お客様各位

カタログ等資料中の旧社名の扱いについて

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

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

2010年4月1日
ルネサスエレクトロニクス株式会社

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

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

ご注意書き

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

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

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

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

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

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

User's Manual

8

RENESAS

改訂一覧は表紙をクリックして直接ご覧になれます。
改訂一覧は改訂箇所をまとめたものであり、詳細については、
必ず本文の内容をご確認ください。

H8/300Lシリーズ

ソフトウェアマニュアル

ルネサスシングルチップマイクロコンピュータ
H8ファミリ

ルネサス エレクトロニクス
www.renesas.com

Rev.2.00　2006.06

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、慎重に制作したものですが万一本資料の記述誤りに起因する損害がお客様に生じた場合には、ルネサス テクノロジはその責任を負いません。
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は負いません。
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。

製品に関する一般的注意事項

1. NC 端子の処理

   【注意】NC 端子には、何も接続しないようにしてください。

   NC(Non-Connection)端子は、内部回路に接続しない場合の他、テスト用端子やノイズ軽減などの目的で使用します。このため、NC 端子には、何も接続しないようにしてください。
   接続された場合については保証できません。

2. 未使用入力端子の処理

   【注意】未使用の入力端子はハイまたはローレベルに固定してください。

   CMOS 製品の入力端子は、一般にハイインピーダンス入力となっています。未使用端子を開放状態で動作させると、周辺ノイズの誘導により中間レベルが発生し、内部で貫通電流が流れて誤動作を起こす恐れがあります。未使用の入力端子は、ハイまたはローレベルに固定してください。

3. 初期化前の処置

   【注意】電源投入時は、製品の状態は不定です。

   すべての電源に電圧が印加され、リセット端子にローレベルが入力されるまでの間、内部回路は不確定であり、レジスタの設定や各端子の出力状態は不定となります。この不定状態によってシステムが誤動作を起こさないようにシステム設計を行ってください。リセット機能を持つ製品は、電源投入後は、まずリセット動作を実行してください。

4. 未定義・リザーブアドレスのアクセス禁止

   【注意】未定義・リザーブアドレスのアクセスを禁止します。

   未定義・リザーブアドレスは、将来の機能拡張用の他、テスト用レジスタなどが割り付けられている場合があります。これらのレジスタをアクセスしたときの動作および継続する動作については、保証できませんので、アクセスしないようにしてください。

はじめに

H8/300L シリーズは、高速 H8/300L CPU をコアとしています。H8/300L CPU は、8 ビット×16 本（または 16 ビット×8 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および、制御機器などの組み込み用に最適なビット操作命令を備えています。ニーモニックは H シリーズ共通です。H8/300L シリーズの応用プログラムの作成は容易に行うことができます。

本マニュアルは、H8/300L の命令の詳細について記載しており、H8/300L シリーズ共通に使用することができます。また、アセンブラについては、別にまとめた「H8/300 シリーズクロスアセンブラユーザーズマニュアル」が用意されていますので、あわせてご活用ください。

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

本版で改訂された箇所

修正項目	ページ	修正箇所
全体	−	社名変更による変更 （修正前）日立製作所　→　（修正後）ルネサス テクノロジ

目次

図目次

表目次

1. CPU

1.1 概要

H8/300L シリーズの CPU は、高速 H8/300L CPU です。H8/300L CPU は、8 ビット×16 本（または 16 ビット×8 本）の汎用レジスタ、ならびに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た高速 CPU です。

1.1.1 特長

H8/300L CPU には、次の特長があります。

■汎用レジスタ方式

8 ビット×16 本（16 ビット×8 本としても使用可能）

■55 種類の基本命令

- 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（@Rn）
- ディスプレースメント付きレジスタ間接（@(d:16, Rn)）
- ポストインクリメント/プリデクリメントレジスタ間接（@Rn＋/@－Rn）
- 絶対アドレス（@aa:8/@aa:16）
- イミディエイト（#xx:8/#xx:16）
- プログラムカウンタ相対（@(d:8, PC)）
- メモリ間接（@@aa:8）

■64K 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
- 動作周波数：5 MHz
- 8/16 ビットレジスタ間加減算 0.4μs
- 8×8 ビット乗算2.8μs
- 16÷8 ビット除算2.8μs

■低消費電力動作

- SLEEP 命令により低消費電力状態に遷移

1.1.2 データ構成

H8/300L CPU は、1 ビット、4 ビット BCD、8 ビット（バイト）、および 16 ビット（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。

バイトデータは、ADDS、SUBS 以外の演算命令で扱われます。また、ワードデータは、MOV.W、ADD.W、SUB.W、CMP.W、ADDS、SUBS、MULXU（8×8 ビット）、DIVXU（16÷8 ビット）命令で扱われます。

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) **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** 1.1 に示します。

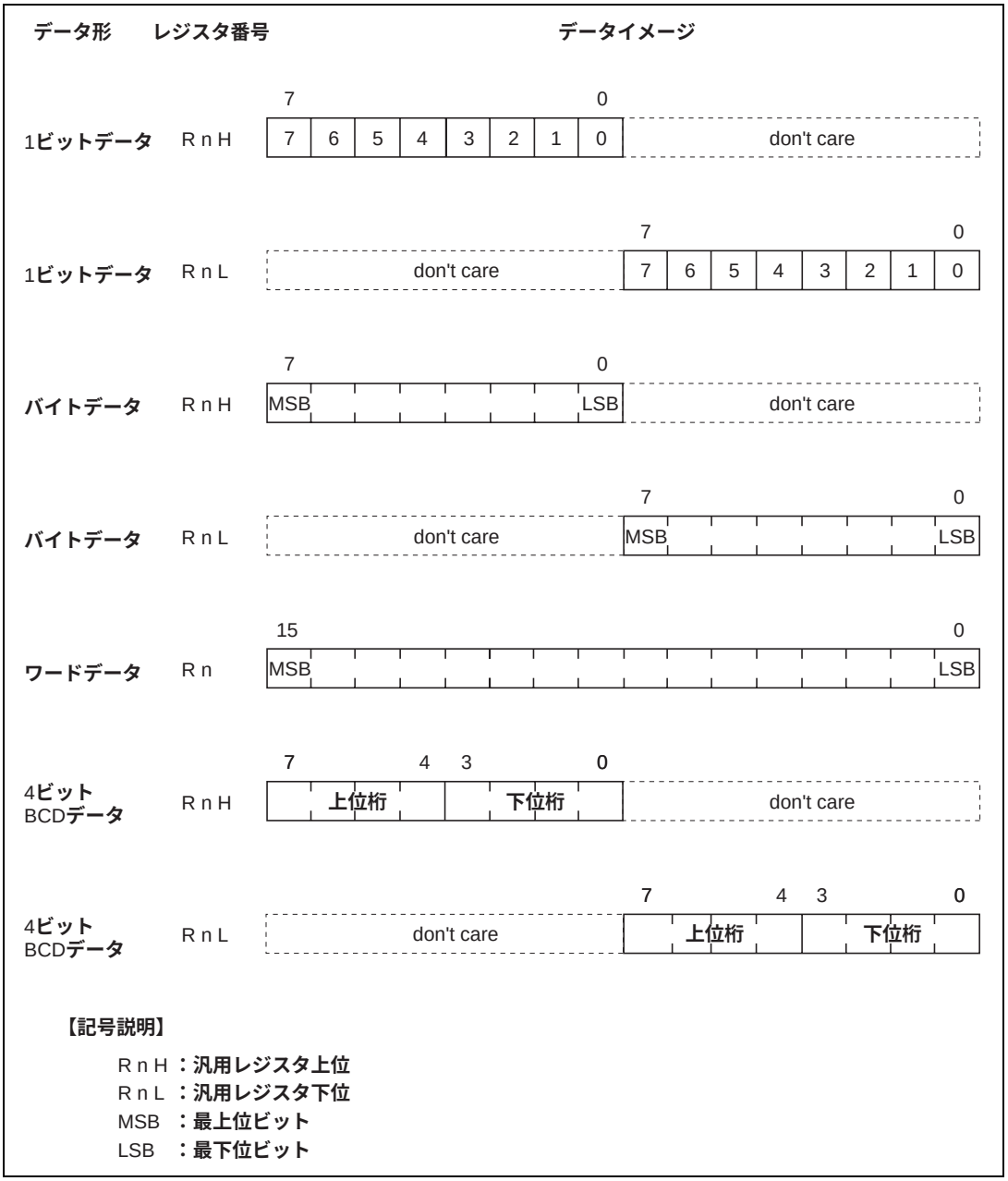


図 1.1 **汎用レジスタのデータ構成**

(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**1.2に示します。H8/300L CPUは、メモリ上のワードデータをアクセスすることができます（MOV.W命令）が、偶数番地から始まるワード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番地前から始まるワード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。

【注】 * H8/300L **シリーズの** LSI **では、ワードサイズのアクセスが不可能な内蔵周辺モジュールがありますので注意してください。**
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。**

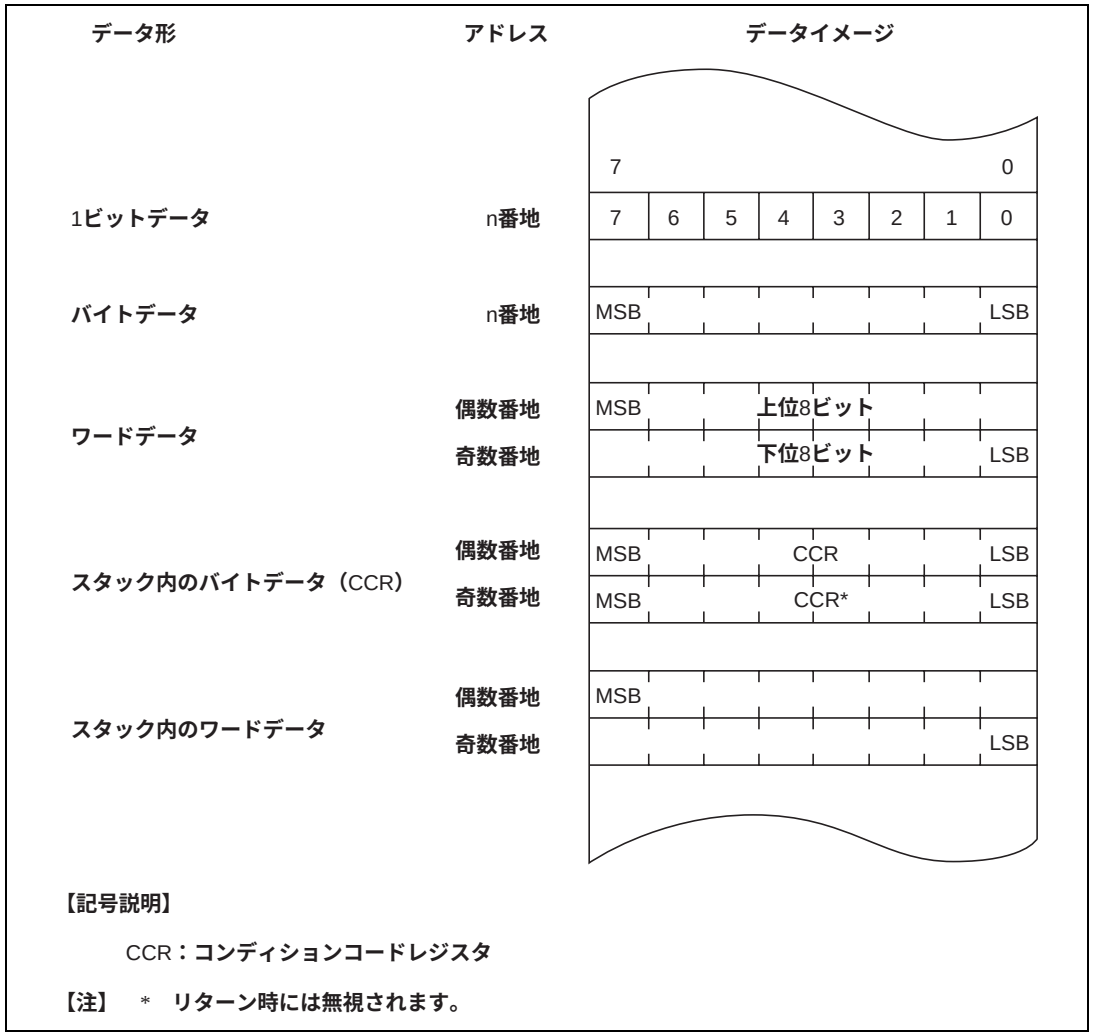


図 1.2 **メモリ上でのデータ構成**

なお、R7 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、必ずワードサイズでアクセスしてください。また、CCR は、ワードデータとして上位 8 ビット、下位 8 ビットに同じ値が格納され、リターン時には下位 8 ビットは無視されます。

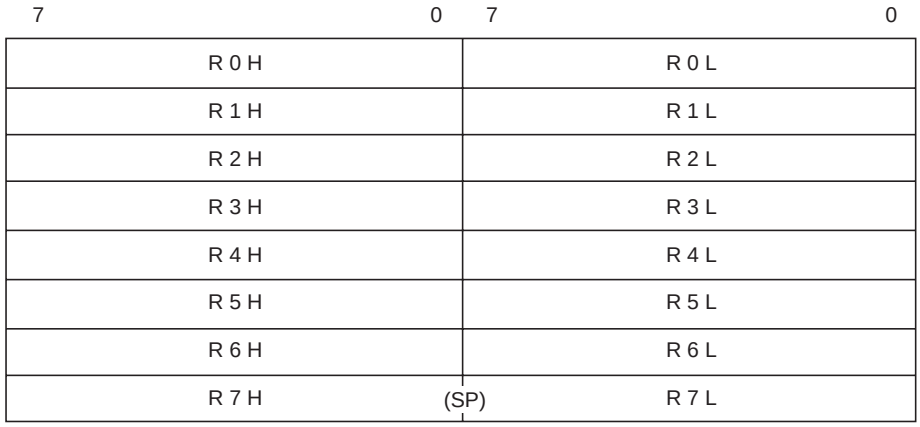
1.1.3 アドレス空間

H8/300L CPU がサポートするアドレス空間は、プログラムコードとデータ領域合計で最大 64K バイトです。メモリマップは、H8/300L シリーズの各 LSI、および各 LSI の動作モード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。

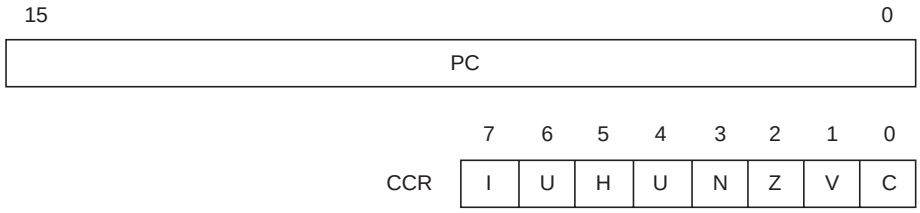
1.1.4 レジスタ構成

H8/300L CPU の内部レジスタ構成を**図** 1.3 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。

汎用レジスタ（Rn）



コントロールレジスタ（CR）



【記号説明】

SP　**：スタックポインタ**
PC　**：プログラムカウンタ**
CCR　**：コンディションコードレジスタ**
I　**：割り込みマスクビット**
U　**：ユーザビット**
H　**：ハーフキャリフラグ**
N　**：ネガティブフラグ**
Z　**：ゼロフラグ**
V　**：オーバフローフラグ**
C　**：キャリフラグ**

図 1.3　CPU **内部レジスタ構成**

1.2 各レジスタの説明

1.2.1 汎用レジスタ

汎用レジスタは、すべて同じ機能を持っており、データレジスタ、アドレスレジスタの区別なく使用できます。

データレジスタとして使用する場合は、8 ビットレジスタとして上位（R0H～R7H）と下位（R0L～R7L）を別々に使用することも、また 16 ビットレジスタ（R0～R7）として使用することもできます。

アドレスレジスタとして使用する場合は、16 ビットレジスタ（R0～R7）として使用します。

レジスタ R7 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コールなどで暗黙的に使用されます。このとき、SP は常にスタック領域の先頭を指しています。スタックの状態を**図** 1.4 に示します。

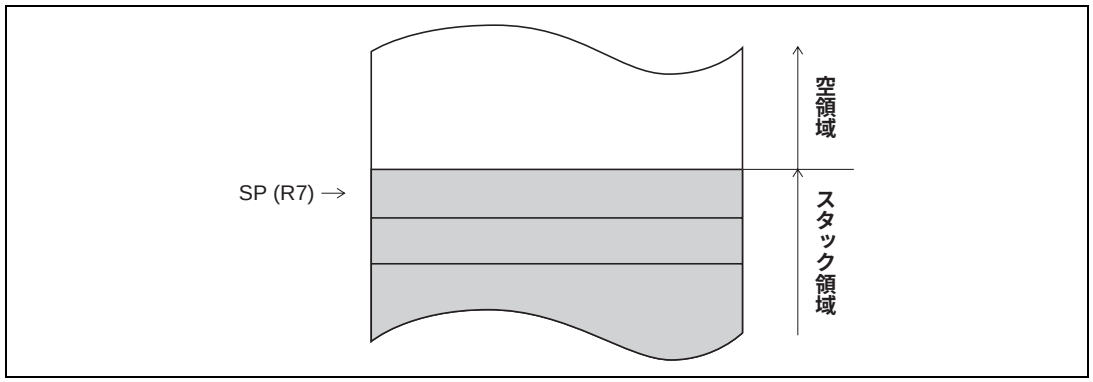


図 1.4 **スタックの状態**

1.2.2 コントロールレジスタ

コントロールレジスタには、16 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）

16 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 16 ビット（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

く常に受け付けられます。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（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 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。

DAA および DAS 命令実行時に、暗黙的に使用されます。

ADD.W、SUB.W、CMP.W 命令ではビット 11 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 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。

1.2.3 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、汎用レジスタおよび CCR の他のビットは初期化されません。レジスタ R7（SP）の初期値も不定です。したがって、リセット直後に、R7 の初期化を行ってください。

1.3 命令

H8/300L CPU の命令には、次の特長があります。

■汎用レジスタアーキテクチャを採用

■高速動作を指向した簡潔で最適化された 55 種類の基本命令

■命令長は 2 バイトまたは 4 バイト

■高速で実行可能な乗除算命令と強力なビット操作命令を用意

■8 種類のアドレッシングモード

1.3.1 命令の分類

H8/300L CPU の命令は合計 55 種類あり、各命令の持つ機能によって、**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、POP*1、PUSH*1	1
算術演算命令	ADD、SUB、ADDX、SUBX、INC、DEC、ADDS、SUBS、DAA、DAS、MULXU、DIVXU、CMP、NEG	14
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	RTE、SLEEP、LDC、STC、ANDC、ORC、XORC、NOP	8
ブロック転送命令/ EEPROM **書き込み専用命令**	EEPMOV	1

合計 55 種類

【注】 *1 POP Rn、PUSH Rn **はそれぞれ** MOV.W @SP＋, Rn、MOV.W Rn, @－SP **と同一です。**

*2 Bcc **は条件分岐命令の総称です。**

1.3.2 命令の機能別一覧

表 1.2 に命令の機能別一覧を示します。また、下表に**表** 1.2 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側）**
Rs	**汎用レジスタ（ソース側）**
Rn	**汎用レジスタ**
(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	3/8/16 **ビット長**

表 1.2　**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	（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。** **ワードデータは** Rn、@Rn、@(d:16, Rn)、@aa:16、#xx:16、@－Rn、@Rn＋**の各アドレッシングモードで扱います。**@aa:8 **はバイトデータのみです。ただし、**@－R7、@R7＋**を使用する場合は必ずワードサイズを指定してください。**
	POP	W	@SP＋→ Rn **スタックから汎用レジスタへデータを復帰します。** **本命令は** MOV.W @SP＋, Rn **と同一です。**
	PUSH	W	Rn → @－SP **汎用レジスタの内容をスタックに退避します。** **本命令は** MOV.W Rn, @－SP **と同一です。**

分類	命令	サイズ*	機能
算術演算命令	ADD SUB	B/W	Rd±Rs → Rd、Rd＋#IMM → Rd 汎用レジスタ間の加減算、または汎用レジスタとイミディエイトデータの加算を行います。汎用レジスタとイミディエイトデータ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。） ワードデータは、汎用レジスタ間の加減算のみで扱います。
	ADDX SUBX	B	Rd±Rs±C → Rd、Rd±#IMM±C → Rd 汎用レジスタ間のキャリ付きの加減算、または汎用レジスタとイミディエイトデータのキャリ付きの加減算を行います。
	INC DEC	B	Rd±1 → Rd 汎用レジスタに 1 を加減算します。
	ADDS SUBS	W	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。
	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	Rd×Rs → Rd 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 ビットの演算が可能です。
	DIVXU	B	Rd÷Rs → Rd 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 ビットの演算が可能です。
	CMP	B/W	Rd－Rs、Rd－#IMM 汎用レジスタ間の比較、または汎用レジスタとイミディエイトデータの比較を行い、その結果を CCR に反映します。 ワードデータは、汎用レジスタ間の比較のみで扱います。
	NEG	B	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。
論理演算命令	AND	B	Rd∧Rs → Rd、Rd∧#IMM → Rd 汎用レジスタ間の論理積、または汎用レジスタとイミディエイトデータの論理積をとります。
	OR	B	Rd∨Rs → Rd、Rd∨#IMM → Rd 汎用レジスタ間の論理和、または汎用レジスタとイミディエイトデータの論理和をとります。
	XOR	B	Rd⊕Rs → Rd、Rd⊕#IMM → Rd 汎用レジスタ間の排他的論理和、または汎用レジスタとイミディエイトデータの排他的論理和をとります。
	NOT	B	～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 SHAR	B	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 SHLR	B	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 ROTR	B	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。
	ROTXL ROTXR	B	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。

分類	命令	サイズ*	機能
ビット操作命令	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 ビットで指定されます。
	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 ビットで指定されます。
	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 ビットで指定されます。
	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 ビットで指定されます。
	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、キャリフラグに結果を格納します。
	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、キャリフラグに結果を格納します。 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。
	BOR	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和をとり、キャリフラグに結果を格納します。
	BIOR	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和をとり、キャリフラグに結果を格納します。 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⊕（<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排他的論理和をとり、キャリフラグに結果を格納します。
	BIXOR	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、キャリフラグとの排他的論理和をとり、キャリフラグに結果を格納します。 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。

<table>
<tr><th>分類</th><th>命令</th><th>サイズ*</th><th>機能</th></tr>
<tr><td rowspan="5">ビット操作命令</td><td>BLD</td><td>B</td><td>（＜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 ビットをキャリフラグに転送します。</td></tr>
<tr><td>BILD</td><td>B</td><td>～（＜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。</td></tr>
<tr><td>BST</td><td>B</td><td>C →（＜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 ビットに、キャリフラグの内容を転送します。</td></tr>
<tr><td>BIST</td><td>B</td><td>～C →（＜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 ビットに、反転された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。</td></tr>
<tr><td rowspan="5">分岐命令</td><td>Bcc</td><td>－</td><td>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>
<tr><th>ニーモニック</th><th>説明</th><th>分岐条件</th></tr>
<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 ∨ Z = 0</td></tr>
<tr><td>BLS</td><td>Low or Same</td><td>C ∨ 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 ⊕ V = 0</td></tr>
<tr><td>BLT</td><td>Less Than</td><td>N ⊕ V = 1</td></tr>
<tr><td>BGT</td><td>Greater Than</td><td>Z ∨ (N ⊕ V) = 0</td></tr>
<tr><td>BLE</td><td>Less or Equal</td><td>Z ∨ (N ⊕ V) = 1</td></tr>
</table></td></tr>
<tr><td>JMP</td><td>－</td><td>指定されたアドレスへ無条件に分岐します。</td></tr>
<tr><td>BSR</td><td>－</td><td>指定されたアドレスへサブルーチン分岐します。</td></tr>
<tr><td>JSR</td><td>－</td><td>指定されたアドレスへサブルーチン分岐します。</td></tr>
<tr><td>RTS</td><td>－</td><td>サブルーチンから復帰します。</td></tr>
</table>

分類	命令	サイズ*	機能
システム制御命令	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	Rs → CCR、#IMM → CCR 汎用レジスタの内容、またはイミディエイトデータを CCR に転送します。
	STC	B	CCR → Rd CCR の内容を汎用レジスタに転送します。
	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。
	NOP	−	PC＋2 → PC PC のインクリメントだけを行います。
ブロック転送命令／EEPROM書き込み専用命令	EEPMOV	−	if R4L ≠ 0 then Repeat @R5＋→ @R6＋, R4L－1 → R4L Until R4L＝0 else next； ブロック転送命令です。R5 で示されるアドレスから始まり、R4L で指定されるバイト数のデータを、R6 で示されるアドレスから始まる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。 H8/300 シリーズの大容量 EEPROM を内蔵した LSI では EEPROM 書き込み専用命令として機能します。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。

【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

【ビット操作命令使用上の注意】

BSET、BCLR、BNOT、BST、BIST の各命令は、バイト単位でデータをリードし、ビット操作後に再びバイト単位でデータをライトします。

したがって、ライト専用ビットを含むレジスタ、またはポートに対してこれらの命令を使用する場合には注意が必要です。

動作順序		動作内容
1	リード	指定したアドレスのデータ（バイト単位）をリードします。
2	ビット操作	リードしたデータの指定された 1 ビットを操作します。
3	ライト	指定したアドレスに、操作したデータ（バイト単位）をライトします。

（例 1）ポート 4 の DDR に BCLR 命令を実行した例を示します。ポート 4 にはプルアップ MOS が内蔵されているものとします。
$P4_7$、$P4_6$は入力端子に設定され、それぞれ"Low レベル"、"High レベル"が入力されているとし、$P4_5$～$P4_0$は出力端子に設定され、それぞれ"Low レベル"出力状態とします。
ここで、BCLR 命令で、$P4_0$を入力ポートにする例を示します。

【A；BCLR 命令を実行前】

	$P4_7$	$P4_6$	$P4_5$	$P4_4$	$P4_3$	$P4_2$	$P4_1$	$P4_0$
入出力	**入力**	**入力**	**出力**	**出力**	**出力**	**出力**	**出力**	**出力**
端子状態	Low **レベル**	High **レベル**	Low **レベル**	Low **レベル**	Low **レベル**	Low **レベル**	Low **レベル**	Low **レベル**
DDR	0	0	1	1	1	1	1	1
DR	1	0	0	0	0	0	0	0
プルアップ MOS	ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF

【B；BCLR 命令を実行】

```
BCLR    #0 , @P4DDR
```

DDR に対して BCLR 命令を実行します。

【C；BCLR 命令を実行後】

	$P4_7$	$P4_6$	$P4_5$	$P4_4$	$P4_3$	$P4_2$	$P4_1$	$P4_0$
入出力	**出力**	**出力**	**出力**	**出力**	**出力**	**出力**	**出力**	**入力**
端子状態	Low **レベル**	High **レベル**	Low **レベル**	Low **レベル**	Low **レベル**	Low **レベル**	Low **レベル**	High **レベル**
DDR	1	1	1	1	1	1	1	0
DR	1	0	0	0	0	0	0	0
プルアップ MOS	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF

【D；BCLR 命令の動作説明】

BCLR 命令を実行すると CPU は、最初に P4DDR をリードします。
P4DDR はライト専用レジスタですので、CPU は H'FF をリードします。
したがってこの例では、DDR は H'3F ですが、CPU がリードしたデータは H'FF となります。
次に、CPU はリードしたデータのビット 0 を"0"にクリアして、データを H'FE に変更します。
最後に、このデータ（H'FE）を DDR に書き込んで BCLR 命令を終了します。
その結果、$P4_0$は DDR が"0"になり、入力ポートになります。
しかし、入力ポートであったビット 7、6 の DDR が 1 になって出力ポートに変化してしまいます。

（例 2）ポート 4 に BSET 命令を実行した例を示します。
$P4_7$、$P4_6$は入力端子に設定され、それぞれ"Low レベル"、"High レベル"が入力されているとし、$P4_5$～$P4_0$は出力端子に設定され、それぞれ"Low レベル"出力状態とします。
ここで、BSET 命令で、$P4_0$に"High レベル"出力を行う例を示します。

【A；BSET 命令を実行前】

	$P4_7$	$P4_6$	$P4_5$	$P4_4$	$P4_3$	$P4_2$	$P4_1$	$P4_0$
入出力	**入力**	**入力**	**出力**	**出力**	**出力**	**出力**	**出力**	**出力**
端子状態	Low **レベル**	High **レベル**	Low **レベル**	Low **レベル**	Low **レベル**	Low **レベル**	Low **レベル**	Low **レベル**
DDR	0	0	1	1	1	1	1	1
DR	1	0	0	0	0	0	0	0
プルアップ MOS	ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF

【B；BSET 命令を実行】

```
BSET    #0 , @PORT4
```

ポート 4 に対して BSET 命令を実行します。

【C；BSET 命令を実行後】

	$P4_7$	$P4_6$	$P4_5$	$P4_4$	$P4_3$	$P4_2$	$P4_1$	$P4_0$
入出力	**入力**	**入力**	**出力**	**出力**	**出力**	**出力**	**出力**	**出力**
端子状態	Low **レベル**	High **レベル**	Low **レベル**	Low **レベル**	Low **レベル**	Low **レベル**	Low **レベル**	High **レベル**
DDR	0	0	1	1	1	1	1	1
DR	0	1	0	0	0	0	0	1
プルアップ MOS	ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF

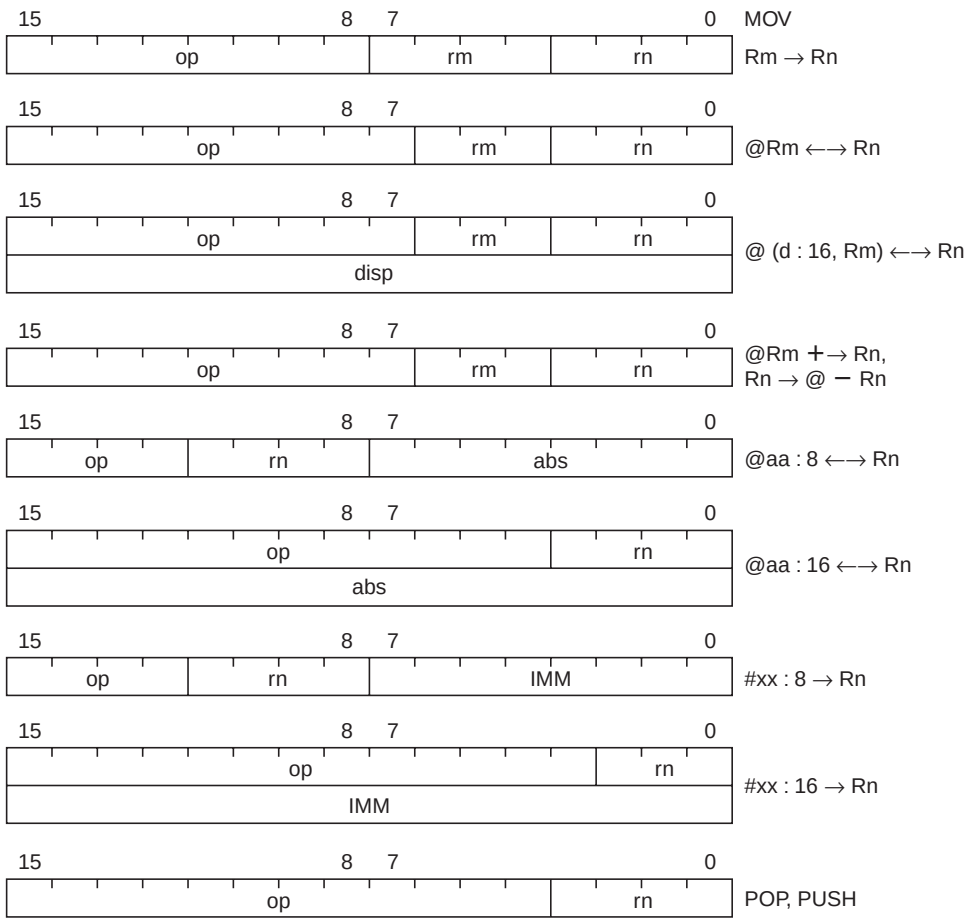
【D；BSET 命令の動作説明】

BSET 命令を実行すると CPU は、最初にポート 4 をリードします。
$P4_7$、$P4_6$は入力端子であるので、CPU は端子の状態（"Low レベル"、"High レベル"入力）をリードします。
$P4_5$～$P4_0$は出力端子であるので、CPU は DR の値をリードします。
したがってこの例では、DR は H'80 ですが、CPU がリードしたデータは H'40 となります。
次に、CPU はリードしたデータのビット 0 を"1"にセットして、データを H'41 に変更します。
最後に、この値（H'41）を DR に書き込んで BSET 命令を終了します。
その結果、$P4_0$は DR が"1"になり、"High レベル"出力になります。

1.3.3 命令の基本フォーマット

(1) データ転送命令の命令フォーマット

データ転送命令の命令フォーマットを**図** 1.5 に示します。



【記号説明】

op	**：オペレーションフィールド**
rm、rn	**：レジスタフィールド**
disp	**：ディスプレースメント**
abs	**：絶対アドレス**
IMM	**：イミディエイトデータ**

図 1.5 **データ転送命令の命令フォーマット**

(2) 算術演算命令・論理演算命令・シフト命令の命令フォーマット

算術演算命令、論理演算命令、およびシフト命令の命令フォーマットを**図** 1.6 に示します。

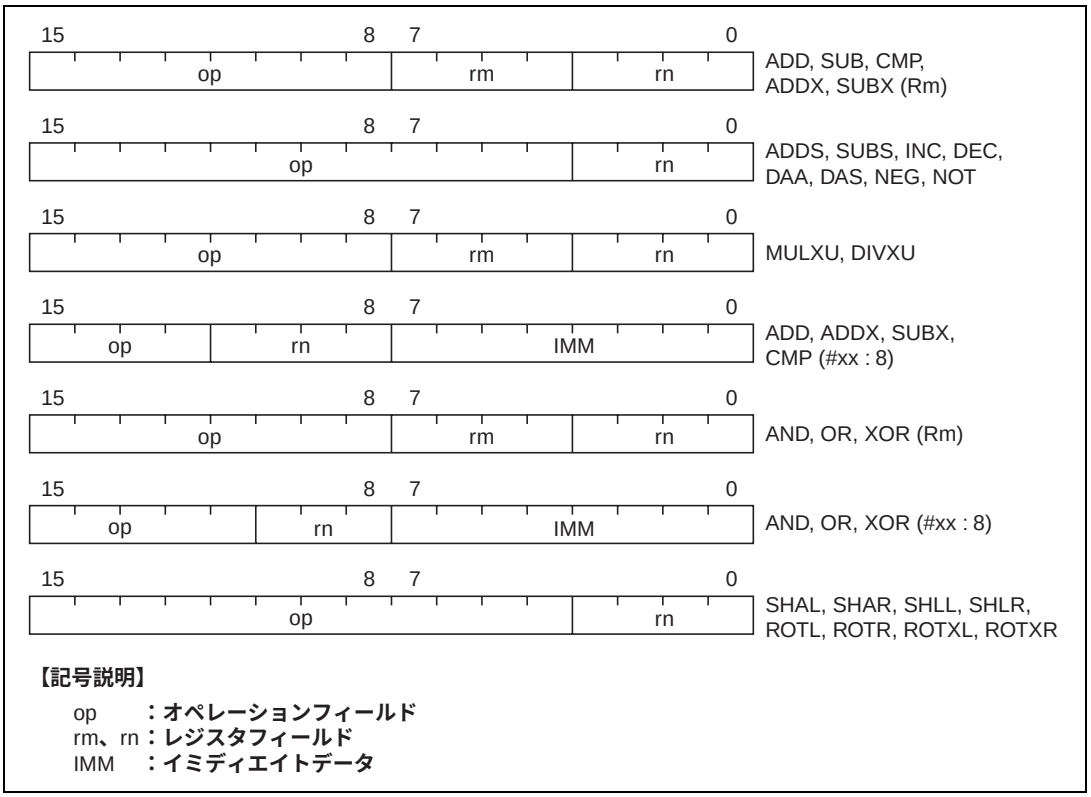


図 1.6 **算術演算命令・論理演算命令・シフト命令の命令フォーマット**

(3) ビット操作命令のフォーマット

ビット操作命令のフォーマットを**図** 1.7 に示します。

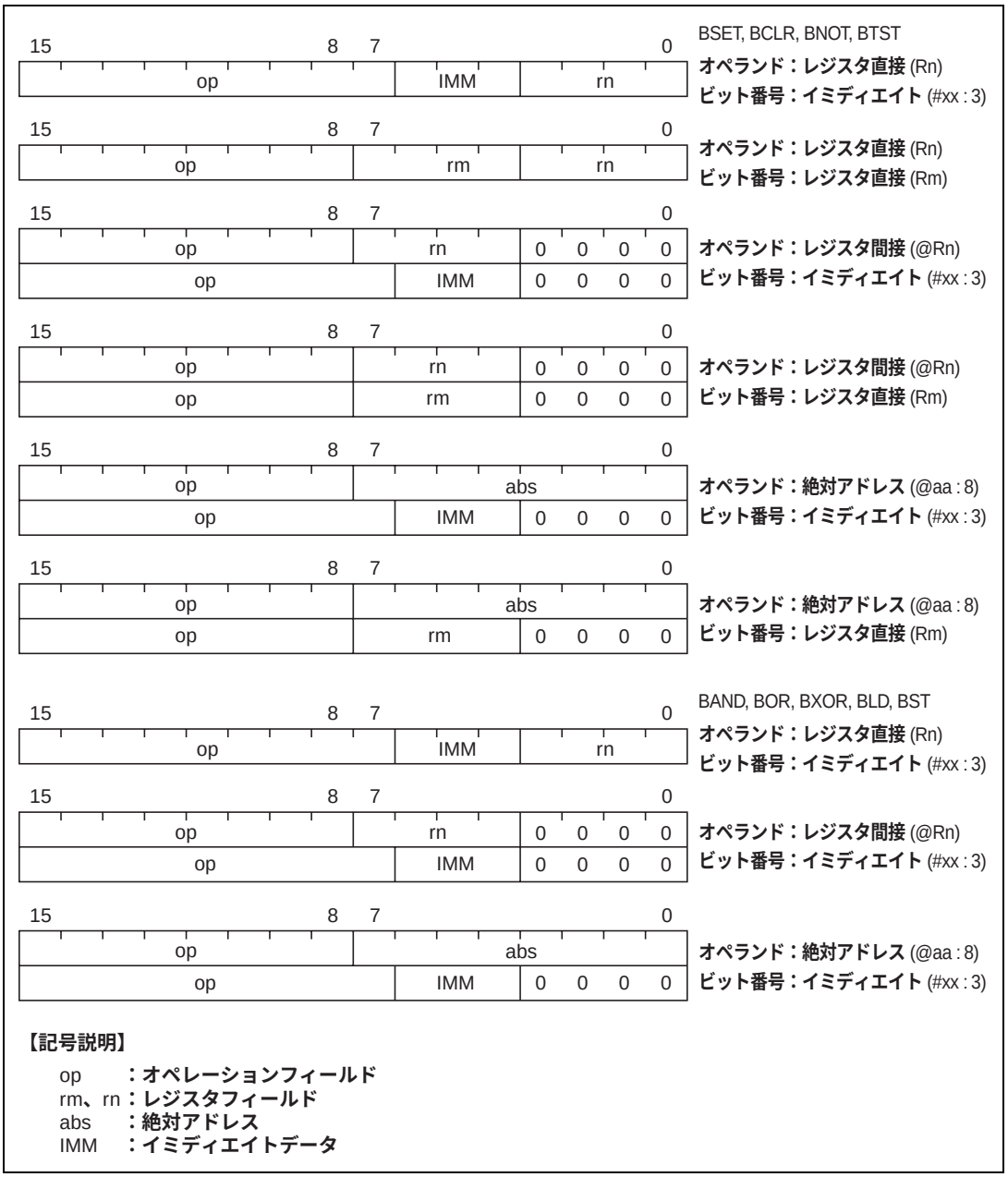


図 1.7 **ビット操作命令の命令フォーマット**(1)

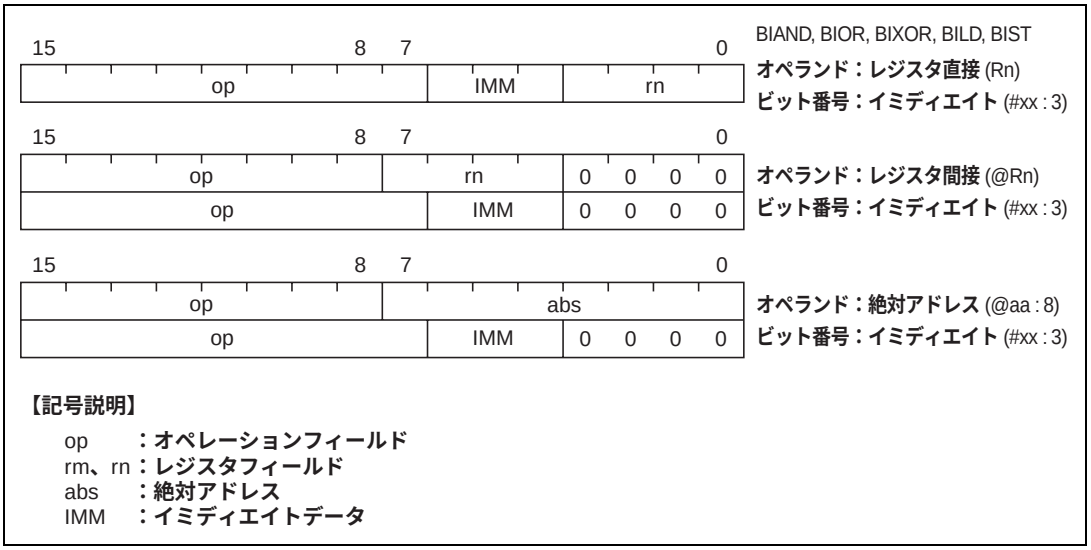
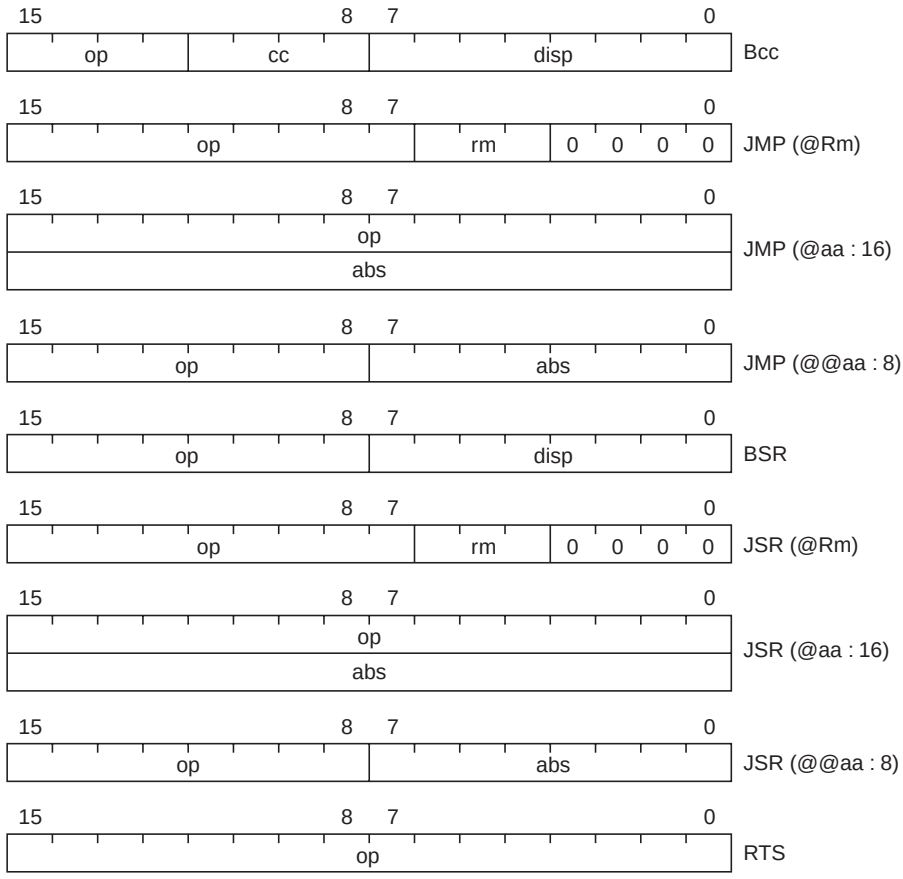


図 1.7 **ビット操作命令の命令フォーマット**(2)

(4) 分岐命令の命令フォーマット

分岐命令の命令フォーマットを**図** 1.8 に示します。



【記号説明】

op	**：オペレーションフィールド**
cc	**：コンディションフィールド**
rm	**：レジスタフィールド**
disp	**：ディスプレースメント**
abs	**：絶対アドレス**

図 1.8 **分岐命令の命令フォーマット**

(5) システム制御命令の命令フォーマット

システム制御命令の命令フォーマットを**図** 1.9 に示します。

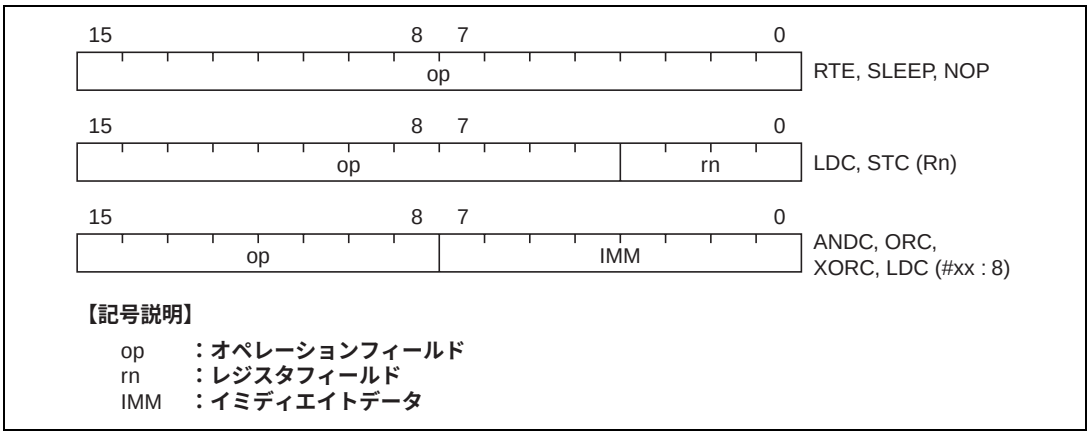


図 1.9 **システム制御命令の命令フォーマット**

(6) ブロック転送命令／EEPROM 書き込み専用命令の命令フォーマット

ブロック転送命令／EEPROM 書き込み専用命令の命令フォーマットを**図** 1.10 に示します。

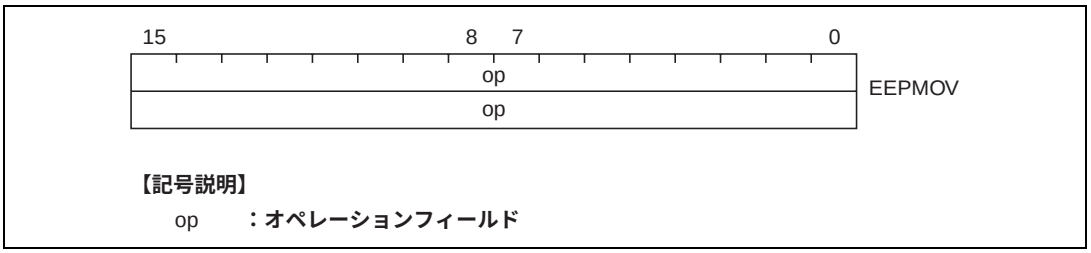


図 1.10 **ブロック転送命令／EEPROM 書き込み専用命令の命令フォーマット**

1.3.4 アドレッシングモードと実効アドレスの計算方法

(1) アドレッシングモード

H8/300L CPU は、**表** 1.3 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①レジスタ直接および⑥イミディエイト（ADD.B、ADDX、SUBX、CMP.B、AND、OR、XOR の各命令）が使用されます。

転送命令では、⑦プログラムカウンタ相対と⑧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可能です。

また、ビット操作命令では、オペランドの指定に①レジスタ直接、②レジスタ間接および⑤絶対アドレス（8 ビット）が使用可能です。さらに、オペランド中のビット番号を指定するために①レジスタ直接（BSET、BCLR、BNOT、BTST の各命令）および⑥イミディエイト（3 ビット）が独立して使

用可能です。

表 1.3 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**	@Rn
③	**ディスプレースメント付レジスタ間接**	@(d:16, Rn)
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**	@Rn+ @−Rn
⑤	**絶対アドレス**	@aa:8/@aa:16
⑥	**イミディエイト**	#xx:8/#xx:16
⑦	**プログラムカウンタ相対**	@(d:8, PC)
⑧	**メモリ間接**	@@aa:8

(a) **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 ビットまたは 16 ビット）がオペランドとなります。

16 ビットレジスタを使用する命令は、MOV.W、ADD.W、SUB.W、CMP.W、ADDS、SUBS、MULXU（8 ビット×8 ビット）、DIVXU（16 ビット÷8 ビット）の各命令です。

(b) **レジスタ間接** @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（16 ビット）の内容をアドレスとしてメモリ上のオペランドを指定します。

(c) **ディスプレースメント付レジスタ間接** @(d:16, 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（16 ビット）の内容に、命令コードの第 2 ワード（第 3、第 4 バイト）の 16 ビットディスプレースメントを加算した内容をアドレスとして、メモリ上のオペランドを指定します。

本アドレッシングモードは、MOV 命令のみで使用されます。特に、MOV.W 命令では、加算結果（アドレス）が偶数となるようにしてください。

(d) **ポストインクリメントレジスタ間接** @Rn+／**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+

MOV（Load from）命令で使用されます。

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（16 ビット）の内容をアドレスとして、メモリ上のオペランドを指定します。その後、レジスタの内容に 1 または 2 が加算され、加算結果がレジスタに格納されます。MOV.B 命令では 1、MOV.W 命令では 2 がそれぞれ加算されます。MOV.W 命令では、レジスタの内容が偶数となるようにしてください。

- **プリデクリメントレジスタ間接** @−Rn

MOV（Store to）命令で使用されます。

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（16 ビット）の内容から 1 または 2 を減算した内容をアドレスとして、メモリ上のオペランドを指定します。その後、減算結果がレジスタに格納されます。MOV.B 命令では 1、MOV.W 命令では 2 がそれぞれ減算されます。MOV.W 命令では、レジスタの内容が偶数となるようにしてください。

(e) **絶対アドレス** @aa:8/@aa:16

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。

このとき、絶対アドレスは 8 ビット（@aa:8）または 16 ビット（@aa:16）で、8 ビット絶対アドレスは MOV.B、ビット操作命令で、16 ビット絶対アドレスは MOV.B、MOV.W、JMP、JSR の各命令で使用されます。

8 ビット絶対アドレスの場合、上位 8 ビットはすべて"1"（H'FF）となります。したがって、アクセス範囲は 65280～65535（H'FF00～H'FFFF）番地です。

(f) **イミディエイト** #xx:8/#xx:16

命令コードの第 2 バイト（#xx:8）または第 3、第 4 バイト（#xx:16）を直接オペランドとして使用します。#xx:16 は、MOV.W 命令のみで使用されます。

なお、ADDS および SUBS 命令では、イミディエイトデータ（1 または 2）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、命令コードの第 2 または第 4 バイトに含まれる場合があります。

(g) **プログラムカウンタ相対** @(d:8, PC)

Bcc、BSR の各命令で使用されます。PC の内容に、命令コードの第 2 バイトの 8 ビットディスプレースメントを加算して、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 16 ビットに符号拡張され、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）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。

(h) **メモリ間接** @@aa:8

JMP および JSR 命令で使用されます。

命令コードの第 2 バイトに含まれる 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。この場合、8 ビット絶対アドレスの上位 8 ビットはすべて"0"（H'00）とされますので、分岐アドレスを格納できるのは 0～255（H'0000～H'00FF）番地です。

ただし、この内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。

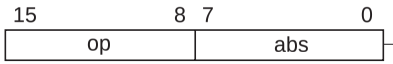
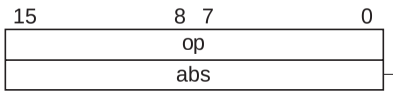
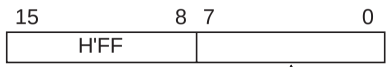
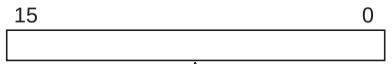
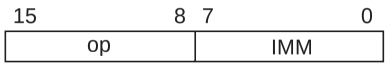
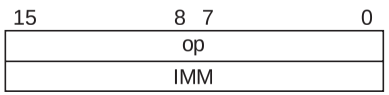
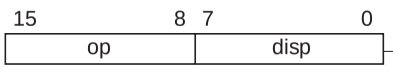
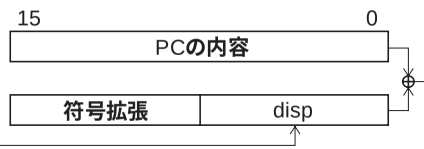
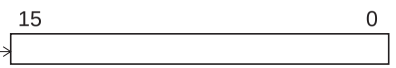
分岐アドレスまたは MOV.W 命令のオペランドアドレスとして奇数アドレスを指定した場合、最下位ビットは 0 とみなされ、1 番地前から始まるワードデータをアクセスします（**「**1.1.2(2)**メモリ上でのデータ構成」**を参照してください）。

(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** 1.4 に示します。

表1.4 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 15 8 7 4 3 0 op regm regn		3 0 3 0 regm regn **オペランドは**regn/n**の内容です。**
②	**レジスタ間接** @Rn 15 7 6 4 3 0 op reg	15 0 reg**の内容**（16**ビット**）	15 0
③	**ディスプレースメント付レジスタ間接** @（d：16, Rn） 15 7 6 4 3 0 op reg disp	15 0 reg**の内容**（16**ビット**） disp	15 0
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** @Rn+ 15 7 6 4 3 0 op reg **・プリデクリメントレジスタ間接** @−Rn 15 7 6 4 3 0 op reg	15 0 reg**の内容**（16**ビット**） 1 or 2 15 0 reg**の内容**（16**ビット**） **オペランドサイズがバイトのとき1、ワードのとき2が加減算されます。** 1 or 2	15 0 15 0

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	**絶対アドレス** @aa : 8  @aa : 16  		 
⑥	**イミディエイト** #xx : 8  #xx : 16  		**オペランドはイミディエイトデータの1または2バイトデータです。**
⑦	**プログラムカウンタ相対**　@ (d : 8, PC)  	 	

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）
⑧	メモリ間接　@@aa : 8 15　8 7　0 op　abs	15　8 7　0 H'00 メモリの内容（16ビット）	15　0

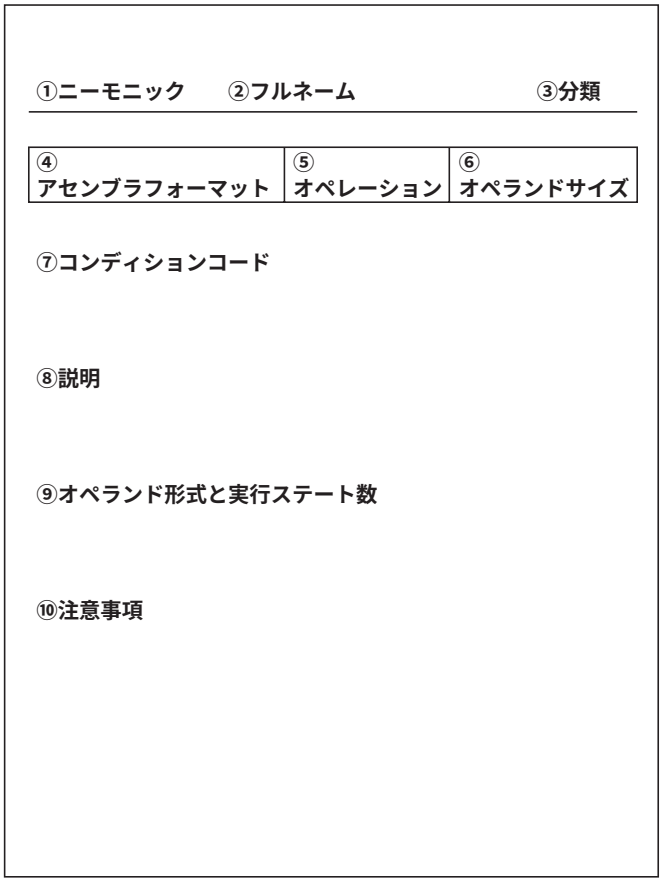
【記号説明】

reg、regm、regn：汎用レジスタ
op　：オペレーションフィールド
disp　：ディスプレースメント
abs　：絶対アドレス
IMM　：イミディエイトデータ

2. 各命令の説明

2.1 表と記号の説明

本書の「2.2 **各命令の説明**」の各命令を説明する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

命令のニーモニックを示します。

②フルネーム

命令のフルネームを示します。

③分類

命令の機能を示します。

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 **アセンブラフォーマット」**を参照）。

⑤オペレーション

命令の操作を簡潔に示します（**「**2.1.2 **オペレーション」**を参照）。

⑥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。

⑦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 **コンディションコード」**を参照）。

⑧説明

命令の動作について詳細に説明します。

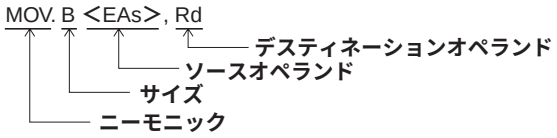
⑨オペランド形式と実行ステート数

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）またはワード（W）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制限されています。

＜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。H8/300L 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**「1.3.4 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。

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

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。

記号	内容
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*
(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（オーバフロー）フラグ
C	CCR の C（キャリ）フラグ
disp	ディスプレースメント
→	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
÷	左辺のオペランドを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）
( )< >	オペランドの実効アドレスの内容

【注】 * 汎用レジスタは、8 ビット（R0H/R0L～R7H/R7L）または 16 ビット（R0～R7）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。
−	実行結果に影響を受けないことを表します。

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（3、8 または 16 ビット）
abs	絶対アドレス（8 または 16 ビット）
disp	ディスプレースメント（8 または 16 ビット）
rs、rd、rn	レジスタ番号（3 または 4 ビット。rs はオペランドの形式の Rs、@Rs などに対応、rd は Rd、@Rd などに対応、rn は Rn 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

rs、rd は、アドレスレジスタとして使用するとき（@Rn, @(d:16, Rn), @Rn＋, @－Rn）は 16 ビットレジスタであり、3 ビットで指定されます。

(2) データレジスタの指定

rs、rd、rn は、データレジスタとして使用するとき（Rn）、サイズがワードのときは 16 ビットレジスタであり、3 ビットで指定されます。

サイズがバイトのときは 8 ビットレジスタであり、4 ビットで指定されます。このとき rs、rd、rn 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"のとき下位（L）が指定され、"0"のとき上位（H）が指定されます。この対応を以下に示します。

16 ビットレジスタ

r			Rn
0	0	0	R0
0	0	1	R1
	⋮		⋮
	⋮		⋮
1	1	1	R7

8 ビットレジスタ

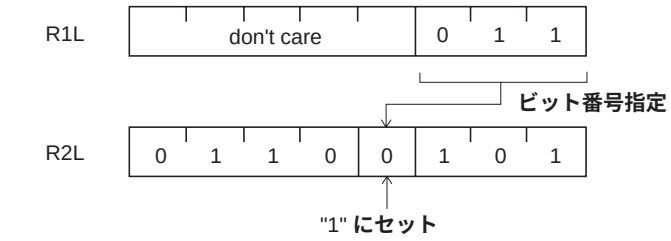
r				Rn
0	0	0	0	R0H
0	0	0	1	R1H
		⋮		⋮
		⋮		⋮
0	1	1	1	R7H
1	0	0	0	R0L
1	0	0	1	R1L
		⋮		⋮
		⋮		⋮
1	1	1	1	R7L

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、……、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。

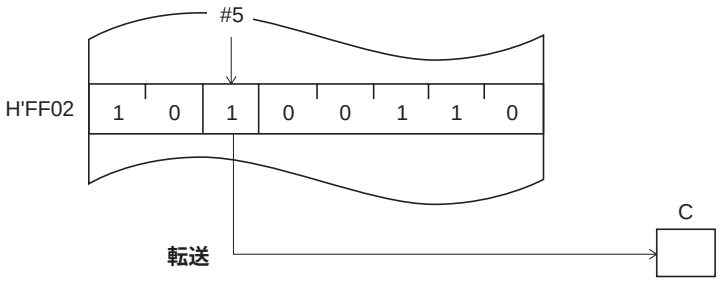
（例 1）R2H のビット 3 を 1 にセットする場合

BSET R1L, R2H



（例 2）H'FF02 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

BLD #5, @H'FF02



なお、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

各命令について説明します。

2.2.1 ADD (B) ADD binary 2 進加算

アセンブラフォーマット	オペレーション	オペランドサイズ
ADD.B ＜EAs＞, Rd	Rd＋(EAs) → Rd	バイト

(1) コンディションコード

I	H		N	Z	V	C
–	–	↕	–	↕	↕	↕	↕

I： 実行前の値が保持されます。
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。
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 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。

(2) 説明

汎用レジスタ 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、結果を汎用レジスタ Rd に格納します。

(3) 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	ADD.B	#xx:8, Rd	8 rd	IMM			2
レジスタ直接	ADD.B	Rs, Rd	0 8	rs rd			2

2.2.2 ADD (W) ADD binary 2 進加算

アセンブラフォーマット	オペレーション	オペランドサイズ
ADD.W Rs, Rd	Rd＋Rs → Rd	ワード

(1) コンディションコード

I	H		N	Z	V	C
–	–	↕	–	↕	↕	↕	↕

I： 実行前の値が保持されます。
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 15 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。

(2) 説明

汎用レジスタ Rd の内容（デスティネーションオペランド）と汎用レジスタ Rs の内容（ソースオペランド）を加算し、結果を汎用レジスタ Rd に格納します。

(3) 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	ADD.W	Rs, Rd	0	9	0	rs	0	rd			2

2.2.3 ADDS ADD with Sign extention アドレスデータ 2 進加算

アセンブラフォーマット	オペレーション	オペランドサイズ
ADDS #1, Rd ADDS #2, Rd	Rd＋1 → Rd Rd＋2 → Rd	ワード

(1) コンディションコード

I	H	N	Z	V	C
–	–	–	–	–	–	–	–

I： 実行前の値が保持されます。
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。

(2) 説明

汎用レジスタ Rd（デスティネーションオペランド）にイミディエイトデータの 1 または 2 を加算します。

ADD 命令と異なり、コンディションコードは実行前の値が保持されます。

(3) 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	ADDS	#1, Rd	0	B	0	0	rd			2
レジスタ直接	ADDS	#2, Rd	0	B	8	0	rd			2

(4) 注意事項

本命令では、バイトサイズのデータは扱えません。

2.2.4 ADDX ADD with eXtend carry キャリ付き加算

アセンブラフォーマット	オペレーション	オペランドサイズ
ADDX ＜EAs＞, Rd	Rd＋(EAs)＋C → Rd	バイト

(1) コンディションコード

I	H		N	Z	V	C
–	–	↕	–	↕	↕	↕	↕

I： 実行前の値が保持されます。
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。
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）のとき実行前の値が保持され、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 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。

(2) 説明

汎用レジスタ 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、結果を汎用レジスタ Rd に格納します。

(3) 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	ADDX	#xx:8, Rd	9 rd	IMM			2
レジスタ直接	ADDX	Rs, Rd	0 E	rs rd			2

2.2.5 AND AND logical 論理積

アセンブラフォーマット	オペレーション	オペランドサイズ
AND ＜EAs＞, Rd	Rd∧(EAs) → Rd	バイト

(1) コンディションコード

I	H			N	Z	V	C
–	–	–	–	↕	↕	0	–

I： 実行前の値が保持されます。
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。

(2) 説明

汎用レジスタ 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、結果を汎用レジスタ Rd に格納します。

(3) 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	#xx:8, Rd	E rd	IMM			2
レジスタ直接	AND	Rs, Rd	1 6	rs rd			2

2.2.6 ANDC AND Control register CCR との論理積

アセンブラフォーマット	オペレーション	オペランドサイズ
ANDC #xx:8, CCR	CCR∧#IMM → CCR	バイト

(1) コンディションコード

I	H	N	Z	V	C
↕	↕	↕	↕	↕	↕

I： 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。
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。

(2) 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。ビット 6 およびビット 4 に対しても、他のビットと同様に操作することができます。

なお、本命令の実行終了時点では、すべての割り込みは受け付けられません。

(3) 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

2.2.7 BAND Bit AND ビット論理積

アセンブラフォーマット	オペレーション	オペランドサイズ
BAND #xx:3, ＜EAd＞	C∧（＜ビット番号＞ of ＜EAd＞）→ C	バイト

(1) コンディションコード

I	H	N	Z	V	C
–	–	–	–	–	↕

I： 実行前の値が保持されます。
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。

(2) 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。

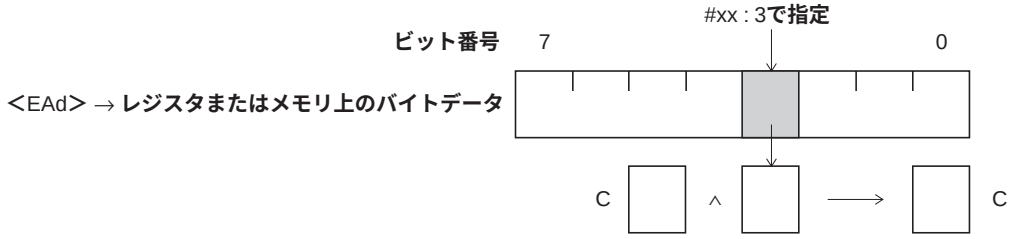
ビット番号は、3 ビットのイミディエイトデータで指定されます。デスティネーションの内容は変化しません。

(3) 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	BAND	#xx:3, Rd	7	6	0	IMM	rd					2
レジスタ間接	BAND	#xx:3, @Rd	7	C	0	rd	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

【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。

(4) 注意事項



2.2.8 Bcc Branch conditionary 条件付き分岐

アセンブラフォーマット	オペレーション	オペランドサイズ
Bcc d:8 └→**コンディションフィールド** **「ニーモニックとコンディションフィールド」参照**	If condition is true, then PC＋d:8 → PC else next;	—

(1) コンディションコード

I	H	N	Z	V	C
–	–	–	–	–	–	–	–

I： 実行前の値が保持されます。
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。

(2) 説明

cc（コンディションフィールド）で指定された条件が成立していると、PC にディスプレース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。

ディスプレースメントは、符号付 8 ビットデータで、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。したがって、分岐できる範囲は、本命令に対して－126～＋128 バイトです。

使用できる条件を以下に示します。

(3) ニーモニックとコンディションフィールド

ニーモニック	説明	cc	条件	符号条件コードの対応
BRA (BT)	Always (True)	0 0 0 0	Always	
BRN (BF)	Never (False)	0 0 0 1	Never	
BHI	HIgh	0 0 1 0	C∨Z=0	X>Y 符号なし
BLS	Low or Same	0 0 1 1	C∨Z=1	X≦Y 符号なし
BCC (BHS)	Carry Clear (High or Same)	0 1 0 0	C=0	X≧Y 符号なし
BCS (BLO)	Carry Set (LOw)	0 1 0 1	C=1	X<Y 符号なし
BNE	Not Equal	0 1 1 0	Z=0	X≠Y 符号なし・あり
BEQ	EQual	0 1 1 1	Z=1	X=Y 符号なし・あり
BVC	oVerflow Clear	1 0 0 0	V=0	
BVS	oVerflow Set	1 0 0 1	V=1	
BPL	PLus	1 0 1 0	N=0	
BMI	MInus	1 0 1 1	N=1	
BGE	Greater or Equal	1 1 0 0	N⊕V=0	X≧Y 符号あり
BLT	Less Than	1 1 0 1	N⊕V=1	X<Y 符号あり
BGT	Greater Than	1 1 1 0	Z∨(N⊕V)=0	X>Y 符号あり
BLE	Less or Equal	1 1 1 1	Z∨(N⊕V)=1	X≦Y 符号あり

(4) 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
プログラムカウンタ相対	BRN (BF)	d : 8	4	1	disp			4
プログラムカウンタ相対	BHI	d : 8	4	2	disp			4
プログラムカウンタ相対	BLS	d : 8	4	3	disp			4
プログラムカウンタ相対	BCC (BHS)	d : 8	4	4	disp			4
プログラムカウンタ相対	BCS (BLO)	d : 8	4	5	disp			4
プログラムカウンタ相対	BNE	d : 8	4	6	disp			4
プログラムカウンタ相対	BEQ	d : 8	4	7	disp			4
プログラムカウンタ相対	BVC	d : 8	4	8	disp			4
プログラムカウンタ相対	BVS	d : 8	4	9	disp			4
プログラムカウンタ相対	BPL	d : 8	4	A	disp			4
プログラムカウンタ相対	BMI	d : 8	4	B	disp			4
プログラムカウンタ相対	BGE	d : 8	4	C	disp			4
プログラムカウンタ相対	BLT	d : 8	4	D	disp			4
プログラムカウンタ相対	BGT	d : 8	4	E	disp			4
プログラムカウンタ相対	BLE	d : 8	4	F	disp			4

(5) 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。また、BRN（BF）の実行ステート数はNOP命令2個と同等です。

2.2.9 BCLR Bit CLeaR ビットクリア

アセンブラフォーマット	オペレーション	オペランドサイズ
BCLR #xx:3, ＜EAd＞ BCLR Rn, ＜EAd＞	0 →（＜ビット番号＞ of ＜EAd＞）	バイト

(1) コンディションコード

I	H	N	Z	V	C
–	–	–	–	–	–	–	–

I： 実行前の値が保持されます。
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。

(2) 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汎用レジスタの内容の下位 3 ビットで指定されます。

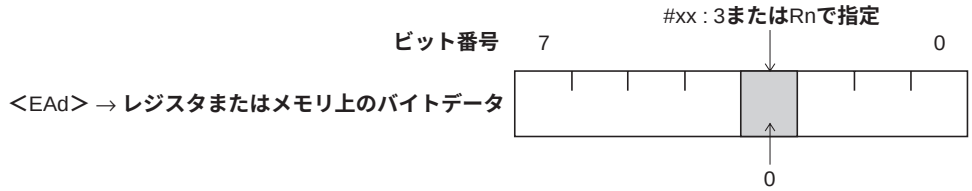
指定された 1 ビットのテストは行いません（コンディションコードは変化しません）。

(3) 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	BCLR	#xx:3, Rd	7	2	0	IMM	rd					2
レジスタ間接	BCLR	#xx:3, @Rd	7	D	0	rd	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, @Rd	7	D	0	rd	0	6	2	rn		0	8
絶対アドレス	BCLR	Rn, @aa:8	7	F	abs			6	2	rn		0	8

【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。

(4) 注意事項



2.2.10 BIAND Bit Invert AND ビット論理積

アセンブラフォーマット	オペレーション	オペランドサイズ
BIAND #xx:3, ＜EAd＞	C∧〔～（＜ビット番号＞ of ＜EAd＞）〕→ C	バイト

(1) コンディションコード

I	H	N	Z	V	C
–	–	–	–	–	↕

I： 実行前の値が保持されます。
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。

(2) 説明

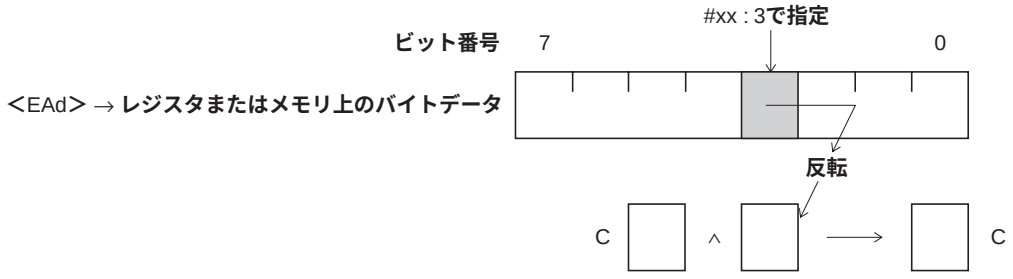
デスティネーションオペランドの指定された 1 ビットを反転し、これとキャリフラグとの論理積をとり、結果をキャリフラグに格納します。ビット番号は、3 ビットのイミディエイトデータで指定されます。デスティネーションの内容は変化しません。

(3) 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	BIAND	#xx:3, Rd	7	6	1	IMM	rd						2
レジスタ間接	BIAND	#xx:3, @Rd	7	C	0	rd	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

【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。

(4) 注意事項



2.2.11 BILD Bit Invert LoaD ビット転送

アセンブラフォーマット	オペレーション	オペランドサイズ
BILD #xx:3, ＜EAd＞	～（＜ビット番号＞ of ＜EAd＞）→ C	バイト

(1) コンディションコード

I	H	N	Z	V	C
–	–	–	–	–	↕

I： 実行前の値が保持されます。
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。

(2) 説明

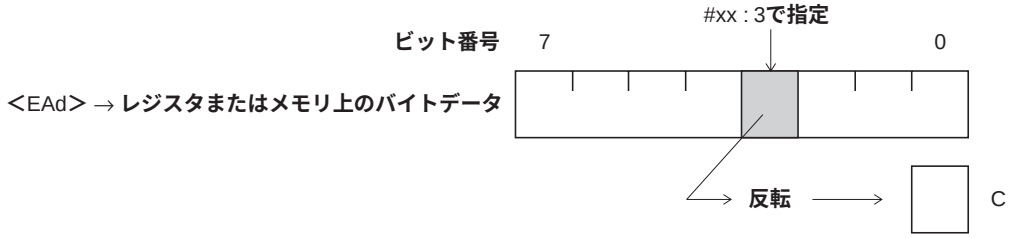
デスティネーションオペランドの指定された 1 ビットを反転し、これをキャリフラグに転送します。ビット番号は、3 ビットのイミディエイトデータで指定されます。デスティネーションの内容は変化しません。

(3) 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, @Rd	7	C	0	rd	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。

(4) 注意事項



2.2.12 BIOR Bit Invert inclusive OR ビット論理和

アセンブラフォーマット	オペレーション	オペランドサイズ
BIOR #xx:3, ＜EAd＞	C∨〔～（＜ビット番号＞ of ＜EAd＞）〕→C	バイト

(1) コンディションコード

I	H	N	Z	V	C
–	–	–	–	–	↕

I： 実行前の値が保持されます。
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。

(2) 説明

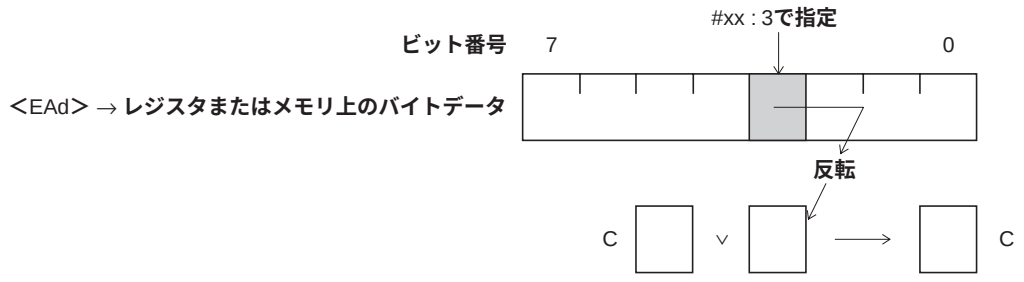
デスティネーションオペランドの指定された 1 ビットを反転し、これとキャリフラグとの論理和をとり、結果をキャリフラグに格納します。ビット番号は、3 ビットのイミディエイトデータで指定されます。デスティネーションの内容は変化しません。

(3) 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	BIOR	#xx:3, Rd	7	4	1	IMM	rd						2
レジスタ間接	BIOR	#xx:3, @Rd	7	C	0	rd	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

【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。

(4) 注意事項



2.2.13 BIST Bit Invert STore ビット転送

アセンブラフォーマット	オペレーション	オペランドサイズ
BIST #xx:3, ＜EAd＞	～C →（＜ビット番号＞ of ＜EAd＞）	バイト

(1) コンディションコード

I	H	N	Z	V	C
–	–	–	–	–	–

I： 実行前の値が保持されます。
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。

(2) 説明

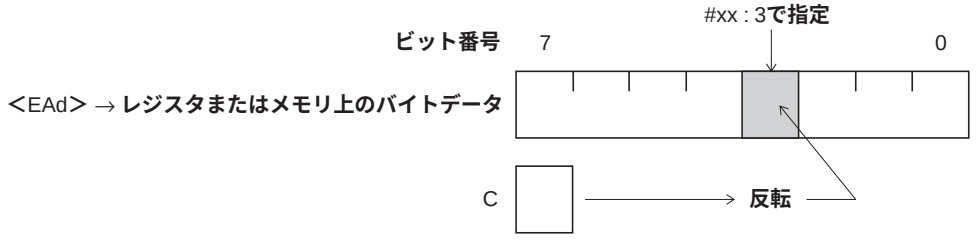
デスティネーションオペランドの指定された 1 ビットのロケーションに、キャリフラグの内容を反転して転送します。ビット番号は、3 ビットのイミディエイトデータで指定されます。なお、デスティネーションオペランドの指定されない他のビットの内容は変化しません。

(3) 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	BIST	#xx:3, Rd	6	7	1	IMM	rd						2
レジスタ間接	BIST	#xx:3, @Rd	7	D	0	rd	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

【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。

(4) 注意事項



2.2.14 BIXOR Bit Invert eXclusive OR ビット排他的論理和

アセンブラフォーマット	オペレーション	オペランドサイズ
BIXOR #xx:3, ＜EAd＞	C⊕〔～（＜ビット番号＞ of ＜EAd＞）〕→ C	バイト

(1) コンディションコード

I	H	N	Z	V	C
–	–	–	–	–	↕

I： 実行前の値が保持されます。
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。

(2) 説明

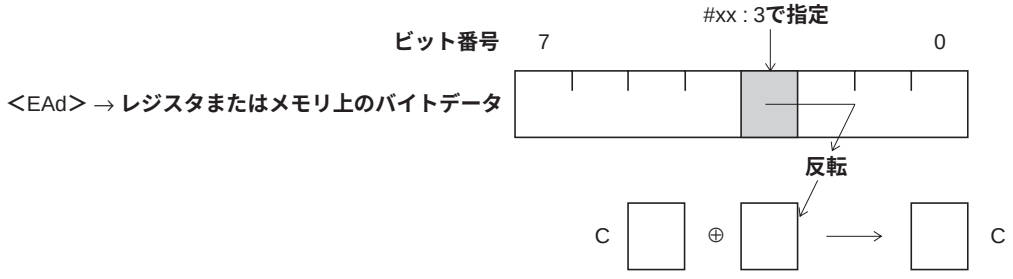
デスティネーションオペランドの指定された 1 ビットを反転し、これとキャリフラグとの排他的論理和をとり、結果をキャリフラグに格納します。ビット番号は、3 ビットのイミディエイトデータで指定されます。デスティネーションの内容は変化しません。

(3) 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	BIXOR	#xx:3, Rd	7	5	1	IMM	rd						2
レジスタ間接	BIXOR	#xx:3, @Rd	7	C	0	rd	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

【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。

(4) 注意事項



2.2.15 BLD Bit LoaD ビット転送

アセンブラフォーマット	オペレーション	オペランドサイズ
BLD #xx:3, ＜EAd＞	（＜ビット番号＞ of ＜EAd＞）→ C	バイト

(1) コンディションコード

I	H	N	Z	V	C
–	–	–	–	–	↕

I： 実行前の値が保持されます。
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。

(2) 説明

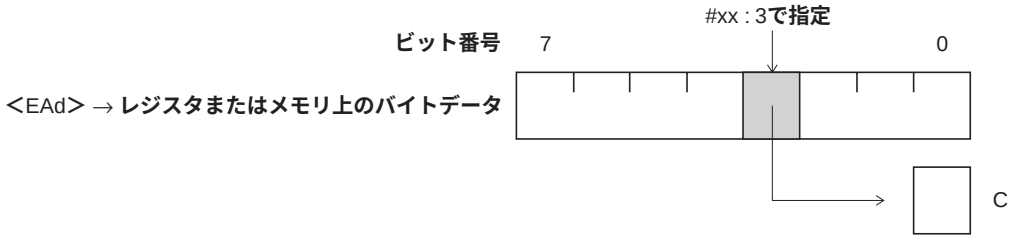
デスティネーションオペランドの指定された 1 ビットをキャリフラグに転送します。ビット番号は、3 ビットのイミディエイトデータで指定されます。デスティネーションの内容は変化しません。

(3) 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	BLD	#xx:3, Rd	7	7	0	IMM	rd						2
レジスタ間接	BLD	#xx:3, @Rd	7	C	0	rd	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

【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。

(4) 注意事項



2.2.16 BNOT Bit NOT ビット反転

アセンブラフォーマット	オペレーション	オペランドサイズ
BNOT #xx:3, ＜EAd＞ BNOT Rn, ＜EAd＞	～（＜ビット番号＞ of ＜EAd＞） →（＜ビット番号＞ of ＜EAd＞）	バイト

(1) コンディションコード

I	H	N	Z	V	C
–	–	–	–	–	–	–	–

I： 実行前の値が保持されます。
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。

(2) 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。

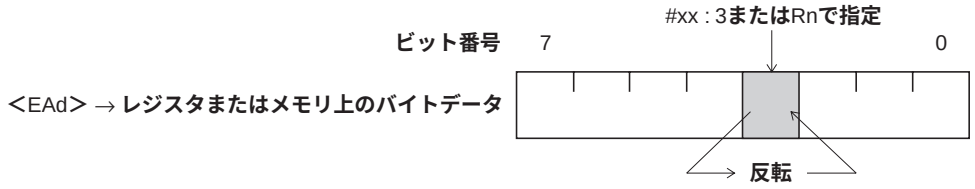
指定された 1 ビットのテストは行いません（コンディションコードは変化しません）。

(3) 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	BNOT	#xx:3, Rd	7	1	0	IMM	rd						2
レジスタ間接	BNOT	#xx:3, @Rd	7	D	0	rd	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, @Rd	7	D	0	rd	0	6	1	rn		0	8
絶対アドレス	BNOT	Rn, @aa:8	7	F	abs			6	1	rn		0	8

【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。

(4) 注意事項



2.2.17 BOR Bit inclusive OR ビット論理和

アセンブラフォーマット	オペレーション	オペランドサイズ
BOR #xx:3, ＜EAd＞	C∨（＜ビット番号＞ of ＜EAd＞）→ C	バイト

(1) コンディションコード

I	H	N	Z	V	C
–	–	–	–	–	↕

I： 実行前の値が保持されます。
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。

(2) 説明

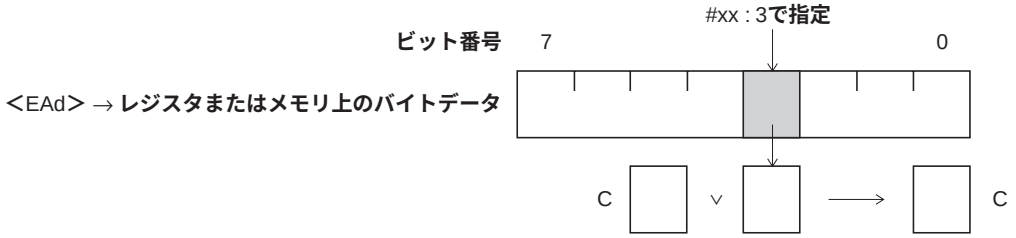
デスティネーションオペランドの指定された 1 ビットとキャリフラグとの論理和をとり、結果をキャリフラグに格納します。ビット番号は、3 ビットのイミディエイトデータで指定されます。デスティネーションの内容は変化しません。

(3) 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, @Rd	7	C	0	rd	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。

(4) 注意事項



2.2.18 BSET Bit SET ビットセット

アセンブラフォーマット	オペレーション	オペランドサイズ
BSET #xx:3, ＜EAd＞ BSET Rn, ＜EAd＞	1 →（＜ビット番号＞ of ＜EAd＞）	バイト

(1) コンディションコード

I	H	N	Z	V	C
–	–	–	–	–	–	–	–

I： 実行前の値が保持されます。
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。

(2) 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"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。

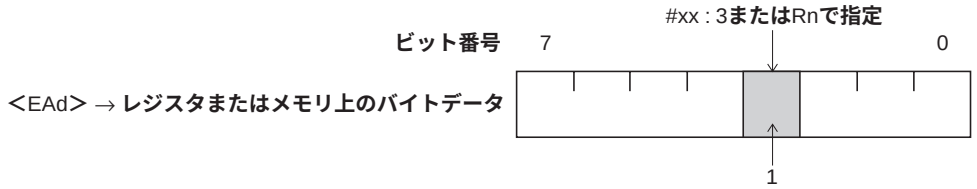
指定された 1 ビットのテストは行いません（コンディションコードは変化しません）。

(3) 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	BSET	#xx:3, Rd	7 0	0 IMM rd			2
レジスタ間接	BSET	#xx:3, @Rd	7 D	0 rd 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, @Rd	7 D	0 rd 0	6 0	rn 0	8
絶対アドレス	BSET	Rn, @aa:8	7 F	abs	6 0	rn 0	8

【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。

(4) 注意事項



2.2.19 BSR Branch to SubRoutine サブルーチン分岐

アセンブラフォーマット	オペレーション	オペランドサイズ
BSR d:8	PC → @－SP PC＋d:8 → PC	—

(1) コンディションコード

I	H			N	Z	V	C
–	–	–	–	–	–	–	–

I： 実行前の値が保持されます。
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。

(2) 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となっています。また、ディスプレースメントは符号付き 8 ビットデータで、分岐できる範囲は本命令に対して－126～＋128 バイトです。

(3) 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	BSR	d：8	5 ┆ 5	disp			6

(4) 注意事項

分岐先アドレスは、必ず偶数になるようにしてください。

2.2.20 BST Bit STore ビット転送

アセンブラフォーマット	オペレーション	オペランドサイズ
BST #xx:3, ＜EAd＞	C →（＜ビット番号＞ of ＜EAd＞）	バイト

(1) コンディションコード

I	H	N	Z	V	C
–	–	–	–	–	–

I： 実行前の値が保持されます。
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。

(2) 説明

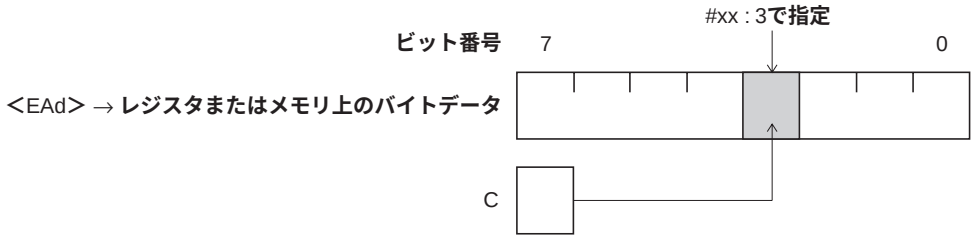
デスティネーションオペランドの指定された 1 ビットのロケーションに、キャリフラグの内容を転送します。ビット番号は、3 ビットのイミディエイトデータで指定されます。

(3) 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	BST	#xx:3, Rd	6	7	0	IMM	rd						2
レジスタ間接	BST	#xx:3, @Rd	7	D	0	rd	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

【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。

(4) 注意事項



2.2.21 BTST Bit TeST ビットテスト

アセンブラフォーマット	オペレーション	オペランドサイズ
BTST #xx:3, ＜EAd＞ BTST Rn, ＜EAd＞	～（＜ビット番号＞ of ＜EAd＞）→ Z	バイト

(1) コンディションコード

I	H	N	Z	V	C
–	–	–	–	–	↕	–	–

I： 実行前の値が保持されます。
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。

(2) 説明

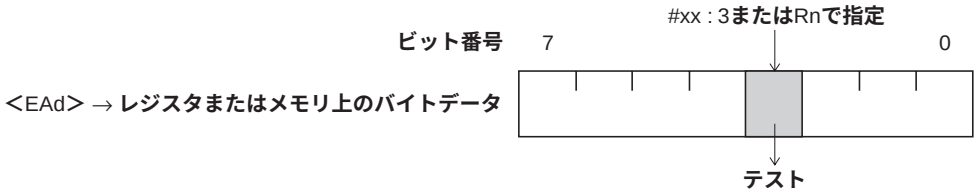
デスティネーションオペランドの指定された 1 ビットの状態を調べて、その結果をゼロフラグに反映します。ビット番号は、3 ビットのイミディエイトデータまたは汎用レジスタの内容の下位 3 ビットで指定されます。デスティネーションの内容は変化しません。

(3) 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	BTST	#xx:3, Rd	7	3	0	IMM	rd						2
レジスタ間接	BTST	#xx:3, @Rd	7	C	0	rd	0	7	3	0	IMM	0	6
絶対アドレス	BTST	#xx:3, @aa:8	7	E	abs			7	3	0	IMM	0	6
レジスタ直接	BTST	Rn, Rd	6	3	rn		rd						2
レジスタ間接	BTST	Rn, @Rd	7	C	0	rd	0	6	3	rn		0	6
絶対アドレス	BTST	Rn, @aa:8	7	E	abs			6	3	rn		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。

(4) 注意事項



2.2.22 BXOR Bit eXclusive OR ビット排他的論理和

アセンブラフォーマット	オペレーション	オペランドサイズ
BXOR #xx:3, ＜EAd＞	C⊕（＜ビット番号＞ of ＜EAd＞）→ C	バイト

(1) コンディションコード

I	H	N	Z	V	C
–	–	–	–	–	↕

I： 実行前の値が保持されます。
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。

(2) 説明

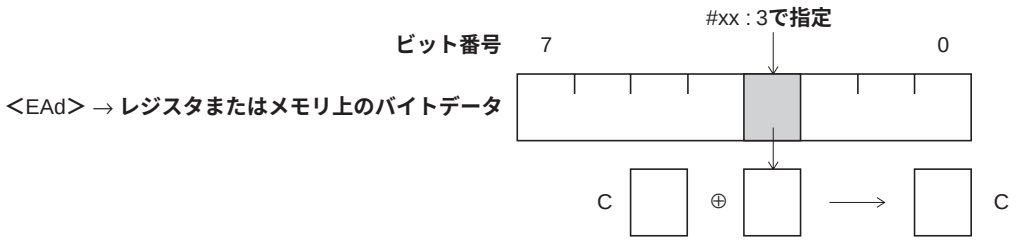
デスティネーションオペランドの指定された 1 ビットとキャリフラグとの排他的論理和をとり、結果をキャリフラグに格納します。ビット番号は、3 ビットのイミディエイトデータで指定されます。デスティネーションの内容は変化しません。

(3) 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	BXOR	#xx:3, Rd	7	5	0	IMM	rd					2
レジスタ間接	BXOR	#xx:3, @Rd	7	C	0	rd	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

【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。

(4) 注意事項



2.2.23 CMP (B) CoMPare 比較

アセンブラフォーマット	オペレーション	オペランドサイズ
CMP.B ＜EAs＞, Rd	Rd－(EAs), CCR セット	バイト

(1) コンディションコード

I	H		N	Z	V	C
–	–	↕	–	↕	↕	↕	↕

I： 実行前の値が保持されます。
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 7 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。

(2) 説明

汎用レジスタ 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、その結果に従って CCR の各ビットをセットまたはクリアします。デスティネーションの内容は変化しません。

(3) 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	CMP.B	#xx:8, Rd	A rd	IMM			2
レジスタ直接	CMP.B	Rs, Rd	1 C	rs rd			2

2.2.24 CMP (W) CoMPare 比較

アセンブラフォーマット	オペレーション	オペランドサイズ
CMP.W Rs，Rd	Rd−Rs, CCR セット	ワード

(1) コンディションコード

I	H		N	Z	V	C
–	–	↕	–	↕	↕	↕	↕

I： 実行前の値が保持されます。
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。
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 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。

(2) 説明

汎用レジスタ Rd の内容（デスティネーションオペランド）から汎用レジスタ Rs の内容（ソースオペランド）を減算し、その結果に従って CCR の各ビットをセットまたはクリアします。デスティネーションの内容は変化しません。

(3) 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	CMP.W	Rs, Rd	1 D	0 rs 0 rd			2

2.2.25 DAA Decimal Adjust Add 10進補正

アセンブラフォーマット	オペレーション	オペランドサイズ
DAA Rd	Rd（10進補正）→ Rd	バイト

(1) コンディションコード

I	H			N	Z	V	C
–	–	*	–	↕	↕	*	↕

I： 実行前の値が保持されます。
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： 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、それ以外のときは実行前の値が保証されます。

(2) 説明

ADD.B、ADDX命令で、4ビットBCDデータを加算した結果が汎用レジスタの内容（デスティネーションオペランド）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、下表に従って汎用レジスタの内容を補正（00、06、60、66を加算）します。

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	0〜2	0	0〜9	60	1
1	0〜2	0	A〜F	66	1
1	0〜3	1	0〜3	66	1

(3) 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	DAA	Rd	0 F	0 rd			2

(4) 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（汎用レジスタの内容、およびC, V, Z, N, Hの各フラグ）は保証しません。

2.2.26 DAS Decimal Adjust Subtract 10進補正

アセンブラフォーマット	オペレーション	オペランドサイズ
DAS Rd	Rd（10進補正）→ Rd	バイト

(1) コンディションコード

I	H			N	Z	V	C
–	–	*	–	↕	↕	*	–

I： 実行前の値が保持されます。
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： 値を保証しません。
C： 実行前の値が保持されます。

(2) 説明

SUB.B、SUBX および NEG 命令で、4 ビット BCD データを減算した結果が汎用レジスタ（デスティネーションオペランド）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、下表に従って汎用レジスタの内容を補正（00、FA、A0、9A を加算）します。

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

(3) 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	DAS	Rd	1 F	0 rd			2

(4) 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（汎用レジスタの内容、および C, V, Z, N, H の各フラグ）は保証しません。

2.2.27 DEC DECrement デクリメント

アセンブラフォーマット	オペレーション	オペランドサイズ
DEC Rd	Rd−1 → Rd	バイト

(1) コンディションコード

I	H			N	Z	V	C
–	–	–	–	↕	↕	↕	–

I： 実行前の値が保持されます。
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（実行前の Rd の内容が H'80 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。
C： 実行前の値が保持されます。

(2) 説明

汎用レジスタ 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、結果を汎用レジスタ Rd に格納します。

(3) 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	DEC	Rd	1 ¦ A	0 ¦ rd			2

2.2.28 DIVXU DIVide eXtend as Unsigned 除算

アセンブラフォーマット	オペレーション	オペランドサイズ
DIVXU Rs, Rd	Rd÷Rs → Rd	バイト

(1) コンディションコード

I	H			N	Z	V	C
–	–	–	–	↕	↕	–	–

I： 実行前の値が保持されます。
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。

(2) 説明

汎用レジスタ Rd の内容（デスティネーションオペランド）を汎用レジスタ Rs の内容（ソースオペランド）で除算し、結果を汎用レジスタ Rd に格納します。Rd は 16 ビットレジスタとして、Rs は 8 ビットレジスタとして指定してください。

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 の下位レジスタ（RdL）に、余りは上位レジスタ（RdH）に格納されます。

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しません。また、オーバフローについては、次ページの「DIVXU **命令とオーバフロー**」を参照してください。

(3) 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	Rs, Rd	5 ⋮ 1	rs ⋮ 0 ⋮ rd			14

(4) DIVXU 命令とオーバフロー

DIVXU 命令は、「16 ビット÷8 ビット → 商 8 ビット 余り 8 ビット」という機能仕様になっているため、オーバフローを生じる場合があります。

たとえば、「H'FFFF÷H'01 → 商 H'FFFF 余り H'00」となり、商 8 ビットを超えてしまいます。

このような場合は、以下のプログラムによって、オーバフローの発生を防ぐことができます。

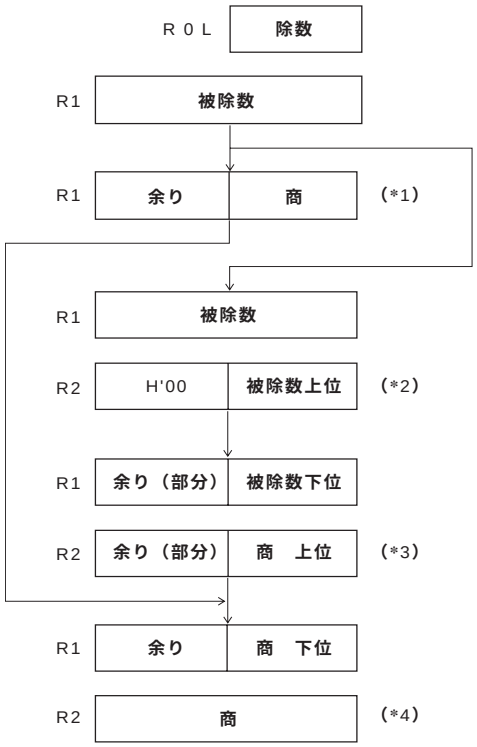
DIVXU R0L, R1 を行う場合：

```
        MOV.B    #H'00, R2H
        CMP.B    R0L, R1H
        BCC      L1
        DIVXU    R0L, R1(*1)
        MOV.B    R1L, R2L
        BRA      L2

L1      MOV.BR1H, R2L(*2)
        DIVXU    R0L, R2
        MOV.B    R2H, R1H(*3)
        DIVXU    R0L, R1
        MOV.B    R2L, R2H
        MOV.B    R1L, R2L

L2      RTS(*4)
```

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。



2.2.29 EEPMOV MOVe data to EEPROM ブロック転送

アセンブラフォーマット	オペレーション	オペランドサイズ
EEPMOV	if R4L≠0 then Repeat @R5＋→ @R6＋ R4L－1 → R4L Until R4L＝0 else next;	―

(1) コンディションコード

I	H	N	Z	V	C
―	―	―	―	―	―	―	―

I： 実行前の値が保持されます。
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。

(2) 説明

ブロック転送命令です。R5 で示されるメモリ上のデータを R6 で示されるメモリへ転送し、R5, R6 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 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。データ転送中は割り込みの検出を行いません。

本命令の実行終了時には、R4L は 0 を、また R5, R6 はそれぞれ（最終アドレス＋1）の内容を保持しています。

(3) 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		
―	EEPMOV		7	B	5	C	5	9	8	F	9＋4n

【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）

(4) 注意事項

本命令ではまず、R5、R6 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。

2.2.30 INC INCrement インクリメント

アセンブラフォーマット	オペレーション	オペランドサイズ
INC Rd	Rd＋1 → Rd	バイト

(1) コンディションコード

I	H			N	Z	V	C
–	–	–	–	↕	↕	↕	–

I： 実行前の値が保持されます。
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（実行前の Rd の内容が H'7F）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。

(2) 説明

汎用レジスタ Rd の内容（デスティネーションオペランド）に"1"を加算し、結果を汎用レジスタ Rd に格納します。

(3) 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	INC	Rd	0 ⋮ A	0 ⋮ rd			2

2.2.31 JMP JuMP 無条件ジャンプ

アセンブラフォーマット	オペレーション	オペランドサイズ
JMP ＜EA＞	実効アドレス → PC	—

(1) コンディションコード

I	H	N	Z	V	C
–	–	–	–	–	–

I： 実行前の値が保持されます。
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。

(2) 説明

指定された実効アドレスに無条件分岐します。

(3) 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	
レジスタ間接	JMP	@Rn	5	9	0	rn	0				4
絶対アドレス	JMP	@aa:16	5	A	0		0	abs			6
メモリ間接	JMP	@@aa:8	5	B	abs						8

(4) 注意事項

分岐先アドレスは必ず偶数になるようにしてください。

2.2.32 JSR Jump to SubRoutine サブルーチンジャンプ

アセンブラフォーマット	オペレーション	オペランドサイズ
JSR ＜EA＞	PC → @－SP **実効アドレス** → PC	—

(1) コンディションコード

I	H	N	Z	V	C
–	–	–	–	–	–	–	–

I： 実行前の値が保持されます。
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。

(2) 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 の値は、本命令の直後の命令の先頭アドレスになります。

(3) 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	
レジスタ間接	JSR	@Rn	5 D	0 rn 0			6
絶対アドレス	JSR	@aa:16	5 E	0 0	abs		8
メモリ間接	JSR	@@aa:8	5 F	abs			8

(4) 注意事項

分岐先アドレスは必ず偶数になるようにしてください。

2.2.33 LDC LoaD to Control register CCR 転送

アセンブラフォーマット	オペレーション	オペランドサイズ
LDC ＜EAs＞, CCR	(EAs) → CCR	バイト

(1) コンディションコード

I	H	N	Z	V	C
↕	↕	↕	↕	↕	↕

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。

(2) 説明

ソースオペランドを CCR に転送します。ビット 6 およびビット 4 に対しても、他のビットと同様に操作することができます。

なお、本命令の実行終了時点では、すべての割り込みは受け付けられません。

(3) 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	#xx:8, CCR	0 7	IMM			2
レジスタ直接	LDC	Rs, CCR	0 3	0 rs			2

2.2.34 MOV (B) MOVe data 転送

アセンブラフォーマット	オペレーション	オペランドサイズ
MOV.B Rs, Rd	Rs → Rd	バイト

(1) コンディションコード

I	H			N	Z	V	C
–	–	–	–	↕	↕	0	–

I： 実行前の値が保持されます。
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、それ以外のときは"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。

(2) 説明

汎用レジスタ Rs の内容を汎用レジスタ 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。

(3) 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	MOV.B	Rs, Rd	0 C	rs rd			2

2.2.35 MOV (W) MOVe data 転送

アセンブラフォーマット	オペレーション	オペランドサイズ
MOV.W Rs, Rd	Rs → Rd	ワード

(1) コンディションコード

I	H			N	Z	V	C
–	–	–	–	↕	↕	0	–

I： 実行前の値が保持されます。
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、それ以外のときは"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。

(2) 説明

汎用レジスタ Rs の内容を汎用レジスタ 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。

(3) 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	MOV.W	Rs, Rd	0	D	0	rs	0	rd			2

2.2.36 MOV (B) MOVe data 転送

アセンブラフォーマット	オペレーション	オペランドサイズ
MOV.B ＜EAs＞, Rd	(EAs) → Rd	バイト

(1) コンディションコード

I	H			N	Z	V	C
–	–	–	–	↕	↕	0	–

I： 実行前の値が保持されます。
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、それ以外のときは"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。

(2) 説明

ソースオペランドの内容を汎用レジスタ 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。

(3) 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	MOV.B	#xx:8, Rd	F	rd	IMM						2
レジスタ間接	MOV.B	@Rs, Rd	6	8	0	rs	rd				4
ディスプレースメント付きレジスタ間接	MOV.B	@(d:16, Rs), Rd	6	E	0	rs	rd	disp			6
ポストインクリメントレジスタ間接	MOV.B	@Rs+, Rd	6	C	0	rs	rd				6
絶対アドレス	MOV.B	@aa:8, Rd	2	rd	abs						4
絶対アドレス	MOV.B	@aa:16, Rd	6	A	0		rd	abs			6

(4) 注意事項

「MOV.B @R7＋, Rd」は、R7 の内容が奇数値となるので使用しないでください。詳細は、**「3.2.3 例外処理の動作」**を参照してください。

2.2.37 MOV (W) MOVe data 転送

アセンブラフォーマット	オペレーション	オペランドサイズ
MOV.W ＜EAs＞, Rd	(EAs) → Rd	ワード

(1) コンディションコード

I	H			N	Z	V	C
–	–	–	–	↕	↕	0	–

I： 実行前の値が保持されます。
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、それ以外のときは"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。

(2) 説明

ソースオペランドの内容を汎用レジスタ 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。

(3) 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	MOV.W	#xx:16, Rd	7	9	0	0	rd	IMM			4
レジスタ間接	MOV.W	@Rs, Rd	6	9	0	rs	0	rd			4
ディスプレースメント付きレジスタ間接	MOV.W	@(d:16, Rs), Rd	6	F	0	rs	0	rd	disp		6
ポストインクリメントレジスタ間接	MOV.W	@Rs+, Rd	6	D	0	rs	0	rd			6
絶対アドレス	MOV.W	@aa:16, Rd	6	B	0		0	rd	abs		6

(4) 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。
(3) H8/300LシリーズのLSIでは、ワードサイズのアクセスが不可能な内蔵周辺モジュールがありますので注意してください。
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。

2.2.38 MOV (B) MOVe data 転送

アセンブラフォーマット	オペレーション	オペランドサイズ
MOV.B Rs, ＜EAd＞	Rs → (EAd)	バイト

(1) コンディションコード

I	H			N	Z	V	C
–	–	–	–	↕	↕	0	–

I： 実行前の値が保持されます。
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、それ以外のときは"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。

(2) 説明

汎用レジスタ 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。

(3) 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	
レジスタ間接	MOV.B	Rs, @Rd	6	8	1	rd	rs			4
ディスプレースメント付きレジスタ間接	MOV.B	Rs, @(d:16, Rd)	6	E	1	rd	rs	disp		6
プリデクリメントレジスタ間接	MOV.B	@Rs, @-Rd	6	C	1	rd	rs			6
絶対アドレス	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

(4) 注意事項

(1) 「MOV.B Rs, @－R7」は、R7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、**「**3.2.3 **例外処理の動作」**を参照してください。

(2) MOV.B RnL, @－RnまたはMOV.B RnH, @－Rnを実行すると（実行前のRnの内容－1）の下位RnLまたは上位RnHが転送されます。

2.2.39 MOV (W) MOVe data 転送

アセンブラフォーマット	オペレーション	オペランドサイズ
MOV.W Rs, ＜EAd＞	Rs → (EAd)	ワード

(1) コンディションコード

I	H	N	Z	V	C
–	–	–	–	↕	↕	0	–

I： 実行前の値が保持されます。
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、それ以外のときは"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。

(2) 説明

汎用レジスタ 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。

(3) 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	
レジスタ間接	MOV.W	Rs, @Rd	6	9	1	rd	0	rs			4
ディスプレースメント付きレジスタ間接	MOV.W	Rs, @(d:16, Rd)	6	F	1	rd	0	rs	disp		6
プリデクリメントレジスタ間接	MOV.W	Rs, @-Rd	6	D	1	rd	0	rs			6
絶対アドレス	MOV.W	Rs, @aa:16	6	B	8		0	rs	abs		6

(4) 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, @－R7」の機械語はPUSH.W Rsと同一です。
(3) MOV.W Rn, @－Rnを実行すると（実行前のRnの内容－2）が転送されます。
(4) H8/300LシリーズのLSIではワードサイズのアクセスが不可能な内蔵周辺モジュールがありますので注意してください。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。

2.2.40 MULXU MULtiply eXtend as Unsigned 乗算

アセンブラフォーマット	オペレーション	オペランドサイズ
MULXU Rs, Rd	Rd×Rs → Rd	バイト

(1) コンディションコード

I	H		N	Z	V	C
–	–	–	–	–	–	–	–

I： 実行前の値が保持されます。
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。

(2) 説明

汎用レジスタ Rd の内容（デスティネーションオペランド）と汎用レジスタ Rs の内容（ソースオペランド）を乗算し、結果を汎用レジスタ Rd に格納します。Rd は 16 ビットレジスタ（上位 8 ビットは無視されます）として、Rs は 8 ビットレジスタとして指定してください。このとき、Rs は Rd の上位または下位レジスタ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。

(3) 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	MULXU	Rs, Rd	5	0	rs	0	rd				14

2.2.41 NEG NEGate 2進符号反転

アセンブラフォーマット	オペレーション	オペランドサイズ
NEG Rd	0−Rd → Rd	バイト

(1) コンディションコード

I	H		N	Z	V	C
−	−	↕	−	↕	↕	↕	↕

I： 実行前の値が保持されます。
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（実行前の Rd の内容が H'80）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 7 にボローが発生した（実行前の Rd の内容が H'00 以外）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。

(2) 説明

汎用レジスタ Rd の内容（デスティネーションオペランド）の 2 の補数をとり（H'00 から減算し）、結果を汎用レジスタ Rd に格納します。ただし、実行前の Rd の内容が H'80 の場合の結果は H'80 となります。

(3) 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	NEG	Rd	1 7	8 rd			2

2.2.42 NOP No OPeration 無操作

アセンブラフォーマット	オペレーション	オペランドサイズ
NOP	PC+2 → PC	—

(1) コンディションコード

I	H			N	Z	V	C
–	–	–	–	–	–	–	–

I： 実行前の値が保持されます。
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。

(2) 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。

(3) 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

2.2.43 NOT NOT＝logical complement 論理反転

アセンブラフォーマット	オペレーション	オペランドサイズ
NOT Rd	～Rd → Rd	バイト

(1) コンディションコード

I	H			N	Z	V	C
–	–	–	–	↕	↕	0	–

I： 実行前の値が保持されます。
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、それ以外のときは"0"にクリアされます。
Z： 実行結果が 0（ゼロ）のとき（実行前の Rd の内容が H'FF 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。

(2) 説明

汎用レジスタ Rd の内容（デスティネーションオペランド）の 1 の補数をとり、結果を汎用レジスタ Rd に格納します。

(3) 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	NOT	Rd	1 ⋮ 7	0 ⋮ rd			2

2.2.44 OR inclusive OR logical 論理和

アセンブラフォーマット	オペレーション	オペランドサイズ
OR ＜EAs＞, Rd	Rd∨(EAs) → Rd	バイト

(1) コンディションコード

I	H			N	Z	V	C
–	–	–	–	↕	↕	0	–

I： 実行前の値が保持されます。
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。

(2) 説明

汎用レジスタ 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、結果を汎用レジスタ Rd に格納します。

(3) 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	#xx:8, Rd	C	rd	IMM			2
レジスタ直接	OR	Rs, Rd	1	4	rs rd			2

2.2.45 ORC inclusive OR Control register CCR との論理和

アセンブラフォーマット	オペレーション	オペランドサイズ
ORC #xx:8 CCR	CCR∨#IMM → CCR	バイト

(1) コンディションコード

I	H	N	Z	V	C
↕	↕	↕	↕	↕	↕	↕	↕

I： 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。
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。

(2) 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。ビット 6 およびビット 4 に対しても、他のビットと同様に操作することができます。本命令の実行終了時点では、すべての割り込みは受け付けられません。

(3) 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	ORC	#xx:8, CCR	0 ┆ 4	IMM			2

2.2.46 POP POP data スタックよりデータ復帰

アセンブラフォーマット	オペレーション	オペランドサイズ
POP Rn	@SP＋→ Rn	ワード

(1) コンディションコード

I	H			N	Z	V	C
–	–	–	–	↕	↕	0	–

I： 実行前の値が保持されます。
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、それ以外のときは"0"にクリアされます。
Z： 転送データが 0 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。

(2) 説明

スタックから汎用レジスタ Rn へデータを復帰します。このとき復帰するデータを検査し、その結果を CCR に反映します。

(3) 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	Rd	6	D	7	0	rn			6

(4) 注意事項

本命令は、MOV.W @SP＋, Rn と同一です。

2.2.47 PUSH PUSH data スタックへデータ退避

アセンブラフォーマット	オペレーション	オペランドサイズ
PUSH Rn	Rn → @－SP	ワード

(1) コンディションコード

I	H			N	Z	V	C
–	–	–	–	↕	↕	0	–

I： 実行前の値が保持されます。
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、それ以外のときは"0"にクリアされます。
Z： 転送データが 0 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。

(2) 説明

汎用レジスタ 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。

(3) 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	
－	PUSH	Rs	6 D	F 0 rn			6

(4) 注意事項

本命令は、MOV.W Rn, @－SP と同一です。

2.2.48 ROTL ROTate Left ローテート

アセンブラフォーマット	オペレーション	オペランドサイズ
ROTL Rd	Rd（左ローテート）→ Rd	バイト

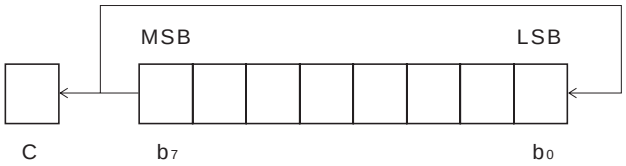
(1) コンディションコード

I	H			N	Z	V	C
–	–	–	–	↕	↕	0	↕

I： 実行前の値が保持されます。
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： 実行前のビット 7 の値が格納されます。

(2) 説明

汎用レジスタ 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。



(3) 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	ROTL	Rd	1 ¦ 2	8 ¦ rd			2

2.2.49 ROTR ROTate Right ローテート

アセンブラフォーマット	オペレーション	オペランドサイズ
ROTR Rd	Rd（右ローテート）→ Rd	バイト

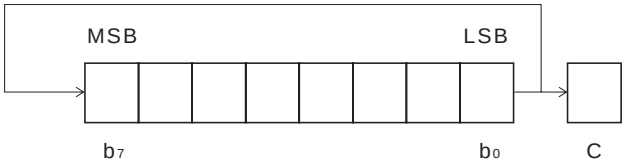
(1) コンディションコード

I	H			N	Z	V	C
–	–	–	–	↕	↕	0	↕

I： 実行前の値が保持されます。
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： 実行前のビット 0 の値が格納されます。

(2) 説明

汎用レジスタ 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。



(3) 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	ROTR	Rd	1 ¦ 3	8 ¦ rd			2

2.2.50 ROTXL ROTate with eXtend carry Left キャリ付きローテート

アセンブラフォーマット	オペレーション	オペランドサイズ
ROTXL Rd	Rd（キャリ付き左ローテート）→ Rd	バイト

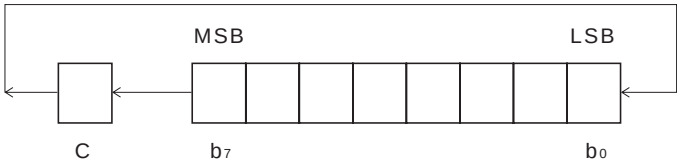
(1) コンディションコード

I	H			N	Z	V	C
–	–	–	–	↕	↕	0	↕

I： 実行前の値が保持されます。
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： 実行前のビット 7 の値が格納されます。

(2) 説明

汎用レジスタ 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。



(3) 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	ROTXL	Rd	1 ¦ 2	0 ¦ rd			2

2.2.51 ROTXR ROTate with eXtend carry Right キャリ付きローテート

アセンブラフォーマット	オペレーション	オペランドサイズ
ROTXR Rd	Rd（キャリ付き右ローテート）→ Rd	バイト

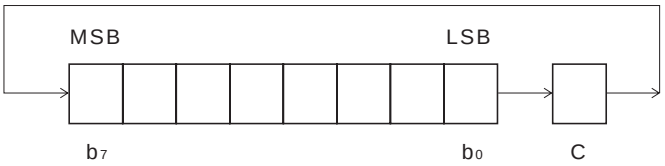
(1) コンディションコード

I	H			N	Z	V	C
–	–	–	–	↕	↕	0	↕

I： 実行前の値が保持されます。
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： 実行前のビット 0 の値が格納されます。

(2) 説明

汎用レジスタ 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。



(3) 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	ROTXR	Rd	1 3	0 rd			2

2.2.52 RTE ReTurn from Exception 例外処理からのリターン

アセンブラフォーマット	オペレーション	オペランドサイズ
RTE	@SP+→ CCR @SP+→ PC	—

(1) コンディションコード

I	H	N	Z	V	C
↕	↕	↕	↕	↕	↕	↕	↕

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。
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。
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。

(2) 説明

例外処理から復帰します。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。

なお、CCR はバイトサイズですが、スタックからの復帰はワードサイズ（下位 8 ビットは無視）で行われます。したがって、本命令によって SP の内容は+4 されます。

(3) 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	
—	RTE		5 ┆ 6	7 ┆ 0			10

2.2.53 RTS ReTurn from Subroutine サブルーチンリターン

アセンブラフォーマット	オペレーション	オペランドサイズ
RTS	@SP+→ PC	—

(1) コンディションコード

I	H			N	Z	V	C
–	–	–	–	–	–	–	–

I： 実行前の値が保持されます。
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。

(2) 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。

(3) 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	
—	RTS		5	4	7	0			8

2.2.54 SHAL SHift Arithmetic Left 算術シフト

アセンブラフォーマット	オペレーション	オペランドサイズ
SHAL Rd	Rd（左算術シフト）→ Rd	バイト

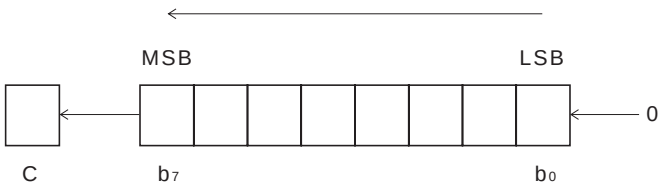
(1) コンディションコード

I	H			N	Z	V	C
–	–	–	–	↕	↕	↕	↕

I： 実行前の値が保持されます。
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： 実行前のビット 7 の値が格納されます。

(2) 説明

汎用レジスタ 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（ゼロ）が格納されます。



(3) 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	Rd	1 0	8 rd			2

(4) 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。

2.2.55 SHAR SHift Arithmetic Right 算術シフト

アセンブラフォーマット	オペレーション	オペランドサイズ
SHAR Rd	Rd（右算術シフト）→ Rd	バイト

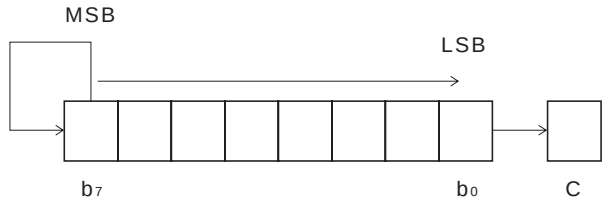
(1) コンディションコード

I	H			N	Z	V	C
–	–	–	–	↕	↕	0	↕

I： 実行前の値が保持されます。
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： 実行前のビット 0 の値が格納されます。

(2) 説明

汎用レジスタ 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 C 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、符号変化は起りません。



(3) 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	SHAR	Rd	1 1	8 rd			2

2.2.56 SHLL SHift Logical Left 論理シフト

アセンブラフォーマット	オペレーション	オペランドサイズ
SHLL Rd	Rd（左論理シフト）→ Rd	バイト

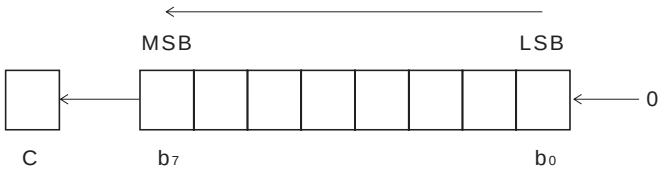
(1) コンディションコード

I	H			N	Z	V	C
–	–	–	–	↕	↕	0	↕

I： 実行前の値が保持されます。
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： 実行前のビット 0 の値が格納されます。

(2) 説明

汎用レジスタ 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（ゼロ）が格納されます。



(3) 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	SHLL	Rd	1 0	0 rd			2

(4) 注意事項

本命令と SHAL 命令とでは、オーバフローフラグの動作が異なります。

2.2.57 SHLR SHift Logical Right 論理シフト

アセンブラフォーマット	オペレーション	オペランドサイズ
SHLR Rd	Rd（右論理シフト）→ Rd	バイト

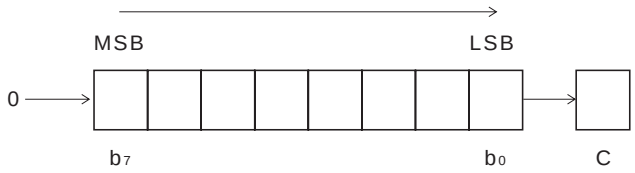
(1) コンディションコード

I	H			N	Z	V	C
–	–	–	–	0	↕	0	↕

I： 実行前の値が保持されます。
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： 実行前のビット 0 の値が格納されます。

(2) 説明

汎用レジスタ 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（ゼロ）が格納されます。



(3) 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	SHLR	Rd	1	1	0	rd			2

2.2.58 SLEEP SLEEP 低消費電力状態命令

アセンブラフォーマット	オペレーション	オペランドサイズ
SLEEP	プログラム実行状態 → 低消費電力状態	—

(1) コンディションコード

I	H	N	Z	V	C
–	–	–	–	–	–	–	–

I： 実行前の値が保持されます。
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。

(2) 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 以外の割り込み要求では、CPU 側で割り込みがマスクされている（I＝1）場合、低消費電力状態は解除されません。

(3) 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

(4) 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。

2.2.59 STC STore from Control register CCR 転送

アセンブラフォーマット	オペレーション	オペランドサイズ
STC CCR, Rd	CCR → Rd	バイト

(1) コンディションコード

I	H	N	Z	V	C
–	–	–	–	–	–	–	–

I： 実行前の値が保持されます。
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。

(2) 説明

CCR の内容を汎用レジスタ Rd に転送します。ビット 6 およびビット 4 についても他のビットと同様に扱うことができます。

(3) 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	STC	CCR, Rd	0 2	0 rd			2

2.2.60 SUB (B) SUBtract binary 2 進減算

アセンブラフォーマット	オペレーション	オペランドサイズ
SUB.B Rs, Rd	Rd－Rs → Rd	バイト

(1) コンディションコード

I	H		N	Z	V	C
–	–	↕	–	↕	↕	↕	↕

I： 実行前の値が保持されます。
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 7 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。

(2) 説明

汎用レジスタ Rd の内容（デスティネーションオペランド）から汎用レジスタ Rs の内容（ソースオペランド）を減算し、結果を Rd に格納します。

(3) 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	SUB.B	Rs, Rd	1	8	rs	rd			2

(4) 注意事項

本命令は汎用レジスタ間の減算のみ可能ですが、汎用レジスタの内容とイミディエイトデータの減算は SUBX.B 命令を使用することにより実現できます。この場合、「SUBX.B #xx:8, Rd」を実行する前に、Z フラグを"1"にセットし、C フラグを"0"にクリアしてください。また、イミディエイト値#IMM ≠0 の場合、次のプログラム例も使用できます。

```
(1)   ORC   #H'05, CCR
      SUBX  # (IMM－1), Rd
(2)   ADD   # (0－IMM), Rd
      XORC  #H'01, CCR
```

2.2.61 SUB (W) SUBtract binary 2 進減算

アセンブラフォーマット	オペレーション	オペランドサイズ
SUB.W Rs, Rd	Rd－Rs → Rd	ワード

(1) コンディションコード

I	H		N	Z	V	C
–	–	↕	–	↕	↕	↕	↕

I： 実行前の値が保持されます。
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。
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 15 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。

(2) 説明

汎用レジスタ Rd の内容（デスティネーションオペランド）から汎用レジスタ Rs の内容（ソースオペランド）を減算し、結果を Rd に格納します。

(3) 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	SUB.W	Rs, Rd	1	9	0 rs	0 rd			2

2.2.62 SUBS SUBtract with Sign extention アドレスデータ 2 進減算

アセンブラフォーマット	オペレーション	オペランドサイズ
SUBS #1, Rd SUBS #2, Rd	Rd−1 → Rd Rd−2 → Rd	ワード

(1) コンディションコード

I	H	N	Z	V	C
−	−	−	−	−	−	−	−

I： 実行前の値が保持されます。
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。

(2) 説明

汎用レジスタ Rd（デスティネーションオペランド）からイミディエイトの 1 または 2 を減算します。

SUB 命令と異なり、コンディションコードは実行前の値が保持されます。

(3) 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	SUBS	#1, Rd	1	B	0	0	rd			2
レジスタ直接	SUBS	#2, Rd	1	B	8	0	rd			2

(4) 注意事項

本命令では、バイトサイズのデータは扱えません。

2.2.63 SUBX SUBtract with eXtend carry キャリ付き減算

アセンブラフォーマット	オペレーション	オペランドサイズ
SUBX ＜EAs＞, Rd	Rd－(EAs)－ C → Rd	バイト

(1) コンディションコード

I		H		N	Z	V	C
–	–	↕	–	↕	↕	↕	↕

I： 実行前の値が保持されます。
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）のとき実行前の値が保持されます。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 7 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。

(2) 説明

汎用レジスタ 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 Rd に格納します。

(3) 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	SUBX	#xx:8, Rd	B rd	IMM			2
レジスタ直接	SUBX	Rs, Rd	1 E	rs rd			2

2.2.64 XOR eXclusive OR logical 排他的論理和

アセンブラフォーマット	オペレーション	オペランドサイズ
XOR ＜EAs＞, Rd	Rd⊕(EAs) → Rd	バイト

(1) コンディションコード

I	H			N	Z	V	C
–	–	–	–	↕	↕	0	–

I： 実行前の値が保持されます。
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。

(2) 説明

汎用レジスタ 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 Rd に格納します。

(3) 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	#xx:8, Rd	D rd	IMM			2
レジスタ直接	XOR	Rs, Rd	1 5	rs rd			2

2.2.65 XORC eXclusive OR Control register CCR との排他的論理和

アセンブラフォーマット	オペレーション	オペランドサイズ
XORC #xx:8, CCR	CCR⊕#IMM → CCR	バイト

(1) コンディションコード

I	H	N	Z	V	C
↕	↕	↕	↕	↕	↕	↕	↕

I： 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。
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。

(2) 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。ビット 6 およびビット 4 に対しても、他のビットと同様に操作することができます。本命令の実行終了時点では、すべての割り込みは受け付けられません。

(3) 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	XORC	#xx:8, CCR	0 ⋮ 5	IMM			2

2.3 オペレーションコードマップ

表 2.1 にオペレーションコードマップを示します。**表** 2.1 では、命令コードの第 1 バイト（第 1 ワードのビット 15～8）についてのみ示しています。

表 2.1　オペレーションコードマップ

第2バイトの最上位ビット（命令コードの第1ワードのビット7）が0の場合を示します。
第2バイトの最上位ビット（命令コードの第1ワードのビット7）が1の場合を示します。

L0 / H1	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NOP	SLEEP	STC	LDC	ORC	XORC	ANDC	LDC	ADD		INC	ADDS	MOV		ADDX	DAA
1	SHLL / SHAL	SHLR / SHAR	ROTXL / ROTL	ROTXR / ROTR	OR	XOR	AND	NOT / NEG	SUB		DEC	SUBS	CMP		SUBX	DAS
2	MOV															
3																
4	BRA	BRN	BHI	BLS	BCC	BCS	BNE	BEQ	BVC	BVS	BPL	BMI	BGE	BLT	BGT	BLE
5	MULXU	DIVXU			RTS	BSR	RTE				JMP				JSR	
6	BSET	BNOT	BCLR	BTST	BOR / BIOR	BXOR / BIXOR	BAND / BIAND	BST / BLD / BILD	MOV*							
7								BIST				EEPMOV		ビット操作命令		
8	ADD															
9	ADDX															
A	CMP															
B	SUBX															
C	OR															
D	XOR															
E	AND															
F	MOV															

【注】　* PUSH、POP命令の機械語はMOV命令と同一です。詳細は、「2.2 各命令の説明」を参照してください。

2.4 命令セット一覧表

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						実行ステート数*
			#xx:8/16	Rn	@Rn	@(d:16, Rn)	@-Rn/@Rn+	@aa:8/16	@(d:8, PC)	@@aa	—		I	H	N	Z	V	C	
MOV	MOV.B #xx:8, Rd	B	2								#xx:8→Rd8		—	—	↕	↕	0	—	2
	MOV.B Rs, Rd	B		2							Rs8→Rd8		—	—	↕	↕	0	—	2
	MOV.B @Rs, Rd	B			2						@Rs16→Rd8		—	—	↕	↕	0	—	4
	MOV.B @(d:16, Rs), Rd	B				4					@(d:16, Rs16)→Rd8		—	—	↕	↕	0	—	6
	MOV.B @Rs+, Rd	B					2				@Rs16→Rd8 Rs16+1→Rs16		—	—	↕	↕	0	—	6
	MOV.B @aa:8, Rd	B						2			@aa:8→Rd8		—	—	↕	↕	0	—	4
	MOV.B @aa:16, Rd	B						4			@aa:16→Rd8		—	—	↕	↕	0	—	6
	MOV.B Rs, @Rd	B			2						Rs8→@Rd16		—	—	↕	↕	0	—	4
	MOV.B Rs, @(d:16, Rd)	B				4					Rs8→@(d:16, Rd16)		—	—	↕	↕	0	—	6
	MOV.B Rs, @-Rd	B					2				Rd16-1→Rd16 Rs8→@Rd16		—	—	↕	↕	0	—	6
	MOV.B Rs, @aa:8	B						2			Rs8→@aa:8		—	—	↕	↕	0	—	4
	MOV.B Rs, @aa:16	B						4			Rs8→@aa:16		—	—	↕	↕	0	—	6
	MOV.W #xx:16, Rd	W	4								#xx:16→Rd		—	—	↕	↕	0	—	4
	MOV.W Rs, Rd	W		2							Rs16→Rd16		—	—	↕	↕	0	—	2
	MOV.W @Rs, Rd	W			2						@Rs16→Rd16		—	—	↕	↕	0	—	4
	MOV.W @(d:16, Rs), Rd	W				4					@(d:16, Rs16)→Rd16		—	—	↕	↕	0	—	6
	MOV.W @Rs+, Rd	W					2				@Rs16→Rd16 Rs16+2→Rs16		—	—	↕	↕	0	—	6
	MOV.W @aa:16, Rd	W						4			@aa:16→Rd16		—	—	↕	↕	0	—	6
	MOV.W Rs, @Rd	W			2						Rs16→@Rd16		—	—	↕	↕	0	—	4
	MOV.W Rs, @(d:16, Rd)	W				4					Rs16→@(d:16, Rd16)		—	—	↕	↕	0	—	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						実行ステート数*
			#xx:8/16	Rn	@Rn	@(d:16, Rn)	@-Rn/@Rn+	@aa:8/16	@(d:8, PC)	@@aa		I	H	N	Z	V	C	
MOV	MOV.W Rs, @-Rd	W					2				Rd16-2→Rd16 Rs16→@Rd16	–	–	↕	↕	0	–	6
	MOV.W Rs, @aa:16	W						4			Rs16→@aa:16	–	–	↕	↕	0	–	6
POP	POP Rd	W					2				@SP→Rd16 SP+2→SP	–	–	↕	↕	0	–	6
PUSH	PUSH Rs	W					2				SP-2→SP Rs16→@SP	–	–	↕	↕	0	–	6
ADD	ADD.B #xx:8, Rd	B	2								Rd8+#xx:8→Rd8	–	↕	↕	↕	↕	↕	2
	ADD.B Rs, Rd	B		2							Rd8+Rs8→Rd8	–	↕	↕	↕	↕	↕	2
	ADD.W Rs, Rd	W		2							Rd16+Rs16→Rd16	–	①	↕	↕	↕	↕	2
ADDX	ADDX.B #xx:8, Rd	B	2								Rd8+#xx:8+C→Rd8	–	↕	↕	②	↕	↕	2
	ADDX.B Rs, Rd	B		2							Rd8+Rs8+C→Rd8	–	↕	↕	②	↕	↕	2
ADDS	ADDS.W #1, Rd	W		2							Rd16+1→Rd16	–	–	–	–	–	–	2
	ADDS.W #2, Rd	W		2							Rd16+2→Rd16	–	–	–	–	–	–	2
INC	INC.B Rd	B		2							Rd8+1→Rd8	–	–	↕	↕	↕	–	2
DAA	DAA.B Rd	B		2							Rd8 10進補正→Rd8	–	*	↕	↕	*	③	2
SUB	SUB.B Rs, Rd	B		2							Rd8-Rs8→Rd8	–	↕	↕	↕	↕	↕	2
	SUB.W Rs, Rd	W		2							Rd16-Rs16→Rd16	–	①	↕	↕	↕	↕	2
SUBX	SUBX.B #xx:8, Rd	B	2								Rd8-#xx:8-C→Rd8	–	↕	↕	②	↕	↕	2
	SUBX.B Rs, Rd	B		2							Rd8-Rs8-C→Rd8	–	↕	↕	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						実行ステート数*
			#xx:8/16	Rn	@Rn	@(d:16, Rn)	@-Rn/@Rn+	@aa:8/16	@(d:8, PC)	@@aa	–		I	H	N	Z	V	C	
SUBS	SUBS.W #1, Rd	W		2								Rd16-1→Rd16	–	–	–	–	–	–	2
	SUBS.W #2, Rd	W		2								Rd16-2→Rd16	–	–	–	–	–	–	2
DEC	DEC.B Rd	B		2								Rd8-1→Rd8	–	–	↕	↕	↕	–	2
DAS	DAS.B Rd	B		2								Rd8 10進補正→Rd8	–	*	↕	↕	*	–	2
NEG	NEG.B Rd	B		2								0-Rd→Rd	–	↕	↕	↕	↕	↕	2
CMP	CMP.B #xx:8, Rd	B	2									Rd8-#xx:8	–	↕	↕	↕	↕	↕	2
	CMP.B Rs, Rd	B		2								Rd8-Rs8	–	↕	↕	↕	↕	↕	2
	CMP.W Rs, Rd	W		2								Rd16-Rs16	–	①	↕	↕	↕	↕	2
MULXU	MULXU.B Rs, Rd	B		2								Rd8×Rs8→Rd16	–	–	–	–	–	–	14
DIVXU	DIVXU.B Rs, Rd	B		2								Rd16÷Rs8→Rd16 (RdH:余り, RdL:商)	–	–	⑤	⑥	–	–	14
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
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
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
NOT	NOT.B Rd	B		2								$\overline{Rd}$→Rd	–	–	↕	↕	0	–	2
SHAL	SHAL.B Rd	B		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					実行ステート数*
			#xx:8/16	Rn	@Rn	@(d:16, Rn)	@-Rn/@Rn+	@aa:8/16	@(d:8, PC)	@@aa		I	H	N	Z	V	C	
SHAR	SHAR.B Rd	B		2								—	—	↕	↕	0	↕	2
SHLL	SHLL.B Rd	B		2								—	—	↕	↕	0	↕	2
SHLR	SHLR.B Rd	B		2								—	—	0	↕	0	↕	2
ROTXL	ROTXL.B Rd	B		2								—	—	↕	↕	0	↕	2
ROTXR	ROTXR.B Rd	B		2								—	—	↕	↕	0	↕	2
ROTL	ROTL.B Rd	B		2								—	—	↕	↕	0	↕	2
ROTR	ROTR.B Rd	B		2								—	—	↕	↕	0	↕	2
BSET	BSET #xx:3, Rd	B		2							(#xx:3 of Rd8)←1	—	—	—	—	—	—	2
	BSET #xx:3, @Rd	B			4						(#xx:3 of @Rd16)←1	—	—	—	—	—	—	8

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					実行ステート数*
			#xx:8/16	Rn	@Rn	@(d:16, Rn)	@-Rn/@Rn+	@aa:8/16	@(d:8, PC)	@@aa	–		I	H	N	Z	V	C	
BSET	BSET #xx:3, @aa:8	B						4				(#xx:3 of @aa:8)←1	–	–	–	–	–	–	8
	BSET Rn, Rd	B		2								(Rn8 of Rd8)←1	–	–	–	–	–	–	2
	BSET Rn, @Rd	B			4							(Rn8 of @Rd16)←1	–	–	–	–	–	–	8
	BSET Rn, @aa:8	B						4				(Rn8 of @aa:8)←1	–	–	–	–	–	–	8
BCLR	BCLR #xx:3, Rd	B		2								(#xx:3 of Rd8)←0	–	–	–	–	–	–	2
	BCLR #xx:3, @Rd	B			4							(#xx:3 of @Rd16)←0	–	–	–	–	–	–	8
	BCLR #xx:3, @aa:8	B						4				(#xx:3 of @aa:8)←0	–	–	–	–	–	–	8
	BCLR Rn, Rd	B		2								(Rn8 of Rd8)←0	–	–	–	–	–	–	2
	BCLR Rn, @Rd	B			4							(Rn8 of @Rd16)←0	–	–	–	–	–	–	8
	BCLR Rn, @aa:8	B						4				(Rn8 of @aa:8)←0	–	–	–	–	–	–	8
BNOT	BNOT #xx:3, Rd	B		2								(#xx:3 of Rd8)←(#xx:3 of Rd8)	–	–	–	–	–	–	2
	BNOT #xx:3, @Rd	B			4							(#xx:3 of @Rd16) ←(#xx:3 of @Rd16)	–	–	–	–	–	–	8
	BNOT #xx:3, @aa:8	B						4				(#xx:3 of @aa:8) ←(#xx:3 of @aa:8)	–	–	–	–	–	–	8
	BNOT Rn, Rd	B		2								(Rn8 of Rd8)←(Rn8 of Rd8)	–	–	–	–	–	–	2
	BNOT Rn, @Rd	B			4							(Rn8 of @Rd16)←(Rn8 of @Rd16)	–	–	–	–	–	–	8
	BNOT Rn, @aa:8	B						4				(Rn8 of @aa:8)←(Rn8 of @aa:8)	–	–	–	–	–	–	8
BTST	BTST #xx:3, Rd	B		2								(#xx:3 of Rd8)→Z	–	–	–	↕	–	–	2
	BTST #xx:3, @Rd	B			4							(#xx:3 of @Rd16)→Z	–	–	–	↕	–	–	6
	BTST #xx:3, @aa:8	B						4				(#xx:3 of @aa:8)→Z	–	–	–	↕	–	–	6
	BTST Rn, Rd	B		2								(Rn8 of Rd8)→Z	–	–	–	↕	–	–	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					実行ステート数*
			#xx:8/16	Rn	@Rn	@(d:16, Rn)	@-Rn/@Rn+	@aa:8/16	@(d:8, PC)	@@aa	-	I	H	N	Z	V	C	
BTST	BTST Rn, @Rd	B			4						(Rn8 of @Rd16)→Z	-	-	-	↕	-	-	6
	BTST Rn, @aa:8	B						4			(Rn8 of @aa:8)→Z	-	-	-	↕	-	-	6
BLD	BLD #xx:3, Rd	B		2							(#xx:3 of Rd8)→C	-	-	-	-	-	↕	2
	BLD #xx:3, @Rd	B			4						(#xx:3 of @Rd16)→C	-	-	-	-	-	↕	6
	BLD #xx:3, @aa:8	B						4			(#xx:3 of @aa:8)→C	-	-	-	-	-	↕	6
BILD	BILD #xx:3, Rd	B		2							(#xx:3 of $\overline{\text{Rd8}}$)→C	-	-	-	-	-	↕	2
	BILD #xx:3, @Rd	B			4						(#xx:3 of @$\overline{\text{Rd16}}$)→C	-	-	-	-	-	↕	6
	BILD #xx:3, @aa:8	B						4			(#xx:3 of @$\overline{\text{aa:8}}$)→C	-	-	-	-	-	↕	6
BST	BST #xx:3, Rd	B		2							C→(#xx:3 of Rd8)	-	-	-	-	-	-	2
	BST #xx:3, @Rd	B			4						C→(#xx:3 of @Rd16)	-	-	-	-	-	-	8
	BST #xx:3, @aa:8	B						4			C→(#xx:3 of @aa:8)	-	-	-	-	-	-	8
BIST	BIST #xx:3, Rd	B		2							$\overline{\text{C}}$→(#xx:3 of Rd8)	-	-	-	-	-	-	2
	BIST #xx:3, @Rd	B			4						$\overline{\text{C}}$→(#xx:3 of @Rd16)	-	-	-	-	-	-	8
	BIST #xx:3, @aa:8	B						4			$\overline{\text{C}}$→(#xx:3 of @aa:8)	-	-	-	-	-	-	8
BAND	BAND #xx:3, Rd	B		2							C∧(#xx:3 of Rd8)→C	-	-	-	-	-	↕	2
	BAND #xx:3, @Rd	B			4						C∧(#xx:3 of @Rd16)→C	-	-	-	-	-	↕	6
	BAND #xx:3, @aa:8	B						4			C∧(#xx:3 of @aa:8)→C	-	-	-	-	-	↕	6
BIAND	BIAND #xx:3, Rd	B		2							C∧($\overline{\text{#xx:3 of Rd8}}$)→C	-	-	-	-	-	↕	2
	BIAND #xx:3, @Rd	B			4						C∧($\overline{\text{#xx:3 of @Rd16}}$)→C	-	-	-	-	-	↕	6
	BIAND #xx:3, @aa:8	B						4			C∧($\overline{\text{#xx:3 of @aa:8}}$)→C	-	-	-	-	-	↕	6
BOR	BOR #xx:3, Rd	B		2							C∨(#xx:3 of Rd8)→C	-	-	-	-	-	↕	2
	BOR #xx:3, @Rd	B			4						C∨(#xx:3 of @Rd16)→C	-	-	-	-	-	↕	6
	BOR #xx:3, @aa:8	B						4			C∨(#xx:3 of @aa:8)→C	-	-	-	-	-	↕	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					実行ステート数*
			#xx:8/16	Rn	@Rn	@(d:16, Rn)	@-Rn/@Rn+	@aa:8/16	@(d:8, PC)	@@aa	-	分岐条件	I	H	N	Z	V	C	
BIOR	BIOR #xx:3, Rd	B		2							C∨(#xx:3 of $\overline{\text{Rd8}}$)→C	-	-	-	-	-	-	2
	BIOR #xx:3, @Rd	B			4						C∨(#xx:3 of @$\overline{\text{Rd16}}$)→C	-	-	-	-	-	-	6
	BIOR #xx:3, @aa:8	B					4				C∨(#xx:3 of @aa:8)→C	-	-	-	-	-	-	6
BXOR	BXOR #xx:3, Rd	B		2							C⊕(#xx:3 of Rd8)→C	-	-	-	-	-	-	2
	BXOR #xx:3, @Rd	B			4						C⊕(#xx:3 of @Rd16)→C	-	-	-	-	-	-	6
	BXOR #xx:3, @aa:8	B					4				C⊕(#xx:3 of @aa:8)→C	-	-	-	-	-	-	6
BIXOR	BIXOR #xx:3, Rd	B		2							C⊕(#xx:3 of $\overline{\text{Rd8}}$)→C	-	-	-	-	-	-	2
	BIXOR #xx:3, @Rd	B			4						C⊕(#xx:3 of @$\overline{\text{Rd16}}$)→C	-	-	-	-	-	-	6
	BIXOR #xx:3, @aa:8	B					4				C⊕(#xx:3 of @aa:8)→C	-	-	-	-	-	-	6
Bcc	BRA d:8 (BT d:8)	-							2		PC←PC+d:8	-	-	-	-	-	-	4
	BRN d:8 (BF d:8)	-							2		PC←PC+2	-	-	-	-	-	-	4
	BHI d:8	-							2		if condition C∨Z=0	-	-	-	-	-	-	4
	BLS d:8	-							2		is true then C∨Z=1	-	-	-	-	-	-	4
	BCC d:8 (BHS d:8)	-							2		PC←PC+d:8 C=0	-	-	-	-	-	-	4
	BCS d:8 (BLO d:8)	-							2		else next; C=1	-	-	-	-	-	-	4
	BNE d:8	-							2		Z=0	-	-	-	-	-	-	4
	BEQ d:8	-							2		Z=1	-	-	-	-	-	-	4
	BVC d:8	-							2		V=0	-	-	-	-	-	-	4
	BVS d:8	-							2		V=1	-	-	-	-	-	-	4
	BPL d:8	-							2		N=0	-	-	-	-	-	-	4
	BMI d:8	-							2		N=1	-	-	-	-	-	-	4
	BGE d:8	-							2		N⊕V=0	-	-	-	-	-	-	4
	BLT d:8	-							2		N⊕V=1	-	-	-	-	-	-	4
	BGT d:8	-							2		Z∨(N⊕V)=0	-	-	-	-	-	-	4
	BLE d:8	-							2		Z∨(N⊕V)=1	-	-	-	-	-	-	4

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						実行ステート数*
			#xx:8/16	Rn	@Rn	@(d:16, Rn)	@-Rn/@Rn+	@aa:8/16	@(d:8, PC)	@@aa	-		I	H	N	Z	V	C	
JMP	JMP @Rn	-			2							PC←Rn16	-	-	-	-	-	-	4
	JMP @aa:16	-						4				PC←aa:16	-	-	-	-	-	-	6
	JMP @@aa:8	-								2		PC←@aa:8	-	-	-	-	-	-	8
BSR	BSR d:8	-							2			SP-2→SP PC→@SP PC←PC+d:8	-	-	-	-	-	-	6
JSR	JSR @Rn	-			2							SP-2→SP PC→@SP PC←Rn16	-	-	-	-	-	-	6
	JSR @aa:16	-						4				SP-2→SP PC→@SP PC←aa:16	-	-	-	-	-	-	8
	JSR @@aa:8	-								2		SP-2→SP PC→@SP PC←@aa:8	-	-	-	-	-	-	8
RTS	RTS	-									2	PC←@SP SP+2→SP	-	-	-	-	-	-	8
RTE	RTE	-									2	CC←@SP SP+2→SP PC←@SP SP+2→SP	↕	↕	↕	↕	↕	↕	10

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						実行ステート数*
			#xx:8/16	Rn	@Rn	@(d:16, Rn)	@-Rn/@Rn+	@aa:8/16	@(d:8, PC)	@@aa	-		I	H	N	Z	V	C	
SLEEP	SLEEP	-									2	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
STC	STC CCR, Rd	B		2								CCR→Rd8	-	-	-	-	-	-	2
ANDC	ANDC #xx:8, CCR	B	2									CCR∧#xx:8→CCR	↕	↕	↕	↕	↕	↕	2
ORC	ORC #xx:8, CCR	B	2									CCR∨#xx:8→CCR	↕	↕	↕	↕	↕	↕	2
XORC	XORC #xx:8, CCR	B	2									CCR⊕#xx:8→CCR	↕	↕	↕	↕	↕	↕	2
NOP	NOP	-									2	PC←PC+2	-	-	-	-	-	-	2
EEPMOV	EEPMOV	-									4	if R4L≠0 Repeat @R5R→@R6 R5+1→R5 R6+1→R6 R4L-1→R4L Until R4L=0 else next;	-	-	-	-	-	-	④

【注】 *：実行ステート数は、オペコードおよびオペランドデータが内蔵メモリに存在する場合の値です。それ以外の場合は、「2.5 命令実行ステート数」を参照してください。
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：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。
③：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。
④：実行ステート数は、R4Lの設定値がnのとき4n＋9となります。
⑤：Eクロック同期転送命令の実行ステート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。

2.5 命令実行ステート数

H8/300L 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** 2.4 に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/ライトなどのサイクル数を示し、**表** 2.3 に各々のサイクル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。

$$\text{実行ステート数} = I \cdot S_I + J \cdot S_J + K \cdot S_K + L \cdot S_L + M \cdot S_M + N \cdot S_N$$
$$= 2 \cdot (I + J + K + L + M + N)$$

■ 実行ステート数計算例

（例）内蔵 ROM より命令をフェッチし、内蔵周辺モジュールをアクセスした場合

1. BSET #0, @FFC7
   表2.4より
   I＝L＝2, J＝K＝M＝N＝0
   表2.3より
   $S_I = 2$, $S_L = 2$
   実行ステート数＝2×(2＋2)＝8
   内蔵ROMより命令をフェッチし、内蔵ROMより分岐アドレスをリード、スタック領域は内蔵RAMとした場合
2. JSR @@30
   表2.4より
   I＝2, J＝K＝1, L＝M＝N＝0
   表2.3より
   $S_I = S_J = S_K = 2$
   実行ステート数＝2×(2＋1＋1)＝8

表 2.3 **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	
		内蔵メモリ	内蔵周辺モジュール
命令フェッチ	S_I	2	
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		2 または 3*
ワードデータアクセス	S_M		
内部動作	S_N	2	

【注】 * **内蔵周辺モジュール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。**

表 2.4 命令の実行状態（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 Rs, Rd	1					
ADDS	ADDS.W #1/2, Rd	1					
ADDX	ADDX.B #xx:8, Rd	1					
	ADDX.B Rs, Rd	1					
AND	AND.B #xx:8, Rd	1					
	AND.B Rs, Rd	1					
ANDC	ANDC #xx:8, CCR	1					
BAND	BAND #xx:3, Rd	1					
	BAND #xx:3, @Rd	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					
BCLR	BCLR #xx:3, Rd	1					
	BCLR #xx:3, @Rd	2			2		
	BCLR #xx:3, @aa:8	2			2		
	BCLR Rn, Rd	1					
	BCLR Rn, @Rd	2			2		
	BCLR Rn, @aa:8	2			2		
BIAND	BIAND #xx:3, Rd	1					
	BIAND #xx:3, @Rd	2			1		
	BIAND #xx:3, @aa:8	2			1		
BILD	BILD #xx:3, Rd	1					
	BILD #xx:3, @Rd	2			1		
	BILD #xx:3, @aa:8	2			1		

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
BIOR	BIOR #xx:3, Rd	1					
	BIOR #xx:3, @Rd	2			1		
	BIOR #xx:3, @aa:8	2			1		
BIST	BIST #xx:3, Rd	1					
	BIST #xx:3, @Rd	2			2		
	BIST #xx:3, @aa:8	2			2		
BIXOR	BIXOR #xx:3, Rd	1					
	BIXOR #xx:3, @Rd	2			1		
	BIXOR #xx:3, @aa:8	2			1		
BLD	BLD #xx:3, Rd	1					
	BLD #xx:3, @Rd	2			1		
	BLD #xx:3, @aa:8	2			1		
BNOT	BNOT #xx:3, Rd	1					
	BNOT #xx:3, @Rd	2			2		
	BNOT #xx:3, @aa:8	2			2		
	BNOT Rn, Rd	1					
	BNOT Rn, @Rd	2			2		
	BNOT Rn, @aa:8	2			2		
BOR	BOR #xx:3, Rd	1					
	BOR #xx:3, @Rd	2			1		
	BOR #xx:3, @aa:8	2			1		
BSET	BSET #xx:3, Rd	1					
	BSET #xx:3, @Rd	2			2		
	BSET #xx:3, @aa:8	2			2		
	BSET Rn, Rd	1					
	BSET Rn, @Rd	2			2		
	BSET Rn, @aa:8	2			2		
BSR	BSR d:8	2		1			
BST	BST #xx:3, Rd	1					
	BST #xx:3, @Rd	2			2		
	BST #xx:3, @aa:8	2			2		
BTST	BTST #xx:3, Rd	1					
	BTST #xx:3, @Rd	2			1		
	BTST #xx:3, @aa:8	2			1		
	BTST Rn, Rd	1					
	BTST Rn, @Rd	2			1		
	BTST Rn, @aa:8	2			1		
BXOR	BXOR #xx:3, Rd	1					
	BXOR #xx:3, @Rd	2			1		
	BXOR #xx:3, @aa:8	2			1		

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
CMP	CMP.B #xx:8, Rd	1					
	CMP.B Rs, Rd	1					
	CMP.W Rs, Rd	1					
DAA	DAA.B Rd	1					
DAS	DAS.B Rd	1					
DEC	DEC.B Rd	1					
DIVXU	DIVXU.B Rs, Rd	1					6
EEPMOV	EEPMOV	2			2n＋2*[1]		
INC	INC.B Rd	1					
JMP	JMP @Rn	2					
	JMP @aa:16	2					1
	JMP @@aa:8	2	1				1
JSR	JSR @Rn	2		1			
	JSR @aa:16	2		1			1
	JSR @@aa:8	2	1	1			
LDC	LDC #xx:8, CCR	1					
	LDC Rs, CCR	1					
MOV	MOV.B #xx:8, Rd	1					
	MOV.B Rs, Rd	1					
	MOV.B @Rs, Rd	1			1		
	MOV.B @(d:16, Rs), Rd	2			1		
	MOV.B @Rs+, Rd	1			1		1
	MOV.B @aa:8, Rd	1			1		
	MOV.B @aa:16, Rd	2			1		
	MOV.B Rs, @Rd	1			1		
	MOV.B Rs, @(d:16, Rd)	2			1		
	MOV.B Rs, @-Rd	1			1		1
	MOV.B Rs, @aa:8	1			1		
	MOV.B Rs, @aa:16	2			1		
	MOV.W #xx:16, Rd	2					
	MOV.W Rs, Rd	1					
	MOV.W @Rs, Rd	1				1	
	MOV.W @(d:16, Rs), Rd	2				1	
	MOV.W @Rs+, Rd	1				1	1
	MOV.W @aa:16, Rd	2				1	
	MOV.W Rs, @Rd	1				1	
	MOV.W Rs, @(d:16, Rd)	2				1	
	MOV.W Rs, @-Rd	1				1	1
	MOV.W Rs, @aa:16	2				1	
MULXU	MULXU.B Rs, Rd	1					6

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
NEG	NEG.B Rd	1					
NOP	NOP	1					
NOT	NOT.B Rd	1					
OR	OR.B #xx:8, Rd	1					
	OR.B Rs, Rd	1					
ORC	ORC #xx:8, CCR	1					
ROTL	ROTL.B Rd	1					
ROTR	ROTR.B Rd	1					
ROTXL	ROTXL.B Rd	1					
ROTXR	ROTXR.B Rd	1					
RTE	RTE	2		2			1
RTS	RTS	2		1			1
SHAL	SHAL.B Rd	1					
SHAR	SHAR.B Rd	1					
SHLL	SHLL.B Rd	1					
SHLR	SHLR.B Rd	1					
SLEEP	SLEEP	1					
STC	STC CCR, Rd	1					
SUB	SUB.B Rs, Rd	1					
	SUB.W Rs, Rd	1					
SUBS	SUBS.W #1/2, Rd	1					
SUBX	SUBX.B #xx:8, Rd	1					
	SUBX.B Rs, Rd	1					
XOR	XOR.B #xx:8, Rd	1					
	XOR.B Rs, Rd	1					
XORC	XORC #xx:8, CCR	1					

【注】 *1 n は R4L 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。

3. 処理状態

CPU の処理状態には、プログラム実行状態、例外処理状態、低消費電力状態の 3 種ウォッチ類があります。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、ウォッチモードがあります。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。

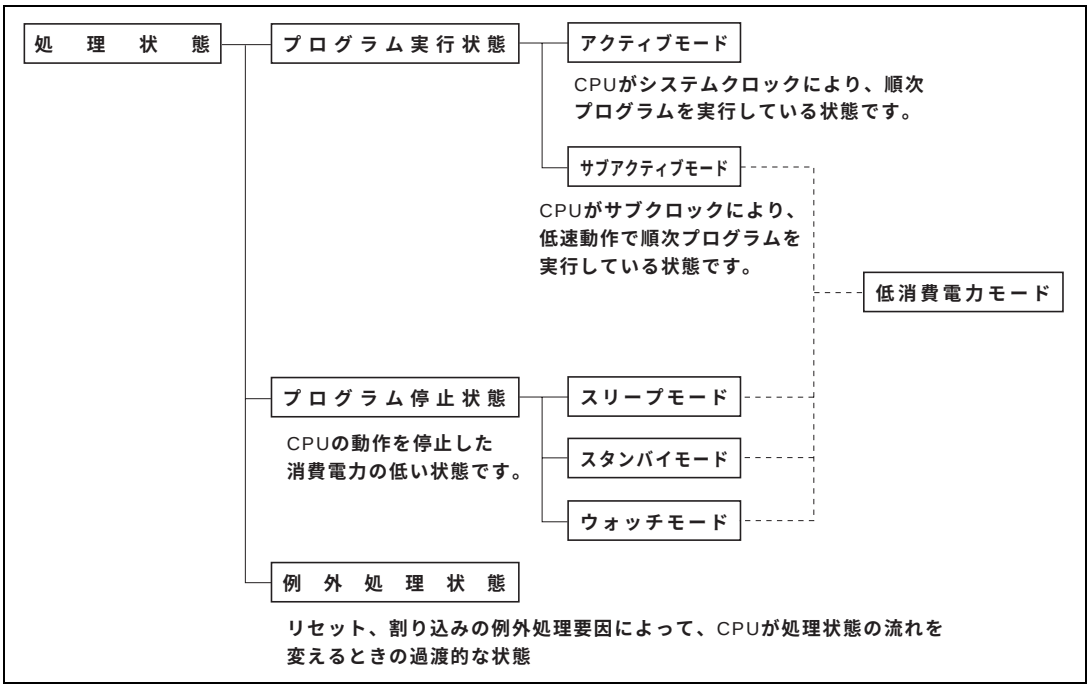


図 3.1　**処理状態の分類**

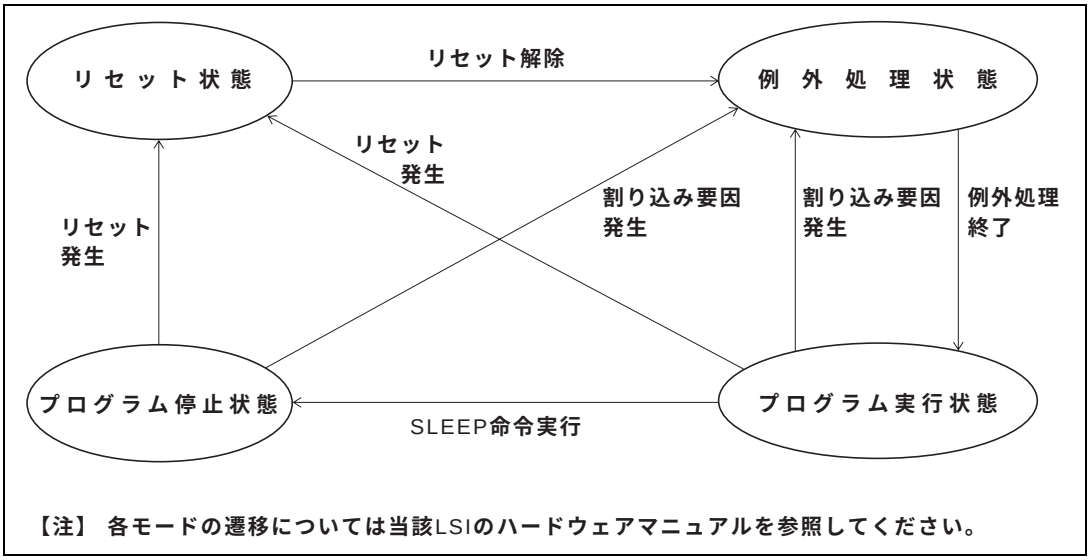


図 3.2 **状態遷移図**

3.1 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。

3.2 例外処理状態

リセット、割り込み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渡的な状態です。割り込み例外処理では、SP（R7）を参照して、PC および CCR の退避を行います。

3.2.1 例外処理の種類と優先度

例外処理には、リセットと割り込みがあります。**表** 3.1 に、例外処理の種類と優先度を示します。

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	$\overline{\text{RES}}$ 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。

【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。

3.2.2 例外処理要因とベクタテーブル

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。

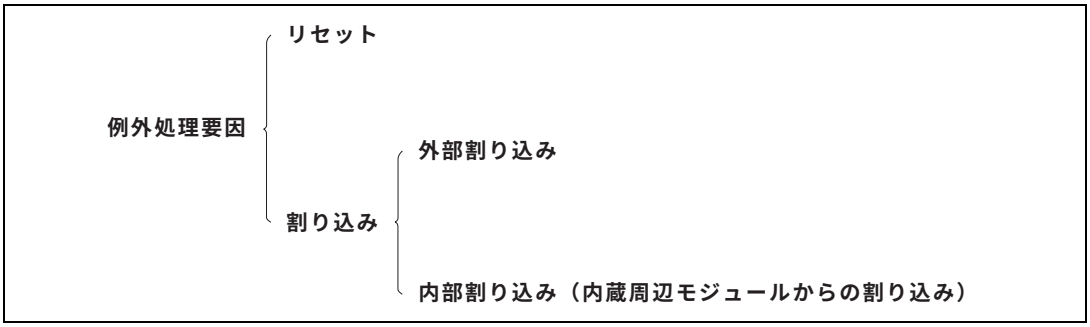


図 3.3 **例外処理要因の分類**

3.2.3 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。$\overline{\text{RES}}$ 端子を"Low"レベルにしてリセット状態にした後、$\overline{\text{RES}}$ 端子を"High"レベルにすると、リセット条件が成立した時点でリセット例外処理が起動されます。リセット条件についての詳細は当該 LSI のハードウェアマニュアルを参照してください。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。

割り込み例外処理が起動されると、CPU は SP（R7）を参照して PC と CCR をスタックに退避します。その後、CCR の I 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、その番地からプログラムの実行を開始します。例外処理終了後のスタックの状態を**図** 3.4 に示します。

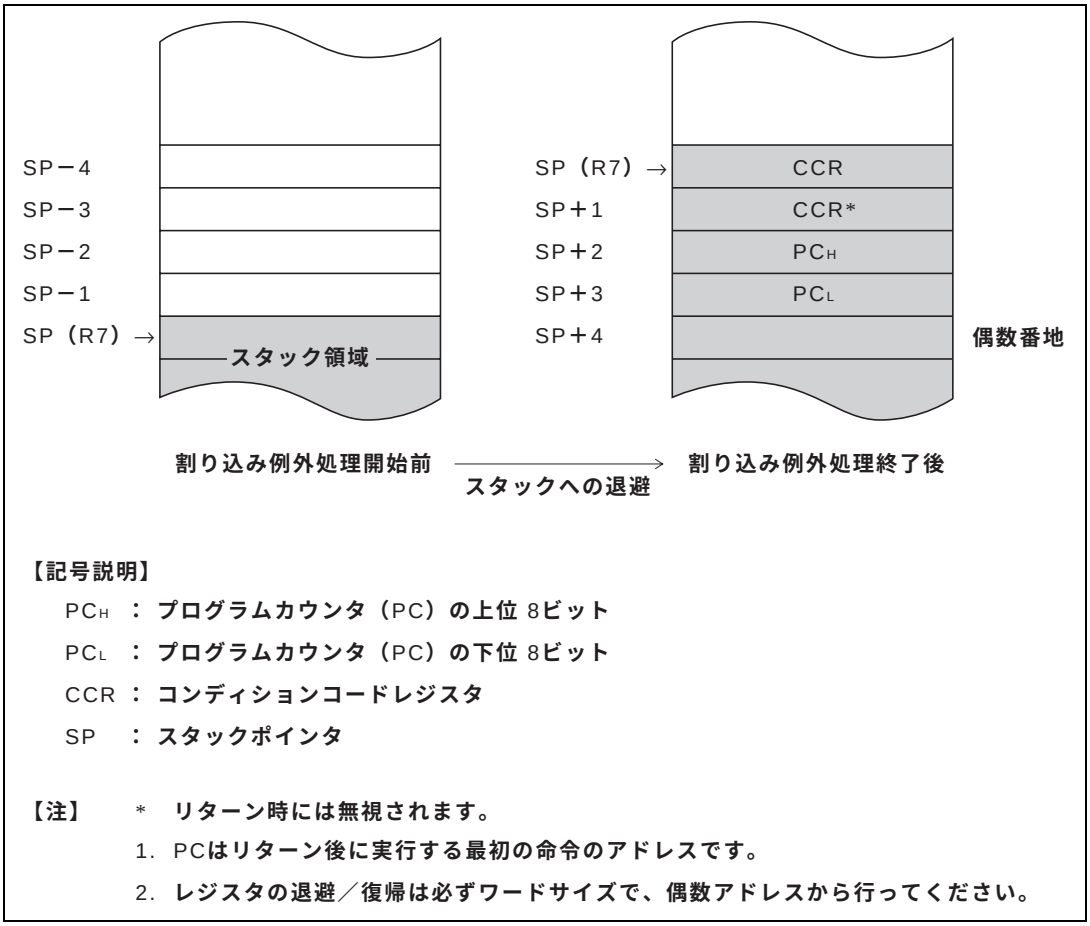


図 3.4 割り込み例外処理終了後のスタック状態

3.3 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。

外部から $\overline{\text{RES}}$ 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。

3.4 低消費電力状態

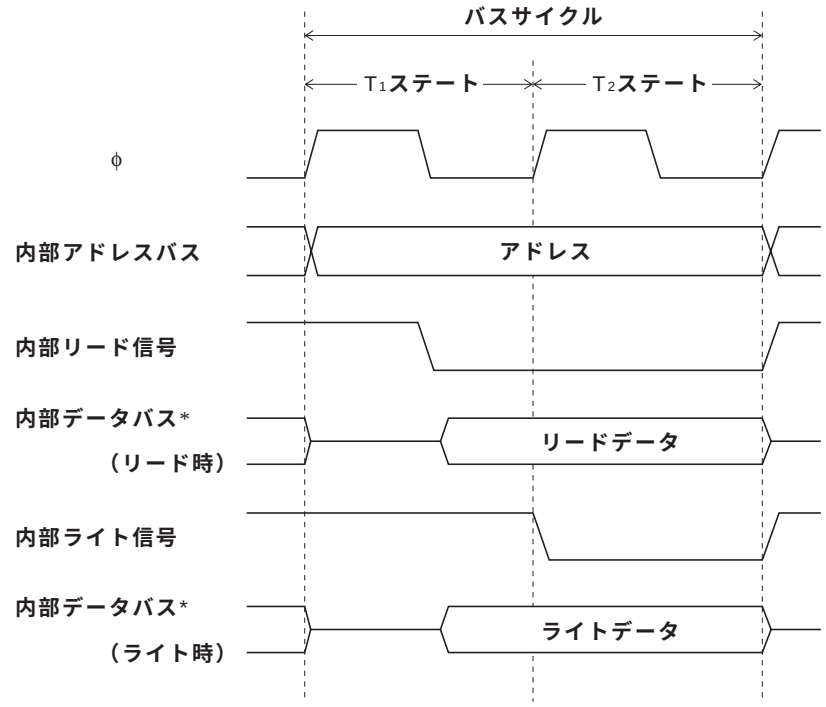
低消費電力状態は CPU の動作を停止して、消費電力を下げる状態です。スリープモード、ソフトウェアスタンバイモード、ウォッチモードがあります。各モードの詳細は当該 LSI のハードウェアマニュアルを参照してください。

4. 基本動作タイミング

CPU は、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ますが、内蔵メモリ、内蔵周辺モジュールによって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。

4.1 内蔵メモリ（RAM、ROM）

内蔵メモリのアクセスは、高速処理を行うために 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 に示します。



【注】 * ワードサイズのデータを2ステートでアクセスするために16ビットのデータバスを使用しています。

図 4.1 **内蔵メモリアクセスサイクル**

4.2 内蔵周辺モジュール

内蔵周辺モジュールのアクセスは、2 または 3 ステートで行われます。このとき、データバス幅は 8 ビットで、バイトデータのみアクセスが可能です。ワードデータおよび命令コードはアクセスできません。内蔵周辺モジュールアクセスサイクルを**図** 4.2 に示します。

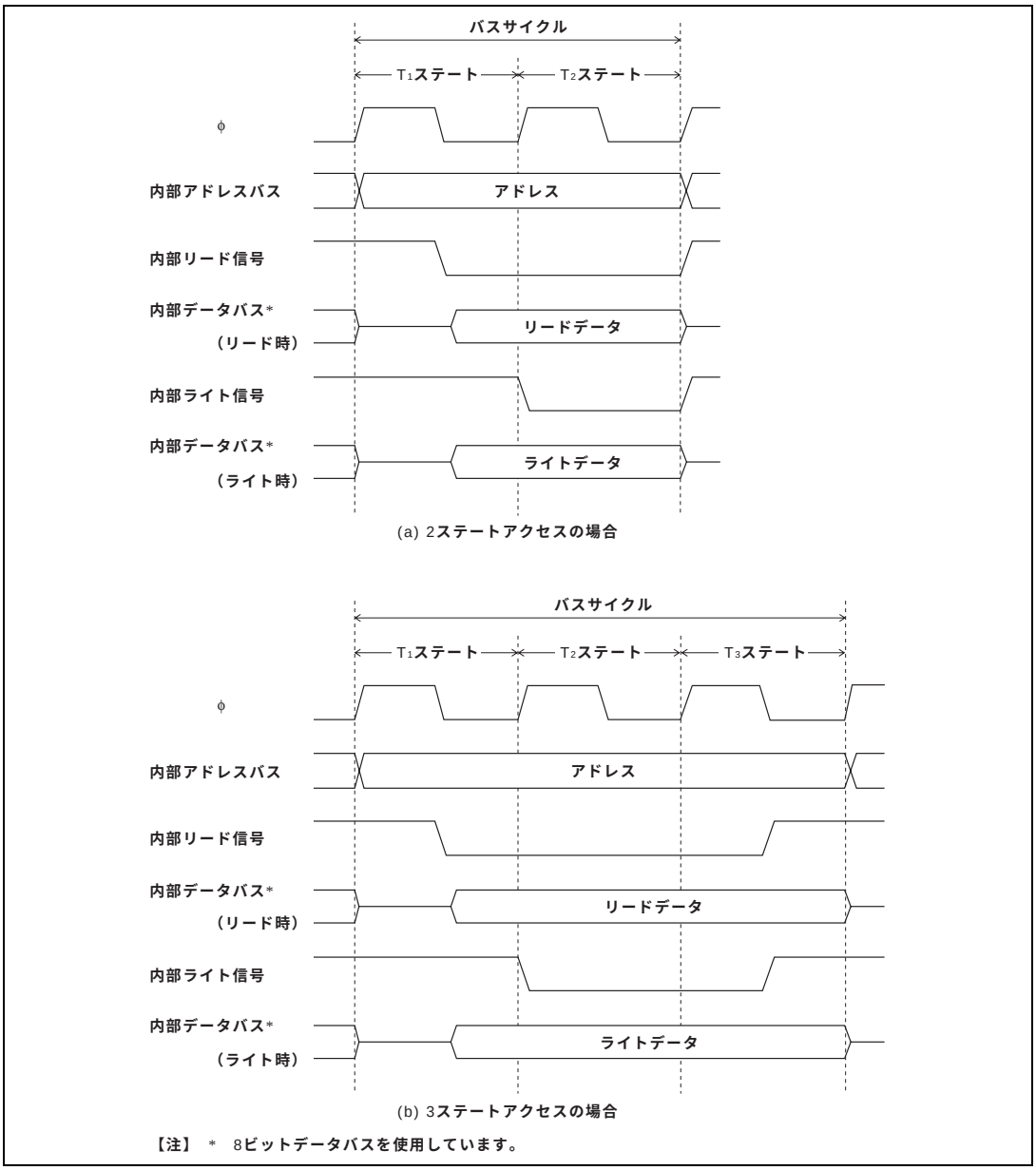


図 4.2 **内蔵周辺モジュールアクセスサイクル**

ルネサスシングルチップマイクロコンピュータ
ソフトウェアマニュアル
H8/300Lシリーズ

発行年月日　1991 年 8 月　　第 1 版
　　2006 年 6 月 23 日　Rev.2.00
発　　行　株式会社ルネサス テクノロジ 営業統括部
　　〒100-0004 東京都千代田区大手町 2-6-2
編　　集　株式会社ルネサスソリューションズ
　　グローバルストラテジックコミュニケーション本部
　　カスタマサポート部

株式会社ルネサス テクノロジ　営業統括部　〒100-0004 東京都千代田区大手町2-6-2 日本ビル

営業お問合せ窓口
株式会社ルネサス販売

http://www.renesas.com

本社	〒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
西東京支社	〒190-0023	立川市柴崎町2-2-23 (第二高島ビル2F)	(042) 524-8701
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
いわき支店	〒970-8026	いわき市平小太郎町4-9 (平小太郎ビル)	(0246) 22-3222
茨城支店	〒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
中部支社	〒460-0008	名古屋市中区栄4-2-29 (名古屋広小路プレイス)	(052) 249-3330
関西支社	〒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
広島支店	〒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
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

■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：コンタクトセンタ　E-Mail: csc@renesas.com

H8/300L シリーズ
ソフトウェアマニュアル

ルネサス エレクトロニクス株式会社
神奈川県川崎市中原区下沼部1753 〒211-8668

RJJ09B0342-0200