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

V850ES

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　U15943JJ4V1UM00（第4版）
発行年月　March 2010 NS

［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，VIL（MAX.）から VIH（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，VIL（MAX.）から VIH（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 VDD 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。

(5) 電源投入切断順序　内部動作および外部インタフェースで異なる電源を使用するデバイスの場合，原則として内部電源を投入した後に外部電源を投入してください。切断の際には，原則として外部電源を切断した後に内部電源を切断してください。逆の電源投入切断順により，内部素子に過電圧が印加され，誤動作を引き起こしたり，異常電流が流れ内部素子を劣化させたりする場合があります。資料中に「電源投入切断シーケンス」についての記載のある製品については，その内容を守ってください。

(6) 電源 OFF 時における入力信号　当該デバイスの電源が OFF 状態の時に，入力信号や入出力プルアップ電源を入れないでください。入力信号や入出力プルアップ電源からの電流注入により，誤動作を引き起こしたり，異常電流が流れ内部素子を劣化させたりする場合があります。資料中に「電源 OFF 時における入力信号」についての記載のある製品については，その内容を守ってください。

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。

- 本資料に記載されている内容は 2010 年 01 月現在のもので，今後，予告なく変更することがあります。量産設計の際には最新の個別データ・シート等をご参照ください。
- 文書による当社の事前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，本資料に記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。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。当社製品をお客様の機器にご使用の際には，当社製品の不具合の結果として，生命，身体および財産に対する損害や社会的損害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。

注 2. 本事項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，製造製品をいう。

（M8E0909J）

はじめに

対象者　このマニュアルは，V850ES 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V850ES 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

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。

本文欄外の★印は，本版で改訂された主な箇所を示しています。
この“★”をPDF上でコピーして「検索する文字列」に指定することによって，改版箇所を容易に検索できます。

★ **製品タイプ**　このマニュアルは，製品名をタイプに分けて説明しています。
次の表より製品に対応するタイプを確認したあと，お読みください。

製品タイプ	製品名
タイプA	μPD703229Y, μPD70F3229Y, V850ES/Fx2, V850ES/FE3[注], V850ES/FF3[注], V850ES/FG3（μPD70F3374, 70F3375）[注], V850ES/FJ3（μPD70F3378）[注], V850ES/Fx3-L, V850ES/Hx2, V850ES/Hx3（μPD70F3757除く）, V850ES/Jx2, V850ES/Jx3, V850ES/Jx3-L, V850ES/Kx1, V850ES/Kx1+, V850ES/Kx2, V850ES/PM1, V850ES/SA2, V850ES/SA3, V850ES/ST2, V850ES/SG1, V850ES/Sx2, V850ES/Sx3,
タイプB	V850ES/FG3（μPD70F3376, 70F3377）, V850ES/FJ3（μPD70F3379, 70F3380, 70F3381, 70F3382）, V850ES/FK3, V850ES/HJ3（μPD70F3757）,V850ES/IE2, V850ES/IK1, V850ES/Jx3-E, V850ES/Jx3-H, V850ES/Jx3-U, V850ES/ST3, V850ES/Sx2-H

注　オプション・バイトで分岐レイテンシを2に設定すると「タイプA」，分岐レイテンシを3に設定すると「タイプB」になります。

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には，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するとともに，命令セットも複雑になり，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倍以上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です。

V850シリーズでは，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。

また，V850シリーズはさらなる高性能化を目指し，「V850 CPU」をベースに，動作周波数の高速化やパイプラインの効率化を行い，かつ命令の上位互換を保ちながら，「V850E1 CPU」，「V850E2 CPU」と進化を続けています。

「V850ES CPU」は，16ビット・マイコンが主流となる分野に対し，高性能でコスト・パフォーマンスの高い製品を展開するためのCPUコアとして開発しました。

150 MHzクラスの製品で採用実績のある「V850E1 CPU」とのコンパチビリティを維持しながら，動作周波数，乗算器，DMAなどの機能を16ビット・マイコン市場向けに最適化し，高性能，コンパクトを実現しています。

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：80
- 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5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：プログラム空間 … 64Mバイト・リニア
  （使用可能領域：16 Mバイト・リニア空間＋内蔵RAM領域60 Kバイト）
  データ空間 … 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または4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

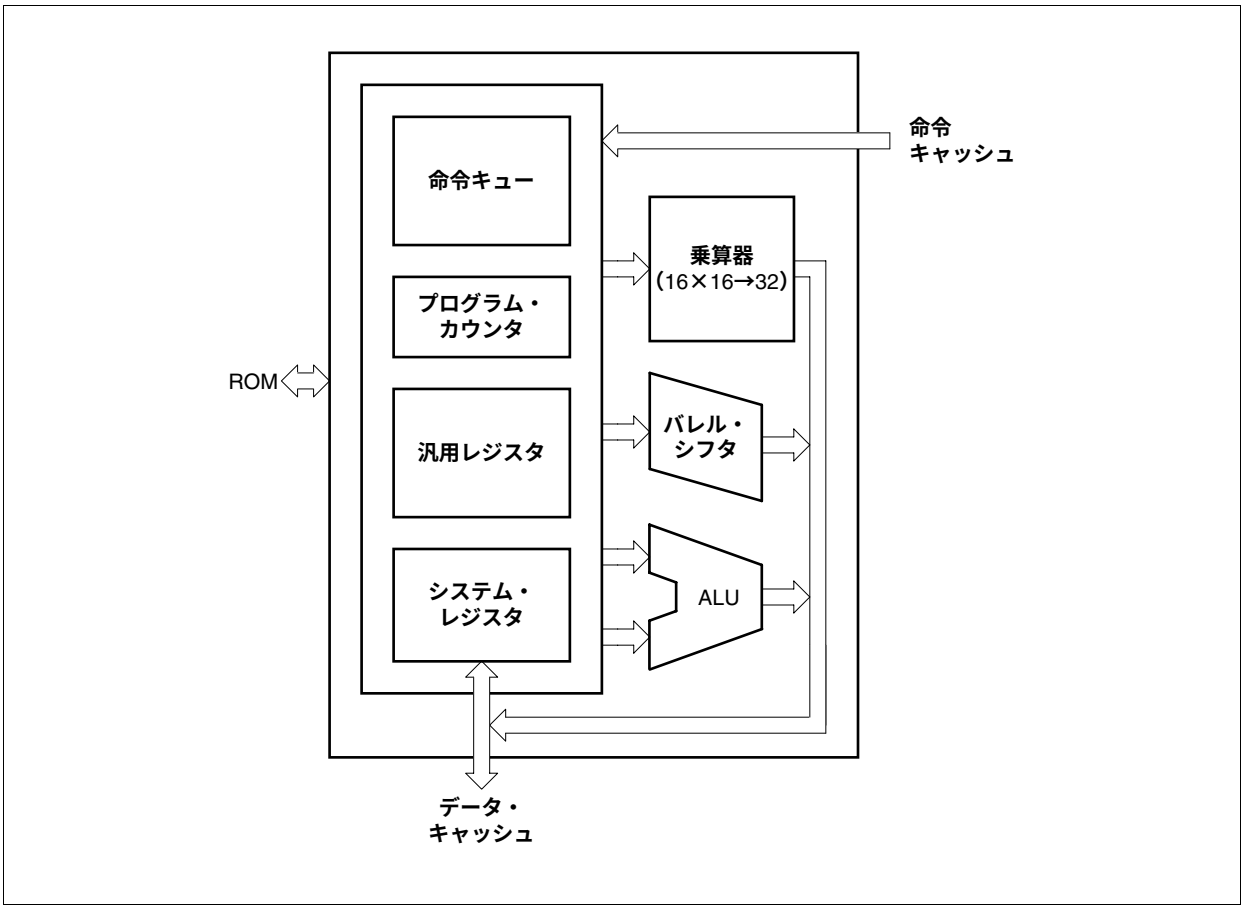
1.2 内部構成

V850ES 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5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（16ビット × 16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。

内部ブロック図を次に示します。

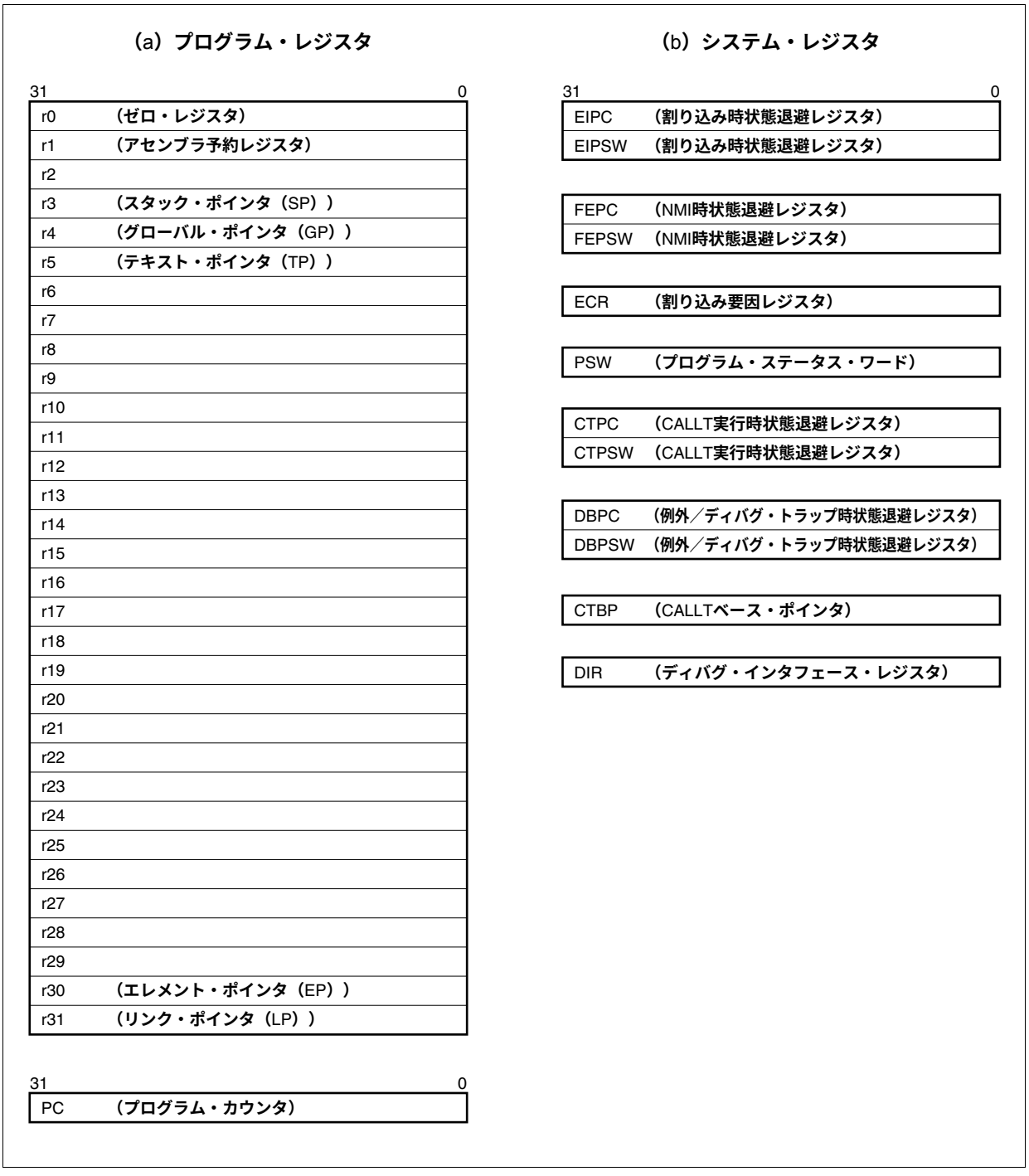
図1－1　V850ES CPUの内部ブロック図



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，実行環境の制御をするシステム・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



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r2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。

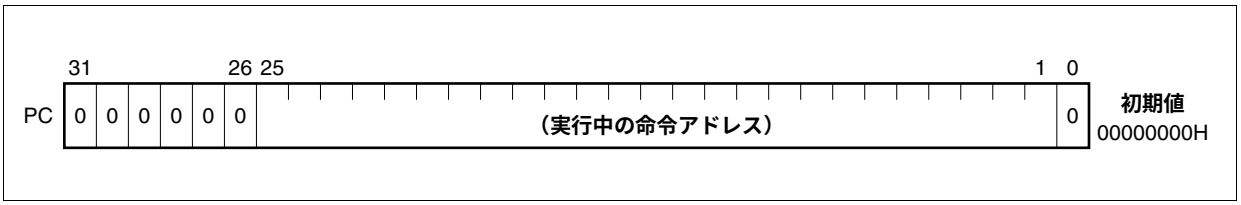
(c) r2

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26ビットが有効で，ビット31-26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25からビット26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，CPU の状態制御，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，システム・レジスタのロード／ストア命令（LDSR, 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・レジスタ番号

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
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）	×	○
22-31	（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））	×	×

注意　LDSR命令によりEIPCかFEPC，またはCTPCのビット0をセット（1）したあと，割り込み処理を行い，RETI命令で復帰する際に，ビット0の値は無視されます（PCのビット0が0固定のため）。EIPC, FEPC, CTPCに値を設定する場合は，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

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とEIPSWがあります。

ソフトウエア例外やマスカブル割り込みが発生した場合，プログラム・カウンタ（PC）の内容がEIPCに，プログラム・ステータス・ワード（PSW）の内容がEIPSW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）。

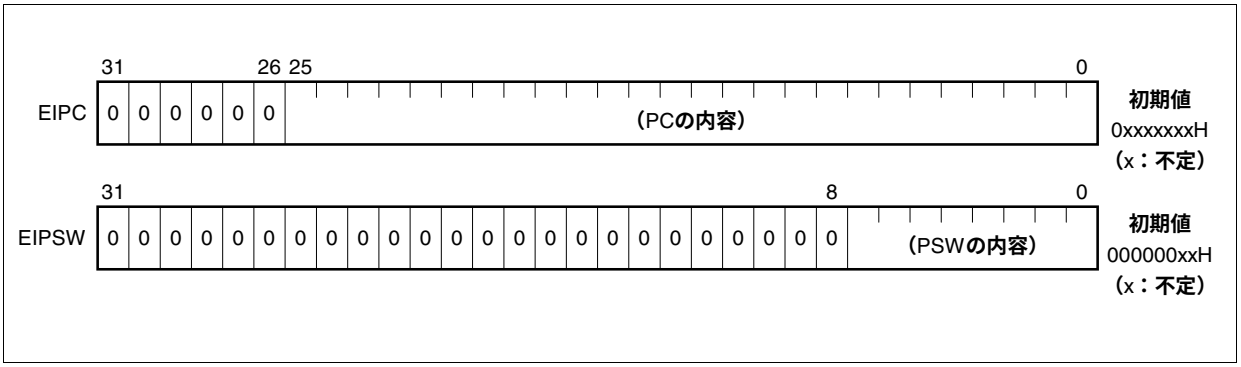
EIPCには，一部の命令を除き，ソフトウエア例外やマスカブル割り込みが発生したときに実行していた命令の次の命令のアドレスが退避されます（表6－1　割り込み，例外コード一覧参照）。

EIPSWには，現在のPSWの内容が退避されます。

割り込み時状態レジスタは，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のビット31-26とEIPSWのビット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2－3　割り込み時状態退避レジスタ（EIPC, EIPSW）



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とFEPSWがあります。

ノンマスカブル割り込み（NMI）が発生した場合，プログラム・カウンタ（PC）の内容がFEPCに，プログラム・ステータス・ワード（PSW）の内容がFEPSW に退避されます。

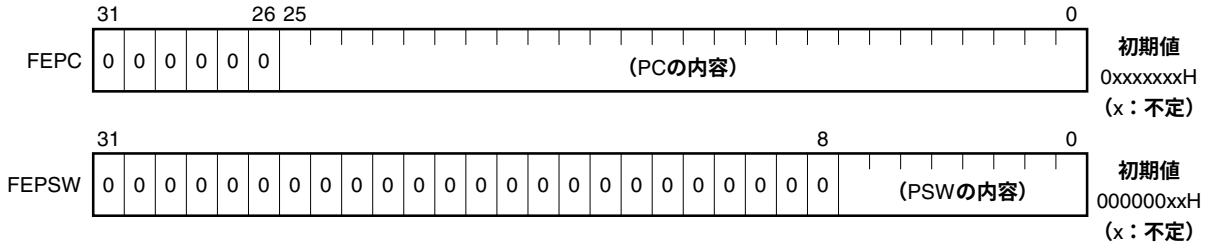
FEPCには，一部の命令を除き，NMIが発生したときに実行していた命令の次の命令のアドレスが退避されます（表6－1　割り込み，例外コード一覧参照）。

FEPSWには，現在のPSWの内容が退避されます。

NMI時状態レジスタは，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のビット31-26とFEPSWのビット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）。

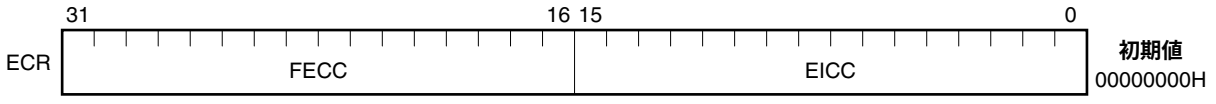
図2－4　NMI時状態退避レジスタ（FEPC, FEPSW）



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
15-0	EICC	例外，マスカブル割り込みの例外コード

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31-8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2－6　プログラム・ステータス・ワード（PSW）（1/2）

	31																						8	7	6	5	4	3	2	1	0	
PSW	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	NP	EP	ID	SAT	CY	OV	S	Z	初期値 00000020H

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。 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。 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 1：割り込み不可
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。 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。 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。 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。 1：演算の結果は負であった。

備考　注の説明は次ページに記載しています。

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
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。 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とCTPSWがあります。

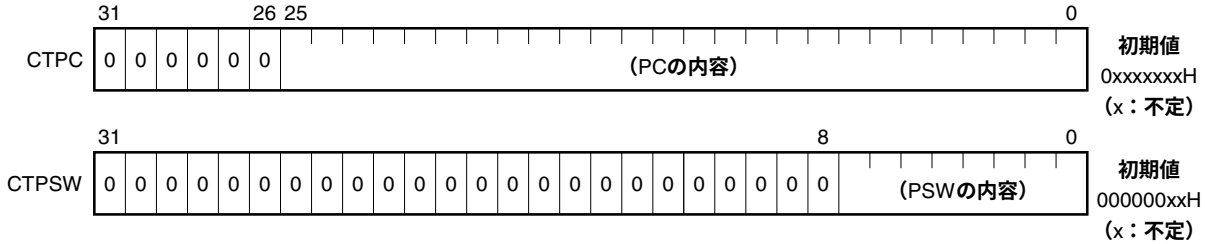
CALLT命令が実行されると，プログラム・カウンタ（PC）の内容がCTPCに，プログラム・ステータス・ワード（PSW）の内容がCTPSWに退避されます。

CTPC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には，現在のPSWの内容が退避されます。

なお，CTPCのビット31-26とCTPSWのビット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2－7　CALLT実行時状態退避レジスタ（CTPC, CTPSW）



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として，DBPCとDBPSWがあります。

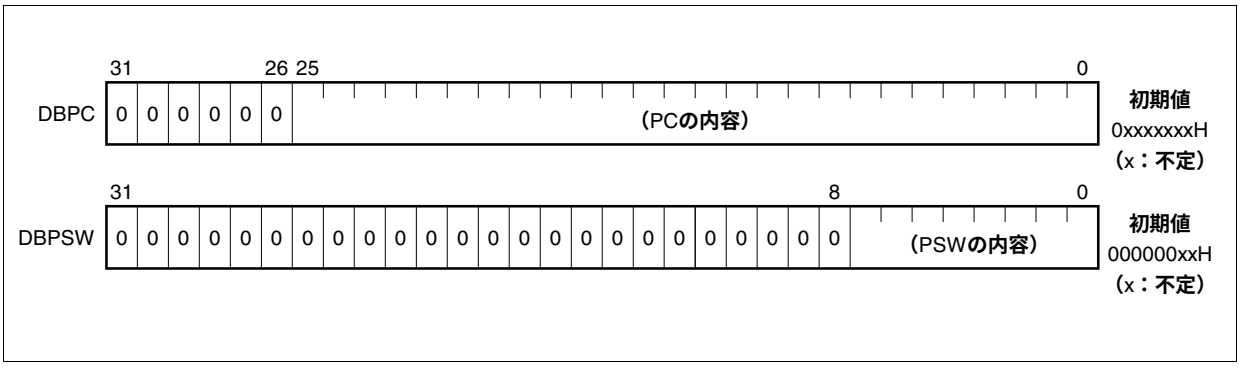
例外トラップ，またはディバグ・トラップが発生すると，プログラム・カウンタ（PC）の内容がDBPCに，プログラム・ステータス・ワード（PSW）の内容がDBPSWに退避されます。

DBPCに退避される内容は，例外トラップ，またはディバグ・トラップ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です。

DBPSWには，現在のPSWの内容が退避されます。

なお，DBPCのビット31-26とDBPSWのビット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2－8　例外／ディバグ・トラップ時状態退避レジスタ（DBPC, DBPSW）

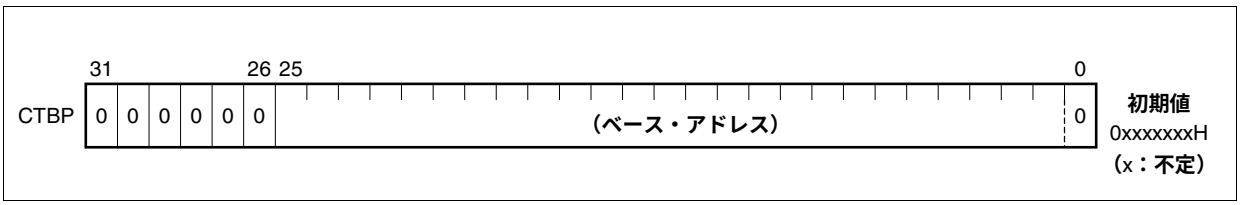


2. 2. 7　CALLTベース・ポインタ（CTBP）

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31-26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）

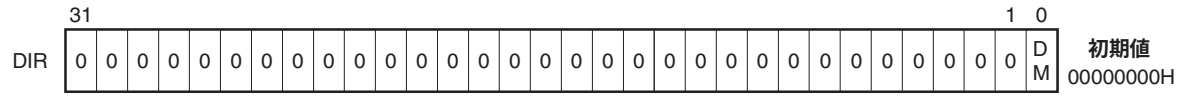
ディバグ・インタフェース・レジスタ（DIR）は，通常モード／ディバグ・モードの状態を示します。

DM ビットは，例外トラップ時と DBTRAP 命令時にセット（1）され，DBRET 命令によりクリア（0）されます。

STSR 命令で DIR レジスタの内容を汎用レジスタに設定することにより，DIR レジスタの内容をリードすることができます。また，DIR レジスタにライトすることはできません。

なお，ビット 31-1 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2－10　ディバグ・インタフェース・レジスタ（DIR）



ビット位置	ビット名	意　味
0	DM	例外トラップ時とDBTRAP命令時にセット（1）され，DBRET命令によりクリア（0）されます。 0：通常モード 1：ディバグ・モード

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**

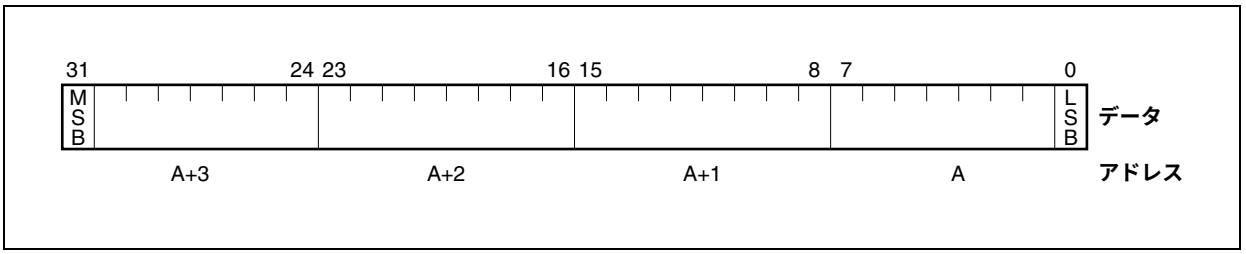
また，データ長として，ワード（32ビット），ハーフワード（16ビット），バイト（8ビット）があります。これらのデータは，バイト0が常に最下位（最右端）バイトである構成になっています（リトル・エンディアン）。

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。

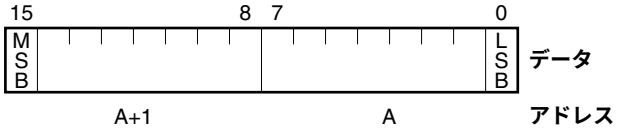
注　ミス・アライン・アクセス許可の状態では，ハーフワード・アクセス，ワード・アクセスにかかわらず，すべてのバイト境界にアクセスできます。3. 3　**データ・アラインメント**を参照してください。



(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。

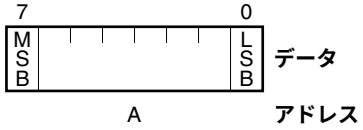
注　ミス・アライン・アクセス許可の状態では，ハーフワード・アクセス，ワード・アクセスにかかわらず，すべてのバイト境界にアクセスできます。3. 3　**データ・アラインメント**を参照してください。



(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。

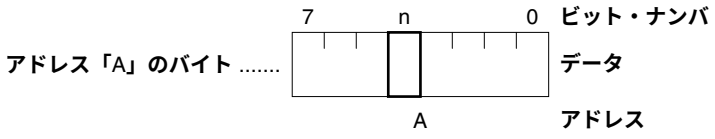
注　ミス・アライン・アクセス許可の状態では，ハーフワード・アクセス，ワード・アクセスにかかわらず，すべてのバイト境界にアクセスできます。3. 3　**データ・アラインメント**を参照してください。



(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・アライン機能を内蔵し，メモリに割り当てられるデータは，データの形式（ワード・データ，ハーフワード・データ）にかかわらず，すべてのアドレスに対して配置できます。ただし，ワード・データ，ハーフワード・データの場合，ワード・データはワード境界（アドレスの下位2ビットが0），ハーフワード・データはハーフワード境界（アドレスの下位1ビットが0）に整列（アライン）していないと，バス・サイクルが最低1回は余分に発生し，バス効率が低下します。

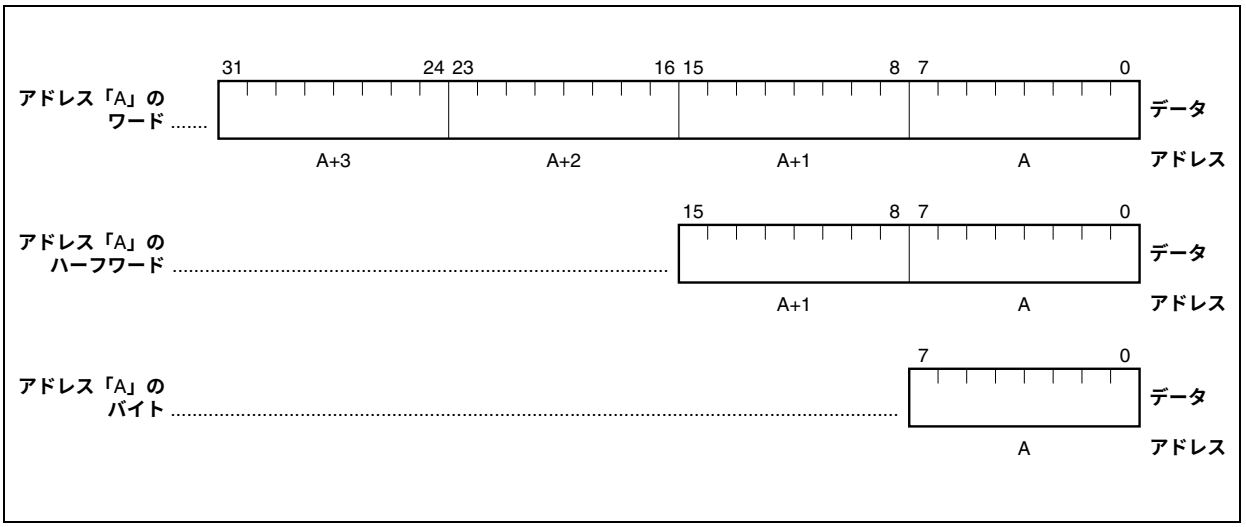
第4章　アドレス空間

V850ES 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）。

2バイト構成のデータ形式をハーフワード，4バイト構成のデータ形式をワードと呼びます。

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。



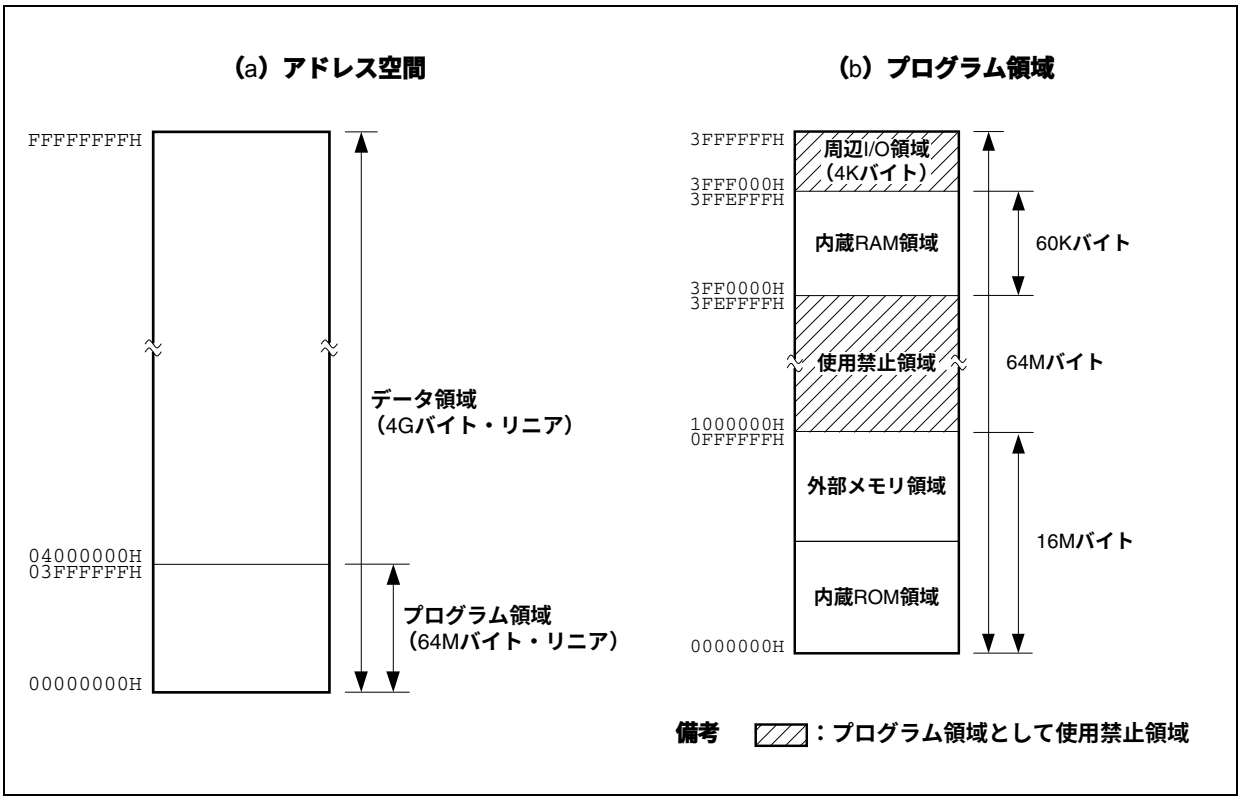
4. 1　メモリ・マップ

V850ES 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64M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。ただし，プログラム領域として使用可能な領域は，最大16 Mバイトのリニア・アドレス空間と内蔵RAM領域（最大60 Kバイト）になります。

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（＋2）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または22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とビット21が符号ビット（S）となります。

JARL disp22, reg2命令，JR disp22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）

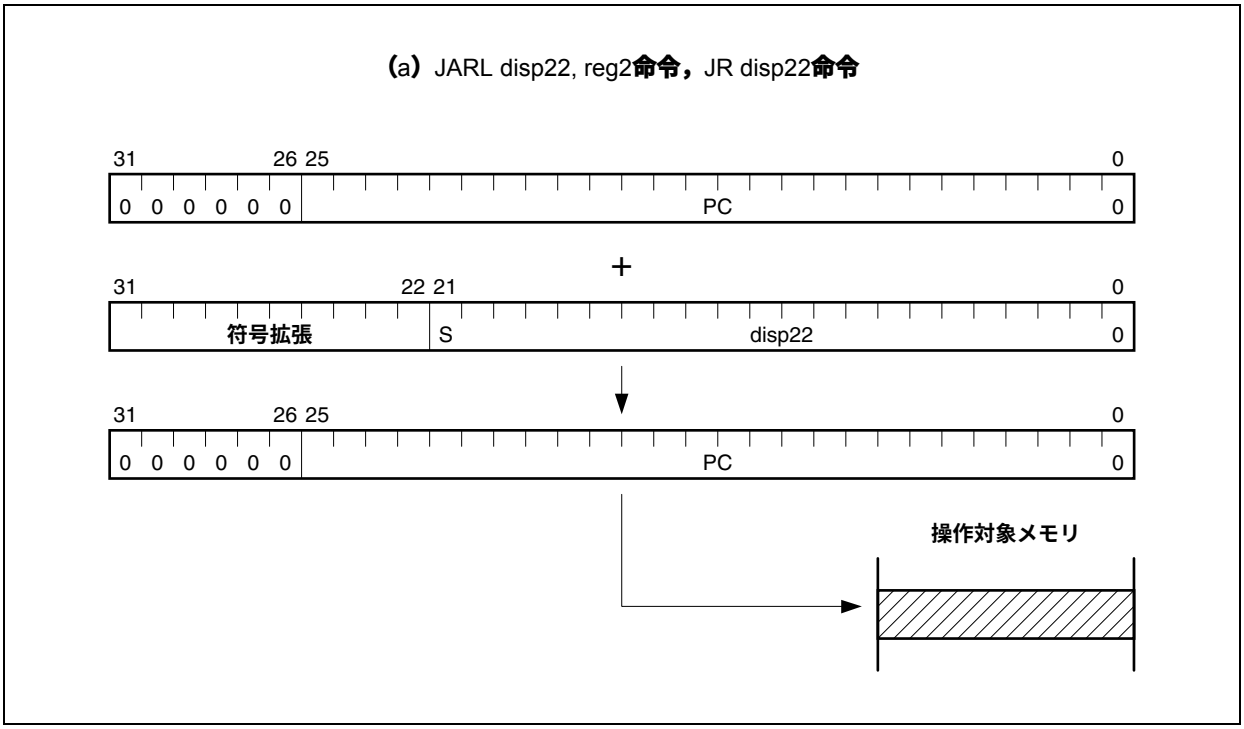
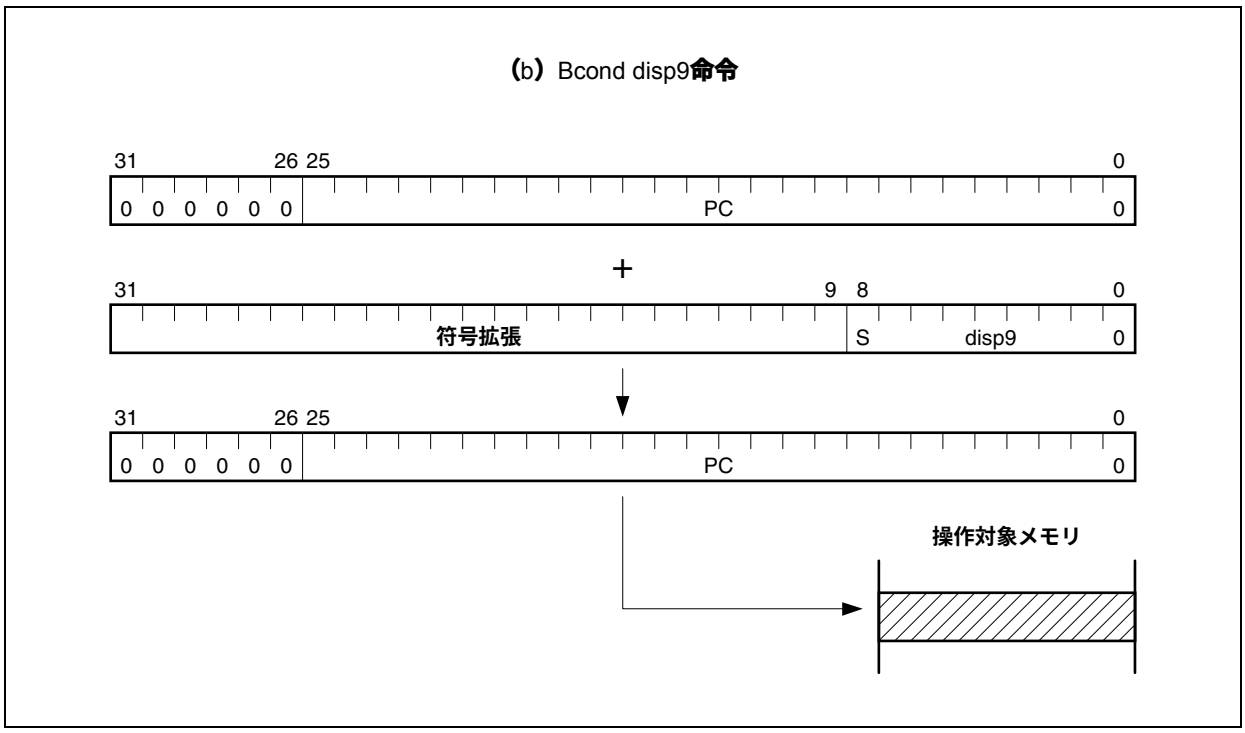


図4－2　レラティブ・アドレシング（2/2）

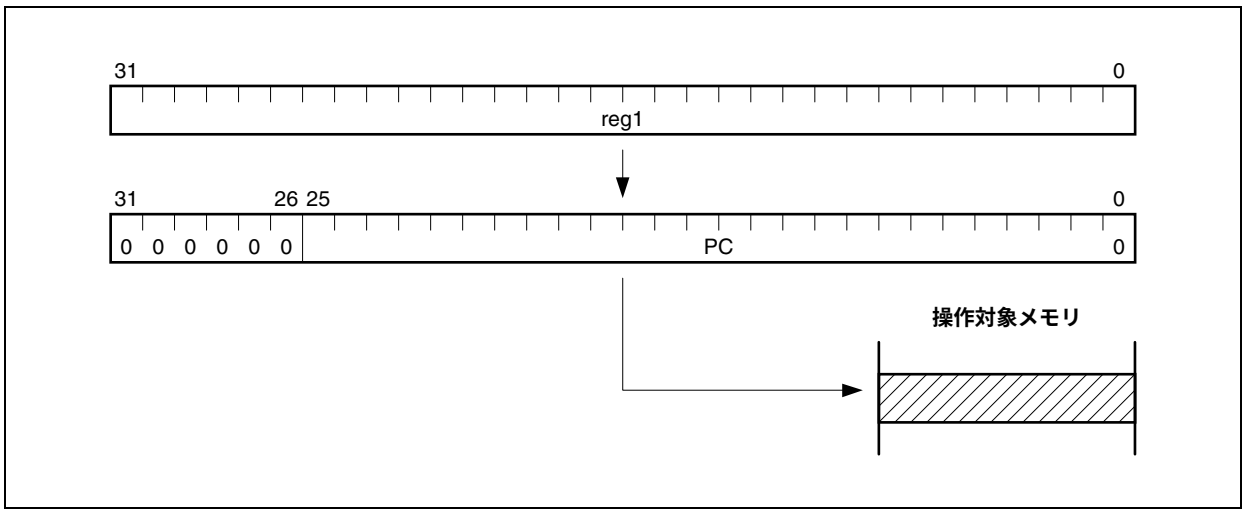


（2）レジスタ・アドレシング（レジスタ間接）

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）



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

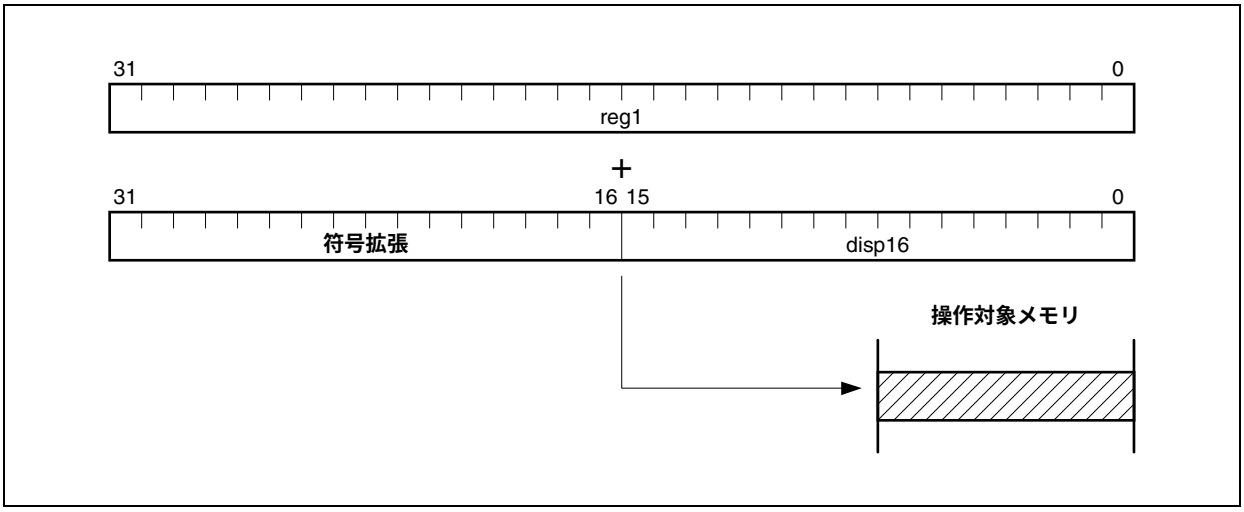
ベースト・アドレシングには，次に示す2種類があります。

(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4－4　ベースト・アドレシング（タイプ1）

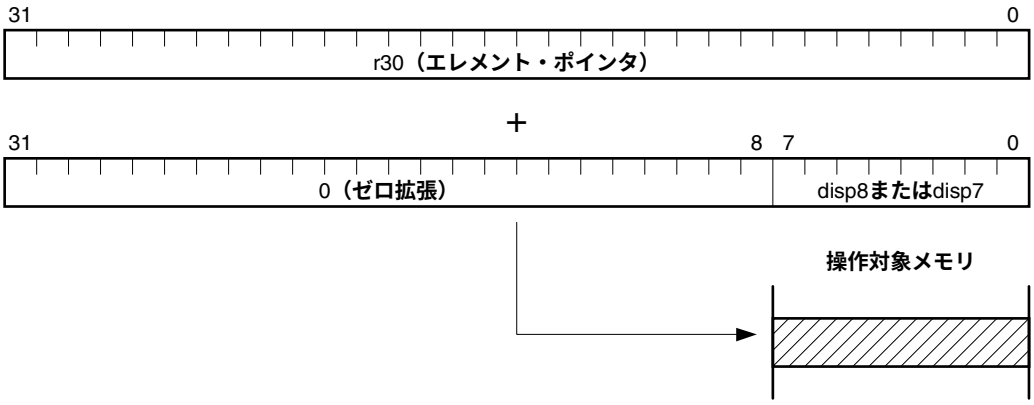


(b) タイプ2

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4－5　ベースト・アドレシング（タイプ2）



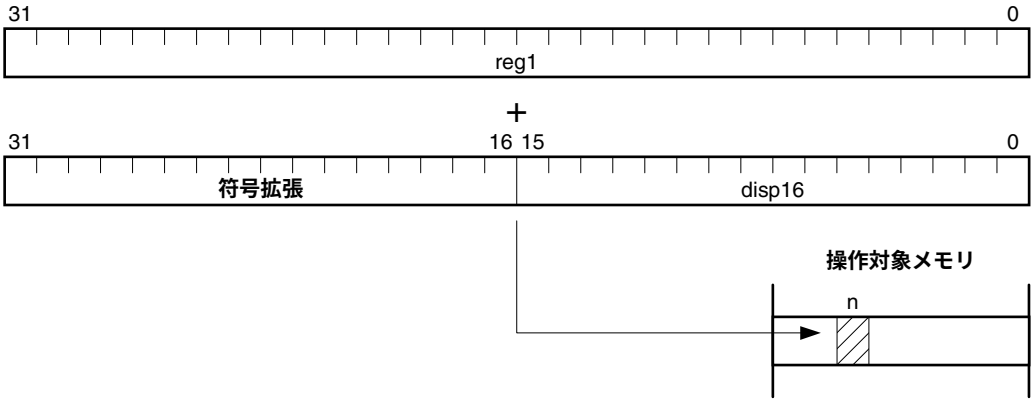
備考　バイト・アクセス：disp7
ハーフワード・アクセス，ワード・アクセス：disp8

(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4－6　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。

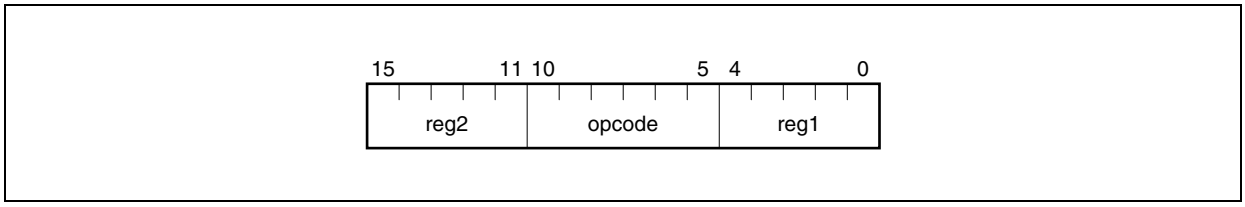
実際に命令がメモリに格納されるときは次のように配置されます。

- 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。

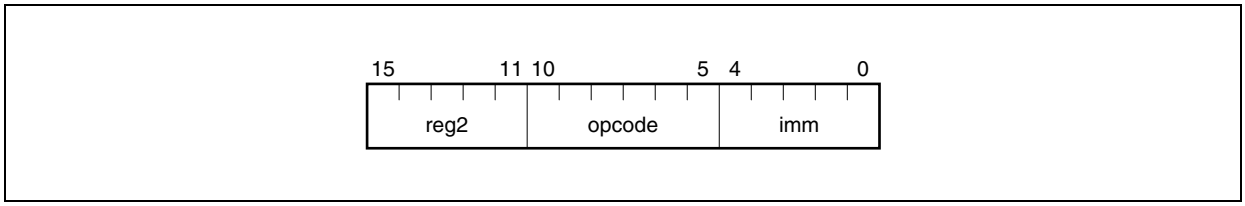
（1）reg-reg**命令形式（**Format I**）**

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。



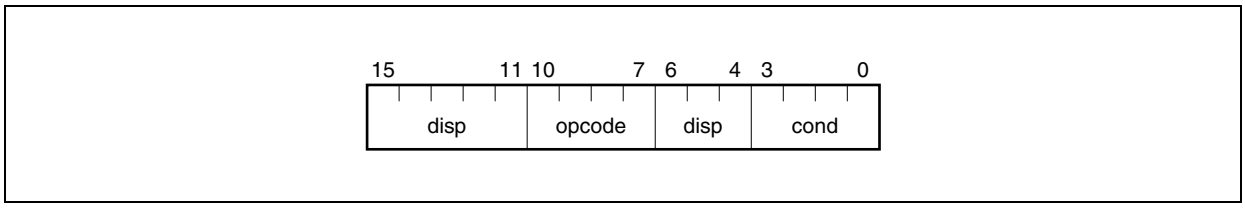
（2）imm-reg**命令形式（**Format II**）**

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。



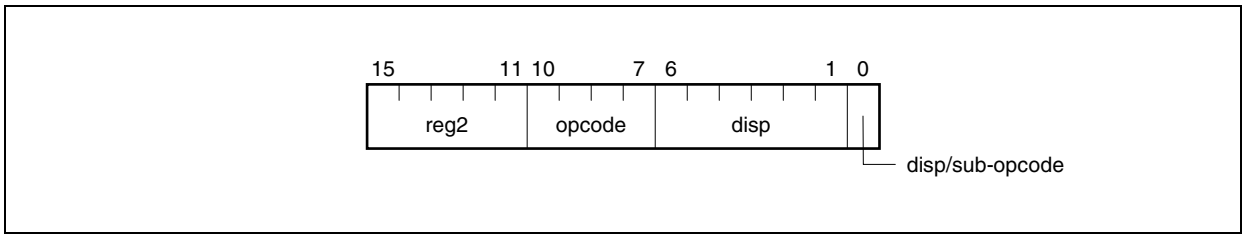
(3) 条件分岐命令形式 (Format III)

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。

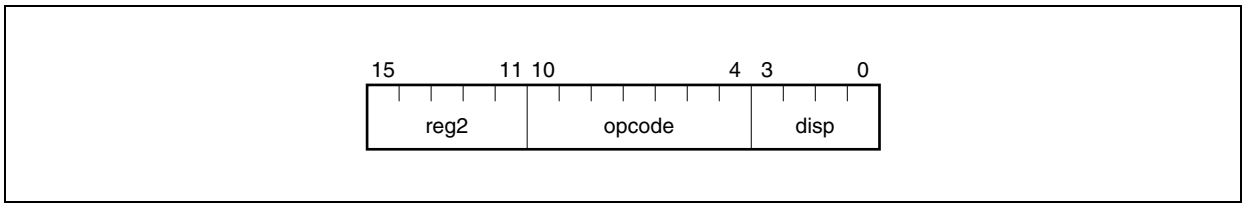


(4) ロード／ストア命令16ビット形式 (Format IV)

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。

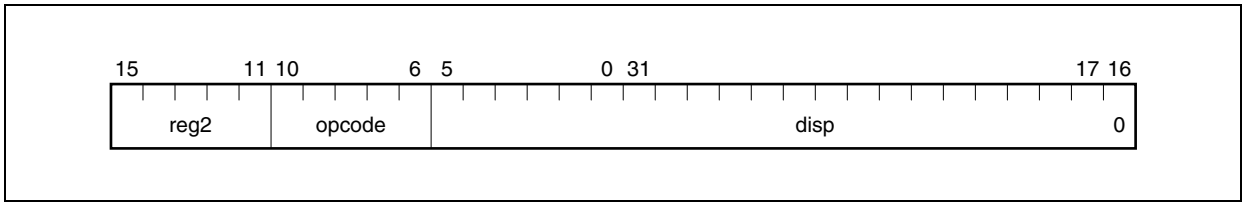


または，7ビットのオペコード・フィールドと1つの汎用レジスタ指定フィールド，4ビットのディスプレースメント・フィールドを持つ16ビット長命令形式。



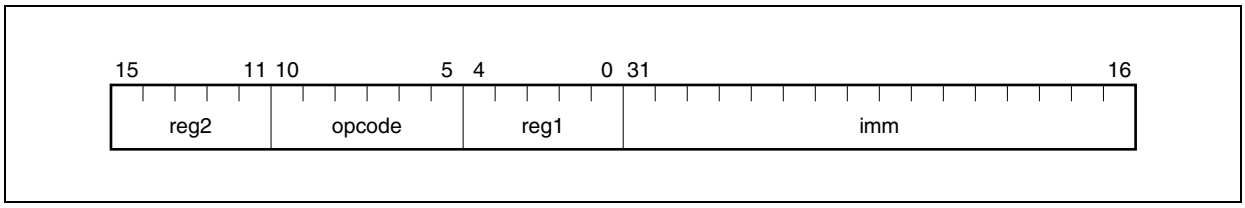
(5) ジャンプ命令形式 (Format V)

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。



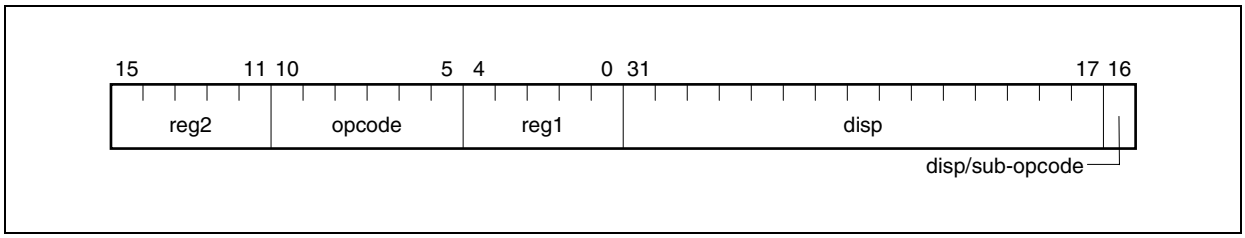
(6) 3オペランド命令形式（Format VI）

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。



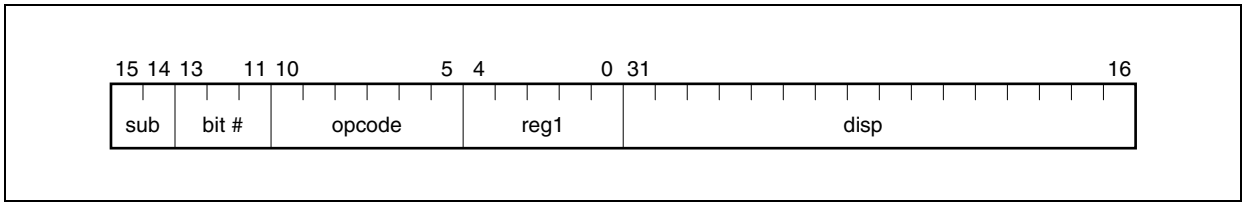
(7) ロード／ストア命令32ビット形式（Format VII）

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。



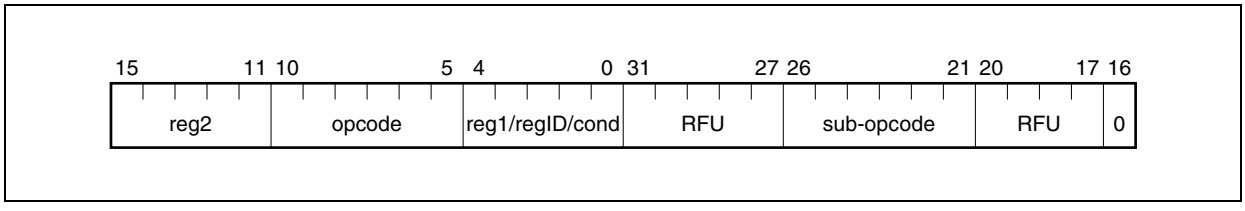
(8) ビット操作命令形式（Format VIII）

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。



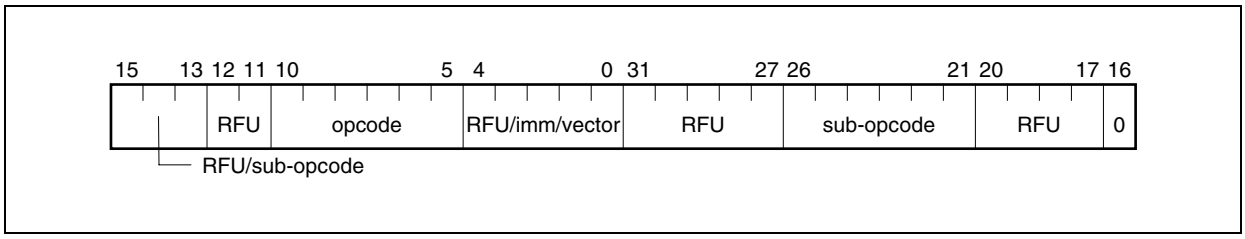
(9) 拡張命令形式1（Format IX）

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（regID）または条件コード・フィールド（cond）の場合あり）を持つ32ビット長命令形式。



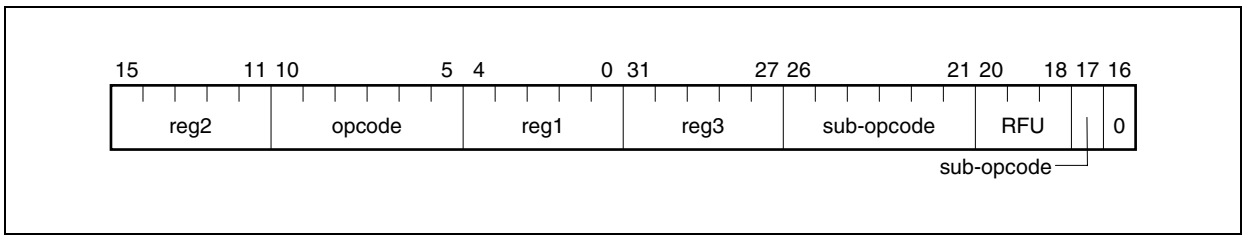
（10）拡張命令形式2（Format X）

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。



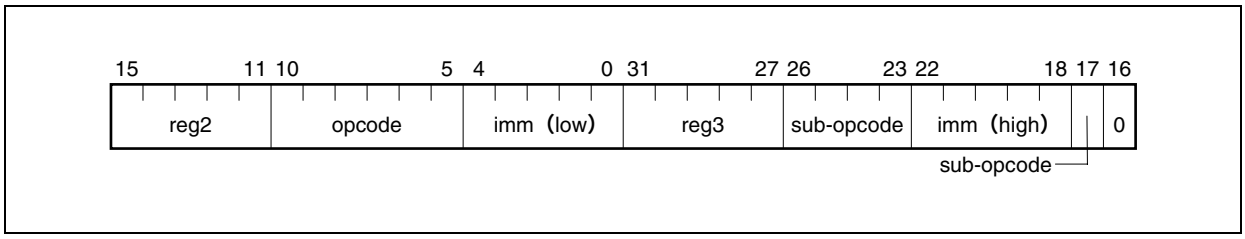
（11）拡張命令形式3（Format XI）

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。



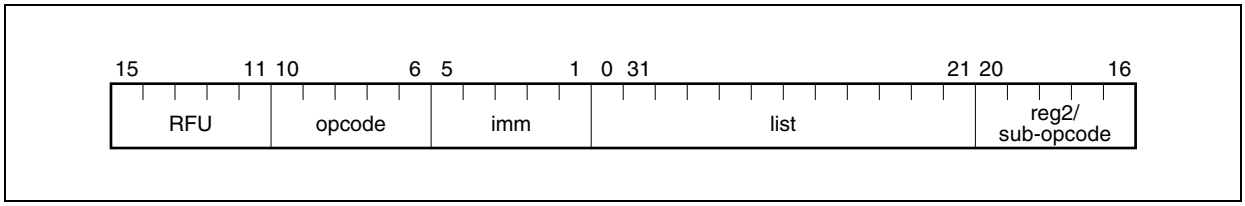
（12）拡張命令形式4（Format XII）

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。



（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。



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，1-5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) 算術演算命令

加減算，除算，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
- CMOV　：Conditional move
- CMP　：Compare
- DIV　：Divide word
- DIVH　：Divide half-word
- DIVHU　：Divide half-word unsigned
- DIVU　：Divide word unsigned
- MOV　：Move
- MOVEA　：Move effective address
- MOVHI　：Move high half-word
- SASF　：Shift and set flag condition
- SETF　：Set flag condition
- SUB　：Subtract
- SUBR　：Subtract reverse

(5) 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

(6) 論理演算命令

論理演算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。

- AND　：AND
- ANDI　：AND immediate
- BSH　：Byte swap half-word
- BSW　：Byte swap word
- HSW　：Half-word swap word
- NOT　：NOT
- OR　：OR
- ORI　：OR immediate

- SAR　：Shift arithmetic right
- SHL　：Shift logical left
- SHR　：Shift logical right
- SXB　：Sign extend byte
- SXH　：Sign extend half-word
- TST　：Test
- XOR　：Exclusive OR
- XORI　：Exclusive OR immediate
- ZXB　：Zero extend byte
- ZXH　：Zero extend half-word

(7) 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

(8) ビット操作命令

メモリの指定されたビット・データに対して，論理演算を行います。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

(9) 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

（10）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，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）
list ×	×個のレジスタ・リスト

表5－2　オペレーションの凡例（1/2）

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。

表5－2　オペレーションの凡例（2/2）

略　号	意　味
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
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
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・リスト中のプログラム・レジスタを指定する1ビット分データ

＜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]

(1)

15															0
r	r	r	r	r	0	0	1	1	1	0	R	R	R	R	R

(2)

15															0
r	r	r	r	r	0	1	0	0	1	0	i	i	i	i	i

[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。

＜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
ddddd1011dddCCCC	

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5－4　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。

＜論理演算命令＞

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。**

＜論理演算命令＞

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の内容をCTPCとCTPSW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。割り込みから復帰したあとに再実行されます。

＜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］

(1)

15	0	31	16
10bbb111110RRRRR		ddddddddddddddddd	

(2)

15	0	31	16
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。

＜算術演算命令＞

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］

(1)

15	0	31	16
rrrrr111111RRRRR		wwwww011001cccc0	

(2)

15	0	31	16
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。表5－5　条件コード一覧で示されているコードのうちの1つを条件コード「cccc」として指定してください。

（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。表5－5　条件コード一覧で示されているコードのうちの1つを条件コード「cccc」として指定してください。

［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]

(1)

15															0
r	r	r	r	r	0	0	1	1	1	1	R	R	R	R	R

(2)

15															0
r	r	r	r	r	0	1	0	0	1	1	i	i	i	i	i

[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から読み出した値が設定される
OV　CTPSWから読み出した値が設定される
S　CTPSWから読み出した値が設定される
Z　CTPSWから読み出した値が設定される
SAT　CTPSW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とCTPSWから取り出します。
<2> 取り出した復帰PCとPSWをPCとPSW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から読み出した値が設定される
OV　DBPSWから読み出した値が設定される
S　DBPSWから読み出した値が設定される
Z　DBPSWから読み出した値が設定される
SAT　DBPSW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＞

Debug trap

DBTRAP

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
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の内容を，それぞれDBPCとDBPSWに退避し，PSWのNP, EP, IDフラグをセット（1）します。
続いて，PCに例外トラップのハンドラ・アドレス（00000060H）をセットし，制御を移します。NP, EP, IDフラグ以外のPSWの各フラグは影響を受けません。
なお，DBPC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＞

Function dispose

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］

(1)

15	0	31	16
0000011001iiiiiL		LLLLLLLLLL00000	

(2)

15	0	31	16
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。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。**

［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に示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に示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，下方にロードされます（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・アクセスがイネーブルであっても，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命令の実行を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。

＜算術演算命令＞

Divide word

DIV

（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。

＜算術演算命令＞

Divide half-word

DIVH

（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]

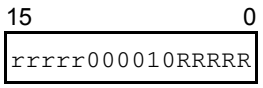
GR [reg3] ← GR [reg2] % GR [reg1]

［フォーマット］　（1）Format I

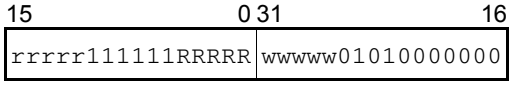
（2）Format XI

［オペコード］

（1）



（2）



［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。

＜算術演算命令＞

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。

＜算術演算命令＞

Divide word unsigned

DIVU

（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（NMI入力）
- 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またはFEPCには，この命令の次の命令アドレスが格納されます。

＜論理演算命令＞

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］　JARL　disp22, reg2

［オペレーション］　GR [reg2] ← PC + 4

PC ← PC + sign-extend (disp22)

［フォーマット］　Format V

［オペコード］

15	0	31	16
rrrrr11110dddddd		ddddddddddddddd0	

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

［フ ラ グ］　CY　–

OV　–

S　–

Z　–

SAT　–

［説　明］　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。

［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］　JMP　[reg1]

［オペレーション］　PC ← GR [reg1]

［フォーマット］　Format I

［オペコード］

15	0
00000000011RRRRR	

［フ ラ グ］　CY　–
OV　–
S　–
Z　–
SAT　–

［説　明］　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。

［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。なお，コールに相当するJARL命令では，復帰PCを汎用レジスタreg2に格納してください。

＜分岐命令＞

JR	Jump relative 無条件分岐（PC相対）

［命令形式］　JR　disp22

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

［フォーマット］　Format V

［オペコード］

15	0 31	16
0000011110dddddd	ddddddddddddddd0	

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

［フ ラ グ］　CY　–
OV　–
S　–
Z　–
SAT　–

［説　明］　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。

［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 **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］　**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。**
アクセス対象の資源（内蔵ROM**，内蔵**RAM**，内蔵周辺**I/O**，外部メモリ）により，バス・サイクルが入れ替わる可能性があります（同じ資源に対するアクセスであれば，バス・サイクルが入れ替わることはありません）。**

＜ロード命令＞

LD.BU	Load byte unsigned 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。

［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。
アクセス対象の資源（内蔵ROM，内蔵RAM，内蔵周辺I/O，外部メモリ）により，バス・サイクルが入れ替わる可能性があります（同じ資源に対するアクセスであれば，バス・サイクルが入れ替わることはありません）。

＜ロード命令＞

LD.H	Load half-word **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］　**ミス・アライン・アクセスが発生した場合の注意事項については，**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。**

アクセス対象の資源（内蔵ROM**，内蔵**RAM**，内蔵周辺**I/O**，外部メモリ）により，バス・サイクルが入れ替わる可能性があります（同じ資源に対するアクセスであれば，バス・サイクルが入れ替わることはありません）。**

＜ロード命令＞

LD.HU	Load half-word unsigned 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，ワード長までゼロ拡張し，汎用レジスタ**reg2**に格納します。**

［注　意］　**ミス・アライン・アクセスが発生した場合の注意事項については，**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。**

アクセス対象の資源（内蔵ROM**，内蔵**RAM**，内蔵周辺**I/O**，外部メモリ）により，バス・サイクルが入れ替わる可能性があります（同じ資源に対するアクセスであれば，バス・サイクルが入れ替わることはありません）。**

＜ロード命令＞

LD.W	Load word **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］　**ミス・アライン・アクセスが発生した場合の注意事項については，**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。**

アクセス対象の資源（内蔵ROM**，内蔵**RAM**，内蔵周辺**I/O**，外部メモリ）により，バス・サイクルが入れ替わる可能性があります（同じ資源に対するアクセスであれば，バス・サイクルが入れ替わることはありません）。**

＜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。

＜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］

(1)

15	0	31	16
rrrrr111111RRRRR		wwwww01000100000	

(2)

15	0	31	16
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。
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。

［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。

［注　意］　（1）「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

★　（2）mul/mulu 命令に関する制限事項については付録 A 注意事項を参照してください。

＜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]

(1)

15	0
rrrrr000111RRRRR	

(2)

15	0
rrrrr010111iiiii	

[フ ラ グ]

CY	–
OV	–
S	–
Z	–
SAT	–

[説　明]　**（1）汇用レジスタ**reg2**の下位ハーフワード・データに汇用レジスタ**reg1**のハーフワード・データを乗算し，その結果を汇用レジスタ**reg2**にワード・データとして格納します。汇用レジスタ**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）汇用レジスタ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，その結果を汇用レジスタ**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，汇用レジスタ**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 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]

(1)

15	0	31	16
rrrrr111111RRRRR		wwwww01000100010	

(2)

15	0	31	16
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。
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。

[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。

［注　意］　（1）「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

★　（2）mul/mulu 命令に関する制限事項については付録 A 注意事項を参照してください。

＜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＞

NOT bit

NOT1

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］

(1)

15	0	31	16
01bbb111110RRRRR		ddddddddddddddddd	

(2)

15	0	31	16
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＞

Function prepare

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

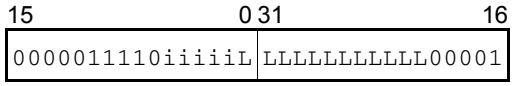
repeat 1 step above until all regs in list12 is stored

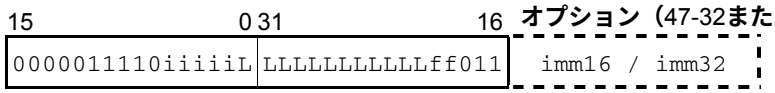
sp ← sp – zero-extend (imm5)

ep ← sp/imm

［フォーマット］　Format XIII

［オペコード］





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，LLLLLLLLLLLL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。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。

[フ ラ グ]

CY –
OV –
S –
Z –
SAT –

[説　明]

（1）list12に表示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に表示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＞

Return from trap or interrupt

RETI

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またはEIPSWから読み出した値が設定される
OV	FEPSWまたはEIPSWから読み出した値が設定される
S	FEPSWまたはEIPSWから読み出した値が設定される
Z	FEPSWまたはEIPSWから読み出した値が設定される
SAT	FEPSWまたはEIPSW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，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) PSWのEPフラグが1の場合，PSWのNPフラグの状態にかかわらず，EIPC, EIPSWから復帰PC，PSWを取り出します。
PSWのEPフラグが0かつ PSWのNPフラグが1の場合，FEPC，FEPSWから復帰PC, PSWを取り出します。
PSWのEPフラグが0かつ PSWのNPフラグが0の場合，EIPC，EIPSW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を正常にリストアするために，RETI命令の直前でPSWの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。

＜論理演算命令＞

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

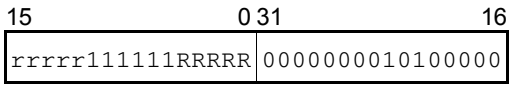
［命令形式］　（1）SAR　reg1, reg2
（2）SAR　imm5, reg2

［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

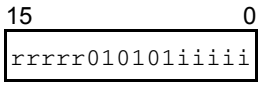
［フォーマット］　（1）Format IX
（2）Format II

［オペコード］

(1)



(2)



［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	ー

［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。

＜算術演算命令＞

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に1がセットされます。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に0がセットされます。
表5－5　条件コード一覧で示されているコードのうちの1つを条件コード「cccc」として指定してください。

［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

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

[フォーマット]　(1) Format I
(2) Format II

[オペコード]

(1)

15															0
r	r	r	r	r	0	0	0	1	1	0	R	R	R	R	R

(2)

15															0
r	r	r	r	r	0	1	0	0	0	1	i	i	i	i	i

[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。

[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］　SATSUB　reg1, reg2

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

［フォーマット］　Format I

［オペコード］

15	0
rrrrr000101RRRRR	

［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。

［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にデータをロードしてください。

＜飽和演算命令＞

Saturated subtract reverse

SATSUBR

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にデータをロードしてください。

＜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。

＜算術演算命令＞

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。条件コード「cccc」には，表5－5　条件コード一覧に示す条件コードを指定します。

［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。

表5－5　条件コード一覧

条件コード（cccc）	条件名	条件式
0000	V	OV = 1
1000	NV	OV = 0
0001	C/L	CY = 1
1001	NC/NL	CY = 0
0010	Z	Z = 1
1010	NZ	Z = 0
0011	NH	(CY or Z) = 1
1011	H	(CY or Z) = 0
0100	S/N	S = 1
1100	NS/P	S = 0
0101	T	always（無条件）
1101	SA	SAT = 1
0110	LT	(S xor OV) = 1
1110	GE	(S xor OV) = 0
0111	LE	((S xor OV) or Z) = 1
1111	GT	((S xor OV) or Z) = 0

＜論理演算命令＞

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

［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)

［フォーマット］　（1）Format IX
（2）Format II

［オペコード］

(1)

15	0	31	16
rrrrr111111RRRRR		0000000011000000	

(2)

15	0
rrrrr010110iiiii	

［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	ー

［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。

＜論理演算命令＞

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

［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）

［フォーマット］　（1）Format IX
（2）Format II

［オペコード］

(1)

15	0	31	16
rrrrr111111RRRRR		0000000010000000	

(2)

15	0
rrrrr010100iiiii	

［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	ー

［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。

＜ロード命令＞

SLD.B	Short format load byte **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。**
アクセス対象の資源（内蔵ROM，内蔵RAM，内蔵周辺I/O，外部メモリ）により，バス・サイクルが入れ替わる可能性があります（同じ資源に対するアクセスであれば，バス・サイクルが入れ替わることはありません）。

［注　意］　**（1）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。この場合，割り込みから復帰したあとに再度この命令を実行します。したがって，リード・サイクルによって状態が変わるI/OやFIFOタイプの資源などに対するアクセスには，割り込みが発生しないことが明確である場合を除き，LD命令を使用してください（LD命令やストア命令は，命令実行中に割り込みが発生してもバス・サイクルは再実行されません）。**
（2）sld命令と割り込み競合に関する制限事項については，付録A　注意事項を参照してください。

＜ロード命令＞

SLD.BU	Short format load byte unsigned **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。**
アクセス対象の資源（内蔵ROM，内蔵RAM，内蔵周辺I/O，外部メモリ）により，バス・サイクルが入れ替わる可能性があります（同じ資源に対するアクセスであれば，バス・サイクルが入れ替わることはありません）。

[注　意]　　**（1）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。この場合，割り込みから復帰したあとに再度この命令を実行します。したがって，リード・サイクルによって状態が変わるI/OやFIFOタイプの資源などに対するアクセスには，割り込みが発生しないことが明確である場合を除き，LD命令を使用してください（LD命令やストア命令は，命令実行中に割り込みが発生してもバス・サイクルは再実行されません）。**
（2）sld命令と割り込み競合に関する制限事項については，付録A　注意事項を参照してください。

＜ロード命令＞

SLD.H	Short format load half-word **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。**
アクセス対象の資源（内蔵ROM，内蔵RAM，内蔵周辺I/O，外部メモリ）により，バス・サイクルが入れ替わる可能性があります（同じ資源に対するアクセスであれば，バス・サイクルが入れ替わることはありません）。

［注　意］　**（1）ミス・アライン・アクセスが発生した場合の注意事項については，**3. 3　**データ・アラインメントを参照してください。**
また，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。この場合，割り込みから復帰したあとに再度この命令を実行します。したがって，リード・サイクルによって状態が変わるI/OやFIFOタイプの資源などに対するアクセスには，割り込みが発生しないことが明確である場合を除き，LD命令を使用してください（LD命令やストア命令は，命令実行中に割り込みが発生してもバス・サイクルは再実行されません）。
（2）sld命令と割り込み競合に関する制限事項については，付録A　**注意事項を参照してください。**

＜ロード命令＞

SLD.HU	Short format load half-word unsigned **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。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。**

［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。**
アクセス対象の資源（内蔵ROM**，内蔵**RAM**，内蔵周辺**I/O**，外部メモリ）により，バス・サイクルが入れ替わる可能性があります（同じ資源に対するアクセスであれば，バス・サイクルが入れ替わることはありません）。**

［注　意］　**（1）ミス・アライン・アクセスが発生した場合の注意事項については，**3. 3　**データ・アライメント**を参照してください。**
また，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。この場合，割り込みから復帰したあとに再度この命令を実行します。したがって，リード・サイクルによって状態が変わるI/O**やFIFO**タイプの資源などに対するアクセスには，割り込みが発生しないことが明確である場合を除き，**LD**命令を使用してください（**LD**命令やストア命令は，命令実行中に割り込みが発生してもバス・サイクルは再実行されません）。**
（2）sld**命令と割り込み競合に関する制限事項については，付録**A　**注意事項**を参照してください。**

＜ロード命令＞

SLD.W	Short format load word **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。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。**

[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。**
アクセス対象の資源（内蔵ROM**，内蔵**RAM**，内蔵周辺**I/O**，外部メモリ）により，バス・サイクルが入れ替わる可能性があります（同じ資源に対するアクセスであれば，バス・サイクルが入れ替わることはありません）。**

[注　意]　　**（1）ミス・アライン・アクセスが発生した場合の注意事項については，**3. 3　**データ・アライメント**を参照してください。**
また，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。この場合，割り込みから復帰したあとに再度この命令を実行します。したがって，リード・サイクルによって状態が変わるI/O**やFIFOタイプの資源などに対するアクセスには，割り込みが発生しないことが明確である場合を除き，**LD**命令を使用してください（**LD**命令やストア命令は，命令実行中に割り込みが発生してもバス・サイクルは再実行されません）。**
（2）sld**命令と割り込み競合に関する制限事項については，付録**A　**注意事項**を参照してください。**

＜ストア命令＞

SST.B	Short format store byte 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］　アクセス対象の資源（内蔵ROM，内蔵RAM，内蔵周辺I/O，外部メモリ）により，バス・サイクルが入れ替わる可能性があります（同じ資源に対するアクセスであれば，バス・サイクルが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 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・データを生成したアドレスに格納します。

［注　意］　ミス・アライン・アクセスが発生した場合の注意事項については，3. 3　**データ・アラインメント**を参照してください。

［補　足］　アクセス対象の資源（内蔵ROM，内蔵RAM，内蔵周辺I/O，外部メモリ）により，バス・サイクルが入れ替わる可能性があります（同じ資源に対するアクセスであれば，バス・サイクルが入れ替わることはありません）。

＜ストア命令＞

SST.W	Short format store word 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・データを生成したアドレスに格納します。**

[注　意]　**ミス・アライン・アクセスが発生した場合の注意事項については，**3. 3　**データ・アラインメントを参照してください。**

[補　足]　**アクセス対象の資源（内蔵**ROM**，内蔵**RAM**，内蔵周辺**I/O**，外部メモリ）により，バス・サイクルが入れ替わる可能性があります（同じ資源に対するアクセスであれば，バス・サイクルが入れ替わることはありません）。**

＜ストア命令＞

ST.B	Store byte 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］　アクセス対象の資源（内蔵ROM，内蔵RAM，内蔵周辺I/O，外部メモリ）により，バス・サイクルが入れ替わる可能性があります（同じ資源に対するアクセスであれば，バス・サイクルが入れ替わることはありません）。

＜ストア命令＞

ST.H	Store half-word 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・データを生成したアドレスに格納します。**

［注　意］　**ミス・アライン・アクセスが発生した場合の注意事項については，**3. 3　**データ・アラインメントを参照してください。**

［補　足］　**アクセス対象の資源（内蔵**ROM**，内蔵**RAM**，内蔵周辺**I/O**，外部メモリ）により，バス・サイクルが入れ替わる可能性があります（同じ資源に対するアクセスであれば，バス・サイクルが入れ替わることはありません）。**

＜ストア命令＞

ST.W	Store word **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・データを生成したアドレスに格納します。**

[注　意]　　**ミス・アライン・アクセスが発生した場合の注意事項については，**3. 3　**データ・アラインメントを参照してください。**

[補　足]　　**アクセス対象の資源（内蔵**ROM**，内蔵**RAM**，内蔵周辺**I/O**，外部メモリ）により，バス・サイクルが入れ替わる可能性があります（同じ資源に対するアクセスであれば，バス・サイクルが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
00000000010RRRRR	

［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。
(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。
(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。
(4)　(3) で生成されたターゲット・アドレスへ分岐。

＜論理演算命令＞

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。**

＜論理演算命令＞

Sign extend half-word

SXH

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 ← 割り込みコード
PSW.EP ← 1
PSW.ID ← 1
PC ← 00000040H（vectorが00H-0FHのとき）
00000050H（vectorが10H-1FH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**PSW**の**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**PSW**の**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・データとワード長までゼロ拡張した16ビット・イミーディエッ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。**

＜論理演算命令＞

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。**

＜論理演算命令＞

Zero extend half-word

ZXH

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

次に内蔵ROM，内蔵RAMを使用した場合における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（1/3）

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		
				i	r	l
ロード命令	LD.B	disp16 [reg1] , reg2	4	1	1	注1
	LD.H	disp16 [reg1] , reg2	4	1	1	注1
	LD.W	disp16 [reg1] , reg2	4	1	1	注1
	LD.BU	disp16 [reg1] , reg2	4	1	1	注1
	LD.HU	disp16 [reg1] , reg2	4	1	1	注1
	SLD.B	disp7 [ep] , reg2	2	1	1	注2
	SLD.BU	disp4 [ep] , reg2	2	1	1	注2
	SLD.H	disp8 [ep] , reg2	2	1	1	注2
	SLD.HU	disp5 [ep] , reg2	2	1	1	注2
	SLD.W	disp8 [ep] , reg2	2	1	1	注2
ストア命令	ST.B	reg2, disp16 [reg1]	4	1	1	1
	ST.H	reg2, disp16 [reg1]	4	1	1	1
	ST.W	reg2, disp16 [reg1]	4	1	1	1
	SST.B	reg2, disp7 [ep]	2	1	1	1
	SST.H	reg2, disp8 [ep]	2	1	1	1
	SST.W	reg2, disp8 [ep]	2	1	1	1
乗算命令	MUL	reg1, reg2, reg3	4	1	4	5
	MUL	imm9, reg2, reg3	4	1	4	5
	MULH	reg1, reg2	2	1	1	2
	MULH	imm5, reg2	2	1	1	2
	MULHI	imm16, reg1, reg2	4	1	1	2
	MULU	reg1, reg2, reg3	4	1	4	5
	MULU	imm9, reg2, reg3	4	1	4	5
算術演算命令	ADD	reg1, reg2	2	1	1	1
	ADD	imm5, reg2	2	1	1	1
	ADDI	imm16, reg1, reg2	4	1	1	1
	CMOV	cccc, reg1, reg2, reg3	4	1	1	1
	CMOV	cccc, imm5, reg2, reg3	4	1	1	1
	CMP	reg1, reg2	2	1	1	1
	CMP	imm5, reg2	2	1	1	1
	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

表5－6　命令実行クロック数一覧（2/3）

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		
				i	r	l
算術演算命令	DIVU	reg1, reg2, reg3	4	34	34	34
	MOV	reg1, reg2	2	1	1	1
	MOV	imm5, reg2	2	1	1	1
	MOV	imm32, reg1	6	2	2	2
	MOVEA	imm16, reg1, reg2	4	1	1	1
	MOVHI	imm16, reg1, reg2	4	1	1	1
	SASF	cccc, reg2	4	1	1	1
	SETF	cccc, reg2	4	1	1	1
	SUB	reg1, reg2	2	1	1	1
	SUBR	reg1, reg2	2	1	1	1
飽和演算命令	SATADD	reg1, reg2	2	1	1	1
	SATADD	imm5, reg2	2	1	1	1
	SATSUB	reg1, reg2	2	1	1	1
	SATSUBI	imm16, reg1, reg2	4	1	1	1
	SATSUBR	reg1, reg2	2	1	1	1
論理演算命令	AND	reg1, reg2	2	1	1	1
	ANDI	imm16, reg1, reg2	4	1	1	1
	BSH	reg2, reg3	4	1	1	1
	BSW	reg2, reg3	4	1	1	1
	HSW	reg2, reg3	4	1	1	1
	NOT	reg1, reg2	2	1	1	1
	OR	reg1, reg2	2	1	1	1
	ORI	imm16, reg1, reg2	4	1	1	1
	SAR	reg1, reg2	4	1	1	1
	SAR	imm5, reg2	2	1	1	1
	SHL	reg1, reg2	4	1	1	1
	SHL	imm5, reg2	2	1	1	1
	SHR	reg1, reg2	4	1	1	1
	SHR	imm5, reg2	2	1	1	1
	SXB	reg1	2	1	1	1
	SXH	reg1	2	1	1	1
	TST	reg1, reg2	2	1	1	1
	XOR	reg1, reg2	2	1	1	1
	XORI	imm16, reg1, reg2	4	1	1	1
	ZXB	reg1	2	1	1	1
	ZXH	reg1	2	1	1	1
★ 分岐命令	Bcond	disp9（条件成立時）	2	2 注3, 4	2 注3, 4	2 注3, 4
		disp9（条件不成立時）	2	1	1	1
	JARL	disp22, reg2	4	2 注4	2 注4	2 注4

表5－6　命令実行クロック数一覧（3/3）

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		
				i	r	l
分岐命令	JMP	[reg1]	2	3注4	3注4	3注4
	JR	disp22	4	2注4	2注4	2注4
ビット操作命令	CLR1	bit#3, disp16 [reg1]	4	3注5	3注5	3注5
	CLR1	reg2, [reg1]	4	3注5	3注5	3注5
	NOT1	bit#3, disp16 [reg1]	4	3注5	3注5	3注5
	NOT1	reg2, [reg1]	4	3注5	3注5	3注5
	SET1	bit#3, disp16 [reg1]	4	3注5	3注5	3注5
	SET1	reg2, [reg1]	4	3注5	3注5	3注5
	TST1	bit#3, disp16 [reg1]	4	3注5	3注5	3注5
	TST1	reg2, [reg1]	4	3注5	3注5	3注5
特殊命令	CALLT	imm6	2	4注4	4注4	4注4
	CTRET	－	4	3注4	3注4	3注4
	DI	－	4	1	1	1
	DISPOSE	imm5, list12	4	n+1注6	n+1注6	n+1注6
	DISPOSE	imm5, list12, [reg1]	4	n+3注6	n+3注6	n+3注6
	EI	－	4	1	1	1
	HALT	－	4	1	1	1
	LDSR	reg2, regID	4	1	1	1
	NOP	－	2	1	1	1
	PREPARE	list12, imm5	4	n+1注6	n+1注6	n+1注6
	PREPARE	list12, imm5, sp	4	n+2注6	n+2注6	n+2注6
	PREPARE	list12, imm5, imm16	6	n+2注6	n+2注6	n+2注6
	PREPARE	list12, imm5, imm32	8	n+3注6	n+3注6	n+3注6
	RETI	－	4	3注4	3注4	3注4
	STSR	regID, reg2	4	1	1	1
	SWITCH	reg1	2	5	5	5
	TRAP	vector	4	3注4	3注4	3注4
ディバグ機能用命令	DBRET	－	4	3注4	3注4	3注4
	DBTRAP	－	2	3注4	3注4	3注4
未定義命令コード			4	3	3	3

注1. **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**2**）。**

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**1**）。**

3. **直前に**PSW**の内容を書き換える命令がある場合は**3**。**

4. **タイプ**B**の製品の場合は**＋1**クロック。**

5. **ウエイト・ステートがない場合（**3＋**リード・アクセス・ウエイト・ステート数）。**

6. n**は，**list12**のロード・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**のレジスタ数。**n = 0**の場合，**n = 1**と同じ動作）。**

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）**
list ×	**×個のレジスタ・リスト**

2. **実行クロックの凡例**

略　号	意　味
i	**命令実行直後に他の命令を実行する場合（issue）**
r	**命令実行直後に同一命令を繰り返す場合（repeat）**
l	**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（latency）**

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（EIPC，EIPSWまたはFEPC，FEPSW）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）[注1]	NMI0入力	割り込み	0010H	00000010H	next PC[注2]
	NMI1入力	割り込み	0020H	00000020H	next PC[注2,3]
	NMI2入力[注4]	割り込み	0030H	00000030H	next PC[注2,3]
マスカブル割り込み	注5	割り込み	注5	注6	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7]
ディバグ・トラップ	DBTRAP命令	例外	0060H	00000060H	next PC

注1. 製品により，搭載しているノンマスカブル割り込みの発生要因は異なります。

2. 次の命令の実行中に割り込みを受け付けた場合を除きます（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）。
   - ロード命令（SLD.B, SLD.BU, SLD.H, SLD.HU, SLD.W），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）
3. 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。
4. PSWのNPフラグがセット（1）されていても受け付けられます。
5. 割り込みの種類ごとに異なります。
6. 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。
7. 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またはFEPC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に退避します。

<2> 現在のPSWをEIPSW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のIDフラグをセット（1）し，EPフラグをクリア（0）します。

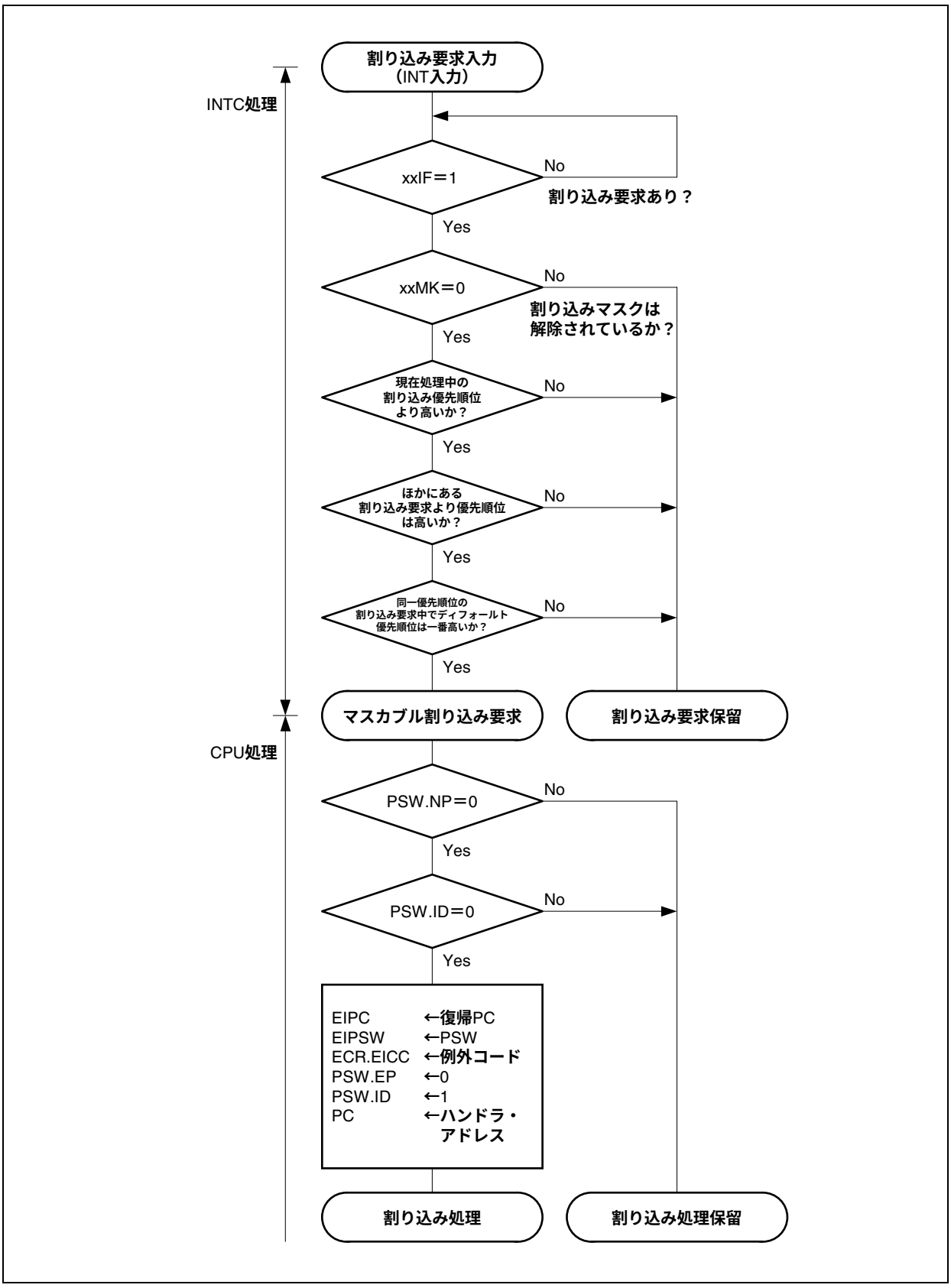
<5> PCに各割り込みに対するハンドラ・アドレスをセットし，制御を移します。

状態退避レジスタにはEIPC，EIPSWを使用します。なお，割り込みコントローラにおいてマスクされているINT入力や割り込み処理中（PSWのNPフラグが1，またはPSW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のNPフラグとPSWの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へ退避します。
<2> 現在のPSWをFEPSW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> PC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。

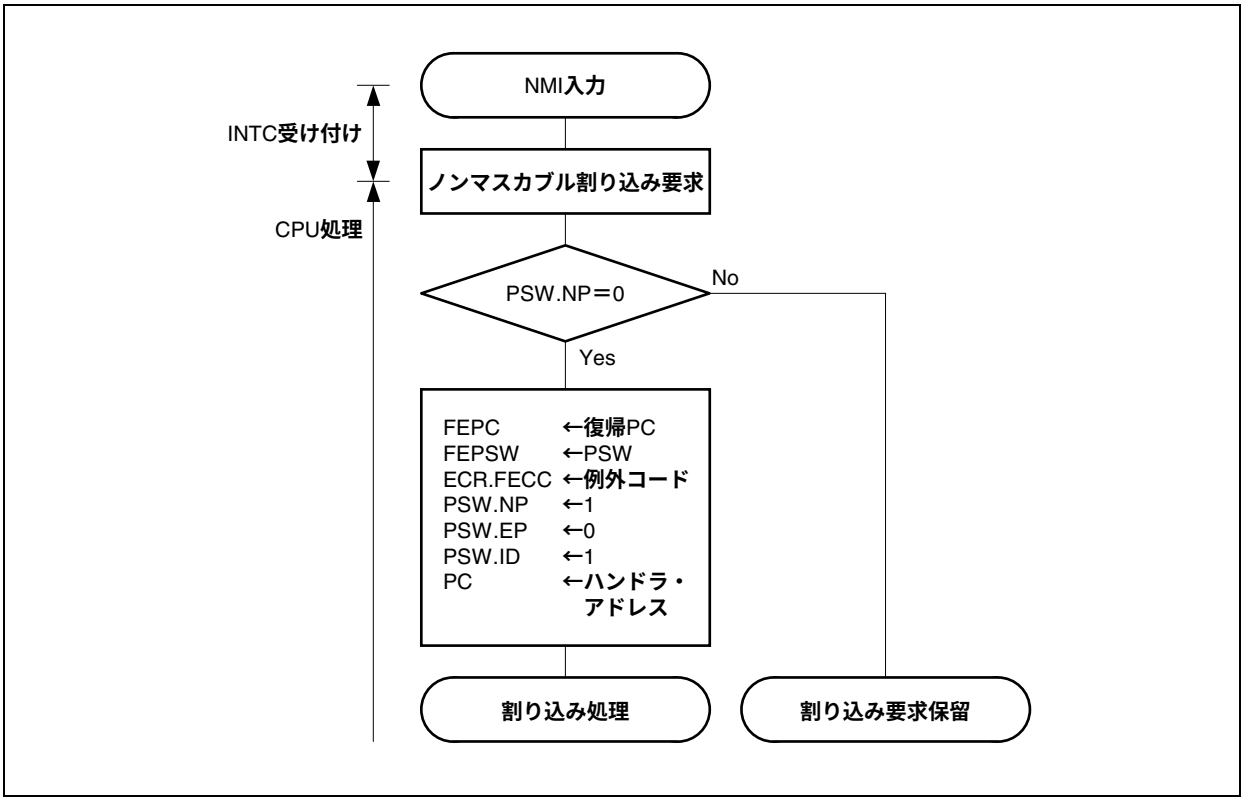
状態退避レジスタには，FEPC, FEPSWを使用します。

また，ノンマスカブル割り込み処理中（PSW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。

NMI2に対する割り込み発生要因を搭載している製品の場合，NMI0, NMI1の割り込み処理中にNMI2が発生した場合だけ，NPフラグの値によらず，NMI2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に退避します。

<2> 現在のPSWをEIPSWへ退避します。

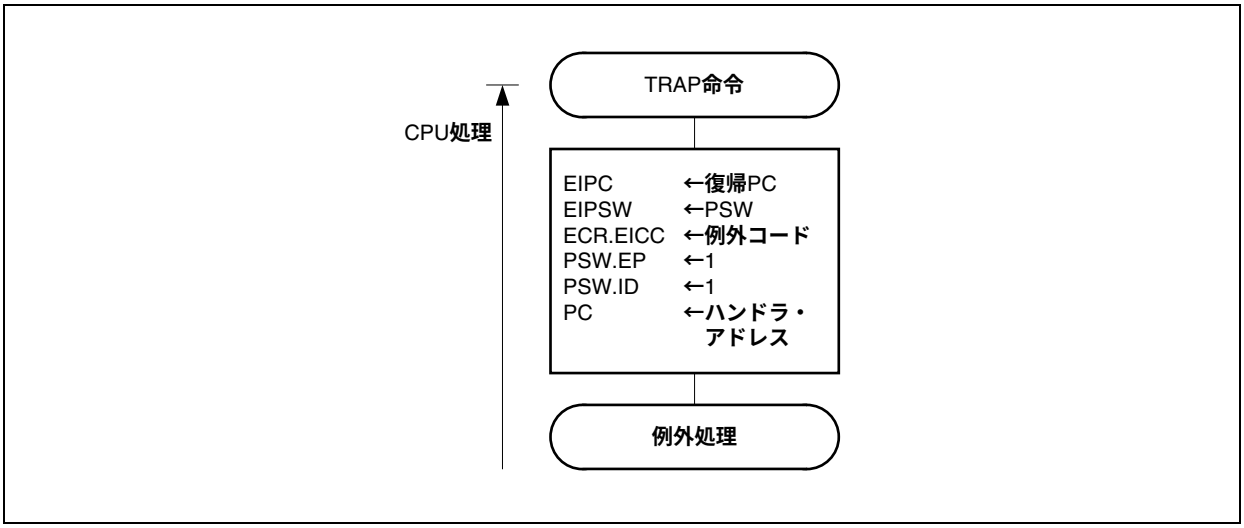
<3> ECR（割り込み要因）の下位16ビット（EICC）に例外コードを書き込みます。

<4> PSWのEP, IDフラグをセット（1）します。

<5> PCにソフトウエア例外に対するハンドラ・アドレス（00000040H または 00000050H）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

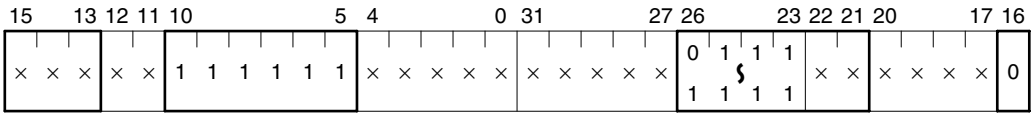


6. 2. 2　例外トラップ

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，命令コードのオペコード（ビット10-5）が111111Bで，サブオペコード（ビット26-23）が0111B-1111B，サブオペコード（ビット16）が0B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



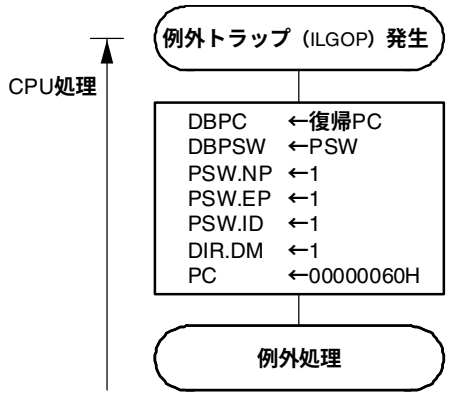
備考　×：任意，□：オペコード／サブオペコード部

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・ルーチンへ制御を移します。

<1> 復帰PCをDBPCに退避します。
<2> 現在のPSWをDBPSWへ退避します。
<3> PSWのNP, EP, IDフラグをセット（1）します。
★ <4> DIRのDMフラグをセット（1）します。
<5> PC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，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・トラップは，DBTRAP命令の実行，またはディバグ機能のトラップにより発生する常時受け付けが可能な例外です。

ディバグ・トラップが発生した場合，CPUは次の処理を行います。

<1> 復帰PCをDBPCに退避します。

<2> 現在のPSWをDBPSWへ退避します。

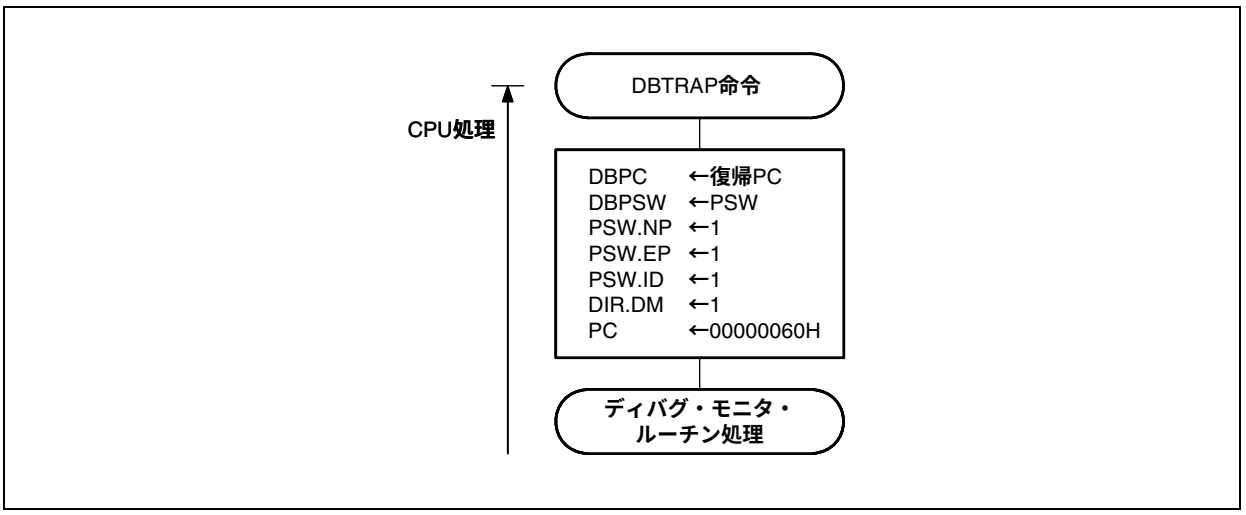
<3> PSWのNP, EP, IDフラグをセット（1）します。

<4> DIRのDMフラグをセット（1）します。

<5> PCにディバグ・トラップに対するハンドラ・アドレス（00000060H）をセットし，制御を移します。

ディバグ・トラップの処理形態を次に示します。

図6－6　ディバグ・トラップの処理形態



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のEPフラグが0かつPSWのNPフラグが1の場合，FEPC, FEPSWから復帰PC, PSWを取り出します。それ以外の場合，EIPC, EIPSWから復帰PC, PSWを取り出します。

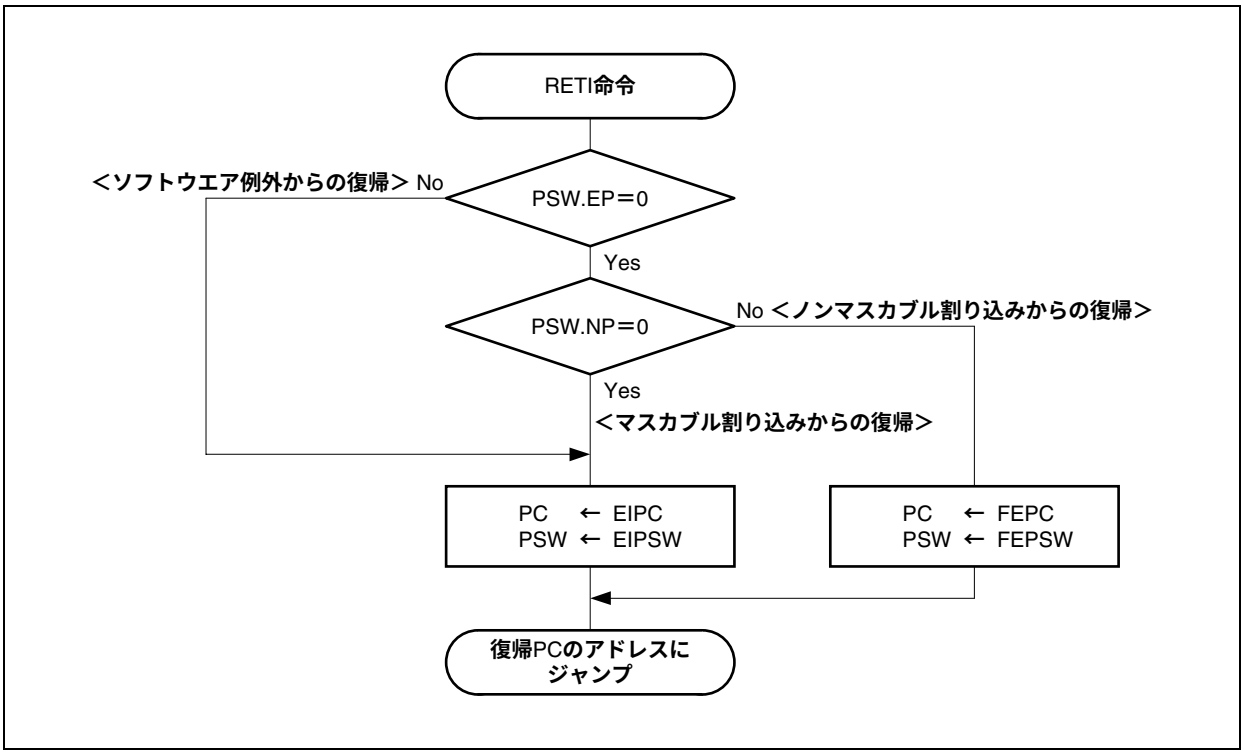
<2> 取り出した復帰PC, PSWのアドレスに制御を移します。

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　：PSW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のNPフラグ = 0，EPフラグ = 0
- 例外処理からの復帰時　：PSWのEPフラグ = 1

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・トラップからの復帰

例外トラップ，ディバグ・トラップからの復帰は，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。

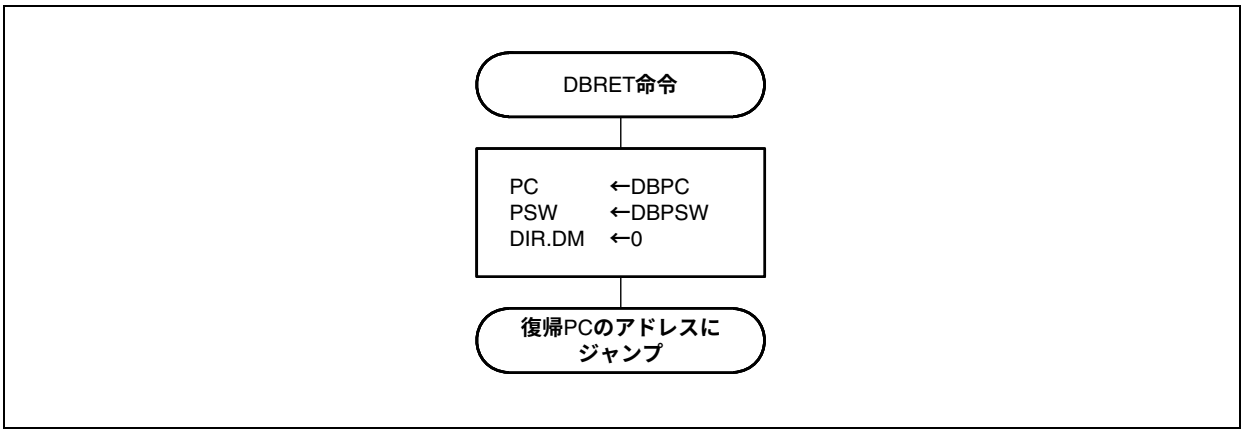
<1> DBPC, DBPSWから復帰PC, PSWを取り出します。

<2> 取り出した復帰PC, PSWのアドレスに制御を移します。

<3> 例外トラップ，ディバグ・トラップから復帰すると，DIRのDMフラグをクリア（0）します。

例外トラップ，ディバグ・トラップ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

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）	0xxxxxxxH
	割り込み時状態退避レジスタ（EIPSW）	00000xxxH
	NMI時状態退避レジスタ（FEPC）	0xxxxxxxH
	NMI時状態退避レジスタ（FEPSW）	00000xxxH
	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）	0xxxxxxxH
	CALLT実行時状態退避レジスタ（CTPSW）	00000xxxH
	例外／ディバグ・トラップ時状態退避レジスタ（DBPC）	0xxxxxxxH
	例外／ディバグ・トラップ時状態退避レジスタ（DBPSW）	00000xxxH
	CALLTベース・ポインタ（CTBP）	0xxxxxxxH
	ディバグ・インタフェース・レジスタ（DIR）	00000000H

備考　x：不定

7.2　起　　動

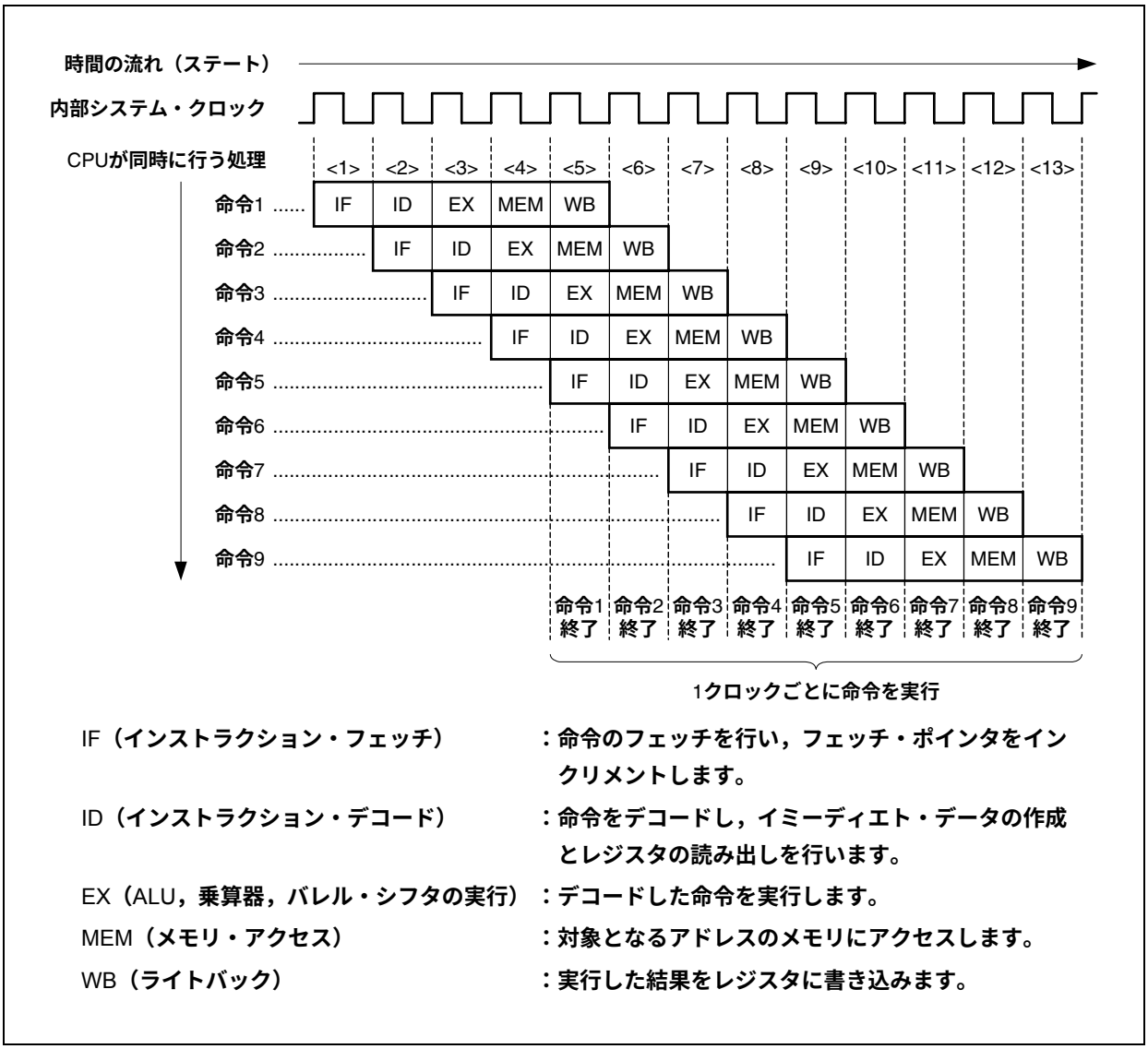
CPUは，リセットにより00000000H番地からプログラムの実行を開始します。

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

V850ES CPUは，RISCアーキテクチャをベースとし，5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，フェッチ（IF）からライトバック（WB）までの5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9つ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9つ続けて実行する例



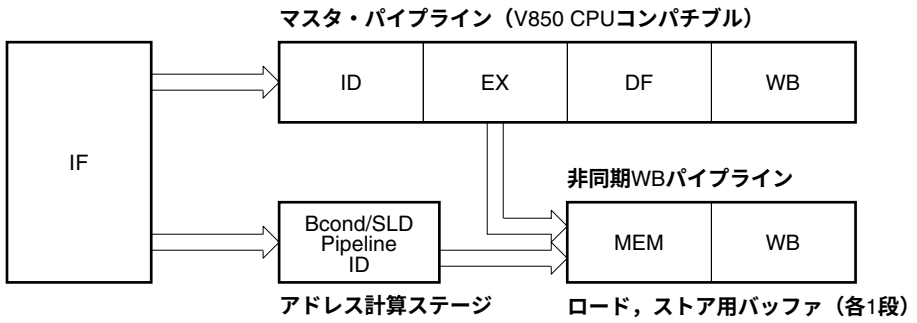
IF（インストラクション・フェッチ）	：命令のフェッチを行い，フェッチ・ポインタをインクリメントします。
ID（インストラクション・デコード）	：命令をデコードし，イミーディエト・データの作成とレジスタの読み出しを行います。
EX（ALU，乗算器，バレル・シフタの実行）	：デコードした命令を実行します。
MEM（メモリ・アクセス）	：対象となるアドレスのメモリにアクセスします。
WB（ライトバック）	：実行した結果をレジスタに書き込みます。

<1> - <13>は，CPUのステートを示します。各ステートでは，命令「n」のライトバック（WB），命令「n+1」のメモリ・アクセス（MEM），命令「n+2」の実行（EX），命令「n+3」のデコード（ID），命令「n+4」のフェッチ（IF）が同時に行われます。標準的な命令では，IFステージからWBステージまで5クロックかかります。しかし，同時に5命令を処理できるため，標準的な命令では平均1クロックごとに実行可能です。

8.1 特　　徴

V850ES CPUは，パイプラインの最適化を行うことにより，CPI（Cycle per instruction）を従来のV850 CPUよりも向上させています。V850ES 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



備考　DF（データ・フェッチ）：実行データをWBステージに転送します。

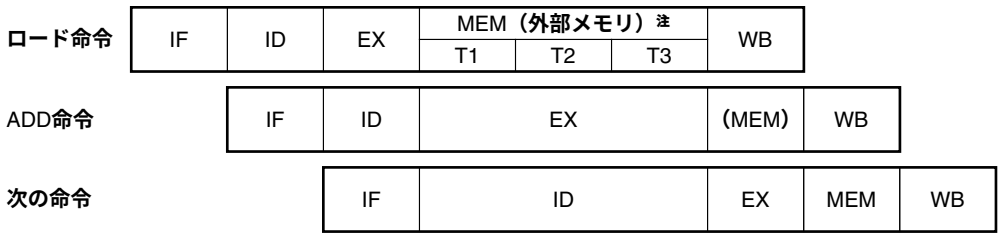
8.1.1　ノンブロッキング・ロード／ストア

外部メモリ・アクセス時にパイプラインが停止することなく，効率的な処理が可能です。

例として，外部メモリに対するロード命令実行後にADD命令を実行する場合のV850 CPUとV850ES CPUのパイプライン動作の比較を図8－3に示します。

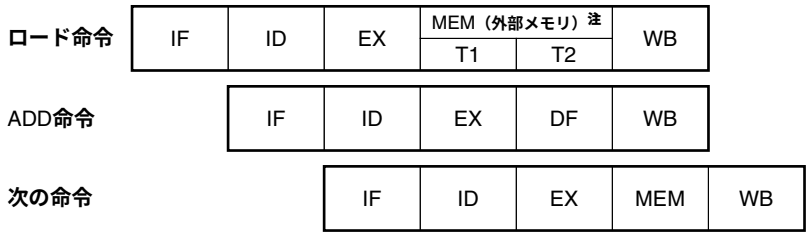
図8－3　ノンブロッキング・ロード／ストア

（a）従来（V850 CPU）：MEMステージ完了までパイプラインが停止



注　外部に対する基本バス・サイクルは3クロックです。

（b）V850ES CPU：非同期WBパイプラインによる効率的なパイプライン処理



注　MEMCの外部に対する基本バス・サイクルは2クロックです。

（1）V850 CPUの場合

ADD命令のEXステージは，本来1クロックで実行されます。しかし，直前のロード命令のMEMステージ実行中，ADD命令のEXステージに待ち時間が発生します。これは，パイプライン上の5つの命令が，同一内部クロック間に同じステージを実行できないためです。この影響により，ADD命令の次の命令のIDステージにも待ち時間が発生します。

（2）V850ES CPUの場合

マスタ・パイプラインのほかに，MEMステージを必要とする命令用に非同期WBパイプラインを持っています。このため，ロード命令のMEMステージは，非同期WBパイプラインで処理されます。図8－3の例ではADD命令はマスタ・パイプラインで処理されるためパイプラインの待ち時間が生じることなく，効率的に命令を実行できます。

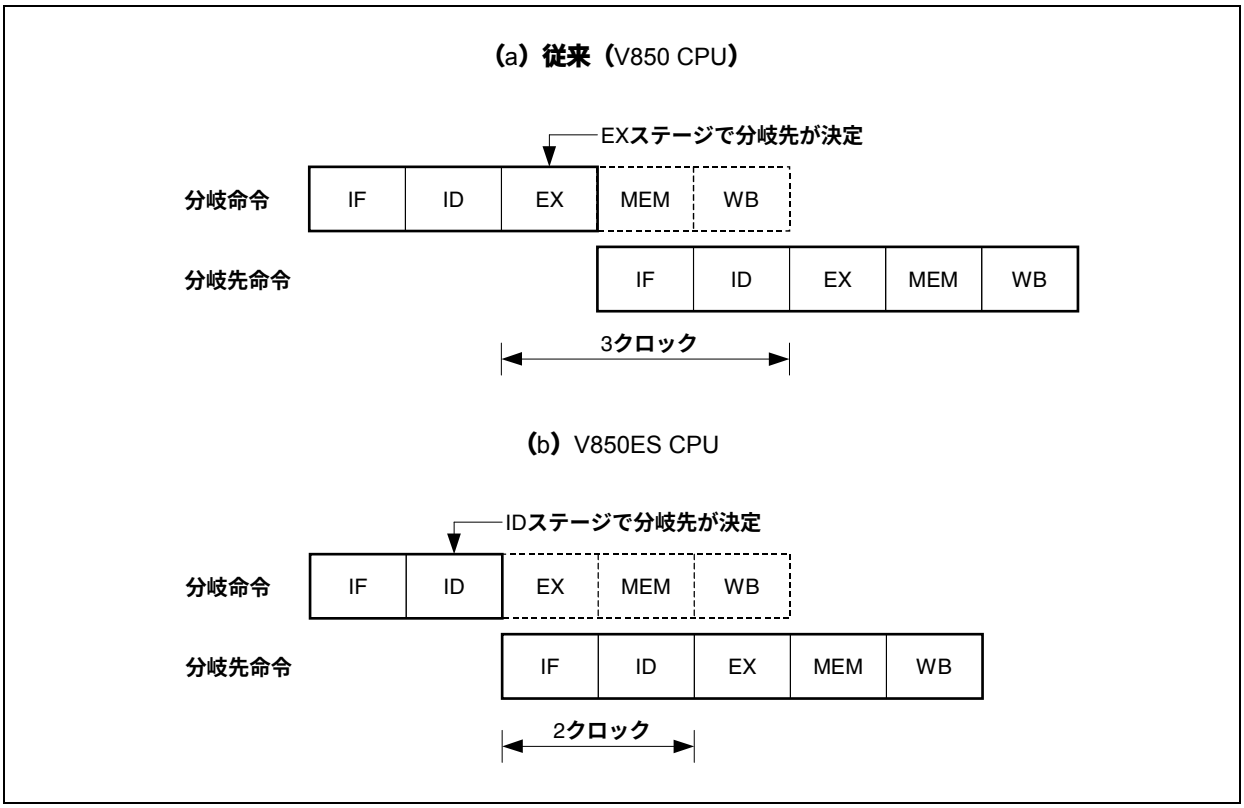
8. 1. 2　2クロック分岐

分岐命令実行時は，IDステージで分岐先が決定します。

従来のV850 CPUの場合，分岐命令実行時の分岐先は，EXステージ実行後に決定されていましたが，V850ES CPUでは，分岐／SLD命令用に追加したアドレス計算ステージにより，IDステージで分岐先が決定します。このため，従来のV850 CPUより，1クロック早く分岐先の命令をフェッチすることができます。

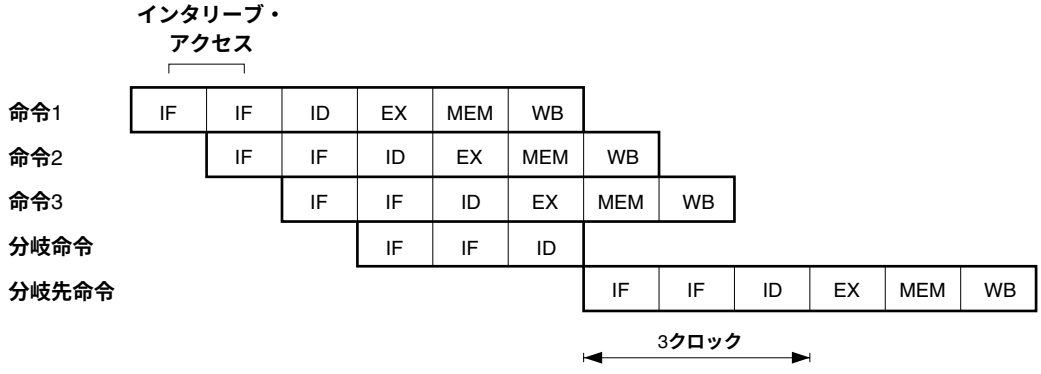
V850 CPUとV850ES CPUの分岐命令でのパイプライン動作の比較を図8－4に示します。

図8－4　分岐命令でのパイプライン動作



備考　タイプ B の製品は，内蔵フラッシュ・メモリまたは内蔵マスク ROM に対しインタリーブ・アクセスを行います。このため割り込み発生直後の命令フェッチや，分岐命令実行後の命令フェッチに 2 クロックかかります。したがって，分岐先命令の ID ステージ実行までに 3 クロックかかります。

例

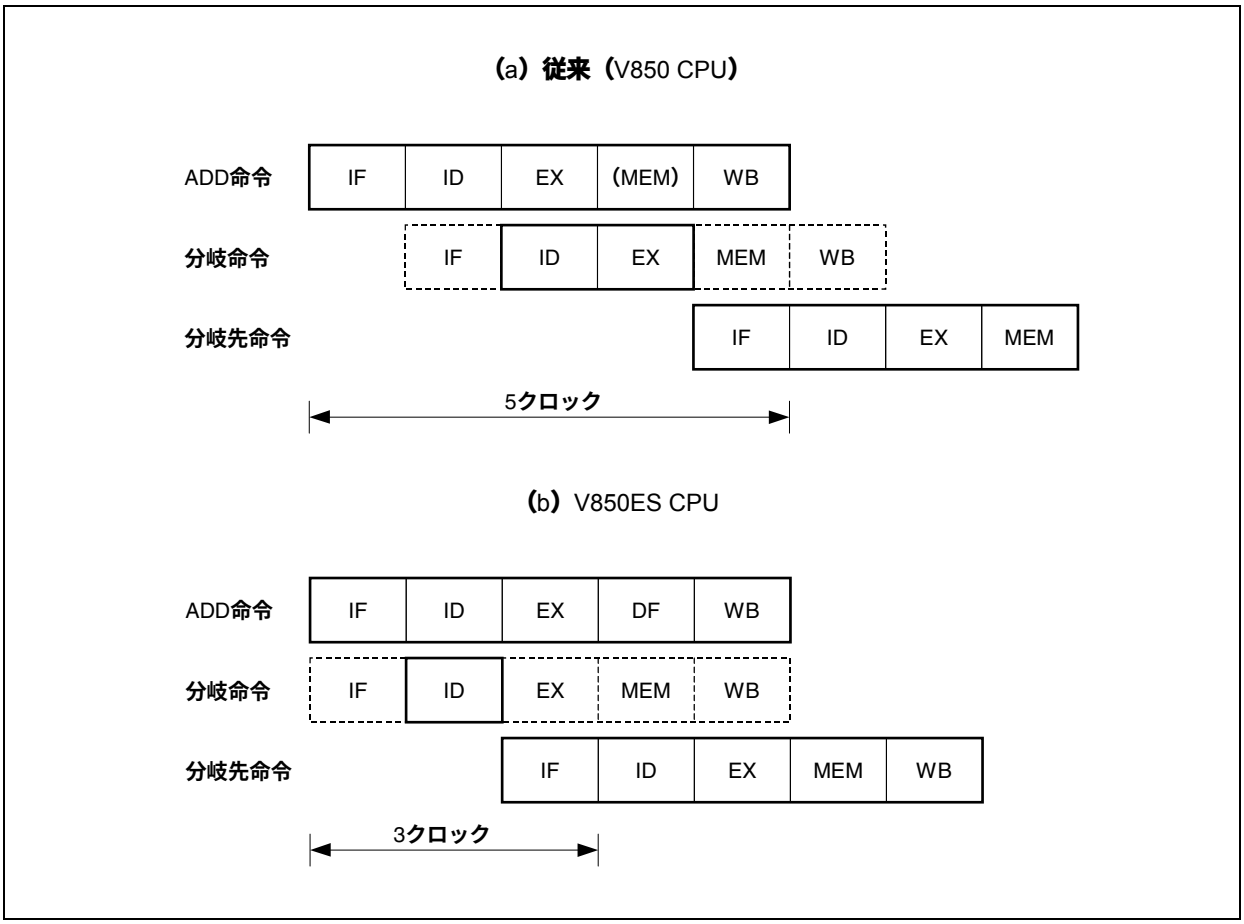


8.1.3　効率的なパイプライン処理

V850ES CPUは，マスタ・パイプライン上のIDステージのほかに，分岐／SLD命令用のIDステージを持っているため，効率的なパイプライン処理が行えます。

例として，ADD命令のIFステージで，次の分岐命令をフェッチした場合のパイプライン動作例を図8－5に示します（専用バスに直結されたROMに対する命令フェッチは，32ビット単位で行われます。図8－5でのADD命令，分岐命令はどちらも16ビット・フォーマットの命令です）。

図8－5　分岐命令の並列実行



（1）V850 CPUの場合

ADD命令のIFステージで次の分岐命令の命令コードまでフェッチしますが，ADD命令のIDステージと分岐命令のIDステージを同一クロック中に実行できません。そのため，分岐命令のフェッチから分岐先命令のフェッチまで，5クロックかかります。

（2）V850ES CPUの場合

マスタ・パイプライン上のIDステージのほかに，分岐／SLD命令用のIDステージを持っているため，同一クロック中に並行してADD命令のIDステージと分岐命令のIDステージを実行できます。このため，分岐命令のフェッチ開始から分岐先の命令フェッチ完了まで3クロックで実行できます。

注意　SLD命令とBcond命令については，ほかの16ビット・フォーマットの命令と同時実行される場合があるため注意が必要です。たとえば，SLD命令とNOP命令が同時に実行された場合，NOP命令によるディレイ・タイムの発生が行われない可能性があります。

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。

パイプライン処理のため，メモリやI/Oのライト・サイクルが発生する時点で，CPUは後続の命令をすでに実行しています。そのため，I/O操作や割り込み要求マスクの反映は，次の命令発行（IDステージ）に対して遅れて実行されます。

内蔵INTCへのアクセスをCPUが検出（IDステージ）して割り込み要求のマスク処理を行うため，割り込みマスク操作を行う場合，直後の命令からマスカブル割り込みの受け付けを禁止します。

8. 2. 1　ロード命令

注意　ノンブロッキング制御のため，MEMステージの間にバス・サイクルが完了している保証はありません。ただし，周辺I/O領域へのアクセスはブロッキング制御となるため，MEMステージでバス・サイクルの完了を待ち合わせます。

（1）LD命令

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

［パイプライン］

	<1>	<2>	<3>	<4>	<5>	<6>
LD命令	IF	ID	EX	MEM	WB	
次命令		IF	ID	EX	MEM	WB

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

（2）SLD命令

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

［パイプライン］

	<1>	<2>	<3>	<4>	<5>	<6>
SLD命令	IF	ID	MEM	WB		
次命令		IF	ID	EX	MEM	WB

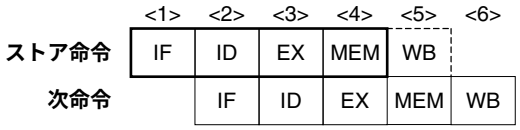
［説　　明］　**パイプラインはIF, ID, MEM, WBの4ステージです。SLD命令の直後に，実行結果を使用する命令を配置すると，データの待ち合わせ時間が発生します。**

8.2.2 ストア命令

注意　ノンブロッキング制御のため，MEMステージの間にバス・サイクルが完了している保証はありません。ただし，周辺I/O領域へのアクセスはブロッキング制御となるため，MEMステージでバス・サイクルの完了を待ち合わせます。

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

[パイプライン]



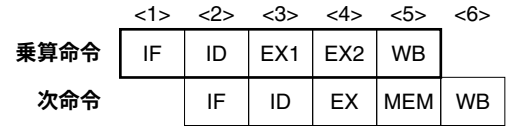
[説　　明]　**パイプラインはIF, ID, EX, MEM, WBの5ステージですが，レジスタへのデータの書き込みがないのでWBステージでは何も行いません。**

8.2.3 乗算命令

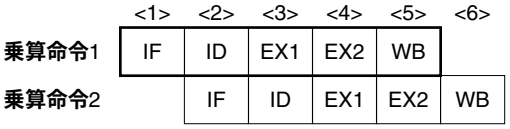
(1) ハーフワード・データ乗算命令

[対象の命令]　MULH, MULHI

[パイプライン]　**(a) 次命令が乗算命令以外の場合**



(b) 次命令が乗算命令の場合



[説　　明]　**パイプラインはIF, ID, EX1, EX2, WBの5ステージです。EXステージは乗算器実行のため2クロックかかりますが，EX1とEX2（通常のEXステージとは異なります）は独立して動作できます。したがって，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。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。**

(2) ワード・データ乗算命令

[対象の命令]　MUL, MULU

[パイプライン]　(a) 続く3命令が乗算命令以外の場合

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

(b) 次命令が乗算命令の場合

	<1>	<2>	<3>	<4>	<5>	<6>	<7>	<8>	<9>
乗算命令1	IF	ID	EX1	EX1	EX1	EX1	EX2	WB	
乗算命令2（ハーフワード）		IF	ー	ー	ー	ID	EX1	EX2	WB

ー：待ちあわせのために挿入されるアイドル

(c) 続く3命令目が乗算命令の場合

	<1>	<2>	<3>	<4>	<5>	<6>	<7>	<8>	<9>
乗算命令1	IF	ID	EX1	EX1	EX1	EX1	EX2	WB	
命令1		IF	ID	EX	MEM	WB			
命令2			IF	ID	EX	MEM	WB		
乗算命令2（ハーフワード）				IF	ー	ID	EX1	EX2	WB

ー：待ちあわせのために挿入されるアイドル

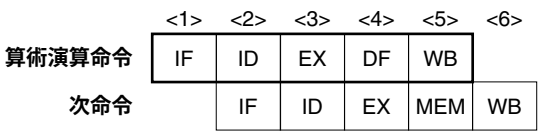
[説　　明]　パイプラインはIF, ID, EX1（4ステージ）, EX2, WBの8ステージです。EXステージは乗算器実行のため5クロックかかりますが，EX1とEX2（通常のEXステージとは異なります）は独立して動作できます。したがって，乗算命令を繰り返しても命令実行クロック数は4クロックとなります。ただし，乗算命令の直後に実行結果を使用する命令を配置すると，データの待ち合わせ時間が発生します。

8.2.4　算術演算命令

(1) 除算／ワード転送命令以外

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

[パイプライン]



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

(2) ワード転送命令

[対象の命令]　MOV　imm32

[パイプライン]

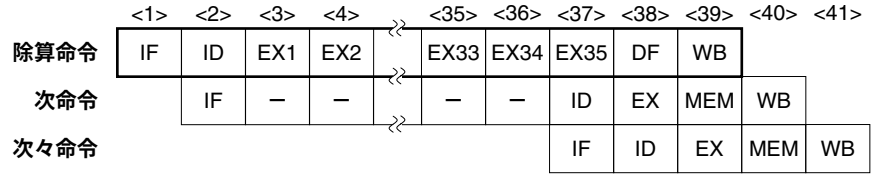
ー：待ちあわせのために挿入されるアイドル

[説　　明]　パイプラインはIF, ID, EX1, EX2（通常のEXステージ）, DF, WBの6ステージです。

(3) 除算命令

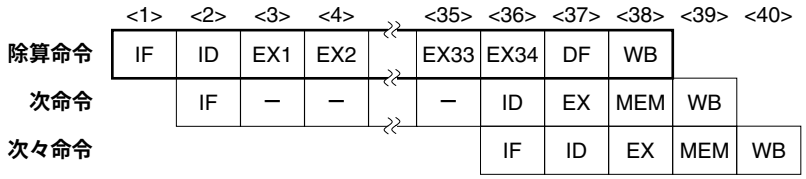
[対象の命令]　DIV, DIVH, DIVHU, DIVU

[パイプライン]　(a) DIV, DIVH命令の場合



ー：待ちあわせのために挿入されるアイドル

(b) DIVHU, DIVU命令の場合



ー：待ちあわせのために挿入されるアイドル

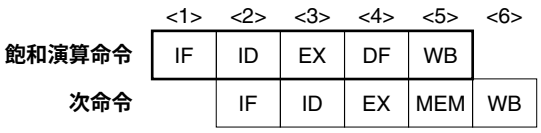
[説　　明]　パイプラインは，DIV, DIVH命令の場合，IF, ID, EX1-EX35（通常のEXステージ）, DF, WBの39ステージ，DIVHU, DIVU命令の場合，IF, ID, EX1-EX34（通常のEXステージ）, DF, WBの38ステージです。

[備　　考]　除算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ます。割り込み処理完了後，この命令を再実行します。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。

8.2.5　飽和演算命令

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

［パイプライン］

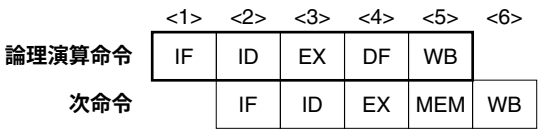


［説　　明］　パイプラインはIF, ID, EX, DF, WBの5ステージです。

8.2.6　論理演算命令

［対象の命令］　AND, ANDI, BSH, BSW, HSW, NOT, OR, ORI, SAR, SHL, SHR, SXB, SXH, TST, XOR, XORI, ZXB, ZXH

［パイプライン］



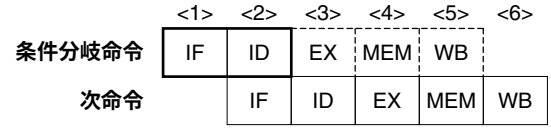
［説　　明］　パイプラインはIF, ID, EX, DF, WBの5ステージです。

8.2.7　分岐命令

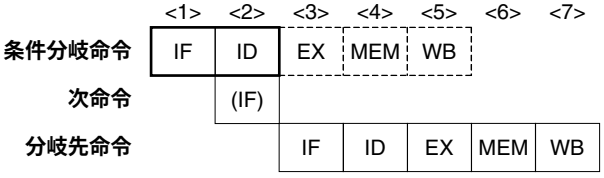
(1) 条件分岐命令（BR命令を除く）

［対象の命令］　Bcond命令（BC, BE, BGE, BGT, BH, BL, BLE, BLT, BN, BNC, BNE, BNH, BNL, BNV, BNZ, BP, BSA, BV, BZ）

［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



(IF)：無効となる命令フェッチ

［説　　明］　パイプラインはIF, ID, EX, MEM, WBの5ステージですが，IDステージで分岐先が決定するため，EXステージ，MEMステージ，WBステージでは何も行いません。

(a) 条件が成立しない場合

分岐命令の命令実行クロック数は1となります。

(b) 条件が成立した場合

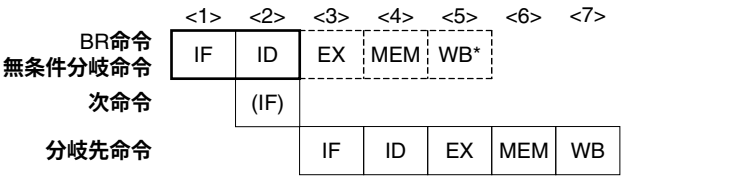
分岐命令の命令実行クロック数は2となります。分岐命令の次命令のIFは無効となります。

ただし，直前にPSWの内容を書き換える命令があると，フラグ・ハザード発生のために命令実行クロック数は3となります。

(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

[パイプライン]

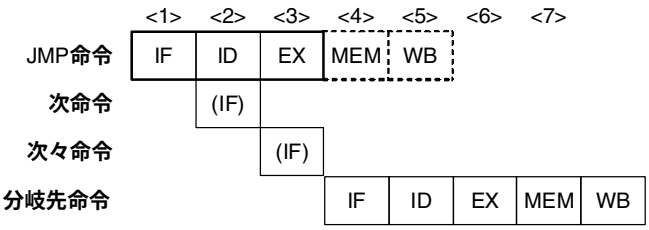


(IF)　：無効となる命令フェッチ
WB*：JR, BR命令の場合は，何も行われませんが，
JARL命令の場合は復帰PCの書き込みが行われます。

[説　　明]　パイプラインはIF, ID, EX, MEM, WBの5ステージですが，IDステージで分岐先が決定するためEXステージ，MEMステージ，WBステージでは何も行いません。ただし，JARL命令の場合にはWBステージにおいて復帰PCの書き込みが行われます。また，分岐命令の次命令のIFは無効となります。

★　(3) JMP命令

[パイプライン]



(IF)：無効となる命令フェッチ

[説　　明]　パイプラインは，IF, ID, EX, MEM, WBの5ステージですが，EXステージで分岐先が決定するため，MEMステージ，WBステージでは何も行いません。

8. 2. 8　ビット操作命令

（1）CLR1, NOT1, SET1**命令**

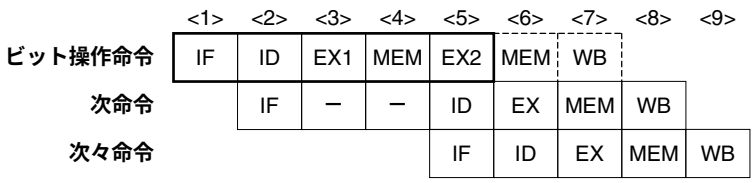
［パイプライン］

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**IF, ID, EX1, MEM, EX2**（通常のステージ）**, MEM, WB**の7ステージですが，レジスタへのデータ書き込みがないので，**WB**ステージでは何も行いません。この命令では，メモリ・アクセスがリード・モディファイ・ライトとなり，**EX**ステージに計2クロック，MEM**ステージに計2サイクルかかります。**

（2）TST1**命令**

［パイプライン］



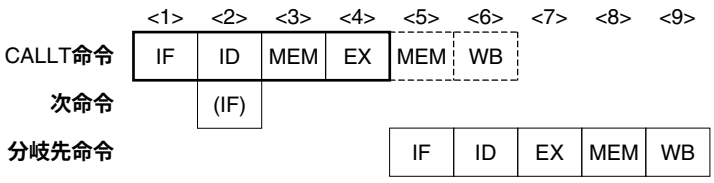
ー：待ちあわせのために挿入されるアイドル

［説　　明］　**パイプラインは**IF, ID, EX1, MEM, EX2**（通常のステージ）**, MEM, WB**の7ステージですが，2回目のメモリ・アクセス，レジスタへのデータ書き込みがないので，2回目の**MEM**ステージ，**WB**ステージでは何も行いません。**
この命令では，計2クロックかかります。

8. 2. 9　特殊命令

（1）CALLT**命令**

［パイプライン］

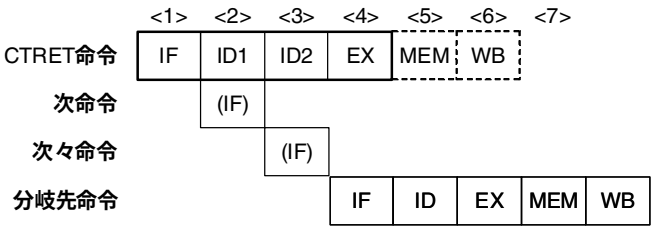


(IF)**：無効となる命令フェッチ**

［説　　明］　**パイプラインは，**IF, ID, MEM, EX, MEM, WB**の6ステージです。ただし，2番目の**MEM**とWBのステージでは，メモリ・アクセスがなく，レジスタにデータ書き込みがないため，何も行いません。**

★　(2) CTRET命令

[パイプライン]

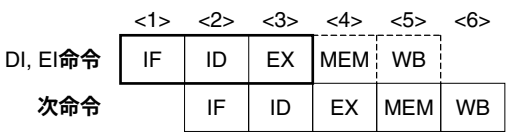


(IF)：無効となる命令フェッチ
ID1 :レジスタ選択
ID2：CTPC読み込み

[説　　明]　パイプラインはIF, ID, EX, MEM, WBの5ステージですが，メモリへのアクセス，レジスタへのデータ書き込みがないので，MEMステージ，WBステージでは何も行いません。また，次命令のIFと次々命令のIFは無効になります。

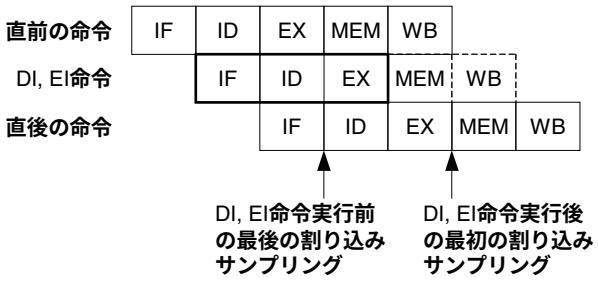
(3) DI, EI命令

[パイプライン]



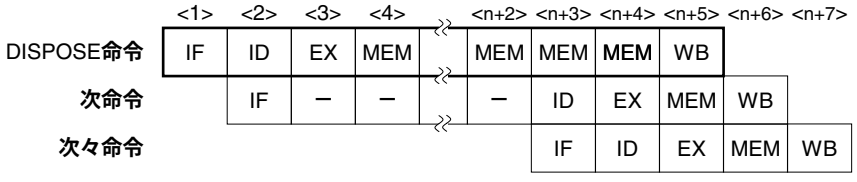
[説　　明]　パイプラインはIF, ID, EX, MEM, WBの5ステージですが，メモリへのアクセス，レジスタへのデータ書き込みがないので，MEMステージ，WBステージでは何も行いません。

[備　　考]　DI, EI命令は，いずれも割り込み要求非サンプル命令です。これらの命令実行時における割り込みサンプリング・タイミングは，次のようになります。



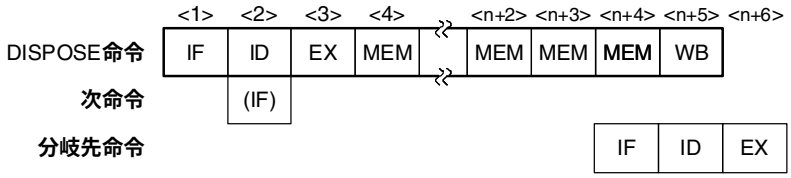
(4) DISPOSE命令

[パイプライン]　(a) 分岐しない場合



－：待ちあわせのために挿入されるアイドル
n：レジスタ・リスト（list12）で指定されるレジスタの数

★ (b) 分岐する場合

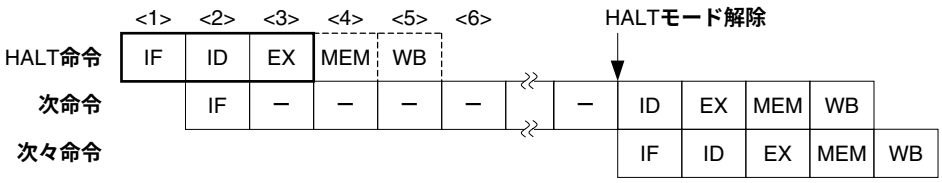


(IF)：無効となる命令フェッチ
n　：レジスタ・リスト（list12）で指定されるレジスタの数

[説　　明]　パイプラインは，IF, ID, EX, n+1回のMEM, WBの「n+5」ステージです（n：レジスタ・リスト・ナンバ）。MEMステージは，n+1サイクル必要です。

(5) HALT命令

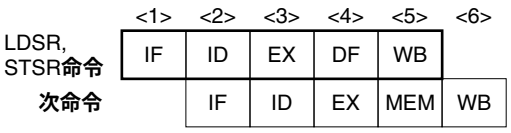
[パイプライン]



[説　　明]　パイプラインはIF, ID, EX, MEM, WBの5ステージですが，メモリへのアクセス，レジスタへのデータ書き込みがないので，MEMステージ，WBステージでは何も行いません。また，次命令では，HALTモードが解除されるまでIDステージが遅れます。

(6) LDSR, STSR命令

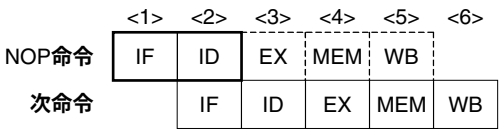
[パイプライン]



[説　　明]　パイプラインはIF, ID, EX, DF, WBの5ステージです。システム・レジスタのEIPC, FEPCを設定するLDSR命令の直後に，同一レジスタを使用するSTSR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。

(7) NOP命令

[パイプライン]

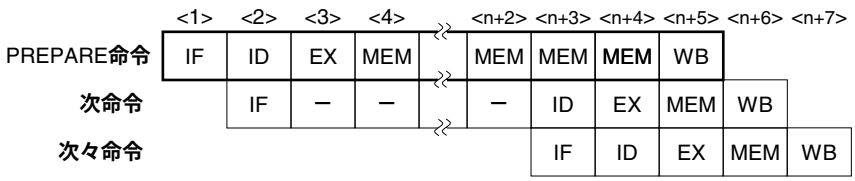


[説　　明]　パイプラインはIF, ID, EX, MEM, WBの5ステージですが，演算，メモリへのアクセス，レジスタへのデータ書き込みがないので，EXステージ，MEMステージ，WBステージでは何も行いません。

[注　　意]　SLD命令とBcond命令については，ほかの16ビット・フォーマットの命令と同時実行される場合があるため注意が必要です。たとえば，SLD命令とNOP命令が同時に実行された場合，NOP命令によるディレイ・タイムの発生が行われない可能性があります。

(8) PREPARE命令

［パイプライン］

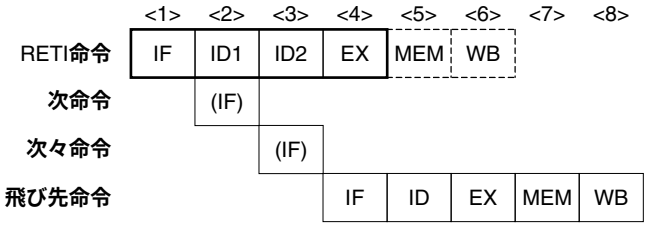


ー：待ちあわせのために挿入されるアイドル
n：レジスタ・リスト（list12）で指定されるレジスタの数

［説　　明］　パイプラインは，IF, ID, EX, n+1回のMEM, WBの「n+5」ステージです（n：レジスタ・リスト・ナンバ）。MEMステージは，n+1サイクル必要です。

(9) RETI命令

［パイプライン］

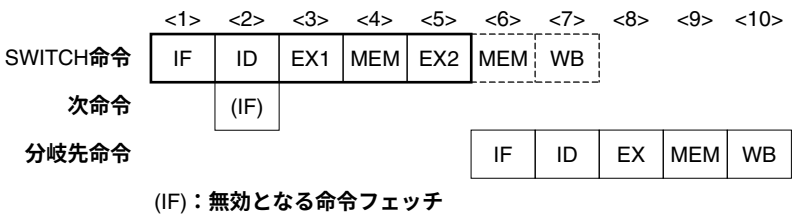


(IF)：無効となる命令フェッチ
ID1：レジスタ選択
ID2：EIPC/FEPC読み込み

［説　　明］　パイプラインはIF, ID1, ID2, EX, MEM, WBの6ステージですが，メモリへのアクセス，レジスタへのデータ書き込みがないので，MEMステージ，WBステージでは何も行いません。IDステージには2クロックかかります。また，次命令のIFと次々命令のIFは無効となります。

(10) SWITCH命令

［パイプライン］

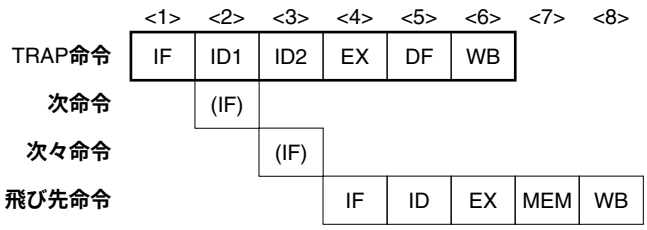


(IF)：無効となる命令フェッチ

［説　　明］　パイプラインは，IF, ID, EX1（通常のEXステージ）, MEM, EX2, MEM, WBの7ステージです。ただし，2番目のMEMとWBのステージでは，メモリ・アクセスがなく，レジスタにデータ書き込みがないため，何も行いません。

（11）TRAP命令

［パイプライン］



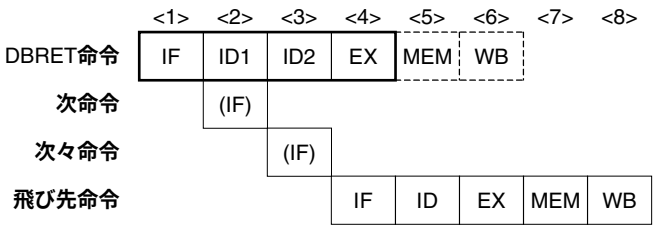
(IF)：無効となる命令フェッチ
ID1：例外コード（004nH, 005nH）検出（n = 0-FH）
ID2：アドレス生成

［説　　明］　パイプラインはIF, ID1, ID2, EX, DF, WBの6ステージです。IDステージには2クロックかかります。また，次命令のIFと次々命令のIFは無効となります。

8. 2. 10　ディバグ機能用命令

（1）DBRET命令

［パイプライン］

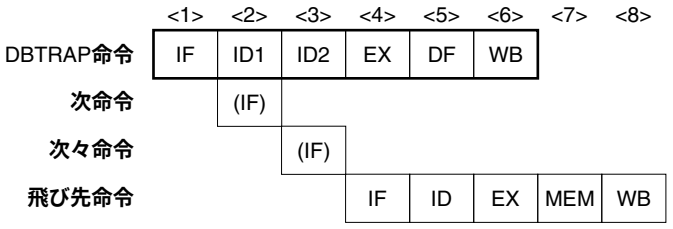


(IF)：無効となる命令フェッチ
ID1：レジスタ選択
ID2：DBPC読み込み

［説　　明］　パイプラインはIF, ID1, ID2, EX, MEM, WBの6ステージですが，メモリへのアクセス，レジスタへのデータ書き込みがないので，MEMステージ，WBステージでは何も行いません。IDステージには2クロックかかります。また，次命令のIFと次々命令のIFは無効となります。

（2）DBTRAP命令

［パイプライン］



(IF)：無効となる命令フェッチ
ID1：例外コード（0060H）検出
ID2：アドレス生成

［説　　明］　パイプラインはIF, ID1, ID2, EX, DF, WBの6ステージです。IDステージには2クロックかかります。また，次命令のIFと次々命令のIFは無効となります。

8. 3　パイプラインの乱れ

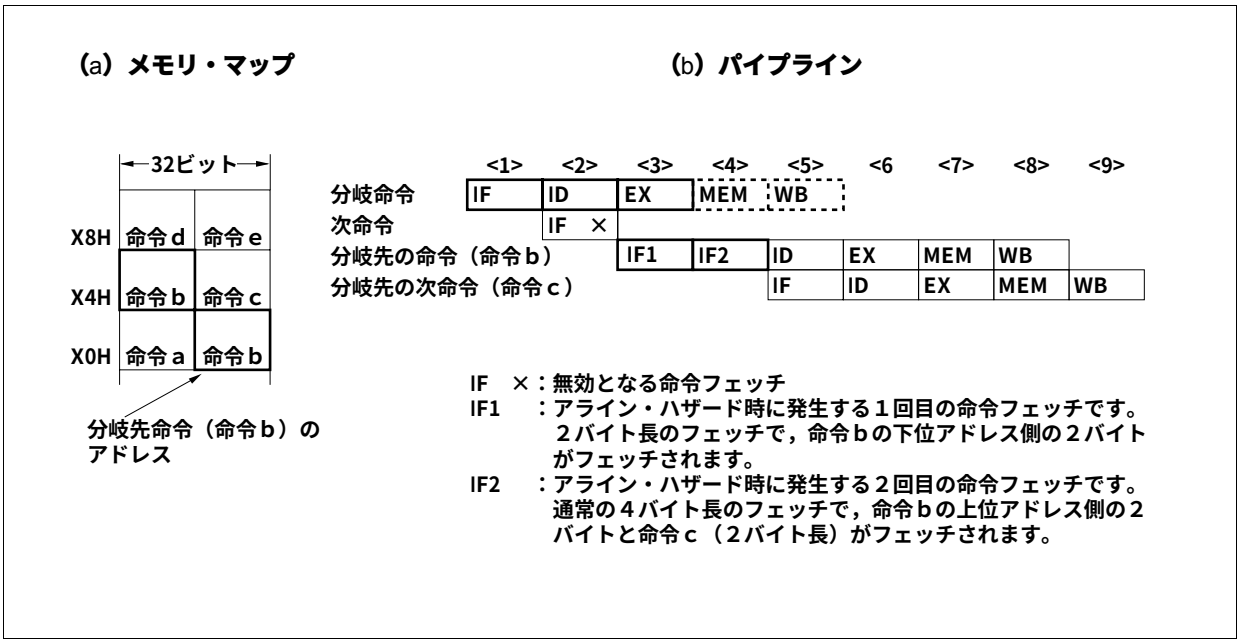
パイプラインはIF（インストラクション・フェッチ）からWB（ライトバック）までの5ステージで構成され，基本的にはそれぞれのステージは1クロックで処理されますが，場合によってはパイプラインが乱れて，実行クロック数が増加する場合があります。この節ではパイプラインを乱す主な要因を示します。

8. 3. 1　アライン・ハザード

分岐先命令のアドレスがワード・アラインでなく（A1 = 1, A0 = 0），かつ4バイト長命令の場合，命令をワード単位にそろえるためにIFを2回続ける必要があります。これをアライン・ハザードと呼びます。

たとえば，命令aから命令eまでがアドレスX0Hから配置されており，命令bは4バイト長命令で，その他の命令は2バイト長命令であるとします。この場合，命令bはX2Hに配置され（A1 = A0 = 0），ワード・アライン（A1 = 0, A0 = 0）となっていません。したがって，この命令bが分岐先命令となる場合，アライン・ハザードが発生します。アライン・ハザードが発生した場合の分岐命令の実行クロック数は4となります。

図8－6　アライン・ハザードの例



アライン・ハザードは，次のような対処によって回避が可能で，命令実行速度の向上が図れます。

- 分岐先命令に2バイト長命令を使用する
- 分岐先命令に，ワード境界（A1 = 0, A0 = 0）に配置した4バイト長命令を使用する

8. 3. 2　ロード命令実行結果の参照

ロード命令（LD, SLD）では，MEMステージで読み出されたデータの格納がWBステージで行われます。したがって，ロード命令の直後の命令で同一のレジスタの内容を使用する場合，ロード命令がレジスタの使用を終えるまで，直後の命令はレジスタの使用を遅らせる必要があります。これをハザードと呼びます。

V850ES CPUは，このハザードを自動的に対処するインタロック機能を持っており，次命令のIDステージを遅らせます。

またV850ES CPUは，MEMステージで読み出したデータを次命令のIDステージで使用できるように，ショート・パスを持っています。このショート・パスによって，ロード命令によってMEMステージでデータを読み出すことと，このデータを次命令のIDステージで使用することを，同一タイミングで行うことができます。

以上のことより，結果を直後の命令で使用する場合，ロード命令の実行クロック数は2になります。

図8－7　ロード命令実行結果の参照例

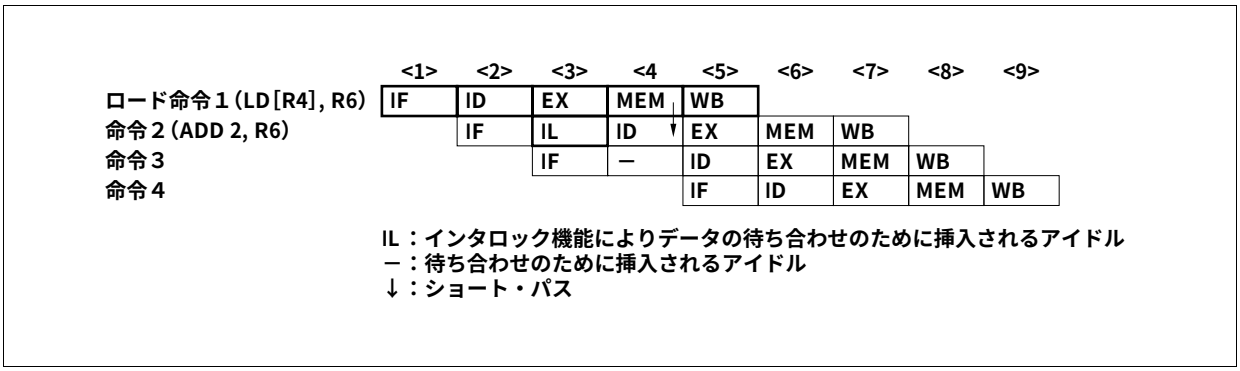


図8－7のように，ロード命令の直後にその結果を使用する命令を配置すると，インタロック機能によるデータの待ち合わせ時間が発生し，実行速度が低下します。ロード命令の結果を使用する命令は，ロード命令の2命令以後に配置することにより，実行速度の低下が防げます。

8. 3. 3　乗算命令実行結果の参照

乗算命令では，演算結果のレジスタへの格納がWBステージで行われます。したがって，乗算命令の直後の命令で同一レジスタの内容を使用する場合，乗算命令がレジスタの使用を終えるまで，直後の命令はレジスタの使用を遅らせる必要があります（ハザードの発生）。

V850ES CPUでは，インタロック機能により直後の命令のIDステージを遅らせます。また，ショート・パスにより，乗算命令のEX2ステージと，この演算結果を直後の命令のIDステージで使用することが，同一タイミングで行えます。

図8－8　乗算命令実行結果の参照例

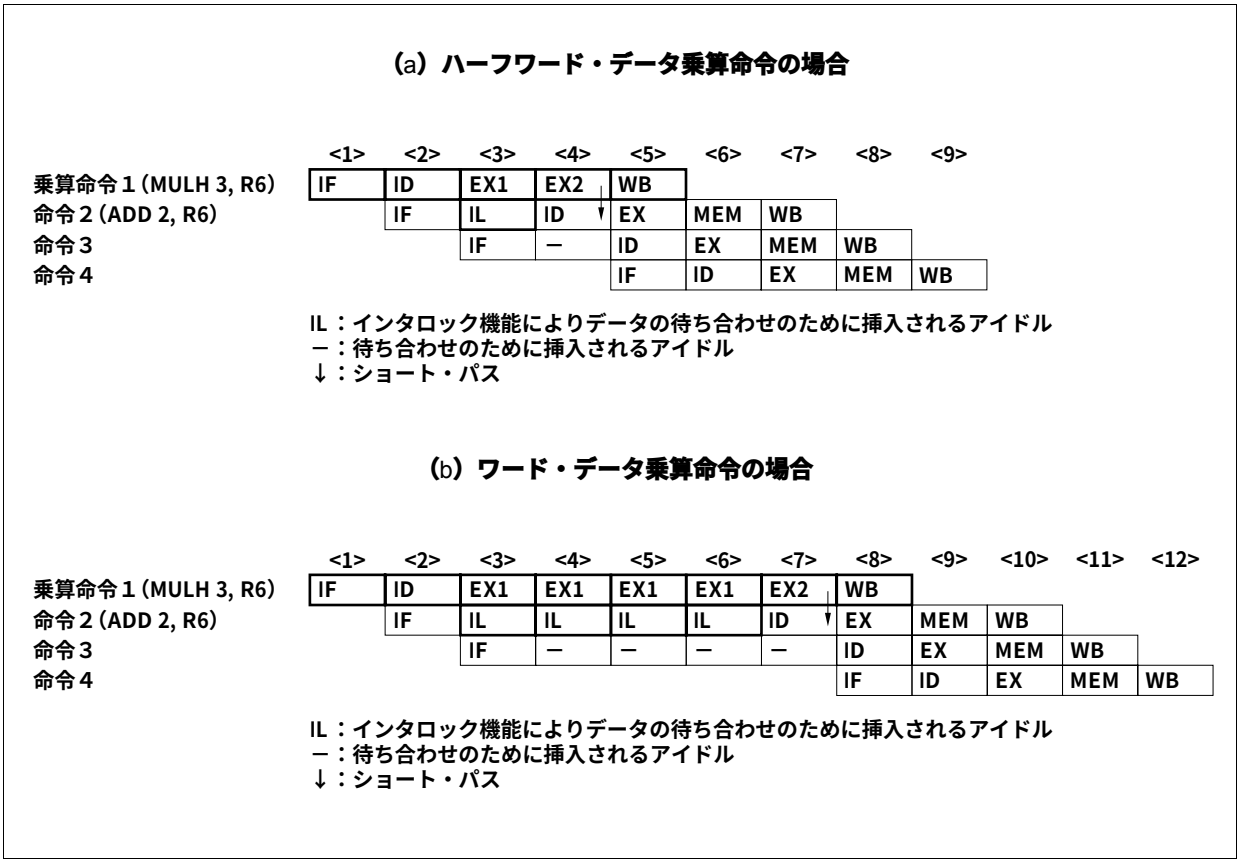


図8－8のように，乗算命令の直後にその結果を使用する命令を配置すると，インタロック機能によるデータの待ち合わせ時間が発生し，実行速度が低下します。乗算命令の結果を使用する命令は，乗算命令の2命令以後に配置することにより，実行速度の低下を防げます。ただし，ワード・データ乗算命令（MUL, MULU）の場合，乗算命令の結果を使用する命令は，乗算命令の5命令以後に配置しなければ，1-4のILステージが挿入されます。

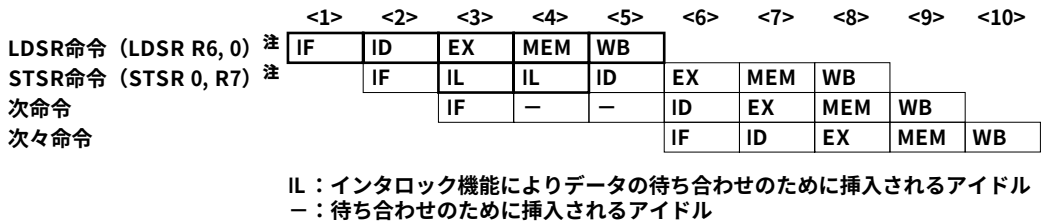
8. 3. 4　EIPC, FEPCを対象とするLDSR命令実行結果の参照

LDSR命令によって，システム・レジスタのEIPC, FEPCのデータ設定を行い，直後にSTSR命令で同一システム・レジスタの参照を行う場合，LDSR命令のシステム・レジスタ設定が終わるまで，直後のSTSR命令はシステム・レジスタの使用が遅れます（ハザードの発生）。

V850ES CPUではインタロック機能により，直後のSTSR命令のIDステージを遅らせます。

以上のことより，EIPC, FEPCのLDSR命令実行結果を直後のSTSR命令で使用する場合，LDSR命令の実行クロック数は3になります。

図8－9　EIPC, FEPCを対象とするLDSR命令実行結果の参照例



注　LDSR, STSR命令で使用しているシステム・レジスタ番号の0はEIPCを表します。

図8－9のように，EIPC，またはFEPCをオペランドとするLDSR命令の直後に，STSR命令によってその結果を使用すると，インタロック機能によるデータの待ち合わせ時間が発生し，実行速度が低下します。LDSR命令の結果を参照するSTSR命令は，LDSR命令の3命令以後に配置することにより，実行速度の低下を防げます。

8. 3. 5　プログラム作成時の注意点

プログラムを作成する場合，次のことに注意するとパイプラインが乱れず，命令実行速度が向上します。

- ロード命令（LD, SLD）の結果を使用する命令は，ロード命令の2命令以後に配置する。
- 乗算命令（MULH, MULHI）の結果を使用する命令は，乗算命令の2命令以後に配置する。
- LDSR命令によるEIPC，またはFEPCへの設定結果をSTSR命令により読み出す場合には，LDSR命令の3命令以後にSTSR命令を配置する。
- 分岐先の最初の命令は，2バイト長命令か，またはワード境界に配置された4バイト長命令を使用する。

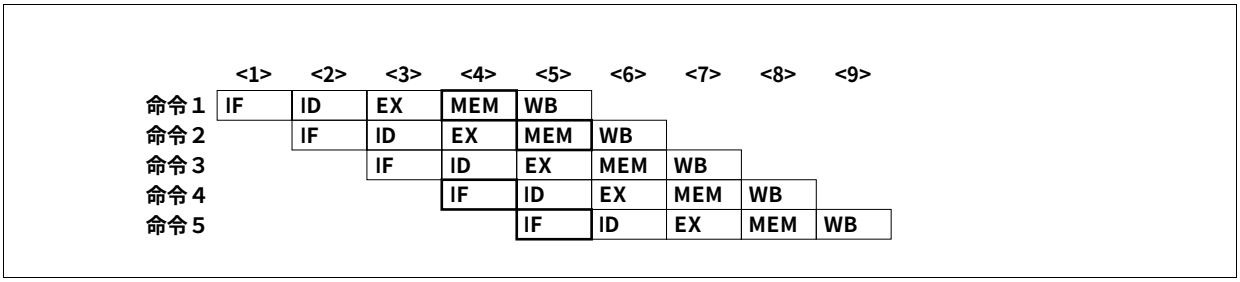
8.4　パイプラインに関する補足事項

8.4.1　ハーバード・アーキテクチャ

V850ES CPUではハーバード・アーキテクチャを採用しており，内蔵ROMからの命令フェッチ用のパスと，内蔵RAMへのメモリ・アクセス用のパスが独立して動作します。これにより，IFステージとMEMステージのバス・アービトレーションの競合が発生せず，パイプラインがスムーズに流れます。

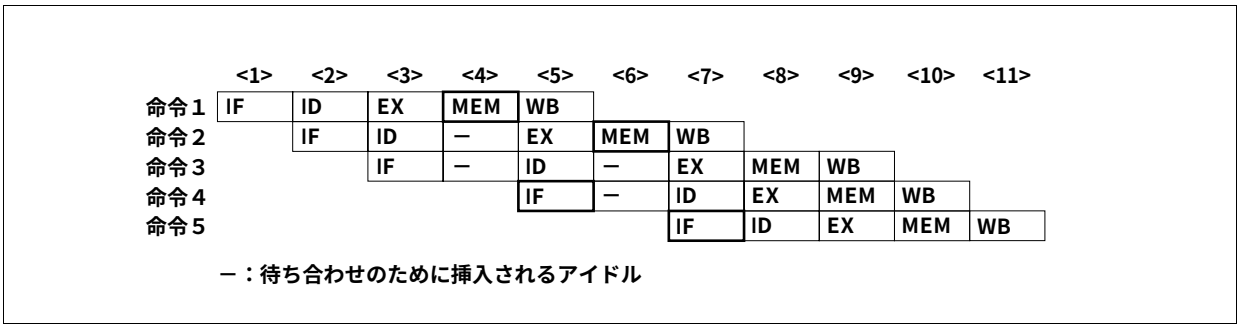
(1) V850ES CPU（ハーバード・アーキテクチャ）の場合

命令1のMEMと命令4のIF，および命令2のMEMと命令5のIFが同時に実行でき，パイプラインが乱れません。



(2) 非ハーバード・アーキテクチャの場合

命令1のMEMと命令4のIF，および命令2のMEMと命令5のIFが競合するためバスの待ち合わせが発生し，パイプラインが乱れ実行速度が低下します。



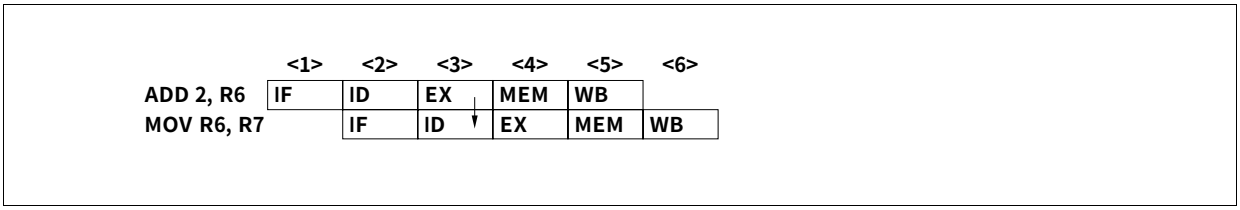
8.4.2　ショート・パス

V850ES CPUはショート・パスを内蔵しているため，前命令のライトバック（WB）が終了する前に，後続の命令がその結果を使用できます。

例1．算術演算命令，論理演算の実行結果を直後の命令で使用する場合

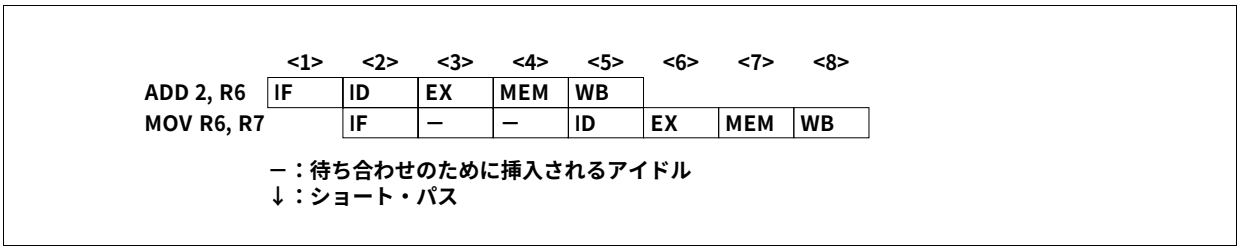
• V850ES CPU**（ショート・パス内蔵）の場合**

前命令のWB**を待たず実行結果が出た時点（**EX**ステージ）で，直後の命令の**ID**を処理できます。**



・ショート・パスがない場合

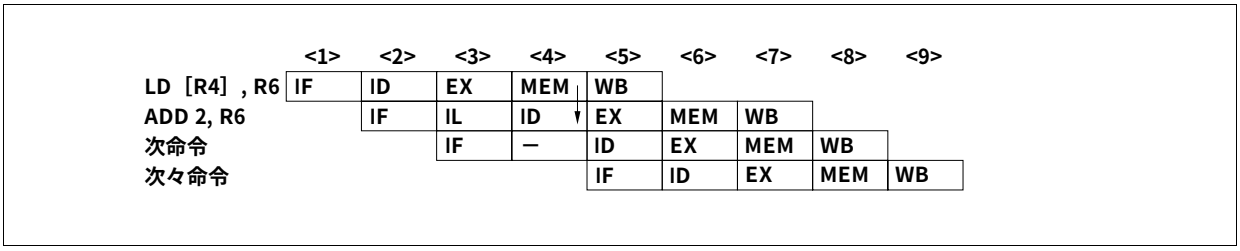
前命令のWB**まで，直後の**ID**が遅れます。**



例2．ロード命令によりメモリから読み出したデータを，直後の命令で使用する場合

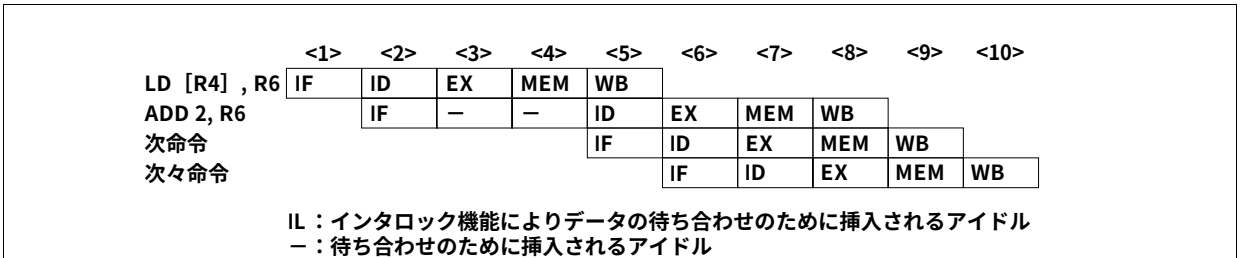
• V850ES CPU**（ショート・パス内蔵）の場合**

前命令のWB**を待たず実行結果が出た時点（**MEM**ステージ）で，直後の命令の**ID**を処理できます。**



・ショート・パスがない場合

前命令のWB**まで，直後の**ID**が遅れます。**



付録A　注意事項

A. 1　sld命令と割り込み競合に関する制限事項

A. 1. 1　内　　容

次の命令<1>の事項が完了する前に，後続のsld命令の直前の命令<2>のデコード動作と割り込み要求が競合した場合，先の命令<1>の実行結果がレジスタに格納されないことがあります。

命令<1>

- ld命令　　：ld.b, ld.h, ld.w, ld.bu, ld.hu
- sld命令　　：sld.b, sld.h, sld.w, sld.bu, sld.hu
- 乗算命令　　：mul, mulh, mulhi, mulu

命令<2>

mov　reg1, reg2	not　reg1, reg2	satsubr　reg1, reg2	satsub　reg1, reg2
satadd　reg1, reg2	satadd　imm5, reg2	or　reg1, reg2	xor　reg1, reg2
and　reg1, reg2	tst　reg1, reg2	subr　reg1, reg2	sub　reg1, reg2
add　reg1, reg2	add　imm5, reg2	cmp　reg1, reg2	cmp　imm5, reg2
mulh　reg1, reg2	shr　imm5, reg2	sar　imm5, reg2	shl　imm5, reg2

<例>

< i >　ld.w　[r11], r10
　　・
　　・
　　・
< ii >　mov　r10, r28
<iii>　sld.w　0x28, r10

< i >のld命令の実行が完了する前に，<iii>のsld命令の直前のmov命令< ii >のデコード動作と割り込み要求が競合した場合，< i >のld命令の実行結果がレジスタに格納されないことがあります。

A. 1. 2　回 避 策

命令< ii >の直後にsld命令を実行する場合は，次のいずれかの方法を用いて，上記動作を回避してください。

- sld命令の直前にnop命令を入れる。
- sld命令のディスティネーション・レジスタと同じレジスタを，sld命令の直前で実行する上記< ii >の命令で使用しない。

★ A. 2　mul/mulu命令に関する制限事項

A. 2. 1　内　容

内蔵RAM領域からのロード命令（ld, sld）の直後にあるmul, mulu命令の乗算処理が完了する前に，後続の内蔵RAMまたは内蔵ROM領域のミスアライン・アドレスに対するロード命令（ld, sld）を実行した場合に，mul, mulu命令とミスアライン･アドレスに対するロード命令の実行結果のレジスタへの格納値が異常になる場合があります。

この現象は，mul, mulu命令のオペランドとして，次のいずれかのレジスタ指定を行った場合にのみ発生する場合があります。

（a）reg3にr0を指定

（b）reg2とreg3に同じレジスタを指定

＜例＞

< ii >のmul命令の乗算処理が完了する前に，< iii >のミスアライン・アドレスへのld命令を実行した場合，< ii >の乗算命令と< iii >のロード命令の実行結果のレジスタへの格納値が異常になる場合があります。

```
＜i＞   ld.h    [r11],r10       :内蔵RAM領域に対するロード命令
< ii >  mul     r12,r13,r0
        ・
        ・
< iii > ld.w    2[r14],r15      ：内蔵RAM または 内蔵ROM領域のミスアライン・
                                  アドレスに対するロード命令
```

A. 2. 2　回避策

mul, mulu命令のオペランドとして，次の両方の条件を満たすレジスタを使用してください。

（a）reg3にr0を使用しない。

（b）reg2とreg3は異なるレジスタを使用する。

付録B　命令一覧

アルファベット順の命令機能一覧を表B－1に，フォーマット順の命令一覧を表B－2に示します。

表B－1　命令機能一覧（アルファベット順）（1/11）

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。
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	0/1	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B－1　命令機能一覧（アルファベット順）（2/11）

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クリアします。
CLR1	reg2 [reg1]	IX	－	－	－	0/1	－	ビット・クリア。 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クリアします。
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。

表B－1　命令機能一覧（アルファベット順）（3/11）

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	
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。
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に示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に示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。
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B－1　命令機能一覧（アルファベット順）（4/11）

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。
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。
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。
LD.B	disp16 [reg1] , reg2	VII	－	－	－	－	－	バイト・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。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・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。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	－	－	－	－	－	ハーフワード・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。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。
LD.HU	disp16 [reg1] , reg2	VII	－	－	－	－	－	符号なしハーフワード・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。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・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。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。

表B－1　命令機能一覧（アルファベット順）（5/11）

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	
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の場合は，PSWのフラグにはreg2の対応するビットの値が設定されます。
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にダブルワード・データとして格納します。
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。
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にワード・データとして格納します。
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にワード・データとして格納します。
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。

表B－1　命令機能一覧（アルファベット順）（6/11）

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	
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にダブルワード・データとして格納します。reg1は影響を受けません。
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に表示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。
PREPARE	list12, imm5, sp/imm	XIII	－	－	－	－	－	スタック・フレームの生成。 list12に表示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。

表B－1　命令機能一覧（アルファベット順）（7/11）

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	
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。
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。
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。

表B－1　命令機能一覧（アルファベット順）（8/11）

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	
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。
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。
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。

表B－1　命令機能一覧（アルファベット順）（9/11）

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	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。
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。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。
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。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。
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。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。
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。
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・データを生成したアドレスに格納します。

表B－1　命令機能一覧（アルファベット順）（10/11）

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.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・データを生成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B－1　命令機能一覧（アルファベット順）（11/11）

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B－2　命令一覧（フォーマット順）（1/3）

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
	0000000010RRRRR	－	SWITCH	reg1
	0000000011RRRRR	－	JMP	[reg1]
	rrrrr000100RRRRR	－	SATSUBR	reg1, reg2
	rrrrr000101RRRRR	－	SATSUB	reg1, reg2
	rrrrr000110RRRRR	－	SATADD	reg1, reg2
	rrrrr000111RRRRR	－	MULH	reg1, reg2
	0000000100RRRRR	－	ZXB	reg1
	0000000101RRRRR	－	SXB	reg1
	0000000110RRRRR	－	ZXH	reg1
	0000000111RRRRR	－	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

表B－2　命令一覧（フォーマット順）（2/3）

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		
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]
V	rrrrr11110dddddd	ddddddddddddddd0	JARL	disp22, reg2
	0000011110dddddd	ddddddddddddddd0	JR	disp22
VI	rrrrr110000RRRRR	iiiiiiiiiiiiiiii	ADDI	imm16, reg1, reg2
	rrrrr110001RRRRR	iiiiiiiiiiiiiiii	MOVEA	imm16, reg1, reg2
	rrrrr110010RRRRR	iiiiiiiiiiiiiiii	MOVHI	imm16, reg1, reg2
	rrrrr110011RRRRR	iiiiiiiiiiiiiiii	SATSUBI	imm16, reg1, reg2
	0000011000 1RRRRR	**注**	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
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
VIII	00bbb111110RRRRR	ddddddddddddddd	SET1	bit#3, disp16 [reg1]
	01bbb111110RRRRR	ddddddddddddddd	NOT1	bit#3, disp16 [reg1]
	10bbb111110RRRRR	ddddddddddddddd	CLR1	bit#3, disp16 [reg1]
	11bbb111110RRRRR	ddddddddddddddd	TST1	bit#3, disp16 [reg1]

注　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
iiiiiiiiiiiiiiii	IIIIIIIIIIIIIIII

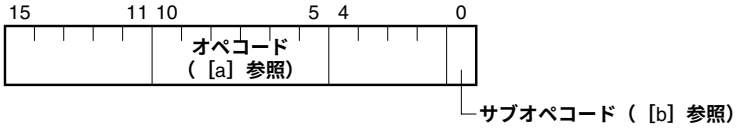
表B－2　命令一覧（フォーマット順）（3/3）

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		
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
X	0000011111iiiii	0000000100000000	TRAP	vector
	0000011111100000	0000000100100000	HALT	－
	0000011111100000	0000000101000000	RETI	－
	0000011111100000	0000000101000100	CTRET	－
	0000011111100000	0000000101000110	DBRET	－
	0000011111100000	0000000101100000	DI	－
	1000011111100000	0000000101100000	EI	－
XI	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
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
XIII	0000011001iiiiiL	LLLLLLLLLLLRRRRR	DISPOSE	imm5, list12, [reg1]
	0000011001iiiiiL	LLLLLLLLLLL00000	DISPOSE	imm5, list12
	0000011110iiiiiL	LLLLLLLLLLL00001	PREPARE	list12, imm5
	0000011110iiiiiL	LLLLLLLLLLLff011	PREPARE	list12, imm5, sp/imm

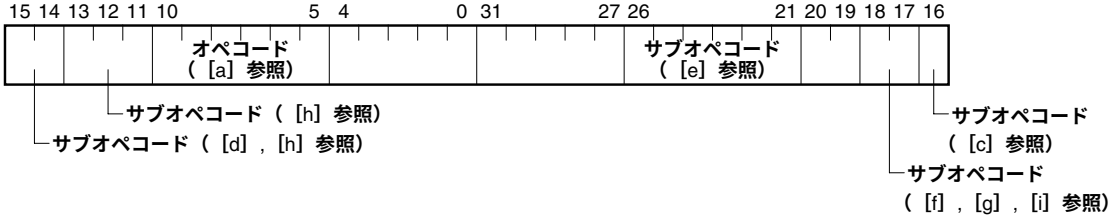
付録C　命令オペコード・マップ

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 未定義[注3]	JMP[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
0	1	0	1	SHR imm5, r	SAR imm5, r	SHL imm5, r	MULH imm5, r 未定義[注4]	
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 未定義[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 JARL 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, 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]	**未定義**	XI, XII
0	1	1	1	**不正命令**				−
1	x	x	x					

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。**

[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	未定義

付録D　V850 CPU, V850E1 CPUとのアーキテクチャ上の相違点

(1/3)

項　目		V850ES CPU	V850E1 CPU	V850 CPU
命令 (オペランドを含む)	BSH　reg2, reg3	あり		なし
	BSW　reg2, reg3			
	CALLT　imm6			
	CLR1　reg2, [reg1]			
	CMOV　cccc, imm5, reg2, reg3			
	CMOV　cccc, reg1, reg2, reg3			
	CTRET			
	DBRET	あり	あり[注]	
	DBTRAP			
	DISPOSE　imm5, list12	あり		
	DISPOSE　imm5, list12 [reg1]			
	DIV　reg1, reg2, reg3			
	DIVH　reg1, reg2, reg3			
	DIVHU　reg1, reg2, reg3			
	DIVU　reg1, reg2, reg3			
	HSW　reg2, reg3			
	LD.BU　disp16 [reg1], reg2			
	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			

注　NB85E, NB85ETではサポートしていません。

（2/3）

項　目		V850ES CPU	V850E1 CPU	V850 CPU
命令フォーマット	Format IV	V850ESおよびV850E1 CPUとV850 CPUで，一部，形式が異なります。		
	Format XI	あり		なし
	Format XII			
	Format XIII			
命令クロック実行数（MUL, MULU命令を除く）		V850ESおよびV850E1 CPUとV850 CPUで，一部，クロック数が異なります。		
	MUL, MULU命令	1/4/5クロック	1/2/2クロック	なし
プログラム空間		64 Mバイト・リニア（使用可能領域16 Mバイト＋60 Kバイト）	64 Mバイト・リニア	16 Mバイト・リニア
プログラム・カウンタ（PC）の有効ビット		下位26ビット		下位24ビット
システム・レジスタ	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）			
	ディバグ・インタフェース・レジスタ（DIR）		あり[注1]	
	ブレークポイント制御レジスタ0, 1（BPC0, BPC1）	なし		
	プログラムIDレジスタ（ASID）			
	ブレークポイント・アドレス設定レジスタ0, 1（BPAV0, BPAV1）			
	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）			
	ブレークポイント・データ・マスク・レジスタ0, 1（BPDM0, BPDM1）			
	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		許可固定	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[注2]		1
	例外コード	0010H, 0020H[注2], 0030H[注2]		0010H
	ハンドラ・アドレス	00000010H, 00000020H[注2], 00000030H[注2]		00000010H
ディバグ・トラップ		あり	あり[注3]	なし

注1. NU85E, NU85ETのみ使用できます。

2. 製品により，搭載していない場合があります。

3. NB85E, NB85ETではサポートしていません。

（3/3）

項　　目		V850ES CPU	V850E1 CPU	V850 CPU
パイプライン	• ワード・データ乗算命令	注1	注1	命令なし
	• ワード・データ乗算命令を除く算術演算命令 • 分岐命令 • ビット操作命令 • 特殊命令（TRAP, RETI）	注2		注2

注1. V850ES CPUコアとV850E1 CPUコアでは，パイプラインの流れが異なります。詳細は，第8章　パイプライン，および，V850E1　ユーザーズ・マニュアル　アーキテクチャ編（U14559J）を参照してください。

2. V850ES CPUコアおよびV850E1 CPUコアと，V850 CPUコアでは，パイプラインの流れが異なります。詳細は，第8章　パイプライン，V850E1　ユーザーズ・マニュアル　アーキテクチャ編（U14559J），および，V850シリーズ　ユーザーズ・マニュアル　アーキテクチャ編（U10243J）を参照してください。

付録E　V850 CPUに対してV850ES CPUで追加した命令

V850ES 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。V850ES 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。

次の表にV850ES CPUで追加した命令コードに対応したV850 CPUの命令を示します。V850 CPUを搭載した製品からV850ES 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。

なお，V850ES CPUは，V850E1 CPUの全命令コードに対して互換性を保っているため，容易に製品を置き換えることができます。

表E－1　V850ES 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）

V850ES CPUで追加した命令	V850ES 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
（RFU）	MULH　imm5, r0
（RFU）	MULHI　imm16, reg1, r0
BSH　reg2, reg3	不正命令
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	
HSW　reg2, reg3	
MUL　imm9, reg2, reg3	
MUL　reg1, reg2, reg3	
MULU　reg1, reg2, reg3	
MULU　imm9, reg2, reg3	
SASF　cccc, reg2	
CLR1　reg2, [reg1]	未定義
DBRET	

表E－1　V850ES 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）

V850ES CPUで追加した命令	V850ES CPUの命令コードと同一のV850 CPUの命令
DBTRAP	未定義
LD.BU　disp16 [reg1], reg2	
LD.HU　disp16 [reg1], reg2	
NOT1　reg2, [reg1]	
PREPARE　list12, imm5	
PREPARE　list12, imm5, sp/imm	
SET1　reg2, [reg1]	
SLD.BU　disp4 [ep], reg2	
SLD.HU　disp5 [ep], reg2	
TST1　reg2, [reg1]	

付録F　改版履歴

F. 1　本版で改訂された主な箇所

箇　所	内　　容
U15943JJ4V0UM00 → U15943JJ4V1UM00	
p.142	6. 2. 1　**ソフトウエア例外を訂正**
U15943JJ3V0UM00 → U15943JJ4V0UM00	
p.6	**はじめに　製品タイプを変更**
p.84	5.3 **命令セット** MUL [**注意**]**の記述を追加**
p.88	5.3 **命令セット** MULU [**注意**]**の記述を追加**
p.135	**表**5－6　**命令実行クロック数一覧を変更**
p.143	6. 2. 2　**例外トラップの記述を変更**
p.143	**図**6－5　**例外トラップの処理形態を変更**
p.158	8. 2. 7（3）JMP**命令を変更**
p.160	8. 2. 9（2）CTRET**命令を変更**
p.161	8. 2. 9（4）DISPOSE**命令を変更**
p.171	A. 2　mul/mulu**命令に関する制限事項を追加**

F. 2　前版までの改版履歴

前版までの改版履歴を次に示します。なお，適用箇所は各版での章を示します。

版　数	前版までの改版内容	適用箇所
第2版	図1－1　V850シリーズのCPU展開　V850ES CPUコアの記述変更	第1章　概　　説
	5. 3　命令セット　MUL　［注意］の記述を追加	第5章　命　　令
	5. 3　命令セット　MULU　［注意］の記述を追加	
	付録C　V850 CPU, V850E1 CPUとのアーキテクチャ上の相違点　パイプラインの記述変更	付録C　V850 CPU, V850E1 CPUとのアーキテクチャ上の相違点
	付録F　改版履歴　追加	付録F　改版履歴
第3版	図2－1　レジスター覧　変更	第2章　レジスタ・セット
	2. 1（1）汎用レジスタ（r0-r31）　説明変更	
	表2－2　システム・レジスタ番号　変更	
	2. 2. 8　ディバグ・インタフェース・レジスタ（DIR）　追加	
	5. 3　命令セット　MUL　注意変更	第5章　命　　令
	5. 3　命令セット　MULU　注意変更	
	5. 3　命令セット　SLD.B　注意（2）追加	
	5. 3　命令セット　SLD.BU　注意（2）追加	
	5. 3　命令セット　SLD.H　注意（2）追加	
	5. 3　命令セット　SLD.HU　注意（2）追加	
	5. 3　命令セット　SLD.W　注意（2）追加	
	表5－6　命令実行クロック数一覧　注4追加	
	6. 2. 3　ディバグ・トラップ　説明追加	第6章　割り込みと例外
	6. 3. 1　割り込み，ソフトウエア例外からの復帰　注削除	
	6. 3. 2　例外トラップ，ディバグ・トラップからの復帰　<3>追加	
	表7－1　リセット後のレジスタの状態　説明追加	第7章　リセット
	8. 1. 2　2クロック分岐　備考追加	第8章　パイプライン
	付録A　注意事項　追加	付録A　注意事項
	付録D　V850 CPU, V850E1 CPUとのアーキテクチャ上の相違点　変更	付録D　V850 CPU, V850E1 CPUとのアーキテクチャ上の相違点
	付録F　改版履歴　変更	付録F　改版履歴

【発行】NEC**エレクトロニクス株式会社（**http://www.necel.co.jp/**）**

【問い合わせ先】http://www.necel.com/contact/ja/