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。

RENESAS

ユーザーズ・マニュアル（暫定）

V850E2

32 ビット・マイクロプロセッサ・コア

アーキテクチャ編

資料番号 U17135JJ1V1UM00（第1版）
発行年月 June 2004 NS CP(K)

［メ　モ］

目次要約

CMOSデバイスの一般的注意事項

① 入力端子の印加波形

入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。

CMOSデバイスの入力がノイズなどに起因して，V_{IL}（MAX.）からV_{IH}（MIN.）までの領域にとどまるような場合は，誤動作を引き起こす恐れがあります。入力レベルが固定な場合はもちろん，V_{IL}（MAX.）からV_{IH}（MIN.）までの領域を通過する遷移期間中にチャタリングノイズ等が入らないようご使用ください。

② 未使用入力の処理

CMOSデバイスの未使用端子の入力レベルは固定してください。

未使用端子入力については，CMOSデバイスの入力に何も接続しない状態で動作させるのではなく，プルアップかプルダウンによって入力レベルを固定してください。また，未使用の入出力端子が出力となる可能性（タイミングは規定しません）を考慮すると，個別に抵抗を介してV_{DD}またはGNDに接続することが有効です。

資料中に「未使用端子の処理」について記載のある製品については，その内容を守ってください。

③ 静電気対策

MOSデバイス取り扱いの際は静電気防止を心がけてください。

MOSデバイスは強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には，当社が出荷梱包に使用している導電性のトレーやマガジン・ケース，または導電性の緩衝材，金属ケースなどを利用し，組み立て工程にはアースを施してください。プラスチック板上に放置したり，端子を触ったりしないでください。

また，MOSデバイスを実装したボードについても同様の扱いをしてください。

④ 初期化以前の状態

電源投入時，MOSデバイスの初期状態は不定です。

電源投入時の端子の出力状態や入出力設定，レジスタ内容などは保証しておりません。ただし，リセット動作やモード設定で定義している項目については，これらの動作ののちに保証の対象となります。

リセット機能を持つデバイスの電源投入後は，まずリセット動作を実行してください。

本製品のうち，外国為替及び外国貿易法の規定により規制貨物等（または役務）に該当するものについては，日本国外に輸出する際に，同法に基づき日本国政府の輸出許可が必要です。

- **本資料は，この製品の企画段階で作成していますので，予告なしに内容を変更することがあります。また本資料で扱う製品の製品化を中止することがあります。**
- **文書による当社の承諾なしに本資料の転載複製を禁じます。当社は，本資料の誤りに関し，一切その責を負いません。**
- **当社は，本資料に掲載された当社製品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，一切その責を負いません。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。**
- **本資料に記載された回路，ソフトウエ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，ソフトウエアおよびこれらに関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，一切その責を負いません。**
- **当社は，当社製品の品質，信頼性の向上に努めておりますが，当社製品の不具合が完全に発生しないことを保証するものではありません。当社製品の不具合により生じた生命，身体および財産に対する損害の危険を最小限度にするために，冗長設計，延焼対策設計，誤動作防止設計等安全設計を行ってください。**
- **当社は，当社製品の品質水準を「標準水準」，「特別水準」およびお客様に品質保証プログラムを指定していただく「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。**

  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

  特別水準：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

  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，生命維持のための装置またはシステム等

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。意図されていない用途で当社製品の使用をお客様が希望する場合には，事前に当社販売窓口までお問い合わせください。

（注）

（1）本事項において使用されている「当社」とは，NECエレクトロニクス株式会社およびNEC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う。

（2）本事項において使用されている「当社製品」とは，（1）において定義された当社の開発，製造製品をいう。

M5　02.11

はじめに

対象者　このマニュアルは，V850E2 CPU コアの機能を理解し，それを用いたアプリケーション・システムを設計しようとするユーザを対象とします。

目　的　このマニュアルは，次の構成に示す V850E2 CPU コアのアーキテクチャをユーザに理解していただくことを目的としています。

構　成　このマニュアルは，おもに次の内容で構成しております。

- レジスタ・セット
- データ・タイプ
- 命令形式と命令セット
- 割り込みと例外
- パイプライン
- ディバグ・モードへの移行

読み方　このマニュアルの読者には，電気，論理回路，マイクロコンピュータの一般知識を必要とします。

ハードウエアの機能について知りたいとき
　→**各製品のユーザーズ・マニュアル　ハードウエア編**をお読みください。

特定の命令の機能を詳細に調べたいとき
　→**第 5 章　命　　令**をお読みください。

凡　例

データ表記の重み	：左が上位桁，右が下位桁
アクティブ・ロウの表記	：xxxB（端子，信号名称のあとに B）
注	：本文中につけた注の説明
注意	：気をつけて読んでいただきたい内容
備考	：本文の補足説明
数の表記	：2 進数 …xxxx または xxxxB
	10 進数…xxxx
	16 進数…xxxxH

2 のべき数を示す接頭語（アドレス空間，メモリ容量）：

K（キロ）… $2^{10} = 1024$

M（メガ）… $2^{20} = 1024^2$

G（ギガ）… $2^{30} = 1024^3$

目　次

第1章　概　　説

リアルタイム制御システムとして，HDD（Hard disk drive），PPC（Plain paper copier），プリンタ，ファクシミリをはじめとするOA機器，エンジン制御，ABS（Antilock braking system）制御などの各種自動車電装機器，NC（Numerical control）工作機，各種コントローラなどのFA機器などがあります。従来，これらの分野では8ビットまたは16ビットのマイクロコンピュータが用いられてきましたが，機器の制御の複雑化にしたがって，マイクロコンピュータに要求される機能も高度化し，命令セットが複雑になり，ハードウエア規模が増大化してきました。同時に，機器の性能向上に対応すべくマイクロコンピュータの動作周波数の高速化もあわせて求められています。

これらの要求に答えるために開発されたV850シリーズは，よりシンプルなハードウエア構成により最大限の性能を実現できる手段としてRISCアーキテクチャを取り入れることにより，従来のCISC型シングルチップ・マイクロコンピュータ78K/IIIシリーズ，78K/IVシリーズの約15倍の性能を低コストで実現する次世代のシングルチップ・マイクロコンピュータとして位置付けられます。

V850シリーズに搭載の「V850 CPU」では，従来のRISC型CPUの基本命令に加えて，各種応用機器，特にディジタル・サーボ制御の応用に最適な命令として，ハードウエア乗算器による乗算命令，飽和演算命令，ビット操作命令などを用意しました。また，コンパイラにおける高コード効率を最優先に意識した命令フォーマットを採用して，オブジェクト・コード・サイズのコンパクト化を図っています。

「V850E2 CPU」は，この「V850 CPU」の後継となる「V850E1 CPU」の性能をさらに強化しています。

「V850E2 CPU」は，割り込み応答性能を重視し，CPUのクロック周波数を向上させるために，パイプラインを7段にしました。また，CPUの処理能力を向上させるために，パイプライン処理の並列化を行い，プログラム領域を512Mバイトまで拡張し，さらに，高速なメモリ・アクセスを実現させるために，命令／データ・キャッシュ・インタフェースを内蔵しました。

なお，命令コードは，V850 CPU, V850E1 CPUに対して，オブジェクト・コード・レベルでの上位互換性を持たせているため，従来のシステムのソフトウエア資産をそのまま使用できます。

1.1 特　徴

(1) 組み込み制御用高性能 32 ビット・アーキテクチャ

- **命令数：**93
- **32 ビット汎用レジスタ：32 本**
- **ロング／ショート形式を持つロード／ストア命令**
- **3 オペランド命令**
- **1 クロック・ピッチの 7 段パイプライン構造**
- **レジスタ／フラグ・ハザードのインタロックをハードウエアにより対処**
- **メモリ空間：プログラム領域** … 512M **バイト・リニア**
  データ領域 … 4G **バイト・リニア**

(2) 各種応用分野に適した命令群

- **飽和演算命令**
- **ビット操作命令**
- **乗算命令（ハードウエア乗算器内蔵により，1 クロックでの乗算処理が可能）**
  16 ビット×16 ビット → **32 ビット**
  32 ビット×32 ビット → **32 ビット，または 64 ビット**
- MAC **演算命令**
  32 ビット×32 ビット＋64 ビット → **64 ビット**

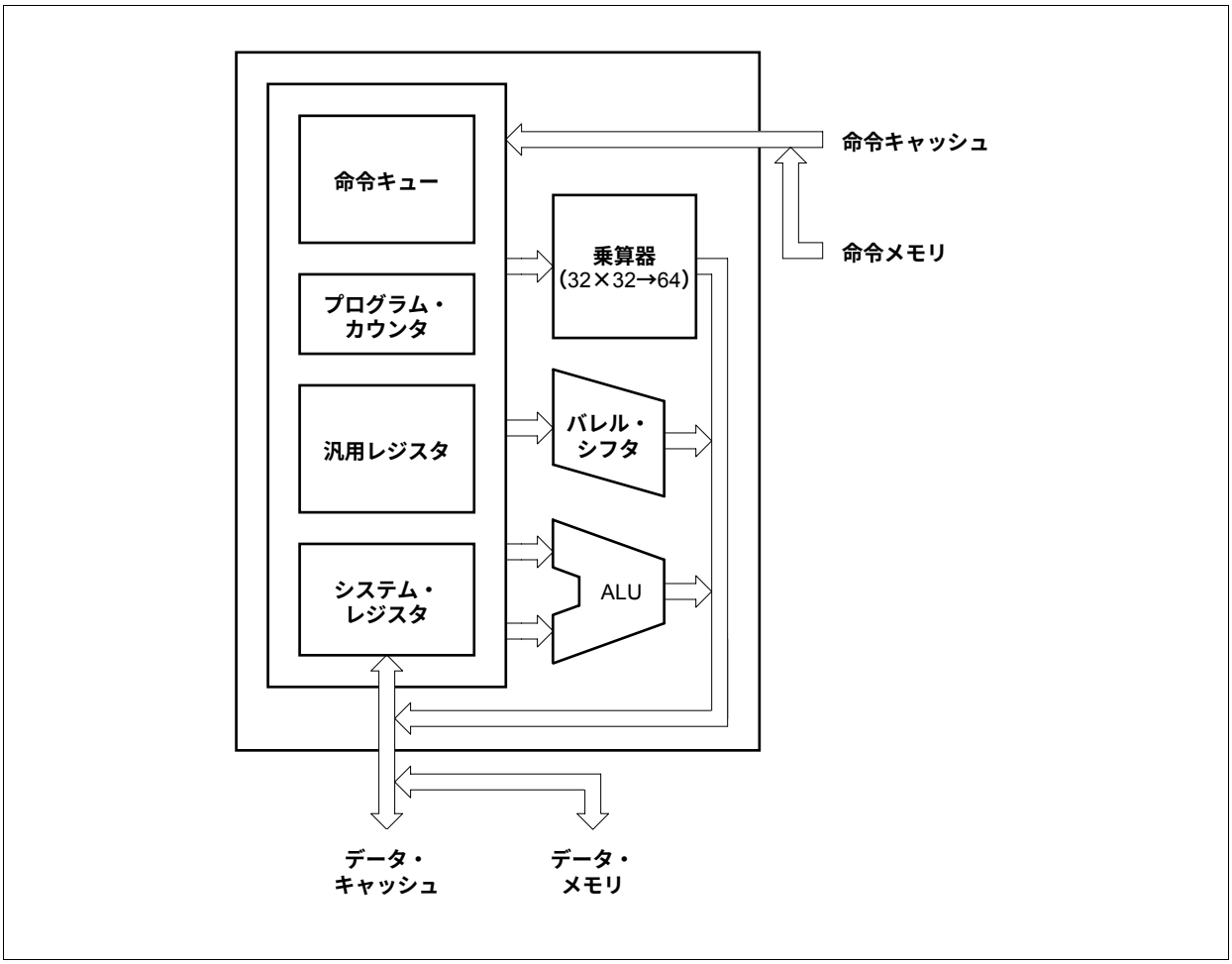
1.2 内部構成

V850E2 CPU は，アドレス計算，算術論理演算，データ転送などのほとんどの命令処理を 7 段パイプライン制御により 1 クロックで実行します。

乗算器（32 ビット ×32 ビット乗算），バレル・シフタ（32 ビット／1 クロック）などの専用ハードウエアを内蔵し，複雑な命令処理の高速化を図っています。

内部ブロック図を図 1－1 に示します。

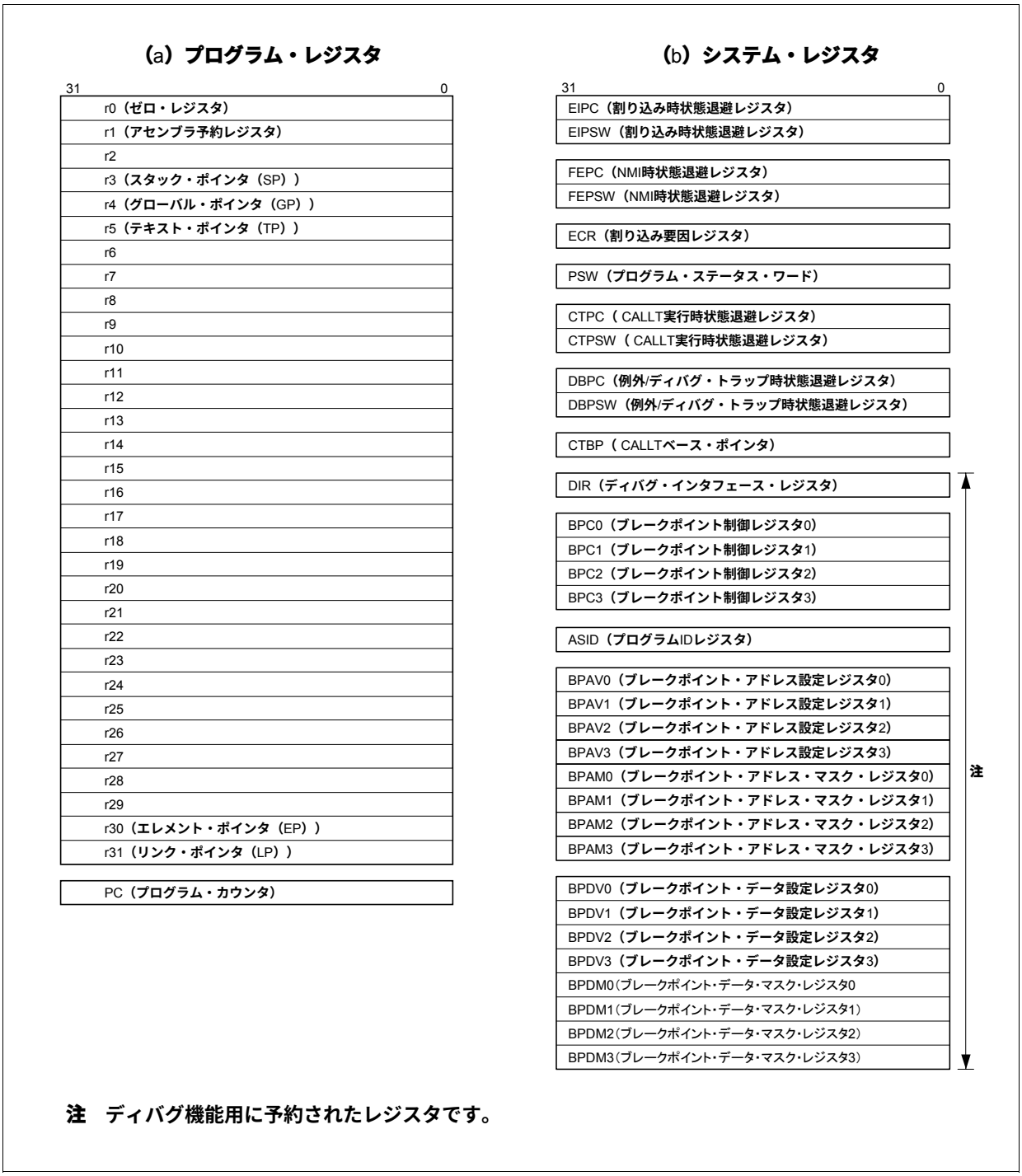
図1－1　内部ブロック図



第2章　レジスタ・セット

レジスタは，一般のプログラム用として使用するプログラム・レジスタと実行環境を制御するシステム・レジスタの2つに分類できます。すべて32ビット・レジスタです。

図2－1　レジスター覧



注　ディバグ機能用に予約されたレジスタです。

2. 1　プログラム・レジスタ

プログラム・レジスタには，汎用レジスタ（r0-r31）とプログラム・カウンタ（PC）があります。

表2－1　プログラム・レジスター覧

プログラム・レジスタ	名　称	機　　能	説　　明
汎用レジスタ	r0	ゼロ・レジスタ	常に 0 を保持
	r1	アセンブラ予約レジスタ	アドレス生成用のワーキング・レジスタとして使用
	r2	アドレス／データ変数用レジスタ（使用するリアルタイム OS がこのレジスタを使用していない場合）	
	r3	スタック・ポインタ（SP）	関数コール時のスタック・フレーム生成時に使用
	r4	グローバル・ポインタ（GP）	データ領域のグローバル変数をアクセスするときに使用
	r5	テキスト・ポインタ（TP）	テキスト領域（プログラム・コードを配置する領域）の先頭を示すレジスタとして使用
	r6-r29	アドレス／データ変数用レジスタ	
	r30	エレメント・ポインタ（EP）	メモリをアクセスするときのアドレス生成用ベース・ポインタとして使用
	r31	リンク・ポインタ（LP）	コンパイラが関数コールをするときに使用
プログラム・カウンタ	PC	プログラム実行中の命令アドレスを保持	

備考　アセンブラや C コンパイラで使用される r1, r3-r5, r31 の詳細な説明は，CA850（C コンパイラ・パッケージ）ユーザーズ・マニュアル　アセンブリ言語編を参照してください。

(1) 汎用レジスタ（r0-r31）

汎用レジスタとして，r0-r31 の 32 本が用意されています。これらのレジスタは，すべてデータ変数用またはアドレス変数用として利用できます。

ただし，r0-r5, r30, r31 を使用する際には次のような注意が必要です。

(a) r0, r30

命令により暗黙的に使用されます。

r0は常に0を保持しているレジスタであり，0を使用する演算やオフセット0のアドレシングで使用されます。

r30はSLD命令とSST命令により，メモリをアクセスするときのベース・ポインタとして使用されます。

(b) r1, r3-r5, r31

アセンブラとCコンパイラにより暗黙的に使用されます。

これらのレジスタを使用する際には，レジスタの内容を破壊しないように退避してから使用し，使用後に元に戻す必要があります。

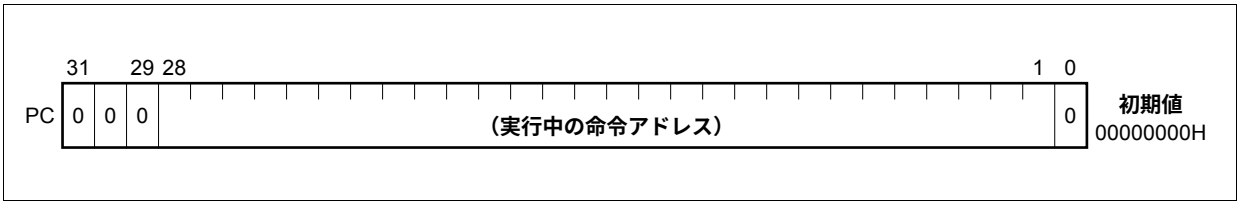
(c) r2

リアルタイムOSが使用する場合があります。使用するリアルタイムOSがr2を使用していない場合は，アドレス変数用またはデータ変数用レジスタとして利用できます。

(2) プログラム・カウンタ（PC）

プログラム実行中の命令アドレスを保持しています。下位の 29 ビットが有効で，ビット 31-29 は将来の機能拡張のために予約されています（0 に固定）。ビット 28 からビット 29 へのキャリーがあっても無視します。また，ビット 0 は 0 に固定されており，奇数番地への分岐はできません。

図2－2　プログラム・カウンタ（PC）



2. 2　システム・レジスタ

システム・レジスタは，状態制御，割り込み情報保持などを行います。

システム・レジスタへのリード／ライトは，LDSR 命令, STSR 命令により，次に示すシステム・レジスタ番号を指定することで行います。

表2－2　システム・レジスター覧

システム・レジスタ番号	システム・レジスタ名	オペランド指定の可否	
		LDSR 命令	STSR 命令
0	割り込み時状態退避レジスタ（EIPC）	○	○
1	割り込み時状態退避レジスタ（EIPSW）	○	○
2	NMI 時状態退避レジスタ（FEPC）	○	○
3	NMI 時状態退避レジスタ（FEPSW）	○	○
4	割り込み要因レジスタ（ECR）	×	○
5	プログラム・ステータス・ワード（PSW）	○	○
6-15	（将来の機能拡張のための予約番号（アクセスした場合の動作は保証しません））	×	×
16	CALLT 実行時状態退避レジスタ（CTPC）	○	○
17	CALLT 実行時状態退避レジスタ（CTPSW）	○	○
18	例外／ディバグ・トラップ時状態退避レジスタ（DBPC）	○	○
19	例外／ディバグ・トラップ時状態退避レジスタ（DBPSW）	○	○
20	CALLT ベース・ポインタ（CTBP）	○	○
21	ディバグ・インタフェース・レジスタ（DIR）	○注	○
22-27	DIR レジスタで設定されるチャネルにより異なります（表 2－3 参照）。	－	－
28-31	（将来の機能拡張のための予約番号（アクセスした場合の動作は保証しません））	×	×

注　ユーザ・モード時に読み出し値が不定となるビットがあります（2. 2. 8　ディバグ・インタフェース・レジスタ（DIR）参照）。

注意　LDSR 命令により EIPC レジスタか FEPC レジスタ，または CTPC レジスタのビット 0 をセット（1）したあと，割り込み処理を行い，RETI 命令で復帰するときにビット 0 の値は無視されます（PC レジスタのビット 0 を 0 に固定してあるため）。EIPC，FEPC，CTPC レジスタに値を設定する場合は，特別な理由のないかぎり，偶数値（ビット 0 = 0）を設定してください。

備考　○：アクセス可能
　　×：アクセス禁止

表2－3　システム・レジスター覧（システム・レジスタ番号：22-27）

システム・レジスタ番号	レジスタ名	オペランド指定の可否	
		LDSR命令	STSR命令
22	ブレークポイント制御レジスタn（BPCn）	○	○
23	プログラムIDレジスタ（ASID）	○	○
24	ブレークポイント・アドレス設定レジスタn（BPAVn）	○	○
25	ブレークポイント・アドレス・マスク・レジスタn（BPAMn）	○	○
26	ブレークポイント・データ設定レジスタn（BPDVn）	○	○
27	ブレークポイント・データ・マスク・レジスタn（BPDMn）	○	○

備考　n = 0-3

○：アクセス可能

2. 2. 1　割り込み時状態退避レジスタ（EIPC, EIPSW）

割り込み時状態退避レジスタには，EIPC レジスタと EIPSW レジスタがあります。

ソフトウエア例外やマスカブル割り込みまたはラン・タイム・エラー例外が発生した場合，プログラム・カウンタ（PC）の内容が EIPC レジスタに，プログラム・ステータス・ワード（PSW）の内容が EIPSW レジスタに退避されます（ノンマスカブル割り込み（NMI）発生時には，NMI 時状態退避レジスタ（FEPC, FEPSW）に退避されます）。

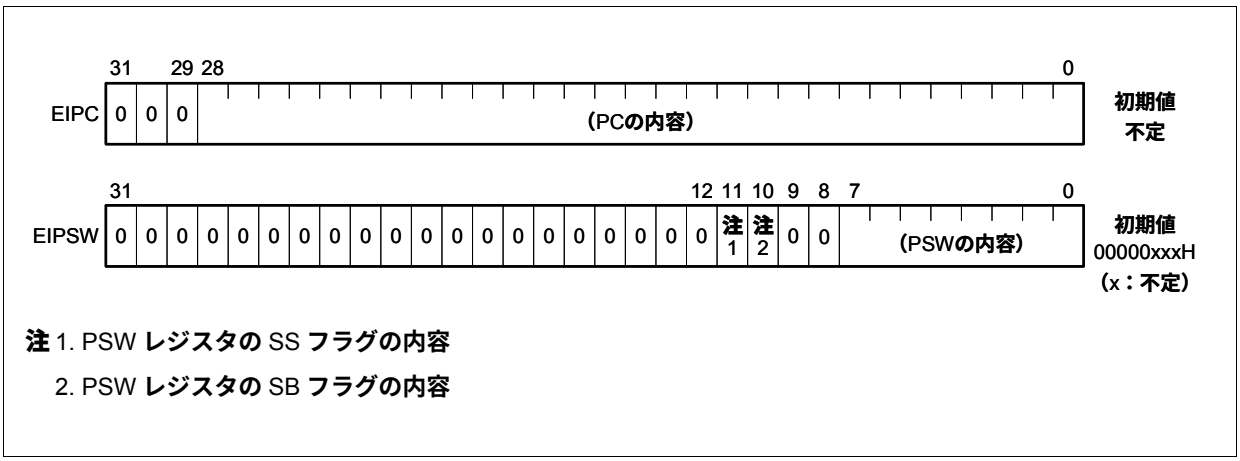
EIPC レジスタには，一部の命令を除き，ソフトウエア例外やマスカブル割り込み，またはラン・タイム・エラー例外が発生したときに実行していた命令の次の命令のアドレスが退避されます（表 6－1　**割り込み，例外コード一覧**参照）。

EIPSW レジスタには，現在の PSW レジスタの内容が退避されます。

割り込み時状態退避レジスタは，1 組しかないため，多重割り込みを行う場合はプログラムによってこれらのレジスタの内容を退避する必要があります。

なお，EIPC レジスタのビット 31-29 と EIPSW レジスタのビット 31-12, 9, 8 は，将来の機能拡張のために予約されています（0 に固定）。

図2－3　割り込み時状態退避レジスタ（EIPC, EIPSW）



注 1. PSW レジスタの SS フラグの内容

2. PSW レジスタの SB フラグの内容

2. 2. 2　NMI 時状態退避レジスタ（FEPC, FEPSW）

NMI 時状態退避レジスタには，FEPC レジスタと FEPSW レジスタがあります。

ノンマスカブル割り込み（NMI），またはラン・タイム・エラー例外が発生した場合，プログラム・カウンタ（PC）の内容が FEPC レジスタに，プログラム・ステータス・ワード（PSW）の内容が FEPSW レジスタに退避されます。

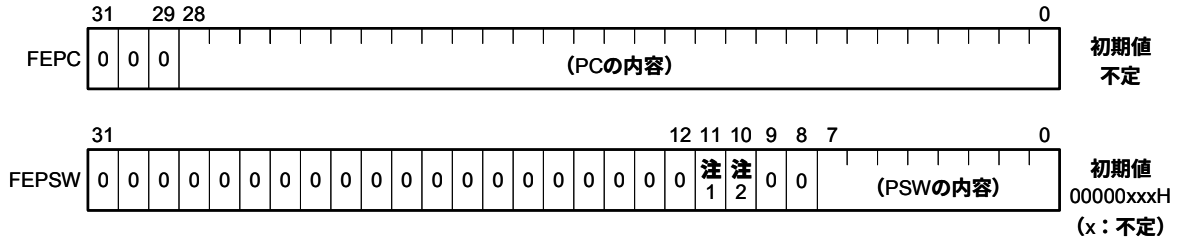
FEPC レジスタには，一部の命令を除き，NMI，またはラン・タイム・エラー例外が発生したときに実行していた命令の次の命令のアドレスが退避されます（表 6－1　割り込み，例外コード一覧参照）。

FEPSW レジスタには，現在の PSW の内容が退避されます。

NMI 時状態退避レジスタは，1 組しかないため，多重割り込みを行う場合はプログラムによってこれらのレジスタの内容を退避する必要があります。

なお，FEPC レジスタのビット 31-29 と FEPSW レジスタのビット 31-12, 9, 8 は，将来の機能拡張のために予約されています（0 に固定）。

図2－4　NMI時状態退避レジスタ（FEPC, FEPSW）



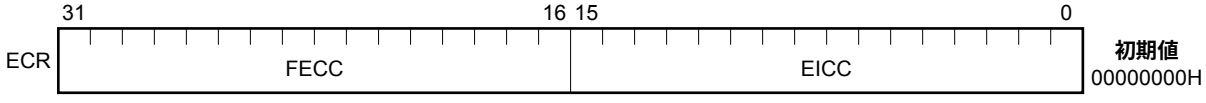
注 1. PSW レジスタの SS フラグの内容

2. PSW レジスタの SB フラグの内容

2. 2. 3　割り込み要因レジスタ（ECR）

割り込み要因レジスタ（ECR）は，例外や割り込みが発生した場合に，その要因を保持するレジスタです。ECR レジスタが保持する値は，割り込み要因ごとにコード化された例外コードです（表 6－1　割り込み，例外コード一覧参照）。なお，このレジスタは読み出し専用のため，LDSR 命令を使ってこのレジスタにデータを書き込むことはできません。

図2－5　割り込み要因レジスタ（ECR）



ビット位置	ビット名	意　味
31-16	FECC	ノンマスカブル割り込み（NMI）の例外コード（初期値：0）
15-0	EICC	例外，マスカブル割り込みの例外コード（初期値：0）

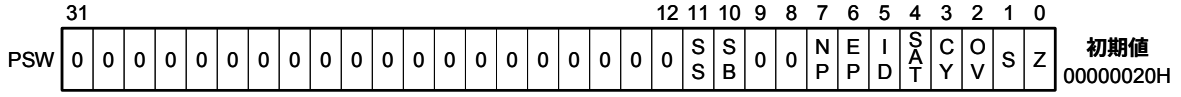
2. 2. 4　プログラム・ステータス・ワード（PSW）

プログラム・ステータス・ワード（PSW）は，プログラムの状態（命令実行の結果）や CPU の状態を示すフラグの集合です。

LDSR 命令を使用してこのレジスタの各ビットの内容を変更した場合は，LDSR 命令実行終了直後から変更内容が有効となります。ただし，ID フラグをセット（1）する場合，LDSR 命令実行中から割り込み要求の受け付けを禁止します。

なお，ビット 31-12, 9, 8 は，将来の機能拡張のために予約されており，0 以外の書き込みを禁止します。また，読み出した場合の値は不定です。

図2－6　プログラム・ステータス・ワード（PSW）（1/2）



ビット位置	フラグ名	意　味
11	SS	このフラグをセット（1）すると，シングルステップ動作（各命令の実行ごとにディバグ・トラップを発生）を行います（初期値：0）。 ただし，割り込み処理ルーチンに分岐する際は，SB フラグの内容が転送されます。そのため，SB フラグがクリア（0）されていると，割り込み処理ルーチンでのシングルステップ動作は行いません。 また，DIR レジスタの SSE ビット = 0 の場合は，このフラグはセットされません（0 に固定）。
10	SB	割り込み処理ルーチンに分岐する際に，このフラグの内容が SS フラグに転送されます（初期値：0）。 したがって，このフラグをセット（1）しておくと，割り込み処理ルーチンでのシングルステップ動作が可能となります。

図2－6　プログラム・ステータス・ワード（PSW）（2/2）

ビット位置	フラグ名	意　味
7	NP	ノンマスカブル割り込み（NMI）処理中であることを示します。NMI 要求が受け付けられるとセット（1）され，多重割り込みを禁止します。 0：NMI 処理中でない（初期値） 1：NMI 処理中である
6	EP	例外処理中であることを示します。例外の発生でセット（1）されます。なお，このビットがセット（1）されても割り込み要求は受け付けます。 0：例外処理中でない（初期値） 1：例外処理中である
5	ID	マスカブル割り込み要求を受け付ける状態かどうかを示します。 0：割り込み可能（EI） 1：割り込み不可（DI）（初期値）
4	SAT 注	飽和演算命令の演算結果がオーバフローし，演算結果が飽和していることを示します。累積フラグのため，飽和演算命令で演算結果が飽和するとセット（1）され，以降の命令の演算結果が飽和しなくてもクリア（0）されません。クリア（0）する場合は，LDSR 命令により行います。なお，算術演算命令の実行では，セット（1）もクリア（0）も行いません。 0：飽和していない（初期値） 1：飽和している
3	CY	演算結果にキャリー，またはボローがあったかどうかを示します。 0：キャリー，またはボローは発生していない（初期値） 1：キャリー，またはボローが発生した
2	OV 注	演算中にオーバフローが発生したかどうかを示します。 0：オーバフローは発生していない（初期値） 1：オーバフローが発生した
1	S 注	演算の結果が負かどうかを示します。 0：演算の結果は，正または 0 であった（初期値） 1：演算の結果は負であった
0	Z	演算の結果が 0 かどうかを示します。 0：演算の結果は 0 でなかった（初期値） 1：演算の結果は 0 であった

注　飽和演算時の OV フラグと S フラグの内容で飽和処理した演算結果が決まります。また，飽和演算時に OV フラグがセット（1）された場合だけ，SAT フラグはセット（1）されます。

演算結果の状態	フラグの状態			飽和処理をした演算結果
	SAT	OV	S	
正の最大値を越えた	1	1	0	7FFFFFFFH
負の最大値を越えた	1	1	1	80000000H
正（最大値を越えない）	演算前の値を保持	0	0	演算結果そのもの
負（最大値を越えない）			1	

2. 2. 5　CALLT 実行時状態退避レジスタ（CTPC, CTPSW）

CALLT 実行時状態退避レジスタには，CTPC レジスタと CTPSW レジスタがあります。

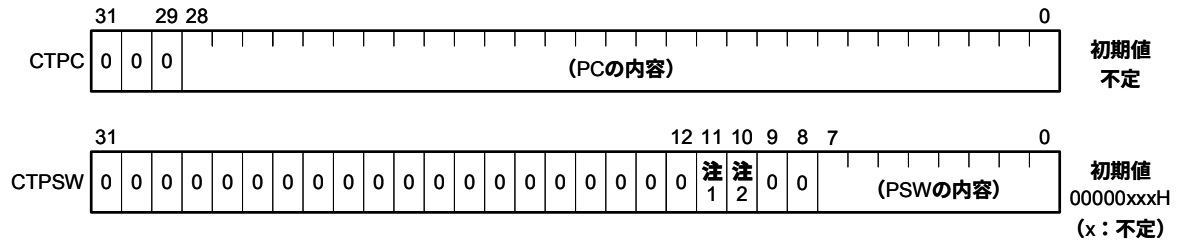
CALLT 命令が実行されると，プログラム・カウンタ（PC）の内容が CTPC レジスタに，プログラム・ステータス・ワード（PSW）の内容が CTPSW レジスタに退避されます。

CTPC レジスタに退避される内容は，CALLT 命令の次の命令のアドレスです。

CTPSW レジスタには，現在の PSW レジスタの内容が退避されます。

なお，CTPC レジスタのビット 31-29 と CTPSW レジスタのビット 31-12, 9, 8 は，将来の機能拡張のために予約されています（0 に固定）。

図2－7　CALLT実行時状態退避レジスタ（CTPC, CTPSW）



注 1. PSW レジスタの SS フラグの内容

2. PSW レジスタの SB フラグの内容

2. 2. 6　例外／ディバグ・トラップ時状態退避レジスタ（DBPC, DBPSW）

例外／ディバグ・トラップ時状態退避レジスタには，DBPC レジスタと DBPSW レジスタがあります。

例外トラップ，ディバグ・トラップ，ディバグ・ブレークの発生，またはシングルステップ動作の実行時に，プログラム・カウンタ（PC）の内容が DBPC レジスタに，プログラム・ステータス・ワード（PSW）の内容が DBPSW レジスタに退避されます。

DBPC レジスタに退避される内容は次のとおりです。

表2－4　DBPCレジスタに退避される内容

退避要因		DBPC レジスタに退避される内容
例外トラップ発生		例外トラップの発生要因となった命令の次の命令のアドレス
ディバグ・トラップ発生		ディバグ・トラップの発生要因となった命令の次の命令のアドレス
ディバグ・ブレーク発生	実行系トラップ	ブレークの発生要因となった命令のアドレス
	ミス・アライン・アクセス例外	
	アラインメント・エラー例外	
	アクセス系トラップ	ブレークの発生要因となった命令の次の命令のアドレス
シングルステップ動作実行		次に実行される命令（ディバグ・モニタ・ルーチンからの復帰時に実行される命令）のアドレス

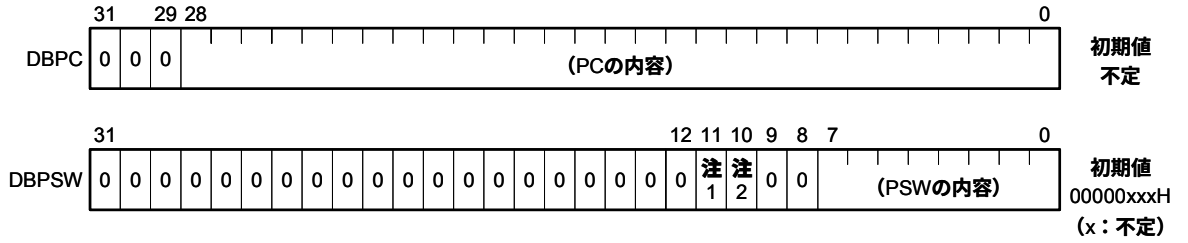
備考　退避要因の詳細については，第 9 章　ディバグ・モードへの移行を参照してください。

DBPSW には，現在の PSW の内容が退避されます。

なお，DBPC レジスタのビット 31-29 と DBPSW レジスタのビット 31-12, 9, 8 は，将来の機能拡張のために予約されています（0 に固定）。

備考　DBPC, DBPSW レジスタは，ユーザ・モードでのリード／ライトが可能です。

図2－8　例外／ディバグ・トラップ時状態退避レジスタ（DBPC, DBPSW）



注 1. PSW レジスタの SS フラグの内容

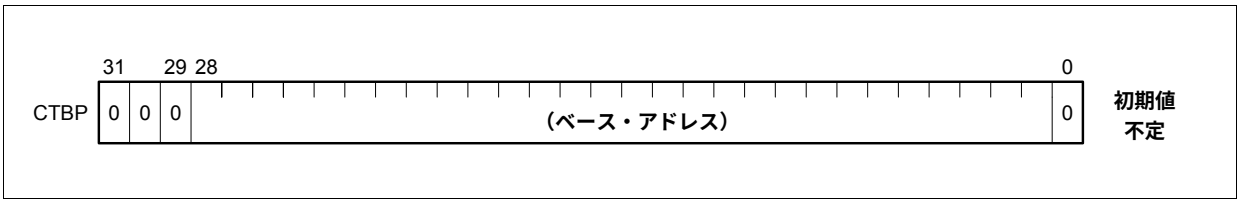
2. PSW レジスタの SB フラグの内容

2. 2. 7　CALLT ベース・ポインタ（CTBP）

CALLT ベース・ポインタ（CTBP）は，テーブル・アドレスの指定，ターゲット・アドレスの生成に使用されます（ビット 0 は 0 に固定）。

なお，ビット 31-29 は，将来の機能拡張のために予約されています（0 に固定）。

図2－9　CALLTベース・ポインタ（CTBP）



2. 2. 8　ディバグ・インタフェース・レジスタ（DIR）

ディバグ・インタフェース・レジスタ（DIR）は，ディバグ機能の制御や状態を示します。

LDSR 命令を使用してこのレジスタの各ビットの内容を変更した場合は，LDSR 命令実行終了直後から変更内容が有効となります。

ディバグ・モード時は，各ビットへのライトは常に可能です（書き込み禁止のビット（31, 27-23, 19-17, 15 ビット），リード・オンリーのビット（ビット 3, 0）を除く）。ユーザ・モード時は CSL ビットのみライト可能になります。

また，ディバグ・モード時，ユーザ・モード時ともリードは常に可能です（31, 27-23, 19-17, 15 ビットの読み出し値は不定）。ただし，ユーザ・モード時に，CSL ビット以外のビットを読み出した場合のリード値は不定となります（リード・オンリーのビット（ビット 3, 0）を除く）。

図2－10　ユーザ・モード時のDIRレジスタのライト可能ビット

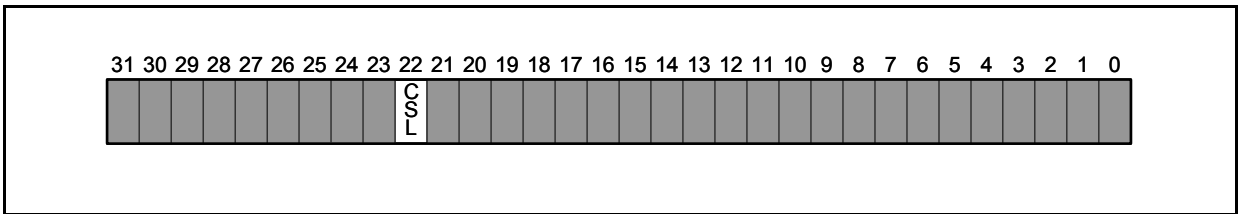


図2－10　ディバグ・インタフェース・レジスタ（DIR）（1/3）

注意 1. SQ1, RE1 **ビットは，常にどちらか一方をセット（1），または両ビットともクリア（0）してください。両ビットともセット（1）した場合の動作は保証しません。**

2. SQ0, RE0 **ビットは，常にどちらか一方をセット（1），または両ビットともクリア（0）してください。両ビットともセット（1）されているとブレークは発生しません。**

3. **ビット** 31, 27-23, 19-17, 15 **は** 0 **以外の書き込みは禁止です。読み出し値は不定です。**

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
DIR	0	SQ1	RE1	CS1	0	0	0	0	0	CSL	BT3	BT2	0	0	0	STT	0	SQ0	RE0	CS0	0	MAE	AEE	SSE	EXT	INI	BT1	BT0	0	MT	AT	DM	初期値 00000040H

ビット位置	ビット名	意　味
30	SQ1 [注1]	チャネル 2, 3 に対するシーケンシャル・ブレーク・モード（チャネル 2→3 の順にブレークが発生した場合にブレーク）を設定します。 0：通常ブレーク・モード（初期値） 1：シーケンシャル・ブレーク・モード
29	RE1 [注1]	チャネル 2, 3 に対するレインジ・ブレーク・モード（チャネル 2, 3 に同時にブレークが発生した場合だけブレーク）を設定します。 0：通常ブレーク・モード（初期値） 1：レインジ・ブレーク・モード
28	CS1 [注1]	チャネル 2, 3 の制御レジスタ（BPCn, BPAVn, BPAMn, BPDVn, BPDMn）への設定を許可します（ブレーク条件の許可／禁止ではありません）（n = 2, 3）。 0：チャネル 2 の制御レジスタ（BPC2, BPxx2）への設定を許可（初期値） 1：チャネル 3 の制御レジスタ（BPC3, BPxx3）への設定を許可
22	CSL	各チャネルの制御レジスタへの設定を許可します（表 2－6　CSL, CS1, CS0 ビットの設定とブレーク条件の設定が許可されるチャネル，レジスタの関係参照）。 CSL：制御レジスタへの設定が許可されるチャネル 0：チャネル 0, 1（初期値） 1：チャネル 2, 3
21	BT3 [注2]	チャネル 3 のブレークが発生するとセット（1）されます（ユーザ・プログラムによる任意のセット（1）はできません）（初期値：0）。
20	BT2 [注2]	チャネル 2 のブレークが発生するとセット（1）されます（ユーザ・プログラムによる任意のセット（1）はできません）（初期値：0）。

注 1. INI **ビットがセット（1）されていると，**SQ1, RE1, CS1 **ビットへのライトはできません。また，**INI **ビットをセット（1）すると，**SQ1, RE1, CS1 **ビットは自動的にクリア（0）されます。**

2. BT2, BT3 **ビットは，**INI **ビットがセット（1）されていると動作しません（ブレークが発生してもセット（1）されません）。また，セット（1）されると，**LDSR **命令によって** 0 **を設定するか，**INI **ビットをセット（1）するまでクリア（0）されません。**

図2－10　ディバグ・インタフェース・レジスタ（DIR）（2/3）

ビット位置	ビット名	意　味
16	STT	ディバグ・トラップ実行時にセット（1）されます（ユーザ・プログラムによる任意のセット（1）はできません）（初期値：0）。 このビットは，セット（1）されると自動的にクリア（0）されません（LDSR 命令によってのみクリア（0）できます）。
14	SQ0 注	チャネル 0, 1 に対するシーケンシャル・ブレーク・モード（チャネル 0→1 の順にブレークが発生した場合にブレーク）を設定します。 0：通常ブレーク・モード（初期値） 1：シーケンシャル・ブレーク・モード
13	RE0 注	チャネル 0, 1 に対するレインジ・ブレーク・モード（チャネル 0, 1 に同時にブレークが発生した場合だけブレーク）を設定します。 0：通常ブレーク・モード（初期値） 1：レインジ・ブレーク・モード
12	CS0 注	チャネル 0, 1 の制御レジスタ（BPCn, BPAVn, BPAMn, BPDVn, BPDMn）への設定を許可します（ブレーク条件の許可／禁止ではありません）（n = 0, 1）。 0：チャネル 0 の制御レジスタ（BPC0, BPxx0）への設定を許可（初期値） 1：チャネル 1 の制御レジスタ（BPC1, BPxx1）への設定を許可
10	MAE	ミス・アライン・アクセス例外の検出の許可／禁止を設定します。 0：ミス・アライン・アクセス例外の検出を禁止（初期値） 1：ミス・アライン・アクセス例外の検出を許可
9	AEE	アラインメント・エラー例外の検出の許可／禁止を設定します。 0：アラインメント・エラー例外の検出を禁止（初期値） 1：アラインメント・エラー例外の検出を許可
8	SSE	PSW レジスタの SS フラグへの書き込みの許可／禁止を設定します。 0：SS フラグへの書き込みを禁止（SS フラグは 0 に固定）（初期値） 1：SS フラグへの書き込みを許可
7	EXT	拡張ディバグ機能（このレジスタのビット 31-15 に割り付けられた機能）の有効／無効を設定します。 0：無効（V850E1 CPU 互換）（初期値） 1：有効

注　INI ビットがセット（1）されていると，SQ0, RE0, CS0 ビットへのライトはできません。また，INI ビットをセット（1）すると，SQ0, RE0, CS0 ビットは自動的にクリア（0）されます。

図2－10　ディバグ・インタフェース・レジスタ（DIR）（3/3）

ビット位置	ビット名	意　味
6	INI 注1	ディバグ機能のリセットにより，セット（1）されます（初期値：1）。 リセット後はセット（1）されるため，必ずクリア（0）してください（このビットがセット（1）されていると，SQn, REn, CSn ビットへのライトができません（n = 0, 1）。また，BT3-BT0 ビットが動作しません）。
5	BT1 注2	チャネル 1 のブレークが発生するとセット（1）されます（ユーザ・プログラムによる任意のセット（1）はできません）（初期値：0）。
4	BT0 注2	チャネル 0 のブレークが発生するとセット（1）されます（ユーザ・プログラムによる任意のセット（1）はできません）（初期値：0）。
2	MT 注1	ミス・アライン・アクセス例外が検出されるとセット（1）されます（ユーザ・プログラムによる任意のセット（1）はできません）（初期値：0）。
1	AT 注1	アラインメント・エラー例外が検出されるとセット（1）されます（ユーザ・プログラムによる任意のセット（1）はできません）（初期値：0）。
0	DM 注3	ディバグ・モードに移行するとセット（1）されます（初期値：0）。 このビットへのライトはできません。

注 1. INI, MT, AT ビットは，セット（1）されると自動的にクリア（0）されません（LDSR 命令によってのみクリア（0）できます）。

2. BT0, BT1 ビットは，INI ビットがセット（1）されていると，動作しません（ブレークが発生してもセット（1）されません）。また，セット（1）されると，LDSR 命令によって 0 を設定するか，INI ビットをセット（1）するまでクリア（0）されません。

3. DM ビットは，次のように変化します。

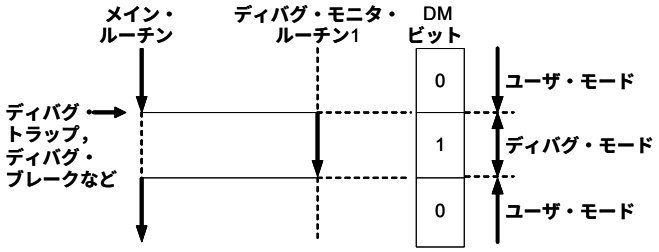


表2－6　CSL, CS1, CS0ビットの設定とブレーク条件の設定が許可されるチャネル，レジスタの関係

CSL	CS1	CS0	ブレーク条件の設定が許可されるチャネル，レジスタ	
			チャネル	レジスタ
0	任意	0	チャネル 0	BPC0, BPAV0, BPAM0, BPDV0, BPDM0
		1	チャネル 1	BPC1, BPAV1, BPAM1, BPDV1, BPDM1
1	0	任意	チャネル 2	BPC2, BPAV2, BPAM2, BPDV2, BPDM2
	1	任意	チャネル 3	BPC3, BPAV3, BPAM3, BPDV3, BPDM3

2. 2. 9　ブレークポイント制御レジスタ 0-3（BPC0-BPC3）

ブレークポイント制御レジスタ 0-3（BPC0-BPC3）は，それぞれ，チャネル 0-3 のディバグ機能の制御や状態を示します。

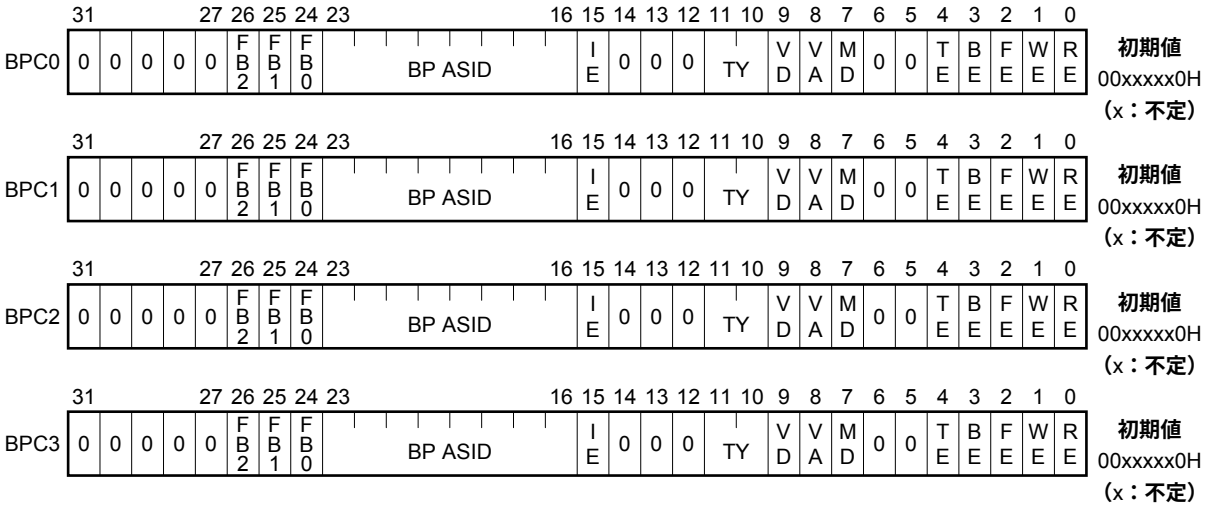
DIR レジスタの CSL, CS1, CS0 ビットの設定で選択されるチャネルに応じて有効となるレジスタが選択されます（表 2－6　CSL, CS1, CS0 ビットの設定とブレーク条件の設定が許可されるチャネル，レジスタの関係参照）。

LDSR 命令を使用してこのレジスタの各ビットの内容を変更した場合は，LDSR 命令実行終了直後から変更内容が有効となります（FE ビットをセット（1）した場合は，有効となるタイミングが遅れますが，DBRET 命令を実行したあとには，必ず設定が反映されます）。

注意 1. BPCn レジスタのビット 31-27, 14-12, 6, 5 は，必ずクリア（0）してください（n = 0-3）。いずれかのビットをセット（1）した場合の動作は保証しません。

2. BPCn レジスタの FB2-FB0 ビットへのライトは，クリア（0）のみ可能です（n = 0-3）。
これらのビットの値を更新する場合は，すべてのビットをクリア（0）してください。いずれかのビットをセット（1）した場合の動作は保証しません。

図2－11　ブレークポイント制御レジスタ0-3（BPC0-BPC3）（1/2）



ビット位置	ビット名	意　味
26-24	FB2-FB0	命令フェッチ・イベントによって発生したブレークの種類を示します。 FB2 / FB1 / FB0 / ブレークの種類 0 / 0 / 0 / ブレーク対象命令の実行中断によるブレーク（初期値） 0 / 1 / 0 / ブレーク対象命令とその1つ前の命令の実行中断によるブレーク 1 / 0 / 0 / ブレーク対象命令とその1つ前と2つ前の命令の実行中断によるブレーク 0 / 0 / 1 / ブレーク対象命令の実行完了によるブレーク （上記以外） / （将来の機能拡張のための予約）
23-16	BP ASID	ブレークを発生させるプログラムIDを設定します（初期値：不定）。 なお，設定値は，IEビットがセット（1）されている場合だけ有効となります。
15	IE	BP ASIDビットとASIDレジスタに設定されているプログラムIDの比較を設定します（初期値：不定）。 0：比較しない 1：比較する
11, 10	TY	ブレークを検出するアクセスの種類を設定します（初期値：不定）。 ビット11 / ビット10 / ブレークを検出するアクセスの種類 0 / 0 / すべてのデータ・タイプによるアクセス 0 / 1 / バイト・アクセス（ビット操作を含む） 1 / 0 / ハーフワード・アクセス 1 / 1 / ワード・アクセス なお，実行系トラップの場合は，このレジスタの設定内容は無視されます。

図2－11　ブレークポイント制御レジスタ0-3（BPC0-BPC3）（2/2）

ビット位置	ビット名	意　味
9	VD	データ・コンパレータの一致条件を設定します（初期値：不定）。 0：一致でブレーク 1：不一致でブレーク
8	VA	アドレス・コンパレータの一致条件を設定します（初期値：不定）。 0：一致でブレーク 1：不一致でブレーク
7	MD	データ・コンパレータの動作を設定します（初期値：不定）。 0：データが条件に一致したときブレーク 1：VD ビットや BPDVx, BPDMx レジスタの設定に関係なくデータの一致判定（データ・コンパレータ）は無視
4	TE	チャネル 0-3 のイベント発生時のトリガ出力の許可／禁止を設定します。 0：トリガ出力禁止（初期値） 1：トリガ出力許可（対応トリガを出力）
3	BE	チャネル 0-3 のイベント発生時にブレークを CPU へ通知する／しないを設定します。 0：通知しない（初期値） 1：通知する（ブレークする）
2	FE	命令フェッチ時のイベント動作を設定します。 0：イベント・マスク（初期値） 1：イベント発生[注1]
1	WE	データ・ライト時のイベント動作を設定します。 0：イベント・マスク（初期値） 1：イベント発生[注2]
0	RE	データ・リード時のイベント動作を設定します。 0：イベント・マスク（初期値） 1：イベント発生[注2]

注 1. FE ビットをセット（1）した場合は，必ず，WE, RE ビットをクリア（0）してください。

2. WE ビット，または RE ビットをセット（1）した場合は，必ず，FE ビットをクリア（0）してください。

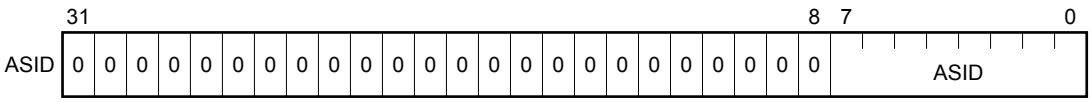
2. 2. 10　プログラム ID レジスタ（ASID）

現在実行中のプログラム ID を設定します。

プログラム ID は，同じアドレス領域の RAM に異なるプログラムをダウンロードして実行する際に，特定のプログラムの実行時のみディバグ・モードへの移行を行う必要のある場合などに使用します。BPCn レジスタの IE ビットをセット（1）している場合は，BP ASID ビットと ASID レジスタに設定したプログラム ID が一致しないと，ブレーク条件が一致してもディバグ・モードには移行しません（n = 0-3）。

なお，ビット 31-8 は，将来の機能拡張のために予約されています（0 に固定）。

図2－12　プログラムIDレジスタ（ASID）



初期値
000000xxH
（x：不定）

ビット位置	フラグ名	意　味
7-0	ASID	現在実行中のプログラム ID

2. 2. 11　ブレークポイント・アドレス設定レジスタ 0-3（BPAV0-BPAV3）

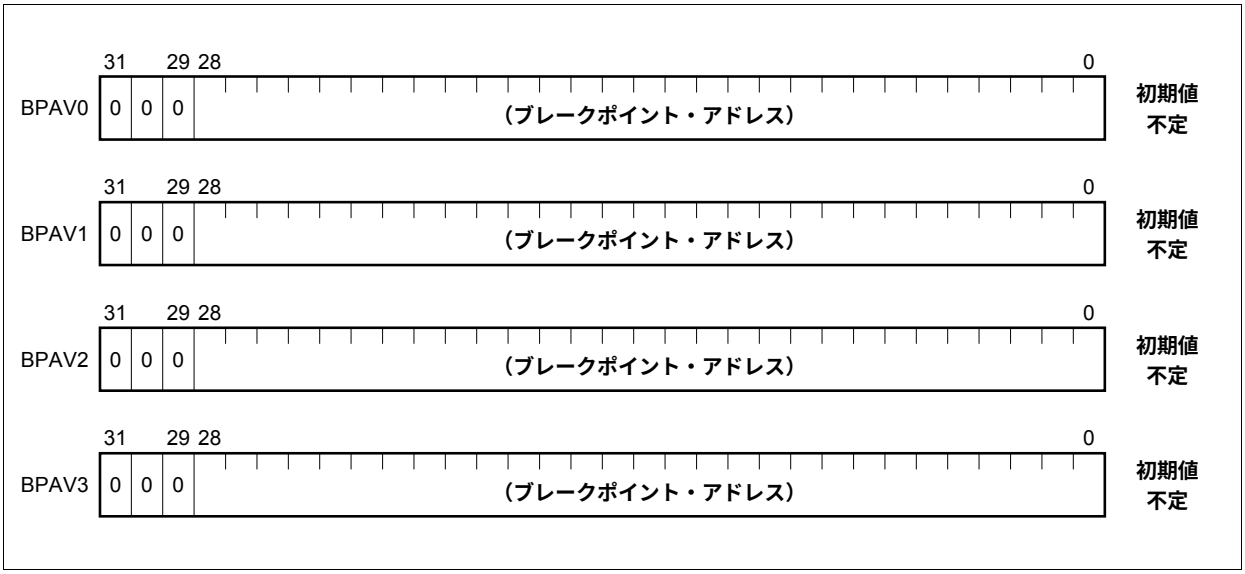
アドレス・コンパレータで使用するブレークポイント・アドレスを設定します。

DIR レジスタの CSL, CS1, CS0 ビットの設定により有効となるレジスタが選択されます（表 2－6　CSL, CS1, CS0 ビットの設定とブレーク条件の設定が許可されるチャネル，レジスタの関係参照）。

なお，使用しない場合は，必ず各ビットをセット（1）してください。

また，ビット 31-29 は，将来の機能拡張のために予約されています（0 に固定）。

図2－13　ブレークポイント・アドレス設定レジスタ0-3（BPAV0-BPAV3）



2. 2. 12　ブレークポイント・アドレス・マスク・レジスタ 0-3（BPAM0-BPAM3）

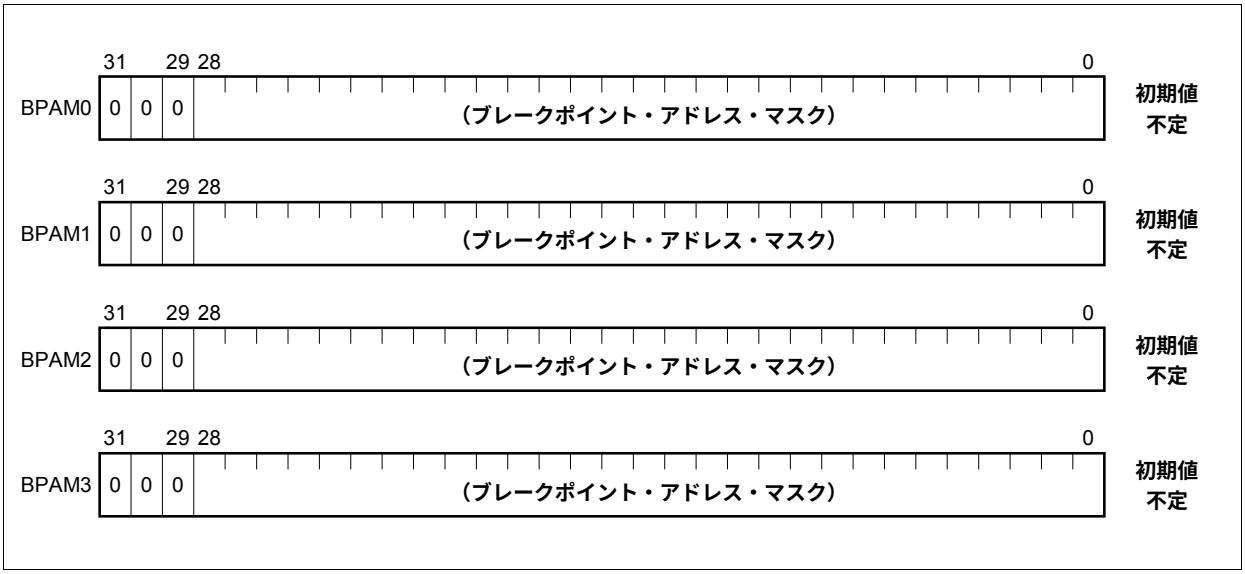
アドレス比較のビット・マスクを設定します（1 でマスク）。

DIR レジスタの CSL, CS1, CS0 ビットの設定により有効となるレジスタが選択されます（表 2－6　CSL, CS1, CS0 ビットの設定とブレーク条件の設定が許可されるチャネル，レジスタの関係参照）。

なお，使用しない場合は，必ず各ビットをセット（1）してください。

また，ビット 31-29 は，将来の機能拡張のために予約されています（0 に固定）。

図 2－14　ブレークポイント・アドレス・マスク・レジスタ 0-3（BPAM0-BPAM3）



2. 2. 13　ブレークポイント・データ設定レジスタ 0-3（BPDV0-BPDV3）

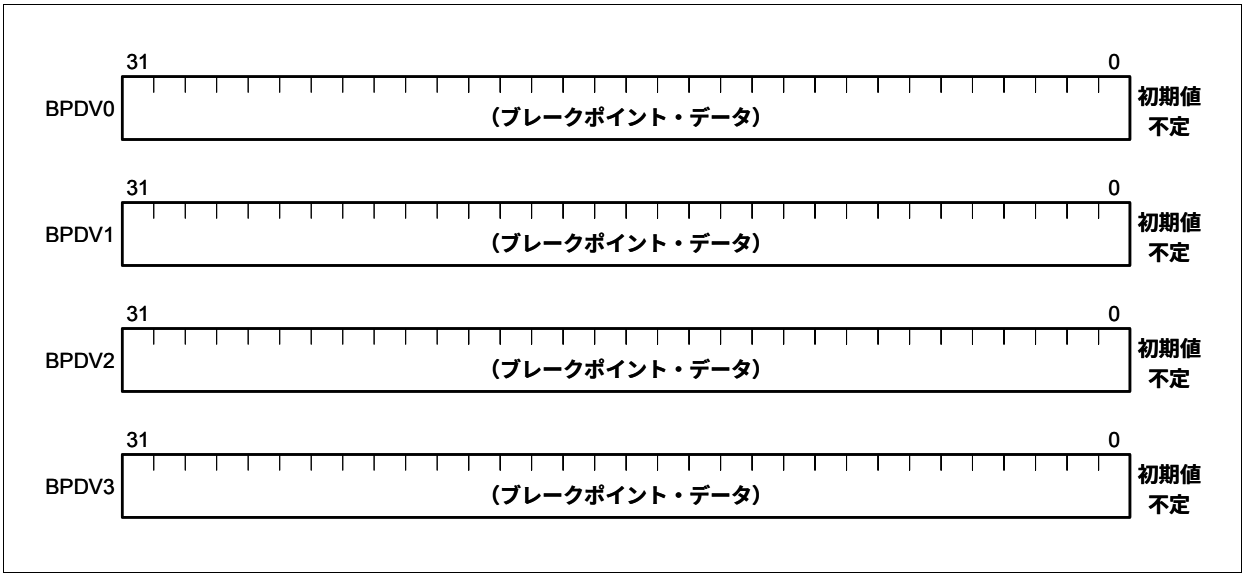
データ・コンパレータで使用するブレークポイント・データを設定します。

DIR レジスタの CSL, CS1, CS0 ビットの設定により有効となるレジスタが選択されます（表 2－6　CSL, CS1, CS0 ビットの設定とブレーク条件の設定が許可されるチャネル，レジスタの関係参照）。

なお，使用しない場合は，必ず各ビットをセット（1）してください。

備考　16 ビット命令の命令コードを設定する場合は，LSB 詰めで設定してください。
32 ビット命令の命令コードを設定する場合は，リトル・エンディアン形式で設定してください。

図2－15　ブレークポイント・データ設定レジスタ0-5（BPDV0-BPDV5）



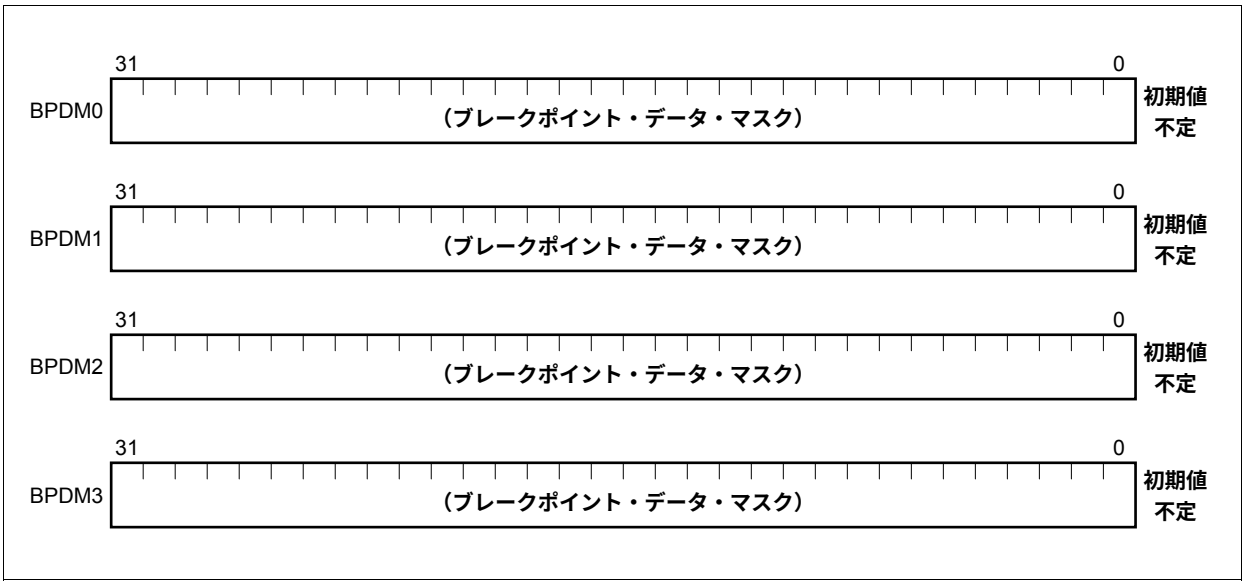
2. 2. 14　ブレークポイント・データ・マスク・レジスタ 0-3（BPDM0-BPDM3）

データ比較のビット・マスクを設定します（1 でマスク）。

DIR レジスタの CSL, CS1, CS0 ビットの設定により有効となるレジスタが選択されます（表 2－6　CSL, CS1, CS0 ビットの設定とブレーク条件の設定が許可されるチャネル，レジスタの関係参照）。

なお，使用しない場合は，必ず各ビットをセット（1）してください。

図2－16　ブレークポイント・データ・マスク・レジスタ0-3（BPDM0-BPDM3）



第3章　データ・タイプ

3.1　データ形式

サポートされているデータ・タイプは次のとおりです（3.2　**データ表現**参照）。

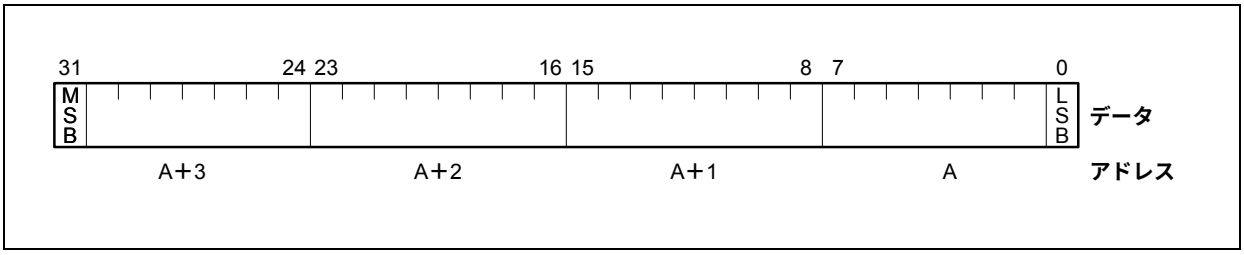
- **整数**（32, 16, 8 **ビット**）
- **符号なし整数**（32, 16, 8 **ビット**）
- **ビット**

また，データ長として，ワード（32 ビット），ハーフワード（16 ビット），バイト（8 ビット）があります。これらのデータは，バイト 0 が常に最下位（最右端）バイトである構成になっています（リトル・エンディアン形式）。

固定長のデータがメモリにある場合のデータ形式を次に示します。

（1）ワード

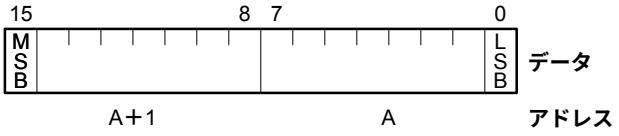
ワードは任意のワード境界[注]から始まる連続した 4 バイト（32 ビット）のデータです。各ビットには 0 から 31 までの番号が付けられており，LSB（Least significant bit）はビット 0，MSB（Most significant bit）はビット 31 に対応します。ワードはそのアドレス「A」（ミス・アライン・アクセス禁止の状態では下位 2 ビットは 0 [注]）で指定され，4 つのバイト「A」，「A＋1」，「A＋2」，「A＋3」を占めます。



注　ミス・アライン・アクセス許可の状態では，ハーフワード・アクセス，ワード・アクセスにかかわらず，すべてのバイト境界にアクセスできます。3.3　**データ・アラインメント**を参照してください。

(2) ハーフワード

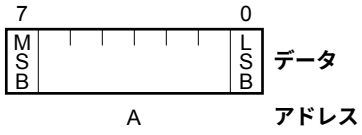
ハーフワードは任意のハーフワード境界[注]から始まる連続した2バイト（16ビット）のデータです。各ビットには，0から15までの番号が付けられており，LSBはビット0，MSBはビット15に対応します。ハーフワードはそのアドレス「A」（下位1ビットは0[注]）で指定され，2つのバイト「A」，「A＋1」を占めます。



注　ミス・アライン・アクセス許可の状態では，ハーフワード・アクセス，ワード・アクセスにかかわらず，すべてのバイト境界にアクセスできます。3.3　**データ・アラインメント**を参照してください。

(3) バイト

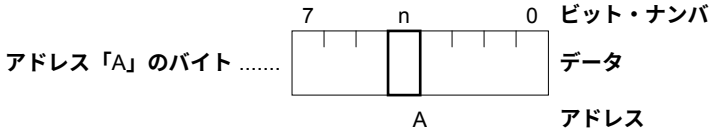
バイトは，任意のバイト境界[注]から始まる連続した8ビットのデータです。各ビットには0から7までの番号が付けられており，LSBはビット0，MSBはビット7に対応します。バイトは，そのアドレス「A」で指定されます。



注　ミス・アライン・アクセス許可の状態では，ハーフワード・アクセス，ワード・アクセスにかかわらず，すべてのバイト境界にアクセスできます。3.3　**データ・アラインメント**を参照してください。

(4) ビット

ビットは，任意のバイト境界[注]から始まる8ビット・データのnビット目の1ビット・データです。ビットはそのバイトのアドレス「A」と，ビット・ナンバ「n」で指定されます。



注　ミス・アライン・アクセス許可の状態では，ハーフワード・アクセス，ワード・アクセスにかかわらず，すべてのバイト境界にアクセスできます。3.3　**データ・アラインメント**を参照してください。

3. 2　データ表現

3. 2. 1　整　　数

整数は 2 の補数による 2 進数表現で表し，32 ビット，16 ビット，8 ビットの 3 通りの長さを持っています。整数の位取りはその長さにかかわらず，ビット 0 を最下位ビットとし，ビット番号が増えるにしたがって位取りを高くします。2 の補数表現であるため，最上位ビットを符号ビットとして使用します。

各データ長の整数の範囲は次のとおりです。

- ワード（32 ビット）　　：－2147483648～＋2147483647
- ハーフワード（16 ビット）：－32768～＋32767
- バイト（8 ビット）　　：－128～＋127

3. 2. 2　符号なし整数

「整数」が，正負両方の値を取るデータであるのに対して，「符号なし整数」は，負でない整数を意味します。整数と同様に，符号なし整数も 2 進数表現で表し，32 ビット，16 ビット，8 ビットの 3 通りの長さを持っています。符号なし整数の位取りは，整数と同様に，その長さにかかわらずビット 0 を最下位ビットとし，ビット番号が増えるに従って位取りを高くします。ただし符号ビットは存在しません。

各データ長の符号なし整数の範囲は次のとおりです。

- ワード（32 ビット）　　：0～4294967295
- ハーフワード（16 ビット）：0～65535
- バイト（8 ビット）　　：0～255

3. 2. 3　ビット

ビット・データとして，クリア（0）またはセット（1）の 2 つの値をとる 1 ビットのデータを扱うことができます。ビットに関する操作は，メモリ空間の 1 バイト・データだけを対象とし，次の 4 種類の操作ができます。

- SET1
- CLR1
- NOT1
- TST1

3. 3　データ・アラインメント

ミス・アライン・アクセスの許可／禁止の設定に応じて，データのアライン（境界整列）を行う必要があります。

ミス・アライン・アクセスとは，処理対象のデータがハーフワード形式の場合はハーフワード境界（アドレスの最下位ビットが 0）以外のアドレスへのアクセスを，処理対象のデータがワード形式の場合はワード境界（アドレスの下位 2 ビットが 0）以外のアドレスへのアクセスを示します。

備考　V850E2 コアでは，IFIMAEN 端子への入力レベルにより，ミス・アライン・アクセスの許可／禁止の設定を行います。

(1) ミス・アライン・アクセス許可の設定がされている場合

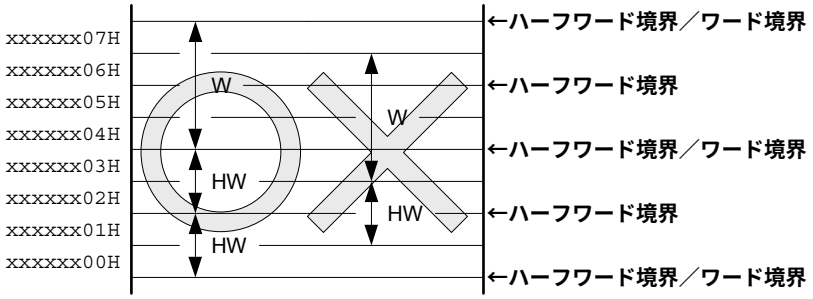
データ形式（バイト／ハーフワード／ワード）にかかわらず，すべてのアドレスにデータの配置が可能です。

ただし，ハーフワード・データ，ワード・データの場合，データがアラインされていないと，バス・サイクルが最低 1 回発生し，バス効率が低下します。

(2) ミス・アライン・アクセス禁止の設定がされている場合

アドレスの下位ビット（ハーフワード・データの場合は，最下位ビット，ワード・データの場合は，下位 2 ビット）が 0 にマスクしてアクセスされるため，正しくアラインされていないとデータの消失や切り捨てが発生します。したがって，処理を行うデータがハーフワード形式の場合はハーフワード境界から，ワード形式の場合はワード境界から配置してください。

図3－1　ミス・アライン・アクセス禁止時のデータ配置例



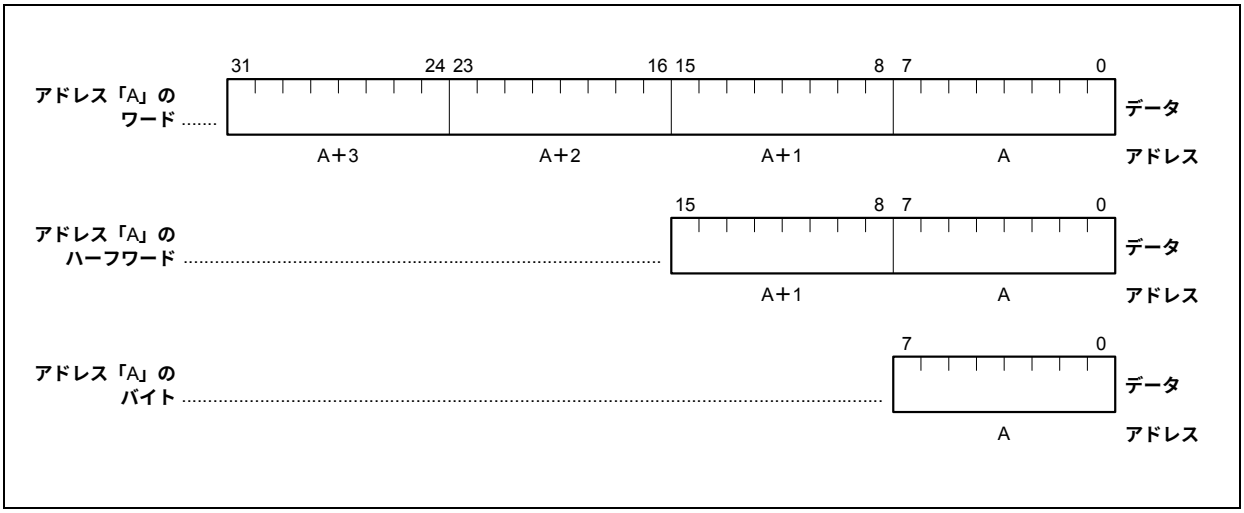
備考　W：ワード・データ
HW：ハーフワード・データ

第4章　アドレス空間

V850E2 CPU は，4G バイトのリニアなアドレス空間をサポートしています。このアドレス空間にはメモリと I/O の両方をマッピングします（メモリ・マップト I/O 方式）。CPU からメモリ，I/O に対して 32 ビットのアドレスが出力され，アドレス番地は最大「$2^{32}-1$」となります。

各アドレスに配置されるバイト・データは，ビット 0 を LSB，ビット 7 を MSB と定義されています。また，複数バイト構成のデータでは特に注意しないかぎり，下位側アドレスのバイト・データが LSB，上位側アドレスのバイト・データが MSB を持つように定義されています（リトル・エンディアン形式）。

このユーザーズ・マニュアルでは，複数バイト構成のデータを表現する場合，次のように右側を下位側アドレス，左側を上位側アドレスとして表現します。



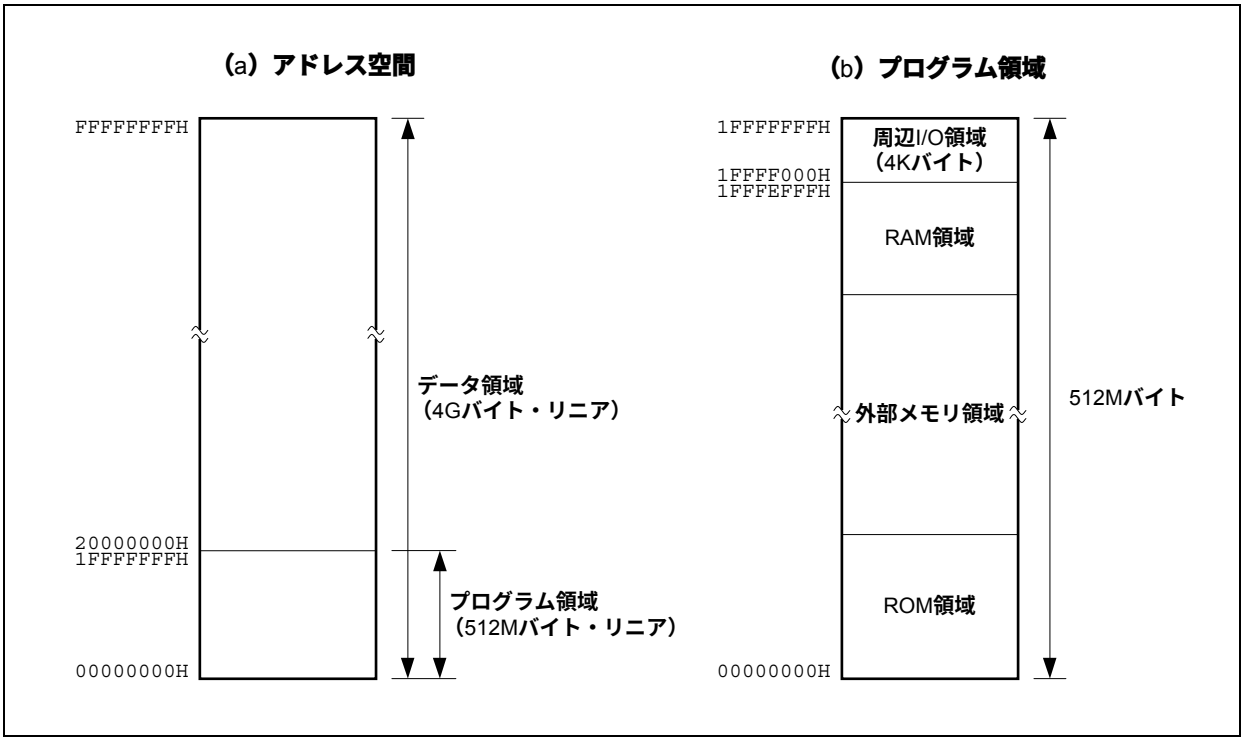
4.1　メモリ・マップ

V850E2 CPU は，32 ビット・アーキテクチャであり，オペランド・アドレシング（データ・アクセス）では，最大 4G バイトのリニア・アドレス空間（データ領域）をサポートします。

一方，命令アドレスのアドレシングにおいては，最大 512M バイトのリニア・アドレス空間（プログラム領域）をサポートします。

メモリ・マップを図 4－1 に示します。

図4－1　メモリ・マップ



4. 2　アドレシング・モード

アドレス生成には，分岐を伴う命令が使用する命令アドレス，データをアクセスする命令が使用するオペランド・アドレスの 2 種類があります。

4. 2. 1　命令アドレス

命令アドレスは，プログラム・カウンタ（PC）の内容によって決定され，実行した命令のバイト数に応じて自動的にインクリメントされます。また，分岐命令を実行する際には，次に示すアドレシングにより，分岐先アドレスを PC レジスタにセットします。

（1）レラティブ・アドレシング（PC 相対）

プログラム・カウンタ（PC）に，命令コードの符号付き 9/22/32 ビット・データ（ディスプレースメント：disp ×）を加算します。このとき，ディスプレースメントは，2 の補数データとして扱われ，それぞれビット 8/21/31 が符号ビット（S）となります。

JARL disp22, reg2 命令，JR disp22 命令，JARL disp32, reg1 命令，JR disp32 命令，Bcond disp9 命令が，このアドレシングの対象となります。

図4－2　レラティブ・アドレシング（1/2）

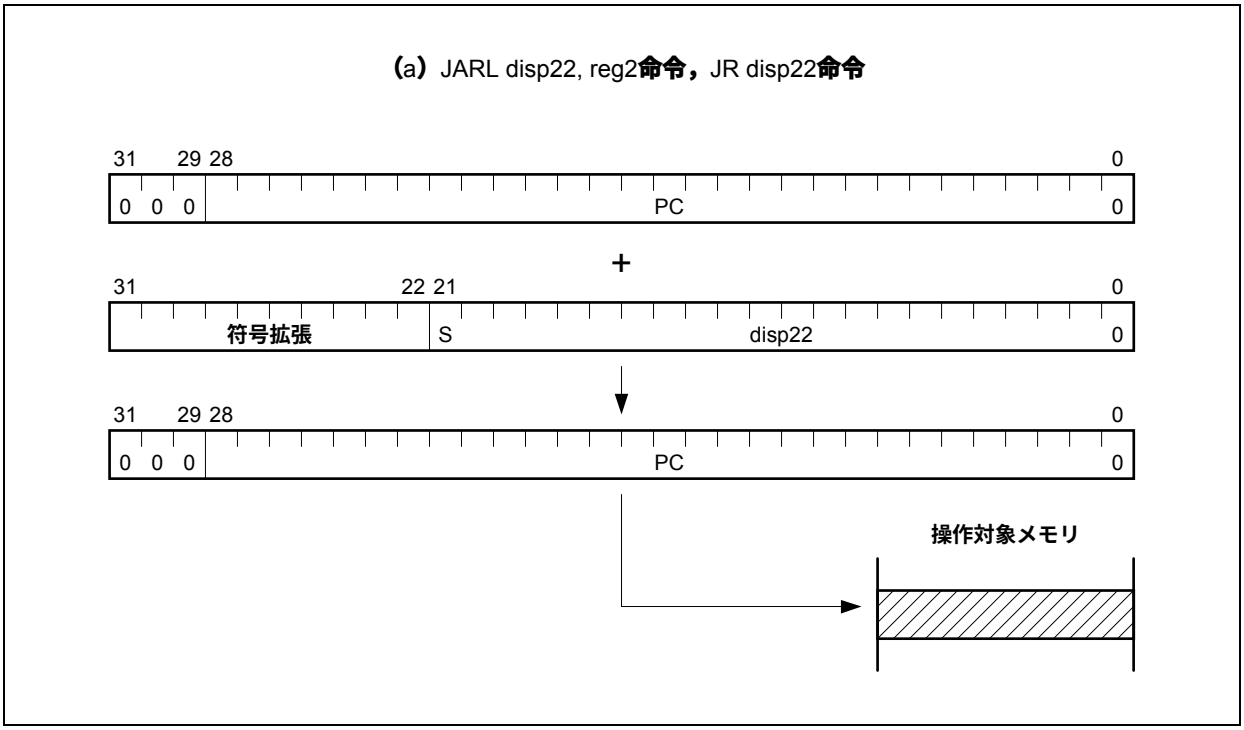
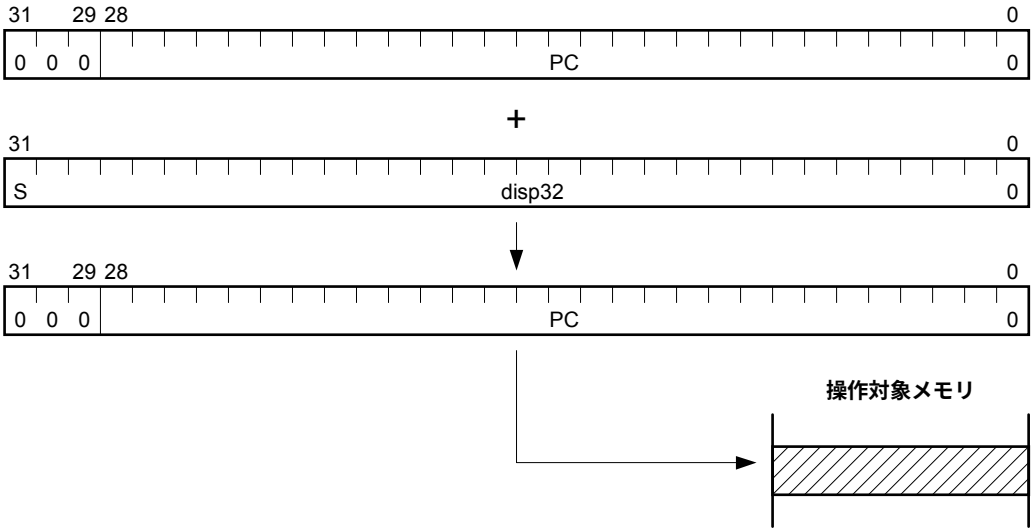
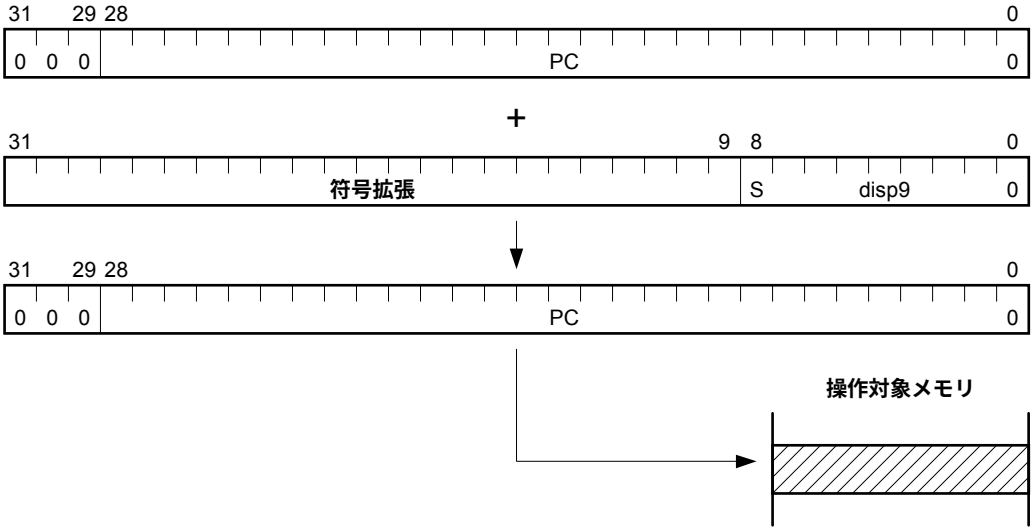


図4－2　レラティブ・アドレシング（2/2）

（b）JARL disp32, reg1命令，JR disp32命令



（c）Bcond disp9命令

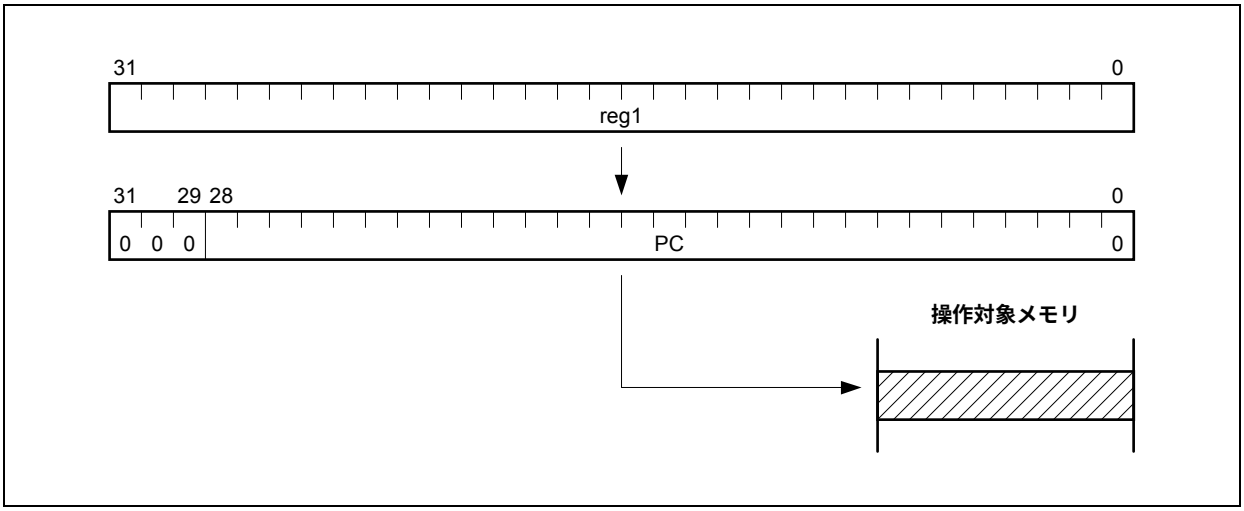


(2) レジスタ・アドレシング（レジスタ間接）

命令によって指定される汎用レジスタ（reg1）の内容をプログラム・カウンタ（PC）に転送します。

JMP [reg1] **命令が，このアドレシングの対象となります。**

図4－3　レジスタ・アドレシング（JMP [reg1] **命令）**

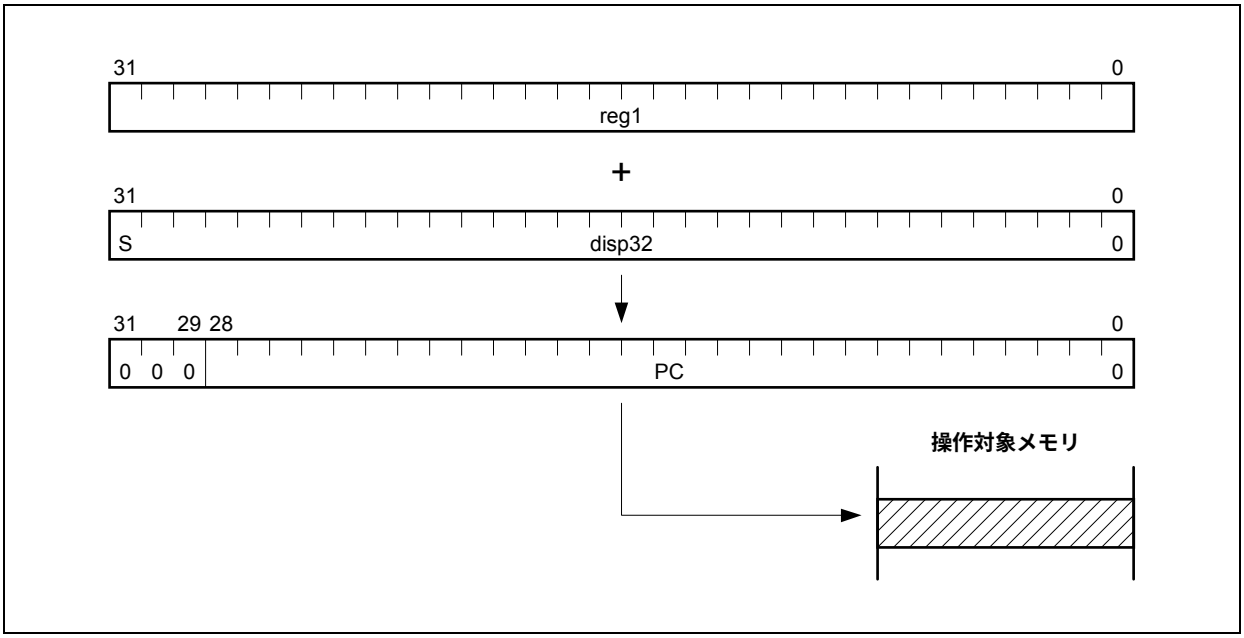


(3) ベースト・アドレシング

命令によって指定される汎用レジスタ（reg1）に，32 ビット・ディスプレースメント（disp32）を加算した内容をプログラム・カウンタ（PC）に転送します。

JMP disp32 [reg1] **命令が，このアドレシングの対象となります。**

図4－4　ベースト・アドレシング（JMP disp32 [reg1] **命令）**



4.2.2　オペランド・アドレス

命令を実行する際に対象となるレジスタやメモリなどをアクセスするために，次に示す方法があります。

(1) レジスタ・アドレシング

汎用レジスタ指定フィールドにより指定される汎用レジスタ，またはシステム・レジスタをオペランドとしてアクセスするアドレシングです。

オペランドに，reg1，reg2，reg3 または regID を含む命令が，このアドレシングの対象となります。

(2) イミーディエト・アドレシング

命令コード中に，操作対象となる 5 ビット・データ，16 ビット・データを持つアドレシングです。

オペランドに，imm5，imm16，vector，または cccc を含む命令が，このアドレシングの対象となります。

備考　vector：トラップ・ベクタ（00H-1FH）を指定する 5 ビット・イミーディエトであり，TRAP 命令で使用されるオペランドです。

cccc：　条件コード指定用の 4 ビット・データであり，CMOV 命令，SASF 命令，SETF 命令で使用されるオペランドです。0 の 1 ビットを上位に付加し，5 ビット・イミーディエト・データとしてオペコード中に割り当てられます。

(3) ベースト・アドレシング

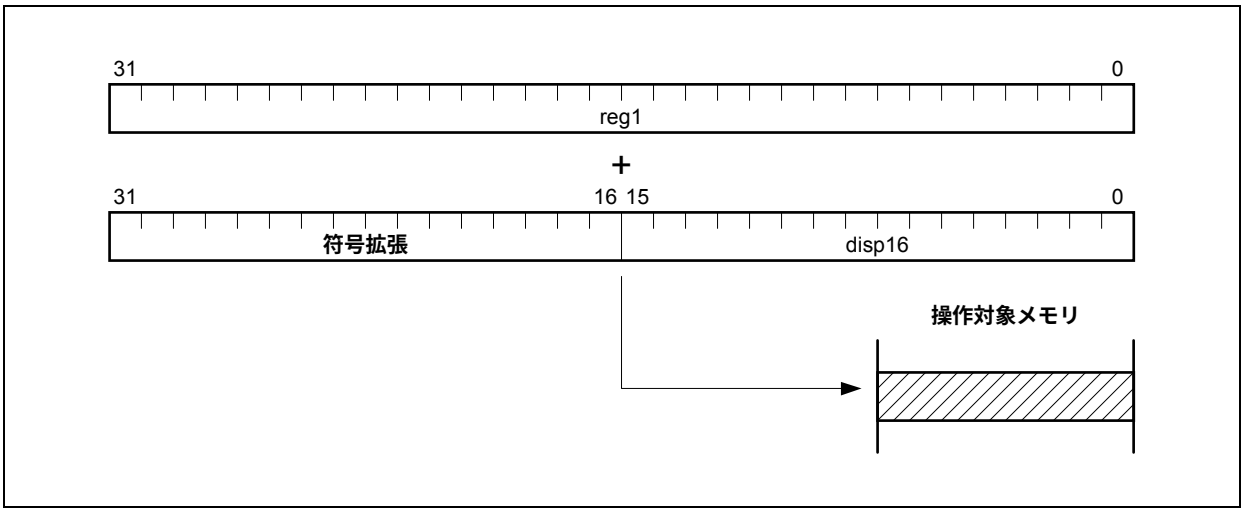
ベースト・アドレシングには，次に示す 2 種類があります。

(a) タイプ 1

命令コード中のアドレシング指定フィールドで指定される汎用レジスタ（reg1）の内容と 16 ビット・ディスプレースメント（disp16）の和がオペランド・アドレスとなって，操作対象となるメモリへのアクセスを行うアドレシングです。

オペランドに，disp16 [reg1] を含む命令が，このアドレシングの対象となります。

図4－5　ベースト・アドレシング（タイプ1）

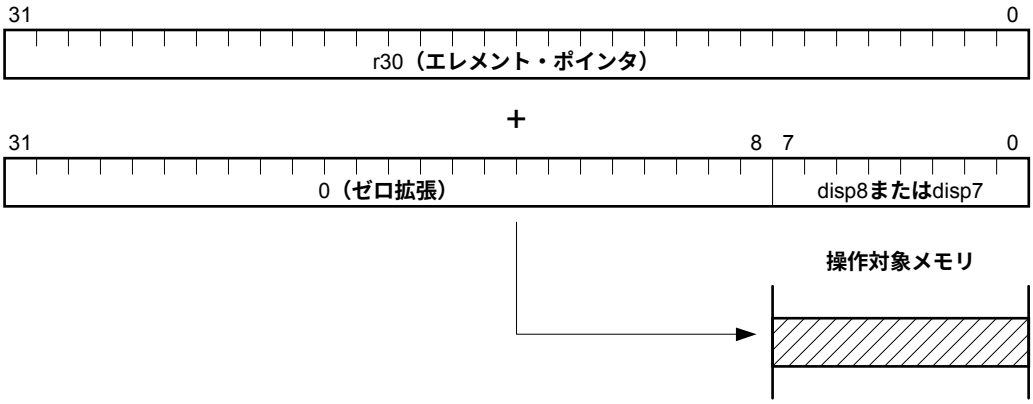


(b) タイプ2

エレメント・ポインタ（r30）の内容と，7 または 8 ビット・ディスプレースメント・データ（disp7, disp8）の和を，オペランド・アドレスとして操作対象となるメモリへのアクセスを行うアドレシングです。

SLD 命令と SST 命令が，このアドレシングの対象となります。

図4－6　ベースト・アドレシング（タイプ2）



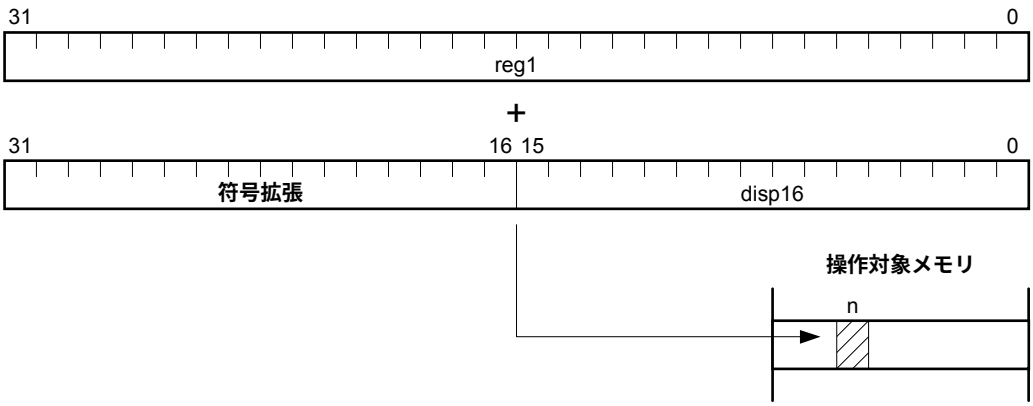
備考　バイト・アクセス：disp7
ハーフワード・アクセス，ワード・アクセス：disp8

(4) ビット・アドレシング

汎用レジスタ（reg1）の内容とワード長まで符号拡張した 16 ビット・ディスプレースメント（disp16）の和をオペランド・アドレスとして，操作対象となるメモリ空間の 1 バイト中の 1 ビット（3 ビット・データ「bit #3」で指定）をアクセスするアドレシングです。

ビット操作命令が，このアドレシングの対象となります。

図4－7　ビット・アドレシング



備考　n：3 ビット・データ（bit #3）で指定されるビット位置（n = 0-7）

第5章　命　　令

5.1　命令フォーマット

命令には，16 ビット・フォーマット，32 ビット・フォーマットの 2 種類があります。16 ビット・フォーマット命令には 2 項演算，制御，条件分岐などがあり，32 ビット・フォーマット命令にはロード，ストア，16 ビット・イミーディエトを扱う命令，ジャンプなどがあります。

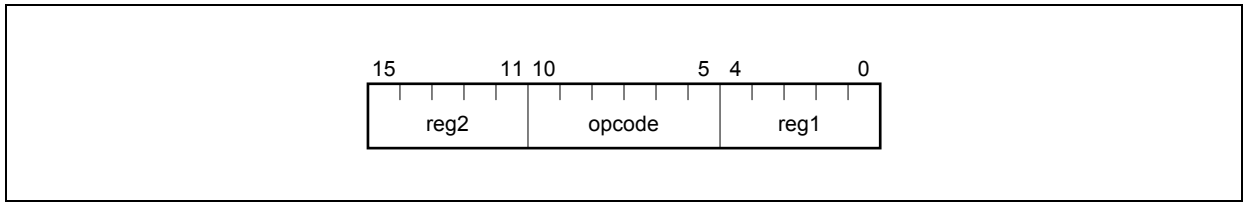
実際に命令がメモリに格納されるときは次のように配置されます。

- 各命令形式の下位部分（ビット 0 を含む）　→下位アドレス側
- 各命令形式の上位部分（ビット 15 またはビット 31 を含む）　→上位アドレス側

注意　一部の命令で未使用フィールド（RFU）がありますが，それらは将来の拡張用なので，0 に固定してください。

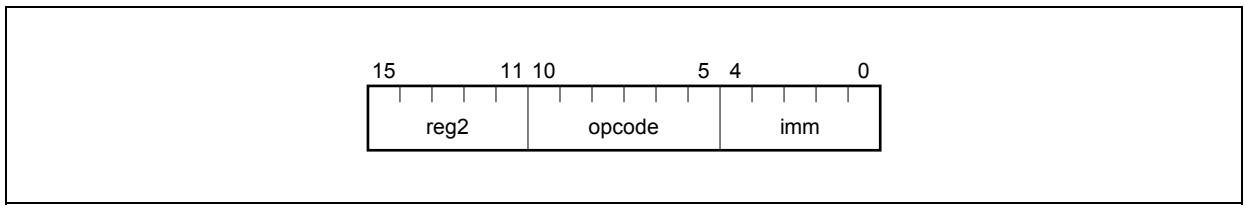
（1）reg-reg 命令形式（Format I）

6 ビットのオペコード・フィールド，2 つの汎用レジスタ指定フィールドを持つ 16 ビット長命令形式です。



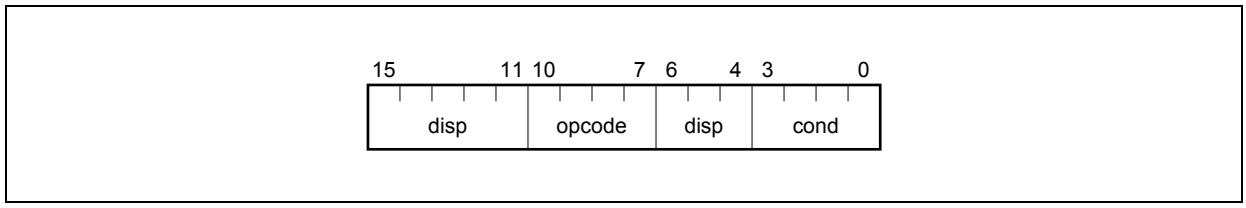
（2）imm-reg 命令形式（Format II）

6 ビットのオペコード・フィールド，5 ビットのイミーディエト・フィールド，1 つの汎用レジスタ・フィールドを持つ 16 ビット長命令形式です。



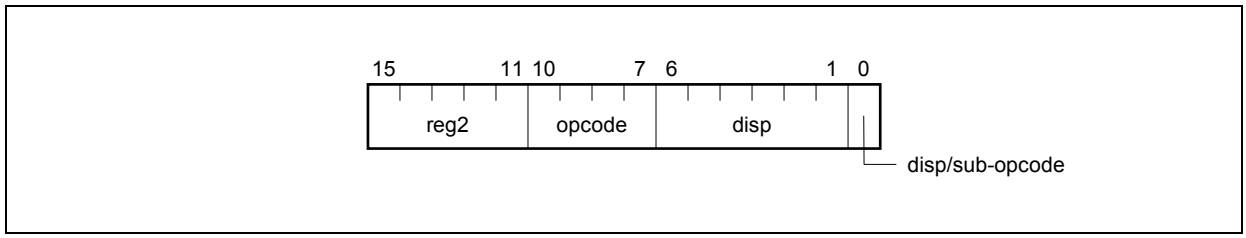
(3) 条件分岐命令形式 (Format III**)**

4 **ビットのオペコード・フィールド，**4 **ビットの条件コード・フィールド，**8 **ビットのディスプレースメント・フィールドを持つ** 16 **ビット長命令形式です。**

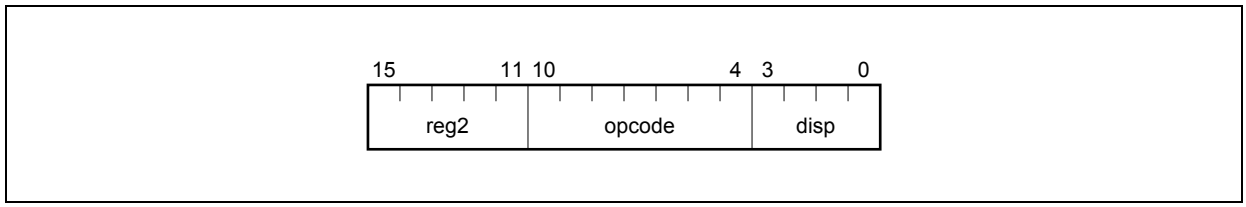


(4) ロード／ストア命令 16 **ビット形式 (**Format IV**)**

4 **ビットのオペコード・フィールド，**1 **つの汎用レジスタ指定フィールド，**7 **ビットのディスプレースメント・フィールド（または** 6 **ビット・ディスプレースメント・フィールドと** 1 **ビット・サブオペコード・フィールド）を持つ** 16 **ビット長命令形式です。**

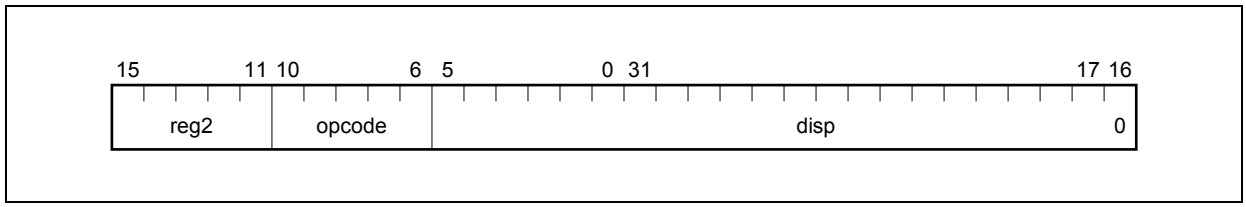


または，7 **ビットのオペコード・フィールドと** 1 **つの汎用レジスタ指定フィールド，**4 **ビットのディスプレースメント・フィールドを持つ** 16 **ビット長命令形式です。**



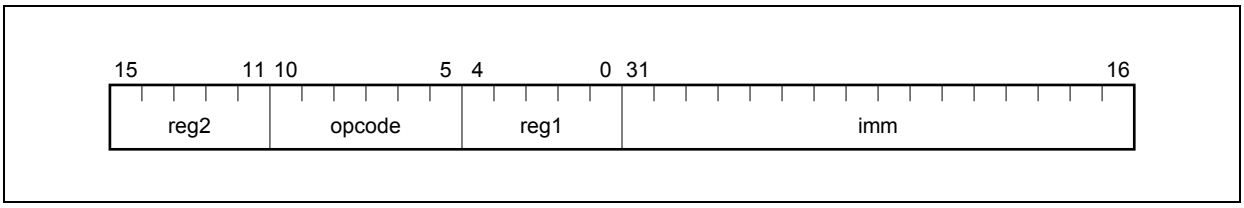
(5) ジャンプ命令形式 (Format V**)**

5 **ビットのオペコード・フィールド，**1 **つの汎用レジスタ指定フィールド，**22 **ビットのディスプレースメント・フィールドを持つ** 32 **ビット長命令形式です。**



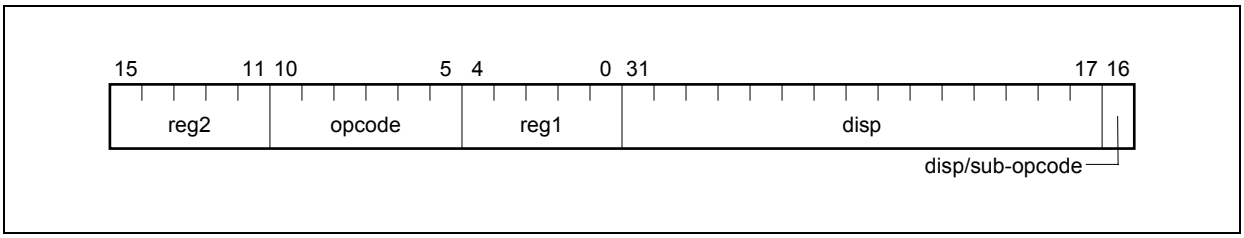
(6) 3 オペランド命令形式 (Format VI)

6 ビットのオペコード・フィールド，2 つの汎用レジスタ指定フィールド，16 ビットのイミーディエト・フィールドを持つ 32 ビット長命令形式です。



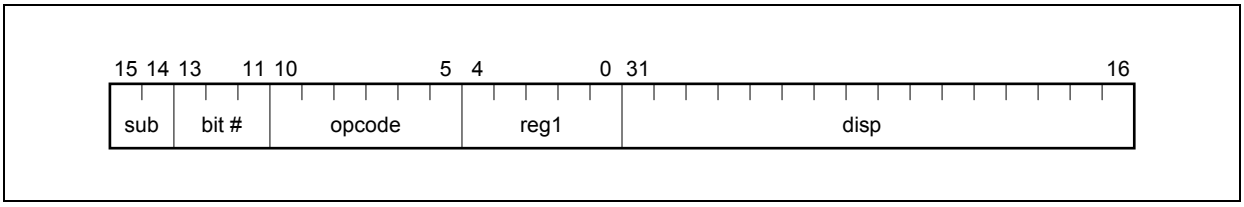
(7) ロード／ストア命令 32 ビット形式 (Format VII)

6 ビットのオペコード・フィールド，2 つの汎用レジスタ指定フィールド，16 ビットのディスプレースメント・フィールド（または 15 ビットのディスプレースメント・フィールドと 1 ビット・サブオペコード・フィールド）を持つ 32 ビット長命令形式です。



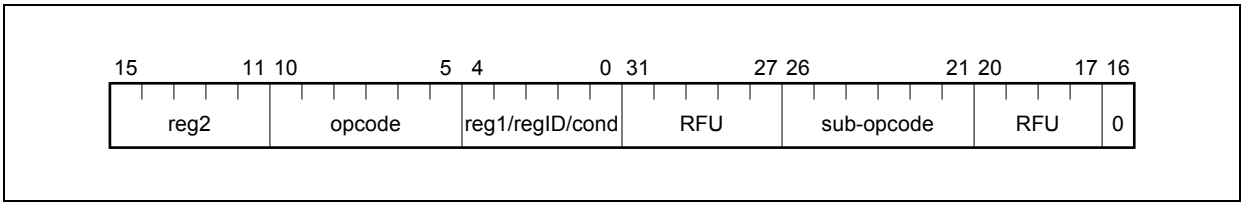
(8) ビット操作命令形式 (Format VIII)

6 ビットのオペコード・フィールドと 2 ビットのサブオペコード・フィールド，3 ビットのビット指定フィールド，1 つの汎用レジスタ指定フィールド，16 ビットのディスプレースメント・フィールドを持つ 32 ビット長命令形式です。



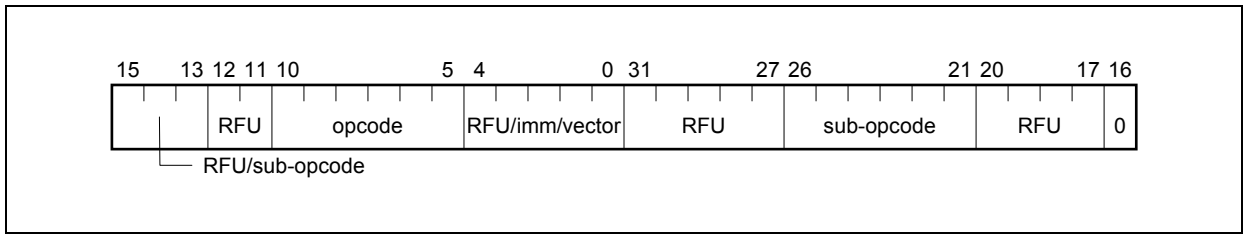
(9) 拡張命令形式 1 (Format IX)

6 ビットのオペコード・フィールドと 6 ビットのサブオペコード・フィールド，2 つの汎用レジスタ指定フィールド（1 つはレジスタ番号フィールドまたは条件コード・フィールドの場合あり）を持つ 32 ビット長命令形式です。



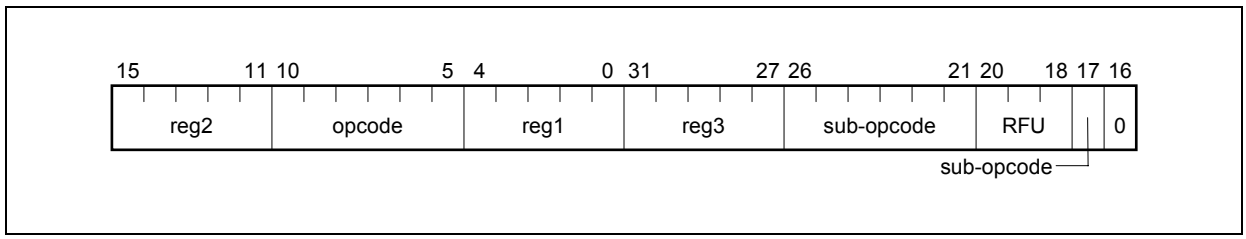
（10）拡張命令形式 2（Format X）

6 ビットのオペコード・フィールド，6 ビットのサブオペコード・フィールドを持つ 32 ビット長命令形式です。



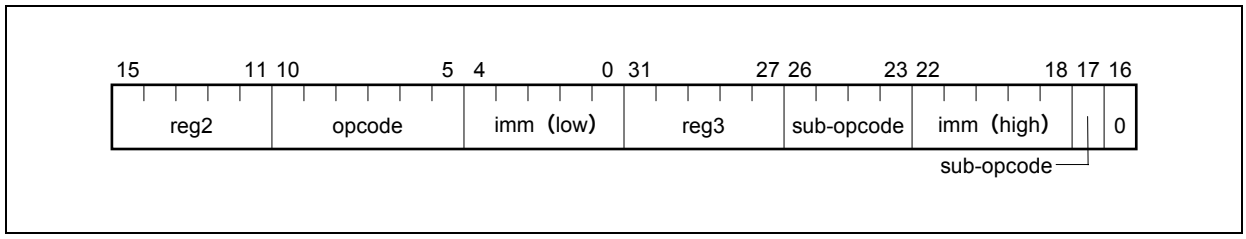
（11）拡張命令形式 3（Format XI）

6 ビットのオペコード・フィールドと 6 ビット＋1 ビットのサブオペコード・フィールド，3 つの汎用レジスタ指定フィールドを持つ 32 ビット長命令形式です。



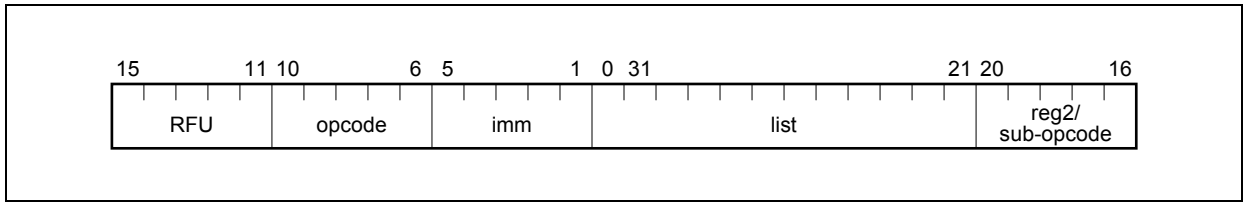
（12）拡張命令形式 4（Format XII）

6 ビットのオペコード・フィールド，4 ビット＋1 ビットのサブオペコード・フィールド，10 ビットのイミーディエト・フィールド，2 つの汎用レジスタ指定フィールドを持つ 32 ビット長命令形式です。



（13）スタック操作命令形式 1（Format XIII）

5 ビットのオペコード・フィールドと 5 ビットのイミーディエト・フィールド，12 ビットのレジスタ・リスト・フィールド，1 つの汎用レジスタ指定フィールド（または 5 ビットのサブオペコード・フィールド）を持つ 32 ビット長命令形式です。



5. 2 命令の概要

(1) ロード命令

メモリからレジスタへのデータ転送を行います。次の命令（ニモニック）があります。

(a) LD 命令

- LD.B　：Load byte
- LD.BU　：Load byte unsigned
- LD.H　：Load half-word
- LD.HU　：Load half-word unsigned
- LD.W　：Load word

(b) SLD 命令

- SLD.B　：Short format load byte
- SLD.BU　：Short format load byte unsigned
- SLD.H　：Short format load half-word
- SLD.HU　：Short format load half-word unsigned
- SLD.W　：Short format load word

(2) ストア命令

レジスタからメモリへのデータ転送を行います。次の命令（ニモニック）があります。

(a) ST 命令

- ST.B　：Store byte
- ST.H　：Store half-word
- ST.W　：Store word

(b) SST 命令

- SST.B　：Short format store byte
- SST.H　：Short format store half-word
- SST.W　：Short format store word

(3) 乗算命令

内蔵のハードウエア乗算器により，1 クロックでの乗算処理を行います。次の命令（ニモニック）があります。

- MUL　：Multiply word
- MULH　：Multiply half-word
- MULHI　：Multiply half-word immediate
- MULU　：Multiply word unsigned

(4) 加算付き乗算命令

乗算後，その結果に対する加算を行います。次の命令（ニモニック）があります。

- MAC　：Multiply word and add
- MACU　：Multiply word unsigned and add

(5) 算術演算命令

加減算，レジスタ間のデータ転送，データ比較を行います。次の命令（ニモニック）があります。

- ADD　：Add
- ADDI　：Add immediate
- CMP　：Compare
- MOV　：Move
- MOVEA　：Move effective address
- MOVHI　：Move high half-word
- SUB　：Subtract
- SUBR　：Subtract reverse

(6) 条件付き演算命令

指定された条件に応じた加減算を行います。次の命令（ニモニック）があります。

- ADF　：Add on condition flag
- SBF　：Subtract on condition flag

(7) 飽和演算命令

飽和加減算を行います。なお，演算の結果が正の最大値（7FFFFFFFH**）を越えたときは** 7FFFFFFFH **を，負の最大値（**80000000H**）を越えたときは** 80000000H **を返します。次の命令（ニモニック）があります。**

- SATADD　：Saturated add
- SATSUB　：Saturated subtract
- SATSUBI　：Saturated subtract immediate
- SATSUBR　：Saturated subtract reverse

(8) 論理演算命令

論理演算を行います。次の命令（ニモニック）があります。

- AND　：AND
- ANDI　：AND immediate
- NOT　：NOT
- OR　：OR
- ORI　：OR immediate
- TST　：Test
- XOR　：Exclusive OR
- XORI　：Exclusive OR immediate

(9) データ操作命令

データ操作とシフト命令があります。シフト命令には，算術シフトと論理シフトがあります。内蔵のバレル・シフタにより，1 クロックで複数ビットのシフトを行います。次の命令（ニモニック）があります。

- BSH　：Byte swap half-word
- BSW　：Byte swap word
- CMOV　：Conditional move
- HSH　：Half-word swap half-word
- HSW　：Half-word swap word
- SAR　：Shift arithmetic right
- SASF　：Shift and set flag condition
- SETF　：Set flag condition
- SHL　：Shift logical left
- SHR　：Shift logical right
- SXB　：Sign extend byte
- SXH　：Sign extend half-word
- ZXB　：Zero extend byte
- ZXH　：Zero extend half-word

(10) ビット・サーチ命令

レジスタに格納されたデータから指定のビットを検索します。

- SCH0L　：Search zero from left
- SCH0R　：Search zero from right
- SCH1L　：Search one from left
- SCH1R　：Search one from right

（11）除算命令

除算を行います。次の命令（ニモニック）があります。

- DIV　：Divide word
- DIVH　：Divide half-word
- DIVHU　：Divide half-word unsigned
- DIVU　：Divide word unsigned

（12）分岐命令

無条件分岐命令（JARL, JMP, JR**）とフラグの状態により制御を変更する条件分岐命令（**Bcond**）があります。分岐命令により指定されたアドレスにプログラムの制御を移します。次の命令（ニモニック）があります。**

- Bcond（BC, BE, BGE, BGT, BH, BL, BLE, BLT, BN, BNC, BNE, BNH, BNL, BNV, BNZ, BP, BR, BSA, BV, BZ）　：Branch on condition code
- JARL　：Jump and register link
- JMP　：Jump register
- JR　：Jump relative

（13）ビット操作命令

メモリのビット・データに対して，論理演算を行います。指定されたビット以外は影響を受けません。次の命令（ニモニック）があります。

- CLR1　：Clear bit
- NOT1　：Not bit
- SET1　：Set bit
- TST1　：Test bit

（14）特殊命令

前項までのカテゴリに含まれない命令です。次の命令（ニモニック）があります。

- CALLT　：Call with table look up
- CTRET　：Return from CALLT
- DI　：Disable interrupt
- DISPOSE　：Function dispose
- EI　：Enable interrupt
- HALT　：Halt
- LDSR　：Load system register
- NOP　：No operation
- PREPARE　：Function prepare
- RETI　：Return from trap or interrupt
- STSR　：Store system register
- SWITCH　：Jump with table look up
- TRAP　：Trap

（15）ディバグ機能用命令

ディバグ機能用に予約された命令です。次の命令（ニモニック）があります。

- DBRET　：Return from debug trap
- DBTRAP　：Debug trap

5. 3 命令セット

この節では，各命令のニモニックごとに（アルファベット順），次の項目に分けて説明します。

- 命令形式　　：命令の記述方法，オペランドを示します（略号については，表5－1参照）。
- オペレーション：命令の機能を示します（略号については，表5－2参照）。
- フォーマット　：命令形式を命令フォーマットで示します（5. 1　**命令フォーマット**参照）。
- オペコード　　：命令のオペコードをビット・フィールドで示します（略号については，表5－3参照）。
- フラグ　　：命令実行により変化するプログラム・ステータス・ワード（PSW）の各フラグの動作を示します。「0」はクリア（リセット）を，「1」はセットを，「－」は変化しないことを示します。
- 説明　　：命令の動作説明をします。
- 補足　　：命令の補足説明をします。
- 注意　　：注意事項を示します。

表5－1　命令形式の凡例

略　号	意　味
reg1	汎用レジスタ（ソース・レジスタとして使用）
reg2	汎用レジスタ（主にデスティネーション・レジスタとして使用。一部の命令で，ソース・レジスタとしても使用）
reg3	汎用レジスタ（主に除算結果の余り，乗算結果の上位32ビットを格納）
bit#3	ビット・ナンバ指定用3ビット・データ
imm ×	× ビット・イミーディエト・データ
disp ×	× ビット・ディスプレースメント・データ
regID	システム・レジスタ番号
vector	トラップ・ベクタ（00H-1FH）を指定する5ビット・データ
cccc	条件コードを示す4ビット・データ
sp	スタック・ポインタ（r3）
ep	エレメント・ポインタ（r30）
list12	レジスタ・リスト

表5－2　オペレーションの凡例

略　号	意　味
←	代入
GR [ ]	汎用レジスタ
SR [ ]	システム・レジスタ
zero-extend (n)	n を，ワード長までゼロ拡張する。
sign-extend (n)	n を，ワード長まで符号拡張する。
load-memory (a, b)	アドレス「a」から，サイズ「b」のデータを読み出す。
store-memory (a, b, c)	アドレス「a」にデータ「b」をサイズ「c」で書き込む。
load-memory-bit (a, b)	アドレス「a」のビット「b」を読み出す。
store-memory-bit (a, b, c)	アドレス「a」のビット「b」に「c」を書き込む。
saturated (n)	n の飽和処理を行う。 計算の結果，n≧7FFFFFFFH となった場合，n = 7FFFFFFFH とする。 計算の結果，n≦80000000H となった場合，n = 80000000H とする。
result	結果をフラグに反映する。
Byte	バイト（8 ビット）
Half-word	ハーフワード（16 ビット）
Word	ワード（32 ビット）
+	加算
−	減算
||	ビット連結
×	乗算
÷	除算
%	除算結果の余り
AND	論理積
OR	論理和
XOR	排他的論理和
NOT	論理否定
logically shift left by	論理左シフト
logically shift right by	論理右シフト
arithmetically shift right by	算術右シフト

表5－3　オペコードの凡例

略　号	意　味
R	reg1 または regID を指定するコードの 1 ビット分データ
r	reg2 を指定するコードの 1 ビット分データ
w	reg3 を指定するコードの 1 ビット分データ
D	ディスプレースメントの 1 ビット分データ（ディスプレースメントの上位ビットを示す）
d	ディスプレースメントの 1 ビット分データ
I	イミーディエトの 1 ビット分データ（イミーディエトの上位ビットを示す）
i	イミーディエトの 1 ビット分データ
cccc	条件コードを示す 4 ビット・データ（表 5－4　条件コード一覧参照）
CCCC	Bcond 命令の条件コードを示す 4 ビット・データ
bbb	ビット・ナンバ指定用 3 ビット・データ
L	レジスタ・リスト中の汎用レジスタを指定する 1 ビット分データ
S	レジスタ・リスト中の EIPC/FEPC, EIPSW/FEPSW レジスタを指定する 1 ビット分データ
P	レジスタ・リスト中の PSW レジスタを指定する 1 ビット分データ

表5－4　条件コード一覧

条件コード（cccc）	条件式
0000	OV = 1
1000	OV = 0
0001	CY = 1
1001	CY = 0
0010	Z = 1
1010	Z = 0
0011	(CY or Z) = 1
1011	(CY or Z) = 0
0100	S = 1
1100	S = 0
0101	always（無条件）
1101	SAT = 1
0110	(S xor OV) = 1
1110	(S xor OV) = 0
0111	((S xor OV) or Z) = 1
1111	((S xor OV) or Z) = 0

＜算術演算命令＞

ADD	Add register/immediate **加算**

[命令形式]　(1) ADD　reg1, reg2
(2) ADD　imm5, reg2

[オペレーション]　(1) GR [reg2] ← GR [reg2] + GR [reg1]
(2) GR [reg2] ← GR [reg2] + sign-extend (imm5)

[フォーマット]　(1) Format I
(2) Format II

[オペコード]

	15	0
(1)	rrrrr001110RRRRR	
(2)	rrrrr010010iiiii	

[フ ラ グ]

CY	MSB **からのキャリーがあれば** 1**，そうでないとき** 0
OV	**オーバフローが起こったとき** 1**，そうでないとき** 0
S	**演算結果が負のとき** 1**，そうでないとき** 0
Z	**演算結果が** 0 **のとき** 1**，そうでないとき** 0
SAT	–

[説　明]　(1) **汎用レジスタ** reg2 **のワード・データに汎用レジスタ** reg1 **のワード・データを加算し，その結果を汎用レジスタ** reg2 **に格納します。汎用レジスタ** reg1 **は影響を受けません。**
(2) **汎用レジスタ** reg2 **のワード・データにワード長まで符号拡張した** 5 **ビット・イミーディエトを加算し，その結果を汎用レジスタ** reg2 **に格納します。**

＜算術演算命令＞

ADDI	Add immediate **加算**

［命令形式］　ADDI　imm16, reg1, reg2

［オペレーション］　GR [reg2] ← GR [reg1] + sign-extend (imm16)

［フォーマット］　Format VI

［オペコード］

15	0	31	16
rrrrr110000RRRRR		iiiiiiiiiiiiiiii	

［フ ラ グ］

CY　MSB **からのキャリーがあれば** 1**，そうでないとき** 0

OV　**オーバフローが起こったとき** 1**，そうでないとき** 0

S　**演算結果が負のとき** 1**，そうでないとき** 0

Z　**演算結果が** 0 **のとき** 1**，そうでないとき** 0

SAT　–

［説　明］　**汎用レジスタ** reg1 **のワード・データにワード長まで符号拡張した** 16 **ビット・イミーディエトを加算し，その結果を汎用レジスタ** reg2 **に格納します。汎用レジスタ** reg1 **は影響を受けません。**

＜条件付き演算命令＞

ADF	Add on condition flag 条件付き加算

［命令形式］　ADF　cccc, reg1, reg2, reg3

［オペレーション］　if conditions are satisfied

then GR [reg3] ← GR [reg1] + GR [reg2] +1

else GR [reg3] ← GR [reg1] + GR [reg2] +0

［フォーマット］　Format XI

［オペコード］

15	0	31	16
rrrrr111111RRRRR		wwwww011101cccc0	

［フ ラ グ］

CY	MSB からのキャリーがあれば 1，そうでないとき 0
OV	オーバフローが起こったとき 1，そうでないとき 0
S	演算結果が負のとき 1，そうでないとき 0
Z	演算結果が 0 のとき 1，そうでないとき 0
SAT	–

［説　明］　汎用レジスタ reg2 のワード・データに汎用レジスタ reg1 のワード・データを加算した結果に 1（加算結果が条件コード「cccc」で指定された条件に満たされた場合）または 0（加算結果が条件コード「cccc」で指定された条件に満たされなかった場合）を加算し，その結果を汎用レジスタ reg3 に格納します。汎用レジスタ reg2 は影響を受けません。

次の表で示されている条件コードのうちの 1 つを「cccc」として指定してください（ただし，cccc ≠ 1101）。

条件コード	条件式	条件コード	条件式
0000	OV = 1	0100	S = 1
1000	OV = 0	1100	S = 0
0001	CY = 1	0101	always（無条件）
1001	CY = 0	0110	(S xor OV) = 1
0010	Z = 1	1110	(S xor OV) = 0
1010	Z = 0	0111	((S xor OV) or Z) = 1
0011	(CY or Z) = 1	1111	((S xor OV) or Z) = 0
1011	(CY or Z) = 0	（1101）	設定禁止

＜論理演算命令＞

AND	AND 論理積

［命令形式］　AND　reg1, reg2

［オペレーション］　GR [reg2] ← GR [reg2] AND GR [reg1]

［フォーマット］　Format I

［オペコード］

15	0
rrrrr001010RRRRR	

［フ ラ グ］

CY　–

OV　0

S　演算結果のワード・データの MSB が 1 のとき 1，そうでないとき 0

Z　演算結果が 0 のとき 1，そうでないとき 0

SAT　–

［説　明］　汎用レジスタ reg2 のワード・データと汎用レジスタ reg1 のワード・データの論理積をとり，その結果を汎用レジスタ reg2 に格納します。汎用レジスタ reg1 は影響を受けません。

＜論理演算命令＞

ANDI	AND immediate 論理積

［命令形式］　ANDI　imm16, reg1, reg2

［オペレーション］　GR [reg2] ← GR [reg1] AND zero-extend (imm16)

［フォーマット］　Format VI

［オペコード］

15	0	31	16
rrrrr110110RRRRR		iiiiiiiiiiiiiiii	

［フ ラ グ］

CY　–

OV　0

S　演算結果のワード・データの MSB が 1 のとき 1，そうでないとき 0

Z　演算結果が 0 のとき 1，そうでないとき 0

SAT　–

［説　明］　汎用レジスタ reg1 のワード・データと 16 ビット・イミーディエトをワード長までゼロ拡張した値の論理積をとり，その結果を汎用レジスタ reg2 に格納します。汎用レジスタ reg1 は影響を受けません。

＜分岐命令＞

Bcond	Branch on condition code with 9-bit displacement 条件分岐

［命令形式］　Bcond　disp9

［オペレーション］　if conditions are satisfied
then PC ← PC + sign-extend (disp9)

［フォーマット］　Format III

［オペコード］　15　0

dddddddddddddddd

ただし，ddddddddは disp9 の上位 8 ビットです。

［フ ラ グ］　CY　–
OV　–
S　–
Z　–
SAT　–

［説　明］　命令が指定する PSW レジスタの各フラグをテストし，条件を満たしているときは分岐し，そうでないときは次の命令に進みます。分岐先 PC は，現在の PC と 8 ビット・イミーディエトを 1 ビット・シフトしてワード長まで符号拡張した 9 ビット・ディスプレースメントを加算した値です。

［補　足］　9 ビット・ディスプレースメントのビット 0 は 0 にマスクされます。なお，計算に使用される現在の PC とは，この命令自身の先頭バイトのアドレスであるためディスプレースメント値が 0 のときは，分岐先はこの命令自身になります。

表5－5　Bcond命令一覧

命　令		条件コード（CCCC）	フラグの状態	分岐条件
符号付き整数	BGE	1110	(S xor OV) = 0	Greater than or equal signed
	BGT	1111	((S xor OV) or Z) = 0	Greater than signed
	BLE	0111	((S xor OV) or Z) = 1	Less than or equal signed
	BLT	0110	(S xor OV) = 1	Less than signed
整数符号なし整数	BH	1011	(CY or Z) = 0	Higher (Greater than)
	BL	0001	CY = 1	Lower (Less than)
	BNH	0011	(CY or Z) = 1	Not higher (Less than or equal)
	BNL	1001	CY = 0	Not lower (Greater than or equal)
共通	BE	0010	Z = 1	Equal
	BNE	1010	Z = 0	Not equal
その他	BC	0001	CY = 1	Carry
	BN	0100	S = 1	Negative
	BNC	1001	CY = 0	No carry
	BNV	1000	OV = 0	No overflow
	BNZ	1010	Z = 0	Not zero
	BP	1100	S = 0	Positive
	BR	0101	–	Always（無条件）
	BSA	1101	SAT = 1	Saturated
	BV	0000	OV = 1	Overflow
	BZ	0010	Z = 1	Zero

注意　飽和演算命令の実行結果で SAT **フラグがセット（**1**）された場合，符号付き整数の条件分岐（**BGE, BGT, BLE, BLT**）は，分岐条件に意味がなくなります。これは，次の理由によるものです。通常の演算では，結果が正の最大値を越えると負の値になり，負の最大値を越えたときは正の値になります。つまり，オーバフローが生じると，**S **フラグが反転（**0→1, 1→0**）します。一方，飽和演算命令では，結果が正の最大値を越えたときは正の値で，負の最大値を越えたときは負の値で飽和します。通常の演算とは異なり，オーバフローが生じても** S **フラグは反転しません。このように，演算結果が飽和したときの** S **フラグは通常の演算とは異なるので，**OV **フラグとの排他的論理和（**XOR**）をとる分岐条件に意味がなくなります。**

＜データ操作命令＞

BSH	Byte swap half-word ハーフワード・データのバイト・スワップ

［命令形式］　BSH　reg2, reg3

［オペレーション］　GR [reg3] ← GR [reg2] (23:16) || GR [reg2] (31:24) || GR [reg2] (7:0) || GR [reg2] (15:8)

［フォーマット］　Format XII

［オペコード］

15	0	31	16
rrrrr11111100000		wwwww01101000010	

［フ ラ グ］

CY　演算結果の下位ハーフワード・データ中に，0 のバイトが 1 つ以上含まれるとき 1，そうでないとき 0

OV　0

S　演算結果のワード・データの MSB が 1 のとき 1，そうでないとき 0

Z　演算結果の下位ハーフワード・データが 0 のとき 1，そうでないとき 0

SAT　–

［説　明］　エンディアン変換します。

＜データ操作命令＞

BSW	Byte swap word ワード・データのバイト・スワップ

［命令形式］　BSW　reg2, reg3

［オペレーション］　GR [reg3] ← GR [reg2] (7:0) || GR [reg2] (15:8) || GR [reg2] (23:16) || GR [reg2] (31:24)

［フォーマット］　Format XII

［オペコード］

15	0	31	16
rrrrr11111100000		wwwww01101000000	

［フ ラ グ］

CY　演算結果のワード・データ中に，0 のバイトが 1 つ以上含まれるとき 1，
そうでないとき 0

OV　0

S　演算結果のワード・データの MSB が 1 のとき 1，そうでないとき 0

Z　演算結果のワード・データが 0 のとき 1，そうでないとき 0

SAT　–

［説　明］　エンディアン変換します。

＜特殊命令＞

CALLT	Call with table look up テーブル参照によるサブルーチン・コール

［命令形式］　CALLT　imm6

［オペレーション］　CTPC ← PC + 2 (return PC)
CTPSW ← PSW
adr ← CTBP + zero-extend (imm6 logically shift left by 1)
PC ← CTBP + zero-extend (Load-memory (adr, Half-word))

［フォーマット］　Format II

［オペコード］

15	0
0000001000iiiiii	

［フ ラ グ］　CY　–
OV　–
S　–
Z　–
SAT　–

［説　明］　次の順に処理を行います。

<1> 復帰 PC と PSW レジスタの内容を CTPC レジスタと CTPSW レジスタに転送
<2> CTBP の値と，1 ビット論理左シフトしワード長までゼロ拡張した 6 ビット・イミーディエト・データを加算して 32 ビット・テーブル・エントリ・アドレスを生成
<3> <2>で生成されたアドレスのハーフワードをロードし，ワード長までゼロ拡張
<4> <3>のデータに CTBP レジスタの値を加算して 32 ビット・ターゲット・アドレスを生成
<5> <4>で生成されたターゲット・アドレスへ分岐

［注　意］　命令実行中に割り込みが発生すると，リード／ライト・サイクルが終了したあとに命令の実行を中止する場合があります。

＜ビット操作命令＞

CLR1	Clear bit ビット・クリア

［命令形式］　（1）CLR1　bit#3, disp16 [reg1]

（2）CLR1　reg2, [reg1]

［オペレーション］　（1）adr ← GR [reg1] + sign-extend (disp16)

Z フラグ← Not (Load-memory-bit (adr, bit#3))

Store-memory-bit (adr, bit#3, 0)

（2）adr ← GR [reg1]

Z フラグ ← Not (Load-memory-bit (adr, reg2))

Store-memory-bit (adr, reg2, 0)

［フォーマット］　（1）Format VIII

（2）Format IX

［オペコード］

	15	0	31	16
(1)	10bbb111110RRRRR		ddddddddddddddddd	
(2)	rrrrr111111RRRRR		0000000011100100	

［フ ラ グ］　CY　–

OV　–

S　–

Z　指定したビットが 0 のとき 1，指定したビットが 1 のとき 0

SAT　–

［説　明］　（1）まず，汎用レジスタ reg1 のワード・データと，ワード長まで符号拡張した 16 ビット・ディスプレースメントを加算して 32 ビット・アドレスを生成します。生成したアドレスのバイト・データを読み出し，3 ビットのビット・ナンバで指定されるビットをクリア（0）し，元のアドレスに書き戻します。

（2）まず，汎用レジスタ reg1 のワード・データを読み出して 32 ビット・アドレスを生成します。生成したアドレスのバイト・データを読み出し，汎用レジスタ reg2 の下位 3 ビットで指定されるビットをクリア（0）し，元のアドレスに書き戻します。

［補　足］　PSW レジスタの Z フラグはこの命令を実行する前に該当ビットが 0 か 1 だったかを示します。この命令実行後の該当ビットの内容を示すものではありません。

＜データ操作命令＞

CMOV	Conditional move 条件付き転送

［命令形式］　（1）CMOV　cccc, reg1, reg2, reg3
（2）CMOV　cccc, imm5, reg2, reg3

［オペレーション］　（1）if conditions are satisfied
then GR [reg3] ← GR [reg1]
else GR [reg3] ← GR [reg2]
（2）if conditions are satisfied
then GR [reg3] ← sign-extended (imm5)
else GR [reg3] ← GR [reg2]

［フォーマット］　（1）Format XI
（2）Format XII

［オペコード］

	15	0	31	16
(1)	rrrrr111111RRRRR		wwwww011001cccc0	
(2)	rrrrr111111iiiii		wwwww011000cccc0	

［フ ラ グ］　CY　–
OV　–
S　–
Z　–
SAT　–

［説　明］　（1）条件コード「cccc」で指定された条件が満たされた場合は汎用レジスタ reg1 のデータを，満たされなかった場合は汎用レジスタ reg2 のデータを，汎用レジスタ reg3 に転送します。次の表で示されている条件コードのうちの 1 つを「cccc」として指定してください。

条件コード	条件式	条件コード	条件式
0000	OV = 1	0100	S = 1
1000	OV = 0	1100	S = 0
0001	CY = 1	0101	always（無条件）
1001	CY = 0	1101	SAT = 1
0010	Z = 1	0110	(S xor OV) = 1
1010	Z = 0	1110	(S xor OV) = 0
0011	(CY or Z) = 1	0111	((S xor OV) or Z) = 1
1011	(CY or Z) = 0	1111	((S xor OV) or Z) = 0

(2) 条件コード「cccc」で指定された条件が満たされた場合はワード長まで符号拡張した5ビット・イミーディエト・データを，満たされなかった場合は汎用レジスタ reg2 のデータを，汎用レジスタ reg3 に転送します。次の表で示されている条件コードのうちの1つを「cccc」として指定してください。

条件コード	条件式	条件コード	条件式
0000	OV = 1	0100	S = 1
1000	OV = 0	1100	S = 0
0001	CY = 1	0101	always（無条件）
1001	CY = 0	1101	SAT = 1
0010	Z = 1	0110	(S xor OV) = 1
1010	Z = 0	1110	(S xor OV) = 0
0011	(CY or Z) = 1	0111	((S xor OV) or Z) = 1
1011	(CY or Z) = 0	1111	((S xor OV) or Z) = 0

［補　足］　SETF 命令を参照してください。

＜算術演算命令＞

CMP	Compare register/immediate (5-bit) 比較

［命令形式］　（1）CMP　reg1, reg2
（2）CMP　imm5, reg2

［オペレーション］　（1）result ← GR [reg2] – GR [reg1]
（2）result ← GR [reg2] – sign-extend (imm5)

［フォーマット］　（1）Format I
（2）Format II

［オペコード］

15　0
（1）rrrrr001111RRRRR

15　0
（2）rrrrr010011iiiii

［フ ラ グ］　CY　MSB へのボローがあれば 1，そうでないとき 0
OV　オーバフローが起こったとき 1，そうでないとき 0
S　演算結果が負のとき 1，そうでないとき 0
Z　演算結果が 0 のとき 1，そうでないとき 0
SAT　–

［説　明］　（1）汎用レジスタ reg2 のワード・データと汎用レジスタ reg1 のワード・データを比較し，結果を PSW レジスタの各フラグに示します。比較は汎用レジスタ reg2 のワード・データから汎用レジスタ reg1 の内容を減算することで行います。汎用レジスタ reg1, reg2 は影響を受けません。

（2）汎用レジスタ reg2 のワード・データとワード長まで符号拡張した 5 ビット・イミーディエトを比較し，結果を PSW レジスタの各フラグに示します。比較は汎用レジスタ reg2 のワード・データから符号拡張したイミーディエトの内容を減算することで行います。汎用レジスタ reg2 は影響を受けません。

＜特殊命令＞

CTRET	Return from CALLT サブルーチン・コールからの復帰

［命令形式］　CTRET

［オペレーション］　PC ← CTPC
PSW ← CTPSW

［フォーマット］　Format X

［オペコード］

15	0	31	16
0000011111100000		0000000101000100	

［フ ラ グ］

CY	CTPSW レジスタから読み出した値が設定される
OV	CTPSW レジスタから読み出した値が設定される
S	CTPSW レジスタから読み出した値が設定される
Z	CTPSW レジスタから読み出した値が設定される
SAT	CTPSW レジスタから読み出した値が設定される

［説　明］　システム・レジスタから復帰 PC と PSW を取り出し，CALLT 命令により呼び出されたルーチンから復帰します。この命令の動作は次のとおりです。

<1> 復帰 PC と PSW を，CTPC レジスタと CTPSW レジスタから取り出します。
<2> 取り出した復帰 PC と PSW を PC レジスタと PSW レジスタに設定し，制御を移します。

＜ディバグ機能用命令＞

DBRET	Return from debug trap ディバグ・トラップからの復帰

［命令形式］　DBRET

［オペレーション］　PC ← DBPC
PSW ← DBPSW

［フォーマット］　Format X

［オペコード］

15	0	31	16
0000011111100000		0000000101000110	

［フ ラ グ］
CY　DBPSW レジスタから読み出した値が設定される
OV　DBPSW レジスタから読み出した値が設定される
S　DBPSW レジスタから読み出した値が設定される
Z　DBPSW レジスタから読み出した値が設定される
SAT　DBPSW レジスタから読み出した値が設定される

［説　明］　システム・レジスタから復帰 PC と PSW を取り出し，ディバグ・モードから復帰します。

［注　意］　DBRET 命令はディバグを目的とした命令のため，基本的にディバグ・ツールが使用しています。このためディバグ・ツールが使用しているときに，アプリケーションが使用すると誤動作する場合があります。

＜ディバグ機能用命令＞

DBTRAP	Debug trap ディバグ・トラップ

［命令形式］　DBTRAP

［オペレーション］　DBPC ← PC + 2 (復帰 PC)
DBPSW ← PSW
PSW.NP ← 1
PSW.EP ← 1
PSW.ID ← 1
DIR.DM ← 1
PC ← 00000060H

［フォーマット］　Format I

［オペコード］

15															0
1	1	1	1	1	0	0	0	0	1	0	0	0	0	0	0

［フ ラ グ］　CY　–
OV　–
S　–
Z　–
SAT　–

［説　明］　復帰 PC（DBTRAP 命令の次の命令のアドレス）と現在の PSW レジスタの内容を，それぞれ DBPC レジスタと DBPSW レジスタに退避し，PSW レジスタの NP, EP, ID フラグと DIR レジスタの DM ビットをセット（1）します。
続いて，PC レジスタに例外トラップのハンドラ・アドレス（00000060H）をセットし，ディバグ・モードに移行します。NP, EP, ID フラグ以外の PSW レジスタの各フラグは影響を受けません。
なお，DBPC レジスタに退避される値は，DBTRAP 命令の次の命令のアドレスです。

［注　意］　DBTRAP 命令はディバグを目的とした命令のため，基本的にディバグ・ツールが使用しています。このためディバグ・ツールが使用しているときに，アプリケーションが使用すると誤動作する場合があります。

＜特殊命令＞

DI	Disable interrupt マスカブル割り込みの禁止

［命令形式］　DI

［オペレーション］　PSW.ID ← 1（マスカブル割り込みの禁止）

［フォーマット］　Format X

［オペコード］

15	0	31	16
0000011111100000		0000000101100000	

［フ ラ グ］

CY	–
OV	–
S	–
Z	–
SAT	–
ID	1

［説　明］　PSW レジスタの ID フラグをセット（1）し，この命令実行中からマスカブル割り込みの受け付けを禁止します。

［補　足］　この命令の実行中は，割り込みのサンプリングをしません。この命令による PSW レジスタのフラグの書き換えが有効になるのは次の命令からですが，割り込みのサンプリングをこの命令実行中に行わないため，実際はこの命令実行中から割り込みを禁止します。ただし，ノンマスカブル割り込み（NMI）は，この命令の実行後も受け付けは禁止されません。

＜特殊命令＞

DISPOSE

Function dispose

スタック・フレームの削除

［命令形式］　（1）DISPOSE　imm5, list12

（2）DISPOSE　imm5, list12, [reg1]

［オペレーション］　（1）sp ← sp + zero-extend (imm5 logically shift left by 2)

GR [reg in list12] ← Load-memory (sp, Word)

sp ← sp + 4

repeat 2 steps above until all regs in list12 is loaded

（2）sp ← sp + zero-extend (imm5 logically shift left by 2)

GR [reg in list12] ← Load-memory (sp, Word)

sp ← sp + 4

repeat 2 states above until all regs in list12 is loaded

PC ← GR [reg1]

［フォーマット］　Format XIII

［オペコード］

	15	0	31	16
(1)	0000011001iiiiiL		LLLLLLLLLLL00000	
(2)	0000011001iiiiiL		LLLLLLLLLLRRRRR	

ただし，RRRRR **は，**00000 **以外です。**

また，LLLLLLLLLLL **は，レジスタ・リスト「**list12**」の中の対応するビットの値を示します（たとえば，オペコード中のビット** 21 **の「**L**」は** list12 **のビット** 21 **の値を示します）。**

list12 **は，次のように定義される** 32 **ビットのレジスタ・リストです。**

31	30	29	28	27	26	25	24	23	22	21	20 … 1	0
r24	r25	r26	r27	r20	r21	r22	r23	r28	r29	r31	−	r30

ビット 31-21 **とビット** 0 **の各ビットに汎用レジスタ（**r20-r31**）が対応しており，セット（**1**）されたビットに対応するレジスタが操作の対象として指定されます。たとえば，**r20, r30 **を指定する場合，**list12 **の値は次のようになります（レジスタが対応付けられていないビット** 20-1 **への設定値は任意です）。**

- **レジスタが対応付けられていないビットの値をすべて** 0 **とした場合：**08000001H
- **レジスタが対応付けられていないビットの値をすべて** 1 **とした場合：**081FFFFFH

［フ ラ グ］　CY　–
OV　–
S　–
Z　–
SAT　–

［説　明］　（1）5 ビット・イミーディエト・データを，2 ビット論理左シフトし，ワード長までゼロ拡張したものを，sp に加算します。そして，list12 で指定されている汎用レジスタに復帰（sp で指定するアドレスからデータをロードし，sp に 4 を加算）します。なお，アドレスのビット 0 は，0 にマスクされます。
（2）5 ビット・イミーディエト・データを，2 ビット論理左シフトし，ワード長までゼロ拡張したものを，sp に加算します。そして，list12 で指定されている汎用レジスタに復帰（sp で指定するアドレスからデータをロードし，sp に 4 を加算）し，汎用レジスタ reg1 で指定されたアドレスに制御を移します。なお，アドレスのビット 0 は，0 にマスクされます。

［補　足］　list12 の汎用レジスタは，降順にロードされます（r31, r30, ... , r20）。
imm5 は，自動変数と一時データのためのスタック・フレームを復元します。
sp で指定された下位 2 ビットのアドレスは，ミス・アライン・アクセスが可能であっても，0 にマスクされます。
また，sp の更新前に割り込みが発生すると，実行を中止し，戻り番地をこの命令の先頭アドレスとして割り込みを処理してから，割り込み処理完了後に再実行します（sp は割り込み実行開始前の元の値を保持します）。

［注　意］　命令実行中に割り込みが発生すると，スタック操作を行うため，リード／ライト・サイクルとレジスタ値の書き換えが終了したあとに中止する場合があります。割り込みから復帰したあとに再実行されます。

＜除算命令＞

DIV	Divide word （符号付き）ワード・データの除算

［命令形式］　DIV　reg1, reg2, reg3

［オペレーション］　GR [reg2] ← GR [reg2] ÷ GR [reg1]
GR [reg3] ← GR [reg2] % GR [reg1]

［フォーマット］　Format XI

［オペコード］

15	0	31	16
rrrrr111111RRRRR		wwwww01011000000	

［フ ラ グ］
CY　–
OV　オーバフローが起こったとき 1，そうでないとき 0
S　演算結果が負のとき 1，そうでないとき 0
Z　演算結果が 0 のとき 1，そうでないとき 0
SAT　–

［説　明］　汎用レジスタ reg2 のワード・データを汎用レジスタ reg1 のワード・データで除算し，その商を汎用レジスタ reg2 に，余りを汎用レジスタ reg3 に格納します。0 で割ったときは，オーバフローを生じ，商は不定となります。汎用レジスタ reg1 は影響を受けません。

［補　足］　オーバフローは負の最大値（80000000H）を–1 で割ったとき（商が 80000000H）と，ゼロによる除算のとき（商は不定）に生じます。
この命令実行中に割り込みが発生すると，実行を中止し，戻り番地をこの命令の先頭アドレスとして割り込みを処理してから，割り込み処理完了後に再実行します。この場合，汎用レジスタ reg1 と汎用レジスタ reg2 はこの命令実行前の値を保持します。
また，汎用レジスタ reg2 と汎用レジスタ reg3 が同じレジスタの場合，そのレジスタには余りが格納されます。

＜除算命令＞

DIVH	Divide half-word （符号付き）ハーフワード・データの除算

［命令形式］　（1）DIVH　reg1, reg2
（2）DIVH　reg1, reg2, reg3

［オペレーション］　（1）GR [reg2] ← GR [reg2] ÷ GR [reg1]
（2）GR [reg2] ← GR [reg2] ÷ GR [reg1]
GR [reg3] ← GR [reg2] % GR [reg1]

［フォーマット］　（1）Format I
（2）Format XI

［オペコード］

（1）

15	0
rrrrr000010RRRRR	

（2）

15	0	31	16
rrrrr111111RRRRR		wwwww01010000000	

［フ ラ グ］　CY　–
OV　オーバフローが起こったとき 1，そうでないとき 0
S　演算結果が負のとき 1，そうでないとき 0
Z　演算結果が 0 のとき 1，そうでないとき 0
SAT　–

［説　明］　（1）汎用レジスタ reg2 のワード・データを汎用レジスタ reg1 の下位ハーフワード・データで除算し，その商を汎用レジスタ reg2 に格納します。0 で割ったときは，オーバフローを生じ，商は不定となります。汎用レジスタ reg1 は影響を受けません。
（2）汎用レジスタ reg2 のワード・データを汎用レジスタ reg1 の下位ハーフワード・データで除算し，その商を汎用レジスタ reg2 に，余りを汎用レジスタ reg3 に格納します。0 で割ったときは，オーバフローを生じ，商は不定となります。汎用レジスタ reg1 は影響を受けません。

［補　足］　(1) 除算結果の余りは格納されません。オーバフローは負の最大値（80000000H）を −1 で割ったとき（商が 80000000H）と，ゼロによる除算のとき（商が不定）に生じます。
この命令実行中に割り込みが発生すると，実行を中止し，戻り番地をこの命令の先頭アドレスとして割り込みを処理してから，割り込み処理完了後に再実行します。この場合，汎用レジスタ reg1 と汎用レジスタ reg2 はこの命令実行前の値を保持します。
なお，reg2 には r0 を指定しないでください。
また，除算の際，汎用レジスタ reg1 の上位 16 ビットを無視します。

(2) オーバフローは負の最大値（80000000H）を −1 で割ったとき（商が 80000000H）と，ゼロによる除算のとき（商が不定）に生じます。
この命令実行中に割り込みが発生すると，実行を中止し，戻り番地をこの命令の先頭アドレスとして割り込みを処理してから，割り込み処理完了後に再実行します。この場合，汎用レジスタ reg1 と汎用レジスタ reg2 はこの命令実行前の値を保持します。
また，除算の際，汎用レジスタ reg1 の上位 16 ビットを無視します。
なお，汎用レジスタ reg2 と汎用レジスタ reg3 が同じレジスタの場合，そのレジスタには余りが格納されます。

＜除算命令＞

DIVHU	Divide half-word unsigned （符号なし）ハーフワード・データの除算

［命令形式］　DIVHU　reg1, reg2, reg3

［オペレーション］　GR [reg2] ← GR [reg2] ÷ GR [reg1]
GR [reg3] ← GR [reg2] % GR [reg1]

［フォーマット］　Format XI

［オペコード］

15	0	31	16
rrrrr111111RRRRR		wwwww01010000010	

［フ ラ グ］
CY　–
OV　オーバフローが起こったとき 1，そうでないとき 0
S　演算結果のワード・データの MSB が 1 のとき 1，そうでないとき 0
Z　演算結果が 0 のとき 1，そうでないとき 0
SAT　–

［説　明］　汎用レジスタ reg2 のワード・データを汎用レジスタ reg1 の下位ハーフワード・データで除算し，その商を汎用レジスタ reg2 に，余りを汎用レジスタ reg3 に格納します。0 で割ったときは，オーバフローを生じ，商は不定となります。汎用レジスタ reg1 は影響を受けません。

［補　足］　オーバフローはゼロによる除算のとき（商は不定）に生じます。
この命令実行中に割り込みが発生すると，実行を中止し，戻り番地をこの命令の先頭アドレスとして割り込みを処理してから，割り込み処理完了後に再実行します。この場合，汎用レジスタ reg1 と汎用レジスタ reg2 はこの命令実行前の値を保持します。
なお，汎用レジスタ reg2 と汎用レジスタ reg3 が同じレジスタの場合，そのレジスタには余りが格納されます。

＜除算命令＞

DIVU	Divide word unsigned （符号なし）ワード・データの除算

［命令形式］　DIVU　reg1, reg2, reg3

［オペレーション］　GR [reg2] ← GR [reg2] ÷ GR [reg1]
GR [reg3] ← GR [reg2] % GR [reg1]

［フォーマット］　Format XI

［オペコード］

15	0	31	16
rrrrr111111RRRRR		wwwww01011000010	

［フ ラ グ］
CY　–
OV　オーバフローが起こったとき 1，そうでないとき 0
S　演算結果のワード・データ MSB が 1 のとき 1，そうでないとき 0
Z　演算結果が 0 のとき 1，そうでないとき 0
SAT　–

［説　明］　汎用レジスタ reg2 のワード・データを汎用レジスタ reg1 のワード・データで除算し，その商を汎用レジスタ reg2 に，余りを汎用レジスタ reg3 に格納します。0 で割ったときは，オーバフローを生じ，商は不定となります。汎用レジスタ reg1 は影響を受けません。

［補　足］　オーバフローはゼロによる除算のとき（商は不定）に生じます。
この命令実行中に割り込みが発生すると，実行を中止し，戻り番地をこの命令の先頭アドレスとして割り込みを処理してから，割り込み処理完了後に再実行します。この場合，汎用レジスタ reg1 と汎用レジスタ reg2 はこの命令実行前の値を保持します。
また，汎用レジスタ reg2 と汎用レジスタ reg3 が同じレジスタの場合，そのレジスタには余りが格納されます。

＜特殊命令＞

EI	Enable interrupt マスカブル割り込みの許可

［命令形式］　EI

［オペレーション］　PSW.ID ← 0（マスカブル割り込みの許可）

［フォーマット］　Format X

［オペコード］

15	0	31	16
1000011111100000		0000000101100000	

［フ ラ グ］　CY –
OV –
S –
Z –
SAT –
ID 0

［説　明］　PSW レジスタの ID フラグをクリア（0）し，次の命令よりマスカブル割り込みの受け付けを許可します。

［補　足］　この命令の実行中は，割り込みのサンプリングをしません。

＜特殊命令＞

HALT	Halt 停止

［命令形式］　HALT

［オペレーション］　停止する

［フォーマット］　Format X

［オペコード］

15	0 31	16
0000011111100000	0000000100100000	

［フ ラ グ］　CY –
OV –
S –
Z –
SAT –

［説　明］　CPU の動作クロックを停止させ，HALT モードに移行します。

［補　足］　HALT モードは次の 3 つの要因によって解除されます。

- リセット入力
- ノンマスカブル割り込み要求
- マスクされていないマスカブル割り込み要求

なお，HALT モード中に割り込みを受け付けた場合，EIPC レジスタまたは FEPC レジスタには，この命令の次の命令アドレスが格納されます。

＜データ操作命令＞

HSH	Half-word swap half-word ハーフワード・データのハーフワード・スワップ

［命令形式］　HSH　reg2, reg3

［オペレーション］　GR [reg3] ← GR [reg2]

［フォーマット］　Format XII

［オペコード］

15	0	31	16
rrrrr11111100000		wwwww01101000110	

［フ ラ グ］

CY　演算結果の下位ハーフワードが 0 のとき 1，そうでないとき 0

OV　0

S　演算結果のワード・データの MSB が 1 のとき 1，そうでないとき 0

Z　演算結果の下位ハーフワードが 0 のとき 1，そうでないとき 0

SAT　–

［説　明］　汎用レジスタ reg2 の内容を汎用レジスタ reg3 に格納し，フラグの判定結果を PSW レジスタに格納します。

＜データ操作命令＞

HSW	Half-word swap word ワード・データのハーフワード・スワップ

［命令形式］　HSW　reg2, reg3

［オペレーション］　GR [reg3] ← GR [reg2] (15:0) || GR [reg2] (31:16)

［フォーマット］　Format XII

［オペコード］

15	0	31	16
rrrrr11111100000		wwwww01101000100	

［フ ラ グ］

CY　演算結果のワード・データ中に，0 のハーフワードが 1 つ以上含まれるとき 1，そうでないとき 0

OV　0

S　演算結果のワード・データの MSB が 1 のとき 1，そうでないとき 0

Z　演算結果のワード・データが 0 のとき 1，そうでないとき 0

SAT　–

［説　明］　エンディアン変換します。

＜分岐命令＞

JARL	Jump and register link 分岐とレジスタ・リンク

［命令形式］　（1）JARL　disp22, reg2
（2）JARL　disp32, reg1

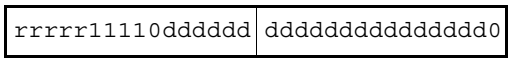
［オペレーション］　（1）GR [reg2] ← PC + 4
PC ← PC + sign-extend (disp22)
（2）GR [reg1] ← PC + 6
PC ← PC + disp32

［フォーマット］　（1）Format V
（2）Format VI

［オペコード］

15　0 31　16

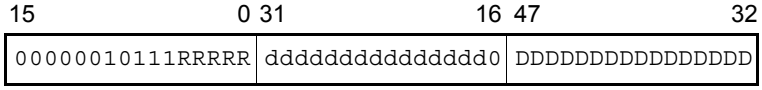
（1）



ただし，ddddddddddddddddddddd は disp22 の上位 21 ビットです。

15　0 31　16 47　32

（2）



ただし，DDDDDDDDDDDDDDDDddddddddddddddd は disp32 の上位 31 ビットです。

［フ ラ グ］　CY　–
OV　–
S　–
Z　–
SAT　–

［説　明］　（1）現在の PC に 4 を加算した値を汎用レジスタ reg2 に退避し，現在の PC とワード長まで符号拡張した 22 ビット・ディスプレースメントを加算した値を PC レジスタに設定し，制御を移します。22 ビット・ディスプレースメントのビット 0 は 0 にマスクされます。
（2）現在の PC に 6 を加算した値を汎用レジスタ reg1 に退避し，現在の PC と 32 ビット・ディスプレースメントを加算した値を PC レジスタに設定し，制御を移します。32 ビット・ディスプレースメントのビット 0 は 0 にマスクされます。

［補　足］　計算に使用される現在の PC とは，この命令自身の先頭バイトのアドレスであるためディスプレースメント値が 0 のときは，分岐先はこの命令自身になります。
この命令は，サブルーチン制御命令のコールに相当し，復帰 PC を汎用レジスタ reg1 または reg2 に格納します。一方，リターンに相当する JMP 命令では，復帰 PC を格納している汎用レジスタを汎用レジスタ reg1 として指定して，使用できます。

＜分岐命令＞

JMP	Jump register 無条件分岐（レジスタ間接）

［命令形式］　（1）JMP　[reg1]

（2）JMP　disp32 [reg1]

［オペレーション］　（1）PC ← GR [reg1]

（2）PC ← GR [reg1] + disp32

［フォーマット］　（1）Format I

（2）Format VI

［オペコード］

（1）

15	0
00000000011RRRRR	

（2）

15	0	31	16	47	32
00000110111RRRRR		ddddddddddddddd0		DDDDDDDDDDDDDDDD	

ただし，DDDDDDDDDDDDDDDDddddddddddddddd **は** disp32 **の上位** 31 **ビットです。**

［フ ラ グ］　CY　–

OV　–

S　–

Z　–

SAT　–

［説　明］　（1）**汎用レジスタ** reg1 **で指定されるアドレスに制御を移します。アドレスのビット** 0 **は** 0 **にマスクされます。**

（2）**汎用レジスタ** reg1 **に** 32 **ビット・ディスプレースメントを加算したアドレスに制御を移します。アドレスのビット** 0 **は** 0 **にマスクされます。**

［補　足］　**この命令をサブルーチン制御命令のリターンとして使用する場合は，復帰** PC **を汎用レジスタ** reg1 **で指定します。**

＜分岐命令＞

JR	Jump relative 無条件分岐（PC 相対）

［命令形式］　（1）JR　disp22
（2）JR　disp32

［オペレーション］　（1）PC ← PC + sign-extend (disp22)
（2）PC ← PC + disp32

［フォーマット］　（1）Format V
（2）Format VI

［オペコード］

（1）

15	0	31	16
0000011110dddddd		dddddddddddddddd0	

ただし，ddddddddddddddddddddd は disp22 の上位 21 ビットです。

（2）

15	0	31	16	47	32
0000001011100000		ddddddddddddddd0		DDDDDDDDDDDDDDDD	

ただし，DDDDDDDDDDDDDDDDddddddddddddddd は disp32 の上位 31 ビットです。

［フ ラ グ］　CY　–
OV　–
S　–
Z　–
SAT　–

［説　明］　（1）現在の PC とワード長まで符号拡張した 22 ビット・ディスプレースメントを加算した値を PC に設定し，制御を移します。22 ビット・ディスプレースメントのビット 0 は 0 にマスクされます。
（2）現在の PC と 32 ビット・ディスプレースメントを加算した値を PC に設定し，制御を移します。32 ビット・ディスプレースメントのビット 0 は 0 にマスクされます。

［補　足］　計算に使用される現在の PC レジスタとは，この命令自身の先頭バイトのアドレスであるため，ディスプレースメント値が 0 の場合の分岐先は，この命令自身になります。

＜ロード命令＞

LD.B	Load byte （符号付き）バイト・データのロード

［命令形式］　LD.B　disp16 [reg1] , reg2

［オペレーション］　adr ← GR [reg1] + sign-extend (disp16)

GR [reg2] ← sign-extend (Load-memory (adr, Byte))

［フォーマット］　Format VII

［オペコード］

15	0 31	16
rrrrr111000RRRRR	ddddddddddddddddd	

［フ ラ グ］　CY　–

OV　–

S　–

Z　–

SAT　–

［説　明］　汎用レジスタ reg1 のワード・データとワード長まで符号拡張した 16 ビット・ディスプレースメントを加算して 32 ビット・アドレスを生成します。生成したアドレスからバイト・データを読み出し，ワード長まで符号拡張し，汎用レジスタ reg2 に格納します。

［補　足］　各バス（VFB, VDB, VSB, NPB, 命令キャッシュ用バス，データ・キャッシュ用バス）に接続される異なる資源に対するアクセスについては，バス・サイクルの順序が入れ替わる可能性があります（同一バスに対するアクセスでは，入れ替わることはありません）。
この命令実行中に割り込みが発生すると，実行を中止し，戻り番地をこの命令の先頭アドレスとして割り込みを処理してから，割り込み処理完了後に再実行します。

＜ロード命令＞

LD.BU	Load byte unsigned （符号なし）バイト・データのロード

［命令形式］　LD.BU　disp16 [reg1] , reg2

［オペレーション］　adr ← GR [reg1] + sign-extend (disp16)

GR [reg2] ← zero-extend (Load-memory (adr, Byte))

［フォーマット］　Format VII

［オペコード］

15	0	31	16
rrrrr11110bRRRRR		ddddddddddddddd1	

ただし，ddddddddddddddd は disp16 の上位 15 ビット，b は disp16 のビット 0 です。

［フ ラ グ］

CY　–

OV　–

S　–

Z　–

SAT　–

［説　明］　汎用レジスタ reg1 のワード・データとワード長まで符号拡張した 16 ビット・ディスプレースメントを加算して 32 ビット・アドレスを生成します。生成したアドレスからバイト・データを読み出し，ワード長までゼロ拡張し，汎用レジスタ reg2 に格納します。reg2 には r0 を指定しないでください。

［補　足］　各バス（VFB, VDB, VSB, NPB, 命令キャッシュ用バス，データ・キャッシュ用バス）に接続される異なる資源に対するアクセスについては，バス・サイクルの順序が入れ替わる可能性があります（同一バスに対するアクセスでは，入れ替わることはありません）。
この命令実行中に割り込みが発生すると，実行を中止し，戻り番地をこの命令の先頭アドレスとして割り込みを処理してから，割り込み処理完了後に再実行します。

＜ロード命令＞

LD.H	Load half-word （符号付き）ハーフワード・データのロード

［命令形式］　LD.H　disp16 [reg1] , reg2

［オペレーション］　adr ← GR [reg1] + sign-extend (disp16)

GR [reg2] ← sign-extend (Load-memory (adr, Half-word))

［フォーマット］　Format VII

［オペコード］

15	0 31	16
rrrrr111001RRRRR	ddddddddddddddd0	

ただし，ddddddddddddddd は disp16 の上位 15 ビットです。

［フ ラ グ］

CY　–

OV　–

S　–

Z　–

SAT　–

［説　明］　汎用レジスタ reg1 のワード・データとワード長まで符号拡張した 16 ビット・ディスプレースメントを加算して 32 ビット・アドレスを生成します。生成したアドレスからハーフワード・データを読み出し，ワード長まで符号拡張し，汎用レジスタ reg2 に格納します。

［注　意］　汎用レジスタ reg1 のデータとワード長まで符号拡張した 16 ビット・ディスプレースメントの加算結果は，ミス・アライン・モード設定により次の 2 種類があります。

- ビット 0 を 0 にマスクしてアドレス生成（ミス・アライン・アクセス禁止の場合）
- ビット 0 をマスクしないでアドレス生成（ミス・アライン・アクセス許可の場合）

ミス・アライン・アクセスについては，3. 3　**データ・アラインメント**を参照してください。

［補　足］　各バス（VFB, VDB, VSB, NPB, 命令キャッシュ用バス，データ・キャッシュ用バス）に接続される異なる資源に対するアクセスについては，バス・サイクルの順序が入れ替わる可能性があります（同一バスに対するアクセスでは，入れ替わることはありません）。
この命令実行中に割り込みが発生すると，実行を中止し，戻り番地をこの命令の先頭アドレスとして割り込みを処理してから，割り込み処理完了後に再実行します。

＜ロード命令＞

LD.HU	Load half-word unsigned （符号なし）ハーフワード・データのロード

［命令形式］　LD.HU　disp16 [reg1] , reg2

［オペレーション］　adr ← GR [reg1] + sign-extend (disp16)

GR [reg2] ← zero-extend (Load-memory (adr, Half-word))

［フォーマット］　Format VII

［オペコード］

15	0	31	16
rrrrr111111RRRRR		ddddddddddddddd1	

ただし，ddddddddddddddd は disp16 の上位 15 ビットです。

［フ ラ グ］　CY　–

OV　–

S　–

Z　–

SAT　–

［説　明］　汎用レジスタ reg1 のワード・データとワード長まで符号拡張した 16 ビット・ディスプレースメントを加算して 32 ビット・アドレスを生成します。生成した 32 ビット・アドレスからハーフワード・データを読み出し，ワード長までゼロ拡張し，汎用レジスタ reg2 に格納します。reg2 には r0 を指定しないでください。

［注　意］　汎用レジスタ reg1 のデータとワード長まで符号拡張した 16 ビット・ディスプレースメントの加算結果は，ミス・アライン・モード設定により次の 2 種類があります。

- ビット 0 を 0 にマスクしてアドレス生成（ミス・アライン・アクセス禁止の場合）
- ビット 0 をマスクしないでアドレス生成（ミス・アライン・アクセス許可の場合）

ミス・アライン・アクセスについては，3. 3　**データ・アラインメント**を参照してください。

［補　足］　各バス（VFB, VDB, VSB, NPB, 命令キャッシュ用バス，データ・キャッシュ用バス）に接続される異なる資源に対するアクセスについては，バス・サイクルの順序が入れ替わる可能性があります（同一バスに対するアクセスでは，入れ替わることはありません）。
この命令実行中に割り込みが発生すると，実行を中止し，戻り番地をこの命令の先頭アドレスとして割り込みを処理してから，割り込み処理完了後に再実行します。

＜ロード命令＞

LD.W	Load word ワード・データのロード

［命令形式］　LD.W　disp16 [reg1] , reg2

［オペレーション］　adr ← GR [reg1] + sign-extend (disp16)

GR [reg2] ← Load-memory (adr, Word)

［フォーマット］　Format VII

［オペコード］

15	0 31	16
rrrrr111001RRRRR	ddddddddddddddd1	

ただし，ddddddddddddddd は disp16 の上位 15 ビットです。

［フ ラ グ］　CY　–

OV　–

S　–

Z　–

SAT　–

［説　明］　汎用レジスタ reg1 のワード・データとワード長まで符号拡張した 16 ビット・ディスプレースメントを加算して 32 ビット・アドレスを生成します。生成した 32 ビット・アドレスからワード・データを読み出し，汎用レジスタ reg2 に格納します。

［注　意］　汎用レジスタ reg1 のワード・データとワード長まで符号拡張した 16 ビット・ディスプレースメントの加算結果は，ミス・アライン・モード設定により次の 2 種類があります。

- ビット 0 とビット 1 を 0 にマスクしてアドレス生成（ミス・アライン・アクセス禁止の場合）
- ビット 0 とビット 1 をマスクしないでアドレス生成（ミス・アライン・アクセス許可の場合）

ミス・アライン・アクセスについては，3. 3　**データ・アラインメント**を参照してください。

［補　足］　各バス（VFB, VDB, VSB, NPB, 命令キャッシュ用バス，データ・キャッシュ用バス）に接続される異なる資源に対するアクセスについては，バス・サイクルの順序が入れ替わる可能性があります（同一バスに対するアクセスでは，入れ替わることはありません）。
この命令実行中に割り込みが発生すると，実行を中止し，戻り番地をこの命令の先頭アドレスとして割り込みを処理してから，割り込み処理完了後に再実行します。

＜特殊命令＞

LDSR	Load to system register システム・レジスタへのロード

［命令形式］　LDSR　reg2, regID

［オペレーション］　SR [regID] ← GR [reg2]

［フォーマット］　Format IX

［オペコード］

15	0	31	16
rrrrr111111RRRRR		0000000000100000	

注意　この命令では，ニモニック記述の都合上，ソース・レジスタを reg2 としていますが，オペコード上は reg1 のフィールドを使用しています。したがって，ニモニック記述とオペコードにおいて，レジスタ指定の意味付けがほかの命令と異なります。

rrrrr：regID 指定
RRRRR：reg2 指定

［フ ラ グ］
CY　－（補足参照）
OV　－（補足参照）
S　－（補足参照）
Z　－（補足参照）
SAT　－（補足参照）

［説　明］　汎用レジスタ reg2 のワード・データをシステム・レジスタ番号（regID）で指定されるシステム・レジスタに設定します。汎用レジスタ reg2 は影響を受けません。

［補　足］　システム・レジスタ番号（regID）が 5（PSW）の場合は，PSW レジスタの各フラグには汎用レジスタ reg2 の対応するビットの値が設定されます。PSW レジスタへの書き込み時のみ，割り込みのサンプリングをしません。この命令によって PSW レジスタの ID フラグをセット（1）する場合，ID フラグが有効になるのは次の命令からですが，割り込みのサンプリングを PSW レジスタへの書き込み時には行わないため，実際はこの命令実行中から割り込みを禁止します。

［注　意］　システム・レジスタ番号は，システム・レジスタを一意に識別するための番号です。予約されているシステム・レジスタ，書き込み禁止のシステム・レジスタに対してこの命令を実行した場合の動作は保証しません。

＜加算付き乗算命令＞

MAC	Multiply word and add （符号付き）ワード・データの加算付き乗算

［命令形式］　MAC　reg1, reg2, reg3, reg4

［オペレーション］　GR [reg4+1] || GR [reg4] ← GR [reg2] × GR [reg1] + GR [reg3+1] || GR [reg3]

［フォーマット］　Format XI

［オペコード］

15	0	31	16
rrrrr111111RRRRR		wwww0011110mmmm0	

［フ ラ グ］

CY　–
OV　–
S　–
Z　–
SAT　–

［説　明］　汎用レジスタ reg2 のワード・データに汎用レジスタ reg1 のワード・データを乗算した結果（64 ビット・データ）と，汎用レジスタ reg3 を下位 32 ビットとして，汎用レジスタ reg3+1（たとえば，reg3 が r6 の場合，「reg3+1」は r7 となります）を上位 32 ビットとして結合した 64 ビット・データを加算し，その結果（64 ビット・データ）の上位 32 ビットを汎用レジスタ reg4+1 に，下位 32 ビットを汎用レジスタ reg4 に格納します。
汎用レジスタ reg1, reg2 の内容を 32 ビットの符号付き整数として扱います。
汎用レジスタ reg1, reg2, reg3, reg3+1 は影響を受けません。

［注　意］　reg3，または reg4 に指定できる汎用レジスタは，偶数番号の付いたレジスタ（r0, r2, r4, …, r30）だけです。奇数番号の付いたレジスタ（r1, r3, …, r31）を指定した場合の結果は不定です。

＜加算付き乗算命令＞

MACU	Multiply word unsigned and add （符号なし）ワード・データの加算付き乗算

［命令形式］　MACU　reg1, reg2, reg3, reg4

［オペレーション］　GR [reg4+1] || GR [reg4] ← GR [reg2] × GR [reg1] + GR [reg3+1] || GR [reg3]

［フォーマット］　Format XI

［オペコード］

15	0	31	16
rrrrr111111RRRRR		wwww0011111mmmm0	

［フ ラ グ］

CY　–
OV　–
S　–
Z　–
SAT　–

［説　明］　汎用レジスタ reg2 のワード・データに汎用レジスタ reg1 のワード・データを乗算した結果（64 ビット・データ）と，汎用レジスタ reg3 を下位 32 ビットとして，汎用レジスタ reg3+1（たとえば，reg3 が r6 の場合，「reg3+1」は r7 となります）を上位 32 ビットとして結合した 64 ビット・データを加算し，その結果（64 ビット・データ）の上位 32 ビットを汎用レジスタ reg4+1 に，下位 32 ビットを汎用レジスタ reg4 に格納します。
汎用レジスタ reg1, reg2 の内容を 32 ビットの符号なし整数として扱います。
汎用レジスタ reg1, reg2, reg3, reg3+1 は影響を受けません。

［注　意］　reg3，または reg4 に指定できる汎用レジスタは，偶数番号の付いたレジスタ（r0, r2, r4, …, r30）だけです。奇数番号の付いたレジスタ（r1, r3, …, r31）を指定した場合の結果は不定です。

＜算術演算命令＞

MOV

Move register/immediate (5-bit) /immediate (32-bit)

データの転送

[命令形式]　(1) MOV　reg1, reg2

(2) MOV　imm5, reg2

(3) MOV　imm32, reg1

[オペレーション]　(1) GR [reg2] ← GR [reg1]

(2) GR [reg2] ← sign-extend (imm5)

(3) GR [reg1] ← imm32

[フォーマット]　(1) Format I

(2) Format II

(3) Format VI

[オペコード]

(1)

15	0
rrrrr000000RRRRR	

(2)

15	0
rrrrr010000iiiii	

(3)

15	0	31	16	47	32
00000110001RRRRR		iiiiiiiiiiiiiiii		IIIIIIIIIIIIIIII	

i（ビット 31-16）は 32 ビット・イミーディエト・データの下位 16 ビットです。

I（ビット 47-32）は 32 ビット・イミーディエト・データの上位 16 ビットです。

[フ ラ グ]　CY　–

OV　–

S　–

Z　–

SAT　–

[説　明]　(1) 汎用レジスタ reg1 のワード・データを，汎用レジスタ reg2 にコピーし転送します。
汎用レジスタ reg1 は影響を受けません。

(2) 5 ビット・イミーディエトをワード長まで符号拡張した値を，汎用レジスタ reg2 にコピーし転送します。
なお，reg2 には r0 を指定しないでください。

(3) 32 ビット・イミーディエトを，汎用レジスタ reg1 にコピーし転送します。

＜算術演算命令＞

MOVEA	Move effective address 実効アドレスの転送

［命令形式］　MOVEA　imm16, reg1, reg2

［オペレーション］　GR [reg2] ← GR [reg1] + sign-extend (imm16)

［フォーマット］　Format VI

［オペコード］

15	0	31	16
rrrrr110001RRRRR		iiiiiiiiiiiiiiii	

［フ ラ グ］

CY　–
OV　–
S　–
Z　–
SAT　–

［説　明］　汎用レジスタ reg1 のワード・データにワード長まで符号拡張した 16 ビット・イミーディエトを加算し，その結果を汎用レジスタ reg2 に格納します。汎用レジスタ reg1 は影響を受けません。加算によってもフラグは変化しません。
なお，reg2 には r0 を指定しないでください。

［補　足］　32 ビット・アドレスを計算する際，フラグを変化させたくない場合に，この命令を使用します。

＜算術演算命令＞

MOVHI	Move high half-word 上位ハーフワードの転送

［命令形式］　MOVHI　imm16, reg1, reg2

［オペレーション］　GR [reg2] ← GR [reg1] + (imm16 || 0^{16})

［フォーマット］　Format VI

［オペコード］

15	0	31	16
rrrrr110010RRRRR		iiiiiiiiiiiiiiii	

［フ ラ グ］

CY　–
OV　–
S　–
Z　–
SAT　–

［説　明］　汎用レジスタ reg1 のワード・データに，上位 16 ビットが 16 ビット・イミーディエト，下位 16 ビットが 0 であるワード・データを加算し，その結果を汎用レジスタ reg2 に格納します。
汎用レジスタ reg1 は影響を受けません。加算によってもフラグは変化しません。
なお，reg2 には r0 を指定しないでください。

［補　足］　32 ビット・アドレスの上位 16 ビットの生成にこの命令を使用します。

＜乗算命令＞

MUL	Multiply word by register/immediate (9-bit) （符号付き）ワード・データの乗算

［命令形式］　（1）MUL　reg1, reg2, reg3

（2）MUL　imm9, reg2, reg3

［オペレーション］　（1）GR [reg3] || GR [reg2] ← GR [reg2] × GR [reg1]

（2）GR [reg3] || GR [reg2] ← GR [reg2] × sign-extend (imm9)

［フォーマット］　（1）Format XI

（2）Format XII

［オペコード］

	15	0	31	16
(1)	rrrrr111111RRRRR		wwwww01000100000	

	15	0	31	16
(2)	rrrrr111111iiiii		wwwww01001IIII00	

iiiii は，9 ビット・イミーディエト・データの下位 5 ビットです。

IIII は，9 ビット・イミーディエト・データの上位 4 ビットです。

［フ ラ グ］　CY　–

OV　–

S　–

Z　–

SAT　–

［説　明］　（1）汎用レジスタ reg2 のワード・データに汎用レジスタ reg1 のワード・データを乗算し，その結果（64 ビット・データ）の上位 32 ビットを汎用レジスタ reg3 に，下位 32 ビットを汎用レジスタ reg2 に格納します。

reg1, reg2 の内容を 32 ビットの符号付き整数として扱います。汎用レジスタ reg1 は影響を受けません。

（2）汎用レジスタ reg2 のワード・データにワード長まで符号拡張した 9 ビット・イミーディエト・データを乗算し，その結果（64 ビット・データ）の上位 32 ビットを汎用レジスタ reg3 に，下位 32 ビットを汎用レジスタ reg2 に格納します。

reg2 の内容を 32 ビットの符号付き整数として扱います。

［注　意］　「MUL reg1, reg2, reg3」命令において，次の条件をすべて満たすレジスタの組み合わせは行わないでください。この条件に当てはまる命令を実行した場合の動作は保証しません。

- reg1 = reg3
- reg1 ≠ reg2
- reg1 ≠ r0
- reg3 ≠ r0

［補　足］　汎用レジスタ reg2 と汎用レジスタ reg3 が同じレジスタの場合，そのレジスタには乗算結果の上位 32 ビットが格納されます。

＜乗算命令＞

MULH	Multiply half-word by register/immediate (5-bit) （符号付き）ハーフワード・データの乗算

［命令形式］　（1）MULH　reg1, reg2
（2）MULH　imm5, reg2

［オペレーション］　（1）GR [reg2] (32) ← GR [reg2] (16) × GR [reg1] (16)
（2）GR [reg2] ← GR [reg2] × sign-extend (imm5)

［フォーマット］　（1）Format I
（2）Format II

［オペコード］

	15	0
(1)	rrrrr000111RRRRR	

	15	0
(2)	rrrrr010111iiiii	

［フ ラ グ］　CY　–
OV　–
S　–
Z　–
SAT　–

［説　明］　（1）汎用レジスタ reg2 の下位ハーフワード・データに汎用レジスタ reg1 の下位ハーフワード・データを乗算し，その結果を汎用レジスタ reg2 に格納します。
汎用レジスタ reg1 は影響を受けません。
なお，reg2 には r0 を指定しないでください。
（2）汎用レジスタ reg2 の下位ハーフワード・データにハーフワード長まで符号拡張した 5 ビット・イミーディエトを乗算し，その結果を汎用レジスタ reg2 に格納します。
なお，reg2 には r0 を指定しないでください。

［補　足］　乗数，被乗数の場合，汎用レジスタ reg1, reg2 の上位 16 ビットを無視します。

＜乗算命令＞

MULHI	Multiply half-word by immediate (16-bit) （符号付き）ハーフワード・イミーディエトの乗算

［命令形式］　MULHI　imm16, reg1, reg2

［オペレーション］　GR [reg2] ← GR [reg1] × imm16

［フォーマット］　Format VI

［オペコード］

15	0	31	16
rrrrr110111RRRRR		iiiiiiiiiiiiiiii	

［フ ラ グ］

CY　–
OV　–
S　–
Z　–
SAT　–

［説　明］　汎用レジスタ reg1 の下位ハーフワード・データに，16 ビット・イミーディエトを乗算し，その結果を汎用レジスタ reg2 に格納します。汎用レジスタ reg1 は影響を受けません。
なお，reg2 には r0 を指定しないでください。

［補　足］　被乗数の場合，汎用レジスタ reg1 の上位 16 ビットを無視します。

＜乗算命令＞

MULU

Multiply word unsigned by register/immediate (9-bit)

（符号なし）ワード・データの乗算

［命令形式］　（1）MULU　reg1, reg2, reg3

（2）MULU　imm9, reg2, reg3

［オペレーション］　（1）GR [reg3] || GR [reg2] ← GR [reg2] × GR [reg1]

（2）GR [reg3] || GR [reg2] ← GR [reg2] × zero-extend (imm9)

［フォーマット］　（1）Format XI

（2）Format XII

［オペコード］

	15	0	31	16
(1)	rrrrr111111RRRRR		wwwww01000100010	

	15	0	31	16
(2)	rrrrr111111iiiii		wwwww01001IIII10	

iiiii は，9 ビット・イミーディエト・データの下位 5 ビットです。

IIII は，9 ビット・イミーディエト・データの上位 4 ビットです。

［フ ラ グ］

CY	–
OV	–
S	–
Z	–
SAT	–

［説　明］　（1）汎用レジスタ reg2 のワード・データに汎用レジスタ reg1 のワード・データを乗算し，その結果（64 ビット・データ）の上位 32 ビットを汎用レジスタ reg3 に，下位 32 ビットを汎用レジスタ reg2 に格納します。

reg1, reg2 の内容を 32 ビットの符号なし整数として扱います。

汎用レジスタ reg1 は影響を受けません。

（2）汎用レジスタ reg2 のワード・データにワード長までゼロ拡張した 9 ビット・イミーディエト・データを乗算し，その結果（64 ビット・データ）の上位 32 ビットを汎用レジスタ reg3 に，下位 32 ビットを汎用レジスタ reg2 に格納します。

reg2 の内容を 32 ビットの符号なし整数として扱います。

［注　意］　「MULU reg1, reg2, reg3」命令において，次の条件をすべて満たすレジスタの組み合わせは行わないでください。この条件に当てはまる命令を実行した場合の動作は保証しません。

- reg1 = reg3
- reg1 ≠ reg2
- reg1 ≠ r0
- reg3 ≠ r0

［補　足］　汎用レジスタ reg2 と汎用レジスタ reg3 が同じレジスタの場合，そのレジスタには乗算結果の上位 32 ビットが格納されます。

＜特殊命令＞

NOP	No operation オペレーションなし

［命令形式］　NOP

［オペレーション］　何もせず最低 1 クロック費やします。

［フォーマット］　Format I

［オペコード］

15	0
0000000000000000	

［フ ラ グ］

CY	–
OV	–
S	–
Z	–
SAT	–

［説　明］　何もせず最低 1 クロック費やします。

［補　足］　PC は +2 されます。また，オペコードは「MOV　r0, r0」と同一になります。

＜論理演算命令＞

NOT

NOT

論理否定（1 の補数をとる）

［命令形式］　NOT　reg1, reg2

［オペレーション］　GR [reg2] ← NOT (GR [reg1])

［フォーマット］　Format I

［オペコード］

15	0
rrrrr000001RRRRR	

［フ ラ グ］

CY　–

OV　0

S　演算結果のワード・データの MSB が 1 のとき 1，そうでないとき 0

Z　演算結果が 0 のとき 1，そうでないとき 0

SAT　–

［説　明］　汎用レジスタ reg1 のワード・データの論理否定（1 の補数）をとり，その結果を汎用レジスタ reg2 に格納します。汎用レジスタ reg1 は影響を受けません。

＜ビット操作命令＞

NOT1	NOT bit ビット・ノット

［命令形式］　（1）NOT1　bit#3, disp16 [reg1]

（2）NOT1　reg2, [reg1]

［オペレーション］　（1）adr ← GR [reg1] + sign-extend (disp16)

Z フラグ ← Not (Load-memory-bit (adr, bit#3))

Store-memory-bit (adr, bit#3, Z フラグ)

（2）adr ← GR [reg1]

Z フラグ ← Not (Load-memory-bit (adr, reg2))

Store-memory-bit (adr, reg2, Z フラグ)

［フォーマット］　（1）Format VIII

（2）Format IX

［オペコード］

	15	0	31	16
(1)	01bbb111110RRRRR		ddddddddddddddddd	
(2)	rrrrr111111RRRRR		0000000011100010	

［フ ラ グ］　CY　–

OV　–

S　–

Z　指定したビットが 0 のとき 1，指定したビットが 1 のとき 0

SAT　–

［説　明］　（1）まず，汎用レジスタ reg1 のワード・データと，ワード長まで符号拡張した 16 ビット・ディスプレースメントを加算して 32 ビット・アドレスを生成します。生成したアドレスのバイト・データを読み出し，3 ビットのビット・ナンバで指定されるビットを反転（0→1, 1→0）し，元のアドレスに書き戻します。

（2）まず，汎用レジスタ reg1 のワード・データを読み出して 32 ビット・アドレスを生成します。生成したアドレスのバイト・データを読み出し，汎用レジスタ reg2 の下位 3 ビットで指定されるビットを反転（0→1, 1→0）し，元のアドレスに書き戻します。

［補　足］　PSW レジスタの Z フラグはこの命令を実行する前に該当ビットが 0 か 1 だったかを示します。この命令実行後の該当ビットの内容を示すものではありません。

＜論理演算命令＞

OR	OR
	論理和

［命令形式］　OR　reg1, reg2

［オペレーション］　GR [reg2] ← GR [reg2] OR GR [reg1]

［フォーマット］　Format I

［オペコード］

15	0
rrrrr001000RRRRR	

［フ ラ グ］

CY　–

OV　0

S　演算結果のワード・データの MSB が 1 のとき 1，そうでないとき 0

Z　演算結果が 0 のとき 1，そうでないとき 0

SAT　–

［説　明］　汎用レジスタ reg2 のワード・データと汎用レジスタ reg1 のワード・データの論理和をとり，その結果を汎用レジスタ reg2 に格納します。汎用レジスタ reg1 は影響を受けません。

＜論理演算命令＞

ORI	OR immediate (16-bit) 論理和

［命令形式］　ORI　imm16, reg1, reg2

［オペレーション］　GR [reg2] ← GR [reg1] OR zero – extend (imm16)

［フォーマット］　Format VI

［オペコード］

15	0	31	16
rrrrr110100RRRRR		iiiiiiiiiiiiiiii	

［フ ラ グ］

CY　–

OV　0

S　演算結果のワード・データの MSB が 1 のとき 1，そうでないとき 0

Z　演算結果が 0 のとき 1，そうでないとき 0

SAT　–

［説　明］　汎用レジスタ reg1 のワード・データと 16 ビット・イミーディエトをワード長までゼロ拡張した値の論理和をとり，その結果を汎用レジスタ reg2 に格納します。汎用レジスタ reg1 は影響を受けません。

＜特殊命令＞

PREPARE	Function prepare スタック・フレームの生成

［命令形式］ （1）PREPARE list12, imm5

（2）PREPARE list12, imm5, sp/imm 注

注 sp/imm の値は，サブオペコードのビット 19, ビット 20 で指定します。

［オペレーション］ （1）Store-memory (sp – 4, GR [reg in list12], Word) sp ← sp – 4

repeat 1 step above until all regs in list12 is stored

sp ← sp – zero-extend (imm5)

（2）Store-memory (sp – 4, GR [reg in list12], Word) sp ← sp – 4

repeat 1 step above until all regs in list12 is stored

sp ← sp – zero-extend (imm5)

ep ← sp/imm

［フォーマット］ Format XIII

［オペコード］

	15	0	31	16	
(1)	0000011110iiiiiL		LLLLLLLLLLL00001		
(2)	0000011110iiiiiL		LLLLLLLLLLff011	imm16 / imm32	

32 ビット・イミーディエト・データ（imm32）の場合，ビット 47-32 が imm32 の下位 16 ビット，ビット 63-48 が imm32 の上位 16 ビットです。

ff=00：sp を ep にロード

ff=01：符号拡張した 16 ビット・イミーディエト・データ（ビット 47-32）を ep にロード

ff=10：16 ビット論理左シフトした 16 ビット・イミーディエト・データ（ビット 47-32）を ep にロード

ff=11：32 ビット・イミーディエト・データ（ビット 63-32）を ep にロード

また，LLLLLLLLLLL は，レジスタ・リスト「list12」の中の対応するビットの値を示します（たとえば，オペコード中のビット 21 の「L」は list12 のビット 21 の値を示します）。

list12 は，次のように定義される 32 ビットのレジスタ・リストです。

31	30	29	28	27	26	25	24	23	22	21	20 … 1	0
r24	r25	r26	r27	r20	r21	r22	r23	r28	r29	r31	–	r30

ビット 31-21 とビット 0 の各ビットに汎用レジスタ（r20-r31）が対応しており，セット（1）されたビットに対応するレジスタが操作の対象として指定されます。たとえば，r20, r30 を指定する場合，list12 の値は次のようになります（レジスタが対応付けられていないビット 20-1 への設定値は任意です）。

- レジスタが対応付けられていないビットの値をすべて 0 とした場合：08000001H
- レジスタが対応付けられていないビットの値をすべて 1 とした場合：081FFFFFH

［フ ラ グ］

CY –
OV –
S –
Z –
SAT –

［説　明］

（1）list12 で指定されている汎用レジスタを退避（sp から 4 を減算し，データをそのアドレスに格納）します。次に，2 ビット論理左シフトしワード長までゼロ拡張した 5 ビット・イミーディエトを sp から減算します。

（2）list12 で指定されている汎用レジスタを退避（sp から 4 を減算し，データをそのアドレスに格納）します。次に，2 ビット論理左シフトしワード長までゼロ拡張した 5 ビット・イミーディエトを sp から減算します。
続いて，第 3 オペランド（sp/imm）で指定されるデータを ep にロードします。

［補　足］

list12 の汎用レジスタは，昇順に格納されます（r20, r21, ..., r31）。
imm5 は，自動変数と一時データ用のスタック・フレームを作るために使用されます。
sp で指定された下位 2 ビットのアドレスは，ミス・アライン・アクセスが可能でも，0 にマスクされます。
また，sp の更新前に割り込みが発生すると，実行を中止し，戻り番地をこの命令の先頭アドレスとして割り込みを処理してから，割り込み処理完了後に再実行します（sp と ep は割り込み実行開始前の元の値を保持します）。

［注　意］

命令実行中に割り込みが発生すると，スタック操作を行うため，リード／ライト・サイクルとレジスタ値の書き換えが終了したあとに中止する場合があります。

＜特殊命令＞

RETI	Return from trap or interrupt ソフトウエア例外または割り込みルーチンからの復帰

［命令形式］　RETI

［オペレーション］　if PSW.EP = 1
then PC ← EIPC
　　PSW ← EIPSW
else if PSW.NP = 1
　　then PC ← FEPC
　　　　PSW ← FEPSW
　　else PC ← EIPC
　　　　PSW ← EIPSW

［フォーマット］　Format X

［オペコード］

15	0	31	16
0000011111100000		0000000101000000	

［フ ラ グ］
CY　FEPSW レジスタまたは EIPSW レジスタから読み出した値が設定される
OV　FEPSW レジスタまたは EIPSW レジスタから読み出した値が設定される
S　FEPSW レジスタまたは EIPSW レジスタから読み出した値が設定される
Z　FEPSW レジスタまたは EIPSW レジスタから読み出した値が設定される
SAT　FEPSW レジスタまたは EIPSW レジスタから読み出した値が設定される

［説　明］　システム・レジスタから，復帰 PC と PSW レジスタを取り出し，ソフトウエア例外または割り込みルーチンから復帰する命令です。この命令の動作は次のとおりです。

<1> EP フラグが 1 の場合，NP フラグの状態にかかわらず，EIPC, EIPSW レジスタから復帰 PC，PSW を取り出します。
EP フラグが 0 かつ NP フラグが 1 の場合，FEPC，FEPSW レジスタから復帰 PC, PSW を取り出します。
EP フラグが 0 かつ NP フラグが 0 の場合，EIPC，EIPSW レジスタから復帰 PC, PSW を取り出します。
<2> 取り出した復帰 PC と PSW を PC，PSW レジスタに設定し，制御を移します。

［注　意］　ノンマスカブル割り込み処理またはソフトウエア例外処理からの RETI 命令による復帰時は，PC, PSW レジスタを正常にリストアするために，RETI 命令の直前で NP フラグ，EP フラグを次の状態にしておく必要があります。

- RETI 命令によるノンマスカブル割り込み処理からの復帰時　：NP = 1 かつ EP = 0
- RETI 命令によるソフトウエア例外処理からの復帰時　　：EP = 1

プログラムによる設定には LDSR 命令を使用します。
割り込みコントローラの動作に関連して，この命令の後半の ID ステージでは割り込みを受け付けません。

＜データ操作命令＞

SAR	Shift arithmetic right by register/immediate (5-bit) 算術右シフト

［命令形式］　（1）SAR　reg1, reg2
（2）SAR　imm5, reg2
（3）SAR　reg1, reg2, reg3

［オペレーション］　（1）GR [reg2] ← GR [reg2] arithmetically shift right by GR [reg1]
（2）GR [reg2] ← GR [reg2] arithmetically shift right by zero-extend
（3）GR [reg3] ← GR [reg2] arithmetically shift right by GR [reg1]

［フォーマット］　（1）Format IX
（2）Format II
（3）Format XI

［オペコード］

(1)

15	0	31	16
rrrrr111111RRRRR		0000000010100000	

(2)

15	0
rrrrr010101iiiii	

(3)

15	0	31	16
rrrrr111111RRRRR		wwwww00010100010	

［フ ラ グ］　CY　最後にシフト・アウトしたビットが 1 のとき 1，そうでないとき 0，ただしシフト数が 0 のときは 0
OV　0
S　演算結果が負のとき 1，そうでないとき 0
Z　演算結果が 0 のとき 1，そうでないとき 0
SAT　－

［説　明］　（1）汎用レジスタ reg2 のワード・データを汎用レジスタ reg1 の下位 5 ビットで示されるシフト数分，0 から+31 までを算術右シフトし（シフト以前の MSB の値を，シフトを実行したあとの MSB にコピーする），汎用レジスタ reg2 に書き込みます。シフト数が 0 のときは，汎用レジスタ reg2 は命令実行前の値を保持します。汎用レジスタ reg1 は影響を受けません。
（2）汎用レジスタ reg2 のワード・データを，ワード長までゼロ拡張した 5 ビット・イミーディエトで示されるシフト数分，0 から+31 までを算術右シフトし（シフト以前の MSB の値を，シフトを実行したあとの MSB にコピーする），汎用レジスタ reg2 に書き込みます。シフト数が 0 のときは，汎用レジスタ reg2 は命令実行前の値を保持します。

(3) 汎用レジスタ reg2 のワード・データを汎用レジスタ reg1 の下位 5 ビットで示されるシフト数分，0 から+31 までを算術右シフトし（シフト以前の MSB の値を，シフトを実行したあとの MSB にコピーする），汎用レジスタ reg3 に書き込みます。シフト数が 0 のときは，汎用レジスタ reg3 は命令実行前の値を保持します。汎用レジスタ reg1, reg2 は影響を受けません。

＜データ操作命令＞

SASF	Shift and set flag condition シフトとフラグ条件の設定

［命令形式］　SASF　cccc, reg2

［オペレーション］　if conditions are satisfied
then GR [reg2] ← (GR [reg2] Logically shift left by 1) OR 00000001H
else GR [reg2] ← (GR [reg2] Logically shift left by 1) OR 00000000H

［フォーマット］　Format IX

［オペコード］

15	0	31	16
rrrrr1111110cccc		0000001000000000	

［フ ラ グ］
CY　－
OV　－
S　－
Z　－
SAT　－

［説　明］　条件コード「cccc」で指定された条件が満たされた場合は，汎用レジスタ reg2 のデータを 1 ビット論理左シフトし，LSB がセット（1）されます。満たされなかった場合は，汎用レジスタ reg2 のデータを 1 ビット論理左シフトし，LSB がクリア（0）されます。
次の表で示されている条件コードのうちの 1 つを「cccc」として指定してください。

条件コード	条件式	条件コード	条件式
0000	OV = 1	0100	S = 1
1000	OV = 0	1100	S = 0
0001	CY = 1	0101	always（無条件）
1001	CY = 0	1101	SAT = 1
0010	Z = 1	0110	(S xor OV) = 1
1010	Z = 0	1110	(S xor OV) = 0
0011	(CY or Z) = 1	0111	((S xor OV) or Z) = 1
1011	(CY or Z) = 0	1111	((S xor OV) or Z) = 0

［補　足］　SETF 命令を参照してください。

＜飽和演算命令＞

SATADD	Saturated add register/immediate (5-bit) **飽和加算**

［命令形式］　（1）SATADD　reg1, reg2
（2）SATADD　imm5, reg2
（3）SATADD　reg1, reg2, reg3

［オペレーション］　（1）GR [reg2] ← saturated (GR [reg2] + GR [reg1])
（2）GR [reg2] ← saturated (GR [reg2] + sigh-extend (imm5))
（3）GR [reg3] ← saturated (GR [reg2] + GR [reg1])

［フォーマット］　（1）Format I
（2）Format II
（3）Format XI

［オペコード］

(1)

15	0
rrrrr000110RRRRR	

(2)

15	0
rrrrr010001iiiii	

(3)

15	0	31	16
rrrrr111111RRRRR		wwwww01110111010	

［フ ラ グ］　CY　MSB **からのキャリーがあれば** 1**，そうでないとき** 0
OV　**オーバフローが起こったとき** 1**，そうでないとき** 0
S　**飽和演算結果が負のとき** 1**，そうでないとき** 0
Z　**飽和演算結果が** 0 **のとき** 1**，そうでないとき** 0
SAT　OV = 1 **であるとき** 1**，そうでないとき変化しない**

［説　明］　（1）**汎用レジスタ** reg2 **のワード・データに汎用レジスタ** reg1 **のワード・データを加算し，その結果を汎用レジスタ** reg2 **に格納します。ただし，結果が正の最大値** 7FFFFFFFH **を越えたときは** 7FFFFFFFH **を，負の最大値** 80000000H **を越えたときは** 80000000H **を** reg2 **に格納し，**SAT **フラグをセット（**1**）します。汎用レジスタ** reg1 **は影響を受けません。**
なお，reg2 **には** r0 **を指定しないでください。**
（2）**汎用レジスタ** reg2 **のワード・データにワード長まで符号拡張した** 5 **ビット・イミーディエトを加算し，その結果を汎用レジスタ** reg2 **に格納します。ただし，結果が正の最大値** 7FFFFFFFH **を越えたときは** 7FFFFFFFH **を，負の最大値** 80000000H **を越えたときは** 80000000H **を** reg2 **に格納し，**SAT **フラグをセット（**1**）します。**
なお，reg2 **には** r0 **を指定しないでください。**

(3) 汎用レジスタ reg2 のワード・データに汎用レジスタ reg1 のワード・データを加算し，その結果を汎用レジスタ reg3 に格納します。ただし，結果が正の最大値 7FFFFFFFH を越えたときは 7FFFFFFFH を，負の最大値 80000000H を越えたときは 80000000H を reg3 に格納し，SAT フラグをセット（1）します。汎用レジスタ reg1 と reg2 は影響を受けません。

［補　足］　SAT フラグは累積フラグであり，飽和演算命令で演算結果が飽和するとセット（1）され，以降の命令の演算結果が飽和しなくてもクリア（0）されません。
SAT フラグがセット（1）されていても，飽和演算命令は正常に実行します。

［注　意］　SAT フラグをクリア（0）するときは，LDSR 命令によって PSW レジスタにデータをロードしてください。

＜飽和演算命令＞

SATSUB	Saturated subtract 飽和減算

［命令形式］　（1）SATSUB　reg1, reg2
（2）SATSUB　reg1, reg2, reg3

［オペレーション］　（1）GR [reg2] ← saturated (GR [reg2] – GR [reg1])
（2）GR [reg3] ← saturated (GR [reg2] – GR [reg1])

［フォーマット］　（1）Format I
（2）Format XI

［オペコード］

	15	0
(1)	rrrrr000101RRRRR	

	15	0	31	16
(2)	rrrrr111111RRRRR		wwwww01110011010	

［フ ラ グ］　CY　MSB へのボローがあれば 1，そうでないとき 0
OV　オーバフローが起こったとき 1，そうでないとき 0
S　飽和演算結果が負のとき 1，そうでないとき 0
Z　飽和演算結果が 0 のとき 1，そうでないとき 0
SAT　OV = 1 であるとき 1，そうでないとき変化しない

［説　明］　（1）汎用レジスタ reg2 のワード・データから汎用レジスタ reg1 のワード・データを減算し，その結果を汎用レジスタ reg2 に格納します。ただし，結果が正の最大値 7FFFFFFFH を越えたときは 7FFFFFFFH を，負の最大値 80000000H を越えたときは 80000000H を reg2 に格納し，SAT フラグをセット（1）します。汎用レジスタ reg1 は影響を受けません。
なお，reg2 には r0 を指定しないでください。
（2）汎用レジスタ reg2 のワード・データから汎用レジスタ reg1 のワード・データを減算し，その結果を汎用レジスタ reg3 に格納します。ただし，結果が正の最大値 7FFFFFFFH を越えたときは 7FFFFFFFH を，負の最大値 80000000H を越えたときは 80000000H を reg3 に格納し，SAT フラグをセット（1）します。汎用レジスタ reg1 と reg2 は影響を受けません。

［補　足］　SAT フラグは累積フラグであり，飽和演算命令で演算結果が飽和するとセット（1）され，以降の命令の演算結果が飽和しなくてもクリア（0）されません。
SAT フラグがセット（1）されていても，飽和演算命令は正常に実行します。

［注　意］　SAT フラグをクリア（0）するときは，LDSR 命令によって PSW レジスタにデータをロードしてください。

＜飽和演算命令＞

SATSUBI	Saturated subtract immediate 飽和減算

［命令形式］　SATSUBI　imm16, reg1, reg2

［オペレーション］　GR [reg2] ← saturated (GR [reg1] – sign-extend (imm16))

［フォーマット］　Format VI

［オペコード］

15	0	31	16
rrrrr110011RRRRR		iiiiiiiiiiiiiiii	

［フ ラ グ］　CY　MSB へのボローがあれば 1，そうでないとき 0
OV　オーバフローが起こったとき 1，そうでないとき 0
S　飽和演算結果が負のとき 1，そうでないとき 0
Z　飽和演算結果が 0 のとき 1，そうでないとき 0
SAT　OV = 1 であるとき 1，そうでないとき変化しない

［説　明］　汎用レジスタ reg1 のワード・データからワード長まで符号拡張した 16 ビット・イミーディエトを減算し，その結果を汎用レジスタ reg2 に格納します。ただし，結果が正の最大値 7FFFFFFFH を越えたときは 7FFFFFFFH を，負の最大値 80000000H を越えたときは 80000000H を reg2 に格納し，SAT フラグをセット（1）します。汎用レジスタ reg1 は影響を受けません。
なお，reg2 には r0 を指定しないでください。

［補　足］　SAT フラグは累積フラグであり，飽和演算命令で演算結果が飽和するとセット（1）され，以降の命令の演算結果が飽和しなくてもクリア（0）されません。
SAT フラグがセット（1）されていても，飽和演算命令は正常に実行します。

［注　意］　SAT フラグをクリア（0）するときは，LDSR 命令によって PSW レジスタにデータをロードしてください。

＜飽和演算命令＞

SATSUBR	Saturated subtract reverse 飽和逆減算

［命令形式］　SATSUBR　reg1, reg2

［オペレーション］　GR [reg2] ← saturated (GR [reg1] – GR [reg2])

［フォーマット］　Format I

［オペコード］

15	0
rrrrr000100RRRRR	

［フ ラ グ］

CY　MSB へのボローがあれば 1，そうでないとき 0

OV　オーバフローが起こったとき 1，そうでないとき 0

S　飽和演算結果が負のとき 1，そうでないとき 0

Z　飽和演算結果が 0 のとき 1，そうでないとき 0

SAT　OV = 1 であるとき 1，そうでないとき変化しない

［説　明］　汎用レジスタ reg1 のワード・データから汎用レジスタ reg2 のワード・データを減算し，その結果を汎用レジスタ reg2 に格納します。ただし，結果が正の最大値 7FFFFFFFH を越えたときは 7FFFFFFFH を，負の最大値 80000000H を越えたときは 80000000H を reg2 に格納し，SAT フラグをセット（1）します。汎用レジスタ reg1 は影響を受けません。
なお，reg2 には r0 を指定しないでください。

［補　足］　SAT フラグは累積フラグであり，飽和演算命令で演算結果が飽和するとセット（1）され，以降の命令の演算結果が飽和しなくてもクリア（0）されません。
SAT フラグがセット（1）されていても，飽和演算命令は正常に実行します。

［注　意］　SAT フラグをクリア（0）するときは，LDSR 命令によって PSW レジスタにデータをロードしてください。

＜条件付き演算命令＞

SBF	Subtract on condition flag 条件付き減算

［命令形式］　SBF　cccc, reg1, reg2, reg3

［オペレーション］　if conditions are satisfied

then GR [reg3] ← GR [reg2] – GR [reg1] –1

else GR [reg3] ← GR [reg2] – GR [reg1] –0

［フォーマット］　Format XI

［オペコード］

15	0 31	16
rrrrr111111RRRRR	wwwww011100cccc0	

［フ ラ グ］

CY　MSB へのボローがあれば 1，そうでないとき 0

OV　オーバフローが起こったとき 1，そうでないとき 0

S　演算結果が負のとき 1，そうでないとき 0

Z　演算結果が 0 のとき 1，そうでないとき 0

SAT　–

［説　明］　汎用レジスタ reg2 のワード・データから汎用レジスタ reg1 のワード・データを減算した結果から 1（減算結果が条件コード「cccc」で指定された条件に満たされた場合）または 0（減算結果が条件コード「cccc」で指定された条件に満たされなかった場合）を減算し，その結果を汎用レジスタ reg3 に格納します。汎用レジスタ reg2 は影響を受けません。

次の表で示されている条件コードのうちの 1 つを「cccc」として指定してください（ただし，cccc ≠ 1101）。

条件コード	条件式	条件コード	条件式
0000	OV = 1	0100	S = 1
1000	OV = 0	1100	S = 0
0001	CY = 1	0101	always（無条件）
1001	CY = 0	0110	(S xor OV) = 1
0010	Z = 1	1110	(S xor OV) = 0
1010	Z = 0	0111	((S xor OV) or Z) = 1
0011	(CY or Z) = 1	1111	((S xor OV) or Z) = 0
1011	(CY or Z) = 0	（1101）	設定禁止

＜ビット・サーチ命令＞

SCH0L	Search zero from left MSB 側からのビット（0）検索

［命令形式］　SCH0L　reg2, reg3

［オペレーション］　GR [reg3] ← search zero from left of GR [reg2]

［フォーマット］　Format IX

［オペコード］

15	0	31	16
rrrrr11111100000		wwwww01101100100	

［フ ラ グ］

CY　最後にビット（0）が見つかったとき 1，そうでないとき 0

OV　0

S　0

Z　ビット（0）が見つからなかったとき 1，そうでないとき 0

SAT　－

［説　明］　汎用レジスタ reg2 のワード・データを左側（MSB 側）から検索し，最初にビット（0）が見つかった位置を汎用レジスタ reg3 に 16 進数で書き込みます（たとえば，reg2 のビット 31 が 0 の場合は，reg3 に 01H を書き込みます）。

ビット（0）が見つからなかった場合は，reg3 に 0 を書き込み，同時に Z フラグをセット（1）します。最後にビット（0）が見つかった場合は CY フラグをセット（1）します。

＜ビット・サーチ命令＞

SCH0R	Search zero from right LSB 側からのビット（0）検索

［命令形式］　SCH0R　reg2, reg3

［オペレーション］　GR [reg3] ← search zero from right of GR [reg2]

［フォーマット］　Format IX

［オペコード］

15	0	31	16
rrrrr11111100000		wwwww01101100000	

［フ ラ グ］

CY　最後にビット（0）が見つかったとき 1，そうでないとき 0

OV　0

S　0

Z　ビット（0）が見つからなかったとき 1，そうでないとき 0

SAT　－

［説　明］　汎用レジスタ reg2 のワード・データを右側（LSB 側）から検索し，最初にビット（0）が見つかった位置を汎用レジスタ reg3 に 16 進数で書き込みます（たとえば，reg2 のビット 0 が 0 の場合は，reg3 に 01H を書き込みます）。

ビット（0）が見つからなかった場合は，reg3 に 0 を書き込み，同時に Z フラグをセット（1）します。最後にビット（0）が見つかった場合は CY フラグをセット（1）します。

＜ビット・サーチ命令＞

SCH1L	Search one from left MSB 側からのビット（1）検索

［命令形式］　SCH1L　reg2, reg3

［オペレーション］　GR [reg3] ← search one from left of GR [reg2]

［フォーマット］　Format IX

［オペコード］

15	0	31	16
rrrrr11111100000		wwwww01101100110	

［フ ラ グ］

CY　最後にビット（1）が見つかったとき 1，そうでないとき 0

OV　0

S　0

Z　ビット（1）が見つからなかったとき 1，そうでないとき 0

SAT　－

［説　明］　汎用レジスタ reg2 のワード・データを左側（MSB 側）から検索し，最初にビット（1）が見つかった位置を汎用レジスタ reg3 に 16 進数で書き込みます（たとえば，reg2 のビット 31 が 1 の場合は，reg3 に 01H を書き込みます）。

ビット（1）が見つからなかった場合は，reg3 に 0 を書き込み，同時に Z フラグをセット（1）します。最後にビット（1）が見つかった場合は CY フラグをセット（1）します。

＜ビット・サーチ命令＞

SCH1R	Search one from right LSB 側からのビット（1）検索

［命令形式］　SCH1R　reg2, reg3

［オペレーション］　GR [reg3] ← search one from right of GR [reg2]

［フォーマット］　Format IX

［オペコード］

15	0	31	16
rrrrr11111100000		wwwww01101100010	

［フ ラ グ］

CY　最後にビット（1）が見つかったとき 1，そうでないとき 0

OV　0

S　0

Z　ビット（1）が見つからなかったとき 1，そうでないとき 0

SAT　－

［説　明］　汎用レジスタ reg2 のワード・データを右側（LSB 側）から検索し，最初にビット（1）が見つかった位置を汎用レジスタ reg3 に 16 進数で書き込みます（たとえば，reg2 のビット 0 が 1 の場合は，reg3 に 01H を書き込みます）。

ビット（1）が見つからなかった場合は，reg3 に 0 を書き込み，同時に Z フラグをセット（1）します。最後にビット（1）が見つかった場合は CY フラグをセット（1）します。

＜ビット操作命令＞

SET1	Set bit ビット・セット

［命令形式］　（1）SET1　bit#3, disp16 [reg1]

（2）SET1　reg2, [reg1]

［オペレーション］　（1）adr ← GR [reg1] + sign-extend (disp16)

Z フラグ ←Not (Load-memory-bit (adr, bit#3))

Store-memory-bit (adr, bit#3, 1)

（2）adr ← GR [reg1]

Z フラグ ←Not (Load-memory-bit (adr, reg2))

Store-memory-bit (adr, reg2, 1)

［フォーマット］　（1）Format VIII

（2）Format IX

［オペコード］

	15	0	31	16
(1)	00bbb111110RRRRR		ddddddddddddddddd	
(2)	rrrrr111111RRRRR		0000000011100000	

［フ ラ グ］　CY　－

OV　－

S　－

Z　指定したビットが 0 のとき 1，指定したビットが 1 のとき 0

SAT　－

［説　明］　（1）まず，汎用レジスタ reg1 のワード・データと，ワード長まで符号拡張した 16 ビット・ディスプレースメントを加算して 32 ビット・アドレスを生成します。生成したアドレスのバイト・データを読み出し，3 ビットのビット・ナンバで指定されるビットをセット（1）し，元のアドレスに書き戻します。

（2）まず，汎用レジスタ reg1 のワード・データを読み出して 32 ビット・アドレスを生成します。生成したアドレスのバイト・データを読み出し，汎用レジスタ reg2 の下位 3 ビットで指定されるビットをセット（1）し，元のアドレスに書き戻します。

［補　足］　PSW レジスタの Z フラグはこの命令を実行する前に該当ビットが 0 か 1 だったかを示します。この命令実行後の該当ビットの内容を示すものではありません。

＜データ操作命令＞

SETF	Set flag condition フラグ条件の設定

［命令形式］　SETF　cccc, reg2

［オペレーション］　if conditions are satisfied
　then GR [reg2] ← 00000001H
　else GR [reg2] ← 00000000H

［フォーマット］　Format IX

［オペコード］

15	0 31	16
rrrrr1111110cccc	0000000000000000	

［フ ラ グ］
CY　－
OV　－
S　－
Z　－
SAT　－

［説　明］　**条件コード「cccc」の示す条件が満たされた場合，汎用レジスタ** reg2 **に** 1 **を，そうでない場合は** 0 **を格納します。**
次の表で示されている条件コードのうちの 1 **つを「**cccc**」として指定してください。**

条件コード	条件式	条件コード	条件式
0000	OV = 1	0100	S = 1
1000	OV = 0	1100	S = 0
0001	CY = 1	0101	always（無条件）
1001	CY = 0	1101	SAT = 1
0010	Z = 1	0110	(S xor OV) = 1
1010	Z = 0	1110	(S xor OV) = 0
0011	(CY or Z) = 1	0111	((S xor OV) or Z) = 1
1011	(CY or Z) = 0	1111	((S xor OV) or Z) = 0

［補　足］　この命令の利用方法の例を示します。

（1）複数の条件節の翻訳

C 言語での if（A）という文において，A が複数の条件節（a_1, a_2, a_3, …）から成り立つとき，通常は if（a_1）then, if（a_2）then というシーケンスに翻訳します。オブジェクト・コードでは a_n に相当する評価の結果を見て「条件分岐」をします。パイプライン・プロセッサでは「条件判断 + 分岐」は通常の演算に比べて遅いので，おのおのの条件節を評価した結果 if（a_n）の結果をレジスタ Ra に覚えておきます。すべての条件節を評価し終わったあとに Ra_n をまとめて論理演算することで，パイプラインによる遅れを回避できます。

（2）倍長演算

Add with Carry のような倍長演算をするときに，CY フラグの結果を汎用レジスタ reg2 に格納できるため，下位からの桁上がりを数値として表現できます。

＜データ操作命令＞

SHL	Shift logical left by register/immediate (5-bit) 論理左シフト

［命令形式］　（1）SHL　reg1, reg2
（2）SHL　imm5, reg2
（3）SHL　reg1, reg2, reg3

［オペレーション］　（1）GR [reg2] ← GR [reg2] logically shift left by GR [reg1]
（2）GR [reg2] ← GR [reg2] logically shift left by zero-extend (imm5)
（3）GR [reg3] ← GR [reg2] logically shift left by GR [reg1]

［フォーマット］　（1）Format IX
（2）Format II
（3）Format XI

［オペコード］

(1)

15	0	31	16
rrrrr111111RRRRR		0000000011000000	

(2)

15	0
rrrrr010110iiiii	

(3)

15	0	31	16
rrrrr111111RRRRR		wwwww00011000010	

［フ ラ グ］　CY　**最後にシフト・アウトしたビットが** 1 **のとき** 1**，そうでないとき** 0**，ただしシフト数が** 0 **のときは** 0
OV　0
S　**演算結果が負のとき** 1**，そうでないとき** 0
Z　**演算結果が** 0 **のとき** 1**，そうでないとき** 0
SAT　－

［説　明］　（1）**汎用レジスタ** reg2 **のワード・データを汎用レジスタ** reg1 **の下位** 5 **ビットで示されるシフト数分，**0 **から**+31 **までを論理左シフトし（**LSB **側に** 0 **を送り込む），汎用レジスタ** reg2 **に書き込みます。シフト数が** 0 **のときは，汎用レジスタ** reg2 **は命令実行前の値を保持します。汎用レジスタ** reg1 **は影響を受けません。**
（2）**汎用レジスタ** reg2 **のワード・データを，ワード長までゼロ拡張した** 5 **ビット・イミーディエトで示されるシフト数分，**0 **から**+31 **までを論理左シフトし（**LSB **側に** 0 **を送り込む），汎用レジスタ** reg2 **に書き込みます。シフト数が** 0 **のときは，汎用レジスタ** reg2 **は命令実行前の値を保持します。**
（3）**汎用レジスタ** reg2 **のワード・データを汎用レジスタ** reg1 **の下位** 5 **ビットで示されるシフト数分，**0 **から**+31 **までを論理左シフトし（**LSB **側に** 0 **を送り込む），汎用レジスタ** reg3 **に書き込みます。シフト数が** 0 **のときは，汎用レジスタ** reg3 **は命令実行前の値を保持します。汎用レジスタ** reg1, reg2 **は影響を受けません。**

＜データ操作命令＞

SHR

Shift logical right by register/immediate (5-bit)

論理右シフト

［命令形式］　（1）SHR　reg1, reg2
（2）SHR　imm5, reg2
（3）SHR　reg1, reg2, reg3

［オペレーション］　（1）GR [reg2] ← GR [reg2] logically shift right by GR [reg1]
（2）GR [reg2] ← GR [reg2] logically shift right by zero-extend（imm5）
（3）GR [reg3] ← GR [reg2] logically shift right by GR [reg1]

［フォーマット］　（1）Format IX
（2）Format II
（3）Format XI

［オペコード］

(1)

15	0	31	16
rrrrr111111RRRRR		0000000010000000	

(2)

15	0
rrrrr010100iiiii	

(3)

15	0	31	16
rrrrr111111RRRRR		wwwww00010000010	

［フ ラ グ］　CY　最後にシフト・アウトしたビットが 1 のとき 1，そうでないとき 0，ただしシフト数が 0 のときは 0
OV　0
S　演算結果が負のとき 1，そうでないとき 0
Z　演算結果が 0 のとき 1，そうでないとき 0
SAT　－

［説　明］　（1）汎用レジスタ reg2 のワード・データを汎用レジスタ reg1 の下位 5 ビットで示されるシフト数分，0 から+31 までを論理右シフトし（MSB 側に 0 を送り込む），汎用レジスタ reg2 に書き込みます。シフト数が 0 のときは，汎用レジスタ reg2 は命令実行前と同じ値を保持します。汎用レジスタ reg1 は影響を受けません。
（2）汎用レジスタ reg2 のワード・データを，ワード長までゼロ拡張した 5 ビット・イミーディエトで示されるシフト数分，0 から+31 までを論理右シフトし（MSB 側に 0 を送り込む），汎用レジスタ reg2 に書き込みます。シフト数が 0 のときは，汎用レジスタ reg2 は命令実行前の値を保持します。
（3）汎用レジスタ reg2 のワード・データを汎用レジスタ reg1 の下位 5 ビットで示されるシフト数分，0 から+31 までを論理右シフトし（MSB 側に 0 を送り込む），汎用レジスタ reg3 に書き込みます。シフト数が 0 のときは，汎用レジスタ reg3 は命令実行前の値を保持します。汎用レジスタ reg1, reg2 は影響を受けません。

＜ロード命令＞

SLD.B	Short format load byte （符号付き）バイト・データのロード

［命令形式］　SLD.B　disp7 [ep] , reg2

［オペレーション］　adr ← ep + zero-extend (disp7)
GR [reg2] ← sign-extend (Load-memory (adr, Byte))

［フォーマット］　Format IV

［オペコード］

15	0
rrrrr0110ddddddd	

［フ ラ グ］　CY　－
OV　－
S　－
Z　－
SAT　－

［説　明］　エレメント・ポインタと，ワード長までゼロ拡張した 7 ビット・ディスプレースメントを加算して 32 ビット・アドレスを生成します。生成したアドレスからバイト・データを読み出し，ワード長まで符号拡張し，reg2 に格納します。

［補　足］　この命令実行中に割り込みが発生すると，実行を中止し，戻り番地をこの命令の先頭アドレスとして割り込みを処理してから，割り込み処理完了後に再実行します。
また，各バス（VFB, VDB, VSB, NPB, 命令キャッシュ用バス，データ・キャッシュ用バス）に接続される異なる資源に対するアクセスについては，バス・サイクルの順序が入れ替わる可能性があります（同一バスに対するアクセスでは，入れ替わることはありません）。

＜ロード命令＞

SLD.BU	Short format load byte unsigned （符号なし）バイト・データのロード

［命令形式］　SLD.BU　disp4 [ep] , reg2

［オペレーション］　adr ← ep + zero-extend (disp4)
GR [reg2] ← zero-extend (Load-memory (adr, Byte))

［フォーマット］　Format IV

［オペコード］

15	0
rrrrr0000110dddd	

ただし，rrrrr は 00000 以外です。

［フ ラ グ］
CY　－
OV　－
S　－
Z　－
SAT　－

［説　明］　エレメント・ポインタと，ワード長までゼロ拡張した 4 ビット・ディスプレースメントを加算して 32 ビット・アドレスを生成します。生成したアドレスからバイト・データを読み出し，ワード長までゼロ拡張し，reg2 に格納します。

［補　足］　この命令実行中に割り込みが発生すると，実行を中止し，戻り番地をこの命令の先頭アドレスとして割り込みを処理してから，割り込み処理完了後に再実行します。
また，各バス（VFB, VDB, VSB, NPB, 命令キャッシュ用バス，データ・キャッシュ用バス）に接続される異なる資源に対するアクセスについては，バス・サイクルの順序が入れ替わる可能性があります（同一バスに対するアクセスでは，入れ替わることはありません）。

＜ロード命令＞

SLD.H	Short format load half-word （符号付き）ハーフワード・データのロード

［命令形式］　SLD.H　disp8 [ep] , reg2

［オペレーション］　adr ← ep + zero-extend (disp8)
GR [reg2] ← sign-extend (Load-memory (adr, Half-word))

［フォーマット］　Format IV

［オペコード］

15	0
rrrrr1000ddddddd	

ただし，ddddddd は disp8 の上位 7 ビットです。

［フ ラ グ］
CY　－
OV　－
S　－
Z　－
SAT　－

［説　明］　エレメント・ポインタと，ワード長までゼロ拡張した 8 ビット・ディスプレースメントを加算して 32 ビット・アドレスを生成します。生成した 32 ビット・アドレスからハーフワード・データを読み出し，ワード長まで符号拡張し，reg2 に格納します。

［補　足］　この命令実行中に割り込みが発生すると，実行を中止し，戻り番地をこの命令の先頭アドレスとして割り込みを処理してから，割り込み処理完了後に再実行します。
また，各バス（VFB, VDB, VSB, NPB, 命令キャッシュ用バス，データ・キャッシュ用バス）に接続される異なる資源に対するアクセスについては，バス・サイクルの順序が入れ替わる可能性があります（同一バスに対するアクセスでは，入れ替わることはありません）。

［注　意］　エレメント・ポインタとワード長までゼロ拡張した 8 ビット・ディスプレースメントの加算結果は，ミス・アライン・モード設定により次の 2 種類があります。

- ビット 0 を 0 にマスクしてアドレス生成（ミス・アライン・アクセス禁止の場合）
- ビット 0 をマスクしないでアドレス生成（ミス・アライン・アクセス許可の場合）

ミス・アライン・アクセスについては，3. 3　**データ・アラインメント**を参照してください。

＜ロード命令＞

SLD.HU	Short format load half-word unsigned （符号なし）ハーフワード・データのロード

［命令形式］　SLD.HU　disp5 [ep] , reg2

［オペレーション］　adr ← ep + zero-extend (disp5)
GR [reg2] ← zero-extend (Load-memory (adr, Half-word))

［フォーマット］　Format IV

［オペコード］　15　0

rrrrr0000111dddd

ただし，dddd は disp5 の上位 4 ビット，rrrrr は 00000 以外です。

［フ ラ グ］
CY　－
OV　－
S　－
Z　－
SAT　－

［説　明］　エレメント・ポインタと，ワード長までゼロ拡張した 5 ビット・ディスプレースメントを加算して 32 ビット・アドレスを生成します。この生成した 32 ビット・アドレスからハーフワード・データを読み出し，ワード長までゼロ拡張し，reg2 に格納します。

［補　足］　この命令実行中に割り込みが発生すると，実行を中止し，戻り番地をこの命令の先頭アドレスとして割り込みを処理してから，割り込み処理完了後に再実行します。
また，各バス（VFB, VDB, VSB, NPB, 命令キャッシュ用バス，データ・キャッシュ用バス）に接続される異なる資源に対するアクセスについては，バス・サイクルの順序が入れ替わる可能性があります（同一バスに対するアクセスでは，入れ替わることはありません）。

［注　意］　エレメント・ポインタとワード長までゼロ拡張した 8 ビット・ディスプレースメントの加算結果は，ミス・アライン・モード設定により次の 2 種類があります。

- ビット 0 を 0 にマスクしてアドレス生成（ミス・アライン・アクセス禁止の場合）
- ビット 0 をマスクしないでアドレス生成（ミス・アライン・アクセス許可の場合）

ミス・アライン・アクセスについては，3. 3　**データ・アラインメント**を参照してください。

＜ロード命令＞

SLD.W	Short format load word ワード・データのロード

［命令形式］　SLD.W　disp8 [ep] , reg2

［オペレーション］　adr ← ep + zero-extend (disp8)
GR [reg2] ← Load-memory (adr, Word)

［フォーマット］　Format IV

［オペコード］　15　0

```
rrrrr1010dddddd0
```

ただし，dddddd は disp8 の上位 6 ビットです。

［フ ラ グ］　CY　－
OV　－
S　－
Z　－
SAT　－

［説　明］　エレメント・ポインタと，ワード長までゼロ拡張した 8 ビット・ディスプレースメントを加算して 32 ビット・アドレスを生成します。この生成した 32 ビット・アドレスからワード・データを読み出し，reg2 に格納します。

［補　足］　この命令実行中に割り込みが発生すると，実行を中止し，戻り番地をこの命令の先頭アドレスとして割り込みを処理してから，割り込み処理完了後に再実行します。
また，各バス（VFB, VDB, VSB, NPB, 命令キャッシュ用バス，データ・キャッシュ用バス）に接続される異なる資源に対するアクセスについては，バス・サイクルの順序が入れ替わる可能性があります（同一バスに対するアクセスでは，入れ替わることはありません）。

［注　意］　エレメント・ポインタとワード長までゼロ拡張した 8 ビット・ディスプレースメントの加算結果は，ミス・アライン・モード設定により次の 2 種類があります。

- ビット 0 とビット 1 を 0 にマスクしてアドレス生成（ミス・アライン・アクセス禁止の場合）
- ビット 0 とビット 1 をマスクしないでアドレス生成（ミス・アライン・アクセス許可の場合）

ミス・アライン・アクセスについては，3. 3　**データ・アラインメント**を参照してください。

＜ストア命令＞

SST.B	Short format store byte バイト・データのストア

［命令形式］　SST.B　reg2, disp7 [ep]

［オペレーション］　adr ← ep + zero-extend (disp7)
Store-memory (adr, GR [reg2] , Byte)

［フォーマット］　Format IV

［オペコード］

15	0
rrrrr0111ddddddd	

［フ ラ グ］
CY　－
OV　－
S　－
Z　－
SAT　－

［説　明］　エレメント・ポインタと，ワード長までゼロ拡張した 7 ビット・ディスプレースメントを加算して 32 ビット・アドレスを生成します。reg2 の最下位バイト・データを生成したアドレスに格納します。

［補　足］　この命令実行中に割り込みが発生すると，実行を中止し，戻り番地をこの命令の先頭アドレスとして割り込みを処理してから，割り込み処理完了後に再実行します。
各バス（VFB, VDB, VSB, NPB, 命令キャッシュ用バス，データ・キャッシュ用バス）に接続される異なる資源に対するアクセスについては，バス・サイクルの順序が入れ替わる可能性があります（同一バスに対するアクセスでは，入れ替わることはありません）。

＜ストア命令＞

SST.H	Short format store half-word ハーフワード・データのストア

［命令形式］　SST.H　reg2, disp8 [ep]

［オペレーション］　adr ← ep + zero-extend (disp8)
Store-memory (adr, GR [reg2] , Half-word)

［フォーマット］　Format IV

［オペコード］

15	0
rrrrr1001ddddddd	

ただし，dddddddは disp8 の上位 7 ビットです。

［フ ラ グ］
CY　－
OV　－
S　－
Z　－
SAT　－

［説　明］　エレメント・ポインタと，ワード長までゼロ拡張した 8 ビット・ディスプレースメントを加算して 32 ビット・アドレスを生成します。reg2 の下位ハーフワード・データを生成した 32 ビット・アドレスに格納します。

［注　意］　エレメント・ポインタとワード長までゼロ拡張した 8 ビット・ディスプレースメントの加算結果は，ミス・アライン・モード設定により次の 2 種類があります。

- ビット 0 を 0 にマスクしてアドレス生成（ミス・アライン・アクセス禁止の場合）
- ビット 0 をマスクしないでアドレス生成（ミス・アライン・アクセス許可の場合）

ミス・アライン・アクセスについては，3. 3　**データ・アラインメント**を参照してください。

［補　足］　この命令実行中に割り込みが発生すると，実行を中止し，戻り番地をこの命令の先頭アドレスとして割り込みを処理してから，割り込み処理完了後に再実行します。
各バス（VFB, VDB, VSB, NPB, 命令キャッシュ用バス，データ・キャッシュ用バス）に接続される異なる資源に対するアクセスについては，バス・サイクルの順序が入れ替わる可能性があります（同一バスに対するアクセスでは，入れ替わることはありません）。

＜ストア命令＞

SST.W	Short format store word ワード・データのストア

［命令形式］　SST.W　reg2, disp8 [ep]

［オペレーション］　adr ← ep + zero-extend (disp8)
Store-memory (adr, GR [reg2] , Word)

［フォーマット］　Format IV

［オペコード］　15　0

rrrrr1010dddddd1

ただし，dddddd は disp8 の上位 6 ビットです。

［フ ラ グ］　CY　－
OV　－
S　－
Z　－
SAT　－

［説　明］　エレメント・ポインタと，ワード長までゼロ拡張した 8 ビット・ディスプレースメントを加算して 32 ビット・アドレスを生成します。reg2 のワード・データを生成した 32 ビット・アドレスに格納します。

［注　意］　エレメント・ポインタとワード長までゼロ拡張した 8 ビット・ディスプレースメントの加算結果は，ミス・アライン・モード設定により次の 2 種類があります。

- ビット 0 とビット 1 を 0 にマスクしてアドレス生成（ミス・アライン・アクセス禁止の場合）
- ビット 0 とビット 1 をマスクしないでアドレス生成（ミス・アライン・アクセス許可の場合）

ミス・アライン・アクセスについては，3. 3　**データ・アラインメント**を参照してください。

［補　足］　この命令実行中に割り込みが発生すると，実行を中止し，戻り番地をこの命令の先頭アドレスとして割り込みを処理してから，割り込み処理完了後に再実行します。
各バス（VFB, VDB, VSB, NPB, 命令キャッシュ用バス，データ・キャッシュ用バス）に接続される異なる資源に対するアクセスについては，バス・サイクルの順序が入れ替わる可能性があります（同一バスに対するアクセスでは，入れ替わることはありません）。

＜ストア命令＞

ST.B	Store byte バイト・データのストア

［命令形式］　ST.B　reg2, disp16 [reg1]

［オペレーション］　adr ← GR [reg1] + sign-extend (disp16)
Store-memory (adr, GR [reg2], Byte)

［フォーマット］　Format VII

［オペコード］

15	0 31	16
rrrrr111010RRRRR	ddddddddddddddddd	

［フ ラ グ］　CY　－
OV　－
S　－
Z　－
SAT　－

［説　明］　汎用レジスタ reg1 のデータと，ワード長まで符号拡張した 16 ビット・ディスプレースメントを加算して 32 ビット・アドレスを生成します。汎用レジスタ reg2 の最下位のバイト・データを生成したアドレスに格納します。

［補　足］　この命令実行中に割り込みが発生すると，実行を中止し，戻り番地をこの命令の先頭アドレスとして割り込みを処理してから，割り込み処理完了後に再実行します。
各バス（VFB, VDB, VSB, NPB, 命令キャッシュ用バス，データ・キャッシュ用バス）に接続される異なる資源に対するアクセスについては，バス・サイクルの順序が入れ替わる可能性があります（同一バスに対するアクセスでは，入れ替わることはありません）。

＜ストア命令＞

ST.H	Store half-word ハーフワード・データのストア

［命令形式］　ST.H　reg2, disp16 [reg1]

［オペレーション］　adr ← GR [reg1] + sign-extend (disp16)
Store-memory (adr, GR [reg2], Half-word)

［フォーマット］　Format VII

［オペコード］

15	0	31	16
rrrrr111011RRRRR		ddddddddddddddd0	

ただし，ddddddddddddddd は disp16 の上位 15 ビットです。

［フ ラ グ］　CY　－
OV　－
S　－
Z　－
SAT　－

［説　明］　汎用レジスタ reg1 のデータと，ワード長まで符号拡張した 16 ビット・ディスプレースメントを加算して 32 ビット・アドレスを生成します。汎用レジスタ reg2 の下位ハーフワード・データを生成した 32 ビット・アドレスに格納します。

［注　意］　汎用レジスタ reg1 のデータとワード長まで符号拡張した 16 ビット・ディスプレースメントの加算結果は，ミス・アライン・モード設定により次の 2 種類があります。

- ビット 0 を 0 にマスクしてアドレス生成（ミス・アライン・アクセス禁止の場合）
- ビット 0 をマスクしないでアドレス生成（ミス・アライン・アクセス許可の場合）

ミス・アライン・アクセスについては，3. 3　**データ・アラインメント**を参照してください。

［補　足］　この命令実行中に割り込みが発生すると，実行を中止し，戻り番地をこの命令の先頭アドレスとして割り込みを処理してから，割り込み処理完了後に再実行します。
各バス（VFB, VDB, VSB, NPB, 命令キャッシュ用バス，データ・キャッシュ用バス）に接続される異なる資源に対するアクセスについては，バス・サイクルの順序が入れ替わる可能性があります（同一バスに対するアクセスでは，入れ替わることはありません）。

＜ストア命令＞

ST.W	Store word ワード・データのストア

［命令形式］　ST.W　reg2, disp16 [reg1]

［オペレーション］　adr ← GR [reg1] + sign-extend (disp16)
Store-memory (adr, GR [reg2], Word)

［フォーマット］　Format VII

［オペコード］

15	0	31	16
rrrrr111011RRRRR		ddddddddddddddd1	

ただし，ddddddddddddddd は disp16 の上位 15 ビットです。

［フ ラ グ］
CY　－
OV　－
S　－
Z　－
SAT　－

［説　明］　汎用レジスタ reg1 のデータと，ワード長まで符号拡張した 16 ビット・ディスプレースメントを加算して 32 ビット・アドレスを生成します。汎用レジスタ reg2 のワード・データを生成した 32 ビット・アドレスに格納します。

［注　意］　汎用レジスタ reg1 のデータとワード長まで符号拡張した 16 ビット・ディスプレースメントの加算結果は，ミス・アライン・モード設定により次の 2 種類があります。

- ビット 0 とビット 1 を 0 にマスクしてアドレス生成（ミス・アライン・アクセス禁止の場合）
- ビット 0 とビット 1 をマスクしないでアドレス生成（ミス・アライン・アクセス許可の場合）

ミス・アライン・アクセスについては，3. 3　**データ・アラインメント**を参照してください。

［補　足］　この命令実行中に割り込みが発生すると，実行を中止し，戻り番地をこの命令の先頭アドレスとして割り込みを処理してから，割り込み処理完了後に再実行します。各バス（VFB, VDB, VSB, NPB, 命令キャッシュ用バス，データ・キャッシュ用バス）に接続される異なる資源に対するアクセスについては，バス・サイクルの順序が入れ替わる可能性があります（同一バスに対するアクセスでは，入れ替わることはありません）。

＜特殊命令＞

STSR	Store contents of system register システム・レジスタの内容のストア

［命令形式］　STSR　regID, reg2

［オペレーション］　GR [reg2] ← SR [regID]

［フォーマット］　Format IX

［オペコード］

15	0 31	16
rrrrr111111RRRRR	0000000001000000	

［フ ラ グ］

CY　－
OV　－
S　－
Z　－
SAT　－

［説　明］　システム・レジスタ番号（regID）で指定されるシステム・レジスタの内容を汎用レジスタ reg2 に設定します。システム・レジスタは影響を受け付けません。

［注　意］　システム・レジスタ番号は，システム・レジスタを一意に識別するための番号です。予約されているシステム・レジスタ番号を指定した場合の動作は保証しません。

＜算術演算命令＞

SUB	Subtract 減算

［命令形式］　SUB　reg1, reg2

［オペレーション］　GR [reg2] ← GR [reg2] – GR [reg1]

［フォーマット］　Format I

［オペコード］

15	0
rrrrr001101RRRRR	

［フ ラ グ］

CY　MSB へのボローがあれば 1，そうでないとき 0

OV　オーバフローが起こったとき 1，そうでないとき 0

S　演算結果が負のとき 1，そうでないとき 0

Z　演算結果が 0 のとき 1，そうでないとき 0

SAT　－

［説　明］　汎用レジスタ reg2 のワード・データから汎用レジスタ reg1 のワード・データを減算し，その結果を汎用レジスタ reg2 に格納します。汎用レジスタ reg1 は影響を受けません。

＜算術演算命令＞

SUBR	Subtract reverse 逆減算

［命令形式］　SUBR　reg1, reg2

［オペレーション］　GR [reg2] ←GR [reg1] – GR [reg2]

［フォーマット］　Format I

［オペコード］

15	0
rrrrr001100RRRRR	

［フ ラ グ］

CY　MSB へのボローがあれば 1，そうでないとき 0

OV　オーバフローが起こったとき 1，そうでないとき 0

S　演算結果が負のとき 1，そうでないとき 0

Z　演算結果が 0 のとき 1，そうでないとき 0

SAT　－

［説　明］　汎用レジスタ reg1 のワード・データから汎用レジスタ reg2 のワード・データを減算し，その結果を汎用レジスタ reg2 に格納します。汎用レジスタ reg1 は影響を受けません。

＜特殊命令＞

SWITCH	Jump with table look up テーブル参照分岐

［命令形式］　SWITCH　reg1

［オペレーション］　adr ← (PC + 2) + (GR [reg1] logically shift left by 1)
PC ← (PC + 2) + (sign-extend (Load-memory (adr, Half-word))) logically shift left by 1

［フォーマット］　Format I

［オペコード］

15	0
0000000010RRRRR	

［フ ラ グ］　CY　－
OV　－
S　－
Z　－
SAT　－

［説　明］　次の順に処理を行います。

<1> テーブルの先頭アドレス（SWITCH 命令の次のアドレス）と 1 ビット論理左シフトした汎用レジスタ reg1 のデータを加算し，32 ビット・テーブル・エントリ・アドレスを生成します。
<2> <1>で生成されたアドレスが指し示すハーフワード・エントリ・データをロードします。
<3> ロードしたハーフワード・データをワード長まで符号拡張し，1 ビット論理左シフトしたあとテーブルの先頭アドレス（SWITCH 命令の次のアドレス）を加算し，32 ビット・ターゲット・アドレスを生成します。
<4> <3>で生成されたターゲット・アドレスへ分岐します。

＜データ操作命令＞

SXB	Sign extend byte バイト・データの符号拡張

［命令形式］　SXB　reg1

［オペレーション］　GR [reg1] ← sign-extend (GR [reg1] (7:0))

［フォーマット］　Format I

［オペコード］

15	0
00000000101RRRRR	

［フ ラ グ］

CY	－
OV	－
S	－
Z	－
SAT	－

［説　明］　汎用レジスタ reg1 の最下位バイトをワード長に符号拡張します。

＜データ操作命令＞

SXH	Sign extend half-word ハーフワード・データの符号拡張

［命令形式］　SXH　reg1

［オペレーション］　GR [reg1] ← sign-extend (GR [reg1] (15:0))

［フォーマット］　Format I

［オペコード］

15	0
00000000111RRRRR	

［フ ラ グ］

CY	–
OV	–
S	–
Z	–
SAT	–

［説　明］　汎用レジスタ reg1 の下位ハーフワードをワード長に符号拡張します。

＜特殊命令＞

TRAP	Trap ソフトウエア例外

［命令形式］　TRAP　vector

［オペレーション］　EIPC ← PC + 4 (復帰 PC)

EIPSW ← PSW

ECR.EICC ← 例外コード（40H-4FH, 50H-5FH）

PSW.EP ← 1

PSW.ID ← 1

PC ← 00000040H（vector が 00H-0FH（例外コード：40H-4FH）のとき）

00000050H（vector が 10H-1FH（例外コード：50H-5FH）のとき）

［フォーマット］　Format X

［オペコード］

15	0	31	16
0000011111iiiii		0000000100000000	

［フ ラ グ］　CY　–

OV　–

S　–

Z　–

SAT　–

［説　明］　復帰 PC，PSW を EIPC，EIPSW レジスタに退避し，例外コードの設定（ECR の EICC），PSW のフラグの設定（EP, ID フラグをセット（1））を行ったあと，「vector」で指定されるトラップ・ベクタ（00H-1FH）に対応するハンドラ・アドレスにジャンプし，例外処理を開始します。

EP, ID フラグ以外の PSW レジスタの各フラグは影響を受けません。

なお，復帰 PC とは，TRAP 命令の次の命令のアドレスです。

＜論理演算命令＞

TST	Test テスト

［命令形式］　TST　reg1, reg2

［オペレーション］　result ← GR [reg2] AND GR [reg1]

［フォーマット］　Format I

［オペコード］

15	0
rrrrr001011RRRRR	

［フ ラ グ］

CY　–

OV　0

S　演算結果のワード・データの MSB が 1 のとき 1，そうでないとき 0

Z　演算結果が 0 のとき 1，そうでないとき 0

SAT　–

［説　明］　汎用レジスタ reg2 のワード・データと汎用レジスタ reg1 のワード・データの論理積をとります。結果は格納されず，フラグだけが影響を受けます。汎用レジスタ reg1, reg2 は影響を受けません。

＜ビット操作命令＞

TST1	Test bit **ビット・テスト**

[命令形式]　（1）TST1　bit#3, disp16 [reg1]

（2）TST1　reg2, [reg1]

[オペレーション]　（1）adr ← GR [reg1] + sign-extend (disp16)

Z **フラグ** ← Not (Load-memory-bit (adr, bit#3))

（2）adr ← GR [reg1]

Z **フラグ** ← Not (Load-memory-bit (adr, reg2))

[フォーマット]　（1）Format VIII

（2）Format IX

[オペコード]

（1）

15	0	31	16
11bbb111110RRRRR		ddddddddddddddddd	

（2）

15	0	31	16
rrrrr111111RRRRR		0000000011100110	

[フ ラ グ]

CY　–

OV　–

S　–

Z　**指定したビットが** 0 **のとき** 1**，指定したビットが** 1 **のとき** 0

SAT　–

[説　明]　（1）**まず，汎用レジスタ** reg1 **のワード・データと，ワード長まで符号拡張した** 16 **ビット・ディスプレースメントを加算して** 32 **ビット・アドレスを生成します。生成したアドレスのバイト・データの，**3 **ビットのビット・ナンバで指定されるビットが** 0 **ならば** Z **フラグをセット（**1**）し，**1 **ならばクリア（**0**）します。指定されたビットも含め，バイト・データは影響を受けません。**

（2）**まず，汎用レジスタ** reg1 **のワード・データを読み出して** 32 **ビット・アドレスを生成します。**

生成したアドレスのバイト・データの，汎用レジスタ reg2 **の下位** 3 **ビットで指定されるビットが** 0 **ならば** Z **フラグをセット（**1**）し，**1 **ならばクリア（**0**）します。指定されたビットも含め，バイト・データは影響を受けません。**

＜論理演算命令＞

XOR	Exclusive OR 排他的論理和

［命令形式］　XOR　reg1, reg2

［オペレーション］　GR [reg2] ← GR [reg2] XOR GR [reg1]

［フォーマット］　Format I

［オペコード］

15	0
rrrrr001001RRRRR	

［フ ラ グ］

CY　–

OV　0

S　演算結果のワード・データの MSB が 1 のとき 1，そうでないとき 0

Z　演算結果が 0 のとき 1，そうでないとき 0

SAT　–

［説　明］　汎用レジスタ reg2 のワード・データと汎用レジスタ reg1 のワード・データとの排他的論理和をとり，その結果を汎用レジスタ reg2 に格納します。汎用レジスタ reg1 は影響を受けません。

＜論理演算命令＞

XORI	Exclusive OR immediate (16-bit) 排他的論理和

［命令形式］　XORI　imm16, reg1, reg2

［オペレーション］　GR [reg2] ← GR [reg1] XOR zero-extend (imm16)

［フォーマット］　Format VI

［オペコード］

15	0	31	16
rrrrr110101RRRRR		iiiiiiiiiiiiiiii	

［フ ラ グ］

CY　–

OV　0

S　演算結果のワード・データの MSB が 1 のとき 1，そうでないとき 0

Z　演算結果が 0 のとき 1，そうでないとき 0

SAT　–

［説　明］　汎用レジスタ reg1 のワード・データとワード長までゼロ拡張した 16 ビット・イミーディエトの排他的論理和をとり，その結果を汎用レジスタ reg2 に格納します。汎用レジスタ reg1 は影響を受けません。

＜データ操作命令＞

ZXB	Zero extend byte **バイト・データのゼロ拡張**

［命令形式］　ZXB　reg1

［オペレーション］　GR [reg1] ← zero-extend (GR [reg1] (7:0))

［フォーマット］　Format I

［オペコード］

15	0
00000000100RRRRR	

［フ ラ グ］

CY	–
OV	–
S	–
Z	–
SAT	–

［説　明］　**汎用レジスタ** reg1 **の最下位バイトをワード長にゼロ拡張します。**

＜データ操作命令＞

ZXH	Zero extend half-word ハーフワード・データのゼロ拡張

［命令形式］　ZXH　reg1

［オペレーション］　GR [reg1] ← zero-extend (GR [reg1] (15:0))

［フォーマット］　Format I

［オペコード］

15	0
00000000110RRRRR	

［フ ラ グ］

CY	–
OV	–
S	–
Z	–
SAT	–

［説　明］　汎用レジスタ reg1 の下位ハーフワードをワード長にゼロ拡張します。

5. 4　命令実行クロック数

表 5－6 に命令実行クロック数一覧を示します。なお，命令実行クロック数は，命令の組み合わせにより異なる場合があります。詳細については，第 8 章 パイプラインを参照してください。

表5－6　命令実行クロック数一覧（1/4）

命令の種類	ニモニック	オペランド	バイト数	実行クロック数			並列発行[注1]
				issue	repeat	latency	
ロード命令	LD.B	disp16 [reg1] , reg2	4	1	1	注2	○（L）
	LD.H	disp16 [reg1] , reg2	4	1	1	注2	○（L）
	LD.W	disp16 [reg1] , reg2	4	1	1	注2	○（L）
	LD.BU	disp16 [reg1] , reg2	4	1	1	注2	○（L）
	LD.HU	disp16 [reg1] , reg2	4	1	1	注2	○（L）
	SLD.B	disp7 [ep] , reg2	2	1	1	注2	○（L）
	SLD.BU	disp4 [ep] , reg2	2	1	1	注2	○（L）
	SLD.H	disp8 [ep] , reg2	2	1	1	注2	○（L）
	SLD.HU	disp5 [ep] , reg2	2	1	1	注2	○（L）
	SLD.W	disp8 [ep] , reg2	2	1	1	注2	○（L）
ストア命令	ST.B	reg2, disp16 [reg1]	4	1	1	1	○（L）
	ST.H	reg2, disp16 [reg1]	4	1	1	1	○（L）
	ST.W	reg2, disp16 [reg1]	4	1	1	1	○（L）
	SST.B	reg2, disp7 [ep]	2	1	1	1	○（L）
	SST.H	reg2, disp8 [ep]	2	1	1	1	○（L）
	SST.W	reg2, disp8 [ep]	2	1	1	1	○（L）
乗算命令	MUL	reg1, reg2, reg3	4	1	1	3	○（L）
	MUL	imm9, reg2, reg3	4	1	1	3	○（L）
	MULH	reg1, reg2	2	1	1	3	○（L）
	MULH	imm5, reg2	2	1	1	3	○（L）
	MULHI	imm16, reg1, reg2	4	1	1	3	○（L）
	MULU	reg1, reg2, reg3	4	1	1	3	○（L）
	MULU	imm9, reg2, reg3	4	1	1	3	○（L）
加算付き乗算命令	MAC	reg1, reg2, reg3, reg4	4	1	1	3	×
	MACU	reg1, reg2, reg3, reg4	4	1	1	3	×
算術演算命令	ADD	reg1, reg2	2	1	1	1	○（R/L）
	ADD	imm5, reg2	2	1	1	1	○（R/L）
	ADDI	imm16, reg1, reg2	4	1	1	1	○（R/L）
	CMP	reg1, reg2	2	1	1	1	○（R/L）
	CMP	imm5, reg2	2	1	1	1	○（R/L）
	MOV	reg1, reg2	2	1	1	1	○（R/L）
	MOV	imm5, reg2	2	1	1	1	○（R/L）
	MOV	imm32, reg1	6	1	1	1	×

表5－6　命令実行クロック数一覧（2/4）

命令の種類	ニモニック	オペランド	バイト数	実行クロック数			並列発行[注1]
				issue	repeat	latency	
算術演算命令	MOVEA	imm16, reg1, reg2	4	1	1	1	○（R/L）
	MOVHI	imm16, reg1, reg2	4	1	1	1	○（R/L）
	SUB	reg1, reg2	2	1	1	1	○（R/L）
	SUBR	reg1, reg2	2	1	1	1	○（R/L）
条件付き演算命令	ADF	cccc, reg1, reg2, reg3	4	1	1	1	×
	SBF	cccc, reg1, reg2, reg3	4	1	1	1	×
飽和演算命令	SATADD	reg1, reg2	2	1	1	1	○（R/L）
	SATADD	imm5, reg2	2	1	1	1	○（R/L）
	SATADD	reg1, reg2, reg3	4	1	1	1	×
	SATSUB	reg1, reg2	2	1	1	1	○（R/L）
	SATSUB	reg1, reg2, reg3	4	1	1	1	×
	SATSUBI	imm16, reg1, reg2	4	1	1	1	○（R/L）
	SATSUBR	reg1, reg2	2	1	1	1	○（R/L）
論理演算命令	AND	reg1, reg2	2	1	1	1	○（R/L）
	ANDI	imm16, reg1, reg2	4	1	1	1	○（R/L）
	NOT	reg1, reg2	2	1	1	1	○（R/L）
	OR	reg1, reg2	2	1	1	1	○（R/L）
	ORI	imm16, reg1, reg2	4	1	1	1	○（R/L）
	TST	reg1, reg2	2	1	1	1	○（R/L）
	XOR	reg1, reg2	2	1	1	1	○（R/L）
	XORI	imm16, reg1, reg2	4	1	1	1	○（R/L）
データ操作命令	BSH	reg2, reg3	4	1	1	1	○（R）
	BSW	reg2, reg3	4	1	1	1	○（R）
	CMOV	cccc, reg1, reg2, reg3	4	1	1	1	○（R）
	CMOV	cccc, imm5, reg2, reg3	4	1	1	1	○（R）
	HSH	reg2, reg3	4	1	1	1	○（R）
	HSW	reg2, reg3	4	1	1	1	○（R）
	SAR	reg1, reg2	4	1	1	1	○（R）
	SAR	imm5, reg2	2	1	1	1	○（R）
	SAR	reg1, reg2, reg3	4	1	1	1	○（R）
	SASF	cccc, reg2	4	1	1	1	○（R）
	SETF	cccc, reg2	4	1	1	1	○（R）
	SHL	reg1, reg2	4	1	1	1	○（R）
	SHL	imm5, reg2	2	1	1	1	○（R）
	SHL	reg1, reg2, reg3	4	1	1	1	○（R）
	SHR	reg1, reg2	4	1	1	1	○（R）
	SHR	imm5, reg2	2	1	1	1	○（R）
	SHR	reg1, reg2, reg3	4	1	1	1	○（R）

表5－6　命令実行クロック数一覧（3/4）

命令の種類	ニモニック	オペランド	バイト数	実行クロック数			並列発行[注1]
				issue	repeat	latency	
データ操作命令	SXB	reg1	2	1	1	1	×
	SXH	reg1	2	1	1	1	×
	ZXB	reg1	2	1	1	1	×
	ZXH	reg1	2	1	1	1	×
ビット・サーチ命令	SCH0L	reg2, reg3	4	1	1	1	○（R）
	SCH0R	reg2, reg3	4	1	1	1	○（R）
	SCH1L	reg2, reg3	4	1	1	1	○（R）
	SCH1R	reg2, reg3	4	1	1	1	○（R）
除算命令	DIV	reg1, reg2, reg3	4	35	35	35	×
	DIVH	reg1, reg2	2	35	35	35	×
	DIVH	reg1, reg2, reg3	4	35	35	35	×
	DIVHU	reg1, reg2, reg3	4	34	34	34	×
	DIVU	reg1, reg2, reg3	4	34	34	34	×
分岐命令	Bcond	disp9（条件成立時）	2	4[注3]	4[注3]	4[注3]	○（B＋R/L）
		disp9（条件不成立時）	2	1	1	1	○（B＋R/L）
	JARL	disp22, reg2	4	4	4	4	○（B＋R/L）
	JARL	disp32, reg1	6	4	4	4	×
	JMP	[reg1]	2	5	5	5	○（B＋R/L）
	JMP	disp32 [reg1]	6	5	5	5	×
	JR	disp22	4	4	4	4	○（B＋R/L）
	JR	disp32	6	4	4	4	×
ビット操作命令	CLR1	bit#3, disp16 [reg1]	4	4[注4]	4[注4]	4[注4]	×
	CLR1	reg2, [reg1]	4	4[注4]	4[注4]	4[注4]	×
	NOT1	bit#3, disp16 [reg1]	4	4[注4]	4[注4]	4[注4]	×
	NOT1	reg2, [reg1]	4	4[注4]	4[注4]	4[注4]	×
	SET1	bit#3, disp16 [reg1]	4	4[注4]	4[注4]	4[注4]	×
	SET1	reg2, [reg1]	4	4[注4]	4[注4]	4[注4]	×
	TST1	bit#3, disp16 [reg1]	4	4[注4]	4[注4]	4[注4]	×
	TST1	reg2, [reg1]	4	4[注4]	4[注4]	4[注4]	×
特殊命令	CALLT	imm6	2	8	8	8	×
	CTRET	－	4	9	9	9	×
	DI	－	4	2	2	2	×
	DISPOSE	imm5, list12	4	n＋1[注5]	n＋1[注5]	n＋1[注5]	×
	DISPOSE	imm5, list12, [reg1]	4	n＋1[注5]	n＋1[注5]	n＋1[注5]	×
	EI	－	4	2	2	2	×
	HALT	－	4	1	1	1	×
	LDSR	reg2, regID	4	2	2	2	×
	NOP	－	2	1	1	1	○（R/L）

表5－6　命令実行クロック数一覧（4/4）

命令の種類	ニモニック	オペランド	バイト	実行クロック数			並列発行[注1]
				issue	repeat	latency	
特殊命令	PREPARE	list12, imm5	4	n＋1[注5]	n＋1[注5]	n＋1[注5]	×
	PREPARE	list12, imm5, sp	4	n＋1[注5]	n＋1[注5]	n＋1[注5]	×
	PREPARE	list12, imm5, imm16	6	n＋1[注5]	n＋1[注5]	n＋1[注5]	×
	PREPARE	list12, imm5, imm32	8	n＋1[注5]	n＋1[注5]	n＋1[注5]	×
	RETI	－	4	不定	不定	不定	×
	STSR	regID, reg2	4	1	1	1	×
	SWITCH	reg1	2	8	8	8	×
	TRAP	vector	4	9	9	9	×
ディバグ機能用命令	DBRET	－	4	不定	不定	不定	×
	DBTRAP	－	2	不定	不定	不定	×
未定義命令コード			4	4	4	4	－

注 1. 「○」は，ほかの命令との並列発行が可能であることを，「×」は，ほかの命令との並列発行が不可能であること（単独で発行）を示します。また，かっこ内は，並列発行を行う際に使用するパイプラインを示します（L：Lpipe, R：Rpipe, B：Bpipe, R/L：Rpipe または Lpipe）。詳細は，第 8 章　パイプラインを参照してください。

2. ウエイト・ステート数による（ウエイト・ステートがない場合は 3）。
3. 直前に PSW レジスタの内容を書き換える命令がある場合は 6。
4. ウエイト・ステートがない場合（4＋リード・アクセス・ウエイト・ステート数）。
5. n は，list12 で指定されるレジスタの合計数（ウエイト・ステート数による。ウエイト・ステートがない場合，n は list12 で指定されるレジスタの合計数。n = 0 の場合は，n = 1 の場合と同じ）。

備考 1. **オペランドの凡例**

略　号	意　味
reg1	**汎用レジスタ（ソース・レジスタとして使用）**
reg2	**汎用レジスタ（主にデスティネーション・レジスタとして使用（一部の命令で，ソース・レジスタとしても使用））**
reg3	**汎用レジスタ（主に除算結果の余り，乗算結果の上位** 32 **ビットを格納）**
bit#3	**ビット・ナンバ指定用** 3 **ビット・データ**
imm ×	× **ビット・イミーディエト・データ**
disp ×	× **ビット・ディスプレースメント・データ**
regID	**システム・レジスタ番号**
vector	**トラップ・ベクタ（**00H-1FH**）を指定する** 5 **ビット・データ**
cccc	**条件コードを示す** 4 **ビット・データ**
sp	**スタック・ポインタ（**r3**）**
ep	**エレメント・ポインタ（**r30**）**
list12	**レジスタ・リスト**

2. **実行クロックの凡例**

略　号	意　味
issue	**命令実行直後に他の命令を実行する場合**
repeat	**命令実行直後に同一命令を繰り返す場合**
latency	**命令実行結果をその命令実行直後の命令で利用する場合**

第6章　割り込みと例外

割り込みは，プログラムの実行とは別に独立に発生する事象で，マスカブル割り込みとノンマスカブル割り込み（NMI）があります。例外は，プログラムの実行に依存して発生する事象で，ソフトウエア例外，例外トラップ，ディバグ・トラップ，およびディバグ・ブレークがあります。

割り込み，または例外が発生した場合には，各要因（割り込み／例外要因）ごとに固定的にアドレスが決まっているハンドラへと制御が移されます。割り込み／例外要因は，割り込み要因レジスタ（ECR）に格納される例外コードで知ることができます。各ハンドラでは ECR レジスタを解析し，適切な割り込み，または例外処理を行います。復帰 PC，復帰 PSW は，各種の状態退避レジスタに書き込まれます。

割り込み処理，ソフトウエア例外からの復帰は，RETI 命令により行われます。また，例外トラップ，ディバグ・トラップ，およびディバグ・ブレークからの復帰は，DBRET 命令により行われます。復帰処理では，状態退避レジスタから復帰 PC，復帰 PSW を取り出し，復帰 PC へ制御を移します。

表6－1　割り込み，例外コード一覧

割り込み／例外要因		分　類	例外コード	ハンドラ・アドレス	復帰 PC
名　称	発生要因				
ノンマスカブル割り込み（NMI）	NMI0 入力	割り込み	0010H	00000010H	next PC 注1
	NMI1 入力	割り込み	0020H	00000020H	next PC 注1, 2
	NMI2 入力注3	割り込み	0030H	00000030H	next PC 注1, 2
マスカブル割り込み	注4	割り込み	注4	注5	next PC 注2
ソフトウエア例外 TRAP0n（n = 0-FH）	TRAP 命令	例外	004nH	00000040H	next PC
ソフトウエア例外 TRAP1n（n = 0-FH）	TRAP 命令	例外	005nH	00000050H	next PC
例外トラップ（ILGOP）	不正命令コード	例外	0060H	00000060H	next PC 注6
ディバグ・トラップ	DBTRAP 命令	例外	0060H	00000060H	next PC
ディバグ・ブレーク	ディバグ・ブレーク	例外	0060H	00000060H	next PC

注 1. 次の命令の実行中に割り込みを受け付けた場合を除く（命令実行中に割り込みを受け付けると実行を中止し，割り込み処理完了後に再実行されます。この場合，中断された命令のアドレスが復帰 PC となります）。
- 除算命令（DIV, DIVH, DIVU, DIVHU）
- PREPARE，DISPOSE 命令（スタック・ポインタの更新前に割り込みが発生した場合のみ）

2. RETI 命令による復帰はできません。割り込み処理後にシステム・リセットを行ってください。
3. PSW レジスタの NP フラグがセット（1）されていても受け付けられます。
4. 割り込みの種類ごとに異なります。
5. 上位 16 ビットは 0000H，下位 16 ビットは例外コードと同一の値です。
6. 不正命令の実行アドレスは，「復帰 PC – 4」で求められます。

備考 復帰 PC：割り込み，例外処理起動時に，EIPC レジスタまたは FEPC レジスタに退避される PC 値
next PC：割り込み，例外処理後に処理を開始する PC 値

6. 1　割り込み処理

6. 1. 1　マスカブル割り込み

割り込みコントローラ（INTC）の割り込み制御レジスタにより割り込み受け付けをマスクできる割り込みです。

INTC では，受け付けた割り込みの最高位の割り込みに基づき，CPU に対して割り込み要求を発生します。

割り込み要求入力（INT 入力）によりマスカブル割り込みが発生した場合，CPU は次の処理を行い，ハンドラ・ルーチンへ制御を移します。

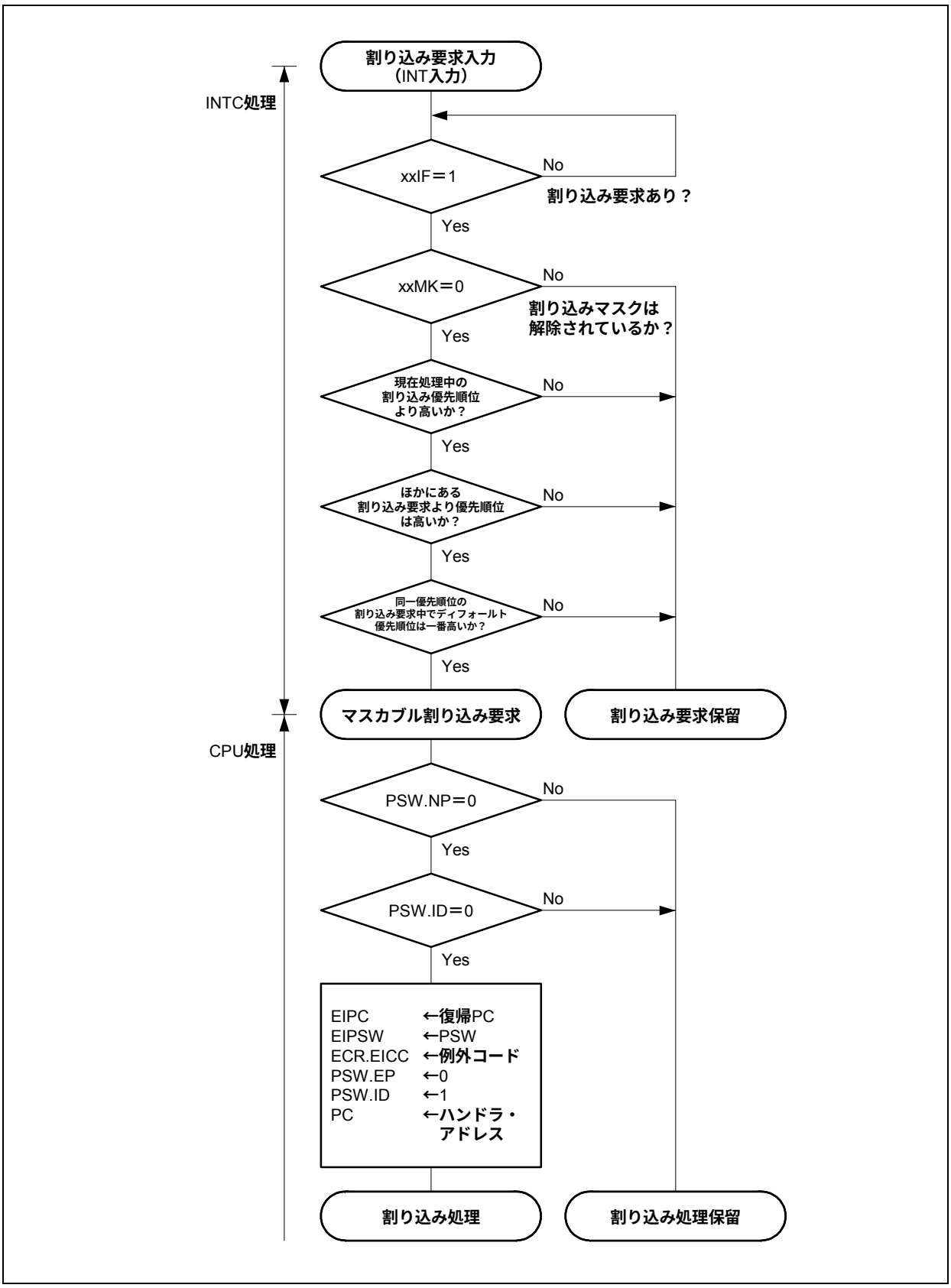
<1>　復帰 PC を EIPC レジスタに退避します。
<2>　現在の PSW を EIPSW レジスタへ退避します。
<3>　ECR の下位ハーフワード（EICC）に例外コードを書き込みます。
<4>　PSW レジスタの ID フラグをセット（1）し，EP フラグをクリア（0）します。
<5>　PSW レジスタの SB フラグの内容を SS フラグに転送します。
<6>　PC レジスタに各割り込みに対するハンドラ・アドレスをセットし，制御を移します。

状態退避レジスタには EIPC，EIPSW レジスタを使用します。なお，INTC においてマスクされている INT 入力や割り込み処理中（PSW レジスタの NP フラグが 1，または PSW レジスタの ID フラグが 1 のとき）に発生した INT 入力は，INTC 内部で保留されます。この場合，マスクを解除するか，LDSR 命令を使用して PSW レジスタの NP フラグと ID フラグを 0 にすると，保留していた INT 入力により，新たなマスカブル割り込み処理が開始されます。

なお，EIPC，EIPSW レジスタは 1 組しかないため，多重割り込みを許可する場合には，プログラムによってこのレジスタを退避する必要があります。

マスカブル割り込みの処理形態を次に示します。

図6－1　マスカブル割り込みの処理形態



6. 1. 2　ノンマスカブル割り込み

ノンマスカブル割り込みは，命令などによる割り込み受け付け禁止ができない常時受け付けが可能な割り込みです。ノンマスカブル割り込みは，NMI 入力により発生します。

ノンマスカブル割り込みが発生した場合は，CPU は次の処理を行いハンドラ・ルーチンへ制御を移します。

<1> 復帰 PC を FEPC レジスタへ退避します。
<2> 現在の PSW を FEPSW レジスタへ退避します。
<3> ECR の上位ハーフワード（FECC）に例外コード（0010H）を書き込みます。
<4> PSW レジスタの NP, ID フラグをセット（1）し，EP フラグをクリア（0）します。
<5> PSW レジスタの SB フラグの内容を SS フラグに転送します。
<6> PC レジスタにノンマスカブル割り込みに対するハンドラ・アドレスをセットし，制御を移します。

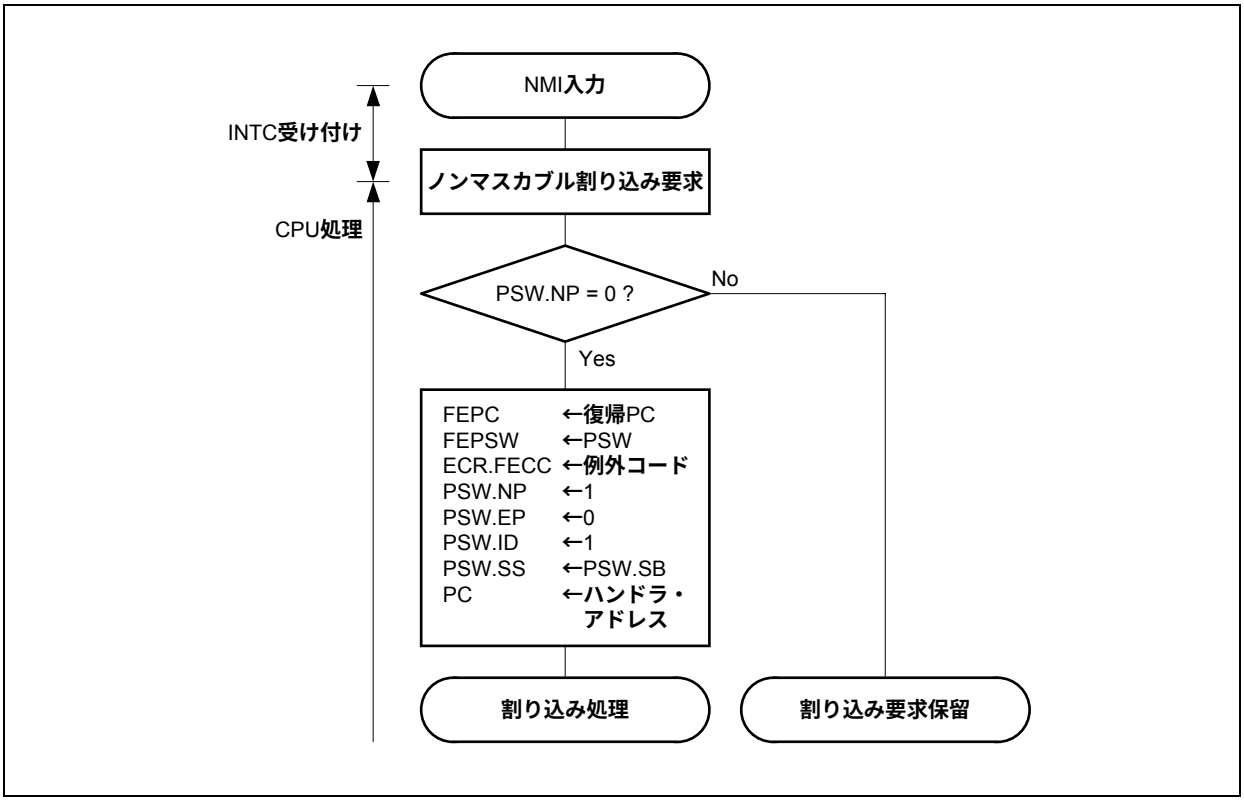
状態退避レジスタには，FEPC, FEPSW レジスタを使用します。

また，ノンマスカブル割り込み処理中（PSW レジスタの NP フラグが 1）に発生したノンマスカブル割り込み要求は，割り込みコントローラ内部で保留されます。この場合，RETI 命令と LDSR 命令を使用して，PSW レジスタの NP フラグを 0 にすると，保留されていたノンマスカブル割り込み要求により新たなノンマスカブル割り込み処理が開始されます。

ただし，NMI2，または DIR レジスタの UTT ビット = 1,1 に設定されたラン・タイム・エラーが発生した場合だけ，NP フラグの値によらず，NMI 処理が実行されます。

ノンマスカブル割り込みの処理形態を次に示します。

図6－2　ノンマスカブル割り込みの処理形態



6. 2　例外処理

6. 2. 1　ソフトウエア例外

ソフトウエア例外は，TRAP 命令の実行により発生する常時受け付けが可能な例外です。

ソフトウエア例外が発生した場合，CPU は次の処理を行いハンドラ・ルーチンへ制御を移します。

<1> 復帰 PC を EIPC レジスタに退避します。

<2> 現在の PSW を EIPSW レジスタへ退避します。

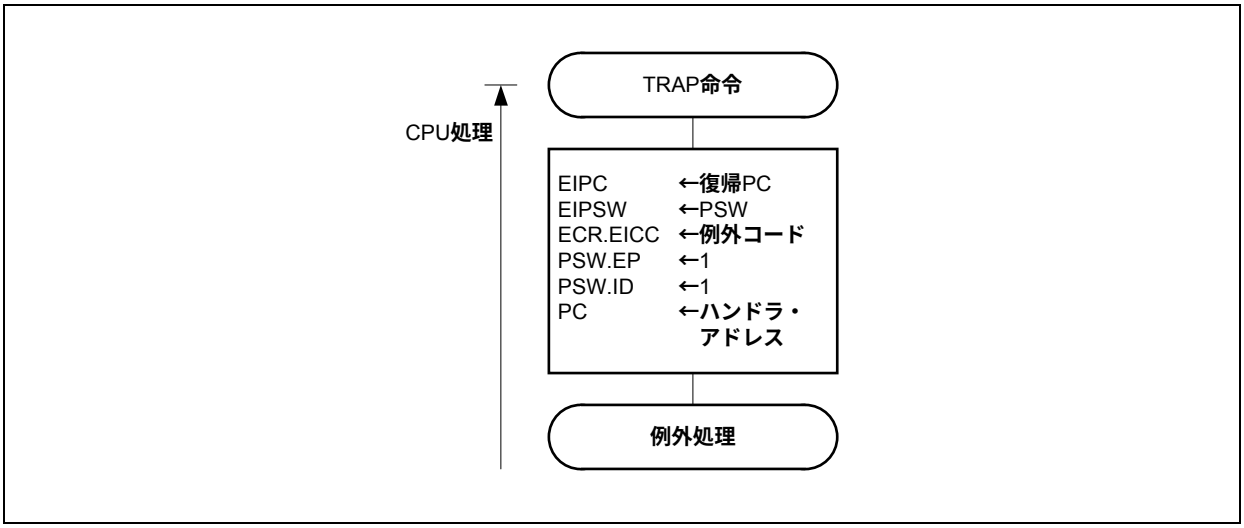
<3> ECR（割り込み要因）の下位 16 ビット（EICC）に例外コードを書き込みます。

<4> PSW レジスタの EP, ID フラグをセット（1）します。

<5> PC レジスタにソフトウエア例外に対するハンドラ・アドレス（00000040H, 00000050H, または 00000070H）をセットし，制御を移します。

ソフトウエア例外の処理形態を次に示します。

図6－3　ソフトウエア例外の処理形態

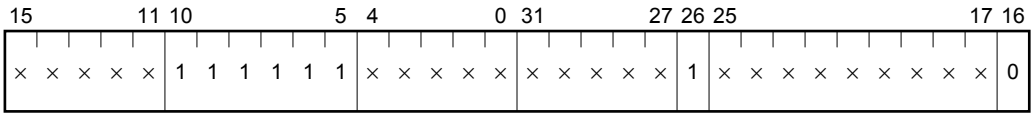


6. 2. 2　例外トラップ

例外トラップは，命令の不正実行が発生した場合に要求される例外です。不正命令コード・トラップ（ILGOP：Illegal opcode trap）が例外トラップにあたります。

不正命令は，命令コードのビット 10-5 がすべて「1」，かつビット 16 が「0」，かつビット 26 が「1」であるものです。この不正命令に当てはまる命令を実行したときに，例外トラップが発生します。

図6－4　不正命令コード

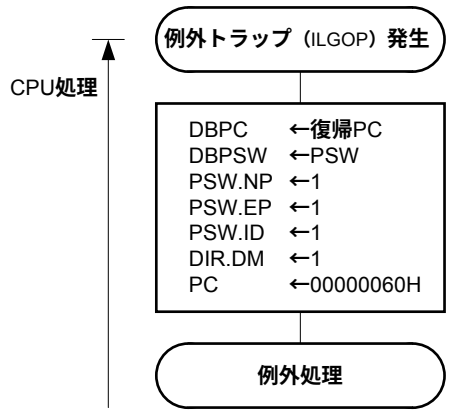


例外トラップが発生した場合，CPU は次の処理を行いハンドラ・ルーチン（ディバグ・モニタ・ルーチン）に制御を移します。

<1> 復帰 PC を DBPC レジスタに退避します。
<2> 現在の PSW を DBPSW レジスタへ退避します。
<3> PSW レジスタの NP, EP, ID フラグをセット（1）します。
<4> DIR レジスタの DM ビットをセット（1）します。
<5> PC レジスタに例外トラップに対するハンドラ・アドレス（00000060H）をセットし，ディバグ・モニタ・ルーチンに制御を移します。

例外トラップの処理形態を次に示します。

図6－5　例外トラップの処理形態



注意　命令，または，不正命令として定義されていない命令を実行したときの動作は保証しません。

備考　不正命令の実行アドレスは，「復帰 PC – 4」で求められます。

6. 2. 3　ディバグ・トラップ，ディバグ・ブレーク

ディバグ・トラップ，ディバグ・ブレークは常時受け付けが可能な例外です。

ディバグ・トラップは，DBTRAP 命令の実行により発生します。

ディバグ・トラップ，ディバグ・ブレークが発生した場合，CPU は次の処理を行いディバグ・モニタ・ルーチンへ制御を移し，ディバグ・モードに移行します。

<1> 復帰 PC を DBPC に退避します。

<2> 現在の PSW を DBPSW レジスタへ退避します。

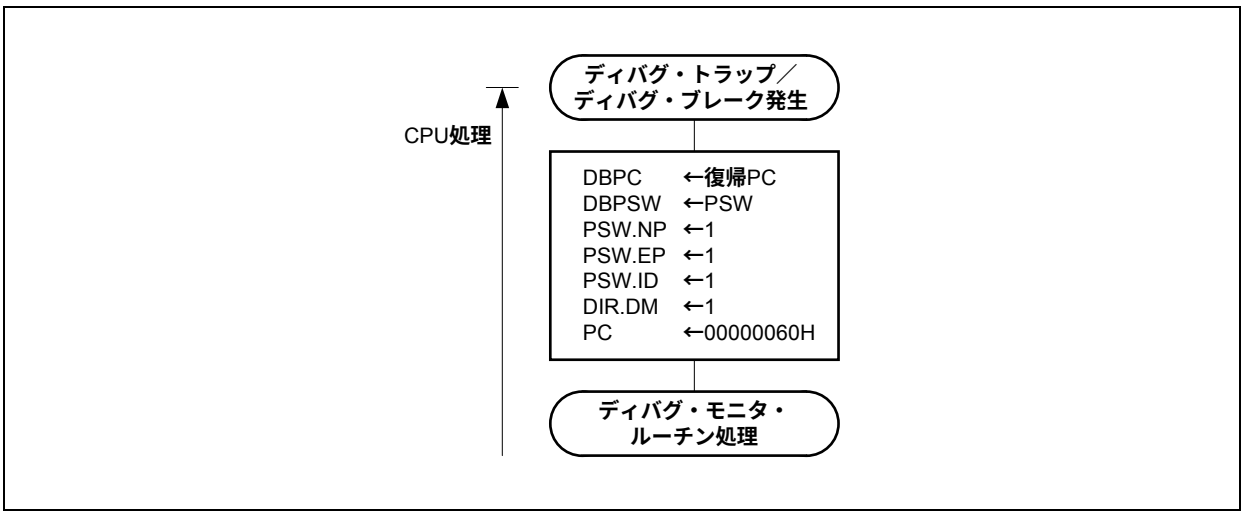
<3> PSW レジスタの NP, EP, ID フラグをセット（1）します。

<4> DIR レジスタの DM ビットをセット（1）します。

<5> PC レジスタにディバグ・トラップ，またはディバグ・ブレークに対するハンドラ・アドレス（00000060H）をセットし，ディバグ・モニタ・ルーチンに制御を移します。

ディバグ・トラップ，ディバグ・ブレークの処理形態を次に示します。

図6－6　ディバグ・トラップ，ディバグ・ブレークの処理形態



6. 3　割り込み，例外処理からの復帰

6. 3. 1　割り込み，ソフトウエア例外からの復帰

割り込み，ソフトウエア例外からの復帰は，すべて RETI 命令により行われます。

RETI 命令の実行により，CPU は次の処理を行い復帰 PC のアドレスへ制御を移します。

<1>　PSW レジスタの EP フラグが 0，かつ NP フラグが 1 の場合，FEPC, FEPSW レジスタから復帰 PC, PSW を取り出します。それ以外の場合，EIPC, EIPSW レジスタから復帰 PC, PSW を取り出します。

<2>　取り出した復帰 PC, PSW のアドレスに制御を移します。

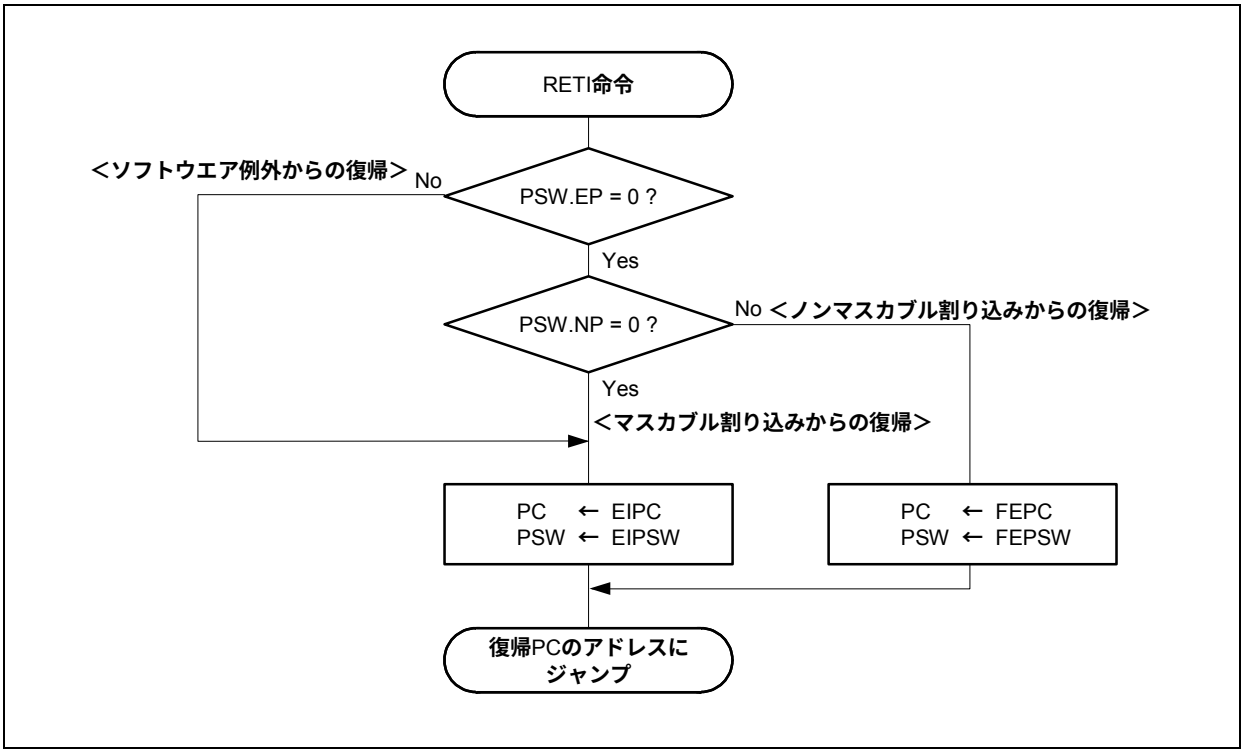
各割り込み処理からの復帰時は，PC, PSW レジスタを正常にリストアするために，RETI 命令の直前で，LDSR 命令を使用し，PSW レジスタの NP フラグ，EP フラグの各フラグを次の状態にしておく必要があります。

- ノンマスカブル割り込み処理からの復帰時[注]　：PSW レジスタの NP フラグ= 1，EP フラグ = 0
- マスカブル割り込み処理からの復帰時　：PSW レジスタの NP フラグ = 0，EP フラグ = 0
- 例外処理からの復帰時　：PSW レジスタの EP フラグ = 1

注　NMI1, NMI2 は，RETI 命令による復帰はできません。割り込み処理後にシステム・リセットを行ってください。また，NMI2 は PSW レジスタの NP フラグがセット（1）されていても受け付けられます。

割り込み，例外処理からの復帰の処理形態を次に示します。

図6－7　割り込み，ソフトウエア例外からの復帰の処理形態



6. 3. 2　例外トラップ，ディバグ・トラップ，ディバグ・ブレークからの復帰

例外トラップ，ディバグ・トラップ，ディバグ・ブレークからの復帰は，DBRET 命令により行われます。

DBRET 命令の実行により，CPU は次の処理を行い復帰 PC のアドレスへ制御を移します。

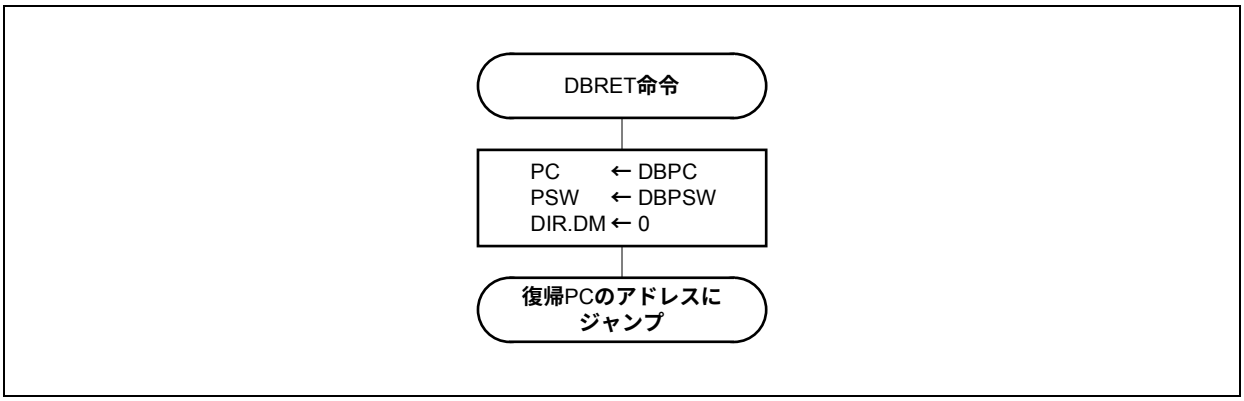
<1> DBPC, DBPSW レジスタから復帰 PC, PSW を取り出します。

<2> 取り出した復帰 PC, PSW のアドレスに制御を移します。

<3> DIR レジスタの DM ビットをクリア（0）します。

例外トラップ，ディバグ・トラップ，ディバグ・ブレークからの復帰の処理形態を次に示します。

図6－8　例外トラップ，ディバグ・トラップ，ディバグ・ブレークからの復帰の処理形態



第7章　リセット

7.1　リセット後のレジスタの状態

リセット端子にロウ・レベルが入力されると，システム・リセットがかかり，プログラム・レジスタとシステム・レジスタは，表7－1に示す状態になります。リセット端子への入力がハイ・レベルになるとリセット状態が解除され，プログラムの実行を開始します。各レジスタの内容は，プログラムの中で必要に応じてイニシャライズしてください。

表7－1　リセット後のレジスタの状態（1/2）

レジスタ		リセット後の状態（初期値）
プログラム・レジスタ	汎用レジスタ（r0）	00000000H（固定）
	汎用レジスタ（r1-r31）	不定
	プログラム・カウンタ（PC）	00000000H
システム・レジスタ	割り込み時状態退避レジスタ（EIPC）	000x xxxx xxxx xxxx xxxx xxxx xxxx xxx0B
	割り込み時状態退避レジスタ（EIPSW）	0000 0000 0000 0000 0000 x00x xxxx xxxxB
	NMI時状態退避レジスタ（FEPC）	000x xxxx xxxx xxxx xxxx xxxx xxxx xxx0B
	NMI時状態退避レジスタ（FEPSW）	0000 0000 0000 0000 0000 x00x xxxx xxxxB
	割り込み要因レジスタ（ECR）	00000000H
	プログラム・ステータス・ワード（PSW）	00000020H
	CALLT実行時状態退避レジスタ（CTPC）	000x xxxx xxxx xxxx xxxx xxxx xxxx xxx0B
	CALLT実行時状態退避レジスタ（CTPSW）	0000 0000 0000 0000 0000 x00x xxxx xxxxB
	例外／ディバグ・トラップ時状態退避レジスタ（DBPC）	000x xxxx xxxx xxxx xxxx xxxx xxxx xxxxB
	例外／ディバグ・トラップ時状態退避レジスタ（DBPSW）	0000 0000 0000 0000 0000 x00x xxxx xxxxB
	CALLTベース・ポインタ（CTBP）	000x xxxx xxxx xxxx xxxx xxxx xxxx xxx0B
	ディバグ・インタフェース・レジスタ（DIR）	00000040H
	ブレークポイント制御レジスタ0（BPC0）	0000 0000 xxxx xxxx x000 xxxx x000 0000B
	ブレークポイント制御レジスタ1（BPC1）	
	ブレークポイント制御レジスタ2（BPC2）	
	ブレークポイント制御レジスタ3（BPC3）	

表7－1　リセット後のレジスタの状態（2/2）

レジスタ		リセット後の状態（初期値）
システム・レジスタ	プログラムIDレジスタ（ASID）	000000xxH
	ブレークポイント・アドレス設定レジスタ0（BPAV0）	000x xxxx xxxx xxxx xxxx xxxx xxxx xxxxB
	ブレークポイント・アドレス設定レジスタ1（BPAV1）	
	ブレークポイント・アドレス設定レジスタ2（BPAV2）	
	ブレークポイント・アドレス設定レジスタ3（BPAV3）	
	ブレークポイント・アドレス・マスク・レジスタ0（BPAM0）	000x xxxx xxxx xxxx xxxx xxxx xxxx xxxxB
	ブレークポイント・アドレス・マスク・レジスタ1（BPAM1）	
	ブレークポイント・アドレス・マスク・レジスタ2（BPAM2）	
	ブレークポイント・アドレス・マスク・レジスタ3（BPAM3）	
	ブレークポイント・データ設定レジスタ0（BPDV0）	不定
	ブレークポイント・データ設定レジスタ1（BPDV1）	
	ブレークポイント・データ設定レジスタ2（BPDV2）	
	ブレークポイント・データ設定レジスタ3（BPDV3）	
	ブレークポイント・データ・マスク・レジスタ0（BPDM0）	不定
	ブレークポイント・データ・マスク・レジスタ1（BPDM1）	
	ブレークポイント・データ・マスク・レジスタ2（BPDM2）	
	ブレークポイント・データ・マスク・レジスタ3（BPDM3）	

備考　x：不定

7.2 起　　動

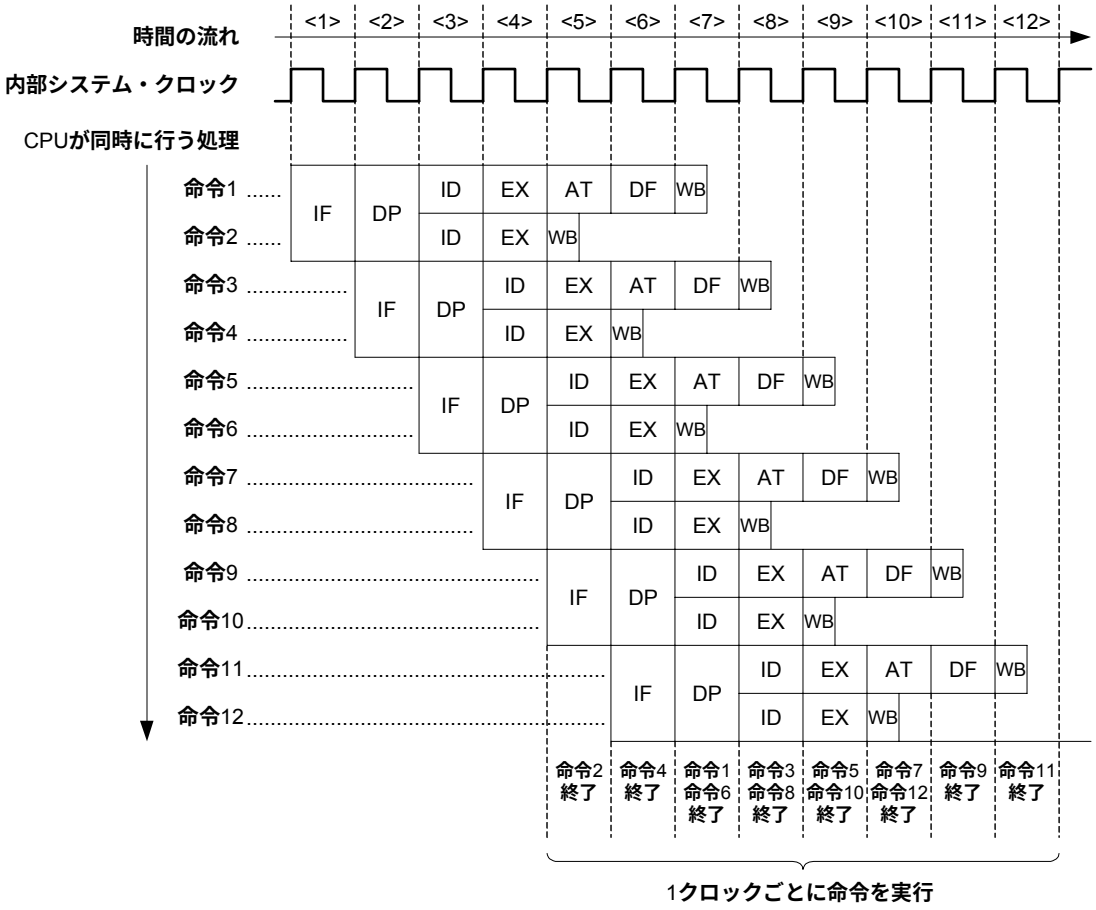
CPU は，リセットにより 00000000H 番地からプログラムの実行を開始します。

なお，リセット直後は，割り込み要求は受け付けられません。プログラムで割り込み処理を使用する場合は，PSW レジスタの ID フラグをクリア（0）してください。

第 8 章　パイプライン

V850E2 CPU は，RISC アーキテクチャをベースとし，7 段パイプラインの制御によりほとんどの命令を 1 クロックで実行します。命令実行手順は，通常，インストラクション・フェッチ（IF）からライトバック（WB）までの 7 つのステージで構成されています。各ステージの実行時間は，命令の種類やアクセスの対象となるメモリの種類などによって異なります。パイプラインの動作例として，標準的な命令を 12 個続けて実行したときの CPU の処理を図 8－1 に示します。

図8－1　標準的な命令を12個続けて実行する例



IF（インストラクション・フェッチ）　：命令のフェッチを行い，フェッチ・ポインタをインクリメントします。

DP（ディスパッチ）　：命令の種類，依存関係を調べ，対応するパイプラインに発行します。

ID（インストラクション・デコード）　：命令をデコードし，イミーディエト・データの作成とレジスタの読み出しを行います。

EX（ALU，乗算器，バレル・シフタの実行）　：デコードした命令を実行します。

AT（アドレス転送）　：対象となるメモリにアドレスを転送します。

DF（データ・フェッチ）　：対象となるメモリからデータを読み出します。

WB（ライトバック）　：実行した結果をレジスタに書き込みます。

<1> - <12>は，CPU のステートを示します。標準的な命令では，1 クロックに 2 つの命令の実行（EX）が並列に行えます。

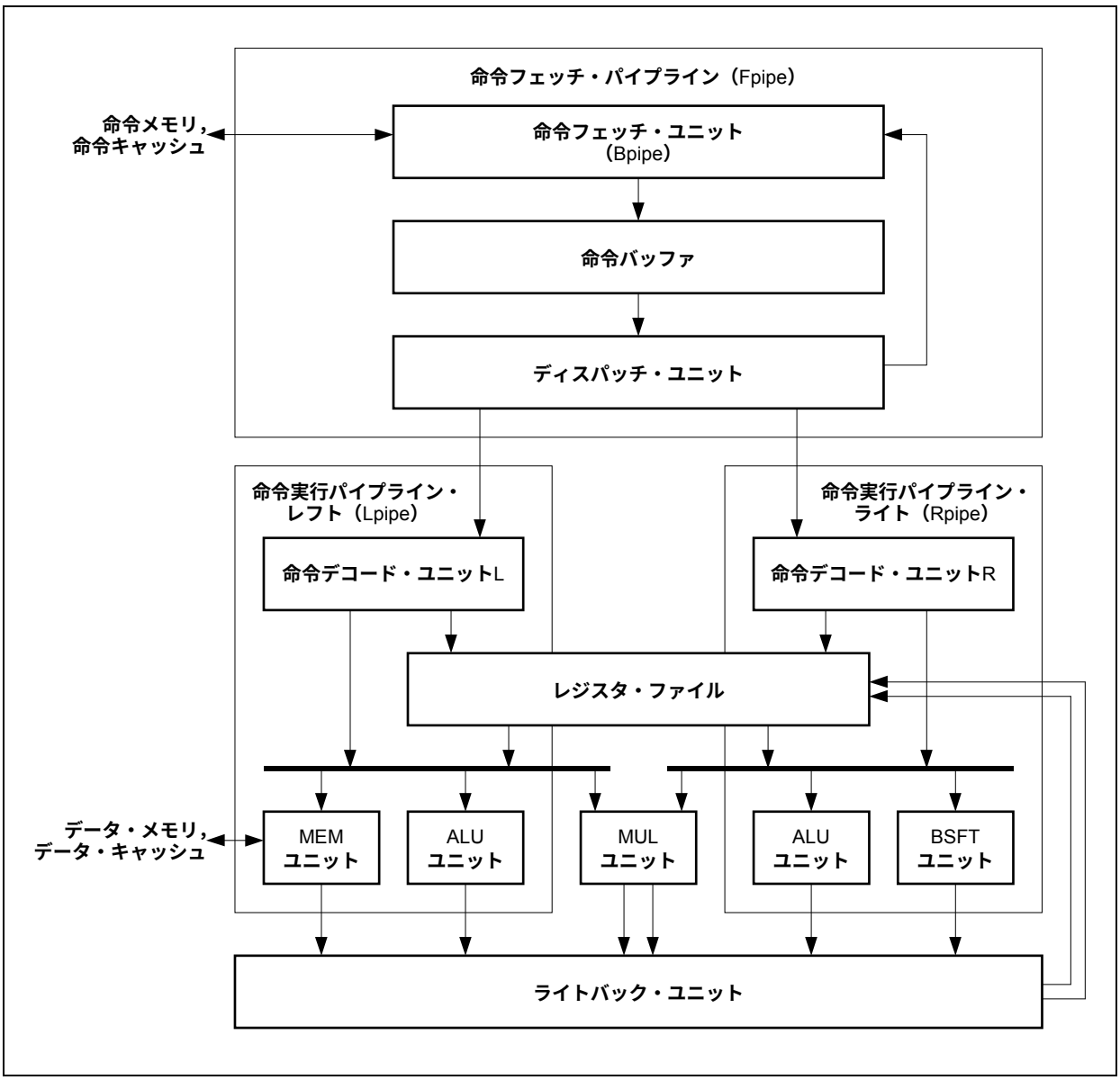
8.1 特　　徴

V850E2 CPU は，次に示す独立した 3 つのパイプラインで構成されます。

- 命令フェッチ・パイプライン（Fpipe）
- 命令実行パイプライン・レフト（Lpipe）
- 命令実行パイプライン・ライト（Rpipe）

V850E2 CPU は，命令の依存関係を検出し，最大で 2 つの命令を同時に発行可能な構成になっています。V850E2 CPU のパイプライン構成を図 8－2 に示します。

図8－2　パイプライン構成



(1) 命令フェッチ・パイプライン（Fpipe）

次に示す 3 つのユニットで構成されています。

(a) 命令フェッチ・ユニット（Bpipe）

128 ビットのフェッチ・バス（iLB）から最大 8 命令（1 命令が 16 ビットの場合）を 1 サイクルでフェッチします。

(b) ディスパッチ・ユニット

128 ビット×2 段の命令キューを内蔵しており，このキューで命令の依存関係を検出し，最大で 2 つの命令を効率良く命令実行パイプラインに発行します。

(c) 命令バッファ

命令フェッチ・ユニット（Bpipe）によってフェッチされた命令を格納します。

(2) 命令実行パイプライン・レフト（Lpipe）

次に示す 3 つのユニットで構成されています。

(a) 命令デコード・ユニット L

ディスパッチ・ユニットから発行された命令をデコードします。

(b) ALU ユニット

整数演算，論理演算を行う命令を実行します。

(c) MEM ユニット

ロード命令，ストア命令を含むメモリ・アクセスを行う命令を実行します。

(3) 命令実行パイプライン・ライト（Rpipe）

次に示す 3 つのユニットで構成されています。

(a) 命令デコード・ユニット R

ディスパッチ・ユニットから発行された命令をデコードします。

(b) ALU ユニット

整数演算，論理演算を行う命令を実行します。

(c) BSFT ユニット

データ操作を行う命令を実行します。

(4) MUL ユニット

整数乗算を行う命令を実行します。

(5) ライトバック・ユニット

レジスタ・ファイルにライトバックする制御をします。

8. 2　各命令実行時のパイプラインの流れ

この節では各命令実行時のパイプラインの流れについて説明します。

8. 2. 1　ロード命令

ロード命令は，命令実行パイプライン・レフト（Lpipe）の MEM ユニットで実行されます。

［対象の命令］　LD.B, LD.BU, LD.H, LD.HU, LD.W, SLD.B, SLD.BU, SLD.H, SLD.HU, SLD.W

［パイプライン］

	<1>	<2>	<3>	<4>	<5>	<6>	<7>	<8>
ロード命令	IF	DP	ID	EX	AT	DF	WB	
次命令		IF	DP	ID	EX	AT	DF	WB

［説　　明］　パイプラインは IF, DP, ID, EX, AT, DF, WB の 7 ステージです。この図では，Lpipe でロード命令が実行され，Lpipe に次命令が発行された場合の動作を示しています。命令実行パイプライン・ライト（Rpipe）はロード命令と依存関係がないかぎり独立に処理を実行します。ロード命令の直後に，実行結果を使用する命令を配置すると，データの待ち合わせ時間が発生します。

各命令は，ほかの命令との並列発行が可能です。

8. 2. 2　ストア命令

ストア命令は，命令実行パイプライン・レフト（Lpipe）の MEM ユニットで実行されます。

［対象の命令］　ST.B, ST.H, ST.W, SST.B, SST.H, SST.W

［パイプライン］

	<1>	<2>	<3>	<4>	<5>	<6>	<7>	<8>
ストア命令	IF	DP	ID	EX	AT	DF	WB	
次命令		IF	DP	ID	EX	AT	DF	WB

［説　　明］　パイプラインは IF, DP, ID, EX, AT, DF, WB の 7 ステージですが，レジスタへのデータの書き込みがないので WB ステージでは何も行いません。

この図では，Lpipe でストア命令が実行され，Lpipe に次命令が発行された場合の動作を示しています。命令実行パイプライン・ライト（Rpipe）はストア命令と依存関係がないかぎり独立に処理を実行します。

各命令は，ほかの命令との並列発行が可能です。

8. 2. 3　乗算命令

乗算命令は，命令実行パイプライン・レフト（Lpipe）の MUL ユニットで実行されます。

［対象の命令］　MUL, MULH, MULHI, MULU

［パイプライン］（a）次命令が乗算命令（または，加算付き乗算命令）以外の場合

	<1>	<2>	<3>	<4>	<5>	<6>	<7>	<8>
乗算命令	IF	DP	ID	EX1	EX2	DF	WB	
次命令		IF	DP	ID	EX	AT	DF	WB

（b）次命令が乗算命令（または，加算付き乗算命令）の場合

	<1>	<2>	<3>	<4>	<5>	<6>	<7>	<8>
乗算命令1	IF	DP	ID	EX1	EX2	DF	WB	
乗算命令2		IF	DP	ID	EX1	EX2	DF	WB

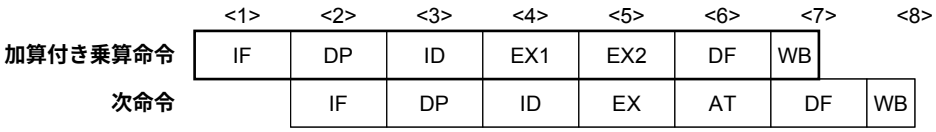
［説　　明］　**パイプラインは** IF, DP, ID, EX, AT, DF, WB **の** 7 **ステージです。**EX **ステージは** 2 **クロックかかりますが，**EX1 **と** EX2 **は独立して動作できます。したがって，乗算命令（または，加算付き乗算命令）を繰り返しても命令実行クロック数は** 1 **クロックとなります。**
この図では，Lpipe **で乗算命令が実行され，**Lpipe **に次命令が発行された場合の動作を示しています。命令実行パイプライン・ライト（**Rpipe**）は乗算命令と依存関係がないかぎり独立に処理を実行します。ただし，乗算命令の直後に実行結果を使用する命令を配置すると，データの待ち合わせ時間が発生します。**
各命令は，ほかの命令との並列発行が可能です。

8. 2. 4　加算付き乗算命令

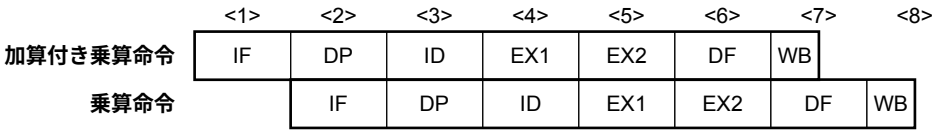
加算付き乗算命令は，命令実行パイプライン・レフト（Lpipe）の MUL ユニットで実行されます。

［対象の命令］　MAC, MACU

［パイプライン］（a）次命令が乗算命令（または，加算付き乗算命令）以外の場合



（b）次命令が乗算命令（または，加算付き乗算命令）の場合



［説　　明］　パイプラインは IF, DP, ID, EX, AT, DF, WB の 7 ステージです。EX ステージは 2 クロックかかりますが，EX1 と EX2 は独立して動作できます。したがって，乗算命令（または，加算付き乗算命令）を繰り返しても命令実行クロック数は 1 クロックとなります。

この図では，Lpipe で乗算命令が実行され，Lpipe に次命令が発行された場合の動作を示しています。命令実行パイプライン・ライト（Rpipe）は乗算命令と依存関係がないかぎり独立に処理を実行します。ただし，乗算命令の直後に実行結果を使用する命令を配置すると，データの待ち合わせ時間が発生します。

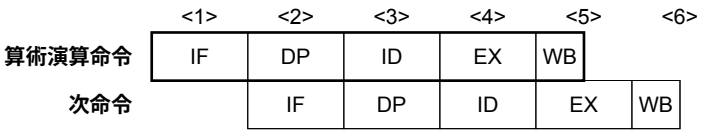
各命令は単独で発行されます。

8. 2. 5　算術演算命令

算術演算命令は，命令実行パイプライン・レフトまたはライト（Lpipe または Rpipe）の ALU ユニットで実行されます。

［対象の命令］　ADD, ADDI, CMP, MOV, MOVEA, MOVHI, SUB, SUBR

［パイプライン］



［説　　明］　パイプラインは IF, DP, ID, EX, WB の 5 ステージです。

この図では，Rpipe で算術演算命令が実行され，Rpipe に次命令が発行された場合の動作を示しています。Lpipe は算術演算命令と依存関係がないかぎり独立に処理を実行します。

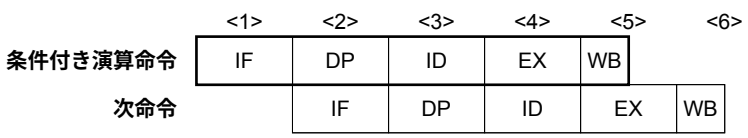
MOV　imm32, reg1 命令を除く各命令は，ほかの命令との並列発行が可能です（MOV imm32, reg1 命令は，単独で発行されます）。

8. 2. 6　条件付き演算命令

条件付き演算命令は，命令実行パイプライン・レフトまたはライト（Lpipe または Rpipe）の ALU ユニットで実行されます。

［対象の命令］　ADF, SBF

［パイプライン］



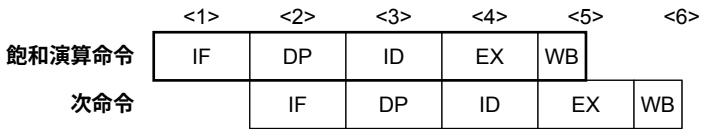
［説　　明］　パイプラインは IF, DP, ID, EX, WB の 5 ステージです。
この図では，Rpipe で算術演算命令が実行され，Rpipe に次命令が発行された場合の動作を示しています。Lpipe は算術演算命令と依存関係がないかぎり独立に処理を実行します。
各命令は単独で発行されます。

8. 2. 7　飽和演算命令

飽和演算命令は，命令実行パイプライン・レフトまたはライト（Lpipe または Rpipe）の ALU ユニットで実行されます。

［対象の命令］　SATADD, SATSUB, SATSUBI, SATSUBR

［パイプライン］



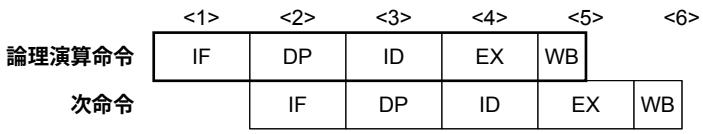
［説　　明］　パイプラインは IF, DP, ID, EX, WB の 5 ステージです。
この図では，Rpipe で飽和演算命令が実行され，Rpipe に次命令が発行された場合の動作を示しています。Lpipe は飽和演算命令と依存関係がないかぎり独立に処理を実行します。
SATADD　reg1, reg2, reg3 命令と SATSUB　reg1, reg2, reg3 命令を除く各命令は，ほかの命令との並列発行が可能です（SATADD　reg1, reg2, reg3 命令と SATSUB　reg1, reg2, reg3 命令は，単独で発行されます）。

8. 2. 8　論理演算命令

論理演算命令は命令実行パイプライン・レフトまたはライト（Lpipe または Rpipe）の ALU ユニットで実行されます。

［対象の命令］　AND, ANDI, NOT, OR, ORI, TST, XOR, XORI

［パイプライン］



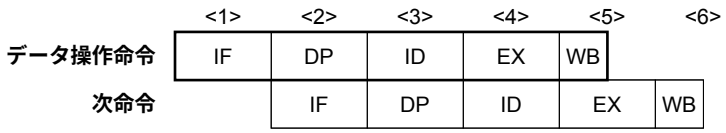
［説　　明］　パイプラインは IF, DP, ID, EX, WB の 5 ステージです。
この図では，Rpipe で論理演算命令が実行され，Rpipe に次命令が発行された場合の動作を示しています。Lpipe は論理演算命令と依存関係がないかぎり独立に処理を実行します。
各命令は，ほかの命令との並列発行が可能です。

8. 2. 9　データ操作命令

データ操作命令は，命令実行パイプライン・ライト（Rpipe）の BSFT ユニットで実行されます。

［対象の命令］　BSH, BSW, CMOV, HSH, HSW, SAR, SASF, SETF, SHL, SHR, SXB, SXH, ZXB, ZXH

［パイプライン］



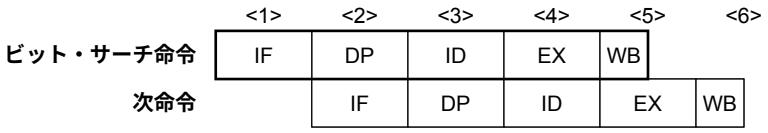
［説　　明］　パイプラインは IF, DP, ID, EX, WB の 5 ステージです。
この図では，Rpipe でデータ操作命令が実行され，Rpipe に次命令が発行された場合の動作を示しています。命令実行パイプライン・レフト（Lpipe）はデータ操作命令と依存関係がないかぎり独立に処理を実行します。
各命令は，ほかの命令との並列発行が可能です。

8. 2. 10　ビット・サーチ命令

ビット・サーチ命令は，命令実行パイプライン・ライト（Rpipe）の BSFT ユニットで実行されます。

［対象の命令］　SCH0L, SCH0R, SCH1L, SCH1R

［パイプライン］



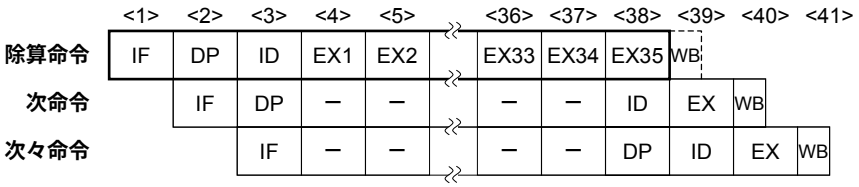
［説　　明］　パイプラインは IF, DP, ID, EX, WB の 5 ステージです。
この図では，Rpipe でデータ操作命令が実行され，Rpipe に次命令が発行された場合の動作を示しています。命令実行パイプライン・レフト（Lpipe）はデータ操作命令と依存関係がないかぎり独立に処理を実行します。
各命令は，ほかの命令との並列発行が可能です。

8. 2. 11　除算命令

除算命令は，命令実行パイプライン・ライト（Rpipe）の ALU ユニットで実行されます。

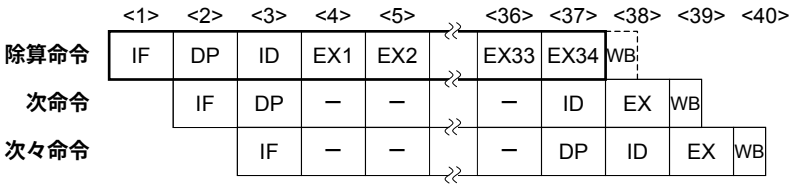
［対象の命令］　DIV, DIVH, DIVHU, DIVU

［パイプライン］（a）DIV, DIVH の場合



ー：待ちあわせのために挿入されるアイドル

（b）DIVU, DIVHU の場合



ー：待ちあわせのために挿入されるアイドル

［説　　明］　パイプラインは DIV, DIVH 命令の場合，IF, DP, ID, EX1-EX35, WB の 39 ステージ，DIVU, DIVHU 命令の場合，IF, DP, ID, EX1-EX34, WB の 38 ステージです。
この図では，Rpipe で除算命令が実行され，Rpipe に次命令が発行された場合の動作を示しています。
ただし，除算命令が ID ステージで命令をデコードしている期間と EX ステージで命令を実行している期間は，ディスパッチ・ユニットは Rpipe に命令を発行しません。
各命令は単独で発行されます。

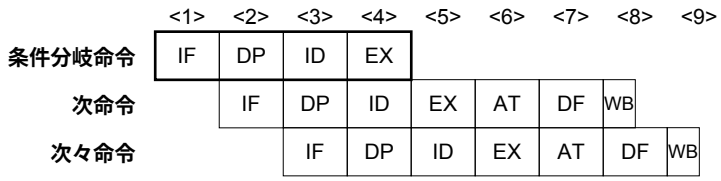
8. 2. 12　分岐命令

分岐命令は，命令フェッチ・ユニット（Bpipe）で実行されます。

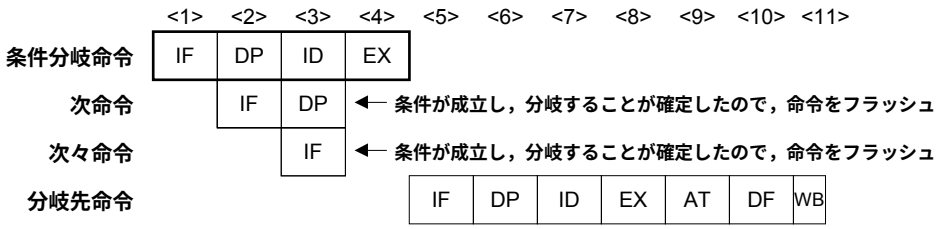
(1) 条件分岐命令（BR 命令を除く）

[対象の命令]　　Bcond 命令

[パイプライン]　(a) 条件が成立しない場合



(b) 条件が成立した場合



[説　　明]　この図では，Bpipe で Bcond 命令が実行され，命令実行パイプライン・レフト（Lpipe）ですべての命令が実行された場合の動作を示しています。
各命令は，ほかの命令との並列発行が可能です。
また，実行クロック数は次のとおりです。

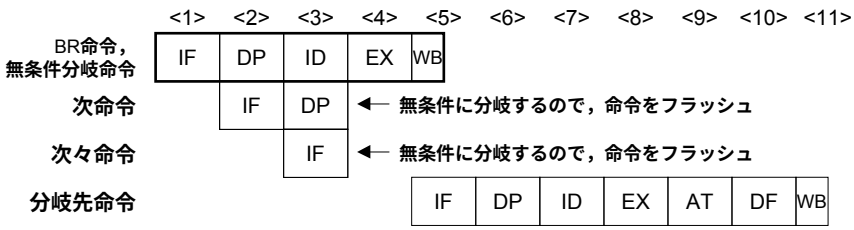
分岐命令	実行クロック数
(a) 条件が成立しない場合	1
(b) 条件が成立した場合	4 注

注　命令バッファにターゲットの命令が存在していた場合は 3（4-1 = 3）
直前に PSW レジスタを書き換える命令がある場合は 6

(2) BR 命令，無条件分岐命令（JMP 命令を除く）

［対象の命令］　BR, JARL, JR 命令

［パイプライン］



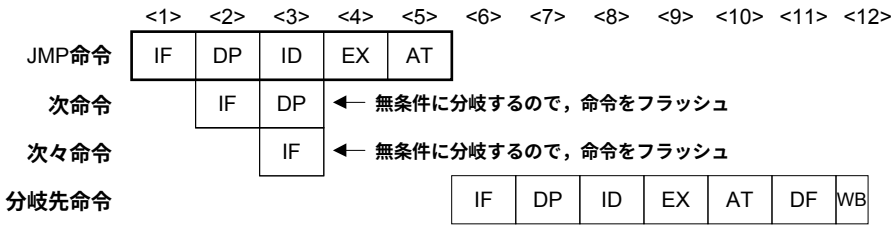
［説　　明］　この図では，Bpipe で分岐命令が実行され，命令実行パイプライン・レフト（Lpipe）ですべての命令が実行された場合の動作を示しています。

JARL disp32, reg1 命令と JR disp32 命令以外の各命令は，ほかの命令との並列発行が可能です。

また，実行クロック数は 4（命令バッファにターゲットの命令が存在していた場合は 3（4-1 = 3））です。

(3) JMP 命令

［パイプライン］



［説　　明］　この図では，Bpipe で JMP 命令が実行され，命令実行パイプライン・レフト（Lpipe）ですべての命令が実行された場合の動作を示しています。

JMP [reg1] 命令は，ほかの命令との並列発行が可能です（JMP disp32 [reg1] 命令は不可）。

また，実行クロック数は 5 です（命令バッファにターゲットの命令が存在していた場合は 4（5-1 = 4））。

8. 2. 13　ビット操作命令

ビット操作命令は，命令実行パイプライン・レフト（Lpipe）の ALU ユニットで実行されます。

（1）CLR1, NOT1, SET1 命令

［パイプライン］

－：待ち合わせのために挿入されるアイドル

［説　　明］　ID ステージで 2 つの命令に分割され，最初にロード命令を，次にビット操作を含むストア命令を実行します。ただし，レジスタへのデータの書き込みがないので，WB ステージでは何も行いません。この図では，Lpipe でビット操作命令が実行され，Lpipe に次命令が発行された場合の動作を示しています。命令実行パイプライン・ライト（Rpipe）はビット操作命令と依存関係がないかぎり独立に処理を実行します。ID ステージで命令をデコードしている期間はディスパッチ・ユニットは Lpipe には命令を発行しません。

各命令は単独で発行されます。

（2）TST1 命令

［パイプライン］

－：待ち合わせのために挿入されるアイドル

［説　　明］　ID ステージで 2 つの命令に分割され，最初にロード命令を，次にビット操作命令を実行します。ただし，レジスタへのデータの書き込みがないので，WB ステージでは何も行いません。この図では，Lpipe で TST1 命令が実行され，Lpipe に次命令が発行された場合の動作を示しています。Rpipe はビット操作命令と依存関係がないかぎり独立に処理を実行します。ID ステージで命令をデコードしている期間はディスパッチ・ユニットは Lpipe には命令を発行しません。

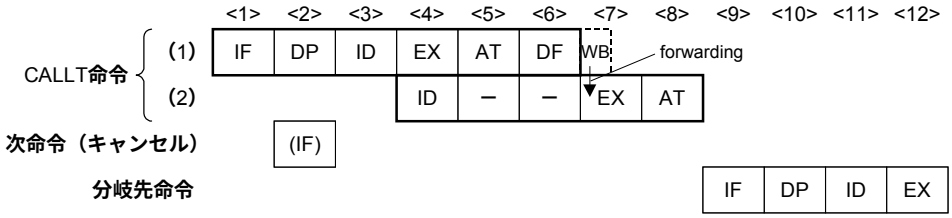
各命令は単独で発行されます。

8. 2. 14　特殊命令

(1) CALLT **命令**

CALLT **命令は，命令実行パイプライン・レフト（**Lpipe**）の** ALU **ユニットで実行されます。**

[パイプライン]



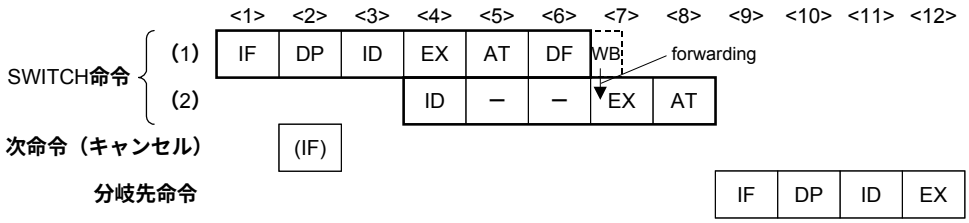
－　：待ち合わせのために挿入されるアイドル
(IF)**：無効となる命令フェッチ**

[説　　明]　ID **ステージで** 2 **つの命令に分割され，最初にロード命令を，次に** CTBP **相対の分岐命令を実行します。ただし，レジスタへのデータの書き込みがないので，**WB **ステージでは何も行いません。この図では，**Lpipe **で** CALLT **命令が実行され，分岐先から命令をフェッチして実行するまでの動作を示しています。命令実行パイプライン・ライト（**Rpipe**）は** CALLT **命令と依存関係がないかぎり独立に処理を実行します。**
この命令は単独で発行されます。
また，実行クロック数は 8 **です。**

(2) SWITCH **命令**

SWITCH **命令は，命令実行パイプライン・レフト（**Lpipe**）の** ALU **ユニットで実行されます。**

[パイプライン]



－　：待ち合わせのために挿入されるアイドル
(IF)**：無効となる命令フェッチ**

[説　　明]　ID **ステージで** 2 **つの命令に分割され，最初にロード命令を，次に** PC **相対の分岐命令を実行します。ただし，レジスタへのデータの書き込みがないので，**WB **ステージでは何も行いません。この図では，**Lpipe **で** SWITCH **命令が実行され，**Lpipe **に次命令が発行された場合の動作を示しています。命令実行パイプライン・ライト（**Rpipe**）は** SWITCH **命令と依存関係がないかぎり独立に処理を実行します。**
この命令は単独で発行されます。
また，実行クロック数は 8 **です。**

(3) DI, EI, LDSR 命令

DI, EI, LDSR 命令は，命令実行パイプライン・ライト（Rpipe）の ALU ユニットで実行されます。

[パイプライン]

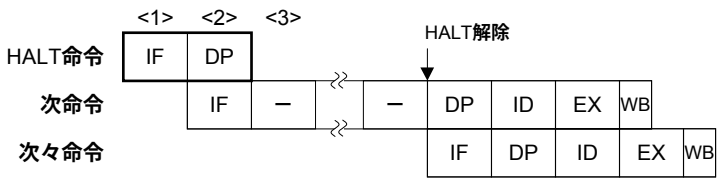
– ：待ち合わせのために挿入されるアイドル

[説　　明]　パイプラインは IF, DP, ID, EX, WB の 5 ステージです。
この図では，Rpipe で DI, EI, LDSR 命令が実行され，Rpipe ですべての命令が実行された場合の動作を示しています。
各命令は単独で発行されます。

(4) HALT 命令

HALT 命令は，命令フェッチ・パイプライン（Fpipe）のディスパッチ・ユニットで実行されます。

[パイプライン]



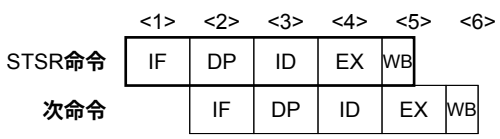
– ：待ち合わせのために挿入されるアイドル

[説　　明]　DP ステージで HALT 命令が検出されると，HALT 命令が解除されるまで，ID ステージへの命令の発行を停止します。したがって，次命令では HALT 命令が解除されるまで ID ステージが遅れます。この図では，命令実行パイプライン・ライト（Rpipe）で HALT 命令が実行され，Rpipe に次命令が発行された場合の動作を示しています。
この命令は単独で発行されます。

(5) STSR 命令

STSR 命令は，命令実行パイプライン・ライト（Rpipe）の ALU ユニットで実行されます。

[パイプライン]

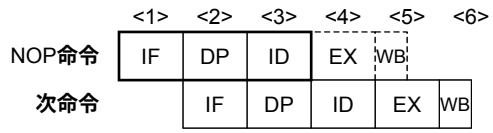


[説　　明]　パイプラインは IF, DP, ID, EX, WB の 5 ステージです。
この図では，Rpipe で STSR 命令が実行され，Rpipe に次命令が発行された場合の動作を示しています。命令実行パイプライン・レフト（Lpipe）は STSR 命令と依存関係がないかぎり独立に処理を実行します。
この命令は単独で発行されます。

(6) NOP **命令**

NOP **命令は，命令実行パイプライン・レフトまたはライト（**Lpipe **または** Rpipe**）の** ALU **ユニットで実行されます。**

[パイプライン]

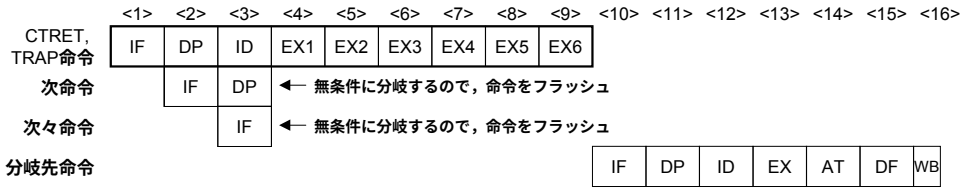


[説　　明]　**パイプラインは** IF, DP, ID, EX, WB **の** 5 **ステージですが，演算，レジスタへのデータの書き込みがないので，**EX, WB **ステージでは何も行いません。**
この図では，Rpipe **で** NOP **命令が実行され，**Rpipe **に次命令が発行された場合の動作を示しています。**Lpipe **は** NOP **命令と依存関係がないかぎり独立に処理を実行します。**
この命令は，ほかの命令との並列発行が可能です。

(7) CTRET, TRAP **命令**

CTRET, TRAP **命令は，命令フェッチ・ユニット（**Bpipe**）で実行されます。**

[パイプライン]

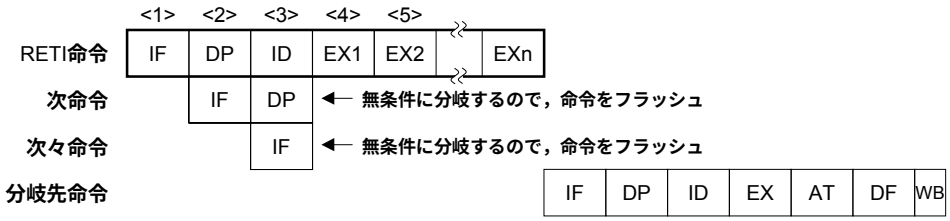


[説　　明]　**この図では，**Bpipe **で** CTRET, TRAP **命令が実行され，**Lpipe **ですべての命令が発行された場合の動作を示しています。**
各命令は単独で発行されます。
また，実行クロック数は 9 **です。**

(8) RETI **命令**

RETI **命令は，命令フェッチ・ユニット（**Bpipe**）で実行されます。**

[パイプライン]

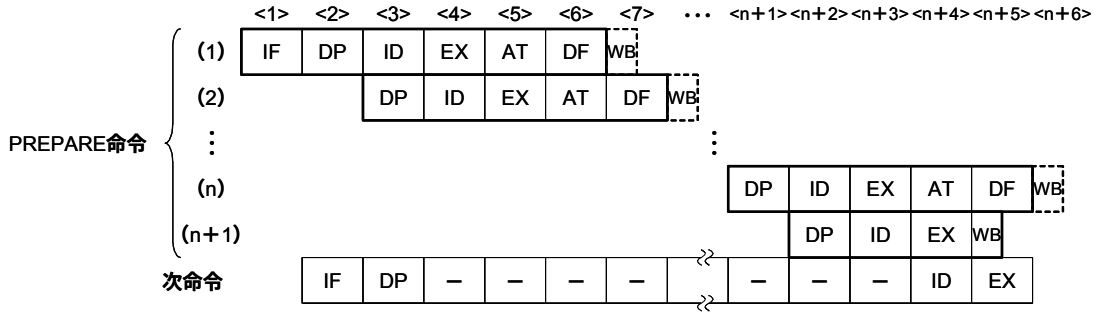


[説　　明]　**この図では，**Bpipe **で** RETI **命令が実行され，命令実行パイプライン・レフト（**Lpipe**）ですべての命令が発行された場合の動作を示しています。**
この命令は単独で発行されます。
また，実行クロック数はシステムによって異なります（割り込みコントローラの動作仕様に依存）。

(9) PREPARE 命令

PREPARE 命令は，命令実行パイプライン・レフト（Lpipe）の ALU ユニットで実行されます。

［パイプライン］



−：待ち合わせのために挿入されるアイドル

備考　n は，レジスタ・リスト（list12）で指定されるレジスタの数です。

［説　　明］　DP ステージで n＋1 個の命令に分割され，最初に n 個のストア命令を，最後にスタック・ポインタ（SP）への書き込み命令を実行します。ただし，ストア命令ではレジスタへのデータの書き込みがないので，WB ステージでは何も行いません。この図では，Lpipe で PREPARE 命令が実行され，Lpipe に次命令が発行された場合の動作を示しています。命令実行パイプライン・ライト（Rpipe）は PREPARE 命令と依存関係がないかぎり独立に処理を実行します。

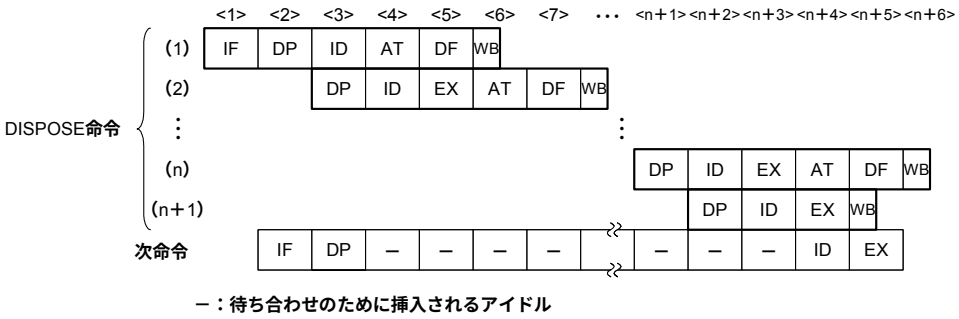
DP ステージで命令をデコードしている期間は，ディスパッチ・ユニットは Lpipe には命令を発行しません。

この命令は単独で発行されます。

（10）DISPOSE 命令

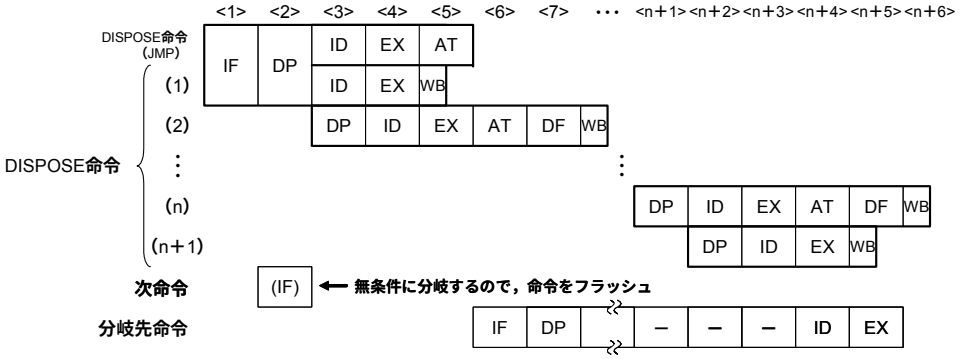
DISPOSE 命令は，命令実行パイプライン・ライト（Rpipe）の ALU ユニットで実行されます。

［パイプライン］（a）分岐しない場合



－：待ち合わせのために挿入されるアイドル

（b）分岐する場合



－　：待ち合わせのために挿入されるアイドル
(IF)：無効となる命令フェッチ

備考　n は，レジスタ・リスト（list12）で指定されるレジスタの数です。

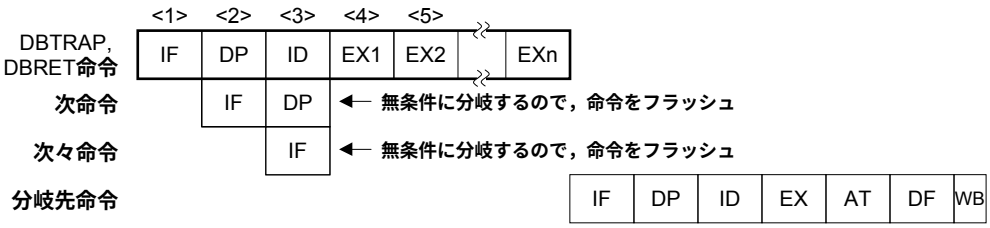
［説　　明］　DP ステージで n＋1 個の命令に分割され，最初に n 個のロード命令を，最後にスタック・ポインタ（SP）への書き込み命令を実行します。この図では，Lpipe で DISPOSE 命令が実行され，Lpipe に次命令が発行された場合の動作を示しています。
命令実行パイプライン・ライト（Rpipe）は DISPOSE 命令と依存関係がないかぎり独立に処理を実行します。
DP ステージで命令をデコードしている期間は，ディスパッチ・ユニットは Lpipe には命令を発行しません。
この命令は単独で発行されます。

8. 2. 15　ディバグ機能用命令

ディバグ機能用命令は，命令フェッチ・ユニット（Bpipe）で実行されます。

［対象の命令］　DBTRAP, DBRET

［パイプライン］



［説　　明］　この図では，Bpipe で DBTRAP, DBRET 命令が実行され，命令実行パイプライン・レフト（Lpipe）ですべての命令が発行された場合の動作を示しています。

各命令は単独で発行されます。

また，各命令は，CPU で保留されているすべての命令の処理が完了するまで分岐先命令の実行を行いません。

第9章　ディバグ・モードへの移行

V850E2 CPU は，ディバグ・トラップ，例外トラップ，ディバグ・ブレークが発生するとプログラム・カウンタ（PC）にハンドラ・アドレス（00000060H）をセットし，ディバグ・モードへ移行します。
また，シングルステップ動作の設定を行うと各命令の実行ごとにディバグ・モードに移行できます。

注意　ディバグ・モードに移行すると，データ・キャッシュ（dCACHE）がホールドされ，データやタグの更新は行われません。ディバグ・モード中にキャッシュ可能領域の外部メモリにアクセスすると，dCACHE が有効でも外部メモリだけにアクセスするため，コヒーレンシ性が崩れてしまいます。したがって，ディバグ・モニタ・ルーチン中でキャッシュ可能領域のデータを操作する場合は，ユーザ・モードに戻る前に，dCACHE をクリア（ライト・スルーの場合），またはフラッシュ，クリア（ライトバックの場合）してください。

9.1　ディバグ・モードへの移行方法

(1) ディバグ・トラップ

DBTRAP 命令の実行によりディバグ・トラップが発生し，ディバグ・モードに移行します。

(2) 例外トラップ

命令の不正実行により例外トラップが発生し，ディバグ・モードに移行します。

(3) ディバグ・ブレーク

ディバグ・ブレークには，次の 3 種類があります。

- ブレークポイント設定によるブレーク（4 チャネル）
- ミス・アライン・アクセス例外発生によるブレーク
- アラインメント・エラー例外発生によるブレーク

ディバグ・ブレークの設定は，次に示すシステム・レジスタにより行います。

- ディバグ・インタフェース・レジスタ（DIR）
- ブレークポイント制御レジスタ 0-3（BPC0-BPC3）
- ブレークポイント・アドレス設定レジスタ 0-3（BPAV0-BPAV3）
- ブレークポイント・アドレス・マスク・レジスタ 0-3（BPAM0-BPAM3）
- ブレークポイント・データ設定レジスタ 0-3（BPDV0-BPDV3）
- ブレークポイント・データ・マスク・レジスタ 0-3（BPDM0-BPDM3）

(a) ブレークポイント設定によるブレーク（4 チャネル）

次に示すブレーク条件の成立によるブレークポイント設定（4 チャネル）に基づいて，ディバグ・モードに移行します。各条件の設定は，BPCn レジスタで行います（n = 0-3）。

注意 BPCn レジスタの IE ビットをセット（1）している場合は，BP ASID ビットと ASID レジスタに設定したプログラム ID が一致しないと，ブレーク条件が成立してもディバグ・モードには移行しません。

表9－1　ブレーク条件

種類	ブレーク条件		ブレーク・タイミング	BPxxn レジスタの設定[注2]				BPCn レジスタの MD, FE, RE, WE ビットの設定		
	アドレス[注1]	データ		BP AVn	BP AMn	BP DVn	BP DMn	MD	FE	RE, WE
実行系トラップ	任意の実行アドレス	－	実行直前	<1>	<1>	<1>	<1>	1	1	0[注4]
	特定の実行アドレス	－		○	<0>	<1>	<1>			
	特定の実行アドレス範囲	－		○	○	<1>	<1>			
アクセス系トラップ	任意のアクセス・アドレス	特定のデータ	実行後[注3]	<1>	<1>	○	<0>	0	0	0/1[注5]
		特定のデータ範囲	実行直後	<1>	<1>	○	○			
	特定のアクセス・アドレス	任意のデータ	実行後[注3]	○	<0>	<1>	<1>	任意		
		特定のデータ		○	<0>	○	<0>	0		
		特定のデータ範囲		○	<0>	○	○			
	特定のアクセス・アドレス範囲	任意のデータ	実行直後	○	○	<1>	<1>	任意		
		特定のデータ	実行後[注3]	○	○	○	<0>	0		
		特定のデータ範囲		○	○	○	○			

注 1. 実行アドレスとは，命令フェッチ時のアドレスを，アクセス・アドレスとは命令の実行に伴いアクセスが発生するアドレスを意味します。

2. 次のように設定してください。

○　：ブレーク条件を設定してください。

<0>：すべてのビットをクリア（0）してください。

<1>：条件設定の必要はありませんが，初期値が不定のため，すべてのビットをセット（1）してください（BPAVn, BPAMn レジスタのビット 31-29 は 0 固定のため，セット（1）できません）。

3. データ・ライト時：実行直後

データ・リード時：数命令実行後（スリップ）

4. 必ず 0 を設定してください（1 を設定した場合の動作は保証しません）。

5. アクセスの種類（リードのみ，ライトのみ，リード／ライト両方）に応じて設定してください。

備考 1. n = 0-3

2. 複数のブレーク条件が設定されている場合，条件のうちの 1 つでも成立すればディバグ・モードに移行します。

チャネル 0, 1／チャネル 2, 3 を連動させて，次の 2 種類の動作を行うことも可能です（同時に動作させることは不可）。

<1> **シーケンシャル実行によるブレーク（シーケンシャル・ブレーク・モード）**

チャネル 0→1，または，チャネル 2→3 の順にブレーク条件が一致した場合だけ，ディバグ・モードに移行します。ディバグ・インタフェース・レジスタ（DIR）の SQ0 ビット（チャネル 0, 1 を使用する場合），または，SQ1 ビット（チャネル 2, 3 を使用する場合）をセット（1）することにより設定されます。

<2> **同時実行によるブレーク（レインジ・ブレーク・モード）**

チャネル 0, 1，または，チャネル 2, 3 のブレーク条件が同時に一致した場合だけ，ディバグ・モードに移行します。ディバグ・インタフェース・レジスタ（DIR）の RE0 ビット（チャネル 0, 1 を使用する場合），または，RE1 ビット（チャネル 2, 3 を使用する場合）をセット（1）することにより設定されます。

注意 1. **実行系トラップとアクセス系トラップでは，ブレーク条件が一致するタイミングが異なります。したがって，シーケンシャル・ブレーク・モードに設定しても，アクセス系トラップのあとに実行系トラップが発生する場合は，正常に動作しないことがあります。**

2. **レインジ・ブレーク・モードでは，実行系トラップとアクセス系トラップのどちらか一方を設定してください（チャネル 0, 1，または，チャネル 2, 3 使用時）。**

(b) ミス・アライン・アクセス例外発生によるブレーク

ディバグ・インタフェース・レジスタ（DIR）の MT ビットをセット（1）することにより，設定されます。

ロード命令，ストア命令実行時にミス・アライン・アクセスが発生するとディバグ・モードに移行します（IFIMAEN 端子入力によるミス・アライン・アクセス許可／禁止の設定には依存しません）。

(c) アラインメント・エラー例外発生によるブレーク

ディバグ・インタフェース・レジスタ（DIR）の AT ビットをセット（1）することにより，設定されます。

アラインメント・エラーが発生するとディバグ・モードに移行します。

アラインメント・エラーが発生するのは，PREPARE, DISPOSE 命令実行時にスタック・ポインタ（SP）がワード境界以外に強制的にアラインされた場合です。

チャネル 0-3 使用時のアクセス系トラップの場合を除くディバグ・ブレークでは，ブレークの発生要因となった命令のアドレスが DBPC レジスタに退避されます（命令実行完了前にディバグ・モードに移行するため）。したがって，ディバグ・モードからユーザ・モードへの移行後に，ブレークの発生要因となった命令が再検出され，再度のディバグ・ブレークが発生します（無視されません）。

(4) シングルステップ動作

PSW レジスタの SS フラグをセット（1）すると，シングルステップ動作が設定され，各命令の実行ごとにディバグ・ブレークを発生します。シングルステップ動作は，次の手順で設定／解除されます。

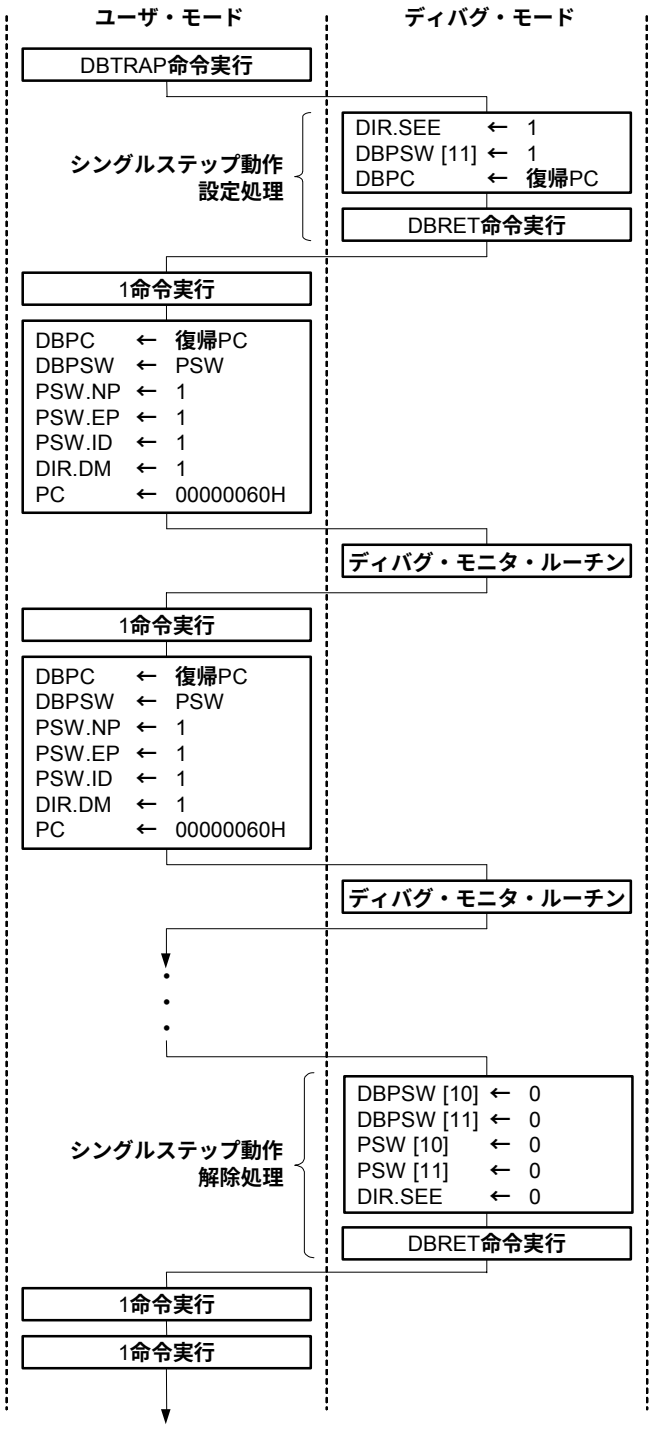
(a) シングルステップ動作の設定手順

<1> ディバグ・トラップ（DBTRAP 命令の実行）により，ディバグ・モードに移行する。
<2>PSW レジスタの SS フラグを制御するために，DIR レジスタの SSE ビットをセット（1）する。
<3> ユーザ・モードへの移行時に PSW レジスタの SS フラグをセット（1）するために DBPSW レジスタのビット 11 をセット（1）する。
<4> DBPC レジスタに復帰 PC 値を転送する。
<5> DBRET 命令によりユーザ・モードへ移行する（移行時に PSW レジスタの SS フラグがセット（1）され，シングルステップ動作が設定される）。

(b) シングルステップ動作の解除手順

<1> ディバグ・モード処理ルーチンでの動作時に，DBPSW レジスタのビット 10（SB ビット）とビット 11（SS ビット）の両方をクリア（0）する。
<2> ディバグ・モード処理ルーチンの動作時に，PSW レジスタのビット 10（SB ビット）とビット 11（SS ビット）の両方をクリア（0）する。
<3> ディバグ・モード処理ルーチンの動作時に，DIR レジスタのビット 8（SSE ビット）をクリア（0）する。
<4> DBRET 命令により，ユーザ・モードに復帰する。

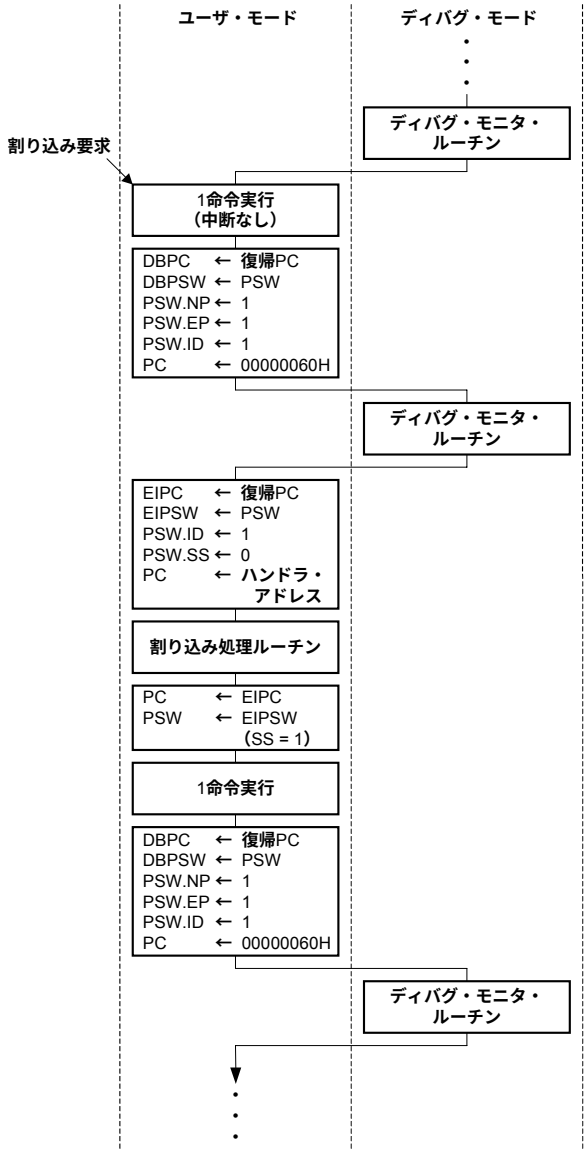
図9－1　シングルステップ動作の実行フロー



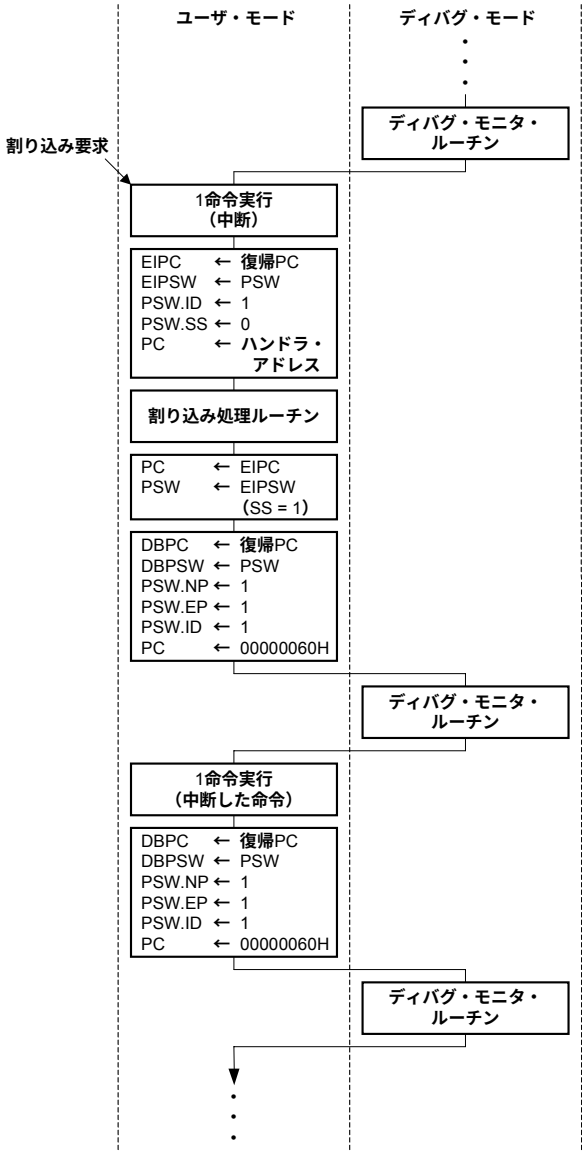
備考　シングルステップ動作時，ユーザ・モードで割り込み要求が発生すると，割り込み処理ルーチンに移行する際に PSW レジスタの SB フラグの内容が SS フラグに転送されます。
したがって，SB フラグがセット（1）されていれば，割り込み処理ルーチンでもシングルステップ動作を行い，クリア（0）されていれば，割り込み処理ルーチンではシングルステップ動作は行いません（V850E1 CPU 互換）。
なお，SS フラグは，割り込み処理ルーチンからの復帰処理（EIPSW→PSW）により，再びセット（1）されます（SB フラグの値にかかわりません）。
また，割り込み発生時に実行されている命令により，処理フローが異なることがあります（**図** 9ー2 参照）。

図9－2　シングルステップ動作時に割り込み要求が発生した場合の処理フロー

（a）割り込み要求により実行を中断しない命令の場合



（b）割り込み要求により実行を中断する命令の場合



備考 1. 割り込み要求により実行を中断する命令の場合（**表** 6－1　**割り込み，例外コード一覧**参照），その命令の実行完了を待たずに割り込み処理を行い，割り込み処理ルーチンからの復帰後，1 命令も実行しないでディバグ・モードに移行することがあります。

2. PSW **レジスタの** SB **フラグがクリア**（0）**されている場合の例です。**

9. 2　注意事項

チャネル 0-3 を使用する場合，ビット操作命令によるアクセス時は，アクセスするアドレスにより，BPDVn レジスタへの設定値が異なります（n = 0-3）。

各アクセス・サイズに対するアクセス・アドレスごとのブレーク条件の設定例を次に示します。

表9－2　ブレーク条件設定例

アクセス・サイズ（かっこ内はデータの例）	アクセス・アドレス[注1]	バス・サイクル	BPCn レジスタの TY ビット	BPAVn レジスタ[注1]	BPDVn レジスタ[注2]
ワード（44332211H）	0H	W	1, 1（ワード・アクセス）	0H	44332211H
	1H			1H	
	2H			2H	
	3H			3H	
ハーフワード（2211H）	0H	HW	1, 0（ハーフワード・アクセス）	0H	xxxx2211H
	1H			1H	
	2H			2H	
	3H			3H	
バイト（11H）	0H	B	0, 1（バイト・アクセス）	0H	xxxxxx11H
	1H			1H	
	2H			2H	
	3H			3H	
ビット（11H）	0H	B	0, 1（バイト・アクセス）	0H	xxxxxx11H
	1H			1H	xxxx11xxH
	2H			2H	xx11xxxxH
	3H			3H	11xxxxxxH

注 1. 下位 2 ビットの値を示します。

2.「x」は，BPDMn レジスタによりマスクされていることを示します。

備考 1. W　：ワード・データ転送サイクル

HW：ハーフワード・データ転送サイクル

B　：バイト・データ転送サイクル

2. n = 0-3

付録 A　命令一覧

アルファベット順の命令機能一覧を表 A－1 に，フォーマット順の命令一覧を表 A－2 に示します。

表A－1　命令機能一覧（アルファベット順）（1/12）

ニモニック	オペランド	フォーマット	フラグ					命令機能
			CY	OV	S	Z	SAT	
ADD	reg1, reg2	I	0/1	0/1	0/1	0/1	－	加算。 reg2 のワード・データに reg1 のワード・データを加算し，その結果を reg2 に格納します。
ADD	imm5, reg2	II	0/1	0/1	0/1	0/1	－	加算。 reg2 のワード・データにワード長まで符号拡張した 5 ビット・イミーディエトを加算し，その結果を reg2 に格納します。
ADDI	imm16, reg1, reg2	VI	0/1	0/1	0/1	0/1	－	加算。 reg1 のワード・データにワード長まで符号拡張した 16 ビット・イミーディエトを加算し，その結果を reg2 に格納します。
ADF	cccc, reg1, reg2, reg3	XI	0/1	0/1	0/1	0/1	－	条件付き加算。 reg2 のワード・データに reg1 のワード・データを加算した結果に 1（加算結果が条件コード「cccc」で指定された条件に満たされた場合）または 0（加算結果が条件コード「cccc」で指定された条件に満たされなかった場合）を加算し，その結果を reg3 に格納します。
AND	reg1, reg2	I	－	0	0/1	0/1	－	論理積。 reg2 のワード・データと reg1 のワード・データの論理積をとり，その結果を reg2 に格納します。
ANDI	imm16, reg1, reg2	VI	－	0	0	0/1	－	論理積。 reg1 のワード・データとワード長までゼロ拡張した 16 ビット・イミーディエトの論理積をとり，その結果を reg2 に格納します。
Bcond	disp9	III	－	－	－	－	－	条件分岐（if Carry）。 命令が指定する PSW レジスタのフラグをテストし，条件を満たしているときは分岐し，そうでないときは次の命令に進みます。分岐先 PC は，現在の PC と 8 ビット・イミーディエトを 1 ビット・シフトしてワード長まで符号拡張した 9 ビット・ディスプレースメントを加算した値です。
BSH	reg2, reg3	XII	0/1	0	0/1	0/1	－	ハーフワード・データのバイト・スワップ。 エンディアン変換を行います。
BSW	reg2, reg3	XII	0/1	0	0/1	0/1	－	ワード・データのバイト・スワップ。 エンディアン変換を行います。
CALLT	imm6	II	－	－	－	－	－	テーブル参照によるサブルーチン・コール。 CTBP の内容に基づき，PC の値を変更し，制御を移します。

表A－1　命令機能一覧（アルファベット順）（2/12）

ニモニック	オペランド	フォーマット	フラグ					命令機能
			CY	OV	S	Z	SAT	
CLR1	bit#3, disp16 [reg1]	VIII	－	－	－	0/1	－	ビット・クリア。 まず，reg1 のワード・データと，ワード長まで符号拡張した 16 ビット・ディスプレースメントを加算して 32 ビット・アドレスを生成します。生成したアドレスのバイト・データを読み出し，3 ビットのビット・ナンバで示されるビットをクリア（0）し，元のアドレスに書き戻します。
CLR1	reg2, [reg1]	IX	－	－	－	0/1	－	ビット・クリア。 まず，reg1 のワード・データを読み出して 32 ビット・アドレスを生成します。生成したアドレスのバイト・データを読み出し，reg2 の下位 3 ビットで示されるビットをクリア（0）し，元のアドレスに書き戻します。
CMOV	cccc, reg1, reg2, reg3	XI	－	－	－	－	－	条件付き転送。 条件コード「cccc」で指定された条件が，満たされた場合は reg1 のデータを，満たされなかった場合は reg2 のデータを，reg3 に転送します。
CMOV	cccc, imm5, reg2, reg3	XII	－	－	－	－	－	条件付き転送。 条件コード「cccc」で指定された条件が，満たされた場合はワード長まで符号拡張した 5 ビット・イミーディエト・データを，満たされなかった場合は reg2 のデータを，reg3 に転送します。
CMP	reg1, reg2	I	0/1	0/1	0/1	0/1	－	比較。 reg2 のワード・データと reg1 のワード・データを比較し，結果を PSW レジスタの各フラグに示します。比較は reg2 のワード・データから reg1 の内容を減算することで行います。
CMP	imm5, reg2	II	0/1	0/1	0/1	0/1	－	比較。 reg2 のワード・データとワード長まで符号拡張した 5 ビット・イミーディエトを比較し，結果を PSW レジスタの各フラグに示します。比較は reg2 のワード・データから符号拡張したイミーディエトの内容を減算することで行います。
CTRET	（なし）	X	0/1	0/1	0/1	0/1	0/1	サブルーチン・コールからの復帰。 システム・レジスタから復帰 PC と PSW を取り出し，CALLT 命令により呼び出されたルーチンから復帰します。
DBRET	（なし）	X	0/1	0/1	0/1	0/1	0/1	ディバグ・トラップからの復帰。 システム・レジスタから復帰 PC と PSW を取り出し，ディバグ・モニタ・ルーチンから復帰します。
DBTRAP	（なし）	I	－	－	－	－	－	ディバグ・トラップ。 復帰 PC, PSW をシステム・レジスタに退避し，PC にハンドラ・アドレス（00000060H）をセットして制御を移します。
DI	（なし）	X	－	－	－	－	－	マスカブル割り込みの禁止。 PSW 中の ID フラグをセット（1）し，この命令実行中からマスカブル割り込みの受け付けを禁止します。

表A－1　命令機能一覧（アルファベット順）（3/12）

ニモニック	オペランド	フォーマット	フラグ					命令機能
			CY	OV	S	Z	SAT	
DISPOSE	imm5, list12	XIII	－	－	－	－	－	スタック・フレームの削除。 5 ビット・イミーディエト・データを，2 ビット論理左シフトし，ワード長までゼロ拡張したものを，sp に加算します。そして，list12 で指定されている汎用レジスタに復帰（sp で指定するアドレスからデータをロードし，sp に 4 を加算）します。
DISPOSE	imm5, list12, [reg1]	XIII	－	－	－	－	－	スタック・フレームの削除。 5 ビット・イミーディエト・データを，2 ビット論理左シフトし，ワード長までゼロ拡張したものを，sp に加算します。そして，list12 で指定されている汎用レジスタに復帰（sp で指定するアドレスからデータをロードし，sp に 4 を加算）し，reg1 で指定されたアドレスに制御を移します。
DIV	reg1, reg2, reg3	XI	－	0/1	0/1	0/1	－	符号付きワード・データの除算。 reg2 のワード・データを reg1 のワード・データで除算し，その商を reg2 に，余りを reg3 に格納します。
DIVH	reg1, reg2	I	－	0/1	0/1	0/1	－	符号付きハーフワード・データの除算。 reg2 のワード・データを reg1 の下位ハーフワード・データで除算し，その商を reg2 に格納します。
DIVH	reg1, reg2, reg3	XI	－	0/1	0/1	0/1	－	符号付きハーフワード・データの除算。 reg2 のワード・データを reg1 の下位ハーフワード・データで除算し，その商を reg2 に，余りを reg3 に格納します。
DIVHU	reg1, reg2, reg3	XI	－	0/1	0/1	0/1	－	符号なしハーフワード・データの除算。 reg2 のワード・データを reg1 の下位ハーフワード・データで除算し，その商を reg2 に，余りを reg3 に格納します。
DIVU	reg1, reg2, reg3	XI	－	0/1	0/1	0/1	－	符号なしワード・データの除算。 reg2 のワード・データを reg1 のワード・データで除算し，その商を reg2 に，余りを reg3 に格納します。
EI	（なし）	X	－	－	－	－	－	マスカブル割り込みの許可。 PSW レジスタ中の ID フラグをクリア（0）し，次の命令からマスカブル割り込みの受け付けを許可します。
HALT	（なし）	X	－	－	－	－	－	CPU 停止。 CPU の動作クロックを停止させ，HALT 状態に入ります。
HSH	reg2, reg3	XII	0/1	0	0/1	0/1	－	ハーフワード・データのハーフワード・スワップ。 reg2 の内容を reg3 に格納し，フラグの判定結果を PSW レジスタに格納します。
HSW	reg2, reg3	XII	0/1	0	0/1	0/1	－	ワード・データのハーフワード・スワップ。 エンディアン交換を行います。

表A－1　命令機能一覧（アルファベット順）（4/12）

ニモニック	オペランド	フォーマット	フラグ					命令機能
			CY	OV	S	Z	SAT	
JARL	disp22, reg2	V	－	－	－	－	－	分岐とレジスタ・リンク。 現在の PC に 4 を加算した値を reg2 に退避し，現在の PC にワード長まで符号拡張した 22 ビット・ディスプレースメントを加算し，その PC に制御を移します。22 ビット・ディスプレースメントのビット 0 は 0 にマスクされます。
JARL	disp32, reg1	VI	－	－	－	－	－	分岐とレジスタ・リンク。 現在の PC に 6 を加算した値を reg1 に退避し，現在の PC に 32 ビット・ディスプレースメントを加算し，その PC に制御を移します。32 ビット・ディスプレースメントのビット 0 は 0 にマスクされます。
JMP	[reg1]	I	－	－	－	－	－	レジスタ間接無条件分岐。 reg1 で指定されるアドレスに制御を移します。アドレスのビット 0 は 0 にマスクされます。
JMP	disp32 [reg1]	VI	－	－	－	－	－	レジスタ間接無条件分岐。 reg1 に 32 ビット・ディスプレースメントを加算し，その PC に制御を移します。アドレスのビット 0 は 0 にマスクされます。
JR	disp22	V	－	－	－	－	－	無条件分岐。 現在の PC にワード長まで符号拡張した 22 ビット・ディスプレースメントを加算し，その PC に制御を移します。22 ビット・ディスプレースメントのビット 0 は 0 にマスクされます。
JR	disp32	VI	－	－	－	－	－	無条件分岐。 現在の PC に 32 ビット・ディスプレースメントを加算し，その PC に制御を移します。32 ビット・ディスプレースメントのビット 0 は 0 にマスクされます。
LD.B	disp16 [reg1] , reg2	VII	－	－	－	－	－	符号付きバイト・データのロード。 reg1 のワード・データと，ワード長まで符号拡張した 16 ビット・ディスプレースメントを加算して 32 ビット・アドレスを生成します。生成したアドレスからバイト・データを読み出し，ワード長まで符号拡張し，reg2 に格納します。
LD.BU	disp16 [reg1] , reg2	VII	－	－	－	－	－	符号なしバイト・データのロード。 reg1 のワード・データと，ワード長まで符号拡張した 16 ビット・ディスプレースメントを加算して 32 ビット・アドレスを生成します。生成したアドレスからバイト・データを読み出し，ワード長までゼロ拡張し，reg2 に格納します。
LD.H	disp16 [reg1], reg2	VII	－	－	－	－	－	符号付きハーフワード・データのロード。 reg1 のワード・データと，ワード長まで符号拡張した 16 ビット・ディスプレースメントを加算して 32 ビット・アドレスを生成します。この生成したアドレスからハーフワード・データを読み出し，ワード長まで符号拡張し，reg2 に格納します。

表A－1　命令機能一覧（アルファベット順）（5/12）

ニモニック	オペランド	フォーマット	フラグ					命令機能
			CY	OV	S	Z	SAT	
LD.HU	disp16 [reg1] , reg2	VII	－	－	－	－	－	符号なしハーフワード・ロード。 reg1 のワード・データと，ワード長まで符号拡張した 16 ビット・ディスプレースメントを加算して 32 ビット・アドレスを生成します。この生成したアドレスからハーフワード・データを読み出し，ワード長までゼロ拡張し，reg2 に格納します。
LD.W	disp16 [reg1] , reg2	VII	－	－	－	－	－	ワード・データのロード。 reg1 のデータと，ワード長まで符号拡張した 16 ビット・ディスプレースメントを加算して 32 ビット・アドレスを生成します。この生成したアドレスからワード・データを読み出し，reg2 に格納します。
LDSR	reg2, regID	IX	－	－	－	－	－	システム・レジスタへのロード。 reg2 のワード・データを regID で指定されるシステム・レジスタに設定します。regID が PSW レジスタの場合は，PSW レジスタのフラグには reg2 の対応するビットの値が設定されます。
MAC	reg1, reg2, reg3, reg4	XI	－	－	－	－	－	（符号付き）ワード・データの加算付き乗算。 reg2 のワード・データに reg1 のワード・データを乗算した結果と，reg3 と reg3＋1 を結合した 64 ビット・データを加算し，その結果（64 ビット・データ）を reg4 と reg4＋1 に格納します。
MACU	reg1, reg2, reg3, reg4	XI	－	－	－	－	－	（符号なし）ワード・データの加算付き乗算。 reg2 のワード・データに reg1 のワード・データを乗算した結果と，reg3 と reg3＋1 を結合した 64 ビット・データを加算し，その結果（64 ビット・データ）を reg4 と reg4＋1 に格納します。
MOV	reg1, reg2	I	－	－	－	－	－	データの転送。 reg1 のワード・データを reg2 にコピーし，転送します。
MOV	imm5, reg2	II	－	－	－	－	－	データの転送。 ワード長まで符号拡張した 5 ビット・イミーディエトを reg2 にコピーし，転送します。
MOV	imm32, reg1	VI	－	－	－	－	－	データの転送。 32 ビット・イミーディエトを reg1 にコピーし，転送します。
MOVEA	imm16, reg1, reg2	VI	－	－	－	－	－	実行アドレスの転送。 reg1 のワード・データにワード長まで符号拡張した 16 ビット・イミーディエトを加算し，その結果を reg2 に格納します。
MOVHI	imm16, reg1, reg2	VI	－	－	－	－	－	上位ハーフワードの転送。 reg1 のワード・データに上位 16 ビット（16 ビット・イミーディエト）と下位 16 ビット（0）を合わせたワード・データを加算し，その結果を reg2 に格納します。
MUL	reg1, reg2, reg3	XI	－	－	－	－	－	符号付きワード・データの乗算。 reg2 のワード・データに reg1 のワード・データを乗算し，その結果を reg2 と reg3 に格納します。
MUL	imm9, reg2, reg3	XII	－	－	－	－	－	符号付きワード・データの乗算。 reg2 のワード・データにワード長まで符号拡張した 9 ビット・イミーディエト・データを乗算し，その結果を reg2 と reg3 に格納します。

表A－1　命令機能一覧（アルファベット順）（6/12）

ニモニック	オペランド	フォーマット	フラグ					命令機能
			CY	OV	S	Z	SAT	
MULH	reg1, reg2	I	－	－	－	－	－	符号付きハーフワード・データの乗算。 reg2 の下位ハーフワード・データに reg1 の下位ハーフワード・データを乗算し，その結果を reg2 に格納します。
MULH	imm5, reg2	II	－	－	－	－	－	符号付きハーフワード・データの乗算。 reg2 の下位ハーフワード・データにハーフワード長まで符号拡張した 5 ビット・イミーディエトを乗算し，その結果を reg2 に格納します。
MULHI	imm16, reg1, reg2	VI	－	－	－	－	－	符号付きハーフワード・イミーディエトの乗算。 reg1 の下位ハーフワード・データに 16 ビット・イミーディエトを乗算し，その結果を reg2 に格納します。
MULU	reg1, reg2, reg3	XI	－	－	－	－	－	符号なしワード・データの乗算。 reg2 のワード・データに reg1 のワード・データを乗算し，その結果を reg2 と reg3 に格納します。
MULU	imm9, reg2, reg3	XII	－	－	－	－	－	符号なしワード・データの乗算。 reg2 のワード・データにワード長までゼロ拡張した 9 ビット・イミーディエト・データを乗算し，その結果を reg2 と reg3 に格納します。
NOP	（なし）	I	－	－	－	－	－	ノー・オペレーション。
NOT	reg1, reg2	I	－	0	0/1	0/1	－	論理否定。 reg1 のワード・データの論理否定（1 の補数）をとり，その結果を reg2 に格納します。
NOT1	bit#3, disp16 [reg1]	VIII	－	－	－	0/1	－	ビット・ノット。 まず，reg1 のデータと，ワード長まで符号拡張した 16 ビット・ディスプレースメントを加算して 32 ビット・アドレスを生成します。生成したアドレスのバイト・データの 3 ビットのビット・ナンバで示されるビットを反転します。
NOT1	reg2, [reg1]	IX	－	－	－	0/1	－	ビット・ノット。 まず，reg1 を読み出して 32 ビット・アドレスを生成します。生成したアドレスのバイト・データの reg2 の下位 3 ビットで示されるビットを反転します。
OR	reg1, reg2	I	－	0	0/1	0/1	－	論理和。 reg2 のワード・データと reg1 のワード・データの論理和をとり，その結果を reg2 に格納します。
ORI	imm16, reg1, reg2	VI	－	0	0/1	0/1	－	論理和。 reg1 のワード・データとワード長までゼロ拡張した 16 ビット・イミーディエトの論理和をとり，その結果を reg2 に格納します。
PREPARE	list12, imm5	XIII	－	－	－	－	－	スタック・フレームの生成。 list12 で指定されている汎用レジスタを退避（sp から 4 を減算し，データをそのアドレスに格納）します。次に，2 ビット論理左シフトし，ワード長までゼロ拡張した 5 ビット・イミーディエトを sp から減算します。

表A－1　命令機能一覧（アルファベット順）（7/12）

ニモニック	オペランド	フォーマット	フラグ					命令機能
			CY	OV	S	Z	SAT	
PREPARE	list12, imm5, sp/imm	XIII	－	－	－	－	－	スタック・フレームの生成。 list12 で指定されている汎用レジスタを退避（sp から 4 を減算し，データをそのアドレスに格納）します。次に，2 ビット論理左シフトし，ワード長までゼロ拡張した 5 ビット・イミーディエトを sp から減算します。続いて，第 3 オペランドで指定されるデータを ep にロードします。
RETI	（なし）	X	0/1	0/1	0/1	0/1	0/1	割り込みまたは例外処理ルーチンから復帰。 システム・レジスタから復帰 PC と PSW を取り出し，割り込みまたは例外処理ルーチンから復帰する命令です。
SAR	reg1,reg2	IX	0/1	0	0/1	0/1	－	算術右シフト。 reg2 のワード・データを reg1 の下位 5 ビットで示されるシフト数分，算術右シフト（シフト以前の MSB の値を順に MSB にコピーする）し，reg2 に書き込みます。
SAR	imm5, reg2	II	0/1	0	0/1	0/1	－	算術右シフト。 reg2 のワード・データをワード長までゼロ拡張した 5 ビット・イミーディエトで示されるシフト数分，算術右シフト（シフト以前の MSB の値を順に MSB にコピーする）し，reg2 に書き込みます。
SAR	reg1, reg2, reg3	XI	0/1	0	0/1	0/1	－	算術右シフト。 reg2 のワード・データを reg1 の下位 5 ビットで示されるシフト数分，算術右シフト（シフト以前の MSB の値を順に MSB にコピーする）し，reg3 に書き込みます。
SASF	cccc, reg2	IX	－	－	－	－	－	シフトとフラグ条件の設定。 条件コード「cccc」で指定された条件が満たされた場合は，reg2 のデータを 1 ビット論理左シフトし，LSB に 1 がセットされます。満たされなかった場合は，reg2 のデータを 1 ビット論理左シフトし，LSB に 0 がセットされます。
SATADD	reg1, reg2	I	0/1	0/1	0/1	0/1	0/1	飽和加算。 reg2 のワード・データに reg1 のワード・データを加算し，その結果を reg2 に格納します。ただし，その結果が正の最大値を越えたときは正の最大値を，負の最大値を越えたときは負の最大値を reg2 に格納し，SAT フラグをセット（1）します。
SATADD	imm5,reg2	II	0/1	0/1	0/1	0/1	0/1	飽和加算。 reg2 のワード・データにワード長まで符号拡張した 5 ビット・イミーディエトを加算し，その結果を reg2 に格納します。ただし，その結果が正の最大値を越えたときは正の最大値を，負の最大値を越えたときは負の最大値を reg2 に格納し，SAT フラグをセット（1）します。

表A－1　命令機能一覧（アルファベット順）（8/12）

ニモニック	オペランド	フォーマット	フラグ					命令機能
			CY	OV	S	Z	SAT	
SATADD	reg1, reg2, reg3	XI	0/1	0/1	0/1	0/1	0/1	飽和加算。 reg2 のワード・データに reg1 のワード・データを加算し，その結果を reg3 に格納します。ただし，その結果が正の最大値を越えたときは正の最大値を，負の最大値を越えたときは負の最大値を reg3 に格納し，SAT フラグをセット（1）します。
SATSUB	reg1, reg2	I	0/1	0/1	0/1	0/1	0/1	飽和減算。 reg2 のワード・データから reg1 のワード・データを減算し，その結果を reg2 に格納します。ただし，その結果が正の最大値を越えたときは正の最大値を，負の最大値を越えたときは負の最大値を reg2 に格納し，SAT フラグをセット（1）します。
SATSUB	reg1, reg2, reg3	XI	0/1	0/1	0/1	0/1	0/1	飽和減算。 reg2 のワード・データから reg1 のワード・データを減算し，その結果を reg3 に格納します。ただし，その結果が正の最大値を越えたときは正の最大値を，負の最大値を越えたときは負の最大値を reg3 に格納し，SAT フラグをセット（1）します。
SATSUBI	imm16, reg1, reg2	VI	0/1	0/1	0/1	0/1	0/1	飽和減算。 reg1 のワード・データからワード長まで符号拡張した 16 ビット・イミーディエトを減算し，その結果を reg2 に格納します。ただし，その結果が正の最大値を越えたときは正の最大値を，負の最大値を越えたときは負の最大値を reg2 に格納し，SAT フラグをセット（1）します。
SATSUBR	reg1, reg2	I	0/1	0/1	0/1	0/1	0/1	飽和逆減算。 reg1 のワード・データから reg2 のワード・データを減算し，その結果を reg2 に格納します。ただし，その結果が正の最大値を越えたときは正の最大値を，負の最大値を越えたときは負の最大値を reg2 に格納し，SAT フラグをセット（1）します。
SBF	cccc, reg1, reg2, reg3	XI	0/1	0/1	0/1	0/1	－	条件付き減算。 reg2 のワード・データから reg1 の内容を減算した結果から 1（減算結果が条件コード「cccc」で指定された条件に満たされた場合）または 0（減算結果が条件コード「cccc」で指定された条件に満たされなかった場合）を減算し，その結果を reg3 に格納します。
SCH0L	reg2, reg3	IX	0/1	0	0	0/1	－	MSB 側からのビット（0）検索。 reg2 のワード・データを左側（MSB 側）から検索し，最初にビット（0）が見つかった位置を reg3 に 16 進数で書き込みます。ビット（0）が見つからなかった場合は，reg3 に 0 を書き込み，同時に Z フラグをセット（1）します。最後にビット（0）が見つかった場合は CY フラグをセット（1）します。

表A－1　命令機能一覧（アルファベット順）（9/12）

ニモニック	オペランド	フォーマット	フラグ					命令機能
			CY	OV	S	Z	SAT	
SCH0R	reg2, reg3	IX	0/1	0	0	0/1	－	LSB 側からのビット（0）検索。 reg2 のワード・データを右側（LSB 側）から検索し，最初にビット（0）が見つかった位置を reg3 に 16 進数で書き込みます。ビット（0）が見つからなかった場合は，reg3 に 0 を書き込み，同時に Z フラグをセット（1）します。最後にビット（0）が見つかった場合は CY フラグをセット（1）します。
SCH1L	reg2, reg3	IX	0/1	0	0	0/1	－	MSB 側からのビット（1）検索。 reg2 のワード・データを左側（MSB 側）から検索し，最初にビット（1）が見つかった位置を reg3 に 16 進数で書き込みます。ビット（1）が見つからなかった場合は，reg3 に 0 を書き込み，同時に Z フラグをセット（1）します。最後にビット（1）が見つかった場合は CY フラグをセット（1）します。
SCH1R	reg2, reg3	IX	0/1	0	0	0/1	－	LSB 側からのビット（1）検索。 reg2 のワード・データを右側（LSB 側）から検索し，最初にビット（1）が見つかった位置を reg3 に 16 進数で書き込みます。ビット（1）が見つからなかった場合は，reg3 に 0 を書き込み，同時に Z フラグをセット（1）します。最後にビット（1）が見つかった場合は CY フラグをセット（1）します。
SET1	bit#3, disp16 [reg1]	VIII	－	－	－	0/1	－	ビット・セット。 まず，reg1 のデータと，ワード長まで符号拡張した 16 ビット・ディスプレースメントを加算して 32 ビット・アドレスを生成します。生成したアドレスのバイト・データの 3 ビットのビット・ナンバで示されるビットをセット（1）します。
SET1	reg2, [reg1]	IX	－	－	－	0/1	－	ビット・セット。 まず，reg1 のデータを読み出して 32 ビット・アドレスを生成します。生成したアドレスのバイト・データの reg2 の下位 3 ビットで示されるビットをセット（1）します。
SETF	cccc, reg2	IX	－	－	－	－	－	フラグ条件の設定。 条件コード「cccc」の示す条件が，対象となる PSW レジスタのフラグと一致した場合には，reg2 に 1 を，そうでない場合には 0 を格納します。
SHL	reg1, reg2	IX	0/1	0	0/1	0/1	－	論理左シフト。 reg2 のワード・データを reg1 の下位 5 ビットで示されるシフト数分，論理左シフト（LSB 側に 0 を送り込む）し，reg2 に書き込みます。
SHL	imm5, reg2	II	0/1	0	0/1	0/1	－	論理左シフト。 reg2 のワード・データをワード長までゼロ拡張した 5 ビット・イミーディエトで示されるシフト数分，論理左シフト（LSB 側に 0 を送り込む）し，reg2 に書き込みます。
SHL	reg1, reg2, reg3	XI	0/1	0	0/1	0/1	－	論理左シフト。 reg2 のワード・データを reg1 の下位 5 ビットで示されるシフト数分，論理左シフト（LSB 側に 0 を送り込む）し，reg3 に書き込みます。

表A－1　命令機能一覧（アルファベット順）（10/12）

ニモニック	オペランド	フォーマット	フラグ					命令機能
			CY	OV	S	Z	SAT	
SHR	reg1, reg2	IX	0/1	0	0/1	0/1	－	論理右シフト。 reg2 のワード・データを reg1 の下位 5 ビットで示されるシフト数分，論理右シフト（MSB 側に 0 を送り込む）し，reg2 に書き込みます。
SHR	imm5, reg2	II	0/1	0	0/1	0/1	－	論理右シフト。 reg2 のワード・データをワード長までゼロ拡張した 5 ビット・イミーディエトで示されるシフト数分，論理右シフト（MSB 側に 0 を送り込む）し，reg2 に書き込みます。
SHR	reg1, reg2, reg3	XI	0/1	0	0/1	0/1	－	論理右シフト。 reg2 のワード・データを reg1 の下位 5 ビットで示されるシフト数分，論理右シフト（MSB 側に 0 を送り込む）し，reg3 に書き込みます。
SLD.B	disp7 [ep], reg2	IV	－	－	－	－	－	バイト・ロード。 エレメント・ポインタと，ワード長までゼロ拡張した 7 ビット・ディスプレースメントを加算して 32 ビット・アドレスを生成します。生成したアドレスからバイト・データを読み出し，ワード長まで符号拡張し，reg2 に格納します。
SLD.BU	disp4 [ep], reg2	IV	－	－	－	－	－	符号なしバイト・ロード。 エレメント・ポインタと，ワード長までゼロ拡張した 4 ビット・ディスプレースメントを加算して 32 ビット・アドレスを生成します。生成したアドレスからバイト・データを読み出し，ワード長までゼロ拡張し，reg2 に格納します。
SLD.H	disp8 [ep], reg2	IV	－	－	－	－	－	ハーフワード・ロード。 エレメント・ポインタと，ワード長までゼロ拡張した 8 ビット・ディスプレースメントを加算して 32 ビット・アドレスを生成します。この，32 ビット・アドレスのビット 0 を 0 にマスクしたアドレスからハーフワード・データを読み出し，ワード長まで符号拡張し，reg2 に格納します。
SLD.HU	disp5 [ep], reg2	IV	－	－	－	－	－	符号なしハーフワード・ロード。 エレメント・ポインタと，ワード長までゼロ拡張した 5 ビット・ディスプレースメントを加算して 32 ビット・アドレスを生成します。この，32 ビット・アドレスのビット 0 を 0 にマスクしたアドレスからハーフワード・データを読み出し，ワード長までゼロ拡張し，reg2 に格納します。
SLD.W	disp8 [ep], reg2	IV	－	－	－	－	－	ワード・ロード。 エレメント・ポインタと，ワード長までゼロ拡張した 8 ビット・ディスプレースメントを加算して 32 ビット・アドレスを生成します。この，32 ビット・アドレスのビット 0 とビット 1 を 0 にマスクしたアドレスからワード・データを読み出し，reg2 に格納します。
SST.B	reg2, disp7 [ep]	IV	－	－	－	－	－	バイト・ストア。 エレメント・ポインタと，ワード長までゼロ拡張した 7 ビット・ディスプレースメントを加算して 32 ビット・アドレスを生成します。reg2 の最下位バイト・データを生成したアドレスに格納します。

表A－1　命令機能一覧（アルファベット順）（11/12）

ニモニック	オペランド	フォーマット	フラグ					命令機能
			CY	OV	S	Z	SAT	
SST.H	reg2,disp8 [ep]	IV	－	－	－	－	－	ハーフワード・ストア。 エレメント・ポインタと，ワード長までゼロ拡張した8ビット・ディスプレースメントを加算して32ビット・アドレスを生成します。reg2の下位ハーフワード・データを生成した32ビット・アドレスのビット0を0にマスクしたアドレスに格納します。
SST.W	reg2, disp8 [ep]	IV	－	－	－	－	－	ワード・ストア。 エレメント・ポインタと，ワード長までゼロ拡張した8ビット・ディスプレースメントを加算して32ビット・アドレスを生成します。reg2のワード・データを生成した32ビット・アドレスのビット0とビット1を0にマスクしたアドレスに格納します。
ST.B	reg2, disp16 [reg1]	VII	－	－	－	－	－	バイト・ストア。 reg1のデータと，ワード長まで符号拡張した16ビット・ディスプレースメントを加算して32ビット・アドレスを生成します。reg2の最下位バイト・データを生成したアドレスに格納します。
ST.H	reg2, disp16 [reg1]	VII	－	－	－	－	－	ハーフワード・ストア。 reg1のデータと，ワード長まで符号拡張した16ビット・ディスプレースメントを加算して32ビット・アドレスを生成します。reg2の下位ハーフワードのデータを生成した32ビット・アドレスのビット0を0にマスクしたアドレスに格納します。
ST.W	reg2, disp16 [reg1]	VII	－	－	－	－	－	ワード・ストア。 reg1のデータと，ワード長まで符号拡張した16ビット・ディスプレースメントを加算して32ビット・アドレスを生成します。reg2のワード・データを生成した32ビット・アドレスのビット0とビット1を0にマスクしたアドレスに格納します。
STSR	regID, reg2	IX	－	－	－	－	－	システム・レジスタの内容のストア。 regIDで指定されるシステム・レジスタの内容をreg2に設定します。
SUB	reg1, reg2	I	0/1	0/1	0/1	0/1	－	減算。 reg2のワード・データからreg1のワード・データを減算し，その結果をreg2に格納します。
SUBR	reg1, reg2	I	0/1	0/1	0/1	0/1	－	逆減算。 reg1のワード・データからreg2のワード・データを減算し，その結果をreg2に格納します。
SWITCH	reg1	I	－	－	－	－	－	テーブル参照分岐。 テーブルの先頭アドレス（SWITCH命令の次のアドレス）と1ビット論理左シフトしたreg1のデータを加算したテーブル・エントリ・アドレスが指し示すハーフワード・エントリ・データをロードします。続いて，ロードしたデータを1ビット論理左シフトし，ワード長まで符号拡張したあと，テーブルの先頭アドレス（SWITCH命令の次のアドレス）を加算したターゲット・アドレスに分岐します。
SXB	reg1	I	－	－	－	－	－	バイト・データの符号拡張。 reg1の最下位バイトをワード長に符号拡張します。
SXH	reg1	I	－	－	－	－	－	ハーフワード・データの符号拡張。 reg1の下位ハーフワードをワード長に符号拡張します。

表A－1　命令機能一覧（アルファベット順）（12/12）

ニモニック	オペランド	フォーマット	フラグ					命令機能
			CY	OV	S	Z	SAT	
TRAP	vector	X	－	－	－	－	－	ソフトウエア・トラップ。 復帰 PC,PSW をシステム・レジスタに退避し，例外コードの設定，PSW レジスタのフラグの設定を行ったあと，vector で示されるトラップ・ベクタに対応するトラップ・ハンドラのアドレスに分岐し，例外処理を開始します。
TST	reg1, reg2	I	－	0	0/1	0/1	－	テスト。 reg2 のワード・データと reg1 のワード・データの論理積をとります。結果は格納されず，フラグのみが影響を受けます。
TST1	bit#3, disp16 [reg1]	VIII	－	－	－	0/1	－	ビット・テスト。 まず，reg1 のデータと，ワード長まで符号拡張した 16 ビット・ディスプレースメントを加算して 32 ビット・アドレスを生成します。生成したアドレスのバイト・データの 3 ビットのビット・ナンバで示されるビットが 0 ならば Z フラグをセット（1）し，1 ならばクリア（0）します。
TST1	reg2, [reg1]	IX	－	－	－	0/1	－	ビット・テスト。 まず，reg1 のデータを読み出して 32 ビット・アドレスを生成します。生成したアドレスのバイト・データの reg2 の下位 3 ビットで示されるビットが 0 ならば Z フラグをセット（1）し，1 ならばクリア（0）します。
XOR	reg1, reg2	I	－	0	0/1	0/1	－	排他的論理和。 reg2 のワード・データと reg1 のワード・データの排他的論理和をとり，その結果を reg2 に格納します。
XORI	imm16, reg1, reg2	VI	－	0	0/1	0/1	－	排他的論理和。 reg1 のワード・データとワード長までゼロ拡張した 16 ビット・イミーディエトの排他的論理和をとり，その結果を reg2 に格納します。
ZXB	reg1	I	－	－	－	－	－	バイト・データのゼロ拡張。 reg1 の最下位バイトをワード長にゼロ拡張します。
ZXH	reg1	I	－	－	－	－	－	ハーフワード・データのゼロ拡張。 reg1 の下位ハーフワードをワード長にゼロ拡張します。

表A－2　命令一覧（フォーマット順）（1/4）

フォーマット	オペコード		ニモニック	オペランド
	15　0	31　16		
I	0000000000000000	－	NOP	－
	rrrrr000000RRRRR	－	MOV	reg1, reg2
	rrrrr000001RRRRR	－	NOT	reg1, reg2
	rrrrr000010RRRRR	－	DIVH	reg1, reg2
	00000000010RRRRR	－	SWITCH	reg1
	00000000011RRRRR	－	JMP	[reg1]
	rrrrr000100RRRRR	－	SATSUBR	reg1, reg2
	rrrrr000101RRRRR	－	SATSUB	reg1, reg2
	rrrrr000110RRRRR	－	SATADD	reg1, reg2
	rrrrr000111RRRRR	－	MULH	reg1, reg2
	00000000100RRRRR	－	ZXB	reg1
	00000000101RRRRR	－	SXB	reg1
	00000000110RRRRR	－	ZXH	reg1
	00000000111RRRRR	－	SXH	reg1
	rrrrr001000RRRRR	－	OR	reg1, reg2
	rrrrr001001RRRRR	－	XOR	reg1, reg2
	rrrrr001010RRRRR	－	AND	reg1, reg2
	rrrrr001011RRRRR	－	TST	reg1, reg2
	rrrrr001100RRRRR	－	SUBR	reg1, reg2
	rrrrr001101RRRRR	－	SUB	reg1, reg2
	rrrrr001110RRRRR	－	ADD	reg1, reg2
	rrrrr001111RRRRR	－	CMP	reg1, reg2
	1111100001000000	－	DBTRAP	－
II	rrrrr010000iiiii	－	MOV	imm5, reg2
	rrrrr010001iiiii	－	SATADD	imm5, reg2
	rrrrr010010iiiii	－	ADD	imm5, reg2
	rrrrr010011iiiii	－	CMP	imm5, reg2
	0000001000iiiiii	－	CALLT	imm6
	rrrrr010100iiiii	－	SHR	imm5, reg2
	rrrrr010101iiiii	－	SAR	imm5, reg2
	rrrrr010110iiiii	－	SHL	imm5, reg2
	rrrrr010111iiiii	－	MULH	imm5, reg2
III	ddddd1011dddCCCC	－	Bcond	disp9
IV	rrrrr0000110dddd	－	SLD.BU	disp4 [ep], reg2
	rrrrr0000111dddd	－	SLD.HU	disp5 [ep], reg2
	rrrrr0110ddddddd	－	SLD.B	disp7 [ep], reg2
	rrrrr0111ddddddd	－	SST.B	reg2, disp7 [ep]
	rrrrr1000ddddddd	－	SLD.H	disp8 [ep], reg2
	rrrrr1001ddddddd	－	SST.H	reg2, disp8 [ep]
	rrrrr1010ddddddd0	－	SLD.W	disp8 [ep], reg2
	rrrrr1010ddddddd1	－	SST.W	reg2, disp8 [ep]

表A－2　命令一覧（フォーマット順）（2/4）

フォーマット	オペコード		ニモニック	オペランド
	15　0	31　16		
V	rrrrr11110ddddd	ddddddddddddddd0	JARL	disp22, reg2
	0000011110ddddd	ddddddddddddddd0	JR	disp22
VI	0000001011100000	注1	JR	disp32
	00000010111RRRRR	注1	JARL	disp32, reg1
	rrrrr110000RRRRR	iiiiiiiiiiiiiiii	ADDI	imm16, reg1, reg2
	rrrrr110001RRRRR	iiiiiiiiiiiiiiii	MOVEA	imm16, reg1, reg2
	rrrrr110010RRRRR	iiiiiiiiiiiiiiii	MOVHI	imm16, reg1, reg2
	rrrrr110011RRRRR	iiiiiiiiiiiiiiii	SATSUBI	imm16, reg1, reg2
	00000110001RRRRR	注2	MOV	imm32, reg1
	rrrrr110100RRRRR	iiiiiiiiiiiiiiii	ORI	imm16, reg1, reg2
	rrrrr110101RRRRR	iiiiiiiiiiiiiiii	XORI	imm16, reg1, reg2
	rrrrr110110RRRRR	iiiiiiiiiiiiiiii	ANDI	imm16, reg1, reg2
	rrrrr110111RRRRR	iiiiiiiiiiiiiiii	MULHI	imm16, reg1, reg2
	00000110111RRRRR	注1	JMP	disp32 [reg1]
VII	rrrrr111000RRRRR	ddddddddddddddd	LD.B	disp16 [reg1], reg2
	rrrrr111001RRRRR	ddddddddddddddd0	LD.H	disp16 [reg1], reg2
	rrrrr111001RRRRR	ddddddddddddddd1	LD.W	disp16 [reg1], reg2
	rrrrr111010RRRRR	ddddddddddddddd	ST.B	reg2, disp16 [reg1]
	rrrrr111011RRRRR	ddddddddddddddd0	ST.H	reg2, disp16 [reg1]
	rrrrr111011RRRRR	ddddddddddddddd1	ST.W	reg2, disp16 [reg1]
	rrrrr11110bRRRRR	ddddddddddddddd1	LD.BU	disp16 [reg1], reg2
	rrrrr111111RRRRR	ddddddddddddddd1	LD.HU	disp16 [reg1], reg2

注1. 32ビット・ディスプレースメント・データです。上位32ビット（ビット16-47）は次のようになります。

31　16	47　32
dddddddddddddd0	DDDDDDDDDDDDDDD

2. 32ビット・イミーディエト・データです。上位32ビット（ビット16-47）は次のようになります。

31　16	47　32
iiiiiiiiiiiiiii	IIIIIIIIIIIIIII

表A－2　命令一覧（フォーマット順）（3/4）

フォーマット	オペコード		ニモニック	オペランド
	15　　0	31　　16		
VIII	00bbb111110RRRRR	ddddddddddddddddd	SET1	bit#3, disp16 [reg1]
	01bbb111110RRRRR	ddddddddddddddddd	NOT1	bit#3, disp16 [reg1]
	10bbb111110RRRRR	ddddddddddddddddd	CLR1	bit#3, disp16 [reg1]
	11bbb111110RRRRR	ddddddddddddddddd	TST1	bit#3, disp16 [reg1]
IX	rrrrr1111110cccc	0000000000000000	SETF	cccc, reg2
	rrrrr111111RRRRR	0000000000100000	LDSR	reg2, regID
	rrrrr111111RRRRR	0000000001000000	STSR	regID, reg2
	rrrrr111111RRRRR	0000000010000000	SHR	reg1, reg2
	rrrrr111111RRRRR	0000000010100000	SAR	reg1, reg2
	rrrrr111111RRRRR	0000000011000000	SHL	reg1, reg2
	rrrrr111111RRRRR	0000000011100000	SET1	reg2, [reg1]
	rrrrr111111RRRRR	0000000011100010	NOT1	reg2, [reg1]
	rrrrr111111RRRRR	0000000011100100	CLR1	reg2, [reg1]
	rrrrr111111RRRRR	0000000011100110	TST1	reg2, [reg1]
	rrrrr1111110cccc	0000001000000000	SASF	cccc, reg2
	rrrrr11111100000	wwwww01101100000	SCH0R	reg2, reg3
	rrrrr11111100000	wwwww01101100010	SCH1R	reg2, reg3
	rrrrr11111100000	wwwww01101100100	SCH0L	reg2, reg3
	rrrrr11111100000	wwwww01101100110	SCH1L	reg2, reg3
X	00000111111iiiii	0000000100000000	TRAP	vector
	0000011111100000	0000000100100000	HALT	－
	0000011111100000	0000000101000000	RETI	－
	0000011111100000	0000000101000100	CTRET	－
	0000011111100000	0000000101000110	DBRET	－
	0000011111100000	0000000101100000	DI	－
	1000011111100000	0000000101100000	EI	－
XI	rrrrr111111RRRRR	wwwww00010000010	SHR	reg1, reg2, reg3
	rrrrr111111RRRRR	wwwww00010100010	SAR	reg1, reg2, reg3
	rrrrr111111RRRRR	wwwww00011000010	SHL	reg1, reg2, reg3
	rrrrr111111RRRRR	wwwww01000100000	MUL	reg1, reg2, reg3
	rrrrr111111RRRRR	wwwww01000100010	MULU	reg1, reg2, reg3
	rrrrr111111RRRRR	wwwww01010000000	DIVH	reg1, reg2, reg3
	rrrrr111111RRRRR	wwwww01010000010	DIVHU	reg1, reg2, reg3
	rrrrr111111RRRRR	wwwww01011000000	DIV	reg1, reg2, reg3
	rrrrr111111RRRRR	wwwww01011000010	DIVU	reg1, reg2, reg3
	rrrrr111111RRRRR	wwwww011001cccc0	CMOV	cccc, reg1, reg2, reg3
	rrrrr111111RRRRR	wwwww011100cccc0	SBF	cccc, reg1, reg2, reg3
	rrrrr111111RRRRR	wwwww01110011010	SATSUB	reg1, reg2, reg3
	rrrrr111111RRRRR	wwwww011101cccc0	ADF	cccc, reg1, reg2, reg3
	rrrrr111111RRRRR	wwwww01110111010	SATADD	reg1, reg2, reg3
	rrrrr111111RRRRR	wwww0011110mmmm0	MAC	reg1, reg2, reg3, reg4
	rrrrr111111RRRRR	wwww0011111mmmm0	MACU	reg1, reg2, reg3, reg4

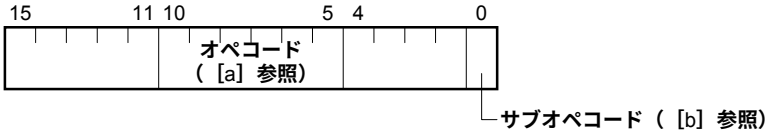
表A－2　命令一覧（フォーマット順）（4/4）

フォーマット	オペコード		ニモニック	オペランド
	15　0	31　16		
XII	rrrrr111111iiiii	wwwww01001IIII00	MUL	imm9, reg2, reg3
	rrrrr111111iiiii	wwwww01001IIII10	MULU	imm9, reg2, reg3
	rrrrr111111iiiii	wwwww011000cccc0	CMOV	cccc, imm5, reg2, reg3
	rrrrr11111100000	wwwww01101000000	BSW	reg2, reg3
	rrrrr11111100000	wwwww01101000010	BSH	reg2, reg3
	rrrrr11111100000	wwwww01101000100	HSW	reg2, reg3
	rrrrr11111100000	wwwww01101000110	HSH	reg2, reg3
XIII	00000110011111iL	LLLLLLLLLLLRRRRR	DISPOSE	imm5, list12, [reg1]
	00000110011111iL	LLLLLLLLLLL00000	DISPOSE	imm5, list12
	00000111101111iL	LLLLLLLLLLL00001	PREPARE	list12, imm5
	00000111101111iL	LLLLLLLLLLLff011	PREPARE	list12, imm5, sp/imm

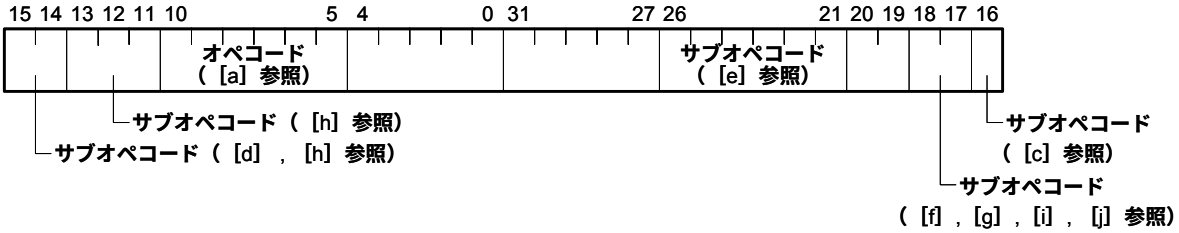
付録 B　命令オペコード・マップ

この章では，次に示す命令コードに対応したオペコード・マップを示します。

（1）16 ビット・フォーマット命令



（2）32 ビット・フォーマット命令



備考　オペランドの凡例

略　　号	意　　味
R	reg1：汎用レジスタ（ソース・レジスタとして使用）
r	reg2：汎用レジスタ（主にデスティネーション・レジスタとして使用。一部の命令で，ソース・レジスタとしても使用。）
w	reg3：汎用レジスタ（主に除算結果の余り，乗算結果の上位 32 ビットを格納）
bit#3	ビット・ナンバ指定用 3 ビット・データ
imm ×	× ビット・イミーディエト・データ
disp ×	× ビット・ディスプレースメント・データ
cccc	条件コードを示す 4 ビット・データ

[a] オペコード部

ビット10	ビット9	ビット8	ビット7	ビット6, 5 0,0	0,1	1,0	1,1	フォーマット
0	0	0	0	MOV R, r NOP[注1]	NOT	DIVH SWITCH[注2] DBTRAP[注3] 未定義[注3]	JMP [reg1] [注4] SLD.BU[注5] SLD.HU[注6]	I, IV
0	0	0	1	SATSUBR ZXB[注4]	SATSUB SXB[注4]	SATADD R, r ZXH[注4]	MULH SXH[注4]	I
0	0	1	0	OR	XOR	AND	TST	
0	0	1	1	SUBR	SUB	ADD R, r	CMP R, r	
0	1	0	0	MOV imm5, r CALLT[注4]	SATADD imm5, r	ADD imm5, r	CMP imm5, r	II, VI
0	1	0	1	SHR imm5, r	SAR imm5, r	SHL imm5, r	MULH imm5, r JR disp32[注1] JARL disp32, reg1[注2]	
0	1	1	0	SLD.B				IV
0	1	1	1	SST.B				
1	0	0	0	SLD.H				
1	0	0	1	SST.H				
1	0	1	0	SLD.W[注7] SST.W[注7]				
1	0	1	1	Bcond				III
1	1	0	0	ADDI	MOVEA MOV imm32, R[注4]	MOVHI DISPOSE[注4]	SATSUBI	VI, XIII
1	1	0	1	ORI	XORI	ANDI	MULHI JMP disp32 [reg1] [注4]	VI
1	1	1	0	LD.B	LD.H[注8] LD.W[注8]	ST.B	ST.H[注8] ST.W[注8]	VII
1	1	1	1	JR disp22 JARL disp22, reg2 LD.BU[注10] PREPARE[注11]		ビット操作1[注9]	LD.HU[注10] 未定義[注11] 拡張1[注12]	V, VII, VIII, IX, X, XI, XII, XIII,

注 1. R（reg1）が r0，かつ r（reg2）が r0（reg1, reg2 を持たない命令）の場合。

2. R（reg1）が r0 でなく，かつ r（reg2）が r0（reg1 を持ち，reg2 を持たない命令）の場合。

3. R（reg1）が r0，かつ r（reg2）が r0 でない（reg1 を持たず，reg2 を持つ命令）場合。

4. r（reg2）が r0（reg2 を持たない命令）の場合。

5. ビット 4 が 0，かつ r（reg2）が r0 でない（reg2 を持つ命令）場合。

6. ビット 4 が 1，かつ r（reg2）が r0 でない（reg2 を持つ命令）場合。

7. 【b】を参照してください。

8. 【c】を参照してください。

9. 【d】を参照してください。

10. ビット 16 が 1，かつ r（reg2）が r0 でない（reg2 を持つ命令）場合。

11. ビット 16 が 1，かつ r（reg2）が r0（reg2 を持たない命令）の場合。

12. 【e】を参照してください。

[b] ショート・フォーマット・ロード／ストア命令（ディスプレースメント／サブオペコード部）

ビット10	ビット9	ビット8	ビット7	ビット0	
				0	1
0	1	1	0	SLD.B	
0	1	1	1	SST.B	
1	0	0	0	SLD.H	
1	0	0	1	SST.H	
1	0	1	0	SLD.W	SST.W

[c] ロード／ストア命令（ディスプレースメント／サブオペコード部）

ビット6	ビット5	ビット16	
		0	1
0	0	LD.B	
0	1	LD.H	LD.W
1	0	ST.B	
1	1	ST.H	ST.W

[d] ビット操作命令1（サブオペコード部）

ビット15	ビット14	
	0	1
0	SET1　bit#3, disp16 [R]	NOT1　bit#3, disp16 [R]
1	CLR1　bit#3, disp16 [R]	TST1　bit#3, disp16 [R]

[e] 拡張1（サブオペコード部）

ビット 26	ビット 25	ビット 24	ビット 23	ビット 22, 21 0,0	0,1	1,0	1,1	フォーマット
0	0	0	0	SETF	LDSR	STSR	**未定義**	IX
0	0	0	1	SHR	SAR	SHL	**ビット操作 2** 注1	
0	0	1	0	TRAP	HALT	RETI 注2 CTRET 注2 DBRET 注2 **未定義**	EI 注3 DI 注3 **未定義**	X
0	0	1	1	**未定義**		**未定義**		−
0	1	0	0	SASF	MUL　R, r, w MULU　R, r, w 注4	MUL　imm9, r, w MULU　imm9, r, w 注4		IX, XI, XII
0	1	0	1	DIVH DIVHU 注4		DIV DIVU 注4		XI
0	1	1	0	CMOV cccc, imm5, r, w	CMOV cccc, R, r, w	BSW 注5 BSH 注5 HSW 注5 HSH 注5	SCH0R 注6 SCH1R 注6 SCH0L 注6 SCH1L 注6	IX, XI, XII
0	1	1	1	SBF SATSUB　R, r, w 注7	ADF SATADD　R, r, w 注7	MAC	MACU	XI
1	x	x	x	**不正命令**				−

注 1. **[f] を参照してください。**

2. **[g] を参照してください。**
3. **[h] を参照してください。**
4. **ビット** 17 **が** 1 **の場合。**
5. **[i] を参照してください。**
6. **[j] を参照してください。**
7. **ビット** 20-17 **が** 1,1,0,1 **の場合。**

[f] ビット操作命令2（サブオペコード部）

ビット 18	ビット 17 0	1
0	SET1　r, [R]	NOT1　r, [R]
1	CLR1　r, [R]	TST1　r, [R]

[g] 復帰命令（サブオペコード部）

ビット 18	ビット 17 0	1
0	RETI	**未定義**
1	CTRET	DBRET

[h] PSWレジスタ操作命令（サブオペコード部）

ビット 15	ビット 14	ビット 13, 12, 11							
		0,0,0	0,0,1	0,1,0	0,1,1	1,0,0	1,0,1	1,1,0	1,1,1
0	0	DI	未定義						
0	1	未定義							
1	0	EI	未定義						
1	1	未定義							

[i] エンディアン変換命令（サブオペコード部）

ビット 18	ビット 17	
	0	1
0	BSW	BSH
1	HSW	HSH

[j] ビット・サーチ命令（サブオペコード部）

ビット 18	ビット 17	
	0	1
0	SCH0R	SCH1R
1	SCH0L	SCH1L

付録C　V850 CPU, V850E1 CPU との アーキテクチャ上の相違点

(1/3)

項　目		V850E2 CPU	V850E1 CPU	V850 CPU
命令（オペランドを含む）	ADF　cccc, reg1, reg2, reg3	**あり**	**なし**	**なし**
	HSH　reg2, reg3			
	JARL　disp32, reg1			
	JMP　disp32 [reg1]			
	JR　disp32			
	MAC　reg1, reg2, reg3, reg4			
	MACU　reg1, reg2, reg3, reg4			
	SAR　reg1, reg2, reg3			
	SATADD　reg1, reg2, reg3			
	SATSUB　reg1, reg2, reg3			
	SBF　cccc, reg1, reg2, reg3			
	SCH0L　reg2, reg3			
	SCH0R　reg2, reg3			
	SCH1L　reg2, reg3			
	SCH1R　reg2, reg3			
	SHL　reg1, reg2, reg3			
	SHR　reg1, reg2, reg3			
	BSH　reg2, reg3		**あり**	
	BSW　reg2, reg3			
	CALLT　imm6			
	CLR1　reg2, [reg1]			
	CMOV　cccc, imm5, reg2, reg3			
	CMOV　cccc, reg1, reg2, reg3			
	CTRET			
	DBRET			
	DBTRAP			
	DISPOSE　imm5, list12			
	DISPOSE　imm5, list12 [reg1]			
	DIV　reg1, reg2, reg3			
	DIVH　reg1, reg2, reg3			
	DIVHU　reg1, reg2, reg3			
	DIVU　reg1, reg2, reg3			
	HSW　reg2, reg3			

(2/3)

項　目		V850E2 CPU	V850E1 CPU	V850 CPU
命令（オペランドを含む）	LD.BU　disp16 [reg1], reg2	あり		なし
	LD.HU　disp16 [reg1], reg2			
	MOV　imm32, reg1			
	MUL　imm9, reg2, reg3			
	MUL　reg1, reg2, reg3			
	MULU　reg1, reg2, reg3			
	MULU　imm9, reg2, reg3			
	NOT1　reg2, [reg1]			
	PREPARE　list12, imm5			
	PREPARE　list12, imm5, sp/imm			
	SASF　cccc, reg2			
	SET1　reg2, [reg1]			
	SLD.BU　disp4 [ep], reg2			
	SLD.HU　disp5 [ep], reg2			
	SWITCH　reg1			
	SXB　reg1			
	SXH　reg1			
	TST1　reg2, [reg1]			
	ZXB　reg1			
	ZXH　reg1			
命令フォーマット	Format IV	一部，形式が異なります。		
	Format XI	あり		なし
	Format XII			
	Format XIII			
命令実行クロック数		一部の命令で数値が異なります。		
プログラム領域		512Mバイト	64Mバイト	16 Mバイト
プログラム・カウンタ（PC）の有効ビット		下位29ビット	下位26ビット	下位24ビット
システム・レジスタ	プログラム・ステータス・ワード（PSW）	機能が異なります。		
	CALLT実行時状態退避レジスタ（CTPC, CTPSW）	あり		なし
	例外／ディバグ・トラップ時状態退避レジスタ（DBPC, DBPSW）			
	CALLTベース・ポインタ（CTBP）			

（3/3）

項　目		V850E2 CPU	V850E1 CPU	V850 CPU
システム・レジスタ	ディバグ・インタフェース・レジスタ（DIR）	あり（機能が異なります）		なし
	ブレークポイント制御レジスタ 0, 1（BPC0, BPC1）			
	プログラム ID レジスタ（ASID）	あり		
	ブレークポイント・アドレス設定レジスタ 0, 1（BPAV0, BPAV1）	あり（初期値が異なります）		
	ブレークポイント・アドレス・マスク・レジスタ 0,1（BPAM0,BPAM1）			
	ブレークポイント・データ設定レジスタ 0, 1（BPDV0, BPDV1）	あり		
	ブレークポイント・データ・マスク・レジスタ 0, 1（BPDM0, BPDM1）			
	ブレークポイント制御レジスタ 2, 3（BPC2, BPC3）	あり	なし	
	ブレークポイント・アドレス設定レジスタ 2, 3（BPAV2, BPAV3）			
	ブレークポイント・アドレス・マスク・レジスタ 2,3（BPAM2, BPAM3）			
	ブレークポイント・データ設定レジスタ 2, 3（BPDV2, BPDV3）			
	ブレークポイント・データ・マスク・レジスタ 2, 3（BPDM2, BPDM3）			
	例外トラップ時の状態退避レジスタ	DBPC, DBPSW		EIPC, EIPSW
不正命令コード		命令コードのコード領域が異なります。		
ミス・アライン・アクセス許可／禁止の設定		製品により設定可能		設定不可（ミス・アライン・アクセス禁止）
ノンマスカブル割り込み（NMI）	入力	3		1
	例外コード	0010H, 0020H, 0030H		0010H
	ハンドラ・アドレス	00000010H, 00000020H, 00000030H		00000010H
ディバグ・トラップ		あり		なし
ディバグ・ブレーク		あり（4要因）	あり（2要因）注	なし
パイプライン		7段	5段	
		各命令で，パイプラインの流れが異なります。		

注　BPC0, BPC1 レジスタ設定により，同等の機能が実行可能です。

付録 D　V850 CPU, V850E1 CPU に対して V850E2 CPU で追加した命令

V850E2 CPU の命令コードは，V850 CPU の命令コードに対し，オブジェクト・コード・レベルでの上位互換性を持たせています。V850E2 CPU では， V850 CPU で実行しても意味を持たない命令（主に r0 レジスタへの書き込みを行う命令）を追加命令として拡張しています。

表 D－1 に，V850E2 CPU で追加した命令コードに対応した V850 CPU の命令を，表 D－2 に V850E2 CPU で追加した命令コードに対応した V850E1 CPU の命令を示します。V850 CPU, V850E1 CPU を搭載した製品から V850E2 CPU を搭載した製品に置き換えるときの参考にしてください。

表D－1　V850E2 CPUで追加した命令と命令コードが同一のV850 CPUの命令（1/2）

V850E2 CPU で追加した命令	V850E2 CPU の命令コードと同一の V850 CPU の命令
CALLT　imm6	MOV　imm5, r0 または SATADD　imm5, r0
DISPOSE　imm5, list12	MOVHI　imm16, reg1, r0 または SATSUBI　imm16, reg1, r0
DISPOSE　imm5, list12 [reg1]	MOVHI　imm16, reg1, r0 または SATSUBI　imm16, reg1, r0
MOV　imm32, reg1	MOVEA　imm16, reg1, r0
SWITCH　reg1	DIVH　reg1, r0
SXB　reg1	SATSUB　reg1, r0
SXH　reg1	MULH　reg1, r0
ZXB　reg1	SATSUBR　reg1, r0
ZXH　reg1	SATADD　reg1, r0
JARL　disp32, reg1	MULH　imm5, r0
JMP　disp32 [reg1]	MULHI　imm16, reg1, r0
JR　disp32	MULH　0x0, r0
ADF　cccc, reg1, reg2, reg3	不正命令
BSH　reg2, reg3	
BSW　reg2, reg3	
CMOV　cccc, imm5, reg2, reg3	
CMOV　cccc, reg1, reg2, reg3	
CTRET	
DIV　reg1, reg2, reg3	
DIVH　reg1, reg2, reg3	
DIVHU　reg1, reg2, reg3	
DIVU　reg1, reg2, reg3	
HSH　reg2, reg3	
HSW　reg2, reg3	
MAC　reg1, reg2, reg3, reg4	
MACU　reg1, reg2, reg3, reg4	
MUL　imm9, reg2, reg3	

表D－1　V850E2 CPUで追加した命令と命令コードが同一のV850 CPUの命令（2/2）

V850E2 CPU で追加した命令	V850E2 CPU の命令コードと同一の V850 CPU の命令
MUL　reg1, reg2, reg3	不正命令
MULU　reg1, reg2, reg3	
MULU　imm9, reg2, reg3	
SASF　cccc, reg2	
SATADD　reg1, reg2, reg3	
SATSUB　reg1, reg2, reg3	
SBF　cccc, reg1, reg2, reg3	
SCH0L　reg2, reg3	
SCH0R　reg2, reg3	
SCH1L　reg2, reg3	
SCH1R　reg2, reg3	
CLR1　reg2, [reg1]	未定義
DBRET	
DBTRAP	
LD.BU　disp16 [reg1], reg2	
LD.HU　disp16 [reg1], reg2	
NOT1　reg2, [reg1]	
PREPARE　list12, imm5	
PREPARE　list12, imm5, sp/imm	
SAR　reg1, reg2, reg3	
SET1　reg2, [reg1]	
SHL　reg1, reg2, reg3	
SHR　reg1, reg2, reg3	
SLD.BU　disp4 [ep], reg2	
SLD.HU　disp5 [ep], reg2	
TST1　reg2, [reg1]	

表D－2　V850E2 CPUで追加した命令と命令コードが同一のV850E1 CPUの命令

V850E2 CPU で追加した命令	V850E2 CPU の命令コードと同一の V850E1 CPU の命令
ADF　cccc, reg1, reg2, reg3	不正命令
SATADD　reg1, reg2, reg3	
SATSUB　reg1, reg2, reg3	
SBF　cccc, reg1, reg2, reg3	
HSH　reg2, reg3	未定義
JARL　disp32, reg1	
JMP　disp32 [reg1]	
JR　disp32	
MACU　reg1, reg2, reg3, reg4	
SAR　reg1, reg2, reg3	
SCH0L　reg2, reg3	
SCH0R　reg2, reg3	
SCH1L　reg2, reg3	
SCH1R　reg2, reg3	
SHL　reg1, reg2, reg3	
SHR　reg1, reg2, reg3	
MAC　reg1, reg2, reg3, reg4	未定義，または積和演算命令の一部

付録 E　注意事項一覧

(1/3)

該当箇所		内　容
2. 2 システム・レジスタ		LDSR 命令により EIPC レジスタか FEPC レジスタ，または CTPC レジスタのビット 0 をセット（1）したあと，割り込み処理を行い，RETI 命令で復帰するときにビット 0 の値は無視されます（PC レジスタのビット 0 を 0 に固定してあるため）。EIPC, FEPC, CTPC レジスタに値を設定する場合は，特別な理由のないかぎり，偶数値（ビット 0 = 0）を設定してください。
2. 2. 8 ディバグ・インタフェース・レジスタ（DIR）		DIR レジスタの SQ1, RE1 ビットは，常にどちらか一方をセット（1），または両ビットともクリア（0）してください。両ビットともセット（1）した場合の動作は保証しません。
2. 2. 9 ブレークポイント制御レジスタ 0-3（BPC0-BPC3）		BPC0-BPC3 レジスタのビット 31-27, 14-12, 6, 5 には，必ず 0 を設定してください。1 を設定した場合の動作は保証しません。
		BPCn レジスタの FB2-FB0 ビットへのライトは，クリア（0）のみ可能です（n = 0-3）。これらのビットの値を更新する場合は，すべてのビットをクリア（0）してください。いずれかのビットをセット（1）した場合の動作は保証しません。
5. 1 命令フォーマット		一部の命令で未使用フィールド（RFU）がありますが，それらは将来の拡張用なので，0 に固定してください。
5. 3 命令セット	Bcond	飽和演算命令の実行結果で SAT フラグがセット（1）された場合，符号付き整数の条件分岐（BGE, BGT, BLE, BLT）は，分岐条件に意味がなくなります。これは，次の理由によるものです。通常の演算では，結果が正の最大値を越えると負の値になり，負の最大値を越えたときは正の値になります。つまり，オーバフローが生じると，S フラグが反転（0→1, 1→0）します。一方，飽和演算命令では，結果が正の最大値を越えたときは正の値で，負の最大値を越えたときは負の値で飽和します。通常の演算とは異なり，オーバフローが生じても S フラグは反転しません。このように，演算結果が飽和したときの S フラグは通常の演算とは異なるので，OV フラグとの排他的論理和（XOR）をとる分岐条件に意味がなくなります。
	CALLT	命令実行中に割り込みが発生すると，リード／ライト・サイクルが終了したあとに命令の実行を中止する場合があります。
	DBRET DBTRAP	この命令はディバグを目的とした命令のため，基本的にディバグ・ツールが使用しています。このためディバグ・ツールが使用しているときに，アプリケーションが使用すると誤動作する場合があります。
	DISPOSE	命令実行中に割り込みが発生すると，スタック操作を行うため，リード／ライト・サイクルとレジスタ値の書き換えが終了したあとに中止する場合があります。割り込みから復帰したあとに再実行されます。
	LD.H LD.HU	汎用レジスタ reg1 のデータとワード長まで符号拡張した 16 ビット・ディスプレースメントの加算結果は，ミス・アライン・モード設定により次の 2 種類があります。 •ビット 0 とビット 1 を 0 にマスクしてアドレス生成（ミス・アライン・アクセス禁止の場合） •ビット 0 とビット 1 をマスクしないでアドレス生成（ミス・アライン・アクセス許可の場合） ミス・アライン・アクセスについては，3. 3　**データ・アラインメント**を参照してください。

(2/3)

該当箇所		内　容
5. 3 命令セット	LD.W	汎用レジスタ reg1 のデータとワード長まで符号拡張した 16 ビット・ディスプレースメントの加算結果は，ミス・アライン・モード設定により次の 2 種類があります。 •ビット 0 とビット 1 を 0 にマスクしてアドレス生成（ミス・アライン・アクセス禁止の場合） •ビット 0 とビット 1 をマスクしないでアドレス生成（ミス・アライン・アクセス許可の場合） ミス・アライン・アクセスについては，3. 3　**データ・アラインメント**を参照してください。
	LDSR	この命令では，ニモニック記述の都合上，ソース・レジスタを reg2 としていますが，オペコード上は reg1 のフィールドを使用しています。したがって，ニモニック記述とオペコードにおいて，レジスタ指定の意味付けがほかの命令と異なります。 rrrrr：regID 指定 RRRRR：reg2 指定
		システム・レジスタ番号は，システム・レジスタを一意に識別するための番号です。予約されているシステム・レジスタ，書き込み禁止のシステム・レジスタに対してこの命令を実行した場合の動作は保証しません。
	MAC MACU	reg3，または reg4 に指定できる汎用レジスタは，偶数番号の付いたレジスタ（r0, r2, r4, …, r30）だけです。奇数番号の付いたレジスタ（r1, r3, …, r31）を指定した場合の結果は不定です。
	MUL MULU	「MUL reg1, reg2, reg3」，「MULU reg1, reg2, reg3」命令において，次の条件をすべて満たすレジスタの組み合わせは行わないでください。この条件に当てはまる命令を実行した場合の動作は保証しません。 ・reg1 = reg3 ・reg1 ≠ reg2 ・reg1 ≠ r0 ・reg3 ≠ r0
	PREPARE	命令実行中に割り込みが発生すると，スタック操作を行うため，リード／ライト・サイクルとレジスタ値の書き換えが終了したあとに中止する場合があります。
	RETI	ノンマスカブル割り込み処理またはソフトウエア例外処理からの RETI 命令による復帰時は，PC, PSW レジスタを正常にリストアするために，RETI 命令の直前で NP フラグ，EP フラグを次の状態にしておく必要があります。 • RETI 命令によるノンマスカブル割り込み処理からの復帰時：NP = 1 かつ EP = 0 • RETI 命令によるソフトウエア例外処理からの復帰時：EP = 1 プログラムによる設定には LDSR 命令を使用します。 割り込みコントローラの動作絡みで，この命令の後半の ID ステージでは割り込みを受け付けません。
	SATADD SATSUB SATSUBI SATSUBR	SAT フラグをクリア（0）するときは，LDSR 命令によって PSW レジスタにデータをロードしてください。

（3/3）

該当箇所		内　容
5. 3 命令セット	SLD.H SLD.HU SST.H	エレメント・ポインタとワード長までゼロ拡張した 8 ビット・ディスプレースメントの加算結果は，ミス・アライン・モード設定により次の 2 種類があります。 • ビット 0 を 0 にマスクしてアドレス生成（ミス・アライン・アクセス禁止の場合） • ビット 0 をマスクしないでアドレス生成（ミス・アライン・アクセス許可の場合） ミス・アライン・アクセスについては，3. 3　**データ・アラインメント**を参照してください。
	SLD.W SST.W	エレメント・ポインタとワード長までゼロ拡張した 8 ビット・ディスプレースメントの加算結果は，ミス・アライン・モード設定により次の 2 種類があります。 •ビット 0 とビット 1 を 0 にマスクしてアドレス生成（ミス・アライン・アクセス禁止の場合） •ビット 0 とビット 1 をマスクしないでアドレス生成（ミス・アライン・アクセス許可の場合） ミス・アライン・アクセスについては，3. 3　**データ・アラインメント**を参照してください。
	ST.H	汎用レジスタ reg1 のデータとワード長まで符号拡張した 16 ビット・ディスプレースメントの加算結果は，ミス・アライン・モード設定により次の 2 種類があります。 • ビット 0 を 0 にマスクしてアドレス生成（ミス・アライン・アクセス禁止の場合） • ビット 0 をマスクしないでアドレス生成（ミス・アライン・アクセス許可の場合） ミス・アライン・アクセスについては，3. 3　**データ・アラインメント**を参照してください。
	ST.W	汎用レジスタ reg1 のデータとワード長まで符号拡張した 16 ビット・ディスプレースメントの加算結果は，ミス・アライン・モード設定により次の 2 種類があります。 •ビット 0 とビット 1 を 0 にマスクしてアドレス生成（ミス・アライン・アクセス禁止の場合） •ビット 0 とビット 1 をマスクしないでアドレス生成（ミス・アライン・アクセス許可の場合） ミス・アライン・アクセスについては，3. 3　**データ・アラインメント**を参照してください。
	STSR	システム・レジスタ番号は，システム・レジスタを一意に識別するための番号です。予約されているシステム・レジスタ番号を指定した場合の動作は保証しません。
6. 2. 2 例外トラップ		命令，または，不正命令として定義されていない命令を実行したときの動作は保証しません。
第 9 章 ディバグ・モードへの移行		ディバグ・モードに移行すると，データ・キャッシュ（dCACHE）がホールドされ，データやタグの更新は行われません。ディバグ・モード中にキャッシュ可能領域の外部メモリにアクセスすると，dCACHE が有効でも外部メモリだけにアクセスするため，コヒーレンシ性が崩れてしまいます。したがって，ディバグ・モニタ・ルーチン中でキャッシュ可能領域のデータを操作する場合は，ユーザ・モードに戻る前に，dCACHE をクリア（ライト・スルーの場合），またはフラッシュ，クリア（ライトバックの場合）してください。
9. 1 ディバグ・モードへの移行方法		BPCn レジスタの IE ビットをセット（1）している場合は，BP ASID ビットと ASID レジスタに設定したプログラム ID が一致しないと，ブレーク条件が成立してもディバグ・モードには移行しません。
		実行系トラップとアクセス系トラップでは，ブレーク条件が一致するタイミングが異なります。したがって，シーケンシャル・ブレーク・モードに設定しても，アクセス系トラップのあとに実行系トラップが発生する場合は，正常に動作しないことがあります。
		レインジ・ブレーク・モードでは，実行系トラップとアクセス系トラップのどちらか一方を設定してください（チャネル 0, 1，または，チャネル 2, 3 使用時）。
9. 2 注意事項		（本文を参照してください。）

付録 F　命令索引

［メ　モ］

【発　行】
NECエレクトロニクス株式会社
〒211-8668　神奈川県川崎市中原区下沼部1753
電話（代表）：044(435)5111

お問い合わせ先

【ホームページ】

NECエレクトロニクスの情報がインターネットでご覧になれます。
URL（アドレス）　http://www.necel.co.jp/

【営業関係，技術関係お問い合わせ先】

半導体ホットライン
（電話：午前 9:00～12:00，午後 1:00～5:00）
電　話　：044−435−9494
E-mail　：info@necel.com

【資料請求先】

NECエレクトロニクスのホームページよりダウンロードいただくか，NECエレクトロニクスの販売特約店へお申し付けください。