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。

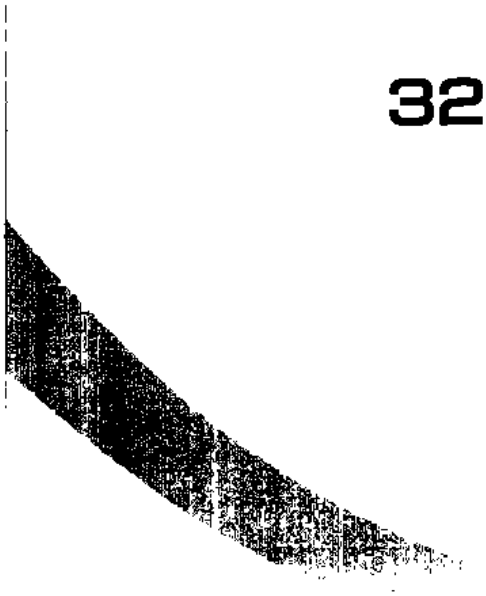
ユーザーズ・マニュアル

RENESAS

保守／廃止

VR3800™

32ビット・マイクロプロセッサ



μPD30380

資料番号 IEU-838B（第3版）
発行年月 October 1994 P

保守／廃止

保守／廃止

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，全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，ノイス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，レシ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。

保守／廃止

V_R3800，V_R3000A，V_Rシリーズは日本電気株式会社の商標です。

本製品は外国為替および外国貿易管理法の規定により戦略物資等（または役務）に該当しますので，日本国外に輸出する場合には，同法に基づき日本国政府の輸出許可が必要です。

本製品は米国の輸出管理規則の規制を受ける技術を用いておりますので，本製品および本製品を組み込んだ装置を輸出する場合，輸出先によっては米国政府の許可も必要です。

M7 94.6

保守／廃止

本版で改訂された主な箇所

箇　所	内　　容
全　般	16 MHz品と20 MHz品を削除
p. 22	**3.4　(6)バス・アービタ**の説明を修正
p. 30	**4.2　シングル・リード・サイクル**に説明を追加
p. 34	**4.4　ブロック・リード・サイクル**に説明を追加
p. 47	**4.6.1　8ビット・ダイナミック・バス・サイジング**に説明を追加
p. 49	**4.6.2　16ビット・ダイナミック・バス・サイジング**に説明を追加
p. 51	**4.7　16ビット固定バス・サイジング**に説明を追加
p. 58	**表4－2　データ・バス上のデータの整列位置とアドレスの関係**に内容を追加
p. 62	**表4－4　16ビット・バス・サイジング時のデータ・バス上の整列位置**に内容を追加
p. 98	**8.2　遅延スロット**に説明を追加
p. 109	**表10－2　ExcCode**フィールドを修正
p. 158, 159, 160, 161, 163, 164, 166, 168	**11.5　命　　令**の次の各命令に説明を追加 LB, LBU, LH, LHU, LW, LWL, LWR, MFC0

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。

保守／廃止

はじめに

対象者　このマニュアルは，μPD30380（別名称V_R3800^{TM}）の機能を理解し，それを用いたアプリケーション・システムを設計しようとするユーザを対象とします。

目　的　このマニュアルは，次の構成に示すV_H3800の持つ各種ハードウエア機能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，大きく分けて次の内容で構成しています。

- 概要
- 端子機能
- ハードウエア・アーキテクチャ
- ソフトウエア・アーキテクチャ
- 例外処理
- 命令セット

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

このマニュアルでは，本文中の製品名を次のように読み替えています。

μPD30310：V_R3000A^{TM}

一通りV_R3800の機能を理解しようとするとき

→目次に従って読んでください。

V_R3800の命令機能の詳細を知りたいとき

→**第11章　命令セット**を読んでください。

電気的特性を知りたいとき

→データ・シートを参照してください。

凡　例

データ表記の重み	：左側が上位桁，右側が下位桁
アクティブ・ロウの表記	×××（端子，信号名称の上に線）
メモリ・マップのアドレス	：上部　上位，下部　下位
注	：本文中に付けた注の説明
注意	：特に気を付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…××××Bまたは××××

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．

V_R3800　データ・シート（IC-8824）

ユーザーズ・マニュアル（このマニュアル）

アプリケーション・ノート

モニタ・プログラムとハードウエア設計編（IEA-753）

V_R3000A　データ・シート（IC-8278）

ユーザーズ・マニュアル　ハードウエア編（IEU-748）

アーキテクチャ編（IEU-730）

保守／廃止

目　　次

図の目次（1/2）

保守／廃止

図の目次（2/2）

表の目次

保守／廃止

第1章　概　　説

1

1.1　特　　徴

V_R3800は，32ビットRISC型マイクロプロセッサV_R3000AをCPUコアとし，周辺として命令／データ・キャッシュ，クロック・ジェネレータ，リセット回路，バス・インタフェース回路を1チップに集積しています．このため，特に組み込み用途のシステムを容易に構築できます．

V_R3800の特徴を次に示します．

●V_R3000Aとオブジェクト互換（浮動小数点演算用コプロセッサと仮想記憶管理機構はサポートしません）

●周辺機能を内蔵

・キャッシュ・メモリ（命令　4Kバイト，データ：1Kバイト）

・PLL（Phase Locked Loop）付きクロック・ジェネレータ（4相クロック生成）

・バス・インタフェース（4段ライト・バッファ，32ビット・アドレス／データ・セパレート・バス）

●システム構築が容易なアーキテクチャ

・単相クロック入力，1倍周波数入力（25 MHz）　★

・リセット時の動作モード設定の簡略化

・Ready制御による汎用バス・サイクル

・8ビット／16ビット・ブートROM対応（ダイナミック・バス・サイジング）

・16ビット固定バス・モード可能

次に，それぞれの特徴について説明します．

(1) CPUコアとしてV_R3000Aを採用

V_R3000Aは，米国MIPS Computer Systems社開発のRISC（Reduced Instruction Set Computer）アーキテクチャを採用した，高性能32ビット・マイクロプロセッサです．

5段の命令パイプラインにより，ほとんどの命令を1CPUサイクルで実行するため，非常に高い性能を実現します．

(2) V_R3000Aとソフトウエア・コンパチブル

V_R3800は，V_R3000Aとソフトウエア・コンパチブルです．このため，V_R3000Aの開発環境をそのまま利用して，プログラムを開発できます　また，V_R3000A用に開発したプログラムも，簡単にV_R3800に移植できます

ただしV$_R$3800は，FPU（Floating-point Processing Unit）を含む外部コプロセッサ（CP1-CP3）をサポートしていません。FPUの使用を前提とするソフトウエアを実行するためには，浮動小数点演算をエミュレートするソフトウエアを別に用意してください．誤動作を避けるため，すべての外部コプロセッサは，システム制御コプロセッサCP0のステータス・レジスタ（Cu |3 1|）を0に設定して，使用不可にしてください

また，仮想記憶管理機構（TLB）もサポートしていません。

(3) キャッシュ・メモリ内蔵

命令キャッシュ4Kバイト，データ・キャッシュ1Kバイトの容量のキャッシュ・メモリを内蔵しています　キャッシュの構成はダイレクト・マップです

命令キャッシュとデータ・キャッシュの内容は，リセットで無効化できます。このため，従来のV$_R$3000Aを使用したシステムで必要だった，ソフトウエアによるリセット直後のキャッシュの無効化という作業は不要です．

また，外部端子（ICCLR，DCCLR）でもキャッシュの内容を無効化できます。

詳細は，**3.2　キャッシュ**を参照してください。

(4) クロック・ジェネレータ内蔵

V$_R$3000Aには，4相の動作クロックを入力します．これに対してV$_R$3800には外付けのクロック回路を簡略化するため，クロック・ジェネレータを内蔵しています　このため，単相でCPU動作周波数のクロックを入力します

詳細は，**3.3　クロック・ジェネレータ**を参照してください

(5) バス・インタフェース回路内蔵

V$_R$3(XX)Aにはメイン・メモリ・システムにアクセスするバスがないため，リード・アクセス，ライト・アクセス時のアドレスとデータをラッチし，メイン・メモリとのインタフェースを行うμPD31311などのデバイスを外付けしていました　V$_R$3800では，4段のライト・バッファと1段のリート・バッファからなるバス・インタフェース・ユニット（BIU）を内蔵し，CPUとメイン・メモリの効率的なインタフェースを実現します．

V$_R$3800は，リード優先のバス制御を行うため，通常の動作ではライト動作の待ち合わせに影響されず，パイプラインの停止時間を最小限にしています。

同一アドレスに対するリードとライトか重なった場合（衝突）には，ライト・データをメイン・メモリに書き込んでからリード動作を行うため，データ転送の順番がずれません。

詳細は，**3.4　バス・インタフェース・ユニット（BIU）**を参照してください．

(6) リセット・シーケンスの簡略化

V$_R$3000Aは，リセット期間中に動作モートを選択します，つまり，リセット信号を解除（ディアサート）する直前の4サイクルがW，X，Y，Zサイクルとして定義され．それぞれのサイクルでのInt

保守／廃止

[5：0] 端子の値でプロセッサ機能を決定するようになっています．このため，外付けのリセット回路が必要でした．

V_R3800では，このリセット・シーケンスを簡略化します．モード選択のうち，いくつかのプロセッサ機能は固定していますが，残りのプロセッサ機能はDBLOCK，IBLOCK，ISTREAM，STPART，BIGEND端子の設定でモードを選択するようになっています．これにより，リセット端子をアクティブにしたあと，必要な期間だけ保持し，インアクティブにするだけの操作でモードを選択できますリセット期間中にINT 5：0 端子を操作する必要はありません．

詳細は，**3.5 リセット**を参照してください

(7) DMAサポート機能内蔵

DMAなどの外部からのバス使用権の要求に応し，自身の出力端子をハイ・インピータンス状態にし，システム・バスから切り離す機能があります．さらに，V_R3800かバスの使用権を明け渡したことを示す端子（HLDAK）とデータ・キャッシュ無効化用の端子（DCCLR）も備えています。

また，キャッシュ領域へのDMA転送も可能です（**3.2 キャッシュ**参照）．

(8) 8ビット 16ビットROMサポート機能内蔵

V_R3000Aのデータ・バスは通常32ビットですが，非キャッシュ領域（キャッシュ領域も条件によっては可能）における命令フェッチとデータのリートに関して，データ・バスをサイクルごとに8ビット幅，または16ビット幅に指定するダイナミック・バス・サイジングを行うことかできます この機能を利用すれば，フログラムをメイン・メモリにダウンロードしてから実行するようなシステムにおいて，リセット直後のブート時に使用するROMの数を削減てきます

ダイナミック・バス・サイジングを使用するかは，バス・サイクルことにSIZRQ端子で設定しますまた，ダイナミック・バス・サイジングを8ビット幅にするか，16ビット幅にするかは，リセット時にDBUSSIZ端子で設定します．

詳細は，**4.6 ダイナミック・バス・サイジング**を参照してください．

(9) 16ビット固定バス・サイジング機能内蔵

常に16ビット・データ・バスで動作するモートです データ・バスの幅が半分になるため配線占有率を抑え，システム・コストを低減てきます ダイナミック・バス・サイジングは非キャッシュ領域のリード・サイクルだけを想定していますか，16ビット固定バス・サイジング・モードては，キャッシュ領域のリート・サイクルやライト・サイクルなど，すべてのサイクルを16ビット幅のバス・モートて動作します ただし，16ビット固定バス・モートでは，CPU内部のハートウエアを簡略化するため，キャッシュ・リフィルてのブロック・リートをサホートしていません．

16ビット固定バス・サイジング・モードはリセット時にSBUSSIZ端子で指定します

詳細は，**4.7 16ビット固定バス・サイジング**を参照してください

保守／廃止

★ 1.2 オーダ情報

オーダ名称	パッケージ	品質水準
μPD30380-GD-25-MBD	160ピン・プラスチックQFP（□128 mm）	標準（一般電子機器用）

品質水準とその応用分野の詳細については当社発行の資料「NEC 半導体デバイスの品質水準」（IEI-620）をご覧ください。

1.3 内部ブロック図

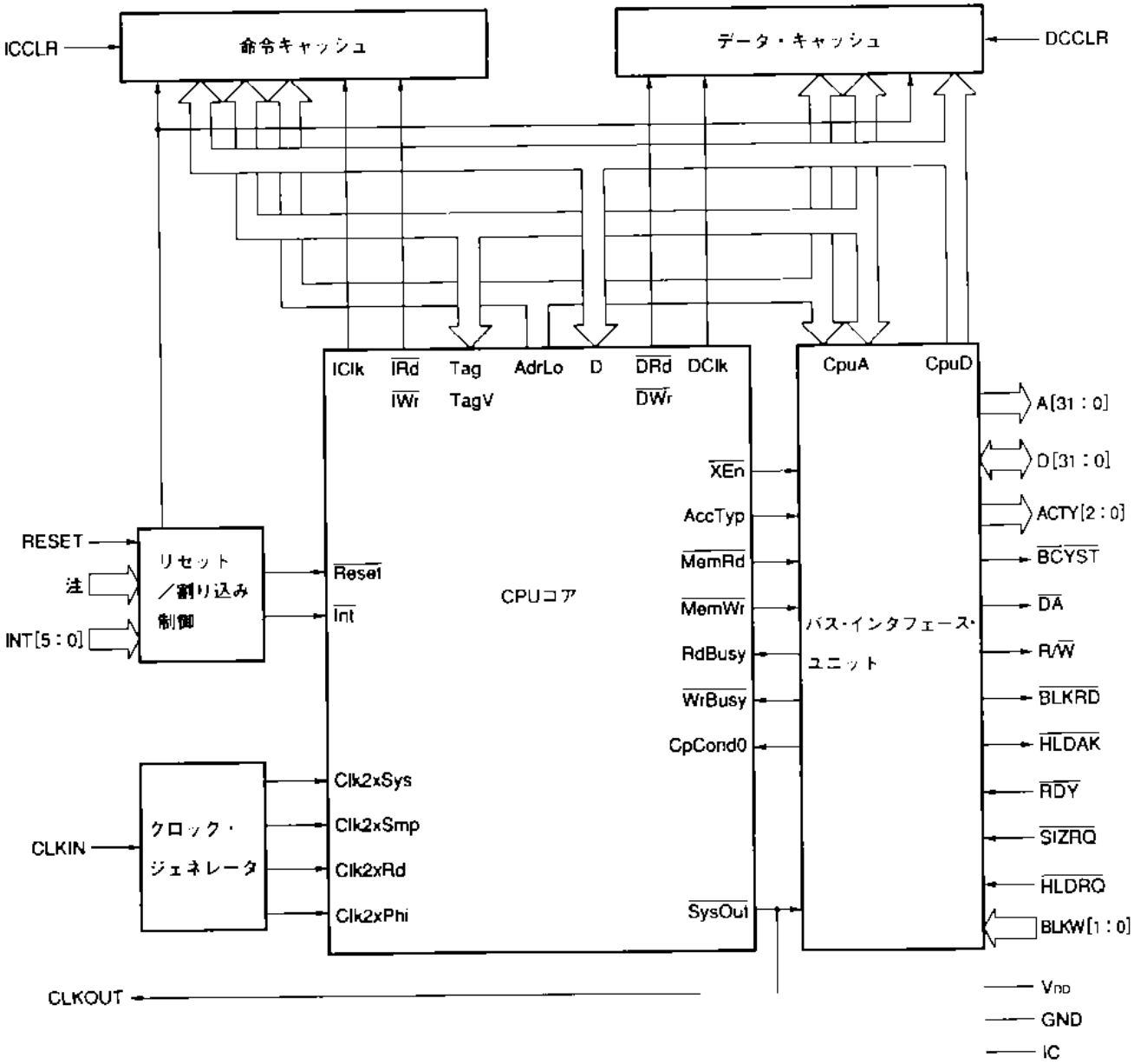
V_R3800は，大きく分けて次の６つのブロックから構成されています

- ●CPUコア（V_R3000A）
- ●クロック・ジェネレータ
- ●命令キャッシュ
- ●データ・キャッシュ
- ●リセット回路
- ●BIU

図１－１に内部ブロック図を示します。

保守／廃止

図1－1 内部ブロック図



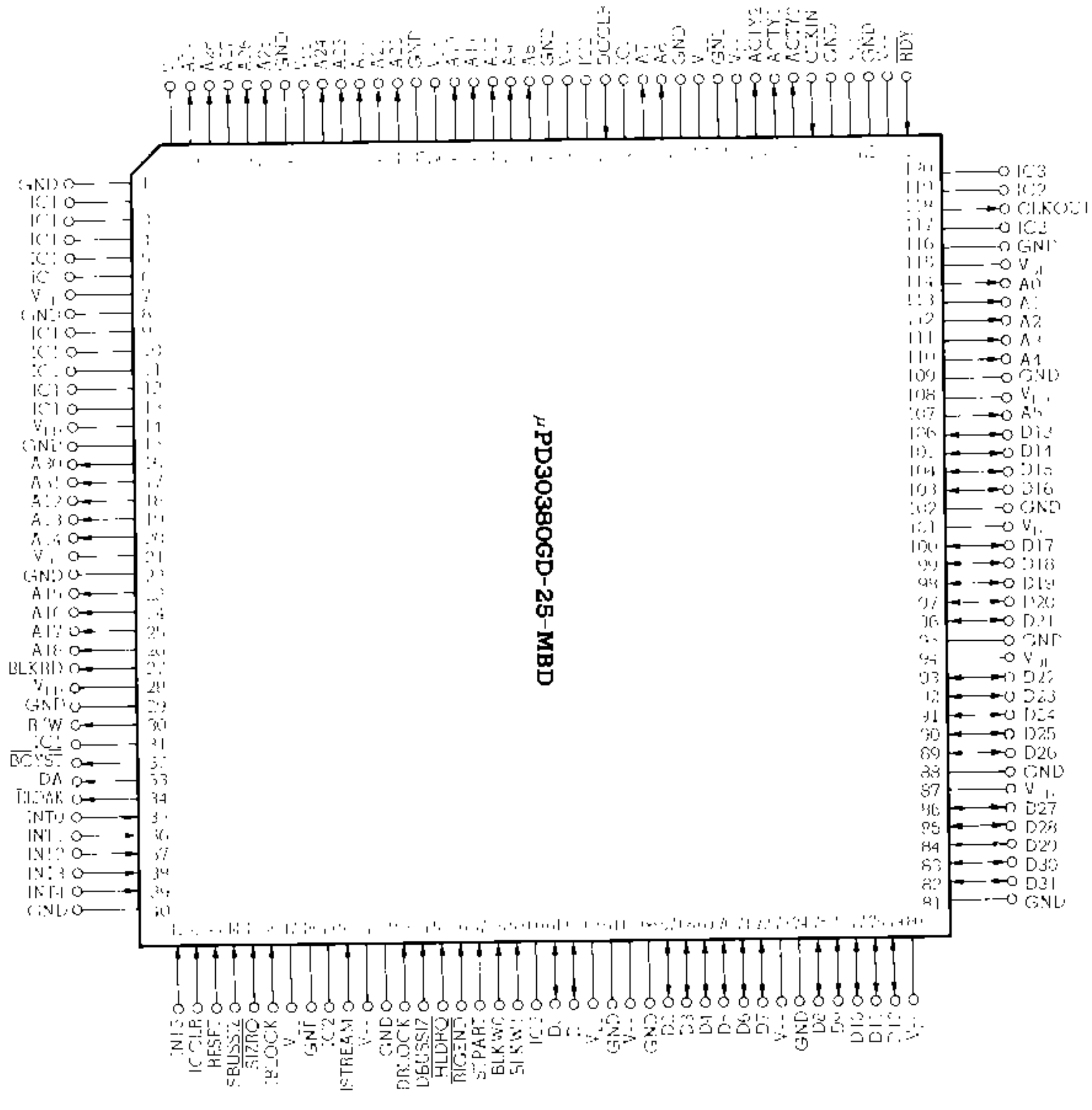
注 SBUSSIZ, DBUSSIZ, DBLOCK, IBLOCK, STPART, ISTREAM, BIGEND

保守／廃止

〔メ　モ〕

第２章　端子機能

2.1　端子接続図（Top View）



注意１．IC1端子には何も接続しないでください。

２．IC2端子はGNDに直接接続してください。

３．IC3端子はV_{DD}に直接接続してください。

端子名称

A [31 : 0]	: Adress Bus
ACTY [2 : 0]	: Access Type
BCYST	: Bus Cycle Start
BIGEND	: Big Endian
BLKRD	: Block Read
BLKW [1 : 0]	: Block Wait
CLKIN	: Clock In
CLKOUT	: Clock Out
D [31 : 0]	: Data Bus
DA	: Data Access
DBLOCK	: Data Cache Block Request
DBUSSIZ	: Dynamic Bus Sizing
DCCLR	: Data Cache Clear
HLDRQ	: Hold Request
HLDAK	: Hold Acknowledge
IBLOCK	: Instruction Cache Block Request
ICCLR	: Instruction Cache Clear
INT [5 : 0]	: Interrupt Request
ISTREAM	Instruction Streaming
R/$\overline{W}$	: Read/Write
RDY	: Ready
RESET	: Reset
SBUSSIZ	: Static Bus Sizing
SIZRQ	: Bus Sizing Request
STPART	: Store Partial
V_{DD}	: Power Supply
GND	: Ground
IC	: Internally Connected

保守／廃止

2.2　機能説明

次に端子機能について示します

表２－１　端子機能（1/5）

端子名	入出力	機　能	バス・ホールド時の状態	リセット時の状態
CLKIN	入力	クロック入力 単相のクロックを入力します。		
CLKOUT	出力	クロック出力 システム・クロックとして使用します。	通常動作	通常動作
A［31：0］	出力	アドレス・バス	Hi-Z	Hi-Z
D［31：0］	入出力	データ・バス	Hi-Z	Hi-Z
ACTY［2：0］	出力	アクセス・タイプ リード・アクセス時，アクセス領域の種類を示します。	Hi-Z	Hi-Z

ACTY［2：0］	種　類
0　×　×	非キャッシュ領域注
1　×　0	データ・キャッシュ領域のアクセス
1　×　1	命令キャッシュ領域のアクセス

注　ACTY［1，0］はCPUコアが要求しているデータ長を示していますが，アドレス出力はワード・アラインさせる（バス・サイジング時にはサイジングされたバス幅の単位でアライン）ため，アドレスとACTY［1：0］の関係が正しくない場合があります。
リード時にはデータ・バス幅に対応したすべてのバイトをアクセスしてください。必要なバイトを認識させる必要はありません。ただし，I/Oなどで特定のバイトしかアクセスがない場合には，そのバイトのアクセスだけでかまいません。

保守／廃止

表2－1　端子機能（2/5）

端子名	入出力	機　能	バス・ホールト時の状態	リセット時の状態
ACTY [2:0]	出力	ライト・アクセス時　データ長を示します。 ACTY [2:0] ／ データ長 × 0 0 ／ バイト × 0 1 ／ ハーフ・ワード × 1 0 ／ 3バイト × 1 1 ／ ワード **備考**　バス・サイジング時には，上記以外の動作を行うことがあります。詳細は，**4.7　16ビット固定バス・サイジング**を参照してください。	Hi-Z	Hi-Z
DA	出力	データ・アクセス バス・サイクルにおけるデータのストローブ信号です。	Hi-Z	Hi-Z
R/W	出力	リード／ライト アクセスがリード・サイクルかライト・サイクルかを示します。	Hi-Z	Hi-Z
BLKRD	出力	ブロック・リード キャッシュのリフィル時に，ブロック・リードを行うことを示します。	Hi-Z	Hi-Z
BCYST	出力	バス・サイクル・スタート バス・サイクルの開始を示します。	Hi-Z	Hi-Z
$\overline{RDY}$	入力	レディ バス・サイクルの完結を指示します。		
HLDRQ	入力	ホールド・リクエスト CPUに対し，バスの使用権を放棄することを要求します。		
HLDAK	出力	ホールド・アクノリッジ CPUがバスの使用権を放棄したことを示します。	アクノリッジ	H
INT [5:0]	入力	マスカブル割り込み要求		

保守／廃止

表2-1　端子機能（3/5）

端子名	入出力	機　能	バス・ホールド時の状態	リセット時の状態
BLKW [1:0]	入力	ブロック・ウエイト ブロック・リード（キャッシュ・リフィル）時の単位バス・サイクル数を指定します。		

BLKW [1:0]		サンプリングの間隔
0	0	1クロック
0	1	2クロック
1	0	3クロック
1	1	4クロック

RESET	入力	リセット要求		
IBLOCK	入力	命令キャッシュ・ブロック要求 命令キャッシュのリフィルをブロック・リードで行うことを指示します。 リセット時に1回だけサンプリングします。 1：ブロック・リードでリフィル 0：シングル・リードでリフィル		
DBLOCK	入力	データ・キャッシュ・ブロック要求 データ・キャッシュのリフィルをブロック・リードで行うことを指示します。 リセット時に1回だけサンプリングします。 1：ブロック・リードでリフィル 0：シングル・リードでリフィル		
SIZRQ	入力	バス・サイジング・リクエスト リード・バス・サイクルを8ビット幅または16ビット幅で行うことを指示します。 バス・サイクルごとにサンプリングします。 サイジング時のバス幅はリセット時にサンプリングするDBUSSIZ端子で決定します。 ブロック・リード・サイクルでサイジングを指示した場合の動作は不定です。		

表２－１　端子機能（4/5）

端子名	入出力	機　能	バス・ホールド時の状態	リセット時の状態
BIGEND	入力	ビッグ・エンディアン メモリ・アクセスをリトル・エンディアンのバイト順序で行うことを指示します． リセット時に１回だけサンプリングします． １：リトル・エンディアン ０：ビッグ・エンディアン		
ISTREAM	入力	命令ストリーム 命令のストリーミング動作を許可します． リセット時に１回だけサンプリングします． １：ストリーミングを行う ０：ストリーミングを行わない		
STPART	入力	ストア・パーシャル 部分ワード・ストア時にデータ・キャッシュに対してリード・モディファイ・ライトを行うように指示します． リセット時に１回だけサンプリングします． １：リード・モディファイ・ライトを行う ０：リード・モディファイ・ライトを行わない		
SBUSSIZ	入力	固定バス・サイジング 常に16ビット・バス・モードで動作させます． リセット時に１回だけサンプリングします． １　16ビット・バス・モード ０　32ビット・バス・モード		
DBUSSIZ	入力	ダイナミック・バス・サイジング ダイナミック・バス・サイジングのバス幅を指定します． リセット時に１回だけサンプリングします． １：16ビット幅 ０：８ビット幅		
ICCLR	入力	命令キャッシュ・クリア 命令キャッシュを無効化します．		
DCCLR	入力	データ・キャッシュ・クリア データ・キャッシュを無効化します．		
V_{DD}		正電源		
GND		グランド電位		
ICU		内部接続端子 何も接続しないでください		

保守／廃止

表2－1　端子機能（5/5）

端子名	入出力	機　能	バス・ホールド時の状態	リセット時の状態
IC2		内部接続端子 GNDに直接接続してください。		
IC3		内部接続端子 V_{DD}に直接接続してください。		

保守／廃止

〔メ　モ〕

保守／廃止

第３章　内部機能

3.1　ハードウエア構成

V_R3800は，CPUコアにV_R3000A，外部にキャッシュ，クロック・ジェネレータ，バス・インタフェース・ユニット（BIU）を付加して，１チップにまとめた32ビットRISC型マイクロプロセッサです。

次にV_μ3800の基本仕様を示します。

表３-１　基本仕様

項目		仕様
品名		μPD30380（V_R3800）
CPUコア		V_R3000A
FPU		なし
TLB		なし
命令キャッシュ	構成	ダイレクト・マップ
	容量	４Kバイト
	リフィル・サイズ	４ワード／１ワード
	ライン・サイズ	１ワード
データ・キャッシュ	構成	ダイレクト・マップ
	容量	１Kバイト
	リフィル・サイズ	４ワード／１ワード
	ライン・サイズ	１ワード
	書き込み制御	ライト・スルー
リード・バッファ		１段（ブロック転送対応）
ライト・バッファ		４段
アドレス／データ・バス		分離バス，32ビット／32ビット
バス・サイジング		８ビット・ダイナミック
		16ビット・ダイナミック
		16ビット・スタティック
クロック入力		単相（25 MHz）

★

保守／廃止

3.2　キャッシュ

V_R3800のキャッシュ・メモリは，ライン・サイズが１ワード（４バイト）のダイレクト・マップ方式（１ウェイ・セット・アソシアティブ）のキャッシュです．

命令キャッシュの容量は４Kバイト（1024エントリ），データ・キャッシュの容量は１Kバイト（256エントリ）です．タグ部のビット数は命令キャッシュが20ビット，データ・キャッシュが22ビットです．

キャッシュにパリティはありません．V_R3000Aにあったデータ／タグ・パリティのチェックは，リセット時の設定で禁止しています．

V_R3000Aのアーキテクチャでは，４Kバイトより小さい容量のキャッシュを接続することを想定していません（タグは最大20ビット）．このため，データ・キャッシュにはタグの残り２ビットを比較するためのコンパレータを備えています．

また，ソフトウエアの負担を軽減するため，リセットで命令キャッシュとデータ・キャッシュを無効化します．また，外部端子（ICCLR，DCCLR）でも無効化できます

CLKOUTの立ち下がりでサンプリングしたICCLR（DCCLR）信号がアクティブならすぐに命令（データ）キャッシュのすべてのバリッド・ビットをクリアし，命令（データ）キャッシュを無効化します．ICCLR（DCCLR）信号がアクティブの間は，命令（データ）キャッシュのバリッド・ビットをクリアします．

通常のキャッシュ・メモリを使用するマイクロプロセッサでは，キャッシュ・メモリの内容とメイン・メモリの内容を一致させるため，キャッシュ領域へのDMA転送はできません．しかしV_R3800では，DMA転送終了後にDCCLR信号をアサートすることでデータ・キャッシュの内容を無効化するため，キャッシュ領域へのDMA転送も可能です．

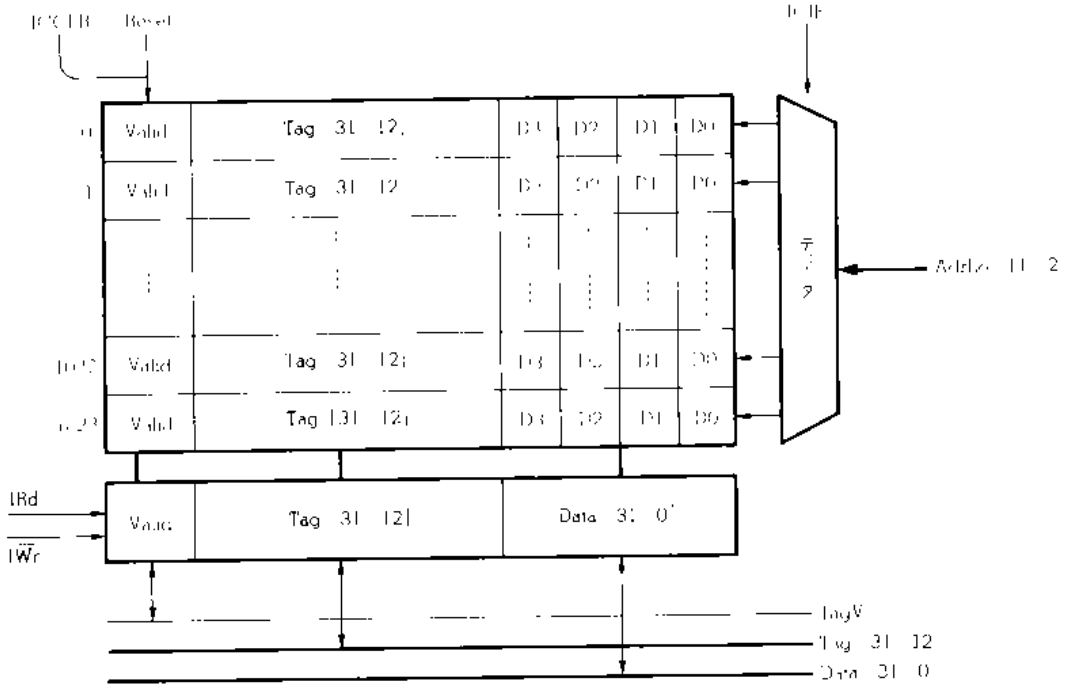
図３－１にキャッシュの構成，図３－２にICCLR信号のタイミング，図３－３にDCCLR信号のタイミングを示します

保守／廃止

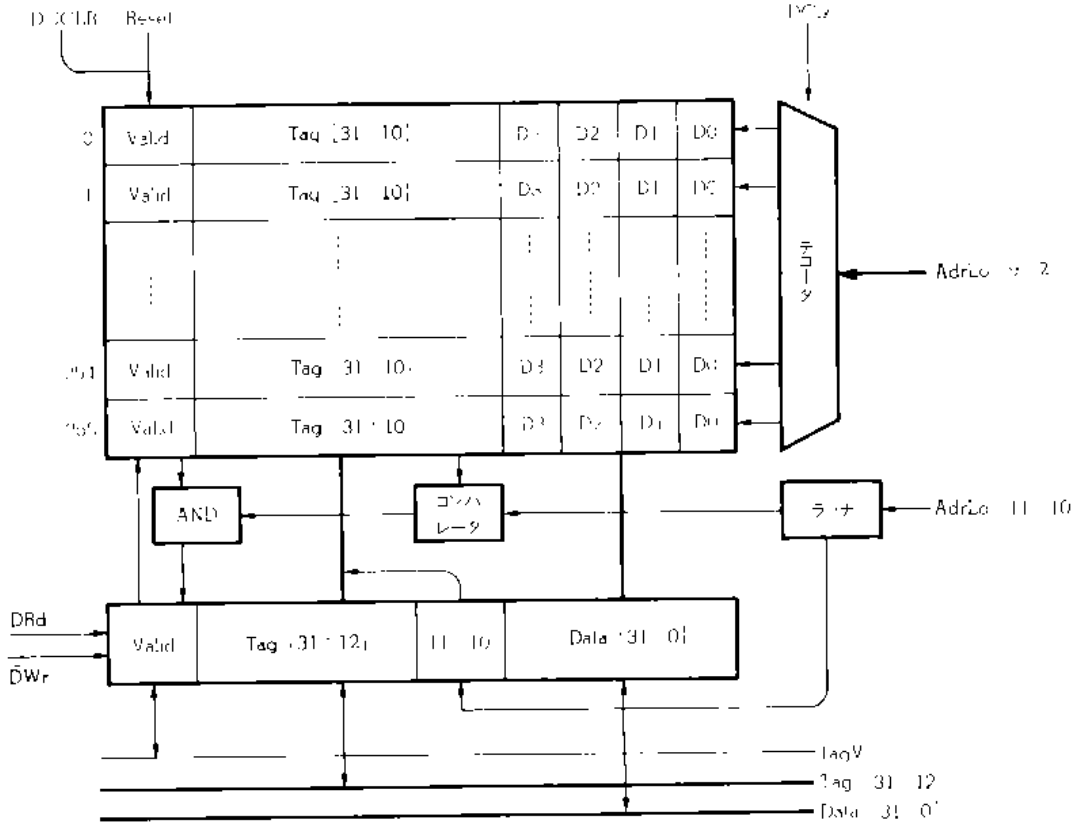
3

図3　1　キャッシュの構成

(a) 命令キャッシュ

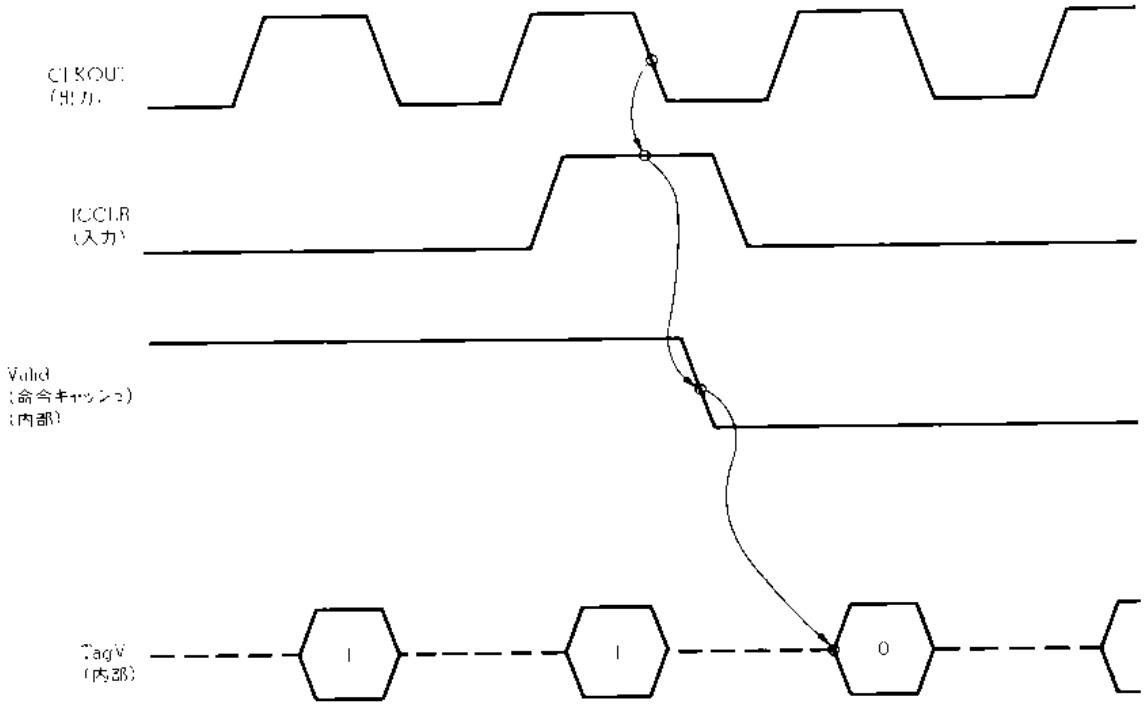


(b) データ・キャッシュ



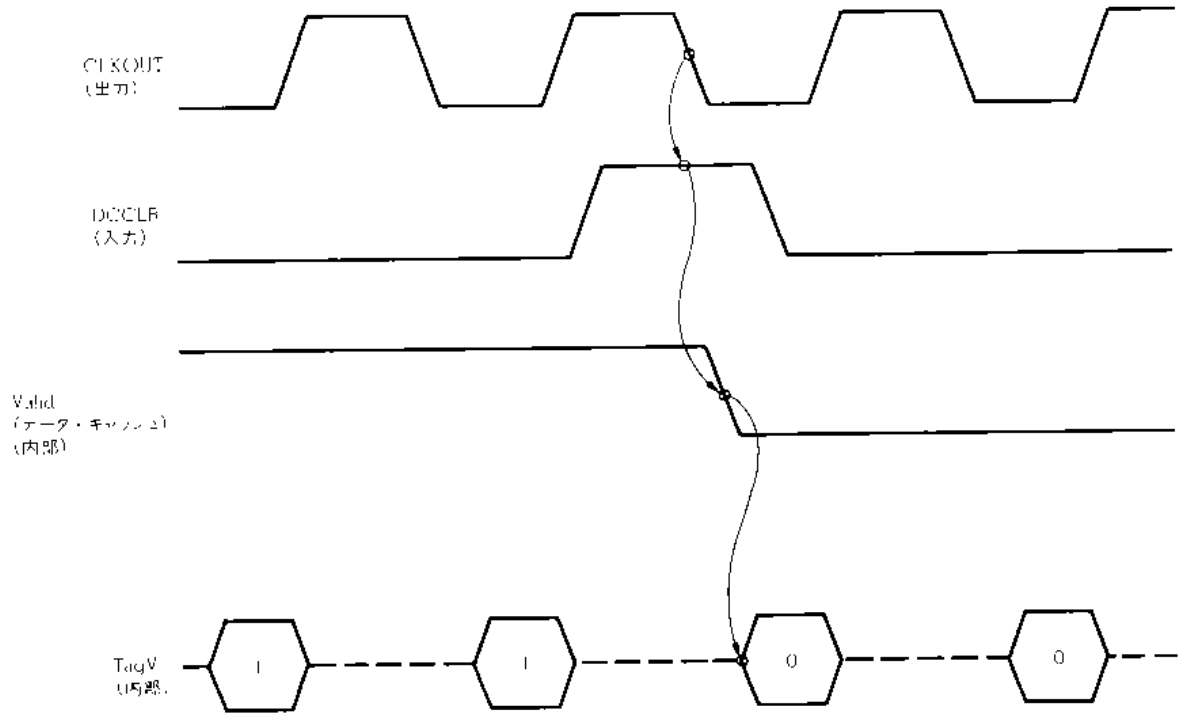
保守／廃止

図３-２　ICCLR信号のタイミング



備考　破線はハイ・インピーダンスを示します。

図３　３　DCCLR信号のタイミング



備考　破線はハイ・インピーダンスを示します。

保守／廃止

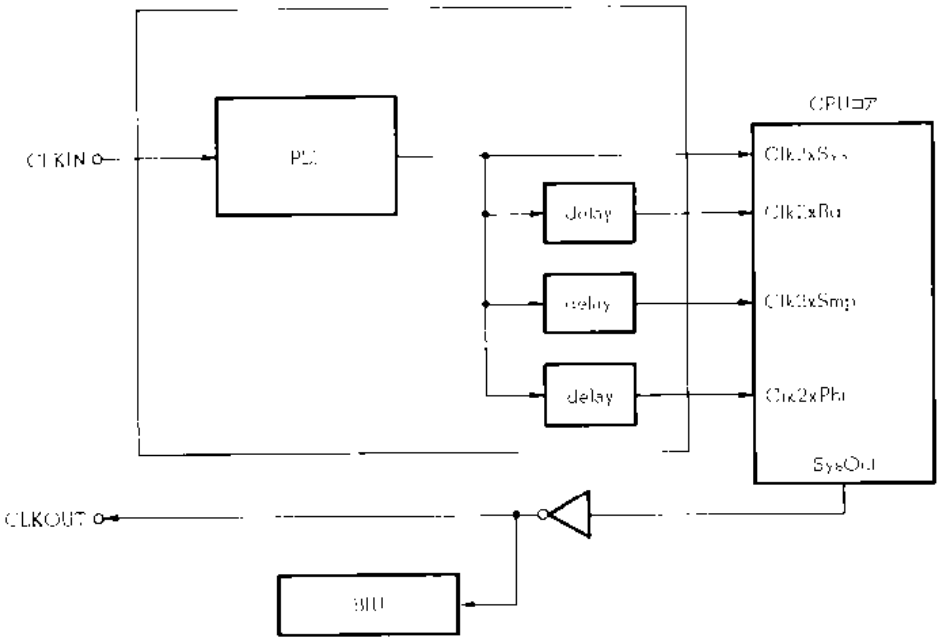
3.3　クロック・ジェネレータ

V_R3000Aは，４相のクロック入力を必要とします。V_R3800にはクロック・ジェネレータを内蔵しているため，外部からのクロック入力は，単相でCPUコアの動作周波数と同じ周波数です。クロック・ジェネレータは，入力クロックを内部のPLLで２逓倍してから遅延回路を通して４相クロックを生成し，これをCPUコアに供給します。

BIUは，CPUコアからのクロック出力（$\overline{SysOut}$）を基準に動作しています。また，SysOut信号を反転した信号はV_R3800の外部にも出力されています（CLKOUT信号），この信号はシステム構成時の基準クロックとなります。

図３-４に，クロック・ジェネレータのブロック図を示します。

図３-４　クロック・ジェネレータのブロック図



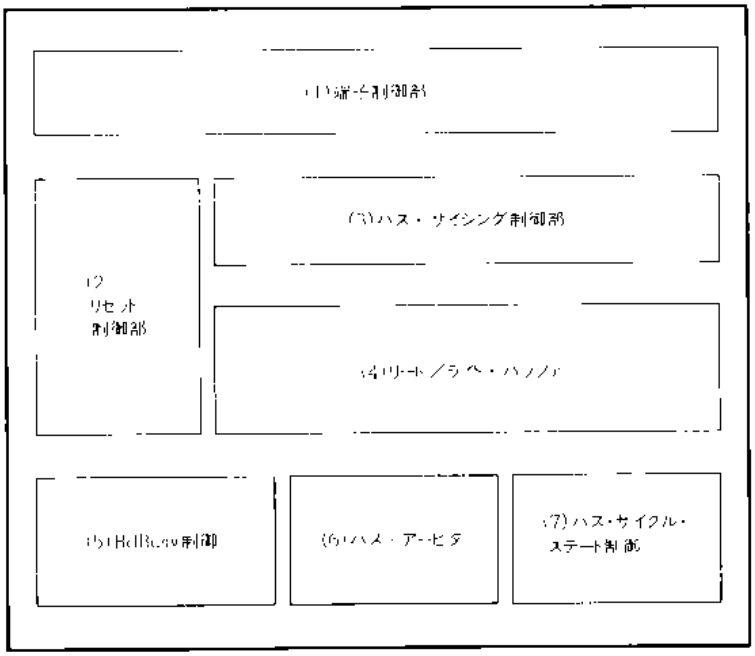
保守／廃止

3.4　バス・インタフェース・ユニット（BIU）

BIUは，CPUコアのアドレス・バスとデータ・バスをメイン・メモリ・システムから分離し，メイン・メモリにアクセスするブロックです。また，CPUコア，キャッシュ・メモリ，クロック・ジェネレータ以外の機能はすべてBIUの中に統合しています。

BIUの構成を図３　5に示します．

図３　5　バス・インタフェース・ユニットの構成



(1) 端子制御部

外部端子の入出力を制御します。入力端子のサンプリング状態を各制御部に伝えるとともに，出力信号を制御してバス・サイクルを生成します。

(2) リセット制御部

リセット制御部は，CPUコアと内部周辺ブロックをリセットします。また，CPUコアのリセット時の初期設定パラメータを入力します。さらに，V_R3800独自の初期設定パラメータであるキャッシュ制御や，バス・サイジングの初期設定も行います。詳細は，**3.5　リセット**を参照してください。

(3) バス・サイジング制御部

各シングル・リード・バス・サイクルにおいてRDYと同じタイミングでSIZRQ端子をサンプリングし，８ビット幅または16ビット幅のダイナミック・バス・サイジングを行います。８ビット幅にするか，16ビット幅にするかは，リセット時にDBUSSIZ端子で指定します

また，リセット時にSBUSSIZ端子をサンプリングして，16ビット幅の固定バス・サイジングを行います。詳細は，**4.6　ダイナミック・バス・サイジング**と**4.7　16ビット固定バス・サイジング**を参照してください。

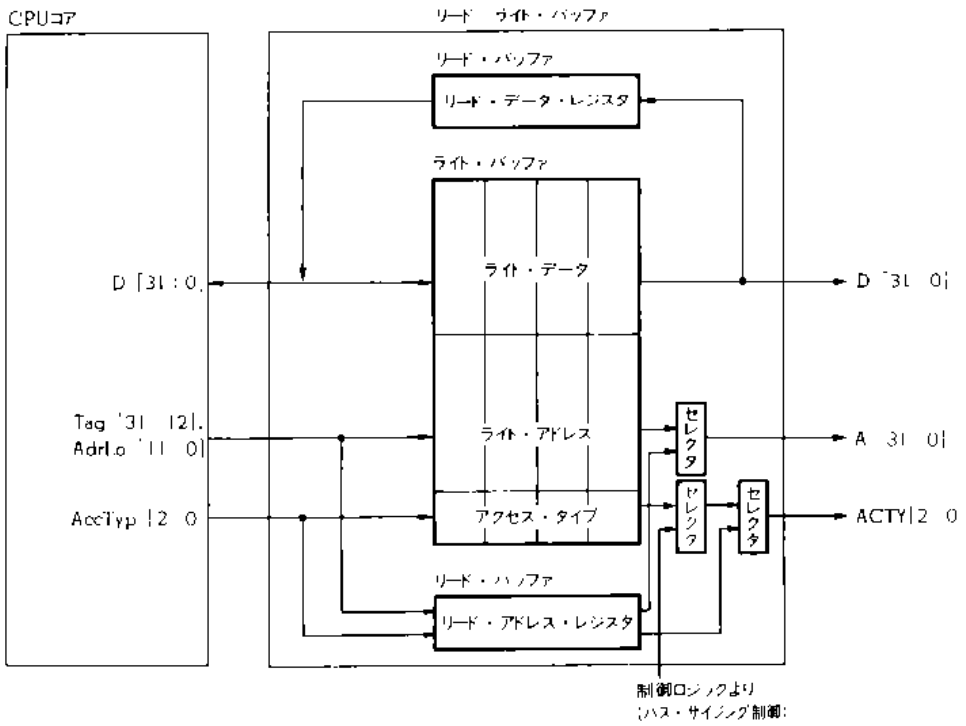
(4) リード・バッファ／ライト・バッファ

これまでのシステムでは，キャッシュ・メモリ・バスとメイン・メモリ・バスを共用しています。したがってライト・スルー方式のキャッシュ・メモリでメイン・メモリへのストア動作が発生した場合，キャッシュ・メモリとメイン・メモリへのストアが終了せず，完了するまで次の命令を実行できません。このため，大幅に性能が低下します。

V_R3000AではCPUとメイン・メモリ・バスの間にアドレス・バッファとデータ・バッファを置き，メイン・メモリへのストアをバッファに対して行うことでキャッシュ・バスとメイン・メモリ・バスを切り離しています。バッファはキャッシュ・メモリと同じで1クロックでストアが可能なため，次の命令実行を待たせることはありません。ただし，バッファを構成するロジック，または読み出し動作の制御も行うBIUが必要でした。

V_R3800ではBIUを内蔵し，さらにブロック・リード転送にも対応します。図3-6に，リード／ライト・バッファのブロック図を示します。

図3-6　リード／ライト・バッファのブロック図



(a) ライト・バッファ

アドレス，データ，およびアクセス・タイプの各４段FIFOバッファとなっており，CPUコアからストア動作が４命令連続して発生しても，パイプラインを止めずに演算を実行できます。

また，ライト・バッファにデータがあるかを調べるための命令（BC0TとBC0F）を用意しています。この命令により，ライト・バッファ内のデータを出力し，ライト・データ間隔の調整やライト優先のメモリ・アクセスを実現できます。詳細は**5.2(1)　ライト・バッファ・エンプティ待ち合わせ機能**を参照してください。

(b) リード・バッファ

アドレスとデータの１段バッファです。連続したロード命令やキャッシュ・ミスが発生しても最初のリード・データをプロセッサに返さなければ次のデータを読み出さないため，多段のバッファは不要です。V_R3800では４ワードのブロック・リード転送をサポートし，ブロック・データの転送レートをBLKW端子で設定できます。

(5) リード・ビジィ（RdBusy）制御

ブロック・リード転送時にBLKW端子で設定したデータ転送レートに応じてV_R3000Aの調停を制御します。

(6) バス・アービタ

バス・アービタはメイン・メモリへのリード要求，ライト要求，バス・ホールド要求の優先順位を制御します。要求の優先順位は次のとおりです。

バス・ホールド　＞　メモリ・リード　＞　メモリ・ライト

リード要求をライト要求より優先させているので，命令フェッチが効率的に行われます。このため，ライト優先で制御することの多いV_R3000Aのシステムと比べてパイプラインの停止時間が最小限となり，より高い性能を実現します。

★　ただし，現在ライト・バッファにあるデータと同じアドレスにリード要求が発生した場合，ライト・データを優先的に出力します。こうして，アクセス順序が変わったことで古いリード・データを読むのを防ぎます。

(7) バス・サイクル・ステート制御

バス・サイクルの状態遷移を制御するステート・マシンです。

詳細は，**4.1　バス・サイクルの遷移**を参照してください。

保守／廃止

3.5　リセット

リセット制御部では，次の３つの制御を行います。

・CPUコアに入力する初期化情報の設定
・V_R3800独自の初期化情報の設定（キャッシュ・メモリ，バス・サイジング）
・内蔵キャッシュ・メモリの無効化

(1) リセット・タイミング

RESET信号をアサートするとリセットされます。

パワーオン・リセットの場合は，V_{DD}が安定してからRESET信号をアサートしてください，またPLL動作を安定させるため，リセットをディアサートするまでの期間は十分に取ってください（詳細は，**V_R3800　データ・シート**を参照してください）。

RESET信号をディアサートした次のクロックの立ち上がりで，動作モードを選択する信号をサンプリングします。モード選択については次の**(2)　モード選択**を参照してください。

(2) モード選択

V_R3000Aでは，リセット時の初期設定情報を割り込み端子Int[5：0]と兼用し，各端子を４回サンプリングして入力していました。V_R3800では，その情報のうち大半は固定とし，一部は専用端子で１回サンプリングします。

また同時に，次のようなV_R3800固有の機能に関する設定も行います。

・キャッシュ・メモリのリフィル・サイズの選択
・ダイナミック・バス・サイジング時のバス幅の選択
・16ビット固定バス・サイジングの選択

表３-２と表３-３にモード選択の一覧，表３-４にV_R3800で固定にしたV_R3000Aの初期情報を示します。

保守／廃止

表３-２　モード選択

端子名	モード	説　　明
ISTREAM	IStream	命令ストリーミング 1：命令ストリーミングを行う。 0：命令ストリーミングを行わない。
STPART	StorePartial	部分ワード・ストア時の動作 1．データ・キャッシュにヒットすると，そのエントリを更新する． 0　データ・キャッシュの対応エントリを無効化する。
BIGEND	BigEndian	エンディアン 1：部分ワード・ストア時のデータ・バス上のデータがリトル・エンディアン順序で整列される。 0：部分ワード・ストア時のデータ・バス上のデータがビッグ・エンディアン順序で整列される。

備考１．命令ストリーミング：

命令キャッシュ・ミスによるブロック・リード・サイクルで，V_R3800が受け取った命令データを命令キャッシュに書き込むと同時に，CPUコアが命令フェッチを行い，命令を実行する機能です．この機能を許可すると，命令キャッシュ・ミスによるパイプラインの停止時間が減少し，トータルの性能が向上します。

２．ストア・パーシャル：

キャッシュ領域に部分ワード・ストアを行うときに，まずデータ・キャッシュの当該ラインを読み出します。タグをチェックしてそのラインがキャッシュ・ヒットなら，読み出したワード・データとストアする部分ワード・データをマージし，ワード・データとしてデータ・キャッシュに書き戻します。メイン・メモリにはワードでストアします。そのラインがミス・ヒットした場合は，メイン・メモリだけに部分ワード・ストアを行います。

この機能を禁止している場合には，キャッシュ・メモリからの読み出しは行いません。データ・キャッシュの当該ラインを無効化するとともにメイン・メモリに部分ワード・ストアを行います。

よって，この機能を使用することで，キャッシュ・メモリへのヒット率の向上が期待できます。ただし，16ビット固定バス・サイジングを行っている場合には，ストア・データとACTYの関係が変わりますので注意してください。詳細は，**4.7　16ビット固定バス・サイジング**を参照してください。

保守／廃止

表３－３　V_R3800固有の動作モード選択

端子名	設定	機　　能
IBLOCK	1	命令キャッシュのリフィルをブロック転送（４ワード）で行う。
	0	命令キャッシュのリフィルをシングル・リードと同じサイクルで行う。
DBLOCK	1	データ・キャッシュのリフィルをブロック転送（４ワード）で行う。
	0	データ・キャッシュのリフィルをシングル・リードと同じサイクルで行う
DBUSSIZ	1	16ビット幅のダイナミック・バス・サイジングを行う。
	0	８ビット幅のダイナミック・バス・サイジングを行う。
SBUSSIZ	1	16ビット幅の固定バス・サイジングを行う。
	0	16ビット幅の固定バス・サイジングを行わない。

表３－４　V_R3800の固定モード

V_R3000Aでのモード	V_R3800での設定	説　　明
DBlkSize[1:0]	1　1（４ワード）	データ・キャッシュのリフィル・サイズ
IBlkSize[1:0]	1　1（４ワード）	命令キャッシュのリフィル・サイズ
DispParRexEnd	0（アクティブ）	パリティ表示　エンディアン反転 ステータス・レジスタのRE（ビット25）を１にすると，ユーザ・モードでのエンディアンが初期設定のものと反転する
PhaseDelayOn	1（インアクティブ）	フェーズ・ロック 浮動小数点演算用コプロセッサがないので無意味。
R3K	0（アクティブ）	V_R3000Aイネーブル CPUコアはV_R3000Aとして動作する。
ExCache	1（アクティブ）	キャッシュ・サイズ拡張 キャッシュ・アドレスとしてAdrLo[17:16]を出力する。
MPAdrDisable	0（アクティブ）	マルチプロセッサ・ストール・アドレス・ディスエーブル マルチプロセッサ・ストールをサポートしないので無意味。
IgnoreParity	0（アクティブ）	パリティ無視 キャッシュ・データのパリティ・チェックを行わない。
MPS	0（インアクティブ）	マルチプロセッサ・ストール サポートしない。
TriState	1（インアクティブ）	３ステート出力 V_R3000Aの出力はハイ・インピーダンスにはならない。 V_R3800の出力をハイ・インピーダンスにするにはHLDRQ端子を使用する。
NoCache	1（インアクティブ）	キャッシュなし動作 キャッシュ付きで動作する。
BusDriveOn	1（アクティブ）	データ　タグ駆動制御 ライトのウエイト・ストール中にデータとタグを駆動する。

保守／廃止

3.6　割り込み

６つのハードウエア割り込みと２つのソフトウエア割り込みをサポートします。また，各割り込み要求を個別にマスクできます。

(1) ハードウエア割り込み

次にハードウエア割り込み受け付けのシーケンスについて説明します。

① INT［5：0］端子にハードウエア割り込み要求を入力する。

② INT［5：0］端子の状態をサンプリングし，原因レジスタのIP［5：0］フィールド（割り込み要求保留）の該当ビットに入力する。

これらの端子はレベルセンスで，CLKOUTの立ち下がりごとにサンプリングします。

③ ②のサンプリングで，次の内容をチェックする」

・CPUコアの状態がラン，またはフィックスアップ・サイクルにあるか。

・ステータス・レジスタのIntMask領域の該当ビットとIEcビットの両方が１にセットされている（割り込み許可）か。

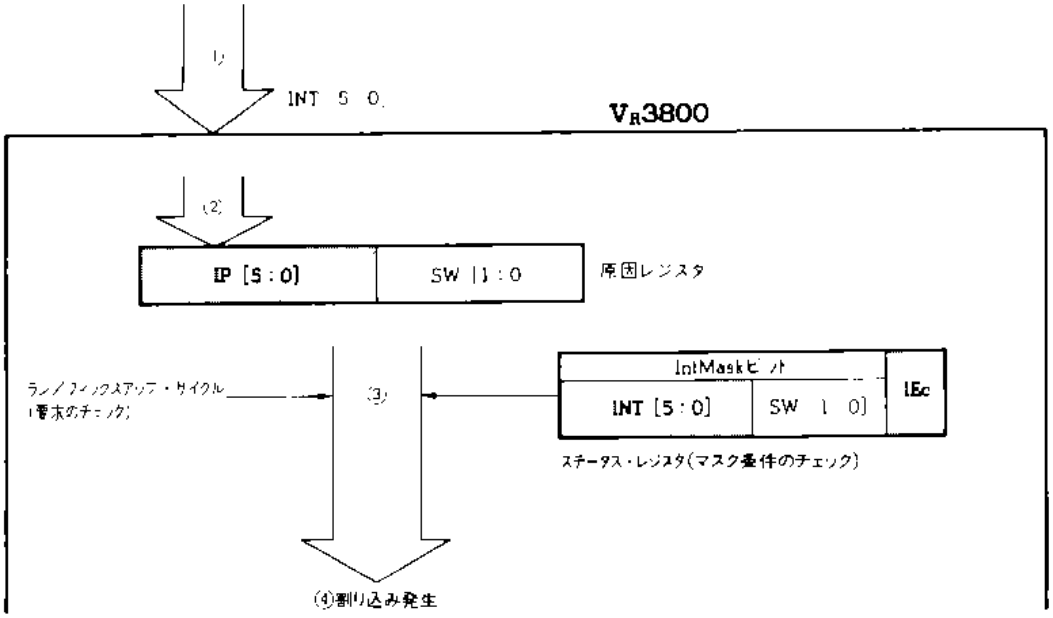
④ ③の条件を満たすと，割り込みが発生する。

注意　原因レジスタのIP［5：0］ビットを調べれば，割り込みの保留状況が分かります。ただしIP［5：0］は，CLKOUTの立ち下がりごとにINT端子の状態を反映するだけなので，割り込みが発生しても，そのときのINT端子の状態を保持しません。このため，割り込み処理ルーチンで原因レジスタを読み出すまでは，該当のINT端子をアサートしてください。そうしないと，６本のうちのどのINT端子で割り込みが要求されたのか判別できなくなります。

備考　割り込みの優先順位はありません。

保守／廃止

図３－７　ハードウエア割り込み処理のシーケンス



3

(2) ソフトウエア割り込み

次にソフトウエア割り込み受け付けのシーケンスについて説明します。

① 原因レジスタのSW［1：0］の該当ビットを直接ソフトウエアで1にセットする。

② 次の内容をチェックする。

・CPUコアの状態がラン，またはフィックスアップ・サイクルにあるか，

・ステータス・レジスタのIntMask領域の該当ビットとIEcビットの両方が1にセットされている（割り込み許可）か。

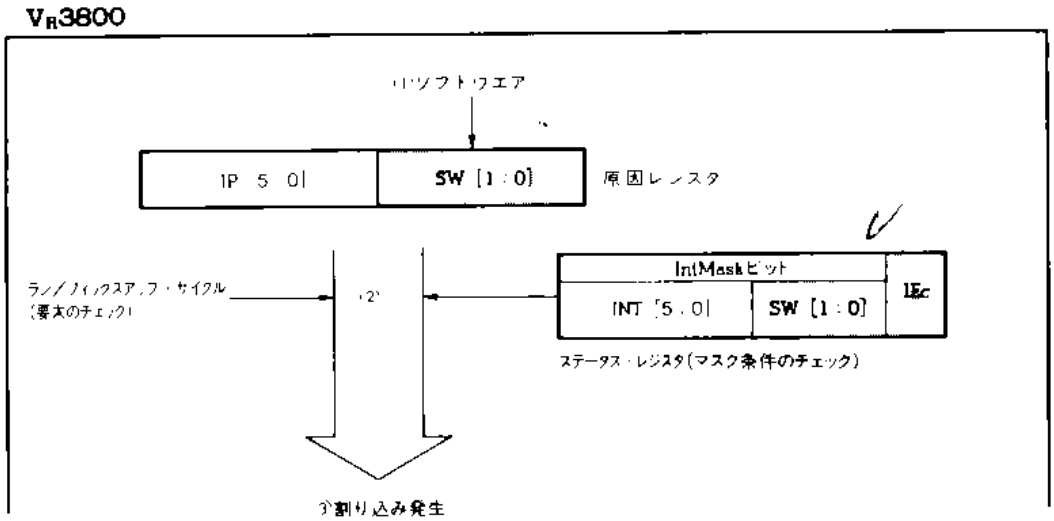
③ ②の条件を満たすと，割り込みが発生する。

注意　SW［1：0］ビットを調べれば，割り込みの保留状況が分かります。
割り込み処理が終了するまで（RFE命令で原因レジスタのKUcビットとIEcビットが例外発生前の状態に戻るまで）に原因レジスタのSWビットを0にクリアしてください。この操作をせずに割り込み許可を行う（IEcビットとIntMaskビットをセット）と，再び同じ割り込みが発生することがあります。

備考　割り込みの優先順位はありません。
なお，原因レジスタ，ステータス・レジスタについての詳細は，**第10章　例外処理**を参照してください。

保守／廃止

図３－８　ソフトウエア割り込み処理のシーケンス



第4章　バス・オペレーション

4.1　バス・サイクルの遷移

V_R3800は，Ready制御（RDY端子）による汎用メモリ・インタフェースを採用しています。メイン・メモリのアクセス時間に合わせて，リード／ライト・バス・サイクルをRDY信号で待ち合わせることができます。

バス・サイクルの状態遷移を図4－1に示します。V_R3800には，次の3つのバス・サイクルの状態があります。

●T1…バス・サイクルの最初の状態
●T2…バス・サイクルの最初以外の状態
●Ti…メモリ・アクセス要求のない状態

ただし，T2にはブロック・リード・サイクルを示す状態と，そうでない状態があります。また，Tiにもバス・ホールド中の状態と，そうでない状態があります。これらの状態は区別なく，ただT2，またはTiとだけ表記してありますので注意してください。

Ti状態に遷移する条件は次の5つです。

① リセット動作の次の状態
② リード要求，ライト要求がないときのTiの次の状態
③ リード要求，ライト要求，バス・サイジング要求のないReady（RDY＝0）状態のT2（ブロック・リード以外）の次の状態
④ リード要求，ライト要求のないブロック・リードの最後のT2の次の状態
⑤ バス・ホールド状態

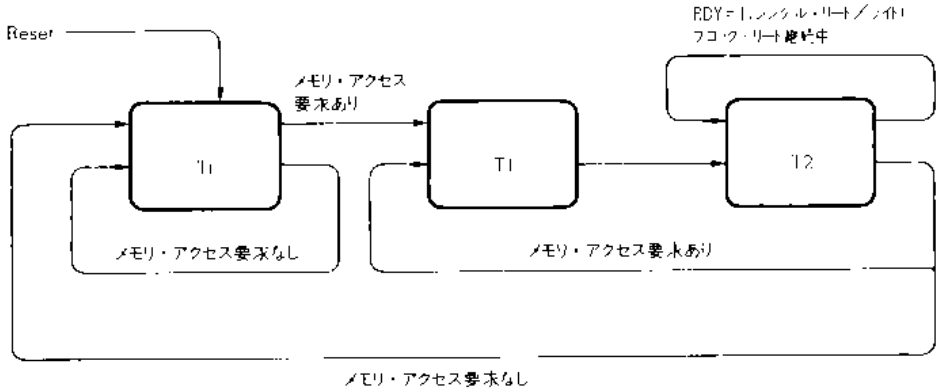
次の状態がTiでもT2でもない場合，T1に遷移します。
T2状態に遷移する条件は次の3つです。

① T1の次の状態
② Not Ready（$\overline{RDY}$＝1）状態のT2の次の状態
③ ブロック・リード・サイクル実行時

バス・ホールド中でない（HLDAK＝H）Ti期間中では，A［31：0］とACTY［2：0］は常に駆動されて

います。ただし，その値は必ずしも前の値を引き継ぐわけではありません。バス・ホールド状態でないTiからBCYST信号がアクティブになるまでのA［31：0］とACTY［2：0］の値は不定です。

図4－1　バス・サイクルの状態遷移



4.2　シングル・リード・サイクル

非キャッシュ領域でのリード要求発生時，およびキャッシュ・ミス・ヒット発生時にリフィル・サイズを1ワードに設定した場合発生します。詳細は**4.4　ブロック・リード・サイクル**を参照してください。

次にシングル・リード・サイクル中の動作について説明します。

★　①　T1状態でアドレスとアクセス・タイプの出力を開始すると同時に，バス・サイクルの開始を示すBCYST信号がアクティブになります。

②　その半クロック後にDA信号がアクティブになります。

★　③　次にT2状態に移ります。この状態ではアドレスとアクセス・タイプをそのまま出力しますが，BCYST信号はインアクティブになります。

また，T2状態のクロックの立ち下がりではRDY信号をサンプリングします。

○サンプリングの時点でRDY信号がインアクティブ（ハイ・レベル）の場合

バス・サイクルは再度T2状態に移ります。これがウエイト状態です。ウエイト状態では，DA信号はアクティブのままです。また，クロックの立ち下がりごとにRDY信号をサンプリングし，RDY信号がインアクティブの間はT2状態を繰り返します。

○サンプリングの時点でRDY信号がアクティブ（ロウ・レベル）の場合

DA信号はインアクティブになり，バス・サイクルはT2状態から終結します。

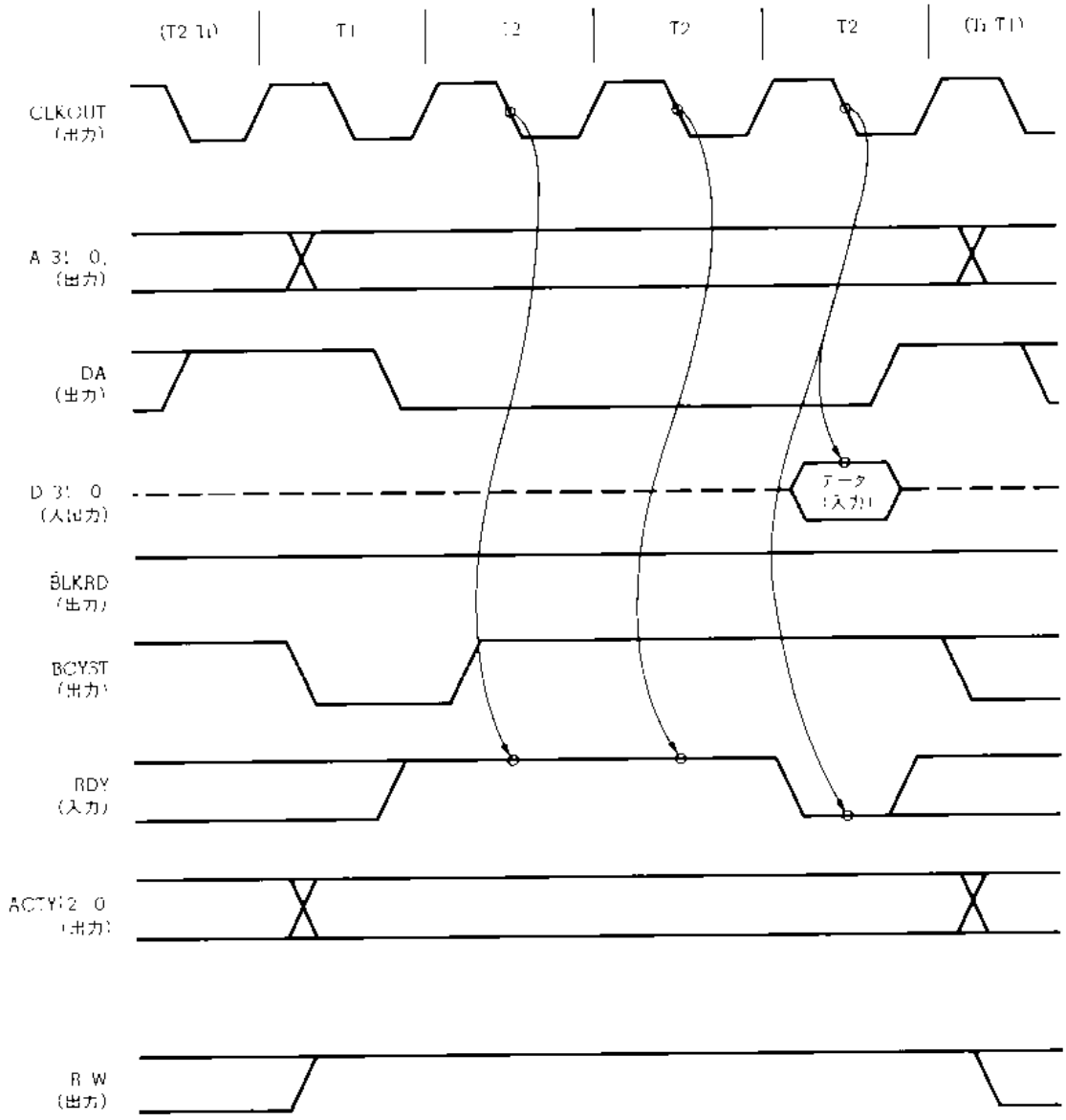
データ・バス上のデータは，最後のT2状態のクロックの立ち下がり時点で読み込まれます。これは，RDY

信号のアクティブ・レベルのサンプリングと同じタイミングです」

シングル・リード・サイクルでは，アドレスはワード・アラインされ，アドレス・バスの下位2ビットは必ず“00”になります。部分ワード・アクセスではACTY［1：0］はCPUコアが要求するデータ長を示しますが，アドレスはワード・アラインされるため，アドレスとACTY［1：0］の関係が正しくならないことがあります。システムはACTY［1：0］に関係なく，該当ワードのすべてをアクセスしてください。必要なバイトを認識させる必要はありません。ただしI/Oなどで特定のバイトしかアクセスがない場合には，そのバイトだけのアクセスが可能です。

図4－2に，シングル・リード・サイクルのタイミングを示します。

図4－2　シングル・リード・サイクル



備考　破線はハイ・インピーダンスを示します。

保守／廃止

4.3　ライト・サイクル

ライト要求発生時に発生します。V_R3800はライト・スルー方式のキャッシュ・メモリ構成のため，キャッシュ領域，非キャッシュ領域に関係なく，すべてのライト要求で行われます」

ライト・サイクルでは，DA信号がアクティブになるのと同じタイミングで，データを書き込み始めます。その後，T1状態かTi状態のクロックの立ち上がりまでデータを書き込み続けます。

ライト・サイクルでは，D［31：0］，R/W以外の端子の状態は，シングル・リード・サイクルと同じです。

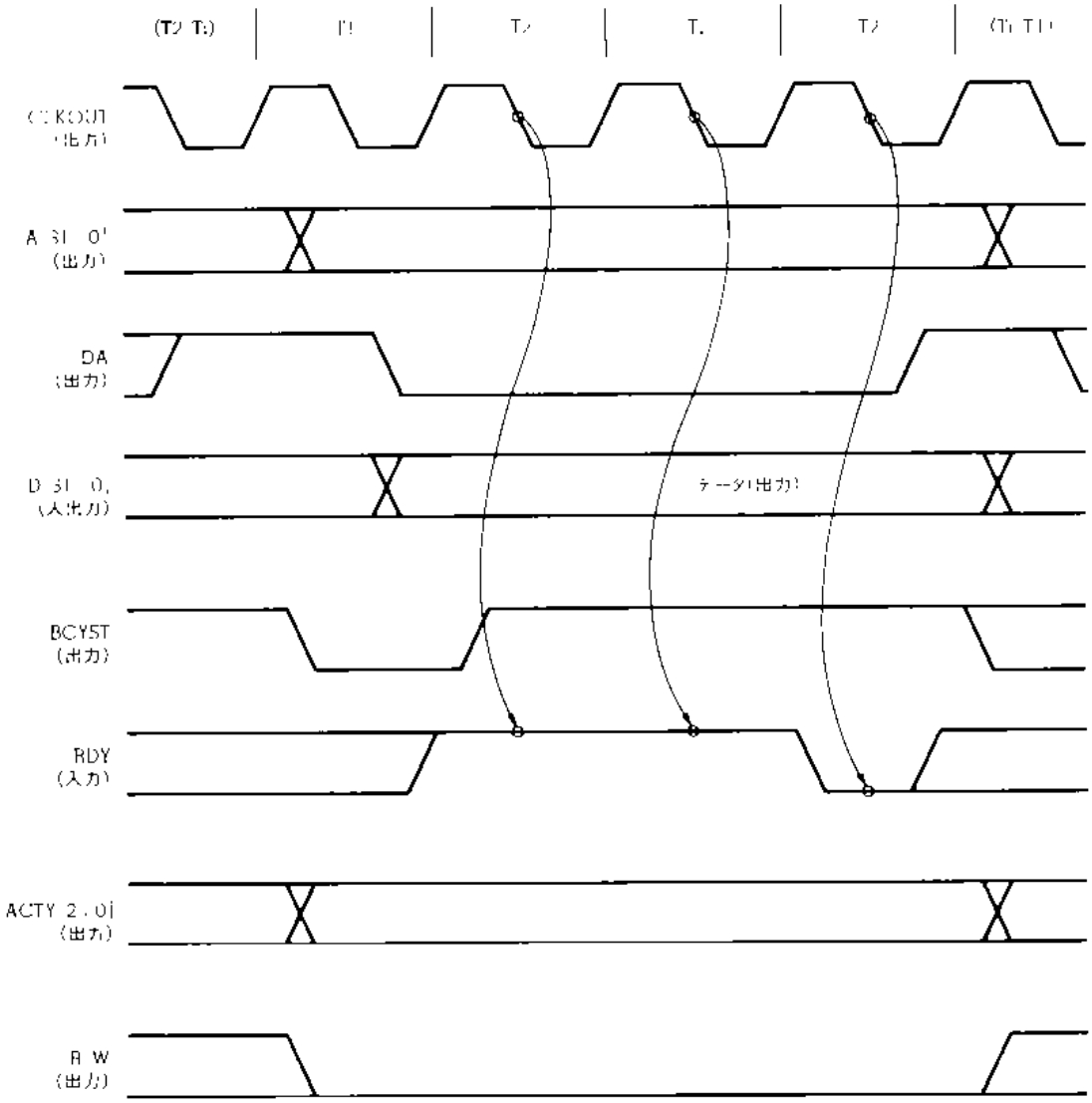
V_R3800がライト・サイクルで出力するアドレスの下位２ビットとアクセス・タイプは，CPUコアが出力する値をそのまま使用しています。このため，ストア・パーシャル端子（STPART）をハイ・レベルに設定し，かつ部分ワード・ストアがデータ・キャッシュにヒットする場合は，アクセス・タイプはワードになります。したがって，次の点に注意してください。

注意　STPART端子をハイ・レベルに設定した場合，アクセス・タイプがワードでも，アドレスの下位２ビットが０とは限りません。アドレスの下位２ビットに関係なく，A31-A2でアドレシングするワード・データすべてが有効です。

図４－３に，ライト・サイクルのタイミングを示します。

保守／廃止

図4－3　ライト・サイクル



4

保守／廃止

4.4　ブロック・リード・サイクル

キャッシュ可能な領域のリード・アクセスに対し，そのデータがキャッシュにない場合は，リフィル・サイクルを起動して必要なデータをキャッシュ内に取り込もうとします。

キャッシュのリフィルは，1ワード（4バイト）または4ワード（16バイト）単位で行います．プログラムのデータ参照には局所性があるため，一般には参照するアドレスに対して，その付近のデータをまとめて（この場合は4ワード）取り込むと効率的です。しかし，分岐が頻発するような局所性のないプログラムでは，一度のメモリ参照で4ワードも読み込んでいたのでは性能が低下します。そこでV_R3800では，プログラムの性質に応じてユーザがリフィル・サイズを選択できるようになっています。リフィルのサイズを1ワードにするか4ワードにするかは，リセット時にIBLOCK端子とDBLOCK端子で指定します。

リフィルのサイクル数に1ワードを指定した場合のバス・サイクルの動作は，シングル・リード・サイクルと同じになります。

一方，リフィルのサイクル数に4ワードを指定した場合は，ブロック転送を行います。図4－4と図4－5に，ブロック・リード・サイクルのタイミングを示します。

また，あとで示すように，4ワードのキャッシュ・リフィルでは最大4クロック周期の同期バス・サイクル（RDY信号の状態にかかわらず，一定クロックで終結するバス・サイクル）で動作します。最大4クロックとは2ウエイト相当です」メイン・メモリの参照に2ウエイトより多くのウエイト・サイクルが必要なシステムにおいて，ブロック・リードのための特殊なハードウエアを設けていないシステムでは，キャッシュのリフィル・サイズを1ワードに指定しておけば，すべてのメモリ・サイクルを$\overline{RDY}$端子で制御できます，つまりキャッシュのリフィル・サイズを1ワードに指定することで，アクセス時間の遅い（ブロック転送で，最低速のクロック・バス・サイクルでも対応できない）メモリを使用したシステムにも対応できます。

次にブロック・リード・サイクル中の動作について説明します。

★ ① T1状態でアドレスとアクセス・タイプの出力を開始すると同時に，バス・サイクルの開始を示す$\overline{BCYST}$信号がアクティブになります。そして，同時にバス・サイクルがブロック・リード・サイクルであることを示す$\overline{BLKRD}$信号がアクティブになります。

② その半クロック後に$\overline{DA}$信号がアクティブになります。

★ ③ 次にT2状態に移ります。この状態ではアドレスとアクセス・タイプをそのまま出力しますが，$\overline{BCYST}$信号はインアクティブになります。

シングル・リード・サイクルと同様に，T2状態のクロックの立ち下がりではRDY信号をサンプリングします，

○サンプリングの時点で$\overline{RDY}$信号がインアクティブ（ハイ・レベル）の場合

バス・サイクルは再度T2状態に移ります．これがウエイト状態です。ウエイト状態ではDA信号はアクティブのままです、また，クロックの立ち下がりごとにRDY信号をサンプリングし，RDY

信号がインアクティブの間は，T2状態を繰り返します。

○サンプリングの時点でRDY信号がアクティブ（ロウ・レベル）の場合

次のクロックの立ち上がりからブロック・リードの読み込みを開始します。

ブロック・リードを行っているときのバスの状態もT2ですが，これはシングル・リード・サイクルやライト・サイクルでのT2とは異なります。バス・サイクルが完結していないという点からT2になっているだけです。シングル・リードやシングル・ライトとは異なり，ブロック・リード・サイクルでデータ転送を開始してからのT2ではRDY信号をサンプリングしません。このため，いったんブロック・リード・サイクルのデータ転送に移ったバス・サイクルは，RDY信号によるウエイト制御を受けません。なお，ブロック・リード・サイクルのT2の期間，T1に引き続いて，BLKRD信号はずっとアクティブになっています。

④ ブロック・リードのデータ転送が始まったときのT2の最初のクロックの立ち下がりで，第1の読み込みデータがサンプリングされます。これは，シングル・リードの場合とは異なり，RDY信号の状態とは無関係です（シングル・リードでは，RDYのロウ・レベルと同時にサンプリング）。その後，第2，第3，第4の読み込みデータをクロックの立ち下がりごとにサンプリングします。これで，ブロック・リード・サイクルが終結します。

なお，入力データをサンプリングする間隔は，メイン・メモリのアクセス時間に応じて，1クロック間隔から4クロック間隔までを，BLKW端子で指定できます。

BLKW [1：0]	サンプリングの間隔
0 0	1クロック
0 1	2クロック
1 0	3クロック
1 1	4クロック

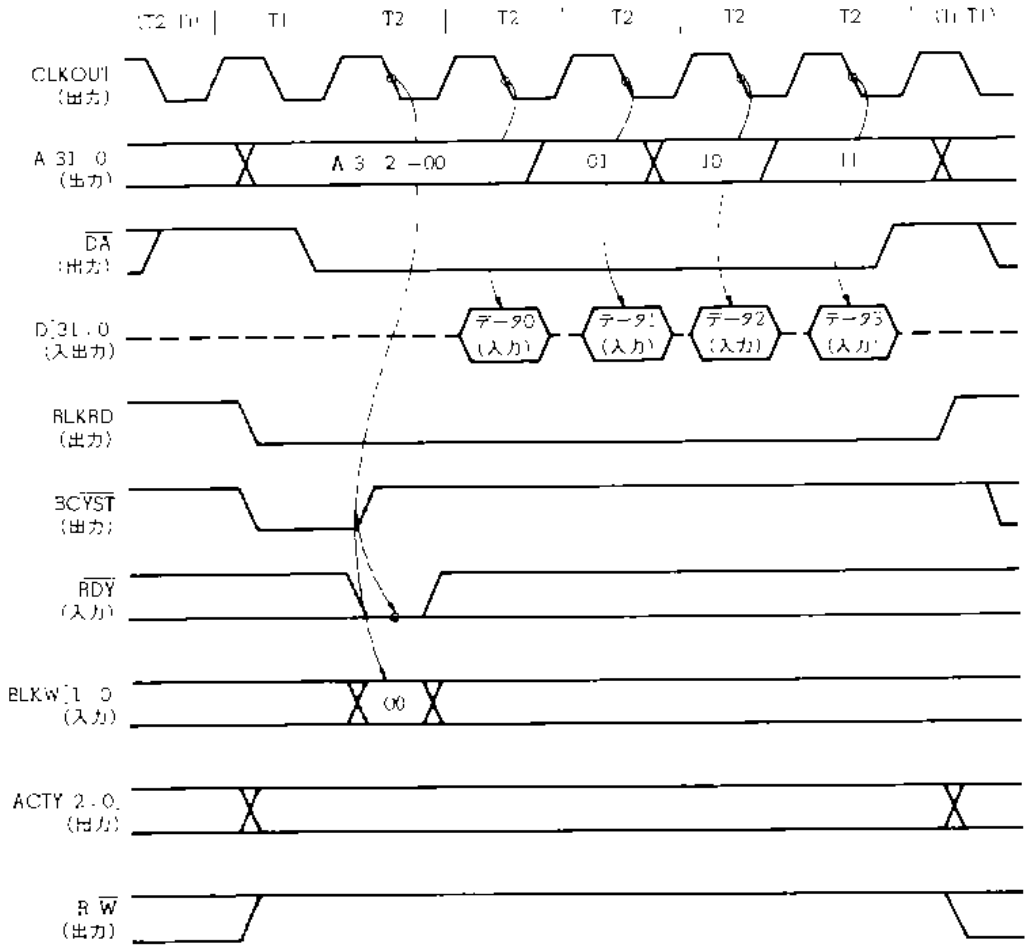
BLKW信号のサンプリングはRDY信号と同タイミングで行います。

ブロック・リード・サイクルでは，読み込みデータをサンプリングすると同時に，アドレスは次のデータのアドレスに切り替わります。通常，アドレス・バスはT1のクロックの立ち上がりで変化しますが，この場合はT2のクロックの立ち下がりで変化します。

また，DA信号はブロック・リード・サイクルの期間中アクティブです。DA信号はT1サイクルのクロックの立ち下がりでアクティブになり，最後のT2のクロックの立ち下がりでインアクティブになります。

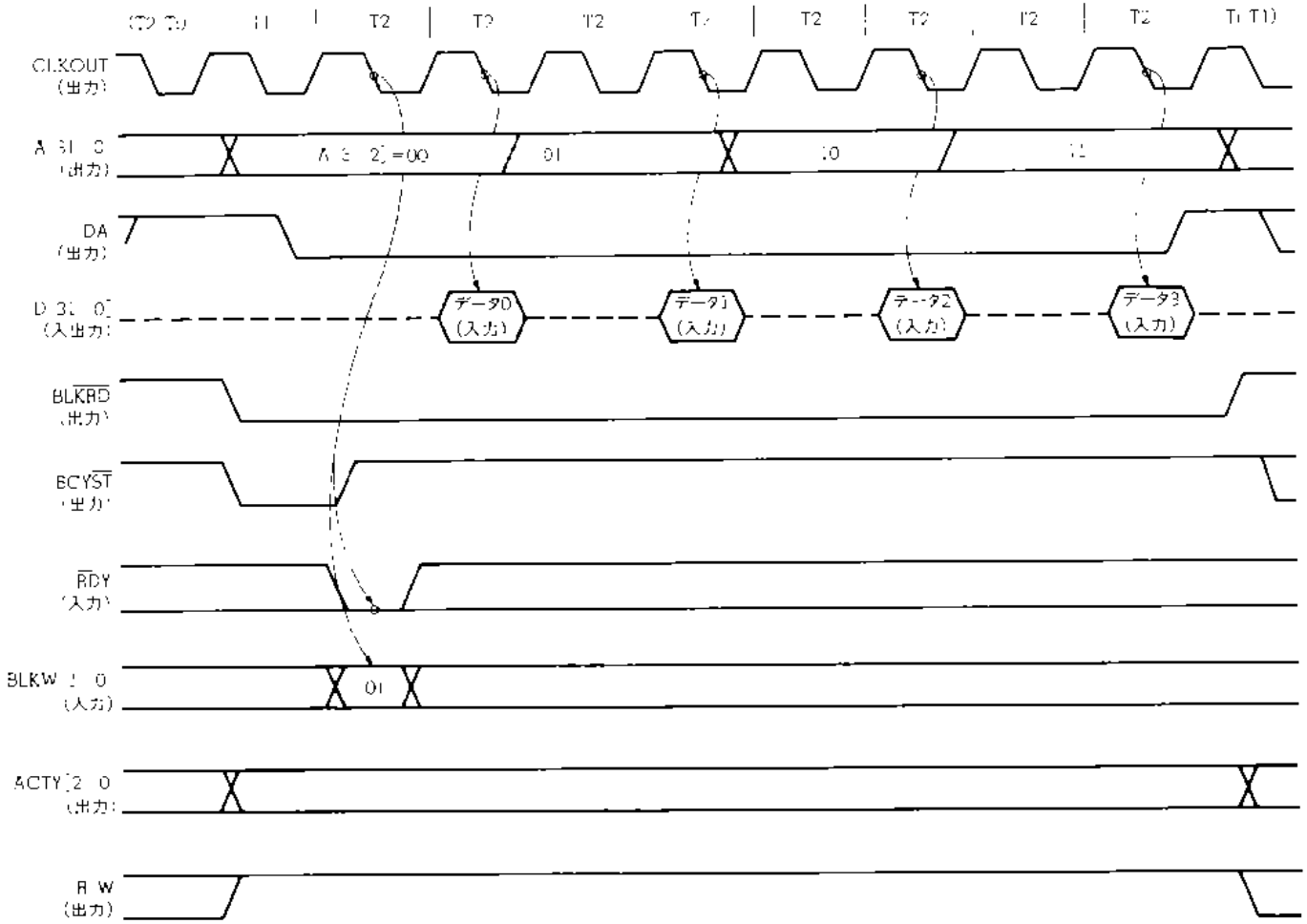
備考　ブロック・リード・サイクルでは，アドレス・バスは必ず16の倍数のアドレスから始まります。つまり，キャッシュ・ミスしたアドレスに下位で一番近い16の倍数のアドレスから4ワード（16バイト）のデータを読み込みます。プログラムのディバグなどでアドレス・バスをトレースする場合は注意してください。

図4-4　ブロック・リード・サイクル（BLKW'1　0　=00の場合）



備考　破線はハイ・インピーダンスを示します

図4－5 ブロック・リード・サイクル（BLKW 1：0 =01の場合）



備考 破線はハイ・インピーダンスを示します。

保守／廃止

4.5　バス・ホールド

ほかのバス・マスタがV_R3800に対してバスの使用権を要求した場合，V_R3800をバス・ホールド状態にできます．このときV_R3800は，ほとんどすべての出力端子をフローティング（ハイ・インピーダンス）状態にして，バスを外部から切り離します．バス・ホールド状態でフローティングになる端子については，**表2-1　端子機能**を参照してください．

4.5.1　バス・ホールド状態への移行

(1) サンプリング時にバス・サイクルが完結していない場合

HLDRQ信号がアクティブな状態をサンプリングした1.5クロック後に，バス・サイクルがまだ完結していなければ（ウエイト状態，バス・サイジング，またはブロック転送中の場合），そのバス・ホールド要求を無視します．

V_R3800を確実にバス・ホールド状態に移行させるためには，バス・サイクルが完結するまで，またはHLDAK信号がアクティブになるまで，バス・ホールド要求を入力してください．

なおこの場合は，バス・サイクルの完結を待ってバス・ホールド状態になります．

(2) サンプリング時にバス・サイクルが完結している場合

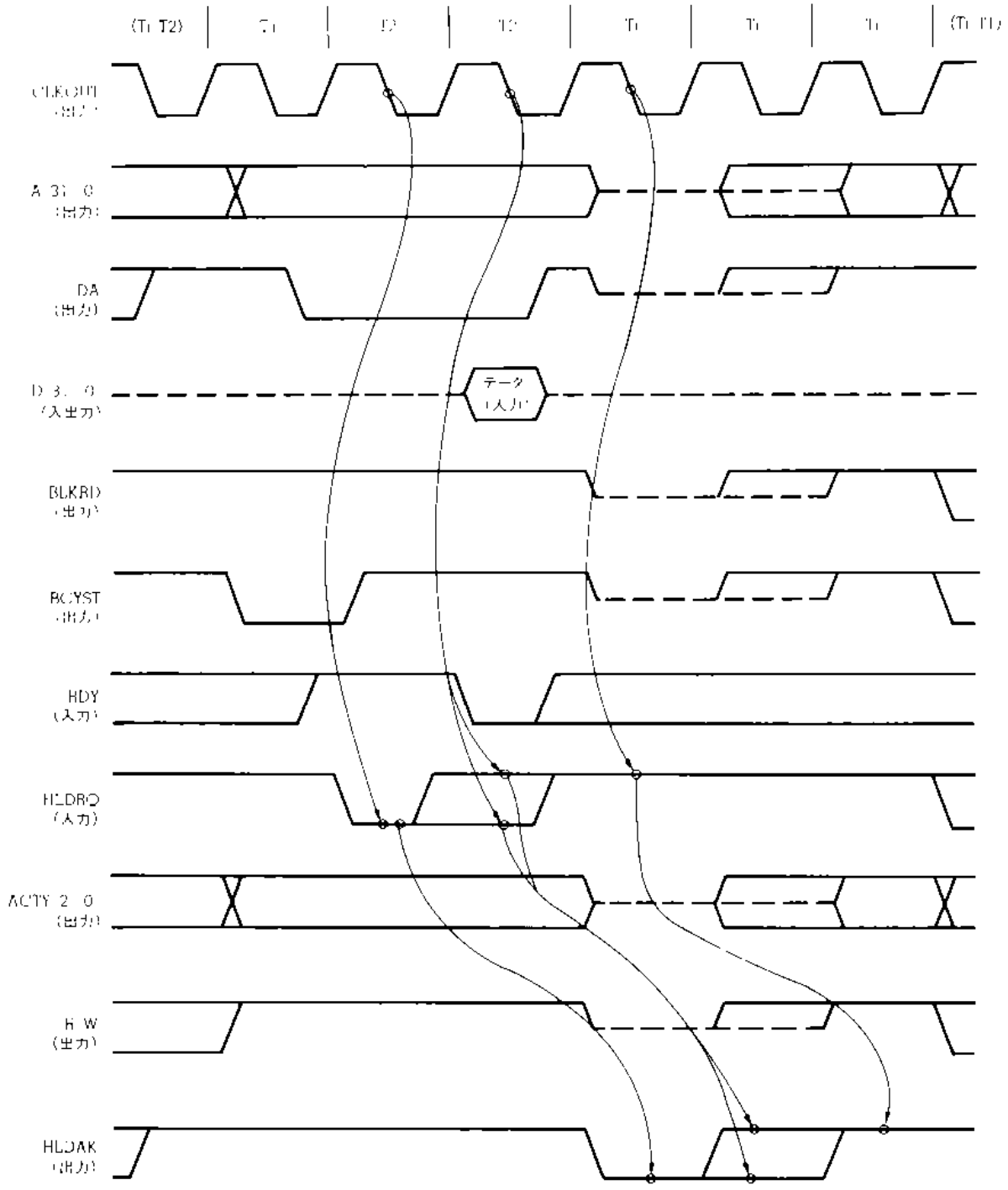
バス・ホールド状態への移行はHLDRQ端子で要求します．HLDRQ信号の状態はクロックの立ち下がりごとにサンプリングします．HLDRQ信号がアクティブな状態をサンプリングした1.5クロック後，バス・サイクルが完結していれば，V_R3800はバス・ホールド状態になり，$\overline{\text{HLDAK}}$信号をアクティブにします．その後はHLDRQ信号がインアクティブな状態をサンプリングするまで，バス・ホールド状態を継続します．

バス・ホールド状態（$\overline{\text{HLDAK}}$=L）でのバス・サイクル状態はTiです．メモリ・アクセス要求がない場合のバス・サイクル状態もTiですが，バス・ホールド状態でない（$\overline{\text{HLDAK}}$=H）ときは，出力信号はハイ・インピーダンスになりません．バス・ホールド状態が解除されると，バス・ホールド状態のTiは，必ずバス・ホールド状態でないTiに移行します．そこでメモリ・アクセスが発生すれば，T1状態に移ってバス・サイクルを開始します．

図4-6から図4　8に，バス・ホールドのタイミングを示します．

図4-6　シングル・リードのバス・ホールド（1/2）

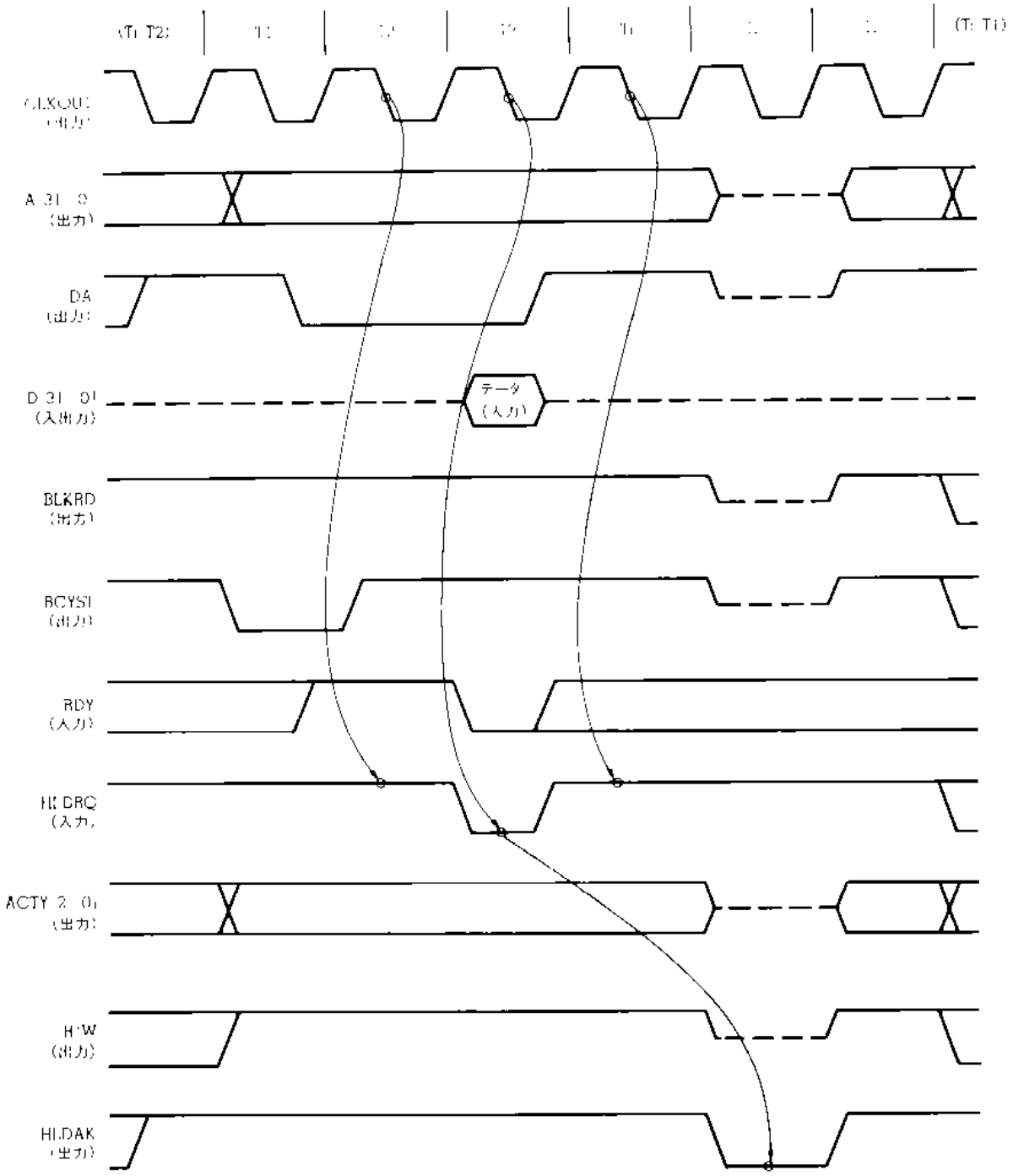
(a) HLDRQ＝Lのサンプリング時にバス・サイクルが完結していない場合



備考　破線はハイ・インピーダンスを示します。

図4-6　シングル・リードのバス・ホールド（2/2）

(b) HLDRQ＝Lのサンプリング時にバス・サイクルが完結している場合

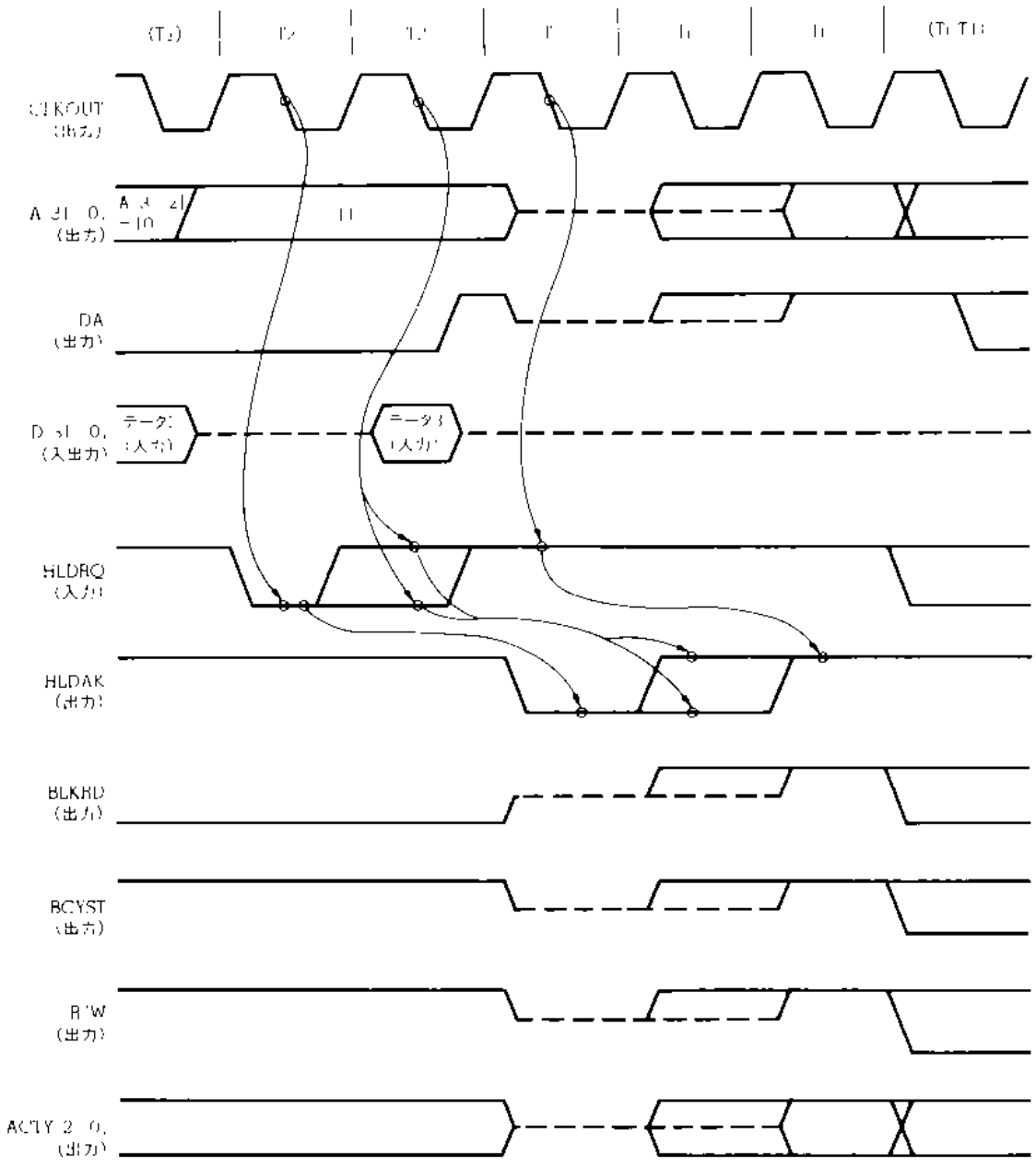


備考　破線はハイ・インピーダンスを示します．

保守／廃止

図4－7　ブロック・リードのバス・ホールド（BLKW｜1　0｜＝01の場合）（1／2）

（a）HLDRQ＝Lのサンプリング時にバス・サイクルが完結していない場合

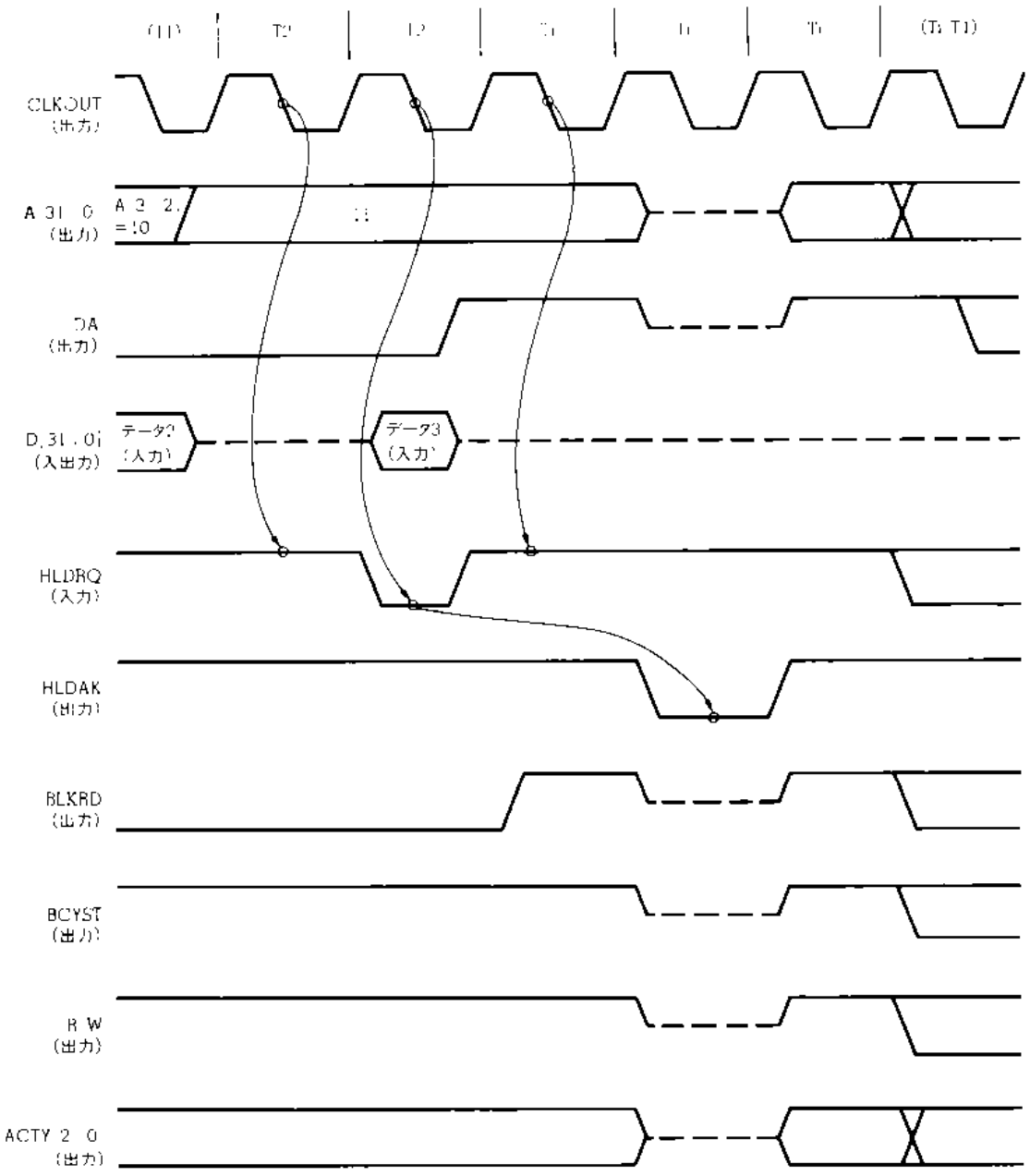


備考　破線はハイ・インピーダンスを示します．

保守／廃止

図4-7　ブロック・リードのバス・ホールド（BLKW　1：0　=01の場合）(2/2)

(b) HLDRQ=Lのサンプリング時にバス・サイクルが完結している場合

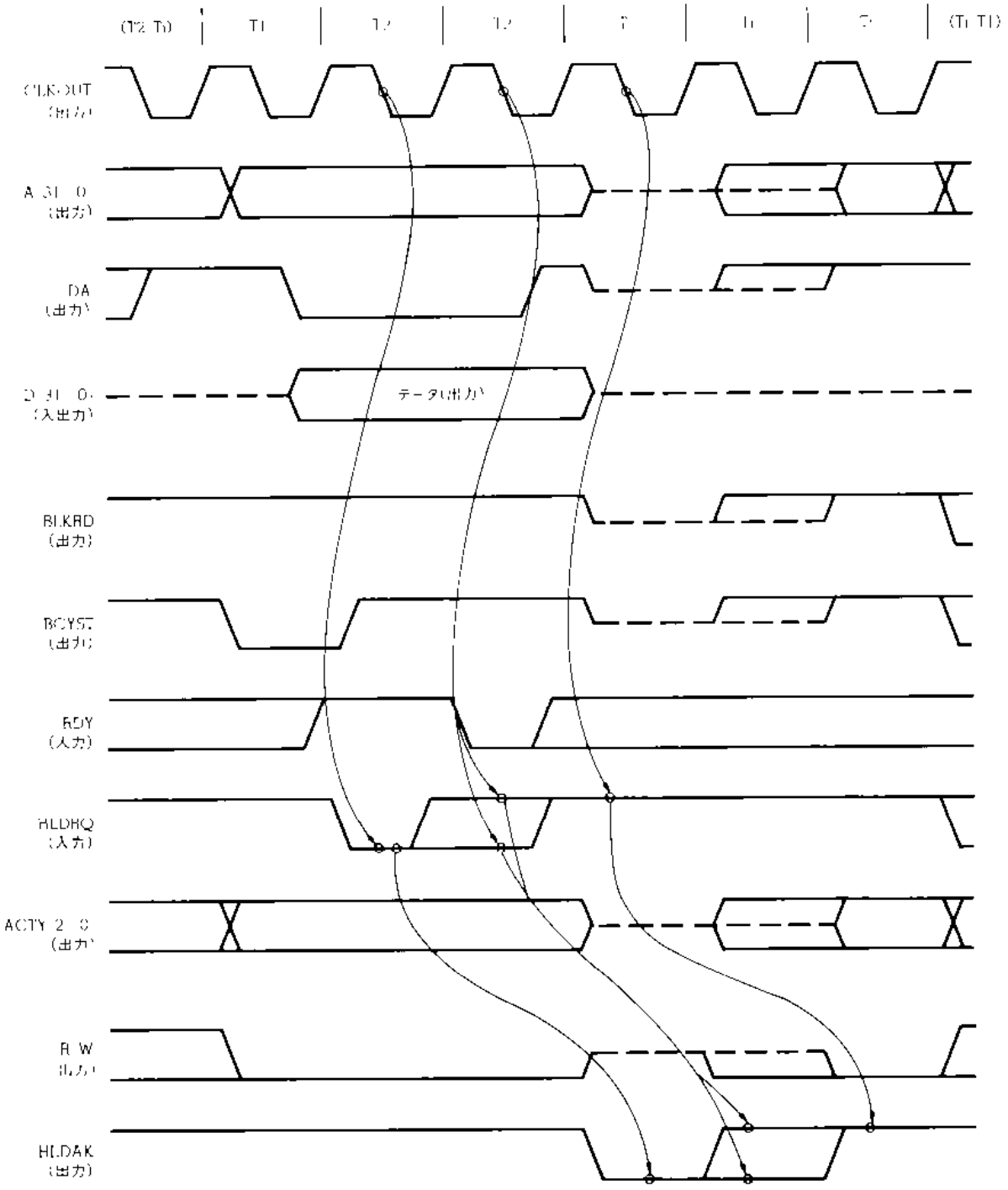


備考　破線はハイ・インピーダンスを示します。

保守／廃止

図4-8　ライトのバス・ホールド（1/2）

（a）HLDRQ＝Lのサンプリング時にバス・サイクルが完結していない場合

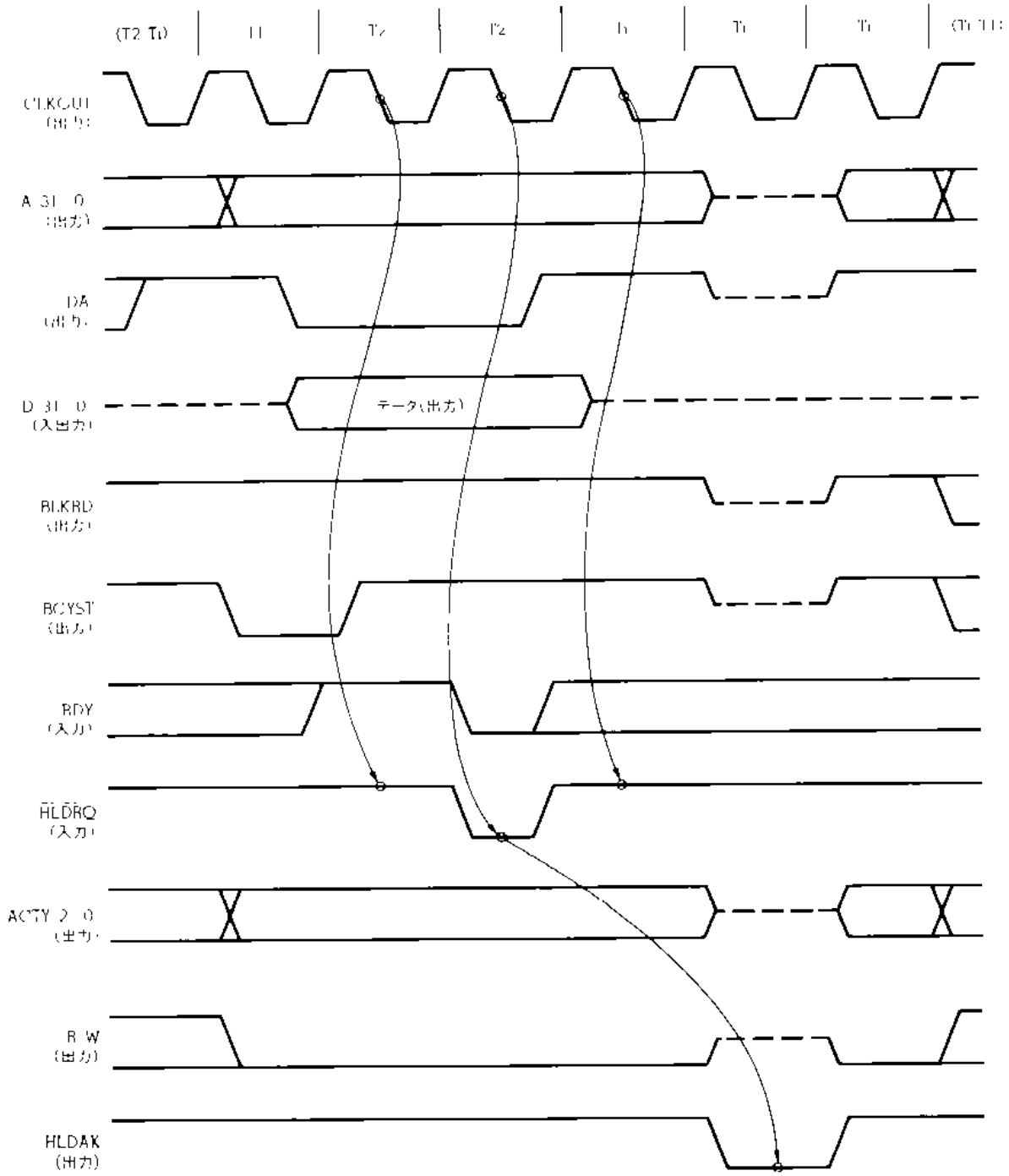


備考　破線はハイ・インピーダンスを示します。

保守／廃止

図4－8　ライトのバス・ホールド（2/2）

(b) $\overline{HLDRQ}$＝Lのサンプリング時にバス・サイクルが完結している場合



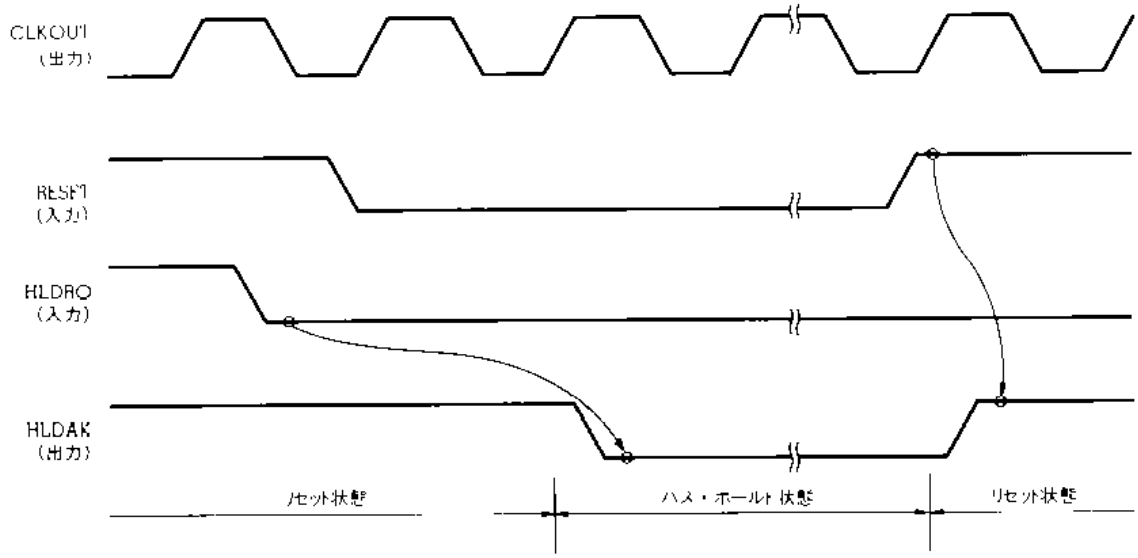
備考　破線はハイ・インピーダンスを示します

バス・ホールド状態では出力信号はハイ・インピーダンスになりますが，内部のCPUコアは，ライト・バッファが一杯になるか，非キャッシュ領域へのリード要求が発生するか，キャッシュ・ミス・ストールが発生するまでは動作を続けます（CPUが停止しているとはかぎりません）。

リセット期間中にHLDRQ信号をアサートした場合は，リセット解除後に1クロックのTi状態（これはリセット状態が遅延したものです）を経てHLDAK信号がアクティブになります。また，バス・ホールド状態にあるときにリセットを入力すると，バス・ホールド状態をすぐに解除してリセット状態に移行します。リセット状態とは，**表2－1　端子機能**の「リセット時の状態」に示されている状態です。これは，HLDAK信号がアクティブでない点以外はバス・ホールド状態と同じです。

図4－9に，リセット入力とバス・ホールドの関係を示します。

図4－9　リセット入力とバス・ホールド



保守／廃止

4.6　ダイナミック・バス・サイジング

プログラムを補助記憶装置などからメイン・メモリにダウンロードして実行するようなシステムでは，IPL（Initial Program Loader）などのブート用プログラムをROMから読み込んで実行します。このブートROMの数を削減できるように，V_R3800にはダイナミック・バス・サイジングの機能があります。

ブートROMでの使用を想定しているため，ダイナミックにバス・サイジングを行うことができるバス・サイクルは，基本的には，リセット例外ベクタを含む非キャッシュ領域（仮想アドレス空間でA0000000H-BFFFFFFFHの範囲）に対するリード・サイクルを対象にしています。

さらに，ブロック・リードを行わなければ（キャッシュのリフィル・サイズを1ワードに設定している場合），キャッシュ可能領域のリード・サイクルをダイナミックにバス・サイジングすることもできます。

注意　ブロック・リード・サイクルやライト・サイクルに対してダイナミック・バス・サイジングを行わないでください。その場合の動作は不定です。

4.6.1　8ビット・ダイナミック・バス・サイジング

8ビット・ダイナミック・バス・サイジングでは，CPUのロード命令のデータ長に関係なく，1回のリード・アクセス要求に対して4回のリード・サイクルを起動します。各バス・サイクルの動作は，シングル・リード・サイクルと同じです。このとき，アドレス・バスの下位2ビットは，"00→01→10→11"の順序で変化します。この間，アドレスの上位30ビット（A [31：2]）は変化しません。 ★

また，アクセス・タイプ（ACTY [1：0]）は現在のバス・サイクルではなく，CPUコアが要求しているデータ長を示します。

8ビット・ダイナミック・バス・サイジングは，リセット時にDBUSSIZ端子を0に設定すると使用できます。バス・サイジングを行うかどうかは，バス・アクセスごとに設定します。$\overline{\text{RDY}}$信号がアクティブになるのと同じタイミングで$\overline{\text{SIZRQ}}$信号の状態をサンプリングして決定します。リセット時にDBUSSIZ信号を0に設定してあり，サンプリングした$\overline{\text{SIZRQ}}$信号がアクティブならば，8ビット・ダイナミック・バス・サイジングを行います。

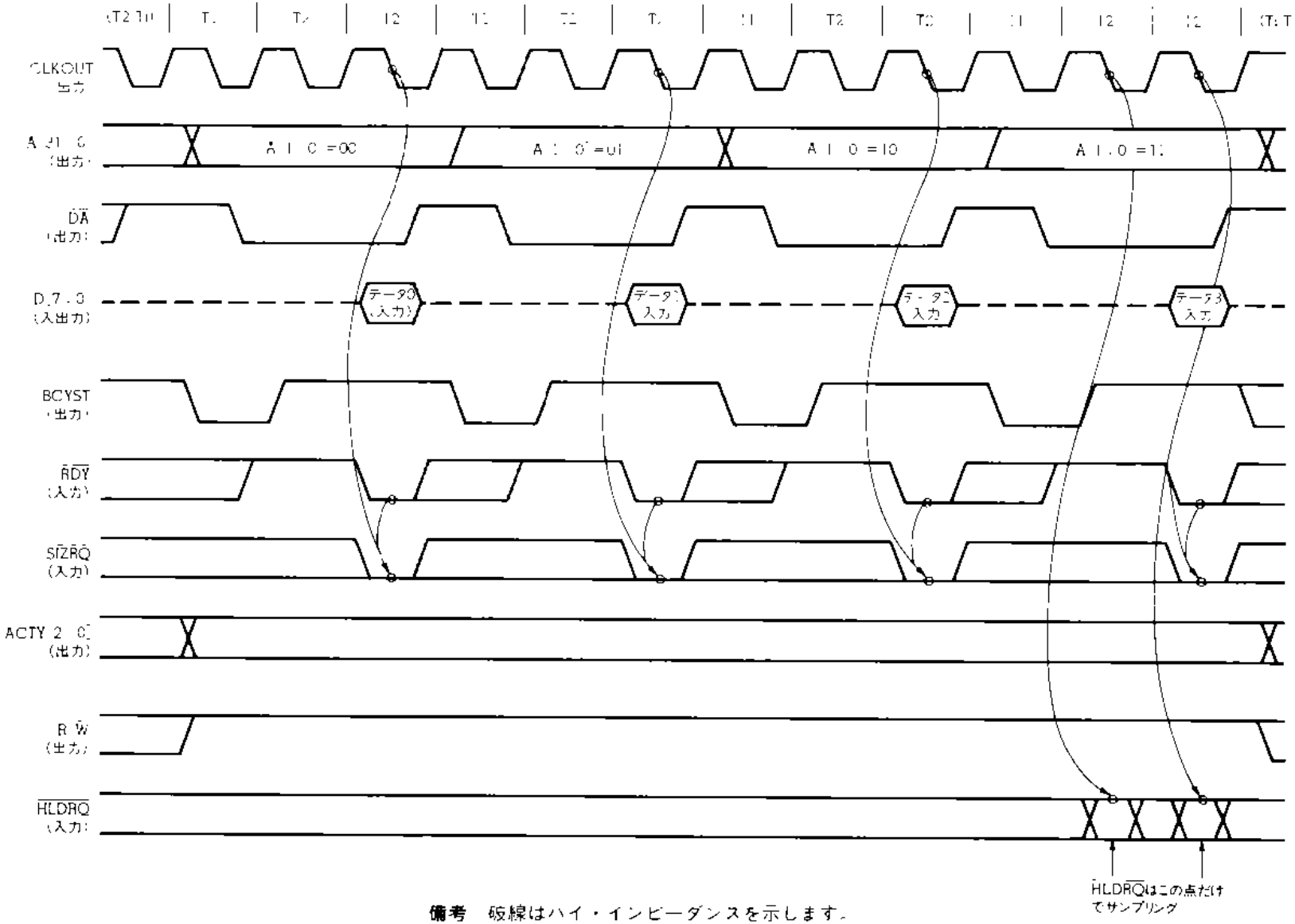
また，8ビット・バス・サイジング中にバス・ホールド要求を行っても，最後のA [1：0] = "11"のバス・サイクルでしか受け付けません。

図4-10に8ビット・ダイナミック・バス・サイジングのタイミングを示します。

注意　4回のバス・サイクルが終了するまでは$\overline{\text{SIZRQ}}$信号は後続の3回のサンプリング時点でもアクティブにしてください。インアクティブにした場合の動作は不定です。

備考　8ビット・ダイナミック・バス・サイジングでは，リード・サイクルにおいてデータ・バスの上位24ビットを駆動する必要はありません。

図4－10　8ビット・ダイナミック・バス・サイジング



備考　破線はハイ・インピーダンスを示します。

保守／廃止

4.6.2　16ビット・ダイナミック・バス・サイジング

16ビット・ダイナミック・バス・サイジングでは，CPUのロード命令のデータ長に関係なく，1回のリード・アクセス要求に対して2回のリード・サイクルを起動します。各バス・サイクルの動作はシングル・リード・サイクルと同じです。このとき，アドレス・バスの下位2ビットは，"00 ・10" の順序で変化します。この間，アドレスの上位30ビット（A [31：2]）は変化しません。 ★

また，アクセス・タイプ（ACTY [1：0]）は，現在のバス・サイクルではなく，V_R3800が要求しているデータ長を示します。

16ビット・ダイナミック・バス・サイジングは，リセット時にDBUSSIZ端子を1に設定すると使用できます。バス・サイジングを行うかどうかは，バス・アクセスごとに設定します。RDY信号がアクティブになるのと同じタイミングでSIZRQ信号の状態をサンプリングして決定します。リセット時にDBUSSIZ信号を1に設定してあり，サンプリングしたSIZRQ信号がアクティブならば，16ビット・ダイナミック・バス・サイジングを行います。

また，16ビット・ダイナミック・バス・サイジング中にバス・ホールド要求を行っても，最後のA [1：0] = "10" のバス・サイクルでしか受け付けません。

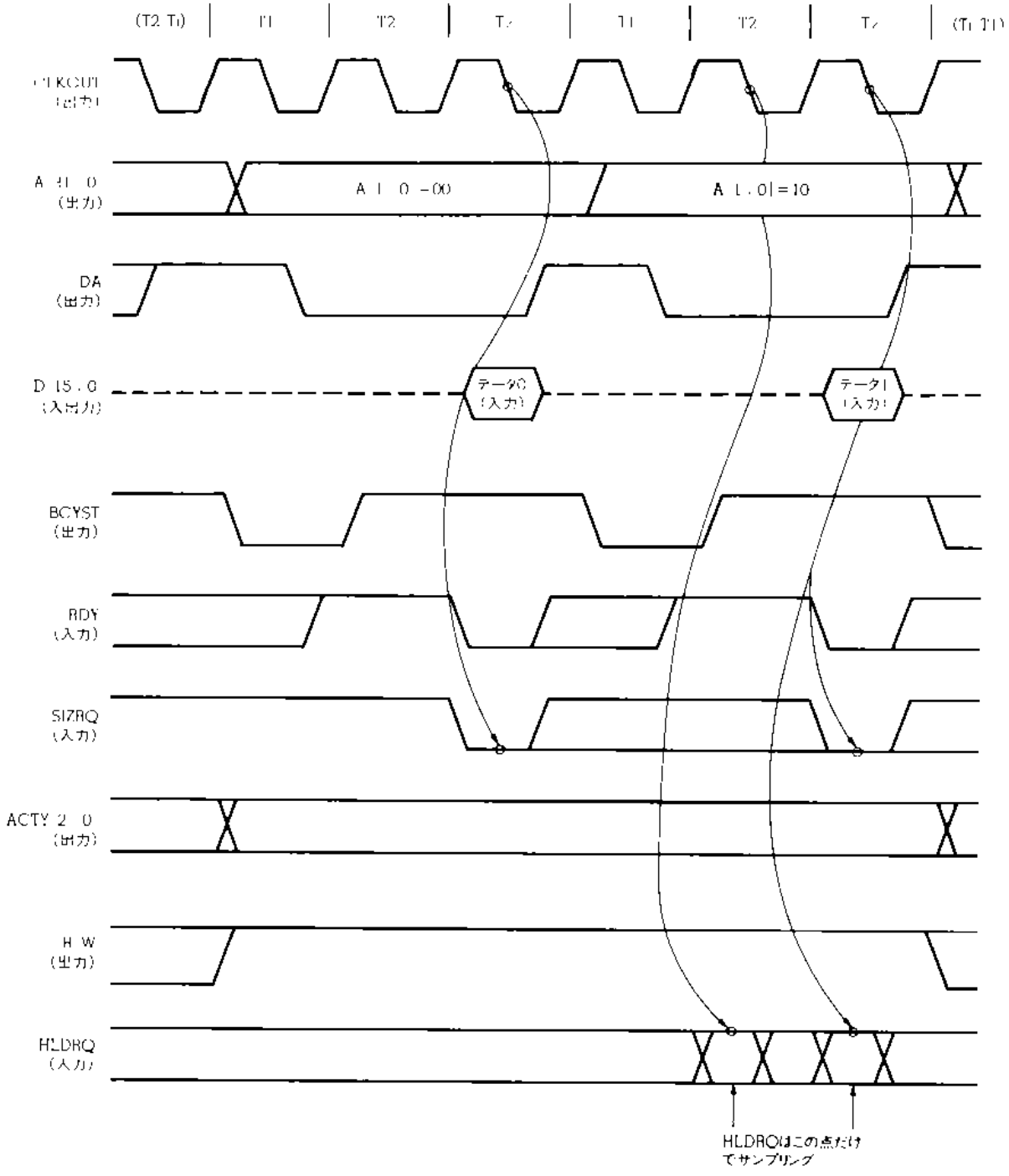
図4－11に，16ビット・ダイナミック・バス・サイジングのタイミングを示します。

注意　2回のバス・サイクルが終了するまで，SIZRQ信号を後続の1回のサンプリング時点でもアクティブのままにしてください。インアクティブの場合の動作は不定です。

16ビット・ダイナミック・バス・サイジングでは，リード・サイクルでデータ・バスの上位16ビットを駆動する必要はありません。

保守／廃止

図4-11　16ビット・ダイナミック・バス・サイジング



備考　破線はハイ・インピーダンスを示します。

保守／廃止

4.7　16ビット固定バス・サイジング

16ビット固定バス・サイジングは，システム・バスのビット幅を減少させることでシステム・コストを低減させます．コスト・パフォーマンスを考慮すると，システム・バスのビット幅を16ビットにした方がよい場合があります．

V_R3800はキャッシュを内蔵しているため，キャッシュのヒット率が高くなる小規模のプログラムを実行する場合，バス幅を16ビットにしても性能はあまり低下しません．たとえば，Dhrystoneベンチマークの実測値では約5％性能が低下するだけです．

16ビット固定バス・サイジングの使用はリセット時に決定します．リセット時にSBUSSIZ信号の状態がアクティブであれば，16ビット固定バス・サイジングを行います．

ダイナミック・バス・サイジングの場合とは異なり，16ビット固定バス・サイジングでは，リード・サイクルとライト・サイクルの両方をバス・サイジングの対象としています．**ただしブロック・リード・サイクルのバス・サイジングはサポートしません．このため，リセット時にキャッシュのリフィル・サイズを1ワードに設定（IBLOCK=0，DBLOCK=0）してください．**キャッシュのリフィル・サイズを4ワードに設定した場合の動作は不定です．

図4－12と図4　13に，それぞれ16ビット固定バス・サイジングのリード・サイクルとライト・サイクルのタイミングを示します．

(1) リード・サイクル

16ビット固定バス・サイジングでは，CPUのロード命令のデータ長に関係なく1回のメモリ・リード・アクセスに対して2回のリード・サイクルを起動します．それぞれのバス・サイクルの動作はシングル・リード・サイクルと同じです．このとき，アドレス・バスの下位2ビットは必ず，"00・10" の順序で変化します．また，アクセス・タイプ（ACTY［1：0］）は，現在のバス・サイクルではなく，CPUコアが要求しているデータ長を示しています．★

(2) ライト・サイクル

16ビット固定バス・サイジングのライト・サイクルの起動回数は，ライト・データのアクセス・タイプによって異なります．アクセス・タイプが1ワード（4バイト）か3バイトの場合は2回のライト・サイクルを起動し，ハーフ・ワード（2バイト）か1バイトの場合は1回のライト・サイクルを起動します．データ・バスの下位16ビットだけが駆動される点を除いて，各バス・サイクルは，バス・サイジングを行わない場合のライト・サイクルと同じです．このときアドレス・バスには，データを書き込むべきバイト・アドレスを出力します

ライト・サイクルでは，データ・バスに出力している16ビットのデータがすべて有効だとはかぎりません．メモリ・システムは，アクセス・タイプ，アドレス・バスの下位2ビットおよびエンディアンから有効な（メモリに書き込んでよい）バイト・データを認識する必要があります．これらの関係については，**4.8.3　バス・サイジング時のデータ・バス上のバイト・データの整列**を参照してください．

保守／廃止

16ビット固定バス・サイジングで2回のライト・サイクルを起動する場合（ワード，3バイト・データのライト），1回目と2回目のバス・サイクルでアクセス・タイプが変化します。

1回目のバス・サイクル：V_R3800が出力しているデータのアクセス・タイプを出力する
2回目のバス・サイクル：次に示すようにアクセス・タイプが変化する

表4－1　アクセス・タイプの変化

1回目のアクセス・タイプ	2回目のアクセス・タイプ
ワード	ハーフ・ワード
3バイト（A［1．0］=00）	バイト
3バイト（A［1．0］=01）	3バイト

これは，3バイト・データを書き込む場合，アクセス・タイプが3バイトのままだと，2回目のバス・サイクルのアドレスの下位2ビットが，"10" のときに，データ・バス上のデータが1バイト有効なのか，2バイト有効なのかが区別できなくなるためです。

注意　V_R3800が1回目のライト・サイクルで出力するアドレスの下位2ビットとアクセス・タイプは，CPUコアが出力する値をそのまま使用しています。このため，ストア・パーシャル端子（STPART）を1にして動作している場合で，部分ストアがデータ・キャッシュにヒットする場合は，アクセス・タイプはワードになります。したがって，次の点に注意してください。

①　アクセス・タイプがワードでも，アドレスの下位2ビットが0だとはかぎりません。アドレスの下位1ビットに関係なく，A31-A1でアドレシングするハーフ・ワード・データすべてが有効です。

★ **②　アクセス・タイプがワードでも，それがバイトまたはハーフ・ワードの部分ストアによるもので，かつアドレスが "10" か "11" で始まる場合は，バス・サイクルは一度しか起動しません。**

16ビット固定バス・サイジングで動作中も，8ビット・ダイナミック・バス・サイジングが可能です。この場合は，リード・サイクルだけがダイナミック・バス・サイジングの対象になります。

16ビット固定バス・サイジング中は，16ビット・ダイナミック・バス・サイジングを要求しないでください。要求した場合の動作は不定です。したがって，SBUSSIZをアクティブにする場合は，DBUSSIZを8ビットに指定（ロウ・レベル）してください。

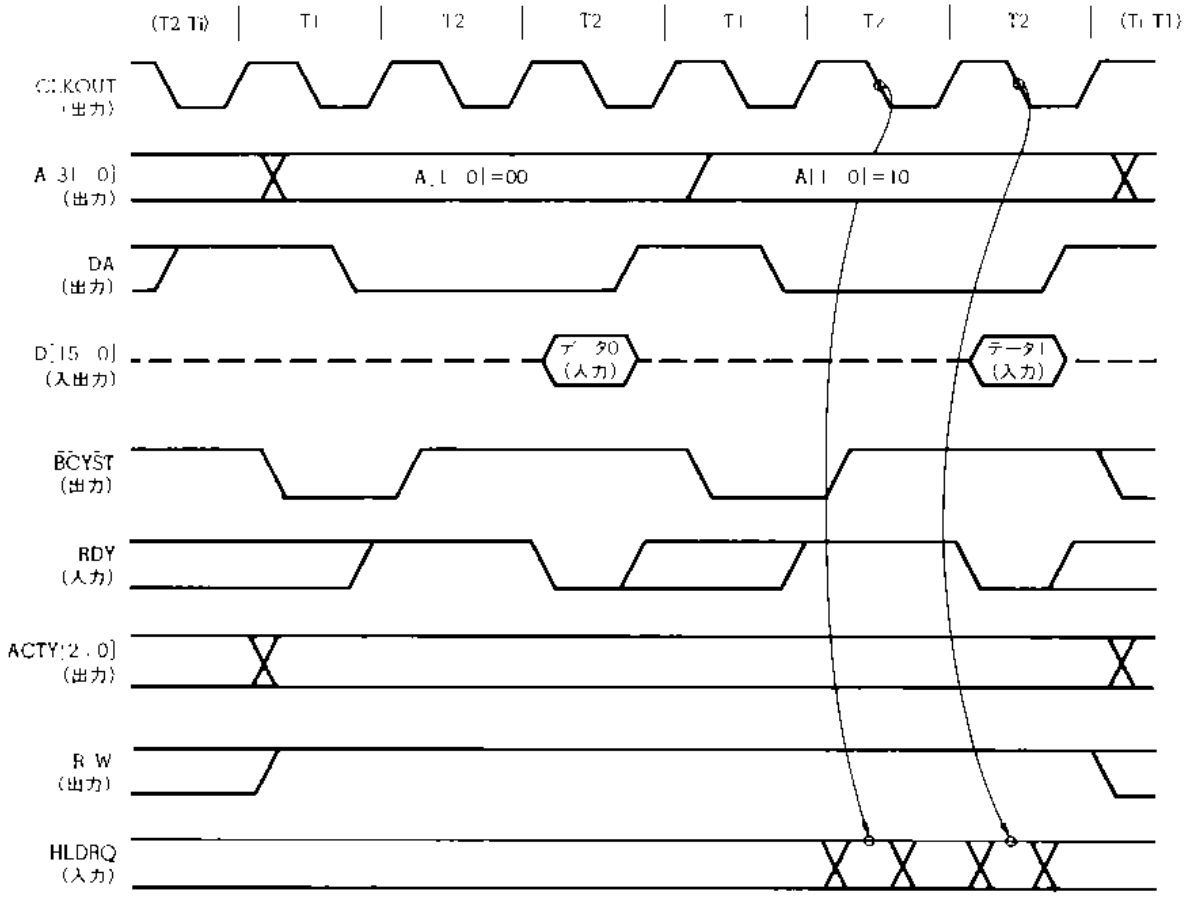
16ビット固定バス・サイジングでは，リード・サイクルでデータ・バスの上位16ビットは駆動する必要

保守／廃止

はありません。ライト・サイクルでは，データ・バス上に無効なバイト・データが存在していても，データ・バスの下位16ビットすべてを駆動します。また，ライト・バス・サイクルでは，データ・バスの上位16ビットはハイ・インピーダンスとなっています。

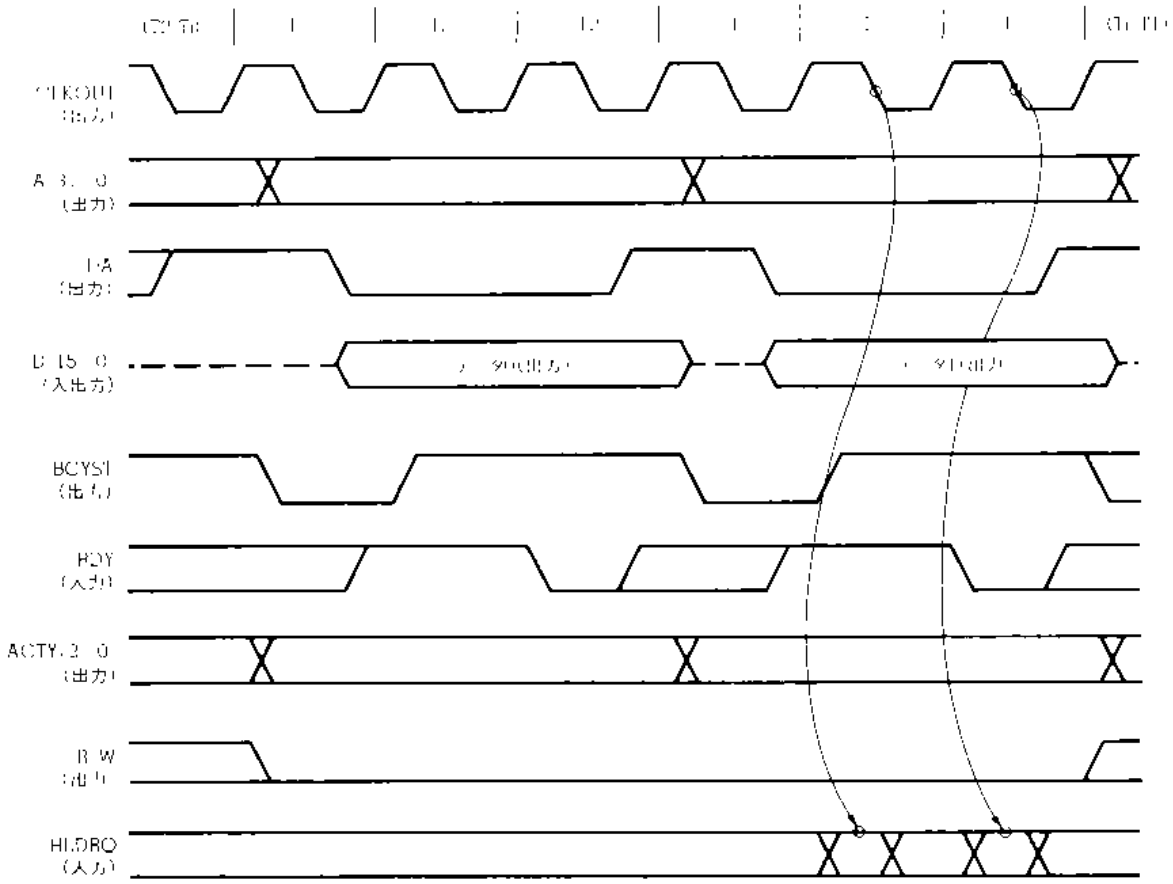
また，2回のバス・アクセスを行う場合は，2回目のバス・アクセス時にだけバス・ホールド要求を受け付けます。

図4-12　16ビット固定バス・サイジング（リード・サイクル）



備考　破線はハイ・インピーダンスを示します。

図4－13　16ビット固定バス・サイジング（3バイト・ワード・ライト）



備考　破線はハイ・インピーダンスを示します．

保守／廃止

4.8　バス・サイジングとエンディアン

4.8.1　ビッグ・エンディアンとリトル・エンディアン

V$_R$3800では32ビット・データをワード，16ビット・データをハーフ・ワード，8ビット・データをバイトと呼びます．それらのバイト順序はリセット時のモード選択（BIGEND端子）でビッグ・エンディアンかリトル・エンディアンに指定できます．

V$_R$3800ではバイト・アドレシングを採用しています．つまり，メモリ内の1バイトごとにアドレスが付けられているメモリ・システムを想定しています．メモリ内の1ワードを例にすると，ビッグ・エンディアンのシステムでは，下位アドレスにあるバイト・データほど上位（MSB側）に配列します．逆にリトル・エンディアンのシステムでは，下位アドレスにあるバイト・データほど下位（LSB側）に配列します．

図4－14に，ビック・エンディアンにおける複数ワード内でのバイトの並びを示します．ビッグ・エンディアンでは，ワード・アドレス内の最上位バイトが最下位アドレスにあります．1ワードは最上位バイトのアドレスで参照されます．

図4－14　ビッグ・エンディアンでのワード内のバイト・アドレス

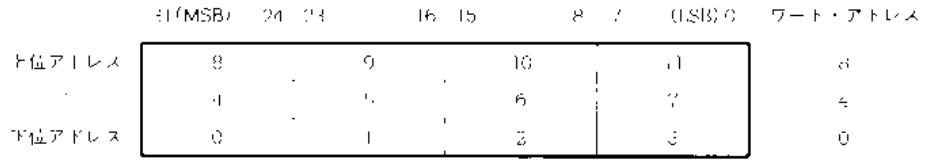
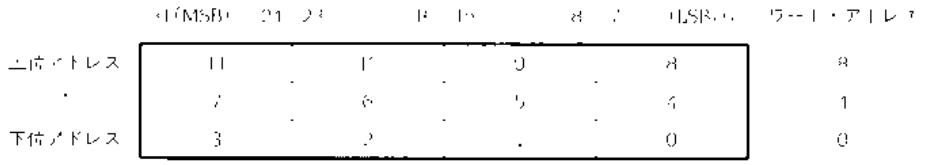


図4－15に，リトル・エンディアンにおける複数ワード内でのバイトの並びを示します．リトル・エンディアンでは，ワード・アドレス内の最下位バイトが最下位アドレスにあります．1ワードは最下位バイトのアドレスで参照されます．

図4－15　リトル・エンディアンでのワード内のバイト・アドレス



V$_R$3800では，ハーフ・ワード・データとワード・データをバイト・アドレスで参照しますが，これには次のような位置合わせの制限があります．

●ハーフ・ワード・データ：ハーフ・ワード境界（偶数アドレス）

●ワード・データ：ワード境界（4の倍数のアドレス）

この制限に反するデータを参照しようとすると，アドレス・エラー例外を発生します。

ただしV_R3800ではV_R3000Aと同様に，ワード境界に位置合わせされていないワード・データを参照するために，次の4つの特殊な命令を用意しています。

(1) LWL (Load Word Left)

(2) LWR (Load Word Right)

(3) SWL (Store Word Left)

(4) SWR (Store Word Right)

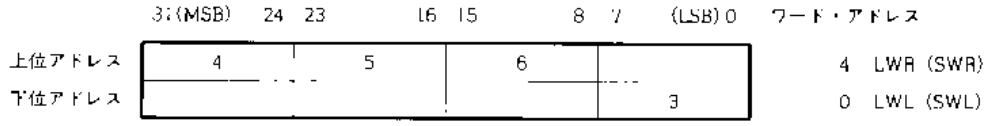
これらの命令は，通常(1)と(2)，および(3)と(4)の組で使用され，ワード境界に位置合わせされていないワード・データを参照します。

図4－16に，ビッグ・エンディアンとリトル・エンディアンの両方において，3番地から始まるワード・データを参照する場合のバイト・アドレスを示します。

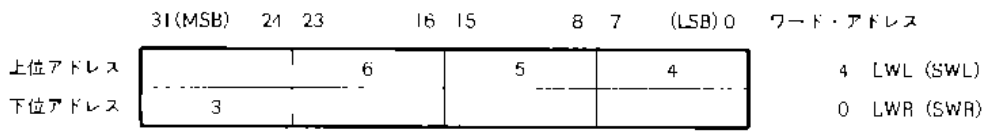
たとえば，ビッグ・エンディアンの場合，3番地から始まるワード・データをロードするために，同じレジスタに対してLWL命令で3番地から，LWR命令で6番地からロードします。一般にN番地から始まるワード・データをロードするためにはLWL命令でN番地，LWR命令で（N＋3）番地からロードします（リトル・エンディアンではLWLとLWRが逆になります）。

図4－16　ワード境界に位置合わせしていないワードのバイト・アドレス

(a) ビッグ・エンディアン



(b) リトル・エンディアン



保守／廃止

4.8.2　データ・バス上のバイト・データの整列

V_R3800は，命令やデータをワード（4バイト）単位で扱います。V_R3800がメモリ・リード・リクエストを発行した場合，メモリ・システムでは，リセット時にモード選択したバイト順序（ビッグ・エンディアンかリトル・エンディアン）に従って，指定したアドレスに存在する4バイトのデータを次のようにデータ・バス上に整列してください。

バイト順序	下位アドレスのバイト・データ	上位アドレスのバイト・データ
ビッグ・エンディアン	データ・バスの上位	データ・バスの下位
リトル・エンディアン	データ・バスの下位	データ・バスの上位

V_R3800が部分ワードのデータ（1-3バイトのデータ）を読み出す場合，そのデータが上に示したバイト順序で，データ・バス上のワード・データ内に整列されているものとみなして，データ・バスの内容を読み出します。

V_R3800がデータを書き込む場合は，リード・データと同じ位置にデータを整列してデータ・バスを駆動します。

表4－2に，データ・バスに対するデータの整列位置と，アドレスおよびデータ長の関係を示します。

ライト・アクセス時には，表中の有効バイトに対応するメモリに書き込みを行ってください。

リード・アクセス時には，アドレスがバス幅の単位でアラインされます。つまり，32ビット・バスならアドレス・バスの下位2ビットは"00"，16ビット・バス・サイジングならアドレス・バスの下位1ビットは"0"となります。

表4－2の有効バイト，無効バイトに関係なく，バス幅（32ビット・バス：D［31：0］，16ビット・バス・サイジング：D［15：0］）に対応したメモリすべてに対して読み出しを行ってください。

部分ワード・リード・アクセスの場合には，プロセッサ内部で必要なバイトの選択を行います，ただしI/Oの接続などの場合で，バス幅のすべてを使用していない場合には，接続されているデバイスだけをアクセスしてください。

ステータス・レジスタのREビット（ビット25）を1にしておくと，ステータス・レジスタのKUcビット（ビット1）が1のときに，データ長に応じて，参照するアドレスの下位2ビットを次のように反転してデータを扱います。

ハーフ・ワード参照：アドレスのビット1を反転
バイト参照：アドレスのビット1とビット0を反転
3バイト参照：アドレスのビット1とビット0を反転
ワード参照：アドレス非反転（参照するデータのエンディアンがリセット時のモード選択と反転したかのようにデータ処理）

保守／廃止

表4－2　データ・バス上のデータの整列位置とアドレスの関係

(a) ビッグ・エンディアン

	アクセス・タイプ（データ長）	アドレスの下位2ビット A1	A0	R/W	データ・バス（数値は対応するバイト・アドレスの下位2ビット） 31-24	23-16	15-8	7-0
	ワード	0	0	R/W	00	01	10	11
		0	1	W	00	01	10	11
		1	0	W	00	01	10	11
		1	1	W	00	01	10	11
★	3バイト	0	0	R	00	01	10	11
★		0	0	W	00	01	10	無効
		0	1	W	無効	01	10	11
★	ハーフ・ワード	0	0	R	00	01	10	11
★		0	0	W	00	01	無効	無効
		1	0	W	無効	無効	10	11
★	バイト	0	0	R	00	01	10	11
★		0	0	W	00	無効	無効	無効
		0	1	W	無効	01	無効	無効
		1	0	W	無効	無効	10	無効
		1	1	W	無効	無効	無効	11

備考　□　ストア・パーシャルを許可にしたときだけ発生します。

保守／廃止

(b) リトル・エンディアン

アクセス・タイプ（データ長）	アドレスの下位2ビット A1	A0	R/W	データ・バス（数値は対応するバイト・アドレスの下位2ビット） 31-24	23 16	15-8	7-0	
ワード	0	0	R/W	11	10	01	00	
	0	1	W	11	10	01	00	
	1	0	W	11	10	01	00	
	1	1	W	11	10	01	00	
3バイト	0	0	R	11	10	01	00	★
	0	0	W	無効	10	01	00	★
	0	1	W	11	10	01	無効	
ハーフ・ワード	0	0	R	11	10	01	00	★
	0	0	W	無効	無効	01	00	★
	1	0	W	11	10	無効	無効	
バイト	0	0	R	11	10	01	00	★
	0	0	W	無効	無効	無効	00	★
	0	1	W	無効	無効	01	無効	
	1	0	W	無効	10	無効	無効	
	1	1	W	11	無効	無効	無効	

備考 ★　：ストア・パーシャルを許可にしたときだけ発生します．

4.8.3　バス・サイジング時のデータ・バス上のバイト・データの整列

V_R3800は次のバス・サイジングをサポートしています。

・８ビット／16ビット・ダイナミック・バス・サイジング
・16ビットの固定バス・サイジング

データ・バス上のデータの整列位置とアドレスの関係について，８ビットのバス・サイジングの場合を表４－3に，16ビットのバス・サイジングの場合を表４－4に示します」

(1) ８ビット・バス・サイジングの場合

エンディアンに関係なく，アドレス・バスに出力するアドレスのバイト・データを，データ・バスの最下位バイトに整列します。

(2) 16ビット・バス・サイジングの場合

リセット時に指定したエンディアンに従って，アドレス・バスに出力するアドレスで示す２バイトのデータを，データ・バスの下位２バイトに整列します。

(a) リード・サイクル

V_R3800は偶数アドレスを出力しますので，そのアドレスから始まる２バイトのデータをデータ・バス上に整列します。データ・バス上の２バイトのデータすべてをV_R3800が要求しているわけではありません（１バイトしか必要でない場合もある）が，外部のメモリ・システムはそれを認識できません。また，認識する必要はありません。

(b) ライト・サイクル

外部のメモリ・システムが，データ・バス上のデータのうち，どこが有効なデータかを認識するようにしてください。そうしなければ，無効なデータをメモリに書き込むことがあります。有効データの認識は，アドレス・バスの下位１ビット，アクセス・タイプ，リセット時に設定したエンディアンによって行います。

V_R3800は，奇数アドレスを出力する場合と偶数アドレスを出力する場合があります」

・奇数アドレスを出力する場合

データ・バス上の上位，または下位の１バイトのデータだけが有効になります。上位が有効か，下位が有効かはエンディアンに依存します。

アクセス・タイプが示しているデータ長には依存しません。アドレスの下位２ビットが"01"で，アクセス・タイプが３バイトのライト・アクセスでも，１バイトのデータだけが有効です。

・偶数アドレスを出力する場合

データ・バス上の2バイトのデータすべてが有効である場合と，1バイトのデータだけが有効な場合があります。この場合は，アクセス・タイプに応じて，データ・バス上にあるデータのどこが有効かを認識できます。

ただし，アクセス・タイプが3バイトのライト・サイクルで，アドレスの下位2ビットが"10"のときは，データ・バス上の有効データが1バイト有効なのか2バイト有効なのかを識別できません（バス・サイクルが1回目か2回目かを認識すれば識別可能）。その識別手段を提供するため，アドレスの下位2ビットが"10"のライト・サイクルで，データ・バスの下位16ビットに無効なバイト・データが存在するような場合が生じると，アクセス・タイプがバイトのバス・サイクルを起動するようになっています。このため，結果としてアクセス・タイプが3バイトで，かつアドレスの下位2ビットが"10"の場合は，データ・バス上の有効データを示すパターンは一通りだけです。この場合は，データ・バス上の16ビットのデータすべてが有効です。

表4－3　8ビット・バス・サイジング時のデータ・バス上の整列位置

アクセス・タイプ（データ長）	アドレスの下位2ビット		データ・バス（数値は対応するバイト・アドレスの下位2ビット）			
	A1	A0	31-24	23-16	15-8	7-0
ワード	0	0	無効	無効	無効	00
	0	1	無効	無効	無効	01
	1	0	無効	無効	無効	10
	1	1	無効	無効	無効	11
3バイト	0	0	無効	無効	無効	00
	0	1	無効	無効	無効	01
	1	0	無効	無効	無効	10
	1	1	無効	無効	無効	11
ハーフ・ワード	0	0	無効	無効	無効	00
	0	1	無効	無効	無効	01
	1	0	無効	無効	無効	10
	1	1	無効	無効	無効	11
バイト	0	0	無効	無効	無効	00
	0	1	無効	無効	無効	01
	1	0	無効	無効	無効	10
	1	1	無効	無効	無効	11

保守／廃止

表4－4　16ビット・バス・サイジング時のデータ・バス上の整列位置（1/2）

(a) ビッグ・エンディアン

アクセス・タイプ（データ長）	アドレスの下位2ビット A1	A0	R/W	データ・バス（数値は対応するバイト・アドレスの下位2ビット） 31-24	23-16	15-8	7 0
ワード	0	0	R/W	無効	無効	00	01
	0	1	W	無効	無効	00	01
	1	0	R/W	無効	無効	10	11
	1	1	W	無効	無効	10	11
3バイト	0	0	R/W	無効	無効	00	01
	0	1	W	無効	無効	無効	01
	1	0	R/W	無効	無効	10	11
ハーフ・ワード	0	0	R/W	無効	無効	00	01
★	1	0	R/W	無効	無効	10	11
★ バイト	0	0	R	無効	無効	00	01
★	0	0	W	無効	無効	00	無効
	0	1	W	無効	無効	無効	01
★	1	0	R	無効	無効	10	11
★	1	0	W	無効	無効	10	無効
	1	1	W	無効	無効	無効	11

備考　[　]：ストア・パーシャルを許可にしたときだけ発生します。

保守／廃止

表4-4　16ビット・バス・サイジング時のデータ・バス上の整列位置（2/2）

(b) リトル・エンディアン

アクセス・タイプ（データ長）	アドレスの下位2ビット A1	A0	R/W	データ・バス（数値は対応するバイト・アドレスの下位2ビット） 31-24	23-16	15-8	7 0	
ワード	0	0	R/W	無効	無効	01	00	
	0	1	W	無効	無効	01	00	
	1	0	R/W	無効	無効	11	10	
	1	1	W	無効	無効	11	10	
3バイト	0	0	R/W	無効	無効	01	00	
	0	1	W	無効	無効	01	無効	
	1	0	R/W	無効	無効	11	10	
ハーフ・ワード	0	0	R/W	無効	無効	01	00	
	1	0	R/W	無効	無効	11	10	★
バイト	0	0	R	無効	無効	01	00	★
	0	0	W	無効	無効	無効	00	★
	0	1	W	無効	無効	01	無効	
	1	0	R	無効	無効	11	10	★
	1	0	W	無効	無効	無効	10	★
	1	1	W	無効	無効	11	無効	

備考　|　　| ストア・バーシャルを許可にしたときだけ発生します。

〔メ　モ〕

第５章　ソフトウエアの互換性

V_R3800は，米国MIPS社のRISCアーキテクチャを採用したV_R3000AをCPUコアとし，V_R3000Aとフルコンパチブルな命令セットを提供します。このため，V_R3000A用に開発したソフトウエアは，V_R3800のシステムに容易に移植できます。

CPUの機能については，リセット時のモード設定で自動的に設定されるモードとTLB機能以外は，V_R3000Aとコンパチブルになっています。リセット時に強制的にOFFとなるマルチプロセッサ・ストールなどはサポートしていません。

5.1　V_R3000Aとの相違点

(1) V_R3800は外部コプロセッサ（CP1-CP3）をサポートしません。このため，外部コプロセッサ命令を実行した場合の動作は不定です。これを避けるために，ソフトウエアでステータス・レジスタのコプロセッサ使用の可能性を指定するフィールドを使用不可に設定しておいてください（ビット31-ビット29に０を設定する）。使用不可に設定したコプロセッサの命令をV_R3800が実行しようとすると，コプロセッサ使用不可例外（Coprocessor Unusable Exception）が発生します。ただし，システム制御コプロセッサ（CP0）は内蔵しており，カーネル・モードの場合にはステータス・レジスタで使用不可に設定しても無視されます。V_R3800は，CP0命令に関しては通常通り実行できます。

(2) V_R3000Aには４本のコプロセッサ条件端子があります。これらの端子はコプロセッサの条件分岐命令であるBCzTまたはBCzF命令（z=0-3）で参照されます。

V_R3800にはこのコプロセッサ条件端子が存在しません。外部コプロセッサをサポートしないためです。外部コプロセッサを使用不可に設定せずに，BC1T，BC1F，BC2T，BC2F，BC3T，BC3F命令を実行した場合の動作は不定です。ただしV_R3800では，BC0T命令とBC0F命令は特殊な目的で使用することができます（後述のライト動作の待ち合わせ）。

(3) V_R3000Aではリセット後，キャッシュ領域をアクセスするまでに，命令キャッシュとデータ・キャッシュをソフトウエアで無効化していました。V_R3800ではリセットで命令キャッシュとデータ・キャッシュを無効化するため，ソフトウエアでの無効化処理は不要です。逆に，リセットしてもキャッシュが以前の状態を保持するようなシステムでは注意してください。

(4) V_R3000Aは，TLBを使用した仮想記憶管理機構をサポートしています。しかし，V_R3800は組み込み制御分野用のため，TLBを削除し，仮想アドレスがそのまま物理アドレスとなるようにしています。ただし，V_R3000AのTLBマッピングなし領域（kseg0，kseg1）については，V_R3800でも同様に物理アドレスの下位領域にマッピングされます，TLBの削除による相違点を次に示します，

保守／廃止

(1)　V_R3000Aで必要だったソフトウエアによるTLBの初期化不要，

(2)　TLB命令（TLBWI，TLBWR，TLBR，TLBP命令）使用禁止（実行結果は不定），

(3)　システム制御コプロセッサ（CP0）のインデクス，ランダム，エントリLO，エントリHIレジスタへのアクセス禁止（アクセスした場合の動作は不定）

(4)　TLB変更例外，TLB不一致例外は発生しない

また，カーネル／ユーザ・モードの概念はそのままなので，ユーザ・モードでカーネル・モード専用のアドレス空間を参照した場合は，アドレス・エラー例外が発生します。

表5－1　V_R3000Aとの相違点

項　　目	V_R3800	V_R3000A
外部コプロセッサ	サポートしない	サポートする
コプロセッサ条件端子	ない	4本
命令／データ・キャッシュの無効化	リセット時に自動処理	ソフトウエアで処理
TLBによる仮想記憶管理機構	サポートしない	サポートする

5.2　V_R3800独自の機能

(1) ライト・バッファ・エンプティ待ち合わせ機能

V_R3800では，システム制御コプロセッサの条件分岐命令であるBC0T命令とBC0F命令で，BIUのライト・バッファの状態を知ることができます．ライト・バッファにデータがあるときはコプロセッサ条件が偽（0）となり，ライト・バッファにデータがないときはコプロセッサ条件が真（1）となるように設計されています。このコプロセッサ条件をBC0T命令やBC0F命令で参照することで，次のような処理ができます。

・ライト・アクセスとリード・アクセスの順序の調停

・連続するライト・サイクルにおけるライト間隔の調節

V_R3800のBIUは，CPUコアかメモリ・アクセスをしていないとき（MemRd＝1），またはラン・サイクル（RUN＝0）のときにライト・バッファの状態をCPUコアのCpCond0端子に出力しますライト・バッファの状態とCpCond0の関係を次に示します

ライト・バッファの状態	CpCond0
データあり	0
データなし	1

保守／廃止

ライト・バッファ・エンプティの待ち合わせは，次の手順で行ってください．

```
        SW    reg1, (reg2)
        NOP
        NOP
Loop :
        BCOF  Loop
        NOP
```

注意　ライト・バッファ・エンプティを確実に検出するためには，次の条件を満たしてください。

(1) SW SH SB SWR SWL命令の実行からBCOF BCOT命令の実行までには，2クロック以上の間隔をあけてください。

(2) 命令キャッシュのリフィルをブロック・リードで行い，命令ストリーミングを行う場合は，BCOF BCOT命令を（16n＋8）番地か（16n＋12）番地に配置してください。

〔メ　モ〕

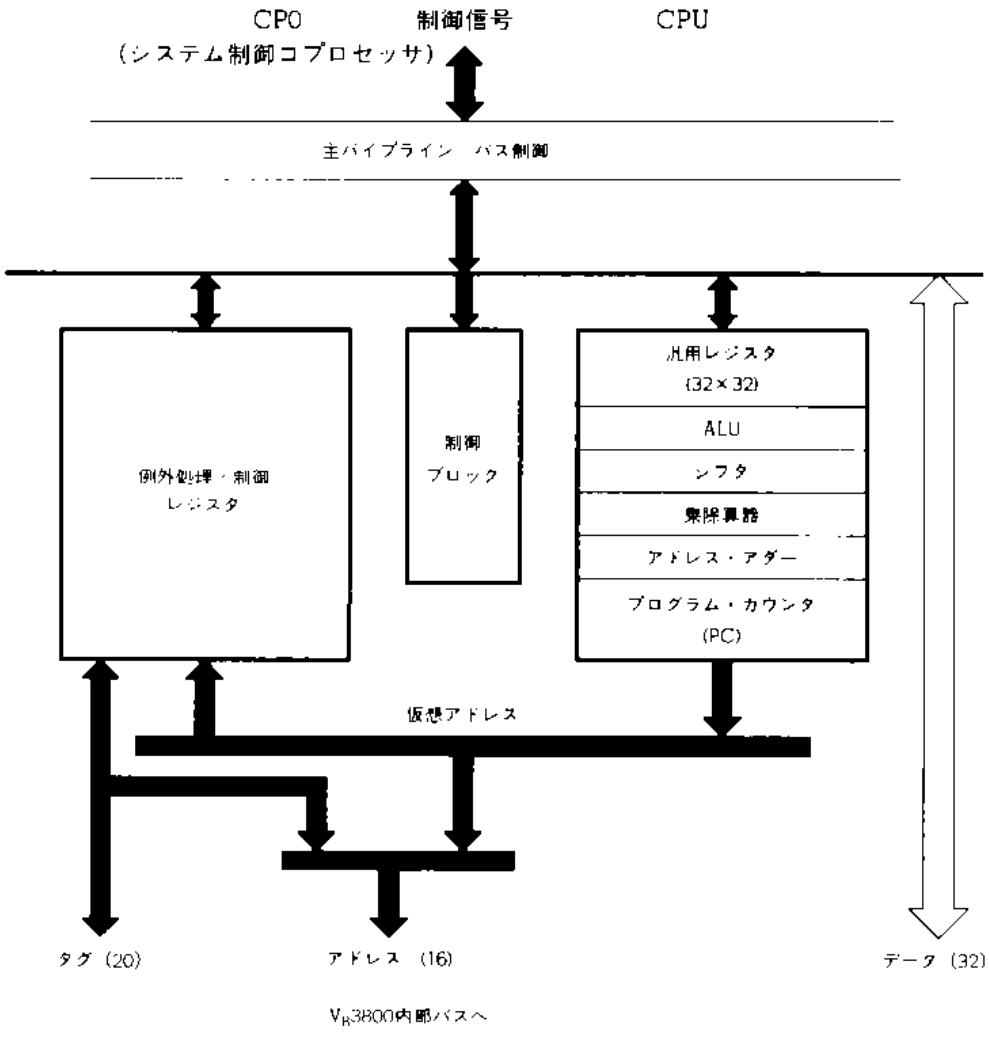
第６章　アーキテクチャ概要

6.1　CPUコア

次に，CPUコアであるV_R3000Aについて簡単に説明します。

V_R3000Aは，単一のチップに実装されている２つの密結合型プロセッサで構成されています。最初のプロセッサは，完全な32ビットRISC型CPUです」２番目のプロセッサはシステム制御コプロセッサ（CP0）で，アドレス・マッピングの管理，例外処理の制御を行います。図６－1に，V_R3000A内部ブロック図を示します。

図６－１　V_R3000A内部ブロック図



保守／廃止

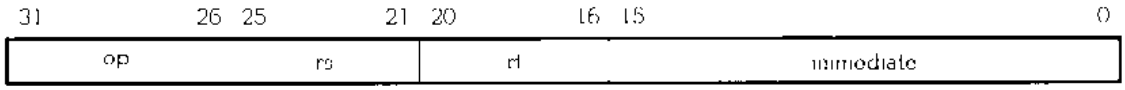
6.2　命令セット概要

V_R3800の命令は，すべて32ビットで，図6－2に示すように命令形式の基本形が3種類しかありません。そのため，命令のデコードは簡単に行われます。

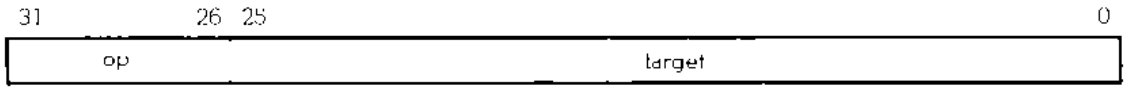
複雑な（使用頻度の低い）オペレーションおよびアドレシング・モードは，一連の簡単な命令を組み合わせて作ります。

図6－2　命令形式

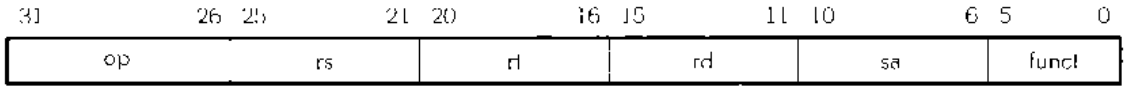
I　タイプ（イミーディエト）



J　タイプ（分岐）



R　タイプ（レジスタ）



備考　変数サブフィールドは，以下のとおりです。

op	6ビットの命令コード
rs	5ビットのソース・レジスタ指定子
rt	5ビットのターゲット（ソース／デスティネーション）・レジスタ，または分岐条件
immediate	16ビットのイミーディエト，分岐ディスプレースメント，またはアドレス・ディスプレースメント
target	26ビットの無条件分岐ターゲット・アドレス
rd	5ビットのデスティネーション・レジスタ指定子
sa	5ビットのシフト量
funct	6ビットの機能フィールド

保守／廃止

V_R3800の命令セットは，以下のクラスに分類できます。

(1) ロード／ストア命令

ロード／ストア命令は，メモリと汎用レジスタ間でデータを転送します．サポートされているアドレシング・モードは，ベース・レジスタに16ビットの符号なしイミーディエト・オフセットを加えるIタイプ命令だけです，

(2) 演算命令

演算命令は，レジスタ内の値に対し算術，論理，およびシフト演算を実行します。この命令は，Rタイプ（オペランドも結果もレジスタ）とIタイプ（1つのオペランドは16ビット・イミーディエト）です，

(3) ジャンプ／ブランチ命令

ジャンプ／ブランチ命令は，プログラムの流れを変更します．ジャンプ命令は常に，26ビット・ターゲット・アドレスとプログラム・カウンタの上位4ビットとを結合して作られるページ絶対アドレス（サブルーチン呼び出しの場合はJタイプ形式）に対して分岐するか，または32ビット・レジスタ・アドレス（制御の戻しおよびディスパッチの場合はRタイプ）に対して分岐が行われます。ブランチ命令は，プログラム・カウンタに16ビット・オフセット値を加算してターケット・アドレスを作ります．ジャンフ・アンド・リンク命令は，戻りアドレスをレジスタ31に退避します

(4) コプロセッサ命令

コプロセッサ命令は，コプロセッサに対してオペレーションを実行します．コフロセッサのロド命令およびストア命令は，Iタイプです。コプロセッサ演算命令の形式は，プロセッサごとに異なります。V_R3800では，コプロセッサ0だけ接続できます。

(5) コプロセッサ0命令

コプロセッサ0命令は，プロセッサの例外処理機能を行うために，システム制御コプロセッサ（CP0）に対してオペレーションを実行します。

(6) 特殊命令

特殊命令は，特殊レジスタと汎用レジスタ間のデータ転送，システム呼び出し，ブレークポイントなどのさまざまな作業を実行します．この命令は，常にRタイプです。

表6-1には，V_R3800プロセッサの命令セットが示されています。さらに詳しい要約は**第7章　命令セット概要**に，各命令の詳しい説明は**第11章　命令セット**に記述されています，

保守／廃止

表６－１　命令の要約

op	説　明
ロード　ストア命令	
LB	Load Byte
LBU	Load Byte Unsigned
LH	Load Halfword
LHU	Load Halfword Unsigned
LW	Load Word
LWL	Load Word Left
LWR	Load Word Right
SB	Store Byte
SH	Store Halfword
SW	Store Word
SWL	Store Word Left
SWR	Store Word Right
算術命令（ALUイミーディエト）	
ADDI	Add Immediate
ADDIU	Add Immediate Unsigned
SLTI	Set on Less Than Immediate
SLTIU	Set on Less Than Immediate Unsigned
ANDI	AND Immediate
ORI	OR Immediate
XORI	Exclusive OR Immediate
LUI	Load Upper Immediate
算術命令（３オペランド，レジスタ・タイプ）	
ADD	Add
ADDU	Add Unsigned
SUB	Subtract
SUBU	Subtract Unsigned
SLT	Set on Less Than
SLTU	Set on Less Than Unsigned
AND	AND
OR	OR
XOR	Exclusive OR
NOR	NOR
シフト命令	
SLL	Shift Left Logical
SRL	Shift Right Logical
SRA	Shift Right Arithmetic
SLLV	Shift Left Logical Variable
SRLV	Shift Right Logical Variable
SRAV	Shift Right Arithmetic Variable
特殊命令	
SYSCALL	System Call
BREAK	Break

op	説　明
乗算　除算命令	
MULT	Multiply
MULTU	Multiply Unsigned
DIV	Divide
DIVU	Divide Unsigned
MFHI	Move From HI
MTHI	Move To HI
MFLO	Move From LO
MTLO	Move To LO
ジャンプ／ブランチ命令	
J	Jump
JAL	Jump And Link
JR	Jump to Register
JALR	Jump And Link Register
BEQ	Branch on Equal
BNE	Branch on Not Equal
BLEZ	Branch on Less than or Equal to Zero
BGTZ	Branch on Greater Than Zero
BLTZ	Branch on Less Than Zero
BGEZ	Branch on Greater than or Equal to Zero
BLTZAL	Branch on Less Than Zero And Link
BGEZAL	Branch on Greater than or Equal to Zero And Link
コプロセッサ命令	
BC0T	Branch on Coprocessor 0 True
BC0F	Branch on Coprocessor 0 False
システム制御コプロセッサ（CP0）命令	
MTC0	Move To CP0
MFC0	Move From CP0
RFE	Restore From Exception

保守／廃止

6.3　プログラミング・モデル

6.3.1　汎用レジスタ

図6－3は，V_R3800CPUレジスタを示しています。汎用レジスタは32個あり，それぞれが単一ワード（32ビット）で構成されています。32個の汎用レジスタは同じように扱われますが，2つのレジスタだけは例外です。つまり，汎用レジスタr0はゼロの値に固定（ハードワイヤ）されています。汎用レジスタr31はジャンプ・アンド・リンク命令に対するリンク・レジスタです。

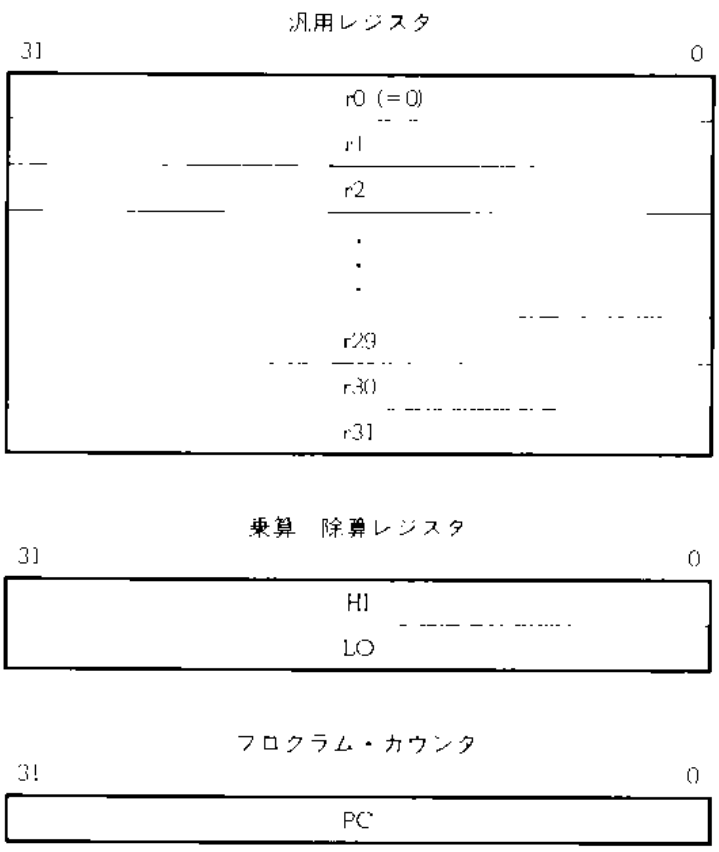
汎用レジスタr0は，演算の結果が不要の場合に命令のターゲット・レジスタとして指定できます。このレジスタを，ソース・レジスタとして使用する場合，値はゼロです。

2つの乗算／除算レジスタ（HI，LO）は，乗算演算のダブル・ワード（64ビット）の結果および除算命令の商と剰余の結果を記憶します。

注意　CPUの汎用レジスタ以外にも，システム制御コプロセッサ（CP0）にはメモリ管理システムに使用されたり，例外処理時に使用されるいくつかの特殊レジスタがあります。
メモリ管理レジスタの説明は第9章　メモリ管理システム，例外処理レジスタの説明は第10章　例外処理を参照してください。

6

図6－3　CPUレジスタ



保守／廃止

6.4　システム制御コプロセッサ（CP0）

システム制御コプロセッサ（CP0）は，V_R3800チップに実装され，V_R3800の仮想アドレシングおよび例外処理機能をサポートします。

また，V_R3000Aで定義できる外部のコプロセッサ（CP1-CP3）は，V_R3800ではサポートしていません。詳細は**5.1　V_R3000Aとの相違点**を参照してください。

図6－4　CP0のレジスタ

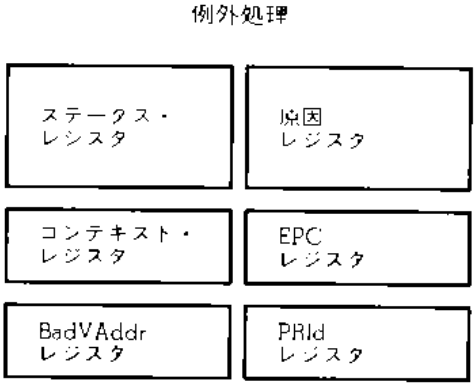


図6－4に示されているCP0レジスタは，V_R3800のメモリ管理および例外処理機能を行うために使用されます。表6－2に，各レジスタを簡単に示します。

表6－2　システム制御コプロセッサ（CP0）レジスター覧

番　号	ニモニック	説　　明
4	Context	カーネルの仮想可能ページ・テーブル配列エントリへのポインタ
8	BadVAddr	最後にエラーを起こした仮想アドレス
12	SR	モード，割り込み可能，診断状態情報
13	Cause	最後の例外の性質を示す
14	EPC	例外プログラム・カウンタ
15	PRid	プロセッサ・リビジョンID

備考　V_R3800はTLBを使用した仮想記憶管理機構をサポートしないため，V_R3000Aにあった次のレジスタは使用できません。

インデクス・レジスタ

ランダム・レジスタ

エントリLOレジスタ

エントリHIレジスタ

保守／廃止

6.5 パイプライン概要

各V_R3800命令の実行は，次の５つのステップで構成されます。

(1) IF　－　命令のフェッチ（Iキャッシュ）。
(2) RD　－　命令のデコード時に，CPUレジスタから必要なオペランドをフェッチします。
(3) ALU　－　命令オペランドに対し必要なオペレーションを実行します。
(4) MEM －　メモリにアクセスします（Dキャッシュ）。
(5) WB　－　レジスタ・ファイルに結果を書き込みます。

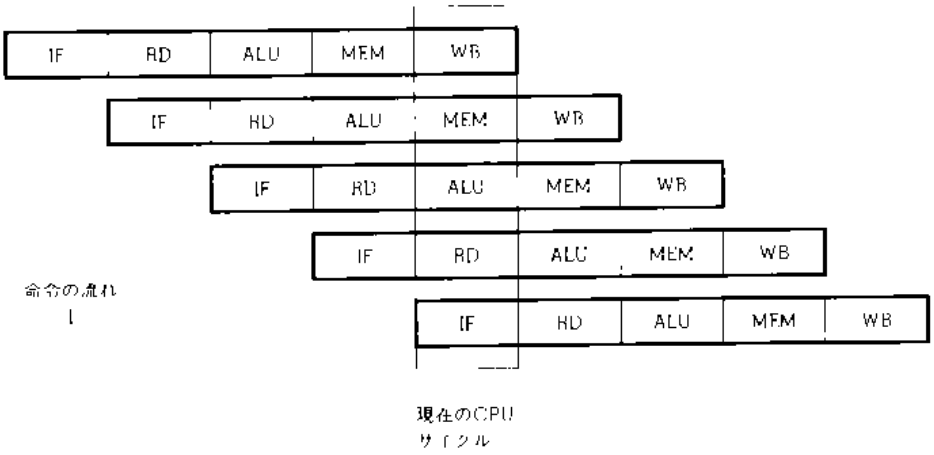
これらの各ステップに必要なサイクルは，図６-５に示されているように，ほぼ1 CPUサイクルです（一部のオペレーションは別のサイクルに重なりますが，1/2サイクルしか必要としないサイクルがあり，平均して約１サイクルです）。

図６-５　命令実行順序

保守／廃止

V_R3800は，５ステージのパイプラインを使用し，ほぼ，１命令１サイクルで命令を実行します。したがって，一度に５つの命令を実行すると，図６－６に示されているようにオーバラップします。

図６－６　命令パイプライン



このパイプラインでは，異なったCPU資源（アドレスおよびデータ・バス・アクセス，ALUオペレーション，レジスタ・アクセスなど）がお互いに干渉せずに使用されているため，効率良く動作します。命令パイプラインの詳細は，**第８章　パイプライン**を参照してください。

保守／廃止

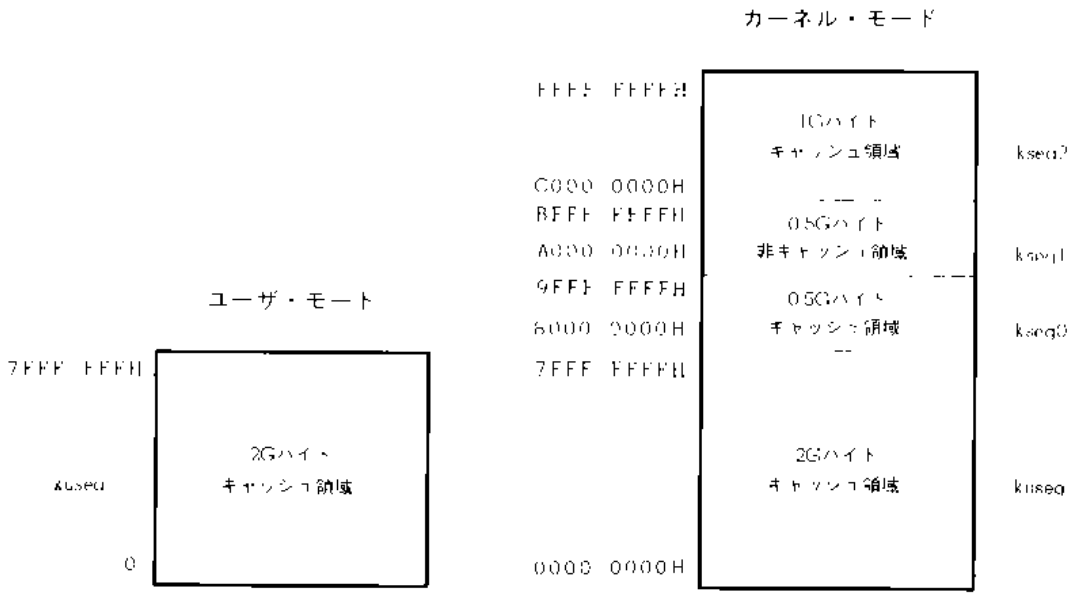
6.6　メモリ管理システム

V_R3800には，プログラム・カウンタが管理する仮想アドレス空間が4 Gバイトあります。ただし，TLBによるアドレス変換機構を持っていないため，ほとんどの空間には仮想アドレスがそのまま物理アドレスとして割り付けられます，ただしアドレス空間の一部は，物理空間の下位アドレスから割り付けられます。なお，キャッシュ・メモリを使用するかどうかもアドレス空間で区別します，

また，カーネルとユーザの2レベルの動作モードがあり，ユーザがカーネル空間を操作できないようにするレベル・プロテクション機構も備えています，詳細は，**第9章　メモリ管理システム**を参照してください。

図6-7に，2種類の動作モードに関する仮想アドレス空間を示します，

図6-7　仮想アドレシング



備考　ksegはカーネル・セグメント，kusegはカーネル・ユーザ・セグメントです，

保守／廃止

〔メ　モ〕

第7章　命令セット概要

本章では，命令を種類ごとに表にし，V_R3800の命令セットの概要を説明します。各命令の詳細については，**第11章　命令セット**を参照してください。

7.1　命令形式

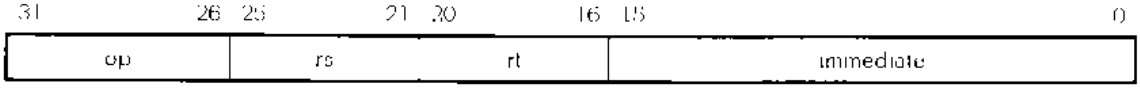
V_R3800は，V_R3000Aと下位互換の命令セットを提供します。FPU命令とTLB命令はサポートしていません。

V_R3800のすべての命令は，ワード境界に位置合わせされた単一ワード（32ビット）で構成されます。命令形式の基本形は，図7－1に示されているように3種類しかありません。したがって，命令デコードを高速で行えます。複雑な（使用頻度の低い）オペレーションおよびアドレシングは，命令を組み合わせて作ります。

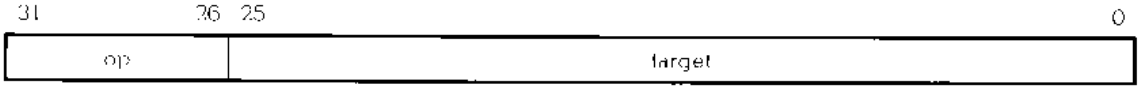
保守／廃止

図7－1　命令形式

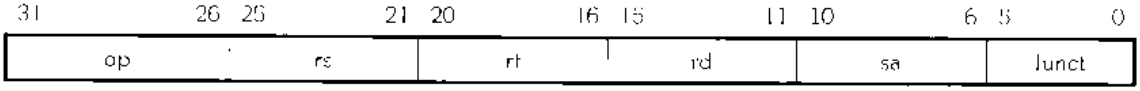
I タイプ（イミーディエト）



J タイプ（分岐）



R タイプ（レジスタ）



備考　変数サブフィールドは，以下のとおりです。

op	6ビットの命令コード
rs	5ビットのソース・レジスタ指定子
rt	5ビットのターゲット（ソース／デスティネーション）・レジスタ，または分岐条件
immediate	16ビットのイミーディエト，分岐ディスプレースメント，またはアドレス・ディスプレースメント
target	26ビットの無条件分岐ターゲット・アドレス
rd	5ビットのデスティネーション・レジスタ指定子
sa	5ビットのシフト量
funct	6ビットの機能フィールド

保守／廃止

7.2　命令の表記法

本書では，命令形式における変数サブフィールド（rs，rt，immediateなど）は，すべて小文字名で示します。分かりやすくするために，特別な命令の形式における変数サブフィールドに，別名を使用しています。たとえば，ロード命令とストア命令では，rs＝baseが使用されています。このような別名は変数サブフィールドなので，常に小文字で記述しています。命令のオペレーション・コードは，すべて大文字です。

すべてのニモニックの実際のビット符号化は，**付録　命令コード表**に示します。

7.3　ロード／ストア命令

ロード／ストア命令は，メモリと汎用レジスタとの間でデータを転送します。これらの命令はすべてIタイプ命令で，16ビットのイミーディエトを32ビットに符号拡張し，rs（＝base）で指定されたベース・レジスタの内容を加算してアドレスを生成し，そのアドレスに対してロード命令またはストア命令を行います。

ロード・オペレーションには，常に1命令分の遅延（待ち時間）があります。そのためロード命令直後の命令は，メモリからレジスタにロードされたデータを使用できません，そのデータを使用できるのは，ロード命令から2番目の命令です。ただし，LWL命令とLWR命令のターゲット・レジスタは例外で，このレジスタは直前のロード命令のデスティネーションとして使用されるレジスタを指定できます（ロード命令の遅延については，**第8章　パイプライン**を参照してください）。

表7－1はロード命令を，表7－2はストア命令を要約したものです。

表7－1　ロード命令の要約

命　令	形式および説明	op	base	rt	offset
Load Byte	LB　rt, offset (base) 16ビットのoffsetをレジスタbaseの内容に加算してアドレスを生成します。アドレス指定されたバイトの内容を符号拡張して汎用レジスタrtにロードします。				
Load Byte Unsigned	LBU　rt, offset (base) 16ビットのoffsetをレジスタbaseの内容に加算してアドレスを生成します。アドレス指定されたバイトの内容をゼロ拡張して汎用レジスタrtにロードします。				
Load Halfword	LH　rt, offset (base) 16ビットのoffsetをレジスタbaseの内容に加算してアドレスを生成します。アドレス指定されたハーフ・ワードの内容を符号拡張して汎用レジスタrtにロードします。				
Load Halfword Unsigned	LHU　rt, offset (base) 16ビットのoffsetをレジスタbaseの内容に加算してアドレスを生成します。アドレス指定されたハーフ・ワードの内容をゼロ拡張して汎用レジスタrtにロードします。				
Load Word	LW　rt, offset (base) 16ビットのoffsetをレジスタbaseの内容に加算してアドレスを生成します。アドレス指定されたワードの内容を汎用レジスタrtにロードします。				
Load Word Left	LWL　rt, offset (base) 16ビットのoffsetをレジスタbaseの内容に加算してアドレスを生成します。アドレス指定されたバイトがワードの最左端バイトになるように，アドレス指定されたワードを左にシフトします。シフトした結果と，汎用レジスタrtの内容とをマージして汎用レジスタrtにロードします。				
Load Word Right	LWR　rt, offset (base) 16ビットのoffsetをレジスタbaseの内容に加算してアドレスを生成します。アドレス指定されたバイトがワードの最右端バイトになるように，アドレス指定されたワードを右にシフトします。シフトした結果と，汎用レジスタrtの内容とをマージして汎用レジスタrtにロードします。				

表7-2　ストア命令の要約

命　令	形式および説明	op	base	rt	offset
Store Byte	SB　rt, offset (base) 16ビットのoffsetをレジスタbaseの内容に加算してアドレスを生成します。汎用レジスタrtの最下位バイトの内容を，アドレス指定されたメモリ位置にストアします。				
Store Halfword	SH　rt, offset (base) 16ビットのoffsetをレジスタbaseの内容に加算してアドレスを生成します。汎用レジスタrtの最下位ハーフ・ワードの内容を，アドレス指定されたメモリ位置にストアします。				
Store Word	SW　rt, offset (base) 16ビットのoffsetをレジスタbaseの内容に加算してアドレスを生成します。汎用レジスタrtの最下位ワードの内容を，アドレス指定されたメモリ位置にストアします。				
Store Word Left	SWL　rt, offset (base) 16ビットのoffsetをレジスタbaseの内容に加算してアドレスを生成します。ワードの最左端バイトがアドレス指定されたバイトの位置になるように，汎用レジスタrtの内容を左にシフトします。元のデータが記憶されている複数バイトを，アドレス指定されたバイト位置に対応する複数バイトにストアします。				
Store Word Right	SWR　rt, offset (base) 16ビットのoffsetをレジスタbaseの内容に加算してアドレスを生成します。ワードの最左端バイトがアドレス指定されたバイトの位置になるように，汎用レジスタrtの内容を右にシフトします。元のデータが記憶されている複数バイトを，アドレス指定されたバイト位置に対応する複数バイトにストアします。				

保守／廃止

7.4　演算命令

演算命令は、レジスタ内の値に対し算術，論理，およびシフト演算を実行します。演算命令は，Rタイプ（オペランドすべてレジスタ）とIタイプ（1つのオペランドは16ビット・イミーディエト）形式があります。演算命令は，以下の4種類に分類できます。

・ALUイミーディエト命令は，表7-3に要約されています。
・3オペランド・レジスタ・タイプ命令は，表7-4に要約されています。
・シフト命令は，表7-5に要約されています。
・乗算／除算命令は，表7-6に要約されています。

保守／廃止

表7-3　ALUイミーディエト命令の要約

命　令	形式および説明	op	rs	rt	immediate
ADD Immediate	ADDI　rt, rs, immediate 16ビットのimmediateを32ビットに符号拡張を行い，汎用レジスタrsと加算し，32ビットの結果を汎用レジスタrtに格納します。2の補数のオーバフローが発生すると，例外が発生します。				
ADD Immediate Unsigned	ADDIU　rt, rs, immediate 16ビットのimmediateを32ビットに符号拡張を行い，汎用レジスタrsに加算し，32ビットの結果を汎用レジスタrtに格納します。オーバフローが発生しても，例外は発生しません。				
Set on Less Than Immediate	SLTI　rt, rs, immediate 16ビットのimmediateを32ビットに符号拡張を行い，符号付き32ビット整数として汎用レジスタrsと比較します。汎用レジスタrsがimmediateより小さい場合は1を，そうでない場合は0を汎用レジスタrtに格納します。				
Set on Less Than Unsigned Immediate	SLTIU　rt, rs, immediate 16ビットのimmediateを32ビットに符号拡張を行い，符号なし32ビット整数として汎用レジスタrsと比較します。汎用レジスタrsがimmediateより小さい場合は1を，そうでない場合は0を汎用レジスタrtに格納します。				
AND Immediate	ANDI　rt, rs, immediate 16ビットのimmediateをゼロ拡張し，汎用レジスタrsの内容とANDを実行し，結果を汎用レジスタrtに格納します。				
OR Immediate	ORI　rt, rs, immediate 16ビットのimmediateをゼロ拡張し，汎用レジスタrsの内容とORを実行し，結果を汎用レジスタrtに格納します。				
Exclusive OR Immediate	XORI　rt, rs, immediate 16ビットのimmediateをゼロ拡張し，汎用レジスタrsの内容とexclusive　ORを実行し，結果を汎用レジスタrtに格納します。				
Load Upper Immediate	LUI　rt, immediate 16ビットのimmediateを左に16ビット・シフトします。ワードの下位16ビットにゼロをセットします。結果を汎用レジスタrtに格納します。				

7

表7-4　3オペランド・レジスタ・タイプ命令の要約

命　令	形式および説明	op	rs	rt	rd	0	funct
ADD	ADD　rd, rs, rt 汎用レジスタrsと汎用レジスタrtの内容を加算し，32ビットの結果を汎用レジスタrdに格納します。2の補数のオーバフローが発生すると，例外が発生します。						
ADD Unsigned	ADDU　rd, rs, rt 汎用レジスタrsと汎用レジスタrtの内容を加算し，32ビットの結果を汎用レジスタrdに格納します。オーバフローが発生しても，例外は発生しません。						
Subtract	SUB　rd, rs, rt 汎用レジスタrsから汎用レジスタrtの内容を減算し，32ビットの結果を汎用レジスタrdに格納します。2の補数のオーバフローが発生すると，例外が発生します。						
Subtract Unsigned	SUBU　rd, rs, rt 汎用レジスタrsから汎用レジスタrtの内容を減算し，32ビットの結果を汎用レジスタrdに格納します。オーバフローが発生しても，例外は発生しません。						
Set on Less Than	SLT　rd, rs, rt 汎用レジスタrtの内容と汎用レジスタrsとを比較します（符号付き32ビット整数として）。汎用レジスタrsが汎用レジスタrtより小さい場合は1，そうでない場合は0を汎用レジスタrdに格納します。						
Set on Less Than Unsigned	SLTU　rd, rs, rt 汎用レジスタrtの内容と汎用レジスタrsとを比較します（符号なし32ビット整数として）。汎用レジスタrsが汎用レジスタrtより小さい場合は1，そうでない場合は0を汎用レジスタrdに格納します。						
AND	AND　rd, rs, rt 汎用レジスタrsと汎用レジスタrtのビット単位にANDを実行し，結果を汎用レジスタrdに格納します。						
OR	OR　rd, rs, rt 汎用レジスタrsと汎用レジスタrtのビット単位にORを実行し，結果を汎用レジスタrdに格納します。						
Exclusive OR	XOR　rd, rs, rt 汎用レジスタrsと汎用レジスタrtのビット単位にExclusive ORを実行し，結果を汎用レジスタrdに格納します。						
NOR	NOR　rd, rs, rt 汎用レジスタrsと汎用レジスタrtのビット単位にNORを実行し，結果を汎用レジスタrdに格納します。						

保守／廃止

表7－5　シフト命令の要約

(a) SLL, SRL, SRA

命　令	形式および説明	op	0	rt	rd	sa	funct
Shift Left Logical	SLL　rd, rt, sa 汎用レジスタrtの内容をsaビット左にシフトし，下位ビットにゼロを挿入します。32ビットの結果を汎用レジスタrdに格納します。						
Shift Right Logical	SRL　rd, rt, sa 汎用レジスタrtの内容をsaビット右にシフトし，上位ビットにゼロを挿入します。32ビットの結果を汎用レジスタrdに格納します。						
Shift Right Arithmetic	SRA　rd, rt, sa 汎用レジスタrtの内容をsaビット右にシフトし，上位ビットを符号拡張します。32ビットの結果を汎用レジスタrdに格納します。						

(b) SLLV, SRLV, SRAV

命　令	形式および説明	op	rs	rt	rd	0	funct
Shift Left Logical Variable	SLLV　rd, rt, rs 汎用レジスタrtの内容を左にシフトします。汎用レジスタrsの下位5ビットは，シフトするビット数を指定します。汎用レジスタrtの下位ビットにゼロを挿入し，32ビットの結果を汎用レジスタrdに格納します。						
Shift Right Logical Variable	SRLV　rd, rt, rs 汎用レジスタrtの内容を右にシフトします。汎用レジスタrsの下位5ビットは，シフトするビット数を指定します。汎用レジスタrtの上位ビットにゼロを挿入し，32ビットの結果を汎用レジスタrdに格納します。						
Shift Right Arithmetic Variable	SRAV　rd, rt, rs 汎用レジスタrtの内容を右にシフトします。汎用レジスタrsの下位5ビットは，シフトするビット数を指定します。汎用レジスタrtの上位ビットを符号拡張し，32ビットの結果を汎用レジスタrdに格納します。						

保守／廃止

表7－6　乗算／除算命令の要約

(a) MULT, MULTU, DIV, DIVU

命　令	形式および説明	op	rs	rt	0	funct
Multiply	MULT　rs, rt 汎用レジスタrsと汎用レジスタrtの内容を2の補数値として乗算します。64ビットの結果を，特殊レジスタHI/LOに格納します。					
Multiply Unsigned	MULTU　rs, rt 汎用レジスタrsと汎用レジスタrtの内容を符号なし値として乗算します。64ビットの結果を，特殊レジスタHI/LOに格納します。					
Divide	DIV　rs, rt 汎用レジスタrsの内容を汎用レジスタrtで除算します。オペランドを2の補数値として扱います。32ビットの商を特殊レジスタLOに，32ビットの剰余をHIに格納します。					
Divide Unsigned	DIVU　rs, rt 汎用レジスタrsの内容を汎用レジスタrtで除算します。オペランドを符号なし値として扱います。32ビットの商を特殊レジスタLOに，32ビットの剰余をHIに格納します。					

(b) MFHI, MFLO

命　令	形式および説明	op	0	rd	0	funct
Move From HI	MFHI　rd 特殊レジスタHIの内容を汎用レジスタrdに転送します。					
Move From LO	MFLO　rd 特殊レジスタLOの内容を汎用レジスタrdに転送します。					

(c) MTHI, MTLO

命　令	形式および説明	op	rs	0	funct
Move To HI	MTHI　rs 汎用レジスタrsの内容を特殊レジスタHIに転送します。				
Move To Lo	MTLO　rs 汎用レジスタrsの内容を特殊レジスタLOに転送します。				

保守／廃止

7.5　ジャンプ／ブランチ命令

ジャンプ命令およびブランチ命令は，プログラムの流れを変更します　ジャンプ命令（およびブランチ命令）は，すべて１命令の遅延が発生します。つまり，ジャンプ命令（またはブランチ命令）の直後の命令は，ジャンプ命令（またはブランチ命令）のターゲット先の命令がメモリから取り出されるまでの間に実行されます，ジャンプ命令（およびブランチ命令）の遅延に関する詳細な説明は，**8.2　遅延スロット**を参照してください。

サブルーチンの呼び出しなどに利用されるジャンプ命令とジャンプ・アンド・リンク命令は，両方ともＪタイプ命令形式が使用されます。この形式では，26ビットのターゲット・アドレスは２ビット左にシフトされ，現在のプログラム・カウンタの上位４ビットと結合され，32ビット絶対アドレスが作られます。

リターンおよびページ間のジャンプ命令には，レジスタに格納されている32ビットのバイト・アドレスを用いるためＲタイプ命令形式が使用されます。

ブランチ命令の場合は，プログラム・カウンタ（分岐命令の次の遅延スロット内の命令のアドレス）に，16ビットのoffsetとを加えます（２ビット左にシフトして32ビット符合拡張），Jump-and-Link命令は，戻りアドレスをレジスタr31にセーブします。

表７－7はジャンプ命令を要約し，表７－8はブランチ命令を要約しています。

表7-7　ジャンプ命令の要約

(a) J, JAL

命　令	形式および説明	op	target
Jump	J　target 26ビットのターゲット・アドレスを左に2ビットシフトし，PCの上位4ビットと結合し，1命令遅れてアドレスへジャンプします。		
Jump And Link	JAL　target 26ビットのターゲット・アドレスを左に2ビットシフトし，PCの上位4ビットと結合し，1命令遅れてアドレスへジャンプします。遅延スロットに続く命令のアドレスをr31に格納します（リンク・レジスタ）。		

(b) JR

命　令	形式および説明	op	rs	0	funct
Jump Register	JR　rs 汎用レジスタrsに格納されているアドレスに，1命令遅れてジャンプします。				

(c) JALR

命　令	形式および説明	op	rs	0	rd	0	funct
Jump And Link Register	JALR　rs, rd 汎用レジスタrsに格納されているアドレスに，1命令遅れてジャンプします。遅延スロットに続く命令のアドレスを汎用レジスタrdに格納します。						

保守／廃止

表7-8　ブランチ命令の要約（1/2）

(a) BEQ, BNE

命　令	形式および説明	op	rs	rt	offset
Branch on Equal	BEQ　rs, rt, offset 汎用レジスタrs＝汎用レジスタrtの場合，ターゲット・アドレスへブランチします。				
Branch on Not Equal	BNE　rs, rt, offset 汎用レジスタrsと汎用レジスタrtが等しくない場合，ターゲット・アドレスへブランチします。				

(b) BLEZ, BGTZ

命　令	形式および説明	op	rs	0	offset
Branch on Less than or Equal Zero	BLEZ　rs, offset 汎用レジスタrsが0以下の場合，ターゲット・アドレスへブランチします。				
Branch on Greater Than Zero	BGTZ　rs, offset 汎用レジスタrsが0より大きい場合，ターゲット・アドレスへブランチします。				

備考　ターゲット・アドレスは，すべて次のように計算されます。遅延スロットの命令のアドレスと16ビットのoffsetとを加えます（2ビット左にシフトして32ビット符合拡張）。すべてのブランチは，1命令遅れて実行されます。

保守／廃止

表7-8　ブランチ命令の要約（2/2）

(c) BLTZ, BGEZ, BLTZAL, BGEZAL

命　令	形式および説明	op	rs	funct	offset
Branch on Less Than Zero	BLTZ　rs, offset 汎用レジスタrsが0より小さい場合，ターゲット・アドレスへブランチします。				
Branch on Greater than or Equal Zero	BGEZ　rs, offset 汎用レジスタrsが0以上の場合，ターゲット・アドレスへブランチします。				
Branch on Less Than Zero And Link	BLTZAL　rs, offset 遅延スロットに続く命令のアドレスをリンク・レジスタr31に格納します。汎用レジスタrsが0より小さい場合，ターゲット・アドレスへブランチします。				
Branch on Greater than or Equal Zero And Link	BGEZAL　rs, offset 遅延スロットに続く命令のアドレスをリンク・レジスタr31に格納します。汎用レジスタrsが0以上の場合，ターゲット・アドレスへブランチします。				

備考　ターゲット・アドレスは，すべて次のように計算されます。遅延スロットの命令のアドレスと16ビットのoffsetとを加えます（2ビット左にシフトして32ビット符合拡張）。すべてのブランチは，1命令遅れて実行されます。

保守／廃止

7.6　特殊命令

２種類の特殊命令が用意されていて，ソフトウエア・トラップが実行できるようにしています。これらの命令は，常にＲタイプです。表７－９に，特殊命令を要約します。

表７－９　特殊命令の要約

(a) SYSCALL

命　令	形式および説明	op	0	funct
System Call	SYSCALL システム呼び出しトラップを開始し，例外処理プログラムにただちに制御権を渡します。			

(b) BREAK

命　令	形式および説明	op	code	funct
Breakpoint	BREAK ブレークポイント・トラップを開始し，例外処理プログラムにただちに制御権を渡します。			

保守／廃止

7.7　コプロセッサ命令

V_R3800は，外付けコプロセッサ（CP1-CP3）をサポートしていません。このため，コプロセッサ命令（BCzF，BCzT，CFCz，COPz，CTCz，LWCz，MFCz，MTCz，SWCz）は実行できません。これらの命令を実行しようとした場合の動作は次のとおりです。

・ステータス・レジスタのCu1-Cu3の該当ビット　＝0…コプロセッサ使用不可例外発生
　　〃　　＝1…未定義

ただしV_R3800の独自機能として，BC0T，BC0F命令でライト・バッファの状態を知ることができます。表7-10に，コプロセッサ命令を要約します。

表7-10　コプロセッサ命令の要約

命　令	形式および説明　op　funct　offset
Branch on Coprocessor 0 True	BC0T　offset 16ビットのoffsetに命令のアドレスを加えて，ブランチ・ターゲット・アドレスを算出します（左に2ビット・シフトし，32ビットに符号拡張）。ライト・バッファにデータがあれば，ターゲット・アドレスへブランチします（1命令の遅れ）。
Branch on Coprocessor 0 False	BC0F　offset 16ビットのoffsetに命令のアドレスを加えて，ブランチ・ターゲット・アドレスを算出します（左に2ビット・シフトし，32ビットに符号拡張）。ライト・バッファにデータがなければ，ターゲット・アドレスへブランチします（1命令の遅れ）。

保守／廃止

7.8 システム制御コプロセッサ（CP0）命令

コプロセッサ0命令は，プロセッサのメモリ管理および例外処理機能を行うために，システム制御コプロセッサ（CP0）レジスタに対するオペレーションを実行します。表7－11に，CP0に使用できる命令を要約します。

表7－11　システム制御コプロセッサ（CP0）命令の要約

(a) MTC0，MFC0

命　令	形式および説明　op ｜ funct ｜ rt ｜ rd ｜ 0
Move To CP0	MTC0　rt，rd CPUレジスタrtの内容をCP0のレジスタrdにロードします
Move From CP0	MFC0　rt，rd CP0レジスタrdの内容をCPUのレジスタrtにロードします。

(b) RFE

命　令	形式および説明　op ｜ CO ｜ 0 ｜ funct
Restore From Exception	RFE ステータス・レジスタの直前の割り込みマスクとモード・ビットを，現在のステータス・ビットに復元します。古いステータス・ビットを直前のステータス・ビットに復元します。

備考　V_R3800にはTLBがないため，V_R3000AにあったTLB命令はサポートしていません。

7

保守／廃止

〔メ　モ〕

第８章　パイプライン

8.1　パイプラインの動き

命令の実行は，以下の５つの主なステップ（パイプ・ステージ）で構成されます。

(1) IF　－　命令取り出し，TLBにアクセスし，Iキャッシュからの命令の読み取りに必要な命令アドレスを計算します。RDパイプ・ステージが開始されるまで（フェーズ１），命令は実際にはプロセッサに読み込まれません，

(2) RD　－　命令をデコードしている間，CPUレジスタから必要なオペランドを読み取ります（RF=Register Fetch)。

(3) ALU　－　命令オペランドに対し，必要なオペレーションを実行します，

(4) MEM　－　必要に応じて（ロード命令またはストア命令），メモリに対してアクセスします（Dキャッシュ)。

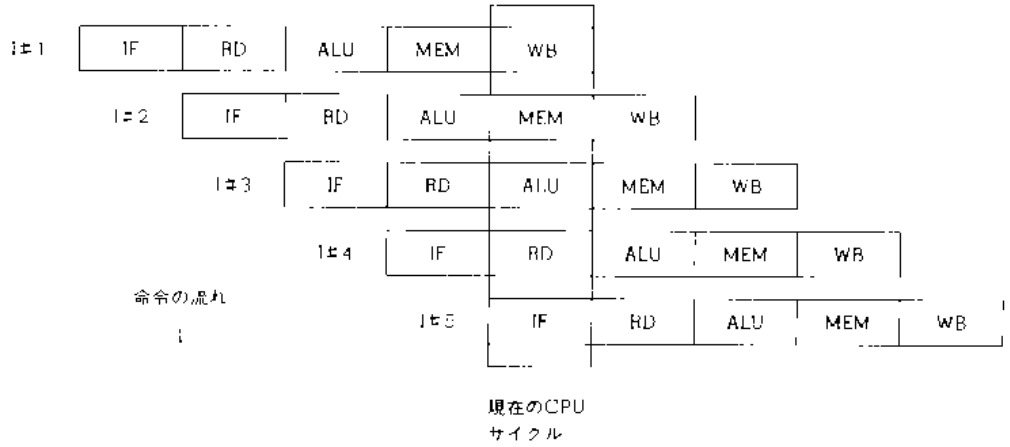
(5) WB　－　ALUの結果またはDキャッシュからロードされた値を，レジスタ・ファイルに書き込みます。

これらの各ステップで必要なサイクルは，図８-１に示されているように，ほぼ，１CPUサイクルです（一部のオペレーションは別のサイクルに重なりますが，1/2サイクルしか必要としないサイクルもあり，平均して約１リイクルです)。

図８-１　命令実行順序

1命令1CPUサイクルに近い命令実行速度を達成するために，5命令パイプラインを使用します。したがって，一度に5つの命令が，図8-2に示しているようにオーバラップします。

図8-2　命令パイプライン



8.2　遅延スロット

V_R3800は，すべての命令を1サイクルで実行できるように，いくつかの手法を用いています。しかし，特別な条件を必要とする2種類の命令があり，パイプラインにおける命令のスムーズな流れを妨げています。

(1) 遅延ロード

ロード命令には，ロードされるデータを別の命令が使用できるようになるまでに1サイクルの遅延（待ち時間）があります。

★　また，MFC0命令にも1サイクルの遅延があります。

(2) 遅延分岐

ジャンプ命令およびブランチ命令にも，分岐を実行する場合はターゲット・アドレスの命令を取り出す間，1サイクルの遅延があります。

★　これらの命令の遅延を処理する1つの方法は，ロード，MFC0，ジャンプ，またはブランチ命令が実行されるときに，パイプラインの命令の流れを停止させる方法です。しかし，この方法は，処理速度に悪い影響を与えるだけでなく，パイプライン論理，例外処理，およびシステム同期も複雑にします。

V_R3800が採用している方法は，パイプラインの遅延をプロセッサに処理させる代わりに，「遅延」命令の処理をソフトウエアで行う方法です。ソフトウエア（コンパイラやアセンブラなど）は，「遅延」命令の直後に遅延による影響を受けないように適当な命令を挿入します。

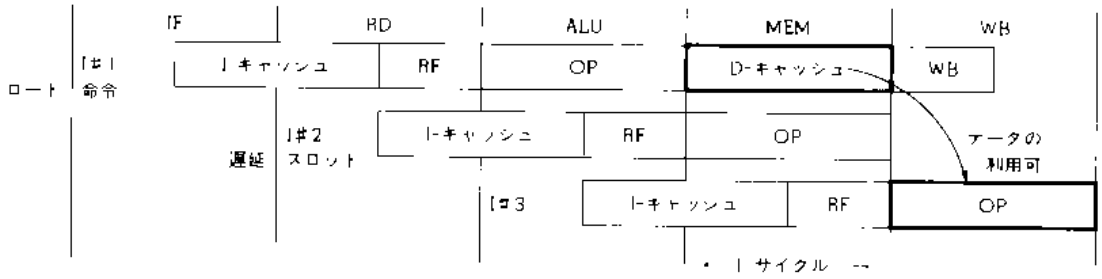
保守／廃止

(1) 遅延ロード

図8-3は，V_R3800パイプラインに3つの命令が入っている状態を示しています。命令1（I＃1）は，ロード命令です」ロードされるデータは，I＃1のMEMサイクルが終わるまで使用できません」したがって，I＃2のALUサイクルには間に合いませんが，I＃3のALUサイクルでは使用できます。そのため，ソフトウエアでは，I＃2がI＃1でロードされるデータに依存しないように命令を再編成し，遅延スロット中に実行の伴う命令を挿入します。他に命令がない場合は，NOP（No Operation）命令を遅延スロットに挿入します。

またMFC0命令では，コプロセッサからデータをロードするのに時間が必要です。ロード命令と同様にIサイクルの遅延があります。 ★

図8-3　遅延ロード

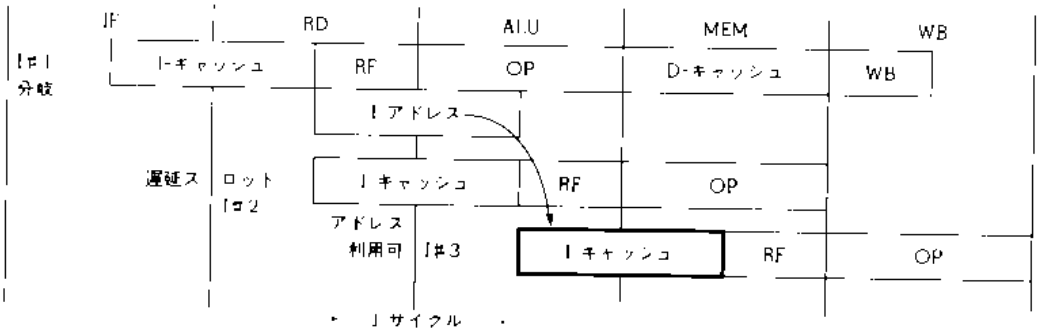


8

(2) 遅延分岐

図8－4も，V_R3800パイプラインに3つの命令が入っている状態を示しています．この場合，命令1（I＃1）は分岐命令です。I＃1は分岐ターゲット・アドレスを計算しなければなりませんが，そのアドレスはI＃1のALUサイクルが開始されるまで使用できません。したがって，I＃2のキャッシュ・アクセスには間に合いませんが，I＃3のIキャッシュ・アクセスでは使用できます。遅延スロット（I＃2）の命令は，実際にジャンプまたはブランチが実行される前に，常に実行されます。

図8－4　遅延分岐



アセンブラは，分岐遅延スロットを有効に利用するために，次のようないくつかの方法を使います。

(a) 分岐命令より論理的に前に実行される命令を，遅延スロットに挿入します。実際には，分岐命令の直後の命令は，分岐命令の直前のブロックに属しているからです。

(b) 分岐命令のターゲット（宛先）となる命令と同じ命令を，遅延スロット内に移動します。ただし，この命令は，分岐命令が実行されない場合でも影響を受けない命令であることが条件です。

(c) 分岐命令より後ろにある命令を，遅延スロットに移動します。ただし，分岐命令が実行されても影響を受けない命令であることが条件です。

(d) 上記のような方法が使用できない場合は，遅延スロットにNOP命令を挿入します。

保守／廃止

第９章　メモリ管理システム

本章では，アドレス・マッピングとカーネル／ユーザ・モードの詳細について説明します．V_{II}3800は組み込み制御分野用のため，V_{II}3000Aにあった TLBを削除し，仮想アドレスがそのまま物理アドレスになるようにしています．ただしkseg0，kseg1については，物理アドレスの下位領域にマッピングされます．

9.1　動作モード

V_R3800にはユーザ・モードとカーネル・モードの２種類の動作モードがあります．通常はユーザ・モードで動作しますが，例外を検出するとカーネル・モードになります．カーネル・モードになるとRFE（Restore From Exception）命令が実行されるまでカーネル・モードのままです．メモリ・アドレスのマッピングは，動作モードに依存しています．

ユーザ・モードでは，カーネル・モードの専用空間（kseg0，kseg1，kseg2）にアクセスできません．アクセスした場合は，アドレス・エラー例外を発生します．

9

保守／廃止

9.1.1　ユーザ・モード

このモードでは，kusegだけが使用できます．仮想アドレス0000 0000Hから始まる2Gバイトで，キャッシュ・メモリを使用する空間です。仮想アドレスがそのまま物理アドレスとして出力されます。

ユーザ・モードのアドレスの最上位ビットは常に0にしてください。最上位ビットを1にセットした状態でアドレスを参照しようとすると，アドレス・エラー例外を発生します。

9.1.2　カーネル・モード

このモードには，次の4つのセグメントがあります

(1) kuseg

ユーザ・モードと同様に，2Gバイトでキャッシュ・メモリを使用する空間です。カーネルからユーザ・データへ容易にアクセスできます，

(2) kseg0

仮想アドレス8000 0000Hから始まる512 Mバイトの空間です　キャッシュ・メモリを使用します，通常時の例外ベクタが置かれます。

(3) kseg1

仮想アドレスA000 0000Hから始まる512 Mバイトの空間です。キャッシュ・メモリは使用できません。ブート時の例外ベクタが置かれます。また，キャッシュ・メモリを使用できないI/Oは，この空間に配置します。

(4) kseg2

仮想アドレスC000 0000Hから始まる1 Mバイトの空間です。キャッシュ・メモリを使用します。

保守／廃止

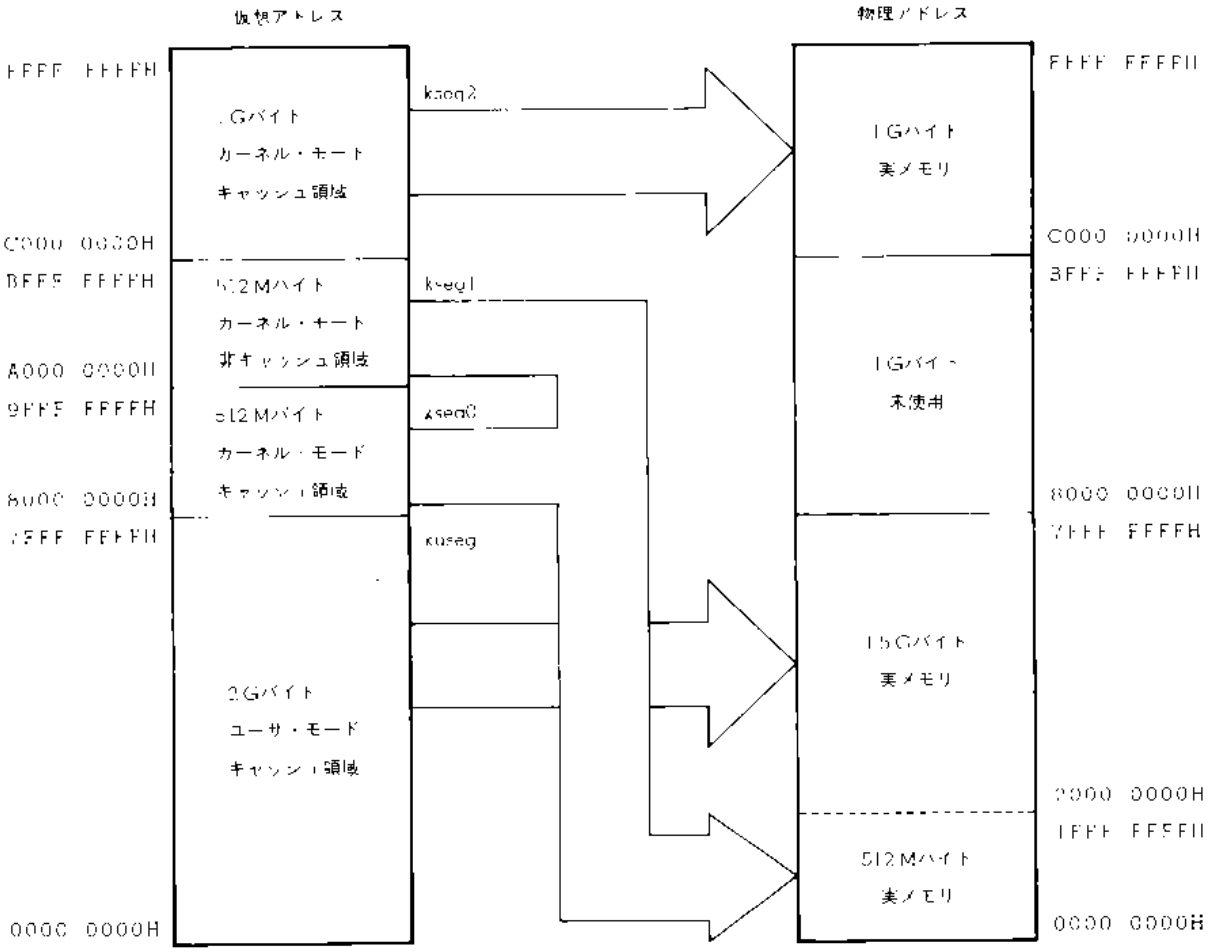
9.2　物理アドレス・マッピング

仮想アドレスのほとんどは変換されず，そのまま物理アドレスとして出力されます．ただしkseg0とkseg1は，どちらも物理アドレスの0番地から割り付けられます．よって，物理アドレスの0000 0000H-1FFF FFFFHの512 Mバイトの空間には，次の3つの空間が多重マッピングされます．

- 仮想アドレス0000 0000H-1FFF FFFFH ：kusegの一部
- 仮想アドレス8000 0000H-9FFF FFFFH ：kseg0
- 仮想アドレスA000 0000H-BFFF FFFFH ：kseg1

また，物理アドレス8000 0000H-BFFF FFFFHの1 Gバイトは未使用空間です．

図9-1　アドレス・マップ



備考　kseg はカーネル・セグメント，kuseg はカーネル・ユーザ・セグメントです．

〔メ　モ〕

第10章　例外処理

本章では，V_H3800プロセッサによる例外処理方法，および例外処理時に使用されるシステム制御コプロセッサ（CP0）レジスタについて説明します。

V_R3800が例外を検出すると，通常の命令実行は中断され，ユーザ・モードからカーネル・モードになり，異常事象に応答します。本章では，例外処理が開始されることになるすべての事象を説明します。

V_H3800例外処理システムは，算術オーバフロー，入出力割り込みおよびシステム呼び出しなどのマシン例外を効率的に処理します。これらの事象は，すべて正常な実行の流れに割り込みをかけます。つまり，V_H3800は例外を引き起こした命令を異常終了させ，すでに実行が開始されている命令パイプライン内の命令もすべて実行を中断させます。異常終了させたあと，V_R3800は指定された例外処理ハンドラ・ルーチンに直接ジャンプします。

例外が発生すると，V_R3800は，その例外が処理されたあとに実行が再開されるリスタート・メモリ位置をEPCレジスタ（例外プログラム・カウンタ）にロードします。EPCレジスタにロードされるリスタート・アドレスは，例外を引き起こした命令のアドレスか，または，命令が分岐遅延スロット内で実行されていた場合は，遅延スロットの直前の分岐命令のアドレスです。

表 10－1に，V_R3800が認識する例外を示します。

保守／廃止

10.1　例外処理一覧

表 10－1　例外

例　外	ニモニック	原　　因
リセット	----	V_R3800のRESET信号がアクティブになったあとにインアクティブになるとリセット例外が起き，制御権は仮想アドレスBFC00000Hのベクタに渡されます。
アドレス・エラー	AdEL（ロード） AdES（ストア）	位置合わせされていないワード（つまり，4または2で割り切れないアドレスのワードまたはハーフ・ワード）に対するロード，フェッチ，またはストア命令時に起こります。また，ユーザ・モードで最上位ビットが1にセットされた仮想アドレスを参照する場合も起こります。
オーバフロー	Ov	加算または減算実行時の2の補数オーバフロー時に起こります。
システム・コール	Sys	syscall命令の実行時に起こります。
ブレークポイント	Bp	break命令の実行時に起こります。
予約命令	RI	未定義または予約メージャ・オペレーション・コード（ビット31-26）の実行または予約マイナ・オペレーション・コード（ビット5-0）が未定義の特殊命令の実行時に起こります。
コプロセッサ使用不可	CpU	ターゲット・コプロセッサに対しCU（Coprocessor Unusable）ビットがセットされていないときのコプロセッサ命令の実行時に起こります。
割り込み	Int	V_R3800の6つのハードウエア割り込み入力のいずれかがオン，または原因レジスタの2つのソフトウエア割り込みビットのいずれかがセットされている時に起こります。

備考　V_R3800には，V_R3000Aにあった TLBと $\overline{\text{BusError}}$端子がないため，UTLB不一致例外，TLB不一致例外，TLB変更例外およびバス・エラー例外は発生しません。

保守／廃止

10.2　例外処理レジスタ

例外が発生すると，６個の例外処理レジスタに例外関連の情報が格納されます。ソフトウエアは，例外処理時にこれらのレジスタを調べ，例外の原因や例外発生時のCPUの状態などを判別できます。これらの各レジスタの説明は，後述します。

(a) 原因レジスタ
(b) EPC（例外プログラム・カウンタ）レジスタ
(c) ステータス・レジスタ
(d) BadVAddr（Bad Virtual Address-誤り仮想アドレス）レジスタ
(e) コンテキスト・レジスタ
(f) PRId（Processor Revision Identifier）レジスタ

図10－1に，CP0の例外処理レジスタを示します。

図10－1　CP0の例外処理レジスタ

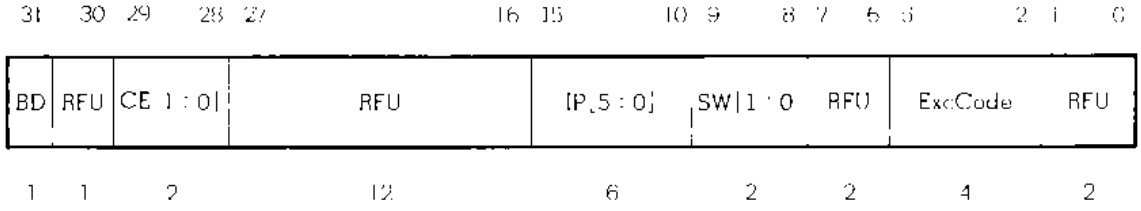
保守／廃止

10.2.1　原因レジスタ

この32ビット・レジスタの内容は，最後に発生した例外の内容を保持しています．４ビットの例外コード（ExcCode）を，表10－2に示します。残りのフィールドには，例外の詳細な情報が入っています。Swビットを除き，このレジスタのすべてのビットは読み取り専用です。SWビットは，ソフトウエア割り込みをセットまたはリセットするために書き込むことができます．

原因レジスタの形式を，図10－2に示します。

図10－2　原因レジスタ



BD	分岐遅延	分岐遅延スロットで実行されているときに最後の例外が発生すると，１にセットされます．
CE	コプロセッサ・エラー	コプロセッサ使用不可例外が発生したときに参照されているユニット番号です。 (CE1，CE0)　コプロセッサ・ユニット番号 （０，０）＝コプロセッサ・ユニット番号 0 （０，１）＝　〃　1 （１，０）＝　〃　2 （１，１）＝　〃　3
IP	割り込み保留	保留されているハードウエア割り込みを示しています．IP[5．0]＝INT[5　0]
SW	ソフトウエア割り込み	２つのソフトウエアのどれが保留中であるかを示しています，このフィールドは，ソフトウエア割り込みをセット，またはリセットするために書き込むことができます
ExcCode	例外コードフィールド	表10　2で説明．
RFU		書き込みを実行しても無視されます　読み出すとゼロが返されます．

保守／廃止

表 10－2　ExcCode フィールド

原因レジスタExcCode フィールド

番　号	ニモニック	説　　明
0	Int	割り込み
1	－－	予約
2	－－	予約
3	－－	予約
4	AdEL	アドレス・エラー例外 （ロード命令または命令フェッチ）
5	AdES	アドレス・エラー例外（ストア命令）
6	－－	予約
7	－－	予約
8	Sys	システム・コール例外
9	Bp	ブレークポイント例外
10	RI	予約命令例外
11	CpU	コプロセッサ使用不可例外
12	Ov	算術オーバフロー例外
13-15	－－	予約

★

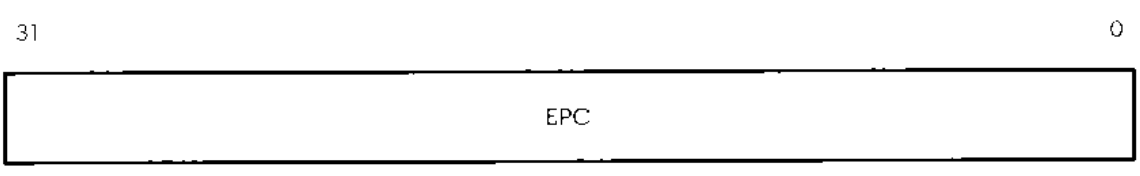
10.2.2　EPC（例外プログラム・カウンタ）レジスタ

EPCレジスタは，32ビットの読み取り専用レジスタで，例外が処理されたあとに処理が再開されるアドレスが格納されています」

このレジスタには，例外を引き起こした命令の仮想アドレスが入ります。その命令が分岐遅延スロット内に含まれている場合は，EPCレジスタには直前のブランチまたはジャンプ命令の仮想アドレスが入ります。また，V_H3800が分岐遅延スロット内で例外が発生した場合は，原因レジスタのBDビットもセットします，EPCレジスタの形式を，図 10－3に示します，

10

図 10－3　EPCレジスタ



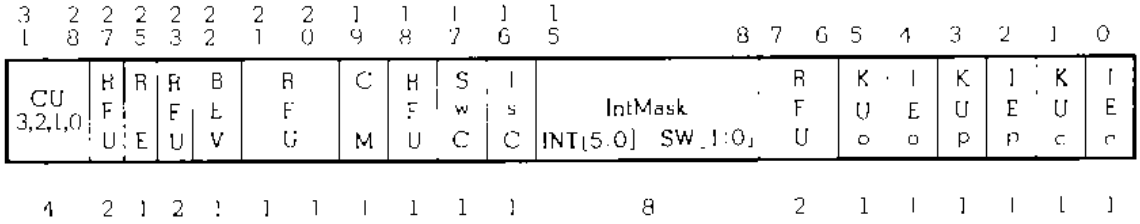
保守／廃止

10.2.3　ステータス・レジスタ

このレジスタには，主要なステータス・ビットがすべて含まれています」例外が発生すると，V_R3800はカーネル・モードになります。ステータス・レジスタのすべてのビットは読み出しおよび書き込み可能です。

図10－4に，ステータス・レジスタの形式を示し，各ビットごとの機能を要約します。各ステータス・レジスタ・ビットの機能に関する詳細は，後述します。

図10－4　ステータス・レジスタ



CU	コプロセッサの使用可能性 4つのコプロセッサ（Cu3，Cu2．Cu1およびCu0）の使用可能性は，これらのビットにより制御されます，CUビットがセットされていると（＝1），そのコプロセッサは使用できます。 なお，V_R3800ではCu0だけ使用できます。
RE	エンディアン反転 初期設定されたエンディアンをダイナミックに反転できます。
BEV[注]	ブートストラップ例外ベクタ 1にセットされた場合にUTLB不一致例外および一般例外が発生するとV_R3800は代替ブートストラップ・ベクタを使用します。
CM[注]	キャッシュ失敗 最後のDキャッシュ・ロードが失敗すると1にセット（Dキャッシュ分離されているときのみ）されます。
SwC[注]	Swap Caches Iキャッシュと Dキャッシュの制御信号の切り替えを制御します。
IsC[注]	Isolate Caches 1にセットされると，メイン・メモリ・システムはDキャッシュから分離されます。
IntMask	Interrupt Mask（割り込みマスク） 1にセットされると，対応するハードウエア割り込み（INT［5：0］）またはソフトウエア割り込み（SW［1：0］）が割り込み可能になります。
KUo	Kernel/User mode, old カーネル・モードの場合0。ユーザ・モードの場合1。

IEo	Interrupt Enable, old 割り込み可能の場合１，割り込み不可能の場合０。
KUp	Kernel/User Mode, previous カーネル・モードの場合０，ユーザ・モードの場合１。
IEp	Interrupt Enable, previous 割り込み可能の場合１，割り込み不可能の場合０。
KUc	Kernel/User Mode, current カーネル・モードの場合０，ユーザ・モードの場合１。
IEc	Interrupt Enable, current 割り込み可能の場合１，割り込み不可能の場合０。
RFU	書き込みを実行しても無視されます，読み出すとゼロが返されます。

注　主な用途は診断およびテストです。

備考　V_R3800ではTLBとパリティをサポートしていないため，V_R3000AにおけるTS（ビット21），PE（ビット20）とPZ（ビット18）はRFUです。

保守／廃止

(1) CU (Coprocessor Usable)

CUビットは，4つの各コプロセッサ（Cu3-Cu0）の使用の可否を制御します。ソフトウエアでは，これらのビットによりコプロセッサに対するプロセス・アクセスを制御できます。ビットが1にセットされていると，対応するコプロセッサは使用できます．ビットがクリア（0）されていると，コプロセッサは使用できません。コプロセッサに対する命令を実行する場合は，ターゲット・コプロセッサには使用可能ビットがセットされていなければなりません．使用可能ビットがセットされていない場合は，コプロセッサ使用不可例外が発生します　システム制御コプロセッサ（CP0）の場合は，V_R3800かカーネル・モードで動作していれば，Cu0ビットの設定に関係なく常に使用できます。

(2) RE (Reversal Endianess)

REビットは，エンディアンをダイナミックに反転します。このビットを1にすると，ユーザ・モードにおけるあらゆるバーシャル・ワード，非アライン・データのバイト順序か，リセット時に選択されたエンディアンとは反転します．

(3) BEV (Bootstrap Exception Vectors)

BEVビットは，ブートストラップ時（リセットの直後）に使用される，UTLB不一致ベクタのメモリ位置および一般例外ベクタのメモリ位置を制御します．このビットが0にセットされていると，通常の例外ベクタが使用されます。このビットが1にセットされていると，ブートストラップ・ベクタ・メモリ位置が使用されます。

このベクタの代替セットは，キャッシュとメイン・メモリ・システムのオペレーションが正常かどうかを検査する前に，診断テストにより例外が発生する場合に使用できます（例外ベクタの説明は，本章で後述する**10.3　例外の詳細説明**を参照してください)。

(4) CM (Cache Miss)

CMビットは．最後のDキャッシュ・ロードでキャッシュ・ミスが発生するとセットされます　このビットは，診断フログラムでキャッシュ・タグか正常に機能しているかどうかを検査するためのものです。このビット設定は，"isolated cache" モードの場合だけ有効です。**(6) IsC (Isolate Caches)**を参照してください。

(5) SwC (Swap Caches)

SwCビットは，データ・キャッシュ（Dキャッシュ）と命令キャッシュ（Iキャッシュ）の制御信号の交換を制御します（0は正常で，1は交換)．キャッシュ交換は．ソフトウエアでのキャッシュ・フラッシュ機構の実現とキャッシュ・テストと診断の実行に使用できます．

(6) IsC (Isolate Caches)

IsCをセットすると，メイン・メモリ・システムはDキャッシュから分離されます（0は正常で，

保守／廃止

1はDキャッシュ分離)。IsC＝1のとき，すべてのロード命令はキャッシュ・ヒットし（タグの比較を行わない)，ストア命令ではメイン・メモリ・ライトが行われなくなります（MemWr信号がアクティブにならない)。このときのデータは不定です。Dキャッシュ・フラッシュ機構の実現とキャッシュ・テストと診断の実行に使用できます。

(7) IntMask（Interrupt Mask)

IntMaskビットは，8種類の割り込み（6種類のハードウエア割り込みと2種類のソフトウエア割り込み）のそれぞれを，個別に割り込み許可/割り込み禁止にします。ビット位置に0がセットされると割り込み禁止で，1がセットされると割り込み許可になります。すべての割り込みは，以下に説明する割り込み許可ビット/IEo/IEp/IEcをクリアすることで割り込み禁止になります。

割り込み処理方法の詳細は**3.6　割り込み**を参照してください，

(8) KUo/KUp/KUc（Kernel/User mode . Old/Previous/Current)

KUo/KUp/KUcの3ビットは，3レベル・スタックを構成しold/previous/currentの動作モートを示しています（0はカーネル・モードで，1はユーザ・モード)。これらはソフトウエアで読み書きできます。例外処理時におけるこれらのビットの操作と使用方法は，以下の項で説明します。

(9) IEo/IEp/IEc（Interrupt Enable：Old/Previous/Current)

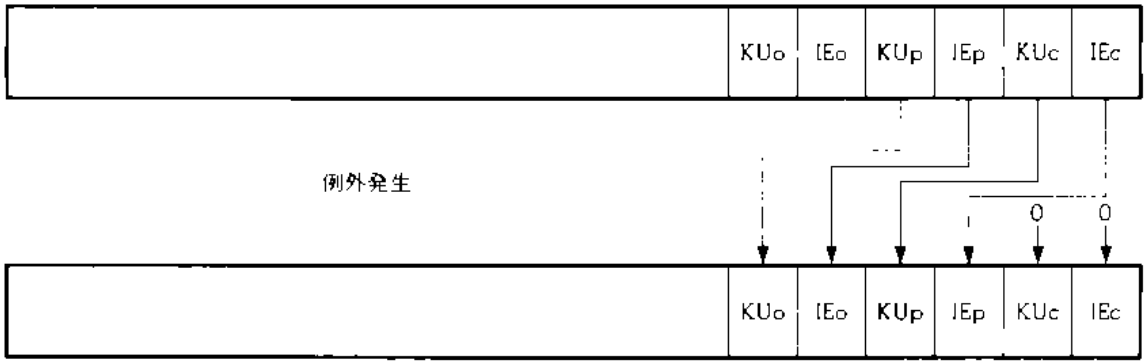
IEo/IEp/IEcの3ビットは，3レベル・スタックを構成しold/previous/currentの割り込み可能設定を示しています（0は割り込み禁止で，1は割り込み許可)。これらはソフトウエアで読み書きできます。例外処理時におけるこれらのビットの操作と使用方法は，以下の節で説明します。

10.2.4　ステータス・レジスタ・モード・ビットと例外処理

V_R3800が例外に応答するとき，V_R3800はステータス・レジスタの現カーネル/ユーザ・モード（KUc）ビットと現割り込み可能モード（IEc）ビットを，前モート・ビット（KUpとIEp）にそれぞれ保存します，前モード・ビット（KUpとIEp）は，旧モード・ビット（KUoとIEo）にそれぞれ保存されます，現モード・ビット（KUcとIEc）はクリアされ，プロセッサはカーネル・オペレーティング・モードになり，割り込みは禁止になります。

この3レベルのモード・ビットが備わっているため，ソフトウエアがステータス・レジスタの内容を保存する前に，V_R3800は2レベルの例外に応答できます。図10－5に，例外処理時におけるV_R3800のステータス・レジスタ保存操作を示します。

図 10－5　例外発生時のステータス・レジスタ

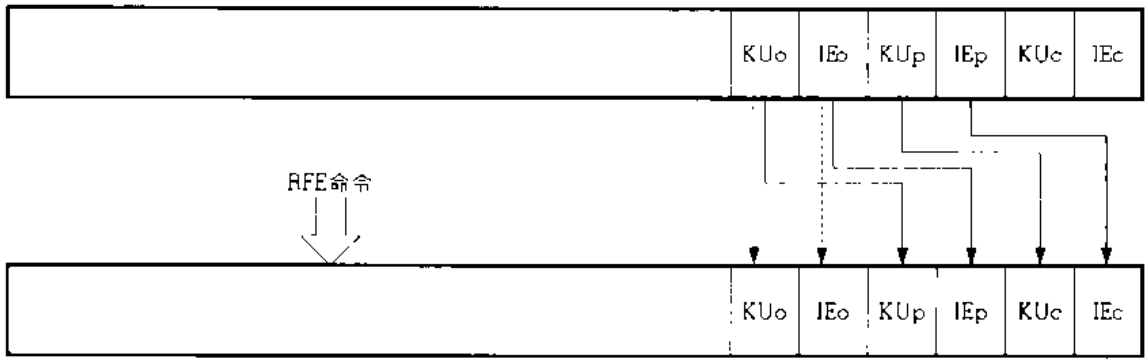


例外ハンドラが実行を完了すると，V_R3800は例外前のシステム状況に復帰しなければなりません。RFE（Restore From Exception）命令は，この復帰のための機構を実行します。

RFE命令は，例外処理が実行される前のプロセスに制御権を返します。RFE命令が実行されると，ステータス・レジスタ内の前割り込みマスク（IEp）ビットと前カーネル/ユーザ・モード（KUp）ビットを，対応する現ステータス・ビット（IEcとKUc）にそれぞれ復元します。また，旧ステータス・ビット（IEoとKUo）を，対応する前ステータス・ビット（IEpとKUp）に復元します。旧ステータス・ビット（IEoとKUo）はそのままです。

RFE命令の動作を，図 10－6に示します。

図 10－6　例外からの復帰

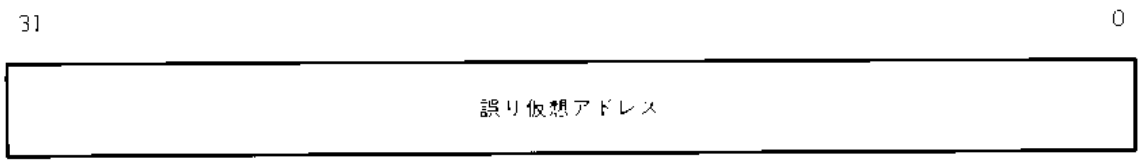


10.2.5　BadVAddrレジスタ

BadVAddrレジスタは，アドレス・エラー例外（AdELとAdES）が発生すると，その誤り仮想アドレスを保存します。図 10－7に，このレジスタの構成を示します。

保守／廃止

図 10－7　BadVAddrレジスタ



10.2.6　コンテキスト・レジスタ

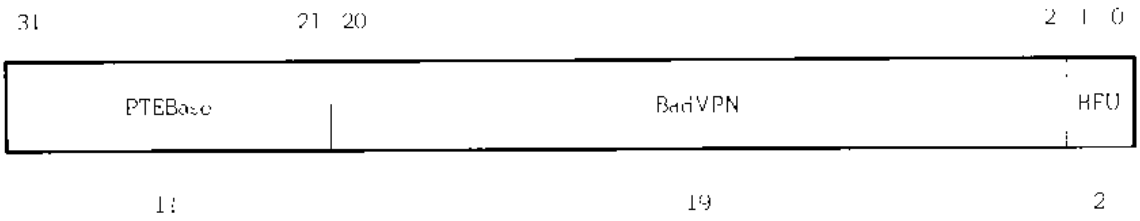
このレジスタはTLBがあるV$_R$3000Aで，TLBに関連する例外を処理するために設けられたレジスタです．V$_R$3800でも使用できますが，BadVAddrレジスタでも機能をカバーしています

PTEBaseはオペレーティング・システムが使用する領域です　ページ・テーブルのベース・フィールドをセットします．ハードウエアでは書き換えられません。TLBがないV$_R$3800では不要のフィールドです。

VPNにはアドレス・エラー例外が発生した仮想ページ番号が格納されます。この情報はBadVAddrレジスタのビット30-ビット12と同じです。

図10－8に，コンテキスト・レジスタの形式を示します

図 10－8　コンテキスト・レジスタ



PTEBase	ページ・テーブル・エントリのベース・アドレス（ソフトウエアによりセット）
BadVPN	障害仮想ページ番号（ハードウエアによりセット） このフィールドは読み出し専用で，BadVAddrレジスタのビット30-12（ユーザ・セグメントVPN）が格納されます．
RFU	書き込みを実行しても無視されます．読み出すとゼロが返されます．

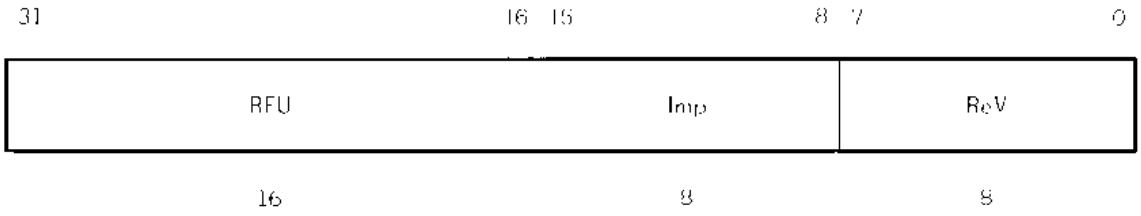
10.2.7　プロセッサ・リビジョン識別子レジスタ

このレジスタは，32ビットの読み出し専用レジスタで，プロセッサおよびシステム制御コプロセッサの構成要素およびリビジョン・レベルの識別情報が入っています．

図10－9 に，このレジスタの形式を示します。

保守／廃止

図10-9　プロセッサ・リビジョン識別子レジスタ



ImP	構成要素識別子（02H）
ReV	リビジョン識別子
RFU	書き込みを実行しても無視されます。読み出すとゼロが返されます。

保守／廃止

10.3　例外の詳細説明

ここでは，以下の各例外（その原因，処理，操作）について説明します。

・アドレス・エラー例外
・ブレークポイント例外
・コプロセッサ使用不可例外
・割り込み例外
・オーバフロー例外
・予約命令例外
・リセット例外
・システム・コール例外

注意　割り込み以外の例外はマスクできません。

例外ベクタのメモリ位置

V_R3800は，次の２つの例外ベクタ・アドレスを使用します。

(1) リセット例外ベクタは，アドレスBFC0　0000Hです。
(2) その他すべての例外タイプに使用される一般例外ベクタは，アドレス8000　0080Hです」

注意　ステータス・レジスタのBEV（ブートストラップ例外ベクタ）ビットが１の場合，一般例外ベクタはBFC0　0180Hに変更されます。

10

保守／廃止

10.3.1　アドレス・エラー例外

●原因

この例外は，ワード境界に位置合わせされていないワード・データのロード/ストアまたはフェッチを実行したときや，ハーフ・ワード境界に位置合わせされていないハーフ・ワード・データのロード/ストアを実行したときにこの例外が発生します．また，ユーザ・モードでカーネル・モードのアドレス空間を参照した場合にも発生します　この例外は，マスクできません．

●処理

V_R3800は，この例外が発生すると一般例外ベクタ（8000　0080H）に分岐します　原因レジスタのExcCodeフィールドには，命令フェッチ，ロード・オペレーション中にアドレス・エラー例外が発生した場合にはAdELコードを，ストア・オペレーション中にアドレス・エラー例外が発生した場合にはAdESコードをセットします．

EPCレジスタは，命令が分岐遅延スロット内の命令でないかぎり，例外を引き起こした命令を指しています　しかし，例外を起こした命令が分岐遅延スロット内の命令の場合には，EPCレジスタは分岐命令を指し，原因レジスタのBDビットをセットします．

V_R3800は，ステータス・レジスタのKUp，IEp，KUcおよびIEcビットをKUo，IEo．KUpおよびIEpビットにそれぞれ保存し，KUcビットとIEcビットをクリアします．BadVAddrレジスタには位置合わせが正しくない仮想アドレス，またはユーザ・モード時にカーネル空間を不正にアドレス指定した仮想アドレスが格納されます．コンテキスト・レジスタのVPNフィールドの内容は，未定義です．

●操作

カーネルは，実行中のプロセスにセグメンテーション違反信号を渡します

注意　位置合わせエラーは，エラーを引き起こした命令をシミュレートすることで処理できますが，このようなエラーは致命的なエラーです．

保守／廃止

10.3.2　ブレークポイント例外

●原因

この例外は，V_R3800がBREAK命令を実行すると発生します　この例外は，マスクできません。

●処理

この例外が発生すると，V_R3800は一般例外ベクタ（8000　0080H）に分岐し，原因レジスタのExcCodeフィールドにBPコードをセットします．V_R3800は，ステータス・レジスタのKUp，IEp，KUcおよびIEcビットをKUo，IEo，KUpおよびIEpビットにそれぞれ保存し，KUcビットとIEcビットをクリアします．

EPCレジスタは，BREAK命令が分岐遅延スロット内でないかぎり，BREAK命令を指しています．しかし，BREAK命令が分岐遅延スロット内の場合には，EPCレジスタは分岐命令を指し，原因レジスタのBDビットをセットします．

●操作

該当するシステム・ルーチンに制御権を渡します．BREAK命令の未使用ビット（ビット26-6）は，追加情報を渡す場合に使用できます．これらのビットを調べるためには，EPCレジスタにより指定されている命令の内容をロードしてください．

実行を再開するためには，V_R3800がBREAK命令を再び実行しないようにEPCレジスタ内のアドレスを変更します．

注意1．　命令が分岐遅延スロット内に存在する場合は，EPCレジスタを読み出して4を加えて，そのアドレス値が指している命令を見つけてください。

2．　BREAK命令が分岐遅延スロット内に存在する場合は，実行を再開するために分岐命令を解釈しなければなりません。

保守／廃止

10.3.3　コプロセッサ使用不可例外

●原因

この例外は，対応するコプロセッサ・ユニットに使用可能のビットがセットされていないときに（ステータス・レジスタ内の該当するCUビットがセットされていないとき），コプロセッサ命令を実行しようとすると発生します。

CP0命令では，プロセスがユーザ・モードで実行されている場合，CUビットがセットされていないときにこの例外が発生します。カーネル・モードでは，CUビットの状態にかかわらず，この例外は発生しません。また，この例外はマスクできません。

●処理

この例外が発生すると，V_R3800は一般例外ベクタ（8000　0080H）に分岐します。原因レジスタのExcCodeフィールドには，CpUコードがセットされます。障害は，一度に1つのコプロセッサにのみ発生します。

4つのコプロセッサ（3，2，1または0）のうち，例外発生時にV_R3800が参照していたコプロセッサは原因レジスタのCE（Coprocessor Error）フィールドを調べれば分かります。

EPCレジスタは，命令が分岐遅延スロット内の命令でないかぎり，例外を引き起こした命令を指しています。しかし，例外を起こした命令が分岐遅延スロット内の命令の場合には，EPCレジスタは分岐命令を指し，原因レジスタのBDビットをセットします。

V_R3800は，ステータス・レジスタのKUp，IEp，KUcおよびIEcビットをKUo，IEo，KUpおよびIEpビットにそれぞれ保存し，KUcビットとIEcビットをクリアします。

●操作

参照されていたコプロセッサを調べるためには，原因レジスタのCEフィールドの内容を調べてください。プロセスがアクセスできる場合は，コプロセッサに使用可能のビットをセットして，対応するユーザ・ステータスをコプロセッサに復元してください。

プロセスがコプロセッサにアクセスできる場合で，コプロセッサが接続されていないことが分かっているかまたは障害が発生していることが分かっている場合，システムはコプロセッサ命令を解釈できます。原因レジスタのBDビットがセットされている場合は，分岐命令を解釈しなければなりません。その後，EPCレジスタをコプロセッサ命令の後続の命令まで進めてコプロセッサ命令をエミュレートできます。

保守／廃止

注意　プロセスがコプロセッサにアクセスできない場合，例外発生時に実行されていたプロセスには，違法命令/特権命令障害信号が通知されます。このようなエラーは，致命的なエラーです。

10.3.4　割り込み例外

●原因

この例外は，８種類の割り込み条件（そのうち，２種類は，ソフトウエアにより生成され，６種類はハードウエアにより生成されます）により発生します。８種類の割り込みのそれぞれは，ステータス・レジスタのIntMaskフィールドの対応するビットをクリアすることで個別にマスクできます。ステータス・レジスタのIEビットをクリアすると，８種類の割り込みすべてをマスクできます，

●処理

この例外が発生すると，V_R3800は一般例外ベクタ（8000　0080H）へ分岐します。V_R3800は，原因レジスタのExcCodeフィールドにIntコードをセットします，

原因レジスタのIPフィールドを調べると，６種類のハードウエア割り込みのうちのどの割り込みが保留されているかが分かり，原因レジスタのSwフィールドを調べると２種類のソフトウエア割り込みのうち，どの割り込みが保留されているかが分かります，複数の割り込みを同時に保留することができます。

V_R3800は，ステータス・レジスタのKUp，IEp，KUcおよびIEcビットをKUo，IEo，KUpおよびIEpビットにそれぞれ保存し，KUcビットとIEcビットをクリアします。

●操作

ソフトウエアが割り込みを生成する場合は，対応する原因レジスタ・ビット（Sw1，Sw0）をゼロにセットして，割り込み状態をクリアしてください，

ハードウエアが割り込みを生成する場合は，割り込み信号INT‘5：0|をオンにしている状態を解決して，割り込み状態をクリアしてください，

10.3.5　オーバフロー例外

●原因

この例外は，ADD，ADDIまたはSUB命令を実行した結果，２の補数のオーバフローになると発生します。この例外は，マスク不可能です。

●処理

この例外が発生すると，V_R3800は一般例外ベクタ（8000　0080H）へ分岐します。V_R3800は原因レジスタのExcCodeフィールドにOvコードをセットします。

EPCレジスタは，命令が分岐遅延スロット内の命令でないかぎり，例外を引き起こした命令を指しています。しかし，例外を起こした命令が分岐遅延スロット内の命令の場合には，EPCレジスタは分岐命令を指し，原因レジスタのBDビットをセットします。

V_R3800は，ステータス・レジスタのKUp，IEp，KUcおよびIEcビットをKUo，IEo，KUpおよびIEpビットにそれぞれ保存し，KUcビットとIEcビットをクリアします。

●操作

この例外が発生すると，カーネルは実行プロセスに浮動小数点例外または整数オーバフロー・エラーを通知します。しかし，このようなエラーは致命的なエラーです。

保守／廃止

10.3.6　予約命令例外

●原因

この例外は，メージャ・オペレーション・コード（ビット31-26）が未定義の命令，またはマイナ・オペレーション・コード（ビット5-0）が未定義のSPECIAL命令をV_R3800が実行すると発生します。

●処理

この例外が発生すると，V_R3800は，一般例外ベクタ（8000　0080H）へ分岐します。V_R3800は，原因レジスタのExcCodeフィールドにRIコードをセットします。

EPCレジスタは，命令が分岐遅延スロット内の命令でないかぎり，例外を引き起こした命令を指しています。しかし，例外を起こした命令が分岐遅延スロット内の命令の場合には，EPCレジスタは分岐命令を指し，原因レジスタのBDビットをセットします。

V_R3800は，ステータス・レジスタのKUp，IEp，KUcおよびIEcビットをKUo，IEo，KUpおよびIEpビットにそれぞれ保存し，KUcビットとIEcビットをクリアします。

●操作

命令が解釈できない場合，カーネルは実行プロセスに対し違法命令/予約オペランド障害信号を通知します。

オペレーティング・システムは未定義命令を解釈し，ソフトウエアでその命令を実現しているルチンに制御権を渡すことができます。未定義命令が分岐遅延スロット内に含まれている場合は，命令を実現しているルーチンは未定義命令が「実行」されたあと，分岐命令をシミュレートしなければなりません。分岐命令のシミュレーションには，次の２点も含まれます。

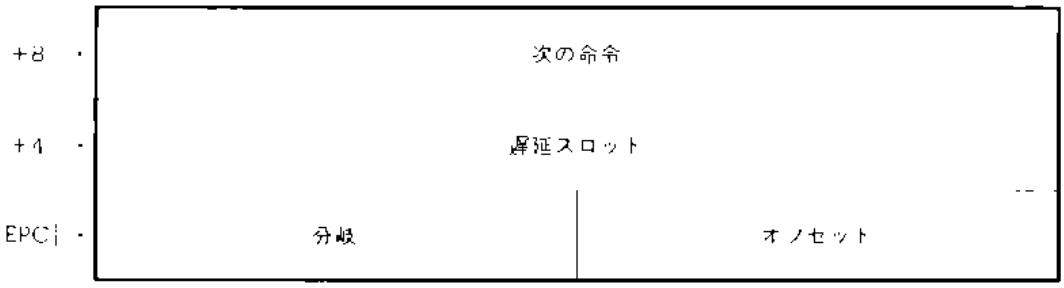
(1) 分岐条件が満足されているかどうかを判別します。

(2) 分岐ターゲット・アドレスへ制御権を渡すか，または分岐命令が実行されない場合は遅延スロットの次の命令への制御権を渡します。分岐命令が実行されない場合は，次に実行される命令のアドレスは［EPC］＋８です。分岐命令が実行される場合は，分岐ターゲット・アドレスは次のように計算されます。

注意　カーネルが実行プロセスに対し違法命令/予約オペランド障害信号を通知するようなエラーは，致命的なエラーです。

保守／廃止

図 10－10　分岐ターゲット・アドレス



ターゲット・アドレス＝（EPC ＋4）＋（オフセット×4）

ターゲット・アドレスは，分岐命令のアドレスではなく，遅延スロットに含まれている命令のアドレスからの相対アドレスです．分岐ターゲット・アドレス計算方法の詳細については，**7.5　ジャンプ／ブランチ命令**を参照してください．

保守／廃止

10.3.7　リセット例外

●原因

この例外は，V_R3800のRESET信号がアクティブになったあとにインアクティブになると発生します。

●処理

V_R3800は，この例外に対し特別な割り込みベクタ（BFC0　0000H）を用意しています。リセット・ベクタはキャッシュを使用しないアドレス空間に存在しています。したがって，ハードウエアは，この例外を処理するためにキャッシュを初期設定する必要はありません。プロセッサは，キャッシュが未定義状態でも命令を取り出し実行できます。

V_R3800内のすべてのレジスタの内容は，この例外が発生する場合は未定義です」ただし，以下の内容は定義されています。

- ステータス・レジスタのSWc，KUcおよびIEcビットは，クリア（0）されています。
- ステータス・レジスタのBEVビットは，セット（1）されています。

●操作

リセット例外の操作は，すべてのプロセッサ・レジスタ，コプロセッサ・レジスタ・キャッシュ，およびメモリ・システムを初期設定することで行われます。初期設定すると，診断機能が実行されてオペレーティング・システムがブートストラップされます。リセット例外ベクタは，キャッシュが未定義状態でも命令を取り出して実行できるように，キャッシュを使用しないメモリ空間に置かれます。

保守／廃止

10.3.8　システム・コール例外

●原因

この例外は，V_R3800がSYSCALL命令を実行すると発生します。

●処理

この例外が発生すると，V_R3800は一般例外ベクタ（8000　0080H）へ分岐し，原因レジスタのExcCodeフィールドにSysコードをセットします。

EPCレジスタは，SYSCALL命令が分岐遅延スロット内の命令でないかぎり，例外を引き起こしたSYSCALL命令を指しています。しかし，例外を起こしたSYSCALL命令が分岐遅延スロット内の命令の場合には，EPCレジスタは分岐命令を指し，原因レジスタのBDビットをセットします。

V_R3800は，ステータス・レジスタのKUp，IEp，KUcおよびIEcビットをKUo，IEo，KUpおよびIEpビットにそれぞれ保存し，KUcビットとIEcビットをクリアします。

●操作

オペレーティング・システムは，該当するシステム・ルーチンに制御権を渡します。実行を再開する場合は，SYSCALL命令が再び実行されないようにEPCレジスタ内のアドレスを変更します。そのためには，制御権を返す前に，EPCレジスタを読み出して4を加えてください。

備考　SYSCALL命令が分岐遅延スロットに含まれている場合は，実行を再開するために分岐命令を解釈しなければなりません。

保守／廃止

第11章　命令セット

本章では，V_R3800の各命令の詳細について説明します．命令はアルファベット順に並べてあります．

なお，V_R3800の命令セットはコプロセッサとシステム制御コプロセッサのサポートを除いてV_R3000Aと完全互換です．

各命令を実行した場合に発生する例外については，各命令の説明のあとに列挙してあります。例外の直接の原因と処理方法についての説明は，省いてあります，例外とその処理方法の詳細については，**第10章　例外処理**を参照してください。

各命令の定数フィールドは**付録　命令コード表**を参照してください

11.1　命令クラス

V_R3800の命令は，次のクラスに分けられます．

(1) ロード／ストア命令は，データをメモリと汎用レジスタ間で転送します．これは，すべてIタイプの命令で，サポートされているアドレシング・モードはベース・レジスタ＋16ビット・ディスプレースメントだけです．

(2) 演算命令は，レジスタに対して，算術，論理およびシフト操作を実行します。そのため演算命令は，（両オペランドがレジスタの）Rタイプか（1つのオペランドが16ビット・イミーディエトの）Iタイプの命令だけです。

(3) 無条件分岐命令と分岐命令は，プログラムの制御フローを変更します。無条件分岐命令は，絶対26ビット・ワード・アドレス（Jタイプ）か，または32ビット・レジスタ・アドレス（Rタイプ）に対して分岐します。分岐命令には，プログラム・カウンタ（遅延スロット内の命令のアドレス）に対して16ビット・オフセット（Iタイプ）の和を求めプログラム・カウンタに対して相対的な分岐をします．Jump and Link命令（JAL，JALR，BLTZAL，BGEZALなど）は，リンク・レジスタr31にリターン・アドレスをセーブします．

(4) コプロセッサ命令は，ライト・バッファの状態を調べるために使用します．コプロセッサ・ゼロ（CP0）命令は，例外処理を行います。

(5) 特殊命令は，特殊レジスタと汎用レジスタ間のデータ転送，およびシステム・コール，ブレークポイントの各種タスクを実行します。これは，常にRタイプです．

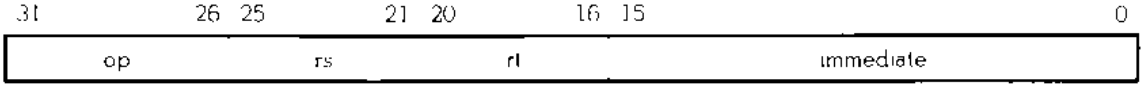
保守／廃止

11.2　命令形式

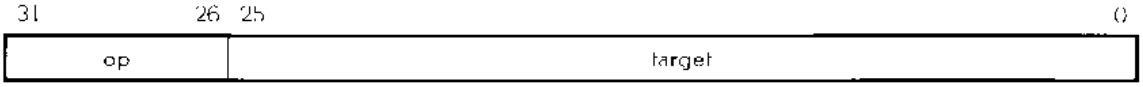
V_R3800の命令は，すべてワード境界に位置合わせされた32ビット・ワードで構成され，命令形式は図11－1に示す3つです。

図11－1　命令形式

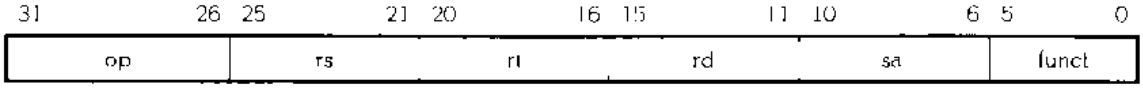
I タイプ（イミーディエト）



J タイプ（分岐）



R タイプ（レジスタ）



備考　変数サブフィールドは，以下のとおりです。

op	6ビットの命令コード
rs	5ビットのソース・レジスタ指定子
rt	5ビットのターゲット（ソース／デスティネーション）・レジスタ，または分岐条件
immediate	16ビットのイミーディエト，分岐ディスプレースメント，またはアドレス・ディスプレースメント
target	26ビットの無条件分岐ターゲット・アドレス
rd	5ビットのデスティネーション・レジスタ指定子
sa	5ビットのシフト量
funct	6ビットの機能フィールド

保守／廃止

11.3　命令表記法

本節では，命令形式内の変数サブフィールド（rs，rt，immediateなど）はすべて小文字で示します。

分かりやすくするために，特定の命令形式における変数サブフィールドに別名を使用する場合があります。たとえば，ロード命令とストア命令の形式では，rsをbaseとして使用しています。このような別名は，変数サブフィールドを指しますので，常に小文字で記述されます。

これから行う説明の「オペレーション」では，各命令で実行されるオペレーションを高級言語で記述してあります。その記述で使用されている記号の意味を，表11－1に示します。

保守／廃止

表 11－1　オペレーション表記法

記　号	意　　味
←	割り当て
‖	ビット・ストリング連結
X^y	ビット・ストリングxをyビットのストリングで反復する．xは常に1ビットの値です．
$X_{y:z}$	ビット・ストリングxについてビットyからzまでの選択。リトル・エンディアン・ビット表記法が常に使用されます　yがz未満の場合には，この表現式は空（ゼロの長さ）のビット・ストリングになります．
+	2の補数加算
−	2の補数減算
*	2の補数乗算
div	2の補数整数除算
mod	2の補数モジュロ
<	2の補数が小さいかの比較
and	ビットごとの論理積
or	ビットごとの論理和
xor	ビットごとの排他的論理和
nor	ビットごとの否定論理和
GPR [x]	汎用レジスタx．GPR [0] の内容は常にゼロです．GPH [0]　の内容は変更できません．
CPR [z，x]	コプロセッサ・ユニットz，汎用レジスタx
CCR [z，x]	コプロセッサ・ユニットz，制御レジスタx
COC　z	コプロセッサ・ユニットz，条件記号
BigEndian	リセット時または，CP0のコントロール・ステータス・レジスタREビットによって設定されたBigEndianモードです．
T + i	演算間の時間的ステップ（CPUサイクル）を示します。T + i で行われると識別される演算は，命令が呼び出された演算の次のサイクルで実行されます　このタイプの演算はロード，ストア，無条件分岐，分岐およびコプロセッサ命令によって行われます．
vAdd	仮想アドレス
pAdd	物理アドレス

保守／廃止

命令表記の例

次に示すのは実際の命令表記の例です。

例 1.

GPR [rt] ← immediate || 0^{16}

16個のゼロ・ビットとイミーディエト値（標準16ビット）とを連結し，32ビットの（下位16ビットがゼロにセットされた）ビット列を汎用レジスタrtに割り当てます。

例 2.

$(immediate_{15})^{16} \parallel immediate_{15..0}$

イミーディエト値の第15ビット（符号ビット）を16ビットに拡張し，その結果をイミーディエト値の第15ビットから０ビットと連結して32ビット符号付き値を生成しています。

保守／廃止

11.4　命令クラス概要

11.4.1　ロード／ストア命令

ロード・オペレーションには，常に1命令分の待ち時間が発生します。つまり，ロード命令の直後の命令は，メモリからフェッチされてレジスタにロードされたデータを使用できません。LWR命令またはLWL命令のターゲット・レジスタは例外で，その直前のロード命令でデスティネーションとして使用されるレジスタを指定できます。

ロード／ストア・オペレーションの指定では，表11－2にある機能が仮想アドレスと物理メモリの処理をするのに使用されます。

表11－2　ロード／ストア共通関数

機　能	説　明
アドレス変換注1	仮想アドレスから物理アドレスに変換します。
ロード・メモリ注2	キャッシュおよびメイン・メモリ上の，指定された物理アドレスのワード・データを読み出します。アドレスの下位2ビットとアクセス・タイプの0，1によって，ワード・データ内の4バイトのどのバイトを読み出すかが分かります。このときキャッシュが使用可能であればメイン・メモリからワード全体が読み出され，キャッシュにロードされます。
ストア・メモリ注3	キャッシュ，およびメイン・メモリを使って，データとして指定されたワードまたはワードの一部を指定された物理アドレスへ格納します。アドレスの下位2ビットとアクセス・タイプの0，1によってワード内の4バイトのうち格納されるバイトを示します。

注1．AddressTranslation（vAddr，I or D）

2．LoadMemory（uncached，accessType，pAddr，vAddr，I or D）

3．StoreMemory（uncached，accessType，memoryWord，pAddr，vAddr，I or D）

保守／廃止

11.4.2　ジャンプ／ブランチ命令

ジャンプ命令とブランチ命令には，１命令の遅延が発生します。つまり，ジャンプ命令またはブランチ命令の直後の命令（遅延スロットを占有しているもの）は，ターゲット命令がメモリからフェッチされている間に実行されます．遅延スロット内に，ジャンプ命令またはブランチ命令を置くことはできません．しかし，この種のエラーは検出されず，そのようなオペレーションの結果は定義されていません。

例外または割り込みが遅延スロット内の正当な命令の完了を妨げる場合には，V_R3800はその直前のジャンプ命令またはブランチ命令を指すようにEPCレジスタをセットします．コード処理が再起動されたとき，例外が引き起こした命令か，または命令が分岐遅延スロット内で実行されていた場合は，遅延スロットの直前の分岐命令から再実行されます．

ジャンプ命令とブランチ命令は，例外や割り込みのあとから再起動される場合があるので，再起動可能になっていなければなりません。したがって，ジャンプ命令やブランチ命令が戻り先アドレスを保持しているとき，レジスタ31をソース・レジスタとして使用できないこともあります。

分岐先はワード単位で位置合わせされていなければならないため，レジスタ・ジャンプ命令またはレジスタ・ジャンプ・リンク命令は下位２ビットがゼロのレジスタを使用しなければなりません。これらの下位ビットがゼロでない場合には，ジャンプ・ターゲット命令をフェッチするときにアドレス・エラー例外が発生します。

保守／廃止

11.4.3　コプロセッサ命令

V_Rシリーズ™・アーキテクチャは，4つのコプロセッサ・ユニットの定義ができます。

しかしV_R3800では，FPUなどの外付けのコプロセッサを接続できません。このため，コプロセッサ命令は実行できません。

CP1-CP3のコプロセッサにアクセスする命令を実行すると，次のように動作します。

・ステータス・レジスタ（Cu1-Cu3）の該当ビット＝0：コプロセッサ使用不可例外発生

〃　　＝1：未定義

ただし，ライト・バッファの状態を調べるためにBC0T，BC0F命令だけ使用できます。

11.4.4　システム制御コプロセッサ（CP0）命令

V_R3800に組み込まれたシステム制御コプロセッサ（CP0）のオペレーションについては，特別の制限が設けられています。一般にデータをコプロセッサから転送したりコプロセッサに転送したりするロード／ストア命令や，制御権を移動する命令はV_R3800アーキテクチャで認められていますが，CP0は例外処理とメモリ管理を担当しているために，保護状態にあります。したがって，コプロセッサへの制御転送命令とコプロセッサからの制御転送命令とがCP0レジスタの読み取りや書き込みをするための唯一の有効な手段です。

コプロセッサのオペレーション命令によってCP0がユーザ・モードや割り込み可能状態に戻る準備のために，オペレーティング・モードを変更します。

11.5　命　　令

本章では，V_R3800の各命令のオペレーションの詳細について説明します。命令はアルファベット順に並べてあります。

各命令を実行した場合に発生しうる例外は，各命令の説明のあとに列挙してあります。

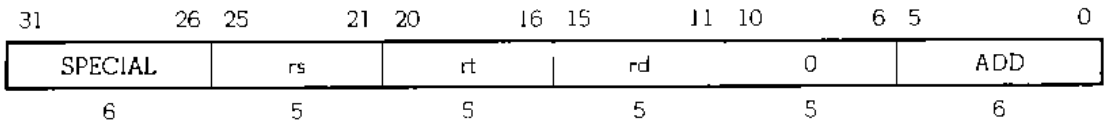
付録　命令コード表は各命令の定数フィールドのビット符号化を示しています。

またV_R3800ではTLBによるアドレス変換機構を削除しているので，TLBに関連するレジスタを操作する各命令を使用できません。これらの命令を実行したときの動作は未定義です。また，実行しても予約例外は発生しません。

保守／廃止

ADD命令

Add



命令形式：

ADD rd, rs, rt

説　　明：

汎用レジスタrsの内容と汎用レジスタrtの内容を加算し，結果を汎用レジスタrdに格納します。

オーバフローが発生すると，例外が発生します。

オペレーション：

```
T:      GPR[rd] ← GPR[rs] + GPR[rt]
```

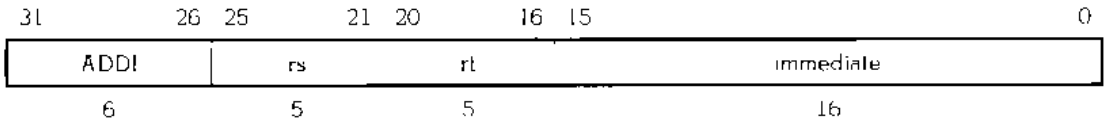
例　　外：

オーバフロー例外

保守／廃止

ADDI命令

Add Immediate



命令形式：

ADDI rt, rs, immediate

説　　明：

16ビットimmediateを32ビットに符号拡張して汎用レジスタrsの内容に加算し，結果を汎用レジスタrtに格納します．

オーバフローが発生すると，例外が発生します

オペレーション：

```
T:      GPR[rt] ← GPR[rs] + (immediate15)16 || immediate15..0
```

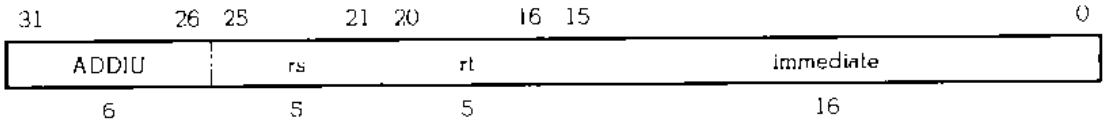
例　　外：

オーバフロー例外

保守／廃止

ADDIU命令

Add Immediate
Unsigned



命令形式：

ADDIU rt, rs, immediate

説　明：

16ビットimmediateを32ビットに符号拡張して汎用レジスタrsの内容に加算し，結果を汎用レジスタrtに格納します。

ADDIU命令はADDI命令と違い，どのような場合でもオーバフロー例外を発生しません。

オペレーション：

```
T:      GPR[rt] ← GPR[rs] + (immediate15)^16 | immediate15..0
```

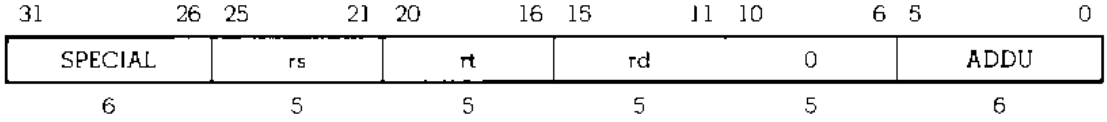
例　外：

なし

保守／廃止

ADDU命令

Add Unsigned



命令形式：

ADDU rd，rs，rt

説　　明：

汎用レジスタrsの内容と汎用レジスタrtの内容を加算し，結果を汎用レジスタrdに格納します。

ADDU命令はADD命令と違い，どのような場合でもオーバフロー例外を発生しません。

オペレーション：

```
T:      GPR[rd]←GPR[rs]+GPR[rt]
```

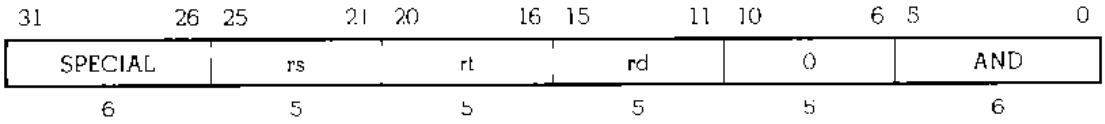
例　　外：

なし

保守／廃止

AND命令

And



命令形式：

AND rd, rs, rt

説　明：

汎用レジスタrsの内容と汎用レジスタrtの内容と論理積をとり，結果を汎用レジスタrdに格納します。

オペレーション：

```
T:      GPR[rd]← GPR[rs] and GPR[rt]
```

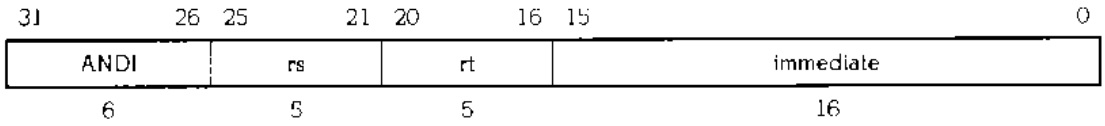
例　外：

なし

保守／廃止

ANDI命令

And Immediate



命令形式：

ANDI rt, rs, immediate

説　明：

16ビットのイミーディエトを32ビットにゼロ拡張して，汎用レジスタrsの内容と論理積をとり，結果を汎用レジスタrtに格納します。

オペレーション：

```
T:      GPR[rt] ← 0^16 || (immediate and GPR[rs]15..0)
```

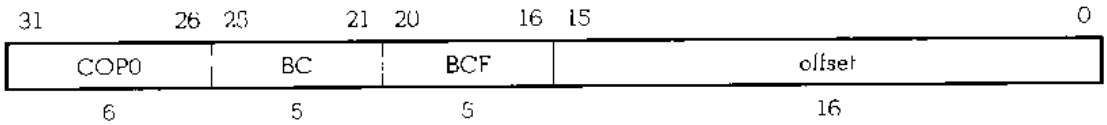
例　外：

なし

保守／廃止

BC0F命令

Branch On
Coprocessor0 False



命令形式：

BC0F offset

説　　明：

16ビットoffsetの下位にゼロを2ビット追加した18ビット・オフセットと，分岐遅延スロット内の命令のアドレスとの和から，分岐ターゲット・アドレスを計算します。

ライト・バッファにデータがある（CpCond0＝0）場合には，プログラムは1命令の遅れでターゲット・アドレスへ分岐します。

オペレーション

```
T－1 :   condition← not COC[0]
T :      target← (offset15)^14 || offset || 0^2
T＋1 :   if condition then
             PC← PC＋target
         endif
```

例　　外：

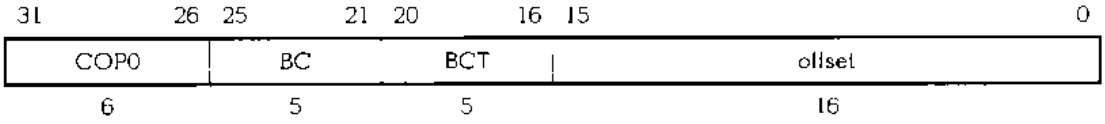
コプロセッサ使用不可例外

注意　V_R3800では，V_R3000AのBCzF命令で使用するCpCond端子が外部に出ていません。CpCond0だけが，ライト・バッファの状態を調べるために内部接続されています。また，BCzF命令でコプロセッサ1-3を指定することはできません。

保守／廃止

BC0T命令

Branch On
Coprocessor0 True



命令形式：

BC0T offset

説　　明：

16ビットoffsetの下位にゼロを２ビット追加した18ビット・オフセットと，分岐遅延スロット内の命令のアドレスとの和から，分岐ターゲット・アドレスを計算します。

ライト・バッファにデータがない（CpCond0＝１）場合には，プログラムは１命令の遅れでターゲット・アドレスへ分岐します。

オペレーション：

```
T－1    condition←COC[0]
T :     target← (offset15)^14 || offset || 0^2
T+1 :   if condition then
            PC← PC+target
        endif
```

例　　外：

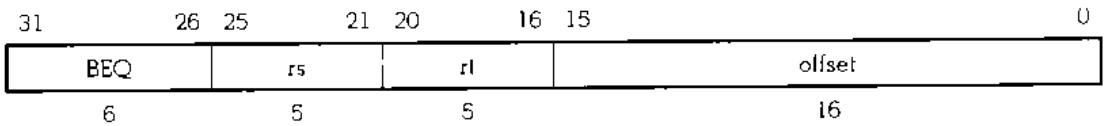
コプロセッサ使用不可例外

注意　V_R3800では，V_R3000AのBCzT命令で使用するCpCond端子が外部に出ていません。CpCond0だけが，ライト・バッファの状態を調べるために内部接続されています。また，BCzT命令でコプロセッサ1-3を指定することはできません。

保守／廃止

BEQ命令

Branch On Equal



命令形式：

BEQ rs, rt, offset

説　明：

16ビットoffsetの下位にゼロを2ビット追加した18ビット・オフセットと，分岐遅延スロット内の命令のアドレスとの和から，分岐ターゲット・アドレスを計算します。

汎用レジスタrsの内容と汎用レジスタrtの内容を比較し，2つのレジスタが等しい場合には，プログラムは1命令の遅れでターゲット・アドレスへ分岐します。

オペレーション：

```
T:      target ← (offset15)^14 || offset || 0^2
        condition ← (GPR[rs] = GPR[rt])
T+1:    if condition then
            PC ← PC + target
        endif
```

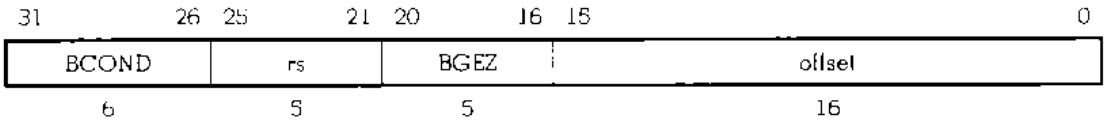
例　外：

なし

保守／廃止

BGEZ命令

Branch On Greater Than Or Equal To Zero



命令形式：

BGEZ rs，offset

説　明：

16ビットoffsetの下位にゼロを2ビット追加した18ビット・オフセットと，分岐遅延スロット内の命令のアドレスとの和から，分岐ターゲット・アドレスを計算します。

汎用レジスタrsの内容の符号ビットがクリアされている場合には，プログラムは1命令の遅れでターゲット・アドレスへ分岐します。

オペレーション：

```
T:      target← (offset15)^14 || offset || 0^2
            condition← (GPR[rs]31=0)
T+1:    if condition then
            PC← PC + target
        endif
```

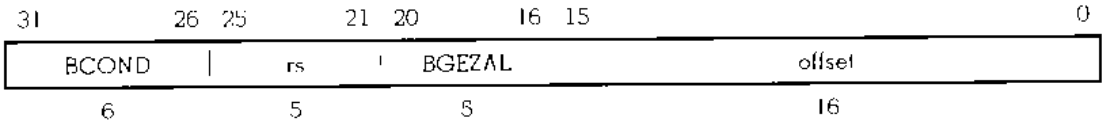
例　外：

なし

保守／廃止

BGEZAL命令

Branch On Greater Than Or Equal To Zero And Link



命令形式：

BGEZAL rs，offset

説　明：

16ビットoffsetの下位にゼロを2ビット追加した18ビット・オフセットと，分岐遅延スロット内の命令のアドレスとの和から，分岐ターゲット・アドレスを計算します。遅延スロットの次の命令のアドレスは，リンク・レジスタr31に格納します。

汎用レジスタrsの内容の符号ビットがクリアされている場合には，プログラムは1命令の遅れでターゲット・アドレスへ分岐します。

通常，汎用レジスタrsはリンク・レジスタr31以外を指定します。汎用レジスタrsにリンク・レジスタr31を指定した場合は，再実行不可能なことがあります。ただし，このような指定をしても例外は起きません。

オペレーション：

```
T:      target← (offset15)^11 || offset || 0^2
            condition← (GPR[rs]31=0)
        GPR[31]← PC+8
T+1:    if condition then
            PC← PC+target
        endif
```

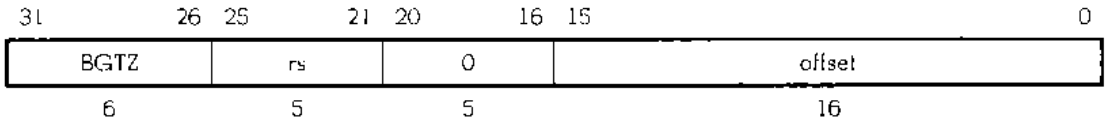
例　外：

なし

保守／廃止

BGTZ命令

Branch On Greater Than Zero



命令形式：

BGTZ rs，offset

説　明：

16ビットoffsetの下位にゼロを２ビット追加した18ビット・オフセットと，分岐遅延スロット内の命令のアドレスとの和から，分岐ターゲット・アドレスを計算します。

汎用レジスタrsの内容とゼロとを比較します。汎用レジスタrsの内容の符号ビットがクリアされていて，かつその内容がゼロでない場合には，プログラムは１命令の遅れでターゲット・アドレスへ分岐します。

オペレーション：

```
T :     target← (offset15)^14 || offset || 0^2
            condition← (GPR[rs]31=0) and (GPR[rs]=0^32)
T+1 :   if condition then
            PC← PC+target
        endif
```

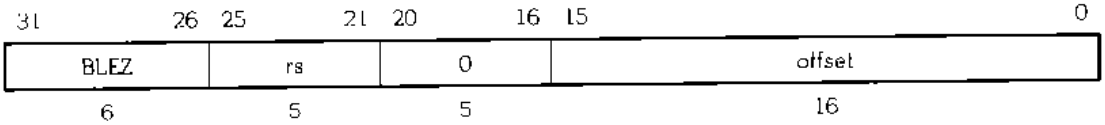
例　外：

なし

保守／廃止

BLEZ命令

Branch On Less Than
Or Equal To Zero



命令形式：

BLEZ rs, offset

説　明：

16ビットoffsetの下位にゼロを2ビット追加した18ビット・オフセットと，分岐遅延スロット内の命令のアドレスとの和から，分岐ターゲット・アドレスを計算します。

汎用レジスタrsの内容とゼロとを比較します。汎用レジスタrsの内容の符号ビットがセットされているか，またはその内容がゼロに等しい場合には，プログラムは1命令の遅れでターゲット・アドレスへ分岐します。

オペレーション：

```
T:      target← (offset15)^14 || offset || 0^2
            condition← (GPR[rs]31 = 1) and (GPR[rs] = 0^32)
T+1:    if condition then
            PC← PC + target
        endif
```

例　外：

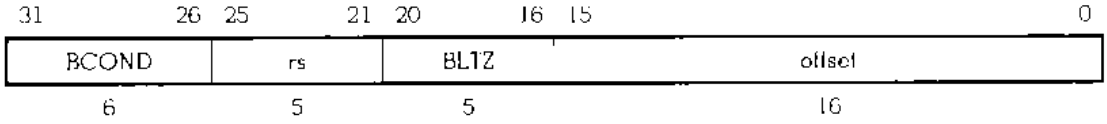
なし

11

保守／廃止

BLTZ命令

Branch On Less Than Zero



命令形式：

BLTZ rs, offset

説　明：

16ビットoffsetの下位にゼロを２ビット追加した18ビット・オフセットと，分岐遅延スロット内の命令のアドレスとの和から，分岐ターゲット・アドレスを計算します。

汎用レジスタrsの内容の符号ビットがセットされている場合には，プログラムは１命令の遅れでターゲット・アドレスへ分岐します。

オペレーション：

```
T:      target ← (offset15)^14 || offset || 0^2
            condition ← (GPR[rs]31 = 1)
T+1:    if condition then
            PC ← PC + target
        endif
```

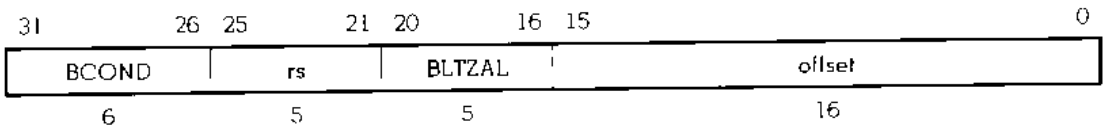
例　外：

なし

保守／廃止

BLTZAL命令

Branch On Less Than Zero And Link



命令形式：

BLTZAL rs, offset

説　明：

16ビットoffsetの下位にゼロを2ビット追加した18ビット・オフセットと，分岐遅延スロット内の命令のアドレスとの和から，分岐ターゲット・アドレスを計算します。遅延スロットのあとの命令のアドレスは，無条件にリンク・レジスタr31に格納します。

汎用レジスタrsの内容の符号ビットがセットされている場合には，プログラムは1命令の遅れでターゲット・アドレスへ分岐します。

通常，汎用レジスタrsはリンク・レジスタr31以外を指定します。リンク・レジスタr31を指定する命令は，再実行不可能なことがあります。ただし，このような指定をしても例外は起きません。

オペレーション：

```
T:      target← (offset15)^14 || offset || 0^2
            condition← (GPR[rs]31 = 1)
            GPR[31]←PC+8
T+1:    if condition then
            PC← PC+target
        endif
```

例　外：

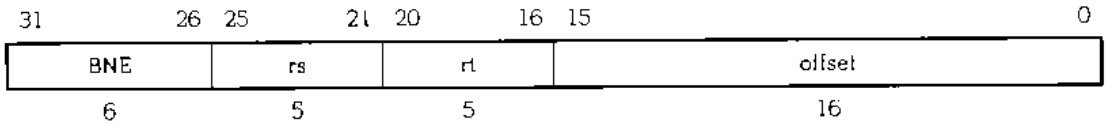
なし

11

保守／廃止

BNE命令

Branch On Not Equal



命令形式：

BNE rs，rt，offset

説　明：

16ビットoffsetの下位にゼロを2ビット追加した18ビット・オフセットと分岐遅延スロット内の命令のアドレスとの和から，分岐ターゲット・アドレスを計算します。

汎用レジスタrsの内容と汎用レジスタrtの内容を，比較します。2つのレジスタが等しくない場合には，プログラムは1命令の遅れでターゲット・アドレスへ分岐します。

オペレーション：

```
T:      target ← (offset15)^14 | offset | 0^2
            condition ← (GPR[rs] ≠ GPR[rt])
T+1:    if condition then
            PC ← PC + target
        endif
```

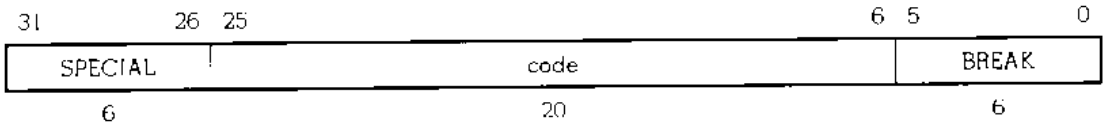
例　外：

なし

保守／廃止

BREAK命令

Break



命令形式：

BREAK code

説　明：

ブレークポイント例外が発生して，無条件に制御権を例外ハンドラに移動します．

コード・フィールドは，ソフトウエア・パラメータとして使用することが可能ですが，例外ハンドラがこれを取り出すためには，命令のあるメモリの内容をロードしなければなりません．

オペレーション：

```
T:      BreakpointException
```

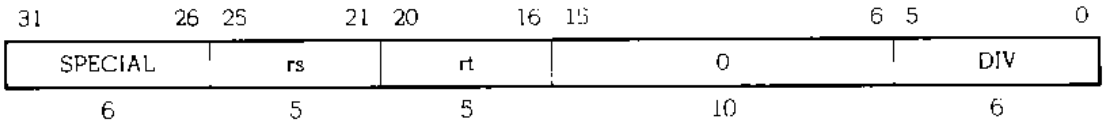
例　外：

ブレークポイント例外

保守／廃止

DIV命令

Divide



命令形式：

DIV rs, rt

説　明：

汎用レジスタrsの内容と汎用レジスタrtの内容を，32ビットの２の補数として扱い，汎用レジスタrsの内容を汎用レジスタrtの内容で除算します。どのような場合でも，オーバフロー例外は発生しません。

演算結果の商は特別レジスタLOに，剰余は特別レジスタHIにロードします。演算が終了する前にMFHI命令とMFLO命令で，特別レジスタLOまたはHIを読み取ろうとしてもインタロックされていて，演算が終了するまでそれらの命令を遅らせるようになっています。

乗除算命令は，V_R3800内の独立した別ユニットによって実行するため，除算オペレーションが開始されたあとも，他の命令の実行は並行して継続できます。そのため乗除算のユニットは，通常命令の実行ができなくなるキャッシュ・ミスやその他の遅延サイクルの間も命令の実行を継続し続けます。

オペレーション：

```
T-2 :   LO←undefined
        HI← undefined
T-1 :   LO← undefined
        HI←undefined
T :     LO← GPR[rs] div GPR[rt]
        HI←GPR[rs] mod GPR[rt]
```

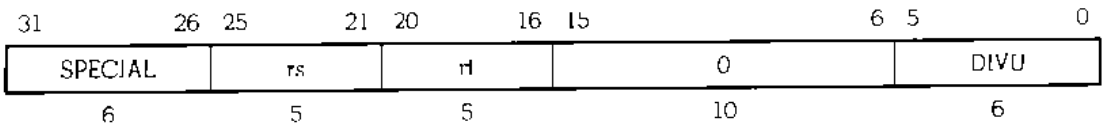
例　外：

なし

保守／廃止

DIVU命令

Divide Unsigned



命令形式：

DIVU rs, rt

説　　明：

汎用レジスタrsの内容と汎用レジスタrtの内容を，32ビットの符号なし数として扱い，汎用レジスタrsの内容を汎用レジスタrtの内容で除算します。どのような場合でも，オーバフロー例外は発生しません。

演算結果の商は特別レジスタLOに，剰余は特別レジスタHIにロードします。演算が終了する前にMFHI命令とMFLO命令で，特別レジスタLOまたはHIを読み取ろうとしてもインタロックされていて，演算が終了するまでそれらの命令を遅らせるようになっています。

乗除算命令は，V_R3800内の独立した別ユニットによって実行するため，除算オペレーションが開始されたあとも，他の命令の実行は並行して継続できます。そのため乗除算ユニットは，通常命令の実行ができなくなるキャッシュ・ミスや，その他の遅延サイクルの間も命令の実行を継続し続けます。

オペレーション：

```
T-2 : LO ← undefined
      HI ← undefined
T-1 : LO ← undefined
      HI ← undefined
T   : LO ← (0 || GPR[rs]) div (0 || GPR[rt])
      HI ← (0 || GPR[rs]) mod (0 || GPR[rt])
```

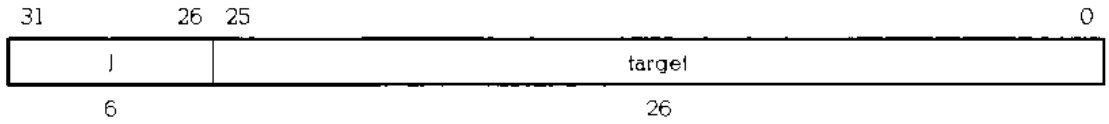
例　　外：

なし

保守／廃止

J命令

Jump



命令形式：

J target

説　　明：

26ビットのターゲット・アドレスを２ビット左にシフトし，現在のプログラム・カウンタの上位４ビットと結合し分岐アドレスとします。結合した分岐アドレスへ，１命令の遅れで無条件に分岐します。

オペレーション：

```
T:      temp ← target
T+1:    PC ← PC31..28 || temp || 0^2
```

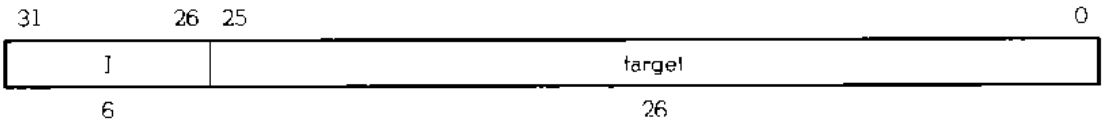
例　　外：

なし

保守／廃止

JAL命令

Jump And Link



命令形式：

JAL target

説　明：

26ビットのターゲット・アドレスを2ビット左にシフトして現在のプログラム・カウンタの上位4ビットと結合し，分岐アドレスとします。結合した分岐アドレスへ，1命令の遅れで無条件に分岐します。遅延スロットのあとの命令のアドレスは，リンク・レジスタr31に格納します。

オペレーション：

```
T:      temp← target
        GPR[31]← PC+8
T+1:    PC← PC31..28 || temp || 0^2
```

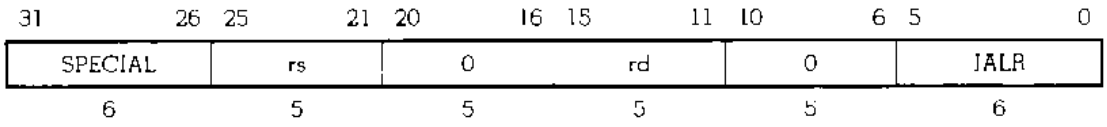
例　外：

なし

保守／廃止

JALR命令

Jump And
Link Register



命令形式：

JALR rs

JALR rd，rs

説　　明：

汎用レジスタrsで指定されたターゲット・アドレスへ，無条件に分岐します。分岐は，1命令遅れて行われます。遅延スロットのあとの命令のアドレスは，汎用レジスタrdに格納します。アセンブリ言語命令で省略してある場合，汎用レジスタrdのディフォールト値はリンク・レジスタr31です。

レジスタ指定子rsと汎用レジスタrdに同じレジスタを選択しても例外は起きませんが，こうした命令を実行した場合の結果は定義されていません。

オペレーション：

```
T:      temp← GPR[rs]
        GPR[rd]← PC+8
T+1:    PC← temp
```

例　　外：

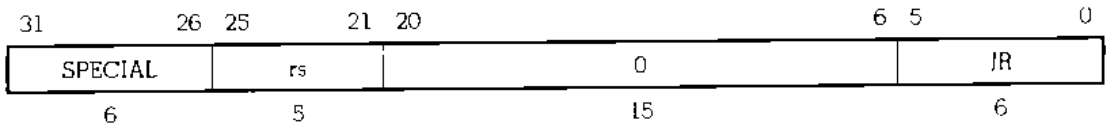
なし

命令はワード単位で位置合わせされていなければならないので，汎用レジスタrsで指定されるターゲット・アドレスの下位2ビットはゼロでなければなりません，この下位ビットがゼロでない場合には，ジャンプ先の命令をフェッチしたときにアドレス・エラー例外が発生します。

保守／廃止

JR命令

Jump Register



命令形式：

JR rs

説　　明：

汎用レジスタrsで指定されたターゲット・アドレスへ，無条件に分岐します。このとき分岐は，1命令遅れて行われます。

オペレーション：

```
T:      temp← GPR[rs]
T+1:    PC← temp
```

例　　外：

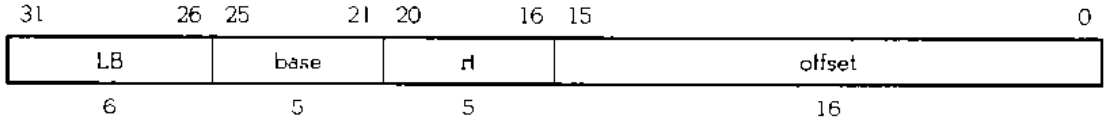
なし

命令はワード単位で位置合わせされていなければならないので，汎用レジスタrsで指定されるターゲット・アドレスの下位2ビットはゼロでなければなりません。この下位ビットがゼロでない場合には，ジャンプ先の命令をフェッチしたときにアドレス・エラー例外が発生します。

保守／廃止

LB命令

Load Byte



命令形式：

LB rt, offset (base)

説　明

16ビットoffsetを汎用レジスタbaseの内容に加算し，32ビット有効アドレスを生成します。有効アドレスで指定したメモリ位置のバイトの内容を符号拡張し，汎用レジスタrtにロードします。

★ この命令の直後の命令では，ロードしたデータを使用できません。

オペレーション：

```
T:      vAddr← (offset15)^16 || offset15..0 + GPR[base]
        (pAddr, uncached)← AddressTranslation (vAddr, DATA)
        mem← LoadMemory (uncached, BYTE, pAddr, vAddr, DATA)
        byte← vAddr1..0 xor BigEndian^2
        GPR[rt]←undefined
T+1:    GPR[rt]← (mem7+8*byte)^24 || mem7+8*byte..8*byte
```

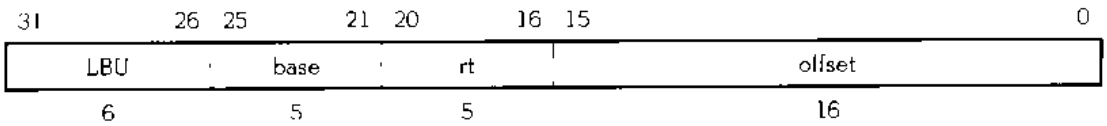
例　外：

アドレス・エラー例外

保守／廃止

LBU命令

Load Byte Unsigned



命令形式：

LBU rt, offset (base)

説　明：

16ビットoffsetを汎用レジスタbaseの内容に加算し，32ビット有効アドレスを生成します。有効アドレスで指定したメモリ位置のバイトの内容を，ゼロ拡張して汎用レジスタrtにロードします。

この命令の直後の命令では，ロードしたデータを使用できません。 ★

オペレーション：

```
T:      vAddr← (offset15)^16 || offset15..0 + GPR[base]
        (pAddr, uncached)← AddressTranslation (vAddr, DATA)
        mem← LoadMemory (uncached, BYTE, pAddr, vAddr, DATA)
        byte← vAddr1..0 xor BigEndian^2
        GPR[rt]← undefined
T+1:    GPR[rt]← 0^24 || mem7+8*byte..8*byte
```

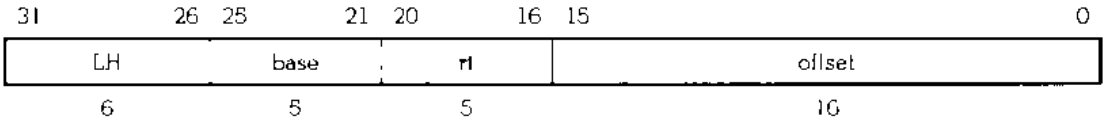
例　外：

アドレス・エラー例外

保守／廃止

LH命令

Load Halfword



命令形式：

LH rt, offset (base)

説　　明：

16ビットoffsetを汎用レジスタbaseの内容に加算し，32ビット有効アドレスを生成します。有効アドレスで指定したメモリ位置のハーフ・ワードの内容を符号拡張して，汎用レジスタrtにロードします。

★ この命令の直後の命令では，ロードしたデータを使用できません。

有効アドレスの最下位ビットがゼロでない場合には，アドレス・エラー例外が発生します。

オペレーション：

```
T:      vAddr ← (offset15)^16 || offset15..0 + GPR[base]
        (pAddr, uncached) ← AddressTranslation(vAddr, DATA)
        mem ← LoadMemory(uncached, HALFWORD, pAddr, vAddr, DATA)
        byte ← vAddr1..0 xor (BigEndian || 0)
        GPR[rt] ← undefined
T+1:    GPR[rt] ← (mem15+8*byte..8*byte)^16 || mem15+8*byte..8*byte
```

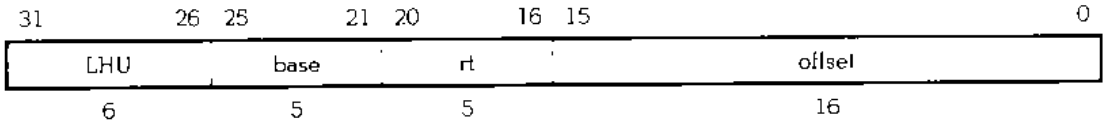
例　　外：

アドレス・エラー例外

保守／廃止

LHU命令

Load Halfword
Unsigned



命令形式：

LHU rt, offset (base)

説　明：

16ビットoffsetを汎用レジスタbaseの内容に加算し，32ビット有効アドレスを生成します。有効アドレスが指定したメモリ位置のハーフ・ワードの内容をゼロ拡張して，汎用レジスタrtにロードします。

この命令の直後の命令では，ロードしたデータを使用できません。 ★

有効アドレスの最下位ビットがゼロでない場合には，アドレス・エラー例外が発生します。

オペレーション：

```
T:      vAddr← (offset15)^16 || offset15..0 + GPR[base]
        (pAddr, uncached)← AddressTranslation(vAddr, DATA)
        mem←LoadMemory(uncached, HALFWORD, pAddr, vAddr, DATA)
        byte←vAddr1..0 xor (BigEndian || 0)
        GPR[rt]← undefined
T+1:    GPR[rt]←0^16 || mem15+8*byte..8*byte
```

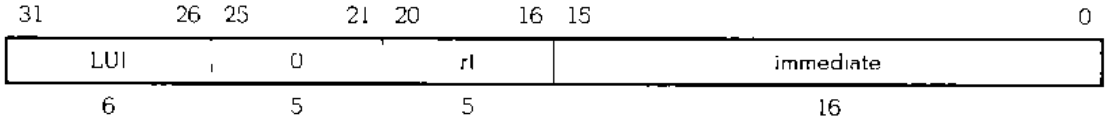
例　外：

アドレス・エラー例外

保守／廃止

LUI命令

Load Upper Immediate



命令形式：

LUI rt, immediate

説　明：

16ビットimmediateを上位16ビット左にシフトして，下位16ビットにゼロを代入し汎用レジスタrtに格納します．

オペレーション：

```
T:      GPR[rt] ← immediate || 0^16
```

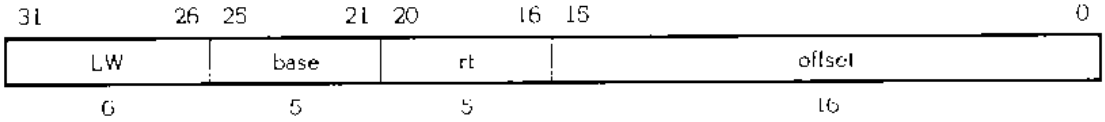
例　外：

なし

保守／廃止

LW命令

Load Word



命令形式：

LW rt, offset (base)

説　明：

16ビットoffsetを汎用レジスタbaseの内容に加算し，32ビット有効アドレスを生成します．有効アドレスが指定したメモリ位置のワードの内容を，汎用レジスタrtにロードします。

この命令の直後の命令では，ロードしたデータを使用できません。 ★

有効アドレスの下位２ビットがゼロでない場合には，アドレス・エラー例外が発生します。

オペレーション：

```
T:      vAddr← (offset15)^16 || offset15..0 + GPR[base]
        (pAddr, uncached)← AddressTranslation(vAddr, DATA)
        mem← LoadMemory(uncached, WORD, pAddr, vAddr, DATA)
        GPR[rt]← undefined
T+1:    GPR[rt]← mem
```

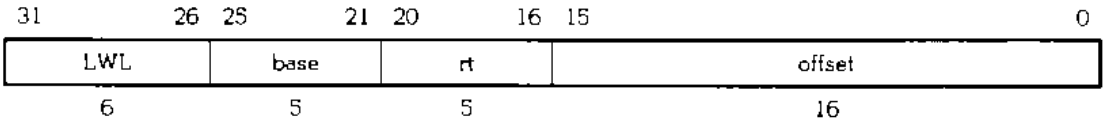
例　外：

アドレス・エラー例外

保守／廃止

LWL命令

Load Word Left



命令形式：

LWL rt, offset (base)

説　明：

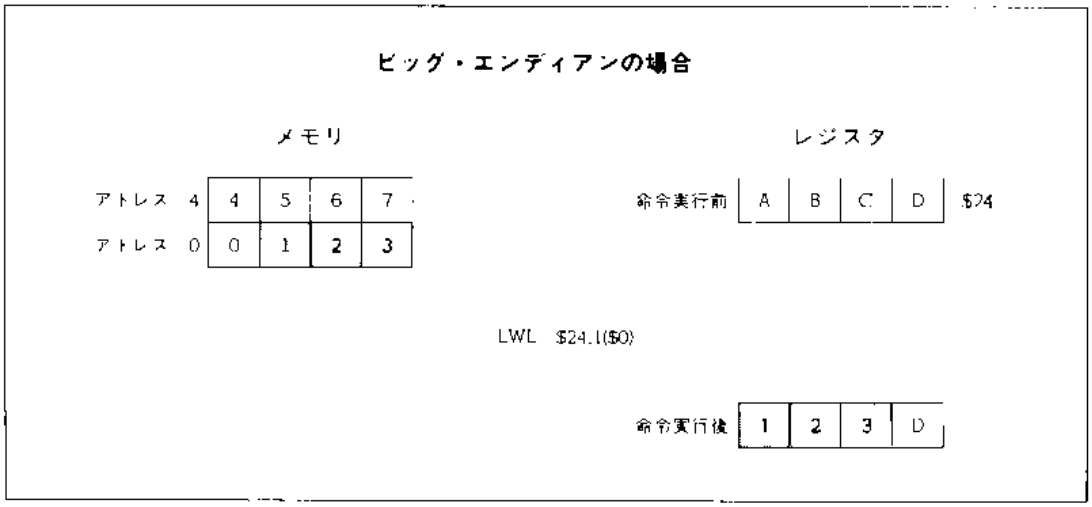
この命令は，LWR命令と一緒に使用して，ワード境界にまたがっているデータをレジスタにロードすることができます。LWL命令は，レジスタの左部分に該当部分をロードし，LWR命令はレジスタの右部分に該当部分をロードします。

LWL命令は，16ビットoffsetを汎用レジスタbaseの内容に加算し，32ビット有効アドレスを生成します。指定された先頭バイトを持つメモリ内のワードから，データを読み取ります。指定された先頭バイトの位置により，1バイトから4バイトのデータをロードします。

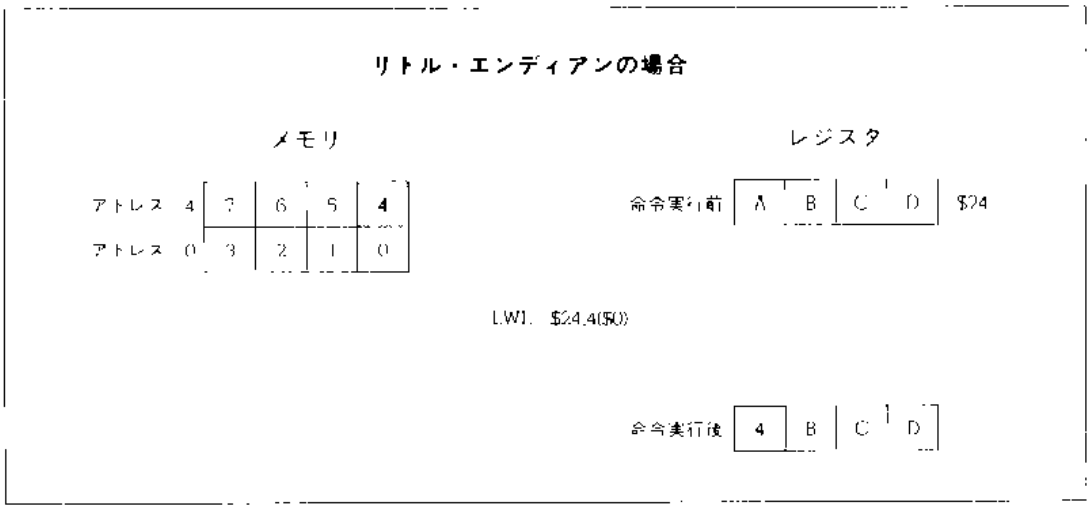
★　この命令の直後の命令では，ロードしたデータを使用できません。

汎用レジスタrtの内容はプロセッサ内部でバイパスされるため，汎用レジスタrtを指定するロード命令と，その直後のLWL命令で同じく汎用レジスタrtを指定しても，この間にはNOP命令は不要です。バイト位置合わせのためのアドレス・エラー例外は，この命令によって抑制します。

この命令は，まずメモリ内の指定されたバイトをレジスタの上位（最左端）バイトにロードし，メモリ内ワードとレジスタ内ワードの下位バイトの方に向かって順にデータのロードを行い，メモリ内ワードの最下位バイトまでそれを続けます。



保守／廃止



オペレーション：

```
T:      vAddr←(offset15)16 || offset15..0 + GPR[base]
        (pAddr, uncached)←AddressTranslation(vAddr, DATA)
        byte←vAddr1..0 xor BigEndian2
        if ¬BigEndian then
          pAddr← pAddr31..2 || 02
        endif
        mem←LoadMemory(uncached, byte, pAddr, vAddr, DATA)
T+1:    GPR[rt]←mem7+8*byte..0 | GPR[rt]23-9*byte..0
```

例　外：

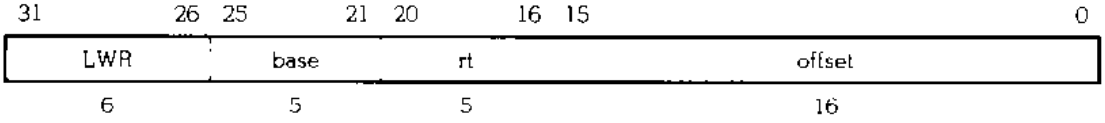
アドレス・エラー例外

11

保守／廃止

LWR命令

Load Word Right



命令形式：

LWR rt, offset (base)

説　　明：

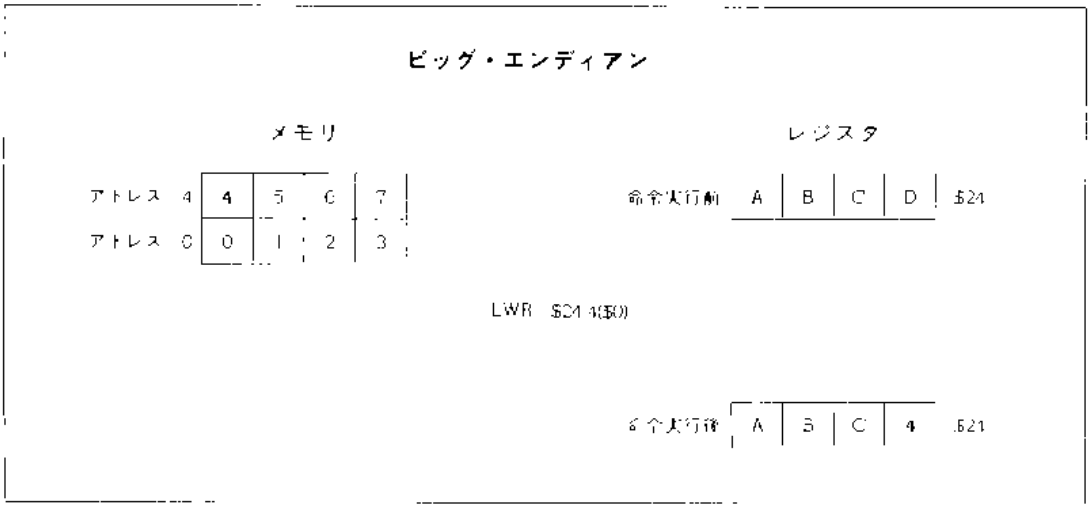
この命令は，LWL命令と一緒に使用して，ワード境界にまたがっているデータをレジスタにロードすることができます。LWR命令は，レジスタの右部分に該当部分をロードし，LWL命令はレジスタの左部分に該当部分をロードします。

LWR命令は，16ビットoffsetを汎用レジスタbaseの内容に加算し，32ビット有効アドレスを生成します。指定された先頭バイトを持つメモリ内のワードから，データを読み取ります。指定された先頭バイトの位置により，1バイトから4バイトのデータをロードします。

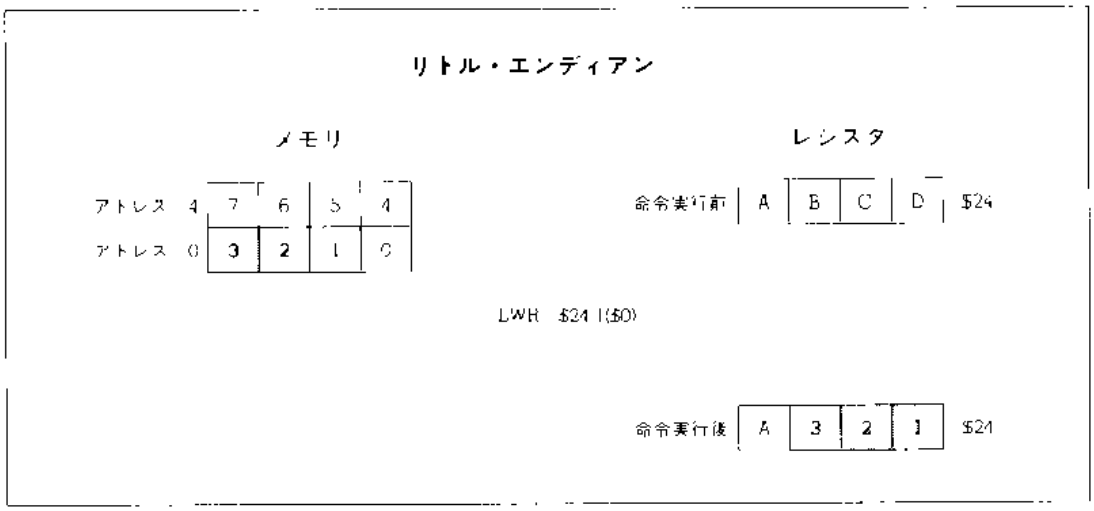
★　この命令の直後の命令では，ロードしたデータを使用できません。

汎用レジスタrtの内容はプロセッサ内部でバイパスされるため，汎用レジスタrtを指定するロード命令と，その直後のLWR命令で同じく汎用レジスタrtを指定しても，この間にはNOP命令は不要です。バイト位置合わせのためのアドレス・エラー例外は，この命令によって抑制します。

この命令は，まずメモリ内の指定されたバイトをレジスタの下位（最右端）バイトにロードし，メモリ内ワードとレジスタ内ワードの上位バイトの方に向かって順にデータのロードを行い，メモリ内のワード最上位バイトまでそれを続けます。



保守／廃止



オペレーション：

```
T:      vAddr←(offset15)^16 || offset15..0+GPR[base]
        (pAddr, uncached)←AddressTranslation(vAddr, DATA)
        byte←vAddr1..0 xor BigEndian^2
        if BigEndian then
          pAddr←pAddr31..2 || 0^2
        endif
        mem←LoadMemory(uncached, WORD-byte, pAddr, vAddrDATA)
T+1:    GPR[rt]←GPR[rt]31..32-8*byte || mem31..8*byte
```

例　外：

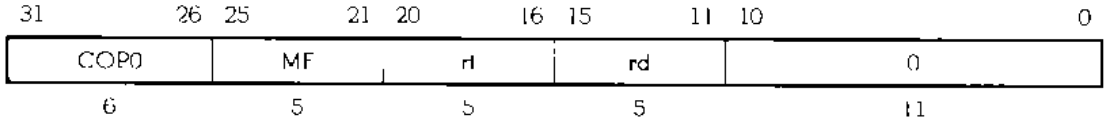
アドレス・エラー例外

11

保守／廃止

MFC0命令

Move From System
Control Coprocessor



命令形式：

MFC0 rt, rd

説　明：

システム制御コプロセッサ（CP0）のコプロセッサ・レジスタrdの内容を，汎用レジスタrtにロードします。

★ この命令の直後の命令では，ロードしたデータを使用できません。

オペレーション：

```
T:      data← CPR[0,rd]
T+1:    GPR[rt]← data
```

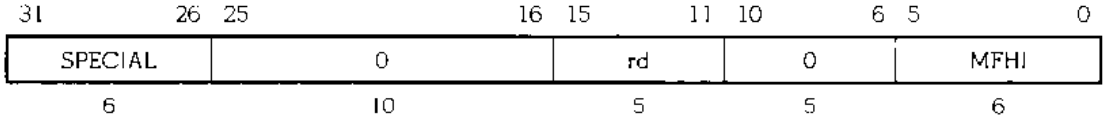
例　外：

コプロセッサ使用不可例外

保守／廃止

MFHI命令

Move From HI



命令形式：

MFHI rd

説　明：

特殊レジスタHIの内容を，汎用レジスタrdにロードします。

割り込みのあとの正しいオペレーションを保証するためには，MFHI命令に続く２つの命令には，HIレジスタの内容を変更する命令（MULT，MULTU，DIV，DIVU，MTHI）を使用しないようにしてください。

オペレーション：

```
T:      GPR[rd] ← HI
```

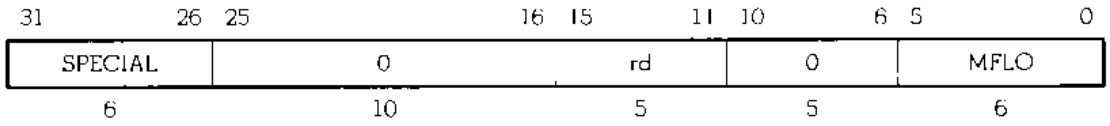
例　外：

なし

保守／廃止

MFLO命令

Move From LO



命令形式

MFLO rd

説　明

特殊レジスタLOの内容を，汎用レジスタrdにロードします。

割り込みのあとの正しいオペレーションを保証するためには，MFLO命令に続く２つの命令には，LOレジスタの内容を変更する命令（MULT，MULTU，DIV，DIVU，MTLO）を使用しないようにしてください。

オペレーション：

```
T:      GPR[rd]← LO
```

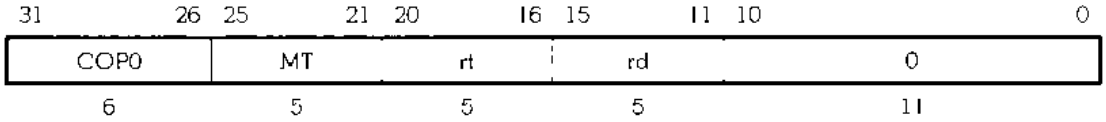
例　外：

なし

保守／廃止

MTC0命令

Move To System
Control Coprocessor



命令形式：

MTC0 rt, rd

説　明：

汎用レジスタrtの内容をシステム制御コプロセッサ（CP0）のコプロセッサ・レジスタrdにロードします。

仮想アドレス変換はこの命令で変更されるために、ロード／ストア命令のオペレーションは定義されていません。

オペレーション：

```
T:      data←GPR[rt]
T+1:    CPR[0,rd]← data
```

例　外：

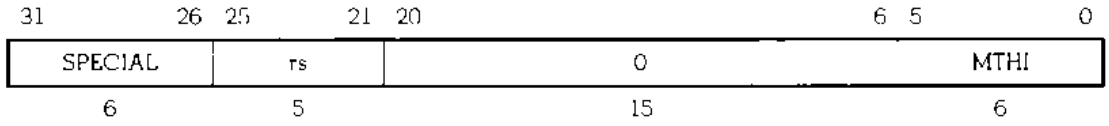
コプロセッサ使用不可例外

11

保守／廃止

MTHI命令

Move To HI



命令形式：

MTHI rs

説　　明：

汎用レジスタrsの内容を，特殊レジスタHIにロードします。

MTHI命令がMULT，MULTU，DIVあるいはDIVU命令のあとに，しかもMFLO，MFHI，MTLOあるいはMTHI命令のどれかに先だって実行される場合には，特殊レジスタLOの内容は定義されません。

オペレーション：

```
T-2 :   HI← undefined
T-1 :   HI← undefined
T :     HI← GPR[rs]
```

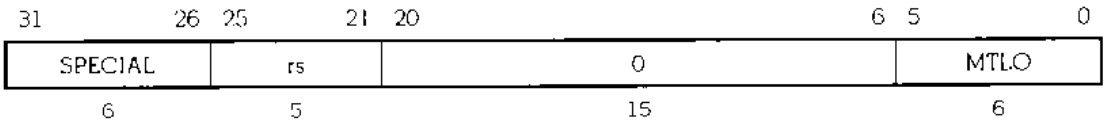
例　　外：

なし

保守／廃止

MTLO命令

Move To LO



命令形式：

MTLO rs

説　明

汎用レジスタrsの内容を，特殊レジスタLOにロードします．

MTLO命令かMULT，MULTU，DIVあるいはDIVU命令のあとに，しかもMFLO，MFHI，MTLOあるいはMTHI命令のどれかに先だって実行される場合には，特殊レジスタHIの内容は定義されません．

オペレーション：

```
T-2:    LO ← undefined
T-1:    LO ← undefined
T:      LO ← GPR[rs]
```

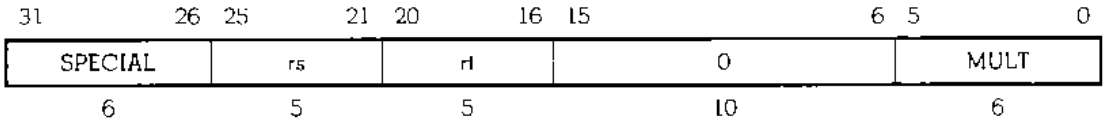
例　外：

なし

保守／廃止

MULT命令

Multiply



命令形式：

MULT rs、rt

説　明：

汎用レジスタrsの内容と汎用レジスタrtの内容を，２の補数の値として扱い，乗算します．どんな場合でも，オーバフロー例外は発生しません。

オペレーションが完了したとき，ダブル・ワードの結果の下位ワードは特殊レジスタLOにロードされ，ダブル・ワードの結果の上位ワードは特殊レジスタHIにロードされます。MFHI命令とMFLO命令は，乗算オペレーションが完了する前にこれを読み取ろうとしても，インタロックされていて実行は遅らされます。

乗算オペレーションは，V_R3800内の独立した別の実行ユニットで行われます。そのため乗算オペレーションが始まってからも，他の命令の実行は並行して継続されます。乗算ユニットや除算ユニットは、命令が実行されなくなるキャッシュ・ミスやその他の遅延サイクルの間も実行を続けます。

オペレーション：

```
T-2 :   LO←undefined
        HI←undefined
T-1 :   LO←undefined
        HI← undefined
T :     t←GPR[rs]*GPR[rt]
        LO← t31..0
        HI← t63..32
```

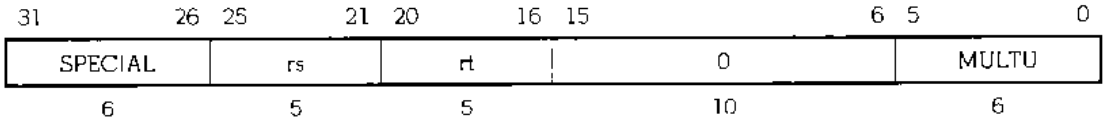
例　外：

なし

保守／廃止

MULTU命令

Multiply Unsigned



命令形式：

MULTU rs，rt

説　明：

汎用レジスタrsの内容と汎用レジスタrtの内容を，無符号値として扱い乗算します。どんな場合でも，オーバフロー例外は発生しません。

オペレーションが完了したとき，ダブル・ワードの結果の下位ワードは特殊レジスタLOにロードされ，ダブル・ワードの結果の上位ワードは特殊レジスタHIにロードされます。MFHI命令とMFLO命令は，乗算オペレーションが完了する前にこれを読み取ろうとしても，インタロックされていて命令の実行は遅らされます。

乗算オペレーションは，V_R3800内の独立した別の実行ユニットで行われます。乗算オペレーションが始まってからも，他の命令の実行は並行して継続されます」乗算ユニットや除算ユニットは，命令が実行されなくなるキャッシュ・ミスやその他の遅延サイクルの間も実行を続けます。

オペレーション：

```
T-2 :   LO←undefined
        HI←undefined
T-1 :   LO← undefined
        HI← undefined
T :     t← (0 || GPR[rs]) * (0 || GPR[rt])
        LO← t31..0
        HI← t63..32
```

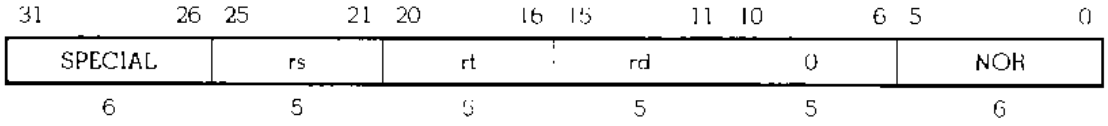
例　外：

なし

保守／廃止

NOR命令

Nor



命令形式：

NOR rd, rs, rt

説　　明：

汎用レジスタrsの内容と汎用レジスタrtの内容との否定論理和をとり，結果を汎用レジスタrdに格納します．

オペレーション：

```
T:      GPR[rd] ← GPR[rs] nor GPR[rt]
```

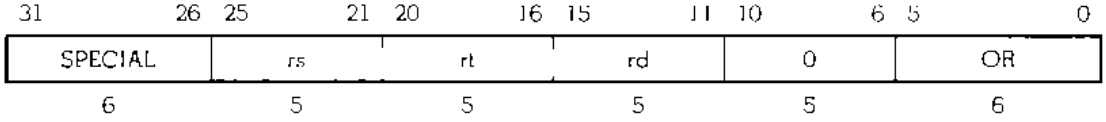
例　　外：

なし

保守／廃止

OR命令

Or



命令形式：

OR rd, rs, rt

説　明：

汎用レジスタrsの内容と汎用レジスタrtの内容との論理和をとり，結果を汎用レジスタrdに格納します。

オペレーション：

```
T:      GPR[rd] ← GPR[rs] or GPR[rt]
```

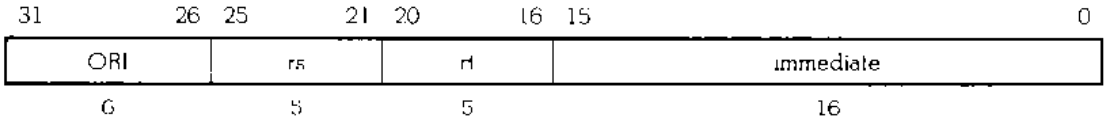
例　外：

なし

保守／廃止

ORI命令

Or Immediate



命令形式：

ORI rt, rs, immediate

説　明

16ビットimmediateをゼロ拡張し，汎用レジスタrsの内容と論理和をとり，結果を汎用レジスタrtに格納します。

オペレーション：

```
T:      GPR[rt] ← GPR[rs]31..16 | (immediate or GPR[rs]15..0)
```

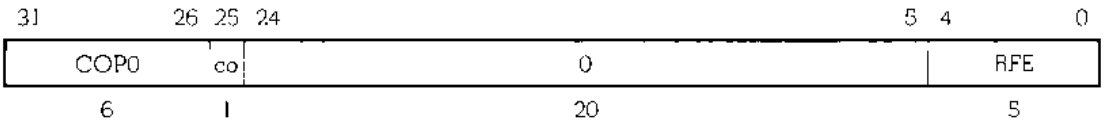
例　外：

なし

保守／廃止

RFE命令

Restore From Exception



命令形式：

RFE

説　　明：

ステータス・レジスタ（SR）の以前の割り込みマスクとカーネル・ユーザ・モード・ビット（IEpとKUp）を対応する現在のモード・ビット（IEcとKUc）に復元して，旧モード・ビット（IEoとKUo）を対応するモード・ビット（IEpとKUp）に復元します．旧モード・ビットは，変更されません

RFE命令の直前のロード／ストア命令に関連するメモリ参照のオペレーションは，指定されていません。普通．RFE命令は，PCを復元させるためにJR（無条件分岐レジスタ）命令の遅延スロットのあとに実行します。

オペレーション

```
T:      SR← SR31-4 || SR5-2
```

例　　外：

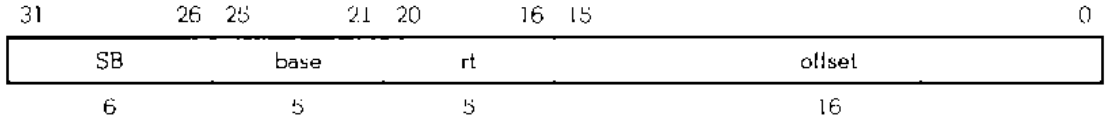
コプロセッサ使用不可例外

11

保守／廃止

SB命令

Store Byte



命令形式：

SB rt, offset (base)

説　明：

16ビットoffsetを汎用レジスタbaseの内容に加算し，32ビットの有効アドレスを生成します．有効アドレスで指定されたメモリ位置に，汎用レジスタrtの最下位バイトの内容を格納します．

オペレーション：

```
T:      vAddr← (offset15)16 || offset15..0 + GPR[base]
        (pAddr, uncached)← AddressTranslation(vAddr, DATA)
        byte← vAddr1..0 xor BigEndian2
        data← GPR'rt'31-8*byte..0 || 0^(2*byte)
        StoreMemory(uncached, BYTE, data, pAddr, vAddr, DATA)
```

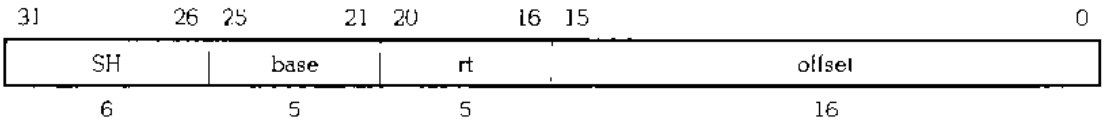
例　外：

アドレス・エラー例外

保守／廃止

SH命令

Store Halfword



命令形式：

SH rt, offset (base)

説　明：

16ビットoffsetを汎用レジスタbaseの内容に加算し，32ビットの有効アドレスを生成します。有効アドレスで指定されたメモリ位置に，汎用レジスタrtの最下位ハーフ・ワードの内容を格納します。

有効アドレスの最下位ビットがゼロでない場合には，アドレス・エラー例外が発生します。

オペレーション：

```
T:      vAddr← (offset15)^16 || offset15..0 + GPR[base]
        (pAddr, uncached)← AddressTranslation(vAddr, DATA)
        byte← vAddr1..0 xor (BigEndian || 0)
        data← GPR[rt]31-8*byte..0 || 0^8*byte
        StoreMemory(uncached, HALFWORD, data, pAddr, vAddr, DATA)
```

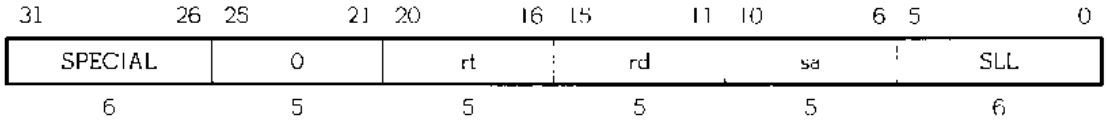
例　外：

アドレス・エラー例外

保守／廃止

SLL命令

Shift Left Logical



命令形式：

SLL rd, rt, sa

説　　明：

ゼロを下位ビットに挿入しながら，汎用レジスタrtの内容をsaビットだけ左にシフトします．結果を，汎用レジスタrdに格納します．

オペレーション：

```
T:      GPR[rd] ← GPR[rt]31-sa..0 || 0^sa
```

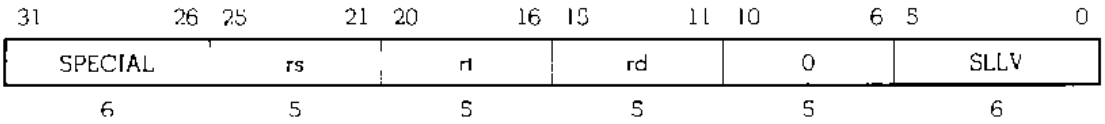
例　　外：

なし

保守／廃止

SLLV命令

Shift Left
Logical Variable



命令形式：

SLLV rd, rt, rs

説　明：

ゼロを下位ビットに挿入しながら、汎用レジスタrtの内容を汎用レジスタrsの内容の下位５ビットが指定するビット数だけ左にシフトします。結果を，汎用レジスタrdに格納します。

オペレーション：

```
T:      s ← GPR[rs]4..0
        GPR[rd] ← GPR[rt](31-s)..0 || 0s
```

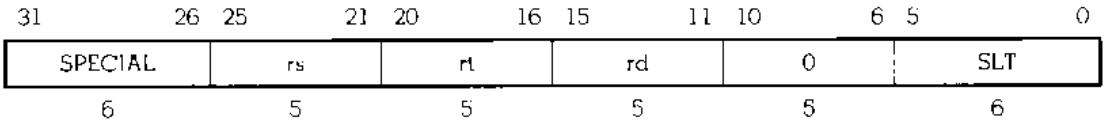
例　外

なし

保守／廃止

SLT命令

Set On Less Than



命令形式

SLT rd, rs, rt

説　　明：

汎用レジスタrtと汎用レジスタrsの内容を，符号付き32ビット整数として比較します。汎用レジスタrsの内容が汎用レジスタrtの内容より小さい場合は，汎用レジスタrdに1を格納します。そうでない場合には，汎用レジスタrdにゼロを格納します。

比較は，その比較演算中に使用される減算がオーバフローを起こしても有効です。そのためどのような場合でも，オーバフロー例外は発生しません。

オペレーション：

```
T:      if GPR[rs] < GPR[rt] then
            GPR[rd] ← 0^31 || 1
        else
            GPR[rd] ← 0^32
        endif
```

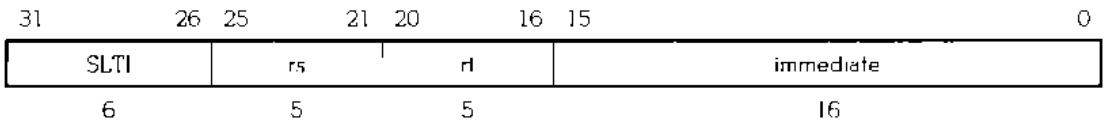
例　　外：

なし

保守／廃止

SLTI命令

Set On Less Than Immediate



命令形式：

SLTI rt, rs, immediate

説　明：

16ビットimmediateを32ビットに符号拡張し，汎用レジスタrsの内容と比較します。汎用レジスタrsの内容が符号拡張したイミーディエト値より小さい場合は，汎用レジスタrtに1を格納します　そうでない場合は，汎用レジスタrtにゼロを格納します。

比較は，その比較演算中に使用する減算がオーバフローを起こしても有効です。そのためどのような場合でも，オーバフロー例外は発生しません。

オペレーション：

```
T:      if GPR[rs] < ((immediate15)^16 | immediate15..0) then
            GPR[rt] <- 0^31 | 1
        else
            GPR[rt] <- 0^32
        endif
```

例　外：

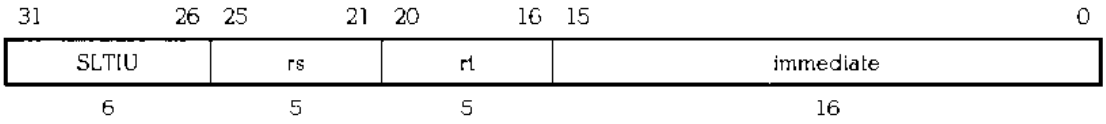
なし

11

保守／廃止

SLTIU命令

Set On Less Than
Immediate Unsigned



命令形式：

SLTIU rt, rs, immediate

説　　明：

16ビットimmediateを32ビットに符号拡張して無符号32ビット整数として汎用レジスタrsの内容と比較します。汎用レジスタrsの内容がイミーディエト値より小さい場合は，汎用レジスタrtに1を格納します，そうでない場合は，汎用レジスタrtにゼロを格納します。

比較は，その比較演算中に使用する減算がオーバフローを起こしても有効です。そのためどのような場合でも，オーバフロー例外は発生しません。

オペレーション：

```
T:      if(0 || GPR[rs] < (0 || (immediate15)16 || immediate15..0)
        then
            GPR[rt] ← 0^31 || 1
        else
            GPR[rt] ← 0^32
        endif
```

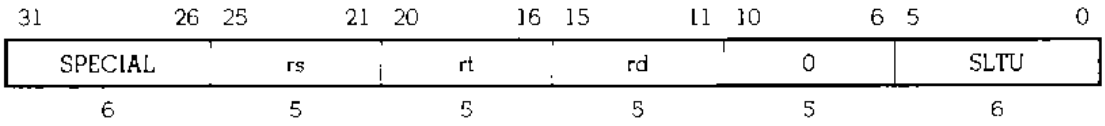
例　　外：

なし

保守／廃止

SLTU命令

Set On Less Than Unsigned



命令形式：

SLTU rd, rs, rt

説　明：

汎用レジスタrtと汎用レジスタrsの内容を，無符号32ビット整数として比較します。汎用レジスタrsの内容が汎用レジスタrtの内容より小さい場合は，汎用レジスタrdに1を格納します。そうでない場合は，汎用レジスタrdにゼロを格納します。

比較は，その比較演算中に使用する減算がオーバフローを起こしても有効です。そのためどのような場合でも，オーバフロー例外は発生しません。

オペレーション：

```
T:      if(0 || GPR[rs]) < (0 | GPR[rt]) then
            GPR[rd]← 0^31 || 1
        else
            GPR[rd]← 0^32
        endif
```

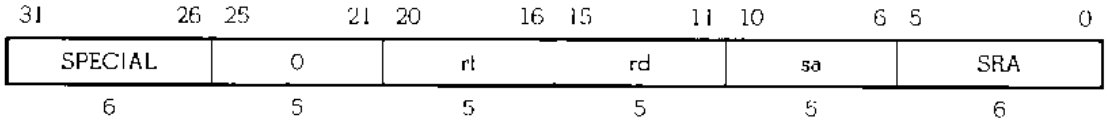
例　外：

なし

保守／廃止

SRA命令

Shift Right Arithmetic



命令形式：

SRA rd, rt, sa

説　明：

汎用レジスタrtを、saビットだけ右にシフトします。シフトした上位ビットには、汎用レジスタrtの符号を格納します。結果を、汎用レジスタrdに格納します。

オペレーション：

```
T:      GPR[rd] ← (GPR[rt]31)^sa || GPR[rt]31..sa
```

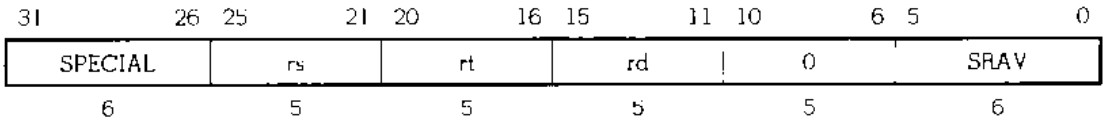
例　外：

なし

保守／廃止

SRAV命令

Shift Right Arithmetic Variable



命令形式：

SRAV rd, rt, rs

説　　明：

汎用レジスタrtの内容を，汎用レジスタrsの内容の下位５ビットが指定するビット数だけ右にシフトします。シフトした上位ビットには汎用レジスタrtの符号を拡張します。結果を，汎用レジスタrdに格納します。

オペレーション：

```
T:      s ← GPR[rs]4..0
        GPR[rd] ← (GPR[rt]31)^s || GPR[rt]31..s
```

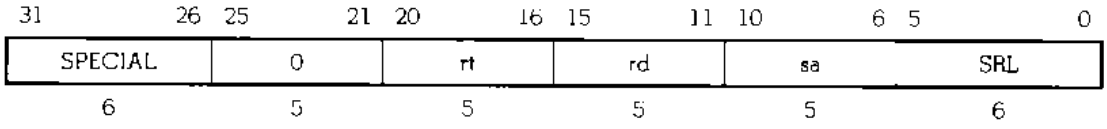
例　　外：

なし

保守／廃止

SRL命令

Shift Right Logical



命令形式：

SRL rd, rt, sa

説　　明：

汎用レジスタrtの内容を，saビットだけ右にシフトします。シフトした上位ビットには，ゼロを挿入します。結果を，汎用レジスタrdに格納します。

オペレーション：

```
T:      GPR[rd] ← 0^(sa) || GPR[rt]_(31..sa)
```

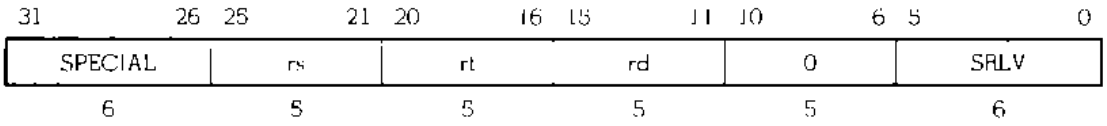
例　　外：

なし

保守／廃止

SRLV命令

Shift Right
Logical Variable



命令形式：

SRLV rd, rt, rs

説　　明：

汎用レジスタrtの内容を，汎用レジスタrsの内容の下位５ビットが指定するビット数だけ右にシフトします．シフトした上位ビットには，符号拡張します。結果を，汎用レジスタrdに格納します。

オペレーション：

```
T:      s ← GPR[rs]4..0
        GPR[rd] ← 0^(s) || GPR[rt]31..s
```

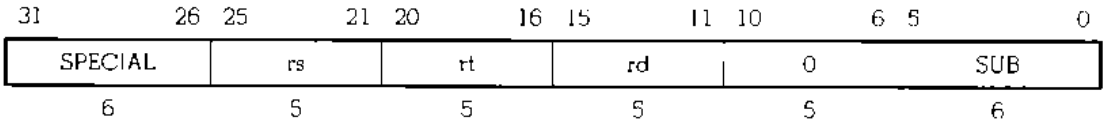
例　　外：

なし

保守／廃止

SUB命令

Subtract



命令形式：

SUB rd, rs, rt

説　明：

汎用レジスタrsの内容から汎用レジスタrtの内容を減算し，結果を汎用レジスタrdに格納します．

オーバフローのときには，オーバフロー例外が発生します。

オペレーション：

```
T:      GPR[rd] ← GPR[rs] - GPR[rt]
```

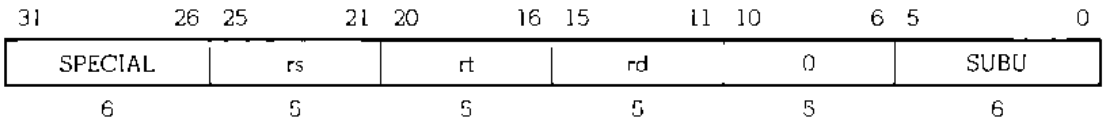
例　外：

オーバフロー例外

保守／廃止

SUBU命令

Subtract Unsigned



命令形式：

SUBU rd, rs, rt

説　明：

汎用レジスタrsの内容から汎用レジスタrtの内容を減算し，結果を汎用レジスタrdに格納します。どんな場合でも，オーバフロー例外は発生しません。

オペレーション：

```
T:      GPR[rd] ← GPR[rs] - GPR[rt]
```

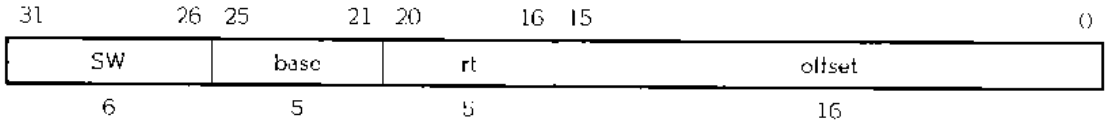
例　外：

なし

保守／廃止

SW命令

Store Word



命令形式：

SW rt, offset (base)

説　　明：

16ビットoffsetを汎用レジスタbaseの内容に加算し，32ビットの有効アドレスを生成します．汎用レジスタrtの内容を，有効アドレスの指定するメモリ位置に格納します。

有効アドレスの下位２ビットがゼロでない場合には，アドレス・エラー例外が発生します。

オペレーション：

```
T:      vAddr← (offset15)^16 || offset15..0 + GPR[base]
        (pAddr, uncached)← AddressTranslation(vAddr, DATA)
        data← GPR[rt]
        StoreMemory(uncached, WORD, data, pAddr, vAddr, DATA)
```

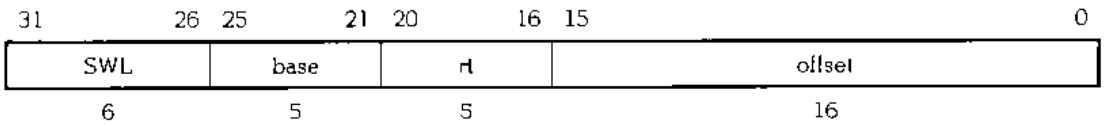
例　　外：

アドレス・エラー例外

保守／廃止

SWL命令

Store Word Left



命令形式：

SWL rt, offset (base)

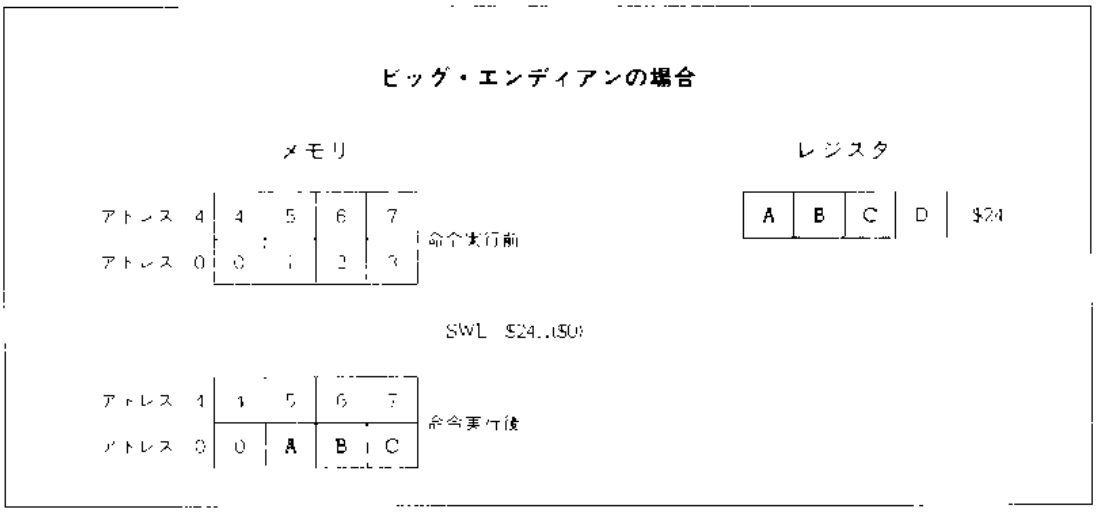
説　明：

この命令は，SWR命令と一緒に使用して，レジスタのワード・データをワードの境界にまたがって，メモリ内の連続した4バイトにストアすることができます。SWL命令は，レジスタの左部分をメモリの該当部分にストアし，SWR命令は，レジスタの右部分を該当部分にストアします。

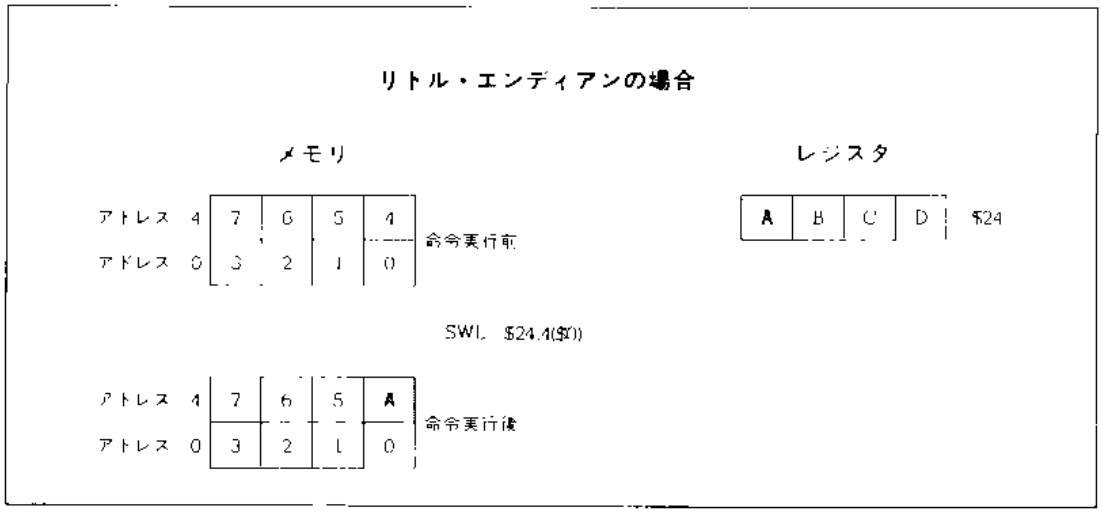
SWL命令は，16ビット・オフセットを汎用レジスタbaseの内容に加算し，32ビットの有効アドレスを生成します。この命令は，このバイトを格納しているメモリ内のワードだけを変更します。指定された先頭バイトに応じて，1バイトから4バイトのデータをストアします。

バイト位置合わせによるアドレス・エラー例外は，この命令では発生しません。

この命令は，まずレジスタの最上位バイトを指定されたメモリ内バイトにストアし，レジスタの下位バイトとメモリ内ワードの下位バイトの方に向かって順にデータのストアを行い，メモリ内ワードの最下位バイトまでそれを続けます。



11



オペレーション：

```
T:      vAddr← (offset15)16 || offset15..0 + GPR[base]
        (pAddr, uncached)←AddressTranslation(vAddr, DATA)
        byte← vAddr1..0 xor BigEndian2
        data←0^(24-8*byte) || GPR[rt]31..24-8*byte
        if ¯BigEndian then
          pAddr←pAddr31..2 || 0^2
        endif
        StoreMemory(uncached, byte, data, pAddr, vAddr, DATA)
```

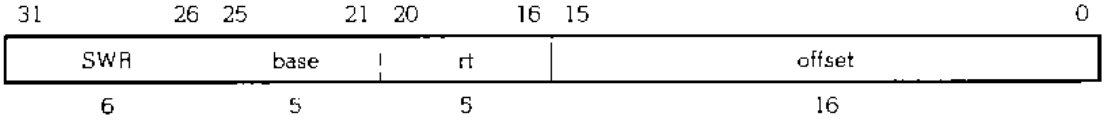
例　外：

アドレス・エラー例外

保守／廃止

SWR命令

Store Word Right



命令形式：

SWR rt, offset (base)

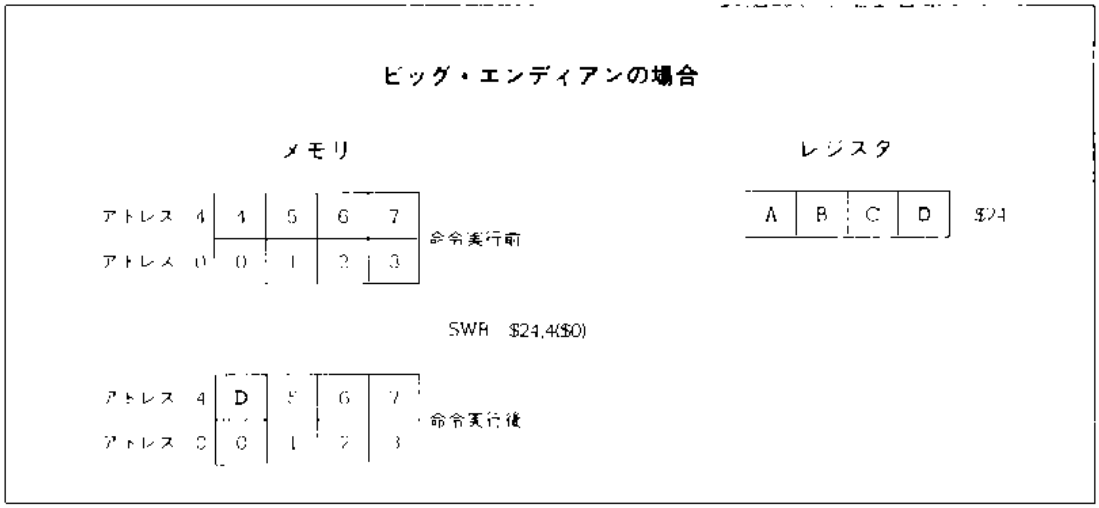
説　　明：

この命令は，SWL命令と一緒に使用して，レジスタのワード・データをワードの境界にまたがって，メモリ内の連続した4バイトにストアすることができます。SWR命令は，レジスタの右部分をメモリの該当部分にストアし，SWL命令は，レジスタの左部分を該当部分にストアします。

SWR命令は，16ビット・オフセットを汎用レジスタbaseの内容に加算し，32ビットの有効アドレスを生成します。この命令は，このバイトを格納しているメモリ内のワードだけを変更します。指定された先頭バイトに応じて，1バイトから4バイトのデータをストアします。

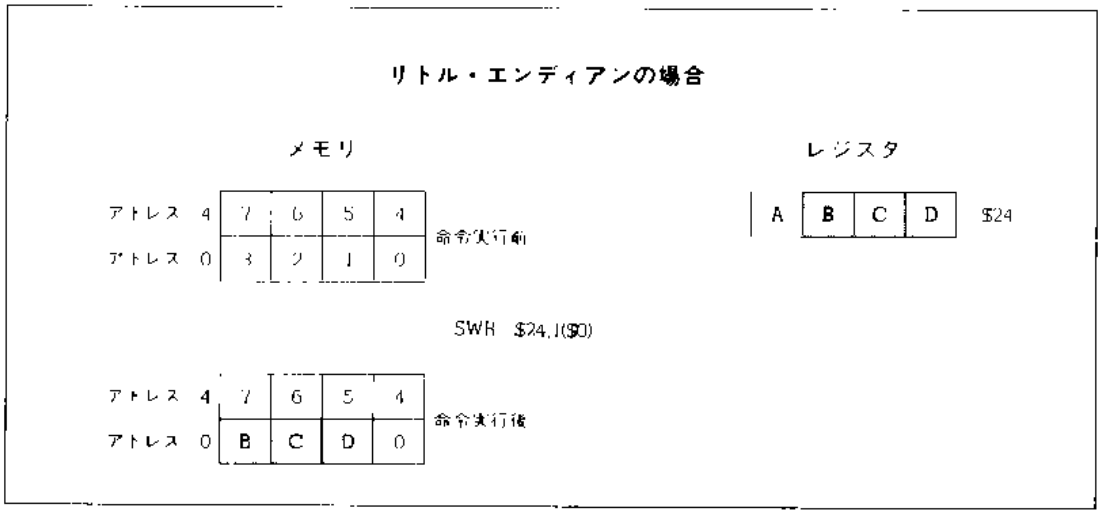
バイト位置合わせによるアドレス・エラー例外は，この命令では発生しません。

この命令は，まずレジスタの最下位バイトを指定されたメモリ内バイトにストアし，レジスタの上位バイトとメモリ内ワードの上位バイトの方に向かって順にデータのストアを行い，メモリ内ワードの最上位バイトまでそれを続けます。



11

保守／廃止



オペレーション：

```
T:      vAddr← (offset15)16 || offset15..0 + GPR[base]
        (pAddr, uncached) ← AddressTranslation (vAddr, DATA)
        byte← vAddr1..0 xor BigEndian2
        if ¬BigEndian then
          pAddr← pAddr31..2 || 02
        endif
        StoreMemory (uncached, WORD-byte, data, pAddr, vAddr, DATA)
```

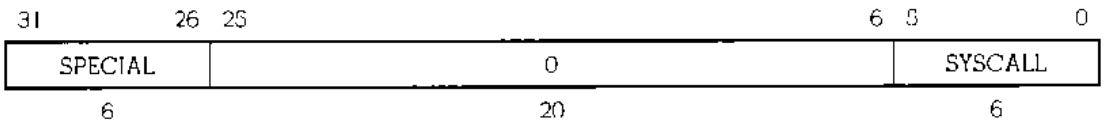
例　外：

アドレス・エラー例外

保守／廃止

SYSCALL命令

System Call



命令形式：

SYSCALL

説　　明：

システム・コール例外が発生し，制御権を例外ハンドラに移動します。

オペレーション：

```
T:      SystemCallException
```

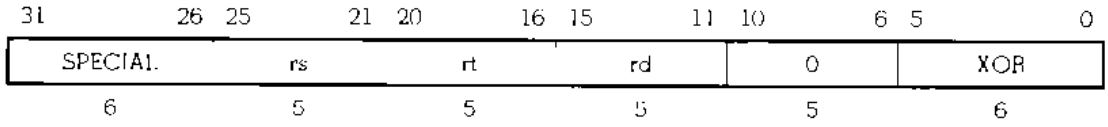
例　　外：

システム・コール例外

保守／廃止

XOR命令

Exclusive Or



命令形式：

XOR rd, rs, rt

説　明：

汎用レジスタrsの内容と汎用レジスタrtの内容との排他的論理和をとり，結果を汎用レジスタrdに格納します。

オペレーション：

```
T:      GPR[rd] ← GPR[rs] xor GPR[rt]
```

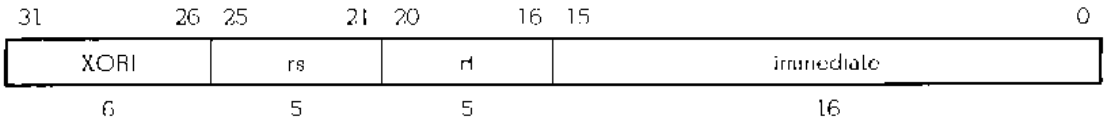
例　外：

なし

保守／廃止

XORI命令

Exclusive Or
Immediate



命令形式

XORI rt, ts, immediate

説　　明：

16ビットimmediateをゼロ拡張して，汎用レジスタrsの内容と排他的論理和をとり，結果を汎用レジスタrtに格納します．

オペレーション：

```
T:      GPR[rt] ← GPR[rs] xor (0^48 || immediate)
```

例　　外：

なし

11

保守／廃止

〔メ　モ〕

保守／廃止

付　録　命令コード表

Opcode

ビット31-29 ＼ ビット28-26	0	1	2	3	4	5	6	7
0	SPECIAL	BCOND	J	JAL	BEQ	BNE	BLEZ	BGTZ
1	ADDI	ADDIU	SLTI	SLTIU	ANDI	ORI	XORI	LUI
2	*	*	*	*	*	*	*	*
3	*	*	*	*	*	*	*	*
4	LB	LH	LWL	LW	LBU	LHU	LWR	*
5	SB	SH	SWL	SW	*	*	SWR	*
6	*	*	*	*	*	*	*	*
7	*	*	*	*	*	*	*	*

SPECIAL

ビット5-3 ＼ ビット2-0	0	1	2	3	4	5	6	7
0	SLL	*	SRL	SRA	SLLV	*	SRLV	SRAV
1	JR	JALR	*	*	SYSCALL	BREAK	*	*
2	MFHI	MTHI	MFLO	MTLO	*	*	*	*
3	MULT	MULTU	DIV	DIVU	*	*	*	*
4	ADD	ADDU	SUB	SUBU	AND	OR	XOR	NOR
5	*	*	SLT	SLTU	*	*	*	*
6	*	*	*	*	*	*	*	*
7	*	*	*	*	*	*	*	*

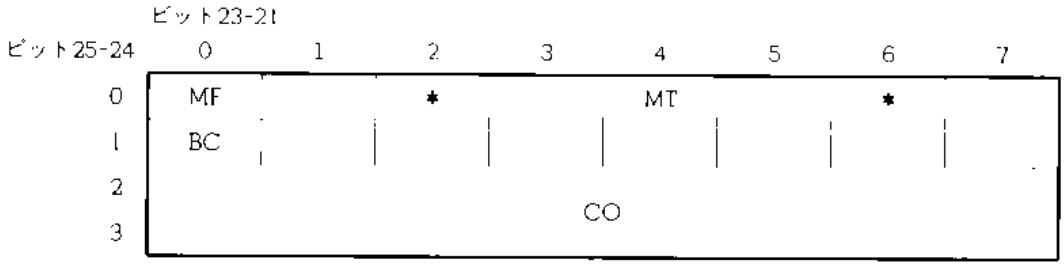
BCOND

ビット20-19 ＼ ビット18-16	0	1	2	3	4	5	6	7
0	BLTZ	BGEZ						
1								
2	BLTZAL	BGEZAL						
3								

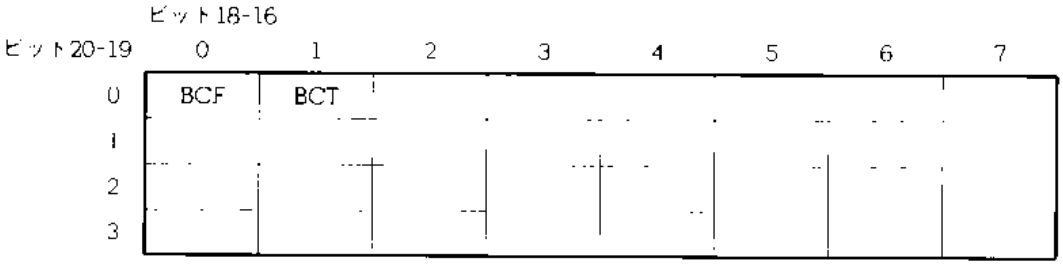
＊：予約

付

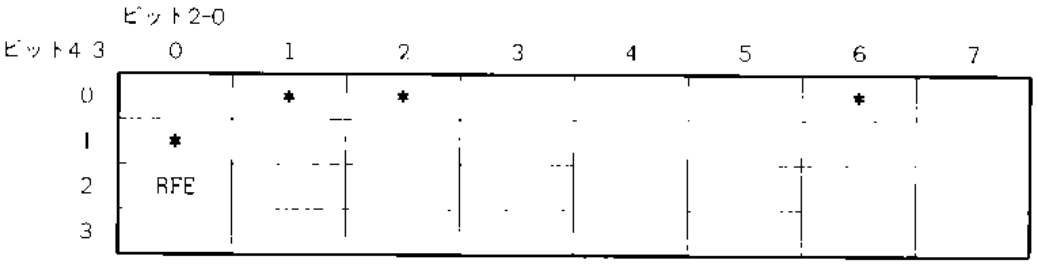
COPz rs



COPz rt



COP0 function ［ビット25＝ "1"］



保守／廃止

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］ V_R3800　ユーザーズ・マニュアル
（IEU-838B（第３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 半導体インフォメーションセンター
FAX：(044)548-7900

キリトリ

保守／廃止

お問い合わせは、最寄りのNECへ

【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	〒108-01 東京都港区芝五丁目7番1号(NEC本社ビル)	東京	(03)3454-1111 (大代表)
中部支社 半導体販売部	〒460 名古屋市中区栄四丁目14番5号(松下中日ビル)	名古屋	(052)242-2755
関西支社 半導体第一販売部 半導体第二販売部 半導体第三販売部	〒540 大阪市中央区城見一丁目4番24号(NEC関西ビル)	大阪 大阪 大阪	(06) 945-3178 (06) 945-3200 (06) 945-3208

北海道支社	札幌 (011)231-0161	小山支店	小山 (0285)24-5011	富山支店	富山 (0764)31-8461
東北支社	仙台 (022)261-5511	長野支社	長野 (0262)35-1444	二重支店	津 (0592)25-7341
盛岡支店	盛岡 (0196)51-4344	松本支店	松本 (0263)35-1666	京都支社	京都 (075)344-7824
山形支店	山形 (0236)23-5511	上諏訪支店	諏訪 (0266)53-5350	神戸支社	神戸 (078)332-3311
郡山支店	郡山 (0249)23-5511	甲府支店	甲府 (0552)24-4141	中国支社	広島 (082)242-5504
いわき支店	いわき (0246)21-5511	大宮支社	大宮 (048)641-1411	鳥取支店	鳥取 (0857)27-5311
長岡支店	長岡 (0258)36-2155	立川支社	立川 (0425)26-5981	岡山支店	岡山 (086)225-4455
土浦支店	土浦 (0298)23-6161	千葉支社	千葉 (043)238-8116	四国支社	高松 (0878)38-1200
水戸支店	水戸 (0292)26-1717	静岡支社	静岡 (054)255-2211	新居浜支店	新居浜 (0897)32-5001
神奈川支社	横浜 (045)324-5511	沼津支店	沼津 (0559)63-4455	松山支店	松山 (0899)45-4111
群馬支店	高崎 (0273)26-1255	浜松支店	浜松 (053)452-2711	九州支社	福岡 (092)271-7700
太田支店	太田 (0276)46-4011	北陸支社	金沢 (0762)23-1621	北九州支店	北九州 (093)541-2887
宇都宮支店	宇都宮 (0286)21-2261	福井支店	福井 (0776)22-1666		

【本資料に関する技術お問い合わせ先】

半導体ソリューション情報本部 マイクロコンピュータ技術部	〒210 川崎市幸区堀川町三丁目484番地	川崎	(044)548-8890
半導体販売技術本部 東日本販売技術部	〒108-01 東京都港区芝五丁目7番1号(NEC本社ビル)	東京	(03)3798-9619
半導体販売技術本部 中部販売技術部	〒460 名古屋市中区栄四丁目14番5号(松下中日ビル)	名古屋	(052)242-2762
半導体販売技術本部 西日本販売技術部	〒540 大阪市中央区城見一丁目4番24号(NEC関西ビル)	大阪	(06) 945-3383

半導体
インフォメーションセンター
FAX(044)548-7900
(FAXにてお願い致します)