



# インテル® IA-64 アーキテクチャ・ ソフトウェア・デベロッパーズ・ マニュアル

スペック・アップデート

---

2001年3月

注意：インテル® Itanium プロセッサは、エラッタと呼ばれる設計上の欠陥またはエラーを含んでいる場合があるため、公表された仕様から外れていることがあります。現在確認されているエラッタについては、本スペック・アップデートに記載されています。

**【輸出規制に関する告知と注意事項】**

本資料に掲載されている製品のうち、外国為替および外国為替管理法に定める戦略物資等または役務に該当するものについては、輸出または再輸出する場合、同法に基づく日本政府の輸出許可が必要です。また、米国産品である当社製品は日本からの輸出または再輸出に際し、原則として米国政府の事前許可が必要です。

**【資料内容に関する注意事項】**

- ・本ドキュメントの内容を予告なしに変更することがあります。
  - ・インテルでは、この資料に掲載された内容について、市販製品に使用した場合の保証あるいは特別な目的に合うことの保証等は、いかなる場合についてもいたしかねます。また、このドキュメント内の誤りについても責任を負いかねる場合があります。
  - ・インテルでは、インテル製品の内部回路以外の使用にて責任を負いません。また、外部回路の特許についても開知いたしません。
  - ・本書の情報はインテル製品を使用できるようにする目的でのみ記載されています。
- インテルは、製品について「取引条件」で提示されている場合を除き、インテル製品の販売や使用に関して、いかなる特許または著作権の侵害をも含み、あらゆる責任を負わないものとします。
- ・いかなる形および方法によっても、インテルの文書による許可なく、この資料の一部またはすべてを複製することは禁じられています。

本資料の内容についてのお問い合わせは、下記までご連絡下さい。

インテル株式会社 資料販売センター  
〒305-8603 筑波学園郵便局 私書箱 115 号  
Fax: 0120-478832

\* 一般にブランド名または商品名は各社の商標または登録商標です。

Copyright © 2001 Intel Corporation.



## 目次

---

改訂履歴 .....	5
はじめに .....	6
変更一覧 .....	7
仕様の変更 .....	8
仕様の明確化 .....	19
文書の修正 .....	28
パフォーマンス監視イベント .....	1



## 改訂履歴

改訂 番号	内容	日付
1.0	初版	2000年4月
2.0	第4巻、7.8節「パフォーマンス監視イベントのリスト」を変更 第2巻、表A-5、pr-writers-intクラスに説明を追加 第2巻、4.4.6.1項、説明を追加	2000年10月
3.0	第2巻、4.1.1.2項、VHPT参照および順方向進行に関して変更 第4巻、第7章「パフォーマンス監視イベント」を改訂(7.6.5項「フロントサイド・パス」を新設、7.8節「パフォーマンス監視イベントのリスト」にパス・モニタを追加、その他変更および修正) 第1巻、4.4.5.3.1項、ALATにヒットするld.c命令のフォルトの説明を追加 第2巻、7.1.1項、IA-32のIBR/DBRの一致に関する説明を追加 第3巻、5-71ページ、IA-32CPUIDの説明を追加 第2巻、8-5、8-26、8-33、8-36の各ページのISRの図を変更	2000年12月
4.0	第2巻、11.8.3項、PAL_CACHE_FLUSHの戻り値を変更 第2巻、11.2節、PAL自己診断コントロールおよびPAL_Aのプロシージャ要件を変更 第2巻、11章、PAL_CACHE_FLUSHの説明を追加 第2巻、4.4.6項、非スペキュレーティブ参照の説明を追加 第2巻、4.1節、RIDおよび推奨ページ・サイズの使い方の説明を追加 第2巻、4.1節、VHPT読み取りのアトミック性の説明を追加 第2巻、4.4.5項、IIPおよびWCフラッシュの説明を追加 第4巻、6.2.4項、IBRおよびDBRのアドレス指定を改訂 第3巻、2.2節、extract、deposit、allocの各命令に関する図を改訂 第2巻、6.4節、RSEおよびPMCの誤植を修正 第2巻、A.4節、DV表を改訂	2001年3月

# はじめに

---

本書は、下表に示した各仕様書の改訂情報である。本書は、仕様の変更、仕様の明確化、文書の修正を1冊にまとめたものである。エラッタについては記載していない。

## 対象文書 / 関連文書

文書名	資料番号
インテル® IA-64 アーキテクチャ・ソフトウェア・デベロッパーズ・マニュアル、第1巻: IA-64 アプリケーション・アーキテクチャ	245317J-002
インテル® IA-64 アーキテクチャ・ソフトウェア・デベロッパーズ・マニュアル、第2巻: IA-64 システム・アーキテクチャ	245318J-002
インテル® IA-64 アーキテクチャ・ソフトウェア・デベロッパーズ・マニュアル、第3巻: 命令セット・リファレンス	245319J-002
インテル® IA-64 アーキテクチャ・ソフトウェア・デベロッパーズ・マニュアル、第4巻: Itanium™ プロセッサ・プログラマーズ・ガイド	245320J-001

## 用語の定義

「仕様の変更」とは、Itanium™ プロセッサの現在文書化されている仕様に対する変更のことである。次のリリースの仕様に反映される。

「仕様の明確化」とは、特定の仕様に関してさらに詳しい説明を加えたり、複雑な設計条件に与える仕様の影響をさらに明確にするものである。次のリリースの仕様に反映される。

「文書の修正」とは、現在文書化されている仕様書の誤字、誤謬、脱落に対する修正である。『インテル® IA-64 アーキテクチャ・ソフトウェア・デベロッパーズ・マニュアル』の次版に反映される。

# 変更一覧

---

下の各表は、『インテル® IA-64 アーキテクチャ・ソフトウェア・デベロッパーズ・マニュアル』に対する仕様の変更、仕様の明確化、文書の修正についてそれぞれまとめたものである。

## 仕様の変更

番号	ページ	仕様の変更
1	8	第2巻：PAL_CACHE_FLUSH の戻り値を追加
2	8	第2巻：PAL 自己診断コントロールおよび PAL_A のプロシージャ要件を変更

## 仕様の明確化

番号	ページ	仕様の明確化
1	19	第2巻：PAL_CACHE_FLUSH についての説明を明確化
2	22	第2巻：非スベキュレーティブ参照に関する説明の明確化
3	22	第2巻：RID および推奨ページ・サイズの使い方に関する説明の明確化
4	23	第2巻：VHPT 読み取りのアトミック性に関する説明を明確化
5	24	第2巻：IIP および WC フラッシュに関する説明を明確化

## 文書の修正

番号	ページ	文書の修正
1	28	第4巻：IBR および DBR のアドレス指定の誤植
2	29	第3巻：extract、deposit、alloc の各命令に関する図を変更
3	30	第2巻：RSE および PMC の誤植
4	31	第2巻：DV 表の誤植

# 仕様の変更

## 1. 第2巻：PAL\_CACHE\_FLUSHの戻り値を追加

11-57 および 11-61 ページに、値 2 の戻り値を新たに追加する。

- a. 11-57 ページの「ステータス」の箇所。次のように、ステータス値 2 の戻り値を新たに追加する。

ステータス値	説明
2	コールはエラーなしで実行を完了したが、このプロシージャの実行中に PMI が発生した。
1	外部イベントが未処理になっているためコールがフラッシュを終了していない。
0	エラーなしでコール終了
-2	引数が無効
-3	エラー発生でコール終了

- b. 11-61 ページの「•Status - 」の箇所。「•Status - 」のすぐ下に次の段落を新たに追加する。

エラーなしでコールの実行は完了したが、このコールの実行中に PMI が発生した場合には、コールから 2 が返される。これは、(*progress\_indicator* をゼロにセットして PAL\_CACHE\_FLUSH を最後にコールしたときに) キャッシュに存在していたキャッシュ・ラインがすべてフラッシュされたにもかかわらず、PMI のハンドリングに関係するコードとデータがそのキャッシュに残っている可能性があることをコール元に通知するものである。

## 2. 第2巻：PAL 自己診断コントロールおよび PAL\_A のプロシージャ要件を変更

11-12 ページ、11.2.1 項の次の部分を差し替える。

～次に、PALE\_RESET は SALE\_ENTRY に分岐し、ファームウェアの更新を必要とするリカバリ条件が存在するかどうかをチェックする。リカバリ条件が存在する場合、SALE\_ENTRY はシステムの更新とリセットを実行する。リカバリ条件が存在しない場合、SAL は PALE\_RESET に戻り、完全なプロセッサ自己診断テストと初期化を実行する。PAL は、IA-32 命令を実行して～

差し替え後（新規段落）：

次に、PALE\_RESET は SALE\_ENTRY に分岐し、ファームウェアの更新を必要とするリカバリ条件が存在するかどうかをチェックする。リカバリ条件が存在する場合、SALE\_ENTRY はシステムの更新とリセットを実行する。ファームウェアのリカバリが不要である場合、SAL は PALE\_RESET に戻り、プロセッサの自己診断テストと初期化を実行する。SAL は PAL のプロセッサ自己診断テストの長さや適用範囲を制御することができる。それには、ファームウェア・リカバリ・ハンドオフ

状態のときに SAL に渡される自己診断テスト・コントロール・ワードを調べて変更するという方法が使われる。自己診断テスト・コントロール・ワードの詳細は、11.2.3 項を参照のこと。

PAL のプロセッサ自己診断テストは 2 つのフェーズに分かれている。第 1 フェーズは、外部メモリがなくても正常に実行できるテスト・プロセッサ機構に組み込まれている。このテストは、SALE\_ENTRY に分岐してファームウェア・リカバリ・チェックをしたあと、SAL が PAL に戻ってきたときに自動的に実行される。この部分は「プロセッサ自己診断テストの第 1 フェーズ」と呼ばれ、プロセッサ・ブート・プロセス中の早い段階で実行されるのが一般的である。第 2 フェーズは、外部メモリがないと正常に実行できないテスト・プロセッサ機構に組み込まれている。このテストは、PAL\_TEST\_PROC という PAL プロシージャの各種パラメータを正しく設定して同プロシージャをコールしたときに実行される。このテストは、通常はプロセッサ・ブート・プロセス中に外部メモリの初期化が終わったあとに実行されるため、「プロセッサ自己診断テストの第 2 フェーズ」と呼ばれている。

PAL は、IA-32 命令を実行して～

11-13 ページ、11.2.2 項「PALE\_RESET 終了状態」の「・GR34 には～」の箇所を次のように変更する。

変更前：

～ SALE\_ENTRY でファームウェアのリカバリを行うのに必要な PAL プロシージャのサブセットだけが使用可能になる。このようなプロシージャとしては以下のものがある。

- PAL\_PLATFORM\_ADDR、PAL\_PROC\_GET\_FEATURES ( 現在の設定値を調べる )
- PAL\_PROC\_SET\_FEATURES ( キャッシュを使用可能 / 使用不可にする )
- PAL\_CACHE\_INIT ( 全レベル、両方、制限なし )
- PAL 認証を行うためのプロセッサ・モデル固有の PAL プロシージャ

変更後：

～ 各種 PAL プロシージャのうち、SALE\_ENTRY でファームウェアのリカバリを行うのに必要な一部のプロシージャだけが利用できる。この種のプロシージャとしては、PAL\_PLATFORM\_ADDR のほか、プロセッサ・モデル固有の PAL プロシージャ (PAL 認証を行うために必要) がある。

11-13 ページ、11.2.2 項「PALE\_RESET 終了状態」に、次のように「・GR37 には～」を追加する。

- GR37 には、11.2.3 項で定義している自己診断テスト・コントロール・ワードが含まれる。このコントロール・ワードはプロセッサ・モデル固有のものであり、自己診断コントロールが実装されているかどうかと、制御可能なビット数を SAL に通知する。自己診断コント

ロールが実装されている場合は、ファームウェアのリカバリ・チェックをしてから SAL が PAL に戻ってきたときに、PAL がこの値を読み取る。自己診断テスト・コントロールがサポートされていない場合は、ファームウェアのリカバリ・チェックをしてから SAL が PAL に戻ってきても、このレジスタは無視される。

11-14 ページの「・キャッシュ：」および「・TLB：」の箇所を次のように変更する。

変更前：

- キャッシュ：プロセッサの内部キャッシュは使用可能な状態で無効化されている。キャッシュ自体およびキャッシュからプロセッサ・コアへの経路はテスト済みである。外部メモリからキャッシュへの経路はテストされていない。
- TLB：TR と TC は無効化されていたすべてのエントリで初期化される。TLB は、 $PSR.it=PSR.dt=PSR.rt=0$  で、かつプロセッサの自己診断テストの 2 番目のフェーズが終了するまで使用できないため、利用できない (PAL\_TEST\_PROCESSOR を参照)。

変更後：

- キャッシュ：プロセッサの内部キャッシュは使用可能であるが無効な状態にある。自己診断テスト・コントロール・ワードによって指示されていない限りは、プロセッサ自己診断テストの第 1 フェーズにおいて、キャッシュ自体とキャッシュからプロセッサ・コアへの経路が検証される。外部メモリからキャッシュへの経路については、プロセッサ自己診断テストの第 2 フェーズまでテストできない。
- TLB：TR と TC は、無効化されていたすべてのエントリで初期化される。 $PSR.it=PSR.dt=PSR.rt=0$  であるため、TLB は使用できない。プロセッサ自己診断テストの第 2 フェーズまでは TLB は完全にはテストできない。

11-16 ページ。自己診断テスト状態パラメータの一部である *state* フィールドの説明箇所を次のように変更する。

変更前：

- *state* - 自己診断テスト後のプロセッサの状態を示す 2 ビット・フィールド。

変更後：

- *state* - 自己診断テスト後のプロセッサの状態を示す 2 ビット・フィールド。自己診断テスト・コントロール・ワードを変更することによって SAL が PAL に対して自己診断テストをいくつか省略するよう指令した場合は、これら自己診断テストに関連していくつか不合格になる項目があったとしても、この状態には反映されない。

11-17 ページ、FUNCTIONALLY RESTRICTED についての説明箇所を次のように変更する。

変更前：

- PAL\_RESET でのテスト中には、プロセッサが制御するキャッシュとレジスタ・ファイル間の経路が機能していて、かつ PAL\_TEST\_PROCESSOR でのテスト中にメモリからキャッシュを経由してレジスタ・ファイルに至る経路がすべてが機能していなければならない。

変更後：

- プロセッサで制御するキャッシュとレジスタ・ファイルを結ぶ経路が機能していることが判明している。プロセッサ・キャッシュとメモリを結ぶ経路は、PAL\_TEST\_PROC プロシージャで呼び出すプロセッサ自己診断テストの第 2 フェーズまでは検証することができない。

11-17 ページ、11.2.3 項「PAL 自己診断テスト・コントロール・ワード」を次のように新設する。

PAL 自己診断テスト・コントロール・ワードは 48 ビット値である。このビット・フィールドは図 11-9 に定義されている。

48 ビットの並びを示す図 11-9 を追加する。ビット 47 には *cs* という名前を付け、ビット 46 ~ 0 には *test\_control* という名前を付ける。

- *test\_control* - これは、プロセッサ・モデル固有の順序付きコントロール・ワードである。ユーザはこれを使って、プロセッサ自己診断テストの長さを実行時間を制御できる。このコントロール・ワードは、最長のテストから最短のテストまで順番どおりに並んでいる。ビット 0 は最長のテストを制御する。

PAL は、47 ビットから成る *test\_control* ワードの一部を実装していないことがある。任意のビットをゼロにすることによってそのビットを制御できる場合は、PAL は他と通信が可能である。そのビットを制御できない場合は、PAL はそのビットに 1 を立てる。

プロセッサ自己診断テストは 2 つのフェーズに分かれているため、PAL はテスト・コントロール・ビットを 2 セット持つことになる。

ファームウェアのリカバリ・チェックを目的として PAL から SAL へハンドオフするときに、実装されている各 *test\_control* ビットに関する情報が PAL から SAL に提供される。これらの *test\_control* ビットにより、プロセッサ自己診断テストの第 1 フェーズを制御する。この情報は、PAL\_TEST\_INFO という PAL プロシージャ・コールによって、第 1 フェーズ、第 2 フェーズのどちらのプロセッサ・テストにも提供される。どちらのフェーズに提供されるかは、コール元がどちらの情報を要求しているかによって異なる。

PAL がこれらのビットを入力パラメータとして解釈するのは、次の 2 つの場合である。1 つは、ファームウェアのリカバリ・チェックのあと、SAL が PAL に制御権を返すときである。もう 1 つは、PAL\_TEST\_PROC をコールするときである。PAL は、これらのビッ

トを解釈するときに、実装されている *test\_control* ビットだけを解釈し、実装されていない *test\_control* ビットの値は無視する。

実装されている各ビットを解釈して、ビットがゼロである場合にはそのテストが実行され、ビットが 1 である場合はそのテストは省略される。

*cs* ビットによって制御権のないことがわかった場合、各 *test\_control* ビットは、無視されるか、あるいはコール元がこれらのビットをセットするとプロシージャ・コールで不正な引数を生成する。

- *cs* - コントロール・サポート (Control Support) : このビットは、自己診断テスト・コントロール・ワードで PAL 自己診断テストの制御ができるかどうかを定義する。このビットが 0 の場合は、自己診断テスト・コントロール・ワードを用いてプロセッサの自己診断テストを制御することはできない。このビットが 1 の場合は、自己診断テスト・コントロール・ワードを用いてプロセッサの自己診断テストを制御することができる。

制御できない場合は、ファームウェアのリカバリ・チェックのあと SAL と PAL と間でハンドオフが行われるときに GR37 は無視され、また、ユーザが自己診断テスト・コントロール機構を使おうとすると、プロセッサ自己診断テストに関連する各 PAL プロシージャから不正な引数が返されることがある。

11-31 ページ、表 11-16 のプロシージャ名と説明を変更する。

変更前 :

- PAL\_MEM\_FOR\_TEST

変更後 :

- PAL\_TEST\_INFO

説明を次のように変更する。

変更前 :

最新のプロセッサ自己診断テストに必要なメモリ量を返す。

変更後 :

PAL\_TEST\_PROC プロシージャに渡されるメモリ・バッファに必要なアラインメント条件とサイズ条件を返す。ほかに、プロセッサ自己診断テストに使われる自己診断テスト・コントロール・ワードに関する情報も返す。

11-91 ページ、PAL\_MEM\_FOR\_TEST プロシージャの説明。

a. 「目的」の箇所を次のように変更する。

PAL\_TEST\_PROC プロシージャに渡されるメモリ・バッファに必要なアラインメント条件とサイズ条件を返す。ほかに、プロセッサ自己診断テストに使われる自己診断テスト・コントロール・ワードに関する情報も返す。

## b. 次の引数を新たに追加する。

引数	説明
test_phase	プロセッサ自己診断テストのどちらのフェーズに関する情報が要求されているのかを示す、符号なし整数。0 はプロセッサ自己診断テストの第 2 フェーズを示し、1 はプロセッサ自己診断テストの第 1 フェーズを示す。ほかの値はすべて予約されている。

## c. 次の戻り値を新たに追加する。

戻り値	説明
st_control	プロセッサ自己診断テストの制御ができるかどうかと、test_control フィールドのどのビットが使えるのかを示す、48 ビット幅のビット・フィールド。

## d. 「説明」の箇所を次のように変更する。

PAL\_TEST\_INFO は、PAL\_TEST\_PROC プロシージャに渡されるメモリ・バッファに対するサイズ条件とアラインメント条件を返すほかに、*test\_phase* 入力引数を基にして自己診断テスト・コントロール・ワードの実装状態に関する情報も返す。自己診断テスト・コントロール・ワードの詳細は、11.2.3 項を参照のこと。

*test\_phase* が 0 のときは、プロセッサ自己診断テストの第 2 フェーズに関する情報が返される。このテストは、外部メモリがないと正しく実行できないテストである。*test\_phase* が 1 のときは、プロセッサ自己診断テストの第 1 フェーズに関する情報が返される。このテストは、通常は PALE\_RESET 時に実行されるテストであり、外部メモリがなくても正しく実行できる。プロセッサ自己診断テストの第 1 フェーズに関する情報が要求された場合は、メモリ・バッファとアラインメント引数も返される。その理由は、PAL\_TEST\_PROC プロシージャでこれらのテストを実行すると、場合によってはプロセッサ状態をこのメモリ・バッファに保存およびリストアしなければならないからである。

11-102 ページ。PAL\_TEST\_PROC プロシージャの説明の入力引数を次のように変更する。

変更前：

引数	説明
index	PAL プロシージャ・リスト内の PAL_TEST_PROC のインデックス。
test_address	プロセッサの自己診断テストで使用するメイン・メモリの 64 ビット物理アドレス。渡されるメモリはキャッシング可能で、ビット 63 が 0 でなければならない。
test_size	プロセッサの自己診断テストに使用するメイン・メモリのバイト数。
attributes	テストするメモリ属性の 16 ビット・マスク。

変更後：

引数	説明
index	PAL プロシージャ・リスト内での PAL_TEST_PROC のインデックス。
test_address	プロセッサの自己診断テストに使われるメイン・メモリ領域の 64 ビット物理アドレス。渡されるメモリ領域はキャッシュ可能でなければならず、ビット 63 が 0 でなければならぬ。
test_info	渡されるメモリ・バッファのサイズを指定する入力引数、および実行されるプロセッサ自己診断テストのフェーズ。図 11-9 を参照のこと。
test_params	自己診断テスト・コントロール・ワードを指定する入力引数、およびメモリ・バッファのメモリ属性として許容できる属性。図 11-9 を参照のこと。

a. 「説明」の箇所を次のように変更する。

PAL\_TEST\_PROC プロシージャは、*test\_info* および *test\_control* という入力パラメータから指令されたとおりに、プロセッサ自己診断テストのいずれか一方のフェーズを実行する。

*test\_address* は、PAL\_TEST\_PROC に使われる連続メモリ領域を指す。このメモリ領域は、PAL\_TEST\_INFO から返されるアラインメント戻り値に指定されているとおりにアラインメントされなければならない。さもないと、このプロシージャは無効な引数戻り値とともに返されることになる。PAL\_TEST\_PROC ルーチンを実行するには、メモリが初期化されていることと、割り当てられているメモリ内に既知の未修復エラーがないことが必要である。

*test\_info* 入力パラメータは、当該プロシージャに渡されるメモリ・バッファのサイズを指定したり、プロセッサ自己診断テストのどちらのフェーズ（第 1 フェーズ、第 2 フェーズ）の実行が要求されているかを指定する。

上位 8 ビット (*test\_phase*) と下位 56 ビット (*buffer\_size*) の並びを示す図を追加する。

- *buffer\_size* は、このプロシージャに渡されるメモリ・バッファのバイト数を示す。*buffer\_size* のサイズは、PAL\_TEST\_INFO の *bytes\_needed* 戻り値以上のサイズでなければならない。さもないと、このプロシージャは無効な引数戻り値とともに返される。
- *test\_phase* は、プロセッサ自己診断テストのどちらのフェーズを実行するよう要求されているかを定義する。値が 0 のときは、プロセッサ自己診断テストの第 2 フェーズが実行されることを示している。プロセッサ自己診断テストの第 2 フェーズは、外部メモリがないと正しく実行することのできないテストである。値が 1 のときは、プロセッサ自己診断テストの第 1 フェーズが実行されることを示している。プロセッサ自己診断テストの第 1 フェーズは、PALE\_RESET のときに実行されるテストであり、外部メモリがなくても正しく実行できる。コール元がこのプロシージャ・コールを使ってプロセッサ自己診断テストの第 1 フェーズを実行するよう要求する場合、PAL の呼び出し規約どおりに状態を保存しリストアするためには、メモリ・バッファが必要になることがある。PAL\_TEST\_INFO プロシージャは、メモリ・バッファの必要条件についてコール元に通知する。

*test\_params* 入力引数は、このプロシージャに渡されるメモリ・バッファと一緒に使えるメモリ属性を指定する。また自己診断テスト・コントロール・ワードの指定にも使われる。*test\_control* という自己診断テスト・コントロール・ワードは、*test\_phase* パラメータに指定されているプロセッサ自己診断テスト・フェーズの実行時間と適用範囲を制御する。

上位 48 ビット (*test\_control*)、それに次ぐ 8 ビット (予約済み)、下位 8 ビット (*attributes*) の並びを示す図を追加する。

- *attributes* は、このプロシージャに渡されるメモリ・バッファと一緒に使えるメモリ属性を指定する。この属性パラメータは一種のベクタであり、各ビットが、アーキテクチャから定義される仮想メモリ属性の 1 つを表現している。ビット・フィールドの位置は、9.4 節「メモリ属性」に定義されている数値メモリ属性エンコーディングと対応している。コール元はメモリ・バッファのキャッシュ可能属性をサポートしていなければならない。さもないと無効な引数が返される。
- *test\_control* は、渡された *test\_phase* に対応している自己診断テスト・コントロール・ワードである。この *test\_control* は、*test\_phase* 入力引数で指定されるプロセッサ自己診断テストの適用範囲と実行時間を指令する。自己診断テスト・コントロール・ワードに関する情報は 11.2.3 項に示してある。また、この機能が実装されているか否かの情報と、対応しているビット数に関する情報は、PAL\_TEST\_INFO プロシージャ・コールにより取得できる。この機能がプロセッサに実装されているのであれば、コール元は *test\_control* のいくつかのビットを 1 にセットすることにより、プロセッサ自己診断テストのいくつかの部分を都合に合わせて省略することができる。ビットが 0 の場合は、そのテストは実行される。実装されていないビットの値は無視される。自己診断テスト・コントロール・ワードが実装されていないことが PAL\_TEST\_INFO によりわかった場合は、コール元がどの *test\_control* ビットをセットしていても、このプロシージャは無効な引数ステータスとともに返される。

PAL\_TEST\_PROC は、自己診断テストの終了後、プロセッサの状態について分類を行う。CATASTROPHIC FAILURE、FUNCTIONALLY RESTRICTED、PERFORMANCE RESTRICTED、HEALTHY という 4 つ状態のうちいずれかに分類される。プロセッサ自己診断テストの各状態については、11-11 ページの表 11-14 で解説している。

PAL\_TEST\_PROC が FUNCTIONALLY RESTRICTED か PERFORMANCE RESTRICTED かのいずれかの状態で返ってきた場合、*self-test\_status* 戻り値によって、発生した障害の性質に関する補足情報が示される。CATASTROPHIC FAILURE の場合はプロシージャは返ってこない。

このプロシージャは、*attributes* ビット・フィールドに示されたメモリ属性を使って、渡されたバッファへのメモリ・アクセスのみを実行する。コール元は、プロシージャに渡されるメモリ領域が必ずコヒーレントな状態になるようにしなければならない。

PAL\_TEST\_PROC は、プロセッサのテストに必要なときには PSR ビットがシステム・レジスタのいずれかを変更することがある。変換キャッシュを除き、これらのビットまたはレジスタは、PAL\_TEST\_PROC から抜けるときにリストアされなければならない。変換キャッシュは、テストの結果に従って削除される。PAL\_TEST\_PROC は、キャッシュの内容をすべて自由に無効にすることができる。コール元がキャッシュの内容に依存している場合は、このコールを実行する前にキャッシュの内容をフラッシュする必要がある。レジスタ・スタックの内容が必要とされる場合は、有効なメモリ・ロケーションへのスピルとフィルを処理できるように RSE が正しく設定されていないと PAL\_TEST\_PROC は使えない。PAL\_TEST\_PROC を使うためには、渡されるメモリ・バッファが、同一システム内でこのプロシージャを同時に実行している他のいくつかのプロセッサと共有されていないことが必要条件となる。PAL\_TEST\_PROC は、非コヒーレントな方式でこのメモリ領域を使用する。

---

### 3. 第 2 巻 : VHPT 参照と順方向進行

4-5 ページ、4.1.1.2 項の「IA-64 コードが順方向に進行することを保証するため、プロセッサおよびソフトウェアは以下の規約に従わなければならない。」に続く 2 つ目の箇条書き項目を次のように変更する。

変更後：

プロセッサは、場合によっては、最後に挿入された TC エントリを無効にすることがある。PSR.ic が 0 である間は、プロセッサは、最後に挿入された TC エントリがどの参照からも可視（参照可能）であることを保証しなければならない。プロセッサは最終的には、rfi 命令によって PSR.ic を 1 にセットし、PSR.ic が 1 の状態で命令が 1 個以上実行され、フォルトも割り込みも発生せずに実行が完了するまで、最後に挿入された TC エントリがどの参照からも可視（参照可能）であることを保証しなければならない。最後に挿入された TC エントリは、場合によってはこの時点よりも前に削除されることがあり、その場合は、次のフォルトが発生したときにこの TC エントリが挿入し直されるようソフトウェア上で準備しておく必要がある。例えば、eager（高頻度）モードか mandatory（強制）モードで RSE が動作している場合は、スペキュレーティブな VHPT 参照か、あるいはリスタート命令による他からの割り込みによって、ソフトウェアによって挿入された TC エントリの位置が動いてしまうことがあるが、あとでソフトウェアによって同じ TC エントリを挿入し直す場合には、最終的にプロセッサはリスタート命令を完了させて、順方向進行が確実に行われるようにしなければならない。これは、リスタート命令が他のフォルトを引き起こし、それを処理しない限りはリスタート命令の実行が完了できないような場合であっても同じことである。rfi 以外の命令で PSR.ic を 1 にセットする場合、プロセッサは順方向進行を保証できない。

4-5 ページ、4.1.1.2 項の上記の箇条書き項目の下に次の 2 つの箇条書き項目を挿入する。

- オーバーラップしているエントリ (同じサイズ以上) が VHPT の中にあるときにソフトウェア上で TLB にエントリを 1 個挿入し、かつ VHPT ウォーカが使用可能な状態にある場合は、順方向進行は保証されない。4-15 ページの 4.1.5.2 項「VHPT の検索」を参照のこと。
- ソフトウェア上では、TC エントリが挿入されてから、PSR.ic を 1 にしてその命令を実行するまでの間は、物理アドレスを持つメモリか、TR でマップされる仮想アドレスを持つメモリか、あるいは挿入されたばかりの変換値によってマップされるアドレスかのいずれかしか参照しないことがある。これは、順方向進行に関するエントリによって決まる。またソフトウェア上で、PSR.ic が 1 である同一命令を繰り返し実行しようとすることもある。この種の参照以外のメモリ参照をソフトウェアで実行する場合は、プロセッサは順方向進行を保証できない。

4-14 ページ、4.1.5 項の末尾に次の段落を追加する。

TLB に挿入されたエントリをもっと明示的にソフトウェアで制御する必要がある場合、あるいは同じ仮想アドレス範囲に対してさまざまなマッピングを持つ VHPT をソフトウェアでプログラムする場合は、さらに何らかの操作をしないと順方向進行を保証できないことがある。4-15 ページの 4.1.5.2 項「VHPT の検索」を参照のこと。

4-15 ページ、4.1.5.2 項の末尾に次の段落を追加する。

VHPT 参照は、プロセッサの VHPT ウォーカによってスペキュレーティブに実行されることがある。また、非スペキュレーティブな方法により実行された命令がきっかけとなって行われる VHPT 参照は、プログラム順に実行するときには不要である。したがって、ウォーカが使用可能な状態にあり、かつ同じ仮想アドレス範囲をマップしている複数のエントリが VHPT に含まれている場合は、それらのエントリをソフトウェアでうまく設定して、この仮想アドレス範囲のどの部分を変換する場合でも当該エントリのどれでも使用できるようにしなければならない。また、ソフトウェア上で順方向進行に必要な TLB に変換値を挿入し、この変換値が、同一アドレスを対象とした VHPT 参照のときに挿入された変換値よりもページ・サイズが小さい場合、ソフトウェア上で VHPT ウォーカを無効にしないと順方向進行を保証できないことがある。これは、挿入されたこの変換値を利用するためには、その前に VHPT ウォーカで当の変換値の格納場所を強制的に変えなければならないことであるからである。

#### 4. **第 4 巻、第 7 章「パフォーマンス監視イベント」(本スペック・アップデートの巻末に付した文章)を改訂**

第 4 巻、第 7 章「パフォーマンス監視イベント」については、新しい内容を追加したほか、読みやすくするために一部修正を加えた。便利のように、本スペック・アップデートの巻末に第 7 章全体を付した。

**注意:** 『インテル® IA-64 ソフトウェア・デベロッパーズ・マニュアル』第4巻の第7章については章全体を差し替えること。

# 仕様の明確化

---

## 1. 第 2 巻 : PAL\_CACHE\_FLUSH についての説明を明確化

11-33 および 11-34 ページの表 11-19 :

- psr.ic ビットの説明欄の脚注参照記号 (a.) を削除する。
- 同表の下の脚注 (a.) を削除する。

11-42 ~ 11-45 ページ。PAL\_CACHE\_FLUSH プロシーダを次のように変更する。

- a. 11-42 ページ。戻りステータス値 1 の説明を次の文に差し替える。

未処理の割り込みがあるためにコールがフラッシュを終了していない。

- b. 11-43 ページの「*int* - 」の箇所を次の文に差し替えて、PMI に関する部分をすべて削除する。

*int* - 指定された *cache\_type* がフラッシュされている間、外部割り込みに対してプロセッサが定期的にポーリングを実行するかどうかを示す 1 ビット・フィールド。

このビットが 0 の場合、マスクされていない外部割り込みはポーリングされない。プロセッサは、指定された *cache\_type* に含まれるキャッシュ・ラインがすべてフラッシュされるまでは、未処理のマスクされていない外部割り込みをすべて無視する。プロセッサのキャッシュのサイズ、バス帯域幅、実装特性によってはキャッシュをフラッシュするのにかなり時間がかかる場合があり、その結果、割り込み応答時間に遅れの出る可能性が高くなり、場合によっては I/O デバイスに障害が発生することもある。

このビットが 1 の場合は、外部割り込みに対して定期的にポーリングが実行され、どれか 1 つでも見つかり、その外部割り込みによってプロシーダは終了する。マスクされていない外部割り込みが未処理状態になると、このプロシーダがリターンし、指定された *cache\_type* のキャッシュ・ラインがすべてフラッシュされる前に、コール元は割り込みを処理することができる。

- c. 11-44 ページの 3 段落目および 5 段落目。psr.ic および psr.i の設定値に関する文をいくつか削除する。

変更前 :

このプロシーダは、1 回のフラッシュ操作で *cache\_type* で指定されたすべてのキャッシュとこれらのキャッシュ内のセットおよびアソシエイティビティをすべてフラッシュする。指定された *cache\_type*

で確実にフラッシュされるのは、最初に *progress\_indicator* を 0 にセットして PAL\_CACHE\_FLUSH をコールする前にそのキャッシュに存在したキャッシュ・ラインだけである。

このプロシージャは PSR.i と PSR.ic を 0 にセットしてコールして、このプロシージャがキャッシュのフラッシュを開始する前やこのプロシージャの終了処理中に外部割り込みが発生しないようにしなければならない。PSR.i と PSR.ic は、*int* フィールドの値に関わらず 0 にセットすべきである。

変更後：

このプロシージャは、*cache\_type* で指定されたすべてのキャッシュについてはもちろんのこと、それらのキャッシュに含まれているセットもアソシエイティビティもすべて 1 回の操作でフラッシュする。*cache\_type* を指定して確実にフラッシュできるのは、*progress\_indicator* を 0 にセットした状態で最初に PAL\_CACHE\_FLUSH を呼び出すより前にそのキャッシュに存在したキャッシュ・ラインだけである。

- d. 11-44 ページ。このプロシージャの終了後もキャッシュに残っている内容について列挙した箇条書き項目において次のように新しく項目を追加し、PMI に関する部分を削除する。

変更前：

以下の条件のために、ソフトウェア側ではこのプロシージャがフラッシュ・パス全体を終了した時点で、指定された *cache\_type* でクリーン状態または変更状態にあるキャッシュ・ラインがすべて空であると想定することはできない。

- 割り込み後、フラッシュ・パスは割り込み発生位置 (*progress\_indicator* で指定される) から再開される。フラッシュ・パス中でも割り込みハンドラは実行されるため、指定されたキャッシュには、すでにフラッシュされたキャッシュ・セクションに新たに変更された可能性のあるキャッシュ・ラインが入ることがある。
- 上に説明したように、このプロシージャがコールされる前に開始されていたプリフェッチはディセーブルにされ、キャッシュからフラッシュされる。ただし、ITLB または DTLB でスペキュレーティブな変換が存在していた場合、スペキュレーティブ命令またはデータ・プリフェッチ操作が、フラッシュ終了後直ちに更新されていないキャッシュ・ラインを再ロードする可能性がある。プリフェッチが発生しないようにするために、ソフトウェア上でこのルーチンをコールする前に、スペキュレーティブ変換をすべて削除する必要がある。あるいは、ソフトウェアで PSR.it、PSR.dt、および PSR.it を 0 にセットして TLB をディセーブルにすることもできる。
- 指定されたキャッシュには、キャッシュをフラッシュするのに必要な PAL ファームウェア・コードのキャッシュ・エントリが含まれていることがある。

## 変更後：

このプロシージャがフラッシュ・バス全体の処理を完了したときであっても、指定された *cache\_type* の中に変更済みキャッシュ・ラインや完全に空になったキャッシュ・ラインが含まれていないとソフトウェア側で想定することはできない。それは以下の諸条件のためである。

- 割り込み後は、*progress\_indicator* で指定される割り込み発生地点からフラッシュ・バスが再開される。フラッシュ・バスの処理中でも割り込みハンドラは実行されるため、指定されたキャッシュは、すでにフラッシュされたキャッシュ領域の中に新しいキャッシュ・ラインを含んでいることがある。また、変更されたキャッシュ・ラインを含んでいることもある。コール元は、*operation* パラメータの *int* ビットを利用して、割り込みに対するポーリングをこのプロシージャで実行するかどうかを指定する。
  - 先に説明したとおり、このプロシージャがコールされる前に開始されたプリフェッチはディセーブルにされ、キャッシュからフラッシュされる。ただし、ITLB か DTLB のいずれかがスベキュレーティブな変換値を含んでいる場合は、フラッシュ終了後直ちに、スベキュレーティブ命令かデータ・プリフェッチ操作かのいずれかによって、変更されていないキャッシュ・ラインをロードし直すことがある。プリフェッチが行われなくするために、このルーチンをコールする前に、スベキュレーティブな変換値のすべてをソフトウェアで削除しなければならない。また、ソフトウェアで *PSR.it*、*PSR.dt*、*PSR.rt* をすべて 0 にセットして TLB をディセーブルにすることもできる。
  - 指定されたキャッシュは、キャッシュをフラッシュするのに必要な PAL ファームウェア・コードのキャッシュ・エントリを含んでいることがある。
  - *psr.ic* = 1 にセットした状態でコールされ、そのコールの実行中に PMI 割り込みが発生した場合、指定されたキャッシュは PAL および SAL の PMI コードを含んでいることがある。
  - これらのハンドラがキャッシュ可能モードで動作していて、マシン・チェックか INIT イベントが発生した場合、指定されたキャッシュは、SAL または OS のマシン・チェックか INIT コードのいずれかを含んでいることがある。
- e. 11-45 ページ。以下の段落から PMI に関する部分を削除する。

## 変更前：

順方向の進行を保証するために、*PAL\_CACHE\_FLUSH* は未処理の外部割り込みまたは PMI をサンプリングする前に、キャッシュ・フラッシュ・シーケンスを少なくとも 1 キャッシュ・ラインだけ進める。割り込みがポーリングされる前にどの程度フラッシュされるかは、プロセッサ・モデルによって異なる。

変更後：

必ず順方向進行となるよう、未処理の外部割り込みに対してサンプリングを行う前に、PAL\_CACHE\_FLUSH は、キャッシュ・フラッシュ・シーケンスを最低でもキャッシュ・ライン 1 本分は先まで進んでいる。割り込みに対してポーリングが実行される前にどのくらいの量がフラッシュされるかは実装状態により異なる。

f. 11-45 ページの「*status* - 」および「*vector* - 」の説明の箇条書き項目を次の文に差し替えて、PMI に関する部分を削除する。

- *status* - エラーが発生しなくても、マスクされていない未処理の外部割り込みが発生するとコールから 1 が返される。続けてキャッシュをフラッシュするときは、コール元は、*progress\_indicator* 戻り値として返された値をセットして PAL\_CACHE\_FLUSH をもう一度コールしなければならない。

エラーなしでコールが終了すると、コールから 0 が返される。*progress\_indicator* を 0 にセットして最後に PAL\_CACHE\_FLUSH をコールしたときにキャッシュに含まれていたキャッシュ・ラインはすべてフラッシュされ、場合によっては無効化されることもある。中間コールもすべて、正しい *progress\_indicator* を使用していなければならない。さもないと、どのような動作になるかわからない。

- *vector* - 戻りステータスが 1 であり、マスクされていない未処理の外部割り込みが原因でこのプロシージャが終了した場合、このフィールドから割り込みベクタ番号が返される。外部割り込みは削除されているはずである。この割り込みは「in-service ( サービス中 ) 」と見なされるため、ソフトウェア上では、指定されたベクタに対するこの割り込みを処理してから EOI を発行しなければならない。戻りステータスが 1 でない場合は、どのような値が返されるかわからない。

## 2. 第 2 巻：非スペキュレーティブ参照に関する説明の明確化

4-30 ページ、4.4.6 項の箇条書き項目のすぐ下に次の文を追加する。

これらの必要条件に合致する参照のことを「非スペキュレーティブ参照」と呼んでいる。これらの条件に合致すれば、Instruction Debug ( 命令デバッグ ) フォルトか External ( 外部 ) 割り込みを発生する命令フェッチもやはり非スペキュレーティブ参照であると言える。

## 3. 第 2 巻：RID および推奨ページ・サイズの使い方に関する説明の明確化

4-11 ページ、4.1.2 項、表 4-5 の「説明」欄の ps フィールドに関して、以下の脚注を追加する。

1. このフィールドの詳しい使い方については、4.1.6 項「VHPT ハッシング」を参照のこと。

4-19 ページ、4.1.6.2 項の末尾の「1. 」を次の文に差し替える。

1. ソフトウェア上では、必ず、一意のリージョン識別子 1 個につき推奨ページ・サイズを 1 個しか使うことができない。さもないとプロセッサの動作がどうなるかわからない。
- 4-19 ページ、4.1.6.2 項の末尾に次のように「4. ~」を追加する。
4. 推奨ページ・サイズの異なるリージョン識別子を再利用するには、当該 RID の変換値のうち挿入可能なものを VHPT が含んでいないことを確認し、当該 RID を使用していた可能性のあるすべてのプロセッサから当該 RID に関する変換値すべてをパージし、そのあとで新しい推奨ページ・サイズでリージョン・レジスタを更新するという一連の処理をソフトウェアで実行しなければならない。

#### 4. 第 2 巻：VHPT 読み取りのアトミック性に関する説明を明確化

4-17 ページ、4.1.5.4 項の次の段落と箇条書き項目について。

ロング形式の VHPT エントリのアトミックな更新を保証するためには、ソフトウェア上で以下のことを行う必要がある。

- メモリ中の VHPT エントリに対して複数の非アトミックな更新を行う前に、そのエントリの  $ti$  ビットを 1 にセットする必要がある。
- メモリ中の VHPT エントリに対して複数の非アトミックな更新を行った後、そのエントリの  $ti$  ビットを 0 にクリアすれば、タグの照合を再びイネーブルにすることができる。

最初の文を次のように変更する。

マルチプロセッサ・システムの場合は、ソフトウェアで次のようにすることにより、ロング形式の VHPT エントリを確実にアトミックに更新することができる。

箇条書き項目に続けて次の段落を追加する。

メモリに格納されている VHPT の更新結果を識別するためには、その前に、この  $ti$  ビットを 1 にセットするストアの識別ができなければならない。それには、 $mf$  命令を実行するか、解放ストアを使って VHPT エントリを更新すればよい。同様に、 $ti$  ビットがクリアされたことを識別するためには、その前に、VHPT エントリの更新結果がすべて識別できなければならない。それには、 $mf$  命令を実行するか、解放ストアを使って  $ti$  ビットをクリアすればよい。

4-19 ページ、4.1.7 項の次の段落について。

VHPT ウォーカによる VHPT の参照は、PSR.cpl の状態に関係なく、特権レベル 0 で行われる。VHPT のバイト・オーダーは、DCR.be の状態によって決定される。DCR.be=1 であれば、VHPT ウォーカの参照はビッグ・エンディアンメモリ形式を使って行われ、0 の場合はリトル・エンディアンで行われる。VHPT ウォーカは、最低 8 バイトのアトミックなアクセスのシーケンスとして、VHPT のエントリを参照する。ロング形式 VHPT の参照は、32 バイトの参照として、データ・ブレイクポイント・レジスタと照合される。

次の文を削除する。

VHPT ウォーカは、最低 8 バイトのアトミックなアクセスのシーケンスとして、VHPT のエントリを参照する。

4-20 ページ、4.1.7 項の次の段落と箇条書き項目について。

プロセッサの VHPT ウォーカは、以下に従って、メモリからの VHPT エントリの読み取りおよび挿入をアトミックに行う必要がある。

- ウォーカがメモリからエントリをアトミックに読み取っていないときに、登録中のエントリの一部に対する更新が検出された場合、ウォーカはその挿入を中止し、Instruction/Data TLB Miss (命令 / データ TLB ミス) フォルトを渡さなければならない。
- ウォーカがメモリからエントリをアトミックに読み取っているときに、登録中のエントリの一部に対する更新が検出された場合、ウォーカはその挿入を中止して Instruction/Data TLB Miss (命令 / データ TLB ミス) フォルトを渡すか、あるいは更新を無視して古いエントリ全体を登録しなければならない。
- TLB パージ操作 (ptc.l、ptc.e、ローカルまたはリモートの ptc.g、ptc.ga、ptr.i または ptr.d) のパージ・アドレス範囲が、ウォーカが挿入しようとしている仮想アドレスとオーバーラップする場合、ウォーカはその挿入を中止して Instruction/Data TLB Miss (命令 / データ TLB ミス) フォルトを渡すか、あるいはウォーカが挿入を完了するか参照を中止するまでパージ操作を遅らせる必要がある。

次の文に差し替える。

プロセッサの VHPT ウォーカがメモリから VHPT エントリを読み取って挿入する処理は、アトミックに実行されなければならない(ショート形式の場合は 8 バイト単位でアトミックな読み取り / 挿入を行い、ロング形式の場合は 32 バイト単位でアトミックな読み取り / 挿入を行う)。

4-32 ページ、4.4.7 項の最初の段落は次のように始まっている。

第 1 巻の 4.4.7 項「メモリ・アクセスの順序」で説明されているとおり、同じメモリ位置への RAW (リード・アフター・ライト)、WAW (ライト・アフター・ライト)、および WAR (ライト・アフター・リード) の依存関係 (メモリ依存関係) は、プロセッサによってプログラムの順序で実行される。

この文に次の脚注を追加する。

VHPT 参照はプログラムの実行を基準にするとやや非同期的に実行されるが、ウォーカで読み取った VHPT の内容を個別に見ると、プログラム順序内での何点かではアトミックに実行されたような形となる。

## 5. 第 2 巻 : IIP および WC フラッシュに関する説明を明確化

4-29 ページ、4.4.5 項の次の第 4 段落を差し替える。

IA-64 の release (解放) 操作 (コーレシング属性を持つページを参照するかどうかを問わない)、または IA-64 の fence (フェンス) タイプの命令においては、命令そのものが可視 (参照可能) になる前に、書き込みコーレシングされたデータが強制的に可視にされる。(release および fence の命令については、表 4-14 のリストを参照のこと。) IA-32 のシリアル化命令、またはキャッシングされないメモリ・タイプへのアクセスでは、それ自体が可視になる前に、書き込みコーレシングされたデータが強制的に可視にされる。

次の文に差し替える。

IA-64 の release (解放) 操作または IA-64 の fence (フェンス) タイプ命令はいずれも、命令そのものが参照できるようになる前に、書き込みコーレシングされたデータを強制的にフラッシュして参照できるようにしている。この解放操作は、コーレシング・メモリ属性を持つページを参照するかしないかには関係ない (release および fence の命令の一覧については、表 4-14 を参照のこと)。IA-32 のシリアル化命令、またはキャッシングされないメモリ・タイプへのアクセスはすべて、それ自体が参照できるようになる前に、書き込みコーレシングされたデータを強制的にフラッシュして参照できるようにしている。

8-36 ページ、8.3 節の「パラメータ」の次の部分を差し替える。

IIP、IPSR、IIPA、IFS が定義される。詳細は、8-1 ページを参照のこと。

差し替え後：

IIP、IPSR、IIPA、IFS が定義される。詳細は、8-1 ページを参照のこと。

**注意：** 非実装命令アドレス・トラップの IIP 値の詳細は、3-18 ページの 3.3.5.3 項を参照のこと。

## 6. 第 1 巻：ALAT にヒットする Id.c 命令のフォルトに関する説明を明確化

4-19 ページ、4.4.5.3.1 項の「2.」の箇所。

1. ターゲット・レジスタを元の状態のままにしておき、かつチェック・ロードのデータ・アクセスまたは変換に関連する例外条件が 1 つ以上発生した場合、プロセッサは、最優先順位を持つこれらのフォルトを発生させるか、あるいは、それらすべてを無視して実行を続けるかのいずれかを選ぶことができる。無視できるフォルトは、データ・アクセスと変換に関連するものに限られる。次のようなフォルトがある。Data Nested TLB フォルト、Alternate Data TLB フォルト、VHPT Data フォルト、Data TLB フォルト、Data Page Not Present フォルト、Data NaT Page Consumption フォルト、Data Key Miss フォルト、Data Key Permission フォルト、Data Access Rights フォルト、Data Dirty bit フォルト、Data Access Bit フォルト、Data Debug フォルト、Unaligned Data Reference フォルト、Unsupported Data Reference フォルト。

## 7. 第2巻：IA-32 の IBR/DBR の一致に関する説明の明確化

7-3 ページ、表 7-1 の addr フィールドの説明を次のように変更する。

変更前：

IA-32 命令セットの参照の場合は、IBR{31:0} が照合に使用される。IA-32 のメモリ参照の場合、一致するには addr{63:32} がゼロでなければならない。

変更後：

IA-32 命令で参照する場合、照合には IBR.addr{31:0} が使われ、一致するには IBR.addr{63:32} がゼロでなければならない。

7-3 ページ、表 7-1 の mask フィールドの説明を次のように変更する。

変更前：

IA-32 のメモリ参照では、mask{63:32} は無視される。

変更後：

IA-32 命令で参照する場合は、IBR.mask{55:32} は無視される。

7-5 ページの最後の箇条書き項目から次の文を削除する。

IA-32 のデータ・メモリ参照を検出するには、DBR の addr フィールドの上位 32 ビットが 0 でなければならない。

## 8. 第3巻：IA-32 CPUID に関する説明を明確化

5-71 ページを次のように変更する。

変更前：

```
BREAK;  
ESAC;
```

変更後：

```
BREAK;  
DEFAULT: (* EAX > highest value recognized by CPUID *)  
    EAX <- Reserved, undefined;  
    EBX <- Reserved, undefined;  
    ECX <- Reserved, undefined;  
    EDX <- Reserved, undefined;  
BREAK;  
ESAC;
```

## 9. 第2巻：表 A-5、pr-writers-int クラス

A-22 ページ、表 A-5 の pr-writers-int クラスの「イベント/命令」欄に pr-and-writers と pr-or-writers を追加する。

10. **第2巻：4.4.6.1 項の PAL\_MC\_DRAIN プロシーダは、単に キャッシュ・ライン・ライトバック・トランザクションがバスに対して強制的に実行されるようにするためのものでしかなく、そのトランザクションがメイン・メモリにまで到達したかどうかを保証するものではない。**

4-32 ページ、4.4.6.1 項の最終段落を次のように変更する。

ページ [X] に属するアドレスが格納されているすべてのキャッシュ・ラインがコヒーレンス・ドメイン内のすべてのキャッシュから追い出されてバスに強制移動させられることをもっと確実にするためには、上記のシーケンスを実行したあと、IPI のしくみを利用して、ソフトウェアを使ってコヒーレンス・ドメイン内のすべてのプロセッサ上で PAL\_MC\_DRAIN 操作を実行しなければならない。この操作はキャッシュ・ラインがメモリにライトバックされたことを保証するものではないので注意すること。

(上記のパラグラフでは「メモリにライトバックされた」が「強制移動させられる」に変更されると共に新しい文章が後に追加されている。

# 文書の修正

## 1. 第4巻：IBR および DBR のアドレス指定の誤植

6-19 ページ、6.2.4 項のページ中央の等式。IBR<sub>i</sub>.addr をすべて IBR<sub>[2\*i]</sub>.addr に置き換える。

変更前：

$$\text{IBRmatch}_i = \text{match}(\text{IP}, \text{IBR}_i.\text{addr}, \text{IBR}_{[2*i]+1}.\text{mask}, \text{IBR}_{[2*i]+1}.\text{plm})$$

変更後：

$$\text{IBRmatch}_i = \text{match}(\text{IP}, \text{IBR}_{[2*i]}.addr, \text{IBR}_{[2*i]+1}.\text{mask}, \text{IBR}_{[2*i]+1}.\text{plm})$$

\* 注 - ほかの等式と整合がとれるように、( ) を [ ] に変更する。

変更前：

$$\text{IBRmatch}_i = (\text{IBR}_{[2*i]+1}.\text{plm}[\text{PSR.cpl}])$$

$$\text{and}(\text{AND}_{b=50..0}((\text{IBR}_i.\text{addr}\{b\} \text{ and } \text{IBR}_{[2*i]+1}.\text{mask}\{b\}) = (\text{IP}\{b\} \text{ and } \text{IBR}_{[2*i]+1}.\text{mask}\{b\})))$$

$$\text{and}(\text{AND}_{b=55..51}((\text{IBR}_i.\text{addr}\{b\} \text{ and } \text{IBR}_{[2*i]+1}.\text{mask}\{b\}) = (\text{IP}\{50\} \text{ and } \text{IBR}_{[2*i]+1}.\text{mask}\{b\})))$$

$$\text{and}(\text{AND}_{b=60..56}(\text{IBR}_i.\text{addr}\{b\} = \text{IP}\{50\}))$$

$$\text{and}(\text{AND}_{b=63..61}(\text{IBR}_i.\text{addr}\{b\} = \text{IP}\{b\}))$$

変更後：

$$\text{IBRmatch}_i = (\text{IBR}_{[2*i]}.plm[\text{PSR.cpl}])$$

$$\text{and}(\text{AND}_{b=50..0}((\text{IBR}_{[2*i]}.addr\{b\} \text{ and } \text{IBR}_{[2*i]+1}.\text{mask}\{b\}) = (\text{IP}\{b\} \text{ and } \text{IBR}_{[2*i]+1}.\text{mask}\{b\})))$$

$$\text{and}(\text{AND}_{b=55..51}((\text{IBR}_{[2*i]}.addr\{b\} \text{ and } \text{IBR}_{[2*i]+1}.\text{mask}\{b\}) = (\text{IP}\{50\} \text{ and } \text{IBR}_{[2*i]+1}.\text{mask}\{b\})))$$

$$\text{and}(\text{AND}_{b=60..56}(\text{IBR}_{[2*i]}.addr\{b\} = \text{IP}\{50\}))$$

$$\text{and}(\text{AND}_{b=63..61}(\text{IBR}_{[2*i]}.addr\{b\} = \text{IP}\{b\}))$$

6-21 ページ、6.2.6 項のページ末尾の等式。DBR<sub>i</sub>.addr をすべて DBR<sub>[2\*i]</sub>.addr に置き換える。

変更前：

$$\text{DBRRangeMatch}_i =$$

$(AND_{b=50..0}((DBR_i.addr\{b\} \text{ and } DBR_{[2*i]+1}.mask\{b\}) = (addr\{b\} \text{ and } DBR_{[2*i]+1}.mask\{b\})))$

$\text{and } (AND_{b=55..51}((DBR_i.addr\{b\} \text{ and } DBR_{[2*i]+1}.mask\{b\}) = (addr\{50\} \text{ and } DBR_{[2*i]+1}.mask\{b\})))$

$\text{and } (AND_{b=60..56}(DBR_i.addr\{b\} = addr\{50\}))$

$\text{and } (AND_{b=63..61}(DBR_i.addr\{b\} = addr\{b\}))$

変更後：

$DBRRangeMatch_i =$

$(AND_{b=50..0}((DBR_{[2*i]}.addr\{b\} \text{ and } DBR_{[2*i]+1}.mask\{b\}) = (addr\{b\} \text{ and } DBR_{[2*i]+1}.mask\{b\})))$

$\text{and } (AND_{b=55..51}((DBR_{[2*i]}.addr\{b\} \text{ and } DBR_{[2*i]+1}.mask\{b\}) = (addr\{50\} \text{ and } DBR_{[2*i]+1}.mask\{b\})))$

$\text{and } (AND_{b=60..56}(DBR_{[2*i]}.addr\{b\} = addr\{50\}))$

$\text{and } (AND_{b=63..61}(DBR_{[2*i]}.addr\{b\} = addr\{b\}))$

## 2. 第3巻：extract、deposit、alloc の各命令に関する図を変更

2-40 ページの extract 命令において、図 2-6 のすぐ上の文を変更する。

変更前：

$extr\ t = r, 7, 50$  の操作の図解を図 2-6 に示す。

変更後：

$extr\ r_1 = r_3, 7, 50$  の操作の図解を図 2-6 に示す。

2-40 ページの図 2-6 のレジスタ値のラベルを変更する。

変更前：

GR r を GR r<sub>3</sub> に変更

GR t を GR r<sub>1</sub> に変更

2-37 ページの deposit 命令において、図 2-5 のすぐ上の文を変更する。

変更前：

$dep\ t = s, r, 36, 16$  の操作の図解を図 2-5 に示す。

変更後：

2-37 ページ、図 2-5 のレジスタ値のラベルを変更する。

変更前：

GR r を GR r<sub>3</sub> に変更

GR s を GR r<sub>2</sub> に変更

GR t を GR r<sub>1</sub> に変更

2-37 ページ、図 2-5 の下に次の段落を新たに追加する。

dep.z r<sub>1</sub> = r<sub>2</sub>, 36, 16 の操作の図解を図 XXX に示す。

図 2-5 と、同図を含む ParaAnchor 段落をコピーする。

コピーしたものを、新しい段落のすぐ下にペーストする。

ペーストした図の中で、GR r<sub>3</sub> のラベルとレジスタを削除し、さらに同レジスタから実行後のレジスタに向かっている 2 本の線も削除する。

GR r<sub>1</sub> レジスタの左右に 1 個ずつ 0 を追加して、この両ビットが 0 になることを示す。ターゲット・レジスタにおける 0 を図で表現した例については、2-4 ページの addp 命令を参照のこと。

2-5 ページの alloc 命令において、図 2-2 の sol 線の下に双方向矢印を 1 本追加する。

新しく追加するこの線の左端の位置は、sol 線の左端に合わせる。

新しく追加するこの線の長さは sol 線のおよそ半分とする。

新しく追加するこの線の中央下に sor と書き込む。

### 3. 第 2 巻 : RSE および PMC の誤植

6-6 ページ、6.4 節の表 6-2 において、CFM 行、br.call 列の内容を次のように変更する。

変更前 :

```
CFM.sof = CFM.sol
```

変更後 :

```
CFM.sof -= CFM.sol
```

7-7 および 7-8 ページ、7.2.1 項の以下の 3 個所の「PMD」を「PMC」に変更する。

図 7-5 のタイトル

変更前 :

```
...(PMC[4]..PMD[p])
```

変更後 :

```
...(PMC[4]..PMC[p])
```

図 7-5、図の左側。

```
...(PMC[4]..PMD[p])”
```

変更後 :

...(PMC[4]..PMC[p])

表 7-4 のタイトルを変更する。

変更前 :

...(PMC[4]..PMD[p])

変更後 :

...(PMC[4]..PMC[p])

#### 4. 第 2 巻 : DV 表の誤植

A-22 ページ、付録 A、A.4 節の表 A-5 (「命令のクラス」) において、fp-writers クラスの「イベント / 命令」欄に setf 命令を追加する。

#### 5. 第 2 巻 : ISR の図と言い回しを変更する。

第 2 巻、8-5 ページ、表 8-2、「非サポート・データ参照フォルト」行の r 列の値を r に変更し、w 列の値を w に変更する。

第 2 巻、8-26 ページ、General Exception (一般例外) ベクタについて説明している箇所で、ISR に関する 2 つ目の図の sp がつねに 0 となるように変更する。

第 2 巻、8-33 ページ、Unsupported Data Reference (非サポート・データ参照) ベクタについて説明している箇所で、ISR のビット 34 の値を r に変更し、ビット 33 の値を w に変更する。

第 2 巻、8-33 ページ、Unsupported Data Reference (非サポート・データ参照) ベクタについて説明している箇所で、ISR の図の下に次の文を追加する。

ldfe 命令および stfe 命令を実行すると、任意でプロセッサに ISR.r と ISR.w の両方を 1 にセットさせることができるが、これは推奨できない。

#### 6. 第 2 巻 : Lower-privilege Transfer Trap (下位特権遷移トラップ) ベクタを説明している箇所の ISR の図を変更する (第 2 巻の新規 8-36 ページ)。

**名称** Lower-privilege Transfer Trap (下位特権遷移トラップ) ベクタ (0x5e00)

**原因** 次の 2 つのトラップ条件により、このベクタに制御が移動するため。

- 実装されていないアドレスで命令を実行しようとする  
と、非実装命令アドレス・トラップが発生する。4-24

ページ「実装されていないアドレス・ビット」を参照のこと。

- PSR.lp ビットが 1 であり、分岐によって特権レベルが下がる。

IA-32 命令でこのトラップが発生することはない。

このベクタでの割り込み：

非実装命令アドレス・トラップ  
下位特権遷移トラップ

パラメータ IIP、IPSR、IIPA、IFS が定義される。詳細は、8-1 ページを参照のこと。

ISR - ISR.ei ビットは、どの命令が原因で例外が発生したかを示すようにセットされる。ISR.code は、直前に実行された命令で発生したすべてのトラップに対して 1 ビットのベクタ (8-5 ページ、表 8-3) を保持する。定義済み ISR ビットは、次のように指定される。

当該トラップの発生原因が非実装命令アドレス・トラップである場合：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0								
0										0										0		fp		トラップ・コード		0		0		1		ss		tb		lp		fp	
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32								
0																		0		ei		0		ni		ir		0		0		0		0		0		0	

当該トラップの発生原因が下位特権遷移トラップである場合：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0								
0										0										0		0		0		0		ss		tb		1		0					
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32								
0																		0		ei		0		ni		ir		0		0		0		0		0		0	

注意 インライン命令のフェッチ、分岐の成立または不成立、rfi 命令のいずれかが実行されると、非実装命令アドレス・トラップが発生することがある。下位特権遷移トラップは、降格分岐が起こるときにのみ発生し、rfi 命令のリターンでは発生しない。

## 7. 第 4 巻：図 6-17、図 6-19、ビットの誤記

6-22 ページの図 6-17「命令イベント・アドレス設定レジスタ (PMC[10])」および 6-25 ページの図 6-19「データ・イベント・アドレス設定レジスタ (PMC[11])」の右から 2 番目の「無視」フィールド (ビット 8 ~ 15) のビット数が 7 となっているが、これを 8 にする。

## 8. 第4巻：第6章、ストールとフラッシュの8つの原因

6-19 ページ、6.1.2 項「プロファイリング」の上の部分を実次の文章に差し替える。

Itanium プロセッサのサイクル・アカウンティング・モニタを利用すると、シングルサイクル、マルチサイクルのいずれについても、ストールとフラッシュを引き起こした条件の主要なものをすべて知ることができる。ストール条件とフラッシュ条件が同時に生じた場合は、パイプラインの並び順とは逆の順序で優先順位が付けられる。すなわち、パイプラインの後段側に近いほうのものがその原因として報告される。ストールとフラッシュの発生原因は8つあるが、その優先順位は次のようになっている。

1. バックエンド・フラッシュ・サイクル：分岐予測ミス、ALAT のフラッシュ、シリアル化フラッシュ、コントロール・スペキュレーション・フラッシュの失敗、MMU-IEU のバイパス、その他の例外により失われるサイクル。
2. データ・アクセス・サイクル：メモリ・サブシステムからソース・オペランドが取得されるのを命令が待機しているとき、およびメモリ・フラッシュが実行されているときに失われるサイクル (L1D ウェイの予測ミス、DTC フラッシュ)。
3. スコアボード依存性サイクル：ロード命令以外の命令からソース・オペランドが取得されるのを命令が待機しているときに失われるサイクル。FP 関連のフラッシュもこれに含まれる。
4. RSE 実行サイクル：メモリ内のバッキング・ストアを相手に実行されるレジスタ・スタック・スピルとレジスタ・スタック・フィルを原因とするストール。
5. 発行リミット・サイクル：ストップ、ポート・オーバーサブスクリプション、非対称性に起因するディスパーサル・ブレイク。
6. 命令アクセス・サイクル：L1I ミスか ITLB ミスを原因とする命令フェッチ・ストール。
7. 分岐成立サイクル：分岐成立予測的中により発生するバブル。
8. ストールしないパイプライン・サイクル：プログラムを実行すると必ず発生するサイクル。

8つのカテゴリのうち4つ(1、2、3、6)は、Itanium プロセッサのイベントとして直接計測することができる。残り4つのカテゴリ(4、5、7、8)は直接には計測されない。その代わりに Itanium プロセッサのイベントとして、パイプライン・フラッシュ・サイクル(1+7)、メモリ・サイクル(2+4)、依存性サイクル(3+5)、ストールしないバックエンド・サイクル(6+8)という4種類の複合カテゴリが利用できる。詳細については、7.4 節「サイクル・アカウンティング・イベント」を参照のこと。



# パフォーマンス監視イベント 7

---

本章では、Itanium プロセッサのアーキテクチャ・イベントおよびマイクロアーキテクチャ・イベントについて説明する。各種イベントの発生回数は、第 6 章の前半に説明しているパフォーマンス監視機構によってカウントできる。本章の前半では、論理的に関係しているイベント同士をいくつかのグループに分け、それぞれの概要を述べる。また、よく使われるパフォーマンス評価規準の計算方法についても解説する。方法としては、ハードウェアのカウンタを用いて直接計測する方法と、いわゆる「派生型」イベントから間接的に取得する方法がある。直接計測できるイベントについては、プロセッサ・イベントをすべてアルファベット順に並べた 7.8 節「パフォーマンス監視イベントのリスト」でさらに詳しく解説する。

## 7.1 イベントのカテゴリ化

パフォーマンス関連のイベントは、以下のカテゴリに分けられる。

- 基本イベント：クロック・サイクル、リタイアした命令 (7.2 節)
- 命令の実行：命令のデコード、発行、実行、データ・スペキュレーションとコントロール・スペキュレーション、メモリ操作 (7.3 節)
- サイクル・アカウンティング・イベント：ストールと実行のサイクルに関する分析 (7.4 節)
- 分岐イベント：分岐予測 (7.5 節)
- メモリ階層：命令キャッシュとデータ・キャッシュ (7.6 節)
- システム・イベント：オペレーティング・システム・モニタ、命令 TLB とデータ TLB (7.7 節)

上記の各節では、直接計測型と派生型の両方について、それぞれ該当する種類のイベントをすべて列挙した表を示してある。直接計測できるイベントは、PMC.umask フィールド (第 6 章、表 6-7 を参照) を使用して、当該イベントの形を変えたもの (バリエーション) として計測することが多い。こうしたイベントのシンボリック・イベント名 (ALAT\_REPLACEMENT.ALL など) には、umask の使用を示すためにピリオドが 1 個含まれている。イベント・リストには、この umask は 4 ビットで指定されているが、その中の x は任意値であることを示す記号である。派生型イベントは、直接計測されたイベントから計算することができる。派生型イベントのイベント・シンボリック名には、.d というサフィックスが 1 個含まれている。関連する各種派生型イベントの計算式についても、それぞれの節に収録してある。ただし、7.8 節のイベント・リストには、派生型イベントについては記載されていない。

以降の各節に収録した各表には、イベントごとにシンボリック・イベント名、イベントの簡易名、詳細説明ページ番号を示してある。派生型イベントについてはイベント・リストには収録していないため、参照先は記していない。

## 7.2 基本イベント

表 7-1 は、4 つの基本実行モニタについてまとめたものである。CPU\_CYCLES イベントを使うと、現在実行中の命令セットに基づいて PMC/PMD の動作を制限することにより、IA-64 サイクル数または IA-32 サイクル数を別々に、あるいは両方を合計してカウントすることができる。リタイアした IA-64 命令数 (IA64\_INST\_RETIRED) には、プレディケート付きの真 / 偽命令と nop が含まれるが、RSE の実行回数は含まれない。

表 7-1. IA-64/IA-32 命令セットの実行 / リタイアメント・モニタ

実行モニタ	名称	ページ
CPU_CYCLES	CPU サイクル	7-48
IA64_INST_RETIRED	リタイアした IA-64 命令	7-52
IA32_INST_RETIRED	リタイアした IA-32 命令	7-52
ISA_TRANSITIONS	IA-64 から IA-32 への ISA の遷移	7-55

表 7-2 に、IPC と、ISA の遷移 1 回あたりの平均 IPC に関する評価規準を示す。

表 7-2. IA-64/IA-32 命令セットの実行 / リタイアメントに関するパフォーマンス評価規準

パフォーマンス評価規準	パフォーマンス・モニタの計算式
1 サイクルあたりの IA-64 命令数	IA64_INST_RETIRED / CPU_CYCLES[IA-64 のみ]
1 サイクルあたりの IA-32 命令数	IA32_INST_RETIRED / CPU_CYCLES[IA-32 のみ]
IA-64 の平均命令数 / 遷移	IA64_INST_RETIRED / (ISA_TRANSITIONS*2)
IA-32 の平均命令数 / 遷移	IA32_INST_RETIRED / (ISA_TRANSITIONS*2)
IA-64 の平均サイクル数 / 遷移	CPU_CYCLES[IA64] / (ISA_TRANSITIONS*2)
IA-32 の平均サイクル数 / 遷移	CPU_CYCLES[IA32] / (ISA_TRANSITIONS*2)

## 7.3 命令の実行

この節では、命令の発行とリタイアメント (表 7-3、表 7-4)、マルチメディアと FP (表 7-5)、データ・スペキュレーションとコントロール・スペキュレーション (表 7-7)、メモリ・モニタ (表 7-9) のそれぞれに関連する各種イベントについて説明する。

**表 7-3. 命令の発行とリタイアメントに関するイベント**

デコード、発行、リタイアメントの各種モニタ	説明	ページ
INST_DISPERSED	配布された命令	7-54
EXPL_STOPS_DISPERSED	配布された明示的ストップ	7-51
ALL_STOPS_DISPERSED	配布された暗黙的ストップと明示的ストップ	7-25
IA64_TAGGED_INST_RETIRED	リタイアしたタグ付き IA-64 命令	7-53
NOPS_RETIRED	リタイアした nop 命令	7-66
PREDICATE_SQUASHED_RETIRED	プレディケート・オフが原因で実行されなかった命令	7-68
RSE_REFERENCES_RETIRED	RSE アクセス	7-68
RSE_LOADS_RETIRED	RSE ロード・アクセス	7-68

**表 7-4. 命令の発行とリタイアメントに関するイベント (派生型)**

デコード、発行、リタイアメントの各種モニタ	説明	パフォーマンス・モニタの計算式
RSE_STORES_RETIRED.d	RSE ストア・アクセス	RSE_REFERENCES_RETIRED - RSE_LOADS_RETIRED

命令キャッシュ・ラインは実行コアに送られ、Itanium プロセッサの機能ユニットに配布される。各サイクルで発行される命令の個数 (INST\_DISPERSED) は、命令ストリーム内の明示的ストップの回数 (EXPL\_STOPS\_DISPERSED) と機能ユニットの可用性によって決まる。リソース制限および分岐バンドルがあると、予測とは無関係に、命令の配布が強制的に中断される。したがって、これらは「暗黙的ストップ」と呼ばれ、ALL\_STOPS\_DISPERSED - EXPL\_STOPS\_DISPERSED という式で計算できる。

リタイアした命令数 (IA64\_TAGGED\_INST\_RETIRED、NOPS\_RETIRED) は、アドレス範囲チェック機能とオペコード・マッチ機能とによって規定されるタグ情報が基となる。リタイアしたタグ付き命令の個数にはプレディケート・オフ命令が含まれる。プレディケート・オフ命令をカウントするときは、別のイベント (PREDICATE\_SQUASHED\_RETIRED) を使う。RSE\_REFERENCES\_RETIRED は、リタイアした RSE の実行回数をカウントする。

リタイアした IA-64 命令の総数をカウントする方法は 2 つある。1 つはタグなしの IA64\_INST\_RETIRED イベントを使用する方法であり、もう 1 つは IA64\_TAGGED\_INST\_RETIRED イベントを使用する方法である。後者のイベントを使用する場合は、PMC[8] オペコード・マッチ・レジスタをその任意値設定に合わせてセットアップする必要がある。

表 7-5 に示す FP モニタ (FP\_SIR\_FLUSH、FP\_FLUSH\_TO\_ZERO) は、浮動小数点演算に起因するパイプライン・フラッシュとゼロ・フラッシュの

発生に関する動的情報を収集する。FP\_OPS\_RETIRED.d は、リタイアした FP 演算の回数をカウントする派生型イベントの 1 つである。

**表 7-5. 浮動小数点実行モニタ**

浮動小数点モニタ	説明	ページ
FP_FLUSH_TO_ZERO	ゼロ・フラッシュされた FP の結果	7-52
FP_SIR_FLUSH	FP SIR フラッシュ	7-52

**表 7-6. 浮動小数点実行モニタ (派生型)**

浮動小数点モニタ	説明	パフォーマンス・モニタの計算式
FP_OPS_RETIRED.d	リタイアした FP 演算	$(4 * \text{FP\_OPS\_RETIRED\_HI}) + \text{FP\_OPS\_RETIRED\_LO}$

表 7-7 に示すように、コントロール・スペキュレーションおよびデータ・スペキュレーション用のモニタは、動的なランタイム情報を取得する。具体的には、chk.s 命令の失敗回数 (INST\_FAILED\_CHKS\_RETIRED.ALL)、ALAT によって検出されるアドバンスト・ロード・チェックとチェック・ロードの回数 (ALAT\_INST\_CHKA\_LDC.ALL)、アドバンスト・ロード・チェックとチェック・ロードの失敗回数 (ALAT\_INST\_FAILED\_CHKA\_LDC.ALL) に関する情報である。リタイアした chk.s 命令の個数は、適切なオペコード・マスクを持つ IA64\_TAGGED\_INST\_RETIRED イベントによって監視される。Itanium プロセッサの ALAT は、分岐経路の予測が外れたときの操作によって更新されるため、アドバンスト・ロード・チェックとチェック・ロードの回数をカウントするには、明示的なイベント (ALAT\_INST\_CHKA\_LDC.ALL) が必要である。最終的には、ALAT\_REPLACEMENT.ALL イベントを使用して、ALAT のオーバーフローを監視できる。

**表 7-7. コントロール/データ・スペキュレーションに関する各種モニタ**

コントロール/データ・スペキュレーションに関する各種モニタ	説明	ページ
INST_FAILED_CHKS_RETIRED.ALL	失敗したスペキュレーティブ・チェック・ロード	7-54
ALAT_INST_CHKA_LDC.ALL	アドバンスト・ロード・チェックおよびチェック・ロード	7-23
ALAT_INST_FAILED_CHKA_LDC.ALL	失敗したアドバンスト・ロード・チェックおよびチェック・ロード	7-24
ALAT_REPLACEMENT.ALL	命令によって置き換えられた ALAT エントリ	7-23

命令タイプのユニット・マスクを用いると、この 4 つのコントロール/データ・スペキュレーション・イベントの監視対象を限定することができる。例えば、整数命令だけ、FP 命令だけ、あるいはすべてのスペキュレーティブ命令だけを対象にすることができる。Itanium プロセッサの各

種スペキュレーション・モニタを使うと、表 7-8 に示すパフォーマンス評価規準の計算ができる。

**表 7-8. コントロール/データ・スペキュレーションに関するパフォーマンス評価規準**

パフォーマンス評価規準	パフォーマンス・モニタの計算式
コントロール・スペキュレーション・ミス率	INST_FAILED_CHKS_RETIRED.ALL / (IA64_TAGGED_INST_RETIRED[chk.s]-PREDICATE_SQUASHED_RETIRED[chk.s])
データ・スペキュレーション・ミス率	ALAT_INST_FAILED_CHKA_LDC.ALL / ALAT_INST_CHKA_LDC.ALL
ALAT の容量性ミス率	ALAT_REPLACEMENT.ALL / IA64_TAGGED_INST_RETIRED[lid.a,lid.sa,lid.c.nc, ldf.a, ldf.sa, ldf.c.nc]-PREDICATE_SQUASHED_RETIRED[lid.a, lid.sa, lid.c.nc, ldf.a, ldf.sa, ldf.c.nc])

表 7-8 に示した計算式でコントロール・スペキュレーション・ミス率や ALAT の容量性ミス率を計算するときは、IA64\_TAGGED\_INST\_RETIRED [ ~ ] から PREDICATE\_SQUASHED\_RETIRED[ ~ ] を減算する必要がある。その理由は、IA64\_TAGGED\_INST\_RETIRED がプレディケート・オフ命令もカウントしていることにある。プレディケート・オフ命令では、アーキテクチャに関する状態が更新されない。そのため、どのパフォーマンス評価規準を計算するときにも、プレディケート・オフ命令の個数を引いて考える必要がある。PMC8 のオペコード・マッチャーと PREDICATE\_SQUASHED\_RETIRED ( および IA64\_TAGGED\_INST\_RETIRED ) を一緒に使うと、目的の命令のプレディケート・オフ・インスタンスの回数をカウントすることができる。ALAT の容量性ミス率を計算するときは、何度か実行しないと、計算に必要なすべての項を取得することができないので注意が必要である。これは、オペコード・マッチャーから課される制約によるものである。

表 7-9 に、リタイアしたロードとストアをカウントする 6 種類のメモリ命令リタイアメント・イベントを示す。RSE の実行回数もカウントされる。このロード回数には、チェック・ロード命令の失敗回数が含まれる。

**表 7-9. メモリ・イベント**

メモリ・モニタ	説明	ページ
LOADS_RETIRED	リタイアしたロード	7-65
STORES_RETIRED	リタイアしたストア	7-69
UC_LOADS_RETIRED	リタイアしたキャッシュ不可ロード	7-69
UC_STORES_RETIRED	リタイアしたキャッシュ不可ストア	7-69
MISALIGNED_LOADS_RETIRED	リタイアしたミスアライン・ロード命令	7-66
MISALIGNED_STORES_RETIRED	リタイアしたミスアライン・ストア命令	7-66

## 7.4 サイクル・アカウンティング・イベント

6.1.1.4 項「サイクル・アカウンティング」で説明しているように、Itanium プロセッサは直接計測型のストール・サイクル・モニタを 8 種類搭載している。表 7-10 に、各種サイクル・アカウンティング・イベントを示す。

Itanium プロセッサでは、あらゆるクロック・サイクルが 4 種類のサイクル・カウンタのいずれかに分類される。すなわち、DEPENDENCY\_ALL\_CYCLE、MEMORY\_CYCLE、UNSTALLED\_BACKEND\_CYCLE、PIPELINE\_ALL\_FLUSH\_CYCLE の 4 つである。この 4 つのカウンタの値を合計したものが CPU\_CYCLES である。

非ロード命令と GR や FR との間に依存性があると、命令ディスパースル・ブレーク（明示的および暗黙的の両方のストップを含む）、FP 関連のフラッシュ、スコアボード・ストールといったものが発生するが、これをカウントするのが DEPENDENCY\_ALL\_CYCLE である。つまり、このモニタは、メモリ・サブシステムからソース・オペランドが取得されるのを命令が待機しているときに発生するストールについてはカウントしない。また、このモニタは、ストールもフラッシュも生じずに命令が実行されているときのサイクル数はカウントしないことに注意が必要である。DEPENDENCY\_SCOREBOARD\_CYCLE モニタも似たような働きをするが、ディスパースル・ブレークはカウントしない。

表 7-10. ストール・サイクル・モニタ

ストール・アカウンティング・モニタ	説明	ページ
PIPELINE_BACKEND_FLUSH_CYCLE	分岐予測ミスが例外に起因するパイプライン・フラッシュ・サイクルの合計	7-67
DATA_ACCESS_CYCLE	データ・アクセスによるストール・サイクル	7-49
DEPENDENCY_SCOREBOARD_CYCLE	スコアボード依存性サイクル	7-50
INST_ACCESS_CYCLE	命令アクセス・サイクル	7-53
PIPELINE_ALL_FLUSH_CYCLE	フロントエンドかバックエンドのソースによるパイプライン・フラッシュ・サイクルの合計	7-67
MEMORY_CYCLE	メモリ・ストール・サイクルの合計	7-66
DEPENDENCY_ALL_CYCLE	スコアボード依存性 / ディスパースル・ブレーク・サイクル	7-49
UNSTALLED_BACKEND_CYCLE	ストールしないバックエンド・サイクル	7-69

MEMORY\_CYCLE は、メモリ・サブシステムからソース・オペランドが取得されるのを命令が待機しているときにパイプラインがストールする回数をカウントするほかに、メモリ・アクセス (L1D ウェイの予測ミス、

DTC フラッシュ)に関連したパイプライン・フラッシュが実行されている間にパイプラインがストールする回数についてもカウントする。同様に、レジスタ・スタック・エンジンがメモリを相手にレジスタをスビルまたはフィルしているときにパイプラインがストールする回数についてもカウントする。DATA\_ACCESS\_CYCLE モニタも似たような働きをするが、RSEの実行回数についてはカウントしない。

バックエンドが遅延せずに命令を処理していて、フロントエンドとバックエンドとの間のデカップリング・バッファが空であるサイクル数をカウントするのが UNSTALLED\_BACKEND\_CYCLE である。この状態だと、フロントエンドに及んだ影響は当該パイプラインのバックエンドに現れる。したがって、このモニタでは、LII および ITLB にヒットするかミスするかに関係なく、バックエンドでストールもフラッシュも発生せず、なおかつデカップリング・バッファが空の状態にあるサイクル数がカウントされる。バックエンドでストールもフラッシュも発生せず、かつデカップリング・バッファが空であり、かつ LII か ITLB にミスが発生したことが原因でフロントエンドがストールして待機状態にあるときのサイクル数をカウントするのは、INST\_ACCESS\_CYCLE モニタである。

PIPELINE\_ALL\_FLUSH\_CYCLE は、分岐関連のリステアにより失われたサイクル数をカウントする。リステアは「分岐予測リステア」か「分岐予測ミス・リステア」のどちらかに分類できる。分岐予測リステアは、フロントエンドが分岐成立を正しく予測したときに発生するものであり、分岐予測ミス・リステアは、分岐成立についても分岐不成立についてもフロントエンドで間違っただけをバックエンドが突き止めたときに発生する。フロントエンドが間違っただけで分岐成立は二度カウントされることはないので、注意すること。バックエンドで発生する分岐予測ミス・フラッシュのほうがフロントエンドのバブルよりも優先される。このモニタは、ALAT フラッシュ、シリアル化フラッシュ、MMU-IEU バイパス・フラッシュ、失敗したコントロール・スペキュレーション・フラッシュ、その他例外フラッシュについてもカウントする。

PIPELINE\_BACKEND\_FLUSH\_CYCLE モニタも似たような働きをするが、いわゆる分岐バブルについてはカウントしない。分岐バブルとは、正しく予測された分岐へフロントエンドで向き直すことを言う。

表 7-11 は、派生型ストール・サイクル・アカウンティング・モニタを直接計測型モニタの計算式で表現したものである。

表 7-11. ストール・サイクル・モニタ (派生型)

ストール・サイクル・モニタ (派生型)	説明	パフォーマンス・モニタの計算式
RSE_ACTIVE_CYCLE.d	RSE アクティブ・サイクル	MEMORY_CYCLE - DATA_ACCESS_CYCLE
ISSUE_LIMIT_CYCLE.d	発行リミット・サイクル	DEPENDENCY_ALL_CYCLE - DEPENDENCY_SCOREBOARD_CYCLE
TAKEN_BRANCH_CYCLE.d	分岐成立サイクル	PIPELINE_ALL_FLUSH_CYCLE - PIPELINE_BACKEND_FLUSH_CYCLE
UNSTALLED_PIPELINE_CYCLE.d	ストールしないパイプライン・サイクル	UNSTALLED_BACKEND_CYCLE - INST_ACCESS_CYCLE

## 7.5 分岐イベント

表 7-12 に、Itanium プロセッサの計測型分岐イベントを 5 つ示す。各イベントは、イベント説明ページで説明しているユニット・マスクを使うことにより、50 種類を超える計測可能な分岐指標に分かれる。BRANCH\_PATH を使って、プレディケート成立 / 不成立予測の精度を検査することができる。一方、ユニット・マスクにより、予測、結果、プレディクタのタイプに基づいて分類することができる。BRANCH\_PREDICTOR は、分岐予測パイプラインを進んでいくときに分岐がどのように各種プレディクタから予測されるかを分類する。ユニット・マスクを使えば更に詳細に分類でき、「予測が当たった」、「プレディケート予測が外れた」、「ターゲット予測が外れた」の 3 種類に各種イベントを分類できる。BRANCH\_MULTIWAY はマルチウェイ分岐バンドルでの予測に関連した各種イベントのみを収集するものであり、それを基にして、対応するシングルウェイ分岐バンドル関連のイベントについて推論することができる。BRANCH\_TAKEN\_SLOT は、実際に成立した分岐が使用しているバンドル内での位置に関する情報を提供する。BRANCH\_EVENT は、ブランチ・トレース・バッファに格納されているイベントの回数をカウントする。

表 7-13 は、派生型分岐モニタを直接計測型モニタの計算式で表現したものである。

表 7-12. 分岐モニタ

分岐イベント	説明
BRANCH_PATH	プレディケート (成立 / 不成立) 予測の精度
BRANCH_PREDICTOR	パイプラインでの分岐予測方法の分類
BRANCH_MULTIWAY	マルチウェイ分岐バンドル予測に関する詳細 (シングルウェイ分岐バンドル予測に関する詳細は、このイベントから推論できる)

表 7-12. 分岐モニタ ( 続き )

分岐イベント	説明
BRANCH_TAKEN_SLOT	バンドル内に分岐成立がある場合のその位置
BRANCH_EVENT	収集された分岐イベント

表 7-13. 分岐モニタ ( 派生型 )

分岐イベント	説明	パフォーマンス・モニタの計算式
BRANCH_MISPREDICTIONS.d	予測ミスした分岐バンドル	(BRANCH_PREDICTOR.ALL.ALL_PREDICTIONS - BRANCH_PREDICTOR.ALL.CORRECT_PREDICTIONS) または (BRANCH_PREDICTOR.ALL.WRONG_PATH + BRANCH_PREDICTOR.ALL.WRONG_TARGET)
BRANCH_1ST_STAGE_PREDICTIONS.d	第 1 パイプライン・ステージで予測された分岐バンドル ( 予測の正否は問わない )	BRANCH_PREDICTOR.1ST_STAGE.ALL_PREDICTIONS
BRANCH_1ST_STAGE_MISPREDICTIONS.d	第 1 パイプライン・ステージで予測された不的中の中の分岐バンドル	(BRANCH_PREDICTOR.ALL.ALL_PREDICTIONS - BRANCH_PREDICTOR.ALL.CORRECT_PREDICTIONS) または (BRANCH_PREDICTOR.1ST_STAGE.WRONG_PATH + BRANCH_PREDICTOR.1ST_STAGE.WRONG_TARGET)
BRANCH_2ND_STAGE_PREDICTIONS.d	第 2 パイプライン・ステージで予測された分岐バンドル ( 予測の正否は問わない )	BRANCH_PREDICTOR.2ND_STAGE.ALL_PREDICTIONS
BRANCH_2ND_STAGE_MISPREDICTIONS.d	第 2 パイプライン・ステージで予測された不的中の中の分岐バンドル	(BRANCH_PREDICTOR.ALL.ALL_PREDICTIONS - BRANCH_PREDICTOR.ALL.CORRECT_PREDICTIONS) または (BRANCH_PREDICTOR.2ND_STAGE.WRONG_PATH + BRANCH_PREDICTOR.2ND_STAGE.WRONG_TARGET)
BRANCH_3RD_STAGE_PREDICTIONS.d	第 3 パイプライン・ステージで予測された分岐バンドル ( 予測の正否は問わない )	BRANCH_PREDICTOR.3RD_STAGE.ALL_PREDICTIONS
BRANCH_3RD_STAGE_MISPREDICTIONS.d	第 3 パイプライン・ステージで予測された不的中の中の分岐バンドル	(BRANCH_PREDICTOR.ALL.ALL_PREDICTIONS - BRANCH_PREDICTOR.ALL.CORRECT_PREDICTIONS) または (BRANCH_PREDICTOR.3RD_STAGE.WRONG_PATH + BRANCH_PREDICTOR.3RD_STAGE.WRONG_TARGET)
BRANCH_MULTIWAY_COMPONENT.d	すべての予測に関連しているマルチウェイ分岐バンドル予測	BRANCH_MULTIWAY.ALL_PATHS.ALL_PREDICTIONS / BRANCH_PREDICTOR.ALL.ALL_PREDICTIONS

どの分岐イベントについても、命令アドレス範囲とオペコード・マッチ機能によって、その適用範囲を絞ることができる。これについては、[6.1.3 項「監視対象となるイベントの制限」](#)で説明している。命令アドレス範囲チェックはバンドルを最小単位としているため、アドレス範囲を使ってマルチウェイ分岐の適用範囲を限定するのは簡単である。ただし、オペコード・マッチ機能を利用するためには、マルチウェイ分岐 (MBB または BBB のバンドル・テンプレート) は、最初の分岐成立 (その分岐自体も含む) まで、次のように適用範囲が制限される。

```
((address range and opcode match on instruction slot 0)
  and (branch in slot 0 is taken))
or ((address range and opcode match on instruction slot 1)
  and (branch in slot 0 is NOT taken)
  and (branch in slot 1 is taken))
or ((address range and opcode match on instruction slot 0 or 1
or 2)
  and (branch in slot 0 is NOT taken)
  and (branch in slot 1 is NOT taken))
```

## 7.6 メモリ階層

この節では、Itanium プロセッサのメモリ階層に関連する各種イベントについて概要を述べる。メモリ階層イベントは次のように分類される。

- L1 命令キャッシュおよび命令プリフェッチ ([7.6.1 節](#))
- L1 データ・キャッシュ ([7.6.2 節](#))
- L2 ユニファイド・キャッシュ ([7.6.3 節](#))
- L3 ユニファイド・キャッシュ ([7.6.4 節](#))

Itanium プロセッサのメモリ階層 (3 段構成) とその各種イベント・モニタの概要を [図 7-1](#) に示してある。命令ストリームとデータ・ストリームは、それぞれ別々の L1 キャッシュを通過する。L1 データ・キャッシュはライトスルー・キャッシュである。ユニファイド型 L2 キャッシュは、L1 命令キャッシュと L1 データ・キャッシュの両方の下位キャッシュとして働く。このユニファイド型 L2 キャッシュの下位には、容量の大きいユニファイド型 L3 キャッシュがある。キャッシュ階層の個々のレベルでの各種イベントについて、以下の 3 つの節で述べる。これらのイベントを使うと、最も一般的なキャッシュ・パフォーマンス比率 ([表 7-15](#) を参照) の計算ができる。

一般的なパフォーマンス評価規準のうちハードウェアで直接計測できないものについては、[表 7-14](#) の数式で求めることができる。

図 7-1. Itanium™ プロセッサのメモリ階層における各種イベント・モニタ

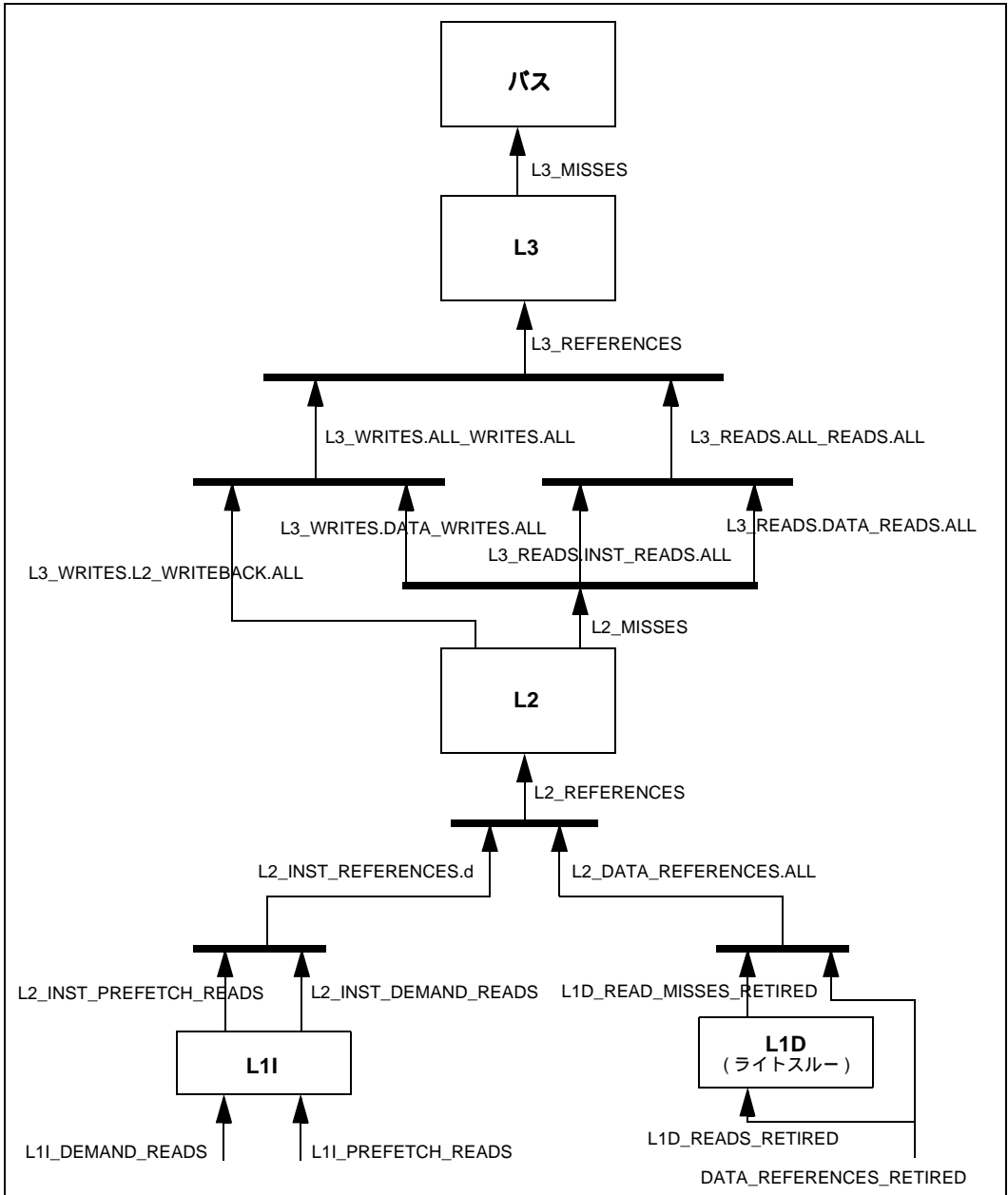


表 7-14. 派生型のメモリ階層モニタ

メモリ階層モニタ (派生型)	説明	パフォーマンス・モニタの計算式
L1I_REFERENCES.d	L1 命令キャッシュ参照	L1I_PREFETCH_READS + L1I_DEMAND_READS
L2_INST_REFERENCES.d	L2 命令参照	L2_INST_DEMAND_READS + L2_INST_PREFETCH_READS
L3_DATA_REFERENCES.d	L3 データ参照	L3_WRITES.DATA_WRITES.ALL + L3_READS.DATA_READS.ALL
L3_CORRECTION_RATIO.d	L3 補正比率	L2_MISSES / (L3_REFERENCES - L3_WRITES.L2_WRITEBACK.ALL)

Itanium のパフォーマンス・モニタのうち L2 キャッシュのモニタには 2 次ミスが含まれ、L3 キャッシュのモニタには含まれないため、L2 と L3 とのパフォーマンス・モニタを直接比較することはできない。次表 7-15 に、各種のパフォーマンス評価規準を示す。場合によっては、L2 ビューか L3 ビューからのイベントだけを使って目的の指標を計算することがある。ただし、この方法では、便利な指標のいくつかは正確には計算できない。この種の指標については、補正比率を使って L3 イベントを推定し、2 次ミスを含んでいる値の概算値を求める。表 7-14 に、この補正比率の定義を列挙する。

表 7-15. キャッシュ・パフォーマンス比率

パフォーマンス評価規準	パフォーマンス・モニタの計算式
L1 命令ミス率	L3_DATA_REFERENCES.d / L1I_REFERENCES.d
L1 命令デマンド・ミス率	L2_INST_DEMAND_READS / L1I_DEMAND_READS
L1 命令プリフェッチ・ミス率	L2_INST_PREFETCH_READS / L1I_PREFETCH_READS
L1 データ読み取りミス率	L1D_READ_MISSES_RETIRED / L1D_READS_RETIRED
L2 ミス率	L2_MISSES / L2_REFERENCES
近似 L2 データ・ミス率	(L3_DATA_REFERENCES.d / L2_DATA_REFERENCES.ALL) * L3_CORRECTION_RATIO.d
近似 L2 命令ミス率 (プリフェッチも含む)	(L3_READS.INST_READS.ALL / L2_INST_REFERENCES.d) * L3_CORRECTION_RATIO.d
近似 L2 データ読み取りミス率	(L3_READS.DATA_READS.ALL / L2_DATA_REFERENCES.READS) * L3_CORRECTION_RATIO.d
近似 L2 データ書き込みミス率	(L3_WRITES.DATA_WRITES.ALL / L2_DATA_REFERENCES.WRITES) * L3_CORRECTION_RATIO.d
L2 命令率	L3_DATA_REFERENCES.d / L2_REFERENCES
L2 データ率	L2_DATA_REFERENCES.ALL / L2_REFERENCES
L3 ミス率	L3_MISSES / (L3_REFERENCES - L3_WRITES.L2_WRITEBACK.ALL)
L3 データ・ミス率	(L3_READS.DATA_READS.MISS + L3_WRITES.DATA_WRITES.MISS) / L3_DATA_REFERENCES.d

表 7-15. キャッシュ・パフォーマンス比率 ( 続き )

パフォーマンス評価規準	パフォーマンス・モニタの計算式
L3 命令ミス率	$L3\_READS.INST\_READS.MISS / L3\_READS.INST\_READS.ALL$
L3 データ読み取り率	$L3\_READS.DATA\_READS.ALL / L3\_DATA\_REFERENCES.d$
L3 データ率	$L3\_DATA\_REFERENCES.d / L3\_REFERENCES$
L3 命令率	$L3\_READS.INST\_READS.ALL / L3\_REFERENCES$

## 7.6.1 L1 命令キャッシュおよび命令プリフェッチ

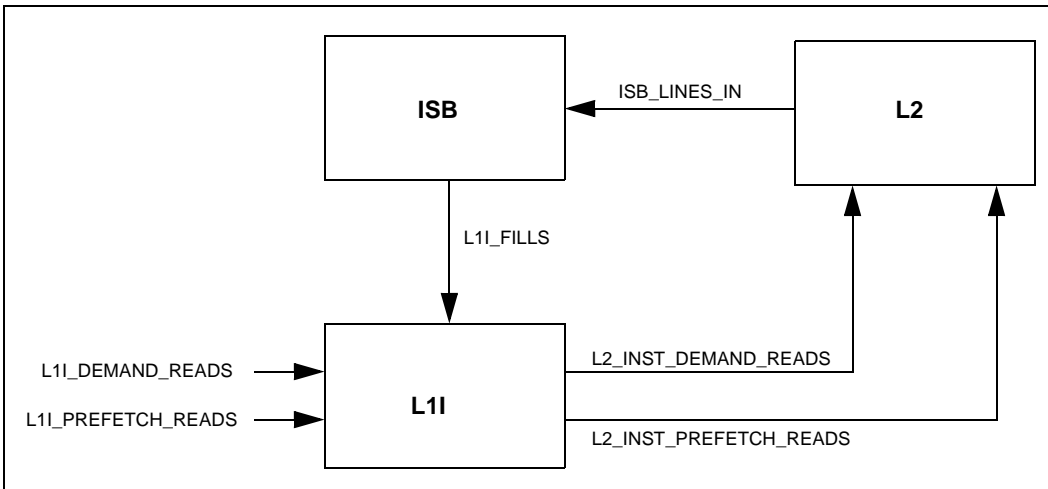
表 7-16 および図 7-2 は、Itanium プロセッサの各種イベントのうち、L1 命令キャッシュにおけるデマンド・フェッチとプリフェッチ動作を監視するイベントについてまとめたものである。表 7-14 には、関連する派生型イベントを示す。この命令フェッチ・モニタはデマンド・フェッチ動作 (L1I\_DEMAND\_READS、L2\_INST\_DEMAND\_READS) とプリフェッチ動作 (L1I\_PREFETCH\_READS、L2\_INST\_PREFETCH\_READS) を別のものとして識別できる。L2 キャッシュから L1 命令キャッシュと命令ストリーミング・バッファに返されるデータ量は、2つのイベント (L1I\_FILLS、ISB\_LINES\_IN) によって監視される。図 7-2 には示していないが、INSTRUCTION\_EAR\_EVENTS モニタは、命令イベント・アドレス・レジスタが命令キャッシュ・ミスまたは ITLB ミスを捕捉した回数をカウントする。

L1 命令キャッシュ/プリフェッチ・イベントについては、命令アドレス範囲チェックを使ってその適用範囲を絞ることができるが、オペコード・マッチャーを使って適用範囲を絞ることはできない。命令キャッシュ/プリフェッチ・イベントは、プロセッサのパイプラインの前半に行われるため、スペキュレーティブ命令、分岐バスの予測が外れた命令、プレディケート・オフ命令によって発生したイベントを含んでいる。アドレス範囲チェックは、実際にリタイアした命令のアドレスではなくスペキュレーティブ命令のアドレスを基準としているため、アドレス範囲チェック機能の機能範囲がプロセッサのパイプライン長よりも狭いアドレス範囲に制限されている場合は、イベント数が不正確になることがある。詳細については、6.2.4 項「IA-64 命令アドレス範囲チェック・レジスタ (PMC[13])」を参照のこと。

表 7-16. L1 命令キャッシュおよび命令プリフェッチ用の各種モニタ

L1 命令/命令プリフェッチ用の各種モニタ	説明	ページ
L1I_DEMAND_READS	L1I および ISB の命令デマンド・ルックアップ	7-57
L1I_FILLS	L1 命令キャッシュ・フィル	7-57
L2_INST_DEMAND_READS	L2 命令デマンド・フェッチ要求	7-59
INSTRUCTION_EAR_EVENTS	命令 EAR イベント	7-55
L1I_PREFETCH_READS	L1I および ISB の命令デマンド・ルックアップ	7-57
L2_INST_PREFETCH_READS	L2 命令プリフェッチ要求	7-60
ISB_LINES_IN	命令ストリーミング・バッファ・ライン入力	7-55

図 7-2. L1 命令キャッシュおよびプリフェッチ用の各種モニタ



## 7.6.2 L1 データ・キャッシュ

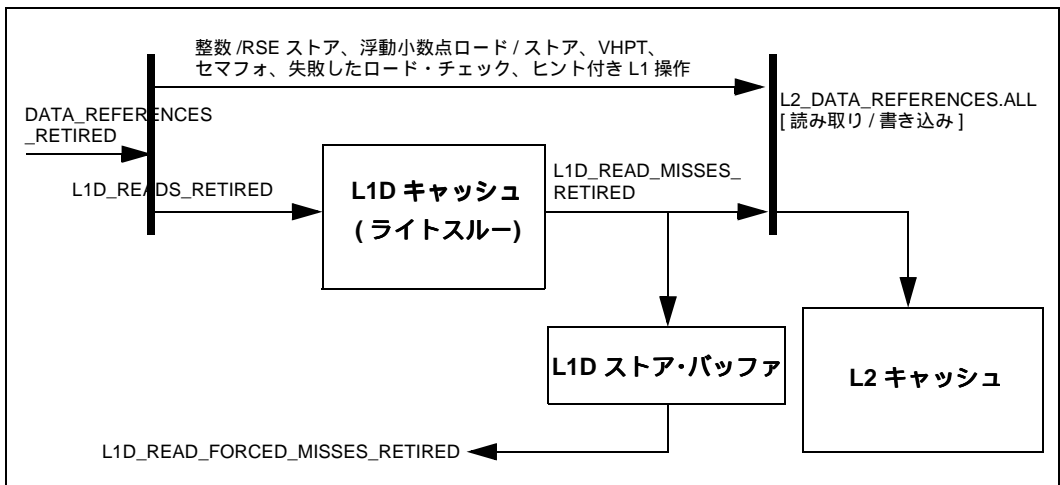
表 7-17 に、Itanium プロセッサの 7 つの L1 データ・キャッシュ・モニタを示す。図 7-3 に示したとおり、ライトスルー型である L1 データ・キャッシュは、キャッシュ可能ロード、整数ストア、RSE ストア、FP メモリ操作、VHPT 参照、セマフォ、チェック・ロード、ヒント付き L2 メモリ参照という各処理に使われる。DATA\_REFERENCES\_RETIRED は、発行されたデータ・メモリ参照の回数である。L1 データ・キャッシュ読み取り (L1D\_READS\_RETIRED) と L1 データ・キャッシュ・ミス (L1D\_READ\_MISSES\_RETIRED) は、L1 データ・キャッシュの読み取りヒット/ミス率を監視する。L2 データ参照の回数

(L2\_DATA\_REFERENCES.ALL) は、キャッシュ・ラインがマージされるよりも前に発行されるデータ要求の回数である。ユニット・マスクを選択すれば、読み取りと書き込みに分けて内訳を調べることができる。図 7-3 には示していないが、DATA\_EAR\_EVENTS モニタは、データ・イベント・アドレス・レジスタがデータ・キャッシュ・ミスまたは DTLB ミスを捕捉した回数をカウントする。RSE の実行回数はどのデータ・キャッシュ・モニタでもカウントされるが、その内訳が明示的に示されることはない。

表 7-17. L1 データ・キャッシュ・モニタ

L1D モニタ	説明	ページ
DATA_REFERENCES_RETIRED	リタイアしたデータ・メモリ参照	7-49
L1D_READS_RETIRED	L1 データ・キャッシュ読み取り	7-57
L1D_READ_MISSES_RETIRED	L1 データ・キャッシュ読み取りミス	7-56
PIPELINE_FLUSH.L1D_WAY_MISPREDICT	パイプライン・フラッシュ	7-67
L1D_READ_FORCED_MISSES_RETIRED	L1 データ・キャッシュの強制ロード・ミス	7-56
L1I_PREFETCH_READS	L2 データの読み取り / 書き込み参照	7-58
DATA_EAR_EVENTS	L1 データ・キャッシュ EAR イベント	7-49

図 7-3. L1 データ・キャッシュ・モニタ



## 7.6.3 L2 ユニファイド・キャッシュ

表 7-18 は、Itanium プロセッサの L2 キャッシュを監視するイベントのうち、直接計測型のイベントについてまとめたものである。表 7-14 には、関連する派生型イベントを示す。L2 キャッシュ・モニタの模式図については図 7-1 を参照のこと。

L2\_REFERENCES、L2\_INST\_DEMAND\_READS、L2\_INST\_PREFETCH\_READS、L2\_DATA\_REFERENCES.ALL、L2\_MISSES はすべて、L2 から見えた要求の回数としてカウントされる。L2\_FLUSHES および L2\_FLUSH\_DETAILS は、アドレス競合、ストア・バッファ競合、バス拒否などが原因で生じた L2 フラッシュの回数をカウントしてその内訳を調べる。L1D\_READ\_FORCED\_MISSES\_RETIRED は、先行するストアからバイパスされてきたロードの回数をカウントする。

表 7-18. L2 キャッシュ・モニタ

L2 モニタ	説明	ページ
L2_REFERENCES	L2 参照	7-60
L2_INST_PREFETCH_READS	L2 命令プリフェッチ要求	7-60
L2_INST_DEMAND_READS	L2 命令デマンド・フェッチ要求	7-59
L2_DATA_REFERENCES.ALL	L2 データの読み取り / 書き込み参照	7-58
L2_DATA_REFERENCES.READS	L2 データ読み取り参照	7-58
L2_DATA_REFERENCES.WRITEs	L2 データ書き込み参照	7-58
L2_MISSES	L2 ミス	7-60
L2_FLUSHES	L2 フラッシュ	7-59
L2_FLUSH_DETAILS	L2 フラッシュの詳細	7-58

## 7.6.4 L3 ユニファイド・キャッシュ

表 7-19 は、直接計測型の L3 キャッシュ・イベントについてまとめたものである。表 7-14 に、関連する派生型イベントを示す。L3 キャッシュ・モニタの模式図については図 7-1 を参照のこと。

表 7-19. L3 キャッシュ・モニタ

L3 モニタ	説明	ページ
L3_REFERENCES	L3 参照 7-63	7-63
L3_MISSES	L3 ミス 7-61	7-61
L3_LINES_REPLACED	置き換えられた L3 キャッシュ・ライン 7-60	7-60
L3_READS.ALL_READS.ALL	命令およびデータの L3 読み取り 7-61	7-61
L3_READS.ALL_READS.HIT	命令およびデータの L3 読み取りヒット 7-61	7-61
L3_READS.ALL_READS.MISS	命令およびデータの L3 読み取りミス	7-61

表 7-19. L3 キャッシュ・モニタ ( 続き )

L3 モニタ	説明	ページ
L3_READS.DATA_READS.ALL	データ L3 読み取り	7-62
L3_READS.DATA_READS.HIT	データ L3 読み取りヒット	7-62
L3_READS.DATA_READS.MISS	データ L3 読み取りミス	7-62
L3_READS.INST_READS.ALL	命令 L3 読み取り	7-62
L3_READS.INST_READS.HIT	命令 L3 読み取りヒット	7-62
L3_READS.INST_READS.MISS	命令 L3 読み取りミス	7-63
L3_WRITES.ALL_WRITES.ALL	L3 書き込み	7-63
L3_WRITES.ALL_WRITES.HIT	L3 書き込みヒット	7-63
L3_WRITES.ALL_WRITES.MISS	L3 書き込みミス	7-64
L3_WRITES.L2_WRITEBACK.ALL	L3 ライトバック	7-64
L3_WRITES.L2_WRITEBACK.HIT	L3 ライトバック・ヒット	7-64
L3_WRITES.L2_WRITEBACK.MISS	L3 ライトバック・ミス	7-64
L3_WRITES.DATA_WRITES.ALL	L3 データ書き込み	7-65
L3_WRITES.DATA_WRITES.HIT	L3 データ書き込みヒット	7-65
L3_WRITES.DATA_WRITES.MISS	L3 データ書き込みミス	7-65

## 7.6.5 フロントサイド・バス

表 7-20 は、フロントサイド・バスまたはシステム・バスの各種トランザクション・モニタを示したものである。

表 7-20. バス・イベント

バス・モニタ	説明	ページ
BUS_ALL	バス・トランザクション	7-40
BUS_PARTIAL	バス・パーシャル・トランザクション	7-44
BUS_BURST	バス・バースト・トランザクション	7-42
BUS_MEMORY	バス・メモリ・トランザクション	7-43
BUS_RD_ALL	バス読み取りトランザクション	7-44
BUS_RD_DATA	バス読み取りデータ・トランザクション	7-44
BUS_RD_PRTL	バス読み取りパーシャル・トランザクション	7-46
BUS_RD_HIT	バス読み取りヒット・クリーン・ノンローカル・キャッシュ・トランザクション	7-45
BUS_RD_HITM	バス読み取りヒット・モディファイド・ノンローカル・キャッシュ・トランザクション	7-45
BUS_RD_INVALID	バス読み取り無効化ライン	7-45
BUS_RD_INVALID_HITM	HITM におけるバス BIL トランザクションの結果	7-46

表 7-20. バス・イベント ( 続き )

バス・モニタ	説明	ページ
BUS_RD_INVALID_BST	バス BRIL パースト・トランザクション	7-45
BUS_RD_INVALID_BST_HITM	HITM におけるバス BRIL パースト・トランザクションの結果	7-46
BUS_HITM	バス・ヒット・モディファイド・ライン・トランザクション	7-42
BUS_WR_WB	バス・ライトバック・トランザクション	7-48
BUS_SNOOPS_HITM	バス・スヌープ・ヒット・モディファイド・キャッシュ・ライン	7-47
BUS_SNOOPS	バス・スヌープ・トータル	7-47
BUS_SNOOP_STALL_CYCLES	バス・スヌープ・ストール・サイクル	7-47
BUS_SNOOPQ_REQ	バス・スヌープ・キュー要求 7-48	7-48
BUS_BRQ_READ_REQ_INSERTED	挿入された BRQ 要求	7-41
BUS_IO	IA-32 互換 I/O バス・トランザクション	7-42
BUS_RD_IO	IA-32 互換 I/O 読み取りトランザクション	7-46
BUS_LOCK	IA-32 互換バス・ロック・トランザクション	7-43
BUS_LOCK_CYCLES	IA-32 互換バス・ロック・サイクル	7-43

表 7-21 に、派生型のフロントサイド・バス・トランザクション・モニタを示す。

表 7-21. フロントサイド・バス・モニタ ( 派生型 )

バス・モニタ ( 派生型 )	説明	パフォーマンス・モニタの計算式
BUS_RD_INSTRUCTIONS.d	バス読み取り命令	BUS_RD_ALL - BUS_RD_DATA
BUS_RD_INVALID_MEMORY.d	メモリから実行されたバス BIL トランザクション	BUS_RD_INVALID - BUS_RD_INVALID_HITM
BUS_RD_INVALID_BST_MEMORY.d	メモリから実行されたバス BRIL パースト・トランザクション	BUS_RD_INVALID_BST - BUS_RD_INVALID_BST_HITM
BUS_ADDR_BPRI.d	I/O エージェントに使われたバス	BUS_MEMORY.IOAGENT
BUS_IOQ_LIVE_REQ.d	インオーダー・バス・キュー要求	$BUS\_IOQ\_LIVE\_REQ\_HI * 4 + BUS\_IOQ\_LIVE\_REQ\_LO$
BUS_BRQ_LIVE_REQ.d	BRQ ライブ要求	$BUS\_BRQ\_LIVE\_REQ\_HI * 4 + BUS\_BRQ\_LIVE\_REQ\_LO$

7.6.5 項で説明するバス・イベントの大部分は、表 7-22 に示す 3 通りのユニット・マスクを使用するバス・トランザクション・イニシエータによって、その適用範囲を限定することができる。

**表 7-22. バス・トランザクション・イベントの適用範囲の限定に使うユニット・マスク (イニシエータ別)**

選択項目	PMC.umask[19:16]	説明
ANY	x001	バス・トランザクションをすべてカウントする (プロセッサ・バス・マスタが開始したものか、プロセッサ以外のバス・マスタが開始したもの)
SELF	x010	ローカル・プロセッサが開始したバス・トランザクションだけをカウントする。
IO	x100	IO エージェント (すなわちプロセッサ以外のバス・マスタ) を起点とするバス・トランザクションをカウントする。

表 7-23 に、7.6.5 項で説明する、Itanium プロセッサのフロントサイド・バス・トランザクション・モニタの説明に使う用語の定義を示す。

**表 7-23. フロントサイド・バス・トランザクションの用語の定義**

名称	説明
BRL	メモリ読み取り (64 バイト・バースト)。WB メモリからのコード・フェッチとデータ・ロードを含む。
BRIL	メモリ読み取り & 無効化 (64 バイト・バースト)。RFO (read for ownership) とも言う。
BIL	メモリ読み取り & 無効化 (0 バイト・サイズのトランザクション)。発生原因は fc (flush cache) 命令のみ。
BWL	メモリ書き込み (64 バイト・バースト)。明示的ライトバック / コーレシングされた書き込み。
BRP	パーシャル・メモリ読み取り (<64 バイト・トランザクション)。通常はキャッシュ不可能読み取り。
BWP	パーシャル・メモリ書き込み (<64 バイト・トランザクション)。通常はキャッシュ不可能書き込み。
IORD	パーシャル IO 読み取り (<64 バイト・トランザクション)。IO ポート空間に対するキャッシュ不可能読み取り
IOWR	パーシャル IO 書き込み (<64 バイト・トランザクション)。IO ポート空間に対するキャッシュ不可能書き込み

表 7-23 に列挙した以外のトランザクションには、Deferred Reply (延期応答)、Special Transactions (特殊トランザクション)、Interrupt (割り込み)、Interrupt Acknowledge (割り込み確認)、Purge TC (ページ TC) がある。7.6.5 項のバス・パフォーマンス・モニタの場合は、どのトランザクションがプライオリティ・エージェントから再試行応答を受信した場合でもカウントするので注意すること。

マルチプロセッサ・システムでのスヌープ・トラフィックの分析を助けるため、Itanium プロセッサにはローカル・プロセッサ・モニタとリモート・レスポンス・モニタが用意されている。ローカル・プロセッサ・スヌープ・イベント (BUS\_SNOOPS\_HITM、BUS\_SNOOPS、BUS\_SNOOPQ\_REQ) は、自分に向かってくるスヌープ・トラフィックを監視する。リモート・レスポンス・イベント (BUS\_RD\_HIT、BUS\_RD\_HITM、BUS\_RD\_INVALID\_HITM、BUS\_RD\_INVALID\_BST\_HITM)

は、監視を行っているプロセッサから生じたバス・トランザクションに対して他のプロセッサがどのようなスヌープ・レスポンスで応じるかを監視する。表 7-24 に、バス・トランザクション別にまとめた各種リモート・スヌープ・イベントを示す。

**表 7-24. バス・イベント (スヌープ・レスポンス別)**

リモート・プロセッサ・レスポンス	BRL	BIL	BRIL
HIT	BUS_RD_HIT	なし	なし
HITM	BUS_RD_HITM	BUS_RD_INVALID_HITM	BUS_RD_INVALID_BST_HITM
ALL	BUS_RD_ALL	BUS_RD_INVALID	BUS_RD_INVALID

Itanium プロセッサの各種フロントサイド・バス・モニタを使うと、表 7-25 のパフォーマンス評価規準の計算ができる。

**表 7-25. バス・パフォーマンス評価規準**

パフォーマンス評価規準	パフォーマンス・モニタの計算式
キャッシュ可能データ・フェッチ・バス・トランザクション率	$BUS\_RD\_DATA / BUS\_ALL$ または $BUS\_RD\_DATA / BUS\_MEMORY$
パーシャル・アクセス率	$BUS\_PARTIAL / BUS\_MEMORY$
読み取りパーシャル・アクセス率	$BUS\_RD\_PRTL / BUS\_MEMORY$
共有ラインに対する読み取りヒット率	$BUS\_RD\_HIT / BUS\_RD\_ALL$ または $BUS\_MEMORY$
モディファイド・ラインに対する読み取りヒット率	$BUS\_RD\_HITM / BUS\_RD\_ALL$ または $BUS\_RD\_HITM / BUS\_MEMORY$
BIL 率	$BUS\_RD\_HIT / BUS\_MEMORY$
モディファイド・ラインに対する BIL ヒット率	$BUS\_RD\_INVALID\_HITM / BUS\_MEMORY$ または $BUS\_RD\_INVALID\_HITM / BUS\_RD\_INVALID$
モディファイド・ラインに対する BRIL ヒット率	$BUS\_RD\_INVALID\_BST\_HITM / BUS\_MEMORY$ または $BUS\_RD\_INVALID\_BST\_HITM / BUS\_RD\_INVALID$
バス・モディファイド・ライン・ヒット率	$BUS\_RD\_HITM / BUS\_MEMORY$ または $BUS\_RD\_HITM / BUS\_BURST$
ライトバック率	$BUS\_WR\_WB / BUS\_MEMORY$ または $BUS\_WR\_WB / BUS\_BURST$
キャッシュ可能読み取り率	$(BUS\_RD\_ALL + BUS\_RD\_INVALID\_BST) / BUS\_MEMORY$
I/O サイクル率	$BUS\_IO / BUS\_ALL$
I/O 読み取り	$BUS\_RD\_IO / BUS\_ALL$

## 7.7 システム・イベント

表 7-26 に、直接計測型のシステム・イベントと TLB イベントを示す。表 7-27 には、関連する派生型イベントを示す。デバッグ・レジスタ・マッチ・イベントとは、命令ブレークポイント・レジスタ (IBR) またはデータ・ブレークポイント・レジスタ (DBR) に格納されているアドレスが、リタイアした現在の命令ポインタに一致した回数 (CODE\_DEBUG\_REGISTER\_MATCHES.d) または現在のデータ・メモリ・アドレスに一致した回数 (DATA\_DEBUG\_REGISTER\_MATCHES.d) をカウントする。PIPELINE\_FLUSH とは、データ変換キャッシュ・ミス、L1 データ・キャッシュ・ウェイの予測ミス、例外フラッシュ、命令シリアル化イベントのいずれかが原因で Itanium プロセッサのパイプラインがフラッシュされた回数をカウントする。CPU\_CPL\_CHANGES とは、割り込み、システム・コール (epc)/ リターン (降格分岐)、rfi 命令が原因で特権レベルが変更された回数をカウントする。CPU\_CYCLES とは、CPU がパワー・ダウンもせずライト HALT 状態にもならないサイクル数をカウントする。

表 7-26. システム・モニタと TLB モニタ

システム・モニタと プロセッサ TLB モニタ	説明	ページ
PIPELINE_FLUSH	パイプライン・フラッシュ	7-67
CPU_CPL_CHANGES	特権レベルの変更	7-48
CPU_CYCLES	CPU サイクル	7-48
ITLB_MISSES_FETCH	ITLB デマンド・ミス	7-56
ITLB_INSERTS_HPW	ハードウェア・ページ・ウォークによる ITLB への挿入	7-56
DTC_MISSES	DTC ミス	7-50
DTLB_MISSES	DTLB ミス	7-51
DTLB_INSERTS_HPW	ハードウェア・ページ・ウォークによる DTLB への挿入	7-50

表 7-27 に、派生型のシステム・イベントと TLB イベントのうち、ハードウェアで直接計測できるイベントから求まるものを示す。

表 7-27. システム・モニタと TLB モニタ (派生型)

派生型のメモリ階層モニタ	説明	パフォーマンス・モニタの計算式
CODE_DEBUG_REGISTER_MATCHES.d	コード・デバッグ・レジスタ・マッチ	IA64_TAGGED_INST_RETIRED
DATA_DEBUG_REGISTER_MATCHES.d	データ・デバッグ・レジスタ・マッチ	LOADS_RETIRED + STORES_RETIRED
ITLB_REFERENCES.d	ITLB 参照	L1I_DEMAND_READS
ITLB_EAR_EVENT.d	ITLB EAR イベント	INSTRUCTION_EAR_EVENTS
DTLB_REFERENCES.d	DTLB 参照	DATA_REFERENCES_RETIRED
DTLB_EAR_EVENT.d	DTLB EAR イベント	DATA_EAR_EVENTS

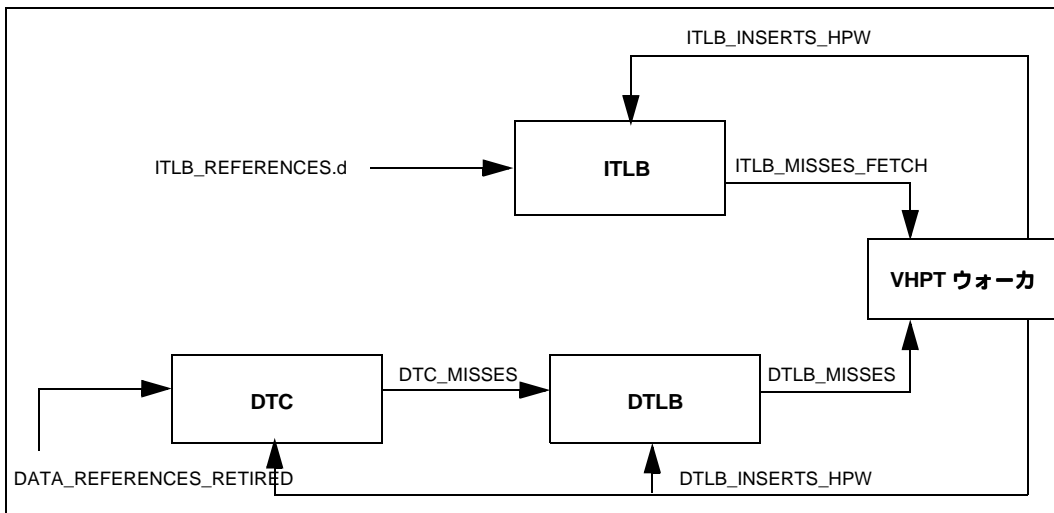
Itanium プロセッサの命令 TLB とデータ用 TLB、および仮想ハッシュ・ページ・テーブル・ウォーカーは、表 7-26、表 7-27 の各種イベントによって監視される。図 7-4 に、その模式図を示す。表 7-28 に、上記イベントから計算できる TLB パフォーマンス評価規準を示す。

ITLB\_REFERENCES.d は命令キャッシュ・アクセス・イベントから、DTLB\_REFERENCES.d はデータ・キャッシュ・アクセス・イベントから求めることができる。ITLB\_REFERENCES.d は L1I キャッシュ (L1I\_PREFETCH\_READS) に対して実行されるプリフェッチ要求を含んでいないので注意すること。その理由は、プリフェッチしようとしても ITLB に目的のものが見つからないとそのプリフェッチはキャンセルされ、VHPT 参照もソフトウェア TLB ミス・ハンドリングも実行されないからである。ITLB\_MISSES\_FETCH および DTLB\_MISSES は、TLB ミスをカウントする。ITLB\_INSERTS\_HPW および DTLB\_INSERTS\_HPW は、仮想ハッシュ・ページ・テーブル・ウォーカーが命令 / データ TLB へ挿入した回数をカウントする。Itanium プロセッサのデータ TLB は 2 段構成であり、その第 1 ステージのデータ TLB ミスの回数をカウントするのが DTC\_MISSES である。

表 7-28. TLB パフォーマンス評価規準

パフォーマンス評価規準	パフォーマンス・モニタの計算式
ITLB ミス率	$ITLB\_MISSES\_FETCH / ITLB\_REFERENCES.d$
DTLB ミス率	$DTLB\_MISSES / DTLB\_REFERENCES.d$
DTC ミス率	$DTC\_MISSES / DTLB\_REFERENCES.d$

図 7-4. 命令 / データ TLB モニタ



## 7.8 パフォーマンス監視イベントのリスト

この節では、Itanium プロセッサの各種パフォーマンス監視イベントを列挙する。

### ALAT\_REPLACEMENT.ALL

- **タイトル** : 命令によって置き換えられた ALAT エントリ、**カテゴリ** : 実行
- **定義** : ALAT\_REPLACEMENT.ALL は、アドバンスト・ロード (ld.a、ld.as、ldfp.a、ldfp.as のいずれか) あるいはクリアなしのチェック・ロード (ld.c.nc および ldf.c.nc の変形) が ALAT 内の有効なエントリを消去した回数をカウントする。
- **イベント・コード** : 0x38、**Umask** : xx11、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 2
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : あり

### ALAT\_REPLACEMENT.FP

- **タイトル** : FP 命令によって置き換えられた ALAT エントリ、**カテゴリ** : 実行
- **定義** : ALAT\_REPLACEMENT.FP は、FP アドバンスト・ロード (ldfp.a または ldfp.as) あるいはクリアなしの FP チェック・ロード (ldf.c.nc の変形) が ALAT 内の有効なエントリを消去した回数をカウントする。
- **イベント・コード** : 0x38、**Umask** : xx10、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 2
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : あり

### ALAT\_REPLACEMENT.INTEGER

- **タイトル** : 整数命令によって置き換えられた ALAT エントリ、**カテゴリ** : 実行
- **定義** : ALAT\_REPLACEMENT.INTEGER は、整数アドバンスト・ロード (ld.a または ld.as) あるいはクリアなしの整数チェック・ロード (ld.c.nc) が ALAT 内の有効なエントリを消去した回数をカウントする。
- **イベント・コード** : 0x38、**Umask** : xx01、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 2
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : あり

### ALAT\_INST\_CHKA\_LDC.ALL

- **タイトル** : アドバンスト・ロード・チェックおよびチェック・ロード、**カテゴリ** : 実行

- **定義** : ALAT\_INST\_CHKA\_LDC.ALL は、ALAT によって検出される、すべてのアドバンスト・ロード・チェック (chk.a)、および、クリアあり / クリアなしの両方の場合におけるチェック・ロード (ld.c.clr または ld.c.nc。FP の変形を含む) の数をカウントする。
- **イベント・コード** : 0x36、**Umask** : xx11、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 2
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : chk.a の場合はなし、ld.c の場合はあり

#### ALAT\_INST\_CHKA\_LDC.FP

- **タイトル** : FP アドバンスト・ロード・チェックおよびチェック・ロード、**カテゴリ** : 実行
- **定義** : ALAT\_INST\_CHKA\_LDC.FP は、ALAT により検出される、すべての FP アドバンスト・ロード・チェック (chk.a)、および、クリアあり / クリアなしの両方の場合におけるすべての FP チェック・ロード (ld.c.clr または ld.c.nc。FP の変形のみ) の数をカウントする。
- **イベント・コード** : 0x36、**Umask** : xx10、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 2
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : chk.a の場合はなし、ld.c の場合はあり

#### ALAT\_INST\_CHKA\_LDC.INTEGER

- **タイトル** : 整数アドバンスト・ロード・チェックおよびチェック・ロード、**カテゴリ** : 実行
- **定義** : ALAT\_INST\_CHKA\_LDC.INTEGER は、ALAT により検出される、すべての整数アドバンスト・ロード・チェック (chk.a)、および、クリアあり / クリアなしの両方の場合におけるすべての整数チェック・ロード (ld.c.clr または ld.c.nc。FP の変形は含まない) の数をカウントする。
- **イベント・コード** : 0x36、**Umask** : xx01、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 2
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : chk.a の場合はなし、ld.c の場合はあり

#### ALAT\_INST\_FAILED\_CHKA\_LDC.ALL

- **タイトル** : 失敗したアドバンスト・ロード・チェックおよびチェック・ロード、**カテゴリ** : 実行
- **定義** : ALAT\_INST\_FAILED\_CHKA\_LDC.ALL は、ALAT により検出される、失敗したアドバンスト・ロード・チェック (chk.a)、および、クリアあり / クリアなしの両方の場合における失敗したチェック・ロード (ld.c.clr または ld.c.nc。FP の変形を含む) をカウントする。
- **イベント・コード** : 0x37、**Umask** : xx11、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 2



- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : `chk.a` の場合はなし、`ld.c` の場合はあり

#### ALAT\_INST\_FAILED\_CHKA\_LDC.FP

- **タイトル** : 失敗した FP アドバンスト・ロード・チェックおよびチェック・ロード、**カテゴリ** : 実行
- **定義** : `ALAT_INST_FAILED_CHKA_LDC.FP` は、ALAT により検出される、失敗した FP アドバンスト・ロード・チェック (`chk.a`)、および、クリアあり / クリアなしの両方の場合における失敗した FP チェック・ロード (`ld.c.clr` または `ld.c.nc`。FP の変形のみ) をカウントする。
- **イベント・コード** : `0x37`、**Umask** : `xx10`、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 2
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : `chk.a` の場合はなし、`ld.c` の場合はあり

#### ALAT\_INST\_FAILED\_CHKA\_LDC.INTEGER

- **タイトル** : 失敗した整数アドバンスト・ロード・チェックおよびチェック・ロード、**カテゴリ** : 実行
- **定義** : `ALAT_INST_FAILED_CHKA_LDC.INTEGER` は、ALAT により検出される、失敗した整数アドバンスト・ロード・チェック (`chk.a`)、および、クリアあり / クリアなしの両方の場合における失敗した整数チェック・ロード (`ld.c.clr` または `ld.c.nc`。FP の変形は含まない) の数をカウントする。
- **イベント・コード** : `0x37`、**Umask** : `xx01`、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 2
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : `chk.a` の場合はなし、`ld.c` の場合はあり

#### ALL\_STOPS\_DISPERSSED

- **タイトル** : 配布された暗黙的ストップと明示的ストップ、**カテゴリ** : 命令の発行
- **定義** : `ALL_STOPS_DISPERSSED` は、プログラマ指定の明示的なストップ (`EXPL_STOPS_DISPERSSED`)、および、リソース制限や分岐命令 (プレディケート予測とは無関係) によるディスパーサル・ブレークの合計をカウントする。この合計には、ハードウェアによるスペキュレ・タイプな分岐パスの予測が外れたとき (すなわち、フラッシュのシャドウ内) に検出されたストップが含まれている。
- **イベント・コード** : `0x2F`、**Umask** : 無視、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 無効、データ・アドレス範囲 : なし

## BRANCH\_EVENT

- **タイトル** : 収集された分岐イベント、**カテゴリ** : 分岐
- **定義** : BRANCH\_EVENT は、分岐イベントの数をカウントする。これには分岐トレース・バッファによって捕捉されたマルチウェイ分岐も含まれる。
- **イベント・コード** : 0x11、**Umask** : 無視、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : なし

## BRANCH\_MULTIWAY.ALL\_PATHS.ALL\_PREDICTIONS

- **タイトル** : マルチウェイ・バンドルでの分岐予測の全回数、**カテゴリ** : 分岐
- **定義** : BRANCH\_MULTIWAY.ALL\_PATHS.ALL\_PREDICTIONS は、マルチウェイ分岐バンドル上で実行されたすべての分岐予測をカウントする。
- **イベント・コード** : 0x0E、**Umask** : 0000、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : なし

## BRANCH\_MULTIWAY.ALL\_PATHS.CORRECT\_PREDICTIONS

- **タイトル** : マルチウェイ・バンドルでの分岐予測の的中回数、**カテゴリ** : 分岐
- **定義** : BRANCH\_MULTIWAY.ALL\_PATHS.CORRECT\_PREDICTIONS は、バックエンドの分岐予測ミス・フラッシュを必要としないマルチウェイ分岐バンドル上で、すべての分岐予測をカウントする。
- **イベント・コード** : 0x0E、**Umask** : 0001、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : なし

## BRANCH\_MULTIWAY.ALL\_PATHS.WRONG\_PATH

- **タイトル** : マルチウェイ・バンドルでのプレディケート予測の不的中回数、**カテゴリ** : 分岐
- **定義** : BRANCH\_MULTIWAY.ALL\_PATHS.WRONG\_PATH は、組み合わせられたプレディケートが正しく予測されないマルチウェイ分岐バンドルの数をカウントする。これに含まれるバンドルには2つある。1つは、分岐命令はどれも分岐しないと予測されているのに、任意の1つの命令が実際に分岐するバンドル、もう1つは、分岐命令が分岐すると予測されて、バンドル内の先行する分岐命令が実際に分岐したか、または予測された命令が処理されなかったバンドルである。どのイベントでも、プロセッサは、フロントエンドを正しいターゲットに向け直す。すなわち、所定のマルチウェイ・バンドルは、1度だけ予測に失敗することがある。



- **イベント・コード** :0x0E、**Umask** :0010、**PMC/PMD** :4, 5, 6, 7、**最大インクリメント/サイクル** :1

- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : なし

#### **BRANCH\_MULTIWAY.ALL\_PATHS.WRONG\_TARGET**

- **タイトル** : マルチウェイ・バンドルでの不的中ターゲット予測、**カテゴリ** : 分岐
- **定義** : BRANCH\_MULTIWAY.ALL\_PATHS.WRONG\_TARGET は、分岐命令が分岐すると正しく予測されているのに、そのターゲットが誤っているマルチウェイ分岐バンドルの数をカウントする。
- **イベント・コード** :0x0E、**Umask** :0011、**PMC/PMD** :4, 5, 6, 7、**最大インクリメント/サイクル** :1
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : なし

#### **BRANCH\_MULTIWAY.NOT\_TAKEN.ALL\_PREDICTIONS**

- **タイトル** : 分岐不成立マルチウェイ・バンドルでの分岐予測の全回数、**カテゴリ** : 分岐
- **定義** : BRANCH\_MULTIWAY.NOT\_TAKEN.ALL\_PREDICTIONS は、BRANCH\_MULTIWAY.ALL\_PATHS.ALL\_PREDICTIONS と似ている。異なるのは、すべての分岐命令が実際に分岐しないマルチウェイ分岐バンドルにしか適用されない点である。
- **イベント・コード** :0x0E、**Umask** :1000、**PMC/PMD** :4, 5, 6, 7、**最大インクリメント/サイクル** :1
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : なし

#### **BRANCH\_MULTIWAY.NOT\_TAKEN.CORRECT\_PREDICTIONS**

- **タイトル** : 分岐不成立マルチウェイ・バンドルでの分岐予測の的中回数、**カテゴリ** : 分岐
- **定義** : BRANCH\_MULTIWAY.NOT\_TAKEN.CORRECT\_PREDICTIONS は、BRANCH\_MULTIWAY.ALL\_PATHS.CORRECT\_PREDICTIONS と似ている。異なるのは、すべての分岐命令が実際に分岐しないマルチウェイ分岐バンドルにしか適用されない点である。
- **イベント・コード** :0x0E、**Umask** :1001、**PMC/PMD** :4, 5, 6, 7、**最大インクリメント/サイクル** :1
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : なし

#### **BRANCH\_MULTIWAY.NOT\_TAKEN.WRONG\_PATH**

- **タイトル** : 分岐不成立マルチウェイ・バンドルでのプレディケート予測の不的中回数、**カテゴリ** : 分岐

- **定義** : BRANCH\_MULTIWAY.NOT\_TAKEN.WRONG\_PATH は、BRANCH\_MULTIWAY.ALL\_PATHS.WRONG\_PATH と似ている。異なるのは、すべての分岐命令が実際に分岐しないマルチウェイ分岐バンドルにしか適用されない点である。
- **イベント・コード** : 0x0E、**Umask** : 1010、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : なし

#### BRANCH\_MULTIWAY.NOT\_TAKEN.WRONG\_TARGET

- **タイトル** : 分岐不成立マルチウェイ・バンドルでの不的中ターゲット予測、**カテゴリ** : 分岐
- **定義** : BRANCH\_MULTIWAY.NOT\_TAKEN.WRONG\_TARGET は、必ず 0 をカウントする。なぜなら、分岐しない分岐は、分岐ターゲットを指定しないからである。
- **イベント・コード** : 0x0E、**Umask** : 1011、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : なし

#### BRANCH\_MULTIWAY.TAKEN.ALL\_PREDICTIONS

- **タイトル** : 分岐成立マルチウェイ・バンドルでの分岐予測の全回数、**カテゴリ** : 分岐
- **定義** : BRANCH\_MULTIWAY.TAKEN.ALL\_PREDICTIONS は BRANCH\_MULTIWAY.ALL\_PATHS.ALL\_PREDICTIONS と似ている。異なるのは、分岐命令が少なくとも 1 つ分岐するマルチウェイ分岐バンドルにしか適用されない点である。
- **イベント・コード** : 0x0E、**Umask** : 1100、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : なし

#### BRANCH\_MULTIWAY.TAKEN.CORRECT\_PREDICTIONS

- **タイトル** : 分岐成立マルチウェイ・バンドルでの分岐予測の的中回数、**カテゴリ** : 分岐
- **定義** : BRANCH\_MULTIWAY.TAKEN.CORRECT\_PREDICTIONS は、BRANCH\_MULTIWAY.ALL\_PATHS.CORRECT\_PREDICTIONS と似ている。異なるのは、分岐命令が少なくとも 1 つ分岐するマルチウェイ分岐バンドルにしか適用されない点である。
- **イベント・コード** : 0x0E、**Umask** : 1101、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : なし



### BRANCH\_MULTIWAY.TAKEN.WRONG\_PATH

- **タイトル** : 分岐成立マルチウェイ・バンドルでのプレディケート予測の的中回数、**カテゴリ** : 分岐
- **定義** : BRANCH\_MULTIWAY.TAKEN.WRONG\_PATH は、BRANCH\_MULTIWAY.ALL\_PATHS.WRONG\_PATH と似ている。異なるのは、分岐命令が少なくとも1つ分岐するマルチウェイ分岐バンドルにしか適用されない点である。
- **イベント・コード** : 0x0E、**Umask** : 1110、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : なし

### BRANCH\_MULTIWAY.TAKEN.WRONG\_TARGET

- **タイトル** : 分岐成立マルチウェイ・バンドルでの的中ターゲット予測、**カテゴリ** : 分岐
- **定義** : BRANCH\_MULTIWAY.TAKEN.WRONG\_TARGET は、BRANCH\_MULTIWAY.ALL\_PATHS.WRONG\_TARGET と同じである。なぜなら、分岐命令が少なくとも1つ分岐するマルチウェイ分岐バンドルに、実際にはターゲットを指定しないからである。
- **イベント・コード** : 0x0E、**Umask** : 1111、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : なし

### BRANCH\_PATH.ALL.NT\_OUTCOMES\_CORRECTLY\_PREDICTED

- **タイトル** : 不成立プレディケート予測の的中回数、**カテゴリ** : 分岐
- **定義** : BRANCH\_PATH.ALL.NT\_OUTCOMES\_CORRECTLY\_PREDICTED は、分岐予測とは無関係に、分岐しない分岐上で、実行しないと正しくプレディケート予測した数をカウントする。
- **イベント・コード** : 0x0F、**Umask** : 0010、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : なし

### BRANCH\_PATH.ALL.TK\_OUTCOMES\_CORRECTLY\_PREDICTED

- **タイトル** : 成立プレディケート予測の的中回数、**カテゴリ** : 分岐
- **定義** : BRANCH\_PATH.ALL.TK\_OUTCOMES\_CORRECTLY\_PREDICTED は、分岐予測とは無関係に、分岐した分岐上で、実行すると正しくプレディケート予測した数をカウントする。このプレディケートのみ正しければよい。分岐が実際に分岐する限り、このモニタは、誤ったターゲットへ分岐してしまった場合もカウントする。
- **イベント・コード** : 0x0F、**Umask** : 0011、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1

- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : なし

#### BRANCH\_PATH.ALL.NT\_OUTCOMES\_INCORRECTLY\_PREDICTED

- **タイトル** : 成立プレディケート予測の不的中回数、**カテゴリ** : 分岐
- **定義** : BRANCH\_PATH.ALL.NT\_OUTCOMES\_INCORRECTLY\_PREDICTED は、分岐予測とは無関係に、分岐しない分岐上の実行されると誤ったプレディケート予測の数をカウントする。
- **イベント・コード** : 0x0F、**Umask** : 0000、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : なし

#### BRANCH\_PATH.ALL.TK\_OUTCOMES\_INCORRECTLY\_PREDICTED

- **タイトル** : 不成立プレディケート予測の不的中回数、**カテゴリ** : 分岐
- **定義** : BRANCH\_PATH.ALL.TK\_OUTCOMES\_INCORRECTLY\_PREDICTED は、分岐予測とは無関係に、分岐した分岐上の実行されないと誤ったプレディケート予測の数をカウントする。
- **イベント・コード** : 0x0F、**Umask** : 0001、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : なし

#### BRANCH\_PATH.1ST\_STAGE.NT\_OUTCOMES\_CORRECTLY\_PREDICTED

- **タイトル** : 第 1 パイプライン・ステージでの不成立プレディケート予測の的中回数、**カテゴリ** : 分岐
- **定義** : BRANCH\_PATH.1ST\_STAGE.NT\_OUTCOMES\_CORRECTLY\_PREDICTED は、必ず 0 をカウントする。なぜなら、TAR はコア・パイプラインの第 1 段で唯一のプレディクタであり、実行するとプレディケート予測するからである。
- **イベント・コード** : 0x0F、**Umask** : 0110、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : なし

#### BRANCH\_PATH.1ST\_STAGE.TK\_OUTCOMES\_CORRECTLY\_PREDICTED

- **タイトル** : 第 1 パイプライン・ステージでの成立プレディケート予測の的中回数、**カテゴリ** : 分岐



- **定義** : BRANCH\_PATH.1ST\_STAGE.TK\_OUTCOMES\_CORRECTLY\_PREDICTED は、コア・パイプラインの1段目で TAR で分岐した分岐上の実行する正しいプレディケート予測の数をカウントする。このプレディケートのみ正しければよい。分岐が実際に分岐する限り、このモニタは、誤ったターゲットへ分岐してしまった場合もカウントする。分岐およびその予測ターゲットの間に、バブルは存在しない。
- **イベント・コード** : 0x0F、Umask : 0111、PMC/PMD : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : なし

#### BRANCH\_PATH.1ST\_STAGE.NT\_OUTCOMES\_INCORRECTLY\_PREDICTED

- **タイトル** : 第1パイプライン・ステージでの成立プレディケート予測の的中回数、**カテゴリ** : 分岐
- **定義** : BRANCH\_PATH.1ST\_STAGE.NT\_OUTCOMES\_INCORRECTLY\_PREDICTED は、コア・パイプラインの1段目で TAR で分岐しない分岐上の実行すると誤ったプレディケート予測の数をカウントする。
- **イベント・コード** : 0x0F、Umask : 0100、PMC/PMD : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : なし

#### BRANCH\_PATH.1ST\_STAGE.TK\_OUTCOMES\_INCORRECTLY\_PREDICTED

- **タイトル** : 第1パイプライン・ステージでの成立プレディケート予測の的中回数、**カテゴリ** : 分岐
- **定義** : BRANCH\_PATH.1ST\_STAGE.TK\_OUTCOMES\_INCORRECTLY\_PREDICTED は、必ず0をカウントする。なぜなら、TAR はコア・パイプラインの第1段で唯一のプレディクタであり、実行する予測しか作成しないからである。
- **イベント・コード** : 0x0F、Umask : 0101、PMC/PMD : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : なし

#### BRANCH\_PATH.2ND\_STAGE.NT\_OUTCOMES\_CORRECTLY\_PREDICTED

- **タイトル** : 第2パイプライン・ステージでの成立プレディケート予測の的中回数、**カテゴリ** : 分岐
- **定義** : BRANCH\_PATH.2ND\_STAGE.NT\_OUTCOMES\_CORRECTLY\_PREDICTED は、コア・パイプラインの2段目で BPT/MBPT により分岐しない分岐上の実行しない正しいプレディケート予測の数をカウントする。

- **イベント・コード** :0x0F、**Umask** :1010、**PMC/PMD** :4, 5, 6, 7、**最大インクリメント / サイクル** :1
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : なし

#### **BRANCH\_PATH.2ND\_STAGE.TK\_OUTCOMES\_CORRECTLY\_PREDICTED**

- **タイトル** : 第 2 パイプライン・ステージでの成立プレディケート予測の的中回数、**カテゴリ** : 分岐
- **定義** : BRANCH\_PATH.2ND\_STAGE.TK\_OUTCOMES\_CORRECTLY\_PREDICTED は、コア・パイプラインの 2 段目で BPT/MBPT または TAC によって分岐した分岐上の実行する正しいプレディケート予測の数をカウントする。このプレディケートのみ正しければよい。分岐が実際に処理される限り、このモニタは、誤ったターゲットへ分岐してしまった場合もカウントする。分岐およびその予測ターゲットの間に、バブルは 1 つ存在する。
- **イベント・コード** :0x0F、**Umask** :1011、**PMC/PMD** :4, 5, 6, 7、**最大インクリメント / サイクル** :1
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : なし

#### **BRANCH\_PATH.2ND\_STAGE.NT\_OUTCOMES\_INCORRECTLY\_PREDICTED**

- **タイトル** : 第 2 パイプライン・ステージでの不成立プレディケート予測の的中回数、**カテゴリ** : 分岐
- **定義** : BRANCH\_PATH.2ND\_STAGE.NT\_OUTCOMES\_INCORRECTLY\_PREDICTED は、コア・パイプラインの 2 段目で BPT/MBPT または TAC によって分岐しない分岐上の実行すると誤ったプレディケート予測の数をカウントする。
- **イベント・コード** :0x0F、**Umask** :1000、**PMC/PMD** :4, 5, 6, 7、**最大インクリメント / サイクル** :1
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : なし

#### **BRANCH\_PATH.2ND\_STAGE.TK\_OUTCOMES\_INCORRECTLY\_PREDICTED**

- **タイトル** : 第 2 パイプライン・ステージでの不成立プレディケート予測の的中回数、**カテゴリ** : 分岐
- **定義** : BRANCH\_PATH.2ND\_STAGE.TK\_OUTCOMES\_INCORRECTLY\_PREDICTED は、コア・パイプラインの 2 段目で BPT/MBPT によって分岐した分岐上の実行しないと誤ったプレディケート予測の数をカウントする。
- **イベント・コード** :0x0F、**Umask** :1001、**PMC/PMD** :4, 5, 6, 7、**最大インクリメント / サイクル** :1



- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : なし

#### **BRANCH\_PATH.3RD\_STAGE.NT\_OUTCOMES\_CORRECTLY\_PREDICTED**

- **タイトル** : 第 3 パイプライン・ステージでの不成立プレディケート予測の的中回数、**カテゴリ** : 分岐
- **定義** : BRANCH\_PATH.3RD\_STAGE.NT\_OUTCOMES\_CORRECTLY\_PREDICTED は、コア・パイプラインの 3 段目で BAC によって分岐しない分岐上の実行しない正しいプレディケート予測の数をカウントする。これには、ループ分岐の最新のインスタンス上で、TAR の実行予測 (1 段目で設定された) をオーバーライドすることも含まれる。
- **イベント・コード** : 0x0F、**Umask** : 1110、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : なし

#### **BRANCH\_PATH.3RD\_STAGE.TK\_OUTCOMES\_CORRECTLY\_PREDICTED**

- **タイトル** : 第 3 パイプライン・ステージでの成立プレディケート予測の的中回数、**カテゴリ** : 分岐
- **定義** : BRANCH\_PATH.3RD\_STAGE.TK\_OUTCOMES\_CORRECTLY\_PREDICTED は、コア・パイプラインの 3 段目で BAC によって分岐した分岐上の実行する正しいプレディケート予測の数をカウントする。このプレディケートのみ正しければよい。分岐が実際に処理される限り、このモニタは、誤ったターゲットへ分岐してしまった場合もカウントする。分岐およびその予測ターゲットの間に、バブルは 2 つ存在する (スロット 0 またはスロット 1 の分岐シラブルについてターゲットを計算する必要がある場合は、バブルが 3 つ存在する)。
- **イベント・コード** : 0x0F、**Umask** : 1111、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : なし

#### **BRANCH\_PATH.3RD\_STAGE.NT\_OUTCOMES\_INCORRECTLY\_PREDICTED**

- **タイトル** : 第 3 パイプライン・ステージでの成立プレディケート予測の的中回数、**カテゴリ** : 分岐
- **定義** : BRANCH\_PATH.3RD\_STAGE.NT\_OUTCOMES\_INCORRECTLY\_PREDICTED は、コア・パイプラインの 3 段目で BAC によって分岐しない分岐上の実行すると誤ったプレディケート予測の数をカウントする。
- **イベント・コード** : 0x0F、**Umask** : 1100、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1

- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : なし

#### BRANCH\_PATH.3RD\_STAGE.TK\_OUTCOMES\_INCORRECTLY\_PREDICTED

- **タイトル** : 第 3 パイプライン・ステージでの不成立プレディケート予測の的不中回数、**カテゴリ** : 分岐
- **定義** : BRANCH\_PATH.3RD\_STAGE.TK\_OUTCOMES\_INCORRECTLY\_PREDICTED は、コア・パイプラインの 3 段目で BAC によって分岐した分岐上の実行しないと誤ったプレディケート予測の数をカウントする。これには、ループ分岐の最新のインスタンス上で、TAR の実行予測 (1 段目で設定された) をオーバーライドすることも含まれる。
- **イベント・コード** : 0x0F、**Umask** : 1101、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : なし

#### BRANCH\_PREDICTOR.ALL.ALL\_PREDICTIONS

- **タイトル** : 分岐予測全回数、**カテゴリ** : 分岐
- **定義** : BRANCH\_PREDICTOR.ALL.ALL\_PREDICTIONS は、プロセッサのフロントエンドで発生する分岐予測をすべてカウントする。ただし、このカウントは、コード内の分岐命令の総数とは必ずしも一致しない。なぜなら、分岐予測はバンドル・ベースで実行されるからである (すなわち、1 つのマルチウェイ分岐バンドルに予測は 1 つだけしか存在しない)。
- **イベント・コード** : 0x10、**Umask** : 0000、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : なし

#### BRANCH\_PREDICTOR.ALL.CORRECT\_PREDICTIONS

- **タイトル** : 全プレディクタによる分岐予測の的中回数、**カテゴリ** : 分岐
- **定義** : BRANCH\_PREDICTOR.ALL.CORRECT\_PREDICTIONS は、プレディクタとは無関係に、バックエンドの分岐予測ミス・フラッシュを必要としない分岐予測をすべてカウントする。分岐予測またはターゲットについて、予測された値と実際の値が一致しないと、分岐予測ミスが生じる。戻り分岐は、さらに特権レベルおよび前の関数状態を予測する必要がある。
- **イベント・コード** : 0x10、**Umask** : 0001、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : なし



### BRANCH\_PREDICTOR.ALL.WRONG\_PATH

- **タイトル** : 全ブレイクタによるブレイクテート予測の不的中回数、  
**カテゴリ** : 分岐
- **定義** : BRANCH\_PREDICTOR.ALL.WRONG\_PATH は、ブレイクタとは無関係に、分岐のブレイクテート予測について予測された値と実際の値が異なっていた場合に生じる分岐予測ミスをカウントする。
- **イベント・コード** : 0x10、**Umask** : 0010、**PMC/PMD** : 4, 5, 6, 7、  
**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : なし

### BRANCH\_PREDICTOR.ALL.WRONG\_TARGET

- **タイトル** : 全ブレイクタによる不的中ターゲット予測、**カテゴリ** : 分岐
- **定義** : BRANCH\_PREDICTOR.ALL.WRONG\_TARGET は、ブレイクタとは無関係に、分岐ターゲットについて予測された値と実際の値が異なっていた場合に生じる分岐予測ミスをカウントする。
- **イベント・コード** : 0x10、**Umask** : 0011、**PMC/PMD** : 4, 5, 6, 7、  
**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : なし

### BRANCH\_PREDICTOR.1ST\_STAGE.ALL\_PREDICTIONS

- **タイトル** : 第 1 パイプライン・ステージでの分岐予測の全回数、  
**カテゴリ** : 分岐
- **定義** : BRANCH\_PREDICTOR.1ST\_STAGE.ALL\_PREDICTIONS は、コア・パイプラインの 1 段目で TAR により実行された分岐予測の数をカウントする。TAR は、このパイプラインの 1 段目で動作しているただ一つのブレイクタで、分岐する予測しか行わない。3 段目の PLP は、ループ分岐上で TAR ブレイクテート予測をオーバーライドする。予測のフローは以下のようなになる。

```
if (TAR Hit)
    monitor++
    Read Target from TAR
```

- **イベント・コード** : 0x10、**Umask** : 0100、**PMC/PMD** : 4, 5, 6, 7、  
**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : なし

### BRANCH\_PREDICTOR.1ST\_STAGE.CORRECT\_PREDICTIONS

- **タイトル** : 第 1 パイプライン・ステージでの分岐予測の的中回数、  
**カテゴリ** : 分岐

- **定義** : BRANCH\_PREDICTOR.1ST\_STAGE.CORRECT\_PREDICTIONS は、プレディケートおよびターゲットの TAR が処理する、正しく予測された分岐の数をカウントする。
- **イベント・コード** : 0x10、**Umask** : 0101、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : なし

#### BRANCH\_PREDICTOR.1ST\_STAGE.WRONG\_PATH

- **タイトル** : 第 1 パイプライン・ステージでのプレディケート予測の的不中回数、**カテゴリ** : 分岐
- **定義** : BRANCH\_PREDICTOR.1ST\_STAGE.WRONG\_PATH は、TAR により予測された、実際には分岐しない分岐の数をカウントする (PLP によるオーバーライドは除く)。
- **イベント・コード** : 0x10、**Umask** : 0110、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : なし

#### BRANCH\_PREDICTOR.1ST\_STAGE.WRONG\_TARGET

- **タイトル** : 第 1 パイプライン・ステージでの的不中ターゲット予測、**カテゴリ** : 分岐
- **定義** : BRANCH\_PREDICTOR.1ST\_STAGE.WRONG\_TARGET は、TAR によって誤ったターゲットへ分岐した分岐の数をカウントする。
- **イベント・コード** : 0x10、**Umask** : 0111、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : なし

#### BRANCH\_PREDICTOR.2ND\_STAGE.ALL\_PREDICTIONS

- **タイトル** : 第 2 パイプライン・ステージでの分岐予測の全回数、**カテゴリ** : 分岐
- **定義** : BRANCH\_PREDICTOR.2ND\_STAGE.ALL\_PREDICTIONS は、コア・パイプラインの 2 段目で実行された分岐予測の数をカウントする。この段では、BPT および MBPT(プレディケート用)、TAC および RSB(ターゲット用)などの構造が機能する。2 段目では、1 段目で予測が実行されなかった場合に限り、予測が実行される。予測は、この段で実行されたらカウントされるが、分岐するプレディケート予測が BPT/MBPT により得られるリターン以外の分岐の場合はカウントされない。また、TAC から利用できるターゲットがない場合もカウントされない。分岐予測構造は、以下の方法で相互に作用する。

```

if ((BPT Hit) or (MBPT Hit))
    if (Predicted Taken)
        if (Predicted Return Branch)

```



```
        monitor++
        Read Target from RSB
    else
        if (TAC Hit)
            monitor++
            Read Target from TAC
        else
            Get Target from BAC in the 3rd Stage
    else
        monitor++
        Follow Sequential Path
else
    if (TAC Hit)
        monitor++
        Read Target from TAC
    else
        Follow Sequential Path
```

- **イベント・コード** :0x10、**Umask** :1000、**PMC/PMD** :4, 5, 6, 7、**最大インクリメント/サイクル** :1
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : なし

#### **BRANCH\_PREDICTOR.2ND\_STAGE.CORRECT\_PREDICTIONS**

- **タイトル** : 第 2 パイプライン・ステージでの分岐予測の的中回数、**カテゴリ** : 分岐
- **定義** : BRANCH\_PREDICTOR.2ND\_STAGE.CORRECT\_PREDICTIONS は、BPT/MBPT、またはコア・パイプラインの 2 段目の TAC により実行される正しいプレディケート予測の数をカウントする。プレディケート予測で分岐する場合は、この段で RSB または TAC により正しいターゲットを指定する必要がある。BPT/MBPT により、正しく分岐をプレディケート予測したもののうち、TAC をミスしたりターン以外の分岐では、パイプラインの 3 段目で BAC により分岐先を得る必要があり、カウントされない。
- **イベント・コード** :0x10、**Umask** :1001、**PMC/PMD** :4, 5, 6, 7、**最大インクリメント/サイクル** :1
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : なし

#### **BRANCH\_PREDICTOR.2ND\_STAGE.WRONG\_PATH**

- **タイトル** : 第 2 パイプライン・ステージでのプレディケート予測の的中回数、**カテゴリ** : 分岐
- **定義** : BRANCH\_PREDICTOR.2ND\_STAGE.WRONG\_PATH は、コア・パイプラインの 2 段目で実行された分岐しないと誤ったプレディケート予測の数をカウントする。また、分岐先が得られた場合は、この段で実行された分岐すると誤ったプレディケート予測の数もカウントする。
- **イベント・コード** :0x10、**Umask** :1010、**PMC/PMD** :4, 5, 6, 7、**最大インクリメント/サイクル** :1

- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : なし

### BRANCH\_PREDICTOR.2ND\_STAGE.WRONG\_TARGET

- **タイトル** : 第 2 パイプライン・ステージでの不的中ターゲット予測、  
**カテゴリ** : 分岐
- **定義** : BRANCH\_PREDICTOR.2ND\_STAGE.WRONG\_TARGET は、BPT/MBPT または TAC によって正しく分岐を予測したが、RSB または TAC によって誤ったターゲットに分岐した数をカウントする。
- **イベント・コード** : 0x10、**Umask** : 1011、**PMC/PMD** : 4, 5, 6, 7、  
**最大インクリメント/サイクル** : 1
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : なし

### BRANCH\_PREDICTOR.3RD\_STAGE.ALL\_PREDICTIONS

- **タイトル** : 第 3 パイプライン・ステージでの分岐予測の全回数、  
**カテゴリ** : 分岐
- **定義** : BRANCH\_PREDICTOR.3RD\_STAGE.ALL\_PREDICTIONS は、コア・パイプラインの 3 段目で BAC により実行された分岐予測の数をカウントする。BAC は、以下の方法により、プレディケート予測を実行することも (分岐のヒント・フィールドに基づいて)、分岐先予測を実行することもできる。

```

if (TAR Hit)
    if (Predicted Last Instance of Loop-Closing Branch)
        monitor++
        PLP Override of TAR Taken Prediction
        Resteer Frontend to Sequential Address
else
    if ((BPT Hit) or (MBPT Hit))
        if (Predicted Taken)
            if (not (TAC Hit))
                if (not (Predicted Return Branch))
                    monitor++
                    Compute Target
        else
            if (not (TAC Hit))
                monitor++
                Read Whether Hint Field for Predicate Prediction
                if (Predicted Taken)
                    Read BType Field for Type Information
                    if (Indirect Branch)
                        Read Target from RSB
                    else
                        Compute Target
            else
                Follow Sequential Path

```



- **イベント・コード** :0x10、**Umask** :1100、**PMC/PMD** :4, 5, 6, 7、**最大インクリメント/サイクル** :1
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : なし

#### **BRANCH\_PREDICTOR.3RD\_STAGE.CORRECT\_PREDICTIONS**

- **タイトル** : 第 3 パイプライン・ステージでの分岐予測の的中回数、**カテゴリ** : 分岐
- **定義** : BRANCH\_PREDICTOR.3RD\_STAGE.CORRECT\_PREDICTIONS は、BAC により実行された正しい分岐予測の数をカウントする。これには、異なるプレディクタによりプレディケート予測された分岐のターゲット予測も含まれる。分岐する予測の場合、プレディケートとターゲットはともに正しくなければならない。
- **イベント・コード** :0x10、**Umask** :1101、**PMC/PMD** :4, 5, 6, 7、**最大インクリメント/サイクル** :1
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : なし

#### **BRANCH\_PREDICTOR.3RD\_STAGE.WRONG\_PATH**

- **タイトル** : 第 3 パイプライン・ステージでのプレディケート予測の的中回数、**カテゴリ** : 分岐
- **定義** : BRANCH\_PREDICTOR.3RD\_STAGE.WRONG\_PATH は、BAC により (分岐のヒント・フィールドに基づいて) 誤ってプレディケート予測された分岐をカウントする。他のプレディクタが分岐をプレディケート予測して BAC が分岐先を提供した、分岐しない分岐もカウントする。
- **イベント・コード** :0x10、**Umask** :1110、**PMC/PMD** :4, 5, 6, 7、**最大インクリメント/サイクル** :1
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : なし

#### **BRANCH\_PREDICTOR.3RD\_STAGE.WRONG\_TARGET**

- **タイトル** : 第 3 パイプライン・ステージでのターゲット予測の的中回数、**カテゴリ** : 分岐
- **定義** : BRANCH\_PREDICTOR.3RD\_STAGE.WRONG\_TARGET は、任意のプレディクタにより正しく分岐を予測されたが、その分岐先は BAC により誤って指定された、分岐した分岐をカウントする。
- **イベント・コード** :0x10、**Umask** :1111、**PMC/PMD** :4, 5, 6, 7、**最大インクリメント/サイクル** :1
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : なし

#### **BRANCH\_TAKEN\_SLOT**

- **タイトル** : 成立分岐の詳細、**カテゴリ** : 分岐

- **定義**：BRANCH\_TAKEN\_SLOT は、分岐した分岐が分岐バンドル（シングルウェイまたはマルチウェイ）内のどのスロット番号を使用するのかを監視したり、所定の分岐バンドル内に分岐した分岐がないことを記録したりする。動的な br.calls と br.rets をカウントするときは、IA64\_TAGGED\_INST\_RETIRED ではなくダウンストリーム・オペコード・マッチ機能と組み合わせてこのモニタを使うこと。
- **イベント・コード**：0x0D、**Umask**：以下を参照、**PMC/PMD**：4, 5, 6, 7、**最大インクリメント / サイクル**：1
- **条件**：命令アドレス範囲：あり、オペコード・マッチ機能：有効、データ・アドレス範囲：なし

表 7-29 に示す SLOT\_MASK ユニット・マスクを使うと、各命令スロット番号に基づいて成立分岐のプロファイリングができる。SLOT\_MASK に複数のビットをセットする場合は、セット・ケースはすべてイベント発生回数に含まれる。プロセッサは、以下の計算式によりサイクルごとのイベントの結果を判定する。

```
(PMC.umask{16}
  and (branch in slot 0 is taken))
or (PMC.umask{17}
  and (branch in slot 0 is NOT taken)
  and (branch in slot 1 is taken))
or (PMC.umask{18}
  and (branch in slot 0 is NOT taken)
  and (branch in slot 1 is NOT taken)
  and (branch in slot 2 is taken))
or (PMC.umask{19}
  and (branch in slot 0 is NOT taken)
  and (branch in slot 1 is NOT taken)
  and (branch in slot 2 is NOT taken))
```

表 7-29. BRANCH\_TAKEN\_SLOT のスロット・ユニット・マスク

SLOT_MASK	PMC.umask {19:16}	説明
命令スロット 0	xxx1	スロット 0 での分岐が最初の成立分岐である場合にカウントする
命令スロット 1	xx1x	スロット 1 での分岐が最初の成立分岐である場合にカウントする
命令スロット 2	x1xx	スロット 2 での分岐が最初の成立分岐である場合にカウントする
成立分岐がない	1xxx	分岐が 1 つも成立しなかった場合にカウントする。

## BUS\_ALL

- **タイトル**：バス・トランザクション、**カテゴリ**：フロントサイド・バス
- **定義**：BUS\_ALL は、バスに向けて発行されたトランザクションをすべてカウントする。これには、BRL、BRIL、BIL、BWL、BRP、BWP、IORD、IOWR、その他のトランザクションが含まれる。



- **イベント・コード** :0x47、**Umask** :7.6.5 項を参照、**PMC/PMD** :4, 5, 6, 7、**最大インクリメント / サイクル** :1
- **条件** : 命令アドレス範囲 : なし、オペコード・マッチ機能 : 無効、データ・アドレス範囲 : なし

### **BUS\_BRQ\_LIVE\_REQ\_LO および BUS\_BRQ\_LIVE\_REQ\_HI**

- **タイトル** : BRQ ライブ要求、**カテゴリ** : フロントサイド・バス
- **定義** : BUS\_BRQ\_LIVE\_REQ は、BRQ (バス要求キュー) に含まれているライブ・エントリの数をカウントする。このイベントには、L3 キャッシュ読み取りのほか、BRL、BRIL、BRP、IORD の各メモリ・トランザクションが含まれる。キャッシュ・ライン・ライトバック、パーシャル・ライト (BWP および IOWR)、ライト・コーレシング RFO はどれもカウントされない。これは、それぞれ独自の書き込みキューを持っているからである。このパフォーマンス・モニタでは、バス・クロックではなくコア・クロックごとにそのカウント数が増えていく。
- **イベント・コード** :0x5c (HI)、0x5b (LO)、**Umask** :無視、**PMC/PMD** :4, 5, 6, 7、**最大インクリメント / サイクル** :各 2
- **条件** : 命令アドレス範囲 : なし、オペコード・マッチ機能 : 無効、データ・アドレス範囲 : なし

これをカウントするには、次の 2 つのハードウェア・イベントが必要である。

BUS\_BRQ\_LIVE\_REQ\_HI - 4 ビットの未処理 BRQ 要求の数の最上位 2 ビットをカウントする。

BUS\_BRQ\_LIVE\_REQ\_LO - 4 ビットの未処理 BRQ 要求の数の最下位 2 ビットをカウントする。

### **BUS\_BRQ\_READ\_REQ\_INSERTED**

- **タイトル** : 挿入された BRQ 要求、**カテゴリ** : フロントサイド・バス
- **定義** : BUS\_BRQ\_READ\_REQ\_INSERTED は、BRQ に挿入された読み取り (BRL) と RFO (BRIL) の要求の数をカウントする。キャッシュ・ライン・ライトバック、パーシャル・ライト、コーレシング・ライトはどれもカウントされない。これは、それぞれが独自の書き込みキューを持っているからである。
- **イベント・コード** :0x5d、**Umask** :無視、**PMC/PMD** :4, 5, 6, 7、**最大インクリメント / サイクル** :1
- **条件** : 命令アドレス範囲 : なし、オペコード・マッチ機能 : 無効、データ・アドレス範囲 : なし

BRQ への挿入を利用することで、離れた位置での複合キャッシュ / バス・レイテンシを直接計測することができる。それには次のようにすればよい。

```
avg_brq_req_outstanding_per_cycle =  
(BUS_BRQ_LIVE_REQ / デルタ (サイクル))  
avg_brq_latency = (BUS_BRQ_LIVE_REQ /  
BUS_BRQ_READ_REQ_INSERTED)
```

ただし、BRQ に挿入されたデータのうちトラッキングされたものは、読み取り値と RFO とについては保持しているが、ライト・コーレシングのライトバックについては保持していないことに注意が必要である。

### BUS\_BURST

- **タイトル** :バス・バースト・トランザクション、**カテゴリ** :フロントサイド・バス
- **定義** :BUS\_BURST は、フル・キャッシュ・ライン (バースト・モード) バス・メモリ・トランザクションの数をカウントする。これには、BRL、BRIL、BWL の各トランザクションが含まれる。
- **イベント・コード** :0x49、**Umask** :7.6.5 項を参照、**PMC/PMD** :4, 5, 6, 7、**最大インクリメント / サイクル** :1
- **条件** :命令アドレス範囲 :なし、オペコード・マッチ機能 :無効、データ・アドレス範囲 :なし

### BUS\_HITM

- **タイトル** :バス・ヒット・モディファイド・ライン・トランザクション、**カテゴリ** :フロントサイド・バス
- **定義** :BUS\_HITM は、HITM がアサートされる原因となったメモリ・トランザクションの数をカウントする。このパフォーマンス・モニタには、BRL、BWL、BRIL、BIL の各メモリ・トランザクションが含まれる。このプロセッサが起点となったイベントだけがカウントされる。
- **イベント・コード** :0x44、**Umask** :無視、**PMC/PMD** :4, 5, 6, 7、**最大インクリメント / サイクル** :1
- **条件** :命令アドレス範囲 :なし、オペコード・マッチ機能 :無効、データ・アドレス範囲 :なし

### BUS\_IO

- **タイトル** :IA-32 互換 I/O バス・トランザクション、**カテゴリ** :フロントサイド・バス
- **定義** :BUS\_IO は、I/O トランザクションの数をカウントする。IORD または IOWR のトランザクションが含まれる。
- **イベント・コード** :0x50、**Umask** :7.6.5 項を参照、**PMC/PMD** :4, 5, 6, 7、**最大インクリメント / サイクル** :1
- **条件** :命令アドレス範囲 :なし、オペコード・マッチ機能 :無効、データ・アドレス範囲 :なし

注 : サポートされているユニット・マスクは、「Any」および「Self」のみ。

### BUS\_IOQ\_LIVE\_REQ\_LO および BUS\_IOQ\_LIVE\_REQ\_HI

- **タイトル** :インオーダー・バス・キュー要求、**カテゴリ** :フロントサイド・バス

- **定義** : BUS\_IOQ\_LIVE\_REQ は、インオーダー・バス・キュー内のライブ・バス要求の数をカウントする。このパフォーマンス・モニタでは、バス・クロックではなくコア・クロックごとにそのカウント数が増えていく。

- **イベント・コード** : 0x58 (HI)、0x57 (LO)、**Umask** : 無視、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 各 3

- **条件** : 命令アドレス範囲 : なし、オペコード・マッチ機能 : 無効、データ・アドレス範囲 : なし

また BUS\_IOQ\_LIVE\_REQ は、 $BUS\_IOQ\_LIVE\_REQ\_HI * 4 + BUS\_IOQ\_LIVE\_REQ\_LO$  という式で計算できる。

これをカウントするには、次の 2 つのハードウェア・イベントが必要である。

BUS\_IOQ\_LIVE\_REQ\_HI - 4 ビットの未処理 IOQ 要求の数の最上位 2 ビットのライトバックをカウントする。

BUS\_IOQ\_LIVE\_REQ\_LO - 4 ビットの未処理 IOQ 要求の数の最下位 2 ビットをカウントする。

## BUS\_LOCK

- **タイトル** : IA-32 互換バス・ロック・トランザクション、**カテゴリ** : フロントサイド・バス

- **定義** : BUS\_LOCK は、IA-32 互換バス・ロック・トランザクションの数をカウントする。

- **イベント・コード** : 0x53、**Umask** : 7.6.5 項を参照、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1

- **条件** : 命令アドレス範囲 : なし、オペコード・マッチ機能 : 無効、データ・アドレス範囲 : なし

**注** : サポートされているユニット・マスクは「Any」のみ。

## BUS\_LOCK\_CYCLES

- **タイトル** : IA-32 互換バス・ロック・サイクル、**カテゴリ** : フロントサイド・バス

- **定義** : BUS\_LOCK\_CYCLES は、IA-32 互換バス・ロック・トランザクションが原因でバスがロックされるバス・クロック数をカウントする。

- **イベント・コード** : 0x54、**Umask** : 7.6.5 項を参照、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1

- **条件** : 命令アドレス範囲 : なし、オペコード・マッチ機能 : 無効、データ・アドレス範囲 : なし

**注** : サポートされているユニット・マスクは「Any」のみ。

## BUS\_MEMORY

- **タイトル** : バス・メモリ・トランザクション、**カテゴリ** : フロントサイド・バス

- **定義** : BUS\_MEMORY は、バス・メモリ・トランザクションの数をカウントする。これには、BRL、BRIL、BIL、BWL、BRP、BWP の各トランザクションが含まれる。
- **イベント・コード** : 0x4a、**Umask** : 7.6.5 項を参照、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : なし、オペコード・マッチ機能 : 無効、データ・アドレス範囲 : なし

#### BUS\_PARTIAL

- **タイトル** : バス・パーシャル・トランザクション、**カテゴリ** : フロントサイド・バス
- **定義** : BUS\_PARTIAL は、パーシャル・バス・メモリ・トランザクションの数をカウントする。これには、BRP および BWP の各トランザクションが含まれる。
- **イベント・コード** : 0x48、**Umask** : 7.6.5 項を参照、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : なし、オペコード・マッチ機能 : 無効、データ・アドレス範囲 : なし

#### BUS\_RD\_ALL

- **タイトル** : バス読み取りトランザクション、**カテゴリ** : フロントサイド・バス
- **定義** : BUS\_RD\_ALL は、BRL メモリ・トランザクションの数をカウントする。これには、コードおよびデータの BRL トランザクションが含まれる。
- **イベント・コード** : 0x4b、**Umask** : 7.6.5 項を参照、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : なし、オペコード・マッチ機能 : 無効、データ・アドレス範囲 : なし

#### BUS\_RD\_DATA

- **タイトル** : バス読み取りデータ・トランザクション、**カテゴリ** : フロントサイド・バス
- **定義** : BUS\_RD\_DATA は、BRL データ・トランザクションの数をカウントする。
- **イベント・コード** : 0x4c、**Umask** : 7.6.5 項を参照、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : なし、オペコード・マッチ機能 : 無効、データ・アドレス範囲 : なし

### BUS\_RD\_HIT

- **タイトル** :バス読み取りヒット・クリーン・ノンローカル・キャッシュ・トランザクション、**カテゴリ** :フロントサイド・バス
- **定義** :BUS\_RD\_HIT は、HIT がアサートされる原因となった BRL メモリ・トランザクションの数をカウントする。このプロセッサを起点とするイベントだけがカウントされる。
- **イベント・コード** :0x40、**Umask** :無視、**PMC/PMD** :4, 5, 6, 7、**最大インクリメント / サイクル** :1
- **条件** :命令アドレス範囲 :なし、オペコード・マッチ機能 :無効、データ・アドレス範囲 :なし

### BUS\_RD\_HITM

- **タイトル** :バス読み取りヒット・モディファイド・ノンローカル・キャッシュ・トランザクション、**カテゴリ** :フロントサイド・バス
- **定義** :BUS\_RD\_HITM は、HITM がアサートされる原因となった BRL メモリ・トランザクションの数をカウントする。このプロセッサを起点とするイベントだけがカウントされる。
- **イベント・コード** :0x41、**Umask** :無視、**PMC/PMD** :4, 5, 6, 7、**最大インクリメント / サイクル** :1
- **条件** :命令アドレス範囲 :なし、オペコード・マッチ機能 :無効、データ・アドレス範囲 :なし

### BUS\_RD\_INVALID

- **タイトル** :バス読み取り無効化ライン、**カテゴリ** :フロントサイド・バス
- **定義** :BUS\_RD\_INVALID は、BIL メモリ・トランザクションの数をカウントする。Itanium プロセッサでは、このようなトランザクションを生成するのはフラッシュ・キャッシュ命令だけである。
- **イベント・コード** :0x4e、**Umask** :7.6.5 項を参照、**PMC/PMD** :4, 5, 6, 7、**最大インクリメント / サイクル** :1
- **条件** :命令アドレス範囲 :なし、オペコード・マッチ機能 :無効、データ・アドレス範囲 :なし

### BUS\_RD\_INVALID\_BST

- **タイトル** :バス BRIL バースト・トランザクション、**カテゴリ** :フロントサイド・バス
- **定義** :BUS\_RD\_INVALID は、BRIL メモリ・トランザクションの数をカウントする。このようなトランザクションは、一般的にメモリ・ストア、RFO (read for ownership) イベントにより生成される。
- **イベント・コード** :0x4f、**Umask** :7.6.5 項を参照、**PMC/PMD** :4, 5, 6, 7、**最大インクリメント / サイクル** :1
- **条件** :命令アドレス範囲 :なし、オペコード・マッチ機能 :無効、データ・アドレス範囲 :なし

**BUS\_RD\_INVALID\_HITM**

- **タイトル** : HITM におけるバス BIL トランザクションの結果、**カテゴリ** : フロントサイド・バス
- **定義** : BUS\_RD\_INVALID\_HITM は、HITM がアサートされる原因となった BIL トランザクションの数をカウントする。このプロセッサを起点とするイベントだけがカウントされる。
- **イベント・コード** : 0x42、**Umask** : 無視、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : なし、オペコード・マッチ機能 : 無効、データ・アドレス範囲 : なし

**BUS\_RD\_INVALID\_BST\_HITM**

- **タイトル** : HITM におけるバス BRIL バースト・トランザクションの結果、**カテゴリ** : フロントサイド・バス
- **定義** : BUS\_RD\_INVALID\_BST\_HITM は、HITM がアサートされる原因となった BRIL トランザクションの数をカウントする。このプロセッサを起点とするイベントだけがカウントされる。
- **イベント・コード** : 0x43、**Umask** : 無視、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : なし、オペコード・マッチ機能 : 無効、データ・アドレス範囲 : なし

**BUS\_RD\_IO**

- **タイトル** : IA-32 互換 I/O 読み取りトランザクション、**カテゴリ** : フロントサイド・バス
- **定義** : BUS\_RD\_IO は、IORD トランザクションの数をカウントする。
- **イベント・コード** : 0x51、**Umask** : 7.6.5 項を参照、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : なし、オペコード・マッチ機能 : 無効、データ・アドレス範囲 : なし

**注** : サポートされているユニット・マスクは、「Any」および「Self」のみ。

**BUS\_RD\_PRTL**

- **タイトル** : バス読み取りパーシャル・トランザクション、**カテゴリ** : フロントサイド・バス
- **定義** : BUS\_RD\_PRTL は、パーシャル読み取りメモリ・トランザクションの数をカウントする。これには、BRP トランザクションが含まれる。
- **イベント・コード** : 0x4d、**Umask** : 7.6.5 項を参照、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : なし、オペコード・マッチ機能 : 無効、データ・アドレス範囲 : なし

## BUS\_SNOOPS

- **タイトル** :バス・スヌープ・トータル、**カテゴリ** :フロントサイド・バス
- **定義** :BUS\_SNOOPS は、バス・メモリ・トランザクションにより生成された内部スヌープの数をカウントする。
- **イベント・コード** :0x46、**Umask** :7.6.5 項を参照、**PMC/PMD** :4, 5, 6, 7、**最大インクリメント/サイクル** :1
- **条件** :命令アドレス範囲 :なし、オペコード・マッチ機能 :無効、データ・アドレス範囲 :なし

**注** : サポートされているユニット・マスクは「Any」のみ。「Any」は、あらゆるバス・トランザクションから生成された内部スヌープの数をカウントする。

## BUS\_SNOOPS\_HITM

- **タイトル** :バス・スヌープ・ヒット・モディファイド・キャッシュ・ライン、**カテゴリ** :フロントサイド・バス
- **定義** :BUS\_SNOOPS\_HITM は、バス・メモリ・トランザクションにより生成された内部スヌープのうち、ローカル・プロセッサ内のモディファイド・ラインにヒットした数をカウントする。
- **イベント・コード** :0x45、**Umask** :7.6.5 項を参照、**PMC/PMD** :4, 5, 6, 7、**最大インクリメント/サイクル** :1
- **条件** :命令アドレス範囲 :なし、オペコード・マッチ機能 :無効、データ・アドレス範囲 :なし

**注** : サポートされているユニット・マスクは「Any」のみ。「Any」は、あらゆるトランザクションから生成された内部スヌープのうち、モディファイド・ラインにヒットした数をカウントする。

## BUS\_SNOOP\_STALL\_CYCLES

- **タイトル** :バス・スヌープ・ストール・サイクル、**カテゴリ** :フロントサイド・バス
- **定義** :BUS\_SNOOP\_STALL\_CYCLES は、スヌープ・ストールが原因でバスがストールしていたバス・クロック数をカウントする。
- **イベント・コード** :0x55、**Umask** :7.6.5 項を参照、**PMC/PMD** :4, 5, 6, 7、**最大インクリメント/サイクル** :1
- **条件** :命令アドレス範囲 :なし、オペコード・マッチ機能 :無効、データ・アドレス範囲 :なし

**注** : サポートされているユニット・マスクは、「Self」および「Any」のみ。「Self」は、ローカル・プロセッサにより開始されたメモリ・トランザクションに起因するスヌープ・ストールの回数をカウントする。「Any」はスヌープ・ストールをすべてカウントするものであり、すなわちローカル・プロセッサ、その他のプロセッサ、プライオリティ・エージェントのそれぞれから開始されたメモリ・トランザクションに起因するものがすべて対象となる。

**BUS\_SNOOPQ\_REQ**

- **タイトル** :バス・スヌープ・キュー要求、**カテゴリ** :フロントサイド・バス
- **定義** :BUS\_SNOOPQ\_REQ は、未処理のメモリ・トランザクションのうち、スヌープ・フェーズをまだ完了していないものの数をカウントする。このパフォーマンス・モニタでは、バス・クロックではなくコア・クロックごとにそのカウント数が増えていく。BUS\_SNOOPQ\_REQ は、スヌープ・キューに含まれている有効なエントリ数と等しくない。それは、例えば暗黙的ライトバックなどのように、スヌープ・フェーズを過ぎてもエントリがスヌープ・キューの中にとどまっていることがあるためである。
- **イベント・コード** :0x56、**Umask** :無視、**PMC/PMD** :4, 5、**最大インクリメント / サイクル** :3
- **条件** :命令アドレス範囲 :なし、オペコード・マッチ機能 :無効、データ・アドレス範囲 :なし

**BUS\_WR\_WB**

- **タイトル** :バス・ライトバック・トランザクション、**カテゴリ** :フロントサイド・バス
- **定義** :BUS\_WR\_WB は、BWL メモリ・トランザクションの数をカウントする。この種のトランザクションは、明示的ライトバックまたはコーレシング・ライトにより生成される。一般に、スヌープがディセーブルになっている場合は BWL がカウントされる。
- **イベント・コード** :0x52、**Umask** :7.6.5 項を参照、**PMC/PMD** :4, 5, 6, 7、**最大インクリメント / サイクル** :1
- **条件** :命令アドレス範囲 :なし、オペコード・マッチ機能 :無効、データ・アドレス範囲 :なし

**CPU\_CPL\_CHANGES**

- **タイトル** :特権レベルの変更、**カテゴリ** :システム
- **定義** :CPU\_CPL\_CHANGES は、特権レベルの変更回数をカウントする。
- **イベント・コード** :0x34、**Umask** :無視、**PMC/PMD** :4, 5, 6, 7、**最大インクリメント / サイクル** :1
- **条件** :命令アドレス範囲 :なし、オペコード・マッチ機能 :無効、データ・アドレス範囲 :なし

**CPU\_CYCLES**

- **タイトル** :CPU サイクル、**カテゴリ** :システム
- **定義** :CPU\_CYCLES は、経過したプロセッサ・サイクル数をカウントする。
- **イベント・コード** :0x12、**Umask** :無視、**PMC/PMD** :4, 5, 6, 7、**最大インクリメント / サイクル** :1
- **条件** :命令アドレス範囲 :なし、オペコード・マッチ機能 :無効、データ・アドレス範囲 :なし



## DATA\_ACCESS\_CYCLE

- **タイトル** : データ・アクセスによるストール・サイクル、**カテゴリ** : ストール
- **定義** : DATA\_ACCESS\_CYCLE は、キャッシュ・ミス、L1D ウェイ予測ミス、DTC ミスが発生して命令がデータ待ちをしていることが原因でパイプラインがストールしたりフラッシュされたりしている最中のサイクル数をカウントする。
- **イベント・コード** : 0x03、**Umask** : 無視、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : なし、オペコード・マッチ機能 : 無効、データ・アドレス範囲 : なし

## DATA\_EAR\_EVENTS

- **タイトル** : L1 データ・キャッシュ EAR イベント、**カテゴリ** : L1 データ・キャッシュ
- **定義** : DATA\_EAR\_EVENTS は、データ・キャッシュ・ユニット・イベント・アドレス・レジスタにより収集されたデータ・キャッシュまたは DTLB イベントの数をカウントする。
- **イベント・コード** : 0x67、**Umask** : 無視、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : なし

## DATA\_REFERENCES\_RETIRED

- **タイトル** : リタイアしたデータ・メモリ参照、**カテゴリ** : L1 データ・キャッシュ
- **定義** : DATA\_REFERENCES\_RETIRED は、プロセッサ・メモリ・パイプラインによってリタイアされたデータ・メモリ参照の数をカウントする。このカウントには、チェック・ロード、キャッシュ不可アクセス、RSE 操作、VHPT メモリ参照、セマフォ、および FP メモリ参照が含まれる。プレディケート・オフの操作は含まれない。
- **イベント・コード** : 0x63、**Umask** : 無視、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 2
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : あり

## DEPENDENCY\_ALL\_CYCLE

- **タイトル** : スコアボード依存性 / ディスパーサル・ブレイク・サイクル、**カテゴリ** : ストール

- **定義** : DEPENDENCY\_ALL\_CYCLE は、整数演算または FP 演算を相手とするデータ (スコアボード) 依存性によるサイクル数をカウントするか (ただしキャッシュ / メモリ・アクセスはカウントしない)、あるいは発行制限ストール (例えば暗黙のおよび明示的なストップ) の数をカウントする。制御 / アプリケーション・レジスタの読み書きによる浮動小数点フラッシュおよびディレイについてもカウントする。
- **イベント・コード** : 0x06、**Umask** : 無視、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : なし、オペコード・マッチ機能 : 無効、データ・アドレス範囲 : なし

#### DEPENDENCY\_SCOREBOARD\_CYCLE

- **タイトル** : スコアボード依存性サイクル、**カテゴリ** : ストール
- **定義** : DEPENDENCY\_SCOREBOARD\_CYCLE は、整数演算または FP 演算を相手とするデータ (スコアボード) 依存性によるサイクル数をカウントするが、キャッシュ / メモリ・アクセスについてはカウントしない。制御 / アプリケーション・レジスタの読み書きによる浮動小数点フラッシュおよびディレイについてもカウントする。
- **イベント・コード** : 0x02、**Umask** : 無視、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : なし、オペコード・マッチ機能 : 無効、データ・アドレス範囲 : なし

#### DTC\_MISSES

- **タイトル** : DTC ミス、**カテゴリ** : システム
- **定義** : DTC\_MISSES は、データ要求に対する DTC ミスの回数をカウントする。
- **イベント・コード** : 0x60、**Umask** : 無視、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : なし

#### DTLB\_INSERTS\_HPW

- **タイトル** : ハードウェア・ページ・ウォークによる DTLB への挿入、**カテゴリ** : システム
- **定義** : DTLB\_INSERTS\_HPW は、ハードウェア・ページ・テーブル・ウォークによる DTLB への挿入回数をカウントする。
- **イベント・コード** : 0x62、**Umask** : 無視、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : なし

## DTLB\_MISSES

- **タイトル** : DTLB ミス、**カテゴリ** : システム
- **定義** : DTLB\_MISSES は、デマンド要求に対する DTLB ミスの回数をカウントする。
- **イベント・コード** : 0x61、**Umask** : 無視、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : なし

## EXPL\_STOPS\_DISPERSED

- **タイトル** : 配布された明示的ストップ、**カテゴリ** : 命令の発行
- **定義** : EXPL\_STOPS\_DISPERSED は、プログラマが指定した明示的なストップの数をカウントする。これには、ハードウェア・スペキュレーティブの分岐予測が外れたときに検出されたストップも含まれる。
- **イベント・コード** : 0x2E、**Umask** : 無視、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 無効、データ・アドレス範囲 : なし

## FP\_OPS\_RETIRED\_HI

- **タイトル** : リタイアした FP 演算 (High)、**カテゴリ** : 実行
- **定義** : FP\_OPS\_RETIRED\_HI および FP\_OPS\_RETIRED\_LO は、両方ペアで使用して、派生イベント FP\_OPS\_RETIRED.d を計算する。これは、リタイアした FP 演算の重み付けされた合計である。
- **イベント・コード** : 0x0A、**Umask** : 無視、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 3
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : なし

導出値である FP\_OPS\_RETIRED.d は、 $FP\_OPS\_RETIRED\_HI * 4 + FP\_OPS\_RETIRED\_LO$  という式で計算される。個々の FP 演算の加重値は、 $f_{norm}=1$ 、 $f_{add}=1$ 、 $f_{mpy}=1$ 、 $f_{ma}=2$ 、 $f_{ms}=2$ 、 $f_{sub}=1$ 、 $f_{pma}=4$ 、 $f_{pmpy}=4$ 、 $f_{pms}=4$ 、 $f_{nma}=2$ 、 $f_{rcpa}=1$ 、 $f_{rsqrta}=1$ 、 $f_{pnma}=4$ 、 $f_{prcpa}=2$ 、 $f_{prsqrta}=2$ 、 $x_{ma}=0$  である。

**注** : 整数乗算命令 ( $x_{ma}$ ) は、浮動小数点乗数の中で実行される場合でも浮動小数点演算とは見なされない。

## FP\_OPS\_RETIRED\_LO

- **タイトル** : リタイアした FP 演算 (Low)、**カテゴリ** : 実行
- **定義** : FP\_OPS\_RETIRED\_HI および FP\_OPS\_RETIRED\_LO は、両方ペアで使用して、派生イベント FP\_OPS\_RETIRED.d を計算する。これは、リタイアした FP 演算の重み付けされた合計である。

- **イベント・コード** :0x09、**Umask** :無視、**PMC/PMD** :4, 5, 6, 7、**最大インクリメント / サイクル** :3
- **条件** : 7-51 ページの FP\_OPS\_RETIRED\_HI を参照のこと。

#### FP\_FLUSH\_TO\_ZERO

- **タイトル** :ゼロ・フラッシュされた FP の結果、**カテゴリ** :実行
- **定義** :FP\_FLUSH\_TO\_ZERO は、0 に近い結果が FTZ モードでゼロにフラッシュされる回数をカウントする。1 つの結果または 2 つの結果をゼロにフラッシュする並列 FP 演算は、イベント・カウントを 1 だけインクリメントする (たとえ 2 つの結果がゼロにフラッシュされた場合でも)。
- **イベント・コード** :0x0B、**Umask** :無視、**PMC/PMD** :4, 5, 6, 7、**最大インクリメント / サイクル** :2
- **条件** : 命令アドレス範囲 :あり、オペコード・マッチ機能 :有効、データ・アドレス範囲 :なし

#### FP\_SIR\_FLUSH

- **タイトル** :FP SIR フラッシュ、**カテゴリ** :実行
- **定義** :FP\_SIR\_FLUSH は、SIR(Safe Instruction Recognition) フラッシュの発生回数をカウントする。
- **イベント・コード** :0x0C、**Umask** :無視、**PMC/PMD** :4, 5, 6, 7、**最大インクリメント / サイクル** :2
- **条件** : 命令アドレス範囲 :あり、オペコード・マッチ機能 :有効、データ・アドレス範囲 :なし

#### IA32\_INST\_RETIRED

- **タイトル** :リタイアした IA-32 命令、**カテゴリ** :システム
- **定義** :IA32\_INST\_RETIRED は、リタイアした IA-32 命令の数をカウントする。
- **イベント・コード** :0x15、**Umask** :無視、**PMC/PMD** :4, 5, 6, 7、**最大インクリメント / サイクル** :2
- **条件** : 命令アドレス範囲 :なし、オペコード・マッチ機能 :無効、データ・アドレス範囲 :なし

#### IA64\_INST\_RETIRED

- **タイトル** :リタイアした IA-64 命令、**カテゴリ** :実行
- **定義** :IA64\_INST\_RETIRED は、リタイアした IA-64 命令をすべてカウントする。このカウントには、プレディケート・オンの命令、プレディケート・オフの命令、NOP が含まれる。ただし、ハードウェアにより挿入される RSE 操作は含まれない。このイベントは、マスクがゼロの IA64\_TAGGED\_INST\_RETIRED と等しい。
- **イベント・コード** :0x08、**Umask** :0000、**PMC/PMD** :4, 5、**最大インクリメント / サイクル** :6

- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : なし

### IA64\_TAGGED\_INST\_RETIRED

- **タイトル** : リタイアしたタグ付き IA-64 命令、**カテゴリ** : 実行
- **定義** : IA64\_TAGGED\_INST\_RETIRED は IA64\_INST\_RETIRED と似ている。異なるのは、IBR レジスタおよび PMC レジスタ内の命令アドレス範囲とオペコード・マッチ機能の設定により、イベントの選択をさらに制限している点である。
- **イベント・コード** : 0x08、**Umask** : 以下を参照、**PMC/PMC** : 4, 5、**最大インクリメント / サイクル** : 6
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : なし

表 7-30 に示すユニット・マスク TAG\_SELECT は、オペコード・マッチ・レジスタ PMC<sub>8</sub> または PMC<sub>9</sub> を用いて

IA64\_TAGGED\_INST\_RETIRED のイベント・カウント数をつねに修飾している。PMC<sub>8</sub> の設定値はダウストリームイベント・モニタすべてを修飾するので注意すること。オペコード・マッチ機能の設定値とは無関係に他の監視対象イベントがカウントされるようにするには、PMC<sub>8</sub> のすべての mifb ビットおよびすべてのマスク・ビットを 1 にセットしなければならない (すなわちすべてのオペコードがマッチする)。PMC<sub>9</sub> の設定値は、他のイベント・モニタには影響しない。

また umask 0011 は、適切なオペコード・マッチ機能によってマッチングされる命令をカウントするだけでなく、アーキテクチャ上は見えない RSE フィル/スピルについても、それを引き起こす親命令 (alloc や br.ret など) が PMC<sub>8</sub> 内の組み合わせによりマッチングされるときにはカウントするという点異なるので注意すること。このように、オペコード・マッチ機能として PMC<sub>8</sub> を使うか PMC<sub>9</sub> を使うかによって取得されるカウント数に違いの出るのは、RSE の実行数が異なることによる。

**表 7-30. オペコード・マッチによるリタイアしたイベントの選択**

TAG_SELECT	PMC.umask {19:16}	説明
PMC <sub>8</sub> タグ	0011	オペコード・マッチャー PMC <sub>8</sub> にタグを付けられた命令
PMC <sub>9</sub> タグ	0010	オペコード・マッチャー PMC <sub>9</sub> にタグを付けられた命令
すべて	0000	リタイアした命令すべて (タグの有無とは無関係)
未定義	その他のすべての umask の設定	未定義のイベント・カウント

### INST\_ACCESS\_CYCLE

- **タイトル** : 命令アクセス・サイクル、**カテゴリ** : ストール

- **定義** : INST\_ACCESS\_CYCLE は、バックエンドのストールもフラッシュも発生せず、デカップリング・バッファが空である場合に、L1I または ITLB のミスによりフロントエンドがストールしてデータ待機状態にあるサイクル数をカウントする。
- **イベント・コード** : 0x01、**Umask** : 無視、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : なし

#### INST\_DISPERSED

- **タイトル** : 配布された命令、**カテゴリ** : 命令の発行
- **定義** : INST\_DISPERSED は、マシンのフロントエンドからバックエンドへ発行された命令 (nop を含む) の数をカウントする。このカウントには、正しくない実行パス上、すなわち、分岐予測ミス・フラッシュや他のバックエンド・フラッシュのシャドウ内での命令ディスパースルが含まれる。
- **イベント・コード** : 0x2D、**Umask** : 無視、**PMC/PMD** : 4, 5、**最大インクリメント / サイクル** : 6
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 無効、データ・アドレス範囲 : なし

#### INST\_FAILED\_CHKS\_RETIRED.ALL

- **タイトル** : 失敗したスペキュレーティブ・チェック・ロード、**カテゴリ** : 実行
- **定義** : INST\_FAILED\_CHKS\_RETIRED.ALL は、失敗したスペキュレーティブ・チェック・ロード命令 (chk.s) の数をカウントする。このカウントには、プレディケート・オフの chk.s 命令は含まれないが、整数および FP のこの命令の変形は含まれる。
- **イベント・コード** : 0x35、**Umask** : xx11、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : なし

#### INST\_FAILED\_CHKS\_RETIRED.FP

- **タイトル** : 失敗したスペキュレーティブ FP チェック・ロード、**カテゴリ** : 実行
- **定義** : INST\_FAILED\_CHKS\_RETIRED.FP は、失敗したスペキュレーティブ・チェック・ロード命令 (chk.s) の数をカウントする。このカウントには、プレディケート・オフの chk.s 命令は含まれず、この命令の FP の変形のみが含まれる。
- **イベント・コード** : 0x35、**Umask** : xx10、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : なし



## INST\_FAILED\_CHKS\_RETIRED.INTEGER

- **タイトル** : 失敗したスペキュレーティブ整数チェック・ロード、  
**カテゴリ** : 実行
- **定義** : INST\_FAILED\_CHKS\_RETIRED.INTEGER は、失敗したスペキュレーティブ・チェック・ロード命令 (chk.s) の数をカウントする。このカウントには、プレディケート・オフの chk.s 命令は含まれず、この命令の整数の変形のみが含まれる。
- **イベント・コード** : 0x35、**Umask** : xx01、**PMC/PMD** : 4, 5, 6, 7、  
**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : なし

## INSTRUCTION\_EAR\_EVENTS

- **タイトル** : 命令 EAR イベント、**カテゴリ** : 命令キャッシュ
- **定義** : INSTRUCTION\_EAR\_EVENTS は、EAR が L1I および ITLB イベントを収集した数をカウントする。
- **イベント・コード** : 0x23、**Umask** : 無視、**PMC/PMD** : 4, 5, 6, 7、  
**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 無効、データ・アドレス範囲 : なし

## ISA\_TRANSITIONS

- **タイトル** : IA-64 から IA-32 への ISA の遷移、**カテゴリ** : システム
- **定義** : ISA\_TRANSITIONS は、IA-64 から IA-32 へ命令セットを変換した回数をカウントする。これは、PSR.is ビットを 0 から 1 へ切り替えた回数の中で、br.ia または rfi から IA-32 コードへ変換したことに伴う動作である。
- **イベント・コード** : 0x14、**Umask** : 無視、**PMC/PMD** : 4, 5, 6, 7、  
**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : なし、オペコード・マッチ機能 : 無効、データ・アドレス範囲 : なし

## ISB\_LINES\_IN

- **タイトル** : 命令ストリーミング・バッファ・ライン入力、**カテゴリ** : 命令キャッシュ
- **定義** : ISB\_LINES\_IN は、命令デマンド・ミスおよび命令プリフェッチ要求の結果として、L2(およびそれ以降) から命令ストリーミング・バッファへ書き込まれた 32 バイトの L1I キャッシュ・ラインの数をカウントする。
- **イベント・コード** : 0x26、**Umask** : 無視、**PMC/PMD** : 4, 5, 6, 7、  
**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : なし、オペコード・マッチ機能 : 無効、データ・アドレス範囲 : なし

**ITLB\_INSERTS\_HPW**

- **タイトル** :ハードウェア・ページ・ウォークによる ITLB への挿入、  
**カテゴリ** :システム
- **定義** :ITLB\_INSERTS\_HPW は、ハードウェア・ページ・テーブル・ウォークによる ITLB への挿入回数をカウントする。
- **イベント・コード** :0x28、**Umask** :無視、**PMC/PMD** :4, 5, 6, 7、  
**最大インクリメント / サイクル** :1
- **条件** :命令アドレス範囲 :あり、オペコード・マッチ機能 :無効、データ・アドレス範囲 :なし

**ITLB\_MISSES\_FETCH**

- **タイトル** :ITLB デマンド・ミス、**カテゴリ** :システム
- **定義** :ITLB\_MISSES\_FETCH は、デマンド ITLB ミスの回数をカウントする。
- **イベント・コード** :0x27、**Umask** :無視、**PMC/PMD** :4, 5, 6, 7、  
**最大インクリメント / サイクル** :1
- **条件** :命令アドレス範囲 :あり、オペコード・マッチ機能 :無効、データ・アドレス範囲 :なし

**L1D\_READ\_FORCED\_MISSES\_RETIRED**

- **タイトル** :L1 データ・キャッシュの強制ロード・ミス、**カテゴリ** :L1 データ・キャッシュ
- **定義** :L1D\_READ\_FORCED\_MISSES\_RETIRED は、メモリ順序付けの制約、予測された L1 データのキャッシュ・ミス、ストア・バッファ・ヒット、レジスタ・ファイルへの L2 データの同時リターンなどによって、L1 データ・キャッシュのミスを強制されたロードの数をカウントする。
- **イベント・コード** :0x6B、**Umask** :無視、**PMC/PMD** :4, 5, 6, 7、  
**最大インクリメント / サイクル** :2
- **条件** :命令アドレス範囲 :あり、オペコード・マッチ機能 :有効、データ・アドレス範囲 :あり

**L1D\_READ\_MISSES\_RETIRED**

- **タイトル** :L1 データ・キャッシュ読み取りミス、**カテゴリ** :L1 データ・キャッシュ
- **定義** :L1D\_READ\_MISSES\_RETIRED は、コミットされた L1 データ・キャッシュの読み取りミスの回数をカウントする。このカウントには、L1 データ・キャッシュ (詳しいリストは L1D\_READS\_RETIRED イベントを参照) によって処理することができたにもかかわらず、キャッシュをミスした読み取り参照が含まれる。イベント・カウントには、正しくないミスが含まれる。L1 データ・キャッシュはライトスルーであるため、書き込みミスはカウントされない。
- **イベント・コード** :0x66、**Umask** :無視、**PMC/PMD** :4, 5, 6, 7、  
**最大インクリメント / サイクル** :2



- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : あり

### L1D\_READS\_RETIRED

- **タイトル** : L1 データ・キャッシュ読み取り、**カテゴリ** : L1 データ・キャッシュ
- **定義** : L1D\_READS\_RETIRED は、コミットされた L1 データ・キャッシュの読み取り (整数および RSE 参照) の数をカウントする。このカウントに含まれないのは、VHPT ロード、チェック・ロード、L1 ヒント・ロード、セマフォ、キャッシュ不可ロードおよび FP ロードである。プレディケート・オフのロードも含まれないが、分岐パスの予測が外れた操作はカウントに含まれる。
- **イベント・コード** : 0x64、**Umask** : 無視、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 2
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : あり

### L1I\_DEMAND\_READS

- **タイトル** : L1I および ISB の命令デマンド・ルックアップ、**カテゴリ** : 命令キャッシュ
- **定義** : L1I\_DEMAND\_READS は、ヒット / ミスの結果とは関係なく、32 バイトの命令デマンド L1I/ISB ルックアップの数をカウントする。
- **イベント・コード** : 0x20、**Umask** : 無視、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 無効、データ・アドレス範囲 : なし  
命令アドレス範囲に基づく条件付けは、不正確になることがある。

### L1I\_FILLS

- **タイトル** : L1 命令キャッシュ・フィル、**カテゴリ** : 命令キャッシュ
- **定義** : L1I\_FILLS は、命令ストリーミング・バッファから L1 命令キャッシュに移動された 32 バイトのラインの数をカウントする。
- **イベント・コード** : 0x21、**Umask** : 無視、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : なし、オペコード・マッチ機能 : 無効、データ・アドレス範囲 : なし

### L1I\_PREFETCH\_READS

- **タイトル** : L1I および ISB の命令プリフェッチ・ルックアップ、**カテゴリ** : 命令キャッシュ
- **定義** : L1I\_PREFETCH\_READS は、ヒット / ミスの結果とは関係なく、64 バイトの L1I/ISB の命令プリフェッチ・ルックアップの数をカウントする。

- **イベント・コード** :0x24、**Umask** :無視、**PMC/PMD** :4, 5, 6, 7、**最大インクリメント / サイクル** :1
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 無効、データ・アドレス範囲 : なし

#### **L2\_DATA\_REFERENCES.ALL**

- **タイトル** :L2 データの読み取り / 書き込み参照、**カテゴリ** :L2 キャッシュ
- **定義** :L2\_DATA\_REFERENCES.ALL は、L2 データの読み取りおよび書き込みアクセスをすべてカウントする。報告されるカウントは、キャッシュ・ラインがマージされる前の要求の数である。セマフォ操作は、1回の読み取り、および1回の書き込みとしてカウントされる。
- **イベント・コード** :0x69、**Umask** :xx11、**PMC/PMD** :4, 5, 6, 7、**最大インクリメント / サイクル** :2
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : あり

#### **L2\_DATA\_REFERENCES.READS**

- **タイトル** :L2 データ読み取り参照、**カテゴリ** :L2 キャッシュ
- **定義** :L2\_DATA\_REFERENCES.READS は、すべてのL2 データ読み取りアクセスをカウントする。報告されるカウントは、キャッシュ・ラインがマージされる前の要求の数である。セマフォ操作は、1回の読み取りとしてカウントされる。
- **イベント・コード** :0x69、**Umask** :xx01、**PMC/PMD** :4, 5, 6, 7、**最大インクリメント / サイクル** :2
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : あり

#### **L2\_DATA\_REFERENCES.WRITES**

- **タイトル** :L2 データ書き込み参照、**カテゴリ** :L2 キャッシュ
- **定義** :L2\_DATA\_REFERENCES.WRITES は、すべてのL2 データ書き込みアクセスをカウントする。報告されるカウントは、キャッシュ・ラインがマージされる前の要求の数である。セマフォ操作は、1回の書き込みとしてカウントされる。
- **イベント・コード** :0x69、**Umask** :xx10、**PMC/PMD** :4, 5, 6, 7、**最大インクリメント / サイクル** :2
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : あり

#### **L2\_FLUSH\_DETAILS**

- **タイトル** :L2 フラッシュの詳細、**カテゴリ** :L2 キャッシュ

- **定義** : L2\_FLUSH\_DETAILS により、L2 パイプライン・フラッシュの内訳を原因ごとに詳しく調べることができる。このイベントは、4 ビットのユニット・マスク (7-59 ページの表 7-31 を参照) で指定される条件により選択される L2 パイプライン・フラッシュの数をカウントする。ユニット・マスクの 4 つの組み合わせは、すべてサポートされる。
- **イベント・コード** : 0x77、**Umask** : 以下を参照、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : なし、オペコード・マッチ機能 : 無効、データ・アドレス範囲 : なし

表 7-31. L2\_FLUSH\_DETAILS イベントのユニット・マスク・ビット {19:16}

L2 フラッシュの原因	PMC.umask {19:16}	説明
L2_ST_BUFFER_FLUSH	xxx1	次のことを原因とする、L2 のストア間競合 (a) ストア・バッファ・エントリが同じ (b) バック・ツー・バック・ストア
L2_ADDR_CONFLICT	xx1x	ストア後のロードを実行するときの MESI のアップデートが原因でフラッシュされた L2
L2_BUS_REJECT	x1xx	バスの制約を原因とする L2 フラッシュ
L2_FULL_FLUSH	1xxx	次のいずれかが原因でフラッシュされた L2 (a) ストア・バッファが一杯 (b) ロード・ミス・バッファが一杯

## L2\_FLUSHES

- **タイトル** : L2 フラッシュ、**カテゴリ** : L2 キャッシュ
- **定義** : L2\_FLUSHES は、ストア・バッファの競合、アドレスの競合、L3 およびバス・キューが一杯になったなどの理由による、L2 パイプライン・フラッシュの数をカウントする。
- **イベント・コード** : 0x76、**Umask** : 無視、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 無効、データ・アドレス範囲 : なし

## L2\_INST\_DEMAND\_READS

- **タイトル** : L2 命令デマンド・フェッチ要求、**カテゴリ** : 命令キャッシュ
- **定義** : L2\_INST\_DEMAND\_READS は、L1I のデマンド・フェッチ・ミスによる L2 命令要求の数をカウントする。モニタは、L1I および ISB の両方でミスしたデマンド・フェッチ・ルックアップの数をカウントする。この場合、ルックアップが、RAB(Request Address Buffer) 内でヒットまたはミスしたかどうかは関係ない。つまり、このカウントには、要求済みのラインに対するミス (2 次ミス) が含まれる。
- **イベント・コード** : 0x22、**Umask** : 無視、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1

- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 無効、データ・アドレス範囲 : なし

## L2\_INST\_PREFETCH\_READS

- **タイトル** : L2 命令プリフェッチ要求、**カテゴリ** : 命令キャッシュ
- **定義** : L2\_INST\_PREFETCH\_READS は、ユニファイド L2 キャッシュに対して発行された命令プリフェッチ要求をすべてカウントする。
- **イベント・コード** : 0x25、**Umask** : 無視、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 無効、データ・アドレス範囲 : なし

## L2\_MISSES

- **タイトル** : L2 ミス、**カテゴリ** : L2 キャッシュ
- **定義** : L2\_MISSES は、L2 キャッシュ・ミス (キャッシュ不可ページへの要求は除く) の数をカウントする。このカウントには、命令のフェッチおよびプリフェッチ、データの読み取りおよび書き込み操作により引き起こされるミスが含まれる。同じ L2 キャッシュ・ラインに対する 2 次ミスは、個々のミスとしてカウントされる。
- **イベント・コード** : 0x6A、**Umask** : 無視、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 2
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 無効、データ・アドレス範囲 : なし

## L2\_REFERENCES

- **タイトル** : L2 参照、**カテゴリ** : L2 キャッシュ
- **定義** : L2\_REFERENCES は、L2 キャッシュ参照 (キャッシュ不可ページへの要求は除く) の数をカウントする。このカウントには、命令のフェッチおよびプリフェッチ、データの読み取りおよび書き込みによる参照が含まれる。1 サイクル当たりの最大インクリメントは 3 で、その内訳は 1 が命令フェッチ、2 がデータ参照である。
- **イベント・コード** : 0x68、**Umask** : 無視、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 3
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 無効、データ・アドレス範囲 : なし

## L3\_LINES\_REPLACED

- **タイトル** : 置き換えられた L3 キャッシュ・ライン、**カテゴリ** : L3 キャッシュ
- **定義** : L3\_LINES\_REPLACED は、犠牲になっている有効な L3 ラインの数をカウントする。
- **イベント・コード** : 0x7F、**Umask** : 無視、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1



- **条件** : 命令アドレス範囲 : なし、オペコード・マッチ機能 : 無効、データ・アドレス範囲 : なし

### L3\_MISSES

- **タイトル** : L3 ミス、**カテゴリ** : L3 キャッシュ
- **定義** : L3\_MISSES は、L3 ミスの回数をカウントする。このカウントには、命令要求、データ要求、L2 ライン・ライトバックにより引き起こされたミスが含まれる。
- **イベント・コード** : 0x7C、**Umask** : 無視、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : なし、オペコード・マッチ機能 : 無効、データ・アドレス範囲 : なし

### L3\_READS.ALL\_READS.ALL

- **タイトル** : 命令およびデータの L3 読み取り、**カテゴリ** : L3 キャッシュ
- **定義** : L3\_READS.ALL\_READS.ALL は、ストリーム・ソース (命令またはデータ) やヒット / ミスの結果とは関係なく、L3 読み取りアクセスの回数をすべてカウントする。
- **イベント・コード** : 0x7D、**Umask** : 1111、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : なし、オペコード・マッチ機能 : 無効、データ・アドレス範囲 : なし

### L3\_READS.ALL\_READS.HIT

- **タイトル** : 命令およびデータの L3 読み取りヒット、**カテゴリ** : L3 キャッシュ
- **定義** : L3\_READS.ALL\_READS.HIT は、ストリーム・ソース (命令またはデータ) とは関係なく、L3 読み取りヒットの回数をすべてカウントする。
- **イベント・コード** : 0x7D、**Umask** : 1101、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : なし、オペコード・マッチ機能 : 無効、データ・アドレス範囲 : なし

### L3\_READS.ALL\_READS.MISS

- **タイトル** : 命令およびデータの L3 読み取りミス、**カテゴリ** : L3 キャッシュ
- **定義** : L3\_READS.ALL\_READS.MISS は、ストリーム・ソース (命令またはデータ) とは関係なく、L3 読み取りミスの回数をすべてカウントする。
- **イベント・コード** : 0x7D、**Umask** : 1110、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : なし、オペコード・マッチ機能 : 無効、データ・アドレス範囲 : なし

**L3\_READS.DATA\_READS.ALL**

- **タイトル** : データ L3 読み取り、**カテゴリ** : L3 キャッシュ
- **定義** : L3\_READS.DATA\_READS.ALL は、ヒット / ミスの結果とは関係なく、データ L3 読み取りアクセスの回数をカウントする。
- **イベント・コード** : 0x7D、**Umask** : 1011、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : なし、オペコード・マッチ機能 : 無効、データ・アドレス範囲 : なし

**L3\_READS.DATA\_READS.HIT**

- **タイトル** : データ L3 読み取りヒット、**カテゴリ** : L3 キャッシュ
- **定義** : L3\_READS.DATA\_READS.HIT は、データ L3 読み取りヒットの回数をカウントする。
- **イベント・コード** : 0x7D、**Umask** : 1001、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : なし、オペコード・マッチ機能 : 無効、データ・アドレス範囲 : なし

**L3\_READS.DATA\_READS.MISS**

- **タイトル** : データ L3 読み取りミス、**カテゴリ** : L3 キャッシュ
- **定義** : L3\_READS.DATA\_READS.MISS は、データ L3 読み取りミスの回数をカウントする。
- **イベント・コード** : 0x7D、**Umask** : 1010、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : なし、オペコード・マッチ機能 : 無効、データ・アドレス範囲 : なし

**L3\_READS.INST\_READS.ALL**

- **タイトル** : 命令 L3 読み取り、**カテゴリ** : L3 キャッシュ
- **定義** : L3\_READS.INST\_READS.ALL は、ヒット / ミスの結果とは関係なく、命令 L3 読み取りアクセスの回数をカウントする。
- **イベント・コード** : 0x7D、**Umask** : 0111、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : なし、オペコード・マッチ機能 : 無効、データ・アドレス範囲 : なし

**L3\_READS.INST\_READS.HIT**

- **タイトル** : 命令 L3 読み取りヒット、**カテゴリ** : L3 キャッシュ
- **定義** : L3\_READS.INST\_READS.HIT は、命令 L3 読み取りヒットの回数をカウントする。



- **イベント・コード** :0x7D、**Umask** :0101、**PMC/PMD** :4, 5, 6, 7、**最大インクリメント / サイクル** :1

- **条件** : 命令アドレス範囲 : なし、オペコード・マッチ機能 : 無効、データ・アドレス範囲 : なし

### L3\_READS.INST\_READS.MISS

- **タイトル** : 命令 L3 読み取りミス、**カテゴリ** : L3 キャッシュ

- **定義** : L3\_READS.INST\_READS.MISS は、命令 L3 読み取りミスの回数をカウントする。

- **イベント・コード** :0x7D、**Umask** :0110、**PMC/PMD** :4, 5, 6, 7、**最大インクリメント / サイクル** :1

- **条件** : 命令アドレス範囲 : なし、オペコード・マッチ機能 : 無効、データ・アドレス範囲 : なし

### L3\_REFERENCES

- **タイトル** : L3 参照、**カテゴリ** : L3 キャッシュ

- **定義** : L3\_REFERENCES は、L3 キャッシュ参照 ( キャッシュ不可ページへの要求は除く ) の数をカウントする。このカウントには、命令のフェッチおよびプリフェッチ、データの読み取りおよび書き込み、L2 キャッシュ・ラインの最上位ビットのライトバックによる参照が含まれる。

- **イベント・コード** :0x7B、**Umask** :無視、**PMC/PMD** :4, 5, 6, 7、**最大インクリメント / サイクル** :1

- **条件** : 命令アドレス範囲 : なし、オペコード・マッチ機能 : 無効、データ・アドレス範囲 : なし

### L3\_WRITES.ALL\_WRITES.ALL

- **タイトル** : L3 書き込み、**カテゴリ** : L3 キャッシュ

- **定義** : L3\_WRITES.ALL\_WRITES.ALL は、ヒット / ミスの結果とは関係なく、L3 書き込みアクセスの回数をカウントする。このカウントには、データ書き込みアクセスと L2 ライトバック・アクセスがともに含まれる ( ストアを行う L3 Read-for-Ownership 要求も含まれる ) 。

- **イベント・コード** :0x7E、**Umask** :1111、**PMC/PMD** :4, 5, 6, 7、**最大インクリメント / サイクル** :1

- **条件** : 命令アドレス範囲 : なし、オペコード・マッチ機能 : 無効、データ・アドレス範囲 : なし

### L3\_WRITES.ALL\_WRITES.HIT

- **タイトル** : L3 書き込みヒット、**カテゴリ** : L3 キャッシュ

- **定義** : L3\_WRITES.ALL\_WRITES.HIT は、L3 書き込みヒットの回数をカウントする。このカウントには、データ書き込みアクセスと L2 ライトバック・アクセスがともに含まれる ( ストアを行う L3 Read-for-Ownership 要求も含まれる ) 。

- **イベント・コード** :0x7E、**Umask** :1101、**PMC/PMD** :4, 5, 6, 7、**最大インクリメント / サイクル** :1
- **条件** : 命令アドレス範囲 : なし、オペコード・マッチ機能 : 無効、データ・アドレス範囲 : なし

### **L3\_WRITES.ALL\_WRITES.MISS**

- **タイトル** :L3 書き込みミス、**カテゴリ** :L3 キャッシュ
- **定義** : L3\_WRITES.ALL\_WRITES.MISS は、L3 書き込みミスの回数をカウントする。このカウントには、データ書き込みアクセスと L2 ライトバック・アクセスがともに含まれる (ストアを行う L3 Read-for-Ownership 要求も含まれる)。
- **イベント・コード** :0x7E、**Umask** :1110、**PMC/PMD** :4, 5, 6, 7、**最大インクリメント / サイクル** :1
- **条件** : 命令アドレス範囲 : なし、オペコード・マッチ機能 : 無効、データ・アドレス範囲 : なし

### **L3\_WRITES.L2\_WRITEBACK.ALL**

- **タイトル** :L3 ライトバック、**カテゴリ** :L3 キャッシュ
- **定義** : L3\_WRITES.L2\_WRITEBACK.ALL は、ヒット / ミスの結果とは関係なく、L2 ライトバックが原因で生じる L3 書き込みアクセスの回数をカウントする。
- **イベント・コード** :0x7E、**Umask** :1011、**PMC/PMD** :4, 5, 6, 7、**最大インクリメント / サイクル** :1
- **条件** : 命令アドレス範囲 : なし、オペコード・マッチ機能 : 無効、データ・アドレス範囲 : なし

### **L3\_WRITES.L2\_WRITEBACK.HIT**

- **タイトル** :L3 ライトバック・ヒット、**カテゴリ** :L3 キャッシュ
- **定義** : L3\_WRITES.L2\_WRITEBACK.HIT は、L2 ライトバックが原因で生じる L3 書き込みヒットの回数をカウントする。
- **イベント・コード** :0x7E、**Umask** :1001、**PMC/PMD** :4, 5, 6, 7、**最大インクリメント / サイクル** :1
- **条件** : 命令アドレス範囲 : なし、オペコード・マッチ機能 : 無効、データ・アドレス範囲 : なし

### **L3\_WRITES.L2\_WRITEBACK.MISS**

- **タイトル** :L3 ライトバック・ミス、**カテゴリ** :L3 キャッシュ
- **定義** : L3\_WRITES.L2\_WRITEBACK.MISS は、L2 ライトバックが原因で生じる L3 書き込みミスの回数をカウントする。
- **イベント・コード** :0x7E、**Umask** :1010、**PMC/PMD** :4, 5, 6, 7、**最大インクリメント / サイクル** :1



- **条件** : 命令アドレス範囲 : なし、オペコード・マッチ機能 : 無効、データ・アドレス範囲 : なし

### L3\_WRITES.DATA\_WRITES.ALL

- **タイトル** : L3 データ書き込み、**カテゴリ** : L3 キャッシュ
- **定義** : L3\_WRITES.DATA\_WRITES.ALL は、ヒット / ミスの結果とは関係なく、L3 データ書き込みアクセスの回数をカウントする。
- **イベント・コード** : 0x7E、**Umask** : 0111、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : なし、オペコード・マッチ機能 : 無効、データ・アドレス範囲 : なし

### L3\_WRITES.DATA\_WRITES.HIT

- **タイトル** : L3 データ書き込みヒット、**カテゴリ** : L3 キャッシュ
- **定義** : L3\_WRITES.DATA\_WRITES.HIT は、L3 データ書き込みヒットの回数をカウントする。
- **イベント・コード** : 0x7E、**Umask** : 0101、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : なし、オペコード・マッチ機能 : 無効、データ・アドレス範囲 : なし

### L3\_WRITES.DATA\_WRITES.MISS

- **タイトル** : L3 データ書き込みミス、**カテゴリ** : L3 キャッシュ
- **定義** : L3\_WRITES.DATA\_WRITES.MISS は、L3 データ書き込みミスの回数をカウントする。
- **イベント・コード** : 0x7E、**Umask** : 0110、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : なし、オペコード・マッチ機能 : 無効、データ・アドレス範囲 : なし

### LOADS\_RETIRED

- **タイトル** : リタイアしたロード、**カテゴリ** : メモリ
- **定義** : LOADS\_RETIRED は、リタイアしたロードの数をカウントする。このカウントには、整数、FP、RSE、VHPT、キャッシュ不可ロード、および失敗したチェック・ロード (ld.c) が含まれる。ALAT 内でヒットしたチェック・ロードや、プレディケート・オフの操作はカウントされない。
- **イベント・コード** : 0x6C、**Umask** : 無視、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 2
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : あり

**MEMORY\_CYCLE**

- **タイトル** : メモリ・ストール・サイクルの合計、**カテゴリ** : ストール
- **定義** : MEMORY\_CYCLE は、キャッシュ・ミス、L1D ウェイ予測ミス、DTC ミス、RSE トラフィックが発生して命令がデータ待ちをしているためにパイプラインがストールあるいはフラッシュされている最中のサイクル数をカウントする。
- **イベント・コード** : 0x07、**Umask** : 無視、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : なし、オペコード・マッチ機能 : 無効、データ・アドレス範囲 : なし

**MISALIGNED\_LOADS\_RETIRED**

- **タイトル** : リタイアしたミスアライン・ロード命令、**カテゴリ** : メモリ
- **定義** : MISALIGNED\_LOADS\_RETIRED は、ハードウェアによって処理された、リタイアしたミスアライン・ロードの数をカウントする。このカウントには、整数、FP、および失敗したチェック・ロード (ld.c) が含まれる。ALAT 内でヒットしたチェック・ロードや、プレディケート・オフの操作はカウントされない。
- **イベント・コード** : 0x70、**Umask** : 無視、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 2
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : あり

**MISALIGNED\_STORES\_RETIRED**

- **タイトル** : リタイアしたミスアライン・ストア命令、**カテゴリ** : メモリ
- **定義** : MISALIGNED\_STORES\_RETIRED は、ハードウェアによって処理された、リタイアしたミスアライン・ストア命令の数をカウントする。このカウントには、整数、FP、キャッシュ不可ストアが含まれる。プレディケート・オフの操作はカウントされない。
- **イベント・コード** : 0x71、**Umask** : 無視、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 2
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : あり

**NOPS\_RETIRED**

- **タイトル** : リタイアした nop 命令、**カテゴリ** : 実行
- **定義** : NOPS\_RETIRED は、リタイアした nop.i、nop.m、または nop.b 命令の数をカウントする。このカウントには、プレディケート・オフの nop 操作は含まれない。
- **イベント・コード** : 0x30、**Umask** : 無視、**PMC/PMD** : 4, 5、**最大インクリメント / サイクル** : 6
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : なし



## PIPELINE\_ALL\_FLUSH\_CYCLE

- **タイトル** : フロントエンドかバックエンドのソースによるパイプライン・フラッシュ・サイクルの合計、**カテゴリ** : ストール
- **定義** : PIPELINE\_ALL\_FLUSH\_CYCLE は、ある一定のサイクルの間、(的中した成立した分岐による)パイプラインのフロントエンド・リステア中に費やされるサイクル数が、(分岐予測ミス、ALATフラッシュ、例外/シリアル化フラッシュのいずれかによる)一定のバックエンド・リステア中に費やされるサイクル数をカウントする。このモニタは、DTCフラッシュ、ウェイ予測ミス、浮動小数点フラッシュについてはカウントしない。
- **イベント・コード** : 0x04、**Umask** : 無視、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : なし、オペコード・マッチ機能 : 無効、データ・アドレス範囲 : なし

## PIPELINE\_BACKEND\_FLUSH\_CYCLE

- **タイトル** : 分岐予測ミスか例外に起因するパイプライン・フラッシュ・サイクルの合計、**カテゴリ** : ストール
- **定義** : PIPELINE\_BACKEND\_FLUSH\_CYCLE は、(分岐予測ミス、ALATフラッシュ、例外/シリアル化フラッシュのいずれかによる)パイプラインのバックエンド・リステア中に費やされるサイクル数をカウントする。このモニタは、DTCフラッシュ、ウェイ予測ミス、浮動小数点フラッシュについてはカウントしない。
- **イベント・コード** : 0x00、**Umask** : 無視、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : なし、オペコード・マッチ機能 : 無効、データ・アドレス範囲 : なし

## PIPELINE\_FLUSH

- **タイトル** : パイプライン・フラッシュ、**カテゴリ** : システム
- **定義** : PIPELINE\_FLUSH は、IEU バイパスの競合(変数シフトなどの非ユニット・レイテンシ MMX 操作により生じる)、データ変換キャッシュ・ミス、L1 データ・キャッシュ・ウェイの予測ミスが原因で Itanium プロセッサのパイプラインがフラッシュされた回数をカウントする。L1 データ・キャッシュ・ウェイの予測ミス、例外フラッシュや命令のシリアル化などが原因となる場合もある。umask の設定によっては、フラッシュの生じる理由が複数に渡ることもある。分岐予測ミスによるフラッシュは含まれない。
- **イベント・コード** : 0x33、**Umask** : 以下を参照、**PMC/PMD** : 4, 5, 6, 7、**最大インクリメント / サイクル** : 1
- **条件** : 命令アドレス範囲 : なし、オペコード・マッチ機能 : 無効、データ・アドレス範囲 : なし

表 7-32. PIPELINE\_FLUSH イベントのユニット・マスク・ビット {19:18}

FLUSH_TYPE	PMC.umask {19:16}	説明
IEU_FLUSH	1xxx	IEU バイパスによるフラッシュ
DTC_FLUSH	x1xx	データ変換キャッシュ・ミスによるフラッシュ
L1D_WAYMP_FLUSH	xx1x	L1 ウェイ予測ミスによるフラッシュ
OTHER_FLUSH	xxx1	フラッシュ発生のおもな原因：例外フラッシュまたは命令のシリアル化

### PREDICATE\_SQUASHED\_RETIRED

- **タイトル** : プレディケート・オフが原因で実行されなかった命令、  
**カテゴリ** : 実行
- **定義** : PREDICATE\_SQUASHED\_RETIRED は、偽のプレディケート修飾のために実行されなかった命令の数をカウントする。このカウントには、nop.b を除いたすべてのプレディケート・オフの nop が含まれる。プレディケート・オフの B シラブル (nop.b を含む) はカウントされない。
- **イベント・コード** : 0x31、**Umask** : 無視、**PMC/PMD** : 4、5、  
**最大インクリメント / サイクル** : 6
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : なし

### RSE\_LOADS\_RETIRED

- **タイトル** : RSE ロード・アクセス、**カテゴリ** : 実行
- **定義** : RSE\_LOADS\_RETIRED は、リタイアした RSE ロードの数をカウントする。
- **イベント・コード** : 0x72、**Umask** : 無視、**PMC/PMD** : 4、5、6、7、  
**最大インクリメント / サイクル** : 2
- **条件** : 7-68 ページの RSE\_REFERENCES\_RETIRED を参照

### RSE\_REFERENCES\_RETIRED

- **タイトル** : RSE アクセス、**カテゴリ** : 実行
- **定義** : RSE\_REFERENCES\_RETIRED は、リタイアした RSE ロードおよびストアの数をカウントする。
- **イベント・コード** : 0x65、**Umask** : 無視、**PMC/PMD** : 4、5、6、7、  
**最大インクリメント / サイクル** : 2
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : あり

alloc、loadrs、flushrs、分岐リターンのおいずれか、あるいは RSE 参照を引き起こした rfi が、命令アドレス範囲またはオペコード・マッチャーによってタグを付けられた場合は、RSE のロードおよびストアはタグ付きであると見なされる。データ・アドレス範囲チェックの場合、RSE 参照にタグが付くのは、プログラムされた DBR 範囲にヒットした場合だけに限られる。

## STORES\_RETIRED

- **タイトル** :リタイアしたストア、**カテゴリ** :メモリ
- **定義** : STORES\_RETIRED は、リタイアしたストアの数をカウントする。このカウントには、整数、FP、RSE、およびキャッシュ不可ストアが含まれる。プレディケート・オフの操作はカウントされない。
- **イベント・コード** :0x6D、**Umask** :無視、**PMC/PMD** :4, 5, 6, 7、**最大インクリメント / サイクル** :2
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : あり

## UC\_LOADS\_RETIRED

- **タイトル** :リタイアしたキャッシュ不可ロード、**カテゴリ** :メモリ
- **定義** : UC\_LOADS\_RETIRED は、リタイアしたキャッシュ不可ロードまたはライト・コーレシング・ロードの数をカウントする。このカウントには、整数、FP、RSE、VHPT の各ロードと、失敗したチェック・ロード (ld.c) が含まれる。ALAT にヒットしたチェック・ロードはカウントされない。プレディケート・オフの操作はカウントされない。
- **イベント・コード** :0x6E、**Umask** :無視、**PMC/PMD** :4, 5, 6, 7、**最大インクリメント / サイクル** :2
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : あり

## UC\_STORES\_RETIRED

- **タイトル** :リタイアしたキャッシュ不可ストア、**カテゴリ** :メモリ
- **定義** : UC\_STORES\_RETIRED は、リタイアしたキャッシュ不可ストアまたはライト・コーレシング・ストアの数をカウントする。このカウントには、整数、FP、RSE、キャッシュ不可のストアが含まれる。プレディケート・オフの操作はカウントされない。
- **イベント・コード** :0x6F、**Umask** :無視、**PMC/PMD** :4, 5, 6, 7、**最大インクリメント / サイクル** :2
- **条件** : 命令アドレス範囲 : あり、オペコード・マッチ機能 : 有効、データ・アドレス範囲 : あり

## UNSTALLED\_BACKEND\_CYCLE

- **タイトル** :ストールしないバックエンド・サイクル、**カテゴリ** :ストール
- **定義** : UNSTALLED\_BACKEND\_CYCLE は、バックエンドが遅延せずに命令を処理し、フロントエンドに対する影響がパイプラインのバックエンドに伝播されるように、フロントエンドとバックエンドの間のデカップリング・バッファが空の時のサイクル数をカウントする。したがって、このモニタでは、L1I および ITLB にヒットするかミスするかに関係なく、バックエンドでストールもフラッシュも発生せず、なおかつデカップリング・バッファが空の状態にあるときのサイクル数がカウントされる。

- **イベント・コード** :0x05、**Umask** :無視、**PMC/PMD** :4, 5, 6, 7、**最大インクリメント / サイクル** :1
- **条件** : 命令アドレス範囲 : なし、オペコード・マッチ機能 : 無効、データ・アドレス範囲 : なし