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

V850 ファミリ™

32 ビット・シングルチップ・マイクロコンピュータ

アーキテクチャ編

資料番号　U10243JJ7V0UM00（第７版）
発行年月　March 2001　N　CP(K)

〔メ　モ〕

目次要約

CMOSデバイスの一般的注意事項

①静電気対策（MOS全般）

注意　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，NEC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。

②未使用入力の処理（CMOS特有）

注意　CMOSデバイスの入力レベルは固定してください。

バイポーラやNMOSのデバイスと異なり，CMOSデバイスの入力に何も接続しない状態で動作させると，ノイズなどに起因する中間レベル入力が生じ，内部で貫通電流が流れて誤動作を引き起こす恐れがあります。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，未使用端子が出力となる可能性（タイミングは規定しません）を考慮すると，個別に抵抗を介してV_{DD}またはGNDに接続することが有効です。

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

③初期化以前の状態（MOS全般）

注意　電源投入時，MOSデバイスの初期状態は不定です。

分子レベルのイオン注入量等で特性が決定するため，初期状態は製造工程の管理外です。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。

注意： μPD703008Y, 70F3008Y, 703014AY, 703014BY, 703015Y, 703015AY, 703015BY, 70F3015BY, 703017AY, 70F3017AY, 703030AY, 703031AY, 703032AY, 70F3032AY, 703033AY, 70F3033AY, 703034AY, 703035AY, 70F3035AY, 703036AY, 703037AY, 70F3037AY, 703078Y, 703079Y, 70F3079Y, 703038Y, 70F3038Y, 703039Y, 703040Y, 703041Y, 70F3040YはI^2Cバス・インタフェース回路を内蔵しています。
I^2Cバス・インタフェースを使用される場合には，カスタム・コードをご発注いただく時に，事前にその旨ご申告下さい。申告に基づき，以下の特典が受けられます。
日本電気株式会社のI^2Cバス対応部品をご購入いただくことにより，これらの部品をI^2Cシステムに使用する実施権がフィリップス社I^2C特許に基づき許諾されることになります。ただし，これらのI^2Cシステムはフィリップス社によって設定されたI^2C標準規格に合致しているものとします。

Purchase of NEC I^2C components conveys a license under the Philips I^2C Patent Rights to use these components in an I^2C system, provided that the system conforms to the I^2C Standard Specification as defined by Philips.

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

- **本資料の内容は予告なく変更することがありますので，最新のものであることをご確認の上ご使用ください。**
- **文書による当社の承諾なしに本資料の転載複製を禁じます。**
- **本資料に記載された製品の使用もしくは本資料に記載の情報の使用に際して，当社は当社もしくは第三者の知的財産権その他の権利に対する保証または実施権の許諾を行うものではありません。上記使用に起因する第三者所有の権利にかかわる問題が発生した場合，当社はその責を負うものではありませんのでご了承ください。**
- **本資料に記載された回路，ソフトウエア，及びこれらに付随する情報は，半導体製品の動作例，応用例を説明するためのものです。従って，これら回路・ソフトウエア・情報をお客様の機器に使用される場合には，お客様の責任において機器設計をしてください。これらの使用に起因するお客様もしくは第三者の損害に対して，当社は一切その責を負いません。**
- **当社は品質，信頼性の向上に努めていますが，半導体製品はある確率で故障が発生します。当社半導体製品の故障により結果として，人身事故，火災事故，社会的な損害等を生じさせない冗長設計，延焼対策設計，誤動作防止設計等安全設計に十分ご注意願います。**
- **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頂く「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の上ご使用願います。**

  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

  特別水準：輸送機器（自動車，列車，船舶等），交通用信号機器，防災／防犯装置，各種安全装置，生命維持を直接の目的としない医療機器

  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。当社製品を上記の「標準水準」の用途以外でご使用をお考えのお客様は，必ず事前に当社販売窓口までご相談頂きますようお願い致します。

M7　98.8

本版で改訂された主な箇所

箇　　所	内　　容
全般	対象デバイス　変更
p. 18	**図１－１　V850ファミリ製品展開**　変更
p. 20	**２.１.１（１）汎用レジスタ**　記述修正
p. 21	**図２－１　プログラム・レジスター覧**　修正
p. 22	**図２－２　プログラム・レジスタ動作一覧**　修正
p. 107	**表５－10　命令実行クロック数一覧**　修正
p. 108	**第６章　割り込みと例外**　記述修正
p. 113	**６.２.２　例外トラップ**　記述修正
p. 113	**図６－５　例外トラップの処理形態**　修正

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。

巻末にアンケート・コーナを設けております。このドキュメントに対するご意見をお気軽にお寄せください。

はじめに

対象者　このマニュアルは，V850ファミリの中のV850 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。

●V850 CPUコア搭載製品

- V851™注1　：μPD703000, 703001, 70P3000
- V852™注2　：μPD703002, 70P3002
- V853™　：μPD703003A，70F3003A，703003A（A）注3，70F3003A（A）注3，703004A, 703025A, 703025A（A）, 70F3025A
- V854™注2　：μPD703006注1, 703008, 70F3008, 703008Y, 70F3008Y
- V850/SA1™　：μPD703014A, 703014AY, 703014B, 703014BY, 703015注2, 703015Y注2, 703015A, 703015AY, 703015B, 703015BY, 70F3015B注3, 70F3015BY注3, 703017A, 70F3017A, 703017AY, 70F3017AY
- V850/SB1™　：μPD703030A注3，703030AY注3，703031A，703031AY，703032A，70F3032A, 703032AY, 70F3032AY, 703033A, 70F3033A, 703033AY, 70F3033AY
- V850/SB2™　：μPD703034A, 703034AY, 703035A, 70F3035A, 703035AY, 70F3035AY, 703036A注3, 703036AY注3, 703037A, 70F3037A, 703037AY, 70F3037AY
- V850/SF1™　：μPD703078Y, 703079Y, 70F3079Y
- V850/SV1™　：μPD703038注3，703038Y注3，70F3038注3，70F3038Y注3，703039, 703039Y, 703040, 703040Y, 703041, 703041Y, 70F3040, 70F3040Y

注1．保守品
　2．非拡販品
　3．開発中

目　的　このマニュアルは，次の構成に示すV850ファミリ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
●パイプラインの操作

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

→各デバイスの**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。

電気的特性を知りたいとき

→各デバイスの**データ・シート**を参照してください。

一通りV850ファミリの機能を理解しようとするとき

→目次に従ってお読みください。

なお，V850ファミリでは２バイト構成のデータをハーフワード，４バイト構成のデータをワードと呼びます。

凡　　例

データの重み	：左が上位桁，右が下位桁
アクティブ・ロウの表記	：$\overline{\times\times\times}$（端子，信号名称の上に線）
メモリ・マップのアドレス	：上部－上位，下部－下位
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…××××または××××B
	10進数…××××
	16進数…××××H

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）…$2^{30} = 1024^3$

データ・タイプ	：ワード…32ビット
	ハーフワード…16ビット
	バイト…８ビット

関連資料　関連資料は暫定版の場合がありますが，この資料では「暫定」の表示をしておりません。
あらかじめご了承ください。

・デバイスに関する資料

品名＼資料名		データ・シート	ユーザーズ・マニュアル	
			ハードウエア編	アーキテクチャ編
V851	μPD703000, 703001	U10987J	U10935J	このマニュアル
	μPD70P3000	U10988J		
V852	μPD703002	U11826J	U10038J	
	μPD70P3002	U11827J		
V853	μPD703003A, 703004A, 703025A	U13188J	U10913J	
	μPD70F3003A, 70F3025A	U13189J		
V854	μPD70F3008	U12756J	U11969J	
	μPD70F3008Y	U12755J		
	μPD703006, 703008, 703008Y	－		
V850/SA1	μPD703014A, 703014AY, 703015A, 703015AY, 703017A, 703017AY	U14526J	U12768J	
	μPD70F3017A, 70F3017AY	U14527J		
V850/SB1	μPD703031A, 703031AY, 703033A, 703033AY, 70F3033A, 70F3033AY	U14734J	U13850J	
	μPD703032A, 703032AY, 70F3032A, 70F3032AY	U14893J		
V850/SB2	μPD703034A, 703034AY, 703035A, 703035AY, 70F3035A, 70F3035AY	U14780J		
	μPD703037A, 703037AY, 70F3037A, 70F3037AY	U14894J		
V850/SF1	μPD703078Y, 703079Y, 70F3079Y	U15183J	U14665J	
V850/SV1	μPD703039, 703039Y, 703040, 703040Y, 703041, 703041Y	U13953J	U14462J	
	μPD70F3040, 70F3040Y	U14662J		

・開発ツールに関する資料（ユーザーズ・マニュアル）

資　料　名		資料番号
IE-703002-MC（V852, V853, V854, V850/SA1, V850/SB1, V850/SB2, V850/SF1, V850/SV1用インサーキット・エミュレータ）		U11595J
IE-703003-MC-EM1（V853用周辺I/Oボード）		U11596J
IE-703008-MC-EM1（V854用周辺I/Oボード）		U12420J
IE-703017-MC-EM1（V850/SA1用周辺I/Oボード）		U12898J
IE-703037-MC-EM1（V850/SB1, V850/SB2用周辺I/Oボード）		U14151J
IE-703040-MC-EM1（V850/SV1用周辺I/Oボード）		U14337J
CA850（Ver. 2.30 以上）（Cコンパイラ・パッケージ）	操作編	U14568J
	C言語編	U14566J
	プロジェクト・マネージャ編	U14569J
	アセンブリ言語編	U14567J
ID850（Ver. 2.20 以上）（統合ディバッガ）	操作編　Windows™ベース	U14580J
SM850（Ver. 2.20 以上）（システム・シミュレータ）	操作編　Windowsベース	U14782J
SM850（Ver. 2.00 以上）（システム・シミュレータ）	外部部品ユーザ・オープン・インタフェース仕様編	U14873J
RX850（Ver. 3.13 以上）（リアルタイムOS）	基礎編	U13430J
	インストレーション編	U13410J
	テクニカル編	U13431J
RX850 Pro（Ver. 3.13）（リアルタイムOS）	基礎編	U13773J
	インストレーション編	U13774J
	テクニカル編	U13772J
RD850（Ver. 3.01）（タスク・ディバッガ）		U13737J
RD850 Pro（Ver. 3.01）（タスク・ディバッガ）		U13916J
AZ850（Ver. 3.0）（システム・パフォーマンス・アナライザ）		U14410J
PG-FP3（フラッシュ・メモリ・プログラマ）		U13502J

目　　次

図の目次

表の目次

第１章　イントロダクション

V850ファミリは，V800シリーズ™におけるRISCマイクロプロセッサの技術を用いたCPUコアを持ち，内蔵ROM/RAM，周辺I/Oなどをワンチップに収めたシングルチップ・マイクロコンピュータです。

また，V850ファミリは，従来のNECオリジナル・シングルチップ・マイクロコンピュータ「78Kシリーズ」の上位製品として，高コスト・パフォーマンスを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ファミリには，V850 CPUを搭載した製品とV850E CPUを搭載した製品があります。このマニュアルでは，V850 CPUを搭載した製品を対象としています。

この章では，V850ファミリの概要を簡単に述べます。

１.１　概　　説

リアルタイム制御システムとしては，HDD（Hard Disk Drive）, PPC（Plain Paper Copier），プリンタ，ファクシミリを始めとする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主として用いられてきましたが，機器の制御の複雑化にしたがって，マイクロコンピュータに要求される機能も高度化し，命令セットが複雑になり，ハードウエア規模が増大化してきました。同時に，機器の性能向上に対応すべくマイクロコンピュータの動作周波数の高速化もあわせて求められています。

これらの要求に答えるためV850ファミリは，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シリーズのおよそ15倍の性能を低コストで実現する次世代のマイクロコンピュータ用のCPUコアとして位置づけられます。

V850ファミリでは，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。

1.2　特　　徴

○組み込み制御用高性能32ビット・アーキテクチャ

●命令数：74

●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 ：16 Mバイト・リニア

　　データ空間　　：４Gバイト・リニア

○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

●乗算器内蔵により，1-2クロックで乗算が可能（16ビット×16ビット→32ビット）

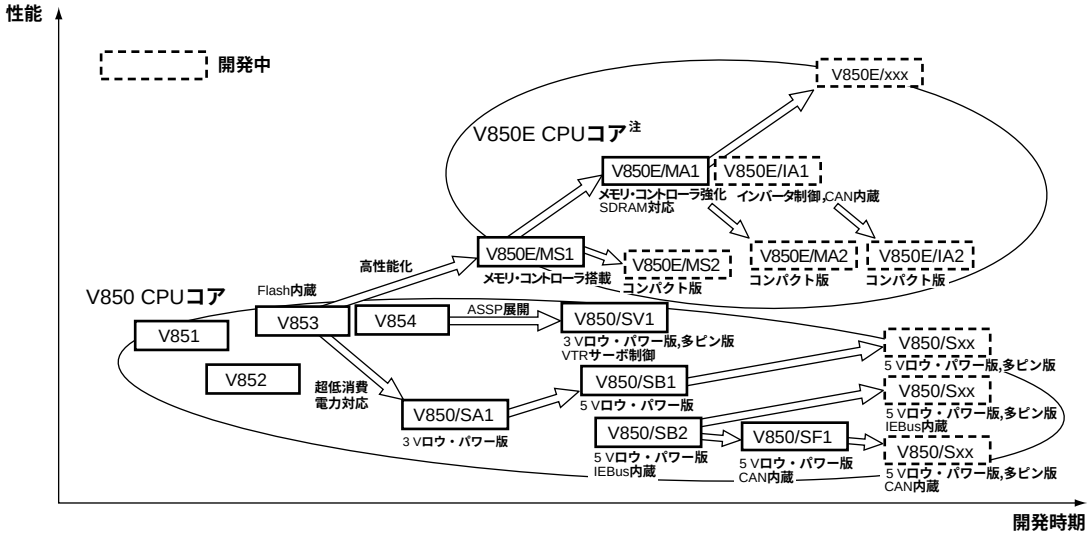
１.３　製品展開

V850ファミリは，RISCマイクロプロセッサをコアとしたシングルチップ・マイクロコンピュータです。

V850ファミリは，V850 CPUを搭載したV851, V852, V853, V854, V850/SA1, V850/SB1, V850/SB2, V850/SF1, V850/SV1と，V850E CPU を搭載したV850E/MS1™，V850E/MS2™，V850E/MA1™, V850E/MA2™，V850E/IA1™，V850E/IA2™，V850E/xxxがあります。

V850 CPUを搭載した製品は，制御系のシングルチップ・マイクロコンピュータであり，V850E CPUを搭載した製品は，外部バス・インタフェースの性能を強化し，制御系だけでなくデータ処理系に対応したシングルチップ・マイクロコンピュータです。

★ **図１－１　V850ファミリ製品展開**

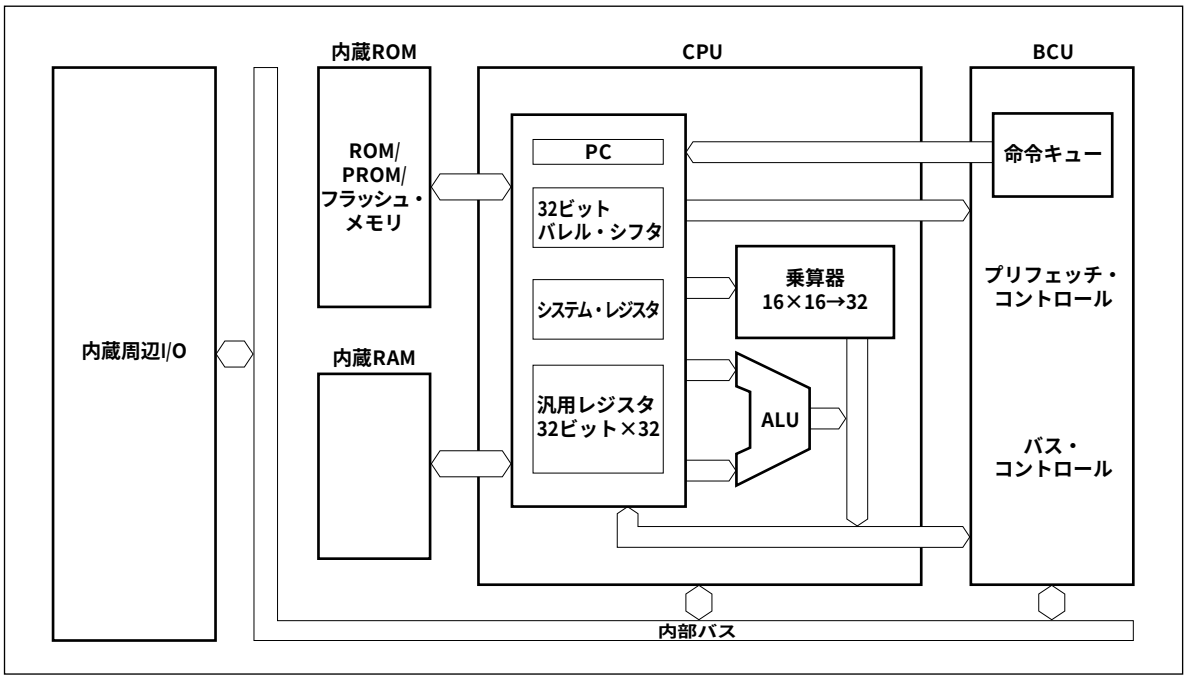


注　V850E CPUコアのアーキテクチャについては，**V850E/MS1　ユーザーズ・マニュアル　アーキテクチャ編（U12197J）**，**V850E1　ユーザーズ・マニュアル　アーキテクチャ編（U14559J）**を参照してください。

１．４　CPU内部構成

V850ファミリの内部構成を図１－２に示します。

図１－２　内部構成



各ハードウエア・ブロックの機能について次に説明します。

CPU …… 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），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。

内蔵ROM …… 00000000H番地からマッピングされるROMまたはEPROMまたはフラッシュ・メモリです。CPUから１クロックでアクセスすることができます（命令フェッチ時）。

内蔵RAM …… FFFFEFFFH番地以前にマッピングされるRAMです。CPUから１クロックでアクセスすることができます（データ・アクセス時）。

内蔵周辺I/O …… FFFFF000H番地からマッピングされる周辺I/O領域です。

BCU …… CPUで得られた物理アドレスに基づいて必要なバス・サイクルを起動します。
CPUからのバス・サイクル起動の要求がない場合は，プリフェッチ・アドレスを生成し，命令コードのプリフェッチを行います。プリフェッチした命令コードは内部の命令キューに取り込まれます。

第２章　レジスタ・セット

V850ファミリのレジスタ・セットは，一般のプログラム用として使用できるプログラム・レジスタ・セットと，実行環境の制御をすることができるシステム・レジスタ・セットの２つに分けることができます。レジスタはどれも32ビット幅を持っています。

２.１　プログラム・レジスタ

２.１.１　プログラム・レジスタ・セット

（１）汎用レジスタ

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，どれでもデータ変数またはアドレス変数として利用できます。

ただし，r0とr30は命令により暗黙的に使用されるので，これらのレジスタを使用する際には注意が必要です。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のアドレシングに使用されます。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。また，r1，r3，r4，ｒ５，r31は，アセンブラとCコンパイラにより暗黙的に使用されるので，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。r2は，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，変数用レジスタとしてr2を使用できます。

図２－１　プログラム・レジスター覧

★

31 0

r0	Zero Register
r1	Reserved for Address Generation
r2	
r3	Stack Pointer（SP）
r4	Global Pointer（GP）
r5	Text Pointer（TP）
r6	
r7	
r8	
r9	
r10	
r11	
r12	
r13	
r14	
r15	
r16	
r17	
r18	
r19	
r20	
r21	
r22	
r23	
r24	
r25	
r26	
r27	
r28	
r29	
r30	Element Pointer（EP）
r31	Link Pointer（LP）

PC	Program Counter

図２－２　プログラム・レジスタ動作一覧

名　称	用　途	動　作
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	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	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	テキスト領域[注]の先頭を指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	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	コンパイラが関数コールをするときに使用
PC	プログラム・カウンタ	プログラム実行中の命令アドレスを保持

注　テキスト領域：プログラム・コードを配置する領域を指します。

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，Cコンパイラ・パッケージ（CA850）のユーザーズ・マニュアルを参照してください。

（2）プログラム・カウンタ

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24ビットが有効でビット31-24は予約フィールドです（０に固定）。ビット23からビット24へのキャリーがあっても無視します。

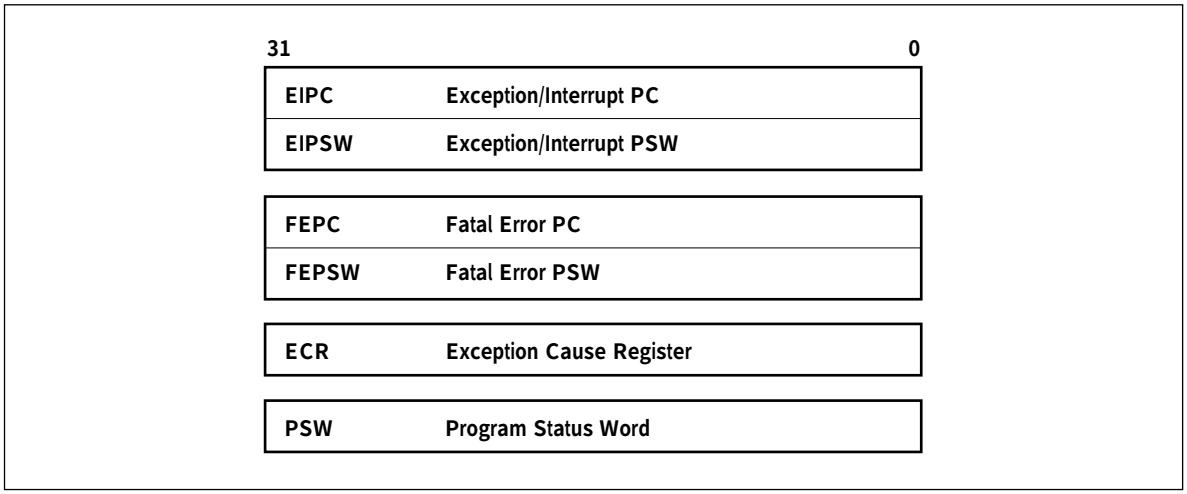
また，ビット０は０に固定されており，奇数番地への分岐はできません。

備考　RFU：予約フィールド（Reserved for Future Use）

２.２　システム・レジスタ

システム・レジスタは，V850ファミリの状態制御，割り込み情報保持などの役割を持ちます。

図２－３　システム・レジスター覧



２.２.１　割り込み時状態退避レジスタ

割り込み時状態退避レジスタには，「EIPC」，「EIPSW」の２個があります。

これらのレジスタには，例外や割り込みが発生した場合，PCおよびPSWがそれぞれ退避されます。ただし，NMI発生時には，NMI時状態退避レジスタに退避します。

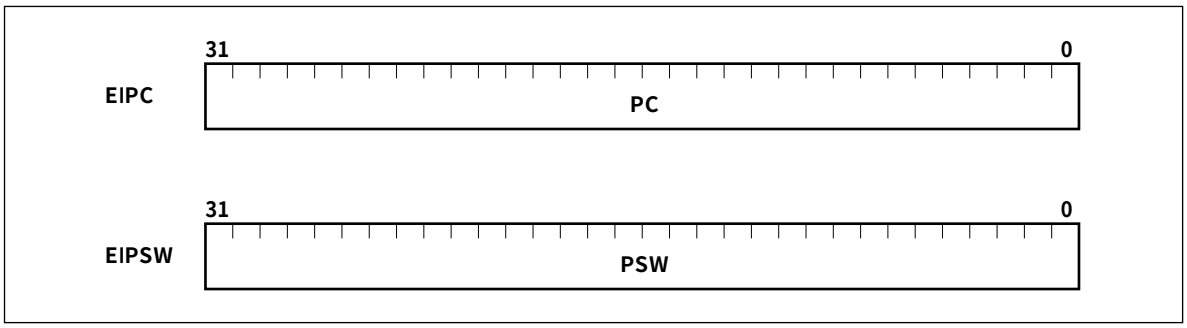
EIPCに退避される値（PC）は，次のようになっています。

例外時は次の命令のアドレスが退避されます。割り込みの場合も基本的には次の命令のアドレスが退避されますが，除算命令（DIVH）実行中に割り込みが発生した場合にかぎり，実行中の除算命令のアドレスが退避されます。

EIPSWには，現在のPSWの値が退避されます。

これらの，割り込み時状態レジスタは１組しかないため，多重割り込みを許す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。

なお，EIPCのビット24-31とEIPSWのビット８-31は０に固定されています。



２.２.２　NMI時状態退避レジスタ

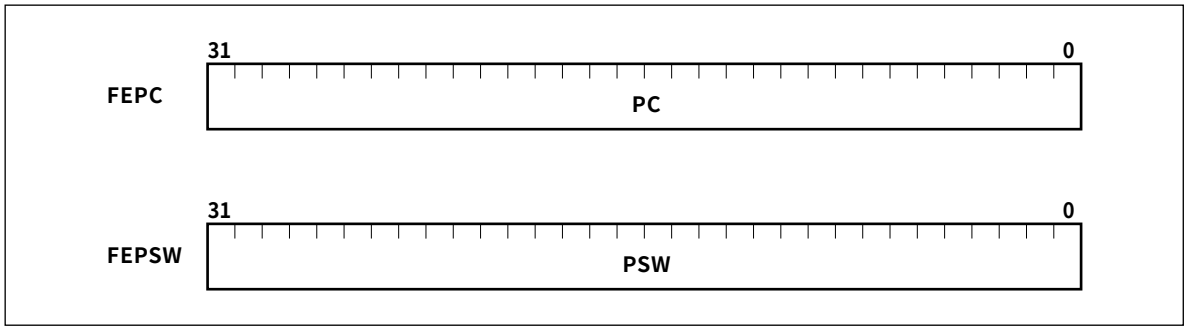
NMI時状態退避レジスタには，「FEPC」と「FEPSW」の２個があります。

これらのレジスタには，NMIが発生した場合，PCおよびPSWがそれぞれ退避されます。

FEPCに退避される値は，EIPCと同様に，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（除算命令実行中にNMIが発生した場合も同様に処理されます）。

FEPSWには，現在のPSWの値が退避されます。

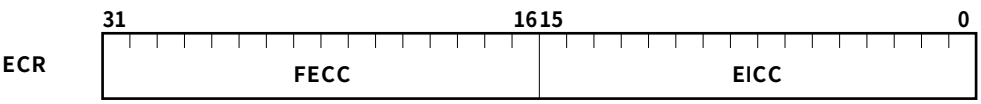
なお，FEPCのビット24-31とFEPSWのビット８-31は０に固定されています。



２.２.３　割り込み要因レジスタ

割り込み要因レジスタは，例外，マスカブル割り込み，NMI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ています。

このレジスタは読み出し専用です。したがって，LDSR命令を使ってこのレジスタに書き込むことはできません。



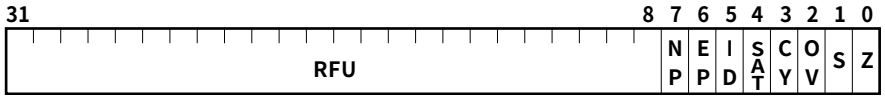
ビット位置	フィールド	意　味
31-16	FECC	Fatal Error Cause Code NMIコード
15-0	EICC	Exception/Interrupt Cause Code 例外／割り込みコード

2.2.4　プログラム・ステータス・ワード

プログラム・ステータス・ワードは，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フィールドの内容を変更した場合には，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，割り込みをLDSR命令実行中から禁止します。

（1/2）

PSW



ビット位置	フラグ	意　味
31-8	RFU	Reserved for Future Use 予約フィールドです（0に固定されています）。
7	NP	NMI Pending NMI処理中であることを示します。NMIが受け付けられるとセットされ，NMI要求をマスクして多重割り込みを禁止します。 NP=0：NMI処理中でない。 NP=1：NMI処理中である。
6	EP	Exception Pending 例外処理中であることを示します。例外の発生でセットされます。 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。 EP=0：例外処理中でない。 EP=1：例外処理中である。
5	ID	Interrupt Disable 外部からの割り込み要求を受け付ける状態かどうかを示します。 ID=0：割り込み可である。 ID=1：割り込み不可である。
4	SAT[注]	Saturated 飽和演算命令で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。累積フラグであり，飽和演算命令で演算結果がひとたび飽和すると，セット（1）され，以降の命令の演算結果が飽和しなくてもリセット（0）されません。リセット（0）するときは，PSWにデータをロードします。 なお，一般の算術演算ではセット（1）もリセット（0）もしません。 SAT=0：飽和していない。 SAT=1：飽和している。

備考　注の説明は次ページに記載しています。

(2/2)

ビット位置	フラグ	意　　味
3	CY	Carry 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。 CY=０：キャリーまたはボローは発生していない。 CY=１：キャリーまたはボローが発生した。
2	OV注	Overflow 演算中にオーバフローが発生したかどうかを示します。 OV=０：オーバフローは発生していない。 OV=１：オーバフローが発生した。
1	S注	Sign 演算の結果が負かどうかを示します。 S=０：演算の結果は正またはゼロであった。 S=１：演算の結果は負であった。
0	Z	Zero 演算の結果がゼロかどうかを示します。 Z=０：演算の結果はゼロでなかった。 Z=１：演算の結果はゼロであった。

注　飽和演算時のSフラグとOV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）した場合のみ，SATフラグはセット（１）されます

演算結果の状態	SAT-S-OV	飽和処理をした演算結果
正の最大値を越えた	1　0　1	7FFFFFFFH
負の最大値を越えた	1　1　1	80000000H
その他	0　×　0	演算結果そのもの

２.２.５　システム・レジスタ番号

システム・レジスタへの入出力は，システム・レジスタ・ロード／ストア命令（LDSR，STSR命令）で次のシステム・レジスタ番号を指定することで行います。

表２－１　システム・レジスタ番号

番号	システム・レジスタ名称	オペランド指定の可否	
		LDSR	STSR
0	EIPC	○	○
1	EIPSW	○	○
2	FEPC	○	○
3	FEPSW	○	○
4	ECR	－	○
5	PSW	○	○
6-31	予約		

－　：アクセス禁止

○　：アクセス可能

予約：アクセスした場合の動作の保証はしません。

注意　LDSR命令によりEIPCあるいはFEPCのビット０に１を設定後，割り込み処理を行い，RETI命令で復帰するときにビット０は無視されます（PCのビット０を０に固定しているため）。EIPC，FEPCにプログラムで値を設定する場合は，特別な理由のないかぎり偶数値（ビット０＝０）にしてください。

第３章　データ・タイプ

３.１　データ形式

V850ファミリがサポートするデータ・タイプを次に示します。

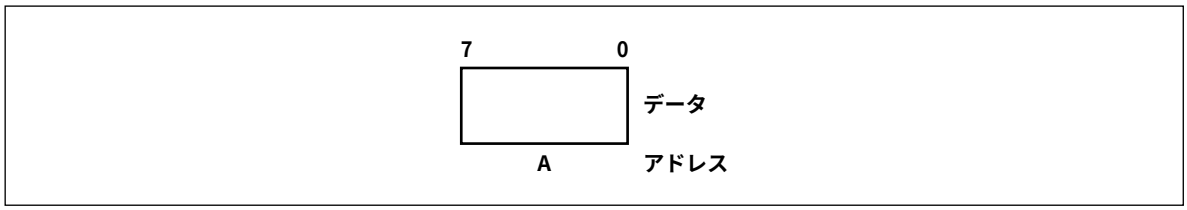
●整数（８，16，32ビット）
●符号なし整数（８，16，32ビット）
●ビット

３.１.１　データ・タイプとアドレシング

V850ファミリで扱うデータ長は，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）バイトである構成をしています（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形式を次に示します。

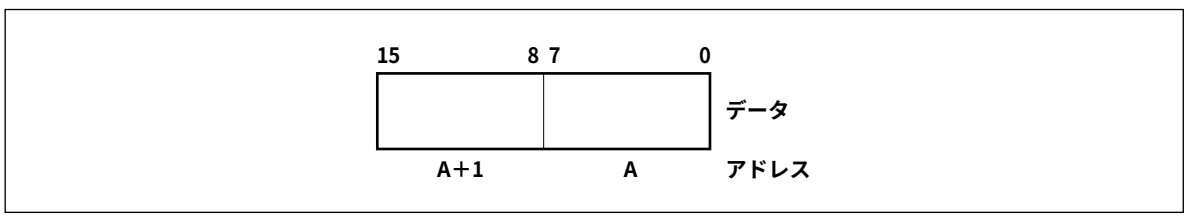
（１）バイト（BYTE）

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（Least Significant Bit）はビット０，MSB（Most Significant Bit）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。



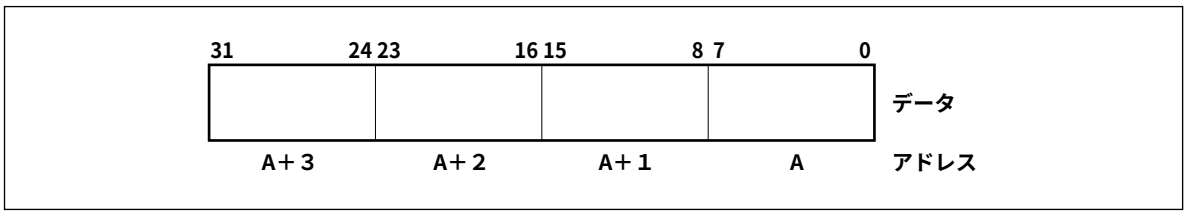
（２）ハーフワード（HALF-WORD）

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）で指定され，２つのバイトA，A＋１を占めます。



（３）ワード（WORD）

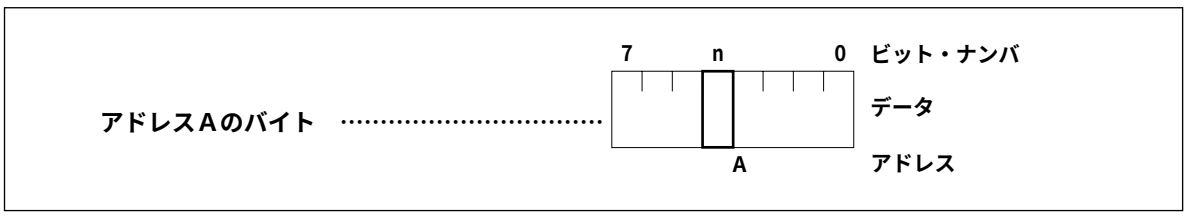
ワードは任意のワード境界[注]から始まる連続した４バイト（32ビット）のデータです。各ビットには０から31までの番号が付けられており，LSBはビット０，MSBはビット31に対応します。ワードはそのアドレスA（下位２ビットは０）で指定され，４つのバイトA，A＋１，A＋２，A＋３を占めます。



（４）ビット（BIT）

ビットは，任意のバイト境界[注]から始まる８ビット・データにおける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　**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

V850ファミリでは，整数は２の補数による２進数表現で表し，８ビット，16ビット，32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。

データ長		範　　囲
バイト	8 bit	－128～＋127
ハーフワード	16 bit	－32768～＋32767
ワード	32 bit	－2147483648～＋2147483647

3.2.2　符号なし整数

前項の整数が，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，８ビット，16ビット，32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したが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。

データ長		範　囲
バイト	8 bit	0 - 255
ハーフワード	16 bit	0 - 65535
ワード	32 bit	0 - 4294967295

3.2.3　ビ ッ ト

V850ファミリでは，ビット・データとしてクリア（０）あるいはセット（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・データに対してのみ行うことができ，次の４種類の操作ができます。

○セット
○クリア
○反転
○テスト

3.3　データ・アラインメント

V850ファミリではメモリに配置するワード・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）にアライン（整列）しなければなりません。アラインされていない場合は，アドレスの下位（ワード・データの場合は下位２ビット，ハーフワード・データの場合は下位１ビット）を自動的に０にマスクしてアクセスを行います。また，バイト・データの場合は，任意のアドレスに置くことができます。

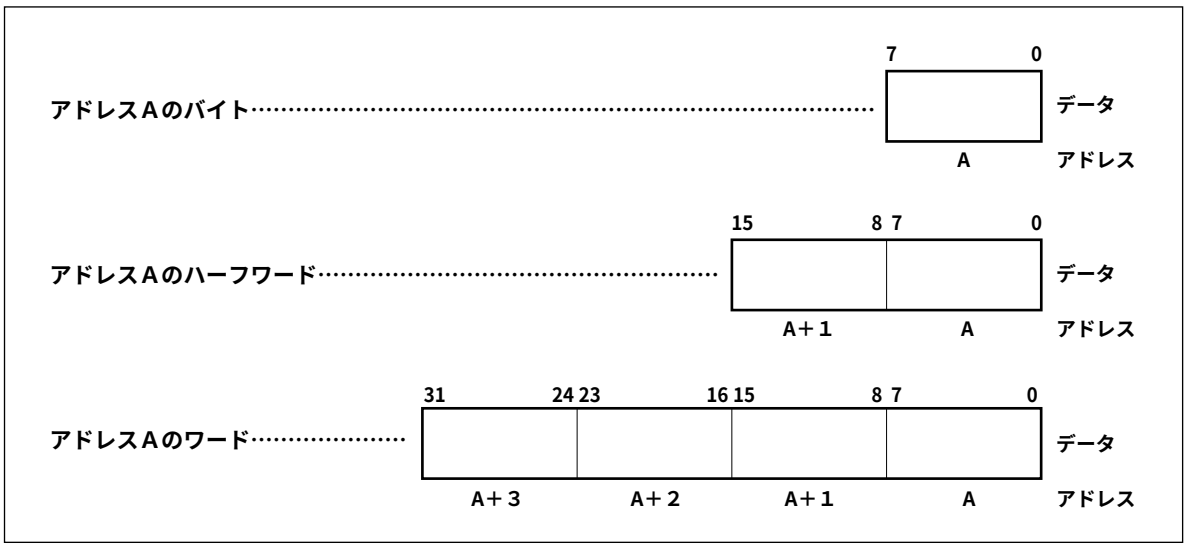
なお，アラインすることをアラインメントといいます。

第４章　アドレス空間

V850ファミリは，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**）。V850ファミリからメモリ，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**）。

V850ファミリでは，２バイト構成のデータ形態をハーフワード，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

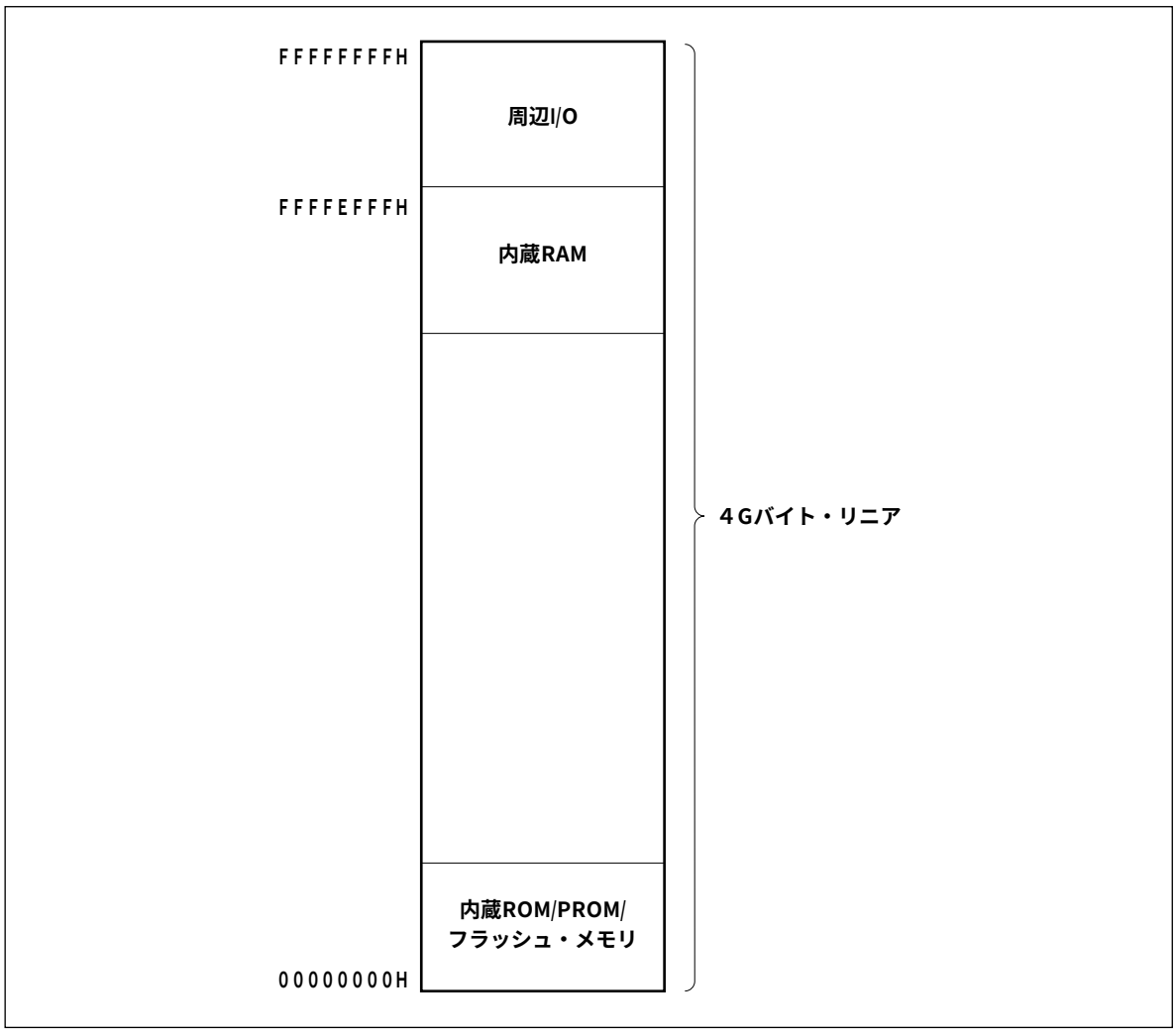
V850ファミリは，32ビット・アーキテクチャであり，オペランド・アドレシングにおいては，最大4Gバイトのリニア・アドレス空間（データ空間）をサポートしています。

一方，命令アドレスのアドレシングにおいては，最大16 Mバイトのリニア・アドレス空間（プログラム空間）をサポートしています。

V850ファミリのメモリ・マップを図４－１に示します。

内蔵ROM, RAMの容量は，製品ごとに異なりますので，詳細については各製品のユーザーズ・マニュアル　ハードウエア編　メモリ・マップの項を参照してください。

図４－１　メモリ・マップ



４.２　アドレシング・モード

V850ファミリのアドレス生成には，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，命令を実行するごとにフェッチする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が符号ビットとなります。

Bcond disp9, JR disp22, JARL disp22, reg2 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（JR disp22/JARL disp22, reg2）

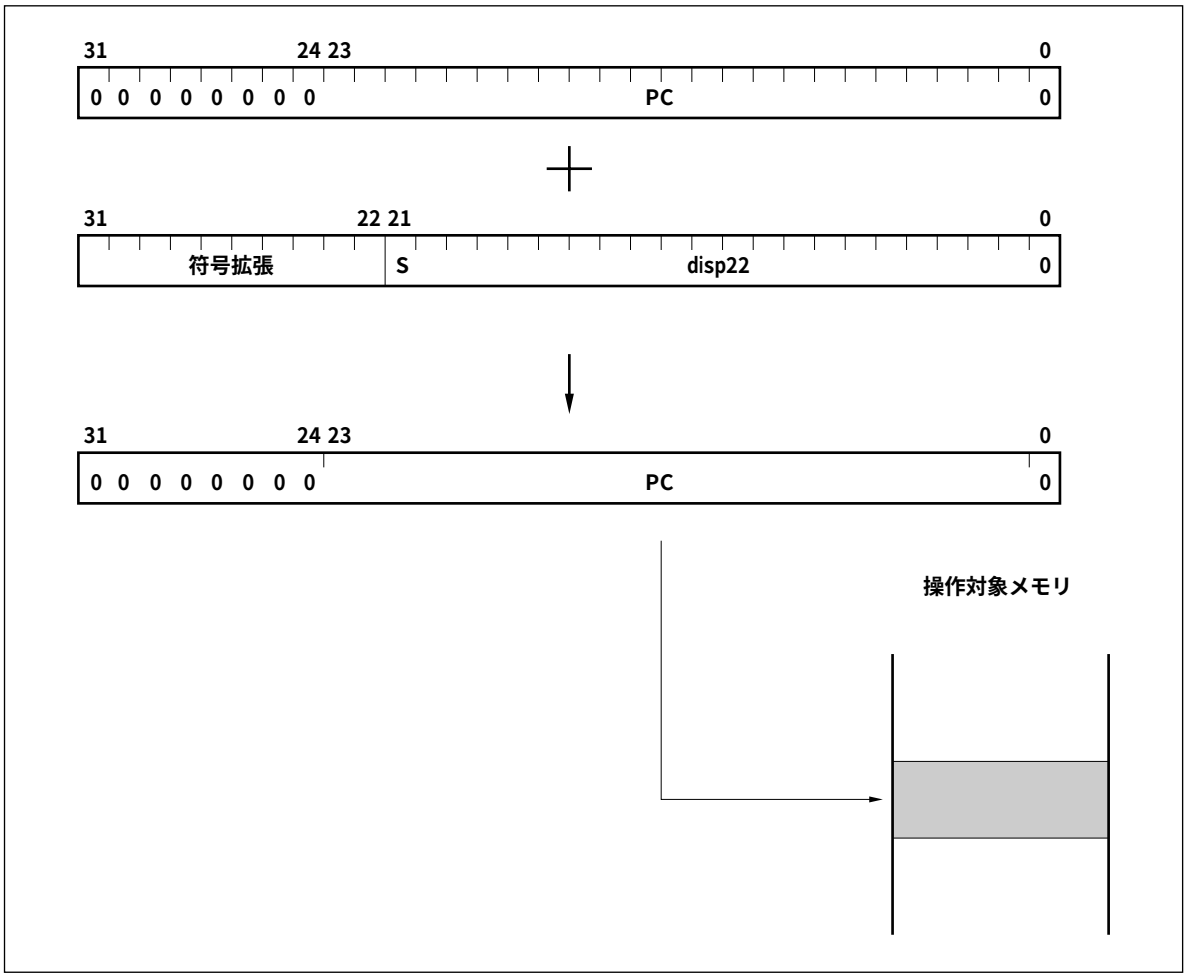
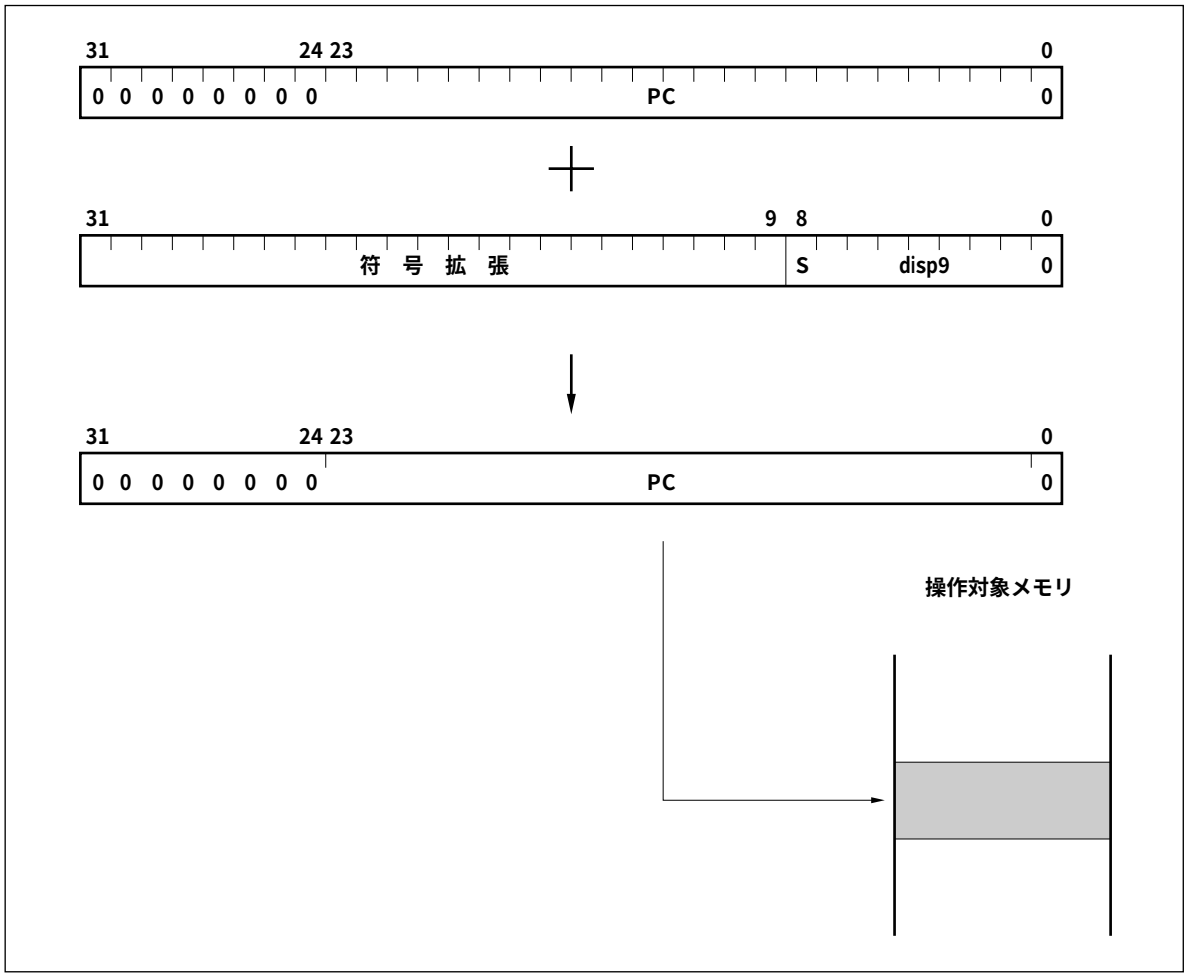


図４－３　レラティブ・アドレシング（Bcond disp9）

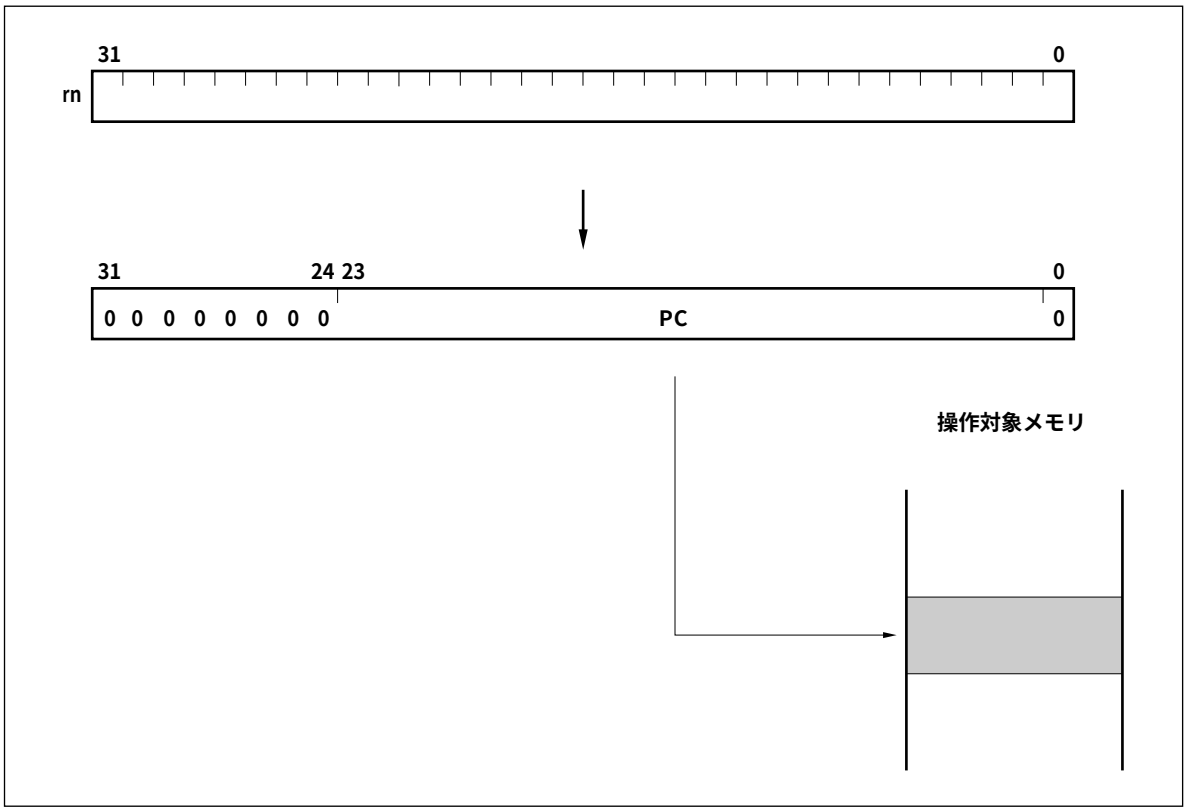


（２）レジスタ・アドレシング（レジスタ間接）

命令によって指定される汎用レジスタ（r0-r31）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４－４　レジスタ・アドレシング（JMP［reg1］）



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，次に示すいくつかの方法があります。

（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（システム・レジスタの場合あり）をオペランドとしてアクセスするアドレシングです。このアドレシングはオペランド形式が，reg1, reg2, またはregIDである命令が対象となります。

（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。このアドレシングはオペランド形式が，imm5，imm16，vector，またはccccである命令が対象となります。

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，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・イミーディエトとして命令コード中に割り当てられます。

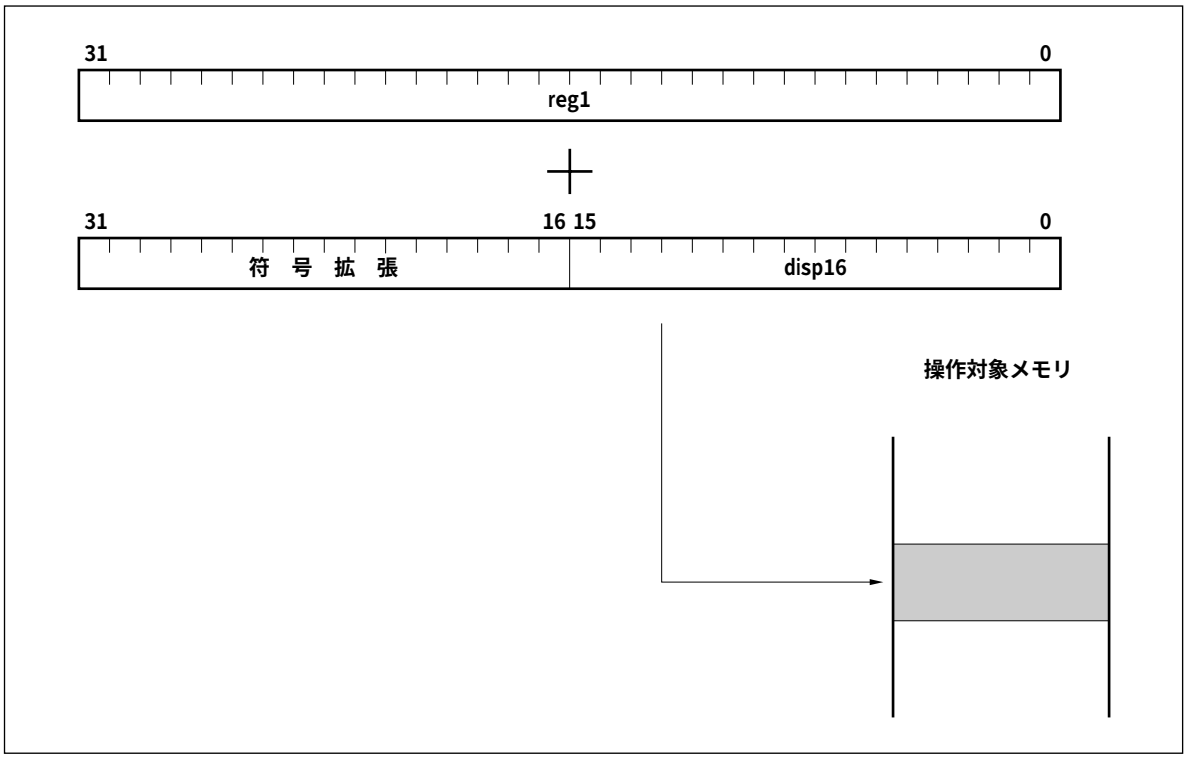
（３）ベースト・アドレシング

ベースト・アドレシングには，以下に示す２種類があります。

（ａ）タイプ１

命令語中のアドレシング指定コードで指定される汎用レジスタの内容と16ビット・ディスプレースメントとの和がオペランド・アドレスとなって，操作対象となるメモリをアドレスするアドレシングです。このアドレシングはオペランド形式が，disp16［reg1］である命令が対象となります。

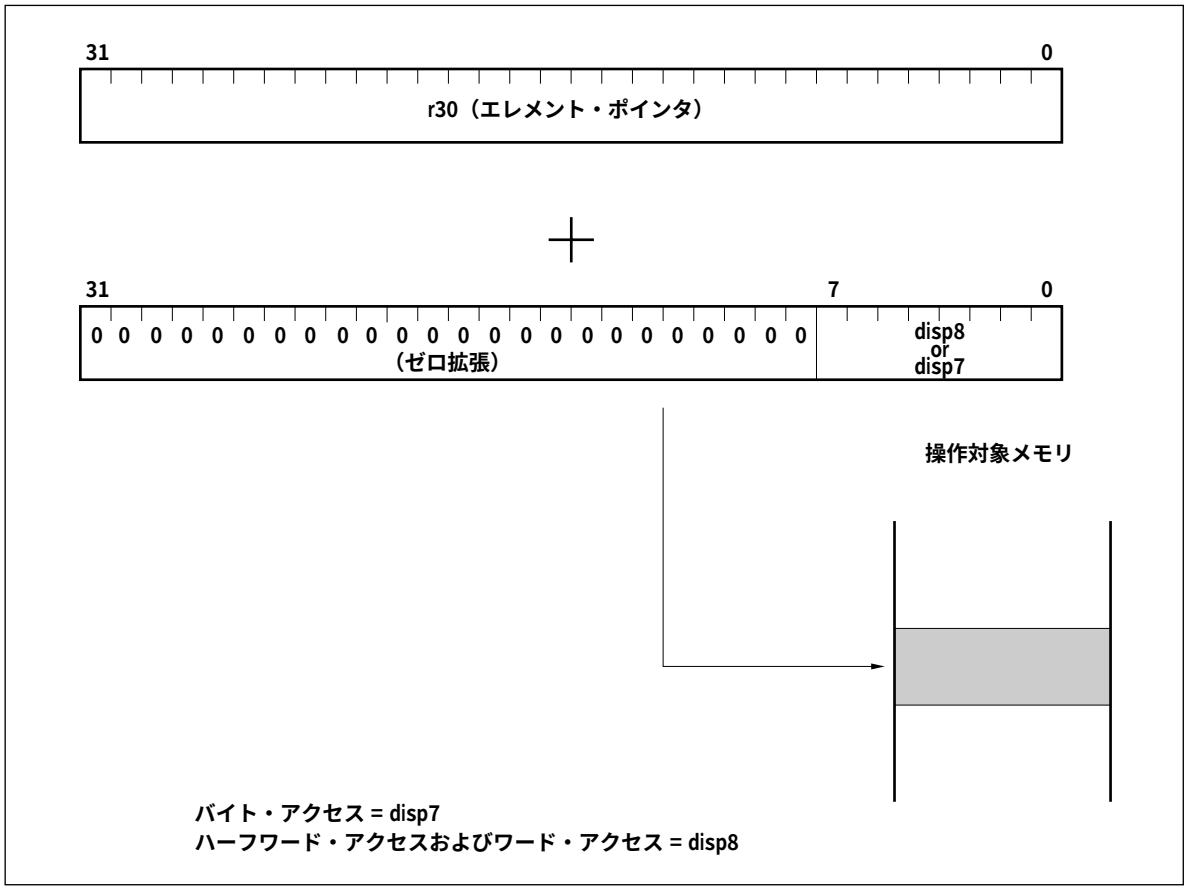
図４－５　ベースト・アドレシング（タイプ１）



（ｂ）タイプ２

32ビット・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・データとの和を，オペランド・アドレスとして操作対象となるメモリをアドレスするアドレシングです。このアドレシングはSLD命令およびSST命令が対象となります。

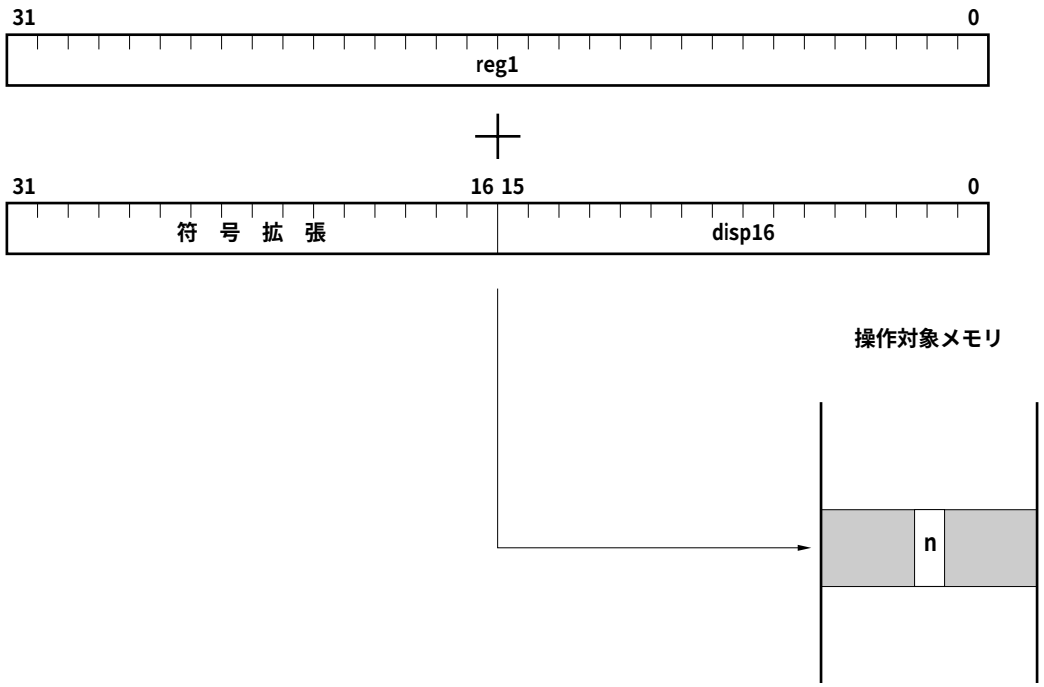
図４－６　ベースト・アドレシング（タイプ２）



（４）ビット・アドレシング

汎用レジスタ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との和をオペランド・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。このアドレシングはビット操作命令だけが対象となります。

図４－７　ビット・アドレシング



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

第５章　命　　令

５.１　命令フォーマット

V850ファミリの命令は16ビット・フォーマット，32ビット・フォーマットの２種類です。16ビット命令には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。

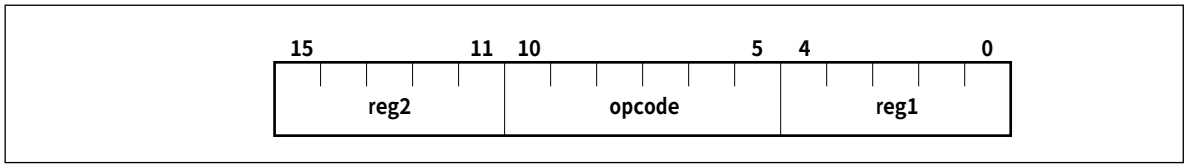
なお，一部の命令で未使用フィールド（RFU）がありますが，それらは将来の拡張用なので，０に固定してください。

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31を含む）　→上位アドレス側

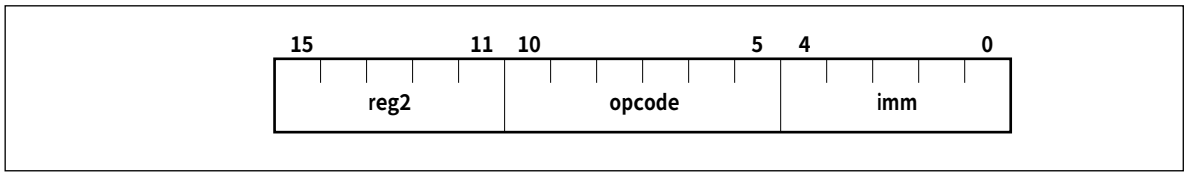
（１）reg-reg命令形式（Format Ⅰ）

６ビットのオペコード・フィールド，オペランド指定に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。



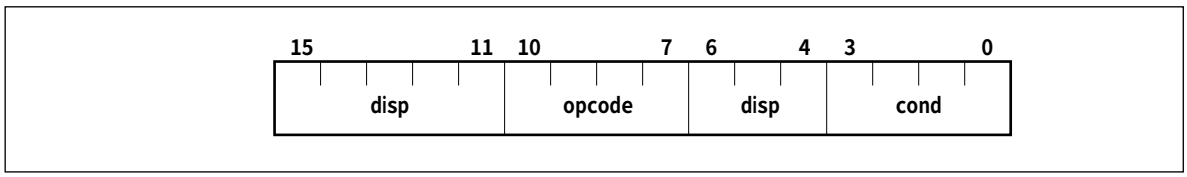
（２）imm-reg命令形式（Format Ⅱ）

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。



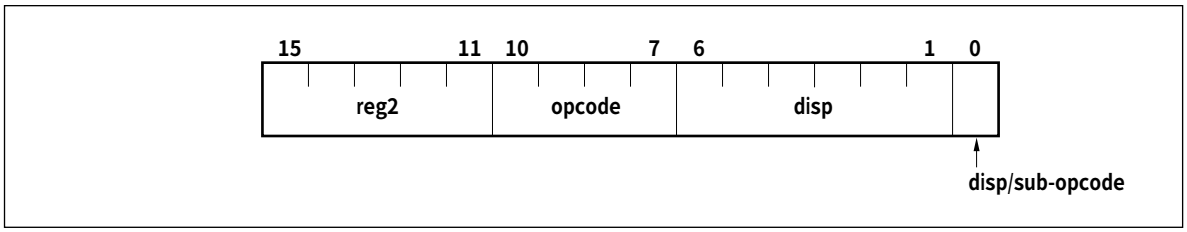
（３）条件分岐命令形式（Format Ⅲ）

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，８ビットのディスプレースメントを持つ16ビット長命令形式。



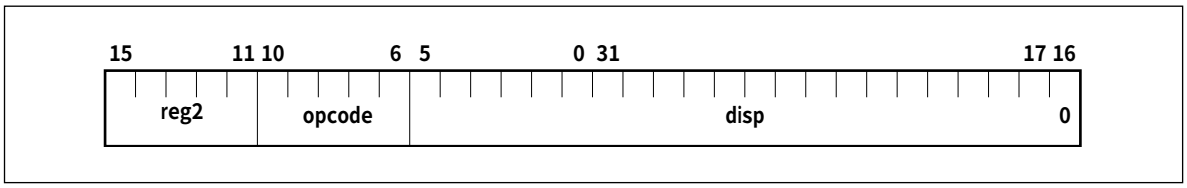
（4）ロード／ストア命令16ビット形式（Format　IV）

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・ディスプレースメント（または６ビット・ディスプレースメント＋１ビット・サブオペコード）を持つ16ビット長命令形式。



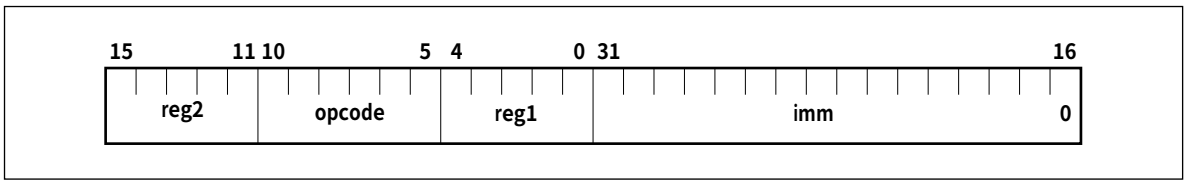
（5）ジャンプ命令形式（Format　V）

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・ディスプレースメントを持つ32ビット長命令形式。



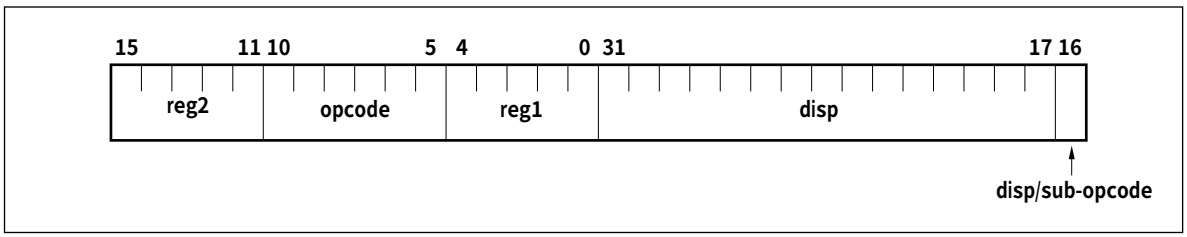
（6）３オペランド命令形式（Format　VI）

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・イミーディエト・フィールドを持つ32ビット長命令形式。



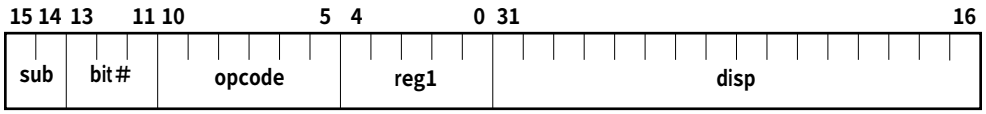
（7）ロード／ストア命令32ビット形式（Format　VII）

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・ディスプレースメント（または15ビット・ディスプレースメント＋１ビット・サブオペコード）を持つ32ビット長命令形式。



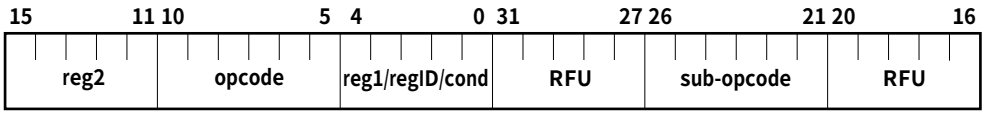
（8）ビット操作命令形式（Format　Ⅷ）

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，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。



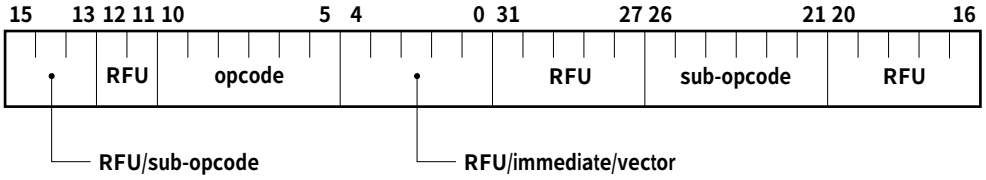
（9）拡張命令形式１（Format　IX）

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，２つの汎用レジスタ指定フィールド（１つはregIDまたは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を持つ32ビット長命令形式。



備考　RFU：予約フィールド（Reserved for Future Use）

5.2　命令の概要

ロード／ストア命令　… ロード／ストア命令によりメモリからレジスタへのデータ転送およびレジスタからメモリへのデータ転送ができます。

表5－1　ロード／ストア命令一覧

SLD
LD
SST
ST

算術演算命令　…… 算術演算命令により加減乗除算およびレジスタ間のデータ転送，データ比較ができます。

表5－2　算術演算命令一覧

MOV
MOVHI
MOVEA
ADD
ADDI
SUB
SUBR
MULH
MULHI
DIVH
CMP
SETF

飽和演算命令 …… 飽和演算により飽和加減算を行います。ただし，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。

表５－３　飽和演算命令一覧

SATADD
SATSUB
SATSUBI
SATSUBR

論理演算命令 …… 論理演算および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。バレル・シフタにより，１クロックで複数のビットをシフトすることができます。

表５－４　論理演算命令一覧

TST
OR
ORI
AND
ANDI
XOR
XORI
NOT
SHL
SHR
SAR

分岐命令　…… 無条件分岐命令とフラグの状態により制御を変更する条件分岐命令があります。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すことができます。

表5－5　分岐命令一覧

JMP
JR
JARL
BGT
BGE
BLT
BLE
BH
BNL
BL
BNH
BE
BNE
BV
BNV
BN
BP
BC
BNC
BZ
BNZ
BR
BSA

ビット操作命令 …… ビット操作命令によりメモリのビット・データに対して，論理演算ができます。指定されたビット以外は影響を受けません。

表５－６　ビット操作命令一覧

SET1
CLR1
NOT1
TST1

特殊命令 …… 前項までのカテゴリに含まれないその他の特殊な命令です。

表５－７　特殊命令一覧

LDSR
STSR
TRAP
RETI
HALT
DI
EI
NOP

5.3 命令セット

命令記述例

命令の意味（英語）

命令のニモニック

命令の意味（日本語）

命 令 形 式　命令の記述方法，オペランドを示します。オペランドの記述で使用する略号は次のとおりです。

略 号	意 味
reg1	汎用レジスタ（ソース・レジスタとして使用する）
reg2	汎用レジスタ（おもにデスティネーション・レジスタとして使用する。一部ソース・レジスタとしても使用する。）
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
disp×	×ビット・ディスプレースメント
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
ep	エレメント・ポインタ（r30）

オペレーション　命令の機能を示します。使用する略号は次のとおりです。

略　　号	意　　味
←	代入
GR []	汎用レジスタ
SR []	システム・レジスタ
zero-extend（n）	nを，ワード長までゼロ拡張する。
sign-extend（n）	nを，ワード長まで符号拡張する。
load-memory（a, b）	アドレスaから，サイズbのデータを読み出す。
store-memory（a, b, c）	アドレスaにデータbをサイズcで書き込む。
load-memory-bit（a, b）	アドレスaのビットbを読み出す。
store-memory-bit（a, b, c）	アドレスaのビットbにcを書き込む。
saturated（n）	nの飽和処理を行う。 nが計算の結果， n≧7FFFFFFFHとなった場合， 7FFFFFFFHとする。 n≦80000000Hとなった場合， 80000000Hとする。
result	結果をフラグに反映する。
Byte	バイト（８ビット）
Halfword	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
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

フォーマット　命令フォーマットを番号で示します。

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。

使用する略号は次のとおりです。

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
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
cccc	条件コードを示す４ビット・データ
bbb	ビット・ナンバ指定３ビット・データ

フ　ラ　グ　フラグの動きを示します。

CY　－　←　変化しないことを示します。

OV　0　←　０に変化することを示します。

S　　1　←　１に変化することを示します。

Z　　－

SAT　－

命　　令　命令の機能を示します。

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　本製品での注意事項について述べます。

命令一覧

ニモニック	機　　能	ニモニック	機　　能
	ロード／ストア命令		論理演算命令
SLD.B	Load Byte	TST	Test
SLD.H	Load Half-word	OR	Or
SLD.W	Load Word	ORI	Or Immediate
LD.B	Load Byte	AND	And
LD.H	Load Half-word	ANDI	And Immediate
LD.W	Load Word	XOR	Exclusive-Or
SST.B	Store Byte	XORI	Exclusive-Or Immediate
SST.H	Store Half-word	NOT	Not
SST.W	Store Word	SHL	Shift Logical Left
ST.B	Store Byte	SHR	Shift Logical Right
ST.H	Store Half-word	SAR	Shift Arithmetic Right
ST.W	Store Word		分岐命令
	算術演算命令	JMP	Jump
MOV	Move	JR	Jump Relative
MOVHI	Move High half-word	JARL	Jump and Register Link
MOVEA	Move Effective Address	Bcond	Branch on Condition Code
ADD	Add		ビット操作命令
ADDI	Add Immediate	SET1	Set Bit
SUB	Subtract	CLR1	Clear Bit
SUBR	Subtract Reverse	NOT1	Not Bit
MULH	Multiply Half-word	TST1	Test Bit
MULHI	Multiply Half-word Immediate		特殊命令
DIVH	Divide Half-word	LDSR	Load System Register
CMP	Compare	STSR	Store System Register
SETF	Set Flag Condition	TRAP	Trap
	飽和演算命令	RETI	Return from Trap or Interrupt
SATADD	Saturated Add	HALT	Halt
SATSUB	Saturated Subtract	DI	Disable Interrupt
SATSUBI	Saturated Subtract Immediate	EI	Enable Interrupt
SATSUBR	Saturated Subtract Reverse	NOP	No Operation

ほとんどのニモニックの名称は機能を示す英単語の頭文字をつなげたものです。

ADD

Add

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］

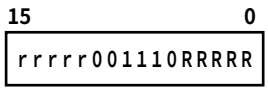
（2）GR［reg2］←GR［reg2］＋sign-extend（imm5）

フォーマット　（1）Format　I

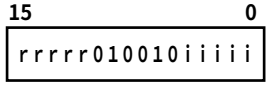
（2）Format　II

オ ペ コ ー ド

（1）



（2）



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　－

命 令　（1）ADD　Add Register

（2）ADD　Add Immediate（5-bit）

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。

ADDI

Add Immediate

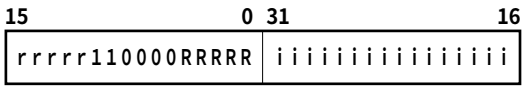
加算

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



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　－

命令　ADDI　Add immediate

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。

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

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　－

命　　令　AND　And

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。

ANDI

And Immediate

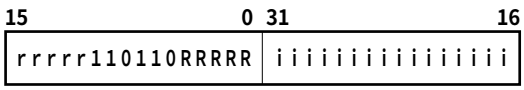
論理積

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



フラグ

CY　－

OV　0

S　0

Z　演算結果が0のとき1，そうでないとき0

SAT　－

命令　ANDI　And Immediate（16-bit）

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。

Bcond

Branch on Condition Code

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
d	d	d	d	d	1	0	1	1	d	d	d	c	c	c	c

ただし，ddddddddはdisp9の上位８ビットです。

フ ラ グ　CY　－
OV　－
S　－
Z　－
SAT　－

命　　令　Bcond　Branch on Condition Code with 9-bit displacement

説　　明　命令が指定する条件フラグをテストし，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，本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の時は，分岐先はこの命令自身になります。

表５－８　条件分岐命令一覧

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

注　　意　飽和演算命令の実行結果でSATフラグが１になった場合，符号付き整数の条件分岐（BGT, BGE, BLT, BLE）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PSWのSフラグは通常の演算とは異なりますので，OVフラグとの排他的論理和をとる分岐条件に意味がなくなります。

CLR1

Clear Bit

ビット・クリア

命 令 形 式　CLR1　bit#3, disp16［reg1］

オペレーション　adr←GR［reg1］＋sign-extend（disp16）
Zフラグ←Not（Load-memory-bit（adr, bit#3））
Store-memory-bit（adr, bit#3, 0）

フォーマット　Format Ⅷ

オ ペ コ ー ド

15	0	31	16
10bbb111110RRRRR		ddddddddddddddd	

フ ラ グ　CY　－
OV　－
S　－
Z　メモリdisp16［reg1］のビットNO.bit#3=0のとき1
　　メモリdisp16［reg1］のビットNO.bit#3=1のとき0
SAT　－

命　　令　CLR1　Clear Bit

説　　明　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をクリア（0）します。指定されたビット以外は影響を受けません。

補　　足　PSWのZフラグは本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。本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。

CMP

Compare

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］

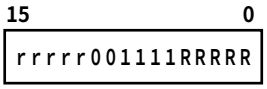
（2）result←GR［reg2］－sign-extend（imm5）

フォーマット　（1）Format　I

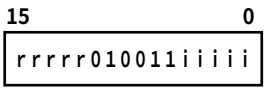
（2）Format　II

オペコード

（1）



（2）



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　－

命令　（1）CMP　Compare Register

（2）CMP　Compare Immediate（5-bit）

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，結果を条件フラグに示します。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，結果を条件フラグに示します。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。

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

命　　令　DI　Disable Interrupt

説　　明　PSW中のIDフラグをセット（1）し，本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　本命令の実行中は，割り込みのサンプリングをしません。本命令による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本命令実行中に行わないため，実際は本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，本命令の実行後も受け付けは禁止されません。

DIVH

Divide Half-word

（符号付き）除算

命令形式　DIVH　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
r	r	r	r	r	0	0	0	0	1	0	R	R	R	R	R

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　－

命令　DIVH　Divide Half-word

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　剰余は格納されません。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。

本命令実行中に割り込みが発生すると，除算を中断し，戻り番地を本命令の先頭アドレスとして割り込みを処理してから，除算を再実行します。中断した場合，汎用レジスタreg1, reg2は本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。

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

命　　令　EI　Enable Interrupt

説　　明　PSW中のIDフラグをリセット（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　本命令の実行中は，割り込みのサンプリングをしません。

HALT

Halt

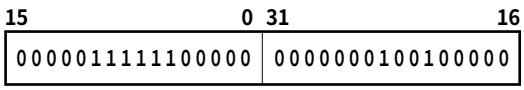
停止

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



フラグ　CY　－

OV　－

S　－

Z　－

SAT　－

命令　HALT　Halt

説明　CPUの動作クロックを停止させ，HALT状態に入ります。

補足　HALT状態は次の３つの要因によって解除されます。

- RESET入力
- NMI入力
- マスカブル割り込み（PSWのID=０のとき）

HALT状態であるときに，割り込みを受け付けた場合，EIPCまたはFEPCには，本命令の次の命令アドレスが格納されます。

JARL

Jump and Register Link

分岐およびレジスタ・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，ddddddddddddddddddddddはdisp22の上位21ビットです。

フ ラ グ　CY　－
OV　－
S　－
Z　－
SAT　－

命　　令　JARL　Jump and Register Link

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，本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。
本命令は，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。

Jump register

JMP

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　－

命　　令　JMP　Jump Register

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　本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。

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

フ ラ グ　CY　－

OV　－

S　－

Z　－

SAT　－

命　　令　JR　Jump Relative

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，本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。

LD

Load

ロード

命 令 形 式　（1）LD.B　disp16［reg1］, reg2

（2）LD.H　disp16［reg1］, reg2

（3）LD.W　disp16［reg1］, reg2

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

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

（2）adr←GR［reg1］＋sign-extend（disp16）

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

（3）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

（1）

15	0	31	16
rrrrr111000RRRRR		ddddddddddddddd	

（2）

15	0	31	16
rrrrr111001RRRRR		ddddddddddddddd0	

ただし，ddddddddddddddはdisp16の上位15ビットです。

（3）

15	0	31	16
rrrrr111001RRRRR		ddddddddddddddd1	

ただし，ddddddddddddddはdisp16の上位15ビットです。

フ ラ グ　CY　－

OV　－

S　－

Z　－

SAT　－

命　　令　（1）LD.B　Load Byte

（2）LD.H　Load Half-word

（3）LD.W　Load Word

説　　明　（１）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。

（２）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。この，32ビット・アドレスのビット０を０にマスクしたアドレスからハーフワード・データを読み出し，ワード長まで符号拡張し，汎用レジスタreg2に格納します。

（３）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。この，32ビット・アドレスのビット０およびビット１を０にマスクしたアドレスからワード・データを読み出し，汎用レジスタreg2に格納します。

注　　意　汎用レジスタreg１のデータとワード長まで符号拡張した16ビット・ディスプレースメントの加算結果は，アクセスするデータ形態（ハーフワード，ワード）に応じ，下位ビットを０にマスクしてアドレス生成されます。

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	

備考　本命令では，ニモニックの記述の都合上，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ので注意が必要です。

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）

命　　令　LDSR　Load to System Register

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。本命令によって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，実際は本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・レジスタに対して本命令を実行した場合の動作は保証しません。

MOV

Move

データの転送

命令形式　（1）MOV　reg1, reg2

（2）MOV　imm5, reg2

オペレーション　（1）GR［reg2］←GR［reg1］

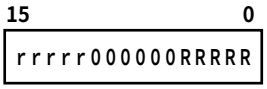
（2）GR［reg2］←sign-extend（imm5）

フォーマット　（1）Format　I

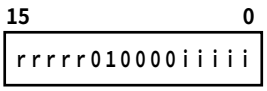
（2）Format　II

オペコード

（1）



（2）



フラグ　CY　－

OV　－

S　－

Z　－

SAT　－

命令　（1）MOV　Move Register

（2）MOV　Move Immediate（5-bit）

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。

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　－

命　　令　MOVEA　Move Effective Address

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。

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，本命令を使用します。

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　－

命令　MOVHI　Move High half-word

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。

補足　32ビット・アドレスの上位16ビットの生成に本命令を使用します。

MULH

Multiply Half-word

（符号付き）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
r	r	r	r	r	0	0	0	1	1	1	R	R	R	R	R

（2）

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

フ ラ グ　CY　－

OV　－

S　－

Z　－

SAT　－

命 令　（1）MULH　Multiply Half-word by Register

（2）MULH　Multiply Half-word by Immediate（5-bit）

説 明　（1）汎用レジスタreg2の下位ハーフワード・データに汎用レジスタreg1のハーフワード・データを乗算し，その結果を汎用レジスタreg2にワード・データとして格納します。汎用レジスタreg1は影響を受けません。

（2）汎用レジスタreg2の下位ハーフワード・データにハーフワード長まで符号拡張した5ビット・イミーディエトを乗算し，その結果を汎用レジスタreg2に格納します。

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

MULHI

Multiply Half-word Immediate

（符号付き）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　－

命令　MULHI　Multiply Half-word by immediate(16-bit)

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。

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。

NOP

No Operation

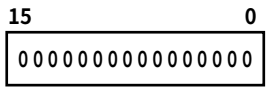
オペレーションなし

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



フ ラ グ　CY　－
OV　－
S　－
Z　－
SAT　－

命　　令　NOP　No Operation

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。

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

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　－

命令　NOT　Not

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。

NOT1

Not Bit

ビット・ノット

命 令 形 式　NOT1　bit#3, disp16［reg1］

オペレーション　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）

フォーマット　Format Ⅷ

オ ペ コ ー ド

15	0	31	16
01bbb111110RRRRR		ddddddddddddddd	

フ ラ グ　CY　－
OV　－
S　－
Z　メモリdisp16［reg1］のビットNO.bit#3=0のとき1
　　メモリdisp16［reg1］のビットNO.bit#3=1のとき0
SAT　－

命　　令　NOT1　Not Bit

説　　明　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を反転（0→1，1→0）します。指定されたビット以外は影響を受けません。

補　　足　PSWのZフラグは本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。本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。

OR

Or

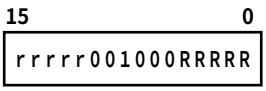
論理和

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



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　－

命　令　OR　Or

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。

ORI

Or Immediate

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　－

命　　令　OR　Or immediate（16-bit）

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。

RETI

Return from Trap or Interrupt

例外または割り込みルーチンから戻る

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

命令　RETI　Return from Trap or Interrupt

説明　システム・レジスタから，復帰PCとPSWを取り出し，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.NP，PSW.EPの各ビットを以下の状態にしておく必要があります。

●RETI命令によりノンマスカブル割り込み処理からの復帰時：

PSW.NP=1　かつ　PSW.EP=0

●RETI命令により例外処理からの復帰時：

PSW.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，本命令の後半のIDステージでは割り込みを受け付けません。

SAR

Shift Arithmetic Right

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］

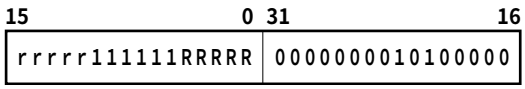
（2）GR［reg2］←GR［reg2］arithmetically shift right by zero-extend

フォーマット　（1）Format　**IX**

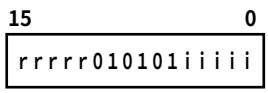
（2）Format　**II**

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　－

命令　（1）SAR　Shift Arithmetic Right by Register

（2）SAR　Shift Arithmetic Right by Immediate（5-bit）

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。

SATADD

Saturated Add

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］）

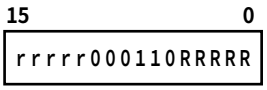
（2）GR［reg2］←saturated（GR［reg2］＋sigh-extend（imm5））

フォーマット　（1）Format　I

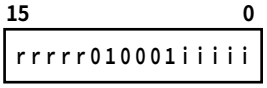
（2）Format　II

オ ペ コ ー ド

（1）



（2）



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

命 令

（1）SATADD　Saturated add register

（2）SATADD　Saturated add Immediate（5-bit）

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。

（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。

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ひとたび飽和すると，セット（1）され，以降の命令の演算結果が飽和しなくてもリセット（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動作を行います。

注　　意　SATフラグをリセット（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。

Saturated Subtract

SATSUB

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

命 令　SATSUB　Saturated Subtract

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。

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ひとたび飽和すると，セット（１）され，以降の命令の演算結果が飽和しなくてもリセット（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動作を行います。

注 意　SATフラグをリセット（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。

Saturated Subtract Immediate

SATSUBI

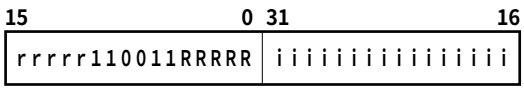
飽和減算

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



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

命令　SATSUBI　Saturated Subtract Immediate

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。

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ひとたび飽和すると，セット（１）され，以降の命令の演算結果が飽和しなくてもリセット（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動作を行います。

注意　SATフラグをリセット（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。

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

命　　令　SATSUBR　Saturated Subtract Reverse

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。

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ひとたび飽和すると，セット（１）され，以降の命令の演算結果が飽和しなくてもリセット（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動作を行います。

注　　意　SATフラグをリセット（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。

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　－

命令　SETF　Set Flag Condition

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５－９に示す条件コードを指定します。

補足　本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5－9　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
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

SET1

Set Bit

ビット・セット

命 令 形 式　SET1　bit#3, disp16［reg1］

オペレーション　adr←GR［reg1］＋sign-extend（disp16）
Zフラグ←Not（Load-memory-bit（adr,bit#3））
Store-memory-bit（adr, bit#3, 1）

フォーマット　Format Ⅷ

オ ペ コ ー ド

15	0	31	16
00bbb111110RRRRR		ddddddddddddddd	

フ ラ グ　CY　－
OV　－
S　－
Z　メモリdisp16［reg1］のビットNO.bit#3=0のとき1
メモリdisp16［reg1］のビットNO.bit#3=1のとき0
SAT　－

命　　令　SET1　Set Bit

説　　明　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をセット（1）します。指定されたビット以外は影響を受けません。

補　　足　PSWのZフラグは本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。本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。

SHL

Shift Logical Left

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］

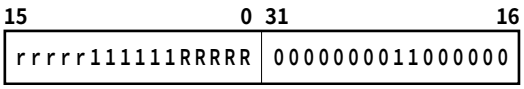
（2）GR［reg2］←GR［reg2］logically shift left by　zero-extend（imm5）

フォーマット　（1）Format　**IX**

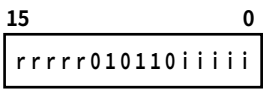
（2）Format　**II**

オ ペ コ ー ド

（1）



（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　－

命 令　（1）SHL　Shift Logical Left by Register

（2）SHL　Shift Logical Left by Immediate（5-bit）

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。

SHR

Shift Logical Right

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］

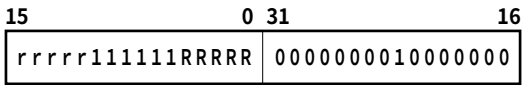
（2）GR［reg2］←GR［reg2］logically shift right by　zero-extend（imm5）

フォーマット　（1）Format　**IX**

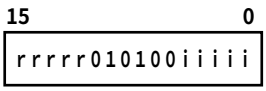
（2）Format　**II**

オペコード

（1）



（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　－

命令　（1）SHR　Shift Logical Right by Register

（2）SHR　Shift Logical Right by Immediate（5-bit）

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。

SLD

Load

ロード

命 令 形 式　（1）SLD.B　disp7［ep］, reg2

（2）SLD.H　disp8［ep］, reg2

（3）SLD.W　disp8［ep］, reg2

オペレーション　（1）adr←ep＋zero-extend（disp7）

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

（2）adr←ep＋zero-extend（disp8）

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

（3）adr←ep＋zero-extend（disp8）

GR［reg2］←Load-memory（adr,Word）

フォーマット　Format Ⅳ

オ ペ コ ー ド

（1）

15															0
r	r	r	r	r	0	1	1	0	d	d	d	d	d	d	d

（2）

15															0
r	r	r	r	r	1	0	0	0	d	d	d	d	d	d	d

ただし，dddddddはdisp8の上位７ビットです。

（3）

15															0
r	r	r	r	r	1	0	1	0	d	d	d	d	d	d	0

ただし，ddddddはdisp8の上位６ビットです。

フ ラ グ　CY　－

OV　－

S　－

Z　－

SAT　－

命　　令　（1）SLD.B　Short format Load Byte

（2）SLD.H　Short format Load Half-word

（3）SLD.W　Short format Load Word

説　　明　（1）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。

（2）エレメント・ポインタと，ワード長までゼロ拡張した8ビット・ディスプレースメントを加算して32ビット・アドレスを生成します。この，32ビット・アドレスのビット0を0にマスクしたアドレスからハーフワード・データを読み出し，ワード長まで符号拡張し，reg2に格納します。

（3）エレメント・ポインタと，ワード長までゼロ拡張した8ビット・ディスプレースメントを加算して32ビット・アドレスを生成します。この，32ビット・アドレスのビット0およびビット1を0にマスクしたアドレスからワード・データを読み出し，reg2に格納します。

注　　意　エレメント・ポインタとワード長までゼロ拡張した8ビット・ディスプレースメントの加算結果は，アクセスするデータ形態（ハーフワード，ワード）に応じ，下位ビットを0にマスクしてアドレス生成されます。

SST

Store

ストア

命 令 形 式　（1）SST.B　reg2, disp7［ep］

（2）SST.H　reg2, disp8［ep］

（3）SST.W　reg2, disp8［ep］

オペレーション　（1）adr←ep＋zero-extend（disp7）

Store-memory（adr,GR［reg2］,Byte）

（2）adr←ep＋zero-extend（disp8）

Store-memory（adr,GR［reg2］, Halfword）

（3）adr←ep＋zero-extend（disp8）

Store-memory（adr,GR［reg2］, Word）

フォーマット　Format Ⅳ

オ ペ コ ー ド

（1）

15															0
r	r	r	r	r	0	1	1	1	d	d	d	d	d	d	d

（2）

15															0
r	r	r	r	r	1	0	0	1	d	d	d	d	d	d	d

ただし，dddddddはdisp8の上位７ビットです。

（3）

15															0
r	r	r	r	r	1	0	1	0	d	d	d	d	d	d	1

ただし，ddddddはdisp8の上位６ビットです。

フ ラ グ　CY　－

OV　－

S　－

Z　－

SAT　－

命　　令　（１）SST.B　Short format Store Byte

（２）SST.H　Short format Store Half-word

（３）SST.W　Short format Store Word

説　　明　（１）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。

（２）エレメント・ポインタと，ワード長までゼロ拡張した８ビット・ディスプレースメントを加算して32ビット・アドレスを生成します。reg2の下位ハーフワード・データを生成した32ビット・アドレスのビット０を０にマスクしたアドレスに格納します。

（３）エレメント・ポインタと，ワード長までゼロ拡張した８ビット・ディスプレースメントを加算して32ビット・アドレスを生成します。reg2のワード・データを生成した32ビット・アドレスのビット０およびビット１を０にマスクしたアドレスに格納します。

注　　意　エレメント・ポインタとワード長までゼロ拡張した８ビット・ディスプレースメントの加算結果は，アクセスするデータ形態（ハーフワード，ワード）に応じ，下位ビットを０にマスクしてアドレス生成されます。

ST

Store

ストア

命 令 形 式　（1）ST.B　reg2, disp16［reg1］

（2）ST.H　reg2, disp16［reg1］

（3）ST.W　reg2, disp16［reg1］

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

Store-memory（adr,GR［reg2］, Byte）

（2）adr←GR［reg1］＋sign-extend（disp16）

Store-memory（adr,GR［reg2］, Halfword）

（3）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

（1）

15	0	31	16
rrrrr111010RRRRR		ddddddddddddddd	

（2）

15	0	31	16
rrrrr111011RRRRR		ddddddddddddddd0	

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

（3）

15	0	31	16
rrrrr111011RRRRR		ddddddddddddddd1	

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

フ ラ グ　CY　－

OV　－

S　－

Z　－

SAT　－

命　　令　（1）ST.B　Store Byte

（2）ST.H　Store Half-word

（3）ST.W　Store Word

説　　明　（1）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。

（2）汎用レジスタreg1のデータと，ワード長まで符号拡張した16ビット・ディスプレースメントを加算して32ビット・アドレスを生成します。汎用レジスタreg2の下位ハーフワード・データを生成した32ビット・アドレスのビット０を０にマスクしたアドレスに格納します。ハーフワード・データのアクセスは，自動的にアラインメントされ，32ビット・アドレスのビット０を０にマスクします。

（3）汎用レジスタreg1のデータと，ワード長まで符号拡張した16ビット・ディスプレースメントを加算して32ビット・アドレスを生成します。汎用レジスタreg2のワード・データを生成した32ビット・アドレスのビット０およびビット１を０にマスクしたアドレスに格納します。ワード・データのアクセスは，自動的にアラインメントされ，32ビット・アドレスのビット０およびビット１を０にマスクします。

注　　意　汎用レジスタreg1のデータとワード長まで符号拡張した16ビット・ディスプレースメントの加算結果は，アクセスするデータ形態（ハーフワード，ワード）に応じ，下位ビットを０にマスクしてアドレス生成されます。

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　－

命　　令　STSR　Store Contents of System Register

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・レジスタに対して本命令を実行した場合の動作は保証しません。

SUB

Subtract

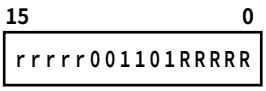
減算

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



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　－

命　　令　SUB　Subtract

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。

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

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　－

命 令　SUBR　Subtract Reverse

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。

Trap

TRAP

ソフトウエア・トラップ

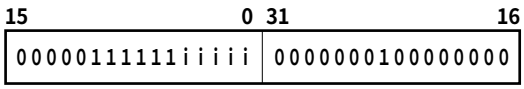
命令形式　TRAP　vector

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



フラグ

```
CY　－
OV　－
S　－
Z　－
SAT　－
```

命令　TRAP　Trap

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）を行ったあと，vectorで指定されるトラップ・ベクタ（0-31）に対応するトラップ・ハンドラのアドレスにジャンプし，例外処理を開始します。
条件フラグは影響を受けません。
なお，復帰PCとは，TRAP命令の次の命令アドレスになります。

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

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　－

命　　令　TST　Test

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，フラグのみが影響を受けます。汎用レジスタreg1，reg2は影響を受けません。

TST1

Test Bit

ビット・テスト

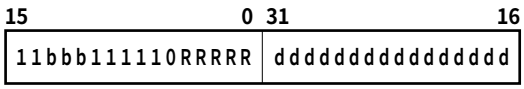
命 令 形 式　TST1　bit#3, disp16［reg1］

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

Zフラグ←Not（Load-memory-bit（adr,bit#3））

フォーマット　Format Ⅷ

オ ペ コ ー ド



フ ラ グ　CY　－

OV　－

S　－

Z　メモリdisp16［reg1］のビットNO.bit#3=0のとき1

メモリdisp16［reg1］のビットNO.bit#3=1のとき0

SAT　－

命　　令　TST1　Test Bit

説　　明　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・ナンバーで指定されるビットが0ならばZフラグをセット（1）し，1ならばリセット（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。

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　－

命　　令　XOR　Exclusive Or

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。

XORI

Exclusive Or Immediate

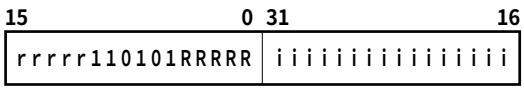
排他的論理和

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



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　－

命令　XORI　Exclusive Or Immediate（16-bit）

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

５．４　命令実行クロック数

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５－10に命令実行クロック数一覧を示します。

表５－10　命令実行クロック数一覧

命令群	ニモニック	オペランド	バイト	実行クロック i－r－l
ロード／ストア命令	SLD.B	disp7［ep］, r	2	1－1－2
	SLD.H	disp8［ep］, r	2	1－1－2
	SLD.W	disp8［ep］, r	2	1－1－2
	SST.B	r, disp7［ep］	2	1－1－1
	SST.H	r, disp8［ep］	2	1－1－1
	SST.W	r, disp8［ep］	2	1－1－1
	LD.B	disp16［R］, r	4	1－1－2
	LD.H	disp16［R］, r	4	1－1－2
	LD.W	disp16［R］, r	4	1－1－2
	ST.B	r, disp16［R］	4	1－1－1
	ST.H	r, disp16［R］	4	1－1－1
	ST.W	r, disp16［R］	4	1－1－1
算術演算命令	MOV	R, r	2	1－1－1
	MOV	imm5, r	2	1－1－1
	MOVEA	imm16, R, r	4	1－1－1
	MOVHI	imm16, R, r	4	1－1－1
	DIVH	R, r	2	36－36－36
	MULH	R, r	2	1－1－2
	MULH	imm5, r	2	1－1－2
	MULHI	imm16, R, r	4	1－1－2
	ADD	R, r	2	1－1－1
	ADD	imm5, r	2	1－1－1
	ADDI	imm16, R, r	4	1－1－1
	CMP	R, r	2	1－1－1
	CMP	imm5, r	2	1－1－1
	SUBR	R, r	2	1－1－1
	SUB	R, r	2	1－1－1
	SETF	cccc, r	4	1－1－1
飽和演算命令	SATSUBR	R, r	2	1－1－1
	SATSUB	R, r	2	1－1－1
	SATADD	R, r	2	1－1－1
	SATADD	imm5, r	2	1－1－1
	SATSUBI	imm16, R, r	4	1－1－1

命令群	ニモニック	オペランド		バイト	実行クロック i－r－l
論理演算命令	NOT	R, r		2	1－1－1
	OR	R, r		2	1－1－1
	XOR	R, r		2	1－1－1
	AND	R, r		2	1－1－1
	TST	R, r		2	1－1－1
	SHR	imm5, r		2	1－1－1
	SAR	imm5, r		2	1－1－1
	SHL	imm5, r		2	1－1－1
	ORI	imm16, R, r		4	1－1－1
	XORI	imm16, R, r		4	1－1－1
	ANDI	imm16, R, r		4	1－1－1
	SHR	R, r		4	1－1－1
	SAR	R, r		4	1－1－1
	SHL	R, r		4	1－1－1
分岐命令	JMP	［R］		2	3－3－3
	JR	disp22		4	3－3－3
	JARL	disp22, r		4	3－3－3
	Bcond	disp9	条件成立時	2	3－3－3
			条件不成立時	2	1－1－1
ビット操作命令	SET1	bit#3, disp16［R］		4	4－4－4
	CLR1	bit#3, disp16［R］		4	4－4－4
	NOT1	bit#3, disp16［R］		4	4－4－4
	TST1	bit#3, disp16［R］		4	3－3－3
特殊命令	LDSR	R, SR		4	1－1－注
	STSR	SR, r		4	1－1－1
	NOP	－		2	1－1－1
	DI	－		4	1－1－1
	EI	－		4	1－1－1
	TRAP	vector		4	4－4－4
	HALT	－		4	1－1－1
	RETI	－		4	4－4－4
未定義命令コード・トラップ				4	4－4－4

注　EIPC, FEPCアクセス時：３
EIPSW, FEPSW, PSWアクセス時：１

オペランド

略　　号	意　　味
R：reg1	汎用レジスタ（ソース・レジスタとして使用する）
r：reg2	汎用レジスタ（おもにデスティネーション・レジスタとして使用する）
SR：System Register	システム・レジスタ
imm×：immediate	×ビット・イミーディエト
disp×：displacement	×ビット・ディスプレースメント
bit#3：bit number	ビット・ナンバ指定用３ビット・データ
ep：Element Pointer	エレメント・ポインタ
B：Byte	バイト（８ビット）
H：Halfword	ハーフワード（16ビット）
W：Word	ワード（32ビット）
cccc：condisions	条件コードを示す４ビット・データ
vector	トラップ・ベクタ（00H-1FH）を指定する５ビット・データ

実行クロック

略　　号	意　　味
i：issue	命令実行直後にほかの命令を実行する場合
r：repeat	命令実行直後に同一命令を繰り返す場合
★ l：latency	命令実行結果を直後の命令で引用する場合

第６章　割り込みと例外

割り込みは，プログラムの実行とは別に独立に発生する事象で，マスカブル割り込みとノンマスカブル割り込みがあります。例外は，プログラムの実行に依存して発生する事象です。割り込みと例外に制御の流れとしての
★ 大きな違いはありません。

V850ファミリでは，オンチップの周辺ハードウエアおよび外部からの各種割り込み要求を処理できます。さらに，命令による例外処理の起動（TRAP命令）や，例外事象の発生による例外処理の起動（例外トラップ）が可能です。

V850ファミリでは，次に示すような割り込みと例外があります。割り込み／例外が発生した場合には，各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。各ハンドラでは割り込み要因レジスタ（ECR）を解析し，適切な割り込み／例外処理を行います。復帰PC，PSWは，状態退避レジスタ（EIPC, EIPSW/FEPC, FEPSW）に書き込まれます。

割り込み／例外処理からの復帰は，RETI命令により行われます。

状態退避レジスタから復帰PC，PSWを取り出し，復帰PCへ制御を移します。

○割り込み／例外処理の種類

V850ファミリの割り込み／例外処理は，次の４種類に分類されます。

- ノンマスカブル割り込み
- マスカブル割り込み
- ソフトウエア例外
- 例外トラップ

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				
NMI	NMI入力	割り込み	0010H	00000010H	next PC注２
マスカブル割り込み	注１	割り込み	注１	注１	next PC注２
TRAP0n（n=0-FH）	TRAP命令	例外	004nH	00000040H	next PC
TRAP1n（n=0-FH）	TRAP命令	例外	005nH	00000050H	next PC
ILGOP	不正命令コード	例外	0060H	00000060H	next PC注３

注１．マスカブル割り込みの種類ごとに異なります。

２．DIVH（除算）命令実行時に割り込みを受け付けた場合は，カレントの命令（DIVH）のPC値となります。

３．不正命令コード例外時の不正命令の実行アドレスは，（復帰PC－4）で求められます。

復帰PCとは，割り込み／例外処理起動時に，EIPCまたはFEPCにセーブされるPCのことです。next PCとは，割り込み／例外処理後に処理を開始するPCのことです。

マスカブル割り込みは，V850ファミリの各デバイスのINTC（割り込みコントローラ）により，ユーザが使用可能な割り込みで，割り込み／例外要因や例外コードは各デバイスにより異なります。

６.１　割り込み処理

６.１.１　マスカブル割り込み

プログラム・ステータス・ワード（PSW）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。

INT入力によりマスカブル割り込みが発生した場合，プロセッサ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ビットをセットし，EPビットをクリアします。

（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。

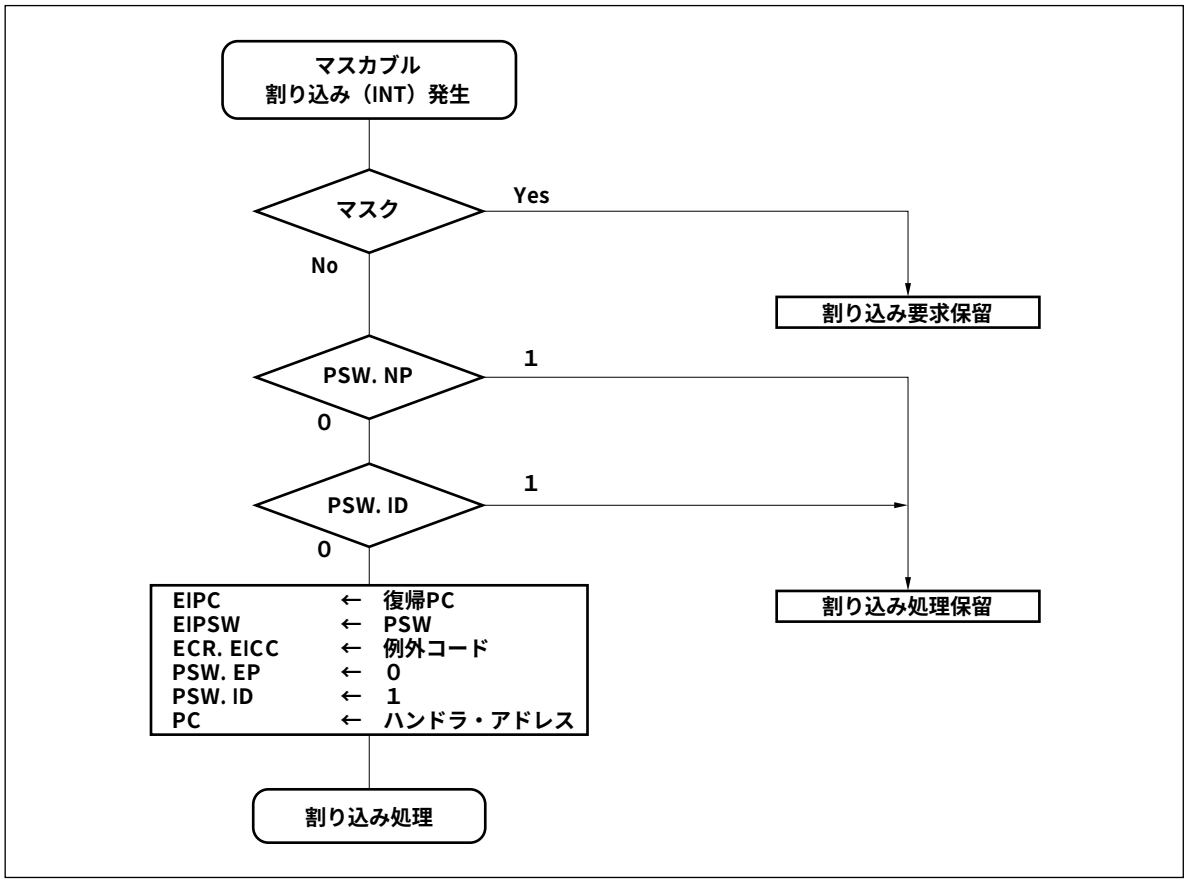
状態退避レジスタにはEIPC，EIPSWを使用します。なお，INTCにおいてマスクされているINT入力，ほかの割り込み処理中（PSWのNPビットが１ or PSWのIDビットが１のとき）に発生したINT入力は，INTC内部で保留されます。この場合，マスクを解除し，RETI命令，LDSR命令を使用してPSWのNPビットとPSWのIDビットを０にすると，保留していたINT入力により新たなマスカブル割り込み処理が開始されます。

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。なお，EIPCのビット31-ビット24と，EIPSWのビット31-ビット８は０に固

定されています。

マスカブル割り込みの処理形態を図６－１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。V850ファミリのノンマスカブル割り込みは，NMI入力により発生します。

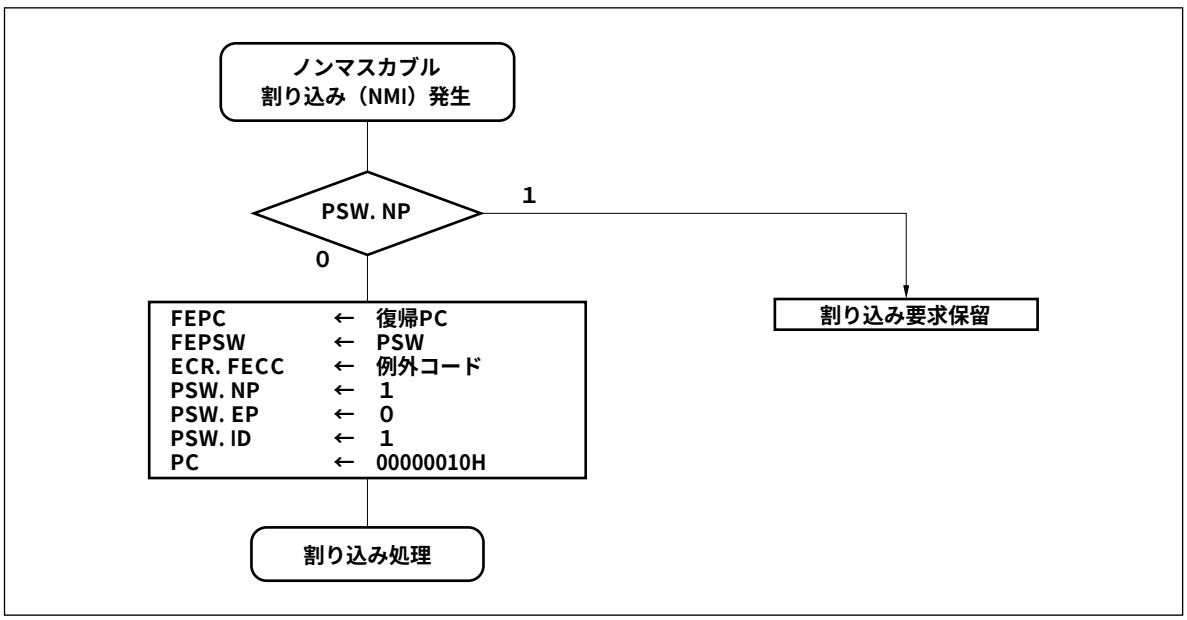
NMI入力により，ノンマスカブル割り込みが発生した場合は，プロセッサは次の処理を行いハンドラ・ルーチンへ制御を移します。

（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）に例外コードを書き込みます。
（４）PSWのNP，IDビットをセットし，EPビットをクリアします。
（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（00000010H）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。なお，ノンマスカブル割り込み処理中（PSWのNPビットが１）に発生したノンマスカブル割り込み要求は，INTC内部で保留されます。この場合，RETI命令およびLDSR命令を使用して，PSWのNPビットを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。

ノンマスカブル割り込みの処理形態を図６－２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，CPUの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。

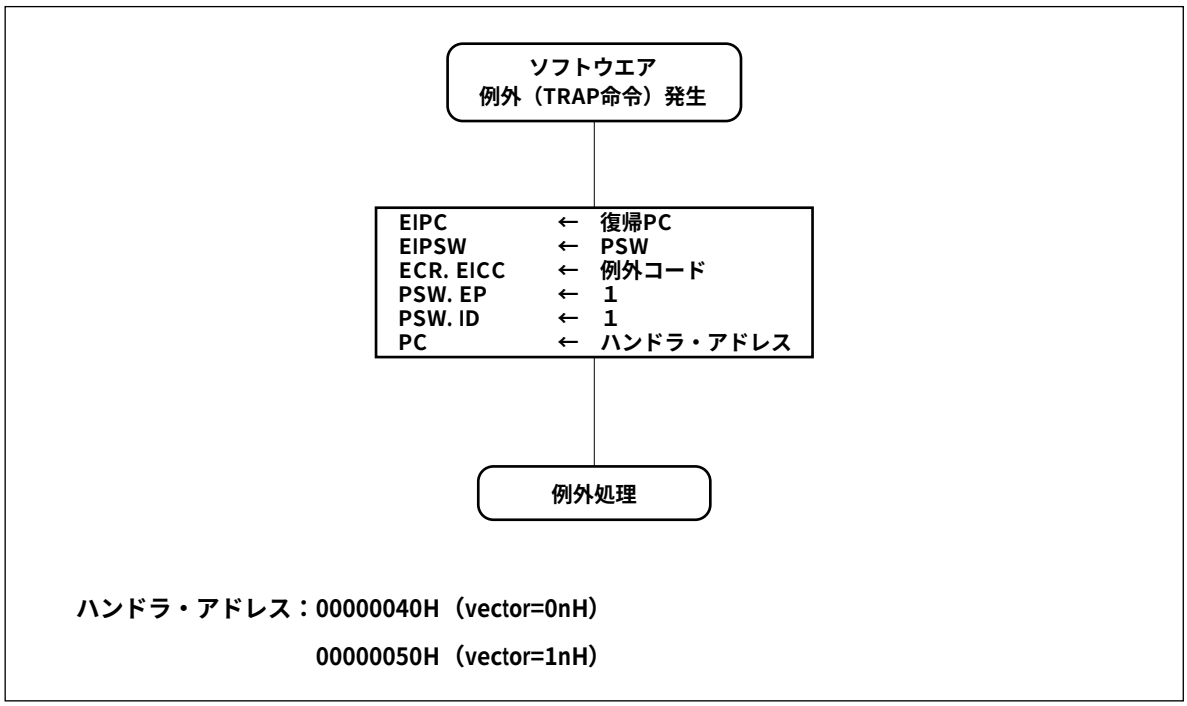
（３）ECR（割り込み要因）の下位16ビット（EICC）に例外コードを書き込みます。

（４）PSWのEP，IDビットをセットします。

（５）PCにソフトウエア例外に対するハンドラ・アドレス（00000040H or 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図６－３に示します。

図６－３　ソフトウエア例外の処理形態

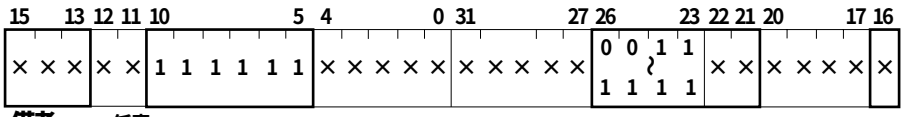


６.２.２　例外トラップ

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割り込みです。V850ファミリの例外トラップは，不正命令コード・トラップ(ILGOP：ILleGal OPcode trap)により発生します。

不正命令は，命令コードのオペコード（ビット５-ビット10）が111111Bで，サブオペコード（ビット23-ビット26）が0011B-1111B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，不正命令コード・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。

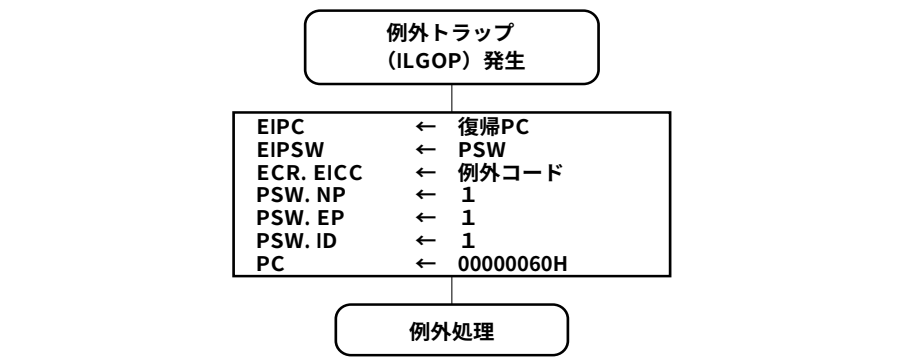
（３）ECRの下位16ビット（EICC）に例外コードを書き込みます。

★　（４）PSWのNP，EP，IDビットをセットします。

（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図６－５に示します。

図６－５　例外トラップの処理形態



★

例外トラップ発生時の，不正命令の実行アドレスは“復帰PC－４”で求められます。

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。

６.３　割り込み／例外からの復帰

割り込み／例外処理からの復帰は，すべてRETI命令により行われます。

RETI命令の実行により，プロセッサ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ビットが０かつPSWのNPビットが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。

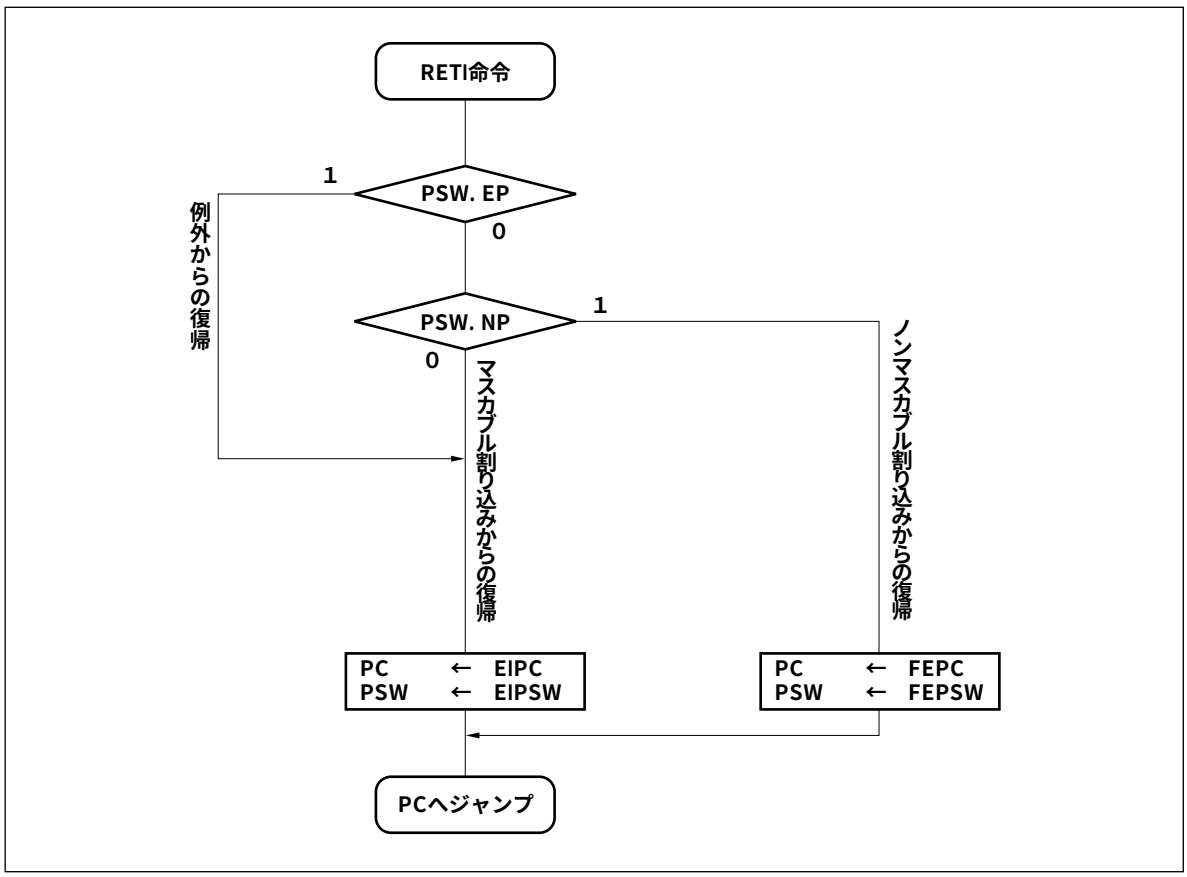
例外処理またはノンマスカブル割り込み処理からの復帰時は，PC，PSWを正常にリストアするために，RETI命令の直前で，LDSR命令を使用し，PSWのNPビット，EPビットの各ビットを以下の状態にしておく必要があります。

ノンマスカブル割り込み処理からの復帰時…PSWのNPビット=１，EPビット=０

例外処理からの復帰時……PSWのEPビット=１

割り込み／例外からの復帰の処理形態を図６－６に示します。

図６－６　割り込み／例外からの復帰の処理形態



第７章　リセット

$\overline{\text{RESET}}$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，オンチップの各ハードウエアは初期状態にイニシャライズされます。

７.１　イニシャライズ

$\overline{\text{RESET}}$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，各ハードウエア関係のレジスタは，表７－１に示すような状態になります。$\overline{\text{RESET}}$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。各種レジスタの内容は，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

ハードウエア（略号）		リセット後の状態
プログラム・カウンタ	PC	00000000H
割り込み時状態退避レジスタ	EIPC	不定
	EIPSW	不定
NMI時状態退避レジスタ	FEPC	不定
	FEPSW	不定
割り込み要因レジスタ　ECR	FECC	0000H
	EICC	0000H
プログラム・ステータス・ワード	PSW	00000020H
汎用レジスタ	r0	00000000H固定
	r1-r31	不定

７.２　起　　動

V850ファミリは，リセットにより00000000Hからプログラムの実行を開始します。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を使用する場合は，プログラム・ステータス・ワード（PSW）のIDビットを０に設定してください。

第８章　パイプライン

V850ファミリは，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。

プロセッサは５ステージのパイプライン構造になっております。

パイプラインの各ステージを次に示します。

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・データを作成，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をアクセスします。

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。

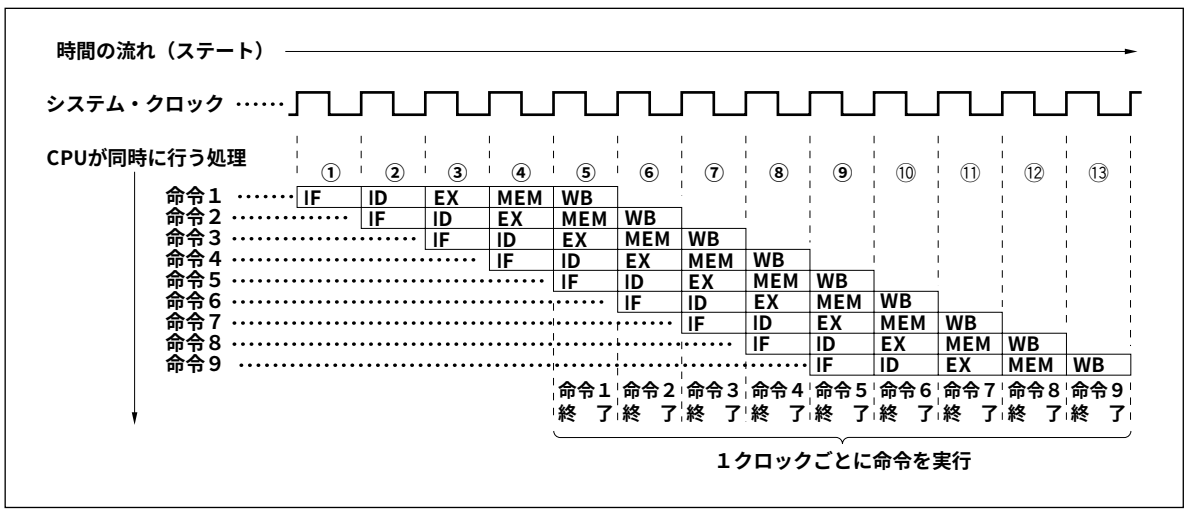
８.１　動作概要

V850ファミリの命令実行手順は，フェッチからライトバックまでの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



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，命令n＋１のメモリ・アクセス，命令n＋２の実行，命令n＋３のデコード，命令n＋４のフェッチが同時に行われます。標準的な命令では，フェッチからライトバックまで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。

8.2　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。

命令フェッチ（IFステージ）は内蔵ROM/PROMを，メモリ・アクセス（MEMステージ）は内蔵RAMを対象にしています。この場合，IFステージ，MEMステージは１クロックで処理されます。それ以外の場合は，所定のアクセス時間と，場合によってはバスの待ち合わせ時間がかかります。アクセス時間は次のとおりです。

表８－１　アクセス時間（クロック数）

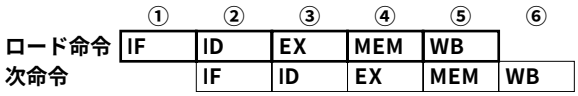
リソース（バス幅）／ステージ	内蔵ROM/PROM（32ビット）	内蔵RAM（32ビット）	内蔵周辺I/O（8/16ビット）	外部メモリ（16ビット）
命令フェッチ（IF）	1	3	不可	3＋n
メモリ・アクセス（MEM）	3	1	3＋n	3＋n

備考　n：ウエイト数

8.2.1　ロード命令

[対 象 の 命 令]　LD, SLD

[パイプライン]

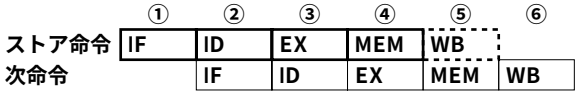


[説　　明]　パイプラインはIF, ID, EX, MEM, WBの５ステージです。ロード命令の直後に，実行結果を使用する命令を配置すると，データの待ち合わせ時間が発生します。詳細は**8.3　パイプラインの乱れ**を参照してください。

8.2.2　ストア命令

[対 象 の 命 令]　ST, SST

[パイプライン]

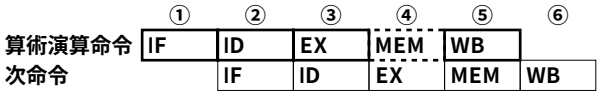


[説　　明]　パイプラインはIF, ID, EX, MEM, WBの５ステージですが，レジスタへのデータの書き込みがないのでWBステージでは何も行いません。

８.２.３　算術演算命令（乗算命令，除算命令を除く）

［対 象 の 命 令］　MOV, MOVEA, MOVHI, ADD, ADDI, CMP, SUB, SUBR, SETF

［パイプライン］

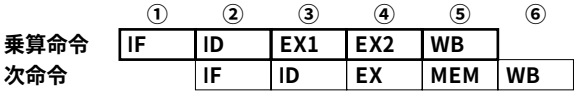


［説　　明］　パイプラインはIF, ID, EX, MEM, WBの５ステージですが，メモリへのアクセスがないのでMEMステージでは何も行いません。

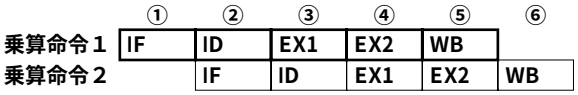
８.２.４　乗算命令

［対 象 の 命 令］　MULH, MULHI

［パイプライン］　（１）次命令が乗算命令以外の場合



（２）次命令が乗算命令の場合

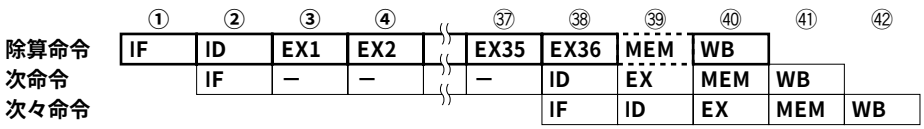


［説　　明］　パイプラインはIF, ID, EX1, EX2, WBの５ステージです。MEMステージはありません。EXステージは２クロックかかりますが，EX1とEX2は独立して動作できます。したがって，乗算命令を繰り返しても命令実行クロック数は１となります。ただし，乗算命令の直後に実行結果を使用する命令を配置すると，データの待ち合わせ時間が発生します。詳細は**８.３　パイプラインの乱れ**を参照してください。

８.２.５　除算命令

［対 象 の 命 令］　DIVH

［パイプライン］



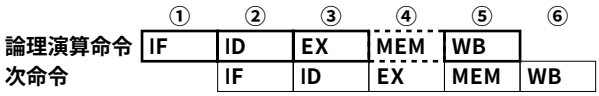
－：待ち合わせのために挿入されるアイドル

[説　　明]　パイプラインはIF, ID, EX1-EX36, MEM, WBの40ステージです。EXステージに36クロックかかります。また，メモリへのアクセスがないのでMEMステージでは何も行いません。

8.2.6　論理演算命令

[対 象 の 命 令]　NOT, OR, ORI, XOR, XORI, AND, ANDI, TST, SHR, SAR, SHL

[パイプライン]

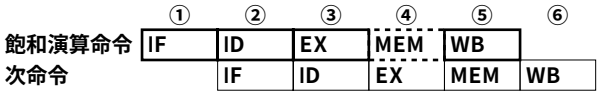


[説　　明]　パイプラインはIF, ID, EX, MEM, WBの５ステージですが，メモリへのアクセスがないのでMEMステージでは何も行いません。

8.2.7　飽和演算命令

[対 象 の 命 令]　SATADD, SATSUB, SATSUBI, SATSUBR

[パイプライン]



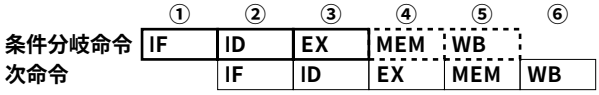
[説　　明]　パイプラインはIF, ID, EX, MEM, WBの５ステージですが，メモリへのアクセスがないのでMEMステージでは何も行いません。

8.2.8　分岐命令

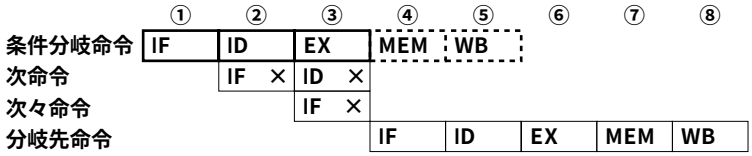
（1）条件分岐命令

[対 象 の 命 令]　Bcond命令（BGT, BGE, BLT, BLE, BH, BNL, BL, BNH, BE, BNE, BV, BNV, BN, BP, BC, BNC, BZ, BNZ, BSA）：BR命令を除く

[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
ID　×：無効となる命令デコード

［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と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。

（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。

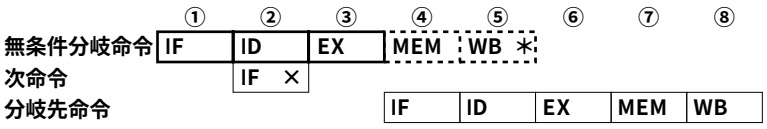
（ｂ）条件が成立した場合

分岐命令の命令実行クロック数は３となります。分岐命令の次命令のIF，次々命令のIFは無効となります。

（２）無条件分岐命令

［対 象 の 命 令］　JMP, JR, JARL, BR

［パイプライン］



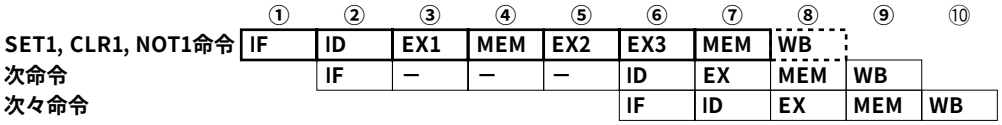
IF　×：無効となる命令フェッチ
WB　＊：JMP命令，JR命令，BR命令の場合は何も行われませんが，JARL命令の場合は復帰PCの書き込みが行われます。

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

8.2.9　ビット操作命令

（1）SET1, CLR1, NOT1

［パイプライン］

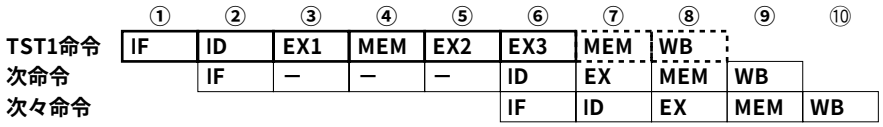


－：待ち合わせのために挿入されるアイドル

［説　　明］　パイプラインはIF, ID, EX1, MEM, EX2, EX3, MEM, WBの８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３クロック，MEMステージに２クロックかかります。

（2）TST1

［パイプライン］



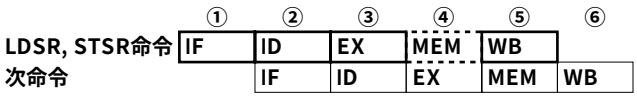
－：待ち合わせのために挿入されるアイドル

［説　　明］　パイプラインはIF, ID, EX1, MEM, EX2, EX3, MEM, WBの８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，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３クロック，MEMステージに２クロックかかります。

8.2.10　特殊命令

（1）LDSR, STSR

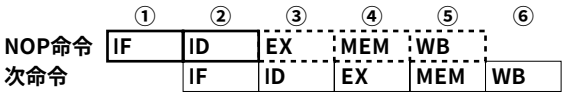
［パイプライン］



［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・アクセスがないためMEMステージでは何も行いません。また，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。詳細は**８.３　パイプラインの乱れ**を参照してください。

（２）NOP

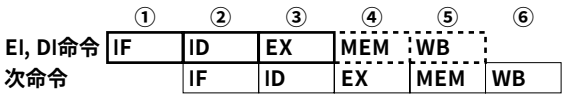
［パイプライン］



［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。

（３）EI, DI

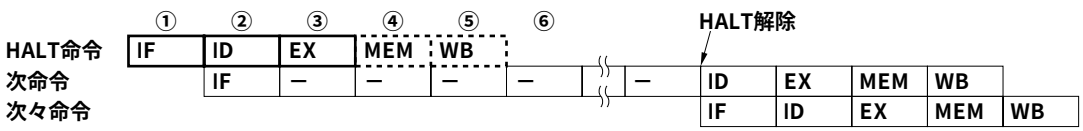
［パイプライン］



［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。

（４）HALT

［パイプライン］

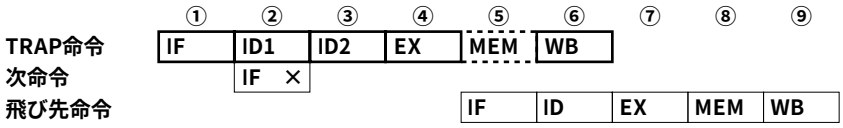


－：待ち合わせのために挿入されるアイドル

［説　　明］　パイプラインはIF, 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状態が解除されるまでIDステージが遅れます。

（5）TRAP

［パイプライン］

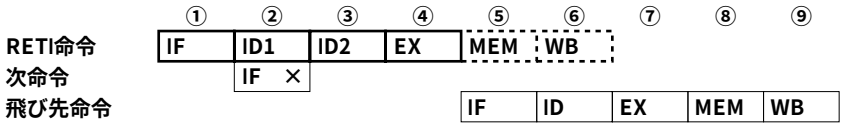


IF　×：無効となる命令フェッチ
ID1　：TRAPコードの検出
ID2　：アドレス生成

［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がないので，MEM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，次々命令のIFは無効となります。

（6）RETI

［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，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

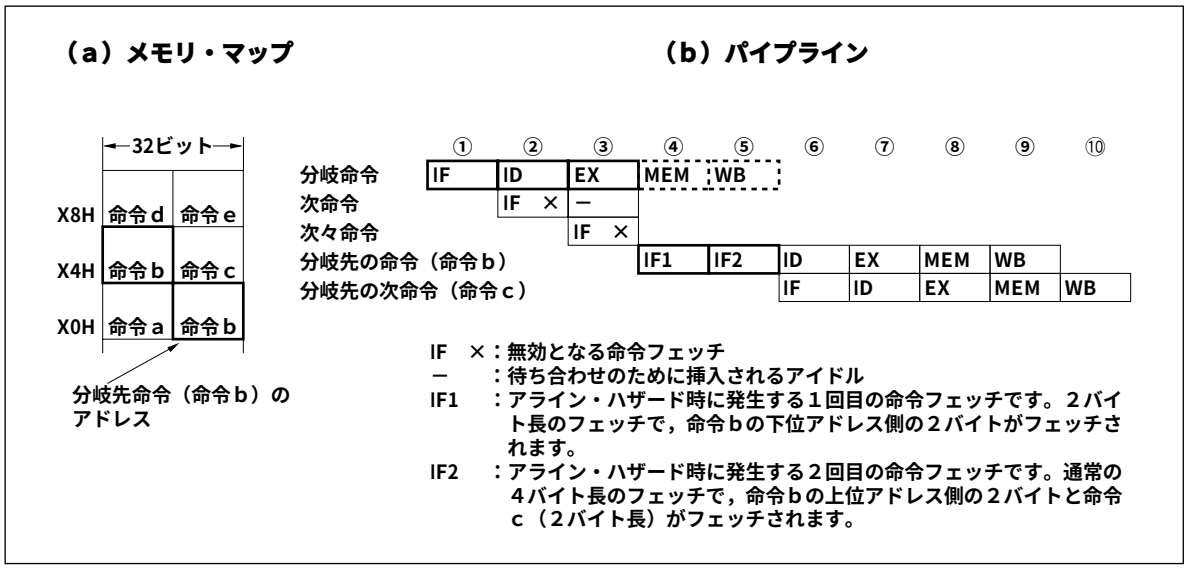
パイプラインはIF（インストラクション・フェッチ）からWB（ライトバック）までの５ステージで構成され，基本的にはそれぞれのステージは１クロックで処理されますが，場合によってはパイプラインが乱れて，実行クロック数が増加する場合があります。この節ではパイプラインを乱す主な要因を示します。

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 = 1, 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８－２　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。V850ファミリは，このハザードをCPUで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V850ファミリは，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８－３　ロード命令実行結果の参照例

図８－３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（MULH, MULHI）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）。

V850ファミリ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８－４　乗算命令実行結果の参照例

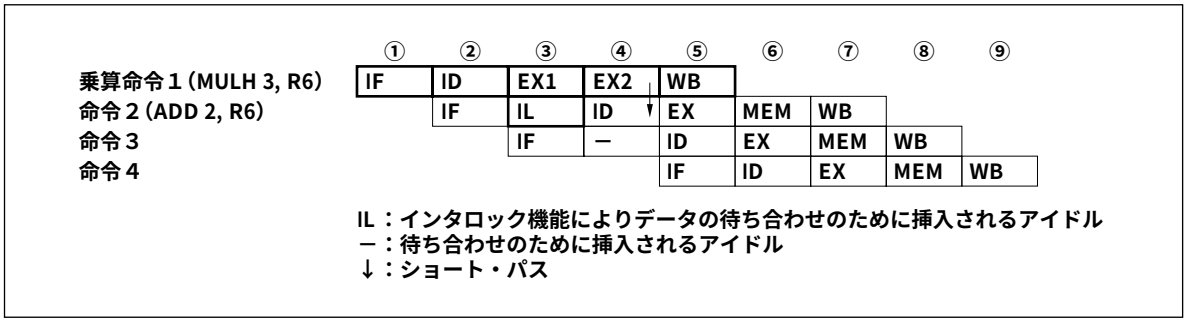


図８－４のように，乗算命令の直後にその結果を使用する命令を配置すると，インタロック機能によるデータの待ち合わせ時間が発生し，実行速度が低下します。乗算命令の結果を使用する命令は，乗算命令の２命令以後に配置することにより，実行速度の低下を防げます。

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）。

V850ファミリ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８－５　EIPC, FEPCを対象とするLDSR命令実行結果の参照例

	①	②	③	④	⑤	⑥	⑦	⑧	⑨	⑩
LDSR命令（LDSR R6, 0）注	IF	ID	EX	MEM	WB					
STSR命令（STSR 0, R7）注		IF	IL	IL	ID	EX	MEM	WB		
次命令			IF	－	－	ID	EX	MEM	WB	
次々命令						IF	ID	EX	MEM	WB

IL：インタロック機能によりデータの待ち合わせのために挿入されるアイドル
－：待ち合わせのために挿入されるアイドル

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８－５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。

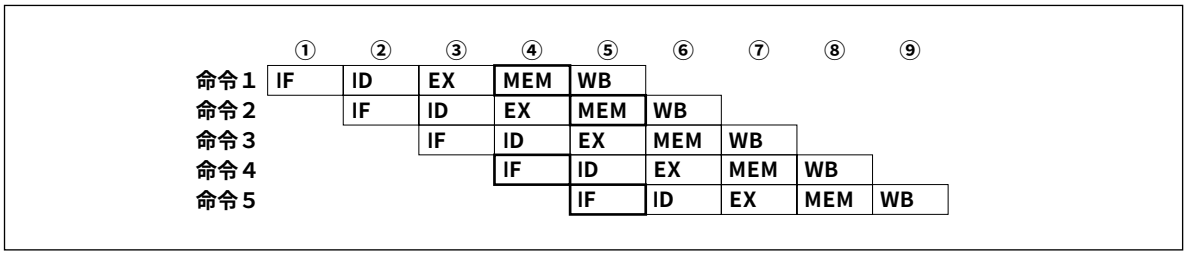
8.4　パイプラインに関する補足事項

8.4.1　ハーバード・アーキテクチャ

V850ファミリではハーバード・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）V850ファミリ（ハーバード・アーキテクチャ）の場合

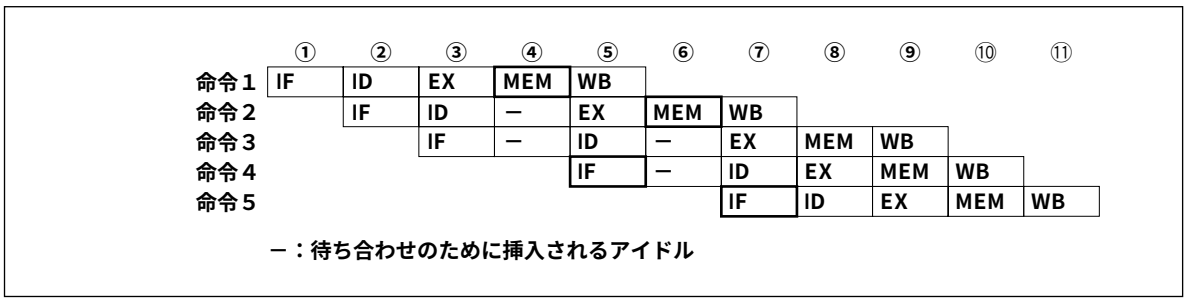
命令１のMEMと命令４のIF，および命令２のMEMと命令５のIFが同時に実行でき，パイプラインが乱れません。



	①	②	③	④	⑤	⑥	⑦	⑧	⑨
命令１	IF	ID	EX	MEM	WB				
命令２		IF	ID	EX	MEM	WB			
命令３			IF	ID	EX	MEM	WB		
命令４				IF	ID	EX	MEM	WB	
命令５					IF	ID	EX	MEM	WB

（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。



	①	②	③	④	⑤	⑥	⑦	⑧	⑨	⑩	⑪
命令１	IF	ID	EX	MEM	WB						
命令２		IF	ID	－	EX	MEM	WB				
命令３			IF	－	ID	－	EX	MEM	WB		
命令４					IF	－	ID	EX	MEM	WB	
命令５							IF	ID	EX	MEM	WB

－：待ち合わせのために挿入されるアイドル

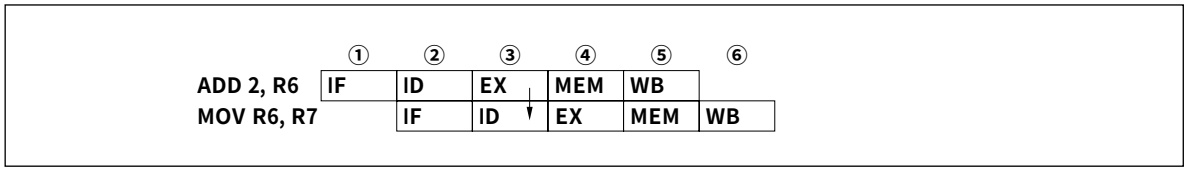
8.4.2　ショート・パス

V850ファミリはショート・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

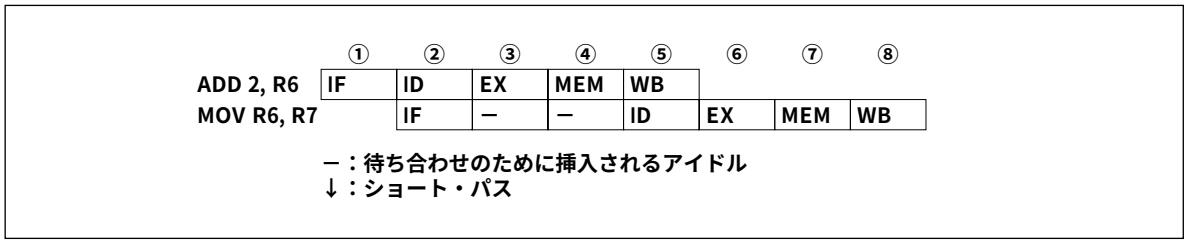
・V850ファミリ（ショート・パス内蔵）の場合

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

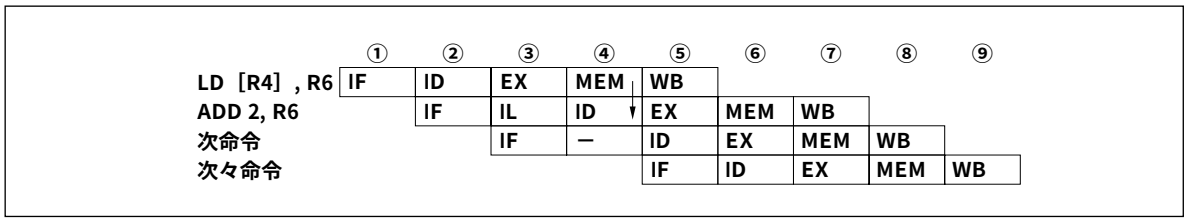
前命令のWBまで，直後のIDが遅れます。



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

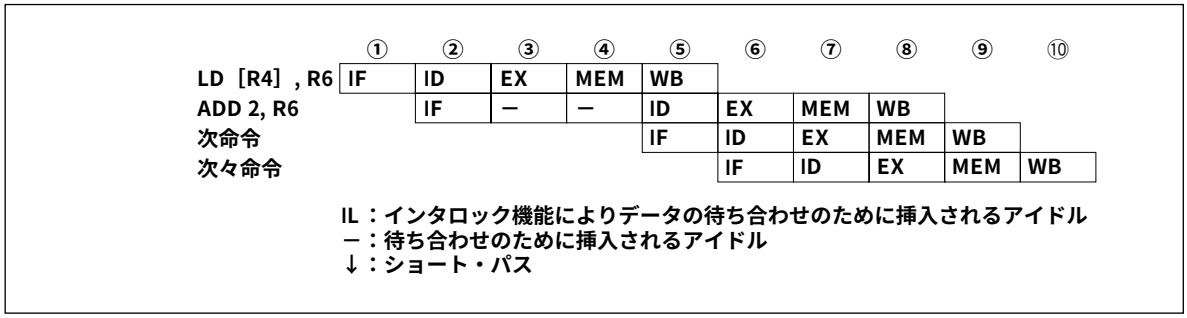
・V850ファミリ（ショート・パス内蔵）の場合

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。



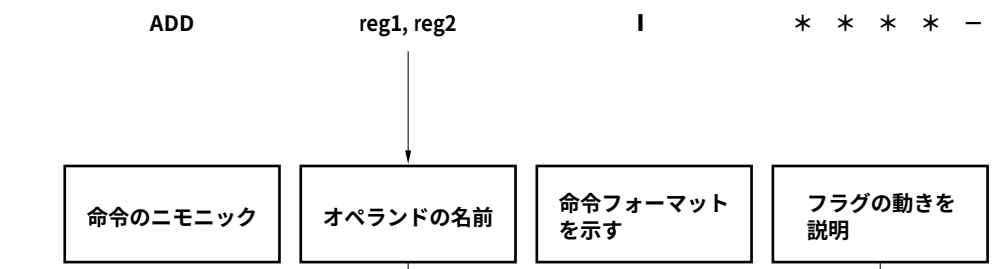
付録A　命令ニモニック（アルファベット順）

付録として，これまでに述べた命令ニモニックの一覧を載せます。

次の命令ニモニックは，アルファベット順に命令が並べられており，辞書感覚で命令を見つけることができます。

命令ニモニック	オペランド	フォーマット	CY	OV	S	Z	SAT

凡例



名　前	意　　味
reg1	汎用レジスタ（ソース・レジスタとして使用する。）
reg2	汎用レジスタ（おもにデスティネーション・レジスタとして使用する。一部のソース・レジスタとしても使用する。）
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
disp×	×ビット・ディスプレースメント
regID	システム・レジスタ番号
vector	トラップ・ベクタに対応するトラップ・ハンドラ・アドレス
cccc	条件コードを示す４ビット・データ

識別子	意　　味
0	リセット（0）される
＊	命令実行結果に従ってセット（1）またはリセット（0）される
—	変化なし

表A－1　命令ニモニック（アルファベット順）（1/10）

命　令ニモニック	オペランド	フォーマット	CY	OV	S	Z	SAT	命令機能
ADD	reg1, reg2	I	＊	＊	＊	＊	－	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	＊	＊	＊	＊	－	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	＊	＊	＊	＊	－	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	＊	＊	－	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	＊	＊	－	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）。命令が指定する条件フラグをテストし，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。
CLR1	bit#3, disp16［reg1］	VIII	－	－	－	＊	－	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。

表A－1　命令ニモニック（アルファベット順）（2/10）

命　令ニモニック	オペランド	フォーマット	CY	OV	S	Z	SAT	命令機能
CMP	reg1, reg2	I	＊	＊	＊	＊	－	比較。reg2のワード・データとreg1のワード・データを比較し，結果を条件フラグに示します。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	＊	＊	＊	＊	－	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，結果を条件フラグに示します。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。
DI	－	X	－	－	－	－	－	マスカブル割り込みの禁止。PSW中のIDフラグをセットし，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。
DIVH	reg1, reg2	I	－	＊	＊	＊	－	符号付き除算。reg2のワード・データをreg1の下位ハードワード・データで除算し，その商をreg2に格納します。
EI	－	X	－	－	－	－	－	マスカブル割り込みの許可。PSW中のIDフラグをリセットし，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	－	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。
JARL	disp22, reg2	V	－	－	－	－	－	分岐およびレジスタ・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。

表A－1　命令ニモニック（アルファベット順）（3/10）

命　令　ニモニック	オペランド	フォーマット	CY	OV	S	Z	SAT	命令機能
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.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。この，32ビット・アドレスのビット０を０にマスクしたアドレスからハーフワード・データを読み出し，ワード長まで符号拡張し，reg2に格納します。
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。この，32ビット・アドレスのビット０およびビット１を０にマスクしたアドレスからワード・データを読み出し，reg2に格納します。
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。

表A－1　命令ニモニック（アルファベット順）（4/10）

命　令 ニモニック	オペランド	フォーマット	CY	OV	S	Z	SAT	命令機能
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。
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。
MULH	reg1, reg2	I	－	－	－	－	－	符号付き乗算。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。
NOP	－	I	－	－	－	－	－	ノー・オペレーション。
NOT	reg1, reg2	I	－	0	＊	＊	－	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。

表A－1　命令ニモニック（アルファベット順）（5/10）

命　令ニモニック	オペランド	フォーマット	CY	OV	S	Z	SAT	命令機能
NOT1	bit#3, disp16［reg1］	Ⅷ	－	－	－	＊	－	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。
OR	reg1, reg2	Ⅰ	－	0	＊	＊	－	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	＊	＊	－	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。
RETI	－	Ⅹ	＊	＊	＊	＊	＊	例外または割り込みルーチンから戻る。システム・レジスタから，復帰PCとPSWを取り出し，例外または割り込みルーチンから復帰する命令です。
SAR	reg1, reg2	Ⅸ	＊	0	＊	＊	－	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	＊	＊	－	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。
SATADD	reg1, reg2	Ⅰ	＊	＊	＊	＊	＊	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フラグをセットします。

表A－1　命令ニモニック（アルファベット順）（6/10）

命　令ニモニック	オペランド	フォーマット	CY	OV	S	Z	SAT	命令機能
SATADD	imm5, reg2	II	＊	＊	＊	＊	＊	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フラグをセットします。
SATSUB	reg1, reg2	I	＊	＊	＊	＊	＊	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フラグをセットします。
SATSUBI	imm16, reg1, reg2	VI	＊	＊	＊	＊	＊	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フラグをセットします。
SATSUBR	reg1, reg2	I	＊	＊	＊	＊	＊	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，条件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。
SET1	bit#3, disp16［reg1］	VIII	－	－	－	＊	－	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します。

表A－1　命令ニモニック（アルファベット順）（7/10）

命　令　ニモニック	オペランド	フォーマット	CY	OV	S	Z	SAT	命令機能
SHL	reg1, reg2	IX	＊	0	＊	＊	－	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	＊	＊	－	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	＊	＊	－	論理右シフト。reg2のワード・データをreg1の下位５ビットで示されるシフト数分，論理右シフト（MSB側に０を送り込む）し，reg2に書き込みます。
SHR	imm5, reg2	II	＊	0	＊	＊	－	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.H	disp8［ep］, reg2	IV	－	－	－	－	－	ハーフワード・ロード。エレメント・ポインタと，ワード長までゼロ拡張した８ビット・ディスプレースメントを加算して32ビット・アドレスを生成します。この，32ビット・アドレスのビット０を０にマスクしたアドレスからハーフワード・データを読み出し，ワード長まで符号拡張し，reg2に格納します。

表A－1　命令ニモニック（アルファベット順）（8/10）

命　令 ニモニック	オペランド	フォーマット	CY	OV	S	Z	SAT	命令機能
SLD.W	disp8［ep］, reg2	IV	－	－	－	－	－	ワード・ロード。エレメント・ポインタと，ワード長までゼロ拡張した８ビット・ディスプレースメントを加算して32ビット・アドレスを生成します。この，32ビット・アドレスのビット０およびビット１を０にマスクしたアドレスからワード・データを読み出し，reg2に格納します。
SST.B	reg2, disp7［ep］	IV	－	－	－	－	－	バイト・ストア。エレメント・ポインタと，ワード長までゼロ拡張した７ビット・ディスプレースメントを加算して32ビット・アドレスを生成します。reg2の最下位バイト・データを生成したアドレスに格納します。
SST.H	reg2, disp8［ep］	IV	－	－	－	－	－	ハーフワード・ストア。エレメント・ポインタと，ワード長までゼロ拡張した８ビット・ディスプレースメントを加算して32ビット・アドレスを生成します。reg2の下位ハーフワード・データを生成した32ビット・アドレスのビット０を０にマスクしたアドレスに格納します。
SST.W	reg2, disp8［ep］	IV	－	－	－	－	－	ワード・ストア。エレメント・ポインタと，ワード長までゼロ拡張した８ビット・ディスプレースメントを加算して32ビット・アドレスを生成します。reg2のワード・データを生成した32ビット・アドレスのビット０およびビット１を０にマスクしたアドレスに格納します。
ST.B	reg2, disp16［reg1］	VII	－	－	－	－	－	バイト・ストア。reg1のデータと，ワード長まで符号拡張した16ビット・ディスプレースメントを加算して32ビット・アドレスを生成します。reg2の最下位バイト・データを生成したアドレスに格納します。

表A－1　命令ニモニック（アルファベット順）（9/10）

命　令　ニモニック	オペランド	フォーマット	CY	OV	S	Z	SAT	命令機能
ST.H	reg2, disp16［reg1］	VII	－	－	－	－	－	ハーフワード・ストア。reg1のデータと，ワード長まで符号拡張した16ビット・ディスプレースメントを加算して32ビット・アドレスを生成します。reg2の下位ハーフワードのデータを生成した32ビット・アドレスのビット０を０にマスクしたアドレスに格納します。
ST.W	reg2, disp16［reg1］	VII	－	－	－	－	－	ワード・ストア。reg1のデータと，ワード長まで符号拡張した16ビット・ディスプレースメントを加算して32ビット・アドレスを生成します。reg2のワード・データを生成した32ビット・アドレスのビット０およびビット１を０にマスクしたアドレスに格納します。
STSR	regID, reg2	IX	－	－	－	－	－	システム・レジスタの内容のストア。regIDで指定されるシステム・レジスタの内容をreg2に設定します。
SUB	reg1, reg2	I	＊	＊	＊	＊	－	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	＊	＊	＊	＊	－	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。
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	＊	＊	－	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。

表A－1　命令ニモニック（アルファベット順）（10/10）

命　令ニモニック	オペランド	フォーマット	CY	OV	S	Z	SAT	命令機能
TST1	bit#3, disp16［reg1］	VIII	－	－	－	＊	－	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フラグをセットし，１ならばリセットします。
XOR	reg1, reg2	I	－	0	＊	＊	－	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	＊	＊	－	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。

付録Ｂ　命令一覧

表Ｂ－１　ニモニック・リスト

ニモニック	機能	ニモニック	機能
	ロード／ストア命令		分岐命令
LD.B	Load Byte	JMP	Jump Register
LD.H	Load Halfword	JR	Jump Relative
LD.W	Load Word	JARL	Jump and Register Link
SLD.B	Load Byte	Bcond	Branch on Condition Code
SLD.H	Load Halfword		ビット操作命令
SLD.W	Load Word	SET1	Set Bit
ST.B	Store Byte	CLR1	Clear Bit
ST.H	Store Halfword	NOT1	Not Bit
ST.W	Store Word	TST1	Test Bit
SST.B	Store Byte		特殊命令
SST.H	Store Halfword	LDSR	Load System Register
SST.W	Store Word	STSR	Store System Register
	整数算術演算／論理演算命令／飽和演算命令（2オペランド・レジスタ）	TRAP	Trap
MOV	Move	RETI	Return from Trap or Interrupt
ADD	Add	HALT	Halt
SUB	Subtract	DI	Disable Interrupt
SUBR	Subtract Reverse	EI	Enable Interrupt
MULH	Multiply Halfword	NOP	No Operation
DIVH	Divide Halfword		
CMP	Compare		
SATADD	Saturated Add		
SATSUB	Saturated Subtract		
SATSUBR	Saturated Subtract Reverse		
TST	Test		
OR	Or		
AND	And		
XOR	Exclusive Or		
NOT	Not		
SHL	Shift Logical Left		
SHR	Shift Logical Right		
SAR	Shift Arithmetic Right		
	（２オペランド・イミーディエト）		
MOV	Move		
ADD	Add		
CMP	Compare		
SATADD	Saturated Add		
SETF	Set Flag Condition		
SHL	Shift Logical Left		
SHR	Shift Logical Right		
SAR	Shift Arithmetic Right		
	（３オペランド）		
MOVHI	Move High Halfword		
MOVEA	Move Effective Address		
ADDI	Add Immediate		
MULHI	Multiply Halfword Immediate		
SATSUBI	Saturated Subtract Immediate		
ORI	Or Immediate		
ANDI	And Immediate		
XORI	Exclusive Or Immediate		

表B－2　命令セット一覧

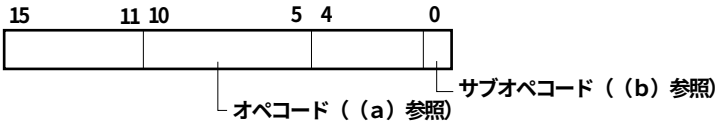
命令コード b10・・・・b5	命令形式		フォーマット	備　　考
0 0 0 0 0 0	MOV	reg1, reg2	I	reg1, reg2=0のとき　NOP
0 0 0 0 0 1	NOT	reg1, reg2		
0 0 0 0 1 0	DIVH	reg1, reg2		
0 0 0 0 1 1	JMP	[reg1]		
0 0 0 1 0 0	SATSUBR	reg1, reg2		
0 0 0 1 0 1	SATSUB	reg1, reg2		
0 0 0 1 1 0	SATADD	reg1, reg2		
0 0 0 1 1 1	MULH	reg1, reg2		
0 0 1 0 0 0	OR	reg1, reg2		
0 0 1 0 0 1	XOR	reg1, reg2		
0 0 1 0 1 0	AND	reg1, reg2		
0 0 1 0 1 1	TST	reg1, reg2		
0 0 1 1 0 0	SUBR	reg1, reg2		
0 0 1 1 0 1	SUB	reg1, reg2		
0 0 1 1 1 0	ADD	reg1, reg2		
0 0 1 1 1 1	CMP	reg1, reg2		
0 1 0 0 0 0	MOV	imm5, reg2	II	
0 1 0 0 0 1	SATADD	imm5, reg2		
0 1 0 0 1 0	ADD	imm5, reg2		
0 1 0 0 1 1	CMP	imm5, reg2		
0 1 0 1 0 0	SHR	imm5, reg2		
0 1 0 1 0 1	SAR	imm5, reg2		
0 1 0 1 1 0	SHL	imm5, reg2		
0 1 0 1 1 1	MULH	imm5, reg2		
0 1 1 0 × ×	SLD.B	disp7［ep］, reg2	IV	
0 1 1 1 × ×	SST.B	reg2, disp7［ep］		
1 0 0 0 × ×	SLD.H	disp8［ep］, reg2		
1 0 0 1 × ×	SST.H	reg2, disp8［ep］		
1 0 1 0 × ×	SLD.W	disp8［ep］, reg2		
1 0 1 0 × ×	SST.W	reg2, disp8［ep］		
1 0 1 1 × ×	Bcond	disp9	III	
1 1 0 0 0 0	ADDI	imm16, reg1, reg2	VI	
1 1 0 0 0 1	MOVEA	imm16, reg1, reg2		
1 1 0 0 1 0	MOVHI	imm16, reg1, reg2		
1 1 0 0 1 1	SATSUBI	imm16, reg1, reg2		
1 1 0 1 0 0	ORI	imm16, reg1, reg2		
1 1 0 1 0 1	XORI	imm16, reg1, reg2		
1 1 0 1 1 0	ANDI	imm16, reg1, reg2		
1 1 0 1 1 1	MULHI	imm16, reg1, reg2		
1 1 1 0 0 0	LD.B	disp16［reg1］, reg2	VII	
1 1 1 0 0 1	LD.H	disp16［reg1］, reg2		
1 1 1 0 1 0	LD.W	disp16［reg1］, reg2		
1 1 1 0 1 0	ST.B	reg2, disp16［reg1］		
1 1 1 0 1 1	ST.H	reg2, disp16［reg1］		
1 1 1 0 1 1	ST.W	reg2, disp16［reg1］		
1 1 1 1 0 ×	JARL	disp22, reg2	V	reg2=r0のとき　JR　disp22
1 1 1 1 1 0	SET1	bit#3, disp16［reg1］	VIII	
1 1 1 1 1 0	CLR1	bit#3, disp16［reg1］		
1 1 1 1 1 0	NOT1	bit#3, disp16［reg1］		
1 1 1 1 1 0	TST1	bit#3, disp16［reg1］		
1 1 1 1 1 1	SETF	cccc, reg2	IX	
1 1 1 1 1 1	LDSR	reg2, regID		
1 1 1 1 1 1	STSR	regID, reg2		
1 1 1 1 1 1	SHR	reg1, reg2		
1 1 1 1 1 1	SAR	reg1, reg2		
1 1 1 1 1 1	SHL	reg1, reg2		
1 1 1 1 1 1	TRAP	vector	X	
1 1 1 1 1 1	HALT			
1 1 1 1 1 1	RETI			
1 1 1 1 1 1	DI			
1 1 1 1 1 1	EI			
1 1 1 1 1 1	未定義命令			

付録C　命令オペコード・マップ

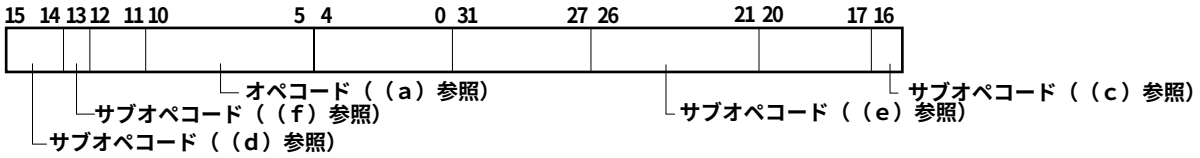
（a）-（f）に命令コードに対応したオペコード・マップを示します。

命令コード

●16ビット長命令形式



●32ビット長命令形式



（a）オペコード部

ビット6-5 / ビット10-7	00	01	10	11	Format
0000	MOV/NOP	NOT	DIVH	JMP	I
0001	SATSUBR	SATSUB	SATADD	MULH	
0010	OR	XOR	AND	TST	
0011	SUBR	SUB	ADD　R, r	CMP　R, r	
0100	MOV　imm5, r	SATADD	ADD　imm5, r	CMP　imm5, r	II
0101	SHR　imm5, r	SAR　imm5, r	SHL　imm5, r	MULH	
0110	SLD.B				IV
0111	SST.B				
1000	SLD.H				
1001	SST.H				
1010	SLD.W/SST.W注1				
1011	Bcond				III
1100	ADDI	MOVEA	MOVHI	SATSUBI	VI
1101	ORI	XORI	ANDI	MULHI	
1110	LD.B	LD.H/LD.W注2	ST.B	ST.H/ST.W注2	V/VII/VIII/IX/X
1111	JARL		ビット操作注3	拡張1注4	

注1．（b）を参照してください。　　注3．（d）を参照してください。

2．（c）　〃　　4．（e）　〃

（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0 / ビット10-7	0	1
0110	SLD.B	
0111	SST.B	
1000	SLD.H	
1001	SST.H	
1010	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16 / ビット6-5	0	1
00	LD.B	
01	LD.H	LD.W
10	ST.B	
11	ST.H	ST.W

（d）ビット操作命令（サブオペコード部）

ビット14 / ビット15	0	1
0	SET1	NOT1
1	CLR1	TST1

（e）拡張1（サブオペコード部）

ビット22-21 / ビット26-23	00	01	10	11
0000	SETF	LDSR	STSR	未定義
0001	SHR R, r	SAR R, r	SHL R, r	未定義
0010	TRAP	HALT	RETI	拡張2[注]
0011 ～ 1111	不正命令			

注　（f）を参照してください。

（f）拡張2（サブオペコード部）

ビット14-13 / ビット15	00	01	10	11
0	DI	未定義		
1	EI			

付録D　総合索引

D.1　50音で始まる語句の索引

【ま行】

【ら行】

【わ行】

D.2　アルファベットで始まる語句の索引

【V】

【W】

【X】

【Z】

〔メ　モ〕

お問い合わせ先

【技術的なお問い合わせ先】

ＮＥＣ半導体テクニカルホットライン
（電話：午前 9:00～12:00，午後 1:00～5:00）

電　話　：044−435−9494
FAX　：044−435−9608
E-mail　：s-info@saed.tmg.nec.co.jp

【営業関係お問い合わせ先】

第一販売事業部

東　京　(03)3798−6106, 6107, 6108
大　阪　(06)6945−3178, 3200, 3208, 3212
仙　台　(022)267−8740
郡　山　(024)923−5591
千　葉　(043)238−8116

第二販売事業部

東　京　(03)3798−6110, 6111, 6112
立　川　(042)526−5981, 6167
松　本　(0263)35−1662
静　岡　(054)254−4794
金　沢　(076)232−7303
松　山　(089)945−4149

第三販売事業部

東　京　(03)3798−6151, 6155, 6586, 1622, 1623, 6156
水　戸　(029)226−1702
広　島　(082)242−5504
前　橋　(027)243−6060
鳥　取　(0857)27−5313
太　田　(0276)46−4014
名古屋　(052)222−2170, 2190
福　岡　(092)261−2806

【資料の請求先】

上記営業関係お問い合わせ先またはNEC特約店へお申しつけください。

【NECエレクトロンデバイス　ホームページ】

NECエレクトロンデバイスの情報がインターネットでご覧になれます。　URL（アドレス）　http://www.ic.nec.co.jp/

アンケート記入のお願い

お手数ですが，このドキュメントに対するご意見をお寄せください。今後のドキュメント作成の参考にさせていただきます。

［ドキュメント名］V850ファミリ　ユーザーズ・マニュアル　アーキテクチャ編
（U10243JJ7V0UM00（第７版））

［お名前など］（さしつかえのない範囲で）

御社名（学校名，その他）（　　）
ご住所（　　）
お電話番号（　　）
お仕事の内容（　　）
お名前（　　）

１．ご評価（各欄に○をご記入ください）

項　　目	大変良い	良　い	普　通	悪　い	大変悪い
全体の構成					
説明内容					
用語解説					
調べやすさ					
デザイン，字の大きさなど					
その他（　　）					
（　　）					

２．わかりやすい所（第　章，第　章，第　章，第　章，その他　　）

理由［　　］

３．わかりにくい所（第　章，第　章，第　章，第　章，その他　　）

理由［　　］

４．ご意見，ご要望

５．このドキュメントをお届けしたのは

NEC販売員，特約店販売員，その他（　　）

ご協力ありがとうございました。
下記あてにFAXで送信いただくか，最寄りの販売員にコピーをお渡しください。

日本電気（株）NECエレクトロンデバイス
半導体テクニカルホットライン
FAX：（044）435-9608

2000.6

キリトリ