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

V830 ファミリ™

32 ビット・マイクロプロセッサ

アーキテクチャ編

資料番号　U12496JJ4V0UM00（第４版）
発行年月　July 1999　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。

- **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
p. 25	２.２.６　プロセッサIDレジスタ（PIR）　修正
p. 114	表５－４　命令実行サイクル数　修正
p. 117	表６－１　例外／割り込み要因コード　注　追加
p. 122	６.４.１　マスカブル割り込みの優先順位　追加
p. 129	７.２.２　命令RAM検索機能（V830, V831）　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，V830ファミリの機能を理解し，それを用いた応用システムを設計するユーザを対象とします。

●V830ファミリ製品

- V830™：μPD705100
- V831™：μPD705101
- V832™：μPD705102

目　的　このマニュアルは，次の構成に示すV830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います。

- レジスタ・セット
- データ・セット
- アドレス空間
- 命　　令
- 割り込みと例外
- 内蔵メモリ
- リセット
- パイプライン

読み方　このマニュアルの読者には，電気，論理回路，およびマイクロコンピュータに関する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。

命令機能の詳細を理解しようとするとき

→**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V830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。

凡　例　データ表記の重み　：左が上位桁，右が下位桁

アクティブ・ロウの表記　：$\overline{\times\times\times}$（端子，信号名称に上線）

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）：$2^{30} = 1024^3$

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。

○V830ファミリに関する資料

製　品　名		データ・シート	ユーザーズ・マニュアル	
愛　称	製　　品		ハードウエア編	アーキテクチャ編
V830	μPD705100	U11483J	U10064J	このマニュアル
V831	μPD705101	U12979J	U12273J	
V832	μPD705102	U13675J	U13577J	

○V830ファミリ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
CA830（Cコンパイラ）		操作編（UNIX™ベース）	U11013J
		操作編（Windows™ベース）	U11068J
		アセンブリ言語編	U11014J
		C言語編	U11010J
		プロジェクト・マネージャ編	U11991J
RX830（リアルタイムOS）	ITRON1	基礎編	U11730J
		インストレーション編	U11731J
	μITRON Ver3.0	基礎編	U13152J
		インストレーション編	U13151J
AZ830（システム・パフォーマンス・アナライザ）			U13621J

目　　次

図の目次

表の目次

〔メ　モ〕

第１章　イントロダクション

V830ファミリは，V830をCPUコアとする，NECの組み込み制御向けRISCマイクロプロセッサです。

1.1　概　　説

V830ファミリは，高性能32ビットRISCマイクロプロセッサです。V830ファミリはマルチメディア機器で使用されるデータ処理を数サイクルで実現できます。高い割り込み応答性，最適化されたパイプライン構造に加え，マルチメディア機能を実現するため，積和演算命令，連結シフト命令，分岐予測を用いた高速分岐命令などを追加しています。

さらに，V810ファミリ™基本命令セットをオブジェクト・レベルで継承することにより，V810ファミリのソフトウエア資産をそのまま使用することができます。

V830ファミリは，ゲーム，カー・ナビゲーション，高機能テレビ，カラーFAX，インターネット，イントラネット機器，各種OA機器など，画像処理を中心として，高速なデータ処理が必要な分野できわめて高いパフォーマンスを実現できます。

1.2　特　　徴

○命令数：102

○最小命令実行サイクル数：１サイクル

○汎用レジスタ：32ビット×32本

○命令セット：V810基本命令セット

積和演算（32ビット×32ビット＋（上位／下位）32ビット）：１-３サイクル

飽和演算（飽和検出機能付き）

連結シフト（64ビット・データのシフト）：１-２サイクル

高速分岐

ブロック転送命令

○メモリ空間

メモリ空間，I/O空間：４Gバイト・リニア・アドレス

○内蔵メモリ

命令キャッシュ（ダイレクト・マップ方式）：４Kバイト

データ・キャッシュ（ダイレクト・マップ方式／ライト・スルー方式）：４Kバイト

命令RAM：４Kバイト

データRAM：４Kバイト

○電力制御

・ストップ・モード

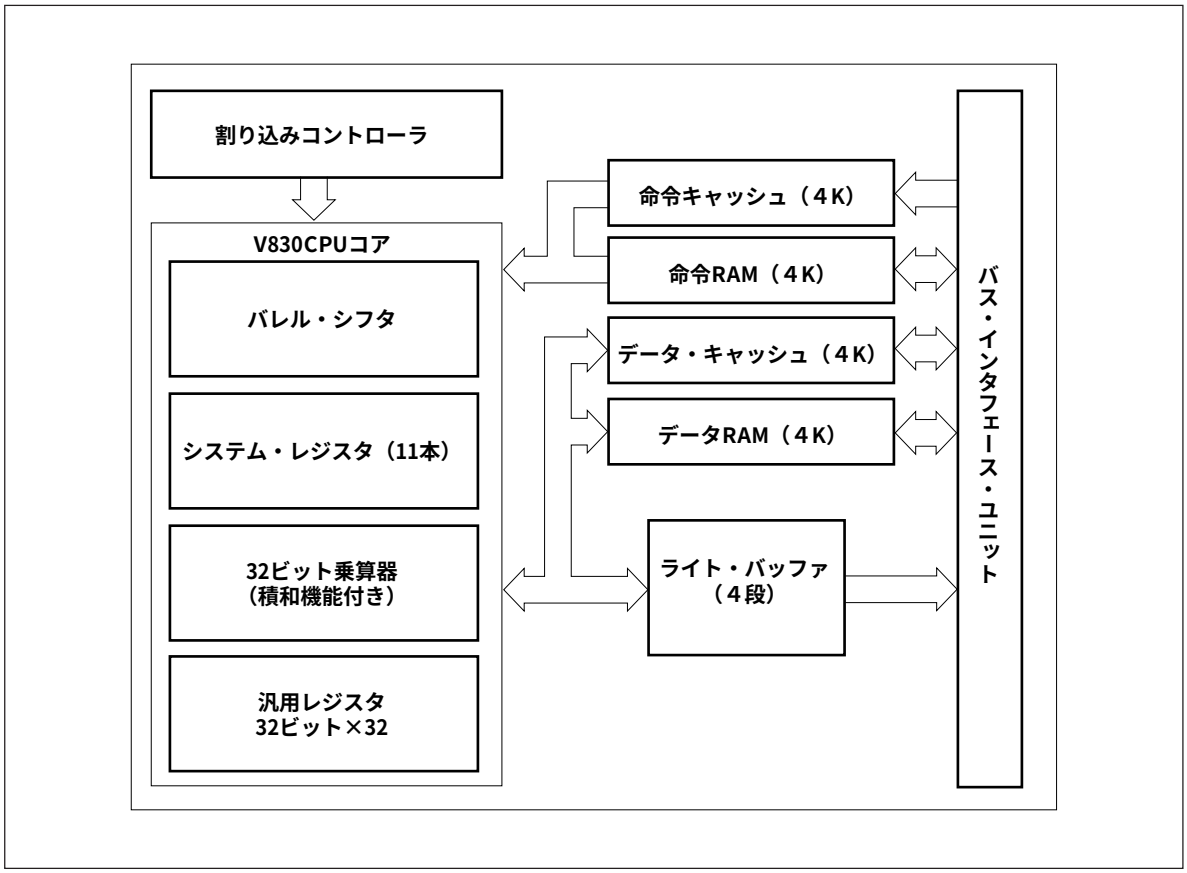
・スリープ・モード

○CMOS構造

１.３　CPU内部構成

V830ファミリの内部構成を図１－１に示します。

図１－１　内部構成図



（１）CPUコア

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サイクルで実行します。

積和機能付き乗算器（32ビット×32ビット＋（上位／下位）32ビット），バレル・シフタ（64ビット・データ・シフトが可能）などの専用ハードウエアを内蔵し，複雑な命令処理を高速に実行できます。

（２）バス・インタフェース

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。バス・インタフェース・ユニットでは，外部データ・バスを32ビットで使用する32ビット・バス・モードと16ビットで使用する16ビット・バス・モードをサポートしています。バス・サイクル起動時に，それぞれのモードに合わせた適切な制御信号を出力します。

（３）割り込みコントローラ

ハードウエア割り込み要求（ノンマスカブル割り込み，マスカブル割り込み）を処理します。マスカブル割り込みの処理ハンドラを内蔵命令RAMへ配置することができます。

（４）ライト・バッファ

CPUが外部ハードウエアへのライト動作を行うときに，ライト・データ（最大４データ）をバッファリングします。ライト・バッファへデータが書き込まれると，CPUはバス・サイクルの終了を待たずに処理を続けることができます。

（５）内蔵メモリ

16 Kバイトのメモリです。メモリは4 Kバイトの４ブロックで構成され，それぞれ命令キャッシュ，データ・キャッシュ，命令RAM，データRAMです。命令RAMはダイレクト・マップ方式，データ・キャッシュはダイレクト・マップ／ライト・スルー方式です。

〔メ　モ〕

第２章　レジスタ・セット

２.１　プログラム・レジスタ・セット

V830ファミリのレジスタ・セットは，一般にプログラマが使用する汎用レジスタ・セットと実行環境の制御をするシステム・レジスタ・セットの２つに分けることができます。レジスタはすべて32ビット幅を持っています。

２.１.１　汎用レジスタ・セット

（１）汎用レジスタ

汎用レジスタは，r0-r31の32本があります。r0-r31は，データ・レジスタまたはアドレス・レジスタとして利用できます。ただし，r0, r30, r31はハードウエア的に値が固定されていたり，暗黙的に命令が使用するので注意してください。

（ａ）ハードウエア依存レジスタ

ハードウエア的に値が固定されていたり，命令により暗黙的に使用されるレジスタです。

r0 ：ゼロ・レジスタ
　　常に０を保持しています。
r30：演算予約レジスタ
　　乗算，除算命令を実行したとき，補助的に演算結果を格納するレジスタです。
r31：リンク・ポインタ
　　JAL命令によって戻り先アドレスを格納するレジスタです。

備考　r1～r31の初期値は不定です。

（ｂ）ソフトウエア予約レジスタ

アセンブラとコンパイラが使用するレジスタです。変数用レジスタとして使用するときは，レジスタの内容を破壊しないように一度退避してから使用してください。使用後は元に戻してください。

r1 ：アセンブラ予約レジスタ
　　32ビット・イミーディエト作成用ワーキング・レジスタです。
　　アセンブラが実効アドレスを計算するとき，暗黙的に使用します。

r2 ：ハンドラ・スタック・ポインタ

ハンドラ用のスタック・ポインタとして予約されています。

r3 ：スタック・ポインタ

関数コール時のスタック・フレーム生成用に予約されています。

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するレジスタです。

r5 ：テキスト・ポインタ

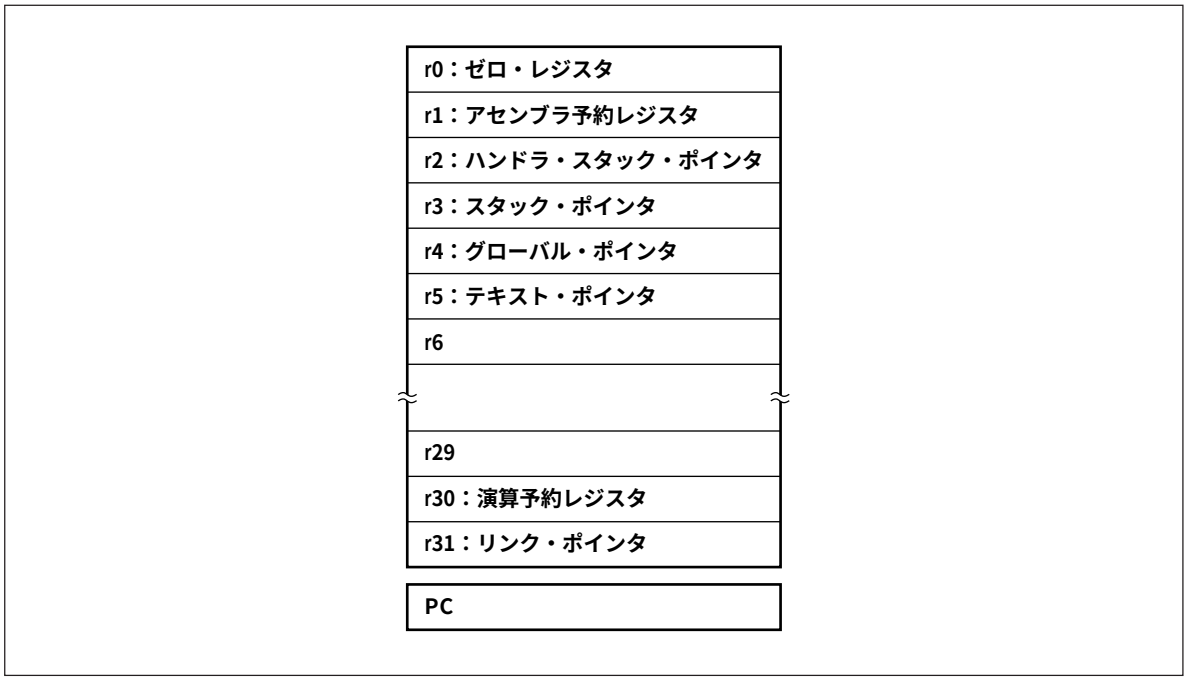
テキスト領域の先頭を指すレジスタです。

2.1.2　プログラム・カウンタ（PC）

実行中の命令の先頭アドレスを保持するレジスタです。プログラム・カウンタ（PC）のビット0は0に固定されています。ハーフワード境界（アドレスのビット0が0）以外への分岐を行ったときは，強制的にビット0を0にマスクします。

リセット時には，FFFFFFF0Hに初期化されます。

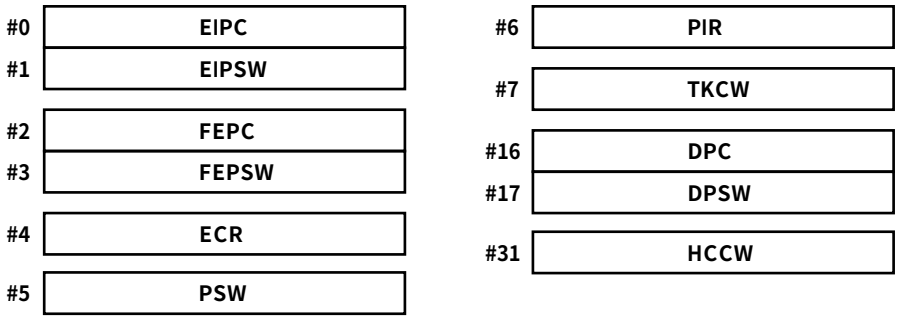
図2－1　プログラム・レジスター覧



2.2　システム・レジスタ・セット

システム・レジスタは，プロセッサの状態制御，例外／割り込み情報の保存，タスクの管理などの役割があり，11本の32ビット・レジスタにより構成されます。システム・レジスタへは，特別な命令（LDSR命令，STSR命令）でアクセスします。

図2－2　システム・レジスター覧



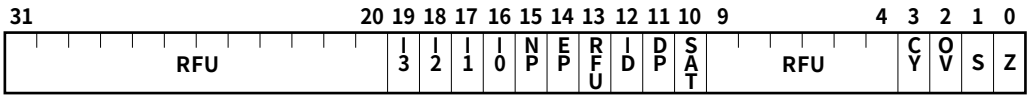
備考　＃はシステム・レジスタ番号を示します。

2.2.1　プログラム・ステータス・ワード（PSW）

プログラム・ステータス・ワードは，プログラムの状態（命令実行の結果）やプロセッサの状態を示すフラグの集まりです。LDSR命令を使用してこのレジスタの各フィールドを変更した場合には，LDSR命令の実行終了直後から変更内容が有効になります。

初期値は，00008000Hです。

PSW (#5)



ビット位置	ビット名	意　　味
31-20	RFU	予約フィールドです（0に固定されています）。
19-16	I3-I0	Interrupt Level マスカブル割り込み許可レベル。
15	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NPビットがセットされ，NMIがマスクされ多重割り込みが禁止されます。 NP = 0：NMI処理中でない。 NP = 1：NMI処理中である。
14	EP	Exception Pending 例外またはトラップ，割り込み処理中であることを示します。例外事象の発生でセットされ，割り込みをマスクします。 EP = 0：例外／トラップ／割り込み処理中でない。 EP = 1：例外／トラップ／割り込み処理中である。
13	RFU	予約フィールドです。（0に固定してください）。
12	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。
11	DP	Debug Pending 致命的例外が処理中であることを示します。 DP = 1：致命的例外処理中である。 DP = 0：致命的例外処理中でない。
10	SAT	Saturate Flag 飽和演算の結果にオーバフローが発生した履歴を示します。なお，SATビットはクリアしないかぎり保持されます。 SAT = 1：オーバフローが発生した。 SAT = 0：オーバフローが発生していない。
9-4	RFU	予約フィールドです。（0に固定してください）。
3	CY	Carry 演算の結果にキャリーがあったかどうかを示します。 CY = 0：キャリーが発生していない。 CY = 1：キャリーが発生した。

ビット位置	ビット名	意　　味
2	OV	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：オーバフローが発生していない。 OV = 1：オーバフローが発生した。
1	S	Sign 演算の結果が負かどうかを示します。 S = 0：演算結果が正またはゼロであった。 S = 1：演算結果が負であった。
0	Z	Zero 演算の結果がゼロかどうかを示します。 Z = 0：演算の結果がゼロでなかった。 Z = 1：演算の結果がゼロであった。

備考　RFU：Reserved for Future Use

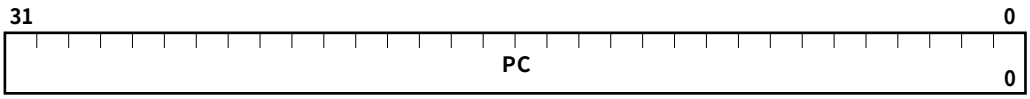
２.２.２　例外／割り込み時状態退避レジスタ（EIPC/EIPSW）

EIPC, EIPSWは，例外またはマスカブル割り込みが発生した場合，その時点のPCおよびPSWを退避するレジスタです。PCはEIPCに，PSWはEIPSWに格納されます。EIPC, EIPSWは１組しかないため，多重例外／多重割り込みを許可する場合はプログラムによって，EIPC, EIPSWを退避してください。

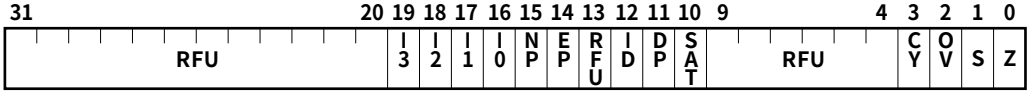
なお，EIPCのビット０とEIPSWのビット31-20，ビット13, ビット9-4は０に固定されています。PSWのEPビットがセットされているときに発生した例外（二重例外）は，EIPC, EIPSWに退避せず，FEPC, FEPSWに退避されます。

初期値は不定です。

EIPC (#0)



EIPSW (#1)



備考　RFU：Reserved for Future Use

2.2.3　NMI／二重例外時状態退避レジスタ（FEPC/FEPSW）

NMIまたは二重例外（PSWのEPビットが１のときに発生する例外）が発生した場合，その時点のPCおよびPSWを退避します。PCはFEPCに，PSWはFEPSWに格納されます。FEPC, FEPSWに退避が発生した場合は緊急であるため，ただちに処理してください。

なお，FEPCのビット０とFEPSWのビット31-20，ビット13，ビット9-4は０に固定されています。

初期値は不定です。

FEPC (#2)

31																															0
PC																															0

FEPSW (#3)

31																							20	19	18	17	16	15	14	13	12	11	10	9							4	3	2	1	0
RFU																								I3	I2	I1	I0	NP	EP	RFU	ID	DP	SAT	RFU							CY	OV	S	Z

備考　RFU：Reserved for Future Use

2.2.4　致命的例外時状態退避レジスタ（DPC/DPSW）

致命的例外が発生した場合（PSWのNPビットが１のときに発生した例外）に，その時点のPCおよびPSWを退避するレジスタです。PCはDPCに，PSWはDPSWに格納されます。このレジスタに退避が発生した場合は緊急であるため，ただちに処理してください。

なお，DPCのビット０とDPSWのビット31-20，ビット13，ビット9-4は０に固定されています。

初期値は不定です。

DPC (#16)

31																															0
PC																															0

DPSW (#17)

31																							20	19	18	17	16	15	14	13	12	11	10	9							4	3	2	1	0
RFU																								I3	I2	I1	I0	NP	EP	RFU	ID	DP	SAT	RFU							CY	OV	S	Z

備考　RFU：Reserved for Future Use

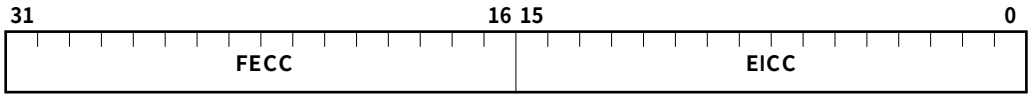
2.2.5　例外要因レジスタ（ECR）

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，例外要因ごとにコード化されています（**第6章　割り込みと例外**を参照）。

ECRは読み出し専用です。LDSR命令を使ってECRにデータを書き込むことはできません。

初期値は，0000FFF0Hです。

ECR (#4)



ビット位置	ビット名	意　　味
31-16	FECC	NMI/二重例外時例外コード
15-0	EICC	例外／割り込み例外コード

★

2.2.6　プロセッサIDレジスタ（PIR）

プロセッサIDレジスタはCPUタイプ番号を識別するレジスタです。PIRの内容は

（1）V830：00008300H

（2）V831：00008301H

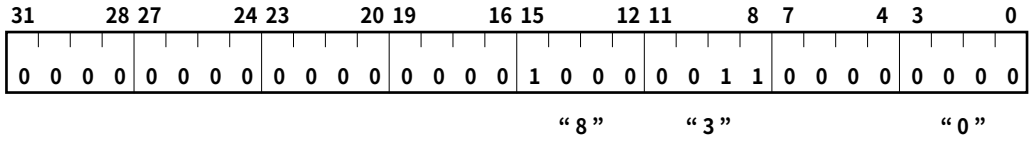
（3）V832：00008302H

となっており，CPUのタイプを示しています。PIRは，読み出し専用です。LDSR命令を使用して，PIRにデータを書き込むことはできません。

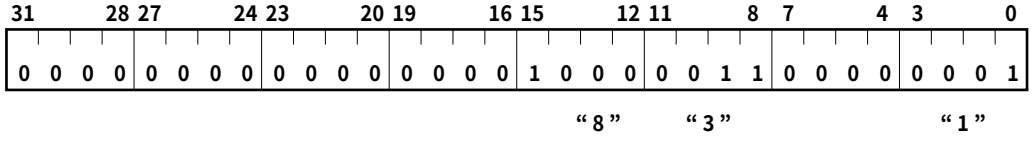
またそれぞれの数値で固定されています。

PIR（#6）

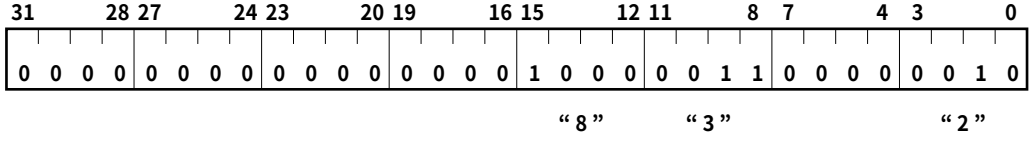
（1）V830



（2）V831



（3）V832

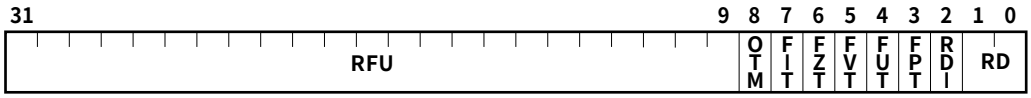


2.2.7　タスク・コントロール・ワード（TKCW）

タスク・コントロール・ワードは，タスクを制御するレジスタです。TKCWは，読み出し専用です。LDSR命令を使ってデータを書き込むことはできません。TKCWは現在は使われていませんが，将来の互換性のために用意されています。

000000E0Hに固定されています。

TKCW (#7)



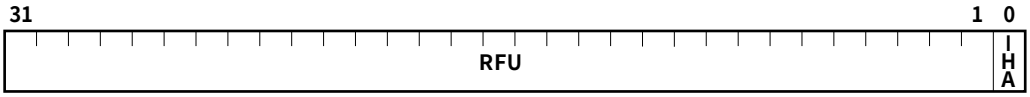
備考　RFU：Reserved for Future Use

2.2.8　ハードウエア・コンフィグレーション・コントロール・ワード（HCCW）

マスカブル割り込みのハンドラ・アドレスを指定するレジスタです。

初期値は，00000000Hです。

HCCW (#31)



ビット位置	ビット名	意　　味
31-1	RFU	予約フィールドです。（0に固定してください）。
0	IHA	Interrupt Handler Address マスカブル割り込みのハンドラ・アドレスを指定します。 IHA = 1：FE0000n0H（内蔵命令RAM） IHA = 0：FFFFFEn0H（外部メモリ） n：割り込みレベル

備考　RFU：Reserved for Future Use

2.3　システム・レジスタ番号

システム・レジスタへの入出力は，LDSR命令，STSR命令で次のシステム・レジスタ番号を指定することにより行われます。

番号	システム・レジスタ	オペランド指定の可否	
		LDSR	STSR
0	EIPC　: Exception/Interrupt PC	○	○
1	EIPSW　: Exception/Interrupt PSW	○	○
2	FEPC　: Fatal Error PC	○	○
3	FEPSW : Fatal Error PSW	○	○
4	ECR　: Exception Cause Register	−	○
5	PSW　: Program Status Word	○	○
6	PIR　: Processor ID Register	−	○
7	TKCW　: Task Control Word	−	○
8-15	予約		
16	DPC　: Debug PC	○	○
17	DPSW　: Debug PSW	○	○
18-30	予約		
31	HCCW　: Hardware Configuration Control Word	○	○

−：禁止（アクセス禁止）

○：許可（アクセス許可）

〔メ　モ〕

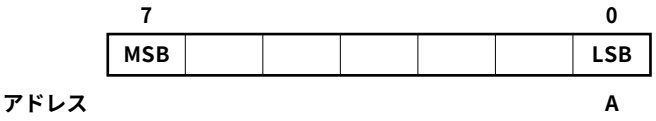
第３章　データ・セット

３.１　データ・タイプ

V830ファミリがサポートするデータ・タイプは，バイト（８ビット），ハーフワード（16ビット），ワード（32ビット）の３通りです。それぞれのデータは，バイト境界，ハーフワード境界，ワード境界に配置しなければならず，アドレシングはリトル・エンディアンに従います。

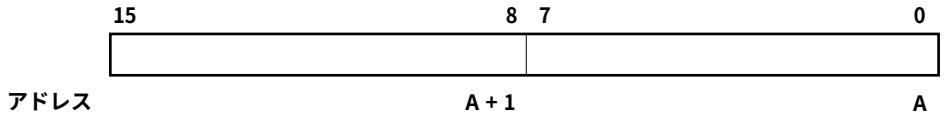
（１）バイト・データ

バイト・データは連続した８ビットで構成され，各ビットには名前が付けられています。この名前のビット０がLSB（Least Significant Bit），ビット７がMSB（Most Significant Bit）に対応します。このデータは，任意のアドレス上に配置することができます。



（２）ハーフワード・データ

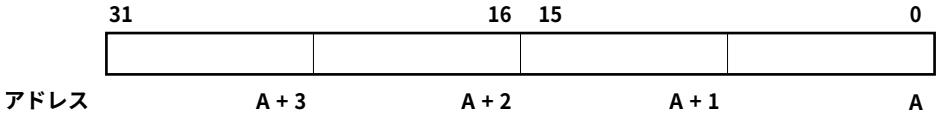
ハーフワード・データは連続した16ビットで構成され，各ビットには名前が付けられています。この名前のビット０はLSBに，ビット15はMSBに対応します。ハーフワード・データは，ハーフワード境界（ビット０が存在する部分のアドレスのビット０が０となるアドレス領域）に配置しなければなりません。



A = 2n（nは正整数）

（3）ワード・データ

ワード・データは連続した32ビットで構成され，各ビットには名前が付けられています。この名前のビット０はLSBに，ビット31はMSBに対応します。ワード・データは，ワード境界（ビット０が存在する部分のアドレスのビット０，ビット１が０となるアドレス領域）に配置しなければなりません。



A = 4n（nは正整数）

3.1.1　整　　数

V830ファミリでは整数は２の補数で表現されます。整数は，バイト，ハーフワード，ワードの３通りのデータで表現できます。整数の位取りはその長さとは独立にビット０を最下位ビットとし，ビット番号が増えるに従って位取りを高くします。

データ長	範囲（10進表記）
バイト　（８ビット）	－128～＋127
ハーフワード（16ビット）	－32768～＋32767
ワード　（32ビット）	－2147483648～＋2147483647

3.1.2　符号なし整数

符号なし整数は，最上位ビットを符号ビットとせずに，正の整数として扱ったデータ型です。データはすべて２進数で表現され，バイト，ハーフワード，ワードの３通りのデータ・サイズを持ちます。符号なし整数の位取りはその長さとは独立に，ビット０を最下位ビットとし，ビット番号が増えるに従って位取りを高くします。

データ長	範囲（10進表記）
バイト　（８ビット）	0～255
ハーフワード（16ビット）	0～65535
ワード　（32ビット）	0～4294967295

3.2　データのアラインメント

V830ファミリでは，ワード・データはワード境界，ハーフワード・データはハーフワード境界，バイト・データはバイト境界にアラインしなければなりません。データ・アラインに違反が生じた際，データのアドレスを自動的にアクセス可能なアドレスへ変更します。この変更により，データ・アクセスが正確に行われるか，間違って行われるかは保証されません。次に変更の方法を示します。

データ・サイズ	変更方法
バイト・データ	−
ハーフワード・データ	ビット0を0にマスク
ワード・データ	ビット0，ビット1を0にマスク

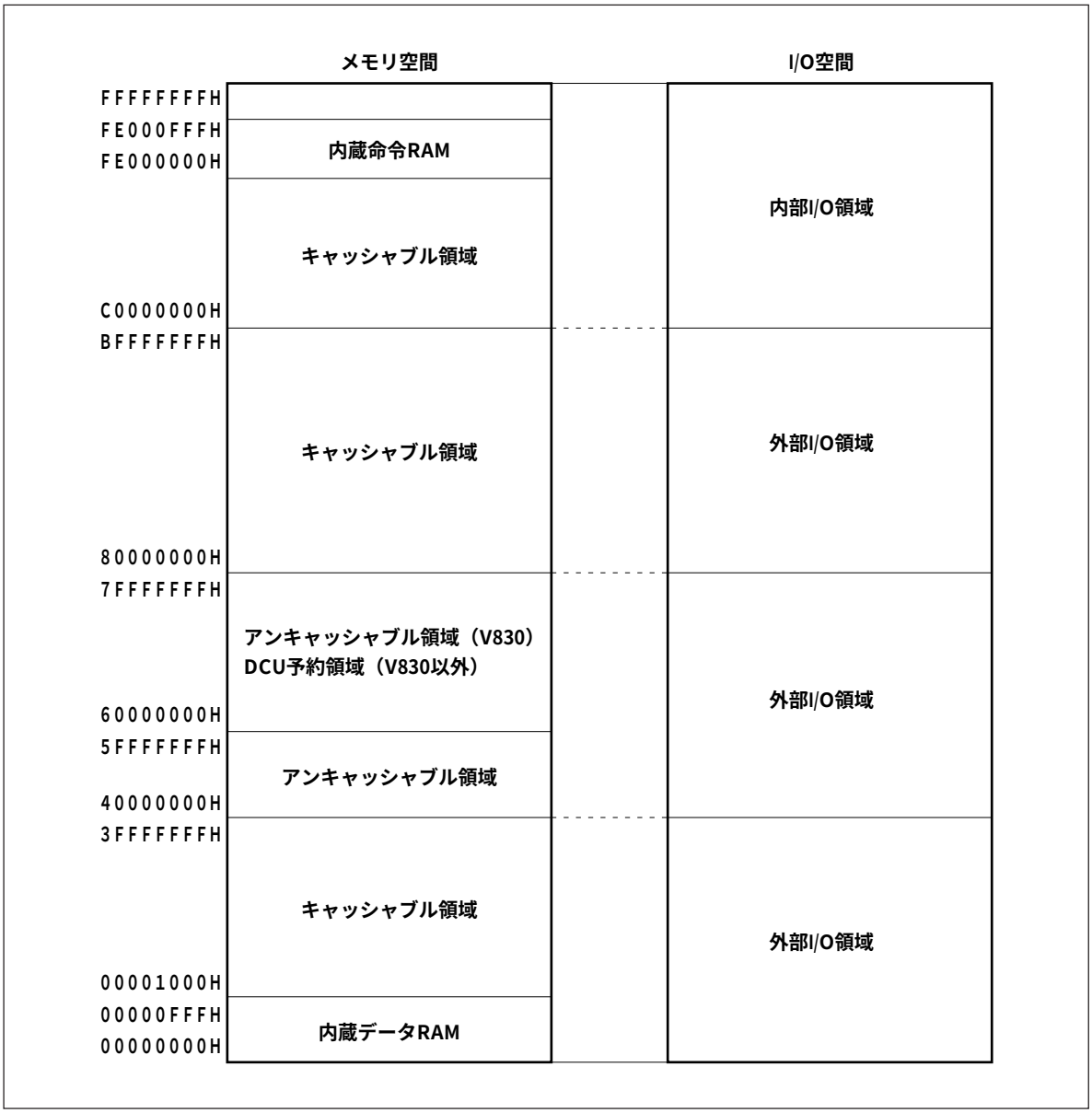
〔メ　モ〕

第４章　アドレス空間

V830ファミリはメモリ空間，I/O空間に対して，それぞれ4 Gバイトのリニアなアドレス空間をサポートしています。V830ファミリからメモリ空間に対して32ビットのアドレスを出力し，アドレスの番地は最大$2^{32}-1$になります。I/O空間に対しても32ビットのアドレスを出力します。

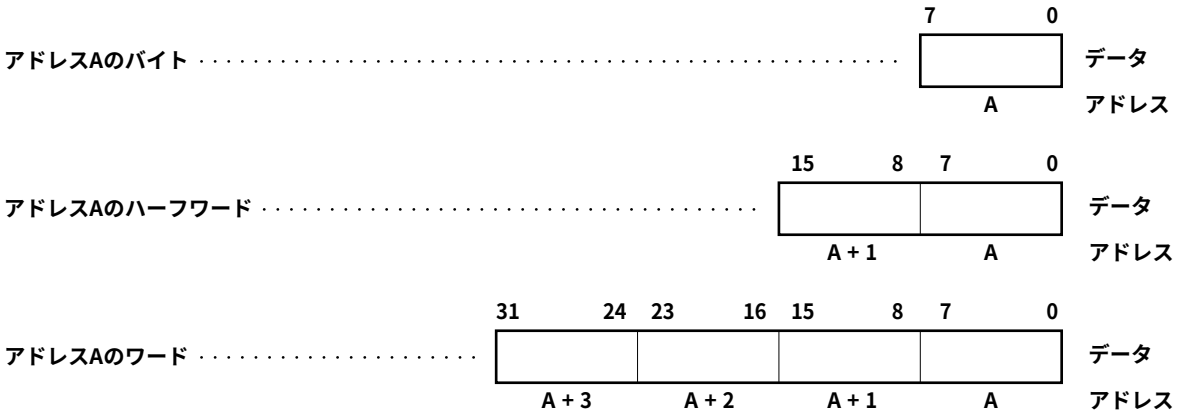
V830ファミリのメモリ・マップを図４－１に示します。

図４－１　メモリ・マップ



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と定義しています。また，複数の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を持つように定義しています。（リトル・エンディアン）。

V830ファミリでは２バイト構成のデータ形式をハーフワード，４バイト構成のデータ形式をワードと呼びます。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　アドレシング・モード

V830ファミリのアドレス生成には次の２種類があります。

・命令アドレス（分岐を伴う命令が使用します。）
・オペランド・アドレス（データをアクセスする命令が使用します。）

４.１.１　命令アドレス

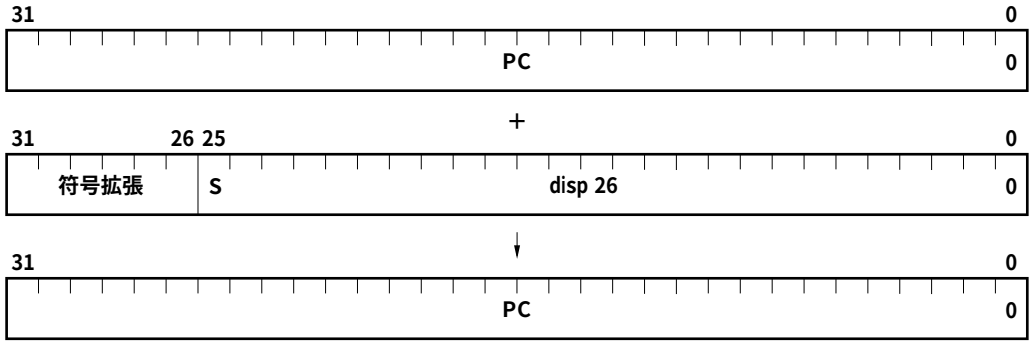
命令アドレスはプログラム・カウンタ（PC）の内容によって決定され，命令を実行するごとにフェッチする命令のバイト数に応じて自動的にインクリメント（＋2/＋4）されます。また，分岐命令を実行するときは，次に示すアドレシングによって分岐先アドレスをPCにセットします。

（1）レラティブ・アドレシング（PC相対）

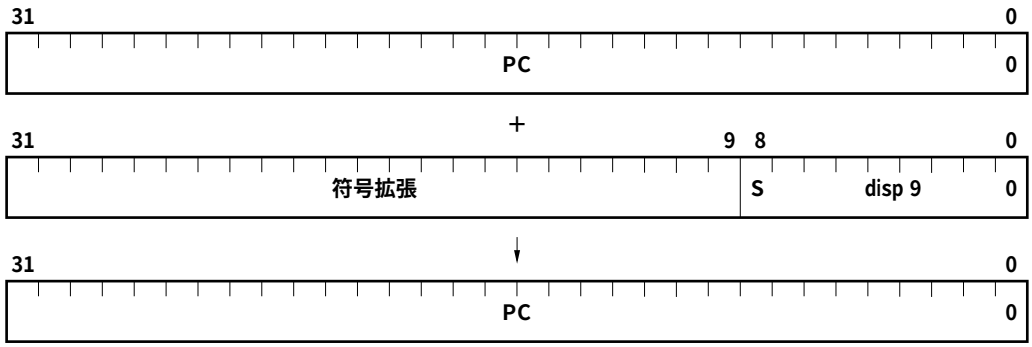
プログラム・カウンタ（PC）に，命令コードの符号付き９ビットまたは26ビット・データ（ディスプレースメント：disp）を加算します。このとき，ディスプレースメントは，２の補数データとして扱われ，それぞれビット８とビット25が符号ビットになります。

JR，JAL，Bcond，ABcond命令がこのアドレシングを使用します。

JR, JAL命令のアドレシング形式



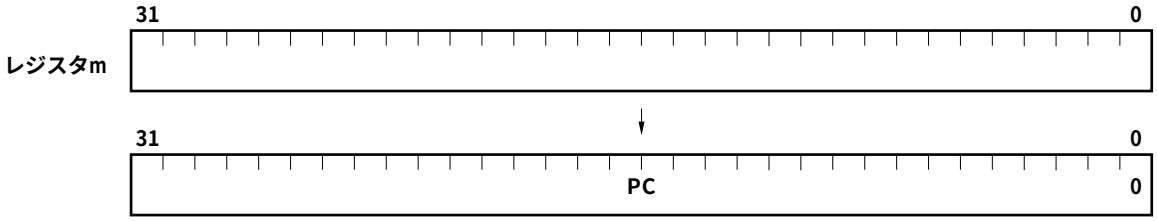
Bcond, ABcond命令のアドレシング形式



（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命令がこのアドレシングを使用します。



4.1.2　オペランド・アドレス

（1）レジスタ・アドレシング

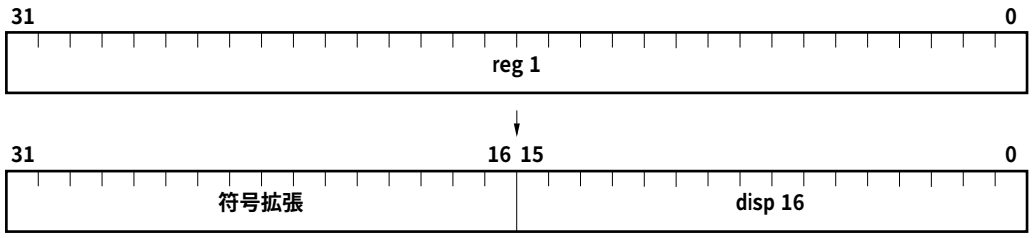
汎用レジスタ指定フィールドにより指定される汎用レジスタをオペランドとしてアクセスするアドレシングです。このアドレシングはオペランド形式がreg1またはreg2である命令が対象となります。

（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である命令が対象となります。

（3）ベースド・アドレシング

命令語中のアドレス指定コードで示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disp16[reg1]である命令が対象となります。



第５章　命　　令

５.１　命令フォーマット

V830ファミリの命令は16ビット・フォーマットと32ビット・フォーマットの２種類があります。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，I/O操作，16ビット・イミーディエトを扱う命令，ジャンプ・アンド・リンクなどがあります。

なお，一部の命令では未使用フィールドが発生しますが，それらは将来の拡張用で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。

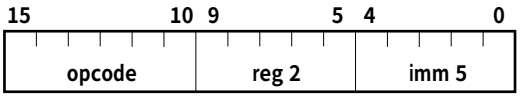
●各命令形式の下位部分（ビット０を含む）→下位アドレス側
●各命令形式の上位部分（ビット15またはビット31を含む）→上位アドレス側

（１）reg-reg命令形式［FORMAT Ｉ］

６ビットのオペコード・フィールドとオペランド指定に２つの汎用レジスタ指定フィールドを持つ命令形式。16ビット長命令。

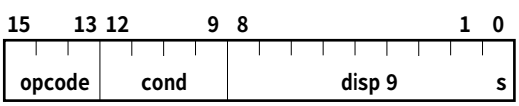
（２）imm-reg命令形式［FORMAT Ⅱ］

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



（３）条件分岐命令形式［FORMATIII］

３ビットのオペコード・フィールドと４ビットの条件コード，９ビットの分岐ディスプレースメント・フィールド（ビット０は０と見なし指定しません），１ビットのサブオペコードを持つ命令形式。16ビット長命令。

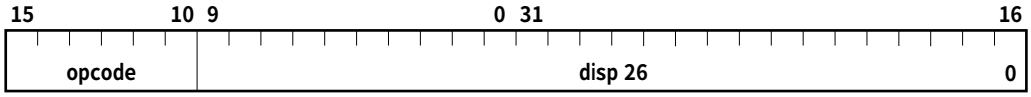


s = 0 : Bcond
s = 1 : ABcond

s : sub-opcode

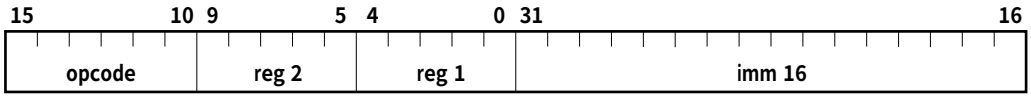
（4）中距離ジャンプ命令形式［FORMAT IV］

６ビットのオペコード・フィールドと26ビットのディスプレースメント（ただし，最下位ビットは０）を持つ中距離分岐命令形式。32ビット長命令。



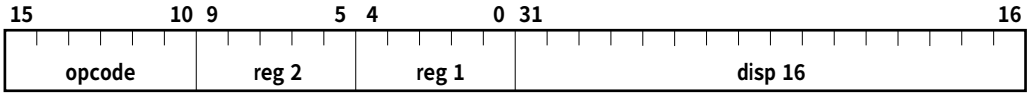
（5）３オペランド命令形式［FORMAT V］

６ビットのオペコード・フィールドと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・フィールドを持つ命令形式。32ビット長命令。



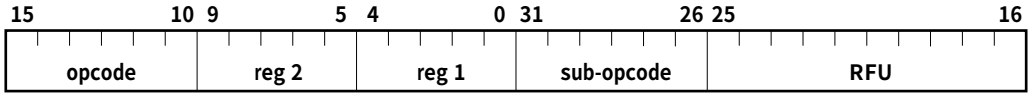
（6）ロード／ストア命令形式［FORMAT VI］

６ビットのオペコード・フィールドと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を持つ命令形式。32ビット長命令。



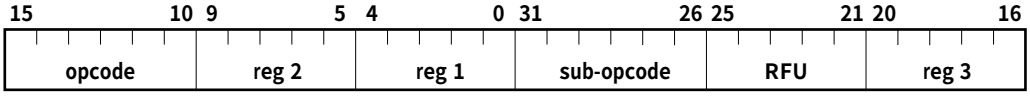
（7）拡張命令形式［FORMAT VII］

６ビットのオペコード・フィールドと２つの汎用レジスタ指定フィールド，6ビットのサブオペコードを持つ命令形式。32ビット長命令。



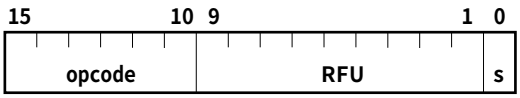
（8）３レジスタ・オペランド命令形式［FORMAT VIII］

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



（9）オペランドなし命令形式［FORMAT IX］

６ビットのオペコード・フィールドと１ビットのサブオペコード・フィールドを持つ命令形式。16ビット長命令。



s : sub-opcode

5.2 命令の概要

（1）ロード／ストア命令 … メモリ-レジスタ間のデータ転送命令です。

ニモニック	命令の意味
LD.B	Load Byte
LD.H	Load Halfword
LD.W	Load Word
ST.B	Store Byte
ST.H	Store Halfword
ST.W	Store Word
BILD	Block Instruction Load to internal memory
BIST	Block Instruction of internal memory Store
BDLD	Block Data Load to internal memory
BDST	Block Data of internal memory Store

（2）入出力命令 … I/O-レジスタ間のデータ転送命令です。

ニモニック	命令の意味
IN.B	Input Byte from Port
IN.H	Input Halfword from Port
IN.W	Input Word from Port
OUT.B	Output Byte to Port
OUT.H	Output Halfword to Port
OUT.W	Output Word to Port

（3）算術演算命令 … 加減乗除算，データ比較，レジスタ間データ転送命令です。

ニモニック	命令の意味
MOV	Move
MOVHI	Add
ADD	Add
ADDI	Add Immediate
MOVEA	Add
SUB	Subtract
MUL	Multiply
MULU	Multiply Unsigned
DIV	Divide
DIVU	Divide Unsigned
CMP	Compare
SETF	Set Flag Condition
MIN3	Minimum
MAX3	Maximum

（4）積和／飽和演算命令

ニモニック	命令の意味
MUL3	Multiply
MAC3	Multiply and Accumulate
MULI	Multiply Immediate
MACI	Multiply and Accumulate Immediate
MULT3	Multiply with Truncation
MACT3	Multiply and Accumulate with Truncation
SATADD3	Saturated Add
SATSUB3	Saturated Subtract

（5）論理演算命令

ニモニック	命令の意味
OR	Or
ORI	Or Immediate
AND	And
ANDI	And Immediate
XOR	Exclusive Or
XORI	Exclusive Or Immediate
NOT	Not
SHL	Shift Logical Left
SHR	Shift Logical Right
SAR	Shift Arithmetic Right
SHLD3	Shift Left Double word
SHRD3	Shift Right Double word

（6）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および分岐履歴を利用した高速分岐命令です。

ニモニック	命令の意味
JMP	Jump Register
JR	Jump Relative
JAL	Jump and Link
ABGT	Advanced Branch on Greater than signed
BGT	Branch on Greater than signed
ABGE	Advanced Branch on Greater than or Equal signed
BGE	Branch on Greater than or Equal signed
ABLT	Advanced Branch on Less than signed
BLT	Branch on Less than signed
ABLE	Advanced Branch on Less than or Equal signed
BLE	Branch on Less than or Equal signed

ニモニック	命令の意味
ABH	Advanced Branch on Higher
BH	Branch on Higher
ABNL	Advanced Branch on Not Lower
BNL	Branch on Not Lower
ABL	Advanced Branch on Lower
BL	Branch on Lower
ABNH	Advanced Branch on Not Higher
BNH	Branch on Not Higher
ABE	Advanced Branch on Equal
BE	Branch on Equal
ABNE	Advanced Branch on Not Equal
BNE	Branch on Not Equal
ABV	Advanced Branch on Overflow
BV	Branch on Overflow
ABNV	Advanced Branch on No Overflow
BNV	Branch on No Overflow
ABN	Advanced Branch on Negative
BN	Branch on Negative
ABP	Advanced Branch on Positive
BP	Branch on Positive
ABC	Advanced Branch on Carry
BC	Branch on Carry
ABNC	Advanced Branch on No Carry
BNC	Branch on No Carry
ABZ	Advanced Branch on Zero
BZ	Branch on Zero
ABNZ	Advanced Branch on Not Zero
BNZ	Branch on Not Zero
ABR	Advanced Branch Always（無条件）
BR	Branch Always（無条件）
NOP	Not Always（分岐しない）

（7）特殊命令 …　（1）-（6）までに含まれないそのほかの特殊命令です。

ニモニック	命令の意味
LDSR	Load to System Register
STSR	Store contents of System Register
TRAP	Trap
RETI	Return from Trap or Interrupt
CAXI	Compare and Exchange Interlocked
HALT	Halt
BRKRET	Break Return
EI	Enable Interrupt
DI	Disable Interrupt
STBY	Standby

5.3 命令セット

命令記述例

命令のニモニック	命令の意味（英語） 命令の意味（日本語）

【命 令 形 式】　命令を記述するときの書き方，オペランドを示します。オペランドの記述で使用され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します。）
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します。一部ソース・レジスタとしても使用します。）
reg3	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します。一部ソース・レジスタとしても使用します。）
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 adr	トラップ・ベクタに対応するトラップ・ハンドラ・アドレス

【オペレーション】　命令の機能を示します。使用される略号は次のとおりです。

略　号	意　味
←	代入
‖	ビット連結
GR[x]	汎用レジスタx
SR[x]	システム・レジスタx
sign-extend(x)	値xをワード長まで符号拡張します。
zero-extend(x)	値xをワード長までゼロ拡張します。
Load-Memory(x, y)	アドレスxからサイズyのデータを読み出します。
Store-Memory(x, y, z)	アドレスxにデータyをサイズzで書き込みます。
Input-Port(x, y)	ポート・アドレスxからサイズyのデータを読み出します。
Output-Port(x, y, z)	ポート・アドレスxにデータyをサイズzで書き込みます。
adr	32ビット符号なしアドレス

【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。

【フ　　ラ　　グ】　フラグの動きを示します。

略　　号	意　　味
－	変化しないことを示します。
0	0に変化することを示します。
1	1に変化することを示します。

【命　　令】　命令機能を示します。

【説　　明】　命令の動作説明を示します。

【補　　足】　命令の補足説明を示します。

【例　　外】　命令を実行すると起こる可能性のある例外について説明します。

ABcond

Advanced Branch on condition
高速分岐機能付き条件分岐

【命 令 形 式】 ABcond　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	9	8	1	0
100$$$$		disp9		1

$$$$フィールドは条件を示します（**表５－１**参照）。

【フ ラ グ】 CY －
OV －
S －
Z －

【命　　令】 ABcond　Advanced Branch on condition Code with 9bit-displacement

【説　　明】 命令で指定された条件フラグがテストされ，もし条件が満たされていると，現PCと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をPCに設定し，制御を移し，分岐履歴を残します。
分岐履歴のある命令を実行するときは，高速に分岐することができます。なお，分岐履歴は１つしか保持できないので，分岐履歴がある命令は最後に実行したABcond命令だけです。
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。計算に使用される現PCはABcond命令自身の先頭アドレスであるため，ディスプレースメントの値が０のときは，分岐先はこの命令自身になります。

【補　　足】 以下の条件が満たされたとき，分岐履歴を消去します。

- リセット時
- BILD命令（外部メモリ→内蔵命令RAM転送）の実行
- IRAMR（内蔵命令RAMの変更）レジスタの書き換え
- 命令キャッシュのクリア
- 命令キャッシュ・タグの書き換え

次にプログラムをロードしたときの注意事項を示します。

・内蔵命令RAMにプログラムをロードするには，BILD命令でしか行えないので，自動的に分岐履歴は消去されます。
・キャッシャブル領域にプログラムをロードしたときは，命令キャッシュをクリア（キャッシュ・メモリ・コントロール・レジスタ（CMCR）のICCビットを１）すると，分岐履歴は消去されます。
・アンキャッシャブル領域にプログラムをロードしたときは，ユーザがABR命令を実行するなどして，今までの分岐履歴を消去してください。消去しない場合，以前の分岐履歴が書き換えたプログラム領域を指していると，誤った分岐が発生します。

【例　　外】　なし

表５－１　条件分岐命令一覧（ABcond命令）

命　令		ビット12-9	条件フラグの状態	分　岐　条　件
整数	ABGT	1111	((S xor OV) or Z) = 0	Greater than signed
	ABGE	1110	(S xor OV) = 0	Greater than or equal signed
	ABLT	0110	(S xor OV) = 1	Less than signed
	ABLE	0111	((S xor OV) or Z) = 1	Less than or equal signed
符号なし整数	ABH	1011	(CY or Z) = 0	Higher（Greater than）
	ABNL	1001	CY = 0	Not lower（Greater than or equal）
	ABL	0001	CY = 1	Lower（Less than）
	ABNH	0011	(CY or Z) = 1	Not higher（Less than or equal）
共通	ABE	0010	Z = 1	Equal
	ABNE	1010	Z = 0	Not equal
その他	ABV	0000	OV = 1	Overflow
	ABNV	1000	OV = 0	No overflow
	ABN	0100	S = 1	Negative
	ABP	1100	S = 0	Positive
	ABC	0001	CY = 1	Carry
	ABNC	1001	CY = 0	No carry
	ABZ	0010	Z = 1	Zero
	ABNZ	1010	Z = 0	Not zero
	ABR	0101	－	Always（無条件）

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）

15 10	9 5	4 0
000001	reg2	reg1

（2）

15 10	9 5	4 0
010001	reg2	imm5

【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 GR[reg2]が負のとき1，そうでないとき0
Z GR[reg2]が0のとき1，そうでないとき0

【命 令】 （1）ADD Add Register
（2）ADD Add Immediate(5-bit)

【説 明】 （1）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。reg1は影響を受けません。
（2）reg2のワード・データに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加算し，その結果をreg2に格納します。

【例 外】 なし

ADDI

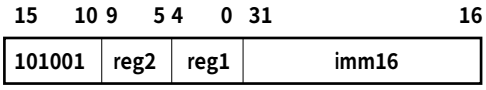
Add Immediate
加算

【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 V

【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　GR[reg2]が負のとき１，そうでないとき０
Z　GR[reg2]が０のとき１，そうでないとき０

【命　　令】　ADDI　Add Immediate(16-bit)

【説　　明】　reg1のワード・データに16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加算し，その結果をreg2に格納します。reg1は影響を受けません。

【例　　外】　なし

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	10 9	5 4	0
001101	reg2	reg1	

【フ　ラ　グ】　CY　－
OV　0
S　GR[reg2]が負のとき１，そうでないとき０
Z　GR[reg2]が０のとき１，そうでないとき０

【命　　令】　AND　And

【説　　明】　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。reg1は影響を受けません。

【例　　外】　なし

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 V

【オ ペ コ ー ド】

15	10 9	5 4	0 31	0
101101	reg2	reg1	imm16	

【フ　ラ　グ】　CY　－
OV　0
S　0
Z　GR[reg2]が0のとき1，そうでないとき0

【命　　令】　ANDI　And Immediate（16-bit）

【説　　明】　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reg2に格納します。reg1は影響を受けません。

【例　　外】　なし

Bcond

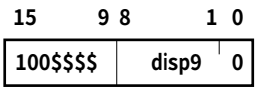
Branch on Condition
条件分岐

【命 令 形 式】 Bcond disp9

【オペレーション】 if condition art satisfied
then PC ← PC＋(sign-extend)disp9

【フォーマット】 Format III

【オ ペ コ ー ド】



$$$$フィールドは条件を示します。（表5－2参照）。

【フ ラ グ】 CY －
OV －
S －
Z －

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

【説 明】 命令で指定される条件フラグがテストされ，もし条件が満たされているならば，現PCと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をPCに設定し，制御を移します。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PCとは，Bcond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。

【例 外】 なし

表５－２　条件分岐命令一覧（Bcond命令）

命　令		ビット12-9	条件フラグの状態	分　岐　条　件
整数	BGT	1111	((S xor OV) 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	((S xor OV) 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）
	NOP	1101	－	Not Always（分岐しない）

BDLD

Block-Data Load to internal memory
内蔵データRAM へのブロック転送

【命 令 形 式】　BDLD　[reg1], [reg2]

【オペレーション】　Store-internal-data-Memory（GR[reg2], Load-Memory（GR[reg1], 16byte）, 16byte）

【フォーマット】　Format VII

【オ ペ コ ー ド】

15　10	9　5	4　0	31　26	25　16
111110	reg2	reg1	100001	RFU

【フ　ラ　グ】　CY　－
OV　－
S　－
Z　－

【命　　令】　BDLD　Block-Data Load to internal memory

【説　　明】　外部メモリから４ワード（16バイト）のデータを内蔵データRAMに転送します。reg1は外部メモリのアドレスを，reg2は内蔵データRAMのオフセット・アドレスを示します。
reg1，reg2の値（アドレス）はビット0-3が０でなければなりません。

【例　　外】　なし

BDST

Block-Data of internal memory Store
内蔵データRAMからのブロック転送

【命 令 形 式】　BDST　[reg2], [reg1]

【オペレーション】　Store-Memory（GR[reg1], Load-internal-data-Memory（GR[reg2], 16byte), 16byte）

【フォーマット】　Format VII

【オ ペ コ ー ド】

15 10	9 5	4 0	31 26	25 16
111110	reg2	reg1	100011	RFU

【フ　ラ　グ】　CY　－
OV　－
S　－
Z　－

【命　　令】　BDST　Block-Data of internal memory Store

【説　　明】　内蔵データRAMから４ワード（16バイト）のデータを外部メモリに転送します。reg2は内蔵データRAMのオフセット・アドレスを，reg1は外部メモリのアドレスを示します。
reg1，reg2の値（アドレス）はビット0-3が０でなければなりません。

【例　　外】　なし

BILD

Block-Instruction Load to internal memory
内蔵命令RAMへのブロック転送

【命 令 形 式】 BILD [reg1], [reg2]

【オペレーション】 Store-internal-instruction-Memory(GR[reg2], Load-Memory(GR[reg1], 16byte), 16byte)

【フォーマット】 Format VII

【オ ペ コ ー ド】

15 10	9 5	4 0	31 26	25 16
111110	reg2	reg1	100000	RFU

【フ ラ グ】 CY －
OV －
S －
Z －

【命 令】 BILD Block-Instruction Load to internal memory

【説 明】 外部メモリから４ワード（16バイト）のデータを内蔵命令RAMに転送します。reg1は外部メモリのアドレスを，reg2は内蔵命令RAMのオフセット・アドレスを示します。reg1, reg2の値（アドレス）はビット0-3が０でなければなりません。

【補 足】 BILD命令を実行すると，ABcond命令（高速分岐）のための分岐履歴を消去します。

【例 外】 なし

BIST

Block-Instruction of internal memory Store
内蔵命令RAMからのブロック転送

【命 令 形 式】　BIST　[reg2], [reg1]

【オペレーション】　Store-Memory(GR[reg1], Load-internal-instruction-Memory(GR[reg2], 16byte), 16byte)

【フォーマット】　Format VII

【オ ペ コ ー ド】

15 10	9 5	4 0	31 26	25 16
111110	reg2	reg1	100010	RFU

【フ ラ グ】　CY －
OV －
S －
Z －

【命　　令】　BIST　Block-Instruction of internal memory Store

【説　　明】　内蔵命令RAMから４ワード（16バイト）のデータを外部メモリに転送します。reg2は内蔵命令RAMのオフセット・アドレスを，reg1は外部メモリのアドレスを示します。reg1, reg2の値（アドレス）はビット0-3が０でなければなりません。

【例　　外】　なし

BRKRET

Break Return

致命的例外からの復帰

【命 令 形 式】 BRKRET

【オペレーション】 PC ← DPC

PSW ← DPSW

【フォーマット】 Format IX

【オ ペ コ ー ド】

15 10	9 1	0
011001	RFU	1

【フ ラ グ】 CY －

OV －

S －

Z －

【命 令】 BRKRET Break Return

【説 明】 システム・レジスタDPC, DPSWからPC, PSWを取り出し，致命的例外から復帰する命令です。

この命令の動作は，DPC, DPSWから復帰PC, PSWを取り出します。取り出した復帰PC, PSWをPC, PSWに設定し，PCにジャンプします。

【補 足】 この命令は致命的例外処理からの復帰のときだけ使用してください。

【例 外】 なし

CAXI

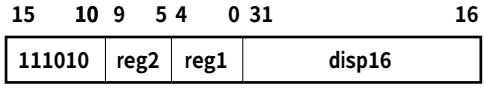
Compare and Exchange Interlocked
比較と交換

【命 令 形 式】　CAXI　disp16[reg1], reg2

【オペレーション】
```
locked
adr ← GR[reg1] + (sign-extend)disp16
tmp ← Load-Memory(adr, Word)
if GR[reg2] = tmp(比較；result ← GR[reg2] −tmp)
    then  Store-Memory(adr, GR[30], Word)
          GR[reg2] ← tmp
    else  Store-Memory(adr, tmp, Word)
          GR[reg2] ← tmp
unlocked
```

【フォーマット】　Format VI

【オペコード】



【フ ラ グ】　CY　比較でMSBからボローがあれば１，そうでないときは０
OV　比較でオーバフローが起こったとき１，そうでないとき０
S　比較結果が負のとき１，そうでないとき０
Z　比較結果が０のとき１，そうでないとき０

【命　　令】　CAXI　Compare and Exchange Interlocked

【説　　明】　この命令はマルチプロセッサ構成のシステムにおけるプロセッサ間同期命令で，disp16[reg1]で指定されるデータは，同期をとるためのデータ（たとえばロック・ワード）です。
命令を実行する前の状態は，次のとおりです。

新しく設定するロック・ワード	GR[30]
前回読み出したときのロック・ワード	GR[reg2]
ロック・ワード	GR[reg1]＋(sign-extend)disp16で指定されるアドレスのワード・データ。アドレスのビット０およびビット１は０にマスクされます。

このような状況のもとで，CAXI命令は，次のような動作を行います。

（１）バスをロックしてほかのプロセッサがアクセスするのを防ぎます。
（２）ロック・ワードをフェッチします。
（３）ロック・ワードの値と前回読み出したときのロック・ワードの値を比較します。
　　比較した結果をフラグに反映します。
（４）一致している場合は，これは前にアクセスしたときと状況が変わっていない（ほかのプロセッサ上のプログラムがアクセスのためにロックしていない）ことを示します。
　　このＣＡＸＩ命令実行で状況が変化するため，新しく設定するロック・ワード（GR[30]）をロック・ワードに設定します。
（５）不一致の場合は，前にアクセスしたときから状況が変化している（ほかのプロセッサ上のプログラムがアクセスのためにロックした）ことを示しているため，ロック・ワードがどのような状態になっているかを知るために，GR[reg2]にそのロック・ワードを設定します。
（６）バス・ロックを解除します。

【例　　外】　なし

CMP

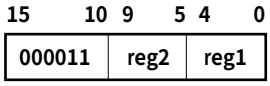
Compare
比較

【命 令 形 式】 （1）CMP　reg1, reg2
（2）CMP　imm5, reg2

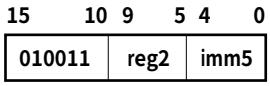
【オペレーション】 （1）result ← GR[reg2] − GR[reg1]
（2）result ← GR[reg2] − sign-extend(imm5)

【フォーマット】 （1）Format I
（2）Format II

【オ ペ コ ー ド】 （1）



（2）



【フ ラ グ】 CY　MSBからのボローがあれば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　resultが負のとき1，そうでないとき0
Z　resultが0のとき1，そうでないとき0

【命　　令】 （1）CMP　Compare Register
（2）CMP　Compare Immediate(5-bit)

【説　　明】 （1）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。reg1, reg2は影響を受けません。
（2）reg2のワード・データと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・データからワード長まで符号拡張した5ビット・イミーディエトを減算することで行います。reg2は影響を受けません。

【例　　外】 なし

DI

Disable Interrupt
マスカブル割り込み禁止

【命 令 形 式】 DI

【オペレーション】 PSW中のIDビットをセットし，マスカブル割り込みを禁止します。

【フォーマット】 Format II

【オ ペ コ ー ド】

15	10	9	0
011110		RFU	

【フ ラ グ】 CY －
OV －
S －
Z －

【命　　令】 DI　Disable Interrupt

【説　　明】 マスカブル割り込みを禁止します。PSWのIDビットを１にします。
LDSR命令を使用して，PSWのIDビットを１にする動作と同じです。

【補　　足】 DI命令ではノンマスカブル割り込みは禁止できません。その場合はLDSR命令を使用して，PSWを書き換えてください。

【例　　外】 なし

DIV

Divide
（符号付き）除算

【命 令 形 式】　DIV reg1, reg2

【オペレーション】　GR[30]　← GR[reg2]MOD GR[reg1]（符号付き）
GR[reg2] ← GR[reg2]÷GR[reg1]（符号付き）

【フォーマット】　Format I

【オ ペ コ ー ド】

15 10	9 5	4 0
001001	reg2	reg1

【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　GR[reg2]が負のとき１，そうでないとき０
Z　GR[reg2]が０のとき１，そうでないとき０

【命　　令】　DIV　Divide

【説　　明】　reg2のワード・データをreg1のワード・データで除算（符号付き）し，その商をreg2に，剰余をr30にそれぞれ格納します。除算は剰余の符号が被除数の符号と一致するように行われます。reg1は影響を受けません。reg2にr30を指定した場合は，商をr30に格納します。オーバフローは，負の最大数（80000000H）を－1（FFFFFFFFH）で除算したときに設定されます。このときreg2には負の最大数，r30には０が格納されます。

【例　　外】　ゼロ除算例外

【注　　意】　もし，reg1のワード・データがゼロである場合には，ゼロ除算例外が発生し，例外処理ハンドラにトラップします。この場合，reg2, r30およびフラグは変化しません。

DIVU

Divide Unsigned
符号なし除算

【命 令 形 式】　DIVU　reg1, reg2

【オペレーション】　GR[30]　← GR[reg2] MOD GR[reg1]（符号なし）
GR[reg2] ← GR[reg2] ÷ GR[reg1]（符号なし）

【フォーマット】　Format I

【オ ペ コ ー ド】

15　10	9　5	4　0
001011	reg2	reg1

【フ　ラ　グ】　CY　－
OV　0
S　GR[reg2]が負のとき１，そうでないとき０
Z　GR[reg2]が０のとき１，そうでないとき０

【命　　令】　DIVU　Divide Unsigned

【説　　明】　reg2のワード・データをreg1のワード・データでともに符号なしデータとして除算し，その商をreg2に，剰余をr30にそれぞれ格納します。reg1は影響を受けません。reg2にr30を指定した場合は，商をr30に格納します。フラグの設定は，結果が符号付きデータであるかのように行われます。

【例　　外】　ゼロ除算例外

【注　　意】　もし，reg1のワード・データがゼロである場合には，ゼロ除算例外が発生し，例外処理ハンドラにトラップします。この場合，reg2, r30およびフラグは変化しません。

EI

Enable Interrupt
マスカブル割り込み許可

【命 令 形 式】　EI

【オペレーション】　PSW中のIDビットをクリアし，マスカブル割り込みを許可します。

【フォーマット】　Format II

【オ ペ コ ー ド】

15	10	9	0
010110		RFU	

【フ　ラ　グ】　CY　－
OV　－
S　－
Z　－

【命　　令】　EI　Enable Interrupt

【説　　明】　マスカブル割り込みを許可します。PSWのIDビットを0にします。
LDSR命令を使用して，PSWのIDビットを0にする動作と同じです。

【補　　足】　EI命令ではノンマスカブル割り込みは許可できません。その場合はLDSR命令を使用して，PSWを書き換えてください。

【例　　外】　なし

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 IX

【オ ペ コ ー ド】

15	10	9	1	0
011010	RFU			0

【フ ラ グ】　CY －
OV －
S －
Z －

【命　　令】　HALT　Halt

【説　　明】　CPUは停止して，スリープ・モードに遷移します。

【例　　外】　なし

IN

Input From Port
ポート入力

【命 令 形 式】 （1）IN.B　disp16[reg1], reg2
（2）IN.H　disp16[reg1], reg2
（3）IN.W　disp16[reg1], reg2

【オペレーション】 （1）adr ← GR[reg1] + (sign-extend)disp16
GR[reg2] ←(zero-extend)— Input-Port(adr, Byte)
（2）adr ← GR[reg1] + (sign-extend)disp16
GR[reg2] ←(zero-extend)— Input-Port(adr, Halfword)
（3）adr ← GR[reg1] + (sign-extend)disp16
GR[reg2] ←— Input-Port(adr, Word)

【フォーマット】 Format VI

【オ ペ コ ー ド】

15 10	9 5	4 0	31 16
1110＊$	reg2	reg1	disp16

(＊$:00=(1),01=(2),11=(3))

【フ ラ グ】 CY －
OV －
S －
Z －

【命 令】 （1）IN.B　Input Byte from Port
（2）IN.H　Input Halfword from Port
（3）IN.W　Input Word from Port

【説 明】 （1）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ビット符号なしポート・アドレスを生成します。生成したポート・アドレスからバイト・データを読み出し，ワード長までゼロ拡張しreg2に格納します。
（2）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ビット符号なしポート・アドレスを生成します。生成したポート・アドレスからハーフワード・データを読み出し，ワード長までゼロ拡張しreg2に格納します。32ビット符号なしアドレスのビット0は0にマスクされます。

（3）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ビット符号なしポート・アドレスを生成します。生成したポート・アドレスからワード・データを読み出し，reg2に格納します。32ビット符号なしアドレスのビット０およびビット１は０にマスクされます。

【例　　外】　なし

JAL

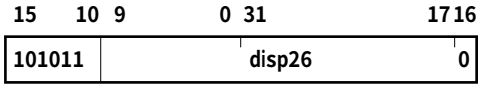
Jump and Link
ジャンプ・アンド・リンク

【命 令 形 式】　JAL　disp26

【オペレーション】　GR[31] ← PC＋4
PC　← PC＋(sign-extend)disp26

【フォーマット】　Format IV

【オ ペ コ ー ド】



【フ　ラ　グ】　CY　－
OV　－
S　－
Z　－

【命　　令】　JAL　Jump and Link

【説　　明】　現PCに4を加算した値をr31に退避し，現PCとワード長まで符号拡張した26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。26ビット・ディスプレースメントの最下位ビットは0にマスクされます。なお，計算に使用される現PCとは，JAL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。

【例　　外】　なし

JMP

Jump register

無条件分岐（レジスタ間接）

【命 令 形 式】　JMP　[reg1]

【オペレーション】　PC ← GR [reg1]

【フォーマット】　Format Ⅰ

【オ ペ コ ー ド】

15	10 9	5 4	0
000110	−	reg1	

【フ　ラ　グ】　CY　−

OV　−

S　−

Z　−

【命　　令】　JMP　Jump register

【説　　明】　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。

【例　　外】　なし

JR

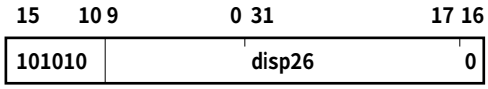
Jump Relative
無条件分岐（PC相対）

【命 令 形 式】　JR　disp26

【オペレーション】　PC ← PC＋(sign-extend)disp26

【フォーマット】　Format IV

【オ ペ コ ー ド】



【フ　ラ　グ】　CY　－
OV　－
S　－
Z　－

【命　　令】　JR　Jump Relative

【説　　明】　現PCとワード長まで符号拡張した26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。26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PCとは，JMP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。

【例　　外】　なし

LD

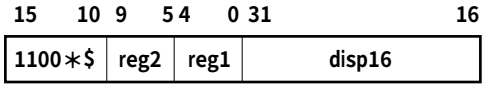
Load
ロード

【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 VI

【オ ペ コ ー ド】



15 10	9 5	4 0	31 16
1100＊$	reg2	reg1	disp16

(＊$:00=(1),01=(2),11=(3))

【フ ラ グ】 CY －
OV －
S －
Z －

【命 令】 （1）LD.B　Load Byte
（2）LD.H　Load Halfword
（3）LD.W　Load Word

【説 明】 （1）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ビット符号なし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）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ビット符号なしアドレスを生成します。生成したアドレスからハーフワード・データを読み出し，ワード長まで符号拡張し，reg2に格納します。32ビット符号なしアドレスのビット0は0にマスクされます。

（３）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ビット符号なしアドレスを生成します。生成したアドレスからワード・データを読み出し，reg2に格納します。32ビット符号なしアドレスのビット０およびビット１は０にマスクされます。

【例　　外】　なし

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 II

【オ ペ コ ー ド】

15	10	9	5	4	0
011100		reg2		regID	

【フ ラ グ】　CY　－　（補足参照）
OV　－　（補足参照）
S　－　（補足参照）
Z　－　（補足参照）

【命　　令】　LDSR　Load to System Register

【説　　明】　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。reg2は影響を受けません。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，書き込み不可能なシステム・レジスタに対するLDSR命令を実行した場合の動作は保証しません。

【例　　外】　なし

【補　　足】　システム・レジスタ番号（regID）が5（PSW）の場合は，各フラグにはreg2の対応するビットの値が設定されます。

MAC3

Multiply and Accumulate
（符号付き32ビット飽和）積和

【命 令 形 式】　MAC3　reg1, reg2, reg3

【オペレーション】　GR[reg3] ← saturate(GR[reg3]＋GR[reg2]×GR[reg1])

【フォーマット】　Format VIII

【オ ペ コ ー ド】

15 10	9 5	4 0	31 26	25 21	20 16
111110	reg2	reg1	011101	RFU	reg3

【フ ラ グ】　CY －
OV －
S －
Z －

【命　　令】　MAC3　Multiply and Accumulate

【説　　明】　reg1のワード・データとreg2のワード・データを32ビット符号付き整数として乗算し，その結果とreg3のデータとを符号付き整数として加算します。演算結果が32ビット符号付き整数として表現できる範囲を越えるとオーバフローとみなします（乗算結果64ビットのうち下位32ビットが有効）。

[オーバフローしない場合]

加算結果をreg3に格納します。

[オーバフローした場合]

SATフラグをセット（1）し，加算結果が正ならば正の最大値（7FFFFFFFH）を，負ならば負の最大値（80000000H）をreg3に格納します。

reg1, reg2は影響を受けません。

【補　　足】　MAC3命令の入力オペランドreg3のタイミングには制限があります。MAC3命令の３サイクル前までに，reg3を更新する命令がない場合には，１サイクル停止（ストール）してからMAC3命令を開始します。

フラグ（CY, OV, S, Z）は変化しません。SATフラグは累積フラグで，飽和演算命令で演算結果が一度飽和するとセット（1）し，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フラグをリセットするには，LDSR命令でPSWに書き込みを行ってください。

【例　　外】　なし

MACI

Multiply and Accumulate Immediate
（符号付き32ビット飽和）積和

【命 令 形 式】 MACI　imm16, reg1, reg2

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

【フォーマット】 Format V

【オ ペ コ ー ド】

15　10	9　5	4　0	31　16
110110	reg2	reg1	imm16

【フ ラ グ】 CY －
OV －
S －
Z －

【命　令】 MACI　Multiply and Accumulate Immediate

【説　明】 reg1のワード・データとイミーディエト・データ（16ビットを32ビットに符号拡張）を符号付き整数として乗算し，その結果とreg2のデータを符号付き整数として加算します。演算結果が32ビット符号付き整数として表現できる範囲を越えるとオーバフローとみなします（乗算結果64ビットのうち下位32ビットが有効）。

[オーバフローしない場合]

演算結果をreg2に格納します。

[オーバフローした場合]

SATフラグをセット（1）し，加算結果が正ならば正の最大値（7FFFFFFFH）を，負ならば負の最大値（80000000H）をreg2に格納します。

reg1は影響を受けません。

【補　足】 フラグ（CY, OV, S, Z）は変化しません。SATフラグは累積フラグで，飽和演算命令で演算結果が一度飽和するとセット（1）し，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フラグをリセットするには，LDSR命令でPSWに書き込みを行ってください。

【例　外】 なし

MACT3

Multiply and Accumulate with Truncation
（符号付き32ビット積和）積和

【命 令 形 式】 MACT3　reg1, reg2, reg3

【オペレーション】 GR[reg3] ← saturate(GR[reg3]＋上位32ビット(GR[reg2]×GR[reg1]))

【フォーマット】 Format VIII

【オ ペ コ ー ド】

15 10	9 5	4 0	31 26	25 21	20 16
111110	reg2	reg1	011100	RFU	reg3

【フ ラ グ】 CY －
OV －
S －
Z －

【命　　令】 MACT3　Multiply and Accumulate with Truncation

【説　　明】 reg2のワード・データとreg1のワード・データを符号付き整数として乗算し，乗算結果（64ビット）の下位32ビットを切り捨て，上位32ビットとreg3のデータを符号付き整数として加算します。

[オーバフローしない場合]

加算結果をreg3に格納します。

[オーバフローした場合]

SATフラグをセット（1）し，加算結果が正ならば正の最大値（7FFFFFFFH）を，負ならば負の最大値（80000000H）をreg3に格納します。

reg1, reg2は影響を受けません。

【補　　足】 MACT3命令の入力オペランドreg3のタイミングには制限があります。MACT3命令の3サイクル前までに，reg3を更新する命令がない場合には，1サイクル停止（ストール）してからMACT3命令を開始します。

フラグ（CY, OV, S, Z）は変化しません。SATフラグは累積フラグで，飽和演算命令で演算結果が一度飽和するとセット（1）し，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フラグをリセットするには，LDSR命令でPSWに書き込みを行ってください。

【例　　外】 なし

MAX3

Maximum
最大値

【命 令 形 式】　MAX3　reg1, reg2, reg3

【オペレーション】　GR[reg3] ← max(GR[reg2], GR[reg1])

【フォーマット】　Format VIII

【オ ペ コ ー ド】

15 – 10	9 – 5	4 – 0	31 – 26	25 – 21	20 – 16
111110	reg2	reg1	010011	RFU	reg3

【フ　ラ　グ】　CY　－
OV　－
S　－
Z　－

【命　　令】　MAX3　Maximum

【説　　明】　reg2のワード・データとreg1のワード・データを符号付き整数として比較し，大きい方のデータをreg3に格納します。reg1, reg2は影響を受けません。

【例　　外】　なし

MIN3

Minimum
最小値

【命 令 形 式】　MIN3　reg1, reg2, reg3

【オペレーション】　GR[reg3] ← min(GR[reg2], GR[reg1])

【フォーマット】　Format VIII

【オ ペ コ ー ド】

15 10	9 5	4 0	31 26	25 21	20 16
111110	reg2	reg1	010010	RFU	reg3

【フ ラ グ】　CY　－
OV　－
S　－
Z　－

【命　　令】　MIN3　Minimum

【説　　明】　reg2のワード・データとreg1のワード・データを符号付き整数として比較し，小さい方のデータをreg3に格納します。reg1, reg2は影響を受けません。

【例　　外】　なし

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）

15 10	9 5	4 0
000000	reg2	reg1

（2）

15 10	9 5	4 0
010000	reg2	imm5

【フ ラ グ】 CY －

OV －

S －

Z －

【命 令】 （1）MOV Move Register

（2）MOV Move Immediate(5-bit)

【説 明】 （1）reg1のワード・データを，reg2にコピーし転送します。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，reg2にコピーし転送します。

【例 外】 なし

MOVEA

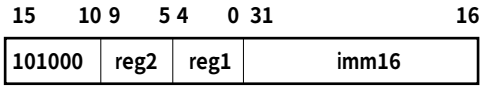
Add
加算

【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 V

【オ ペ コ ー ド】



15	10 9	5 4	0 31	16
101000	reg2	reg1	imm16	

【フ　ラ　グ】　CY　－
OV　－
S　－
Z　－

【命　　令】　MOVEA　Add Immediate(16-bit)

【説　　明】　reg1のワード・データに16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加算し，その結果をreg2に格納します。reg1は影響を受けません。フラグは変化しません。

【例　　外】　なし

MOVHI

Add
加算

【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 V

【オ ペ コ ー ド】

15	10 9	5 4	0 31	16
101111	reg2	reg1	imm16	

【フ ラ グ】　CY　－
OV　－
S　－
Z　－

【命　　令】　MOVHI　Add

【説　　明】　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reg2に格納します。reg1は影響を受けません。フラグは変化しません。

【例　　外】　なし

MUL

Multiply
（符号付き）乗算

【命 令 形 式】　MUL　reg1, reg2

【オペレーション】　result　← GR[reg2]×GR[reg1]（符号付き）
GR[30]　← result（上位32ビット）
GR[reg2] ← result（下位32ビット）

【フォーマット】　Format I

【オ ペ コ ー ド】

15	10 9	5 4	0
001000	reg2	reg1	

【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　GR[reg2]が負のとき１，そうでないとき０
Z　GR[reg2]が０のとき１，そうでないとき０

【命　　令】　MUL　Multiply

【説　　明】　reg2のワード・データにreg1のワード・データを乗算（符号付き）し，その結果（ダブル・ワード長）の上位32ビットをr30に，下位32ビットをreg2にそれぞれ格納します。reg1は影響を受けません。reg2にr30を指定した場合は，結果の下位32ビットをr30に格納します。オーバフローは，ダブル・ワード長の結果が，下位32ビットをダブル・ワード長まで符号拡張したものと等しくないときに設定されます。

【例　　外】　なし

MUL3

Multiply
（符号付き32ビット飽和）乗算

【命 令 形 式】　MUL3　reg1, reg2, reg3

【オペレーション】　GR[reg3] ← saturate（GR[reg2] × GR[reg1]）

【フォーマット】　Format VIII

【オ ペ コ ー ド】

15 10	9 5	4 0	31 26	25 21	20 16
111110	reg2	reg1	011111	RFU	reg3

【フ ラ グ】　CY　－
OV　－
S　－
Z　－

【命　　令】　MUL3　Multiply（3-operand）

【説　　明】　reg2のワード・データとreg1のワード・データを32ビット符号付き整数として乗算します。
乗算結果が32ビット符号付き整数として表現できる範囲を越えるとオーバフローとみなします（乗算結果64ビットのうち下位32ビットが有効）。
［オーバフローしない場合］
乗算結果をreg3に格納します。
［オーバフローした場合］
SATフラグをセット（１）し演算結果が正ならば正の最大値（7FFFFFFFH）を，負ならば負の最大値（80000000H）をreg3に格納します。
reg1, reg2は影響を受けません。

【補　　足】　フラグ（CY, OV, S, Z）は変化しません。SATフラグは累積フラグで，飽和演算命令で演算結果が一度飽和するとセット（１）し，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フラグをリセットするには，LDSR命令でPSWに書き込みを行ってください。

【例　　外】　なし

MULI

Multiply Immediate
（符号付き32ビット飽和）乗算

【命 令 形 式】　MULI　imm16, reg1, reg2

【オペレーション】　GR[reg2] ← saturate(GR[reg1] × sing-extend(imm16))

【フォーマット】　Format V

【オ ペ コ ー ド】

15 10	9 5	4 0	31 16
110010	reg2	reg1	imm16

【フ ラ グ】　CY －
OV －
S －
Z －

【命　　令】　MULI　Multiply Immediate

【説　　明】　reg1のワード・データとイミーディエト・データ（16ビットを32ビットに符号拡張）を符号付き整数として乗算します。乗算結果が32ビット符号付き整数として表現できる範囲を越えるとオーバフローとみなします（乗算結果64ビットのうち下位32ビットが有効）。

[オーバフローしない場合]

演算結果をreg2に格納します。

[オーバフローした場合]

SATフラグをセット（1）し，演算結果が正ならば正の最大値（7FFFFFFFH）を，負ならば負の最大値（80000000H）をreg2に格納します。

reg1は影響を受けません。

【補　　足】　フラグ（CY, OV, S, Z）は変化しません。SATフラグは累積フラグで，飽和演算命令で演算結果が一度飽和するとセット（1）し，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フラグをリセットするには，LDSR命令でPSWに書き込みを行ってください。

【例　　外】　なし

MULT3

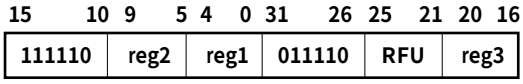
Multiply with Truncation
（符号付き32ビット）乗算

【命 令 形 式】　MULT3　reg1, reg2, reg3

【オペレーション】　GR[reg3] ← 上位32ビット（GR[reg2]×GR[reg1]）

【フォーマット】　Format VIII

【オ ペ コ ー ド】



15 10	9 5	4 0	31 26	25 21	20 16
111110	reg2	reg1	011110	RFU	reg3

【フ ラ グ】　CY　－
OV　－
S　－
Z　－

【命　　令】　MULT3　Multiply word with Truncation

【説　　明】　reg2のワード・データとreg1のワード・データを符号付き整数として乗算します。乗算結果（64ビット）の下位32ビットを切り捨て，上位32ビットだけをreg3に格納します。reg1, reg2は影響を受けません。

【例　　外】　なし

MULU

Multiply Unsigned
符号なし乗算

【命 令 形 式】　MULU　reg1, reg2

【オペレーション】　result　← GR[reg2]×GR[reg1]（符号なし）
GR[30]　← result（上位32ビット）
GR[reg2] ← result（下位32ビット）

【フォーマット】　Format I

【オ ペ コ ー ド】

15	10 9	5 4	0
001010	reg2	reg1	

【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　GR[reg2]が負のとき１，そうでないとき０
Z　GR[reg2]が０のとき１，そうでないとき０

【命　　令】　MULU　Multiply Unsigned

【説　　明】　reg2のワード・データにreg1のワード・データを符号なしデータとして乗算し，その結果（ダブル・ワード長）の上位32ビットをr30に，下位32ビットをreg2にそれぞれ格納します。reg1は影響を受けません。reg2にr30を指定した場合は，結果の下位32ビットをr30に格納します。フラグの設定は，結果が符号付きデータであるかのように行われます。オーバフローは，ダブル・ワード長の結果が，下位32ビットをダブル・ワード長までゼロ拡張したものと等しくないときに設定されます。

【例　　外】　なし

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	10 9	5 4	0
001111	reg2	reg1	

【フ　ラ　グ】　CY　－

OV　0

S　GR[reg2]が負のとき1，そうでないとき0

Z　GR[reg2]が0のとき1，そうでないとき0

【命　　令】　NOT　Not

【説　　明】　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。

reg1は影響を受けません。

【例　　外】　なし

OR

Or

論理和

【命 令 形 式】　OR　reg1, reg2

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

【フォーマット】　Format I

【オ ペ コ ー ド】

15	10 9	5 4	0
001100	reg2	reg1	

【フ　ラ　グ】　CY　－

OV　0

S　GR[reg2]が負のとき１，そうでないとき０

Z　GR[reg2]が０のとき１，そうでないとき０

【命　　令】　OR　Or

【説　　明】　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。reg1は影響を受けません。

【例　　外】　なし

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 V

【オ ペ コ ー ド】

15	10 9	5 4	0 31	16
101100	reg2	reg1	imm16	

【フ　ラ　グ】　CY　－

OV　0

S　GR[reg2]が負のとき１，そうでないとき０

Z　GR[reg2]が0 のとき１，そうでないとき０

【命　　令】　ORI　Or Immediate(16-bit)

【説　　明】　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reg2に格納します。reg1は影響を受けません。

【例　　外】　なし

OUT

Output to Port
ポート出力

【命 令 形 式】 （1）OUT.B　reg2, disp16[reg1]
（2）OUT.H　reg2, disp16[reg1]
（3）OUT.W　reg2, disp16[reg1]

【オペレーション】 （1）adr ← GR[reg1]＋(sign-extend)disp16
Output-Port(adr, GR[reg2], Byte)
（2）adr ← GR[reg1]＋(sign-extend)disp16
Output-Port(adr, GR[reg2], Halfword)
（3）adr ← GR[reg1]＋(sign-extend)disp16
Output-Port(adr, GR[reg2], Word)

【フォーマット】 Format VI

【オ ペ コ ー ド】

15	10	9	5	4	0	31	16
1111＊$		reg2		reg1		disp16	

(＊$:00=(1),01=(2),11=(3))

【フ ラ グ】 CY －
OV －
S －
Z －

【命 令】 （1）OUT.B　Output Byte to Port
（2）OUT.H　Output Halfword to Port
（3）OUT.W　Output Word to Port

【説 明】 （1）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，32ビット符号なしポート・アドレスを生成します。reg2の下位1バイトのデータを生成したポート・アドレスに出力します。
（2）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，32ビット符号なしポート・アドレスを生成します。reg2の下位2バイトのデータを生成したポート・アドレスに出力します。32ビット符号なしアドレスのビット0は0にマスクされます。

（3）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，32ビット符号なしポート・アドレスを生成します。reg2のワード・データを生成したポート・アドレスに出力します。32ビット符号なしアドレスのビット0およびビット1は0にマスクされます。

【例　　外】　なし

RETI

Return from Trap or Interrupt
トラップ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.NP = 1
　then PC　← FEPC
　　PSW ← FEPSW
　else PC　← EIPC
　　PSW ← EIPSW

【フォーマット】　Format IX

【オ ペ コ ー ド】

15	10	9	1	0
011001	RFU			0

【フ　ラ　グ】　CY　読み出した値が設定される
OV　読み出した値が設定される
S　読み出した値が設定される
Z　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，トラップ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NPフラグが1の場合は，FEPC, FEPSWから，NPフラグが0の場合は，
　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，PCにジャンプします。

【例　　外】　なし

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(imm5)

【フォーマット】 （1）Format I
（2）Format II

【オ ペ コ ー ド】 （1）

15 10	9 5	4 0
000111	reg2	reg1

（2）

15 10	9 5	4 0
010111	reg2	imm5

【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　GR[reg2]が負のとき1，そうでないとき0
Z　GR[reg2]が0のとき1，そうでないとき0

【命　　令】 （1）SAR　Shift Arithmetic Right by Register
（2）SAR　Shift Arithmetic Right by Immediate(5-bit)

【説　　明】 （1）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，算術右シフトし（MSBの値を順にMSBにコピーする），reg2に書き込みます。シフト数が0のときは，reg2は命令実行前と同じ値を保持したままです。シフト数は，5ビット・データですので0から＋31まで指定できます。
（2）reg2のワード・データを5ビット・イミーディエトをワード長までゼロ拡張した値で示されるシフト数分, 算術右シフトし（MSBの値を順にMSBにコピーする），reg2に書き込みます。シフト数が0のときは，reg2は命令実行前と同じ値を保持します。シフト数は，0から＋31まで指定できます。

【例　　外】 なし

SATADD3

Saturated Add
飽和加算

【命 令 形 式】　SATADD3　reg1, reg2, reg3

【オペレーション】　GR[reg3] ← saturate(GR[reg2]＋GR[reg1])

【フォーマット】　Format VIII

【オ ペ コ ー ド】

15 10	9 5	4 0	31 26	25 21	20 16
111110	reg2	reg1	010000	RFU	reg3

【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　GR[reg3]が負のとき１，そうでないとき０
Z　GR[reg3]が０のとき１，そうでないとき０

【命　　令】　SATADD3　Saturated Add

【説　　明】　reg2のワード・データにreg1のワード・データを符号付き整数として加算します。
[オーバフローしない場合]
加算結果をreg3に格納します。
[オーバフローした場合]
SATフラグをセット（１）し，加算結果が正ならば正の最大値（7FFFFFFFH）を，負ならば負の最大値（80000000H）をreg3に格納します。
reg1, reg2は影響を受けません。

【補　　足】　SATフラグは累積フラグで，飽和演算命令で演算結果が一度飽和するとセット（１）し，以降の命令の演算結果が飽和しなくてもリセット（０）しません。SATフラグをリセット（０）するにはLDSR命令でPSWに書き込みを行ってください。この命令によって演算結果が飽和した場合，フラグはreg1, reg2の大小を表しません。したがって，ABGT, ABGE, ABLT, ABLE, BGT, BGE, BLT, BLE命令では正常に分岐できません。ABE, ABNE, ABN, ABP, BE, BNE, BN, BP命令を使用してください。

【例　　外】　なし

SATSUB3

Saturated Subtract
飽和減算

【命 令 形 式】 SATSUB3 reg1, reg2, reg3

【オペレーション】 GR[reg3] ← saturate(GR[reg2] − GR[reg1])

【フォーマット】 Format VIII

【オ ペ コ ー ド】

15 10	9 5	4 0	31 26	25 21	20 16
111110	reg2	reg1	010001	RFU	reg3

【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 GR[reg3]が負のとき１，そうでないとき０
Z GR[reg3]が０のとき１，そうでないとき０

【命 令】 SATSUB3 Saturated Subtract

【説 明】 reg2のワード・データにreg1のワード・データを符号付き整数として減算します。
[オーバフローしない場合]
減算結果をreg3に格納します。
[オーバフローした場合]
SATフラグをセット（１）し，減算結果が正ならば正の最大値（7FFFFFFFH）を，負ならば負の最大値（80000000H）をreg3に格納します。
reg1, reg2は影響を受けません。

【補 足】 SATフラグは累積フラグで，飽和演算命令で演算結果が一度飽和するとセット（１）し，以降の命令の演算結果が飽和しなくてもリセット（０）しません。SATフラグをリセット（０）するにはLDSR命令でPSWに書き込みを行ってください。この命令によって演算結果が飽和した場合，フラグはreg1, reg2の大小を表しません。したがって，ABGT, ABGE, ABLT, ABLE, BGT, BGE, BLT, BLE命令では正常に分岐できません。ABE, ABNE, ABN, ABP, BE, BNE, BN, BP命令を使用してください。

【例 外】 なし

SETF

Set Flag Condition
フラグ条件の設定

【命 令 形 式】　SETF　imm5, 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 II

【オ ペ コ ー ド】

15	10 9	5 4	0
010010	reg2	imm5	

【フ　ラ　グ】　CY　－
OV　－
S　－
Z　－

【命　　令】　SETF　Set Flag Condition

【説　　明】　５ビット・イミーディエトの下位４ビットの示す条件が満たされた場合，reg2に１を，そうでない場合は０を格納します。５ビット・イミーディエトの下位４ビットには，表５－３に示す条件コードを指定します。上位１ビットは無視されます。

【例　　外】　なし

表5－3　条件コード一覧

条件コード	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 1
1101	F	always 0
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

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 I
（2）Format II

【オ ペ コ ー ド】 （1）

15 10	9 5	4 0
000100	reg2	reg1

（2）

15 10	9 5	4 0
010100	reg2	imm5

【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 GR[reg2]が負のとき1，そうでないとき0
Z GR[reg2]が0のとき1，そうでないとき0

【命 令】 （1）SHL Shift Logical Left by Register
（2）SHL Shift Logical Left by Immediate(5-bit)

【説 明】 （1）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，論理左シフトし（LSB側に0を送り込む），reg2に書き込みます。シフト数が0のときは，reg2は命令実行前と同じ値を保持したままです。シフト数は，5ビット・データですので0から＋31まで指定できます。
（2）reg2のワード・データを，5ビット・イミーディエトをワード長までゼロ拡張した値で示されるシフト数分，論理左シフトし（LSB側に0を送り込む），reg2に書き込みます。シフト数が0のときは，reg2は命令実行前と同じ値を保持したままです。シフト数は，0から＋31まで指定できます。

【例 外】 なし

SHLD3

Shift Left Double word
連結左シフト

【命 令 形 式】　SHLD3　reg1, reg2, reg3

【オペレーション】　GR[reg3] ← (GR[reg3], GR[reg2])<<reg1

【フォーマット】　Format VIII

【オ ペ コ ー ド】

15 10	9 5	4 0	31 26	25 21	20 16
111110	reg2	reg1	011000	RFU	reg3

【フ　ラ　グ】　CY　－
OV　－
S　－
Z　－

【命　　令】　SHLD3　Shift Left Double word

【説　　明】　reg3（上位），reg2（下位）のデータを連結した64ビット・データをreg1の下位５ビットで示されるシフト数だけ論理左シフトをして，上位32ビットをreg3に出力します。reg1が０のときはreg3のデータがreg3に格納されます。reg1の上位27ビットは無視されます。reg1, reg2は影響を受けません。

【補　　足】　SHLD3命令の入力オペランドreg3のタイミングには制限があります。SHLD3命令の３サイクル前までにreg3を更新する命令がない場合，１サイクル停止（ストール）してからSHLD3命令を開始します。

【例　　外】　なし

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 I
（2）Format II

【オペコード】 （1）

15 10	9 5	4 0
000101	reg2	reg1

（2）

15 10	9 5	4 0
010101	reg2	imm5

【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　GR[reg2]が負のとき1，そうでないとき0
Z　GR[reg2]が0のとき1，そうでないとき0

【命　　令】 （1）SHR　Shift Logical Right by Register
（2）SHR　Shift Logical Right by Immediate(5-bit)

【説　　明】 （1）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。シフト数が0のときは，reg2は命令実行前と同じ値を保持したままです。シフト数は，5ビット・データですので0から＋31まで指定できます。
（2）reg2のワード・データを5ビット・イミーディエトをワード長までゼロ拡張した値で示されるシフト数分，論理右シフトし（MSB側に0を送り込む），reg2に書き込みます。シフト数が0のときは，reg2は命令実行前と同じ値を保持したままです。シフト数は，0から＋31まで指定できます。

【例　　外】 なし

SHRD3

Shift Right Double word
連結右シフト

【命 令 形 式】　SHRD3　reg1, reg2, reg3

【オペレーション】　GR[reg3] ← (GR[reg3], GR[reg2])>>reg1

【フォーマット】　Format VIII

【オ ペ コ ー ド】

15 – 10	9 – 5	4 – 0	31 – 26	25 – 21	20 – 16
111110	reg2	reg1	011001	RFU	reg3

【フ　ラ　グ】　CY　－
OV　－
S　－
Z　－

【命　　令】　SHRD3　Shift Right Double word

【説　　明】　reg3（上位），reg2（下位）のデータを連結した64ビット・データをreg1の下位５ビットで示されるシフト数だけ論理右シフトをして，下位32ビットをreg3に出力します。reg1が０のときはreg2のデータがreg3に格納されます。reg1の上位27ビットは無視されます。reg1, reg2は影響を受けません。

【補　　足】　SHRD3命令の入力オペランドreg3のタイミングには制限があります。SHRD3命令の３サイクル前までにreg3を更新する命令がない場合，１サイクル停止（ストール）してからSHRD3命令を開始します。

【例　　外】　なし

ST

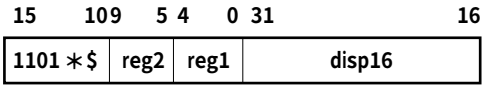
Store
ストア

【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 VI

【オ ペ コ ー ド】



15 10	9 5	4 0	31 16
1101＊$	reg2	reg1	disp16

(＊$:00=(1),01=(2),11=(3))

【フ ラ グ】 CY －
OV －
S －
Z －

【命　　令】 （1）ST.B　Store Byte
（2）ST.H　Store Halfword
（3）ST.W　Store Word

【説　　明】 （1）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，32ビット符号なしアドレスを生成します。reg2の下位１バイトのデータを生成したアドレスに格納します。
（2）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，32ビット符号なしアドレスを生成します。reg2の下位２バイトのデータを生成したアドレスに格納します。32ビット符号なしアドレスのビット０は０にマスクされます。

（３）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，32ビット符号なしアドレスを生成します。reg2のワード・データを生成したアドレスに格納します。32ビット符号なしアドレスのビット０およびビット１は０にマスクされます。

【例　　外】　なし

STBY

Standby
停止

【命 令 形 式】　STBY

【オペレーション】　停止

【フォーマット】　Format IX

【オ ペ コ ー ド】

15	10	9	1	0
011010		RFU		1

【フ　ラ　グ】　CY　－
OV　－
S　－
Z　－

【命　　令】　STBY　Standby

【説　　明】　CPUは停止して，ストップ・モードへ遷移します。

【例　　外】　なし

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 II

【オ ペ コ ー ド】

15	10 9	5 4	0
011101	reg2	regID	

【フ　ラ　グ】　CY　－
OV　－
S　－
Z　－

【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reg2に設定します。システム・レジスタは影響を受けません。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・レジスタに対するSTSR命令を実行した場合の動作は保証しません。

【例　　外】　なし

SUB

Subtract
減算

【命　令　形　式】　SUB　reg1, reg2

【オペレーション】　GR[reg2] ← GR[reg2]－GR[reg1]

【フォーマット】　Format　I

【オ　ペ　コ　ー　ド】

15	10 9	5 4	0
000010	reg2	reg1	

【フ　　ラ　　グ】　CY　MSBからのボローがあれば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　GR[reg2]が負のとき１，そうでないとき０
Z　GR[reg2]が０のとき１，そうでないとき０

【命　　令】　SUB　Subtract

【説　　明】　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。reg1は影響を受けません。

【例　　外】　なし

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】
```
if  PSW.NP = 1
  then  致命的例外（MACHINE FAULT）
else if PSW.EP = 1
  then  FEPC      ← 復帰PC
        FEPSW     ← PSW
        ECR.FECC  ← 例外コード
        PSW.NP    ←1
        PSW.ID    ←1
        PC        ← ＜NMIハンドラ・アドレス＞
  else  EIPC      ← 復帰PC
        EIPSW     ← PSW
        ECR.EICC  ← 例外コード
        PSW.EP    ←1
        PSW.ID    ←1
        PC        ← ＜vector adr＞
```

【フォーマット】　Format II

【オペコード】

15　10	9　0
011000	vector

【フ　ラ　グ】
```
CY  －
OV  －
 S  －
 Z  －
```

【命　　令】　TRAP　Trap

【説　　明】 PSWのNPフラグが１の場合は，致命的例外となり，プロセッサは致命的例外処理を行います。

PSWのNPフラグが０で，EPフラグが１の場合は，二重例外となります。この場合，復帰PC, PSWをFEPC, FEPSWに退避し，例外コードの設定（ECRのFECC），PSWのフラグの設定（NP, IDフラグをセット）を行ったあと，NMI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。

PSWのNPフラグとEPフラグがともに０の場合は，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。

【例　　外】 なし

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	10 9	5 4	0
001110	reg2	reg1	

【フ　ラ　グ】　CY　－
OV　0
S　GR[reg2]が負のとき１，そうでないとき０
Z　GR[reg2]が０のとき１，そうでないとき０

【命　　令】　XOR　Exclusive Or

【説　　明】　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。reg1は影響を受けません。

【例　　外】　なし

XORI

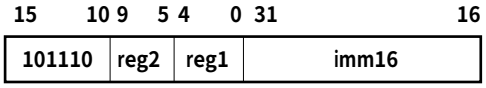
Exclusive Or Immediate
排他的論理和

【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 V

【オ ペ コ ー ド】



15 – 10	9 – 5	4 – 0	31 – 16
101110	reg2	reg1	imm16

【フ　ラ　グ】　CY　－
OV　0
S　GR[reg2]が負のとき１，そうでないとき０
Z　GR[reg2]が0 のとき１，そうでないとき０

【命　　令】　XORI　Exclusive Or Immediate(16-bit)

【説　　明】　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排他的論理和をとり，その結果をreg2に格納します。reg1は影響を受けません。

【例　　外】　なし

5.4　命令実行サイクル数

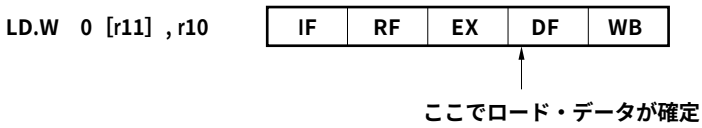
各命令の実行サイクル数を示します。実際の実行サイクル数はRepeatとLatencyの間の数値になります。

（1）Latency（レイテンシ）

命令実行を開始してから終了するまでの時間です。

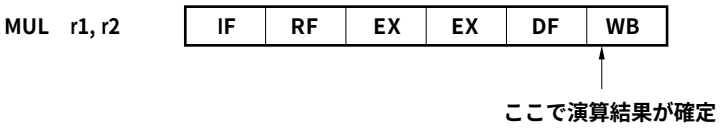
［例1］LD命令

LD命令のLatencyは2で実行はEXステージ，DFステージの2サイクルで終了します。



［例2］MUL命令

MUL命令のLatencyは4で実行はEXステージ，EXステージ，DFステージ，WBステージの4サイクルで終了します。



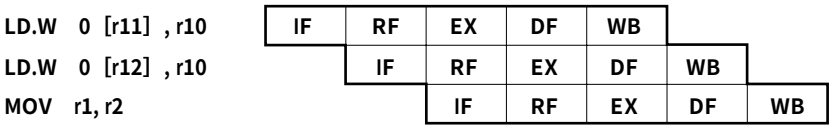
備考　パイプラインの流れの詳細は**第9章　パイプライン**を参照してください。

（2）Repeat（リピート）

現行命令と後続命令が同一演算器を使用する場合，現行命令の実行を開始してから後続命令の実行が可能になるまでの時間です。命令の実行を開始するタイミングは，その命令がオペランドを獲得できたときです。

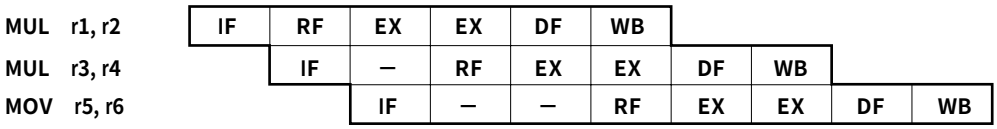
［例1］　LD命令

LD命令のRepeatは1で，EXステージは1サイクルです。このため後続命令は1サイクルでEXステージを使うことができます。



［例２］MUL命令

MUL命令のRepeatは２で，EXステージが２サイクルです。このため後続命令は２サイクル後にEXステージを使うことができ，毎サイクルごとにEXステージが使えるとは限りません。



備考　パイプラインの流れの詳細は**第９章　パイプライン**を参照してください。

（３）Latency-Repeat（レイテンシとリピートの差）

演算結果の出力されるパイプライン・ステージを示します。

差が０：EXステージで演算結果を出力

差が１：DFステージで演算結果を出力

差が２：WBステージで演算結果を出力

表５－４で使用する略号は次のとおりです。

①：データ・キャッシュにヒットまたは内部RAMアクセス

②：データ・キャッシュ・ミス

③：外部RAM（アンキャッシャブル領域）のアクセス

B ：バースト・バス・サイクル実行クロック数（外部クロック）

S ：シングル・バス・サイクル実行クロック数（外部クロック）

n ：内部クロックと外部クロックの周波数比（n = 2またはn = 3）

s ：外部クロックに同期するための待ち合わせ時間

n = 2のとき → s = 0, 1

n = 3のとき → s = 0, 1, 2

表５－４　命令実行サイクル数（1/3）

	ニモニック	オペランド	命令長（バイト）	Latency	Repeat
ロード／ストア	LD.B	disp16[reg1], reg2	4	①２ ② n×B＋10＋s[注1] ③ n×S＋９＋s[注1]	①１ ② n×B＋９＋s ③ n×S＋８＋s
	LD.H	disp16[reg1], reg2	4		
	LD.W	disp16[reg1], reg2	4		
★	ST.B	reg2, disp16[reg1]	4	①３ ② n×S＋５＋s[注1] ③ n×S＋５＋s[注1]	１
	ST.H	reg2, disp16[reg1]	4		
	ST.W	reg2, disp16[reg1]	4		
	BILD	[reg1], [reg2]	4	n×B＋10＋s[注1]	n×B＋10＋s
	BIST	[reg2], [reg1]	4	n×B＋７＋s[注1]	n×(B－１)＋10＋s
	BDLD	[reg1], [reg2]	4	n×B＋10＋s[注1]	n×B＋10＋s
	BDST	[reg2], [reg1]	4	n×B＋７＋s[注1]	n×(B－１)＋10＋s
入出力	IN.B	disp16[reg1], reg2	4	n×S＋10＋s[注1]	n×S＋９＋s
	IN.H	disp16[reg1], reg2	4		
	IN.W	disp16[reg1], reg2	4		
	OUT.B	reg2, disp16[reg1]	4	n×S＋６＋s[注1]	n×S＋９＋s
	OUT.H	reg2, disp16[reg1]	4		
	OUT.W	reg2, disp16[reg1]	4		
算術演算	MOV	reg1, reg2	2	１	１
		imm5, reg2			
	MOVHI	imm16, reg1, reg2	4	１	１
	ADD	reg1, reg2	2	１	１
		imm5, reg2			
	ADDI	imm16, reg1, reg2	4	１	１
	MOVEA	imm16, reg1, reg2	4	１	１
	SUB	reg1, reg2	2	１	１
	MUL	reg1, reg2	2	４[注2]	２
	MULU	reg1, reg2	2	４[注2]	２
	DIV	reg1, reg2	2	37	37
	DIVU	reg1, reg2	2	35	35
	CMP	reg1, reg2	2	１	１
		imm5, reg2			
	SETF	imm5, reg2	2	２	１
	MIN3	reg1, reg2, reg3	4	２	１
	MAX3	reg1, reg2, reg3	4	２	１

注１． ライト・バッファを空にしてから実行してするため，ライト・バス・サイクル時間が加わることがあります。

２． フラグのレイテンシは３サイクルです。次の命令がフラグを参照する命令（条件分岐など）の場合，フラグ・ハザードとなります。

表５－４　命令実行サイクル数（2/3）

	ニモニック	オペランド	命令長（バイト）	Latency	Repeat
積和／飽和演算	MUL3	reg1, reg2, reg3	4	3	1
	MAC3	reg1, reg2, reg3	4	3 注1	1 注1
	MULI	imm16, reg1, reg2	4	3	1
	MACI	imm16, reg1, reg2	4	3	1
	MULT3	reg1, reg2, reg3	4	3	1
	MACT3	reg1, reg2, reg3	4	3 注1	1 注1
	SATADD3	reg1, reg2, reg3	4	2	1
	SATSUB3	reg1, reg2, reg3	4	2	1
論理演算	OR	reg1, reg2	2	1 注2	1
	ORI	imm16, reg1, reg2	4	1 注2	1
	AND	reg1, reg2	2	1 注2	1
	ANDI	imm16, reg1, reg2	4	1 注2	1
	XOR	reg1, reg2	2	1 注2	1
	XORI	imm16, reg1, reg2	4	1 注2	1
	NOT	reg1, reg2	2	1 注2	1
	SHL	reg1, reg2	2	2 注2	1
		imm5, reg2			
	SHR	reg1, reg2	2	2 注2	1
		imm5, reg2			
	SAR	reg1, reg2	2	2 注2	1
		imm5, reg2			
	SHLD3	reg1, reg2, reg3	4	2 注1	1 注1
	SHRD3	reg1, reg2, reg3	4	2 注1	1 注1

注１．reg3をデスティネーションとする命令が３サイクル前までに実行されていないと１サイクル停止します。

２．フラグのレイテンシは２サイクルです。次の命令がフラグを参照する命令（条件分岐など）の場合，フラグ・ハザードとなります。

表５－４　命令実行サイクル数（3/3）

	ニモニック	オペランド	命令長（バイト）	Latency	Repeat
分岐	JMP	[reg1]	2	3 注1	3
	JR	disp26	4	3 注1	3
	JAL	disp26	4	3 注1	3
	Bcond	disp9	2	3（taken）注1 1（not taken）注2	3（taken） 1（not taken）
	ABcond	disp9	2	1（履歴あり）注1 3（履歴なし）	1（履歴あり） 3（履歴なし）
特殊	LDSR	reg2, regID	2	5	5
	STSR	regID, reg2	2	5	2
	TRAP	vector	2	5	5
	RETI	－	2	5 注1	5
	CAXI	disp16[reg1], reg2	4	n×S＋18＋s 注3	n×S＋18＋s
	HALT	－	2	5 注3	－
	STBY	－	2	5 注3	－
	BRKRET	－	2	5 注1	5
	EI	－	2	4	4
	DI	－	2	4	4

注１．分岐先のアドレスが４の倍数ではなく，分岐先の命令32ビットの場合，１サイクル停止します。

２．高速分岐命令（ABcond）の後続命令が，32ビット長でアドレスが４の倍数でなければ，ループを抜けるときに１サイクル停止します。

３．ライト・バッファを空にしてから実行してするため，ライト・バス・サイクル時間が加わることがあります。

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独立に発生する事象で，マスカブル割り込み，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大きな違いはありませんが，割り込みは例外より優先的に処理されます。ただし，致命的例外は割り込みより優先的に処理されます。

V830ファミリ・アーキテクチャでは，次に示すような例外と割り込みがあります。例外，マスカブル割り込み，ノンマスカブル割り込み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。例外要因は，ECR（Exception Cause Register）に格納される例外コードで知ることができます。各ハンドラではECRを解析し，適切な例外／割り込み処理を行います。

表６－１　例外／割り込み要因コード

例外と割り込み		分　類	例外コード ECR[注1]	割り込み要求名	ハンドラ・アドレス[注1]	復帰PC
リセット		割り込み	FFF0H	RESET	FFFFFFF0H	不定
致命的例外		例外	－	FAULT	FFFFFFE0H	current PC
NMI		割り込み	FFD0H	NMI	FFFFFFD0H	next PC
二重例外		例外	注２	NMI	FFFFFFD0H	current PC
TRAP命令（パラメータ 0x1n）		例外	FFBnH	TRAP1n	FFFFFFB0H	next PC
TRAP命令（パラメータ 0x0n）		例外	FFAnH	TRAP0n	FFFFFFA0H	next PC
不正命令コード		例外	FF90H	I_OPC	FFFFFF90H	current PC
ゼロ除算		例外	FF80H	DIV0	FFFFFF80H	current PC
割り込みレベルn（n = 0-15）[注3]	HWCC.IHA = 0	割り込み	FEn0H	INT0n	FFFFFEn0H	next PC
	HWCC.IHA = 1			INT1n	FE0000n0H	

注１．レベルnは16進数表記になります（n = 0-F）。

２．二重例外の要因となった例外の例外コードです。

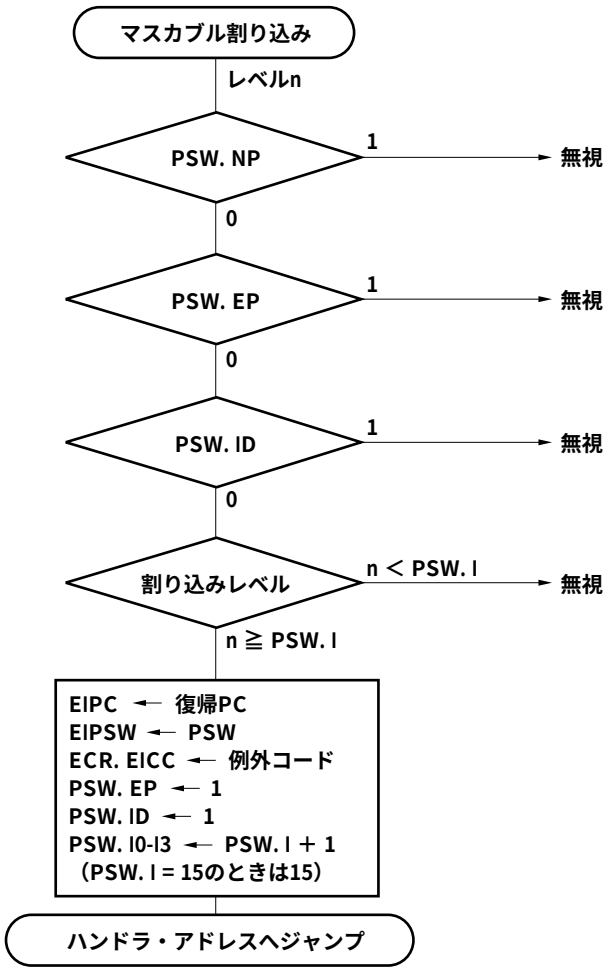
★ **３．**V831, V832は割り込みコントローラを内蔵しており，内部／外部の割り込み要因をINT0n，およびINT1nに割り当てています。詳しくは**各製品のユーザーズ・マニュアル　ハードウエア編　第４章　割り込み／例外処理機能**を参照してください。

６.１　割り込み処理

６.１.１　マスカブル割り込み

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。状態退避レジスタにはEIPC, EIPSWを使用します。

マスカブル割り込みは，PSWのNP, EP, IDの論理和でマスクされます。さらに，$\overline{\text{INTV0}}$-$\overline{\text{INTV3}}$に示される割り込みレベルnが，PSWの割り込み許可レベル（PSWのI0-I3ビット）より低い場合（n＜I0-I3）は割り込みは受け付けられません。したがって，最高位の割り込みレベル（n = 15）を割り込み許可レベルで禁止することはできません。



① 復帰PCをEIPCへ退避します。

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EPビットをセットします。

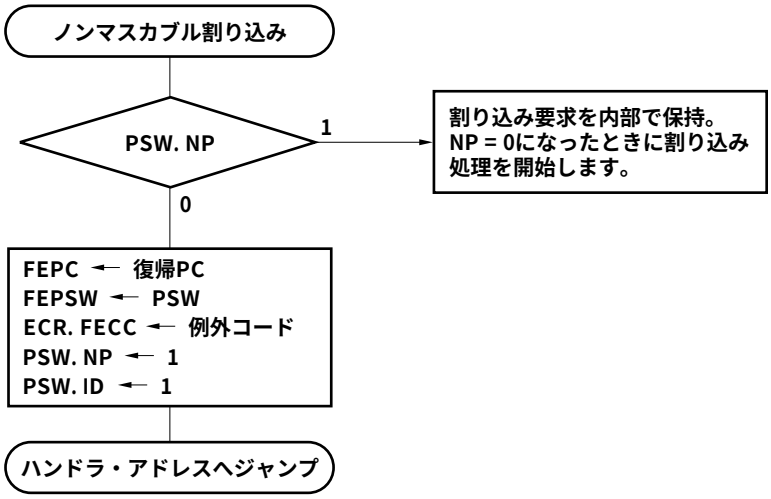
⑤ PSWのIDビットをセットします。

⑥ PSWのI（I0-I3）フィールドに受け付けた割り込みレベルnに１を加算した値（n＋１）をセットします。ただし，受け付けた割り込みレベルが最高位（n = 15）の場合は15をセットします。

⑦ ハンドラ・アドレスへジャンプします。

６.１.２　ノンマスカブル割り込み

$\overline{\text{NMI}}$入力により，ノンマスカブル割り込みが発生した場合は，次の処理を行い，ハンドラ・ルーチンへ制御を移します。状態退避レジスタにはFEPC, FEPSWを使用します。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，プロセッサ内部で保持されます（ノンマスカブル割り込み処理開始直後，内部処理によってラッチをクリアする期間に発生したノンマスカブル割り込み要求は，プロセッサ内部のラッチに保持されません）。プロセッサは，ノンマスカブル割り込みを$\overline{\text{NMI}}$入力の立ち下がりで検出します。たのため，$\overline{\text{NMI}}$入力を一度インアクティブにしてから$\overline{\text{NMI}}$入力をアクティブにしてください。



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16ビット（FECC）に例外コードを書き込みます。

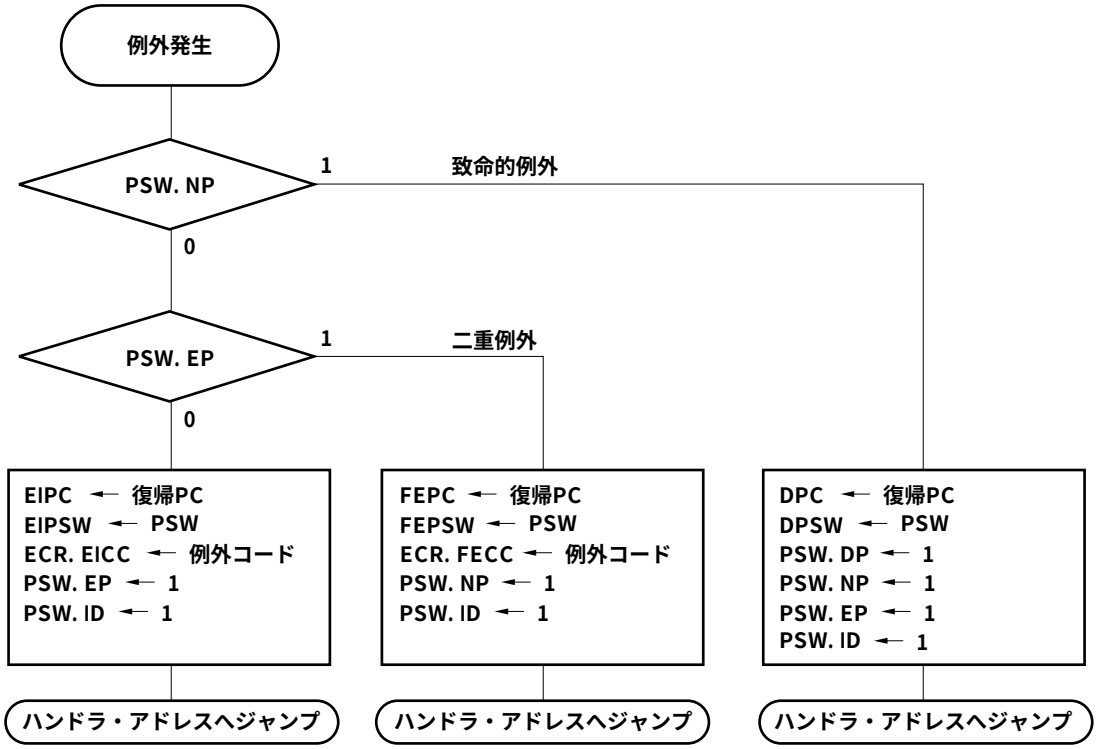
④ PSWのNPビットをセットします。

⑤ PSWのIDビットをセットします。

⑥ ハンドラ・アドレス（FFFFFFD0H）へジャンプします。

6.2　例外処理

例外が発生した場合，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 すでにPSWのNPビットがセットされていたら，⑧致命的例外処理へ

② すでにPSWのEPビットがセットされていたら，⑨二重例外処理へ

③ 復帰PCをEIPCへ退避します。

④ 現在のPSWをEIPSWへ退避します。

⑤ 例外コードをECRの下位16ビット（EICC）に書き込みます。

⑥ PSWのEP, IDビットをセットします。

⑦ ハンドラ・アドレスへジャンプします。

⑧ 致命的例外処理

（a）復帰PCをDPCへ退避します。

（b）現在のPSWをDPSWへ退避します。

（c）PSWのDP, NP, EP, IDビットをセットします。

（d）ハンドラ・アドレス（FFFFFFE0H）へジャンプします。

⑨ 二重例外処理

（a）復帰PCをFEPCへ退避します。

（b）現在のPSWをFEPSWへ退避します。

（c）例外コードをECRの上位16ビット（FECC）に書き込みます。

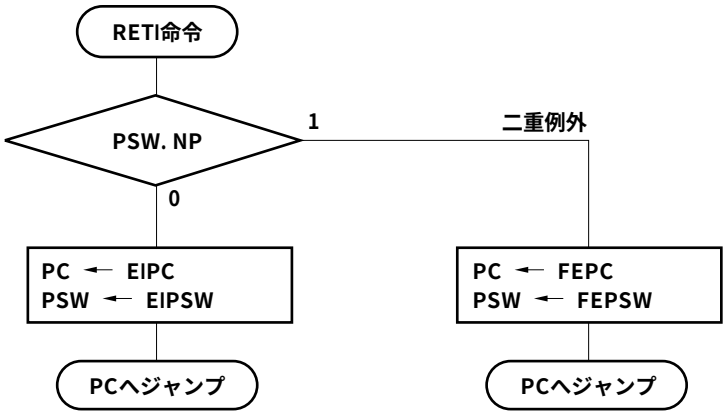
（d）PSWのNP, IDビットをセットします。

（e）ハンドラ・アドレス（FFFFFFD0H）へジャンプします。

６.３　例外／割り込みからの復帰

６.３.１　例外／割り込みからの復帰

致命的例外処理以外の例外および割り込み事象からの復帰は，すべてRETI命令により行います。

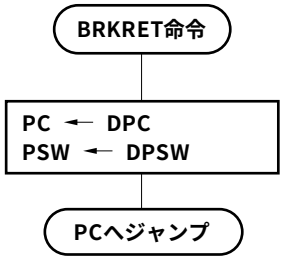


① PSWのNP = 1ならFEPC, FEPSWから，NP = 0ならEIPC, EIPSWから復帰PC, PSWを取り出します。

② 復帰PC, PSWを元に戻し，PCへジャンプします。

６.３.２　致命的例外処理ルーチンからの復帰

致命的例外処理からの復帰は，BRKRET命令により行います。



① DPC, DPSWから復帰PC, PSWを取り出します。

② 復帰PC, PSWを元に戻し，PCへジャンプします。

６.４　割り込みと例外の優先順位

割り込みと例外の優先順位を次に示します。割り込みと例外のいくつかが同時に発生した場合は，この優先順位に従います。

	RESET	NMI	INT	トラップ命令	不正命令コード例外	ゼロ除算例外
RESET		＊	＊	＊	＊	＊
NMI	×		←	←	←	←
INT	×	↑		←	←	←
トラップ命令	×	↑	↑		−	−
不正命令コード例外	×	↑	↑	−		−
ゼロ除算例外	×	↑	↑	−	−	

＊：左部の項目は上部の項目を無視する。

×：左部の項目は上部の項目に無視される。

−：左部の項目と上部の項目は同時には発生しない。

←：左部の項目は上部の項目より優先順位が高い。

↑：上部の項目は左部の項目より優先順位が高い。

★　６.４.１　マスカブル割り込みの優先順位

V831, V832は内蔵の割り込みコントローラにより，複数の割り込み要因が発生した場合の優先順位制御を行います。詳しくは**各製品のユーザーズ・マニュアル　ハードウエア編　第４章　割り込み／例外処理機能**を参照してください。

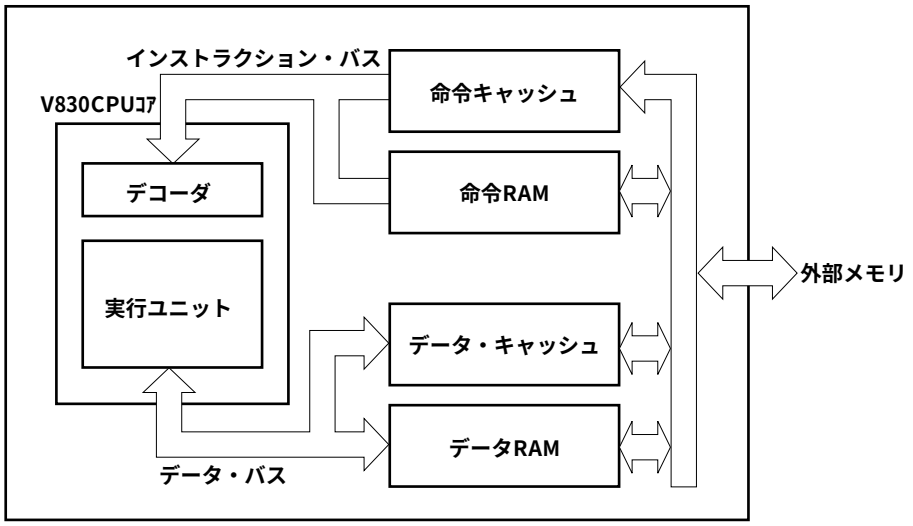
第７章　内蔵メモリ

この章では，内蔵キャッシュ，内蔵RAMの機能およびその検索機能について説明します。

７.１　内蔵キャッシュ

V830ファミリには4 Kバイト×４の内蔵メモリがあります。内蔵メモリは４つのブロック（命令キャッシュ，データ・キャッシュ，命令RAM，データRAM）で構成されています。V830ファミリはこれらの内蔵メモリに１サイクルでアクセスできます。

図７－１　内蔵キャッシュの構成



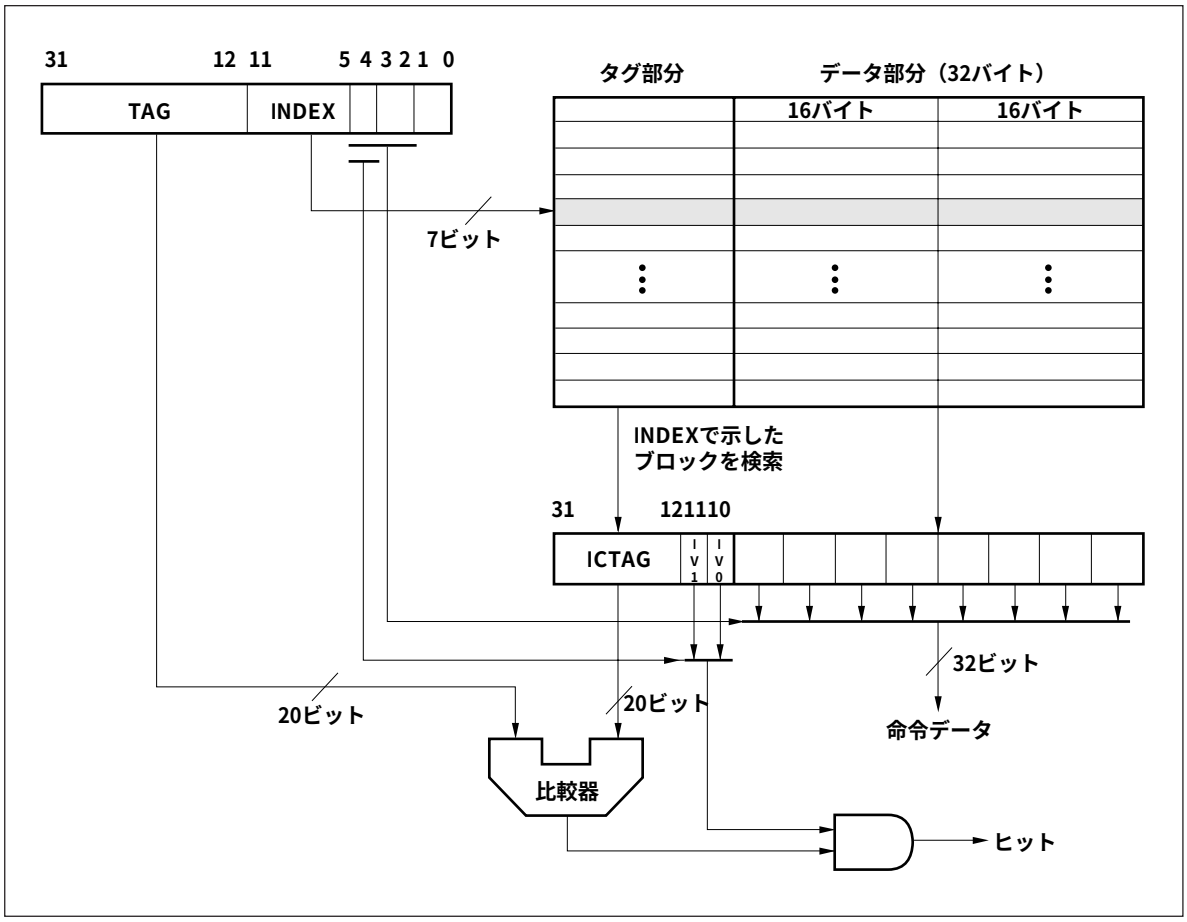
注意　命令キャッシュ，命令RAMにデータは書き込めません。
　　データ・キャッシュ，データRAMに命令は書き込めません。

７.１.１　命令キャッシュ

命令キャッシュ・メモリは，128個の32バイト・ブロックで構成され，合計4 Kバイトの容量です。各ブロックは，２個のサブブロック（16バイト）で構成され，タグと２つのバリッド・ビット（32バイト・ブロックの上位16バイトがIV1，下位16バイトがIV0）があります。バリッド・ビットは，サブブロックの内容が有効であるか，無効であるかを示し，キャッシュ・ミスが発生した場合は，サブブロック単位のリフィルを行います。

なお，命令キャッシュにキャッシングされる命令は，キャッシャブル領域から命令フェッチされた命令列に限ります。ただし，内蔵命令RAM上にある命令はキャッシングされません。

図７－２　命令キャッシュの構成



7.1.2　命令キャッシュ・タグ検索機能

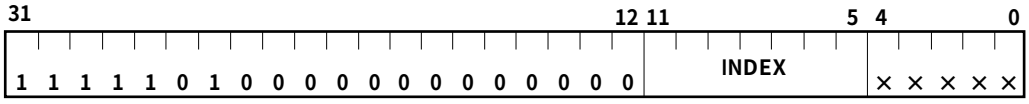
V830ファミリは命令キャッシュにキャッシングされている命令のタグを検索することができます。このタグからキャッシングされている命令のアドレスを生成して，キャッシングされている命令列を知ることができます。

タグの検索はICTRレジスタを使用します。ICTRレジスタはI/O空間（FA000000H-FA000FFFH）にマップされていて，全部で128個あります。各レジスタはICTR0-ICTR127の番号が付けられて，ビット4-0が０となるアドレス上にマップされています。また，この番号はキャッシュのブロック番号に対応しています。

（1）命令キャッシュ・タグ・レジスタ

命令キャッシュのタグを検索するときに使用するレジスタです。これらのレジスタにアクセスするときはIN.W命令，OUT.W命令を使用してください。

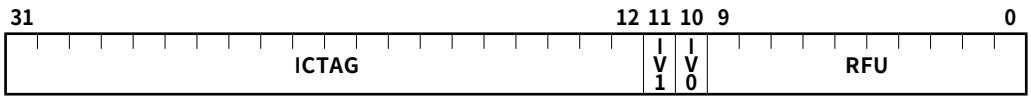
ICTRのアドレス指定方法（FA000×××H）



×：Don't care

ビット位置	ビット名	意　　味
11-5	INDEX	Index 内蔵命令キャッシュ・タグのアドレスを指定します。

ICTRの内容



ビット位置	ビット名	意　　味
31-12	ICTAG	Instruction Cache Tag 命令キャッシュのインデクスで指定したブロックのタグ。
11	IV1	Instruction Cache Valid Bit インデクスで指定した上位側のサブブロックが有効であることを示します。 IV1 = 0：無効 IV1 = 1：有効（サブブロックはICTAGで示した外部メモリの内容と一致）
10	IV0	Instruction Cache Valid Bit インデクスで指定した下位側のサブブロックが有効であることを示します。 IV0 = 0：無効 IV0 = 1：有効（サブブロックはICTAGで示した外部メモリの内容と一致）
9-0	RFU	予約フィールドです（0に固定してください）。

（2）キャッシュ・タグの読み出し

検索したい命令キャッシュのブロックのICTRnレジスタを読み出します。読み出したデータのビット31-12がタグを示し，ビット11，10がサブブロックのバリッド・ビットに対応します。

なお，読み出すときはIN.W命令を使用してください。

ICTRから読み出したデータ

（３）キャッシュ・タグの書き込み

キャッシュ・タグ，バリッド・ビットを指定したデータを，検索したい命令キャッシュのブロックのICTRnレジスタに書き込みます。これにより，キャッシュのタグを変更することができます。このときに書き込んだブロックの分岐履歴（ABcond命令を使用）は消去されます。

なお，書き込むときはOUT.W命令を使用してください。

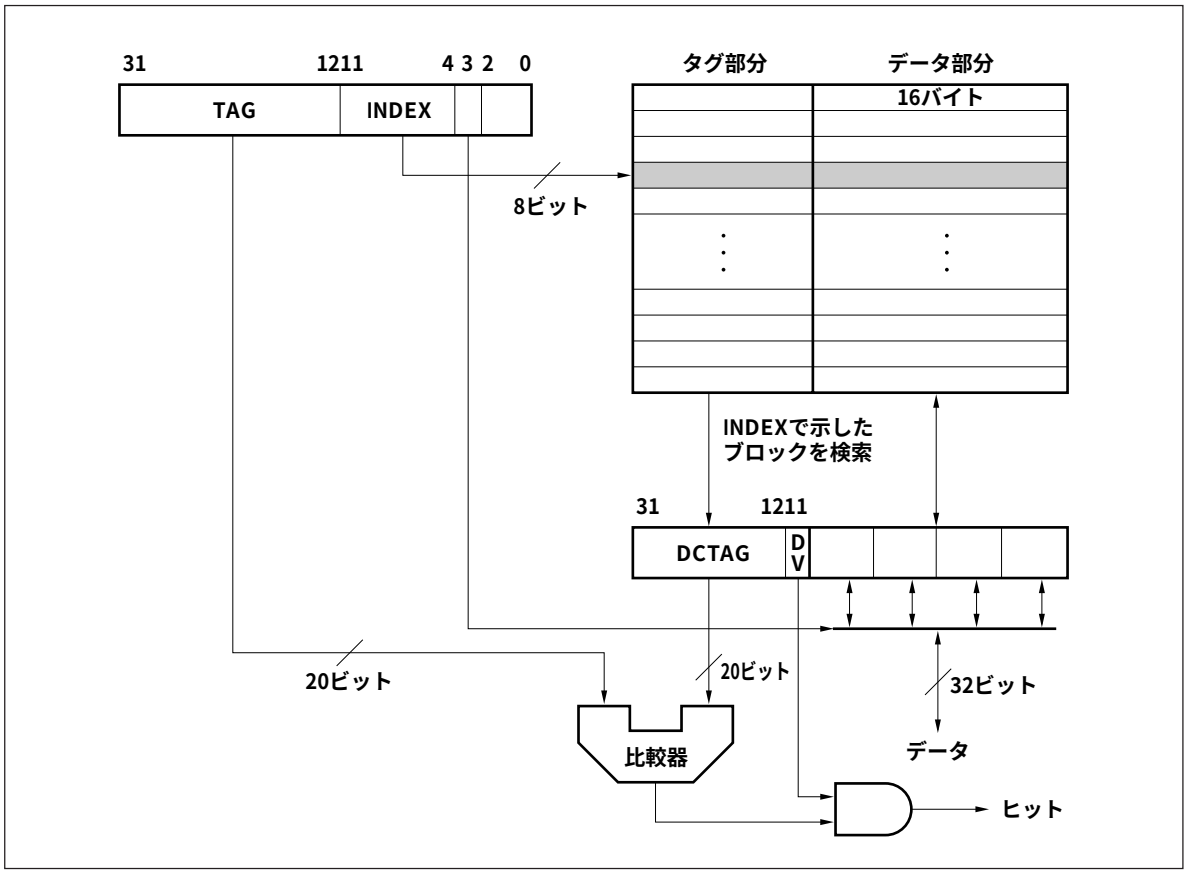
ICTRに書き込むデータ

7.1.3　データ・キャッシュ

データ・キャッシュ・メモリは256個の16バイト・ブロックで構成され，合計4 Kバイトの容量です。各ブロックはタグとバリッド・ビットがあります。バリッド・ビットはブロックの内容が有効か，無効かを示し，キャッシュ・ミスが発生した場合には，ブロック単位のリフィルを行います。リフィル動作は，V830ファミリがデータ・リード時のキャッシュ・ミスが発生したときだけ行います（ライト・スルー方式）。ライト時にはリフィル動作は行われません。

なお，データ・キャッシュにキャッシングされるデータは，キャッシャブル領域にあるデータです。内蔵データRAMやアンキャッシャブル領域のデータはキャッシングされません。

図７－３　データ・キャッシュの構成



７.１.４　データ・キャッシュ・タグ検索機能

V830ファミリにはデータ・キャッシュにキャッシングされているデータのタグを検索することができます。このタグからキャッシングされているデータのアドレスを生成して，キャッシングされているデータを知ることができます。

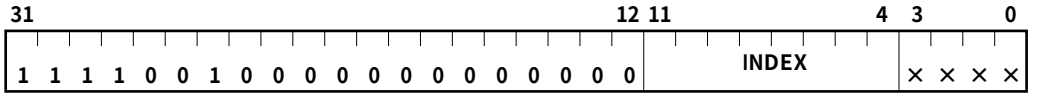
タグの検索はDCTRレジスタを使用します。DCTRレジスタはI/O空間（F2000000H-F2000FFFH）にマップされており，全部で256個あります。各レジスタはDCTR0-DCTR255の番号が付けられて，ビット3-0が０となるアドレス上にマップされています。また，この番号はキャッシュのブロック番号に対応しています。

（1）データ・キャッシュ・タグ・レジスタ

データ・キャッシュのタグを検索するときに使用するレジスタです。

なお，検索するときはIN.W命令，OUT.W命令を使用してください。

DCTRのアドレス指定方法　（F2000×××H）



×：Don't care

ビット位置	ビット名	意　　味
11-4	INDEX	Index 内蔵データ・キャッシュ・タグのアドレスを指定します。

DCTRの内容

ビット位置	ビット名	意　　味
31-12	DCTAG	Data Cache Tag データ・キャッシュのインデクスで指定したブロックのタグ。
11	DV	Data Cache Valid Bit インデクスで指定したブロックが有効であることを示します。 DV = 0：無効 DV = 1：有効（ブロックはDCTAGで示した外部メモリの内容と一致）
10-0	RFU	予約フィールドです（０に固定してください）。

（2）キャッシュ・タグの読み出し

検索したいデータ・キャッシュのブロックのDCTRnレジスタを読み出します。読み出したデータのビット31-12がタグを示し，ビット11がバリッド・ビットに対応します。

なお，読み出すときはIN.W命令を使用してください。

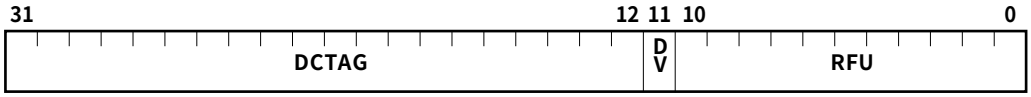
DCTRから読み出したデータ

（3）キャッシュ・タグの書き込み

キャッシュ・タグ，バリッド・ビットを指定したデータを，検索したいデータ・キャッシュのブロックのDCTRnレジスタに書き込みます。これにより，キャッシュのタグを変更できます。

なお，書き込むときはOUT.W命令を使用してください。

DCTRに書き込むデータ

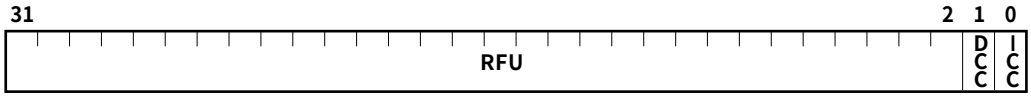


7.1.5　キャッシュ・メモリ・コントロール・レジスタ

キャッシュ・クリアを制御するためのレジスタです。このレジスタは書き込み専用です。読み出しを行った場合には，ゼロが読み出されます。

なお，書き込むときはOUT.W命令を使用してください。

CMCR（FFFFFFF4H）



ビット位置	ビット名	意　　味
31-2	RFU	予約フィールドです（０に固定してください）。
1	DCC	Data Cache Clear このビットに１を書き込むと，データ・キャッシュがクリアされます。 外部バス・マスタ（DMAなど）により，外部メモリにデータを転送したあと，そのデータをアクセスする前にデータ・キャッシュをクリアしてください。DMA転送先がアンキャッシャブル領域の場合は，データ・キャッシュをクリアする必要はありません。
0	ICC	Instruction Cache Clear このビットに１を書き込むと，命令キャッシュがクリアされます。 外部メモリにプログラムを転送したあと，そのプログラムを実行する前に命令キャッシュをクリアしてください。同時に分岐履歴もクリアしてください。

７.２　内蔵RAM

７.２.１　命令RAM

内蔵命令RAMは，メモリ空間のFE000000H-FE000FFFHに割り付けられています。この空間から命令をフェッチするには，１サイクルで命令のフェッチができます。内蔵命令RAMに命令列を配置すると，外部メモリにアクセスを行わずに命令フェッチができます。外部メモリと内蔵命令RAM間の命令転送に，BILD, BIST命令を使用します。

なお，内蔵命令RAMにロード，ストア（LD, ST）命令を使用して，アクセスすることはできません。LD, ST命令を使用してアクセスした場合には，動作保証はできません。

★ ７.２.２　命令RAM検索機能（V830, V831）

V830, V831は命令RAMに格納されている命令を読み書きする機能があります。この機能を使用すると，低速ですが命令の書き込み，読み出しができます。

命令RAMは，I/O空間のFE000000H-FE000FFFHより参照できます。ただし，この空間に命令の書き込みを行うと，書き込みを行った部分の分岐履歴は消去されます。

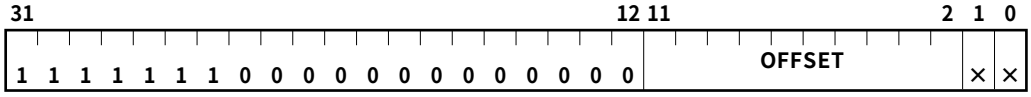
注意　V832では命令RAM検索機能を用いて，IN.WおよびOUT.W命令による命令RAMへの読み書きはできません。したがって命令RAMの命令フェッチ以外のソフトウエアによるアクセスは，BILD, BIST命令により４ワード単位で行ってください。

（1）命令RAMレジスタ

命令RAMの内容を読み出すときに使用するレジスタです。

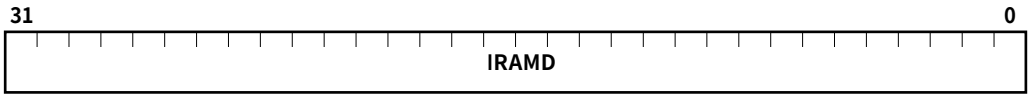
なお，これらのレジスタにアクセスするときはIN.W命令，OUT.W命令を使用してください。

IRAMRのアドレス指定方法（FE000×××H）



ビット位置	ビット名	意　　味
11-2	OFFSET	Offset 内蔵命令RAMのアドレスを指定します。

IRAMRの内容



ビット位置	ビット名	意　　味
31-0	IRAMD	Instruction RAM Data 内蔵命令RAMの内容

（2）命令RAMの検索

IN.W命令を使用して，読み出したい命令RAMのアドレスから読み込みます。

（3）命令RAMへの書き込み

書き込みたい命令を，命令RAMのアドレスにOUT.W命令を使用して書き込みます。

7.2.3　データRAM

内蔵データRAMは，メモリ空間の00000000H-00000FFFHに割り付けられています。この空間からデータのロード／ストアは，１サイクルで処理できます。外部メモリと内蔵データRAM間のデータ転送には，BDLD, BDST命令を使用すると高速に転送できます。

なお，内部データRAMからの命令フェッチはできません。内部データRAMから命令フェッチを行った場合には，動作保証はできません。

注意　内蔵RAM領域とアドレスが重なる外部メモリは使用できません。

第8章　リセット

$\overline{\text{RESET}}$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が初期化されます。

8.1　イニシャライズ

$\overline{\text{RESET}}$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，システム・レジスタおよび内部レジスタは，表８－１に示すような状態になります。

$\overline{\text{RESET}}$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。各種のレジスタはプログラムの中で必要に応じて設定してください。

表８－１　リセット後のレジスタの状態

	レジスタ	略　号	リセット後の状態
システム・レジスタ	プログラム・カウンタ	PC	FFFFFFF0H
	例外／割り込み時状態退避レジスタ	EIPC	不定
		EIPSW	不定
	NMI/二重例外時状態退避レジスタ	FEPC	不定
		FEPSW	不定
	例外要因レジスタ	ECR	0000FFF0H
	プログラム・ステータス・ワード	PSW	00008000H
	プロセッサIDレジスタ	PIR	00008300H
	タスク・コントロール・ワード	TKCW	000000E0H
	ディバグ例外時状態退避レジスタ	DPC	不定
		DPSW	不定
	ハードウエア・コンフィグレーション・コントロール・ワード	HCCW	00000000H
内部レジスタ	PLLコントロール・レジスタ[注]	PLLCR	0000000×H
	キャッシュ・メモリ・コントロール・レジスタ	CMCR	00000000H
	命令キャッシュ・タグ・レジスタ	ICTR	XXXXX000H
	データ・キャッシュ・タグ・レジスタ	DCTR	XXXXX000H
	命令RAMレジスタ	IRAMR	不定

注　CMODEによって，リセット後の状態が異なります。

8.2　起　　動

V830ファミリはリセットによりFFFFFFF0H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。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NPビットに0を設定してください。

注意　V830ではFFFFFFFBH以降に命令を置かないでください。

第９章　パイプライン

V830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以下に示します。

IF（Instruction Fetch）　… 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。

RF（Register Fetch）…… 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（Execute）　…… デコードした命令を実行します。

DF（Data Fetch）　…… 演算フラグの生成，メモリ（キャッシュ）の読み出しを行います。

WB（Write Back）　…… 実行結果のレジスタ・ファイルへの書き込み，メモリ（キャッシュ）への書き込みを行います。

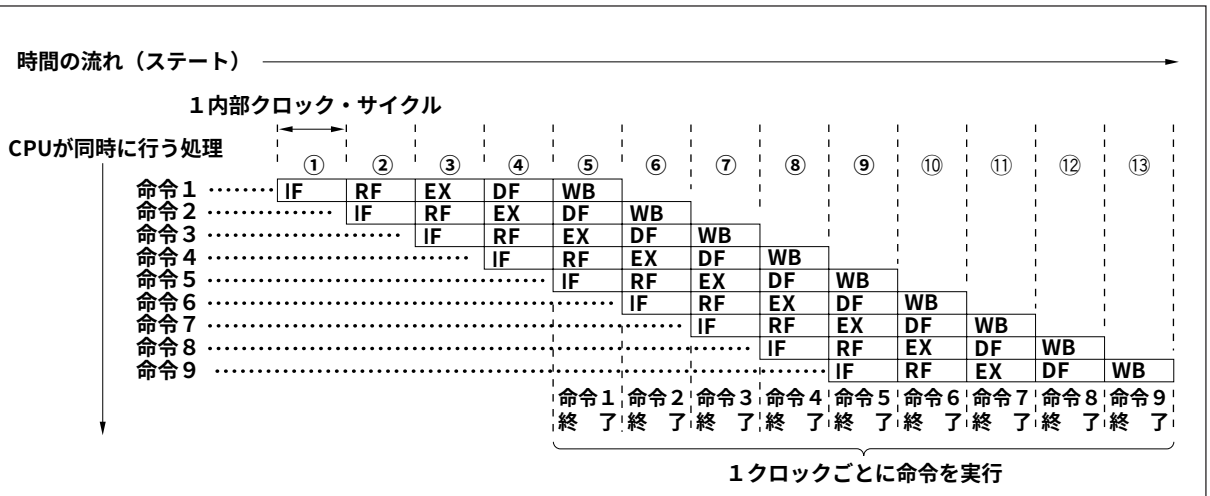
９.１　動作概要

V830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９－１に示します。

図９－１　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。

９.２　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。

ここで用いたパイプラインでは内部ブロックと外部ブロックの周波数比を２対１としています。

また，ライト・バッファによる待ち合わせ，内部ブロックと外部ブロックの同期化のためにおこる待ち合わせは考慮されていません。パイプラインの待ち合わせが外部バスのアクセス頻度，命令の組み合わせなどにより発生することがあります。具体的なプログラムのパイプライン動作はシミュレータなどによりご確認ください。

９.２.１　ロード命令

[対象の命令]　LD.B, LD.H, LD.W

[パイプライン]　ここではロード命令の基本的な流れを示します。

	①	②	③	④	⑤	⑥
ロード命令	IF	RF	EX	DF	WB	
次命令		IF	RF	EX	DF	WB

[説　　明]　アドレス計算はEXステージで行います。ロード・データはDFステージにおいてデータ・キャッシュまたはデータ・メモリから読み出します。また，DFステージではデータ・キャッシュのヒット／ミス判定も行います。

９.２.２　ストア命令

[対象の命令]　ST.B, ST.H, ST.W

[パイプライン]　ここではストア命令の基本的な流れを示します。

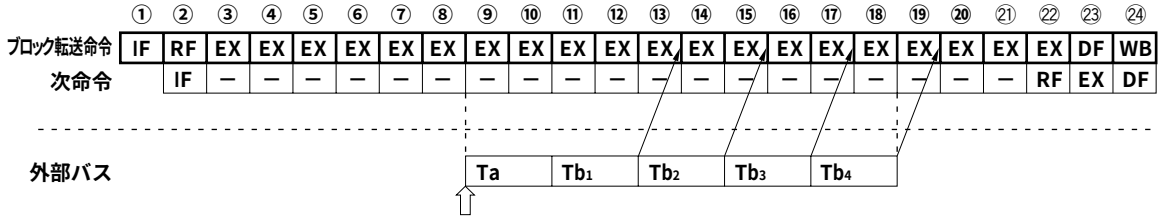
	①	②	③	④	⑤	⑥
ストア命令	IF	RF	EX	DF	WB	
次命令		IF	RF	EX	DF	WB

[説　　明]　アドレス計算はEXステージで行います。データ・キャッシュのヒット／ミス判定は，DFステージで行います。データ・キャッシュまたはデータ・メモリへのストアは，WBステージで行います。

９.２.３　ブロック転送命令

（１）BILD, BDLD

［パイプライン］　ここではブロック転送命令の基本的な流れを示します。



［説　　明］　⑭のEXステージでは，Tb_1で読んだデータを内蔵RAMへ書き込みます。

⑯のEXステージでは，Tb_2で読んだデータを内蔵RAMへ書き込みます。

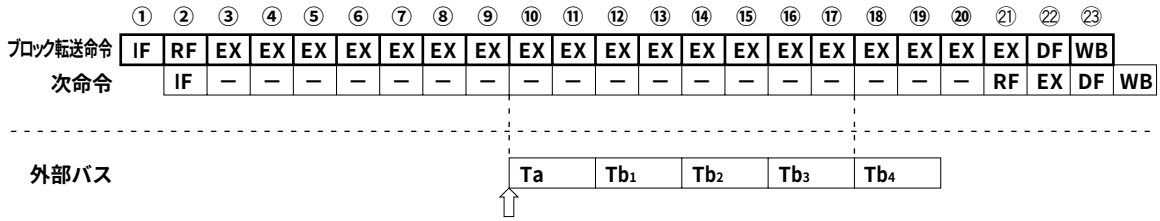
⑱のEXステージでは，Tb_3で読んだデータを内蔵RAMへ書き込みます。

⑳のEXステージでは，Tb_4で読んだデータを内蔵RAMへ書き込みます。

外部バスからデータを取り込む最後のバス・サイクル（Tb_4）の終了から３クロック後に，パイプラインのホールドが解除されます。

（２）BIST, BDST

［パイプライン］　ここではブロック転送命令の基本的な流れを示します。



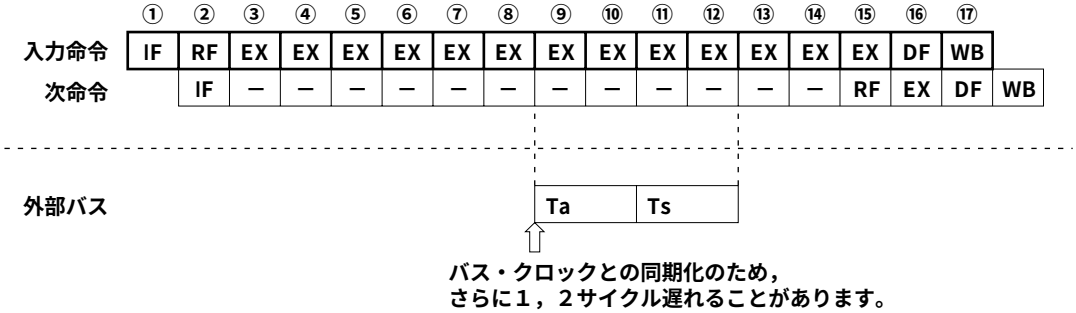
［説　　明］　外部バスにデータを出力する最後のバス・サイクル（Tb_4）の始めから３クロック後にパイプラインのホールドが解除されます。

９.２.４　入出力命令

（１）　入力命令

［対象の命令］　IN.B, IN.H, IN.W

［パイプライン］　ここでは入力命令の基本的な流れを示します。



［説　　明］　外部バスからデータを取り込むまでパイプラインを停止（ホールド）します。

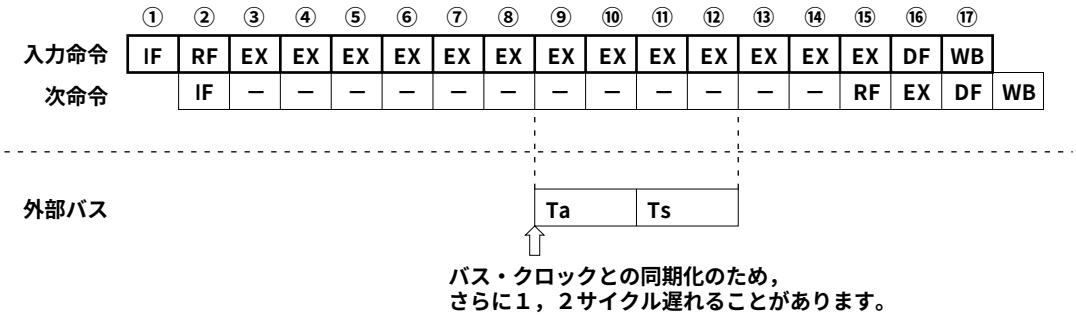
外部バスからデータを取り込むバス・サイクルの終了から２クロック後に，パイプラインのホールドを解除します。

DFステージにおいて外部バスから読み込んだデータを取り込みます。そして，WBステージでレジスタ・ファイルに書き込みます。

（２）出力命令

［対象の命令］　OUT.B, OUT.H, OUT.W

［パイプライン］　ここでは出力命令の基本的な流れを示します。



［説　　明］　外部バスにデータを出力するバス・サイクルの終了から２クロック後にパイプラインのホールドを解除します。

9.2.5　算術演算命令（乗算命令，除算命令を除く）

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

[パイプライン]　ここでは算術演算命令（乗除算命令を除く）の基本的な流れを示します。

	①	②	③	④	⑤	⑥
算術演算命令	IF	RF	EX	DF	WB	
次命令		IF	RF	EX	DF	WB

[説　　明]　演算結果はEXステージで得られます。フラグもEXステージで生成されます（MOV, MOVHI, MOVEAは除く）。
レジスタ・ファイルへの書き込みはWBステージで行います（CMPは除く）。

9.2.6　乗算命令

[対象の命令]　MUL, MULU

[パイプライン]　ここでは乗算命令の基本的な流れを示します。

	①	②	③	④	⑤	⑥	⑦
乗算命令	IF	RF	EX	EX	DF	WB	
次命令		IF	―	RF	EX	DF	WB

[説　　明]　演算結果は上位ワードと下位ワードを分けて出力します。
DFステージでは上位ワードを出力し，レジスタへの書き込みを行います。
フラグの生成もDFステージで行います。WBステージでは下位ワードを出力し，レジスタへの書き込みを行います。

9.2.7　除算命令

（1）DIV

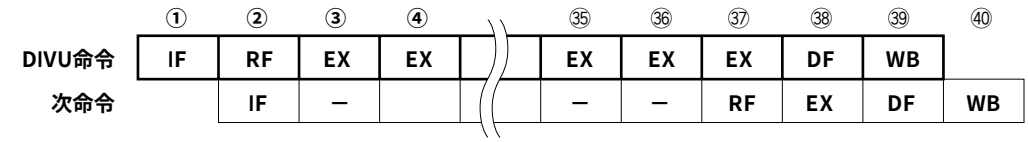
[パイプライン]　ここではDIV命令の基本的な流れを示します。

	①	②	③	④		㊲	㊳	㊴	㊵	㊶	㊷
DIV命令	IF	RF	EX	EX		EX	EX	EX	DF	WB	
次命令		IF	―			―	―	RF	EX	DF	WB

［説　　明］　DIV命令は，EXステージの36サイクル目（㊳）まで次命令を停止します。EXステージの37サイクル目（㊴）で，次命令をRFステージから再開します。
剰余の出力はEXステージの35サイクル目（㊲），書き込みはEXステージの37サイクル目（㊴）で行います。商の出力はEXステージの37サイクル目（㊴），書き込みはWBステージで行います。フラグはDFステージで生成されます。

（２）DIVU

［パイプライン］　ここではDIVU命令の基本的な流れを示します。

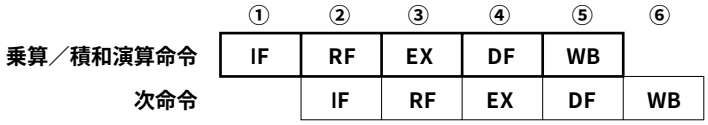


［説　　明］　DIVU命令は，EXステージの34サイクル目（㊱）まで次命令を停止します。EXステージの35サイクル目（㊲）で，次命令をRFステージから再開します。
剰余の出力はEXステージの34サイクル目（㊱），書き込みはDFステージで行います。商の出力はEXステージの35サイクル目（㊲），書き込みはWBステージで行います。フラグはDFステージで生成されます。

９.２.８　乗算／積和演算命令

［対象の命令］　MUL3, MULI, MULT3, MAC3, MACI, MACT3

［パイプライン］　ここでは乗算／積和演算命令の基本的な流れを示します。

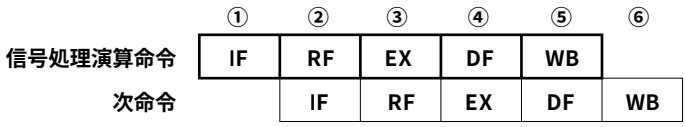


［説　　明］　演算結果はWBステージで生成されます。
RFステージでは第１，第２オペランドのレジスタ・ファイルの読み出しを行います。第３オペランドの値がフォワーディングされない場合は，構造ハザード（２）が発生します。詳細は**９.３　パイプラインの乱れ**を参照してください。

9.2.9　信号処理演算命令

[対象の命令]　SATADD3, SATSUB3, MIN3, MAX3

[パイプライン]　ここでは信号処理演算命令の基本的な流れを示します。

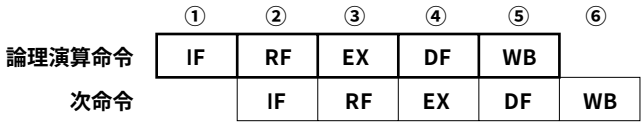


[説　　明]　フラグはEXステージで生成されます。ただし，MIN3, MAX3命令はフラグを生成しません。演算結果はDFステージで得られます。

9.2.10　論理演算命令

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

[パイプライン]　ここでは論理演算命令の基本的な流れを示します。

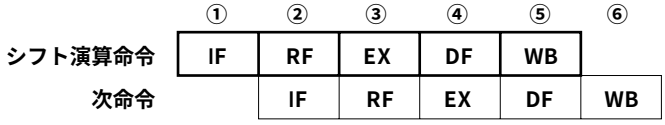


[説　　明]　演算結果はEXステージで生成します。フラグはDFステージで生成します。

9.2.11　シフト演算命令

[対象の命令]　SHL, SHR, SAR, SHLD3, SHRD3

[パイプライン]　ここではシフト演算命令の基本的な流れを示します。



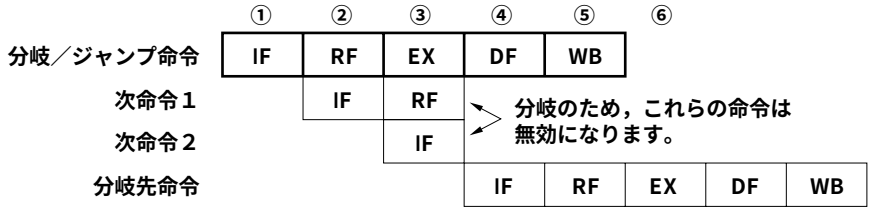
[説　　明]　フラグと演算結果はDFステージで生成します。ただし，SHLD3, SHRD3命令はフラグを生成しません。

９.２.12　分岐／ジャンプ命令

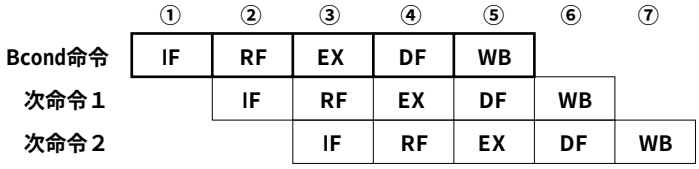
[対象の命令]　Bcond命令（BGT, BGE, BLT, BLE, BH, BNL, BL, BNH, BE, BNE, BV, BNV, BN, BP, BC, BNC, BZ，BNZ, BR, NOP）, JMP, JR

[パイプライン]　ここでは分岐／ジャンプ命令の基本的な流れを示します。

（ａ）ジャンプ命令またはBcond命令の分岐条件が成立した場合



（ｂ）Bcond命令の分岐条件が成立しない場合



[説　　明]　**（ａ）ジャンプ命令またはBcond命令の分岐条件が成立した場合**

EXステージで分岐先アドレスをPCに入れ，分岐します。

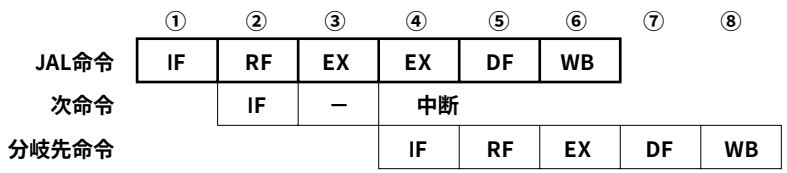
（ｂ）Bcond命令の分岐条件が成立しない場合

EXステージで分岐不成立と判定し，後続命令をそのまま実行します。

９.２.13　ジャンプ・アンド・リンク命令

[対象の命令]　JAL

[パイプライン]　ここではJAL命令の基本的な流れを示します。

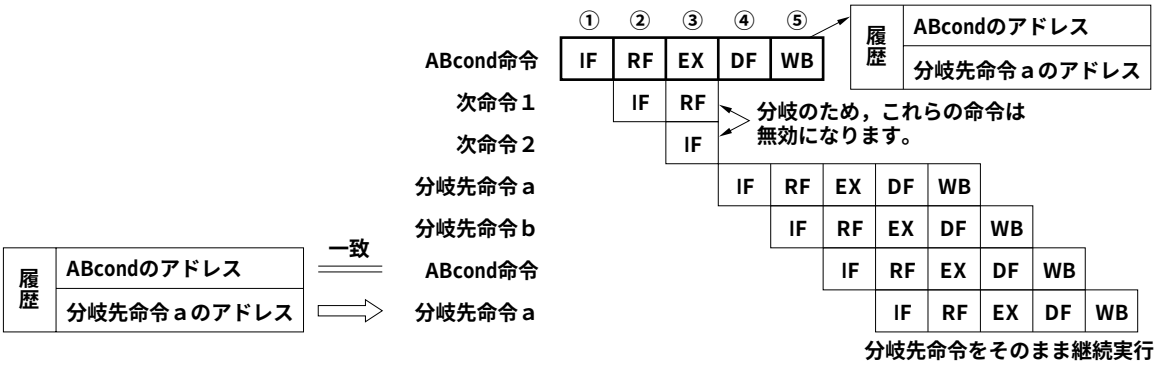


[説　　明]　１回目のEXステージで分岐先アドレスをPCに入れます。２回目のEXステージでリンクPCを計算し，WBステージでr31へその値を書き込みます。

９.２.14　高速分岐命令

[対象の命令]　ABcond命令（ABGT, ABGE, ABLT, ABLE, ABH, ABNL, ABL, ABNH, ABE, ABNE, ABV, ABNV, ABN, ABP, ABC, ABNC, ABZ, ABNZ, ABR）

[パイプライン]　ここではABcond命令の基本的な流れを示します。

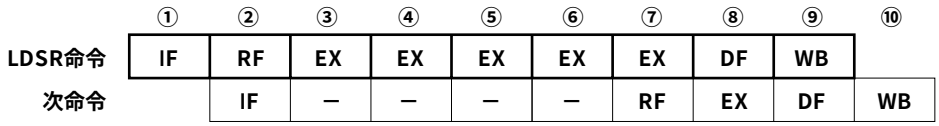


[説　　明]　ABcond命令のEXステージで条件が成立し，分岐と判定されると，初回分岐時には，まだ分岐履歴がないために通常の分岐命令と同様の動きをします。ただし，WBステージにおいてABcond命令のアドレスと分岐先アドレスを分岐履歴に書き込みます。ABcond命令の２回目以降分岐時（分岐履歴がクリアされていないとき）は，分岐履歴のアドレスとPCの値の一致により，分岐履歴に格納されている分岐先命令のアドレスがABcond命令の次のPCの値として設定されます。

９.２.15　特殊命令

（１）　LDSR

[パイプライン]　ここではLDSR命令の基本的な流れを示します。

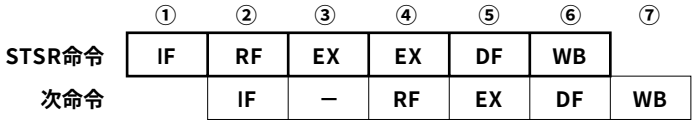


[説　　明]　システム・レジスタの新しい値は次命令から適用されます。次命令はEXステージの４サイクル目（⑥）まで停止します。

LDSR命令直後に条件分岐命令があってもフラグ・ハザードにはなりません。

（２）STSR

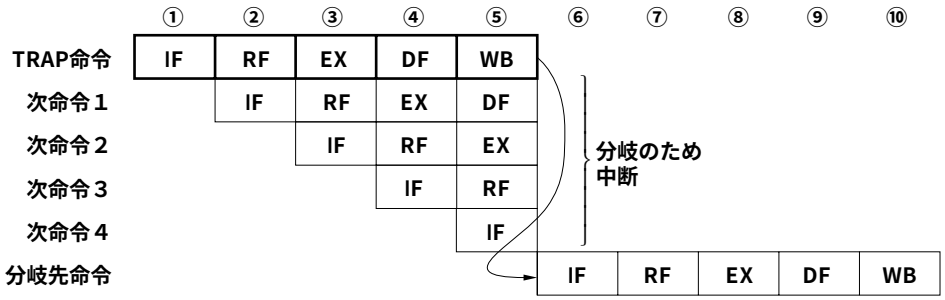
［パイプライン］　ここではSTSR命令の基本的な流れを示します。



［説　　明］　最初のEXステージ（③）でシステム・レジスタ（EIPC, EIPSW, FEPC, FEPSW, PPC, PPSW）の読み出しを行います。WBステージでシステム・レジスタ（ECR, PSW, PIR, TKCW, HCCW）の読み出しを行います。レジスタ・ファイルへの書き込みがWBステージで行われるため，STSR命令直後にSTSR命令の結果を使用する場合は，レジスタ・ハザードになります。詳細は**９.３　パイプラインの乱れ**を参照してください。

（３）TRAP

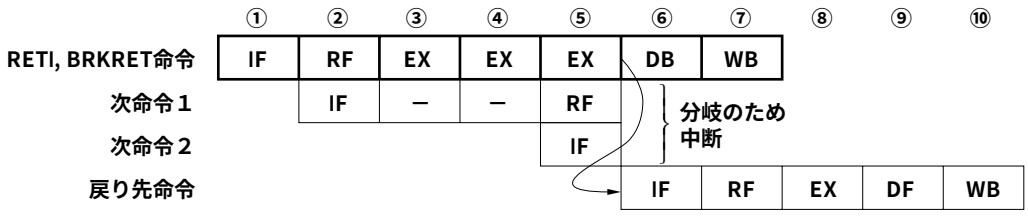
［パイプライン］　ここではTRAP命令の基本的な流れを示します。



［説　　明］　TRAP命令以前の命令で例外などが発生していないことを確認してから分岐するために，TRAP命令のWBステージが終了してから分岐します。

（４）　RETI, BRKRET

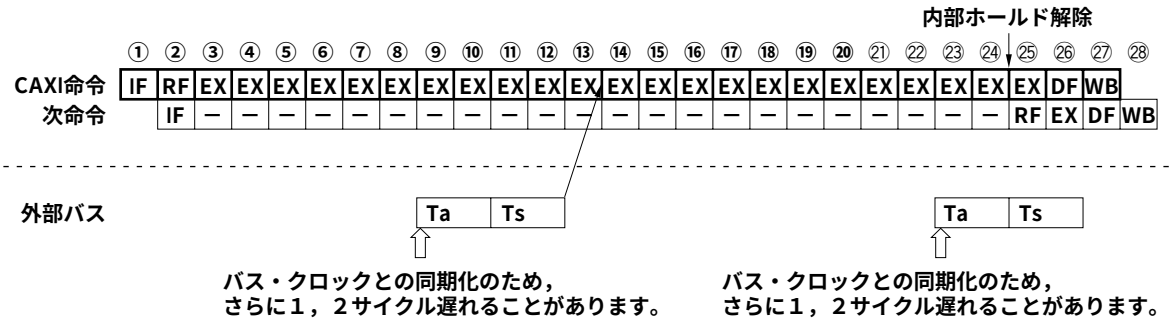
［パイプライン］　ここではRETI, BRKRET命令の基本的な流れを示します。



[説　　明]　EXステージの３サイクル目（⑤）で，戻り先アドレスをPCに設定して分岐します。次命令はEXステージの２サイクル目（④）まで停止します。

（５）CAXI

[パイプライン]　ここではCAXI命令の基本的な流れを示します。



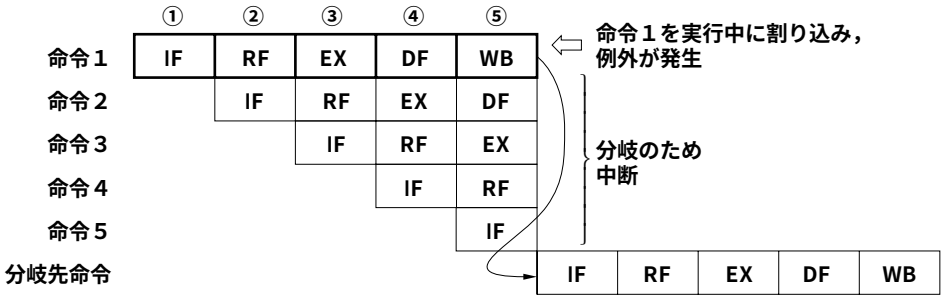
[説　　明]　外部バスにデータを出力する最初のバス・サイクルから２サイクル後（⑬）に，パイプラインのホールドが解除されます。

解除された次のEXステージ（⑭）で外部バスから読み込んだデータを取り込み，その次のEXステージ（⑮）で読み込んだデータと比較し，また次のEXステージ（⑯）で比較結果をストアします。

９.２.16　アドレス・トラップ，割り込み

[対象の処理]　割り込み，不正コード例外，ゼロ除算例外

[パイプライン]　ここでは割り込み／例外処理の基本的な流れを示します。



[説　　明]　命令１の実行中に割り込みが発生した場合は，命令１をWBステージまで完了してから割り込み処理に移ります。

命令１の実行中に例外が発生した場合も，例外の発生した命令１のWBステージまで完了してから例外処理に移ります。

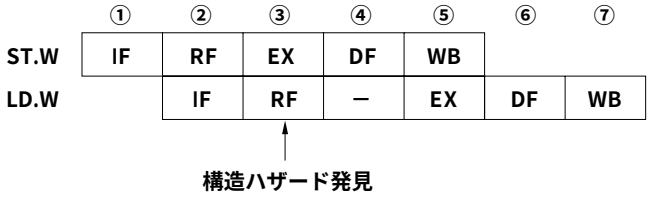
９.３　パイプラインの乱れ

９.３.１　構造ハザード（１）

[対象の処理]　後続の命令が同じハードウエアを同時にアクセスするタイミングが発生する可能性があるときに発生します。

[プログラム例]　ST.W　　r3, 200［r2］
LD.W　　100［r1］, r4

[パイプライン]

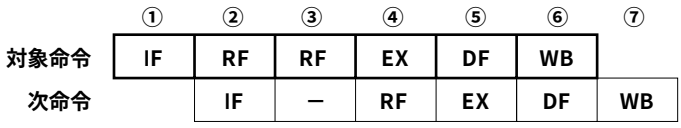


[説　　明]　後続命令が同じハードウエアを同時にアクセスし誤動作することを防ぐため，後続命令の実行を途中で停止し実行タイミングをずらします。
プログラム例のストア命令（ST.W）ではWBステージで内蔵データRAM，データ・キャッシュへのアクセスが発生し，後続命令であるロード命令（LD.W）ではDFステージで内蔵データRAM，データ・キャッシュへのアクセスが発生するため，同じサイクルで同じバスを使用することになります。このタイミングをずらすため，LD.W命令がハザード機能により命令を１サイクル停止します。

９.３.２　構造ハザード（２）

[対象の処理]　MAC3, MACT3, SHLD3, SHRD3において，第３オペランドの値が前命令からフォワーディングされないときに発生します。

[パイプライン]



[説　　明]　対象命令の最初のRFステージ（②）では第１オペランド，第２オペランドを読み出し，第３オペランドの読み出しは行いません。したがって，第３オペランドがレジスタ・フォワーディングの対象となっていない場合は，再度RFステージが発生し第３オペランドの読み出しを行います。そのため次命令は停止し，前命令がEXステージを実行するときに次命令のRFステージを実行します（④）。

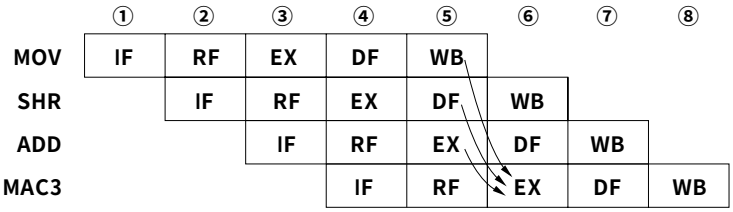
９.３.３　レジスタ・フォワーディング

[対象の処理]　レジスタ・ファイルへの書き込みを行うWBステージより前に後続命令が演算の結果を使用するときに発生します。

[プログラム例]

```
MOV     r8, r5
SHR     #1, r4
ADD     r2, r3
MAC3    r5, r4, r3
```

[パイプライン]

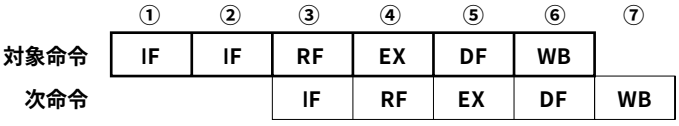


[説　　明]　後続命令で使用する演算結果をEX, DF, WBの各ステージから後続命令のEXステージへ中継します（フォワーディング機能またはバイパス機能）。プログラム例ではMOV命令の演算結果（r5）をMAC3（MOVのWB→MAC3のEX）へ，SHR命令の演算結果（r4）をMAC3（SHRのDF→MAC3のEX）へ，ADDの演算結果（r3）をMAC3（ADDのEX→MAC3のEX）へフォワーディングします。この機能により後続命令は前命令のWBステージの終了を待つことなく実行を開始することができます。

９.３.４　命令コード・ハザード

[対象の処理]　ワード境界にまたがる32ビット命令に分岐（分岐／ジャンプ命令による分岐，割り込みによる分岐）したときに発生します。

[パイプライン]



[説　　明]　CPUは１ワードごとに命令キャッシュまたは命令RAMから命令フェッチを行います。したがって分岐直後に実行するワード境界にまたがる32ビット命令は，１回目のIFステージでは前半のコードしかフェッチすることができません。後半のコード・フェッチのために２回目のIFステージを起動するため，次の命令のIFステージは前命令のコード・フェッチが終了するまで待たされます。

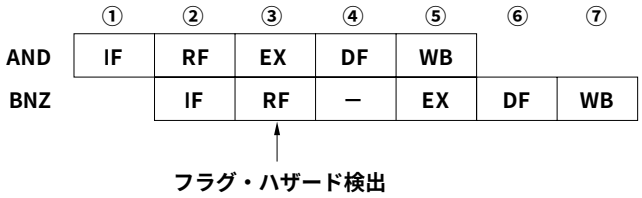
９.３.５　フラグ・ハザード

[対象の処理]　DFステージ以降でフラグを生成する命令（論理演算命令はDFステージ，乗算命令はWBステージ）後の条件分岐命令，SETF命令のときに発生します。

[プログラム例]

```
AND     #1, r3
BNZ     skip
```

[パイプライン]



[説　　明]　EXステージでフラグ生成されないため後続命令は，フラグ・フォワーディングまで実行を停止します。

付録A　命令一覧

A.1　命令形態

A.1.1　V810™と共通の命令

ロード／ストア	LD.B	Load Byte
	LD.H	Load Halfword
	LD.W	Load Word
	ST.B	Store Byte
	ST.H	Store Halfword
	ST.W	Store Word
整数算術演算	MOV	Move
	MOVHI	Add
	ADD	Add
	ADDI	Add Immediate
	MOVEA	Add
	SUB	Subtract
	MUL	Multiply
	MULU	Multiply Unsigned
	DIV	Divide
	DIVU	Divide Unsigned
	CMP	Compare
	SETF	Set Flag Condition
論理演算	OR	Or
	ORI	Or Immediate
	AND	And
	ANDI	And Immediate
	XOR	Exclusive Or
	XORI	Exclusive Or Immediate
	NOT	Not
	SHL	Shift Logical Left
	SHR	Shift Logical Right
	SAR	Shift Arithmetic Right

入出力	IN.B	Input Byte from Port
	IN.H	Input Halfword from Port
	IN.W	Input Word from Port
	OUT.B	Output Byte to Port
	OUT.H	Output Halfword to Port
	OUT.W	Output Word to Port
プログラム制御	JMP	Jump Register
	JR	Jump Relative
	JAL	Jump and Link
	BGT	Branch on Greater than signed
	BGE	Branch on Greater than or Equal signed
	BLT	Branch on Less than signed
	BLE	Branch on Less than or Equal signed
	BH	Branch on Higher
	BNH	Branch on Not Higher
	BL	Branch on Lower
	BNL	Branch on Not Lower
	BE	Branch on Equal
	BNE	Branch on Not Equal
	BV	Branch on Overflow
	BNV	Branch on No Overflow
	BN	Branch on Negative
	BP	Branch on Positive
	BC	Branch on Carry
	BNC	Branch on No Carry
	BZ	Branch on Zero
	BNZ	Branch on Not Zero
	BR	Branch Always
	NOP	Not always
特殊	LDSR	Load to System Register
	STSR	Store Contents of System Register
	TRAP	Trap
	RETI	Return from Trap or Interrupt
	CAXI	Compare and Exchange Interlocked
	HALT	Halt

A.1.2　V810に対して追加した命令

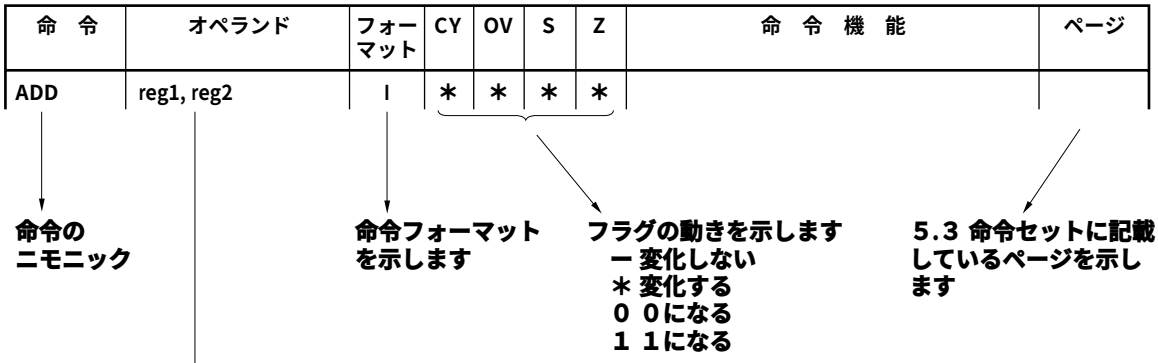
内蔵メモリ対応	BILD	Block Instruction Load to internal memory
	BDLD	Block Data Load to internal memory
	BIST	Block Instruction of internal memory Store
	BDST	Block Data of internal memory Store
V830制御	EI	Enable Interrupt
	DI	Disable Interrupt
	STBY	Standby
	BRKRET	Break Return
マルチメディア対応	MUL3	Multiply
	MAC3	Multiply and Accumulate
	MULI	Multiply Immediate
	MACI	Multiply and Accumulate Immediate
	MULT3	Multiply with Truncation
	MACT3	Multiply and Accumulate with Truncation
	SATADD3	Saturated Add
	SATSUB3	Saturated Subtract
	MIN3	Minimum
	MAX3	Maximum
	SHLD3	Shift Left Double word
	SHRD3	Shift Right Double word
	ABGT	Advanced Branch on Greater than signed
	ABGE	Advanced Branch on Greater than or Equal signed
	ABLT	Advanced Branch on Less than signed
	ABLE	Advanced Branch on Less than or Equal signed
	ABH	Advanced Branch on Higher
	ABNH	Advanced Branch on Not Higher
	ABL	Advanced Branch on Lower
	ABNL	Advanced Branch on Not Lower
	ABE	Advanced Branch on Equal
	ABNE	Advanced Branch on Not Equal
	ABV	Advanced Branch on Overflow
	ABNV	Advanced Branch on No Overflow
	ABN	Advanced Branch on Negative
	ABP	Advanced Branch on Positive
	ABC	Advanced Branch on Carry
	ABNC	Advanced Branch on No Carry
	ABZ	Advanced Branch on Zero
	ABNZ	Advanced Branch on Not Zero
	ABR	Advanced Branch Always

A.2　命令一覧（アルファベット順）

命令ニモニックの一覧をアルファベット順に示します。

凡　例

命　令	オペランド	フォーマット	CY	OV	S	Z	命　令　機　能	ページ
ADD	reg1, reg2	I	＊	＊	＊	＊		



オペランドの略号

略　号	意　　味
reg1	汎用レジスタ（ソース・レジスタとして使用します。）
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します。一部ソース・レジスタとしても使用します。）
reg3	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します。一部ソース・レジスタとしても使用します。）
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 adr	トラップ・ベクタに対応するトラップ・ハンドラ・アドレス

命　令	オペランド	フォーマット	CY	OV	S	Z	命　令　機　能	ページ
ABC	disp9	III	－	－	－	－	高速条件分岐（if Carry）。PC相対分岐。	46
ABE	disp9	III	－	－	－	－	高速条件分岐（if Equal）。PC相対分岐。	
ABGE	disp9	III	－	－	－	－	高速条件分岐（if Greater than or Equal）。PC相対分岐。	
ABGT	disp9	III	－	－	－	－	高速条件分岐（if Greater than）。PC相対分岐。	
ABH	disp9	III	－	－	－	－	高速条件分岐（if Higher）。PC相対分岐。	
ABL	disp9	III	－	－	－	－	高速条件分岐（if Lower）。PC相対分岐。	
ABLE	disp9	III	－	－	－	－	高速条件分岐（if Less than or Equal）。PC相対分岐。	
ABLT	disp9	III	－	－	－	－	高速条件分岐（if Less than）。PC相対分岐。	
ABN	disp9	III	－	－	－	－	高速条件分岐（if Negative）。PC相対分岐。	
ABNC	disp9	III	－	－	－	－	高速条件分岐（if Not Carry）。PC相対分岐。	
ABNE	disp9	III	－	－	－	－	高速条件分岐（if Not Equal）。PC相対分岐。	
ABNH	disp9	III	－	－	－	－	高速条件分岐（if Not Higher）。PC相対分岐。	
ABNL	disp9	III	－	－	－	－	高速条件分岐（if Not Lower）。PC相対分岐。	
ABNV	disp9	III	－	－	－	－	高速条件分岐（if Not Overflow）。PC相対分岐。	
ABNZ	disp9	III	－	－	－	－	高速条件分岐（if Not Zero）。PC相対分岐。	
ABP	disp9	III	－	－	－	－	高速条件分岐（if Positive）。PC相対分岐。	
ABR	disp9	III	－	－	－	－	高速無条件分岐（Always）。PC相対分岐。	
ABV	disp9	III	－	－	－	－	高速条件分岐（if Overflow）。PC相対分岐。	
ABZ	disp9	III	－	－	－	－	高速条件分岐（if Zero）。PC相対分岐。	
ADD	reg1, reg2	I	＊	＊	＊	＊	加算。reg2にreg1を加算し，その結果をreg2に格納します。	48
	imm5, reg2	II	＊	＊	＊	＊	加算。reg2にimm5をワード長まで符号拡張した値を加算し，その結果をreg2に格納します。	

命　令	オペランド	フォーマット	CY	OV	S	Z	命　令　機　能	ページ
ADDI	imm16, reg1, reg2	V	＊	＊	＊	＊	加算。reg1にimm16をワード長まで符号拡張した値を加算し，その結果をreg2に格納します。	49
AND	reg1, reg2	I	－	0	＊	＊	論理積。reg2とreg1の論理積をとり，その結果をreg2に格納します。	50
ANDI	imm16, reg1, reg2	V	－	0	0	＊	論理積。reg1とimm16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reg2に格納します。	51
BC	disp9	III	－	－	－	－	条件分岐（if Carry）。PC相対分岐。	52
BDLD	[reg1], [reg2]	VII	－	－	－	－	ブロック転送。外部メモリから内蔵データRAMに４ワードのデータを転送します。	54
BDST	[reg2], [reg1]	VII	－	－	－	－	ブロック転送。内蔵データRAMから外部メモリに４ワードのデータを転送します。	55
BE	disp9	III	－	－	－	－	条件分岐（if Equal）。PC相対分岐。	52
BGE	disp9	III	－	－	－	－	条件分岐（if Greater than or Equal）。PC相対分岐。	
BGT	disp9	III	－	－	－	－	条件分岐（if Greater than）。PC相対分岐。	
BH	disp9	III	－	－	－	－	条件分岐（if Higher）。PC相対分岐。	
BILD	[reg1], [reg2]	VII	－	－	－	－	ブロック転送。外部メモリから内蔵命令RAMに４ワードのデータを転送します。	56
BIST	[reg2], [reg1]	VII	－	－	－	－	ブロック転送。内蔵命令RAMから外部メモリに４ワードのデータを転送します。	57
BL	disp9	III	－	－	－	－	条件分岐（if Lower）。PC相対分岐。	52
BLE	disp9	III	－	－	－	－	条件分岐（if Less than or Equal）。PC相対分岐。	
BLT	disp9	III	－	－	－	－	条件分岐（if Less than）。PC相対分岐。	
BN	disp9	III	－	－	－	－	条件分岐（if Negative）。PC相対分岐。	
BNC	disp9	III	－	－	－	－	条件分岐（if Not Carry）。PC相対分岐。	
BNE	disp9	III	－	－	－	－	条件分岐（if Not Equal）。PC相対分岐。	
BNH	disp9	III	－	－	－	－	条件分岐（if Not Higher）。PC相対分岐。	
BNL	disp9	III	－	－	－	－	条件分岐（if Not Lower）。PC相対分岐。	
BNV	disp9	III	－	－	－	－	条件分岐（if Not Overflow）。PC相対分岐。	
BNZ	disp9	III	－	－	－	－	条件分岐（if Not Zero）。PC相対分岐。	
BP	disp9	III	－	－	－	－	条件分岐（if Positive）。PC相対分岐。	
BR	disp9	III	－	－	－	－	無条件分岐（Always）。PC相対分岐。	
BRKRET		IX	－	－	－	－	致命的例外処理から復帰。	58

命　令	オペランド	フォーマット	CY	OV	S	Z	命　令　機　能	ページ
BV	disp9	III	－	－	－	－	条件分岐（if Overflow）。PC相対分岐。	52
BZ	disp9	III	－	－	－	－	条件分岐（if Zero）。PC相対分岐。	
CAXI	disp16[reg1], reg2	VI	＊	＊	＊	＊	マルチプロセッサ構成のシステムにおけるプロセッサ間同期。	59
CMP	reg1, reg2	I	＊	＊	＊	＊	比較。reg2とreg1をワード長まで符号拡張した値を比較し，結果を条件フラグに示します。比較はreg2からreg1を減算することで行います。	61
	imm5, rag2	II	＊	＊	＊	＊	比較。reg2とimm5をワード長まで符号拡張した値を比較し，結果を条件フラグに示します。比較はreg2からワード長まで符号拡張したimm5を減算することで行います。	
DI		II	－	－	－	－	割り込み禁止。マスカブル割り込みを禁止します。DI命令ではノンマスカブル割り込みは禁止できません。	62
DIV	reg1, reg2	I	－	＊	＊	＊	符号付き除算。reg2をreg1で除算（符号付き）し，その商をreg2に，剰余をr30に格納します。除算は剰余の符号が被除数の符号と一致するように行われます。	63
DIVU	reg1, reg2	I	－	0	＊	＊	符号なし除算。reg2をreg1で，ともに符号なしデータとして除算し，その商をreg2に，剰余をr30に格納します。除算は剰余の符号が被除数の符号と一致するように行われます。	64
EI		II	－	－	－	－	割り込み許可。マスカブル割り込みを許可します。EI命令ではノンマスカブル割り込みは許可できません。	65
HALT		IX	－	－	－	－	プロセッサ停止。スリープ・モードに入ります。	66
IN.B	disp16[reg1], reg2	VI	－	－	－	－	ポート入力。reg1とワード長まで符号拡張したdisp16を加算して32ビット符号なしポート・アドレスを生成します。生成したポート・アドレスからバイト・データを読み出し，ワード長までゼロ拡張しreg2に格納します。	67

命　令	オペランド	フォーマット	CY	OV	S	Z	命　令　機　能	ページ
IN.H	disp16[reg1], reg2	VI	−	−	−	−	ポート入力。reg1とワード長まで符号拡張したdisp16を加算して32ビット符号なしポート・アドレスを生成します。生成したポート・アドレスからハーフワード・データを読み出し，ワード長までゼロ拡張しreg2に格納します。32ビット符号なしポート・アドレスのビット0は0にマスクされます。	67
IN.W	disp16[reg1], reg2	VI	−	−	−	−	ポート入力。reg1とワード長まで符号拡張したdisp16を加算して32ビット符号なしポート・アドレスを生成します。生成したポート・アドレスからワード・データを読み出し，reg2に格納します。32ビット符号なしポート・アドレスのビット0, 1は，0にマスクされます。	
JAL	disp26	IV	−	−	−	−	ジャンプ・アンド・リンク。現在のPCに4を加算した値をr31に退避し，PCとワード長まで符号拡張したdisp26を加算した値をPCに設定し制御を移します。disp26のビット0はマスクされます。	69
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。	70
JR	disp26	IV	−	−	−	−	無条件分岐。現在のPCにワード長まで符号拡張したdisp26を加算し，その値に制御を移します。disp26のビット0は0にマスクされます。	71
LD.B	disp16[reg1], reg2	VI	−	−	−	−	バイト・ロード。reg1とワード長まで符号拡張したdisp16を加算して32ビット符号なし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。	72
LD.H	disp16[reg1], reg2	VI	−	−	−	−	ハーフワード・ロード。reg1とワード長まで符号拡張したdisp16を加算して32ビット符号なしアドレスを生成します。生成したアドレスからハーフワード・データを読み出し，ワード長まで符号拡張し，reg2に格納します。32ビット符号なしアドレスのビット0は0にマスクされます。	

命　令	オペランド	フォーマット	CY	OV	S	Z	命　令　機　能	ページ
LD.W	disp16[reg1], reg2	VI	－	－	－	－	ワード・ロード。reg1とワード長まで符号拡張したdisp16を加算して32ビット符号なしアドレスを生成します。生成したアドレスからワード・データを読み出し，reg2に格納します。32ビット符号なしアドレスのビット0, 1は0にマスクされます。	72
LDSR	reg2, regID	II	＊	＊	＊	＊	システム・レジスタへのロード。reg2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。	74
MAC3	reg1, reg2, reg3	VIII	－	－	－	－	符号付き32ビット飽和演算。reg1とreg2を符号付き整数として乗算し，その結果をreg3と加算します。 [オーバフローが発生しない場合] 結果をreg3に格納します。 [オーバフローが発生した場合] SATビットをセットし，結果が正ならば正の最大値を，負ならば負の最大値をreg3に格納します。	75
MACI	imm16, reg1, reg2	V	－	－	－	－	符号付き32ビット飽和演算。reg1と32ビット長に符号拡張したimm16を符号付き整数として乗算します。その結果とreg2を符号付き整数として加算します。 [オーバフローが発生しない場合] 結果をreg2に格納します。 [オーバフローが発生した場合] SATフラグをセットし，結果が正ならば正の最大値を，負ならば負の最大値をreg2にセットします。	76
MACT3	reg1, reg2, reg3	VIII	－	－	－	－	符号付き32ビット積和演算。reg1とreg2を符号付き整数として乗算し，結果の上位32ビットとreg3の32ビットを符号付き整数として加算します。 [オーバフローが発生しない場合] 結果をreg3に格納します。 [オーバフローが発生した場合] SATフラグをセットし，結果が正ならば正の最大値を，負ならば負の最大値をreg3に格納します。	77

命　令	オペランド	フォーマット	CY	OV	S	Z	命　令　機　能	ページ
MAX3	reg1, reg2, reg3	Ⅷ	－	－	－	－	最大値。reg2とreg1を符号付き整数として比較し，大きい方をreg3に格納します。	78
MIN3	reg1, reg2, reg3	Ⅷ	－	－	－	－	最小値。reg2とreg1を符号付き整数として比較し，小さい方をreg3に格納します。	79
MOV	reg1, reg2	Ⅰ	－	－	－	－	データの転送。reg1をreg2へコピーし，転送します。	80
	imm5, reg2	Ⅱ	－	－	－	－	データの転送。imm5をワード長まで符号拡張した値を，reg2にコピーし転送します。	
MOVEA	imm16, reg1, reg2	Ⅴ	－	－	－	－	加算。reg1に上位16ビット（imm16）をワード長まで符号拡張した値を加算し，その結果をreg2に格納します。	81
MOVHI	imm16, reg1, reg2	Ⅴ	－	－	－	－	加算。reg1に上位16ビット（imm16）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。	82
MUL	reg1, reg2	Ⅰ	－	＊	＊	＊	符号付き乗算。reg2にreg1を乗算（符号付き）し，その結果（ダブルワード長）の上位32ビットをr30に，下位32ビットをreg2に格納します。	83
MUL3	reg1, reg2, reg3	Ⅷ	－	－	－	－	符号付き32ビット乗算。reg1とreg2を符号付き整数として乗算し，結果の上位32ビットをreg3に格納します。	84
MULI	imm16, reg1, reg2	Ⅴ	－	－	－	－	符号付き32ビット飽和乗算。reg1と32ビット長に符号拡張したimm16を符号付き整数として乗算します。 [オーバフローが発生しない場合] 結果をreg2に格納します。 [オーバフローが発生した場合] SATフラグをセットし，結果が正ならば正の最大値を，負ならば負の最大値をreg2に格納します。	85
MULT3	reg1, reg2, reg3	Ⅷ	－	－	－	－	符号付き32ビット飽和乗算。reg1とreg2を符号付き整数として乗算し，結果の上位32ビットをreg3に格納します。	86
MULU	reg1, reg2	Ⅰ	－	＊	＊	＊	符号なし乗算。reg2にreg1を符号なしデータとして乗算し，その結果（ダブルワード長）の上位32ビットをr30に，下位32ビットをreg2に格納します。	87

命　令	オペランド	フォーマット	CY	OV	S	Z	命　令　機　能	ページ
NOP		III	−	−	−	−	ノー・オペレーション。	52
NOT	reg1, reg2	I	−	0	＊	＊	論理否定。reg1の論理否定（１の補数）をとり，その結果をreg2に格納します。	88
OR	reg1, reg2	I	−	0	＊	＊	論理和。reg2とreg1の論理和をとり，その結果をreg2に格納します。	89
ORI	imm16, reg1, reg2	V	−	0	＊	＊	論理和。reg1とimm16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reg2に格納します。	90
OUT.B	reg2, disp16[reg1]	VI	−	−	−	−	ポート出力。reg1のデータとワード長まで符号拡張したdisp16を加算し，32ビット符号なしポート・アドレスを生成します。reg2の下位１バイトのデータを生成したポート・アドレスに出力します。	91
OUT.H	reg2, disp16[reg1]	VI	−	−	−	−	ポート出力。reg1のデータとワード長まで符号拡張したdisp16を加算し，32ビット符号なしポート・アドレスを生成します。reg2の下位２バイトのデータを生成したポート・アドレスに出力します。32ビット符号なしポート・アドレスのビット０は０にマスクされます。	
OUT.W	reg2, disp16[reg1]	VI	−	−	−	−	ポート出力。reg1のデータとワード長まで符号拡張したdisp16を加算し，32ビット符号なしポート・アドレスを生成します。reg2のワード・データを生成したポート・アドレスに出力します。32ビット符号なしポート・アドレスのビット0, 1は０にマスクされます。	
RETI		IX	＊	＊	＊	＊	トラップまたは割り込みルーチンから戻ります。システム・レジスタから，復帰PCとPSWを取り出し，トラップまたは割り込みルーチンから復帰します。	93
SAR	reg1, reg2	I	＊	0	＊	＊	算術右シフト。reg2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（ＭＳＢの値を順にＭＳＢにコピー）し，reg2に書き込みます。	94
	imm5, reg2	II	＊	0	＊	＊	算術右シフト。reg2をimm5をワード長までゼロ拡張した値で示されるシフト数分算術シフトし，reg2に書き込みます。	

命　令	オペランド	フォーマット	CY	OV	S	Z	命　令　機　能	ページ
SATADD3	reg1, reg2, reg3	Ⅷ	＊	＊	＊	＊	飽和加算。reg2にreg1を符号付き整数として加算します。 [オーバフローが発生しない場合] 結果をreg3に格納します。 [オーバフローが発生した場合] SATフラグをセットし，結果が正ならば正の最大値を，負ならば負の最大値をreg3にセットします。	95
SATSUB3	reg1, reg2, reg3	Ⅷ	＊	＊	＊	＊	飽和減算。reg2からreg1を符号付き整数として減算します。 [オーバフローが発生しない場合] 結果をreg3に格納します。 [オーバフローが発生した場合] SATフラグをセットし，結果が正ならば正の最大値を，負ならば負の最大値をreg3にセットします。	96
SETF	imm5, reg2	Ⅱ	－	－	－	－	フラグ条件の設定。imm5の下位４ビットの示す条件が，条件フラグと一致した場合にはreg2に１を，そうでない場合は０を格納します。	97
SHL	reg1, reg2	Ⅰ	＊	0	＊	＊	論理左シフト。reg2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。	99
	imm5, reg2	Ⅱ	＊	0	＊	＊	論理左シフト。reg2をimm5をワード長までゼロ拡張した値で示されるシフト数分論理左シフトし，reg2に書き込みます。	
SHLD3	reg1, reg2, reg3	Ⅷ	－	－	－	－	連結左シフト。reg3（上位）とreg2（下位）を連結した64ビットを，reg1の下位５ビットで示されるシフト数分だけ論理左シフトし，上位32ビットをreg3に格納します。	100
SHR	reg1, reg2	Ⅰ	＊	0	＊	＊	論理右シフト。reg2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。	101
	imm5, reg2	Ⅱ	＊	0	＊	＊	論理右シフト。reg2をimm5をワード長までゼロ拡張した値で示されるシフト数分論理右シフトし，reg2に書き込みます。	

命　令	オペランド	フォーマット	CY	OV	S	Z	命　令　機　能	ページ
SHRD3	reg1, reg2, reg3	Ⅷ	－	－	－	－	連結右シフト。reg3（上位）とreg2（下位）を連結した64ビットをreg1の下位５ビットで示されるシフト数分だけ論理右シフトし，下位32ビットをreg3に格納します。	102
ST.B	reg2, disp16[reg1]	Ⅵ	－	－	－	－	バイト・ストア。reg1のデータと，ワード長まで符号拡張したdisp16を加算し，32ビット符号なしアドレスを生成します。reg2の下位１バイトのデータを生成したアドレスに格納します。	103
ST.H	reg2, disp16[reg1]	Ⅵ	－	－	－	－	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disp16を加算し32ビット符号なしアドレスを生成します。reg2の下位２バイトのデータを生成したアドレスに格納します。32ビット符号なしアドレスのビット０は０にマスクされます。	
ST.W	reg2, disp16[reg1]	Ⅵ	－	－	－	－	ワード・ストア。reg1のデータとワード長まで符号拡張したdisp16を加算し，32ビット符号なしアドレスを生成します。reg2のワード・データを生成したアドレスに格納します。32ビット符号なしアドレスのビット0, 1は０にマスクされます。	
STBY		Ⅸ	－	－	－	－	プロセッサ停止。ストップ・モードに入ります。	105
STSR	regID, reg2	Ⅱ	－	－	－	－	システム・レジスタの内容のストア。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reg2に設定します。	106
SUB	reg1, reg2	Ⅰ	＊	＊	＊	＊	減算。reg2からreg1を減算し，その結果をreg2に格納します。	107

命　令	オペランド	フォーマット	CY	OV	S	Z	命　令　機　能	ページ
TRAP	vector	II	－	－	－	－	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， PSW.EP= 1 → FEPC, FEPSWに退避 PSW.EP = 0 → EIPC, EIPSWに退避 例外コードをECRに設定し， PSW.EP = 1 → FECCに設定 PSW.EP = 0 → EICCに設定 PSWのフラグを設定し， PSW.EP = 1 → NP, IDをセット PSW.EP = 0 → EP, IDをセット 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。	108
XOR	reg1, reg2	I	－	0	＊	＊	排他的論理和。reg2とreg1の排他的論理和をとり，その結果をreg2に格納します。	110
XORI	imm16, reg1, reg2	V	－	0	＊	＊	排他的論理和。reg1とimm16をワード長までゼロ拡張した値の排他的論理和をとりその結果をreg2に格納します。	111

付録B　オペコード・マップ

命令コード・マップ

ビット 15-10	命　令　形　式		フォーマット	サブオペコード
000000	MOV	reg1, reg2	I	
000001	ADD	reg1, reg2	I	
000010	SUB	reg1, reg2	I	
000011	CMP	reg1, reg2	I	
000100	SHL	reg1, reg2	I	
000101	SHR	reg1, reg2	I	
000110	JMP	[reg1]	I	
000111	SAR	reg1, reg2	I	
001000	MUL	reg1, reg2	I	
001001	DIV	reg1, reg2	I	
001010	MULU	reg1, reg2	I	
001011	DIVU	reg1, reg2	I	
001100	OR	reg1, reg2	I	
001101	AND	reg1, reg2	I	
001110	XOR	reg1, reg2	I	
001111	NOT	reg1, reg2	I	
010000	MOV	imm5, reg2	II	
010001	ADD	imm5, reg2	II	
010010	SETF	imm5, reg2	II	
010011	CMP	imm5, reg2	II	
010100	SHL	imm5, reg2	II	
010101	SHR	imm5, reg2	II	
010110	EI		II	
010111	SAR	imm5, reg2	II	
011000	TRAP	vector	II	
011001	RETI		IX	0
011001	BRKRET		IX	1
011010	HALT		IX	0
011010	STBY		IX	1
011100	LDSR	reg2, regID	II	
011101	STSR	regID, reg2	II	
011110	DI		II	

ビット 15-10	命　令　形　式		フォーマット	サブオペコード
100XXX	Bcond		III	0
100XXX	ABcond		III	1
101000	MOVEA	imm16, reg1, reg2	V	
101001	ADDI	imm16, reg1, reg2	V	
101010	JR	disp26	IV	
101011	JAL	disp26	IV	
101100	ORI	imm16, reg1, reg2	V	
101101	ANDI	imm16, reg1, reg2	V	
101110	XORI	imm16, reg1, reg2	V	
101111	MOVHI	imm16, reg1, reg2	V	
110000	LD.B	disp16[reg1], reg2	VI	
110001	LD.H	disp16[reg1], reg2	VI	
110010	MULI	imm16, reg1, reg2	V	
110011	LD.W	disp16[reg1], reg2	VI	
110100	ST.B	reg2, disp16[reg1]	VI	
110101	ST.H	reg2, disp16[reg1]	VI	
110110	MACI	imm16, reg1, reg2	V	
110111	ST.W	reg2, disp16[reg1]	VI	
111000	IN.B	disp16[reg1], reg2	VI	
111001	IN.H	disp16[reg1], reg2	VI	
111010	CAXI	disp16[reg1], reg2	VI	
111011	IN.W	disp16[reg1], reg2	VI	
111100	OUT.B	reg2, disp16[reg1]	VI	
111101	OUT.H	reg2, disp16[reg1]	VI	
111110	Special		VII／VIII	
111111	OUT.W	reg2, disp16[reg1]	VI	

オペコード部

ビット 12-10 / ビット 15-13	000	001	010	011	100	101	110	111
000	MOV	ADD	SUB	CMP	SHL	SHR	JMP	SAR
001	MUL	DIV	MULU	DIVU	OR	AND	XOR	NOT
010	MOV	ADD	SETF	CMP	SHL	SHR	EI	SAR
011	TRAP	RETI BRKRET	HALT STBY		LDSR	STSR	DI	
100	Bcond/ABcond							
101	MOVEA	ADDI	JR	JAL	ORI	ANDI	XORI	MOVHI
110	LD.B	LD.H	MULI	LD.W	ST.B	ST.H	MACI	ST.W
111	IN.B	IN.H	CAXI	IN.W	OUT.B	OUT.H	スペシャル	OUT.W

条件分岐（Bcond/ABcond）条件コード部

ビット 11-9 / ビット 12	000	001	010	011	100	101	110	111
0	BV	BC/BL	BZ/BE	BNH	BN	BR	BLT	BLE
1	BNV	BNC/BNL	BNZ/BNE	BH	BP	NOP	BGE	BGT

スペシャル命令コード部

ビット 28-26 / ビット 31-29	000	001	010	011	100	101	110	111
000								
001								
010	SATADD3	SATSUB3	MIN3	MAX3				
011	SHLD3	SHRD3			MACT3	MAC3	MULT3	MUL3
100	BILD	BDLD	BIST	BDST				
101								
110								
111								

〔メ　モ〕

付録C　総合索引

C.1　50音で始まる語句の索引

C.2　アルファベットで始まる語句の索引

〔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（インフォメーションセンター）
（電話：午前 9:00～12:00，午後 1:00～5:00）

電　話　：044−548−8899
FAX　：044−548−7900
E-mail ：s-info@saed.tmg.nec.co.jp

【営業関係お問い合わせ先】

半導体第一販売事業部 半導体第二販売事業部 半導体第三販売事業部	〒108-8001	東京都港区芝５－７－１　（日本電気本社ビル）	(03)3454−1111
中部支社	〒460-8525	愛知県名古屋市中区錦１－17－１　（日本電気中部ビル）	(052)222−2170 (052)222−2190
関西支社	〒540-8551	大阪府大阪市中央区城見１－４－24　（日本電気関西ビル）	(06)6945−3178 (06)6945−3200 (06)6945−3208

北海道支社	札　幌	(011)251−5599	宇都宮支店	宇都宮	(028)621−2281	北陸支社	金　沢	(076)232−7303
東北支社	仙　台	(022)267−8740	小山支店	小　山	(0285)24−5011	京都支社	京　都	(075)344−7824
岩手支店	盛　岡	(019)651−4344	甲府支店	甲　府	(055)224−4141	神戸支社	神　戸	(078)333−3854
郡山支店	郡　山	(024)923−5511	長野支社	松　本	(0263)35−1662	中国支社	広　島	(082)242−5504
いわき支店	いわき	(0246)21−5511	静岡支社	静　岡	(054)254−4794	鳥取支店	鳥　取	(0857)27−5311
長岡支店	長　岡	(0258)36−2155	立川支社	立　川	(042)526−5981,6167	岡山支店	岡　山	(086)225−4455
水戸支店	水　戸	(029)226−1717	埼玉支社	大　宮	(048)649−1415	松山支店	松　山	(089)945−4149
土浦支店	土　浦	(0298)23−6161	千葉支社	千　葉	(043)238−8116	九州支社	福　岡	(092)261−2806
群馬支店	高　崎	(027)326−1255	神奈川支社	横　浜	(045)682−4524			
太田支店	太　田	(0276)46−4011	三重支店	津	(059)225−7341			

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］V830ファミリ　ユーザーズ・マニュアル　アーキテクチャ編
（U12496JJ4V0UM00（第４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，NEC半導体ソリューション技術本部員，
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。

NEC半導体テクニカルホットライン
FAX：（044）548-7900

キリトリ