SONY

SPU アセンブリ言語の仕様

Version 1.4

CBEA JSRE Series
Cell Broadband Engine Architecture
Joint Software Reference
Environment Series

2006 年 10 月 11 日

ソニー株式会社
〒141-0001 東京都品川区北品川 6-7-35
（2007年2月より 〒108-0075 東京都港区港南 1-7-1）

株式会社ソニー・コンピュータエンタテインメント
〒107-0062 東京都港区南青山 2-6-21

ソニーのホームページ　**http://www.sony.net**
SCEIのホームページ　**http://www.scei.co.jp**

Cell Broadband Engine のホームページ　**http://cell.scei.co.jp**

2006年 10月 11日

目次

表目次

SONY

本ドキュメントについて

本ドキュメントは、Cell Broadband Engine Architecture（CBEA）に準拠しているプロセッサのための Synergistic Processor Unit（SPU）アセンブリ言語構文について説明します。

対象者

本ドキュメントは、SPU 用のアセンブリ言語プログラムを書くことを望んでいるシステムおよびアプリケーション・プログラマを対象としています。

変更履歴

本セクションでは、本ドキュメントに関する重要な変更点をバージョンごとにまとめてあります。

バージョン番号および日付	変更点
v. 1.4 2006 年 10 月 11 日	SPU アセンブラ命令の表中のいくつかのオペランドを `rt` から `rc` に変更しました（TWG_RFC00049-0: CORRECTION NOTICE、および jsre-tool メール 00468、00488）。 SPU アセンブラ命令の表中の `wrch` 命令の記述を修正しました。 TWG_RFC00061-1 および TWG_RFC00062-0 による変更を適用しました。
v. 1.3 2005 年 10 月 20 日	「Broadband Processor Architecture」を「Cell Broadband Engine Architecture」に、「BPA」を「CBEA」に、それぞれ修正しました（TWG RFC 00037-0: CORRECTION NOTICE）。 一部の BE リビジョン DD1.0 および DD2.0 への参照を削除しました（TWG RFC 00040-0 CORRECTION NOTICE）。
v. 1.2 2005 年 8 月 1 日	「本ドキュメントについて」の一部のセクションを削除しました（TWG RFC 00032-0: CORRECTION NOTICE）。 いくつかのドキュメントの誤りを修正しました。たとえば表「SPU アセンブラ命令」で、「ハーフワード要素 `rt`」という記述を「レジスタ `rt` のハーフワード要素 1」に修正しました（TWG RFC 00033-0: CORRECTION NOTICE）。
v. 1.1 2005 年 6 月 10 日	「Broadband Engine」または「BE」を「Broadband Processor Architecture に準拠しているプロセッサ」または「BPA に準拠しているプロセッサ」に変更し、Synergistic Processing Unit を Synergistic Processor Unit に変更しました。最初の（主な）例で、PPU を PowerPC Processor Unit と定義しました。いくつかの参考文献を修正し、商標の所有者が明らかになるように著作権のページを変更しました。（変更点は全て TWG RFC 00031-0: CORRECTION NOTICE に従っています。） その他、「本ドキュメントについて」のセクションを変更しました。
v. 0.9 - 1.0	なし。JSRE のバージョン番号が IBM の公開リリースに使われているバージョン番号と同期するように、バージョン番号を変更しました。
v. 0.8 2005 年 5 月 12 日	PU を PPU に変更し、「PU-to-SPU」および「SPU-to-PU」（メールボックス）をそれぞれ「inbound」および「outbound」に変更しました（TWG RFC 00028-1: CORRECTION NOTICE）。 BPA チャネル名に合わせて、チャネル名を更新しました（TWG RFC 00029-1）。

バージョン番号および日付	変更点
v. 0.7 2004 年 7 月 16 日	命令の別名の表から、分岐の別名を全て削除しました（TWG RFC 00009-0）。 SPU 命令 `orx` を追加しました（TWG RFC 00010-0）。 イベントマスクおよびタグマスクの読み込みをサポートするチャネルのためにニーモニックを追加しました（TWG RFC 00011-0）。 `hbrp`命令からオペランドを削除し、この命令の新しい説明を提供しました。また、「2.6. エラーおよび警告」の表からもこの命令を削除しました（TWG RFC 00012-0）。 様々な編集上の変更を行いました。
v. 0.6 2004 年 3 月 12 日	様々な編集上の変更を行いました。
v. 0.5 2004 年 2 月 25 日	ixページに記載されている表記上の規則が反映されるように、ドキュメントのフォーマットを変更しました。最低限の編集上の変更を行いました。
v. 0.4 2004 年 1 月 20 日	ドキュメントのフォーマットを新しくし、前付けを追加しました。 様々な編集上の変更を行いました。
v. 0.3 2003 年 8 月 31 日	PC 関連のアドレス指定スタイルを修正しました。 下位および上位のハーフワードアドレス構文を追加しました。 stopd 命令を追加しました。
v. 0.2 2003 年 5 月 13 日	isolation control チャネルを追加しました。 `aci`、`asc`、`sbi`、および `ssb` 命令を `addx`、`cg`、`cgx`、`sfx`、`bg`、および `bgx` に置き換えました。
v. 0.1 2003 年 3 月 7 日	最初のリリース

関連ドキュメント

以下の表は、本ドキュメントの参考文献および資料の一覧です。

ドキュメント名	バージョン	日付
PowerPC User Instruction Set Architecture: Book I	2.02	2005 年 1 月 28 日
PowerPC Virtual Environment Architecture: Book II	2.02	2005 年 1 月 28 日
PowerPC Operating Environment Architecture: Book III	2.02	2005 年 1 月 28 日
PowerPC Microprocessor Family: The Programming Environments for 32-Bit Microprocessors (G522-0290-01)	1.0	2000 年 2 月 21 日
Cell Broadband Engine Architecture	1.01	2006 年 10 月
Synergistic Processor Unit 命令セット・アーキテクチャ	1.11	2006 年 10 月

本ドキュメントの構成

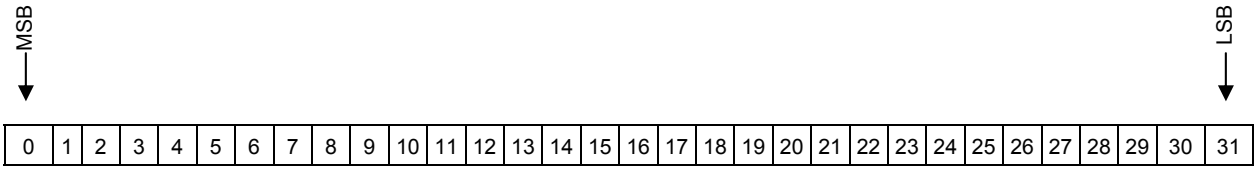
本ドキュメントは、主に以下のセクションを含みます。

1. はじめに
2. 命令セットおよび命令構文

本ドキュメント内で用いられるビット表記および書体の表記上の規則

ビット表記について

本ドキュメントでは、標準ビット表記を採用しています。ビット番号およびバイト番号は、左から右へ昇順に並んでいます。従って 4 バイトワードの場合、以下のように、bit 0 が MSB（最上位ビット）で、bit 31 が LSB（最下位ビット）になります。



MSB = Most significant bit（最上位ビット）

LSB = Least significant bit（最下位ビット）

ビットエンコーディングの表記法は以下の通りです。

- 16 進数の値の前には、`0x` が付いています。例：`0x0A00`
- 文中に出てくるバイナリ値は、シングルクオーテーション（’）で囲んでいます。例：‘1010’

書体の表記上の規則

本ドキュメントでは、ビット表記に加え、以下の書体の表記上の規則を採用しています。

規則	意味
`courier`	プログラミングコード、処理命令、レジスタ名、データ型、イベント、ファイル名、および他のリテラルを表します。関数名およびマクロ名を表す場合にも使用されます。この書体は、理解に役立つ場合にのみ使用され、特に説明文に使用されます。
`courier + 斜字体`	引数、パラメータ、および変数（`const` タイプの変数を含む）を表します。この書体は、理解に役立つ場合にのみ使用され、特に説明文に使用されます。
斜字体（courier なし）	強調を表します。ハイパーリンクの場合を除いて、文献の参照にはイタリック体が使用されます。言葉を初めて定義する場合、イタリック体を使用することが多々あります。
青	ハイパーリンクを表します（カラープリンタまたはオンライン専用）。

1. はじめに

本ドキュメントは、SPU アセンブリ言語構文および GNU アセンブラ（as）の機械依存機能について説明します。本ドキュメントは、GNU アセンブラに焦点を合わせていますが、他の SPU アセンブラの仕様の例にもなるかもしれません。

2. 命令セットおよび命令構文

2.1. 表記法および規則

このドキュメントにおいて、全ての命令、レジスタの別名、およびチャネル名に小文字が使われています。しかし、これらのトークンは、大文字で表したり、小文字と大文字を混ぜて表したりすることもできます。表 2-1は、本ドキュメントで用いられている表記法について説明します。

表2-1：表記法および規則

表記法／規則	意味
ch	チャネル番号。チャネルは、`$ch`の後にチャネル番号を付けて（例：`$ch3`）、または、特定のチャネルニーモニックとして表されます。チャネルニーモニックのリストに関しては、セクション「2.4. チャネルニーモニック」を参照してください。
ra, rb, rc	ソースレジスタ。レジスタは、ドル記号（`$`）の後にレジスタ番号（0 から 127）を付けて表されます。例えば、`$38` はレジスタ 38 を表します。その他のレジスタの別名に関しては、表 2-3を参照してください。
rt	ターゲットレジスタ。レジスタは、ドル記号（`$`）の後にレジスタ番号（0 から 127）を付けて表されます。例えば、`$38` はレジスタ 38 を表します。その他のレジスタの別名に関しては、表 2-3を参照してください。
s3, s6	それぞれ、3 ビットまたは 6 ビットの符号付き値です。7 ビットの符号付き即値としてエンコードされていますが、ビットの一部しか使用しません。
s7	7 ビットの符号拡張される値。
s10	10 ビットの符号拡張される値。
s11	11 ビットの符号拡張される値。
s14	14 ビットの符号拡張される値。
s16	16 ビットの符号拡張される値。
s18	相対アドレス計算。
scale7	7 ビットのスケール指数。0 から 127 の範囲の値。
spr	special purpose レジスタ。
u3, u5, u6	それぞれ、3 ビット、5 ビット、または 6 ビットの符号なし値。7 ビットの符号付き即値としてエンコードされていますが、ビットの一部しか使用しません。
u7	7 ビットの符号なし値。
u14	14 ビットの符号なし値。
u16	16 ビットの符号なし値。
u18	18 ビットの符号なし値。

2.2. 命令セット

本セクションは、SPU 命令セットとその構文の概要を提供します。本セクションには、以下が含まれます。

- サポートされている命令とその構文
- サポートされているデータ型
- 命令パラメータのサポートされている範囲

特定の機械命令の詳細については、Synergistic Processor Unit 命令セット・アーキテクチャ (SPU ISA)仕様書を参照してください。

表2-2：SPU アセンブラ命令

命令／使用法	説明
a rt, ra, rb	**Add word** ワードの加算。レジスタ ra の各ワード要素と、レジスタ rb の対応するワード要素を加算し、結果をレジスタ rt の対応するワード要素に格納します。
absdb rt, ra, rb	**Absolute difference of bytes** バイトの絶対差。レジスタ ra の各バイト要素を、レジスタ rb の対応するバイト要素から減算し、結果の絶対値をレジスタ rt の対応する要素に格納します。
addx rt, ra, rb	**Add word extended** ワードの加算（拡張版）。レジスタ ra の各ワード要素と、レジスタ rb の対応するワード要素と、レジスタ rt の対応するワード要素の最下位ビットを加算し、結果をレジスタ rt の対応するワード要素に格納します。
ah rt, ra, rb	**Add halfword** ハーフワードの加算。レジスタ ra の各ハーフワード要素と、レジスタ rb の対応するハーフワード要素を加算し、結果をレジスタ rt の対応するハーフワード要素に格納します。
ahi rt, ra, s10	**Add halfword immediate** ハーフワードと即値の加算。レジスタ ra の各ハーフワード要素と、符号拡張される即値 s10 を加算し、結果をレジスタ rt の対応するハーフワード要素に格納します。
ai rt, ra, s10	**Add word immediate** ワードと即値の加算。レジスタ ra の各ワード要素と、符号拡張される即値 s10 を加算し、結果をレジスタ rt の対応するワード要素に格納します。
and rt, ra, rb	**And** 論理積。レジスタ ra の値とレジスタ rb の値との論理積を求め、結果をレジスタ rt に格納します。
andbi rt, ra, s10	**And byte immediate** バイトと即値の論理積。レジスタ ra の各バイト要素と s10 の最下位 8 ビットとの論理積を求め、結果をレジスタ rt の対応する要素に格納します。
andc rt, ra, rb	**And with complement** 補数との論理積。レジスタ ra の値とレジスタ rb の補数との論理積を求め、結果をレジスタ rt に格納します。
andhi rt, ra, s10	**And halfword immediate** ハーフワードと即値の論理積。レジスタ ra の各ハーフワード要素と符号拡張される即値 s10 との論理積を求め、結果をレジスタ rt の対応する要素に格納します。
andi rt, ra, s10	**And word immediate** ワードと即値の論理積。レジスタ ra の各ワード要素と符号拡張される即値 s10 との論理積を求め、結果をレジスタ rt の対応する要素に格納します。
avgb rt, ra, rb	**Average bytes** バイトの平均化。レジスタ ra と rb の対応するバイト要素の平均をとり（(a+b+1) >> 1）、結果をレジスタ rt の対応するバイト要素に格納します。
bg rt, ra, rb	**Borrow generate word** ボローの生成。レジスタ ra の各符号なしワード要素と、レジスタ rb の対応する符号なしワード要素とを比較し、ra の値のほうが rb の値より大きければ、レジスタ rt の対応する要素に 0 を格納します。そうでなければ、1 を格納します。

命令／使用法	説明
bgx rt, ra, rb	**Borrow generate word extended** ボローの生成（拡張版）。レジスタ ra の各ワード要素を、レジスタ rb の対応するワード要素から減算し、ワード要素 rt の最下位ビットが 0 の場合、その結果からさらに 1 を減算します。結果が 0 より小さい場合、レジスタ rt の対応する要素に 0 を格納します。そうでなければ、1 を格納します。
bi ra	**Branch indirect** 間接分岐。レジスタ ra のワード要素 0 によって指定されたアドレスに分岐します。アドレスの最下位 2 ビットは無視します。
bid ra	**Branch indirect, disable** 間接分岐（割り込み禁止）。レジスタ ra のワード要素 0 によって指定されたアドレスに分岐し、割り込みを禁止します。アドレスの最下位 2 ビットは無視します。
bie ra	**Branch indirect, enable** 間接分岐（割り込み許可）。レジスタ ra のワード要素 0 によって指定されたアドレスに分岐し、割り込みを許可します。アドレスの最下位 2 ビットは無視します。
bihnz rc, ra	**Branch indirect if not zero halfword** 間接分岐（ハーフワード非 0）。レジスタ rc のハーフワード要素 1 が 0 の場合、次の命令を実行します。そうでなければ、レジスタ ra のワード要素 0 のアドレスに分岐します。アドレスの最下位 2 ビットは無視します。
bihnzd rc, ra	**Branch indirect if not zero halfword, disable** 間接分岐（ハーフワード非 0、割り込み禁止）。レジスタ rc のハーフワード要素 1 が 0 の場合、次の命令を実行します。そうでなければ、分岐を成立させ、レジスタ ra のワード要素 0 のアドレスに分岐します。アドレスの最下位 2 ビットは無視します。分岐が成立すると、割り込みを禁止します。そうでなければ、割り込み許可状態は変更されません。
bihnze rc, ra	**Branch indirect if not zero halfword, enable** 間接分岐（ハーフワード非 0、割り込み許可）。レジスタ rc のハーフワード要素 1 が 0 の場合、次の命令を実行します。そうでなければ、分岐を成立させ、レジスタ ra のワード要素 0 のアドレスに分岐します。アドレスの最下位 2 ビットは無視します。分岐が成立すると、割り込みを許可します。そうでなければ、割り込み許可状態は変更されません。
bihz rc, ra	**Branch indirect if zero halfword** 間接分岐（ハーフワード 0）。レジスタ rc のハーフワード要素 1 が 0 の場合、レジスタ ra のワード要素 0 のアドレスに分岐します。アドレスの最下位 2 ビットは無視します。そうでなければ、次の命令を実行します。
bihzd rc, ra	**Branch indirect if zero halfword, disable** 間接分岐（ハーフワード 0、割り込み禁止）。レジスタ rc のハーフワード要素 1 が 0 の場合、分岐を成立させ、レジスタ ra のワード要素 0 のアドレスに分岐します。アドレスの最下位 2 ビットは無視します。そうでなければ、次の命令を実行します。分岐が成立すると、割り込みを禁止します。そうでなければ、割り込み許可状態は変更されません。
bihze rc, ra	**Branch indirect if zero halfword, enable** 間接分岐（ハーフワード 0、割り込み許可）。レジスタ rc のハーフワード要素 1 が 0 の場合、分岐を成立させ、レジスタ ra のワード要素 0 のアドレスに分岐します。アドレスの最下位 2 ビットは無視します。そうでなければ、次の命令を実行します。分岐が成立すると、割り込みを許可します。そうでなければ、割り込み許可状態は変更されません。

命令／使用法	説明
binz rc, ra	**Branch indirect if not zero word** 間接分岐（ワード非 0）。レジスタ rc のワード要素 0 が 0 の場合、次の命令を実行します。そうでなければ、レジスタ ra のワード要素 0 のアドレスに分岐します。アドレスの最下位 2 ビットは無視します。
binzd rc, ra	**Branch indirect if not zero word, disable** 間接分岐（ワード非 0、割り込み禁止）。レジスタ rc のワード要素 0 が 0 の場合、次の命令を実行します。そうでなければ、分岐を成立させ、レジスタ ra のワード要素 0 のアドレスに分岐します。アドレスの最下位 2 ビットは無視します。分岐が成立すると、割り込みを禁止します。そうでなければ、割り込み許可状態は変更されません。
binze rc, ra	**Branch indirect if not zero word, enable** 間接分岐（ワード非 0、割り込み許可）。レジスタ rc のワード要素 0 が 0 の場合、次の命令を実行します。そうでなければ、分岐を成立させ、レジスタ ra のワード要素 0 のアドレスに分岐します。アドレスの最下位 2 ビットは無視します。分岐が成立すると、割り込みを許可します。そうでなければ、割り込み許可状態は変更されません。
bisl rt, ra	**Branch indirect and set link** リンク付き間接分岐。次の命令の実効アドレスを、レジスタ ra のワード要素 0 から取得します。アドレスの最下位 2 ビットは無視します。この命令の次の命令のアドレスをレジスタ rt のワード要素 0 に格納し、rt のその他のワード要素に 0 の値を割り当てます。
bisld rt, ra	**Branch indirect and set link, disable** リンク付き間接分岐（割り込み禁止）。次の命令の実効アドレスを、レジスタ ra のワード要素 0 から取得します。アドレスの最下位 2 ビットは無視します。この命令の次の命令のアドレスをレジスタ rt のワード要素 0 に格納し、rt のその他のワード要素に 0 の値を割り当てます。割り込みを禁止します。
bisle rt, ra	**Branch indirect and set link, enable** リンク付き間接分岐（割り込み許可）。次の命令の実効アドレスを、レジスタ ra のワード要素 0 から取得します。アドレスの最下位 2 ビットは無視します。この命令の次の命令のアドレスをレジスタ rt のワード要素 0 に格納し、rt のその他のワード要素に 0 の値を割り当てます。割り込みを許可します。
bisled rt, ra	**Branch indirect and set link on external data** リンク付き間接分岐（外部条件）。この命令の次の命令のアドレスをレジスタ rt のワード要素 0 に格納し、レジスタ rt のその他の要素に 0 の値を割り当てます。チャネル 0 のカウントが 0 ではない場合、レジスタ ra のワード要素 0 の実効アドレスに分岐します。アドレスの最下位 2 ビットは無視します。チャネル 0 のカウントが 0 の場合、次の命令を実行します。
bisledd rt, ra	**Branch indirect and set link on external data, disable** リンク付き間接分岐（外部条件、割り込み禁止）。この命令の次の命令のアドレスをレジスタ rt のワード要素 0 に格納し、レジスタ rt のその他の要素に 0 の値を割り当てます。チャネル 0 のカウントが 0 ではない場合、分岐を成立させ、レジスタ ra のワード要素 0 の実効アドレスに分岐します。アドレスの最下位 2 ビットは無視します。チャネル 0 のカウントが 0 の場合、次の命令を実行します。分岐が成立すると、割り込みを禁止します。そうでなければ、割り込み許可状態は変更されません。

命令／使用法	説明
bislede rt, ra	**Branch indirect and set link on external data, enable** リンク付き間接分岐（外部条件、割り込み許可）。この命令の次の命令のアドレスをレジスタ rt のワード要素 0 に格納し、レジスタ rt のその他の要素に 0 の値を割り当てます。チャネル 0 のカウントが 0 ではない場合、分岐を成立させ、レジスタ ra のワード要素 0 の実効アドレスに分岐します。アドレスの最下位 2 ビットは無視します。チャネル 0 のカウントが 0 の場合、次の命令を実行します。分岐が成立すると、割り込みを許可します。そうでなければ、割り込み許可状態は変更されません。
biz rc, ra	**Branch indirect if zero word** 間接分岐（ワード 0）。レジスタ rc のワード要素 0 が 0 の場合、レジスタ ra のワード要素 0 の実効アドレスに分岐します。アドレスの最下位 2 ビットは無視します。rc のワード要素 0 が 0 ではない場合、次の命令を実行します。
bizd rc, ra	**Branch indirect if zero word, disable** 間接分岐（ワード 0、割り込み禁止）。レジスタ rc のワード要素 0 が 0 の場合、分岐を成立させ、レジスタ ra のワード要素 0 の実効アドレスに分岐します。アドレスの最下位 2 ビットは無視します。rc のワード要素 0 が 0 ではない場合、次の命令を実行します。分岐が成立すると、割り込みを禁止します。そうでなければ、割り込み許可状態は変更されません。
bize rc, ra	**Branch indirect if zero word, enable** 間接分岐（ワード 0、割り込み許可）。レジスタ rc のワード要素 0 が 0 の場合、分岐を成立させ、レジスタ ra のワード要素 0 の実効アドレスに分岐します。アドレスの最下位 2 ビットは無視します。rc のワード要素 0 が 0 ではない場合、次の命令を実行します。分岐が成立すると、割り込みを許可します。そうでなければ、割り込み許可状態は変更されません。
br s18	**Branch relative** 相対分岐。現在の命令のアドレスと符号拡張される値 s18 との合計によって指定されるアドレスに分岐します。s18 の最下位 2 ビットは無視します。
bra s18	**Branch absolute** 絶対分岐。符号拡張される値 s18 によって指定されるアドレスに分岐します。s18 の最下位 2 ビットは無視します。
brasl rt, s18	**Branch absolute and set link** リンク付き絶対分岐。符号拡張される値 s18 によって指定されるアドレスに分岐します。s18 の最下位 2 ビットは無視します。現在の命令の次の命令をレジスタ rt のワード要素 0 に格納し、rt のその他の要素に 0 の値を割り当てます。
brhnz rc, s18	**Branch if not zero halfword** 分岐（ハーフワード非 0）。レジスタ rc のハーフワード要素 1 が 0 ではない場合、現在の命令のアドレスと符号拡張される値 s18 との合計によって指定されるアドレスに分岐します。s18 の最下位 2 ビットは無視します。rc のハーフワード要素 1 が 0 の場合、次の命令を実行します。
brhz rc, s18	**Branch if zero halfword** 分岐（ハーフワード 0）。レジスタ rc のハーフワード要素 1 が 0 の場合、現在の命令のアドレスと符号拡張される値 s18 との合計によって指定されるアドレスに分岐します。s18 の最下位 2 ビットは無視します。レジスタ rc のハーフワード要素 1 が 0 ではない場合、次の命令を実行します。
brnz rc, s18	**Branch if not zero word** 分岐（ワード非 0）。レジスタ rc のワード要素 0 が 0 ではない場合、現在の命令のアドレスと符号拡張される値 s18 との合計によって指定されるアドレスに分岐します。s18 の最下位 2 ビットは無視します。レジスタ rc のワード要素 0 が 0 の場合、次の命令を実行します。

命令／使用法	説明
brsl rt, s18	**Branch relative and set link** リンク付き相対分岐。現在の命令のアドレスと符号拡張される値 s18 との合計によって指定されるアドレスに分岐します。s18 の最下位 2 ビットは無視します。現在の命令の次の命令をレジスタ rt のワード要素 0 に格納し、rt のその他の要素に 0 の値を割り当てます。
brz rc, s18	**Branch if zero word** 分岐（ワード 0）。レジスタ rc のワード要素 0 が 0 の場合、現在の命令のアドレスと符号拡張される値 s18 との合計によって指定されるアドレスに分岐します。s18 の最下位 2 ビットは無視します。レジスタ rc のワード要素 0 が 0 ではない場合、次の命令を実行します。
cbd rt, u7(ra)	**Generate controls for byte insertion (d-form)** バイト挿入マスクの生成（d-form）。レジスタ ra と符号なし値 u7 との合計によって計算された実効アドレスにおいてバイトを挿入するための、命令 shufb によって用いられる制御マスクを生成します。制御マスクは、レジスタ rt に格納します。
cbx rt, ra, rb	**Generate controls for byte insertion (x-form)** バイト挿入マスクの生成（x-form）。レジスタ ra および rb の合計によって計算された実効アドレスにおいてバイトを挿入するための、命令 shufb によって用いられる制御マスクを生成します。制御マスクは、レジスタ rt に格納します。
cdd rt, u7(ra)	**Generate controls for doubleword insertion (d-form)** ダブルワード挿入マスクの生成（d-form）。レジスタ ra と符号なし値 u7 との合計によって計算された実効アドレスにおいてダブルワードを挿入するための、命令 shufb によって用いられる制御マスクを生成します。制御マスクは、レジスタ rt に格納します。
cdx rt, ra, rb	**Generate controls for doubleword insertion (x-form)** ダブルワード挿入マスクの生成（x-form）。レジスタ ra および rb の合計によって計算された実効アドレスにおいてダブルワードを挿入するための、命令 shufb によって用いられる制御マスクを生成します。制御マスクは、レジスタ rt に格納します。
ceq rt, ra, rb	**Compare equal word** ワードの比較（等しい）。レジスタ ra の各ワード要素と、レジスタ rb の対応するワード要素とを比較します。これらの要素が等しい場合、レジスタ rt の対応するワード要素の全ビットを 1 にします。これらの要素が等しくない場合、レジスタ rt の対応するワード要素に 0 を格納します。
ceqb rt, ra, rb	**Compare equal byte** バイトの比較（等しい）。レジスタ ra の各バイト要素と、レジスタ rb の対応するバイト要素とを比較します。これらの要素が等しい場合、レジスタ rt の対応するバイト要素の全ビットを 1 にします。これらの要素が等しくない場合、レジスタ rt の対応するバイト要素に 0 を格納します。
ceqbi rt, ra, s10	**Compare equal byte immediate** バイトと即値の比較（等しい）。レジスタ ra の各バイト要素と、s10 の最下位 8 ビットとを比較します。これらの値が等しい場合、レジスタ rt の対応するバイト要素の全ビットを 1 にします。これらの値が等しくない場合、レジスタ rt の対応するバイト要素に 0 を格納します。
ceqh rt, ra, rb	**Compare equal halfword** ハーフワードの比較（等しい）。レジスタ ra の各ハーフワード要素と、レジスタ rb の対応するハーフワード要素とを比較します。これらの要素が等しい場合、レジスタ rt の対応するハーフワード要素の全ビットを 1 にします。これらの要素が等しくない場合、レジスタ rt の対応するハーフワード要素に 0 を格納します。

命令／使用法	説明
ceqhi rt, ra, s10	**Compare equal halfword immediate** ハーフワードと即値の比較（等しい）。レジスタ ra の各ハーフワード要素と、16 ビットの符号拡張される値 s10 とを比較します。これらの値が等しい場合、レジスタ rt の対応するハーフワード要素の全ビットを 1 にします。これらの値が等しくない場合、レジスタ rt の対応するハーフワード要素に 0 を格納します。
ceqi rt, ra, s10	**Compare equal word immediate** ワードと即値の比較（等しい）。レジスタ ra の各ワード要素と、32 ビットの符号拡張される値 s10 とを比較します。これらの値が等しい場合、レジスタ rt の対応するワード要素の全ビットを 1 にします。これらの値が等しくない場合、レジスタ rt の対応するワード要素に 0 を格納します。
cflts rt, ra, scale7	**Convert floating to signed integer** 浮動小数点数を符号付き整数に変換。レジスタ ra の各浮動小数点要素に 2^{scale7} を掛け、32 ビットの符号付き整数に変換し、レジスタ rt の対応するワード要素に格納します。-2^{31} から $2^{31}-1$ の範囲外の値はクランプします（最も近いバウンドにサチュレート演算）。
cfltu rt, ra, scale7	**Convert floating to unsigned integer** 浮動小数点数を符号なし整数に変換。レジスタ ra の各浮動小数点要素に 2^{scale7} を掛け、32 ビットの符号なし整数に変換し、レジスタ rt の対応するワード要素に格納します。0 から $2^{32}-1$ の範囲外の値はクランプします（最も近いバウンドにサチュレート演算）。
cg rt, ra, rb	**Carry generate word** キャリーの生成。レジスタ ra の各ワード要素を、レジスタ rb の対応するワード要素に加算し、桁上がりをレジスタ rt の対応するワード要素の最下位ビットに格納し、rt の残りのビットには 0 を格納します。
cgt rt, ra, rb	**Compare greater than word** ワードの比較（より大）。レジスタ ra の各ワード要素と、レジスタ rb の対応するワード要素とを比較します。ra のワードのほうが rb の対応するワードより大きい場合、レジスタ rt の対応するワード要素の全ビットを 1 にします。ra のワードが rb の対応するワードより小さい、または、それに等しい場合、レジスタ rt の対応するワード要素に 0 を格納します。
cgtb rt, ra, rb	**Compare greater than byte** バイトの比較（より大）。レジスタ ra の各バイト要素と、レジスタ rb の対応するバイト要素とを比較します。ra のバイトのほうが rb の対応するバイトより大きい場合、レジスタ rt の対応するバイト要素の全ビットを 1 にします。ra のバイトが rb の対応するバイトより小さい、または、それに等しい場合、レジスタ rt の対応するバイト要素に 0 を格納します。
cgtbi rt, ra, s10	**Compare greater than byte immediate** バイトと即値の比較（より大）。レジスタ ra の各バイト要素と、s10 の最下位 8 ビットとを比較します。ra のバイトのほうが s10 より大きい場合、レジスタ rt の対応するバイト要素の全ビットを 1 にします。ra のバイトが s10 より小さい、または、それに等しい場合、レジスタ rt の対応するバイト要素に 0 を格納します。
cgth rt, ra, rb	**Compare greater than halfword** ハーフワードの比較（より大）。レジスタ ra の各ハーフワード要素と、レジスタ rb の対応するハーフワード要素とを比較します。ra のハーフワードのほうが rb の対応するハーフワードより大きい場合、レジスタ rt の対応するハーフワード要素の全ビットを 1 にします。ra のハーフワードが rb の対応するハーフワードより小さい、または、それに等しい場合、レジスタ rt の対応するハーフワード要素に 0 を格納します。

命令／使用法	説明
cgthi rt, ra, s10	**Compare greater than halfword immediate** ハーフワードと即値の比較（より大）。レジスタ ra の各ハーフワード要素と、16 ビットの符号拡張される値 s10 とを比較します。ra のハーフワードのほうが s10 より大きい場合、レジスタ rt の対応するハーフワード要素の全ビットを 1 にします。ra のハーフワードが s10 より小さい、または、それに等しい場合、レジスタ rt の対応するハーフワード要素に 0 を格納します。
cgti rt, ra, s10	**Compare greater than word immediate** ワードと即値の比較（より大）。レジスタ ra の各ワード要素と、32 ビットの符号拡張される値 s10 とを比較します。ra のワードのほうが s10 より大きい場合、レジスタ rt の対応するワード要素の全ビットを 1 にします。ra のワードが s10 より小さい、または、それに等しい場合、レジスタ rt の対応するワード要素に 0 を格納します。
cgx rt, ra, rb	**Carry generate word extended** キャリーの生成（拡張版）。レジスタ ra および rb の各ワード要素に関して、レジスタ ra の要素と、rb の対応する要素と、rt の最下位ビットとを合計して、桁上がりを得ます。桁上がりは rt の対応するワード要素の最下位ビットに格納し、残りのビットには 0 を格納します。
chd rt, u7(ra)	**Generate controls for halfword insertion (d-form)** ハーフワード挿入マスクの生成（d-form）。レジスタ ra と符号なし値 u7 との合計によって計算された実効アドレスにおいてハーフワードを挿入するための、命令 shufb によって用いられる制御マスクを生成します。制御マスクは、レジスタ rt に格納します。
chx rt, ra, rb	**Generate controls for halfword insertion (x-form)** ハーフワード挿入マスクの生成（x-form）。レジスタ ra および rb の合計によって計算された実効アドレスにおいてハーフワードを挿入するための、命令 shufb によって用いられる制御マスクを生成します。制御マスクは、レジスタ rt に格納します。
clgt rt, ra, rb	**Compare logical greater than word** ワードの論理比較（より大）。レジスタ ra の各ワード要素と、レジスタ rb の対応するワード要素とを論理的に比較します。ra のワードのほうが rb の対応するワードより大きい場合、レジスタ rt の対応するワード要素の全ビットを 1 にします。ra のワードが rb の対応するワードより小さい、または、それに等しい場合、レジスタ rt の対応するワード要素に 0 を格納します。
clgtb rt, ra, rb	**Compare logical greater than byte** バイトの論理比較（より大）。レジスタ ra の各バイト要素と、レジスタ rb の対応するバイト要素とを論理的に比較します。ra のバイトのほうが rb の対応するバイトより大きい場合、レジスタ rt の対応するバイト要素の全ビットを 1 にします。ra のバイトが rb の対応するバイトより小さい、または、それに等しい場合、レジスタ rt の対応するバイト要素に 0 を格納します。
clgtbi rt, ra, s10	**Compare logical greater than byte immediate** バイトと即値の論理比較（より大）。レジスタ ra の各バイト要素と、s10 の最下位 8 ビットとを論理的に比較します。ra のバイトのほうが s10 より大きい場合、レジスタ rt の対応するバイト要素の全ビットを 1 にします。ra のバイトが s10 より小さい、または、それに等しい場合、レジスタ rt の対応するバイト要素に 0 を格納します。

命令／使用法	説明
clgth rt, ra, rb	**Compare logical greater than halfword** ハーフワードの論理比較（より大）。レジスタ ra の各ハーフワード要素と、レジスタ rb の対応するハーフワード要素とを論理的に比較します。ra のハーフワードのほうが rb の対応するハーフワードより大きい場合、レジスタ rt の対応するハーフワード要素の全ビットを 1 にします。ra のハーフワードが rb の対応するハーフワードより小さい、または、それに等しい場合、レジスタ rt の対応するハーフワード要素に 0 を格納します。
clgthi rt, ra, s10	**Compare logical greater than halfword immediate** ハーフワードと即値の論理比較（より大）。レジスタ ra の各ハーフワード要素と、16 ビットの符号拡張される値 s10 とを論理的に比較します。ra のハーフワードのほうが s10 より大きい場合、レジスタ rt の対応するハーフワード要素の全ビットを 1 にします。ra のハーフワードが s10 より小さい、または、それに等しい場合、レジスタ rt の対応するハーフワード要素に 0 を格納します。
clgti rt, ra, s10	**Compare logical greater than word immediate** ワードと即値の論理比較（より大）。レジスタ ra の各ワード要素と、32 ビットの符号拡張される値 s10 とを論理的に比較します。ra のワードのほうが s10 より大きい場合、レジスタ rt の対応するワード要素の全ビットを 1 にします。ra のワード要素が s10 より小さい、または、それに等しい場合、レジスタ rt の対応するワード要素に 0 を格納します。
clz rt, ra	**Count leading zeros** 先頭の 0 の数。レジスタ ra の各ワード要素の最初の 1 の左側にある 0 の数を数え、結果をレジスタ rt の対応する要素に格納します。
cntb rt, ra	**Count ones in bytes** バイト内の 1 の数。レジスタ ra の各バイト要素の 1 の数を数え、結果をレジスタ rt の対応する要素に格納します。
csflt rt, ra, scale7	**Convert signed integer to floating** 符号付き整数を浮動小数点数に変換。レジスタ ra の各符号付きワード要素を浮動小数点数に変換し、それに $2^{-scale7}$ を掛け、レジスタ rt の対応する浮動小数点要素に格納します。
cuflt rt, ra, scale7	**Convert unsigned integer to floating** 符号なし整数を浮動小数点数に変換。レジスタ ra の各符号なしワード要素を浮動小数点数に変換し、それに $2^{-scale7}$ を掛け、レジスタ rt の対応する浮動小数点要素に格納します。
cwd rt, u7(ra)	**Generate controls for word insertion (d-form)** ワード挿入マスクの生成（d-form）。レジスタ ra と符号なし値 u7 との合計によって計算された実効アドレスにおいてワードを挿入するための、命令 shufb によって用いられる制御マスクを生成します。制御マスクは、レジスタ rt に格納します。
cwx rt, ra, rb	**Generate controls for word insertion (x-form)** ワード挿入マスクの生成（x-form）。レジスタ ra および rb の合計によって計算された実効アドレスにおいてワードを挿入するための、命令 shufb によって用いられる制御マスクを生成します。制御マスクは、レジスタ rt に格納します。
dfa rt, ra, rb	**Double floating add** 倍精度の加算。レジスタ ra の各倍精度浮動小数点要素を、レジスタ rb の対応する倍精度浮動小数点要素に加算し、結果をレジスタ rt の対応する要素に格納します。
dfm rt, ra, rb	**Double floating multiply** 倍精度の乗算。レジスタ ra の各倍精度浮動小数点要素に、レジスタ rb の対応する倍精度浮動小数点要素を掛け、結果をレジスタ rt の対応する要素に格納します。

命令／使用法	説明
dfma rt, ra, rb	**Double floating multiply and add** 倍精度の乗算および加算。レジスタ ra の各倍精度浮動小数点要素に、レジスタ rb の対応する倍精度浮動小数点要素を掛け、その積に、レジスタ rt の対応する倍精度浮動小数点要素を加算します。結果は、レジスタ rt の対応する要素に格納します。
dfms rt, ra, rb	**Double floating multiply and subtract** 倍精度の乗算および減算。レジスタ ra の各倍精度浮動小数点要素に、レジスタ rb の対応する倍精度浮動小数点要素を掛け、その積から、レジスタ rt の対応する倍精度浮動小数点要素を引きます。結果は、レジスタ rt の対応する要素に格納します。
dfnma rt, ra, rb	**Double floating negative multiply and add** 倍精度の乗算および加算（符号反転）。レジスタ ra の各倍精度浮動小数点要素に、レジスタ rb の対応する倍精度浮動小数点要素を掛け、その積に、レジスタ rt の対応する倍精度浮動小数点要素を加算します。各結果の符号を反転させ、レジスタ rt の対応する要素に格納します。
dfnms rt, ra, rb	**Double floating negative multiply and subtract** 倍精度の乗算および減算（符号反転）。レジスタ ra の各倍精度浮動小数点要素に、レジスタ rb の対応する倍精度浮動小数点要素を掛け、その積を、レジスタ rt の対応する倍精度浮動小数点要素から引きます。結果は、レジスタ rt の対応する要素に格納します。
dfs rt, ra, rb	**Double floating subtract** 倍精度浮動小数点数の減算。レジスタ rb の各倍精度浮動小数点要素を、レジスタ ra の対応する倍精度浮動小数点要素から引き、結果をレジスタ rt の対応する要素に格納します。
dsync	**Synchronize data** データの同期。プロセッサは、ローカルストレージメモリに対する全ての待ち状態のストア命令が完了するのを待ってから、次の命令に進みます。
eqv rt, ra, rb	**Equivalent** 同値。レジスタ ra の値とレジスタ rb の値との排他的論理和をとり、結果の補数をレジスタ rt に格納します。
fa rt, ra, rb	**Floating add** 浮動小数点数の加算。レジスタ ra の各浮動小数点要素を、レジスタ rb の対応する浮動小数点要素に加算し、結果をレジスタ rt の対応する要素に格納します。
fceq rt, ra, rb	**Floating compare equal** 浮動小数点数の比較（等しい）。レジスタ ra の各浮動小数点要素と、レジスタ rb の対応する浮動小数点要素とを比較します。これらの要素が等しい場合、レジスタ rt の対応するワード要素の全ビットを 1 にします。これらの要素が等しくない場合、レジスタ rt の対応するワード要素に 0 を格納します。
fcgt rt, ra, rb	**Floating compare greater than** 浮動小数点数の比較（より大）。レジスタ ra の各浮動小数点要素と、レジスタ rb の対応する浮動小数点要素とを比較します。ra の要素のほうが rb の対応する要素より大きい場合、レジスタ rt の対応するワード要素の全ビットを 1 にします。ra の要素が rb の対応する要素より小さい、または、それに等しい場合、レジスタ rt の対応するワード要素に 0 を格納します。
fcmeq rt, ra, rb	**Floating compare magnitude equal** 浮動小数点数の絶対値の比較（等しい）。レジスタ ra の各浮動小数点要素の絶対値と、レジスタ rb の対応する浮動小数点要素の絶対値とを比較します。これらの要素が等しい場合、レジスタ rt の対応するワード要素の全ビットを 1 にします。これらの要素が等しくない場合、レジスタ rt の対応するワード要素に 0 を格納します。

命令／使用法	説明
fcmgt rt, ra, rb	**Floating compare greater than** 浮動小数点数の絶対値の比較（より大）。レジスタ ra の各浮動小数点要素の絶対値と、レジスタ rb の対応する浮動小数点要素の絶対値とを比較します。ra の値のほうが rb の対応する値より大きい場合、レジスタ rt の対応するワード要素の全ビットを 1 にします。ra の値が rb の対応する値より小さい、または、それに等しい場合、レジスタ rt の対応するワード要素に 0 を格納します。
fesd rt, ra	**Floating extend single to double** 単精度から倍精度への変換。レジスタ ra の偶数の単精度浮動小数点要素をそれぞれ倍精度に変換し、レジスタ rt の対応する要素に格納します。
fi rt, ra, rb	**Floating interpolate** 浮動小数点数の補間。レジスタ ra の各浮動小数点要素を補間して、より正確な見積もりを行います。その際、レジスタ rb の対応する要素に含まれるベースおよびステップを使用します。ただし rb は、命令 frest または命令 frsqest の出力フォーマットになっているとします。補間結果は、レジスタ rt の対応する要素に格納します。
fm rt, ra, rb	**Floating multiply** 浮動小数点数の乗算。レジスタ ra の各浮動小数点要素に、レジスタ rb の対応する浮動小数点要素を掛け、その積をレジスタ rt の対応する要素に格納します。
fma rt, ra, rb, rc	**Floating multiply and add** 浮動小数点数の乗算および加算。レジスタ ra の各浮動小数点要素に、レジスタ rb の対応する浮動小数点要素を掛け、その積に、レジスタ rc の対応する浮動小数点要素を加算します。結果は、レジスタ rt の対応する要素に格納します。
fms rt, ra, rb, rc	**Floating multiply and subtract** 浮動小数点数の乗算および減算。レジスタ ra の各浮動小数点要素に、レジスタ rb の対応する浮動小数点要素を掛け、その積から、レジスタ rc の対応する浮動小数点要素を引きます。結果は、レジスタ rt の対応する要素に格納します。
fnms rt, ra, rb, rc	**Floating negative multiply and subtract** 浮動小数点数の乗算および減算（符号反転）。レジスタ ra の各浮動小数点要素に、レジスタ rb の対応する浮動小数点要素を掛け、その積を、レジスタ rc の対応する浮動小数点要素から引きます。結果は、レジスタ rt の対応する要素に格納します。
frds rt, ra	**Floating round double to single** 倍精度から単精度への丸め処理。レジスタ ra の各倍精度浮動小数点要素を単精度に丸め、レジスタ rt の対応する偶数要素に格納します。同時に、rt の対応する奇数要素に 0 を格納します。
frest rt, ra	**Floating reciprocal estimate** 浮動小数点数の逆数の見積もり。レジスタ ra の各浮動小数点要素の逆数を見積もるためにベースおよびステップを計算し、結果をレジスタ rt の対応する要素に格納します。この命令によって返される結果は、命令 fi のオペランドとして使われることを意図しています。
frsqest rt, ra	**Floating reciprocal square root estimate** 浮動小数点数の平方根の逆数の見積もり。レジスタ ra の各浮動小数点要素の平方根の逆数を見積もるためにベースおよびステップを計算し、結果をレジスタ rt の対応する要素に格納します。この命令によって返される結果は、命令 fi のオペランドとして使われることを意図しています。

命令／使用法	説明
fs rt, ra, rb	**Floating subtract** 浮動小数点数の減算。レジスタ rb の各浮動小数点要素を、レジスタ ra の対応する浮動小数点要素から引き、結果をレジスタ rt の対応する要素に格納します。
fscrrd rt	**Floating-point status control register read** 浮動小数点状態および制御レジスタの読み込み。浮動小数点状態および制御レジスタ（FPSCR）の内容を読み込み、レジスタ rt に格納します。
fscrwr ra fscrwr rc, ra	**Floating-point status control register write** 浮動小数点状態および制御レジスタの書き込み。128 ビットレジスタ ra を浮動小数点状態および制御レジスタ（FPSCR）に書き込みます。レジスタ rc は擬似ターゲットであり、そこに値が書き込まれることは決してありません。レジスタ rc が指定されていない場合、レジスタ 0 を擬似ターゲットとして使用します。
fsm rt, ra	**Form select mask for words** ワードマスクの生成。レジスタ ra のワード要素 0 の最下位 4 ビットをそれぞれ 32 回ずつ繰り返すことによって、マスクを生成します。128 ビットの結果は、レジスタ rt に返します。
fsmb rt, ra	**Form select mask for bytes** バイトマスクの生成。レジスタ ra のワード要素 0 の下位 16 ビットをそれぞれ 8 回ずつ繰り返すことによって、マスクを生成します。128 ビットの結果は、レジスタ rt に返します。
fsmbi rt, u16	**Form select mask for byte immediate** 即値によるバイトマスクの生成。u16 の 16 ビットをそれぞれ 8 回ずつ繰り返すことによって、マスクを生成します。128 ビットの結果は、レジスタ rt に返します。
fsmh rt, ra	**Form select mask for halfwords** ハーフワードマスクの生成。レジスタ ra のワード要素 0 の最下位 8 ビットをそれぞれ 16 回ずつ繰り返すことによって、マスクを生成します。128 ビットの結果は、レジスタ rt に返します。
gb rt, ra	**Gather bits from words** ワードからビットを収集。レジスタ ra の各ワード要素の最下位ビットを連結することによって、4 ビット値を形成します。その 4 ビット値は、レジスタ rt のワード要素 0 の最下位ビットに格納し、残りのビットには 0 を格納します。
gbb rt, ra	**Gather bits from bytes** バイトからビットを収集。レジスタ ra の各バイト要素の最下位ビットを連結することによって、16 ビット値を形成します。その 16 ビット値は、レジスタ rt のワード要素 0 の最下位ビットに格納し、残りのビットには 0 を格納します。
gbh rt, ra	**Gather bits from halfwords** ハーフワードからビットを収集。レジスタ ra の各ハーフワード要素の最下位ビットを連結することによって、8 ビット値を形成します。その 8 ビット値は、レジスタ rt のワード要素 0 の最下位ビットに格納し、残りのビットには 0 を格納します。
hbr s11, ra	**Hint for branch (r-form)** 分岐のヒント（r-form）。この命令のアドレスと符号拡張される値 s11 との合計によってアドレス指定される分岐命令のために、レジスタ ra のワード要素 0 に含まれる分岐ターゲットアドレスにおける、命令 prefetch の発生を許可します。s11 の最下位 2 ビットは無視します。

命令／使用法	説明
hbra s11, s18	**Hint for branch (a-form)** 分岐のヒント（a-form）。この命令のアドレスと符号拡張される値 s11 との合計によってアドレス指定される分岐命令のために、符号拡張される値 s18 によって指定される分岐ターゲットアドレスにおける、命令 prefetch の発生を許可します。s11 および s18 の最下位 2 ビットは無視します。
hbrp	**Hint for branch, prefetch (r-form)** 分岐のヒント、プリフェッチ（r-form）。フェッチユニットのスロットをインラインプリフェッチ用に確保します。この命令は、P ビットをセットした命令 hbr に変換されます。分岐命令へのオフセットを含む命令 hbr のフィールドは、0 に設定します。
hbrr s11, s18	**Hint for branch relative** 相対分岐のヒント。この命令のアドレスと符号拡張される値 s11 との合計によってアドレス指定される分岐命令のために、この命令のアドレスと符号拡張される値 s18 との合計によってアドレス指定される分岐ターゲットにおける、命令 prefetch の発生を許可します。s18 および s11 の最下位 2 ビットは無視します。
heq ra, rb heq rt, ra, rb	**Halt if equal** 停止（等しい）。レジスタ ra および rb のワード要素 0 が等しい場合、プロセッサは停止します。レジスタ rt は擬似ターゲットであり、書き込みが行われることは決してありません。レジスタ rt が指定されていない場合、レジスタ 0 を擬似ターゲットとして使用します。
heqi ra, s10 heqi rt, ra, s10	**Halt if equal immediate** 停止（即値と等しい）。レジスタ ra のワード要素 0 と符号拡張される値 s10 とが等しい場合、プロセッサは停止します。レジスタ rt は擬似ターゲットであり、そこに値が書き込まれることは決してありません。レジスタ rt が指定されていない場合、レジスタ 0 を擬似ターゲットとして使用します。
hgt ra, rb hgt rt, ra, rb	**Halt if greater than** 停止（より大）。レジスタ ra の符号付きワード要素 0 が、レジスタ rb のワード要素 0 より大きい場合、プロセッサは停止します。レジスタ rt は擬似ターゲットであり、そこに値が書き込まれることは決してありません。レジスタ rt が指定されていない場合、レジスタ 0 を擬似ターゲットとして使用します。
hgti ra, s10 hgti rt, ra, s10	**Halt if greater than immediate** 停止（即値より大）。レジスタ ra の符号付きワード要素 0 が、符号拡張される値 s10 より大きい場合、プロセッサは停止します。レジスタ rt は擬似ターゲットであり、そこに値が書き込まれることは決してありません。レジスタ rt が指定されていない場合、レジスタ 0 を擬似ターゲットとして使用します。
hlgt ra, rb hlgt rt, ra, rb	**Halt if logically greater than** 停止（論理的により大）。レジスタ ra の符号なしワード要素 0 が、レジスタ rb の符号なしワード要素 0 より大きい場合、プロセッサは停止します。レジスタ rt は擬似ターゲットであり、そこに値が書き込まれることは決してありません。レジスタ rt が指定されていない場合、レジスタ 0 を擬似ターゲットとして使用します。
hlgti ra, s10 hlgti rt, ra, s10	**Halt if logically greater than immediate** 停止（即値より論理的に大）。レジスタ ra の符号なしワード要素 0 が、符号拡張される値 s10 より論理的に大きい場合、プロセッサは停止します。レジスタ rt は擬似ターゲットであり、そこに値が書き込まれることは決してありません。レジスタ rt が指定されていない場合、レジスタ 0 を擬似ターゲットとして使用します。

命令／使用法	説明
il rt, s16	**Immediate load word** 即値のロード（ワード）。符号拡張される値 s16 を rt の各ワード要素にロードします。
ila rt, u18	**Immediate load address** 即値のロード（アドレス）。符号なし値 u18 を rt の各ワード要素にロードします。
ilh rt, u16	**Immediate load halfword** 即値のロード（ハーフワード）。値 u16 を rt の 8 ハーフワード要素の各々にロードします。
ilhu rt, u16	**Immediate load halfword upper** 即値のロード（上位ハーフワード）。値 u16 を rt の 4 ワード要素の各々の上位 16 ビットにロードします。
iohl rt, u16	**Immediate OR halfword lower** 即値の論理和（下位ハーフワード）。値 u16 と rt の各ワード要素との論理和をとります。
iretd iretd ra	**Interrupt return, disable** 割り込みからの復帰（割り込み禁止）。マシン状態の保存／復元レジスタ 0（SRR0）によって指定されるアドレスにジャンプします。割り込みを禁止します。レジスタ ra は擬似ソースであり、その内容は無視します。ra が指定されていない場合、レジスタ 0 を擬似ソースとして使用します。
irete irete ra	**Interrupt return, enable** 割り込みからの復帰（割り込み許可）。マシン状態の保存／復元レジスタ 0（SRR0）によって指定されるアドレスにジャンプします。割り込みを許可します。レジスタ ra は擬似ソースであり、その内容は無視します。ra が指定されていない場合、レジスタ 0 を擬似ソースとして使用します。
iret iret ra	**Interrupt return** 割り込みからの復帰。マシン状態の保存／復元レジスタ 0（SRR0）によって指定されるアドレスにジャンプします。レジスタ ra は擬似ソースであり、その内容は無視します。ra が指定されていない場合、レジスタ 0 を擬似ソースとして使用します。
lnop	**Nop operation (load)** ノーオペレーション（ロード）。ノーオペレーションをロードパイプライン上で行います。
lqa rt, s18	**Load quadword (a-form)** クワッドワードのロード（a-form）。クワッドワードを、符号拡張される値 s18 によって指定された実効アドレスからレジスタ rt にロードします。s18 の最下位 2 ビットは無視します。
lqd rt, s14(ra)	**Load quadword (d-form)** クワッドワードのロード（d-form）。クワッドワードを、レジスタ ra および符号拡張される値 s14 との合計によって計算された実効アドレスからレジスタ rt にロードします。s14 の最下位 4 ビットは無視します。
lqr rt, s18	**Load quadword instruction relative (a-form)** クワッドワードの相対ロード（a-form）。クワッドワードを、現在の命令のアドレスと s18 との合計によって指定された実効アドレスからレジスタ rt にロードします。s18 の最下位 2 ビットは無視します。
lqx rt, ra, rb	**Load quadword (x-form)** クワッドワードのロード（x-form）。クワッドワードを、レジスタ ra および rb の合計によって計算された実効アドレスからレジスタ rt にロードします。

命令／使用法	説明
mfspr rt, spr	**Move from special purpose register** special purpose レジスタからの移動。指定された special purpose レジスタ spr の内容をレジスタ rt のワード要素 0 に移動します。
mpy rt, ra, rb	**Multiply** 乗算。レジスタ ra および rb の対応するワード要素の符号付き下位 16 ビットを掛け合わせ、32 ビットの積をレジスタ rt の対応するワード要素に格納します。
mpya rt, ra, rb, rc	**Multiply and add** 乗算および加算。レジスタ ra および rb の対応するワード要素の符号付き下位 16 ビットを掛け合わせ、32 ビットの積をレジスタ rc の対応するワード要素に加算します。結果は、レジスタ rt の対応する要素に格納します。
mpyh rt, ra, rb	**Multiply high** 乗算（上位ハーフワードと下位ハーフワード）。レジスタ ra のワード要素の上位 16 ビットにレジスタ rb の対応する要素の下位 16 ビットを掛け、32 ビットの積を左に 16 ビットシフトさせ、その結果をレジスタ rt の対応するワード要素に格納します。
mpyhh rt, ra, rb	**Multiply high high** 乗算（上位ハーフワード同士）。レジスタ ra および rb のワード要素の符号付き上位 16 ビットを掛け合わせ、32 ビットの積をレジスタ rt の対応するワード要素に格納します。
mpyhha rt, ra, rb	**Multiply high high and add** 乗算および加算（上位ハーフワード同士）。レジスタ ra および rb のワード要素の符号付き上位 16 ビットを掛け合わせ、32 ビットの積をレジスタ rt の対応するワード要素に加算し、その合計をレジスタ rt に格納します。
mpyhhau rt, ra, rb	**Multiply high high unsigned and add** 符号なし乗算および加算（上位ハーフワード同士）。レジスタ ra および rb のワード要素の符号なし上位 16 ビットを掛け合わせ、32 ビットの積をレジスタ rt の対応するワード要素に加算し、その合計をレジスタ rt に格納します。
mpyhhu rt, ra, rb	**Multiply high high unsigned** 符号なし乗算（上位ハーフワード同士）。レジスタ ra および rb のワード要素の符号なし上位 16 ビットを掛け合わせ、32 ビットの積をレジスタ rt の対応するワード要素に格納します。
mpyi rt, ra, s10	**Multiply immediate** 乗算（下位ハーフワードと即値）。レジスタ ra の各ワード要素の下位 16 ビットに符号拡張される値 s10 を掛け、32 ビットの積をレジスタ rt の対応するワード要素に格納します。
mpys rt, ra, rb	**Multiply and shift right** 乗算および右シフト。レジスタ ra および rb の対応するワード要素の上位 16 ビットを掛け合わせ、32 ビットの積の上位 16 ビットをレジスタ rt の対応するワード要素の最下位ビットに格納します。
mpyu rt, ra, rb	**Multiply unsigned** 符号なし乗算（下位ハーフワード同士）。レジスタ ra および rb の対応するワード要素の符号なし下位 16 ビットを掛け合わせ、32 ビットの積をレジスタ rt の対応するワード要素に格納します。
mpyui rt, ra, s10	**Multiply unsigned immediate** 符号なし乗算（下位ハーフワードと即値）。レジスタ ra の各ワード要素の下位 16 ビットに、符号拡張される値 s10 を掛けます。両オペランドを、符号なしとして扱います。32 ビットの積は、レジスタ rt の対応するワード要素に格納します。

命令／使用法	説明
mtspr spr, ra	**Move to special purpose register** special purpose レジスタへの移動。レジスタ rt のワード要素 0 の内容を、special purpose レジスタ spr に移動します。
nand rt, ra, rb	**Nand** 否定論理積。レジスタ ra の値とレジスタ rb との論理積をとり、結果の補数をレジスタ rt に格納します。
nop nop rt	**Nop operation (execute)** ノーオペレーション（実行）。ノーオペレーションを、実行パイプライン上で行います。レジスタ rt は擬似ターゲットであり、そこに値が書き込まれることは決してありません。レジスタ rt が指定されていない場合、レジスタ 0 を擬似ターゲットとして使用します。
nor rt, ra, rb	**Nor** 否定論理和。レジスタ ra の値とレジスタ rb の論理和をとり、結果の補数をレジスタ rt に格納します。
or rt, ra, rb	**Or** 論理和。レジスタ ra の値とレジスタ rb との論理和をとり、結果をレジスタ rt に格納します。
orbi rt, ra, s10	**Or byte immediate** バイトと即値の論理和。レジスタ ra の各バイト要素と s10 の最下位 8 ビットとの論理和をとり、結果をレジスタ rt の対応する要素に格納します。
orc rt, ra, rb	**Or with complement** 補数との論理和。レジスタ ra の値とレジスタ rb の補数との論理和をとり、結果をレジスタ rt に格納します。
orhi rt, ra, s10	**Or halfword immediate** ハーフワードと即値の論理和。レジスタ ra の各ハーフワード要素と符号拡張される即値 s10 との論理和をとり、結果をレジスタ rt の対応する要素に格納します。
ori rt, ra, s10	**Or word immediate** ワードと即値の論理和。レジスタ ra の各ワード要素と符号拡張される値 s10 との論理和をとり、結果をレジスタ rt の対応する要素に格納します。
orx rt, ra	**Or word across** ワードの論理和（横方向）。レジスタ ra の 4 つのワード要素の論理和をとり、結果をレジスタ rt のワード要素 0 に格納します。レジスタ rt のワード要素 1、2、および 3 には、値 0 を格納します。
rchcnt rt, ch	**Read channel count** チャネルカウントの読み込み。チャネル ch のチャネルカウントを読み込み、そのカウントをレジスタ rt に格納します。
rdch rt, ch	**Read channel** チャネルの読み込み。チャネル ch の内容を読み込み、その内容をレジスタ rt に格納します。
rot rt, ra, rb	**Rotate word** ワードの回転。レジスタ ra の各ワード要素の内容を、レジスタ rb の対応するワード要素に応じて左に回転させ、結果をレジスタ rt の対応するワード要素に格納します。
roth rt, ra, rb	**Rotate halfword** ハーフワードの回転。レジスタ ra の各ハーフワード要素の内容を、レジスタ rb の対応するハーフワード要素に応じて左に回転させ、結果をレジスタ rt の対応するハーフワード要素に格納します。

命令／使用法	説明
rothi rt, ra, s7	**Rotate halfword immediate** ハーフワードの回転（即値指定）。レジスタ ra の各ハーフワード要素の内容を、s7 の最下位 4 ビットに応じて左に回転させ、結果をレジスタ rt の対応するハーフワード要素に格納します。
rothm rt, ra, rb	**Rotate and mask halfword** ハーフワードのマスク付き回転。レジスタ ra の各ハーフワード要素の内容を、レジスタ rb の対応するハーフワード要素の最下位 5 ビットの 2 の補数に応じて右にシフトさせ、結果をレジスタ rt の対応するハーフワード要素に格納します。
rothmi rt, ra, s6	**Rotate and mask halfword immediate** ハーフワードのマスク付き回転（即値指定）。レジスタ ra の各ハーフワード要素の内容を、符号付き値 s6 の 2 の補数に応じて右にシフトさせ、結果をレジスタ rt の対応するハーフワード要素に格納します。
roti rt, ra, s7	**Rotate word immediate** ワードの回転（即値指定）。レジスタ ra の各ワード要素の内容を、符号付値 s7 に応じて左に回転させ、結果をレジスタ rt の対応するワード要素に格納します。
rotm rt, ra, rb	**Rotate and mask word** ワードのマスク付き回転。レジスタ ra の各ワード要素の内容を、レジスタ rb の対応するワード要素の最下位 6 ビットの 2 の補数に応じて右にシフトさせ、結果をレジスタ rt の対応するワード要素に格納します。
rotma rt, ra, rb	**Rotate and mask algebraic word** ワードのマスク付き算術回転。レジスタ ra の各ワード要素の内容を、レジスタ rb の対応するワード要素の最下位 6 ビットの 2 の補数に応じて右にシフトさせます。空いたビットは、符号ビットで埋めます。結果は、レジスタ rt の対応するワード要素に格納します。
rotmah rt, ra, rb	**Rotate and mask algebraic halfword** ハーフワードのマスク付き算術回転。レジスタ ra の各ハーフワード要素の内容を、レジスタ rb の対応するハーフワード要素の最下位 5 ビットの 2 の補数に応じて右にシフトさせます。空いたビットは、符号ビットで埋めます。結果は、レジスタ rt の対応するハーフワード要素に格納します。
rotmahi rt, ra, s6	**Rotate and mask algebraic halfword immediate** ハーフワードのマスク付き算術回転（即値指定）。レジスタ ra の各ハーフワード要素の内容を、符号付き値 s6 に応じて右にシフトさせます。空いたビットは、符号ビットで埋めます。結果は、レジスタ rt の対応するハーフワード要素に格納します。
rotmai rt, ra, s7	**Rotate and mask algebraic word immediate** ワードのマスク付き算術回転（即値指定）。レジスタ ra の各ワード要素の内容を、符号付き値 s7 の 2 の補数に応じて右にシフトさせます。空いたビットは、符号ビットで埋めます。結果は、レジスタ rt の対応するワード要素に格納します。
rotmi rt, ra, s7	**Rotate and mask word immediate** ワードのマスク付き回転（即値指定）。レジスタ ra の各ワード要素の内容を、符号付き値 s7 の 2 の補数に応じて右にシフトさせます。結果は、レジスタ rt の対応するワード要素に格納します。
rotqbi rt, ra, rb	**Rotate quadword by bits** クワッドワードの回転（ビット単位）。レジスタ ra の内容を、レジスタ rb のワード要素 0 の最下位 3 ビットによって指定されるビット数だけ左に回転させ、結果をレジスタ rt に格納します。

命令／使用法	説明
rotqbii rt, ra, u3	**Rotate quadword by bits immediate** クワッドワードの回転（ビット単位、即値指定）。レジスタ ra の内容を、値 u3 に応じたビット数だけ左に回転させ、結果をレジスタ rt に格納します。
rotqby rt, ra, rb	**Rotate quadword by bytes** クワッドワードの回転（バイト単位）。レジスタ ra の内容を、レジスタ rb のワード要素 0 の最下位 4 ビットによって指定されるバイト数だけ左に回転させ、結果をレジスタ rt に格納します。
rotqbybi rt, ra, rb	**Rotate quadword by bytes from bit shift count** クワッドワードの回転（バイト単位）。レジスタ ra の内容を、レジスタ rb のワード要素 0 のビット 24-28 によって指定されるバイト数だけ左に回転させ、結果をレジスタ rt に格納します。
rotqbyi rt, ra, s7	**Rotate quadword by bytes immediate** クワッドワードの回転（バイト単位、即値指定）。レジスタ ra の内容を、符号付き値 s7 に応じたバイト数だけ左に回転させ、結果をレジスタ rt に格納します。
rotqmbi rt, ra, rb	**Rotate and mask quadword by bits** クワッドワードのマスク付き回転（ビット単位）。レジスタ ra の内容を、レジスタ rb のワード要素 0 の最下位 3 ビットの 2 の補数によって指定されるビット数だけ右にシフトさせ、結果をレジスタ rt に格納します。
rotqmbii rt, ra, s3	**Rotate and mask quadword by bits immediate** クワッドワードのマスク付き回転（ビット単位、即値指定）。レジスタ ra の内容を、符号付き値 s3 の 2 の補数によって指定されるビット数だけ右にシフトさせ、結果をレジスタ rt に格納します。
rotqmby rt, ra, rb	**Rotate and mask quadword by bytes** クワッドワードのマスク付き回転（バイト単位）。レジスタ ra の内容を、レジスタ rb のワード要素 0 の最下位 5 ビットの 2 の補数によって指定されるバイト数だけ右にシフトさせ、結果をレジスタ rt に格納します。
rotqmbybi rt, ra, rb	**Rotate and mask quadword by bytes from bit shift count** クワッドワードのマスク付き回転（バイト単位）。レジスタ ra の内容を、レジスタ rb のワード要素 0 のビット 25-28 の 2 の補数によって指定されるバイト数だけ右にシフトさせ、結果をレジスタ rt に格納します。
rotqmbyi rt, ra, s6	**Rotate and mask quadword by bytes immediate** クワッドワードのマスク付き回転（バイト単位、即値指定）。レジスタ ra の内容を、符号付き値 s6 の 2 の補数によって指定されるバイト数だけ右にシフトさせ、結果をレジスタ rt に格納します。
selb rt, ra, rb, rc	**Select bits** ビットの選択。レジスタ rc のビットで値が 0 のものは、レジスタ ra から対応するビットを選択します。値が 1 のビットは、レジスタ rb から対応するビットを選択します。クワッドワード結果は、レジスタ rt に格納します。
sf rt, ra, rb	**Subtract from word** ワードの逆減算。レジスタ ra の各ワード要素を、レジスタ rb の対応するワード要素から減算し、結果をレジスタ rt の対応するワード要素に格納します。
sfh rt, ra, rb	**Subtract from halfword** ハーフワードの逆減算。レジスタ ra の各ハーフワード要素を、レジスタ rb の対応するハーフワード要素から減算し、結果をレジスタ rt の対応するワード要素に格納します。

命令／使用法	説明
sfhi rt, ra, s10	**Subtract from halfword immediate** ハーフワードと即値の逆減算。レジスタ ra の各ハーフワード要素を、符号拡張される値 s10 から減算し、結果をレジスタ rt の対応するハーフワード要素に格納します。
sfi rt, ra, s10	**Subtract from word immediate** ワードと即値の逆減算。レジスタ ra の各ワード要素を、符号拡張される値 s10 から減算し、結果をレジスタ rt の対応するワード要素に格納します。
sfx rt, ra, rb	**Subtract from word extended** ワードの逆減算（拡張版）。レジスタ ra の各ワード要素を、レジスタ rb の対応するワード要素から減算し、ワード要素 rt の最下位ビットが 0 の場合、その結果からさらに 1 を減算します。結果は、レジスタ rt の対応するワード要素に格納します。
shl rt, ra, rb	**Shift left word** ワードの左シフト。レジスタ ra の各ワード要素の内容を、レジスタ rb の対応するワード要素の最下位 6 ビットに応じて左にシフトさせ、結果をレジスタ rt の対応するワード要素に格納します。
shlh rt, ra, rb	**Shift left halfword** ハーフワードの左シフト。レジスタ ra の各ハーフワード要素の内容を、レジスタ rb の対応するハーフワード要素の最下位 5 ビットに応じて左にシフトさせ、結果をレジスタ rt の対応するハーフワード要素に格納します。
shlhi rt, ra, u5	**Shift left halfword immediate** ハーフワードの左シフト（即値指定）。レジスタ ra の各ハーフワード要素の内容を、符号なし値 u5 に応じて左にシフトさせ、結果をレジスタ rt の対応するハーフワード要素に格納します。
shli rt, ra, u6	**Shift left word immediate** ワードの左シフト（即値指定）。レジスタ ra の各ワード要素の内容を、符号なし値 u6 に応じて左にシフトさせ、結果をレジスタ rt の対応するワード要素に格納します。
shlqbi rt, ra, rb	**Shift left quadword by bits** クワッドワードの左シフト（ビット単位）。レジスタ ra の内容を、レジスタ rb のワード要素 0 の最下位 3 ビットによって指定されるビット数だけ左にシフトさせ、結果をレジスタ rt に格納します。
shlqbii rt, ra, u3	**Shift left quadword by bits immediate** クワッドワードの左シフト（ビット単位、即値指定）。レジスタ ra の内容を、符号なし値 u3 によって指定されるビット数だけ左にシフトさせ、結果をレジスタ rt に格納します。
shlqby rt, ra, rb	**Shift left quadword by bytes** クワッドワードの左シフト（バイト単位）。レジスタ ra の内容を、レジスタ rb のワード要素 0 の最下位 5 ビットによって指定されるバイト数だけ左にシフトさせ、結果をレジスタ rt に格納します。
shlqbybi rt, ra, rb	**Shift left quadword by bytes from bit shift count** クワッドワードの左シフト（バイト単位）。レジスタ ra の内容を、レジスタ rb のワード要素 0 のビット 24-28 によって指定されるバイト数だけ左にシフトさせ、結果をレジスタ rt に格納します。
shlqbyi rt, ra, u5	**Shift left quadword by bytes immediate** クワッドワードの左シフト（バイト単位、即値指定）。レジスタ ra の内容を、符号なし値 u5 によって指定されるバイト数だけ左にシフトさせ、結果をレジスタ rt に格納します。
shufb rt, ra, rb, rc	**Shuffle bytes** バイトのシャッフル。レジスタ rc の各バイトを用いて、レジスタ ra、rb、定数（0、0x80、0xFF）のいずれかのバイト値を選択します。結果は、レジスタ rt の対応するバイトに格納します。

命令／使用法	説明
stop u14	**Stop and signal** 停止およびシグナル。実行を停止し、現在のアドレスを SPU NPC レジスタに書き込み、値 u14 を SPU ステータスレジスタに書き込み、割り込みを PowerPC® Processor Unit（PPU）に送ります。
stopd ra, rb, rc	**Stop and signal with dependencies** レジスタ依存つきの停止およびシグナル。レジスタ依存を満たした後、実行を停止します。これには、現在のアドレスの SPU NPC レジスタへの書き込み、値 0x3FFF の SPU ステータスレジスタへの書き込み、および PPU への割り込みが伴います。
stqa rc, s18	**Store quadword (a-form)** クワッドワードのストア（a-form）。レジスタ rc のクワッドワードを、符号拡張される値 s18 によって指定される実効アドレスにストアします。s18 の最下位 2 ビットは無視します。
stqd rc, s14(ra)	**Store quadword (d-form)** クワッドワードのストア（d-form）。レジスタ rc のクワッドワードを、レジスタ ra と符号拡張される値 s14 との合計によって計算される実効アドレスにストアします。s14 の最下位 4 ビットは無視します。
stqr rc, s18	**Store quadword instruction relative (a-form)** クワッドワードの相対ストア（a-form）。レジスタ rc のクワッドワードを、現在の命令のアドレスと s18 との合計によって指定される実効アドレスにストアします。s18 の最下位 2 ビットは無視します。
stqx rc, ra, rb	**Store quadword (x-form)** クワッドワードのストア（x-form）。レジスタ rc のクワッドワードを、レジスタ ra と rb との合計によって計算される実効アドレスにストアします。
sumb rt, ra, rb	**Sum bytes into halfword** バイトの合計。レジスタ ra の各ワード要素の 4 バイトを合計して、その結果をレジスタ rt の対応する奇数ハーフワード要素に格納し、レジスタ rb の各ワード要素の 4 バイトを合計して、その結果をレジスタ rt の対応する偶数ハーフワード要素に格納します。
sync	**Synchronize** 同期。プロセッサは、全ての待ち状態のストア命令が完了するのを待ってから、次の命令をフェッチします。
syncc	**Synchronize channel** チャネルの同期。プロセッサは、チャネルの準備が整い、全ての待ち状態のストア命令が完了するのを待ってから、次の命令をフェッチします。
wrch ch, ra	**Write channel** チャネルの書き込み。レジスタ ra の内容をチャネル ch に書き込みます。
xor rt, ra, rb	**Xor** 排他的論理和。レジスタ ra の値とレジスタ rb の排他的論理和をとり、結果をレジスタ rt に格納します。
xorbi rt, ra, s10	**Exclusive or byte immediate** バイトと即値の排他的論理和。レジスタ ra の各バイト要素と s10 の最下位 8 ビットの排他的論理和をとり、結果をレジスタ rt の対応する要素に格納します。
xorhi rt, ra, s10	**Exclusive or halfword immediate** ハーフワードと即値の排他的論理和。レジスタ ra の各ハーフワード要素と符号拡張される値 s10 の下位 16 ビットの排他的論理和をとり、結果をレジスタ rt の対応する要素に格納します。

命令／使用法	説明
xori rt, ra, s10	**Exclusive or word immediate** ワードと即値の排他的論理和。レジスタ ra の各ワード要素と符号拡張される値 s10 の排他的論理和をとり、結果をレジスタ rt の対応する要素に格納します。
xsbh rt, ra	**Extend sign byte to halfword** バイトからハーフワードへの符号拡張。レジスタ ra の各ハーフワード要素の最下位 8 ビットを 16 ビットに符号拡張し、レジスタ rt の対応するハーフワード要素に格納します。
xshw rt, ra	**Extend sign halfword to word** ハーフワードからワードへの符号拡張。レジスタ ra の各ワード要素の下位 16 ビットを 32 ビットに符号拡張し、レジスタ rt の対応するワード要素に格納します。
xswd rt, ra	**Extend sign word to doubleword** ワードからダブルワードへの符号拡張。レジスタ ra の各ダブルワード要素の最下位 32 ビットを 64 ビットに符号拡張し、レジスタ rt の対応するダブルワード要素に格納します。

2.3. 別名

プログラマの便宜上、アセンブラは、表 2-3に示されるレジスタおよび命令の別名に対応しています。

表2-3：レジスタおよび命令の別名

別名	レジスタおよび命令	説明
$LR	$0	リンクレジスタ（リターンアドレス）。
$SP	$1	スタックポインタ。
lr rt, ra	ori rt, ra, 0	レジスタ ra からレジスタ rt へのロード。

2.4. チャネルニーモニック

表 2-4および 表 2-5に示されるチャネルニーモニックに対応しています。アセンブラは、可能な全てのチャネル 0-127 に、$ch#の形式の一般的なチャネルニーモニックを提供します。#はチャネル番号を示します。例えば、$ch0 はイベントステータス読み込みチャネルです。

全ての SPU チャネルニーモニックに対応していなければなりません。その一方で、MFC チャネルニーモニックに対応していなければならないのは、MFC に対応しているターゲットシステムだけです。

表2-4：SPU チャネル

チャネル番号	ニーモニック	説明
0 - 127	$ch0 - $ch127	一般的なチャネルニーモニック。
0	$SPU_RdEventStat	イベントステータスの読み込み。
1	$SPU_WrEventMask	イベントマスクの書き込み。
2	$SPU_WrEventAck	イベント受け付けフラグの書き込み。
3	$SPU_RdSigNotify1	シグナル通知 1。
4	$SPU_RdSigNotify2	シグナル通知 2。
7	$SPU_WrDec	デクリメンタのカウントの書き込み。
8	$SPU_RdDec	デクリメンタのカウントの読み込み。
11	$SPU_RdEventMask	イベントマスクの読み込み。
13	$SPU_RdMachStat	SPU 実行ステータスの読み込み。

チャネル番号	ニーモニック	説明
14	$SPU_WrSRR0	SPU マシン状態の保存／復元レジスタ 0（SRR0）の書き込み。
15	$SPU_RdSRR0	SPU マシン状態の保存／復元レジスタ 0（SRR0）の読み込み。
28	$SPU_WrOutMbox	アウトバウンドメールボックスの内容の書き込み。
29	$SPU_RdInMbox	インバウンドメールボックスの内容の読み込み。
30	$SPU_WrOutIntrMbox	アウトバウンド割り込みメールボックスの内容の書き込み（PPU に対する割り込み）。

表2-5：MFC チャネル

チャネル番号	ニーモニック	説明
9	$MFC_WrMSSyncReq	マルチソース同期リクエストの書き込み。
12	$MFC_RdTagMask	タグマスクの読み込み。
16	$MFC_LSA	ローカルメモリアドレスコマンドパラメータの書き込み。
17	$MFC_EAH	上位 DMA 実効アドレスコマンドパラメータの書き込み。
18	$MFC_EAL	下位 DMA 実効アドレスコマンドパラメータの書き込み。
19	$MFC_Size	DMA 転送サイズコマンドパラメータの書き込み。
20	$MFC_TagID	タグ識別子コマンドパラメータの書き込み。
21	$MFC_Cmd	DMA コマンドを関連するクラス ID と共に書き込み、キューに入れる。
22	$MFC_WrTagMask	タグマスクの書き込み。
23	$MFC_WrTagUpdate	条件付きまたは無条件のタグステータス更新のリクエストの書き込み。
24	$MFC_RdTagStat	マスク適用後のタグステータスの読み込み。
25	$MFC_RdListStallStat	DMA リストのストール通知ステータスの読み込み。
26	$MFC_WrListStallAck	DMA リストのストール通知受領応答の書き込み。
27	$MFC_RdAtomicStat	最後に完了した即時実行 `MFC` アトミック更新コマンドの終了ステータスの読み込み（" Cell Broadband Engine Architecture"ドキュメントのセクション 9.4 を参照）。

2.5. 即値

様々な長さの符号付きまたは符号なし即値が、多くの命令によって受け入れられています。これらの値を、以下の方法でコード化することが可能です。

- *即値定数または即値定数式。*例えば、命令「`ai $3, $3, -32`」は、レジスタ 3 の各ワード要素から 32 を減算します。
- *PC 相対アドレス。*現在のプログラムカウンタを、ドット（.）記号で表します。例えば、命令「`br .-4`」は、その命令の直前の命令に分岐します。

- *シンボリックラベルアドレス*。これらのアドレスは、リンクエディット時に解釈されます。その間、適切な命令値が、記号の場所にコード化されます。例えば、相対アドレス命令は、相対アドレスによってコード化されます。絶対アドレス命令は、ラベルまたは記号のアドレスによってコード化されます。ハーフワードアドレスを指定する際に`@h`または`@l`を用いると、それぞれ、上位または下位のハーフワードを指定することができます。例えば、以下の命令シーケンスは、*variable* の 32 ビットアドレスをレジスタ 3 にロードします。

```
ilhu $3, variable@h    # load high halfword address of variable
iohl $3, variable@l    # logically OR low halfword address of variable
```

2.6. エラーおよび警告

コード化のエラーを早期に識別するために、アセンブラは、即値がそれぞれの命令によって予期されている範囲を超えるたびに、警告またはエラーを発行します。いくつかの命令に関しては、値が範囲を超えた場合に警告またはエラーを発行することは適していません。表 2-6は、即値オペランドの有効な範囲と、その範囲に対する特例とを示します。

表2-6：有効な即値

即値	最小値	最大値	特例
s3	-4	3	命令 `rotqmbii` に対する制限はありません。指定された即値の最下位 `7` ビットがコード化されます。
s6	-32	31	オプションとして、命令 `rothmi`、`rotmahi`、および `rotqmbyi` の範囲[-31, 0]を超えた値に対して、警告を発行することができます。
s7	-64	63	命令 `rothi`、`roti`、および `rotqbyi` に対する制限はありません。指定された即値の最下位 `7` ビットがコード化されます。オプションとして、命令 `rotmai` および `rotmi` の範囲[-63, 0]を超えた値に対して、警告を発行することができます。
s10	-512	511	オプションとして、命令 `andbi`、`ceqbi`、`cgtbi`、`clgtbi`、`orbi`、および `xorbi` の範囲[-128, 255]を超えた値に対して、警告を発行することができます。
s11	-1024	1023	オプションとして、命令 `hbr`、`hbra`、および `hbrr` に関して、最下位 `2` ビットが `0` ではない値に対して警告を発行することができます。
s14	-8192	8191	オプションとして、命令 `lqd` および `stqd` に関して、最下位 4 ビットが 0 ではない値に対して警告を発行することができます。
s16	-32768	32767	
s18	-131072	131071	オプションとして、命令 `br`、 `bra`、`brasl`、`brhnz`、`brhz`、`brnz`、`brsl`、`brz`、`hbra`、`hbrr`、`lqa`、`lqr`、`stqa`、および `stqr` に関して、最下位 2 ビットが 0 ではない値に対して警告を発行することができます。
scale7	0	127	
u3	0	7	命令 `rotqbii` に対する制限はありません。指定された即値の最下位 `7` ビットがコード化されます。
u5	0	31	

即値	最小値	最大値	特例
u6	0	63	
u7	0	127	命令 cbd、cdd、chd、および cwd に対する制限はありません。アセンブラは、即値の最下位 7 ビットを *u7* パラメータとして無警告でコード化します。
u14	0	16383	
u16	0	65535	即値に先行ビットを追加しない命令に関して、最小値は-32768 に拡張されます。これには、命令 fsmbi、ilh、ilhu、および iohl が含まれます。
u18	0	262143	

以上