



インテル® Itanium® 2 プロセッサ リファレンス・マニュアル

ソフトウェアの開発と最適化

2002 年 6 月

© 1999-2003 Intel Corporation
無断での引用、転載を禁じます。
資料番号 : 251110J-001

Web: www.intel.co.jp/jp/developer/ (日本語)
<http://developer.intel.com> (英語)

【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は責任を負いません。また、外部回路の特許についても関知いたしません。
- ・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写することは禁じられています。

インテル、Itanium、Pentium、VTune は、アメリカ合衆国およびその他の国における Intel Corporation またはその子会社の商標または登録商標です。

© 2003 Intel Corporation. 無断での引用、転載を禁じます。

* 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。

目次

第 1 章	本書について	1
	1.1 概要	1
	1.2 構成	1
	1.3 用語	2
	1.4 参考文献	2
	1.5 改訂履歴	2
第 2 章	Itanium® 2 プロセッサ の拡張	3
	2.1 サポートしている命令	3
	2.2 機能ユニットと発行規則	3
	2.3 操作のレイテンシ	3
	2.4 データ操作	4
	2.4.1 データ・スペキュレーションと ALAT	4
	2.4.2 データ・アライメント	4
	2.4.3 コントロール・スペキュレーション	6
	2.5 メモリ階層	6
	2.6 分岐予測	8
	2.7 命令プリフェッチ	8
第 3 章	機能ユニットと発行規則	9
	3.1 実行モデル	9
	3.2 機能ユニットの数とタイプ	9
	3.3 命令スロットと機能ユニットの対応関係	10
	3.3.1 実行幅	12
	3.3.2 ディスパーサル規則	13
	3.3.3 スプリット発行とバンドル・タイプ	15
第 4 章	レイテンシとバイパス	17
	4.1 コントロール・スペキュレーションとデータ・スペキュレーションのペナルティ	17
	4.2 分岐に関連するレイテンシとペナルティ	17
第 5 章	データ操作	19
	5.1 データ・スペキュレーションと ALAT	19
	5.1.1 割り当て / 置換ポリシー	20
	5.1.2 規則と特殊な場合	20
	5.2 スペキュレーティブ・ロード / ストアとプレディケート付きロード / ストア	20
	5.3 浮動小数点ロード	21
	5.4 データ・キャッシュのプリフェッチ機能とロード・ヒント	22
	5.4.1 lfetch のサポート	22
	5.4.2 ロードのテンポラルな局所性を制御するコンプリータ	22
	5.5 データ・アライメント	24
	5.6 ライト・コアリング	24
	5.6.1 WC バッファの排出条件	25
	5.6.2 WC バッファのフラッシュ動作	25
	5.7 レジスタ・スタック・エンジン	26
第 6 章	メモリ・サブシステム	27
	6.1 トランスレーション・ルックアサイド・バッファ	28
	6.1.1 命令 TLB	28
	6.1.2 データ TLB	28

6.2	ハードウェア・ページ・ウォーク	29
6.3	キャッシュのまとめ	30
6.4	1次命令キャッシュ	30
6.5	命令ストリーム・バッファ	31
6.6	1次データ・キャッシュ	31
6.6.1	L1D ロード	32
6.6.2	L1D ストア	32
6.6.3	ロードとストアの留意点	32
6.6.4	L1D ミス	34
6.7	2次ユニファイド・キャッシュ	34
6.7.1	L2 に対する L1D 要求	35
6.7.2	L2 OzQ	35
6.7.3	L2 のキャンセル	37
6.7.4	L2 の再循環	39
6.7.5	順序づけ	39
6.7.6	L2 命令プリフェッチ FIFO	39
6.7.7	システム・バス / L3 との相互作用	40
6.8	3次ユニファイド・キャッシュ	41
6.9	システム・バス	41
第 7 章	分岐命令と分岐予測	43
7.1	分岐予測ヒント	44
7.2	間接分岐	44
7.3	完全ループ予測	44
第 8 章	命令プリフェッチ機能	47
8.1	ストリーミング・プリフェッチ	47
8.2	ヒント・プリフェッチ	48
8.3	プリフェッチ・フラッシュ・ヒント	49
8.4	brl 命令	49
第 9 章	Itanium [®] 2 プロセッサ向けの最適化	51
9.1	スケジューリングのヒント	51
9.2	lfetch の最適な使用	51
9.3	データ・ストリーミング	52
9.3.1	浮動小数点データ・ストリーム	52
9.3.2	整数データ・ストリーム	53
9.3.3	ストア・データ・ストリーム	53
9.4	コントロール・スペキュレーションとデータ・スペキュレーション	54
9.5	確認済みの L2 ミス・バンドル配置	54
9.6	確認済みの L2 キャンセル条件および再循環条件の回避	54
9.7	命令バンドリング	54
9.8	分岐	54
9.8.1	1 サイクル分岐	55
9.8.2	完全ループ予測	55
9.8.3	分岐ターゲット	55
第 10 章	パフォーマンス監視機能	57
10.1	概要	57
10.2	パフォーマンス・モニタのプログラミング・モデル	57
10.2.1	作業負荷の特性評価	58
10.2.2	プロファイリング	61
10.2.3	監視対象となるイベントの制限	63

	10.2.4 参考資料	68
10.3	パフォーマンス・モニタのステート	69
	10.3.1 パフォーマンス・モニタの制御とアクセス	70
	10.3.2 パフォーマンス・カウンタ・レジスタ	72
	10.3.3 パフォーマンス・モニタ・オーバーフロー・ステータス・レジスタ (PMC0,1,2,3)	74
	10.3.4 オペコード・マッチ・チェック (PMC8,9,15)	74
	10.3.5 命令アドレス範囲マッチング	77
	10.3.6 データ・アドレス範囲マッチング (PMC13)	79
	10.3.7 イベント・アドレス・レジスタ (PMC10,11/PMD0,1,2,3,17)	80
	10.3.8 データ EAR (PMC11、PMD2,3,17)	83
	10.3.9 分岐トレース・バッファ	87
	10.3.10 割り込み	91
	10.3.11 プロセッサ・リセット、PAL コール、および低消費電力状態	92
第 11 章	パフォーマンス監視イベント	93
	11.1 概要	93
	11.2 イベントのカテゴリ化	93
	11.3 基本イベント	94
	11.4 命令ディスパースル・イベント	95
	11.5 命令実行イベント	95
	11.6 ストール・イベント	97
	11.7 分岐イベント	98
	11.8 メモリ階層	99
	11.8.1 L1 命令キャッシュと命令プリフェッチ・イベント	101
	11.8.2 L1 データ・キャッシュ・イベント	102
	11.8.3 L2 ユニファイド・キャッシュ・イベント	104
	11.8.4 L3 キャッシュ・イベント	108
	11.9 システム・イベント	109
	11.10 TLB イベント	109
	11.11 システム・バス・イベント	111
	11.12 RSE イベント	115
	11.13 イベント・コードによって並べ替えられるパフォーマンス・モニタ	116
	11.14 パフォーマンス監視イベント・リスト	122
第 12 章	モデル固有の機能とオプションの機能	175
	12.1 メモリ属性	175
	12.2 ptc.e のページ動作	175
	12.3 CPUID の戻り値	175
	12.3.1 Itanium [®] 命令 CPUID の戻り値	175
	12.3.2 IA-32 CPUID の戻り値	176
付録 A	Itanium [®] 2 プロセッサのパイプライン	177
	A.1 コア・パイプライン	177
	A.2 パイプライン・ステージ	177
	A.2.1 IPG STAGE	177
	A.2.2 ROT ステージ	178
	A.2.3 EXP ステージ	178
	A.2.4 REN ステージ	178
	A.2.5 REG ステージ	178
	A.2.6 EXE ステージ	178
	A.2.7 DET ステージ	178
	A.2.8 WRB ステージ	178

A.3	命令バッファ (IB)	179
A.4	マイクロパイプライン	179
A.4.1	FPU マイクロパイプライン	179
A.4.2	L1D マイクロパイプライン	179
A.4.3	L2 マイクロパイプライン	179

図目次

6-1	Itanium [®] 2 プロセッサの 3 レベル・キャッシュ階層	27
10-1	時間ベースのサンプリング	58
10-2	Itanium [®] プロセッサ・ファミリのサイクル・アカウンティング	60
10-3	プログラム・カウンタによるイベント・ヒストグラム	62
10-4	Itanium [®] 2 プロセッサの監視対象となるイベントの制限	64
10-5	Itanium [®] 2 プロセッサの命令タグ付け機構	65
10-6	1 つのプロセスの監視	67
10-7	複数のプロセスの監視	68
10-8	システム全体の監視	68
10-9	Itanium [®] 2 プロセッサのパフォーマンス監視レジスタのモデル	70
10-10	プロセッサ・ステータス・レジスタ (PSR) のパフォーマンス監視用フィールド	71
10-11	Itanium [®] 2 プロセッサの汎用 PMC レジスタ (PMC4,5,6,7)	72
10-12	Itanium [®] 2 プロセッサの汎用 PMD レジスタ (PMD4,5,6,7)	73
10-13	Itanium [®] 2 プロセッサのパフォーマンス・モニタ・オーバーフロー・ ステータス・レジスタ (PMC0,1,2,3)	74
10-14	オペコード・マッチ・レジスタ (PMC8,9)	75
10-15	オペコード・マッチ設定レジスタ (PMC15)	75
10-16	命令アドレス範囲設定レジスタ (PMC14)	77
10-17	メモリ・パイプライン・イベント制限設定レジスタ (PMC13)	80
10-18	命令イベント・アドレス設定レジスタ (PMC10)	81
10-19	命令イベント・アドレス・レジスタのフォーマット (PMD0,1)	81
10-20	データ・イベント・アドレス設定レジスタ (PMC11)	83
10-21	データ・イベント・アドレス・レジスタのフォーマット (PMD2,3,17)	84
10-22	分岐トレース・バッファ設定レジスタ (PMC12)	88
10-23	分岐トレース・バッファ・レジスタのフォーマット (PMD8-15、PMC ₁₂ .ds == 0)	89
10-24	分岐トレース・バッファ・レジスタのフォーマット (PMD8-15、PMC ₁₂ .ds == 1)	89
10-25	分岐トレース・バッファ・インデックス・レジスタのフォーマット (PMD16)	90
11-1	Itanium [®] 2 プロセッサのメモリ階層内のイベント・モニタ	100
A-1	Itanium [®] 2 プロセッサのコア・パイプライン	177

表目次

2-1	Itanium [®] 2 プロセッサ / Itanium [®] プロセッサの操作レイテンシ	5
2-2	L1I キャッシュの相違点	6
2-3	L1D キャッシュの相違点	6
2-4	L2 ユニファイド・キャッシュの相違点	7
2-5	L3 キャッシュの相違点	7
2-6	命令 TLB の相違点	7
2-7	データ TLB の相違点	7
2-8	分岐予測のレイテンシ (サイクル単位)	8

3-1	A タイプの命令とポートの対応関係	11
3-2	I タイプの命令とポートの対応関係	11
3-3	M タイプの命令とポートの対応関係	11
3-4	デュアル発行できるバンドル・タイプ	14
4-1	スペキュレーティブ・ロードのリカバリ・レイテンシ	17
4-2	分岐予測のレイテンシ	17
4-3	バイパス使用時の実行レイテンシのまとめ	18
5-1	ALAT エントリの比較サイズ	19
5-2	コントロール・スペキュレーションのペナルティ	21
5-3	プロセッサ・キャッシュ・ヒント	23
5-4	Itanium [®] 2 プロセッサの WCB 排出条件	25
6-1	Itanium [®] 2 プロセッサによる仮想メモリのサポート	27
6-2	命令 TLB とデータ TLB の主な機能	28
6-3	最良の場合の HPW ペナルティ	29
6-4	キャッシュのまとめ	30
6-5	ストアからロードへのフォワーディングのペナルティ	33
6-6	L2 の発行の優先度	40
6-7	システム・バス / L3 への要求と最終的な L2 ステート	40
7-1	分岐予測のレイテンシ	43
8-1	ストリーミング・プリフェッチ動作のまとめ	48
8-2	プリフェッチ機構	48
10-1	要求 1 回当たりの平均レイテンシと 1 サイクル当たりの要求数の計算例	59
10-2	Itanium [®] 2 プロセッサの EAR と分岐トレース・バッファ	63
10-3	Itanium [®] 2 プロセッサのイベント制限モード	66
10-4	Itanium [®] 2 プロセッサのパフォーマンス監視レジスタ・セット	69
10-5	パフォーマンス・モニタ設定 (PMC) レジスタの制御フィールド (PMC4,5,6,7,10,11,12)	71
10-6	Itanium [®] 2 プロセッサの汎用 PMC レジスタ・フィールド (PMC4,5,6,7)	72
10-7	Itanium [®] 2 プロセッサの汎用 PMD レジスタのフィールド	73
10-8	Itanium [®] 2 プロセッサのパフォーマンス・モニタ・オーバーフロー・ レジスタのフィールド (PMC0,1,2,3)	74
10-9	オペコード・マッチ・レジスタのフィールド (PMC8,9)	75
10-10	オペコード・マッチ設定レジスタのフィールド (PMC15)	76
10-11	命令セットによる Itanium [®] 2 プロセッサの命令アドレス範囲チェック	77
10-12	命令アドレス範囲設定レジスタのフィールド (PMC14)	78
10-13	メモリ・パイプライン・イベント制限のフィールド (PMC13)	79
10-14	命令イベント・アドレス設定レジスタのフィールド (PMC10)	81
10-15	キャッシュ・モード (PMC10.ct=1x) の命令 EAR (PMC10) の umask フィールド	82
10-16	キャッシュ・モード (PMC10.ct=1x) の命令 EAR (PMD0,1)	82
10-17	TLB モード (PMC10.ct=00) の命令 EAR (PMC10) の umask フィールド	83
10-18	TLB モード (PMC10.ct=00) の命令 EAR (PMD0,1)	83
10-19	データ・イベント・アドレス設定レジスタのフィールド (PMC11)	84
10-20	データ・キャッシュ・モード (PMC11.mode=00) のデータ EAR (PMC11) の Umask フィールド	84
10-21	データ・キャッシュ・ロード・ミス・モード (PMC11.mode=00) の PMD2,3,17 のフィールド	85
10-22	TLB モード (PMC10.ct=01) のデータ EAR (PMC11) の Umask フィールド	86
10-23	TLB ミス・モード (PMC11.mode=01) の PMD2,3,17 のフィールド	86
10-24	ALAT ミス・モード (PMC11.mode=1x) の PMD2,3,17 のフィールド	87
10-25	分岐トレース・バッファ設定レジスタのフィールド (PMC12)	88
10-26	分岐トレース・バッファ・レジスタのフィールド (PMD8-15)	89
10-27	分岐トレース・バッファ・インデックス・レジスタのフィールド (PMD16)	90

10-28	Itanium® 2 プロセッサについて PAL_PERF_MON_INFO が返す情報	92
11-1	基本イベントのパフォーマンス・モニタ	94
11-2	基本イベントの派生モニタ	94
11-3	命令ディスパースル・イベントのパフォーマンス・モニタ	95
11-4	命令実行イベントのパフォーマンス・モニタ	96
11-5	命令実行イベントの派生モニタ	96
11-6	ストール・イベントのパフォーマンス・モニタ	97
11-7	分岐イベントのパフォーマンス・モニタ	98
11-8	L1 命令キャッシュ・イベントおよびプリフェッチ・イベントの パフォーマンス・モニタ	101
11-9	L1 命令キャッシュ・イベントおよびプリフェッチ・イベントの派生モニタ	102
11-10	L1 データ・キャッシュ・イベントのパフォーマンス・モニタ	102
11-11	L1D キャッシュ・セット 0 のパフォーマンス・モニタ	103
11-12	L1D キャッシュ・セット 1 のパフォーマンス・モニタ	103
11-13	L1D キャッシュ・セット 2 のパフォーマンス・モニタ	103
11-14	L1D キャッシュ・セット 3 のパフォーマンス・モニタ	103
11-15	L1D キャッシュ・セット 4 のパフォーマンス・モニタ	104
11-16	L2 ユニファイド・キャッシュ・イベントのパフォーマンス・モニタ	104
11-17	L2 ユニファイド・キャッシュ・イベントの派生モニタ	105
11-18	L2 キャッシュ・セット 0 のパフォーマンス・モニタ	106
11-19	L2 キャッシュ・セット 1 のパフォーマンス・モニタ	106
11-20	L2 キャッシュ・セット 2 のパフォーマンス・モニタ	106
11-21	L2 キャッシュ・セット 3 のパフォーマンス・モニタ	107
11-22	L2 キャッシュ・セット 4 のパフォーマンス・モニタ	107
11-23	L2 キャッシュ・セット 5 のパフォーマンス・モニタ	107
11-24	L3 ユニファイド・キャッシュ・イベントのパフォーマンス・モニタ	108
11-25	L3 ユニファイド・キャッシュ・イベントの派生モニタ	108
11-26	システム・イベントのパフォーマンス・モニタ	109
11-27	システム・イベントの派生モニタ	109
11-28	TLB イベントのパフォーマンス・モニタ	110
11-29	TLB イベントの派生モニタ	110
11-30	システム・バス・イベントのパフォーマンス・モニタ	111
11-31	システム・バス・イベントの派生モニタ	113
11-32	システム・バス・トランザクションの表記規則	114
11-33	スヌープ応答によるバス・イベント	115
11-34	RSE イベントのパフォーマンス・モニタ	115
11-35	RSE イベントの派生モニタ	115
11-36	コードによって並べ替えられるすべてのパフォーマンス・モニタ	116
11-37	ALAT_CAPACITY_MISS のユニット・マスク	122
11-38	BACK_END_BUBBLE のユニット・マスク	122
11-39	BE_BR_MISPREDICT_DETAIL のユニット・マスク	123
11-40	BE_EXE_BUBBLE のユニット・マスク	123
11-41	BE_FLUSH_BUBBLE のユニット・マスク	124
11-42	BE_L1D_FPU_BUBBLE のユニット・マスク	124
11-43	BE_LOST_BW_DUE_TO_FE のユニット・マスク	125
11-44	BE_RSE_BUBBLE のユニット・マスク	126
11-45	BR_MISPRED_DETAIL のユニット・マスク	127
11-46	BR_MISPREDICT_DETAIL2 のユニット・マスク	128
11-47	BR_PATH_PRED のユニット・マスク	129
11-48	BR_PATH_PRED2 のユニット・マスク	130
11-49	BUS_ALL のユニット・マスク	131
11-50	BUS_BACKSNP_REQ のユニット・マスク	131

11-51	BUS_IO のユニット・マスク	133
11-52	BUS_LOCK のユニット・マスク	133
11-53	BUS_MEMORY のユニット・マスク	134
11-54	BUS_MEM_READ のユニット・マスク	135
11-55	BUS_RD_DATA のユニット・マスク	137
11-56	BUS_RD_IO のユニット・マスク	138
11-57	BUS_RD_PRTL のユニット・マスク	138
11-58	BUS_SNOOPS のユニット・マスク	138
11-59	BUS_SNOOPS_HITM のユニット・マスク	139
11-60	BUS_SNOOP_STALL_CYCLES のユニット・マスク	139
11-61	BUS_WR_WB のユニット・マスク	140
11-62	ENCBR_MISPRED_DETAIL のユニット・マスク	142
11-63	EXTERN_DP_PINS_0_TO_3 のユニット・マスク	143
11-64	EXTERN_DP_PINS_4_TO_5 のユニット・マスク	143
11-65	FE_BUBBLE のユニット・マスク	144
11-66	FE_LOST_BW のユニット・マスク	145
11-67	IA64_INST_RETIRED のユニット・マスク	147
11-68	IA64_TAGGED_INST_RETIRED のユニット・マスク	147
11-69	IDEAL_BE_LOST_BW_DUE_TO_FE のユニット・マスク	148
11-70	INST_CHKA_LDC_ALAT のユニット・マスク	149
11-71	INST_FAILED_CHKA_LDC_ALAT のユニット・マスク	149
11-72	INST_FAILED_CHKS_RETIRED のユニット・マスク	150
11-73	ITLB_MISSES_FETCH のユニット・マスク	150
11-74	L1D_READ_MISSES のユニット・マスク	152
11-75	L1I_PREFETCH_STALL のユニット・マスク	154
11-76	L2_BAD_LINES_SELECTED のユニット・マスク	155
11-77	L2_BYPASS のユニット・マスク	156
11-78	L2_DATA_REFERENCES のユニット・マスク	156
11-79	L2_FILLB_FULL のユニット・マスク	157
11-80	L2_FORCE_RECIRC のユニット・マスク	157
11-81	L2_GOT_RECIRC_IFETCH のユニット・マスク	158
11-82	L2_IFET_CANCELSELS のユニット・マスク	159
11-83	L2_ISSUED_RECIRC_IFETCH のユニット・マスク	160
11-84	L2_L3ACCESS_CANCEL のユニット・マスク	161
11-85	L2_OPS_ISSUED のユニット・マスク	162
11-86	L2_OZDB_FULL のユニット・マスク	162
11-87	L2_OZQ_CANCELSELS0 のユニット・マスク	163
11-88	L2_OZQ_CANCELSELS1 のユニット・マスク	163
11-89	L2_OZQ_CANCELSELS2 のユニット・マスク	164
11-90	L2_OZQ_FULL のユニット・マスク	165
11-91	L2_STORE_HIT_SHARED のユニット・マスク	165
11-92	L2_VICTIMB_FULL のユニット・マスク	166
11-93	L3_READS のユニット・マスク	167
11-94	L3_WRITES のユニット・マスク	168
11-95	MEM_READ_CURRENT のユニット・マスク	168
11-96	RSE_REFERENCES_RETIRED のユニット・マスク	171
11-97	SYLL_NOT_DISPERSSED のユニット・マスク	172
11-98	SYLL_OVERCOUNT のユニット・マスク	173
12-1	Itanium® 2 プロセッサ CPUID の戻り値	175
12-2	キャッシュ戻り値のエンコード	176
A-1	FPU パイプライン	179

A-2	L1D マイクロパイプライン	179
A-3	L2 マイクロパイプライン	179

1.1 概要

インテル® Itanium® 2 プロセッサは、インテル Itanium プロセッサ・ファミリの 2 番目の製品である。本書では、Itanium 2 プロセッサによる Itanium アーキテクチャの機能のサポート方法と、パフォーマンス・チューニング、コンパイル、アセンブラのプログラミングに関連する Itanium 2 プロセッサ固有の機能について説明する。特に断らない限り、本書で説明するすべての制限、規則、サイズ、および容量は、Itanium 2 プロセッサにのみ適用され、Itanium プロセッサ・ファミリの他の製品には適用されない。

本書の読者は、プロセッサのコンポーネントと Itanium 命令についてよく理解している必要がある。本書は、Itanium アーキテクチャのアーキテクチャ・リファレンスとしては使用できない。Itanium アーキテクチャの詳細は、『インテル® Itanium® 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』を参照のこと。

1.2 構成

第 2 章「Itanium® 2 プロセッサの拡張」では、Itanium プロセッサと Itanium 2 プロセッサの相違点を説明する。ソフトウェアを Itanium 2 プロセッサ向けに最適化する際に必要ないくつかの留意点は強調表示されている。

第 3 章「機能ユニットと発行規則」では、使用可能な機能ユニットの数とタイプ、命令の発行規則、マシン・リソースと発行規則に基づく効率的な命令スケジューリングの経験的手法を説明する。

第 4 章「レイテンシとバイパス」では、Itanium 2 プロセッサ上で各種の命令タイプを実行する際のレイテンシとバイパスについて説明する。

第 5 章「データ操作」では、スペキュレーティブ・ロード/ストア、プレディケート付きロード/ストア、浮動小数点ロード、プリフェッチなどのデータ操作に関する留意点を説明する。データ・アライメントに関する留意点も説明する。

第 6 章「メモリ・サブシステム」では、インテル Itanium 2 プロセッサ上のメモリ・サブシステム階層の概要を説明する。

第 7 章「分岐命令と分岐予測」では、Itanium 2 プロセッサ上で分岐予測のヒントおよび命令プリフェッチをどのようにサポートしているかを説明する。

第 8 章「命令プリフェッチ機能」では、インテル Itanium 2 プロセッサ上でプリフェッチ機能をどのようにサポートしているかを説明する。

第 9 章「Itanium® 2 プロセッサ向けの最適化」では、第 8 章までの重要な内容から導き出した結論をまとめている。

第 10 章「パフォーマンス監視機能」では、Itanium 2 プロセッサ固有のパフォーマンス監視レジスタとパフォーマンス監視機能を説明する。

第 11 章「パフォーマンス監視イベント」では、Itanium 2 プロセッサのイベントと、頻繁に使用されるパフォーマンス評価基準の算出方法を説明する。

第 12 章「モデル固有の機能とオプションの機能」では、CPUID 命令の実行など、Itanium 2 プロセッサのモデル固有の動作を説明する。

1.3 用語

本書全体を通して使用される用語の定義を以下に示す。

- ディスペーサル** バンドル内の命令と機能ユニットを対応付けるプロセス。
- バンドル・ローテーション**
2 バンドルの発行ウィンドウに新しいバンドルを送り込むプロセス。
- スプリット発行** ある命令がその直前の命令と同時に発行されないときの命令の実行。
- アドバンスト・ロード・アドレス・テーブル (ALAT)**
ALAT は、アドバンスト・ロード / チェック操作に必要なステートを格納する。
- トランスレーション・ルックアサイド・バッファ (TLB)**
TLB には、仮想アドレスと物理アドレスの対応関係を格納する。
- 仮想ハッシュ・ページ・テーブル (VHPT)**
VHPT は、TLB 階層を拡張したものであり、仮想メモリ空間に置かれ、仮想アドレス変換のパフォーマンスを向上させる。
- ハードウェア・ページ・ウォーカ (HPW)**
HPW は、第 3 レベルのアドレス変換機構である。HPW は、VHPT からのページ参照を実行し、可能なときにプロセッサの TLB 内に変換を挿入するエンジンである。
- レジスタ・スタック・エンジン (RSE)**
RSE は、レジスタ・スタックとメモリ内のバッキング・ストアの間でレジスタを移動する。
- イベント・アドレス・レジスタ (EAR)**
EAR には、データ・キャッシュ・ミスの命令アドレスとデータ・アドレスを記録する。

1.4 参考文献

本書の読者は、以下のマニュアルで説明する内容と概念についても理解している必要がある。

- 『インテル® Itanium® 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: アプリケーション・アーキテクチャ
- 『インテル® Itanium® 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: システム・アーキテクチャ
- 『インテル® Itanium® 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

1.5 改訂履歴

改訂番号	説明	改訂時期
-001	本書の刊行	2002 年 6 月

本章では、Itanium 2 プロセッサ と Itanium プロセッサの主な相違点を説明する。本章はすべての相違点を示しているわけではない。各項目には参照箇所が記載されている。

2.1 サポートしている命令

Itanium 2 プロセッサ は、64 ビット長の分岐命令 (brl) をハードウェア上で直接サポートしている。この命令は、Itanium プロセッサではサポートしていない。この命令によって、プログラムは、64 のアドレス・ビットをすべて使用するアドレスへの分岐を指示できる。brl 命令の詳細は、『インテル® Itanium® 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』を参照のこと。brl 命令を使用すると、分岐予測のパフォーマンスに多少影響がある。これについては、第 7 章「分岐命令と分岐予測」で説明する。

2.2 機能ユニットと発行規則

一般的に、Itanium 2 プロセッサ は、Itanium プロセッサより多くの機能ユニットを搭載している。

- 特に、Itanium 2 プロセッサ は、算術演算、比較、ほとんどのマルチメディア命令などを実行する、6 つの ALU (Arithmetic Logic Unit) を搭載している。Itanium プロセッサは、これらのタイプの命令を 1 サイクル当たり 4 つ発行できる。
- Itanium 2 プロセッサ には、4 つのメモリ・ポートがあり、1 サイクル当たり 2 つの整数ロードと 2 つの整数ストアを発行できる。Itanium プロセッサには、2 つのメモリ・ポートがある。
- Itanium 2 プロセッサ は、1 サイクル当たり 1 つの SIMD 浮動小数点 (FP) 命令を発行できる。Itanium プロセッサは、1 サイクル当たり 2 つの SIMD FP 命令を発行できる。
- 一定の条件下では、Itanium 2 プロセッサ は I タイプの命令をメモリ機能ユニットに対して発行できるため、1 サイクルで発行できるテンプレート・ペアのタイプが増える。Itanium プロセッサでは、I タイプの命令は、整数機能ユニットに対してのみ発行される。
- Itanium 2 プロセッサ は、1 次命令キャッシュ (L1D) ミス、マルチメディア演算、浮動小数点演算などのマルチサイクル操作をスコアボード処理する。

つまり、整数演算がマルチメディア演算の結果を使用する場合、レイテンシを隠蔽するように整数演算がスケジューラされていないときは、依存する命令グループは、マルチメディア・データが使用可能になるまで待たなければならない。

スコアボード処理されるオペランドを使用するプレディケート・オフ操作は、プレディケートが直前のサイクルで生成された場合、発行グループを 1 サイクルの間ストールさせる。2 サイクル以上に生成されたプレディケートを使用するプレディケート・オフ命令では、オペランドがスコアボード処理される場合でも、パイプラインのストールは発生しない。

2.3 操作のレイテンシ

Itanium 2 プロセッサ 上のレイテンシは、2、3 の例外を除いて、Itanium プロセッサのレイテンシと同じか、それより小さい。Itanium 2 プロセッサ では、メモリ操作のレイテンシも浮動小数点演算のレイテンシも短縮されている。さらに、非対称性を除去するバイパスがいくつか追加されている。表 2-1 「Itanium® 2 プロセッサ / Itanium® プロセッサの操作レイテンシ」に、Itanium 2 プロセッサ と Itanium プロセッサのレイテンシを示す。表中の明るい色の部分が相違点を示している。スラッシュで区切られた 2 つの数値は、左側が Itanium 2 プロセッサ、右側が Itanium プロセッサを示している。

2.4 データ操作

2.4.1 データ・スペキュレーションと ALAT

Itanium 2 プロセッサ のアドバンスド・ロード・アドレス・テーブル (ALAT) はフル・アソシアティブであるが、Itanium プロセッサの ALAT は 2 ウェイ・アソシアティブである。

Itanium プロセッサでは、`ld.c` が ALAT にヒットしなかった場合、10 サイクルのパイプライン・フラッシュが発生する。Itanium 2 プロセッサ では、このペナルティは 8 サイクルである。

Itanium プロセッサでは、`chk.a` または `chk.s` が失敗すると、トラップ・ハンドラによって OS ハンドラが起動され、`chk.a/chk.s` 命令のターゲット・フィールドで指定された位置にあるリカバリ・コードに実行を渡す。Itanium 2 プロセッサ では、通常はハードウェアがオペレーティング・システムの介入なしでこの移行を実行する。これによって、再移行のコストが約 200 サイクルから 18 サイクルに軽減される。以下のいずれかの条件が満たされない場合、Itanium 2 プロセッサは、OS へのトラップを実行して `chk.a/chk.s` を処理させる。

```
psr.ic=1
psr.it=1
psr.ss=0
psr.tb=0
```

Itanium プロセッサでは、`chk.a` が同じサイクル内のストアの後に続く場合、`chk.a` は常に失敗する。Itanium 2 プロセッサ では、ALAT エントリとの間で 12 ビット・アドレスの比較が行われる。詳細は、5.1 節「データ・スペキュレーションと ALAT」を参照のこと。

2.4.2 データ・アライメント

Itanium プロセッサは、16 バイト・ブロック内のアライメントの合わない整数アクセスをサポートする。Itanium 2 プロセッサ は、8 バイト・ブロック内のアライメントの合わない整数アクセスをサポートする。Itanium 2 プロセッサ によるアライメントの合わないアクセスのサポートについては、5.5 節「データ・アライメント」を参照のこと。

表 2-1. Itanium® 2 プロセッサ / Itanium® プロセッサの操作レイテンシ

		結果を参照する側									
		修飾 プレディ ケート	分枝 プレディ ケート	ALU	アドレスの ロード/ ストア	マルチ メディア	データの ストア	Fmac	Fmisc	getf	setf
結果を生成する側	加算操作: add, cmp, tbit, addp4, shladd, shladdp4, sum, 論理演算、64 ビット即値移動、movl, inc 後の操作 (inc 後のストア、ロード、lfetch を含む)	なし	なし	1	1/(1-2) ¹	3	1	なし	なし	なし	1
	マルチメディア	なし	なし	3	3	2	3	なし	なし	なし	3
	getf	なし	なし	5/9	6/9	6/9	5/9	なし	なし	なし	6/9
	setf	なし	なし	なし	なし	なし	6/2	6/2	6/2	6/2	なし
	Fmac: fma, fms, fnma, fpma, fms, fpmna, fadd, fnmpy, fsub, fpmpy, fpmmpy, fmpy, fnorm, xma, frcpa, fprcpa, frsqta, fpsqta, fcvt, fpcvt	なし	なし	なし	なし	なし	4/5	4/5	4/5	4/5	なし
	Fmisc: fselect, fcmp, fclass, fmin, fmax, famin, famax, fpmin, fpmax, fpamin, fpcmp, fmerge, fmix, fsxt, fpack, fswap, fand, fandcm, for, fxor, fpmerge, fneg, fnegabs, fpabs, fpneg, fpnegabs	なし	なし	なし	なし	なし	4/5	4/5	4/5	4/5	なし
	INT 側プレディケートの書き込み: cmp, tbit, tnat	1	0	なし	なし	なし	なし	なし	なし	なし	なし
	FP 側プレディケートの書き込み: fcmp	2	1/1	なし	なし	なし	なし	なし	なし	なし	なし
	FP 側プレディケートの書き込み: frcpa, fprcpa, frsqta, fpsqta	2	2	なし	なし	なし	なし	なし	なし	なし	なし
	Int ロード ²	なし	なし	N	N+1	N+1	N	N	N	N	N
	FP ロード ³	なし	なし	M+1	M+2	M+2	M+1	M+1	M+1	M+1	M+1
	IEU2: move_from_br, alloc	なし	なし	2	2	3	2	なし	なし	なし	2
	cr, ar ⁴ との間の移動	なし	なし	C	C	C	C	なし	なし	なし	C
	pr への移動	1	0	2	2	3	2	なし	なし	なし	なし
間接移動 ⁵	なし	なし	D	D	D	D	なし	なし	なし	D	

1. Itanium プロセッサでは、レイテンシ・サイクルが増えないように、アドレス計算命令は M スロット・タイプに入れなければならない。
2. N はヒットしたキャッシュのレベルによって異なる。Itanium プロセッサでは、L1D の場合は N=2、L2 の場合は N=6、L3 の場合は N=21 である。Itanium 2 プロセッサ では、L1D の場合は N=1、L2 の場合は N=5、L3 の場合は N=(12 ~ 15) である。これらの値は最小レイテンシである。
3. M はヒットしたキャッシュのレベルによって異なる。Itanium プロセッサでは、L2 の場合は M=8、L3 の場合は M=24 である。Itanium 2 プロセッサ では、L2 の場合は M=5、L3 の場合は M=(12 ~ 15) である。これらの値は最小レイテンシである。「+1」の項目は、フォーマットの変換に 1 サイクル余分にかかることを示す。
4. C の最良の場合の値は、アクセスするレジスタによって、2 ~ 35 サイクルの範囲である。EC および LC のアクセスは 2 サイクルである。FPSR および CR のアクセスは 10 ~ 12 サイクルである。
5. D の最良の場合の値は、アクセスする間接レジスタによって、6 ~ 35 サイクルの範囲である。lreg の pkr および rr のアクセスは高速 (6 サイクル) である。

2.4.3 コントロール・スペキュレーション

Itanium 2 プロセッサ は、誤ったコントロール・スペキュレーションのコストを減らすことによってアプリケーションのパフォーマンスを向上させる機能を備えている。このために、Itanium 2 プロセッサ には、次の2つの解決策がある。

- 第1に、スペキュレーティブ・ロード (これには、.fault コンプリータを持たない lfetch が含まれる) は、データ・トランスレーション・ルックアサイド・バッファ (TLB) ミスの発生時に、アボートして NaT ビットをセットできる。これに対して、Itanium プロセッサは、ハードウェア・ページ・ウォーク (HPW) がウォークを完了するまで待ってから、NaT ビットをセットする。
- 第2に、chk.s 命令 (および chk.a 命令) は、OS の介入なしに、修復コードに直接に分岐できる。Itanium プロセッサは、chk.s または chk.a 命令が失敗するとフォルトを生成し、修復コードに分岐するように OS に要求する。

このように、Itanium 2 プロセッサ では、抑止が迅速に行われ、修復コードへの分岐も迅速に行われる。

割り込みハンドラ内 (PSR.is = 1 のとき) では、データ TLB ミスの発生時の抑止機能はオフにされる。これによって、ld.s および lfetch 命令は、TLB ウォークを完了し、通常はデータを返すことができる。dcr.dm ビットをクリアすれば、データ TLB ミスの発生時のスペキュレーティブ操作の抑止も行われない。高速抑止機能を有効にするには、dcr.dm ビットをセットする必要がある。詳細は、5.2 節「スペキュレーティブ・ロード/ストアとプレディケート付きロード/ストア」を参照のこと。

2.5 メモリ階層

Itanium マイクロアーキテクチャと Itanium 2 マイクロアーキテクチャは、両方とも 3 レベルのキャッシュ構造を統合している。一般的に、Itanium 2 プロセッサ のライン・サイズは Itanium プロセッサ の 2 倍である。また、Itanium 2 プロセッサ のレイテンシは Itanium プロセッサ より小さい。Itanium 2 プロセッサ の 3 次キャッシュ (L3) は Itanium プロセッサ の L3 より小さいが、オンチップであり高速のコア周波数で動作するため、レイテンシははるかに短縮されている。Itanium 2 プロセッサ は 2 レベルの命令 TLB と 2 レベルのデータ TLB を搭載しているが、Itanium プロセッサ は 1 レベルの命令 TLB と 2 レベルのデータ TLB を搭載している。Itanium 2 プロセッサ の TLB のサイズは、Itanium プロセッサ より大きい。以下の表は、キャッシュと TLB の相違点をまとめたものである。詳細は、第 6 章「メモリ・サブシステム」を参照のこと。

表 2-2. L1I キャッシュの相違点

	Itanium 2 プロセッサ	Itanium プロセッサ
サイズ	16KB	16KB
ライン・サイズ	64 バイト	32 バイト
アソシアティビティ	4 ウェイ	4 ウェイ
レイテンシ	1 サイクル	1 サイクル

表 2-3. L1D キャッシュの相違点

	Itanium 2 プロセッサ	Itanium プロセッサ
サイズ	16KB	16KB
ライン・サイズ	64 バイト	32 バイト
アソシアティビティ	4 ウェイ	4 ウェイ

表 2-3. L1D キャッシュの相違点 (続き)

	Itanium 2 プロセッサ	Itanium プロセッサ
レイテンシ	1 サイクル	2 サイクル
書き込みポリシー	ライトスルー、 ライト・アロケートなし	ライトスルー、 ライト・アロケートなし

表 2-4. L2 ユニファイド・キャッシュの相違点

	Itanium 2 プロセッサ	Itanium プロセッサ
サイズ	256KB	96KB
ライン・サイズ	128 バイト	64 バイト
アソシアティビティ	8 ウェイ	6 ウェイ
整数レイテンシ	最小 5 サイクル	最小 6 サイクル
浮動小数点レイテンシ	最小 6 サイクル	最小 9 サイクル
書き込みポリシー	ライトバック、 ライト・アロケート	ライトバック、 ライト・アロケート

表 2-5. L3 キャッシュの相違点

	Itanium 2 プロセッサ	Itanium プロセッサ
サイズ	3MB または 1.5MB、 オンチップ	4MB または 2MB、 オフチップ
ライン・サイズ	128 バイト	64 バイト
アソシアティビティ	12 ウェイ	4 ウェイ
整数レイテンシ	最小 12 サイクル	最小 21 サイクル
浮動小数点レイテンシ	最小 13 サイクル	最小 24 サイクル
帯域幅	32 バイト / サイクル	16 バイト / サイクル

表 2-6. 命令 TLB の相違点

	Itanium 2 プロセッサ	Itanium プロセッサ
階層	2 レベル : L1 ITLB、L2 ITLB	1 レベル : ITLB
サイズ	32 エントリ、128 エントリ	64 エントリ
アソシアティビティ	フル、フル	フル

表 2-7. データ TLB の相違点

	Itanium 2 プロセッサ	Itanium プロセッサ
階層	2 レベル : L1 DTLB、 L2 DTLB	2 レベル : L1 DTLB、 L2 DTLB
サイズ	32 エントリ、128 エントリ	32 エントリ、96 エントリ
アソシアティビティ	フル、フル	直接、フル
L1 DTLB ミスのペナルティ	2 サイクル	10 サイクル

2.6 分岐予測

Itanium 2 プロセッサ と Itanium プロセッサの分岐予測機能には、次のような主な相違点がある。

- レイテンシが短縮されている。
- 分岐予測には、`brp` 命令は無視される。つまり、ゼロバブル分岐を達成するのに、`brp.imp` は不要である。
- 間接分岐ターゲットは、ハードウェア・テーブルからではなく、ソース分岐レジスタから予測される。
- 予測エンコーディングによって、BBB バンドルの予測を減らせる可能性がある。
- 分岐予測ミスのリターン後には予測構造の修復方法の安定性が向上している。
- `brl` (64 ビット相対分岐) 命令をハードウェア上でサポートしている。
- 完全ループ予測のために、`ar.ec = 1` に設定する必要はない。

詳細には、第 7 章「分岐命令と分岐予測」を参照のこと。

表 2-8. 分岐予測のレイテンシ (サイクル単位)

	Itanium 2 プロセッサ	Itanium プロセッサ
実行されると正しく予測された IP 相対分岐	0	1
実行されると正しく予測された間接分岐	2	0
実行されると正しく予測されたリターン分岐	1	1
完全ループ予測内の最後の分岐	0	2
分岐予測ミスのレイテンシ	6+	9

2.7 命令プリフェッチ

Itanium 2 プロセッサ では、ストリーミングおよびヒント・プリフェッチのサポートが強化されている。詳細は、第 8 章「命令プリフェッチ機能」を参照のこと。

本章では、使用可能な機能ユニットの数とタイプ、命令の発行規則、マシン・リソースと発行規則に基づく効率的な命令スケジューリングの経験的手法を説明する。

3.1 実行モデル

Itanium® 2 プロセッサは、命令の発行と実行をアセンブリ・コードの順序で行う。したがって、高性能のアセンブリ・コードを生成するために、プログラマはストールの条件を理解している必要がある。

一般的に、ある命令とその直前の命令が同時に発行されない場合、命令の実行はスプリット発行と呼ばれる。スプリット発行状態が発生すると、スプリット・ポイント以降のすべての命令は、命令の実行リソースが十分にあっても、1 クロック以上ストールする。Itanium 2 プロセッサのスプリット発行の一般的な原因には、以下のものがある。

- 明示的なストップが検出された。
- 命令の実行に必要なタイプのマシン・リソースが不足している。
- 命令が Itanium 2 プロセッサの発行規則に従って配置されていない。

Itanium 2 プロセッサは、静的スケジュールによって定義された順序で命令を発行する。コード・ジェネレータは、1 つの発行グループ内のレジスタの依存関係を避けるように注意する必要がある。Itanium 2 プロセッサは、WAW ハザードを解決するために明示的なストップ・ビットを挿入しない。したがって、ロードとストアの間の WAW ハザードがあると、プレディケートが真である場合、8 サイクルのペナルティが発生する。他の WAW ハザード (ALU 演算による WAW ハザードなど) がある場合の結果は不定である (この場合も、プレディケートが考慮に入れられる)。

複数の命令がグループとして発行されると、それらの命令はパイプライン内でグループとして処理される。発行グループ内の 1 つの命令がストール状態になると、グループ全体がストールする。このストールによって、パイプライン内でそれに続く (より若い) すべての命令もストールする。

3.2 機能ユニットの数とタイプ

並列命令グループを構成するバンドルの数や、各グループに含まれる各タイプの命令の数に制限はないが、Itanium 2 プロセッサの実行リソースは有限である。並列命令グループに、使用可能な実行ユニットの数より多くの命令が含まれている場合は、適切なユニットが見つからなかった最初の命令でスプリット発行が発生し、並列命令グループは分割される。

Itanium 2 プロセッサ・パイプラインのフロントエンドは、1 サイクルあたり最大 2 バンドルをフェッチできる。パイプラインのバックエンドは、1 サイクルあたり最大 2 バンドルを発行できる。1 バンドルあたり 3 個の命令があるとすると、Itanium 2 プロセッサを 6 命令発行マシンと考えることができる。パイプラインの詳細は、付録 A 「Itanium® 2 プロセッサのパイプライン」を参照のこと。

Itanium 2 プロセッサは、さまざまなタイプの機能ユニットを多数搭載している。これによって、1 サイクルごとにさまざまな組み合わせの命令を発行できる。発行される命令は 1 サイクルあたり 6 個までに制限されているため、以下に説明する Itanium 2 プロセッサの機能ユニットの一部だけが、各サイクルで使用される。

Itanium 2 プロセッサには、6 つの汎用 ALU ユニット (ALU0、1、2、3、4、5)、2 つの整数ユニット (I0、1)、1 つのシフト・ユニット (ISHIFT、汎用シフトおよびその他の特殊な命令に使用される) がある。これらのタイプの命令は、1 サイクルあたり最大 6 個まで発行できる。

データ・キャッシュ・ユニット (DCU) には、4 つのメモリ・ポートが含まれる。通常は、2 つのポートがロード操作に使用され、2 つのポートがストア操作に使用される。これらのタイプの命令は、1 サイクル当たり最大 4 個まで発行できる。2 つのストア・ポートは、一部の特殊な浮動小数点ロード命令にも使用される。

Itanium 2 プロセッサには、6 つのマルチメディア機能ユニット (PALU0、1、2、3、4、5)、2 つの並列シフト・ユニット (PSMU0、1)、1 つの並列乗算ユニット (PMUL)、および 1 つのポピュレーション・カウント・ユニット (POPCNT) がある。これらのユニットは、マルチメディア、並列乗算、popcnt の各命令タイプを処理する。Itanium 2 プロセッサは、1 サイクル当たり最大 6 個の PALU 命令を発行できる。また、1 サイクル当たり最大 1 個の pmul または popcnt 命令を発行できる。

Itanium 2 プロセッサには、4 つの浮動小数点機能ユニットがある。そのうち、2 つの FMAC ユニットは浮動小数点乗算 / 加算を実行し、2 つの FMISC ユニットは fcmp、fmerge などのその他の浮動小数点演算を実行する。1 サイクル当たり最大 2 つの浮動小数点操作を実行できる。

Itanium 2 プロセッサには、3 つの分岐ユニットがあり、1 サイクル当たり 3 つの分岐を実行できる。

すべての計算機能ユニットはフルにパイプライン化されているため、各機能ユニットは、他のタイプのストールがない場合、1 クロック・サイクル当たり 1 個の新しい命令を受け入れられる。ただし、システム命令とシステム・レジスタへのアクセスには、この規則が適用されない場合がある。

3.3 命令スロットと機能ユニットの対応関係

フェッチされた各命令は、発行ポートを介して機能ユニットに割り当てられる。多数の機能ユニットが、少数の発行ポートを共有している。11 個の発行ポートがあり、そのうち 8 つは非分岐命令用ポート、3 つは分岐命令用ポートである。各ポートは、M0、M1、M2、M3、I0、I1、F0、F1、B0、B1、B2 と呼ばれる。バンドル内の命令を機能ユニットに対応付けるプロセスは、ディスパーサルと呼ばれる。

各命令がどの発行ポートに割り当てられるかは、命令のタイプと発行グループ内の位置によって決まる。命令は、ALU (A)、メモリ (M)、整数 (I) などの命令タイプに基づいて、発行ポートのサブセットに対応付けられる。次に、命令は、ディスパーサルの対象になる命令グループ内の命令の位置に基づいて、サブセット内の特定の発行ポートに対応付けられる。

表 3-1 「A タイプの命令とポートの対応関係」、表 3-2 「I タイプの命令とポートの対応関係」、および表 3-3 「M タイプの命令とポートの対応関係」に、命令タイプとポート / 機能ユニットの対応関係を示す。命令の位置に基づく特定のポートの選択については、3.3.2 項で説明する。

注： 以下の表の暗い色の部分は、その命令タイプを発行できるポートを示す。

A タイプの命令は、すべての M ポートおよび I ポート (M0 ~ M3、I0、および I1) で発行できる。I タイプの命令は、I0 または I1 にのみ発行できる。I ポートは非対称ポートであり、一部の I タイプ命令は、ポート I0 でのみ発行できる。M ポートには多くの非対称性がある。M タイプの命令には、すべてのポートで発行できるもの、M0 および M1 でのみ発行できるもの、M2 および M3 でのみ発行できるもの、M0 でのみ発行できるもの、M2 でのみ発行できるものがある。

表 3-1. A タイプの命令とポートの対応関係

命令タイプ	説明	例	ポート
A1 ~ A5	ALU	add, shladd	M0 ~ M3, I0, I1
A4, A5	即値の加算	addp4, addl	M0 ~ M3, I0, I1
A6, A7, A8	比較	cmp, cmp4	M0 ~ M3, I0, I1
A9	MM ALU	pcmp[1 2 4]	M0 ~ M3, I0, I1
A10	MM シフトおよび加算	pshladd2	M0 ~ M3, I0, I1

表 3-2. I タイプの命令とポートの対応関係

命令タイプ	説明	例	I ポート	
			I0	I1
I1	MM 乗算 / シフト	pmpy2.[l r], pmpyshr2{.u}		
I2	MM ミックス / パック	mix[1 2 4].[l r] pmin, pmax		
I3, I4	MM mux	mux1, mux2		
I5	可変右シフト	shr{.u} =ar,ar pshr[2 4] =ar,ar		
I6	MM 右シフト固定	pshr[2 4] =ar,c		
I7	可変左シフト	shl{.u} =ar,ar pshl[2 4] =ar,ar		
I8	MM 左シフト固定	pshl[2 4] =ar,c		
I9	MM ポピュレーション・カウント	popcnt		
I10	ペアの右シフト	shrp		
I11 ~ I17	抽出、デポジット テスト NaT	extr{.u}, dep{.z} tnat		
I19	ブレーク、NOP	break.i, nop.i		
I20	整数スペキュレーション・チェック	chk.s.i		
I21 ~ 28	BR/PR/IP/AR との間の移動	mov =[br pr ip ar] mov [br pr ip ar]=		
I29	sxt/zxt/czx	sxt, zxt, czx		

表 3-3. M タイプの命令とポートの対応関係

命令タイプ	説明	例	メモリ・ポート			
			M0	M1	M2	M3
M1, 2, 3	整数ロード	ldsz, ld8.fill				
M4, 5	整数ストア	stsz, st8.spill				
M6, 7, 8	浮動小数点ロード	ldffsz, ldffsz.s, ldf.fill				
	浮動小数点アドバンスト・ロード	ldffsz.a, ldffsz.c.[clr nc]				
M9, 10	浮動小数点ストア	stffsz, stf.spill				
M11, 12	浮動小数点ロード・ペア	ldfpfsz				

表 3-3. M タイプの命令とポートの対応関係 (続き)

命令タイプ	説明	例	メモリ・ポート			
			M0	M1	M2	M3
M13、14、15	ライン・プリフェッチ	lfetch				
M16	比較および交換	cmpxchgsz.[acq rel]				
M17	フェッチおよび加算	fetchaddsz.[acq rel]				
M18	浮動小数点レジスタのセット	setf.[s d exp sig]				
M19	浮動小数点レジスタの取得	getf.[s d exp sig]				
M20、21	スペキュレーション・チェック	chk.s{.m}				
M22、23	アドバンスト・ロード・チェック	chk.a{clr nc}				
M24	ALAT の無効化	invala				
	メモリ・フェンス、同期化、シリアル化	fwb、mf{.a}、srlz.{d i}、sync.li				
M25	RSE の制御	flushrs、loadrs				
M26、27	ALAT の無効化	invala.e				
M28	キャッシュのフラッシュ、TC エントリのバージ	fc、ptc.e				
M29、30、31	アプリケーション・レジスタとの間の移動	mov{.m} ar= mov{.m} =ar				
M32、33	制御レジスタとの間の移動	mov cr=、mov =cr				
M34	レジスタ・スタック・フレームの割り当て	alloc				
M35、36	プロセッサ・ステータス・レジスタとの間の移動	mov psr.[l um] mov =psr.[l m]				
M37	ブレーク、nop.m	break.m、nop.m				
M38、39、40	プローブ・アクセス	probe.[r w].{fault}				
M41	トランスレーション・キャッシュの挿入	itc.{d i}				
M42、43	間接レジスタの移動 トランスレーション・レジスタの挿入	mov ireg=、move =ireg、 itr.{d i}				
M44	ユーザ・マスク / システム・マスクのセット / リセット	sum、rum、ssm、rsm				
M45	トランスレーション・キャッシュ / レジスタのバージ	ptc.{d i g ga}				
M46	仮想アドレス変換	tak、thash、tpa、ttag				

3.3.1 実行幅

Itanium 2 プロセッサは、命令を機能ユニットにディスパーサルするとき、特別なアライメントの必要条件なしに、一度に最大 2 つのバンドルを観察する。本書では、これらのバンドルを第 1 バンドルと第 2 バンドルと呼ぶ。新しいバンドルは、バンドル・ローテーションによって、発行の対象になる 2-バンドルの命令ウィンドウに送り込まれる。バンドル内のすべての命令が発行されると、バンドル・ローテーションが行われる。発行された命令の数に応じて、1 つまたは 2 つのバンドルがローテートされる。

3.3.2 ディスパーサル規則

Itanium 2 プロセッサのハードウェアは、ストールを避けるために命令の順序を変更しようとしません。したがって、コード・ジェネレータは、不要なストールを避けるために、並列命令グループ内の命令の数、タイプ、順序に注意しなければならない。プレディケートを使用しても、ディスパーサルには影響を与えない。真のプレディケート、偽のプレディケート、プレディケートなしを問わず、すべての命令は同じようにディスパーサルされる。同様に、`nop` 命令は、通常の命令と同じように機能ユニットにディスパーサルされる。実行ユニットのディスパーサル規則は、スロットのタイプ (I、M、F、B、または L) によって異なる。各スロット・タイプの規則について、以下に説明する。

F スロット命令のディスパーサル規則

- 第 1 バンドル内の F スロット命令は、F0 に対応付けられる。
- 第 2 バンドル内の F スロット命令は、F1 に対応付けられる。
- SIMD FP 命令は、基本的に F0 と F1 の両方に対応付けられる。SIMD FP 命令の発行規則の詳細については、[3.3.3 項](#)を参照のこと。

B スロット命令のディスパーサル規則

- MBB または BBB バンドル内の各 B スロット命令は、それに対応する B ユニットに対応付けられる。すなわち、テンプレートの最初の位置にある B スロット命令は B0 に対応付けられ、2 番目の位置にある命令は B1 に対応付けられ、3 番目の位置にある命令は B2 に対応付けられる。
- MIB/MFB/MMB バンドル内の B 命令は、その命令が `brp` または `nop.b` であり、第 1 バンドル内にある場合は、B0 に対応付けられる。それ以外の場合は、B2 に対応付けられる。
- ディスパーサルのために、`break.b` は分岐のように扱われる。

L スロット命令のディスパーサル規則

- MLX バンドルは、MFI バンドルと同じポートを使用する。MLX バンドルが第 1 バンドルである場合は、L スロット命令は F0 に対応付けられる。それ以外の場合は、L スロット命令は F1 に対応付けられる。ただし、MLX テンプレートと MMF または MIF バンドルと一緒に発行され、F 命令が SIMD FP 命令である場合は、競合は発生しない。

I スロット命令のディスパーサル規則

- 2 バンドルの発行グループの最初の I スロットの命令は、I0 に対して発行される。2 番目の I スロット命令は、I1 に対して発行される。
- 2 番目の I スロット命令が I0 ポート専用である場合は ([表 3-2](#)を参照)、暗黙的なストップが挿入され、2 番目の I スロット命令は次のサイクルで発行される。したがって、I0 専用の命令は、バンドル・ペアの最初の I スロットに置く必要がある。I0 専用の命令は、1 サイクル当たり 1 個しか発行できない。
- I スロット内の命令は、必ずしも I ポートに対して発行されるとは限らない。最初の 2 つの I スロット命令がすでに I ポートに対して発行され、その発行グループ内の追加の I スロット命令が [表 3-1](#) の A タイプの命令を含む場合、M ポートが使用可能であれば、これらの命令は使用可能な M ポートに対応付けられる。これによって、MII-MII バンドル・ペアのデュアル発行が可能になる。これは Itanium 2 プロセッサの新機能であり、Itanium プロセッサではサポートしていない。
- MLI テンプレートでは、第 1 バンドルの場合、I スロット命令は常にポート I0 に対応付けられ、第 2 バンドルの場合、I スロット命令はポート I1 に対応付けられる。したがって、バンドル・ペア MII-MLI のデュアル発行はできない。

M スロット命令のディスパースル規則

Itanium 2 プロセッサでは、M スロット命令は以下の 4 つのサブタイプに分類される (表 3-3 を参照)。

- ロード・サブタイプ。M0 または M1、あるいはその両方で発行できる (例えば、整数ロード、sync)。
- ストア・サブタイプ。M2 または M3、あるいはその両方で発行できる (例えば、整数ストア、alloc、getf)。
- 一般サブタイプ。4 つの M ポートのうちどれでも発行できる (例えば、ALU、浮動小数点ロード)。
- 特殊な命令。M2 ポートでのみ発行できる (例えば、getf、AR への mov)。

発行ロジックでは、異なるサブタイプの M スロット命令の順序は変更できるが、同じサブタイプに属する命令の順序は変更できない。例えば、発行グループ内で整数ストアが整数ロードに先行していても、スプリット発行は起こらない。2 つの命令は異なるサブタイプに属するため、ストアは M2 に対応付けられ、ロードは M0 に対応付けられる。

ただし、ストアが getf に先行する場合は、ストアが M2 に発行され、getf は必ず M2 に発行されるため、スプリット発行が起こる。同じサブタイプに属する命令の順序は変更されない。したがって、コード・スケジューラは、ポートの重複を避けるために、getf 命令をストアの前に配置し、getf 命令が M2 に対応付けられ、ストアが M3 に対応付けられるように保証する必要がある。

発行グループ内で先行する一般サブタイプ命令が M ポートを使用する場合は、ディスパースルはさらに複雑になる。このような場合の総合的な規則はない。発行グループ内ではより限定的なサブタイプが先行するように、スケジューリングすることが望ましい。例 3-1 と例 3-2 に、ディスパースルの可能性の例を示す。

注: M_A は一般サブタイプ命令、 M_L は整数ロード命令、 M_S はストア・サブタイプ命令である。

例 3-1. $M_A M_L I - M_S M_A I$

バンドル・ペア $M_A M_L I - M_S M_A I$ は、ポート M2 M0 I0 - M3 M1 I1 に対応付けられる。

最初の一般サブタイプ命令が M2 に対応付けられるため、 M_S 命令は M3 に対応付けられる。 M_S が getf 命令である場合は、スプリット発行が起こる。

例 3-2. $M_A M_A I - M_S M_A I$

バンドル・ペア $M_A M_A I - M_S M_A I$ は、ポート M0 M1 I0 - M2 M3 I1 に対応付けられる。この場合は、 M_S はより望ましい M2 ポートを取得できる。

表 3-4 に、Itanium 2 プロセッサがデュアル発行できるバンドル・タイプの組み合わせを (暗い色の部分で) 示す。行はバンドル・ペアの第 1 バンドル、列は第 2 バンドルを示す。

表 3-4. デュアル発行できるバンドル・タイプ

	MII	MLI	MMI	MFI	MMF	MIB	MBB	BBB	MBB	MFB
MII										
MLI										
MMI										
MFI										
MMF										
MIB ¹										

表 3-4. デュアル発行できるバンドル・タイプ (続き)

	MII	MLI	MMI	MFI	MMF	MIB	MBB	BBB	MBB	MFB
MBB										
BBB										
MBB										
MFB										

1. B は nop.b または brp でなければならない。

注：浮動小数点ロードは一般サブタイプ命令であるため、1 サイクル当たり 4 個の命令を発行できる。これには、すべてのサイズの通常の浮動小数点ロードとスペキュレーティブ浮動小数点ロードが含まれるが、アドバンスド浮動小数点ロード、ロード・ペア命令、チェック・ロード命令、浮動小数点ストア命令は含まれない。

3.3.3 スプリット発行とバンドル・タイプ

Itanium 2 プロセッサでは、機能ユニットの数が増加し、場合によっては I スロット命令を M ポートに発行できるため、多くのバンドル・ペアをデュアル発行できる。リソースの重複が起こるのは、非常にまれである。バンドル・ペアをデュアル発行できない場合は、明示的なストップか、前の項で説明したディスパースルの問題が原因である。さらに、スプリット発行の原因となる、Itanium 2 プロセッサ-固有の (アーキテクチャ的ではない) 特殊な場合がいくつかある。これらの特殊な場合は、以下に説明する。

分岐

BBB/MBB これらのバンドルの後、常にスプリット発行が起こる。

MIB/MFB/MMB B スロットに nop.b または brp 命令が入っている場合を除いて、これらのバンドルの後はスプリット発行が起こる。これらのバンドル・タイプでは、br 命令を使用すると、暗黙的なストップ・ビットが必ず追加される。

MIB BBB このペアの第 1 バンドルの後、B ポートの重複によって、スプリット発行が起こる。

SIMD FP

命令が SIMD FP 命令である場合は、1 サイクルで発行できる FP 命令は 1 つだけである。例えば、以下のバンドル・ペアの場合、

$MF_p I MFI$

(F_p は SIMD FP 演算)、第 2 バンドルの F 命令が nop.f であっても、第 2 バンドルの M 命令と F 命令の間に暗黙的なストップが挿入される。

同様に、以下のバンドル・ペアの場合、

$MFI MF_p I$

最初の F 命令は既に F0 に対応付けられているが、 F_p 命令は必ず F0 ポートに発行されるため、第 2 バンドルの M 命令と F_p 命令の間に暗黙的なストップが挿入される。

以下のバンドル・ペアの場合は、スプリット発行が起こりそうだが、実際には発生しない。

$MF_p I MLX$

L スロットが F ポートに対応付けられる場合でも、これらの 2 バンドルはデュアル発行される。

本章では、Itanium® 2 プロセッサ上で各種の命令タイプを実行する際のレイテンシとバイパスについて説明する。

一般的に、整数命令のレイテンシは1 サイクル、浮動小数点命令のレイテンシは4 サイクル、マルチメディア命令のレイテンシは2 サイクル、L1 キャッシュ・ヒットのレイテンシは1 サイクルである。ただし、非対称バイパスのために、この規則に該当しない多くの特殊な場合がある。

4.1 コントロール・スペキュレーションとデータ・スペキュレーションのペナルティ

Itanium 2 プロセッサは、OS フォルト・ハンドラへのトラップを実行せずに、chk.a/chk.s 命令内のオフセットから、リカバリ・コードのアドレスを計算できる。表 4-1 に示したスペキュレティブ・ロードのリカバリ・レイテンシは、chk.s/chk.a のリタイアメントから修復コードの最初の命令の完了までの時間差に基づく概算値である。これらのレイテンシには、キャッシュ・レイテンシや TLB レイテンシは含まれず、リカバリ・コードそのもののコストも含まれない。アドバンスト・ロードの詳細は、5.1 節「データ・スペキュレーションと ALAT」を参照のこと。

表 4-1. スペキュレティブ・ロードのリカバリ・レイテンシ

命令	レイテンシ (サイクル)
chk.a, int および fp (ALAT ヒット)、chk.s (NaT/NatVal なし)	0
chk.a, int および fp (ALAT ミス)、chk.s (NaT/NatVal)	18
ld*.c, ldf*.c (ALAT ヒット、L1/L2 ヒット)	0
ld*.c, ldf*.c (ALAT ミス、L1/L2 ヒット)	8

4.2 分岐に関連するレイテンシとペナルティ

表 4-2 に、分岐操作および分岐に関連するフラッシュのレイテンシを示す。詳細は、第 7 章「分岐命令と分岐予測」を参照のこと。

表 4-2. 分岐予測のレイテンシ

分岐のタイプ	分岐有無予測	分岐ターゲット予測	フロントエンド・バブル
IP 相対	正しい	正しい	0
IP 相対	正しい	誤り	1/6 ¹
リターン	正しい	正しい	1
リターン	正しい	誤り	6

1. 40 ビット境界を超える IP 相対分岐の場合は、6 サイクルのペナルティが発生する。予測ミスになったループ分岐は、7 サイクルを必要とする。これらの場合は、完全な分岐予測ミス・ペナルティが発生する。

表 4-3. バイパス使用時の実行レイテンシのまとめ

結果を参照する側 (右方向) 結果を生成する側 (下方向)	修飾 プレディ ケート	分岐 プレディ ケート	ALU	アドレスの ロード/ ストア	マルチ メディア	データの ストア	Fmac	Fmisc	getf	setf
加算操作: add, cmp, tbit, addp4, shladd, shladdp4, sum, 論理演算、64ビット即値移動、movl、inc 後の操作 (inc 後のストア、ロード、lfetch を含む)	なし	なし	1	1	3	1	なし	なし	なし	1
マルチメディア	なし	なし	3	3	2	3	なし	なし	なし	3
thash, ttag, tak, tpa, probe ¹			5	6	6	5				
getf ²	なし	なし	5	6	6	5	なし	なし	なし	5
setf ²	なし	なし	なし	なし	なし	6	6	6	6	なし
Fmac: fma, fms, fnma, fpma, fpms, fpnma, fadd, fnmpy, fsub, fpmpy, fpnmpy, fmpy, fnorm, xma, frcpa, fprcpa, frsqta, fpsqta, fcvt, fpvct	なし	なし	なし	なし	なし	4	4	4	4	なし
Fmisc: fselect, fcmp, fclass, fmin, fmax, famin, famax, fpmin, fpmax, fpamin, fpcomp, fmerge, fmix, fsxt, fpack, fswap, fand, fandcm, for, fxor, fpmerge, fneg, fnegabs, fpabs, fpneg, fpnegabs	なし	なし	なし	なし	なし	4	4	4	4	なし
整数側プレディケートの書き込み: cmp, tbit, tnat	1	0	なし	なし	なし	なし	なし	なし	なし	なし
FP 側プレディケートの書き込み: fcmp	2	1	なし	なし	なし	なし	なし	なし	なし	なし
FP 側プレディケートの書き込み: frcpa, fprcpa, frsqta, fpsqta	2	2	なし	なし	なし	なし	なし	なし	なし	なし
整数ロード ³	なし	なし	N	N+1	N+1	N	N	N	N	N
FP ロード ⁴	なし	なし	M+1	M+2	M+2	M+1	M+1	M+1	M+1	M+1
IEU2: move_from_br, alloc	なし	なし	2	2	3	2	なし	なし	なし	2
CR または AR ⁵ との間の移動	なし	なし	C	C	C	C	なし	なし	なし	C
pr への移動	1	0	2	2	3	2	なし	なし	なし	なし
間接移動 ⁶	なし	なし	D	D	D	D	なし	なし	なし	D

- これらの操作は L1D 上で実行されるため、L1D パイプラインおよび L2 パイプラインと相互作用する。これらの値は最小レイテンシであるが、この相互作用のために、実際のレイテンシはこれよりはるかに大きくなる場合がある。
- これらの操作は L1D 上で実行されるため、L1D パイプラインおよび L2 パイプラインと相互作用する。これらの値は最小レイテンシであるが、この相互作用のために、実際のレイテンシはこれよりはるかに大きくなる場合がある。
- N はヒットしたキャッシュのレベルによって異なる。L1D の場合は N=1、L2 の場合は N=5、L3 の場合は N=12 ~ 15、メイン・メモリの場合は N=約 180 ~ 225 である。これらの値は最小レイテンシであり、キャッシュのレベルが高いほど大きくなる。
- M はヒットしたキャッシュのレベルによって異なる。L2 の場合は M=5、L3 の場合は M=12 ~ 15、メイン・メモリの場合は M=約 180 ~ 225 である。これらの値は最小レイテンシであり、キャッシュのレベルが高いほど大きくなる。表中の「+1」の項目は、フォーマットの変換に 1 サイクル余分にかかることを示す。
- C の最良の場合の値は、アクセスするレジスタによって、2 ~ 35 サイクルの範囲である。EC および LC のアクセスは 2 サイクルである。FPSR および CR のアクセスは 10 ~ 12 サイクルである。
- D の最良の場合の値は、アクセスする間接レジスタによって、6 ~ 35 サイクルの範囲である。lreg の pr および rr は高速側にあるため、6 サイクル・アクセスになる。

本章では、スペキュレーティブ・ロード/ストア、プレディケート付きロード/ストア、浮動小数点ロード、プリフェッチなどのデータ操作に関する留意点を説明する。ロード・ヒント、データ・アライメント、ライト・コアレンシングに関する留意点も説明する。

5.1 データ・スペキュレーションと ALAT

`ld.a/ldf.a/ldfp.a`、`ld.c/ldf.c/ldfp.c`、`chk.a` で構成される一連の命令は、ロードとストアの間のメモリ・アドレスを動的に明確にする機能を持つ。アーキテクチャ的には、`ld.c` 命令と `chk.a` 命令のレイテンシは、結果を参照する側の命令に対して 0 サイクルである。したがって、`ld.c/ldf.c/ldfp.c/chk.a` とそれに対応する参照側の命令は、同じサイクル内にスケジューリングできる。ただし、`ld.c/ldf.c/ldfp.c/chk.a` が ALAT ミスになった場合は、レイテンシが発生する。

`ld.c`、`ldf.c`、または `ldfp.c` が ALAT ミスになると、L1 キャッシュへのアクセスが開始される。発行グループ内の他の命令は再実行される。この 8 サイクルのペナルティは、同じ発行グループ内に結果を参照する側の命令があったかどうかに関係なく、チェック・ロード以降に発行されるすべての操作に影響を与える。結果を参照する側の命令は、キャッシュ・レイテンシの増加の影響を受ける。L1 内でチェック・ロードが見つかった場合は、ペナルティは 8 サイクルだけで済むが、チェック・ロードが L2 内にある場合は、レイテンシはさらに大きくなる。

`chk.a` が ALAT ミスになると、リカバリ・コードへの分岐が実行される。Itanium® 2 プロセッサでは、分岐ターゲットは、ほとんどの場合、`chk.a` 命令に含まれているオフセットから計算できる。これによって、Itanium プロセッサで行われていた、オペレーティング・システムへのトラップを避けられる。`chk.a` が ALAT ミスになった場合のコストは、リカバリ・コードへの分岐に少なくとも 18 サイクル + リカバリ・コードのコスト + リターンである。修復コードへの実際の移行は 10 サイクル以内に行われるが、修復コードの最初の命令が完了するのに少なくとも 8 サイクルかかる。修復コードへの分岐が、L2 ITLB や L1I などのキャッシュ・レベルにヒットしなかった場合は、レイテンシは 8 サイクルより大きくなる。

Itanium 2 プロセッサの ALAT は、32 エントリ、フル・アソシアティブである。各エントリには、レジスタの番号、タイプ、物理アドレスの下位 20 ビットが格納される。このアドレスを使用して、競合する可能性のあるストアとの比較が行われる。レジスタのインデックスとタイプは、チェック操作をサポートする。ALAT にはアドレスの一部だけが保存されるため、ストア・アドレスと ALAT エントリのアドレスが異なっていても、下位 20 ビットが一致していると、偽りの競合が検出される可能性がある。さらに、`ld.c` または `chk.a` がストアのすぐ後に続く場合は、ALAT アドレスの比較は、20 ビットに満たないアドレスに基づいて行われる。この現象は、最小 4K のページ・サイズのサポートと、20 ビットの比較を実行するにはストア・アドレスとチェック・アドレスの両方をフルに変換する必要があることの結果である。表 5-1 に、ストアとチェックの間の距離と、比較されるアドレスのサイズを示す。

注：表 5-1 では、`ld.c` には `ldf.c` と `ldfp.c` も含まれる。

表 5-1. ALAT エントリの比較サイズ

距離	比較サイズ
st と ld.c が同じサイクル内にある。	12 ビット
st が ld.c より 1 サイクル先行する。	12 ビット
st が ld.c より 2 サイクル以上先行する。	20 ビット

表 5-1. ALAT エントリの比較サイズ (続き)

距離	比較サイズ
st と chk.a が同じサイクル内にある。	12 ビット
st が chk.a より 1 サイクル先行する。	12 ビット
st が chk.a より 2 サイクル以上先行する。	20 ビット

注： Itanium プロセッサでは、ストアと chk.a が同じサイクル内に発生すると、chk.a は常に失敗する。この制限は、Itanium 2 プロセッサには適用されない。

5.1.1 割り当て / 置換ポリシー

ALAT に新しいエントリが追加されたとき、どのエントリが置き換えられるかについての規則を、優先度の順に次に示す。

- 新しいエントリと同じレジスタ番号を持つエントリ
- 最初の無効なエントリ
- 有効なエントリは、ポート M0 および M1 に関連する前進ポイントに基づいて置き換えられる。この方法は、FIFO (First In - First Out) アルゴリズムによく似ている。

5.1.2 規則と特殊な場合

以下の規則と特殊な場合に注意する必要がある。

- Itanium アーキテクチャの定義では、ld.a 命令と ld.c 命令が同じターゲット・レジスタを持つ場合、両方の命令を同じサイクル内にスケジューリングすることは禁止されている。同様に、ld.a 命令と chk.a 命令が同じターゲット・レジスタを持つ場合、両方の命令を同じサイクル内にスケジューリングできない。ただし、2 つの命令の間隔を 1 サイクル以上空ければ、ALAT は正常に動作する。ldf.a と ldfp.a についても、同じ規則が適用される。
- ld.a でフォルトが発生した場合は、ALAT への書き込みは行われない。このようなフォルトについては、『インテル® Itanium® 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』を参照のこと。これらのフォルトには、データ・ページ不在フォルト、データ TLB フォルト、アライメントの合っていないデータ参照フォルトが含まれる。この場合、対応するこれ以降の ld.c または chk.a は、必ず ALAT ミスになる。
- ALAT セット命令と ALAT 無効化命令の両方が同じサイクル内で発行された場合は、ALAT のセットは実行されない。例えば、chk.a.clr rx と rx = ld.a[B] が同じサイクル内で発行された場合、ld.a[B] のアドレスは ALAT に入力されない。

5.2 スペキュレーティブ・ロード / ストアとプレディケート付きロード / ストア

スペキュレーティブなメモリ操作またはスペキュレーティブ入力を使用するメモリ操作は、次のように動作する。

- 通常のロード / ストアのソース・レジスタに NaT 値が含まれている場合は、レジスタ NaT 参照フォルトが発生する。
- スペキュレーティブ・ロードのソース・レジスタに NaT 値が含まれている場合は、NaT ビットがセットされ、0 の値が返される。

- スペキュレーティブ・ロードが TLB ミスになった場合、スペキュレーティブ・ロードの動作は、`psr.ic` ビットの設定によって決まる。このビットがクリアされている場合は、HPW ウォークが行われ、NaT ビットがセットされる。`psr.ic` がセットされている場合は、HPW ウォークが行われ、HPW ウォークの結果によって NaT の設定が決まる。HPW ウォークには数サイクルかかり、その間は発行グループ内のすべての命令がストールする。
- スペキュレーティブ・ロードでメモリ管理フォルトまたはデータ TLB ミスが発生した場合の動作は、`psr.ic` ビットと `dcr.dm` の設定によって決まる。`psr.ic=0` で `dcr.dm=1` の場合は、NaT ビットがセットされ、0 の値が返される。`psr.ic=1` または `dcr.dm=0` の場合は、スペキュレーティブ・ロードで HPW アクセスが開始され、変換が完了した場合はデータが返される。この HPW アクセスの結果、メモリ管理フォルトが発生した場合は、NaT ビットがセットされ、返されるデータ値は 0 になる。

注： スペキュレーティブ・ロードは、`ld.s` 命令だけに限られない。`lfetch` 命令は、通常はスペキュレーティブ命令であり、`ld.s` 命令と同じように動作するが、NaT ビットをセットしない点とデータを返さない点が異なる。`lfetch` 命令は、`.fault` コンプリータによって、非スペキュレーティブ命令に変更できる。

表 5-2 に、スペキュレーティブ・ロードがデータを返すか最終的にデスティネーション NaT ビットをセットするまでのレイテンシを示す。各例外による抑止のコストは、HPW のレイテンシに応じて、1 サイクル～数サイクルの範囲である。HPW に関連するこれらのペナルティは、スケジュールによって回避できないため、発行グループ内のすべての命令に影響を与える。また、抑止の原因になる例外が抑止される場合は、その例外が解決されない可能性がある。したがって、`chk.s` に到達しないループが実行されるたびに、抑止によるストールが発生するときがある。

表 5-2. コントロール・スペキュレーションのペナルティ

結果	ヒット/ミス	ペナルティ (早期抑止モード)	ペナルティ (後期抑止モード)
有効なデータを返す。	L1 DTLB ヒット	1	1
	L2 DTLB ヒット	4 + L2 のレイテンシ	4 + L2 のレイテンシ
	VHPT ヒット	5 (HPW ウォークなし)	20 + L2 のレイテンシ
NaT ビットをセットする。	NaT ソース	1	1
	L1 DTLB ヒット	2	2
	L2 DTLB ヒット	2 または 4 + L2 のレイテンシ	2 または 4 + L2 のレイテンシ
	VHPT ヒット	5 (HPW ウォークなし)	22 または 2 + L2 のレイテンシ
	VHPT ミス	5 (HPW ウォークなし)	20 + L2 のレイテンシ
	VHPT フォルト	5 (HPW ウォークなし)	17 + L2 のレイテンシ

5.3 浮動小数点ロード

浮動小数点ロードは、L1D 内にキャッシュされず、L2 によって直接に処理される。L1D のサイズと帯域幅は限られているため、このデータを L1D にキャッシュするのは有益でない。L2 の追加レイテンシを隠蔽するように、FP メモリ・アクセスをスケジューリングできるはずである。

浮動小数点ロードは、フォーマット変換のために、整数アクセスよりレイテンシのクロック数が増える。したがって、浮動小数点ロードが L2 にヒットした場合、ロードには 6 サイクルかかる。また、浮動小数点ロード・ペア命令 (倍精度と単精度の両方) も L2 キャッシュにアクセスするため、L2 にヒットすると仮定すると、ロード・ペア命令のレイテンシも 6 サイクルになる。

5.4 データ・キャッシュのプリフェッチ機能とロード・ヒント

Itanium アーキテクチャは、データをロードする時間と場所を制御する 2 つのソフトウェア機構を持っている。lfetch 命令を使用して、L1D、L2、または L3 キャッシュ内にデータを明示的にプリフェッチできる。また、データの局所性を強めるために、テンポラルなヒントを使用して、ロードされたデータが置かれるキャッシュ階層のレベルを制御できる。

5.4.1 lfetch のサポート

Itanium 2 プロセッサでは、lfetch は次のように処理される。

- lfetch.none は、例外が検出されない場合にのみ実行される。例外は報告されない。メモリ管理フォルトが検出された場合の lfetch 命令の動作は、5.2 節を参照のこと。
- lfetch.fault は、例外の有無に関係なく実行される。例外が検出された場合は、その例外の処理を OS に渡して命令を完了する。TLB ミスは、通常のロードと同じように解決される。
- lfetch が L1D ミスになり、L2 にヒットした場合は、lfetch テンポラル・ヒントに基づいて L1D キャッシュが割り当てられる。lfetch 命令のテンポラルな局所性の動作は、整数ロードの場合と同じである。
- lfetch タイプの命令が、1 次データ TLB ミスになり、2 次データ TLB にヒットした場合は、メイン・パイプラインはストールし、通常のロード操作と同じように 1 次データ TLB のフィルが行われる。lfetch が L2 DTLB ミスになった場合の動作は、5.2 節で説明した早期抑止モードと後期抑止モードのどちらを使用するかによって異なる。早期抑止モードでは、L2 DTLB ミスが発生すると、lfetch は中止される。後期抑止モードでは、lfetch が L2 DTLB ミスになると、HPW アクセスが開始される。この HPW アクセスが失敗すると、lfetch は中止される。ただし、両方のデータ TLB 内でミスになったときに HPW アクセスを開始するのは、lfetch.fault 命令だけである。
- lfetch.excl は、他のキャッシュ・レベルおよびシステム・バスに対するストアとして機能する。つまり、この操作は、ラインをキャッシュ内の M ステートに置く。データが実際に変更される確率が高い場合以外は、.excl コンプリータを使用してはならない。データが変更されない場合にこのコンプリータを使用すると、変更されていないデータが当該キャッシュからキャッシュ構造上に排出され、最終的には変更されていないメモリに排出される。
- Itanium アーキテクチャの定義により、キャッシュ不可メモリ上の位置に対する lfetch は、L2 キャッシュに到達しない。

注： lfetch 命令は、コアに対して特定のデータを返さないロード操作として機能する。したがって、メモリ階層内の任意の位置で通常のロードに適用される制限の大部分は、lfetch 命令にも同じように影響を与える。ただし、コンパイラが簡単に lfetch 命令を使用してパフォーマンスを向上できるように、lfetch 命令には制限が適用されない場合もある。

5.4.2 ロードのテンポラルな局所性を制御するコンプリータ

Itanium アーキテクチャは、データ・キャッシュ階層の管理に、メモリ局所性ヒントを使用する。Itanium 2 プロセッサは、t1、nt1、nt2、nta の 4 種類のメモリ局所性ヒントをサポートしている。Itanium 2 プロセッサは、非テンポラルなバッファをサポートしていない。非テンポラルな L2 アクセスは、バイアスされた置換を使用して、L2 内に割り当てられる。メモリ局所性ヒントは、次のように処理される。

- t1 ヒントは、通常のアクセスを表す。ロードの場合、ラインは L1D、L2、L3 に割り当てられる。ストアの場合、ラインは L2 および L3 に割り当てられ、L1D には割り当てられない。
- nt1 ヒント付きロードでは、ラインは L2 および L3 にのみ割り当てられる。また、ラインは L2 内で置換されるようにバイアスされる。L2 の LRU ビットを更新しないことで、そのラインがバイアスされる。これによって、ラインが置換される確率が高くなるが、そのラインが次に置き換えられるかどうかは保証されない。

- nt2 ヒント付きロードは、nt1 ヒント付きロードと同じように処理される。
- nta ヒント付きロードおよびストアでは、ラインは L2 内にも割り当てられ、置換されるようにバイアスされる。ラインは L3 には割り当てられない。

表 5-3 に、各種のヒントについて、L1D、L2、L3 がラインの割り当てと LRU の更新をどのように処理するかを示す。次の点に注意する。

- L1D はライトスルー・キャッシュであり、FP ロード/ストアをサポートしていない。
- L1D キャッシュ内の有効ビットの更新と L3 キャッシュ内の LRU ビットの更新は、ヒント・ビットとは無関係である。L2 LRU の更新だけが、非テンポラル・バッファの動作を模倣するようにバイアスされる。

表 5-3. プロセッサ・キャッシュ・ヒント

アクセス	ヒント	L1D		L2		L3	
		Alloc ¹	LRU ビットを更新するか?	Alloc	LRU ビットを更新するか?	Alloc	LRU ビットを更新するか?
lfetch	t1	行う	行う	行う	行う	行う	行う
	nt1	行わない	行わない	行う	行う	行う	行う
	nt2	行わない	行わない	行う	行わない	行う	行う
	nta	行わない	行わない	行う	行わない	行わない	行わない
整数ロード ²	t1	行う	行う	行う	行う	行う	行う
	nt1	行わない	行わない	行う	行う	行う	行う
	nta	行わない	行わない	行う	行わない	行わない	行わない
整数ストア ³	t1	行わない	行わない	行う	行う	行う	行う
	nta	行わない	行わない	行う	行わない	行わない	行わない
FP ロード	t1	行わない	行わない	行う	行う	行う	行う
	nt1	行わない	行わない	行う	行わない	行う	行う
	nta	行わない	行わない	行う	行わない	行わない	行わない
FP ストア	t1	行わない	行わない	行う	行う	行う	行う
	nta	行わない	行わない	行う	行わない	行わない	行わない

1. Alloc は、キャッシュ・ミスの発生時に当該キャッシュ・レベル内にエントリが割り当てられることを示す。
2. 整数ロードおよび FP ロード - t1、nt1、および nta 属性だけを使用できる。
3. 整数ストアおよび FP ストア - t1 と nta だけを使用できる。

注： これ以外の命令とヒントの組み合わせは、Itanium アーキテクチャでは使用できない。

5.4.2.1 ヒントの一般的な説明

各メモリ局所性ヒントについて以下に説明する。

.none: ロードはデータを読み込み、L1D と L2 の両方にロードする。

.nt1: このヒントは、参照されるデータを保持できる最初のキャッシュ・レベル内の非テンポラルな局所性を意味する。Itanium アーキテクチャでは、このヒントは、ロードがデータを読み込み、ラインを 1 次キャッシュに割り当てないように指示する。Itanium 2 プロセッサでは、このヒントがあると、整数キャッシュ・ミスの発生時にラインは L1D に割り当てられない。ラインがすでに L1D キャッシュ内にある場合は、そのラインは割り当て解除されない。

.nt2: このヒントは、命令を保持できる 2 番目のキャッシュ・レベル内の非テンポラルな局所性を意味する。Itanium 2 プロセッサでは、このヒントがあると、ラインに対する整数アクセスは L2 に割り当てられるが、そのラインの LRU 情報は更新されない(すなわち、そのラインは、特定のセット内で次に置き換えられるラインになる)。ラインがすでに L2 キャッシュ内にある場合は、そのラインは割り当て解除されない。

.nta: このヒントは、キャッシュ階層のすべてのレベル内の非テンポラルな局所性を意味する。Itanium 2 プロセッサでは、このヒントがあると、ラインは L2 に割り当てられるが、そのラインの LRU 情報は更新されない(すなわち、そのラインは、特定のセット内で次に置き換えられるラインになる)。このラインは、L3 キャッシュには割り当てられない。ラインがいずれかのキャッシュ内にある場合、そのラインは当該キャッシュから割り当て解除されない(ただし、コヒーレンスに関する理由で、ラインが割り当て解除されることがある)。

注: lfetch 命令を使用しても、L2 に影響を与えずに L3 にのみラインを割り当てる方法はない。

非テンポラルなデータは短時間で置換されるため、非テンポラルな L2 データを 1 ウェイで割り当てると、テンポラル・データ・ストリームの L2 データが置き換えられるときがある。L2 の 1 ウェイは、32KB を保持できる。このサイズは、1 つの .nt ストリームには十分な大きさであるが、2 つの非テンポラルなストリームを使用しようとする、1 つのストリームがもう 1 つのストリームで置き換えられる場合がある。

5.5 データ・アライメント

Itanium 2 プロセッサは、8 バイト境界を超える整数アクセスと 16 バイト境界を超える任意のアクセスを除いて、任意のアライメントのロード/ストア・アクセスをサポートしている。

psr.ac = 1 の場合は、アライメントの合っていないメモリ参照はすべてフォルトになる。

psr.ac = 0 の場合は、次の規則に従っていればフォルトは回避される。

- 整数ロードおよびストアは、8 バイト境界ウィンドウの範囲内でアライメントされていなければならない。
- すべての 4 バイトおよび 8 バイト FP ロード操作は、16 バイト境界ウィンドウの範囲内ではアライメントが合っていないくてもよい。
- すべての FP ロード・ペアは、自然境界にアライメントされていなければならない。つまり、単精度操作は 8 バイト境界、倍精度操作は 16 バイト境界、ldpr.8 は 16 バイト境界にアライメントされていなければならない。
- すべての 10 バイト FP ロードは、16 バイト・ウィンドウの範囲内ではアライメントが合っていないくてもよい。
- FP フィル/スピル命令は、16 バイト境界ウィンドウの範囲内でアライメントされていなければならない。
- FP ストアは、16 バイト境界ウィンドウの範囲内ではアライメントが合っていないくてもよい。
- セマフォ(cmpxchg、xchg、fetchadd) は、自然境界にアライメントされていなければならない。
- 8 バイト境界を超えるキャッシュ不可(UC、WC)アクセスは、すべてフォルトになる。

5.6 ライト・コアレスティング

フレーム・バッファに対するキャッシュ不可参照のパフォーマンスを上げるために、以前の世代の IA-32 プロセッサは、ライト・コアレスティング(WC)メモリ・タイプを定義していた。WC は、データ書き込みのストリームを結合し、1 つの大きなバス書き込みトランザクションにまとめられる。Itanium 2 プロセッサは、インテル® Pentium® III プロセッサで定義されたライト・コアレスティング

グを十分にサポートしている。Pentium III プロセッサと同様に、Itanium 2 プロセッサの WC ロードは、コアリング・バッファからではなく、メモリから直接に実行される。

Itanium 2 プロセッサは、WC アクセスにのみ使用される、別個の 2 エントリ 128 バイト・バッファ (WCB) を搭載している。ライン内の各バイトが 1 つの有効ビットを持つ。すべての有効ビットが真である場合、そのラインはフルと呼ばれ、プロセッサによって排出される。

5.6.1 WC バッファの排出条件

メモリに対する整合性を保証するために、WCB は以下の条件でフラッシュされる (両方のエントリがフラッシュされる)。表 5-4 に、プロセッサが Itanium ベースのシステム環境で動作しているときの排出条件を示す。

表 5-4. Itanium® 2 プロセッサの WCB 排出条件

排出条件	Itanium 命令
メモリ・フェンス (mf)	mf
WCB フラッシュのアーキテクチャ的な条件	
メモリ解放の順序づけ (op.rel)	st.rel, cmpxchg.rel, fetchadd.rel, ptc.g
フラッシュ・キャッシュ (fc) が WCB にヒット	排出する
フラッシュ・ライト・バッファ (fwb)	排出する
任意の UC ロード	排出しない ¹
任意の UC ストア	排出しない ¹
UC ロードまたは ifetch が WCB にヒット	排出しない ¹
UC ストアが WCB にヒット	排出しない ¹
WC ロード /ifetch が WCB にヒット	排出しない
WC ストアが WCB にヒット	排出しない ²

1. Itanium アーキテクチャは、UC ロード / ストア操作に対する WC バッファのコヒーレンスを要求していない。
2. WC ストアが WCB にヒットすると、ヒットしたエントリがフルでない場合は、そのエントリが更新される。ヒットしたエントリがフルである場合は、そのエントリがもう 1 つの WCB エントリより古いか新しいかのチェックが行われる。そのエントリの方が新しい場合は、古い方の WCB エントリが (フルでない場合でも) フラッシュされる。新しい方の WCB エントリは、後でフラッシュされる。ヒットした WCB エントリの方が古い場合は、そのエントリだけがフラッシュされる。

5.6.2 WC バッファのフラッシュ動作

すでに説明したように、Itanium 2 プロセッサの WCB は 2 つのエントリを持つ。2 つの WC エントリは、割り当てられた順序と同じ順序でフラッシュされる。つまり、エントリは書き込まれた順にフラッシュされる。このフラッシュ順序は、「行儀のよい」ストリームにのみ適用される。「行儀のよい」ストリームは、一度に 1 つの WC エントリに書き込み、最初の WC エントリがフルになってから 2 番目の WC エントリに書き込む。

プラットフォーム上の再試行または抑止機能がない場合、このフラッシュ規則は、「行儀のよい」ストリームでは、割り込みがあった場合でも、WCB エントリは常にプログラムに書かれた順序でフラッシュされるのを意味する。例えば、次の場合を考える。ソフトウェアが「行儀のよい」ストリームを発行したが、途中で割り込みが発生したために、WC エントリのうち 1 つが部分的にフィルされたとする。この WCB (部分的にフィルされたエントリを含む) は、OS カーネル・コードまたは他のプロセスによってフラッシュされるときがある。中断されたコンテキストが再開されると、残りのラインが送信され、もう 1 つのエントリのフィルが開始される。ただし、もう 1 つのエントリのフィルの途中で、再開されたコンテキストに対して再び割り込みが行われる可能

性がある。この場合は、割り込みが発生した時点で、両方のエントリが部分的にフィルされた状態になる。

ストリームが上記の「行儀のよい」ストリームの規則に従わない場合は、WC バッファのフラッシュ順序は無作為に決定される。

WCB の排出は、フル・ラインの場合、1 つの 128 ビット・バスのトランザクションによって、16 バイト・パッケージのストリームで実行される。部分的フル・ラインの場合、WCB は、適切なバイト・イネーブルを使用して、16 バイトまたは 32 バイト・トランザクションによって排出される。フラッシュが行われるときは、WC トランザクションがすべての外部バス動作の中で最高の優先度を与えられる。

5.7 レジスタ・スタック・エンジン

Itanium 2 プロセッサのレジスタ・スタック・エンジン (RSE) は、レイジー・モード (`ar.rsc.mode = 0`) でのみ動作する。その他のモード設定はすべて無視される。

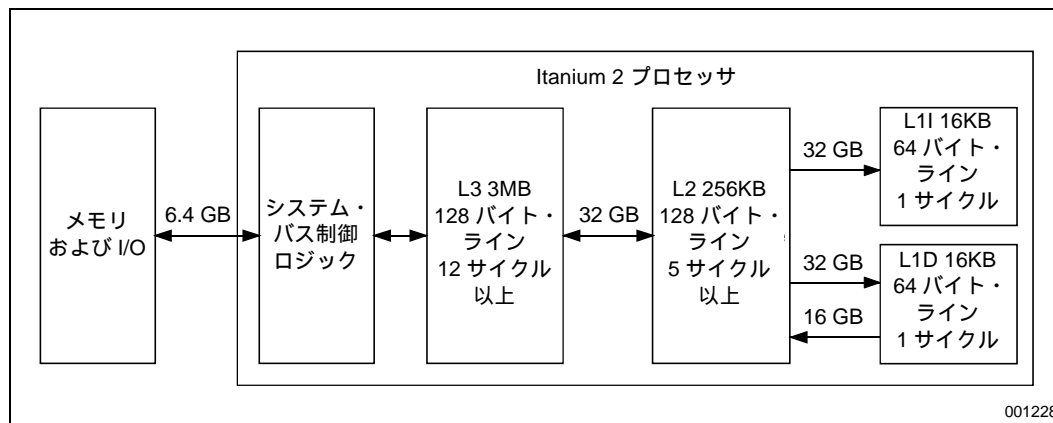
RSE は、各サイクルで最大 2 つのロードまたは 2 つのストアを実行できるが、ロードとストアの両方を同時に実行できない。

一般的に、RSE ロードおよびストアは L1D キャッシュにヒットし、L1D は RSE キャッシュ・ラインを L1D 内に保持できると考えられる。

Itanium® 2 プロセッサのメモリ・システムは、1 次命令キャッシュ (L1I)、1 次データ・キャッシュ (L1D)、ユニファイド 2 次キャッシュ (L2)、およびユニファイド 3 次キャッシュ (L3) の 3 レベルのキャッシュ構造を持つ。

本章では、L1D、L2、L3、システム・バスの機能について詳しく説明する。この情報が、最適化の推奨事項の基礎となる。本書で特に説明しないボトルネックを見つけるには、本章の内容をよく理解する必要がある。第 9 章「Itanium® 2 プロセッサ向けの最適化」では、Itanium 2 プロセッサのメモリ・サブシステム向けにソフトウェアを最適化する際の重要な推奨事項を示す。

図 6-1. Itanium® 2 プロセッサの 3 レベル・キャッシュ階層



Itanium 2 プロセッサは、命令参照用とデータ参照用の 2 レベルのトランスレーション・ルックアサイド・バッファ (TLB) を使用する。プロセッサは、命令用に 1 次命令 TLB (L1 ITLB) と 2 次命令 TLB、データ用に 1 次データ TLB (L1 DTLB) と 2 次データ TLB を搭載している。

Itanium 2 プロセッサは、仮想メモリのサポートに関する Itanium アーキテクチャの必要な機能をすべてサポートしている。表 6-1 に、Itanium 2 プロセッサがサポートする仮想メモリ機能を示す。

表 6-1. Itanium® 2 プロセッサによる仮想メモリのサポート

仮想メモリ	Itanium 2 プロセッサがサポートするサイズ
ページ・サイズ	4K、8K、16K、64K、256K、1M、4M、16M、64M、256M、1G、4G バイト
物理アドレス	50 ビット
仮想アドレス	64 ビット
領域レジスタ	8 個のレジスタ (各レジスタは 24 ビット)
保護キー・レジスタ	16 個のレジスタ (各レジスタは 24 ビット)

6.1 トランスレーション・ルックアサイド・バッファ

表 6-2 に、Itanium 2 プロセッサの TLB の主な機能を示す。命令 TLB とデータ TLB の機能はほぼ同じである。1 次 TLB は 1 次命令キャッシュおよび 1 次データ・キャッシュと密接に結合されている。これによって、L1 キャッシュに 1 サイクルでアクセスできるが、1 次 TLB ミスがあると必ず 1 次キャッシュ・ミスが発生する。

表 6-2. 命令 TLB とデータ TLB の主な機能

	命令 TLB	データ TLB
構造	L1 ITLB、L2 ITLB	L1 DTLB、L2 DTLB
エントリ数	32、128	32、128
アソシアティビティ	フル、フル	フル、フル
1 次ミスのペナルティ	2 サイクル	4 サイクル

6.1.1 命令 TLB

L1 ITLB は、32 エントリ、フル・アソシアティブ、デュアル・ポート・バッファである。1 つのポートは、正規の命令フェッチにのみ使用される。もう 1 つのポートは、命令プリフェッチ、スヌープ、TLB パージの間で共有される。L1 ITLB は、十分な情報、領域レジスタ、保護キーを持っているため、より大きな L2 ITLB の厳密なサブセットである必要はない。

L1 ITLB のページ変換エントリが置き換えられると、ビクティムにされたページからの L1I キャッシュ内のすべてのエントリは無効にされる。ビクティム・エントリは、真の LRU を使用して決定される。L1 ITLB が直接にサポートするページ・サイズは、4KB だけである。その他のページ・サイズは、大きなページの各 4K バイト・セグメントが参照されるたびに追加の L1 ITLB エントリを割り当てると、間接的にサポートされる。

L2 ITLB は、128 エントリ、フル・アソシアティブ、シングル・ポート・バッファである。L2 ITLB のうち最大 64 エントリが、トランスレーション・レジスタ (TR) として割り当てられる。TR は、実質的には L2 ITLB 内にロックされた変換エントリであるため、LRU 置換ポリシーは適用されない。L2 ITLB は、4KB、8KB、16KB、64KB、256KB、1MB、4MB、16MB、64MB、256MB、1GB、4GB のページ・サイズを直接にサポートしている。

デマンド・フェッチの場合、L1 ITLB ミス (および対応する L1I キャッシュ・ミス) のペナルティを小さくするために、L1 ITLB と L2 ITLB は並行してアクセスされる。命令アクセスが L1 ITLB ミスになり、L2 ITLB にヒットした場合は、L2 ITLB から L1 ITLB にページ情報を転送するために、1 次命令キャッシュへのアクセスに (2 次キャッシュのレイテンシと並行して) 2 サイクルのペナルティが発生する。L1 ITLB ミスがあると L1I キャッシュ・ミスが発生するため、上位のキャッシュまたはシステム・メモリから命令にアクセスする必要があるため、通常はペナルティはさらに大きくなる。

6.1.2 データ TLB

L1 DTLB は、32 エントリ、フル・アソシアティブ、デュアル・ポート・バッファである。L1 DTLB は整数ロード操作だけをサポートするため、必要なポートは 2 つだけである。L1 ITLB とは異なり、L1 DTLB は保護情報やページ属性情報を持たない。したがって、L1D ヒットの場合、L1 DTLB は L2 DTLB と並行してアクセスされ、L2 DTLB の厳密なサブセットでなければならない。

L1 DTLB のページ変換エントリが置き換えられると、ビクティムにされたページからの L1D 内のすべてのエントリは無効にされる。L1 DTLB は、4KB の固定ページ・サイズを持つ。これより大きいページ・サイズは、大きなページの 4KB 部分として追加の L1 DTLB エントリを割り当ててサポートされる。

L2 DTLB は、128 エントリ、フル・アソシアティブ、4 ポート・バッファである。整数ロード、ストア、浮動小数点ロードのすべての組み合わせを並行して参照するために、4 つのポートが必要である。整数ロードは、保護情報とページ属性情報を L2 DTLB に依存する。その他のアクセスは、仮想アドレスから物理アドレスへのマッピング、保護情報、ページ属性を L2 DTLB から取得する。

L2 DTLB のうち最大 64 エントリが、TR として割り当てられる。TR は、実質的には L2 DTLB 内にロックされた変換エントリであるため、LRU 置換ポリシーは適用されない。L2 DTLB は、4KB、8KB、16KB、64KB、256KB、1MB、4MB、16MB、64MB、256MB、1GB、4GB のページ・サイズを直接にサポートしている。

ストアまたは浮動小数点アクセスが L1 DTLB ミスになっても、L1 DTLB ミスによるペナルティは発生しない。整数ロードが L1 DTLB ミスになり、L2 DTLB にヒットした場合は、L2 DTLB から L1 DTLB にデータを転送するために、(L2 キャッシュ・レイテンシ以外に) 4 サイクルのペナルティが発生する。また、L1 DTLB ミスになったロード・アクセスは、L1D にヒットしない。

6.2 ハードウェア・ページ・ウォーカー

ハードウェア・ページ・ウォーカー (HPW) は、第 3 レベルのアドレス変換機構である。HPW は、仮想ハッシュ・ページ・テーブル (VHPT) からのページ・ルックアップを実行するエンジンである。L2 DTLB ミスまたは L2 ITLB ミスが検出されると、HPW は、(必要に応じて) L2 キャッシュ、L3 キャッシュ、最後にメモリにアクセスし、ページ・エントリを取得する。L2、L3、またはメモリ内でページ・エントリが見つからない場合は、割り込みが生成されて、ソフトウェア・ハンドラが呼び出され、アドレス変換を完了する (要求する側の命令が例外を据え置いていない場合)。HPW は、データ TLB ミスを処理しているとき、新しい命令 TLB ミスを受け入れる。同様に、命令 TLB ミスを処理しているとき、新しいデータ TLB ミスを受け入れる。ただし、HPW は、データ TLB ミスと命令 TLB ミスを同時に処理できない。要求は実質的にはシリアル化される。

キャッシュ・アクセスは、TLB の変換が完了するまで待たなければならない。

- L1D は、L1 DTLB と L2 DTLB の両方に並行してアクセスする。
- L1I アクセスは、L1 ITLB のルックアップだけを必要とする (L1 ITLB ミスのときは、L2 ITLB のルックアップが必要になる)。
- L2/L3 データ・アクセスは、L2 DTLB のルックアップだけを必要とする。
- L2/L3 命令アクセスは、L2 ITLB のルックアップだけを必要とする。

L2 DTLB ミスまたは L2 ITLB ミスが発生すると、HPW ルックアップが実行される。HPW に移行するための最小ペナルティを表 6-3 に示す。HPW ルックアップは、L1D キャッシュの参照やファイルを行わない。

L2 DTLB ミスまたは L2 ITLB ミスは、L1D または L1I 内にもミスがあることを意味する。したがって、表 6-3 のペナルティは、HPW ウォークのレイテンシに最良の場合の L2 キャッシュ・レイテンシを加算した値である。

表 6-3. 最良の場合の HPW ペナルティ

イベント	ペナルティ (サイクル)
L2 内のヒット	25
L2 内のミス、L3 内のヒット	31
L2 と L3 の両方でミス	OS トラップ

6.3 キャッシュのまとめ

表 6-4 は、Itanium 2 プロセッサのオンチップ・キャッシュの主なパラメータをまとめたものである。

表 6-4. キャッシュのまとめ

	L1I	L1D	L2	L3
サイズ	16KB	16KB	256KB	3MB または 1.5MB
アソシアティブ ティ	4 ウェイ	4 ウェイ	8 ウェイ	12 ウェイ
ライン・サイズ	64 バイト	64 バイト	128 バイト	128 バイト
レイテンシ	1 サイクル	1 サイクル	最小 5 サイクル (整数ロード) 最小 6 サイクル (浮動小数点ロード) 7 サイクル + ROT 段 階の 6 サイクルの ストール・ペナルティ (命令ロード)	最小 12 サイクル (ロード)
タグ読み出し帯域 幅	2/ サイクル	4/ サイクル	4/ サイクル	1/ サイクル
データ読み出し帯 域幅	1 × 32B/ サイクル	2 × 8B/ サイクル	4 × 8B/ サイクル	1 × 32B/ サイクル
データ・バンク	なし	8 バイト/バンク (ストア専用)	16 バイト/バンク	なし
書き込み帯域幅	なし	2 × 8B/ サイクル	4 × 16B/ サイクル	1 × 32B/ サイクル
フィル帯域幅	64 バイト アセンブリ 2 サイク ル 書き込み - 1 サイクル	64 バイト アセンブリ 2 サイク ル 書き込み - 1 サイクル	128 バイト アセンブリ 4 サイク ル 書き込み - 1 サイクル	128 バイト (4 サイクル)
未処理のミス	7 つのプリフェッチ	8 つの個別ライン	16 の個別ライン	22 (L2 と共有される 16 の読み出し、6 つ の書き込み)
ライン・サイズ	64 バイト	64 バイト	128 バイト	128 バイト

6.4 1 次命令キャッシュ

1 次命令キャッシュ (L1I) は、16KB、4 ウェイ・セット・アソシアティブ、物理アドレス指定キャッシュ (ライン・サイズ 64 バイト) である。最小仮想ページを表す下位の仮想アドレス・ビット 11:0 は変換されず、キャッシュのインデックスとして使用される。L1I は、1 サイクルおきに 1 回ずつ 64 バイト・ラインをフィルできる。L1I は、オンデマンド・フェッチ・ミスブロックするが、プリフェッチ・ミスについてはノンブロッキングであり、L2 キャッシュに対して最大 7 つのプリフェッチを実行待ちにできる。

L1I は、1 サイクル当たり 32 バイトの読み出しレートを維持し、1 サイクル当たり 2 バンドルのフェッチ・レートをサポートする。フロントエンドは、常に、アライメントの合った 32 バイトのバンドル・ペアを L1I からフェッチする。32 バイトのバンドル・ペアの始点ではなく中間に対する分岐の場合は、第 2 バンドルだけがフェッチされる。したがって、L1I からの最大のフェッチ帯域幅を利用するには、分岐ターゲットが 32 バイト境界にアライメントされていなければならない。

タグ配列はデュアル・ポートである。1つのポートは命令デマンド・フェッチ専用であり、もう1つのポートはキャッシュ・スヌープと命令プリフェッチによって共有される。キャッシュ・スヌープはプリフェッチより高い優先度を持つ。データ配列はデュアル・ポートであり、1つの読み出しポートと1つのフィル・ポートを持つ。さらに、LIIは、読み出しとフィルを同時に行えるように設計されている。したがって、LIIミスが結局LIIに書き込めないようなイベントは、ほとんど存在しない。

6.5 命令ストリーム・バッファ

Itanium 2 プロセッサの命令ストリーム・バッファ (ISB) は、LII と L2 の間に置かれている。ISB は、LII のライン・フィル・バッファとして機能し、命令プリフェッチ機能を補助する。ISB は、8つの64バイト・キャッシュ・ラインと8つのダブルバンドル・ペアを含むフル・アソシアティブ・バッファである。

L2 から返される LII ライン (デマンド・ミスまたはプリフェッチ) は、すべて ISB 内にストアされる。返されるキャッシュ・ラインがデマンド・ミスである場合、そのラインは命令パイプラインに転送され、LII 内に移動される。このキャッシュ・ラインは、アイドル期間になるまで ISB 内にとどまり、それから LII に移動することも、LII に移動する前にピクティムにされることもある。ただし、LII は読み出しとフィルが同時に行えるため、ISB エントリの存続期間は非常に短い。ISB エントリがピクティムにされるのは非常にまれである。

ISB は LII と並行してアクセスされる。ISB ヒットは、LII ヒットと同じレイテンシを持つ。ターゲット・ラインが ISB と LII の両方にヒットした場合は、ISB 内のラインが無効にされる。

6.6 1 次データ・キャッシュ

1 次データ・キャッシュ (L1D) は、マルチポート、16KB、4 ウェイ・セット・アソシアティブ、物理アドレス指定キャッシュ (ライン・サイズ 64 バイト) である。L1D は、ノンブロッキング、インオーダー・キャッシュである。最小仮想ページを表す下位の仮想アドレス・ビット 11:0 は変換されず、キャッシュのインデックスとして使用される。

L1D は、2つの専用ロード・ポートと2つの専用ストア・ポートを持つ。これらのポートは固定ポートであるが、発行ロジックは発行グループ内のロードとストアを移動して、適切なメモリ・ポートに送り込める。ロード・ポートはデュアル・ポートであり、任意の2つのロード・アドレスを競合なしに並行して読み出せる。ストアは、幅 8 バイト、8 グループの L1D データ配列にアクセスする。複数のストアが競合する可能性があるが、これらの競合がパフォーマンスに与える影響を制限する特殊なハードウェアが用意されている。

L1D のアクセス・レイテンシは 1 サイクルである。ただし、他のロード操作のアドレスのために使用される場合 (つまり、ポインタ追跡)、レイテンシは 2 サイクルになる。L1D は、ライトアロケートなしのライトスルー・ポリシーを使用する。すべてのストアは、L1D にヒットするかミスになるかを問わず、L2 キャッシュに送られる。ストアが L1D にヒットした場合は、L1D を更新するためのデータ配列が使用可能になるまで、そのデータはストア・バッファに保持される。これらのストア・バッファは、ストア・データをマージし、それを制限なしでその後のロードに転送できる。ロード・ミスになったときは、テンポラルなヒント、ロードのタイプ、使用可能なリソースに従って、L1D の割り当てが行われる。

L1D は、整数データ・バスに緊密に統合されている。すべての整数ロードは、L1D を通して、レジスタ・ファイル/バイパス網にデータを返さなければならない。したがって、整数 L1D ミスも、L2、L3、またはメモリによって処理された後、整数レジスタ・ファイルへの L1D データバスを使用する。このとき、同じ L1D データバスを要求するコア・ロードはすべてブロックされる。

浮動小数点ロードは L1D にアクセスしないため、4つのメモリ・ポートのどれでも制限付きで発行できる。浮動小数点ロード・ペアと、ALAT との相互作用を伴う任意の浮動小数点ロードは、ロード・ポートだけがディスペーサルできる。lfetch 命令はデータをコアに配布しないが、2つのロード・ポートだけがこの命令を発行できる。これは、lfetch によって L1D のフィルが行われる場合があるが、この機能は 2つのロード・メモリ・ポートだけがサポートしているためである。

アライメントの合っていない整数ロードが 8 バイト境界を超えた場合、アライメントの合っていないデータ参照の例外が発生する。サポートしているアライメントの詳細は、5.5 節「データ・アライメント」を参照のこと。

6.6.1 L1D ロード

コア・ロード要求が L1D にアクセスするとき、L1D データ配列へのアクセスが L1D ヒットになった場合は、データをコアに配布する。ヒットになるには、L1 DTLB、DTLB、L1D タグ内に対応するエントリが必要である。ロードがミスになるか、強制的に L1D ミスになった場合は、要求は (リソースが十分にあれば) L2 に渡され、L1D のフィルが行われる。浮動小数点ロードと順序づけされた操作は、強制的に L1D ミスになるが、L1D のフィルは行われない。

L1D のリソースは、L2 に対する実行待ちの L1D フィル要求を最大 8 つまで保持できる。9 個以上のミスが未処理になっている場合は、その次のミスは L2 に渡され、L1D のフィルは行われない。同じ L1D ラインに 2 つ以上のアクセスがあり、それらが L1D ミスになった場合は、L1D のフィルを要求するアクセスは 1 つだけである。その他のアクセスは、処理のために L2 に渡され、その L1D ラインのフィル要求を行わない。

6.6.2 L1D ストア

L1D はライトスルー・キャッシュであるため、すべてのストア要求は L2 キャッシュに渡される。ストアは、L1D ミスになっても L1D に影響を与えない。ただし、ストアがヒットした場合は、L1D は、その後のロードが新しいデータを参照できるように、データ配列を更新しなければならない。この処理のために、ストア・データがレジスタから読み出され、L1D パイプラインに送られる。各ストア・パイプライン (M2/M3) は、別々のストア・リソースと制御ロジックを持つ。

L1D データ配列を更新するデータの準備ができた時点で、競合がなければ、L1D データ配列が更新される。しかし、同時に他の操作がデータ配列に書き込む場合 (例えば、L1D のフィルまたはロードが同じ 8 バイト・バンクにアクセスしたり、ストアが別の M ポート上で同じバンクにアクセスする場合)、必要な更新ができなくなる。この場合は、ストア・データはストア・バッファに移され、データ配列が使用可能になるのを待つ。ストア・バッファは、同じ幅 8 バイトの L1D データ・バンクにアクセスする、新しいストアのコアリングができる。ストア・バッファがデータ配列を更新できないとき、コアリングできない新しいストアがストア・バッファを要求する場合は、L1D パイプラインがストールし、その間にストア・バッファがデータを排出する。

この構造では、同じグループをターゲットとするストアは、同じ L1D パイプラインに発行する方がよい。例えば、バンク 0 に対するアクセスはすべて M2 に発行し、バンク 1 に対するアクセスはすべて M3 に発行するとよい。データ配列を更新するとき、M2 と M3 が競合しないため、遅延なしに配列を更新できる。

6.6.3 ロードとストアの留意点

一部のメモリ要求は、時間の間隔があっても、互いに影響を与える場合がある。この項では、ヒットとミスの両方について、こうした相互作用を説明する。

6.6.3.1 ロード / ロードの競合

L1D は真のデュアル・ポート・キャッシュであるため、L1D ロード・ヒットは互いに競合しない。しかし、L1D ロード・ミスは、バンクの競合のために L2 上で競合することがある。レイテンシを短縮する必要がある場合は、同じ発行グループ内の複数のロードが同じ L2 バンクにアクセスしないように、特に注意する必要がある。すなわち、L2 に対するアクセスについては、A[7:4] は一意でなければならない。

L1D への発行待ちのロードが、L2/L3 またはシステム・バスから返される、より以前のロードに割り込まれた場合は、さらに複雑なロード / ロードの競合が発生する。この場合は、古い方のロードが優先され、新しい方のロードが待たなければならない。このようなイベントは予測しにくいいため、コードによって避けることが難しい。ただし、L2 キャッシュは、1 サイクル内に返される整

数ロードが1つだけである場合、常に M1 ポートを使用する。したがって、ロードのために M1 ポートを使用しないことで競合を回避できる。ただし、クリティカル・パス内では、この方法を使用してはならない。

ロードと、L2 データ・パスを使用してコアに情報を伝達する特殊な要求の間にも、これと同様の競合が発生する。これらの要求とは、probe、thash、ttag、tpa、tak 命令である。

6.6.3.2 ロード / ストアの競合

ロードとストアの競合は、ロードとストアのどちらが先に発生したかによって、全く異なる影響を与える。発行グループは本質的に並列的であるが、ロードとストアは発行グループ内の位置に従って順序づけられる。

ロードがストアに先行し、ロードがヒットになった場合は、競合は発生しない。しかし、ロードがストアに先行し、ロードとストアの両方が L1D ミスである場合は、大きな影響を与える。この場合、ロードは L1D ミスになり、L1D のフィルを要求しそうである。ロードに関連するフィルより前に、L1D によってストアが検出された場合は、ストアは L1D ミスになる。したがって、このストアは、L1D に関連するフィル・バッファ・エントリを無効にし、L1D のフィルを中止させる。この処理が必要なのは、ストアが L1D フィルより前に入力データを更新する機会がないためである。Itanium 2 プロセッサは、後続するロードが先行するストアを参照するように保証しなければならないため、フィルはキャンセルされ、キャッシュ・ラインへのストアのマージは L2 によって処理される。ストアより前にフィルが行われた場合は、フィルが実行され、L1D の通常のストア更新が行われる。この項の説明は、ロードとストアが A[49:6]、すなわち L1D キャッシュ・ライン全体を共有している場合に適用される。

6.6.3.3 ストア / ロードの競合

ストアがロードに先行する場合は、ストア・データはロードによって参照されなければならない。これらの要求が L1D ミスになった場合は、L2 がこの参照可能性を保証する。これらの操作が L1D にヒットした場合は、ロードに対する応答は、共通のアドレス・ビットと、ストアからロードまでの間隔が何サイクルかによって決まる。

表 6-5 に、ストア / ロードの競合のさまざまなペナルティを示す。ペナルティは、ロードがストアと同じデータにアクセスするのか、ストア・データの一部にアクセスするのか、ストアに全く依存しないのかによって異なる。

表 6-5. ストアからロードへのフォワーディングのペナルティ

ストアからロードまでの間隔	ストアの内部をロード	ストアの外部をロード	アドレスの比較
0 サイクル	17 サイクル	17 サイクル	11:2
1 サイクル	3 サイクル	5 サイクル	11:2
2 サイクル	3 サイクル	3 サイクル	49:2
3 サイクル	1 サイクル	3 サイクル	49:2

5 サイクルのペナルティと 17 サイクルのペナルティは、ストア要求とロード要求の両方が L2 に渡され、L2 の競合状態に直面するために発生する。3 サイクルのペナルティと 1 サイクルのペナルティは、L1D がロード要求を再循環するために発生する。再循環の間に、L1D はデータ配列をストア・データで更新し、ストアがなかったかのようにロードを実行できる。競合を避けるには、ストアからロードまでの間隔が 4 サイクル以上必要である。

6.6.3.4 整数アクセスと浮動小数点アクセスの相互作用

浮動小数点ロードと浮動小数点ストアは、L1D を無視して直接 L2 に渡される。L1D キャッシュ内のラインに対する浮動小数点ストアが発生すると、その L1D ラインは無効にされる。整数データと浮動小数点データが同じ L1D キャッシュ・ラインを共有している場合、これによって問題が起

こるときがある。この状況は、整数データと浮動小数点データの両方がスタック内にある場合や、Cプログラムの同じ構造体の一部として存在する場合に起こり得る。1つの整数値と1つの浮動小数点値が、64バイトにアライメントされた同じブロックを共有している場合を考える。整数ロードが行われると、このラインがL1Dに入れられる。その後の浮動小数点ストアは、L2に書き込み、このL1Dラインを無効にする。したがって、次にこの整数値をロードすると、L1Dミスが起こる。

この問題を回避するには、ストアの発行後に `lfetch` を使用してラインをL1Dに戻すか、あるいは整数アクセス時に `.nt1` ヒントを使用してL1Dのフィルを防ぎ、L2のレイテンシに合わせてアクセスをスケジューリングすればよい。

6.6.3.5 ストア/ストアの競合

L1Dは、ロード用ポートは真のデュアル・ポートであるが、ストア用ポートは疑似デュアル・ポートである。つまり、2つのストアが、データ配列内の全く同じ位置を同時に更新はできない(6.6.2項「L1Dストア」を参照)。ストア・バッファがコアレスリング機能を持つため、ほとんどのストア/ストアの競合は防止される。ただし、2つのストアが同じL1Dバンクを同時に更新しようとする競合が発生する。競合が発生した場合は、新しい方のストアはストア・バッファに移され、L1Dパイプラインに影響を与えずに、後でL1Dデータ配列を更新できる。ただし、ストア・バッファが一杯の場合は、ストア・バッファが排出されるまで、L1Dはストールする。

6.6.4 L1Dミス

L1D要求がミスになると、L2が新しい要求を保持するのに十分なリソースを持っていれば、その要求はL2に渡される。このリソースには、少なくとも1つのL2OzQエントリとL2データ・エントリ(ストアの場合)が含まれる。これらのリソースが必要なときに使用できない場合は、これらのリソースが使用可能になるまで、その操作とその発行グループ内の他のすべての操作がストールする。

L2制御ロジックは、要求がL1DパイプラインからL2に送られたときにL2OzQエントリを使用できるように、L2OzQエントリを予約しておく。制御ロジックは、3サイクルの両義性の各サイクルにつき、4つのエントリを予約する。その結果、ストリーミングなどの場合、32個のL2OzQエントリのうち約20個のみが使用可能になる。Itanium 2プロセッサは、L2に送らなければならない要求がL1D内にあり、かつL2が一杯になっているときにのみ、L1Dパイプラインをストールさせる。

一部のロード命令は、強制的にL1Dミスになる。浮動小数点ロードは、常にL1Dミスになる。順序づけされたロード(`ld.acq`)は、常にL1Dミスになる。さらに、順序づけされたロードには、順序づけされたロードが参照可能なことをL2が指示するまで、それ以降のすべてのロードも(アドレスや順序づけの制約に関係なく)強制的にL1Dミスになる特徴がある。

6.7 2次ユニファイド・キャッシュ

2次ユニファイド・キャッシュ(L2)は、ユニファイド、256Kバイト、8ウェイ・セット・アソシアティブ・キャッシュ(ライン・サイズ128バイト)である。L2タグは真の4ポート型であり、L1Dパイプラインの一部としてアクセスされる。L2はライトバック・ポリシーとライトアロケート・ポリシーを使用する。L2に対する整数アクセスのレイテンシは、5サイクル、7サイクル、または9サイクル以上である。浮動小数点アクセスには、6サイクル、8サイクル、または10サイクル以上かかる。これには浮動小数点変換ステージが含まれる。L1DミスがL2にヒットし、L2の5サイクルのバイパスを使用する場合、7サイクルのレイテンシと6サイクルのストール・ペナルティが発生する。

L2キャッシュは、ノンブロッキング、アウト・オブ・オーダー・キャッシュである。L2にアクセスするすべてのメモリ操作(L1Dミスとすべてのストア)は、L2タグをチェックし、L2OzQと呼ばれる32エントリのキューイング構造内に割り当てられる。すべてのストアは、後でL2データ配列を更新するデータを保持するために、24個のL2データ・エントリのうち1つを必要とする。

競合が解決され、リソースが使用可能になると、操作が一度に最大 4 つまで発行され、L2 データ配列にアクセスする。L1I 命令ミスも L2 に送られるが、命令プリフェッチ FIFO (IPF) にストアされる。L2 OzQ 要求と IPF 要求の間で、データ配列と L3/ システム・バスへのアクセスを求めてアービトレーションが行われる。

L2 データ配列は、幅 16 バイトのバンクを 16 個持つ。これによって、複数の操作が、それぞれ異なるバンクに同時にアクセスできる。L2 は FP ユニット / レジスタ・ファイルに対する 4 つのデータパスを持っているため、浮動小数点ロードは、L2 OzQ から発行され、一度に 4 つずつ L2 データ配列にアクセスできる。L2 は、整数ユニット / レジスタ・ファイルに対する直接のデータパスを持たない。整数ロードは、L1D を介してデータを配布する (L1D は整数ユニット / レジスタ・ファイルに対する 2 つのデータパスを持つ)。ストアは、L2 OzQ から発行され、一度に 4 つずつ L2 データ配列にアクセスできる (各ストアが異なるバンクにアクセスする場合)。

L2 から L1D および L1I へのフィル・パスの幅は、32 バイトである。L3 またはメモリから L2 へのフィル帯域幅は、1 サイクル当たり 32 バイトである。L2 フィル・バッファに 32 バイト・データが 4 つ累積されると、128 バイト・キャッシュ・ラインが 1 サイクルで L2 に書き込まれ、タグ配列とデータ配列の両方が更新される。

注： キャッシュ・ラインの置換には、NRU アルゴリズムが使用される。

L2 キャッシュは、L1D キャッシュと L1I キャッシュを含まない。L2 は各ラインのステート情報を保持し、ストアされるデータが、修正済み (M)、排他的 (E)、共有 (S)、無効 (I)、またはベンディング・アップデート (P) であるかどうかを監視する。これにより、L2 は、MESIP プロトコルを使用して、キャッシュ・コヒーレンスを維持し、ビクティムにされるラインを保持できる。

6.7.1 L2 に対する L1D 要求

L1D は、1 サイクルごとに最大 4 つの要求を L2 に発行できる。これらの要求は、L1D ロード / ストア・ミス、L2 再循環、L2 フィル、命令フェッチ、またはスヌープである。L2 タグは真の 4 ポート型であり、L1D バイブラインの一部である。これによって、4 つの L1D ロード要求またはストア要求がすべて L2 タグにアクセスでき、L2 キューイング構造内に割り当てられる前に L2 ヒットか L2 ミスかを判定できる。この機能によって、L2 ミスを識別し、迅速にシステム・バス / L3 に渡すことができる。また、L2 ヒット要求のレイテンシも小さくなる。

L1D のすべてのロード要求、ストア要求、セマフォ要求は、L2 OzQ に入れられる。L1D を介して L2 に発行されるすべての L1I 命令ミスは、IPF に入れられる。IPF 内の要求と L2 OzQ 内の要求の間で、L2 データ配列およびシステム・バス / L3 へのアクセスを求めてアービトレーションが行われる。L1D から送られるその他の要求 (スヌープやフィルなど) は、テンポラルなものであり、キューに入れられない。

L2 データ配列の読み出し (ロード) 操作は、L2 データ配列の書き込み (ストア) より 2 サイクル前に行われる。データ配列のロード / ストアの競合を判定するときは、このタイミング関係が重要である。

L2 は、L2 ミスを保持する 16 個のフィル・バッファを持つ。L2 ミスがあるたびに、修正済みのデータが排出される。L2 は、ビクティム・データを保持する 16 個のビクティム・バッファを持つ。ただし、一度に保持できる未処理の L2 ビクティムは 6 つまでである。

6.7.2 L2 OzQ

L2 がノンブロッキングであるのは、L2 OzQ によって可能になる。この構造は、L1D が処理できなかった最大 32 個の操作を保持できる。これらの操作には、すべてのストア、セマフォ、キャッシュ不可アクセス、L1D ロード・ミス、L1D の未解決の競合が含まれる。L2 キャッシュの設計では、競合がない場合、L2 OzQ エントリが 32 個より少なくとも、最大数の L1D 要求を保持できるはずである。しかし実際には、L2 内には多くの競合が発生する。競合があると、要求が L2 OzQ 内にとどまる時間が長くなる。L2 OzQ エントリの数を増やせば、L2 が競合を解決している間に、

L1D パイプラインはヒットの処理を続けて、L2 に対する次の要求を送り出せる。競合があると、L2 のレイテンシが大きくなり、L2 レイテンシの予測が不可能になる。

6.7.2.1 L2 OzQ の割り当てと解放

L2 OzQ の制御ロジックは、直前のサイクルで割り当てられた最後のエントリから始めて、1 サイクルごとに最大 4 つの連続するエントリを割り当てる。使用可能なエントリが足りない場合は、操作がそれ以上 L2 に渡されないように、L1D パイプラインがストールする。L2 OzQ 内の要求は、L2 内で処理が完了したとき、つまり、ストアがデータ配列を更新するか、ロードが適切なデータをコアに返すか、L2 ミスの要求がシステム・バス / L3 によって受け入れられたとき、L2 OzQ から削除される。

6.7.2.2 L2 OzQ の動作

L2 OzQ の制御ロジックは、アーキテクチャ的な順序づけの必要条件を使用する。また、アーキテクチャ上で許される場合は、操作をアウト・オブ・オーダーで完了できる。競合または発行の制限のために操作がブロックされても、それより新しい操作の完了を妨げることはない。これによって、L2 内のリソースの利用効率が向上し、パフォーマンスのメリットが得られる。さらに、アウト・オブ・オーダー発行により、L2 の制御ロジックが一時的に要求をリタイアできなくなった場合、L2 はその状態から迅速に回復できる。

L2 OzQ はアウト・オブ・オーダー、ノンブロッキングであるため、操作の間の時間的な関係が取り除かれる。例えば、コード・ジェネレータが 2 つの操作の間隔を 4 サイクル空けた場合、それらの操作は 4 サイクルの間隔で L1D パイプライン内に現れる。しかし、競合のために第 1 の操作がただちに発行されず、L2 OzQ 内で待たされるときがある。この場合、コード・ストリーム内では 2 つの操作の間に 4 サイクルの間隔があるにもかかわらず、順序づけの制約がないとすれば、実際には L2 内では第 2 の操作が第 1 の操作より前に完了する。

L2 のレイテンシは、通常は 5 サイクル、7 サイクル、または 9 サイクル以上である。さまざまなレイテンシがあるのは、一部の操作は、必要なリソースとその要求に先行する操作に基づいて、異なる時間に発行され、L2 データ配列にアクセスするためである。L1D 要求が L2 OzQ 内で割り当てられる前に L2 データ配列にアクセスできる場合は、レイテンシは小さくなる。これが 5 サイクルおよび 7 サイクルの L2 OzQ バイパスである。これらのバイパスを使用できない操作では、レイテンシは 9 サイクル以上になる。これらの操作は、L2 OzQ 内に割り当てられた後、L2 OzQ から発行され、L2 データ配列にアクセスする。

6.7.2.3 5 サイクル / 7 サイクルのバイパス

新しい L1D 要求は、L2 OzQ 内の以前の操作との競合がない場合は、L2 OzQ の 5 サイクル・バイパスを使用して、直接 L2 データ配列に発行される。発行グループ内に競合がない場合は、発行グループ全体がこのバイパスを使用できる。競合が発生した場合は、古い方の要求がバイパスを使用し、新しい方の要求はバイパスを使用できない。

L2 バンクの競合については、6.7.3 節で説明する。ここでは、L2 ができるだけレイテンシを小さくするために要求の順序を変更する方法の例として、L2 バンクの競合をとりあげる。競合は、通常は、複数の要求が同じ L2 データ配列にアクセスしようとしたときに発生する (バンク競合)。以下の L1D 要求 (発行) グループの場合を考える。

```
ldfs f20 = [0x004]   (L2 Bank 0)
ldfs f21 = [0x008]   (L2 Bank 0)
ldfs f22 = [0x00c]   (L2 Bank 0)
ldfs f23 = [0x010]   (L2 Bank 1)
```

第 1 のロードが、5 サイクル・バイパスを使用する。第 1 のロードと第 2 のロードのバンク競合のために、第 2 のロードと第 3 のロードは 5 サイクル・バイパスを使用できない。第 4 のロードは、古い要求とのバンク競合やアーキテクチャ的な順序づけの必要条件がないため、5 サイクル・バイパスを使用する。

要求が 5 サイクル・バイパスを使用できない場合、7 サイクル・バイパスが次の選択肢になる。第 2 のロードと第 3 のロードのバンク競合のために、第 3 のロードは 7 サイクル・バイパスを使用できない。

上記の命令の後に、L2 によって処理される命令がさらに続く場合は、この状況はさらに複雑になる。前の例においてロードの発行グループの直後に、以下のロードの発行グループが続く場合を考える。

```
ldfs f25 = [0x014] (L2 Bank 1)
ldfs f26 = [0x018] (L2 Bank 1)
ldfs f27 = [0x01c] (L2 Bank 1)
ldfs f28 = [0x020] (L2 Bank 2)
```

この例では、f20 と f24 のロードが 5 サイクル・バイパスを使用する。f21、f22、f23 のロードは、7 サイクル・バイパスを使用しようとする。しかし、これらのロードが 7 サイクル・バイパスを使用する前に、f25、f26、f27、f28 の新しい要求グループが発行される。この発行グループ内では、f25 と f28 が 5 サイクル・バイパスを使用する。これによって、先行する発行グループは、7 サイクル・バイパスを使用できなくなる。f21 と f22 の要求は、L2 OzQ から発行されなければならない。このため、これらの要求のレイテンシは、6 サイクルから 12 サイクルに増える。各操作のレイテンシ(カッコ内の数値)は、次のようになる。

```
ldfs f20 = [0x004] (6)
ldfs f21 = [0x008] (12)
ldfs f22 = [0x00c] (13)
ldfs f23 = [0x010] (6)
ldfs f25 = [0x014] (6)
ldfs f26 = [0x018] (13)
ldfs f27 = [0x01c] (14)
ldfs f28 = [0x020] (6)
```

6.7.2.4 L2 OzQ の発行

L2 OzQ は、1 サイクルごとに、L2 データ配列に発行される要求 (L2 ヒット)、システム・バス /L3 に発行される要求 (L2 ミス)、または他の L2 タグの参照のために L1D に戻される要求 (再循環) がないかどうか検索する。L2 のキャンセルの条件は、6.7.3 項を参照のこと。L2 の再循環の条件については、6.7.4 項を参照のこと。

L2 は、それらの操作に競合がなく、それ以前に発行された操作にも競合がない場合、1 サイクルごとに最大 4 つの L2 ヒット・アクセスを発行できる。L2 ヒットの競合には、L2 データ配列バンクの競合、レジスタ・ポートの競合、L1D フィルの競合、順序づけの競合が含まれる。L1D フィルの場合、1 サイクルに発行できるロードは 1 つだけである。また、L2 はロードのために L1D レジスタのリターン・パスを使用するため、1 サイクルごとに発行できるロードは 2 つだけである。

L2 は、システム・バス /L3 に対するアクセスを一度に 1 つだけ発行できる。同じ L1D 要求グループ内に L2 ヒットと L2 ミスがある場合、L3 レイテンシを最小限に抑えるには、L2 ミスを M0 ポート上に発行する必要がある。他のポート上に L2 ミスを発行すると、レイテンシが多少大きくなる。

システム・バス /L3 の制御ロジックは、システム・バス /L3 のリソースと競合条件に基づいて、要求を受け入れるか、または拒否する。要求を受け入れられると、その要求は L2 OzQ から削除される。L2 OzQ は、L2 ミスの要求をパイプライン処理する。つまり、L2 OzQ は、システム・バス /L3 が要求を受け入れるのを待たずに、次の要求を発行する。

6.7.3 L2 のキャンセル

L2 のキャンセルは、一般的に、5 サイクルまたは 7 サイクルの L2 OzQ バイパスを使用する要求にのみ適用される。これは、ほとんどの場合、発行ロジックは、競合の条件を考慮に入れて、競合が解決されるまで発行を保留するからである。L2 OzQ からの発行が保留される最も良い例は、バンクの競合である。しかし、起こり得る発行時間の競合を回避するのに必要なすべての情報が、常に利用できるとは限らない。したがって、L2 OzQ から発行された要求の一部は、後でキャンセルされ、再発行されなければならない。バイパスを使用する操作がキャンセルされると、その操作は L2 OzQ から再発行される。これは、L1D 要求グループだけがバイパスを使用できるためであ

る。L2 OzQ から要求が発行され、後でキャンセルされた場合、その要求のレイテンシは4 サイクルだけ大きくなる。

また、キャンセル・ロジックは、発行ロジックを簡単にするためか、必要な情報が利用できないために、予想より多くの発行をキャンセルまたはブロックするときがある。例えば、再循環される要求が、発行の対象となる他の命令のキャンセル/ブロック計算に含まれる場合や、発行ロジックが、2つ以上の要求を再循環できないにもかかわらず、再循環する必要がある要求を最大4つまで発行しようとする場合が、これに該当する。

L2 OzQ の発行のキャンセルまたはブロックには、多くの理由がある。これらの理由は、予測した上で回避できる条件と予測不可能な条件の2つのカテゴリに分類される。

6.7.3.1 予測した上で回避できるキャンセル条件

L2 データ配列の競合 : L2 データ配列は、幅 16 バイトのバンクを持つ。アドレスのビット 7:4 がバンクを指定する。同じバンクに対する要求は、キャッシュ・ラインに関係なく、バンク競合を発生させる可能性がある。L1D の要求グループに、同じバンクをターゲットとする複数のロードが含まれている場合は、新しい方の要求がキャンセルされるか、L2 OzQ の発行がブロックされる。同じバンクをターゲットとする複数のストアについても、同じ規則が適用される。

L2 ロードと L2 ストアは、L2 データ配列に同時にアクセスしないため、同じ要求グループ内にロードとストアが含まれていても、バンク競合は発生しない。しかし、全く異なる L1D 要求グループの間で、ロードとストアのバンク競合が発生する可能性がある。ストア要求がロード要求の3 サイクル後にデータ配列にアクセスする場合を考える。ロードとストアが同じ L2 バンクにアクセスする場合、時間 X に発行されたストアによって、時間 X+3 に発行されるはずのロードがブロックまたはキャンセルされる。

以下に、競合処理ロジックが、L2 データ配列へのアクセス時間を考慮に入れてバンク競合を判定する方法の例を示す。以下の2つの例では、バンクの競合は発生しない。

```
ld8 r20 = [0x008] ;;
ld8 r21 = [0x010]
```

および

```
st8 [0x008] = r20
ld8 r21 = [0x010]
```

次の例では、ストアと最後のロードの間にバンク競合が発生し、他の要求の間にはバンク競合は発生しない。

```
st8 [0x000] = r0 ;;
ld8 r19 = [0x000] ;;
ld8 r20 = [0x008] ;;
ld8 r22 = [0x110]
```

L1D フィル要求によるバンク競合は、予測が多少難しくなる。これらのバンク競合の原因は、L1D フィルが 64 バイトのデータを(したがって、一度に4つのバンクを)必要とすることである。さらに、L1D へのデータ・パスが受け入れるフィルは、1 サイクルおきに1つだけである。すべての L1D ミスが L1D フィルを要求するわけではないので、この競合の予測は簡単ではない。どの要求が L1D フィルを要求するかについては、6.6.1 項を参照のこと。

6.7.3.2 予測不可能なキャンセル条件

一般に予測不可能なバンク競合もいくつか存在する。これらのイベントは、システム・バスおよび L3 からの予測不可能なデータ・リターンに密接に関連している。予測不可能なキャンセル条件があると、原因不明の L2 レイテンシの増加が起こる。

6.7.4 L2 の再循環

要求がヒットかミスか明確でない場合や、L2 ミス処理の完了に必要なリソースが利用できない場合は、L2 OzQ は要求を再循環しなければならない。

要求の再循環の最も予測しやすい理由は、システム・バス /L3 によってすでに処理されているが、まだ L2 に返されていないラインに対する要求ミスである。L2 は、L2 ヒットと 1 次的な L2 ミスだけを L2 ラインにリタイアする。L2 は、複数の L2 ミス要求をリタイアしない。2 つ目以降のミスは、L2 OzQ 内にとどまり、タグ・ルックアップがヒットを返すまで再循環される。タグにヒットした要求は、L2 OzQ から発行され、通常の L2 ヒット要求と同じように、データを返すか (ロードの場合)、配列を更新する (ストアの場合)。

6.7.4.1 lfetch と再循環

上記の 2 次的な L2 ミスの再循環条件には、1 つの重要な例外がある。lfetch 命令は、以下の基準を満たす場合、L2 OzQ 内への割り当てを回避するように最適化されている。

- L2 ミスに対する 2 次的なアクセスである
- L1D のフィルを行わない

このような lfetch 命令は、L2 OzQ 内に割り当てられないため、再循環されない。lfetch 命令が L1D をフィルしないように保証する方法は、.nt1、.nt2、または .nta などのテンポラルなヒントを使用すること以外にない。

6.7.5 順序づけ

Itanium アーキテクチャの順序づけ規則では、アクワイア・セマンティクスを持つ要求は、後続のすべての操作より前に検出可能にならないといけない。リリース・セマンティクスを持つ要求は、先行する操作より前に検出可能になってはならない。

L2 発行ロジックは、L2 OzQ 内の最も古い操作になるまでリリース要求の発行をブロックすれば、リリース・セマンティクスのアーキテクチャ的な順序づけを実現する。L2 発行ロジックは、最も古い操作になる前にリリース操作を発行した場合、その操作をキャンセルして再発行する。

順序づけされた操作が L2 にヒットしない場合、L2 制御ロジックは、そのラインのシステム・バス /L3 要求を見込み的に生成するか、または順序づけされた要求をプリフェッチに変換できる。順序づけされた要求に先行する他の L2 OzQ エントリが競合しない場合、プリフェッチには、順序づけの必要条件に違反せずに、アクセスを早く開始できるメリットがある。競合がある場合は、適切な順序づけを保証するために、その要求は再発行される。

L2 はアーキテクチャ的な順序づけを維持する責任を負うため、ld.acq の影響を受けるすべてのロードは、L2 によって検出されなければならない。したがって、ld.acq が検出可能になるまで、これらのロードは強制的に L1D ミスになる。

6.7.6 L2 命令プリフェッチ FIFO

命令プリフェッチ FIFO (IPF) は、L1I 要求を保持する 8 エントリのキューである。8 つのエントリのうち 7 つは、プリフェッチ要求を格納できる。1 つのスロットは、デマンド要求のために常に予約されている。L2 OzQ と同じように、IPF は、L2 ヒット、L2 ミス、バンク競合、または再循環になった要求を格納できる。これらの要求のそれぞれについて、IPF には、L2 OzQ と同じ発行制限が適用される。しかし、L2 OzQ のヒット要求とは異なり、L2 データ配列に対して 1 サイクルごとに発行される IPF の L2 ヒットは 1 つだけである。これは、すべての IPF 要求は L1I キャッシュにデータを返すが、L1I に戻るデータ・パスは、1 サイクルごとに 1 つのフィルしか受け入れないためである。

L2 は命令アクセスとデータ・アクセスの両方をサポートするため、すべての L2 発行制御ロジックは、表 6-6 に従って、命令要求とデータ要求を選択する。

表 6-6. L2 の発行の優先度

優先度	要求
1	デマンド命令フェッチ (IPF)
2	デマンド命令フェッチ (7 サイクル・バイパス)
3	データ (L2 OzQ)
4	データ (7 サイクル・バイパス)
5	プリフェッチ命令フェッチ (IPF)
6	プリフェッチ命令フェッチ (7 サイクル・バイパス)
7	データ (7 サイクル・バイパス)

6.7.7 システム・バス / L3 との相互作用

L2 で処理できないすべての要求は、RL (Read Line) 要求または RFO (Read For Ownership) 要求としてシステム・バス / L3 に送られる。RL 要求は、コード操作と一般的なロード操作に使用される。RL 要求の場合、L2 は、L3 ステートまたはシステム・バス上のスヌープ応答に基づいて、ラインを M、E、または S ステートで受け入れる。RFO 要求は、L2 がデータをストアするためにラインを修正しようとしていることを示す。ストアと、`lfetch.excl` および `ld.bias` 命令は、RFO 要求を生成させる。RFO 要求は、L2 内に常に M ステートで存在する。表 6-7 は、この動作をまとめたものである。

表 6-7. システム・バス / L3 への要求と最終的な L2 ステート

L3/ システム・バスへの要求	L2 への要求	L3 ステート			システム・バス	
		S	E	M	ヒット	ヒットなし
RL (Read Line)	コード読み出し	S	E	M	S	S
	データ読み出し	S	E	M	S	E
RFO (Read For Ownership)	ストア	ミス	M	M	M	なし
	<code>lfetch.excl</code>	ミス	M	M	M	
	<code>ld.bias</code>	ミス	M	M	M	

L2 は、システム・バスに対して部分的なライン要求を生成するときがある。これは UC 属性アクセスの場合に限られる。これらの要求には一貫性がなく、パフォーマンス上の問題にもならないため、ここでは扱わない。

L2 は、システム・バス / L3 に対して、1 サイクルごとに 1 つの RL または RFO を生成する。ビクティムとして選択された L2 ウェイが M ステートになっている場合、各 RL/RFO 要求は、それに関連する 1 つのダーティ・ビクティムを持つ。L2 は、システム・バス / L3 に対して要求を発行した後、その要求を確認する。このプロトコルによって、システム・バス / L3 に対して発行した要求を、後でキャンセルまたは再循環できる。L2 は、使用可能なリソースが足りない場合は、要求を発行しても確認を行わない。L2 は、同じ L2 ラインに対する 2 つの要求を発行しない。確認されなかった要求は、少なくとも 4 サイクル待機してから再発行される。

システム・バス / L3 は、アドレスの競合、読み出し要求に使用できるリソース、および関連するダーティ・ビクティムに基づいて、要求を受け入れるかどうかを決定し、L2 に通知する。システム・バス / L3 が要求を受け入れる場合は、L2 は L2 OzQ から要求を解放する。L2 の要求は、拒否されるときがある (6.9 節を参照)。拒否された要求は、少なくとも 4 サイクル待ってから再発行される。

システム・バス / L3 が L2 にデータを配布する準備ができると、それが L2 に通知され、L2 はデータを受け取る準備をする。システム・バス / L3 からのデータは、重要なチャンクから先に、一度に

32 バイトずつ返される。L3 からのリターンは、システム・バスからのデータ・リターンより優先度が高く、連続的に行われる。多くの場合、L2 ミスがあると、L1D フィルも行われる。L1D ラインの幅は 64 バイトしかないため、システム・バスまたは L3 から 2 チャンクを受け取っただけで、L1D フィルを実行するのに十分なデータになる。L1D フィル要求は、L1D パイプラインにアクセスしなければならない。これらの要求は、そのサイクルの間、コアからの要求が L1D パイプラインに入らないようにブロックできる。1 つの L2 ミスで 2 つの L1D フィルが行われる場合は、L2 が最後の 2 チャンクを受け取った時点で、2 回目のフィルが行われる。

6.8 3 次ユニファイド・キャッシュ

3 次ユニファイド・キャッシュ (L3) は、ユニファイド、3M バイト、12 ウェイ・セット・アソシアティブ・キャッシュ (ライン・サイズ 128 バイト) である。Itanium 2 プロセッサの一部のバージョンには、1.5M バイトの 6 ウェイ・セット・アソシアティブ L3 キャッシュがある。これらの 2 種類のキャッシュは、他のすべての点では同じである。

すべての L3 アクセスは、128 バイト・ライン全体に対して行われる。不完全なライン・アクセスはサポートしていない。アクセス・レイテンシは、12 サイクル、16 サイクル、またはそれ以上である。このレイテンシは、L2 がどの程度迅速に要求を発行するかと、要求の時点での L3 の動作によって決まる。

Itanium 2 プロセッサでは、L3 アクセスがフルにパイプライン処理されるため、実効帯域幅は Itanium プロセッサの L3 よりはるかに大きい。L3 タグ配列はシングル・ポートであり、1 サイクルごとに新しいタグにアクセスできるようにパイプライン化されている。L3 データ配列もシングル・ポートであるが、L3 ダーティ・ビクティムの場合に L2 キャッシュまたはシステム・バスにライン全体のデータを転送するには、最大 4 サイクルを必要とする。

L3 はノンブロッキングであり、複数の実行待ち要求を保持できる 8 エントリのキューを持つ。このキューは、要求を順序づけて、タグ読み出し / 書き込みおよびデータ読み出し / 書き込み要求の優先度を指定し、要求された操作に対して最高のパフォーマンスを実現する。

L3 の利用率は、オンボードの消費電力制御機能によって調整できる。

6.9 システム・バス

Itanium 2 プロセッサのシステム・バスは、200MHz で動作する。このシステム・バスは、アドレス / 要求、スヌープ、応答、データ、据え置きなどの各種の機能に対応する複数のサブバスで構成される。データ・バスの幅は 128 ビットで、ソースと同期して動作し、ピーク帯域幅で 1 秒当たり 4 億回のメモリ・トランザクション (6.4GB) を実現している。

システム・バス制御ロジックは、インオーダー・キュー (IOQ) とアウト・オブ・オーダー・キュー (OOQ) である。これらのキューは、システム・バス上で完了が保留されているすべてのトランザクションを保持する。IOQ は、要求のインオーダー・フェーズを保持し、すべてのプロセッサに対して同じ内容になる。OOQ は、据え置かれたプロセッサ要求だけを保持する。IOQ は 8 つのエントリを保持でき、OOQ は 18 個の要求を保持できる。これによって、1 つの Itanium 2 プロセッサからのトランザクションを、システム・バス上で最大 19 個まで実行待ちにできる。

処理が完了していない L2 要求 (すなわち、まだ L3 にアクセスしていない要求と、システム・バス上のデータ・フェーズを完了していない要求) は、以下のサイズの構造として保持される。

- L2 からの 16 個の実行待ち読み出し要求
- L2 からの 6 つの実行待ちダーティ・ライトバック要求
- メイン・メモリによって処理される、6 つの実行待ち L3 ライトバック (ダーティなラインの置換)
- メイン・メモリによって処理される、16 個の実行待ち L3 ライトバックまたは L3 キャストアウトの組み合わせ (キャストアウトとは、コヒーレンシ・メカニズムに基づくクリーンなラインの置換であり、メモリ・トラフィックが発生する場合がある)

- WC ストアをサポートする、2 つの 128 バイト・コアレシング・バッファ

読み出しトランザクション (L2 ミスになったストア命令を含む) は、16 個のバス要求キュー (BRQ) のうち 1 つに入れられる。BRQ 内の各要求は L3 に送られ、L3 がその要求を満たすかどうか確認される。この要求がさらに L3 ミスになった場合は、システム・バス要求 (ストアの場合、Bus Read Line または Bus Read Invalidate Line) を生成するようにスケジューリングされる。システム・バスがデータを返すと、そのラインは、テンポラルな局所性のヒントとアクセスのタイプに基づいて、L2 と L3 に書き込まれる。

Itanium® 2 プロセッサでは、静的な手法と動的な手法の両方を使用して分岐予測を行う。静的な分岐予測の場合、Itanium 2 プロセッサは、分岐命令内のヒント・コンプリータを使用する。動的な予測の場合、Itanium 2 プロセッサは、複数のハードウェア構造を使用する。

本章では、分岐予測がソフトウェアの実行に与える影響について説明する。フロントエンドの命令フェッチ機能とバックエンドの命令実行機能は、8 バンドルの命令バッファによって分離されている。命令バッファの詳細は、付録 A「Itanium® 2 プロセッサのパイプライン」を参照のこと。本章全体を通して、「バブル」の用語は、フロントエンドが有効なデータを配布できない時間（サイクル）を指す。このペナルティをバブルと呼ぶのは、他のイベントがバックエンドの命令リタイアを妨げている場合、フロントエンドのペナルティがパフォーマンスの低下に現れないことがあるためである。バックエンドがフロントエンドを待機しなければならない場合は、そのペナルティをストールと呼ぶ。

表 7-1「分岐予測のレイテンシ」は、Itanium 2 プロセッサの分岐予測のレイテンシをまとめたものである。正しく予測された IP 相対分岐では、フロントエンド・バブルは発生しない。

表 7-1. 分岐予測のレイテンシ

分岐のタイプ	分岐有無予測	ターゲット予測	ペナルティ
IP 相対	正しい	正しい	0 サイクルのフロントエンド・バブル
IP 相対	正しい	誤り	1 サイクルのフロントエンド・バブル
リターン	正しい	正しい	1 サイクルのフロントエンド・バブル
リターン	正しい	誤り	6 サイクル以上のパイプライン・ストール
間接	正しい	正しい	2 サイクルのフロントエンド・バブル
間接	正しい	誤り	6 サイクル以上のパイプライン・ストール ¹
ループ	誤り	なし	7 サイクル以上のパイプライン・ストール ¹
任意のタイプ	誤り	なし	6 サイクル以上のパイプライン・ストール ¹

1. + は、一部の分岐がフロントエンドをストールさせる可能性があることを示している。これは、誤って予測された短い（最大 16 バンドルの）前方分岐の場合に限られる。追加のレイテンシは最大 8 サイクルで、フロントエンドが予測ミスの分岐を検出した後、フロントエンドが分岐をいくつ検出したかによって変化する。

Itanium 2 プロセッサの分岐予測マイクロアーキテクチャは、Itanium プロセッサの分岐予測マイクロアーキテクチャとはかなり異なっている。分岐予測機構は L1I キャッシュに密接に結合されているため、ゼロ・バブルでの実行が可能である。

1 サイクル分岐には、1 サイクルおきに一度ストールが発生する。すなわち、1 サイクルのループを 3 回反復するのに 4 サイクルかかる。1 サイクルのループは避けた方がよい。また、複数の分岐が連続して検出された場合も、ストールが発生する可能性がある。例えば、フロントエンドが 3 サイクルにわたって各サイクルで分岐を検出すると、1 サイクルのストールが発生する。

7.1 分岐予測ヒント

分岐の動作に関する情報をプロセッサに与えることで、分岐予測の精度を向上できる。この情報は、分岐ヒントによって分岐命令の中にコード化される。分岐ヒントは、プログラムの機能的な動作には影響を与えない。また、プロセッサは分岐ヒントを無視するときがある。

分岐命令の中で指定されたヒントだけが、分岐予測に使用される。brp または mov br 命令のヒントは、分岐予測機構によって無視される。

Itanium 2 プロセッサの分岐ヒントは、.sptk、.spnt、.dptk、または .dpnt である (sp= 静的予測、dp= 動的予測、tk= 発生する、nt= 発生しない)。「静的」ヒントと「動的」ヒントの用語は、コード・ジェネレータが分岐の動作を確実に知っているかどうかを示す。例えば、.sptk は、分岐が発生するのをコード・ジェネレータが確実に知っているという意味である。.dptk は、コード・ジェネレータは分岐が発生すると予想しているが、確実性はないという意味である。

これらの分岐ヒントの影響は、2 バンドル・ウィンドウ内の他の分岐とプロセッサ内の他の分岐情報によって異なる。したがって、.dpnt ヒントを持つ分岐は、最初に検出されたときは、発生すると予測されるときがある。プロセッサは、この予測から迅速に回復し、その後はこの分岐を正しく予測する。

デフォルトでは、.dpxx の使用を推奨する。ただし、ループが ctop または cloop である場合は、.spxx を推奨する。

7.2 間接分岐

リターン以外の間接分岐の予測されるターゲットは、ハードウェア・テーブルからではなく、その間接分岐のソース分岐レジスタから抽出される。このことはさまざまな意味を持っている。

Itanium 2 プロセッサでは、間接分岐には常にペナルティが発生する。正しく予測された間接分岐では、2 サイクルのフロントエンド・バブルが発生する。分岐有無予測またはアドレス予測が誤っている場合は、6 サイクル以上のパイプライン・ストールが発生する。アドレス予測は、フロントエンドが認識する、分岐が参照する分岐レジスタの内容に基づいて行われる。フロントエンドは分岐レジスタの段階的な更新を認識しないため、ターゲットの予測が間違っている可能性がある。ターゲットを正しく予測するには、間接分岐より数サイクル前に、分岐レジスタの書き込みが行われている必要がある。パイプラインのフロントエンドとバックエンドが分離されているため、この間隔は変化する。コード・ジェネレータは、以下の方法で、この影響を最小限に抑えられる。

- 分岐レジスタの書き込みと間接分岐の間隔を、少なくとも 6 回のフロントエンド LII キャッシュ・アクセスだけ空ける。
- 真の分岐レジスタ書き込みの上に、ターゲットのヒントを与える書き込みを追加する。
- 間接分岐ごとに異なる分岐レジスタを使用して、他の間接分岐との競合を最小限に抑える。

7.3 完全ループ予測

多くの場合、完全ループ予測機構は、カウントされるループの最後の分岐 (すなわち、clloop または ctop タイプの分岐) を、フォールスルー分岐やループの後方反復を含めて、正しく予測できる。Itanium プロセッサとは異なり、Itanium 2 プロセッサは、この処理を行うために brp を必要としない。

Itanium 2 プロセッサは、ループの最後の反復にのみ PLP を使用する。最初のループ予測は、使用されるヒントと L2B 内の利用可能な履歴に基づく動的情報または静的情報によって決定される。

ループの最後の分岐が正しく予測された場合でも、この正しい予測を得るために、1 サイクルまたは2 サイクルのバブルが発生することがある。ループの反復回数が少ないほど、2 サイクルのバブルが発生する可能性が高くなる。逆に、ループの反復回数が多いほど、バブルが0 になる可能性が高くなる。PLP は、`ar.lc` および `ar.ec` の現在値を予測に使用する。したがって、正しい予測を保証するには、これらのレジスタへの書き込みからカウントされるループの分岐までの間に十分な間隔が必要である。

Itanium プロセッサでは、場合によっては、正しい予測のために `ar.ec` が 1 に設定されている必要があった。Itanium 2 プロセッサには、この必要条件は適用されない。実際に、Itanium 2 プロセッサは、エピソードがないときは、`ar.ec = 0` と予想する。

Itanium® 2 プロセッサは、さまざまな形式の命令プリフェッチ機能をサポートしている。命令プリフェッチとは、命令キャッシュ・ラインを上位のキャッシュまたはメモリから LII に移動する操作である。ストリーミング・プリフェッチは、シーケンシャルに、または発生すると予測される分岐のターゲットとして、次のキャッシュ・ラインのハードウェア・プリフェッチを開始する。ヒント・プリフェッチは、ソフトウェアがプリフェッチされるラインを指定できる。Itanium 2 プロセッサでは、コード・ジェネレータがプリフェッチ・エージェントを広範囲にわたって制御でき、また Itanium 2 プロセッサのキャッシュは特にプリフェッチ機能を考慮に入れて設計されている。したがって、命令プリフェッチ機能は、命令キャッシュ・ミスを減らすための効果的な方法になるはずである。

8.1 ストリーミング・プリフェッチ

ストリーミング・プリフェッチを開始するには、分岐命令上で `.many` コンプリータを使用する。フロントエンドが `.many` コンプリータを持つ分岐を処理する場合、プリフェッチ・エンジンは、プリフェッチ要求を 1 サイクルに 1 つずつ連続的に発行し、これ以降の命令ラインをプリフェッチ・パイプラインに入れていく。このプリフェッチ要求は、LII と L1 ITLB に対してチェックされる。プリフェッチ要求が L1 ITLB にヒットし、LII でミスになった場合は、その要求は L2 に送られる。それ以外の場合は、要求は廃棄される。命令ラインは、(分岐ターゲットのアライメントに基づいて) 分岐ターゲットに 64 バイトまたは 128 バイトを加算した位置を始点としてプリフェッチされる。ストリーミング・プリフェッチは、以下のいずれかのストップ条件が発生するまで続けられる。

- 発生すると予測された分岐をフロントエンドが検出する
- 分岐予測ミスが発生する
- `.imp` コンプリータなしの `brp` 命令をフロントエンドが検出する¹

LII キャッシュは、フィルとルックアップが同時に行えるように設計されている。したがって、ISB 内の要求の存続期間は、通常は非常に短い。このため、プリフェッチ・エンジンが命令をプリフェッチしたとき、プリフェッチしたラインが使用される前に上書きされることはほとんどない。フロントエンドが分岐が発生すると予測した場合は、フロントエンドでプリフェッチが開始される。フロントエンドが分岐が発生しないと誤って予測した場合は、予測が修正された時点で、バックエンドがプリフェッチを開始する。ただし、これとは逆の場合、つまり、フロントエンドが分岐が発生すると誤って予測した場合は、プリフェッチは終了し、バックエンドがこの予測を修正してもプリフェッチは再開されない。最後に、フロントエンドが分岐が発生すると誤って予測した場合は、バックエンドがこの予測を修正した時点で、プリフェッチは終了する。

1. `brp` 命令は、それに対応する `br.many` が近くにあるのを示唆する。プリフェッチ・エンジンは `br.many` の後に既にプリフェッチを実行しているため、これ以上のプリフェッチは不要であると考えられる。`brp.imp` ではプリフェッチが終了しない理由は、Itanium プロセッサのコードに関係がある。Itanium プロセッサでは、`brp.imp` 命令は分岐予測に使用されるため、`br.many` に対応していない可能性がある。

表 8-1. ストリーミング・プリフェッチ動作のまとめ

	発生すると予測された場合	発生しないと予測された場合
実際に発生した場合	現在のストリーミング・プリフェッチは、フロントエンドで中止される。 分岐が many コンプリータを持つ場合は、フロントエンドによって新しいストリームが開始される。	現在のストリーミング・プリフェッチは、バックエンドで中止される。 分岐が many コンプリータを持つ場合は、バックエンドで新しいストリームが開始される。
実際には発生しなかった場合	現在のストリーミング・プリフェッチは、フロントエンドで中止される。予測ミスが検出されても、プリフェッチは再開されない。 分岐が many コンプリータを持つ場合は、フロントエンドで新しいストリームが開始される。バックエンドによって予測ミスが検出された時点で、このストリームは終了する。	現在のストリーミング・プリフェッチに影響を与えない。 新しいストリームは開始されない。

8.2 ヒント・プリフェッチ

ヒント・プリフェッチを開始するには、`brp` または `mov br` 命令を使用する。Itanium プロセッサとは異なり、Itanium 2 プロセッサでは、プリフェッチの開始は分岐予測ステートに影響を与えない。ただし、Itanium 2 プロセッサには、Itanium プロセッサと同じように、以下の制限がある。すなわち、`brp` 命令が処理されるためには、`brp` 命令はバンドルの最後の命令スロット (スロット 2) 内になければならない。それ以外の場合は、`brp` 命令は無視される。`brp` は、分岐予測に影響を与えない。表 8-2 に、各分岐ヒントに対応するプリフェッチ機構を示す。

表 8-2. プリフェッチ機構

分岐ヒント	プリフェッチ機構
<code>brp.(sptk,loop,dptk).few</code>	生成された 1 つのキャッシュ・ラインを通常の方法でプリフェッチする。
<code>brp.(sptk,loop,dptk).many</code>	ターゲットから 2 つのキャッシュ・ラインをプリフェッチする。
<code>brp.(sptk,loop,dptk).imp.few</code>	プリフェッチ仮想アドレス・バッファ (PVAB) をフラッシュし、1 つのキャッシュ・ラインをプリフェッチする。
<code>brp.(sptk,loop,dptk).imp.many</code>	プリフェッチ仮想アドレス・バッファ (PVAB) をフラッシュし、2 つのキャッシュ・ラインをプリフェッチする。
<code>move_to_br.(sptk,dptk).few</code>	他のすべてのフィールドを無視して、1 つのキャッシュ・ラインをプリフェッチする。
<code>.many hint</code>	発生すると予測された IP 相対分岐で、ストリーミング・プリフェッチをトリガする。

`.few` コンプリータは、関連する分岐ターゲットのアライメントに基づいて、1/2 または 1 つの L2 ラインをプリフェッチする。`.many` コンプリータは、関連する分岐ターゲットのアライメントに基づいて、1.5 または 2 つの L2 ラインをプリフェッチする。ヒント・プリフェッチは、8 エントリのプリフェッチ仮想アドレス・バッファ (PVAB) に送られる。1 サイクルごとに最大 2 つのヒント・プリフェッチを PVAB に送ることができる。

特定のサイクルで、プリフェッチ・パイプラインがストールせず、`br.many` がアクティブになっていない場合は、プリフェッチ要求は PVAB から削除され、L1I と L1 ITLB に対してチェックされる。プリフェッチ要求が L1 ITLB にヒットし、L1I でミスになった場合は、その要求は L2 に送られる。それ以外の場合は、要求は廃棄される。この操作の意図は、ヒント・プリフェッチを使用して分岐ターゲットの位置にある命令の最初の「チャンク」をプリフェッチし、ストリーミング・プリフェッチを使用してそれ以降の命令をプリフェッチすることである。L2 ヒットのレイテンシ全体を隠蔽するためには、ヒント・プリフェッチは、分岐に対して 9 フェッチ・サイクル先行し

ている必要がある。br.many の前に brp.many があると、2つの命令によって生成されるプリフェッチが一部重複する。この重複は無駄であるが、(br.many の前に brp.few がある場合と比べて)より早くより多くのラインをプリフェッチできる利点がある。brp.few プリフェッチは、8.1 節で説明したように、ストリーミング・プリフェッチと組み合わせたときに効果的である。

8.3 プリフェッチ・フラッシュ・ヒント

特定の形式の brp 命令には、PVAB の内容と (場合によっては) プリフェッチ・パイプラインをフラッシュする副次的作用がある。これらの命令は、コンパイラがプリフェッチ機能の状態をある程度制御できるように用意されている。

- brp.few.imp - すべての brp.few プリフェッチを PVAB から削除する (すでにプリフェッチ・パイプライン内にあるものは削除しない)。
- brp.exit.imp - PVAB 内のすべてのプリフェッチとプリフェッチ・パイプライン内のプリフェッチを削除し、ストリーミング・プリフェッチ・エンジンを停止する (したがって、br.many、brp.many、brp.exit プリフェッチは中止される)。
- brp.*.imp - (.imp コンプリータなしの brp) br.many 命令によって開始されたストリーミング・プリフェッチをすべてキャンセルする。この命令によって、コンパイラは、br.many による必要以上のプリフェッチを止められる。

これらのプリフェッチ命令を使用すると、通常の動作以外に、副次的作用としてフラッシュが実行される。ただし、すでにパイプラインに到達したプリフェッチについては、プリフェッチのフラッシュが有効でない (つまり、L2 およびそれ以上のレベルに対して発行される) 場合がある。

8.4 brl 命令

Itanium 2 プロセッサは、64 ビット相対分岐をサポートする brl 命令を使用できる。これらの長い相対分岐命令のコストは、Itanium プロセッサ内で使用する場合より小さいが、短い相対分岐命令 (br 命令) のコストより大きい。厳密には、Itanium 2 プロセッサの分岐予測機構は、LII キャッシュ・ラインが書き込まれるときに分岐ターゲットが設定されない限り、brl 命令の予測ターゲットを正確に計算できない。したがって、brl の予測ターゲットがバンドル・ペア内の他の分岐によって別名参照される場合は、ターゲットは不正確になり、分岐予測ミスのペナルティがフルに発生する。しかも、この誤りは修正されない。

このようなコストにもかかわらず、brl 命令は、複数の短いジャンプを使用するよりはるかに効率的である。ただし、リンカは、特に必要な場合にのみ brl 命令を使用するべきである。

本章は、前章までに説明した重要な内容から得られる結論をまとめたものである。これらのガイドラインは、すべての状況に適用できるとは限らない。最適化手法の使用ガイドとして、プロファイリングを利用する必要がある。

9.1 スケジューリングのヒント

可能な限り以下の経験的手法に従えば、暗黙的なストップや、ディスパーサルに関連する予想不可能なストールを最小限に抑えられる。

- 最も制約条件の厳しい命令を、バンドル内で先にスケジューリングする。これによって、一般サブタイプ命令が占有しているポートを、後で制約条件の厳しい命令が要求する可能性が小さくなる。
- 場合によっては、A タイプの命令を M スロットではなく I スロットに入れると、バンドリングの密度が高くなる。この場合は、可能な限り I タイプの命令 (I スロットに入れなければならない) を発行グループ内で先に配置すること。この方法で、これ以降の I スロット内の命令を、使用可能な M ポートに発行できる。ただし、Itanium プロセッサなど、プロセッサの中にはこの方法をサポートしていないものもあるため、通常は A タイプの命令は M スロットに入れる方がよい。
- ほとんどの浮動小数点ロード・タイプは、M0 と M1 だけでなく、4 つのメモリ・ポートのうちどれに対しても発行できる。しかし、コントロール・スペキュレーションに関連するロード (アドバンスト・ロードとチェック・ロード) と、浮動小数点ロード・ペアは、ポート M0 と M1 以外には発行できない。FP ロード、アドバンスト FP ロード、整数ロード、lfetch 命令を混在させてスケジューリングする場合は、通常の FP ロードは、必要に応じて M2 ポートと M3 ポートに発行できるように、発行グループ内で後にスケジューリングすること。これによって、lfetch 命令または制約条件の厳しいロード・タイプが要求する M0 ポートと M1 ポートが解放される。
- nop.f の使用を避ける。nop.f を使用すると、レイテンシの長い未処理の命令のために、予想不可能なストールが発生するおそれがある。例えば、FPSR への書き込みはマルチサイクル操作である。この書き込みが完了するまで、nop.f を含むすべての浮動小数点操作はストールする。
- Itanium プロセッサでは、デュアル発行が簡単に行えるように、MFI テンプレートがよく使用されていた。Itanium 2 プロセッサには、MFI 以外の多くのデュアル発行テンプレート・ペアがあるので、MFI テンプレートを使用する必要はない。

9.2 lfetch の最適な使用

Itanium 2 プロセッサが、メモリに関連する多くの状況ですぐれたパフォーマンスを発揮するためには、lfetch 命令の使い方が重要である。整数データに lfetch を使用すると、通常は L1D ヒットになる。これには、L1D キャッシュによって L2 に対する要求を選別できるメリットがある。整数ロードが L1D にヒットし、L2 で検出されないのを保証すれば、多くの L2 競合が避けられる。L2 で検出される要求が少ないほど、競合する要求は少なくなる。

lfetch 命令は、注意深く使用する必要がある。lfetch 命令を不注意に使用すると、パフォーマンスが低下する。Itanium 2 プロセッサのキャッシュ構造の詳細は、第 6 章「メモリ・サブシステム」を参照のこと。メモリ・サブシステムについては、以下のガイドラインが経験的に得られている。

- L3 またはメモリに対する実行待ちの `lfetch` 操作の最大数 (データ要求と命令要求の合計) が、16 を超えてはならない。
- `lfetch` 命令はメモリ・ポート M0 および M1 のみに制限されているが、FP ロード (`ldfpd` と `ldfps` を除く) は 4 つのメモリ・ポートのうちどれでも発行できる。したがって、`lfetch` 命令と FP ロードを混在させる場合は、発行グループ内で `lfetch` 命令を先にスケジューリングすべきである。例えば、2 つの FP ロードと `lfetch` を同じサイクルにスケジューリングする場合は、`lfetch` を第 1 バンドルにスケジューリングし、最初の 2 つのメモリ・ポートのうちいずれかで発行されるようにする。2 つの FP ロードを先にスケジューリングすると、ハードウェアは、`lfetch` 命令を発行する前に暗黙的なストップを挿入する。
- Itanium 2 プロセッサの `lfetch.excl` 命令は、L2 キャッシュ内に M ステートでデータを入れる。`.excl` コンプリータは、`lfetch` によってキャッシュに入れられたデータがストア命令によってすぐに変更されそうな場合のみ使用すること。
- Itanium 2 プロセッサの `lfetch` 命令は、変換情報と保護情報を提供する DTLB エントリが使用できない場合は、データをキャッシュに入れない。`lfetch` 命令が HPW ウォークを完了し、必要に応じて TLB 変換フォルトまたは保護フォルトを生成するように保証するには、`.fault` コンプリータを使用する必要がある。これらのイベントには大きなコストがかかるため、スペキュレーティブ・アドレスには `.fault` コンプリータを使用しない方がよい。
- `lfetch` 命令がキャッシュ階層に与える影響のために、この命令の使用に高いコストがかかることがある。この影響には、以下のものがある。
 - L2 OzQ などの L2 リソースを取得する
 - L2 データ配列へのアクセスのアービトレーションが行われ、L2 バンクの競合の候補になる
 - 2 次的な L2 ミスの場合の `lfetch` の再循環

2 次的な L2 ミスによる L2 再循環の影響を緩和するには、`lfetch` 命令に `.nt` コンプリータを追加する。`.nt` ヒントがあると、`lfetch` による L1D フィルが発生せず、L2 OzQ から `lfetch` を削除できる。ただし、この非テンポラルなコンプリータは必ずしも必要ではない。これは、`lfetch` 命令が 2 次的な L2 ミスになると、L2 OzQ ロジックがそれを認識し、`lfetch` が L2 OzQ 内に割り当てられないように、L1D フィルの実行を中止するからである。

`lfetch` が L2 にヒットすると、`.nt` ヒントや実際の L1D フィルの必要性に関係なく、`lfetch` が L2 OzQ リソースを取得し、他の要求がキャンセルされる。また、L2 にヒットした `lfetch` が、実際に L2 データ配列を読み込むかのように動作し、`lfetch` そのものがキャンセルされることがある。

9.3 データ・ストリーミング

高帯域幅の長いデータ・ストリームを処理するには、いくつかの方法がある。この節では、いくつかの可能な方法について、それぞれのメリットとコストを説明する。

9.3.1 浮動小数点データ・ストリーム

浮動小数点データは、L2 キャッシュに置かれる。ソース・ストリームの場合は、`lfetch.fault.nt1` 命令を 1 つの L2 キャッシュ・ラインにつき一度だけ発行する。デスティネーション・ストリームの場合は、`lfetch.fault.excl.nt1` 命令を 1 つの L2 キャッシュ・ラインにつき一度だけ発行する。`.fault` コンプリータは、データが L2 DTLB ミスまたは VHPT ミスになった場合でも、そのキャッシュ階層内に入れられることを保証する。`.nt1` コンプリータは、浮動小数点データが L1D 内のデータを置き換えないように保証する。`.nt1` コンプリータには、2 次的な L2 ミスになった `lfetch` 命令が、L2 OzQ 内に割り当てられないようにする機能もある。この機能は、データ・ストリーミング・コードの設計上、パフォーマンス低下の原因となる L2 ラインへの追加要求が避けられない場合に重要である。デスティネーション・ストリームに対する `.excl` コンプリータは、データが修正できる状態であることを保証する。

データが L2 ヒットとしてアクセスされる場合は、要求グループ間の L2 バンク競合を避けるように注意する必要がある。L2 の 5 サイクル / 7 サイクル・バイパスが使用できるのを保証するには、L2 バンク競合を避ける必要がある。浮動小数点コードについては、一般的に、レイテンシは問題にならない。ただし、ストリーミングの場合、L2 OzQ のサイズに対して L2 OzQ 内での操作の存続期間が長すぎると、OzQ が一杯であると L2 制御ロジックが考えるために、コアがストールするときがある。レイテンシが小さいほど、その操作の OzQ 内での存続期間が短くなり、実質的により多くの OzQ エントリが使用できる。

9.3.2 整数データ・ストリーム

整数データ・ストリームの処理は、浮動小数点データ・ストリームより複雑である。これは、整数データでは、場合によってはパフォーマンス上の理由で L1D 内にデータを入れる必要があるためである。L1D からのストリーミングには、いくつかの問題がある。第 1 に、各ロード操作は L1D にヒットし、(L1D ミスになる場合でも) 整数レジスタへのリターン・リソースを要求する。このため、L1D ミスは、新しい L1D ミスのフローに影響を与えずにレジスタ・ファイルにデータを返すことが難しくなる。第 2 に、各フィル操作の実行に 1 サイクルかかる。第 3 に、L1D をフィルする必要があるため、L2 OzQ から 2 次的な L2 ミスの `lfetch` 命令を削除できなくなる。L1D のライン・サイズは L2 の半分であり、1 つの L1D ラインにつき 1 つの `lfetch` によって、各 L2 ラインにつき少なくとも 1 つの 2 次的な L2 ミス・アクセスが発生し、L2 OzQ のスループットを制限するため、この問題は重要である。

1 つの解決策は、3 つの `lfetch` 命令を個別に使用することである。`lfetch.fault.nt1` がデータを L2 に入れる。その後、データが L2 内にあるとき、2 つの `lfetch.fault` 命令が L2 キャッシュにヒットし、データを L1D に入れる。これらの `lfetch` 命令は非対称であり、複数のロード・メモリ・スロットを必要とする。

上記の 3 つの `lfetch` 手法を最適化した手法では、2 つの `lfetch.fault` 命令のみを段階的に使用する。つまり、最初の命令が L2 と L1D にデータを入れる。次に、最初の要求によって L2 がフィルされたとき、第 2 の `lfetch` は、2 次的な L2 ミスにならずに、L1D にデータを入れられる (L2 がフィルされているため、第 2 の `lfetch` は L2 ヒットになる)。これによって、追加のロード・メモリ・スロットが解放され、`lfetch` 命令が再使用可能になる。

実行待ちの L1D フィルは、同じラインに対するストアによって無効にされる。同じラインに対するストアが検出される前に `lfetch` で L1D をフィルできれば、小さなデータ・ストリームであっても、`lfetch` 命令を使用することでパフォーマンスが大きく向上する。

また、L1D にヒットしたすべてのロードは L2 OzQ 内に割り当てられないため、`lfetch` 命令を使用して L1D ヒットを保証すれば、L2 OzQ の内容をストア・データと `lfetch` 要求のみに制限でき、パフォーマンス上のメリットがある。これによって、限られた OzQ リソースへの圧迫が軽減され、OzQ エントリ間の競合の可能性が小さくなる。

9.3.3 ストア・データ・ストリーム

ストア命令は常に L2 によって検出されるため、ストアのデスティネーション・データを L1D に入れてもメリットはない。デスティネーション・ストリームに対して `lfetch.fault.excl.nt1` コンプリータを使用すると、多くのメリットがある。例えば、`.nt1` ヒントにより、2 次的な L2 ミスを取り除くことができ、L1D フィルによるコアのストールを回避できる。また、`.excl` ヒントは、L2 データがストア・データを受け入れられることを保証する。

9.4 コントロール・スペキュレーションとデータ・スペキュレーション

Itanium 2 プロセッサは、早期据え置きモードとレイテンシの小さい修正機能により、ALAT 内のコントロール・スペキュレーションとデータ・スペキュレーションに関連するコストを軽減する。したがって、コードの生成を調整してスペキュレーションを積極的に使用すると、パフォーマンスの向上を実現できる。スペキュレーションに関する注意事項のうちいくつかは Itanium プロセッサに固有であり、Itanium 2 プロセッサには適用されない。スペキュレーションを積極的に利用するほど、修正コードの呼び出し回数が増える。Itanium プロセッサでは、多くの場合、修正コードは実際のスペキュレーションから遠く離れたコールド・ページに移動させられていた。スペキュレーション・ポイントからどの程度の距離に修正コードを置くかについての経験的手法を再検討し、決定マトリックスにプロファイル情報を含める必要がある。

9.5 確認済みの L2 ミス・バンドル配置

Itanium 2 プロセッサの設計では、L2 キャッシュ・ミスになることがわかっている命令をメモリ・ポート 0 に入れる（発行グループ内の最初のメモリ・オペレーションを割り当てると、多少メリットがある。これによって、可能な場合は、L3 に対する見込み的な要求を生成できる。L2 に送られる必要があるメモリ要求が M1、M2、または M3 内にあると、その要求は L2 OzQ から再発行可能になるまで待たなければならない。

9.6 確認済みの L2 キャンセル条件および再循環条件の回避

最も予測しやすい L2 キャンセル条件は、L2 バンクの競合である。これを回避するには、L2 アクセスを注意深く順序づけるか、あるいは `lfetch` 命令を使用して LID にデータを入れ、L2 へのアクセスを避ける。

最も予測しやすい L2 再循環条件は、2 次的な L2 ミスのアクセスである。これを回避するには、`lfetch` 命令を使用して、L2 にデータを入れる。2 次的なアクセスとしてカウントされないのは、LID のフィルを行わない `lfetch` 命令だけである。`lfetch` が 1 次的な L2 ミスになり、ロードが 2 次的な L2 ミスになった場合は、ロードは最終的にはコアにデータを返さなければならないため、ロードの再循環が必要になる。この状態を避けるには、L2 ミスになる `lfetch` 命令を、ロードより十分に先行してスケジューリングする必要がある。

9.7 命令バンドリング

Itanium 2 プロセッサは、ほぼすべてのバンドル・テンプレートの組み合わせを漏れなく発行できる。ILP が使用可能である場合、適切なバンドリングと命令スケジューリングを行えば、パフォーマンスは向上する。ここで 2 つの推奨事項がある。それは、制約条件の厳しい命令を発行グループ内で先に配置することと、制約条件の厳しい命令をできるだけ変換することである。例えば、簡単な命令 `nop.i` は I ポートに発行しなければならないが、加算命令は M ポートと I ポートのどちらでも発行できる。この場合、`nop.i` が必ず I ポートを使用できるように、この命令を先にスケジューリングすべきである。また、I ポートと M ポートのどちらでも発行できる、実質的に `nop` と同じ機能を持つ命令 (`add r3=r0, r3` など) で、`nop.i` を置き換える方法もある。

9.8 分岐

分岐および分岐予測に関連する以下の最適化の推奨事項は、第 7 章「分岐命令と分岐予測」で詳しく説明している。この項は、第 7 章の内容をまとめたものである。

9.8.1 1 サイクル分岐

Itanium 2 プロセッサで 1 サイクル・ループ分岐を処理すると、ループが何回か反復され、多少のペナルティが発生する。ループを少なくとも 2 サイクルにアンロールすれば、予想どおりのパフォーマンスが得られる。ただし、コードのサイズは多少大きくなる。

9.8.2 完全ループ予測

完全ループ予測は、ループの最後の反復だけを予測する。したがって、Itanium 2 プロセッサは、この分岐を予測する際に分岐ヒントを考慮に入れる。Itanium 2 プロセッサでは、`ar.ec` が正しく設定されている必要がある。つまり、エピローグがない場合は、`ar.ec=0` に設定する必要がある (Itanium プロセッサでは、この値を 1 に設定する必要があった)。

9.8.3 分岐ターゲット

フロントエンドが 1 サイクルにつき 2 バンドルをバックエンドに配布できるように、分岐ターゲットは 32 バイト境界にアライメントが合っている必要がある。

10.1 概要

本章では、Itanium® 2 プロセッサのパフォーマンス監視機能について説明する。Itanium 2 プロセッサは、4つの48ビット・パフォーマンス・カウンタ、100以上の監視可能イベント、各種の高度な監視機能を備えている。本章では、各パフォーマンス・モニタの使用モデル、パフォーマンス監視ソフトウェアのインターフェイスとプログラミング・モデル、一連の監視対象イベントを説明する。

Itanium アーキテクチャに組み込まれたアーキテクチャ上の機構によって、パフォーマンス監視ソフトウェアは、分岐予測構造、プロセッサのデータ/命令キャッシュ、仮想メモリ変換構造などの高性能が要求されるプロセッサ・リソースを、アクティブかつ直接に管理できる。最高の性能レベルを達成するために、状況に応じて変化するプロセッサ動作を監視し、その結果をコード生成プロセスにフィードバックすると、観察されたランタイム動作のエンコードや命令レベルでの並列性を向上できる。Itanium プロセッサの第2世代となる Itanium 2 プロセッサでは、実際の Itanium ベースのアプリケーションやオペレーティング・システムの動作だけではなく、IA-32 ベースのコードと Itanium アーキテクチャ・ベースのコードが混在したコードの動作も計測できる。これらの計測結果は、コンパイラの最適化の動作、スペキュレーションや分岐予測などのアーキテクチャ上の機能の使用、あるいは ALAT、キャッシュ、TLB などのマイクロアーキテクチャ構造の効率を理解するために重要である。これらの計測結果によって、アプリケーションのチューニングおよび将来のプロセッサ、コンパイラ、オペレーティング・システムの設計を容易にするためのデータが得られる。

以降では、以下の内容について説明する。

- 10.2 節「パフォーマンス・モニタのプログラミング・モデル」では、パフォーマンス・モニタの使用手法と、Itanium 2 プロセッサのパフォーマンス・モニタの各種プログラミング・モデルを説明する。
- 10.3 節「パフォーマンス・モニタのステート」では、Itanium 2 プロセッサ固有の PMC/PMD パフォーマンス監視レジスタを定義する。
- 第 11 章「パフォーマンス監視イベント」では、Itanium 2 プロセッサのイベント・リストの概要を説明する。

10.2 パフォーマンス・モニタのプログラミング・モデル

本節では、Itanium 2 プロセッサのパフォーマンス監視機能について、プログラミング・モデルの見地から説明し、各種のイベント監視機構の効果的な使用方法を説明する。Itanium 2 プロセッサのパフォーマンス監視アーキテクチャは、次の2つの使用モデルに焦点をあてている。

- 作業負荷の特性評価：パフォーマンス分析の出発点は、計測される作業負荷のパフォーマンス特性を理解することである。10.2.1 項「作業負荷の特性評価」では、作業負荷の特性評価を可能にする Itanium 2 プロセッサの機能を説明する。
- プロファイリング：プロファイリングは、アプリケーション開発者とプロファイル方式のコンパイラによって使用される。アプリケーション開発者は、パフォーマンスのボトルネックを特定し、それらをコードに関連付ける必要がある。アプリケーション開発者の主な目的は、モジュール・レベル、関数レベル、基本ブロック・レベルで、パフォーマンス低下の原因になっているプログラムの箇所を理解することである。データの配置を最適化し、重要なループを分析するためには、命令レベルを分析の最小単位にする必要がある。また、プロファイル方式のコンパイラは、分岐予測やスペキュレーションなどの高度な Itanium アーキテクチャ機能によって、ランタイム・プロファイル情報を使用して命令のスケジュールを最適化する。Itanium 2 プロセッサは、分岐の予測ミスとキャッシュ・ミスに関して、命令レベルでの統計

的プロファイリングをサポートする。Itanium 2 プロセッサのプロファイリング機能の詳細は、[10.2.2 項「プロファイリング」](#)を参照のこと。

10.2.1 作業負荷の特性評価

パフォーマンス分析の出発点は、計測される作業負荷のパフォーマンス特性を理解することである。このために、イベント発生率とプログラム・サイクルの内訳の 2 つの基本的な計測基準が用意されている。

- イベント発生率の監視：計測可能なイベント発生率には、アプリケーション全体で計測される、1 クロック当たりの平均リタイア命令数 (IPC)、データ / 命令キャッシュ・ミス率、分岐の予測ミス率などがある。オペレーティング・システムまたは大規模な市販アプリケーションの特性評価 (例えば、OLTP 分析) を行うには、TLB ミス率、VHPT ウォーク / 秒、割り込み数 / 秒、またはバス利用率などのパフォーマンスに関連するイベントを、システム・レベルで観察する必要がある。イベント発生率の監視については、[10.2.1.1 項「イベント発生率の監視」](#)で説明する。
- サイクル・アカウンティング：作業負荷のサイクルの内訳は、プログラムによって消費されたサイクルの原因ごとの内訳を示す。サイクル数の増加は、プログラム内部の実行レイテンシ以外に、通常はパイプラインのストールおよびフラッシュによって発生する。サイクル・アカウンティングについては、[10.2.1.4 項「サイクル・アカウンティング」](#)で説明する。

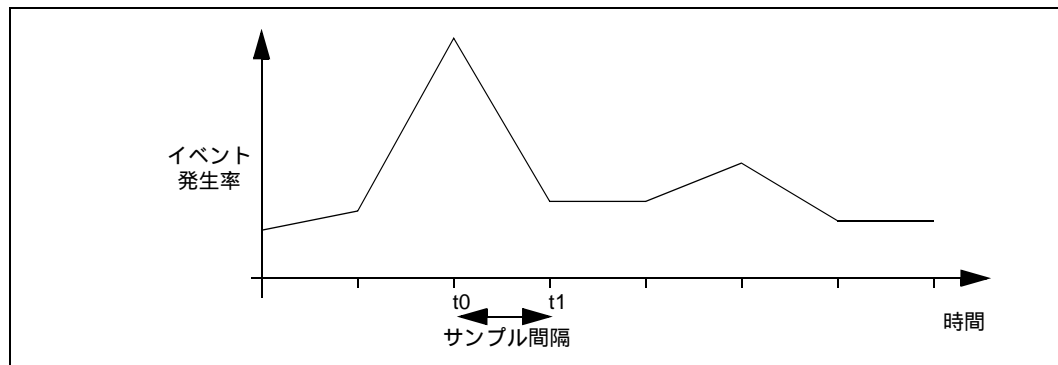
10.2.1.1 イベント発生率の監視

イベント発生率の監視は、作業負荷の実行の前後にプロセッサ・イベント発生カウンタを読み取って、監視対象のイベント発生率を計算する。例えば、Itanium 2 プロセッサの 2 つの基本的なイベント (リタイアした Itanium 命令の数をカウントするイベント (IA64_INST_RETIRED.u) と、経過したクロック・サイクル数をカウントするイベント (CPU_CYCLES)) を使用して、作業負荷の 1 サイクル当たりの命令数 (IPC) を次のように計算できる。

$$\text{IPC} = (\text{IA64_INST_RETIRED.u}_{t1} - \text{IA64_INST_RETIRED.u}_{t0}) / (\text{CPU_CYCLES}_{t1} - \text{CPU_CYCLES}_{t0})$$

時間ベースのサンプリングは、多くのパフォーマンス・デバッグ・ツール [VTune(TM)、gprof、WinNT] の基礎となる。[図 10-1](#) に示すように、時間ベースのサンプリングを使用して、時間とともに変化するイベント発生率をプロットできる。これによって、作業負荷が移行していくさまざまな状態を推定できる。

図 10-1. 時間ベースのサンプリング



Itanium プロセッサ上では、多くのイベント・タイプ (例えば、TLB ミスや分岐の予測ミス) の発生回数は、1 クロック・サイクル当たり 1 回までに制限されている。このようなイベントは、「単一発生」イベントと呼ばれる。しかし、Itanium 2 プロセッサ上では、同じタイプの複数のイベントが同一クロック内で発生する場合がある。このようなイベントは、「複数発生」イベントと呼ばれる。Itanium 2 プロセッサの複数発生イベントの例は、データ・キャッシュ読み出しミス (1 クロック当たり最大 2 つ) である。メモリ要求キュー内のエントリ数などの複数発生イベントを使用

して、メモリ・アクセスの平均回数と平均レイテンシを計算できる。次の2つの項では、単一発生イベントと複数発生イベントを監視するための Itanium 2 プロセッサの基本的な機構を説明する。

10.2.1.2 単一発生イベントと持続時間のカウンタ

単一発生イベントは、Itanium 2 プロセッサの任意のパフォーマンス・カウンタによって監視できる。すべての単一発生イベントについて、それに対応するカウンタは、1クロック・サイクル当たり最大1ずつインクリメントされる。ある状態が持続するクロック・サイクル数をカウントする持続時間カウンタは、「単一発生」イベントと見なされる。Itanium 2 プロセッサの単一発生イベントの例としては、TLB ミス、分岐の予測ミス、サイクルベースの評価基準などがある。

10.2.1.3 複数発生イベント、スレッシュホールド、平均計算

ハードウェアの並列性のために、1クロック・サイクル当たり2つ以上発生する可能性があるイベントは、「複数発生」イベントと呼ばれる。Itanium 2 プロセッサの複数発生イベントの例には、リタイアした命令数や、メモリ要求キュー内の有効エントリ数などがある。

スレッシュホールド機能は、Itanium 2 プロセッサの複数発生カウンタ内で使用できる。この機能を使用して、イベント分布ヒストグラムをプロットできる。ゼロでないスレッシュホールドが指定されている場合は、検出されたイベント数が設定されたスレッシュホールドを超えたサイクルごとに、モニタが1ずつインクリメントされる。これによって、「メモリ要求キュー内に3つ以上のエントリが入っていたサイクルの数は？」または「マシンが4つ以上の命令をリタイアさせたサイクルの数は？」などの問いに答えられる。この機能は、マイクロアーキテクチャ上のバッファ・サイズのテストを、実際の計測値によって支援できる。さまざまなスレッシュホールド値を指定してベンチマークを実行すると、あるバッファ・サイズでのパフォーマンスを特定するヒストグラムを作成できる。

未処理のメモリ操作など、重複する同時イベントは、同時に未処理になっている要求の平均数と、要求が未処理になっている期間の平均サイクル数が問題になる。メモリ・キュー内の複数の未処理の要求の平均数または平均レイテンシを計算するには、要求の合計回数 (n_{total}) と、1サイクル当たりの有効な要求の数 ($n_{live}/cycle$) がわかっていなければならない。複数発生カウンタを使用して有効な要求の数 ($n_{live}/cycle$) を合計すると、 $\sum n_{live}$ はハードウェアによって直接計測される。要求の平均数と平均レイテンシを次のように計算できる。

- 1サイクル当たりの未処理の要求の平均数 = $\sum n_{live} / \Delta t$
- 要求1回当たりの平均レイテンシ = $\sum n_{live} / n_{total}$

表 10-1 に、この計算の例を示す。この表では、1サイクル当たりの未処理の要求の平均数 = $15/8 = 1.825$ 、要求1回当たりの平均レイテンシ = $15/5 = 3$ サイクルである。

表 10-1. 要求1回当たりの平均レイテンシと1サイクル当たりの要求数の計算例

時間 [サイクル]	1	2	3	4	5	6	7	8
要求数 (In)	1	1	1	1	1	0	0	0
要求数 (Out)	0	0	0	1	1	1	1	1
n_{live}	1	2	3	3	3	2	1	0
$\sum n_{live}$	1	3	6	9	12	14	15	15
n_{total}	1	2	3	4	5	5	5	5

Itanium 2 プロセッサは、イベント発生率を監視するための以下の機能を搭載している。

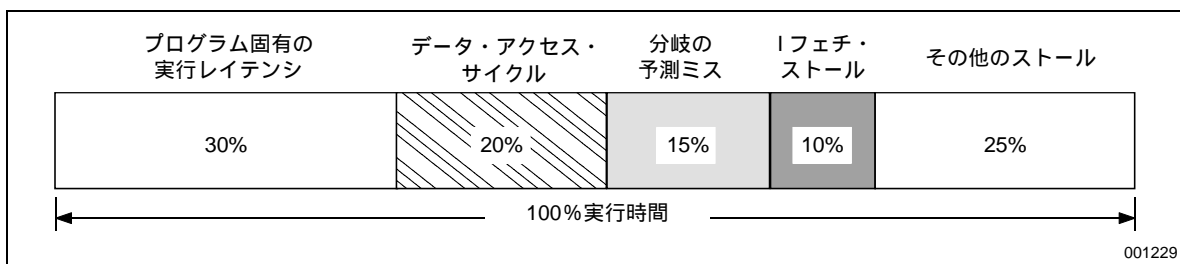
- クロック・サイクル・カウンタ
- リタイア命令カウンタ
- イベント発生カウンタおよび持続時間カウンタ

- ・ スレッシュホールド機能付き複数発生カウンタ

10.2.1.4 サイクル・アカウンティング

イベント発生率の監視では、イベントの数をカウントするが、検出されたイベントがパフォーマンスに影響を与えているかどうかはわからない。よく使用される手法は、複数のイベント発生率をプロットし、1サイクル当たりの命令数 (IPC) の計測値に関連付ける方法である。キャッシュ・ミス動作のピークと同時に IPC の値が低下している場合は、キャッシュ・ミスがパフォーマンス上の問題の原因であると推定される。このような推定作業を不要にするために、Itanium 2 プロセッサは、一連のサイクル・アカウンティング・モニタを搭載しており、各種のマイクロアーキテクチャ上のイベントによって失われたサイクル数の内訳を調べられる。図 10-2 に示すように、この機能は、プログラムによって消費されたサイクルの内訳を示すため、アプリケーションのマイクロアーキテクチャ上の動作を推定できる。ただし、サイクル・アカウンティングは、単なるストールまたはフラッシュの持続時間のカウントとは異なる。サイクル・アカウンティングは、マシンの実際のストール/フラッシュ状態に基づいて、重複したパイプラインの遅延の原因を示す。単なるストールやフラッシュの持続時間カウンタは、このような機能を持っていない。サイクル・アカウンティングは、ストールとフラッシュを原因とするプログラムのサイクルの内訳を調べられる。単なる持続時間カウンタは、ストールまたはフラッシュの累積レイテンシを計算するときにより便利である。

図 10-2. Itanium® プロセッサ・ファミリのサイクル・アカウンティング



Itanium 2 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。バックエンド・ストールとフラッシュの発生原因は 6 つあるが、その優先順位は次のとおりである。

1. 例外 / 割り込みサイクル : 割り込みおよび例外が原因のパイプのフラッシュに費やされるサイクル数
2. 分岐の予測ミス・サイクル : 分岐の予測ミスが原因のパイプのフラッシュに費やされるサイクル数
3. データ / FPU アクセス・サイクル : メモリ・パイプライン・フル、データ TLB ストール、load-use ストール、浮動小数点ユニットへのアクセス
4. 実行レイテンシ・サイクル : スコアボード・ストールおよびその他のレジスタ依存性ストール
5. RSE 実行サイクル : RSE によるスピル / フィル・ストール
6. フロントエンド・ストール : フロントエンド上で待機しているバックエンドが原因のストール

フロントエンド・ストールの発生には 7 つの原因が考えられるが、そうした原因が詳しくわかるフロントエンド・ストール・カウンタもさらに用意されている。ただし、バックエンド・ストール・イベントとフロントエンド・ストール・イベントは、それぞれ別のパイプライン・ステージでカウントされるため、比較するべきではない。

詳細は、11.6 節「ストール・イベント」を参照のこと。

10.2.2 プロファイリング

プロファイリングは、アプリケーション開発者とプロファイル方式のコンパイラが、リンクとランタイム・システムを最適化するときに使用する。アプリケーション開発者は、パフォーマンスのボトルネックを特定し、それらをソース・コードに関連付ける必要がある。アプリケーション開発者は、プロファイルのフィードバックに基づいて、プログラムの高レベル・アルゴリズムとデータ構造を修正できる。コンパイラは、分岐予測やスペキュレーションなどの高度な Itanium アーキテクチャ機能によって、プロファイルのフィードバックを使用して命令のスケジュールを最適化できる。

プロファイリングを実行するには、パフォーマンス・モニタのカウントが、プログラム内の位置に関連付けられる必要がある。Itanium 2 プロセッサのパフォーマンス・モニタは、以下の機構を直接サポートする。

- プログラム・カウンタのサンプリング
- ミス・イベント・アドレスのサンプリング : Itanium 2 プロセッサのイベント・アドレス・レジスタ (EAR) は、きめ細かな分析を必要とするイベント (命令キャッシュ、データ・キャッシュ、分岐の予測ミス、命令 TLB、データ TLB) のために、パイプライン長の一部を最小単位とする分析を可能にする。
- 監視対象となるイベントの制限 : イベント監視機能の対象を、特定の命令アドレス範囲、オペコード、または特権レベルだけに制限する。

次の 3 つの項では、これらのプロファイリング機能について説明する。

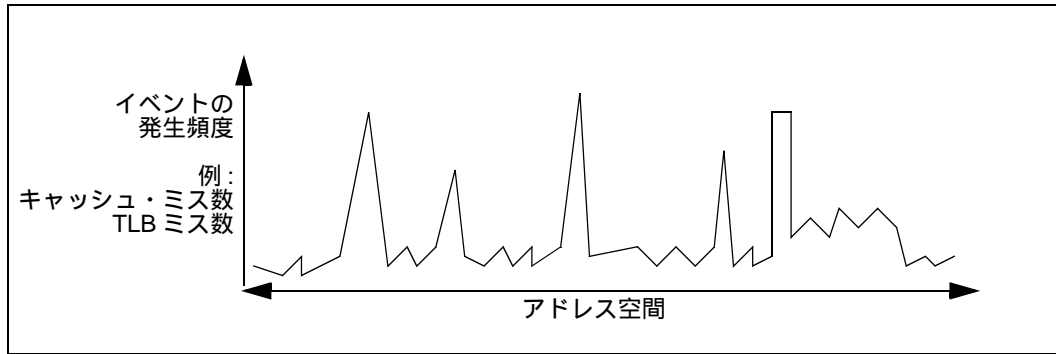
10.2.2.1 プログラム・カウンタのサンプリング

[VTune, gprof] などのアプリケーション・チューニング・ツールでは、プログラム・カウンタなどのイベント・カウンタを時間ベースまたはイベントベースでサンプリングして、パフォーマンスに大きな影響を与える関数と基本ブロックを特定する。図 10-3 のように、命令アドレスを横軸にして、サンプリングされたポイントをヒストグラムで示すことができる。従来、アプリケーションのチューニングには、統計的サンプリング手法が非常に効果的であった。これは、プログラマが、処理に時間がかかる箇所や特定のイベント数が増える箇所など、コードのホットスポットを簡単に特定できるためである。

プログラム・カウンタ・サンプリングでは、パフォーマンス分析者はコードのホットスポットを特定できるが、パフォーマンスの問題の原因はわからない。パフォーマンスの問題の根本的な原因を特定するには、ホットスポットの範囲を手作業で分析した上で、難しい推定を行わなければならない。Itanium 2 プロセッサは、(10.2.1.4 項「サイクル・アカウンティング」で説明した) サイクル・アカウンティング機構を使用して、アプリケーションのマイクロアーキテクチャ上の動作を直接計測できる。

時間ベースのプログラム・カウンタ・サンプリングには、Itanium アーキテクチャのインターバル・タイマ機能 (ITC および ITM レジスタ) を使用できる。イベントベースのプログラム・カウンタ・サンプリングは、専用のパフォーマンス・モニタ・オーバーフロー割り込み機構によって可能になる。詳細は、『インテル® Itanium® 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 7.2.2 項「パフォーマンス・モニタ・オーバーフロー・ステータス・レジスタ (PMC[0] ~ PMC[3])」を参照のこと。

図 10-3. プログラム・カウンタによるイベント・ヒストグラム



Itanium 2 プロセッサは、プログラム・カウンタ・サンプリングをサポートするために、以下の機構を搭載している。

- ・ 時間ベースのプログラム・カウンタ・サンプリング用のタイマ割り込み
- ・ イベントベースのプログラム・カウンタ・サンプリング用のイベント数オーバーフロー割り込み
- ・ ハードウェア上でサポートされるサイクル・アカウンティング機能

10.2.2.2 ミス・イベント・アドレス・サンプリング

プログラム・カウンタ・サンプリングとサイクル・アカウンティングによって、累積的なマイクロアーキテクチャ動作を正確に認識できる。しかし、アプリケーション開発者は、マイクロアーキテクチャ上の「ミス・イベント」を繰り返し発生させる、特定のプログラム要素（コード内の位置およびデータ構造）へのポインタを必要とする。SPEC92 ベンチマークのキャッシュに関する研究で、[Lebeck] は、トレースベースのキャッシュ・ミス・プロファイリングを使用して、ソース・コードに簡単な変更を加えると、各種のベンチマーク上で 1.02 ~ 3.46 に相当するパフォーマンスの向上を達成した。このような分析を行うには、キャッシュ・ミス、分岐の予測ミス、TLB ミスなどのマイクロアーキテクチャ上の「ミス・イベント」に対応する命令/データ・アドレスを特定する必要がある。これらのアドレスは、シンボル・テーブルまたはコンパイラの注釈を使用して、重要なソース・コード要素に対応付けられる。従来のパフォーマンス分析者は、Lebeck と同じように、ハードウェアのトレースを収集し、トレース方式のシミュレーションを実行してきた。

今日のマイクロアーキテクチャは、スーパースケラ、深いパイプライン、命令のアウト・オブ・オーダー実行を利用するため、サンプリングされたプログラム・カウンタの値が、ミス・イベントの原因となった命令アドレスに対応しない場合がある。Pentium® プロセッサのパイプライン上では、サンプリングされたプログラム・カウンタが、ミス・イベントの原因となった命令から、動的命令 2 つ分だけずれることがある。Pentium Pro プロセッサ上では、この距離が約 32 個の動的命令に増える。Itanium 2 プロセッサ上では、この距離は約 48 個の動的命令に相当する。Itanium 2 プロセッサ上でプログラム・カウンタ・サンプリングを使用してミス・イベント・アドレスを特定する場合、ミス・イベントは、実際に発生した位置から動的な基本ブロック約 5 つ分だけ離れた命令に対応付けられるときがある（すべての命令のうち 10% が分岐命令とする）。したがって、ハードウェアがイベントのアドレスを正確に特定する必要がある。

Itanium 2 プロセッサは、一連のイベント・アドレス・レジスタ (EAR) を搭載している。これらのレジスタは、ロード時のデータ・キャッシュ・ミスの命令アドレスとデータ・アドレス、データ TLB ミスの命令アドレスとデータ・アドレス、命令 TLB ミスと命令キャッシュ・ミスの命令アドレスを記録する。4 つの深い分岐トレース・バッファは、分岐命令のシーケンスを収集する。表 10-2 は、Itanium 2 プロセッサの EAR と分岐トレース・バッファの機能をまとめたものである。ソフトウェアがミス・イベント・アドレスを参照できるようにすると、サンプリングまたはコードのインストルメンテーションにより、ミス・イベント・アドレスを監視できる。これによって、パフォーマンスの問題を特定して解決するためにトレースを生成する必要がなくなるため、はるかに多くの開発者が、ハードウェアの追加なしでパフォーマンス分析を実行できるようになる。

表 10-2. Itanium® 2 プロセッサの EAR と分岐トレース・バッファ

イベント・アドレス・レジスタ	トリガ条件	記録内容
命令キャッシュ	L1 命令キャッシュ・ミスになった命令フェッチ (デマンド・フェッチのみ)	命令アドレス フェッチが遅延したサイクル数
命令 TLB (ITLB)	L1 ITLB ミスになった命令フェッチ (デマンド・フェッチのみ)	命令アドレス L1 ITLB ミスを発生させたコンポーネント : L2 ITLB VHPT またはソフトウェア
データ・キャッシュ	L1 データ・キャッシュ・ミスになったロード命令	命令アドレス データ・アドレス ロードが遅延したサイクル数
データ TLB (DTLB)	L1 DTLB ミスになったデータ参照	命令アドレス データ・アドレス L1 DTLB ミスを発生させたコンポーネント : L2 DTLB、VHPT、またはソフトウェア
分岐 トレース・ バッファ	分岐の結果	分岐命令アドレス 分岐ターゲットと命令アドレス 予測ミスの状態と原因

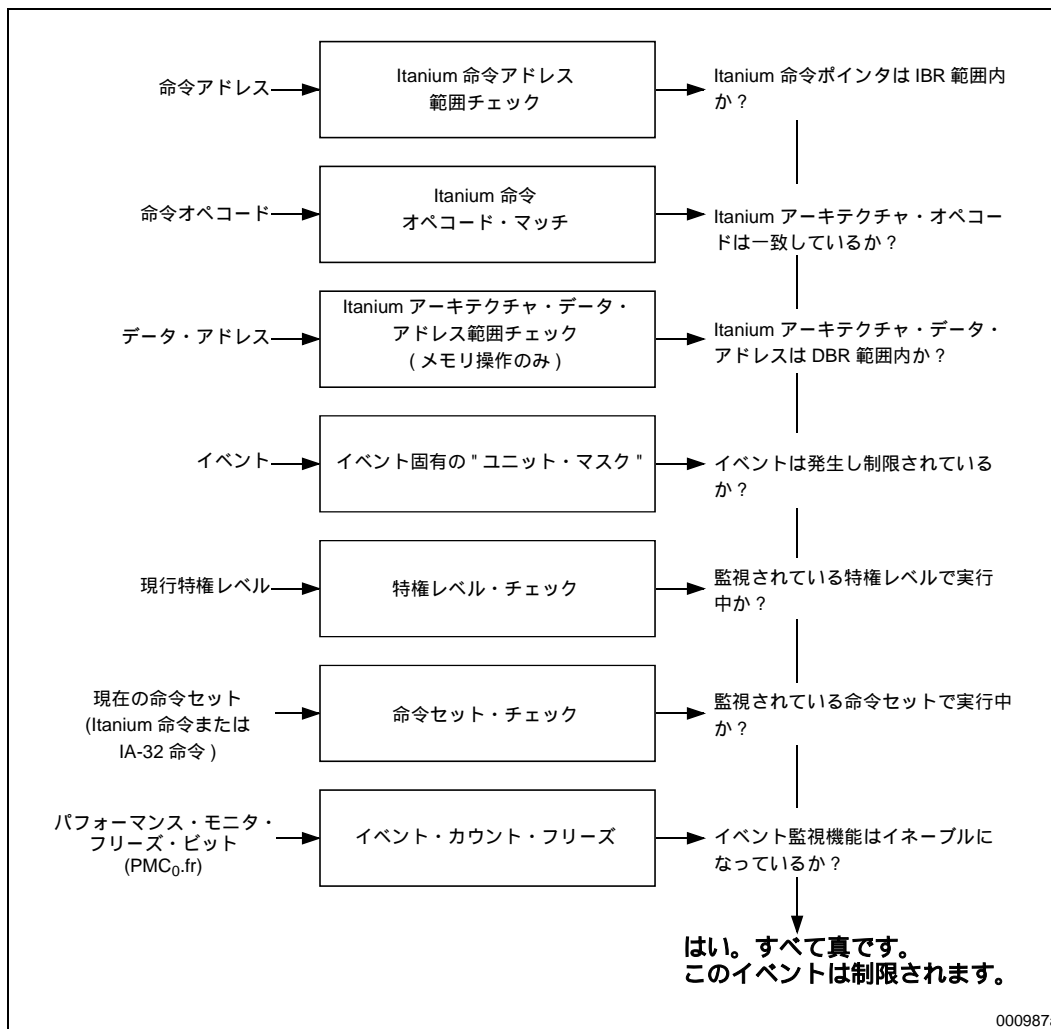
Itanium 2 プロセッサの EAR を使用して、例えば、データ・キャッシュ・ミスまたはリタイアした命令数をカウントするようにパフォーマンス・カウンタを設定して、統計的サンプリングを実行できる。パフォーマンス・カウンタの値は、あらかじめ指定された数のイベントが検出された時点でプロセッサに割り込みをかけるように設定される。データ・キャッシュ・イベント・アドレス・レジスタは、実際のデータ・キャッシュ・ロード・ミスの命令アドレスとデータ・アドレスを繰り返し収集する。カウンタがオーバーフローするたびに、パフォーマンス監視ソフトウェアがイベント・アドレス・レジスタを読み取るまで、ミス・イベント・アドレスの収集は中止される (これにより、ソフトウェア自体の監視で発生したミス・イベントが収集されるのを防ぐ)。カウンタがオーバーフローすると、ソフトウェアに割り込みが伝えられ、監視されるイベントのアドレスが収集され、パフォーマンス・カウンタ・レジスタを書き換えることによって新しい監視区間が設定可能になる。時間ベースの (イベントベースではない) サンプリング手法では、イベント・アドレス・レジスタは、監視対象となるイベントが収集されたかどうかをソフトウェアに示す。統計的サンプリングは、1つの監視区間内のイベントの数を変えたり、監視区間の数を増やすと、任意の単位でイベントを分析できる。

10.2.3 監視対象となるイベントの制限

Itanium 2 プロセッサ上では、パフォーマンス監視の対象となるイベントを、一部のイベントだけに制限できる。図 10-4 に示すように、命令アドレス範囲、特定の命令オペコード、データ・アドレス範囲、イベント固有の「単位マスク」、イベントを発生させた特権レベルと命令セット、およびパフォーマンス・モニタ・フリーズ・ビット (PMC_{0,fr}) の状態に基づいて、監視対象となるイベントを制限できる。

- Itanium 命令アドレス範囲チェック : Itanium 2 プロセッサでは、設定可能な命令アドレス範囲に基づいて、監視対象となるイベントを制限できる。これによって、大規模な Itanium ベース・アプリケーション内の特定のダイナミック・リンク・ライブラリ (DLL)、関数、またはループの監視が可能になる。Itanium 命令アドレス範囲チェックは、パイプラインの命令フェッチ段階で適用される。チェック条件を満たした命令は、そのパイプライン全体を通じて監視対象となる。この方法で、パイプラインの動的命令長 (約 48 個の命令) より小さい長さを最小単位とする条件付きイベント・カウントが可能になる。Itanium 2 プロセッサの命令アドレス範囲チェックは、Itanium アーキテクチャ・ベースのコードの実行中 (すなわち、PSR.is が 0 の場合) にのみ有効である。詳細は、Itanium アーキテクチャ・オペコード・マツチ・レジスタおよびアドレス範囲チェック・レジスタ (PMC_{8,9}) を参照のこと。

図 10-4. Itanium® 2 プロセッサの監視対象となるイベントの制限



- Itanium 命令オペコード・マッチ : Itanium 2 プロセッサは、独立した 2 つの Itanium アーキテクチャ・オペコード・マッチ・レジスタを搭載している。各マッチ・レジスタによって、現在発行されている命令のエンコーディングと設定可能なオペコード・マッチ / マスク関数が照合される。一致したイベントを、パフォーマンス・カウンタによってカウントされる命令タイプとして選択できる。この方法で、命令タイプのヒストグラムの作成や、(タグ付き NOP の挿入による) デスティネーション・レジスタおよびプレディケート・レジスタと基本ブロックの使用状況のプロファイリングが可能になる。オペコード・マッチャーは、Itanium アーキテクチャ・ベースのコードの実行中 (すなわち、PSR.is が 0 の場合) にのみ有効である。詳細は、10.3.4 節を参照のこと。
- Itanium アーキテクチャ・データ・アドレス範囲チェック : Itanium 2 プロセッサは、設定可能なデータ・アドレス範囲に基づいて、監視対象となるメモリ操作イベントを制限できる。この方法で、特定のデータ構造のデータ・キャッシュ・ミス動作を選択した上で監視できる。詳細は、10.3.6 項を参照のこと。
- イベント固有のユニット・マスク : いくつかのイベントは、「ユニット・マスク」を指定して、監視されるユニットで直接に、必要なイベントを選別することができる。例えば、イベント固有のユニット・マスクを使用して、カウントするバス・トランザクションの数を制限すれば、バス・エージェントからのトランザクション、プロセッサ自体からのトランザクション、またはその他の I/O バス・マスタからのトランザクションを含めることができる。この場合、

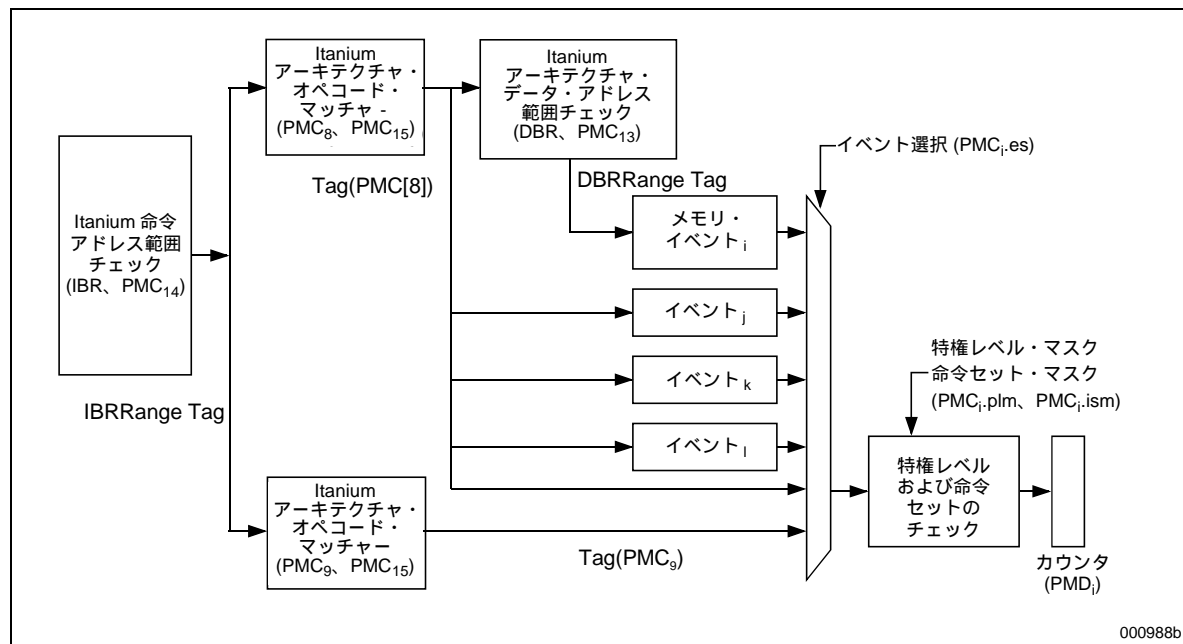
バス・ユニットは、カウント対象のトランザクションを指定する3ウェイのユニット・マスク(任意のバス・エージェント、プロセッサ自体、またはI/Oバス・マスタ)を使用する。Itanium 2 プロセッサの場合、分岐、メモリ、バス・ユニットからのイベントは、各種のユニット・マスクをサポートする。詳細は、第11章「パフォーマンス監視イベント」のイベントに関するページを参照のこと。

- 特権レベル: プロセッサ・ステータス・レジスタ内の2ビットを使用して、プロセスに基づきイベントを選択した上で監視ができる。Itanium 2 プロセッサは、現行特権レベルに基づいた条件付きイベント・カウント機能をサポートしている。これによって、パフォーマンス監視ソフトウェアは、ユーザによるイベントとオペレーティング・システムによるイベントを区別して、イベント数の内訳を調べられる。特権レベルによる監視の制限は、10.3.1項「パフォーマンス・モニタの制御とアクセス」を参照のこと。
- 命令セット: Itanium 2 プロセッサは、各イベント・モニタにつき2つの命令セット・マスク・ビットによる、現在実行中の命令セット (Itanium 命令または IA-32 命令) に基づいた条件付きイベント・カウント機能をサポートしている。これによって、パフォーマンス監視ソフトウェアは、Itanium 命令によるイベントと IA-32 命令によるイベントを区別して、イベント数の内訳を調べられる。詳細は、10.3.1項「パフォーマンス・モニタの制御とアクセス」を参照のこと。
- パフォーマンス・モニタ・フリーズ: イベント・カウンタのオーバーフローまたはソフトウェアによって、イベント監視機能がフリーズされるときがある。監視機能がフリーズされると、ソフトウェアがモニタ・フリーズ・ビット (PMC_{0,fr}) をクリアするまで、すべてのイベント監視は中止される。これによって、パフォーマンス監視ルーチンそれ自体 (例えば、カウンタ・オーバーフロー割り込みハンドラまたはパフォーマンス監視コンテキスト・スイッチ・ルーチン) が、観察されているシステムのイベント数に影響を与えることを防げる。詳細は、『インテル® Itanium® 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、7.2.4項を参照のこと。

10.2.3.1 オペコード・マッチ、命令アドレス範囲チェック、データ・アドレス範囲チェックの組み合わせ

Itanium 2 プロセッサは、図 10-5 に示す命令タグ付け機構を使用して、各種のイベント制限機構を組み合わせられる。

図 10-5. Itanium® 2 プロセッサの命令タグ付け機構



Itanium 命令の実行時には (PSR.is=0)、最初に命令アドレス範囲チェックが適用される。得られたアドレス範囲チェック・タグ (IBRRangeTag) は、2つのオペコード・マッチャーに渡される。これらのオペコード・マッチャーは、命令アドレス範囲チェックとオペコード・マッチを行う。得られた2つのタグ (Tag(PMC₈) および Tag(PMC₉)) のそれぞれは、リタイア命令カウント・イベントとしてカウントできる (詳細は、11-147 ページの「IA64_TAGGED_INST_RETIRED」のイベントの説明を参照のこと)。

アドレス範囲 / オペコード・マッチ・タグのうち1つ (Tag(PMC₈)) が、すべてのダウン・ストリームのパイプライン・イベントの選択条件を指定する。メモリ階層内のイベント (L1/L2 データ・キャッシュ・イベントおよびデータ TLB イベント) は、データ・アドレスの DBRRangeTag によってさらに制限ができる。

表 10-3 にまとめたように、Itanium 2 プロセッサ上では、データ・アドレス範囲チェックをオペコード・マッチおよび命令範囲チェックと組み合わせられる。また、現行特権レベルおよび現在実行中の命令セットに基づいた追加のイベント制限を、すべてのイベントに適用できる。これについては、10.2.3.2 項「特権レベルによる制限」および 10.2.3.3 項「命令セットによる制限」で説明する。

表 10-3. Itanium® 2 プロセッサのイベント制限モード

イベント制限モード	オペコード・マッチ・イネーブル PMC ₁₅ .ibrp0-pmc8	オペコード・マッチ PMC ₈	命令アドレス範囲チェック・イネーブル PMC ₁₄ .ibrp0	データ・アドレス範囲チェック [PMC ₁₃ .enable-dbrp# PMC ₁₃ .dbrp#] (メモリ・パイプ・イベントのみ)
制限なしの監視 (すべてのイベント)	x	0xffff_ffff_ffff_ffff	x	[1,11] または [0,xx]
命令アドレス範囲チェックのみ	x	0xffff_ffff_ffff_ffe	0	[1,00]
オペコード・マッチのみ	1	希望のオペコード	x	[1,01]
データ・アドレス範囲チェックのみ	x	0xffff_ffff_ffff_ffff	x	[1,10]
命令アドレス範囲チェックとデータオペコード・マッチ	1	希望のオペコード	0	[1,01]
命令アドレス範囲チェックとデータ・アドレス範囲チェック	x	0xffff_ffff_ffff_ffe	0	[1,00]
オペコード・マッチとデータ・アドレス範囲チェック	1	希望のオペコード	x	[1,00]

10.2.3.2 特権レベルによる制限

パフォーマンス監視ソフトウェアは、オペレーティング・システムによるコンテキスト・スイッチのサポートを常に期待することはできない。このため、一般的に、マルチプロセッシング・システムまたはマルチプロセス作業負荷の中の1つのプロセスのパフォーマンス分析は不可能であった。このような分析をハードウェア上でサポートするために、Itanium アーキテクチャは、3つのグローバル・ビット (PSR.up, PSR.pp, DCR.pp) とモニタごとの「特権モニタ」ビット (PMC_i.pm) を定義する。オペレーティング・システムとユーザレベルのアプリケーションがパフォーマンスに与える影響の内訳を調べるために、各モニタは、4ビットの特権レベル・マスク (PMC_i.plm) を指定する。このマスクは、プロセッサ・ステータス・レジスタ内の現行特権レベル (PSR.cpl) と比較され、PMC_i.plm[PSR.cpl] が1の場合は、イベント・カウント機能が有効になる。Itanium 2 プロセッサのパフォーマンス・モニタの制御は、10.3.1 項「パフォーマンス・モニタの制御とアクセス」で説明する。

PMC レジスタは、ユーザレベル・モニタとして設定することも (PMC_i.pm=0)、システムレベル・モニタとして設定することもできる (PMC_i.pm=1)。ユーザレベル・モニタは、PSR.up が1になったときに有効になる。PSR.up は、sum/rum 命令を使用して、アプリケーションによって制御でき

る。この方法で、アプリケーションは、特定のコード・セクションに対するパフォーマンス監視機能を有効 / 無効にできる。システムレベル・モニタは、PSR.pp が 1 になったときに有効になる。PSR.pp は、特権レベル 0 以外では制御できないため、ユーザレベルのプロセスの影響を受けずにモニタの制御が可能である。割り込みが発生するたびに、デフォルト制御レジスタの pp フィールド (DCR.pp) が PSR.pp にコピーされる。これによって、割り込み中に発生したイベントを別に処理ができる。DCR.pp が 0 の場合は、割り込み中のイベントはカウントされない。DCR.pp が 1 の場合は、割り込み中のイベントはカーネル・カウントに追加される。

図 10-6、図 10-7、および図 10-8 に示すように、PSR ビットと DCR ビットを適切に組み合わせれば、1 つのプロセス、複数のプロセス、システムレベルのパフォーマンス監視が可能になる。これらのビットにより、オペレーティング・システムの明示的なサポートがなくても、カーネル・レベルのデバイス・ドライバからパフォーマンス監視機能を制御できる。あるプロセスのプロセッサ・ステータス・レジスタ (PSR) 内で希望の監視設定を一度設定すれば、特に修正していない「通常の」オペレーティング・コンテキスト・スイッチ・コードによって、パフォーマンス監視機能が自動的に有効 / 無効になる。

オペレーティング・システムのサポートがあれば、『インテル® Itanium® 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』のパフォーマンス監視機能の章の説明に従って、プロセスごとのイベント数の内訳を調べられる。

10.2.3.3 命令セットによる制限

Itanium 2 プロセッサ上では、PSR.is によって定義される現在実行中の命令セットに基づいて、監視機能をさらに制限できる。この機能は、4 つの汎用パフォーマンス・カウンタ、オペコード・マッチ・レジスタ、命令 / データ・イベント・アドレス・レジスタによってサポートされる。ただし、分岐トレース・バッファは、Itanium アーキテクチャ・ベースのコードの実行時にのみ有効になる。Itanium アーキテクチャ専用機能を使用するときは、それに対応する PMC レジスタの命令セット・マスク (PMC_i.ism) を Itanium アーキテクチャ専用 (01) に設定して、IA-32 コードによって生成されたイベントが Itanium 2 プロセッサのイベント数に影響を与えないようにする必要がある。

図 10-6. 1 つのプロセスの監視

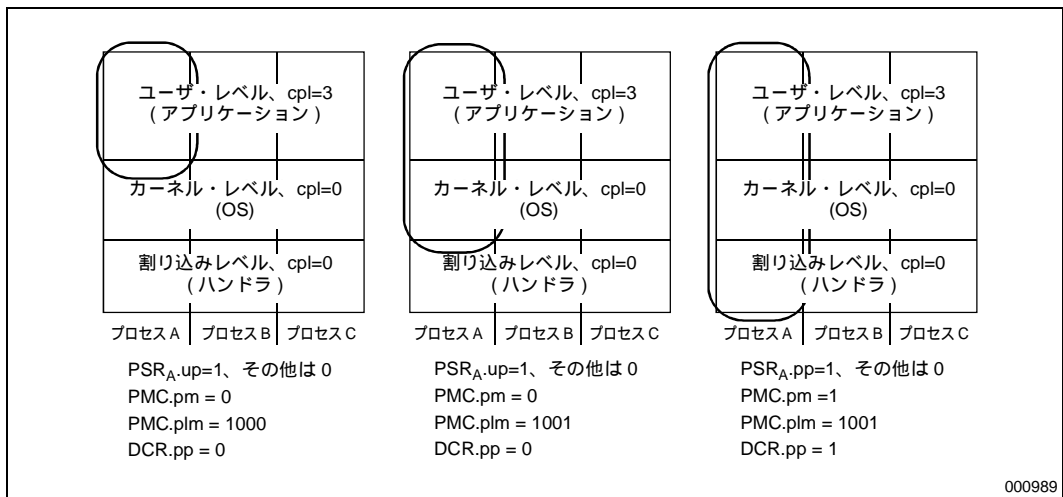


図 10-7. 複数のプロセスの監視

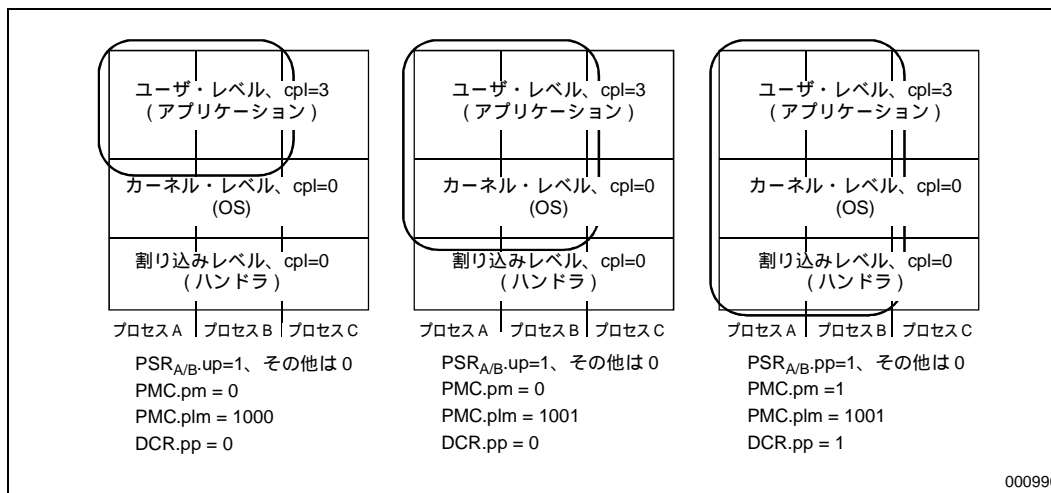
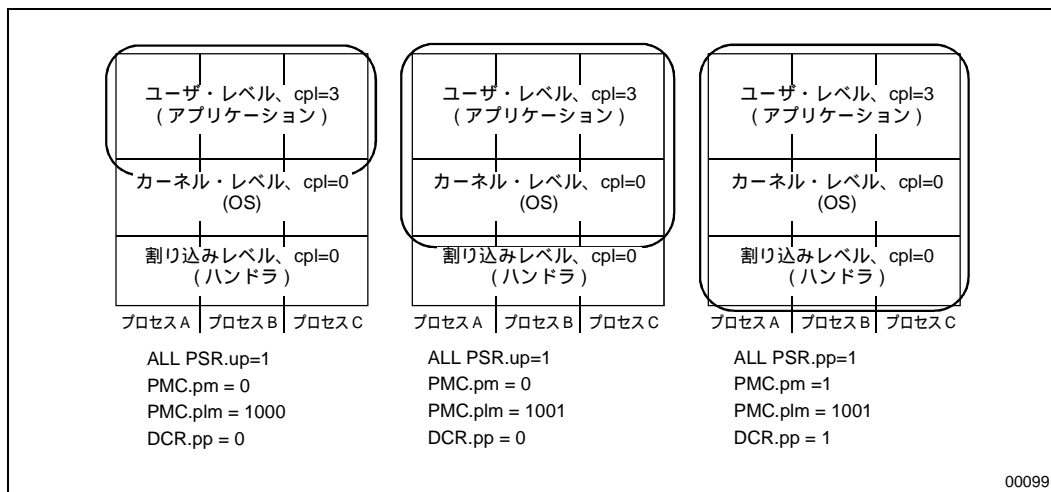


図 10-8. システム全体の監視



10.2.4 参考資料

- [gprof] S.L. Graham, P.B. Kessler, M.K. McKusick 著 『gprof: A Call Graph Execution Profiler』 (コンパイラの構築に関する SIGPLAN'82 シンポジウムの議事録)、SIGPLAN Notices, Vol. 17, No. 6, pp. 120 ~ 126, 1982 年 6 月。
- [Lebeck] Alvin R. Lebeck, David A. Wood 著 『Cache Profiling and the SPEC benchmarks: A Case Study』, Tech Report 1164, Computer Science Dept., University of Wisconsin - Madison, 1993 年 7 月。
- [VTune] Mark Atkins および Ramesh Subramaniam 著 『PC Software Performance Tuning』, IEEE Computer, Vol. 29, No. 8, pp. 47 ~ 54, 1996 年 8 月。
- [WinNT] Russ Blake 著 『Optimizing Windows NT(tm)』 (Microsoft "Windows NT Resource Kit for Windows NT Version 3.51" Volume 4)、Microsoft Press, 1995 年。

10.3 パフォーマンス・モニタのステート

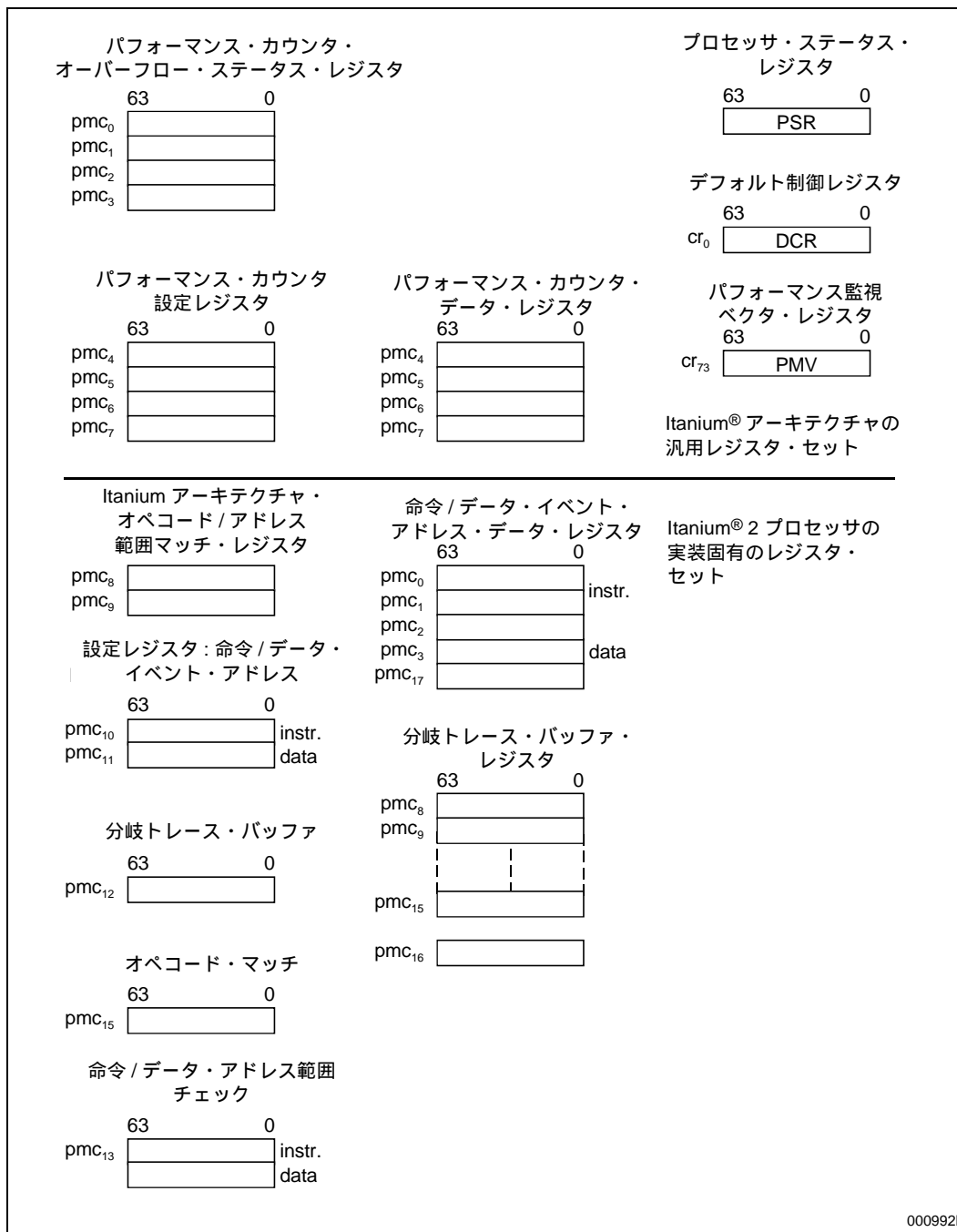
Itanium アーキテクチャは、2 種類のパフォーマンス監視レジスタを定義している。パフォーマンス・モニタ設定 (PMC) レジスタは、監視機能の設定に使用される。パフォーマンス・モニタ・データ (PMD) レジスタは、モニタのデータ値を格納する。本節では、Itanium アーキテクチャの定義によって拡張された、Itanium 2 プロセッサのパフォーマンス監視レジスタを説明する。図 10-9 に示すように、Itanium 2 プロセッサは、4 つの 48 ビット・パフォーマンス・カウンタ (PMC/PMD_{4,5,6,7} のペア) と、モデル固有の監視レジスタを搭載している。モデル固有の監視レジスタには、キャッシュ・ミスおよび TLB ミス監視用の命令 / データ・イベント・アドレス・レジスタ (EAR)、分岐トレース・バッファ、2 つのオペコード・マッチ・レジスタ、命令アドレス範囲チェック・レジスタがある。

表 10-4 に、各監視機能に対する PMC/PMD レジスタの割り当てを示す。割り込みステータス・レジスタは、PMC_{0,1,2,3} に割り当てられる。4 つの汎用パフォーマンス・カウンタのペアは、PMC/PMD_{4,5,6,7} に割り当てられる。イベント・アドレス・レジスタと分岐トレース・バッファは、3 つの設定レジスタ (PMC_{10,11,12}) によって制御される。パフォーマンス監視ソフトウェアは、5 つのイベント・アドレス・データ・レジスタ (PMD_{0,1,2,3,17}) と分岐トレース・バッファ (PMD₈₋₁₆) を介して、収集されたイベント・アドレスとキャッシュ・ミス・レイテンシを参照できる。Itanium 2 プロセッサ上では、命令ブレークポイント・レジスタ (IBR) と命令アドレス範囲チェック・レジスタ (PMC₁₃) を設定し、オペコード・マッチ・レジスタ (PMC_{8,9}) 内のチェック機構を調整すると、設定可能な命令アドレス範囲に基づき、監視対象となるイベントを制限できる。2 つのオペコード・マッチ・レジスタ (PMC_{8,9}) とオペコード・マッチ設定レジスタ (PMC₁₅) は、設定可能なオペコードに基づいて、監視対象となるイベントを制限する。メモリ操作については、データ・ブレークポイント・レジスタ (DBR) とデータ・アドレス範囲設定レジスタ (PMC₁₄) を設定すると、設定可能なデータ・アドレス範囲に基づき、監視対象となるイベントを制限できる。

表 10-4. Itanium® 2 プロセッサのパフォーマンス監視レジスタ・セット

監視機能	設定レジスタ (PMC)	データ・レジスタ (PMD)	説明
割り込みステータス	PMC _{0,1,2,3}	なし	10.3.3 項「パフォーマンス・モニタ・オーバーフロー・ステータス・レジスタ (PMC _{0,1,2,3})」を参照。
イベント・カウンタ	PMC _{4,5,6,7}	PMD _{4,5,6,7}	10.3.2 項「パフォーマンス・カウンタ・レジスタ」を参照。
オペコード・マッチ	PMC _{8,9,15}	なし	10.3.4 項「オペコード・マッチ・チェック (PMC _{8,9,15})」を参照。
命令 EAR	PMC ₁₀	PMD _{0,1}	10.3.7.1 項「命令 EAR (PMC ₁₀ , PMD _{0,1})」を参照。
データ EAR	PMC ₁₁	PMD _{2,3,17}	10.3.8 項「データ EAR (PMC ₁₁ , PMD _{2,3,17})」を参照。
分岐トレース・バッファ	PMC ₁₂	PMD ₈₋₁₆	10.3.9.2 項「分岐トレース・バッファの読み出し」を参照。
命令アドレス範囲チェック	PMC ₁₄	なし	10.3.5 項「命令アドレス範囲マッチング」を参照。
メモリ・パイプライン・イベントの制限	PMC ₁₃	なし	10.3.6 項「データ・アドレス範囲マッチング (PMC ₁₃)」を参照。

図 10-9. Itanium® 2 プロセッサのパフォーマンス監視レジスタのモデル



10.3.1 パフォーマンス・モニタの制御とアクセス

パフォーマンス監視機能を使用するには、 $PMC_4.enable$ に 1 をセットして、PMU への電源をオンにする必要がある。このビットはリセット時にセットされる。このビットをクリアすると、すべての PMD、PMC (PMC_4 を除く)、その他の重要でない回路へのクロックをオフにできるため、節電が可能になる。

電源をオンにすると、イベント収集機能は、パフォーマンス・モニタ設定 (PMC) レジスタとプロセッサ・ステータス・レジスタ (PSR) によって制御される。PSR の 4 つのフィールド (PSR.up、PSR.pp、PSR.cpl、PSR.sp) とパフォーマンス・モニタ・フリーズ・ビット (PMC₀.fr) は、すべてのパフォーマンス監視レジスタの動作に影響を与える。

PMC レジスタの 3 つのフィールド (PMC_i.plm、PMC_i.ism、PMC_i.pm) を使用して、各モニタを制御できる。PMC_i.ism に基づいた命令セット・マスクは、Itanium 2 プロセッサのモデル固有機能である。Itanium 2 プロセッサ上では、モニタのイベント収集機能は、次の条件を満たす場合に有効になる。

Monitor Enable_i = (not PMC₀.fr) and PMC_i.plm[PSR.cpl] and ((not PMC_i.ism[PSR.is]) or (PMC_i=12)) and ((not (PMC_i.pm) and PSR.up) or (PMC_i.pm and PSR.pp))

図 10-10 に、パフォーマンス監視機能に影響を与える、PSR の制御フィールドを示す。PSR ビットによってイベント監視機能と PMD レジスタへのアクセスを制御する方法は、『インテル® Itanium® 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、3.3.2 項および 7.2.1 項を参照のこと。

表 10-5 は、PMC_{4,5,6,7,10,11,12} に適用される各モニタの制御をまとめたものである。表 10-4 「Itanium® 2 プロセッサのパフォーマンス監視レジスタ・セット」に示すように、これらの各 PMC レジスタは、関連するパフォーマンス監視データ・レジスタ (PMD) の動作を制御する。命令 / データ EAR および分岐トレース・バッファ (PMD_{0,1,2,3,8 ~ 17}) に関連付けられている Itanium 2 プロセッサのモデル固有の PMD レジスタは、イベント監視機能がフリーズしている場合 (PMC₀.fr=1) にのみ読み取れる。

図 10-10. プロセッサ・ステータス・レジスタ (PSR) のパフォーマンス監視用フィールド

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
予約済み				その他								pp	sp	その他								予約済み				その他		up	その他	予約済み	
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
予約済み																		その他										is	cpl		

表 10-5. パフォーマンス・モニタ設定 (PMC) レジスタの制御フィールド (PMC_{4,5,6,7,10,11,12})

フィールド	ビット	説明
plm	3:0	特権レベル・マスク - 特定の特権レベルでのパフォーマンス・モニタの動作を制御する。各ビットが 4 つの特権レベルのうち 1 つに対応する。例えば、ビット 0 が特権レベル 0、ビット 1 が特権レベル 1 に対応する。ビット値が 1 の場合は、その特権レベルでモニタが有効になる。すべての plm ビットに 0 を書き込むと、モニタは無効になる。この状態では、Itanium® 2 プロセッサは、この PMC レジスタに対応する PMD レジスタの値を保持しない。
pm	6	特権モニタ - 0 の場合は、パフォーマンス・モニタはユーザ・モニタとして設定され、PSR.up によって有効にされる。PMC.pm が 1 の場合は、パフォーマンス・モニタは特権モニタとして設定され、PSR.pp によって有効にされる。この場合は、特権を持つソフトウェアだけが PMD の値を読み取れる。特権を持たないソフトウェアが PMD を読み取ると、0 が戻される。 注：PMC ₁₀ の場合、このフィールドはビット [4] に実装されている。

表 10-5. パフォーマンス・モニタ設定 (PMC) レジスタの制御フィールド (PMC_{4,5,6,7,10,11,12}) (続き)

フィールド	ビット	説明
ism	25:24	<p>命令セット・マスク - 現在実行中の命令セットに基づき、パフォーマンス・モニタの動作を制御する。命令セット・マスクは、PMC_{4,5,6,7,10,11} に適用され、PMC₁₂ には適用されない。</p> <p>00: Itanium 命令および IA-32 命令の実行中に監視機能が有効になる (PSR.is の値には無関係)。 10: ビット 24 が 0 の場合は、Itanium 命令の実行中に監視機能が有効になる (PSR.is=0 の場合)。 01: ビット 25 が 0 の場合は、IA-32 命令の実行中に監視機能が有効になる (PSR.is=1 の場合)。 11: 監視機能は無効になる。</p> <p>注: PMC₁₀ の場合、これは [15:14] に実装されている。PMC₁₂ はこのフィールドを持たない。</p>

10.3.2 パフォーマンス・カウンタ・レジスタ

Itanium 2 プロセッサは、4 つの汎用パフォーマンス・カウンタ (PMC/PMD_{4,5,6,7} のペア) を搭載している。Itanium 2 プロセッサ上に実装されたカウンタ幅は 48 ビットである ([47] はオーバーフロー条件を示す)。Itanium 2 プロセッサ上の PMC/PMD のペアは、Itanium プロセッサよりも対称的である (つまり、どのカウンタでも、ほぼすべてのイベント・タイプを監視できる)。ただし、キャッシュ・カウンタによっては例外がある。詳細は、11.8.2 項「L1 データ・キャッシュ・イベント」および 11.8.3 項「L2 ユニファイド・キャッシュ・イベント」を参照のこと。これらのカウンタは、1 サイクル当たりのイベントの最大増加数が 7 のイベントを監視できる。

図 10-11 と表 10-6 に、Itanium 2 プロセッサのパフォーマンス・カウンタ設定レジスタ (PMC_{4,5,6,7}) のレイアウトを示す。これらの設定レジスタの主なタスクは、個々のパフォーマンス監視データ・カウンタで監視されるイベントを選択することである。これらのイベントの選択は、PMC レジスタ内のイベント選択 (es) フィールドおよびユニット・マスク (umask) フィールドによって行われる。PMC 内の残りのフィールドは、カウントを実行する条件 (plm、pm、ism)、カウンタのインクリメントの大きさ、(threshold)、カウンタがオーバーフロー (ev、oi) した場合に必要な処置を指定する。

図 10-11. Itanium® 2 プロセッサの汎用 PMC レジスタ (PMC_{4,5,6,7})

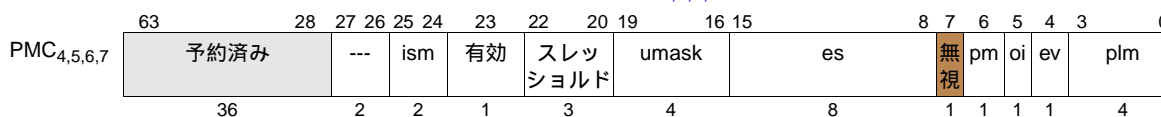


表 10-6. Itanium® 2 プロセッサの汎用 PMC レジスタ・フィールド (PMC_{4,5,6,7})

フィールド	ビット	説明
plm	3:0	特権レベル・マスク。表 10-5「パフォーマンス・モニタ設定 (PMC) レジスタの制御フィールド (PMC _{4,5,6,7,10,11,12})」を参照のこと。
ev	4	外部への通知 - 1 の場合、カウンタがオーバーフローしたとき、外部通知 (BPM ピンのストローブ) が生成される。この外部通知は、oi ビットの設定に関係なく生成される (以下を参照)。Itanium 2 プロセッサ上では、PMC ₄ の外部通知は BPM0 ピン、PMC ₅ は BPM1 ピン、PMC ₆ は BPM2 ピン、PMC ₇ は BPM3 ピンをそれぞれストローブする。
oi	5	オーバーフロー割り込み - 1 の場合、モニタがオーバーフローしたとき、パフォーマンス・モニタ割り込みが発生し、パフォーマンス・モニタ・フリーズ・ビット (PMC ₀ .fr) が設定される。0 の場合、割り込みは発生せず、パフォーマンス・モニタ・フリーズ・ビット (PMC ₀ .fr) は変更されない。カウンタのオーバーフローは、割り込みを 1 回だけ生成する。オーバーフロー時の対応する PMC ₀ ビットの設定は、このビットとは無関係になる。

表 10-6. Itanium® 2 プロセッサの汎用 PMC レジスタ・フィールド (PMC_{4,5,6,7}) (続き)

フィールド	ビット	説明
pm	6	特権モニタ。表 10-5「パフォーマンス・モニタ設定 (PMC) レジスタの制御フィールド (PMC _{4,5,6,7,10,11,12})」を参照のこと。
ig	7	予約済み
es	15:8	イベント選択 - 監視対象となるパフォーマンス・イベントを選択する。Itanium 2 プロセッサのイベントのエンコーディングについては、第 11 章「パフォーマンス監視イベント」で説明している。
umask	19:16	ユニット・マスク - イベント固有のマスク・ビット (詳細は、イベントの定義を参照のこと)
スレッシュ ホールド	22:20	スレッシュホールド - 「複数発生」イベントのスレッシュホールドの設定を可能にする。スレッシュホールドが 0 の場合、カウンタはすべての検出されたイベント値を合計する。スレッシュホールドが 0 でない場合、検出されたイベント値がそのスレッシュホールドを超えたサイクルごとに、カウンタは 1 ずつインクリメントされる。
有効	23	PMC ₄ のみ。PMU を使用可能にする。PMU を機能させるには、1 を書き込む必要がある。1 をセットすると、電源がオンになる。
ism	25:24	命令セット・マスク。表 10-5「パフォーマンス・モニタ設定 (PMC) レジスタの制御フィールド (PMC _{4,5,6,7,10,11,12})」を参照のこと。
---	27:26	適切な PMU 操作に対して 0 を書き込む必要がある。
無視	63:28	0 として読み出され、書き込みは無視される。

図 10-12 と表 10-7 に、Itanium 2 プロセッサのパフォーマンス・カウンタ・データ・レジスタ (PMD_{4,5,6,7}) のレイアウトを示す。カウンタがラップしたとき (つまり、ビット 46 からの桁の繰り上げが検出されたとき)、カウンタ・オーバーフローが発生する。パフォーマンス監視ソフトウェアは、カウンタ値 $2^{47} - N$ をモニタにあらかじめロードしておく、N 個のイベントが発生したときに外部割り込みまたは外部通知を発生させられる。ビット 47 はオーバーフロー・ビットであり、レジスタを初期化する際は必ず 0 に初期設定する必要がある。

ソフトウェアは、(パフォーマンス監視レジスタにアクセスできる場合は) イベントの収集を中止せずに、パフォーマンス・カウンタ・レジスタ PMD_{4,5,6,7} の読み出しを続行できる。適切な特権レベル以外でソフトウェアから PMD を読み出すと、0 が戻される (表 10-6 の「plm」を参照)。プロセッサは、ソフトウェアが単調に増加するカウンタ値を読み出すことを保証する。

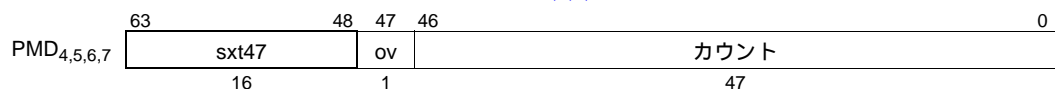
図 10-12. Itanium® 2 プロセッサの汎用 PMD レジスタ (PMD_{4,5,6,7})

表 10-7. Itanium® 2 プロセッサの汎用 PMD レジスタのフィールド

フィールド	ビット	説明
sxt47	63:48	書き込みは無視される。読み出しはビット 47 の値を返すため、カウンタ値は符号拡張された値として読み取られる。
ov	47	オーバーフロー・ビット (ビット 46 から桁が繰り上げられた)。 注: PMD を初期化するには、このビットに 0 を書き込む必要がある。
カウンタ	46:0	イベント数。カウンタは、カウンタ・フィールドがラップした (ビット 46 から桁が繰り上げられた) ときにオーバーフローするように定義されている。

10.3.3 パフォーマンス・モニタ・オーバーフロー・ステータス・レジスタ (PMC_{0,1,2,3})

前述したように、Itanium 2 プロセッサは、4つのパフォーマンス監視カウンタを搭載している。これらの4つのカウンタのオーバーフロー・ステータスは、レジスタ PMC₀ に示される。図 10-13 と表 10-8 に示すように、PMC₀ の [7:4,0] ビットだけが値を与えられる。その他のオーバーフロー・ビットは、すべて無視される（つまり、0として読み出され、書き込みは無視される）。

図 10-13. Itanium® 2 プロセッサのパフォーマンス・モニタ・オーバーフロー・ステータス・レジスタ (PMC_{0,1,2,3})

63	8 7 6 5 4 3 2 1 0
予約済み (PMC ₀)	オーバー フロー 予約済み fr
	4 3 1
予約済み (PMC ₁)	
予約済み (PMC ₂)	
予約済み (PMC ₃)	

表 10-8. Itanium® 2 プロセッサのパフォーマンス・モニタ・オーバーフロー・レジスタのフィールド (PMC_{0,1,2,3})

レジスタ	フィールド	ビット	説明
PMC ₀	fr	0	パフォーマンス・モニタ「フリーズ」ビット - 1 の場合、イベント監視機能は無効になる。0 の場合、イベント監視機能は有効になる。パフォーマンス・モニタに対応するオーバーフロー割り込みビット (PMC _{0i}) が 1 にセットされている場合、そのモニタのオーバーフローが発生すると、このビットがハードウェアによってセットされる。このビットをクリアする役割は、ソフトウェアが受け持つ。PMC _{0i} ビットがセットされていない場合は、カウンタ・オーバーフローが発生してもこのビットはセットされない。
PMC ₀	無視	3:1	0 として読み出され、書き込みは無視される。
PMC ₀	オーバー フロー	7:4	イベント・カウンタ・オーバーフロー - ビット n が 1 の場合は、PMDn がオーバーフローしたことを示す。これは、どのパフォーマンス・モニタがオーバーフローしたかを示すビット・ベクトルである。これらのオーバーフロー・ビットは、PMC _{0i} ビットの状態に関係なく、そのビットに対応するカウンタがオーバーフローしたときにセットされる。これらのビットは、ソフトウェアによってもセットできる。これらのビットはスティッキー・ビットであり、複数のビットをセットできる。
PMC ₀	無視	63:8	0 として読み出され、書き込みは無視される。
PMC _{1,2,3}	無視	63:0	0 として読み出され、書き込みは無視される。

10.3.4 オペコード・マッチ・チェック (PMC_{8,9,15})

Itanium 2 プロセッサは、命令アドレスや命令の Itanium アーキテクチャ・エンコーディング（オペコード）に基づいて、監視対象となるイベントを制限できる。これらの機能を有効にするには、レジスタ PMC₁₅ および PMC₁₄ (10.3.5 項「命令アドレス範囲マッチング」を参照) を使用する。レジスタ PMC_{8,9} を使用すると、これらの機能を設定できる。メモリ関連のイベントの場合、この機能を有効にするには、該当するビットを PMC₁₃ に設定する必要がある。詳細は、10.3.6 項「データ・アドレス範囲マッチング (PMC₁₃)」を参照のこと。Itanium プロセッサの場合と異なり、Itanium 2 プロセッサのオペコード・マッチャーは、Itanium アーキテクチャ・ベースのコード実行中および IA32 のコード実行中の両方で動作する。IA32 モードで動作中、このオペコード・マッチャーは、Itanium アーキテクチャ・オペコードがないかどうかをチェックする。

図 10-14 および表 10-9 に、PMC_{8,9} レジスタのフィールドを示す。図 10-15 および表 10-10 に、レジスタ PMC₁₅ を示す。ビット [63:60] のすべての組み合わせがサポートされている。A スロット命令に一致するためには、ビット [63:62] に 11 を設定する。すべての命令タイプに一致するためには、ビット [63:60] に 1111 を設定する必要がある。オペコード・マッチャーに関係なく、すべてのイベントがカウントされるようにするには、PMC_{8,9} のすべての mifb ビットとすべてのマスク・ビットに 1 をセットする (すべてのオペコードが一致する) 必要がある。

PMC₉ は、イベント IA64_TAGGED_INST_RETIRED のみを制限する。IA64_TAGGED_INST_RETIRED イベントに対する Itanium 2 プロセッサのオペコード制限により、PMC₉ の結果と IBRP₁ マッチおよび IBRP₃ マッチの AND 演算、PMC₈ の結果と IBR₀ マッチおよび IBRP₂ マッチの AND 演算が行われる。ただし、PMC₈ は、その他のダウストリーム・イベントも制限する。オペコード・マッチャーに関係なく、すべてのイベントがカウントされるようにするには、ビット [63:60] とビット [29:3] にすべて 1 をセットする必要がある。

図 10-14. オペコード・マッチ・レジスタ (PMC_{8,9})

63	62	61	60	59					32	31	30	29					3	2	1	0
m	i	f	b		マッチ				予約				マスク				inv			ig_ad
1	1	1	1		28				2				28				1			1

表 10-9. オペコード・マッチ・レジスタのフィールド (PMC_{8,9})

フィールド	ビット	幅	説明
ig_ad	0	1	命令アドレス範囲チェックを無視する。1 をセットすると、命令アドレスのすべてがイベント用に考慮される。0 にすると、IBR 0-1 がアドレス制限用に使用される。 注: PMC ₉ では、このビットは無視される。
inv	1	1	範囲チェックを反転する。1 をセットすると、IBR0-1 によって指定されたアドレス範囲が反転する。これは、ig_ad ビットを 0 にした場合にのみ有効である。 注: PMC ₉ では、このビットは無視される。
マスク	29:2	28	Itanium 命令エンコーディング・ビットをマスクするビット [2] は、1 にセットする必要がある。 [15:3] は、オペコード・ビット [12:0] のビットをマスクする。 [29:16] は、オペコード・ビット [40:27] のビットをマスクする。 マスク・ビットに 1 をセットした場合、オペコード・マッチでは、対応するオペコード・ビットは使用されない。
マッチ	59:33	27	Itanium 命令エンコーディングが一致されるオペコード・ビット [45:33] は、オペコード・ビット [12:0] のビットに一致する。 [59:46] は、オペコード・ビット [40:27] のビットに一致する。
b	60	1	1 の場合は、オペコードが B スロットの場合に一致する。
f	61	1	1 の場合は、オペコードが F スロットの場合に一致する。
i	62	1	1 の場合は、オペコードが I スロットの場合に一致する。
m	63	1	1 の場合は、オペコードが M スロットの場合に一致する。

図 10-15. オペコード・マッチ設定レジスタ (PMC₁₅)

63					4	3	2	1	0
予約済み					ibrp3	ibrp2	ibrp1	ibrp0	
					pmc9	pmc8	pmc9	pmc8	
					1	1	1	1	

表 10-10. オペコード・マッチ設定レジスタのフィールド (PMC₁₅)

フィールド	ビット	説明
ibrp0-pmc8	0	1: PMU イベントは、オペコードによって制限されない。 0: PMU イベント (IA64_TAGGED_INST_RETIRED.00 を含む) は、PMC ₈ によって制限されるオペコードになる。
ibrp1-pmc9	1	1: IA64_TAGGED_INST_RETIRED.01 は、オペコードによって制限されない。 0: IA64_TAGGED_INST_RETIRED.01 は、PMC ₉ によって制限されるオペコードになる。
ibrp2-pmc8	2	1: IA64_TAGGED_INST_RETIRED.10 は、オペコードによって制限されない。 0: IA64_TAGGED_INST_RETIRED.10 は、PMC ₈ によって制限されるオペコードになる。
ibrp3-pmc9	3	1: IA64_TAGGED_INST_RETIRED.11 は、オペコードによって制限されない。 0: IA64_TAGGED_INST_RETIRED.11 は、PMC ₉ によって制限されるオペコードになる。

オペコード・マッチの目的で、Itanium 命令は、命令タイプ "itype" (M、I、F、または B のうちのひとつ) と 41 ビットエンコーディングの 2 つの項目によって定義される (『インテル® Itanium® 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』を参照)。各命令が、各オペコード・マッチ・レジスタ (PMC_{8,9}) に対して、次のように評価される。

$$\text{Match}(\text{PMC}_i) = (\text{imatch}(\text{itype}, \text{PMC}_i.\text{mifb}) \text{ AND } \text{ematch}(\text{enco}, \text{PMC}_i.\text{match}, \text{PMC}_i.\text{mask}))$$

各項目は次のとおりである。

$$\text{imatch}(\text{itype}, \text{PMC}[i].\text{mifb}) = (\text{itype}=\text{M} \text{ AND } \text{PMC}[i].\text{m}) \text{ OR } (\text{itype}=\text{I} \text{ AND } \text{PMC}[i].\text{i}) \text{ OR } (\text{itype}=\text{F} \text{ AND } \text{PMC}[i].\text{f}) \text{ OR } (\text{itype}=\text{B} \text{ AND } \text{PMC}[i].\text{b})$$

$$\text{ematch}(\text{enco}, \text{match}, \text{mask}) = \text{AND}_{b=40..27} ((\text{enco}\{b\}=\text{match}\{b-14\}) \text{ OR } \text{mask}\{b-14\}) \text{ AND } \text{AND}_{b=12..0} ((\text{enco}\{b\}=\text{match}\{b\}) \text{ OR } \text{mask}\{b\})$$

この関数は、エンコーディング・ビット {40:27} (メジャー・オペコード) とエンコーディング・ビット {12:0} (デスティネーション・プレディケートおよび修飾プレディケート) のみ一致する。オペコード・マッチャーは、命令エンコーディングのビット {26:13} を無視する。

IBRP マッチは、オペコードが配布される点を指す命令ポインタを使用して進められる。オペコード・マッチャーからのマッチは、この時点での IBRP マッチを使用して、AND 演算される。

この処理で、2 つのオペコード・マッチ・イベントが得られる。これらの結果は、次のように命令範囲チェック・タグ (IBRRangeTag) と組み合わせられる (IBRRangeTag については、10.3.5 項「命令アドレス範囲マッチング」を参照)。

$$\text{Tag}(\text{PMC}_8) = \text{Match}(\text{PMC}_8) \text{ and } \text{IBRRangeTag}$$

$$\text{Tag}(\text{PMC}_9) = \text{Match}(\text{PMC}_9) \text{ and } \text{IBRRangeTag}$$

図 10-5 に示したように、Tag(PMC₈) と Tag(PMC₉) の 2 つのタグは、命令がリタイアされるまで、プロセッサのパイプライン内の下位の段に渡されていく。これらのタグは、リタイア命令カウント・イベントとして選択できる (リタイア命令カウント・イベントについては、11-147 ページの「IA64_TAGGED_INST_RETIRED」のイベントの説明を参照のこと)。この方法で、パフォーマンス・カウンタ (PMC/PMD_{4,5,6,7}) を使用して、指定したオペコードに一致する、設定されたアドレス範囲内のリタイアした命令数をカウントできる。

別のパイプラインに配布されるオペコードは、PMC₈ と比較される。オペコード・マッチは、多数のユーザ設定可能ビット (本書の PMC₁₅ の定義を参照) でさらに制限され、IBRP0 マッチと AND 演算された後、別の場所に配布される。

注: レジスタ PMC15 には、事前設定値である 0xfffff0 が含まれていなければならない。表 10-10 に示されていないビットをソフトウェアで変更した場合、プロセッサの動作は未定義である。

10.3.5 命令アドレス範囲マッチング

Itanium 2 プロセッサは、命令アドレスの範囲に基づいて、監視対象となるイベントを制限できる。4つのアーキテクチャ上の命令ブレークポイント・レジスタ・ペア IBRP₀₋₃ (IBR₀₋₁₇) を使用すると、希望のアドレス範囲を指定できる。この制限は、一度設定すると、すべてのイベントに適用される。Itanium 2 プロセッサでは、レジスタ PMC_{8,14} を使用して、一致したアドレスをどのようにパフォーマンス・モニタに適用するかを指定する。IA64_INST_RETIRED およびプリフェッチ・イベントを除き、IBRP₀ は、使用される唯一の IBR ペアであり、このセクションのデフォルトであると見なされる。メモリ関連のイベントの場合、この機能を有効にするには、該当するビットを PMC₁₃ に設定する必要がある。詳細は、10.3.6 項「データ・アドレス範囲マッチング (PMC13)」を参照のこと。

図 10-16 および表 10-12 に、レジスタ PMC₁₄ のフィールドを示す。命令アドレス範囲チェックは、「アドレス範囲チェックの無視」ビット (PMC₈.ig_ad および PMC₁₄.ibrp0) によって制御される。PMC₈.ig_ad が 1 (または PMC₁₄.ibrp0 が 1) の場合は、IBR の設定に関係なく、すべての命令にタグが付けられる。このモードでは、IA-32 コードのイベントと Itanium アーキテクチャ・ベース・コードのイベントがいずれもイベント数に含まれる。PMC₈.ig_ad および PMC₁₄.ibrp0 がともに 0 の場合は、IBR の設定に基づいて、すべての Itanium アーキテクチャ・コード・フェッチに対して命令アドレス範囲チェックが適用される。このモードでは、IA-32 命令にはタグが付けられないため、IA-32 コードによって発生したイベントは無視される。表 10-11 に、PSR.is と、PMC₈.ig_ad または PMC₁₄.ibrp0 のさまざまな組み合わせについて、命令アドレス範囲チェックの動作を示す。

表 10-11. 命令セットによる Itanium® 2 プロセッサの命令アドレス範囲チェック

PMC ₈ .ig_ad OR PMC ₁₄ .ibrp0	PSR.is	
	0 (Itanium)	1 (IA-32)
0	IBR 範囲に一致する Itanium 命令にのみタグを付ける。	IA-32 命令にはタグを付けない。
1	すべての Itanium 命令および IA-32 命令にタグを付ける。IBR の範囲は無視する。	

プロセッサは、各 Itanium 命令のフェッチ・アドレス IP{63:0} と、アーキテクチャ上の命令ブレークポイント・レジスタ・ペア IBRP₀ に設定されたアドレス範囲とを比較する。命令ブレークポイント・フォルト有効ビット (IBR の x ビット) の値に関係なく、Itanium 2 プロセッサの IBRP₀ について、次の式が評価される。

$$IBRmatch = match(IP, IBR_0.addr, IBR_1.mask, IBR_1.plm)$$

この制限されたマッチ (IBRmatch) が真である場合は、命令配布ステージ以前に発生するイベントが実行される。この制限されたマッチは、オペコード・マッチャー PMC₈ の結果と AND 演算され、より多くのユーザ定義可能ビット (表 10-12 を参照) でさらに制限された後、別の場所に配布される。命令配布ステージ以後に発生するイベントでは、この新しい制限されたマッチ (ibrp0-pmc8 マッチ) が使用される。

図 10-16. 命令アドレス範囲設定レジスタ (PMC₁₄)

63	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
予約済み			fine	予約済み	ibrp3	予約済み	ibrp2	予約済み	ibrp1	予約済み	ibrp0	予約済み			
50			1	2	1	2	1	2	1	2	1	2	1	1	

表 10-12. 命令アドレス範囲設定レジスタのフィールド (PMC₁₄)

フィールド	ビット	説明
ibrp0	1	1: 制限なし。 0: 非プリフェッチ PMU イベント (IA64_TAGGED_INST_RETIRED.00 を含む) が IBRP ₀ で制限される。
ibrp1	4	1: 制限なし。 0: プリフェッチ PMU イベント (IA64_TAGGED_INST_RETIRED.01 を含む) が IBRP ₁ で制限される。
ibrp2	7	1: 制限なし。 0: 非プリフェッチ PMU イベント (IA64_TAGGED_INST_RETIRED.10 を含む) が IBRP ₂ で制限される。
ibrp3	10	1: 制限なし。 0: 非プリフェッチ PMU イベント (IA64_TAGGED_INST_RETIRED.11 を含む) が IBRP ₃ で制限される。
fine	13	任意の範囲チェックを有効にする (2 の累乗ではない)。 1: IBRP _{0,2} と IBRP _{1,3} は、下限ビット / 上限ビットとしてペアにされる。 0: 標準モード このビットの機能は、次のとおりである。1 をセットすると、ibrp0 (下限) と ibrp2 (上限) がペアにされ、ibrp1 (下限) と ibrp3 (上限) がペアにされる。上限および下限のビット [63:12] は、全く同じにしなければならないが、任意の値を持つことができる。上限のビット [11:0] は、下限のビット [11:0] よりも大きくなければならない。アドレスが上限と下限の間に入っている場合は、使用される ibr ペアの両方について、マッチが示される (ibrp0 と ibrp2 は、同時にマッチを示す)。 注: このモードでは、ビット [11:0] の IBR 1、3、5、7 にマスク・ビットをセットしても効果がない。

IBRP₀ マッチは、次の方法で生成される。fine モードを使用していない場合は、任意の範囲チェックを実行できないことに注意する必要がある。なぜなら、マスク・ビットが 2 の累乗であるからである。fine モードでは、2 つの IBR ペアを使用して、ページ内の範囲の上限と下限を指定する (下限と上限の上位ビットは、全く同じでなければならない)。

```
If PMC14.Fine=0, IBRmatch0 = match[IP(63:0), IBR0(63:0), IBR1(55:0)]
Else, IBRmatch0 = match[IP(63:12), IBR0(63:12), IBR1(55:12)] and [IP(11:0) >
IBR0(11:0)] and [IP(11:0) < IBR4(11:0)]
IBRadrmatch0 = IBRmatch0
ibrp0 match = (PMC8.ign or PMC14.ibrp0) or (IBRadmatch0 and match[PSR.cpl,
IBR1(59:56)])
```

命令範囲チェック・タグ (IBRRangeTag) が IBR アドレス範囲を考慮するのは、PMC₈.ig_ad、PMC₁₄.ibrp0、PSR.is がすべて 0 であり、IBR の x ビットまたは PSR.db がどれもセットされていない場合に限られる。

IBR の一部をパフォーマンス監視用に使用し、それと同時に、残りの IBR をデバッグ (これらのレジスタの構成された目的) 用に使用できるようにするために、IBR を有効にすることを目的とした個別の機構が提供されている。PMU で使用される IBRP では、x ビットを 0 にクリアする必要がある。

10.3.5.1 命令アドレス範囲チェック用の IBRP0 の使用 - 例外 1

プリフェッチ・イベントのアドレス範囲制限は、プリフェッチ命令のアドレスではなく、プリフェッチ・イベントのターゲット・アドレスに適用される。したがって、それらのイベントを制限するには、IBRP₁ を使用する必要がある。IBR_{0,1,4} の代わりに IBR_{2,3,6} を使用する点を除けば、IBRP₁ マッチの計算は、IBRP₀ マッチの計算と同じである。

注: レジスタ PMC₁₄ には、事前設定値である 0xdb6 が含まれていなければならない。表 10-12 に示されていないビットをソフトウェアで変更した場合、プロセッサの動作は未定義である。PMC₁₃[48:45]=0000 と PMC₈[0]=0、および ((PMC₁₄[2:1]=10 または 00) あるいは (PMC₁₄[5:4]=10 または 00)) を設定するのは禁止されている。このように設定すると、L1D および L2 内の I サイド・イベントにタグを付ける際、矛盾が発生する。

10.3.5.2 命令アドレス範囲チェック用の IBRP0 の使用 - 例外 2

IA64_TAGGED_INST_RETIRED イベントに対するアドレス範囲制限では、4 つの IBR ペアがすべて使用される。IBR₀ の代わりに IBR_{4,5} (非 fine モード) を使用する点を除けば、IBRP₂ マッチの計算は、IBRP₀ マッチの計算と同じである。IBR_{2,3} の代わりに IBR_{6,7} (非 fine モード) を使用する点を除けば、IBRP₃ マッチの計算は、IBRP₁ マッチの計算と同じである。

10.3.6 データ・アドレス範囲マッチング (PMC₁₃)

メモリを参照する命令については、Itanium 2 プロセッサは、データ・アドレス範囲に基づき、カウントするイベントを制限できる。4 つのアーキテクチャ上のデータ・ブレイクポイント・レジスタ (DBR) を使用すると、希望するアドレス範囲を指定できる。データ・アドレス範囲チェック機能は、メモリ・パイプライン・イベント制限レジスタ (PMC₁₃) によって制御される。

図 10-17 および表 10-11 に、レジスタ PMC₁₃ のフィールドを示す。データ・アドレス範囲チェックは、有効にされている場合 (使用する 4 つの DBR のいずれかに対応するビット内の [1,x0])、ロード命令、ストア命令、セマフォ操作命令、lfetch 命令に適用される。

表 10-13. メモリ・パイプライン・イベント制限のフィールド (PMC₁₃)

フィールド	ビット	説明
cfg dbrp0	4:3	これらのビットは、メモリ・パイプライン・イベントを制限するために DBRP ₀ を使用するかどうか、またどのように使用するかを決定する (該当する場合)。 00: IBR/Opc/DBR - 制限のために IBRP ₀ /PMC ₈ と DBRP ₀ を使用する (つまり、これらがカウントされるのは、命令アドレス、オペコード、データ・アドレスが、これらのレジスタ内に設定されている IBRP ₀ と一致する場合のみである)。 01: IBR/Opc - 制限のために IBRP ₀ /PMC ₈ を使用する。 10: DBR - 制限のために DBRP ₀ のみを使用する。 11: 制限なし。 注: "fine" モードと一緒に使用する場合は、下限 DBR ペア (DBRP0 または DBRP1) の設定のみを行う必要がある (fine モードは、PMC14 の説明を参照)。上限 DBR ペアの設定は、制限なしのままにしておく必要がある。したがって、IBRP _{0,2} が "fine" モードで選択された場合は、希望する制限に応じて cfg_dbrp0 を設定し、cfg_dbrp2 は 11 のまま (制限なし) にする。
cfg dbrp1	12:11	これらのビットは、メモリ・パイプライン・イベントを制限するために DBRP ₁ を使用するかどうか、またどのように使用するかを決定する (該当する場合)。これらは、DBRP ₀ に定義されたものと bit for bit で一致する。
cfg dbrp2	20:19	これらのビットは、メモリ・パイプライン・イベントを制限するために DBRP ₂ を使用するかどうか、またどのように使用するかを決定する (該当する場合)。これらは、DBRP ₀ に定義されたものと bit for bit で一致する。
cfg dbrp3	48,28:27	これらのビットは、メモリ・パイプライン・イベントを制限するために DBRP ₃ を使用するかどうか、またどのように使用するかを決定する (該当する場合)。これらは、DBRP ₀ に定義されたものと bit for bit で一致する。
イネーブル dbrp0	45	0 - 制限なし。 1 - cfg dbrp0 で設定された制限

表 10-13. メモリ・パイプライン・イベント制限のフィールド (PMC₁₃) (続き)

フィールド	ビット	説明
イネーブル dbrp1	46	0 - 制限なし。 1 - cfg dbrp1 で設定された制限
イネーブル dbrp2	47	0 - 制限なし。 1 - cfg dbrp2 で設定された制限
イネーブル dbrp3	48	0 - 制限なし。 1 - cfg dbrp3 で設定された制限

図 10-17. メモリ・パイプライン・イベント制限設定レジスタ (PMC₁₃)

63	49	48	47	46	45	44	29	28	27	26	21	20	19	18	13	12	11	10	5	4	3	2	0
予約 済み	イネーブル dbrp 3 2 1 0				予約済み				cfg dbrp 3	予約 済み	cfg dbrp 2	予約 済み	cfg dbrp 1	予約 済み	cfg dbrp 0	予約 済み							
15	1	1	1	1	1	16	2	6	2	6	2	6	2	6	2	3							

DBRP_x マッチは、次の方法で生成される。マスク・ビットが 2 の累乗であるため、任意の範囲チェックは実行できない。一度に複数の DBRP を有効にしてチェックを行うのは可能であるが、推奨できない。結果である 4 つのマッチは PSR.db と組み合わせられ、単一の DBR マッチが生成される。

```
DBRRangeMatch = ((DBRRangeMatch0 or DBRRangeMatch1 or DBRRangeMatch2 or
DBRRangeMatch3) and (not PSR.db))
```

メモリ命令が EXE ステージに到達した後に発生するイベントは、この制限されたマッチ (DBRP_x マッチ) が真である場合にのみ実行される。データ・アドレスは DBRP_x と比較される。アドレス・マッチは、PMC₁₃ 内の多数のユーザ設定可能ビットでさらに制限された後、別の場所に配布される。パフォーマンス監視用の DBR マッチは、DBR の読み出し、書き込み、plm の各フィールドの設定を無視する。

DBR の一部をパフォーマンス監視用に使用し、それと同時に、残りの DBR をデバッグ (これらのレジスタの構成された目的) 用に同時に使用できるようにするために、DBR を有効にするのを目的とした個別の機構が提供されている。PMU で使用される DBRP では、読み出し / 書き込みビットを 0 にクリアする必要がある。

注: レジスタ PMC13 には、事前設定値である 0x2078fefefefe が含まれていなければならない。表 10-11 に示されていないビットをソフトウェアで変更した場合、プロセッサの動作は未定義である。PMC13[48:45]=0000 と PMC8[0]=0、((PMC14[2:1]=10 または 00) あるいは (PMC14[5:4]=10 または 00)) を設定することは禁止されている。このように設定すると、L1D および L3 内の I サイド・イベントにタグを付ける際、矛盾が発生する。

10.3.7 イベント・アドレス・レジスタ (PMC_{10,11}/PMD_{0,1,2,3,17})

本項では、Itanium 2 プロセッサの命令 / データ・イベント・アドレス・レジスタ (EAR) のレジスタ・レイアウトを説明する。Itanium 2 プロセッサは、命令キャッシュ・ミス、命令 TLB ミス、データ・キャッシュ・ロード・ミス、データ TLB ミス、ALAT ミス、フロントエンド・ストロールの 6 つのイベントの検出をサポートしている。EAR は、2 つの PMC レジスタ (PMC_{10,11}) によって設定される。パフォーマンス監視ソフトウェアは、EAR 固有の単位マスクによって、イベント収集パラメータをハードウェアに指示できる。命令アドレス、データ・アドレス、操作のレイテンシなど、収集されたイベントのパラメータは、5 つの PMD レジスタ (PMD_{0,1,2,3,17}) に格納される。命令 / データ・キャッシュ EAR は、収集されたキャッシュ・イベントのレイテンシを報告する。また、レイテンシのスレッシュホルドに基づいて、収集されるイベントを制限できる。イベント・アドレス・データ・レジスタ (PMD_{0,1,2,3,17}) は、イベント収集機能がフリーズされているときのみ (PMC_{0,fr}=1)、有効なデータを読み出せる。イベント収集機能が有効になっている間に PMD_{0,1,2,3,17} を読み取った場合は、戻り値は未定義である。

10.3.7.1 命令 EAR (PMC₁₀、PMD_{0,1})

命令イベント・アドレス設定レジスタ (PMC₁₀) の設定によって、L1 命令キャッシュ・ミス・イベントまたは命令 TLB ミス・イベントを監視できる。図 10-18 と表 10-14 に、PMC₁₀ レジスタのレイアウトを詳しく示す。表 10-19 に、PMC₁₀ に対応するイベント・アドレス・データ・レジスタ PMD_{0,1} を示す。

図 10-18. 命令イベント・アドレス設定レジスタ (PMC₁₀)

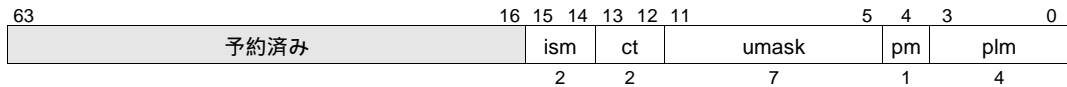
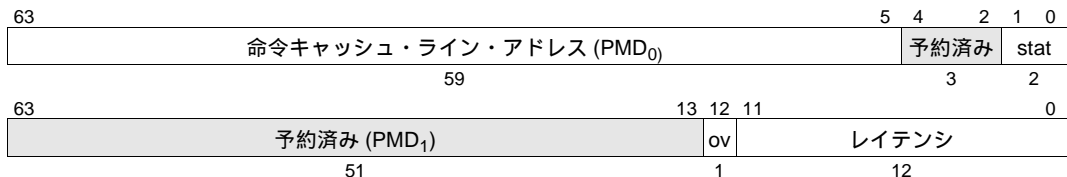


表 10-14. 命令イベント・アドレス設定レジスタのフィールド (PMC₁₀)

フィールド	ビット	説明
plm	3:0	表 10-5 「パフォーマンス・モニタ設定 (PMC) レジスタの制御フィールド (PMC4,5,6,7,10,11,12)」を参照。
pm	4	表 10-5 「パフォーマンス・モニタ設定 (PMC) レジスタの制御フィールド (PMC4,5,6,7,10,11,12)」を参照。
umask	11:5 12:5	監視対象のイベントを選択する。 [13]=1 の場合は、[12:5] が umask に使用される。
ct	13:12	cache_tlb ビット。命令 EAR セレクタ。命令キャッシュまたは TLB ストールを選択する。
	if =1x:	デマンド命令キャッシュ・ミスを監視する。 注: ISB ヒットは、考慮されるミスではない。 PMD _{0,1} レジスタの解釈 (表 10-16 「キャッシュ・モード (PMC10.ct='1x) の命令 EAR (PMD0,1)」を参照)
	if =01:	何も監視されない。
	if =00:	L1 命令 TLB ミスを監視する。 PMD _{0,1} レジスタの解釈 (表 10-16 「キャッシュ・モード (PMC10.ct='1x) の命令 EAR (PMD0,1)」を参照)
ism	15:14	表 10-5 「パフォーマンス・モニタ設定 (PMC) レジスタの制御フィールド (PMC4,5,6,7,10,11,12)」を参照。
無視	31:16 47:32	読み出し時、ビット [15:0] の値がそれぞれ返される。

図 10-19. 命令イベント・アドレス・レジスタのフォーマット (PMD_{0,1})



cache_tlb ビット (PMC₁₀.ct) が 0 にされている場合は、命令キャッシュ・ミスが監視される。cache_tlb ビットが 1 にセットされている場合は、命令 TLB ミスが監視される。umask フィールドとパフォーマンス・モニタ・データ・レジスタ PMD_{0,1} の意味は、cache_tlb ビットの設定によって異なる。命令キャッシュの監視は、10.3.7.2 項「キャッシュ・モードの命令 EAR (PMC10.ct='1x)」を参照のこと。命令 TLB の監視は、10.3.7.3 項「TLB モードの命令 EAR (PMC10.ct=00)」を参照のこと。

10.3.7.2 キャッシュ・モードの命令 EAR (PMC₁₀.ct='1x)

PMC₁₀.ct が 1x の場合、命令イベント・アドレス・レジスタは、L1 命令キャッシュ・ミスの命令アドレスとアクセス・レイテンシを収集する。レイテンシがスレッシュホールドを超えたミスだけが収集される。スレッシュホールドは、設定レジスタ PMC₁₀ 内の 4 ビットの umask フィールドで指定される。表 10-15 は、設定可能なスレッシュホールドの値を示している。

表 10-15. キャッシュ・モード (PMC10.ct='1x) の命令 EAR (PMC10) の umask フィールド

umask ビット 12:5	レイテンシのスレッシュホールド [CPU サイクル]	umask ビット 12:5	レイテンシのスレッシュホールド [CPU サイクル]
01xxxxxx	>0 (すべて L1 ミス)	11100000	>=256
11111111	>=4	-----	>=512
11111110	>=8	11000000	>=1024
11111100	>=16	-----	>=2048
11111000	>=32	10000000	>=4096
-----	>=64	その他	未定義
11110000	>=128	00000000	RAB ヒット (RAB 内でヒットしたすべての L1 ミス)

表 10-16 に示すように、L1 命令キャッシュ・ミスになった命令キャッシュ・ラインのアドレスは、PMD₀ に格納される。監視対象となるイベントが 1 つも収集されなかった場合は、PMD₀ の有効ビットは 0 になる。収集された命令キャッシュ・ミスのレイテンシ (CPU のクロック・サイクル単位) は、PMD₁ のレイテンシ・フィールドに格納される。キャッシュ・モードでは、PMD₀ の TLB ミス・ビットは未定義である。

表 10-16. キャッシュ・モード (PMC₁₀.ct='1x) の命令 EAR (PMD_{0,1})

レジスタ	フィールド	ビット	説明
PMD ₀	ステータス	1:0	ステータス x0: EAR は監視対象となるイベントを収集しなかった。 x1: EAR は有効なイベント・データを格納している。
	命令キャッシュ・ラインのアドレス	63:5	キャッシュ・ミスの原因となった命令キャッシュ・ラインのアドレス
PMD ₁	レイテンシ	11:0	レイテンシ (CPU クロック単位)
	オーバーフロー	12	1 の場合、レイテンシ・カウンタは、データが返される前に、1 回または複数回オーバーフローしている。

10.3.7.3 TLB モードの命令 EAR (PMC₁₀.ct=00)

PMC₁₀.ct が 00 の場合、命令イベント・アドレス・レジスタは、命令 TLB ミスのアドレスを収集する。ユニット・マスクによって、特定の命令 TLB ミスのイベント・アドレスだけを収集するように設定できる。表 10-17 は、命令 TLB の umask の設定をまとめたものである。マスク・ビットのすべての組み合わせがサポートされている。

表 10-17. TLB モード (PMC_{10.ct=00}) の命令 EAR (PMC₁₀) の umask フィールド

ITLB ミスのタイプ	PMC.umask[7:5]	説明
---	000	ディセーブル。何もカウントされない。
L2TLB	xx1	L2 TLB をヒットした L1 ITLB ミス
VHPT	x1x	VHPT をヒットした L1 命令 TLB ミス
FAULT	1xx	ITLB ミス・フォルトにより生成された命令 TLB ミス
ALL	111	L1 ITLB ミスをすべて選択する。 注：すべての組み合わせがサポートされる。

表 10-18 に示すように、L1 ITLB ミスになった命令キャッシュ・ライン・フェッチのアドレスは、PMD₀ に格納される。ステータス・ビット [1] は、収集された TLB ミスが、VHPT 内でヒットしたか、それともソフトウェアによる処理を必要としたかを示す。監視対象となるイベントが 1 つも収集されなかった場合は、PMD₀ の有効ビットは 0 になる。TLB モードでは、PMD₁ のレイテンシ・フィールドは未定義である。

表 10-18. TLB モード (PMC_{10.ct=00}) の命令 EAR (PMD_{0,1})

レジスタ	フィールド	ビット	説明
PMD ₀	ステータス	1:0	ステータス・ビット 00: EAR は監視対象となるイベントを収集しなかった。 01: L2 ITLB 内でヒットした L1 ITLB ミス 10: VHPT 内でヒットした L1 ITLB ミス 11: ITLB ミス・フォルトを生成した L1 ITLB ミス
	命令キャッシュ・ラインのアドレス	63:5	TLB ミスの原因となった命令キャッシュ・ラインのアドレス
PMD ₁	レイテンシ	11:2	TLB モードでは未定義である。

10.3.8 データ EAR (PMC₁₁、PMD_{2,3,17})

データ・イベント・アドレス設定レジスタ (PMC₁₁) の設定によって、L1 データ・キャッシュ・ロード・ミス、L1 データ TLB ミス、または ALAT ミスを監視することができる。図 10-20 と表 10-19 に、PMC₁₁ レジスタのレイアウトを詳しく示す。図 10-21 に、PMC₁₁ に対応するイベント・アドレス・データ・レジスタ PMD_{2,3,17} を示す。設定レジスタ PMC₁₁ 内のモード・ビットは、データ・キャッシュ、データ TLB、または ALAT 監視を選択する。mask フィールドとレジスタ PMD_{2,3,17} の意味は、モード・ビットの設定によって異なる。データ・キャッシュ・ロード・ミスの監視は 10.3.8.1 項「データ・キャッシュ・ロード・ミスの監視 (PMC_{11.mode=00})」、データ TLB の監視は 10.3.8.2 項「データ TLB ミスの監視 (PMC_{11.mode=01})」、ALAT の監視は 10.3.8.3 項「ALAT ミスの監視 (PMC_{11.mode=1x})」をそれぞれ参照のこと。

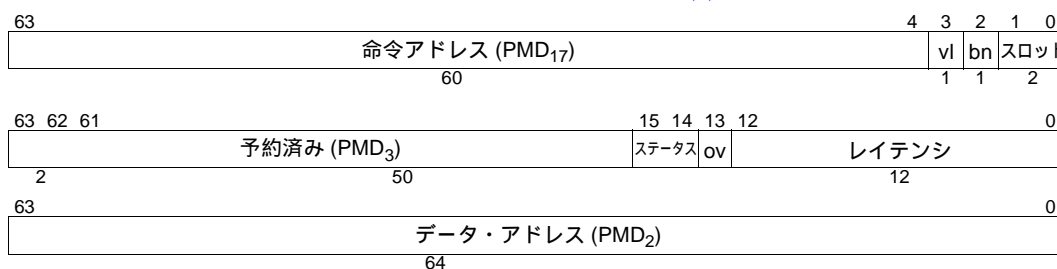
図 10-20. データ・イベント・アドレス設定レジスタ (PMC₁₁)

63	26	25	24	23	20	19	16	15	9	8	7	6	5	4	3	0
予約済み		ism		予約済み		umask		予約済み		モード		pm	予約済み	plm		
38		2		4		4		7		2		1	2	4		

表 10-19. データ・イベント・アドレス設定レジスタのフィールド (PMC₁₁)

フィールド	ビット	説明
plm	3:0	表 10-5 「パフォーマンス・モニタ設定 (PMC) レジスタの制御フィールド (PMC4,5,6,7,10,11,12)」を参照。
pm	6	表 10-5 「パフォーマンス・モニタ設定 (PMC) レジスタの制御フィールド (PMC4,5,6,7,10,11,12)」を参照。
モード	8:7	データ EAR モード・セクタ: `00: L1 データ・キャッシュ・ロード・ミス `01: L1 データ TLB ミス `1x: ALAT ミス
umask	19:16	データ EAR ユニット・マスク モード 00: データ・キャッシュのユニット・マスク (定義は、表 10-20 「データ・キャッシュ・モード (PMC11.mode=00) のデータ EAR (PMC11) の Umask フィールド」を参照) モード 01: データ TLB のユニット・マスク (定義は、表 10-22 「TLB モード (PMC10.ct=01) のデータ EAR (PMC11) の Umask フィールド」を参照)
ism	25:24	表 10-5 「パフォーマンス・モニタ設定 (PMC) レジスタの制御フィールド (PMC4,5,6,7,10,11,12)」を参照。

図 10-21. データ・イベント・アドレス・レジスタのフォーマット (PMD_{2,3,17})



10.3.8.1 データ・キャッシュ・ロード・ミスの監視 (PMC₁₁.mode=00)

データ EAR がデータ・キャッシュ・ロード・ミスを監視するように設定されている場合、umask は、表 10-20 に示すようなロード・レイテンシのスレッシュホールドとして使用される。

表 10-22 に示すように、収集されたデータ・キャッシュ・ロード・ミスの命令 / データ・アドレスとロード・レイテンシは、PMD_{2,3,17} の 3 つのレジスタに格納され、ソフトウェアに渡される。監視対象となるイベントが 1 つも収集されなかった場合は、PMD₃ の有効ビットは 0 になる。

HPW アクセスは監視されない。また、setf、ccv からの読み出し値は監視されない。L1D キャッシュ・ミスは、収集されたミスの後の 7 クロック以内に存在しない場合は、収集されない。セマフォ命令はカウントされる。

表 10-20. データ・キャッシュ・モード (PMC₁₁.mode=00) のデータ EAR (PMC₁₁) の Umask フィールド

umask ビット 19:16	レイテンシのスレッシュホールド [CPU サイクル]	umask ビット 19:16	レイテンシのスレッシュホールド [CPU サイクル]
0000	>= 4 (任意のレイテンシ)	0110	>= 256
0001	>= 8	0111	>= 512
0010	>= 16	1000	>= 1024
0011	>= 32	1001	>= 2048
0100	>= 64	1010	>= 4096
0101	>= 128	1011.. 1111	イベントは収集されていない。

表 10-21. データ・キャッシュ・ロード・ミス・モード (PMC₁₁.mode=00) の PMD_{2,3,17} のフィールド

レジスタ	フィールド	ビット範囲	説明
PMD ₂	データ・アドレス	63:0	ミスの原因となったデータ・アイテムの 64 ビット・アドレス
PMD ₃	レイテンシ	12:0	レイテンシ (CPU クロック単位)
	オーバーフロー	13	オーバーフロー - 1 の場合、レイテンシ・カウンタは、データが戻される前に、1 回または複数回オーバーフローしている。
	ステータス	15:14	ステータス・ビット 00: PMD ₃ の残り、および PMD _{2,17} には、有効な情報はない。 01: PMD _{2,3} 内に有効な情報があり、PMD ₁₇ 内にも有効な情報がある可能性がある。 注: これらのビットは、EAR を再利用する前にクリアする必要がある。
PMD ₁₇	スロット	1:0	スロット・ビット。"vl" が 1 の場合は、メモリ命令の命令バンドル・スロット
	bn	2	バンドル・ビット。"vl" が 1 の場合、これは、収集されたミスに関連のある実行済みのバンドルを示す。
	vl	3	有効ビット 0: 無効なアドレス (EAR は監視対象となるイベントを収集しなかった)。 1: EAR は有効なイベント・データを格納している。 注: このビットは、EAR を再利用する前にクリアする必要がある。
	命令アドレス	63:4	ミス時に実行されていた 2 バンドル配布ウィンドウ内の先頭のバンドルのアドレス。"bn" が 1 の場合、2 番目のバンドルにはメモリ命令が含まれ、アドレスに 16 が追加される。

データ・キャッシュ・ロード・ミスを検出するためには、ロード命令の発行からキャッシュ・ミスの発生までの複数のクロック・サイクルの間、ロード命令を監視しなければならない。任意の時点で複数のロードが未処理になる可能性があるが、Itanium 2 プロセッサのデータ・キャッシュ・ミス・イベント・アドレス・レジスタは一度に 1 つのロードしか監視できないため、一部のデータ・キャッシュ・ロード・ミスが収集されない可能性がある。プロセッサ・ハードウェアは、ロード (監視対象ロードと呼ばれる) のアドレスを検出すると、その監視対象ロードが L1 データ・キャッシュ・ミスかどうか判明するまで、同時に重複する他のすべてのロードを無視する。監視対象ロードがキャッシュ・ミスと判明した場合は、そのパラメータが PMD_{2,3,17} 内に保持される。プロセッサは、どのロード命令を監視するかを無作為に選択することによって、(通常の重複するデータ・キャッシュ・ロード・ミスのシーケンスの中から) 常に同じデータ・キャッシュ・ロード・ミスが収集されるのを防ぐ。この機構は、特定の重複するロード・シーケンス内のすべてのデータ・キャッシュ・ロード・ミスを常に検出するとは限らないが、統計的サンプリングまたはコードのインストルメンテーションには十分な精度を持っている。

10.3.8.2 データ TLB ミスの監視 (PMC₁₁.mode='01)

データ EAR がデータ TLB ミスを監視するように設定されている場合、データ EAR で収集されるデータ TLB ミスは、表 10-23 に示す umask によって決まる。TLB の監視のために、マスク・ビットのすべての組み合わせがサポートされている。

表 10-23 に示すように、収集された DTLB ミスの命令アドレスとデータ・アドレスは、PMD_{2,17} に格納され、ソフトウェアに渡される。監視対象となるイベントが 1 つも収集されなかった場合は、PMD₁₇ の有効ビットは 0 になる。データ TLB 監視モードでは、PMD₃ のレイテンシ・フィールドの内容は未定義である。

ロード TLB ミスおよびストア TLB ミスは、両方収集される。到達されない命令も一部収集される。例えば、あるロードが、L1DTLB でミスしたが、L2 DTLB でヒットし、分岐処理後の命令が

ループに入っている場合、そのロードは収集される。ストア操作および浮動小数点操作は、L1DTLB でミスすることはない。ただし、L2 DTLB をミスするか、フォルトになりソフトウェアで処理される場合がある。

注： このモードでは、PMC₁₂ は 0 でなければならない。そうでない場合は、予測ミスされた分岐の直後に、ミスが原因の誤った IP が提供される。

表 10-22. TLB モード (PMC_{10.ct=01}) のデータ EAR (PMC₁₁) の Umask フィールド

L1 DTLB ミスのタイプ	PMC.umask[19:16]	説明
---	000x	ディセーブル。何もカウントされない。
L2DTLB	xx1x	L2 DTLB をヒットした L1 DTLB ミス
VHPT	x1xx	VHPT をヒットした L1 DTLB ミス
FAULT	1xxx	データ TLB ミスがフォルトを生成した。
ALL	111x	L1 DTLB ミスをすべて選択する。 注：すべての組み合わせがサポートされる。

表 10-23. TLB ミス・モード (PMC_{11.mode=01}) の PMD_{2,3,17} のフィールド

レジスタ	フィールド	ビット範囲	説明
PMD ₂	データ・アドレス	63:0	ミスの原因となったデータ・アイテムの 64 ビット・アドレス
PMD ₃	レイテンシ	12:0	TLB ミス・モードでは未定義である。
	ov	13	TLB ミス・モードでは未定義である。
	ステータス	15:14	ステータス 00: PMD3 の残りおよび PMD2,17 の情報は無効である。 01: L2 データ TLB ヒット 10: VHPT ヒット 11: データ TLB ミスがフォルトを生成した。 注：これらのビットは、EAR を再利用する前にクリアする必要がある。
PMD ₁₇	スロット	1:0	スロット・ビット。"vl" が 1 の場合は、メモリ命令の命令バンドル・スロット。
	bn	2	バンドル・ビット。"vl" が 1 の場合、これは、収集されたミスに関連のある実行済みのバンドルを示す。
	vl	3	有効ビット 0: 無効なアドレス (EAR は監視対象となるイベントを収集しなかった)。 1: EAR は有効なイベント・データを格納している。 注：このビットは、EAR を再利用する前にクリアする必要がある。
	命令アドレス	63:4	ミス時に実行されていた 2 バンドル配布ウィンドウ内の先頭のバンドルのアドレス。".bn" が 1 の場合、2 番目のバンドルにはメモリ命令が含まれ、アドレスに 16 が追加される。

10.3.8.3 ALAT ミスの監視 (PMC_{11.mode=1x})

表 10-24 に示すように、ALAT ミスを引き起こしている命令 (失敗している chk.a および ld.c) のアドレスは、PMD₁₇ に格納され、ソフトウェアに渡される。監視対象となるイベントが 1 つも収集されなかった場合は、PMD₁₇ の有効ビットは 0 になる。ALAT 監視モードでは、PMD₃ のレイテンシ・フィールド、PMD₂ の内容は未定義である。

注： このモードでは、PMC₁₂ は 0 でなければならない。そうでない場合は、予測ミスされた分岐の直後に、ミスが原因の誤った IP が提供される。

表 10-24. ALAT ミス・モード (PMC₁₁.mode='1x) の PMD_{2,3,17} のフィールド

レジスタ	フィールド	ビット範囲	説明
PMD ₂	データ・アドレス	63:0	ALAT ミス・モードでは未定義である。
PMD ₃	レイテンシ	12:0	ALAT ミス・モードでは未定義である。
	ov	13	ALAT ミス・モードでは未定義である。
	ステータス	15:14	ステータス・ビット 00: PMD ₃ の残り、PMD _{2,17} には、有効な情報はない。 01: PMD _{2,3} 内に有効な情報があり、PMD ₁₇ にも有効な情報がある可能性がある。 注: これらのビットは、EAR を再利用する前にクリアする必要がある。
PMD ₁₇	スロット	1:0	スロット・ビット。"vl" が 1 の場合は、メモリ命令の命令バンドル・スロット。
	bn	2	バンドル・ビット。"vl" が 1 の場合、これは、収集されたミスに関連のある実行済みのバンドルを示す。
	vl	3	有効ビット 0: 無効なアドレス (EAR は監視対象となるイベントを収集しなかった)。 1: EAR は有効なイベント・データを格納している。 注: このビットは、EAR を再利用する前にクリアする必要がある。
	命令アドレス	63:4	ミス時に実行されていた 2 バンドル配布ウィンドウ内の先頭のバンドルのアドレス。"bn" が 1 の場合、2 番目のバンドルにはメモリ命令が含まれ、アドレスに 16 が追加される。

10.3.9 分岐トレース・バッファ

分岐トレース・バッファは、最近の Itanium アーキテクチャ分岐命令およびそれらの予測結果に関する情報を提供する。Itanium 2 アーキテクチャ分岐トレース・バッファ設定レジスタ (PMC₁₂) は、どのような条件の下で分岐命令が収集されるかを定義し、トレース・バッファが特定の分岐イベントだけを収集できるようにする。分岐トレース・バッファは、Itanium アーキテクチャ・ベースのコードの実行中 (すなわち、PSR.is が 0 の場合) にのみ有効になる。IA-32 ISA を実行しているときは、分岐トレース・バッファは更新されない。

監視対象となる Itanium アーキテクチャ分岐命令がリタイアされるサイクルごとに、その分岐のソース・バンドル・アドレスとスロット番号が分岐トレース・バッファに書き込まれる。分岐のターゲット・アドレスは、バッファ内の次の位置に書き込まれる。分岐先の命令バンドルにも監視対象となる Itanium アーキテクチャ分岐命令が入っている場合は、分岐トレース・バッファは、(b ビットを 1 にセットして) 1 つのトレース・バッファ・エントリを記録するか、または 2 つのトレース・バッファ・エントリを作成する。2 つのエントリを作成する場合は、1 つは分岐先の命令を分岐ターゲットとして記録し (b ビットをクリアする)、もう 1 つは分岐先の命令を分岐ソースとして記録する (b ビットをセットする)。その結果、分岐トレース・バッファ内に、一連の分岐とターゲットが混在することがある。

10.3.9.1 分岐トレース・バッファ収集の条件

分岐トレース・バッファ設定レジスタ (PMC₁₂) は、分岐命令が収集される条件を定義する。☒ 10-22 と表 10-25 に、これらの条件を示す (この図と表は、分岐予測に関連する条件を示すものである)。これらの条件には以下のものがある。

- 分岐ターゲットを収集するのか、それとも予測に関する追加情報を収集するのか
- 分岐のパス (分岐しない / 分岐した)
- 分岐パスの予測が外れたかどうか
- 分岐ターゲットの予測が外れたかどうか

- どのようなタイプの分岐を収集するのか

図 10-22. 分岐トレース・バッファ設定レジスタ (PMC₁₂)

63	予約済み	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		brt	ppm	ptm	tm	ds	pm	予約 済み										plm
	48	2	2	2	2	1	1	2										4

表 10-25. 分岐トレース・バッファ設定レジスタのフィールド (PMC₁₂)

フィールド	ビット	説明
plm	3:0	表 10-5「パフォーマンス・モニタ設定 (PMC) レジスタの制御フィールド (PMC4,5,6,7,10,11,12)」を参照。
pm	6	表 10-5「パフォーマンス・モニタ設定 (PMC) レジスタの制御フィールド (PMC4,5,6,7,10,11,12)」を参照。
ds	7	データ・セクタ 1: 分岐予測に関する情報を収集する。 0: 分岐ターゲットを収集する。
tm	9:8	分岐処理マスク 11: すべての Itanium アーキテクチャ分岐命令を収集する。 10: 実行された Itanium アーキテクチャ分岐命令だけを収集する。 01: 実行されない Itanium アーキテクチャ分岐命令だけを収集する。 00: 分岐を収集しない。
ptm	11:10	ターゲット・アドレス予測マスク 11: ターゲットの予測結果に関係なく、分岐を収集する。 10: ターゲット・アドレスの予測が当たった分岐だけを収集する。 01: ターゲット・アドレスの予測が外れた分岐だけを収集する。 00: 分岐を収集しない。
ppm	13:12	ブレディケート予測マスク 11: ブレディケートの予測結果に関係なく、分岐を収集する。 10: 分岐のパス (実行された / 実行されない) の予測が当たった分岐だけを収集する。 01: 分岐のパス (実行された / 実行されない) の予測が外れた分岐だけを収集する。 00: 分岐を収集しない。
brt	15:14	分岐タイプ・マスク 11: リターン以外の間接分岐だけを収集する。 10: リターン分岐だけを収集する。 01: IP 相対分岐だけを収集する。 00: すべての分岐を収集する。

これらの条件をまとめると、Itanium アーキテクチャ分岐命令およびそのターゲットは、次の式が成り立つ場合に、トレース・バッファによって収集される。

```
(not PSR.is)
  and (
    (tm[1] and branch taken)
    or (tm[0] and branch not taken)
  )
  and (
    (ptm[1] and hardware predicted target address correctly)
    or (ptm[0] and hardware mispredicted target address)
  )
  and (
    (ppm[1] and hardware predicted the branch path correctly)
    or (ppm[0] and hardware mispredicted the branch path)
  )
  and (
    not ds
  )
)
```

予測が外れた Itanium アーキテクチャ分岐命令をすべて収集するには、PMC₁₂ の Itanium 2 アーキテクチャ分岐トレース・バッファ設定フィールドを、ds=0、tm=11、ptm=01、ppm=01、および brt=11 に設定する必要がある。

命令アドレス範囲マッチング (10.3.5 項) およびオペコード・マッチング (10.3.4 項「オペコード・マッチ・チェック (PMC8,9,15)」) を使用して、分岐トレース・バッファで収集されたものを制限することもできる。

10.3.9.2 分岐トレース・バッファの読み出し

図 10-23. 分岐トレース・バッファ・レジスタのフォーマット (PMD₈₋₁₅、PMC₁₂.ds == 0)

63	アドレス	4	3	2	1	0
60		2	1	1	1	1
		mp	b			
		mp	b			

図 10-24. 分岐トレース・バッファ・レジスタのフォーマット (PMD₈₋₁₅、PMC₁₂.ds == 1)

63	61	60	41	40	4	3	2	1	0
3	20	37	2	1	1	1	1	1	1
3	20	37	2	1	1	1	1	1	1
3	20	37	2	1	1	1	1	1	1

表 10-26. 分岐トレース・バッファ・レジスタのフィールド (PMD₈₋₁₅)

フィールド	ビット範囲	説明
b	0	分岐ビット 1: レジスタの内容は分岐命令である。 0: レジスタの内容は分岐ターゲットであるか、またはレジスタの内容に分岐予測の詳細が含まれる。
mp	1	予測ミス・ビット b=1 かつ mp=1 の場合: (ターゲット予測ミスまたはプレディケート予測ミスによって) 予測ミスになった分岐 b=1 かつ mp=0 の場合: 予測が当たった分岐 b=0 かつ mp=1 の場合: 有効なターゲット・アドレス b=0 かつ mp=0 の場合: 無効な分岐トレース・バッファ・レジスタ
スロット	3:2	b=0 の場合: 未定義 b=1 の場合: バンドル内の最初に処理された分岐命令のスロット・インデックス 00: Itanium スロット 0 の分岐 / ターゲット 01: Itanium スロット 1 の分岐 / ターゲット 10: Itanium スロット 2 の分岐 / ターゲット 11: 分岐しなかった分岐である。
アドレス	63:4	b=1 の場合: Itanium アーキテクチャ分岐命令の 60 ビット・バンドル・アドレス ds=0 かつ b=0 の場合: Itanium アーキテクチャ分岐命令の 60 ビット・ターゲット・バンドル・アドレス ds=0 かつ b=1 の場合: Itanium アーキテクチャ分岐命令のバンドル・アドレスの上位 3 ビットと下位 37 ビット、および収集された分岐に関連する L1 IBR の下位 20 ビット

8 つの分岐トレース・バッファ・レジスタ PMD₈₋₁₅ は、収集された一連の分岐の結果に関する情報を格納する。分岐トレース・バッファ・レジスタ (PMD₈₋₁₅) は、イベント収集機能がフリーズされているときのみ (PMC₀.fr=1)、有効なデータを読み出せる。イベント収集機能が有効になっている間に PMD₈₋₁₅ を読み出した場合は、戻り値は未定義である。これらのレジスタは、図 10-23、図 10-24、表 10-26 に示すレイアウトに従って、収集された分岐命令のアドレス (b ビット=1 の場合)、分岐ターゲットのアドレス (b ビット=0 の場合)、または分岐予測の詳細を格納する。分岐命令の場合、mp ビットは分岐の予測ミスを示す。分岐トレース・レジスタの b ビットが 0、mp ビットが 0 の場合は、無効な分岐トレース・バッファ・エントリを示す。スロット・フィールドは、収集された命令バンドル内の最初に処理された Itanium アーキテクチャ分岐命令のスロット番号を格納する。スロット番号が 3 の場合は、実行されなかった分岐を示す。IA-32 への分岐 (br.ia) のターゲット・アドレス・バンドルは記録される。IA-32 の JMPE 分岐命令およびその Itanium アーキテクチャ・ターゲットは記録されない。

監視対象となる Itanium アーキテクチャ分岐命令がリタイアされるサイクルごとに¹、その分岐のソース・バンドル・アドレスとスロット番号が分岐トレース・バッファに書き込まれる。次のクロック内に、リタイアして同じ条件を満たす分岐がターゲット命令バンドルに格納される場合は、

2 番目の分岐のアドレスが格納される。それ以外の場合は、分岐のターゲット・アドレス (PMC₁₂.ds=0) または分岐予測の詳細 (PCM₁₂.ds=1) が、バッファ内の次の位置に書き込まれる。その結果、分岐トレース・バッファ内に、一連の分岐とターゲットが混在する場合がある。

Itanium 2 アーキテクチャ分岐トレース・バッファは、監視対象となる Itanium アーキテクチャ分岐命令のうち最後の 4 ~ 8 個を格納する循環バッファである。図 10-25、表 10-27 に示す分岐トレース・バッファ・インデックス・レジスタ (PMD₁₆) は、最近記録された分岐またはターゲットを示す。監視対象となる分岐またはターゲットが記録されるサイクルごとに、分岐バッファ・インデックス (bbi) はポストインクリメントされる。8 つのエントリが記録されると、分岐インデックスはラップアラウンドし、トレース・バッファの最初のエントリは、次の分岐で上書きされる。このラップ操作それ自体は、PMD₁₆ のフル・ビットに記録される。PMD₁₆ の bbi フィールドは、次に書き込まれる分岐バッファのインデックスを定義する。以下の公式では、PMD₁₆ の内容から、最後に書き込まれた分岐トレース・バッファの PMD インデックスを計算する。

$$\text{最後に書き込まれた PMD インデックス} = 8 + ((8 * \text{PMD}_{16}.\text{full}) + (\text{PMC}_{16}.\text{bbi} - 1)) \% 8$$

PMD₁₆ のフル・ビットと bbi フィールドが 0 になっている場合は、分岐トレース・バッファは、監視対象となる分岐を 1 つも収集していない。分岐トレース・バッファが PMD₁₅ から PMD₈ にラップするたびに、PMD₁₆ のフル・ビットがセットされる。このフル・ビットは、一度セットされると、ソフトウェアが明示的にクリアするまでセットされたままになる (つまり、このビットはスティッキー・ビットである)。ソフトウェアは、PMD16 への書き込みによって、bbi インデックスとフル・ビットをリセットできる。

図 10-25. 分岐トレース・バッファ・インデックス・レジスタのフォーマット (PMD₁₆)

63	36	35	32	31	28	27	24	23	20	19	16	15	12	11	8	7	4	3	2	0	
予約済み				pmd15	pmd14	pmd13	pmd12	pmd11	pmd10	pmd9	pmd8	フル	bbi								
				ext	ext	ext	ext	ext	ext	ext	ext	ext	フル	bbi							
28				4	4	4	4	4	4	4	4	4	1	3							

表 10-27. 分岐トレース・バッファ・インデックス・レジスタのフィールド (PMD₁₆)

フィールド	ビット範囲	説明
bbi	2:0	分岐バッファ・インデックス [範囲は 0..7 - インデックス 0 は PMD ₈ を示す] 次書き込まれる分岐トレース・バッファ・エントリへのポインタ フル・ビット = 1 の場合: 記録済みの分岐 / ターゲットのうち最も古いものを指す。 フル・ビット = 0 の場合: 次に書き込まれる位置を指す。
フル	3	フル・ビット (スティッキー) フル・ビット = 1 の場合: 分岐トレース・バッファがラップした。 フル・ビット = 0 の場合: 分岐トレース・バッファがラップしていない。
pmd8 ext	7:4	ビット [7:6] は未使用 ビット [5] (brflush): PMD8. ビット [1:0] = 01 の場合、 1 = バックエンドが分岐を予測ミスし、パイプラインがそれによってフラッシュされた。 0 = この分岐に関連しているパイプライン・フラッシュはない。 bit[4] (b1): b = 1 の場合 1 = 分岐はバンドル 1 からのものであり、0x1 が PMD8. ビット [63:4] に追加される。 0 = 分岐はバンドル 0 からのものである。
pmd9 ext	11:8	PMD9 については、上記と同じ。

- Itanium 2 プロセッサの分岐トレース・バッファは、分岐以外の命令のソース・アドレスを収集するときがある (ターゲット・アドレスを収集することはない)。この状態は、分岐命令のトラップや、br.ia、break.b、マルチウェイ分岐のフォルトで発生する。

表 10-27. 分岐トレース・バッファ・インデックス・レジスタのフィールド (PMD_i) (続き)

フィールド	ビット範囲	説明
pmd10 ext	15:12	PMD10 については、上記と同じ。
pmd11 ext	19:16	PMD11 については、上記と同じ。
pmd12 ext	23:20	PMD12 については、上記と同じ。
pmd13 ext	27:24	PMD13 については、上記と同じ。
pmd14 ext	31:28	PMD14 については、上記と同じ。
pmd15 ext	35:32	PMD15 については、上記と同じ。

10.3.10 割り込み

表 10-6 で説明したように、以下の条件が該当する場合は、レジスタ PMD_{4,5,6,7} のそれぞれが割り込みを発生させる。

- $PMC_i.oi=1$ (すなわち、PMD_i に対してオーバーフロー割り込みが有効) であり、PMD_i がオーバーフローする場合。どの PMC/PMD の設定がこの条件を満たしているかに関係なく、発生する割り込みラインは 1 つだけであることを注意する。

この割り込みは、ベクタ = 0x3000 の「外部割り込み」であり、以下の条件が該当する場合にのみ認識される。

- $PMV.m=0$ および $PMV.vector$ が正常に設定されている場合。すなわち、パフォーマンス・モニタ割り込みはマスクされず、"mov cr73=r2" を実行して、適切なベクタがこの割り込みに設定される場合。
- $PSR.i=1$ かつ $PSR.ic=1$ である場合。すなわち、割り込みがマスク解除され、"ssm imm" 命令または "mov psr.l=r2" 命令を実行して、プロセッサ・ステータス・レジスタで割り込み収集が有効にされる場合。
- $TPR.mmi=0$ (すなわち、すべての外部割り込みがマスクされない) であり、 $TPR.mic$ の値が、パフォーマンス・モニタ割り込みが属しているプライオリティ・クラスがマスクされないような値である場合。例えば、『インテル® Itanium® 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 巻の表 5-7「割り込みの優先順位、イネーブルおよびマスク」に従って、パフォーマンス・モニタ割り込みにベクタ 0xD2 を割り当てると、この値はプライオリティ・クラス 13 になる。そのため、この割り込みを認識するには、 $TPR.mic$ の値が 13 未満であればよい。このレジスタへの書き込みは、"mov cr66=r1" によって実行される。
- より高い優先順位のフォルト、トラップ、または未処理の外部割り込みはない。

処理が必要な最高優先順位の外部割り込みを理解するには、IVR レジスタ "mov r1=cr65" を割り込みサービス・ルーチンで読み出す必要がある。

パフォーマンス・モニタは、割り込みサービス・ルーチンから戻る前に、割り込みがクリアされるように初期化する必要がある。これを行うには、 $PMC.oi$ をクリアするか、割り込みを発生させた PMD を再初期化すればよい (割り込みを発生させた PMD は、 PMC_0 を読み出すことでわかる)。この他に、さらに監視を行う必要がある場合は、 PMC_0 のビットをすべてクリアする必要がある。

10.3.10.1 外部イベント

表 10-6 で説明したように、以下の条件が該当する場合は、各 PMD により、BPM# ピン上に外部イベントが発生する。現在信号は、PMD [47] のオーバーフロー・ビットの値を示す。ビット W=0 を指定して PMD が再書き込みされた場合、または PMD がもう一度オーバーフローする場合、0 から 1 への遷移が行われると、1 から 0 の遷移も行われる。

- $PMC_i.ev=1$ であり (すなわち、外部イベントが PMD_i 用に有効にされる)、PMD_i がオーバーフローする場合 ($PMD_i=1$ の読み出しビット 47)。これに割り当てられるのは、プロセッサ・ピンの 1 つのみである。これは、どの PMD がこの条件を満たしているかに関係なく発生する。このピンは、これらの条件が満たされている限り、high であり続ける。

- BPM[5:0] は、デバッグおよびパフォーマンス監視用に割り当てられた、双方向のプロセッサ・ピンである。現時点では、これらのピンを有効にする正確な方法は公開されていない。ただし、これらのピンの方向（入力または出力）、ピンに現れる情報（出力）、情報の用途（入力）を決める方法が、今後提供される予定である。

10.3.11 プロセッサ・リセット、PAL コール、および低消費電力状態

プロセッサ・リセット：すべての PMC レジスタのプロセッサ・ハードウェア・リセット・ビット oi および ev が 0 になっており、 $PMV.m$ が 1 にセットされているとき。この状態では、割り込みは生成されず、イベントは外部に通知されない。リセット時には、命令アドレス範囲チェック、オペコード・マッチャー、データ・アドレス範囲チェックは、PAL ファームウェアによって以下のように初期化される。

- $PMC_{8,9} = 0xffffffffffff$, (すべてのオペコードにマッチ)
- $PMC_{13} = 0x2078fefefefe$, (メモリ・パイプライン・イベントの制限なし)
- $PMC_{14} = 0xdb6$, (命令アドレス範囲の制限なし)
- $PMC_{15} = 0xfffff0$, (オペコード・マッチの制限なし)

パフォーマンス監視機能に関連する、その他のすべての状態は未定義である。

表 10-28. Itanium® 2 プロセッサについて PAL_PERF_MON_INFO が返す情報

PAL_PERF_MON_INFO の戻り値	説明	Itanium® 2 プロセッサ固有の値
PAL_RETIRE	リタイアしたタグなし Itanium 命令数をカウントするイベント・タイプを示す符号なし 8 ビット値	0x08
PAL_CYCLES	実行中の CPU サイクル数をカウントするイベント・タイプを示す符号なし 8 ビット値	0x12
PAL_WIDTH	実装されているカウンタ・ビット数を示す符号なし 8 ビット値	48
PAL_GENERIC_PM_PAIRS	汎用 PMC/PMD のペアの数を示す符号なし 8 ビット値	4
PAL_PMCmask	どの PMC レジスタに値が与えられるかを定義する 256 ビット・マスク	0x3FFF
PAL_PMDmask	どの PMD レジスタに値が与えられるかを定義する 256 ビット・マスク	0x3FFFF
PAL_CYCLES_MASK	どの PMC/PMD カウンタが実行中の CPU サイクルをカウントできるかを定義する 256 ビット・マスク (PAL_CYCLES によって定義されたイベント)	0xF0
PAL_RETIRE_MASK	どの PMC/PMD カウンタが、リタイアしたタグなし Itanium 命令の数をカウントできるかを定義する 256 ビット・マスク (PAL_RETIRE によって定義されたイベント)	0xF0

PAL コール：『インテル® Itanium® 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 巻に定義されているように、PAL コール PAL_PERF_MON_INFO は、実装されているパフォーマンス・モニタに関する情報をソフトウェアに提供する。表 10-28 に、Itanium 2 プロセッサ固有の値を示す。

低消費電力状態：プロセッサが低消費電力状態に移行したときにモニタのカウントが保持されるように、PAL_LIGHT_HALT は、プロセッサを低消費電力状態にする前にイベント監視機能をフリーズする。その結果、低消費電力状態（スヌープなど）で発生しているバス・イベントは、カウントされない。PAL_LIGHT_HALT は、 PMC_0 レジスタの元の値を保持する。

11.1 概要

本章では、第 10 章の初めに説明したパフォーマンス監視機構で計測できる、Itanium® 2 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・イベントを説明する。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、よく使用されるパフォーマンス評価規準の計算方法も解説する。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は、プロセッサ・イベントをアルファベット順に並べた第 11 章「イベントのカテゴリ化」でさらに詳しく説明する。

Itanium 2 プロセッサでは、多数のイベントを監視できる。大部分のイベントは任意の PMD₄₋₇ への入力として選択できる。それを行うには、イベント・リストの「イベント・コード」列に示された 16 進値を、対応する PMC のビット [15:8] に設定すればよい。さらに特別な要件を持つイベントについては、11.8.2 項および 11.8.3 項を参照のこと。

11.2 イベントのカテゴリ化

パフォーマンスに関連したイベントは、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 (11.3 節)
- 命令ディスパースル・イベント：命令のデコードと発行 (11.4 節)
- 命令実行イベント：命令の実行、データ・スペキュレーション、コントロール・スペキュレーション、メモリ操作 (11.5 節)
- ストール・イベント：ストールと実行のサイクルに関する分析 (11.6 節)
- 分岐イベント：分岐予測 (11.7 節)
- メモリ階層：命令キャッシュとデータ・キャッシュ (11.8 節)
- システム・イベント：オペレーティング・システム・モニタ (11.9 節)
- TLB イベント：命令 TLB とデータ TLB (11.10 節)
- システム・バス・イベント：(11.11 節)
- RSE イベント：レジスタ・スタック・エンジン (11.12 節)

上記の各節には、直接計測可能なイベントの情報を説明した表が記載されている。また、直接計測可能なイベントから派生するイベントの表も必要に応じて記載されている。これらの派生イベントでは、既存のイベントまたは現在のステップを単純にリネームして、よく使用されるパフォーマンス評価基準の値を決定できる。ただし、派生イベントは、11.14 節のイベント・リストには記載されていない。

直接計測できるイベントは、PMC.umask フィールド (10.3.2 項「パフォーマンス・カウンタ・レジスタ」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) として計測するケースが多い。こうしたイベントのシンボリック・イベント名には、umask の使用を示すためにピリオドが 1 つ含まれている。イベント・リストには、この unmask は 4 ビットで指定されているが、その中の x は任意値を示す記号である。

以降の各節に記載した各表では、イベントごとに以下の属性を示している。

- **シンボル名** - 当該イベントを示すために使用するシンボリック名。
- **イベント・コード** - 当該イベントを計測するために、該当する PMC レジスタのビット [15:8] に設定する 16 進値。
- **IAR** - 当該イベントを IAR (Instruction Address Range) レジスタで制限できるかどうか。
- **DAR** - 当該イベントを DAR (Data Address Range) レジスタで制限できるかどうか。
- **OPC** - 当該イベントをオペコード・マッチ・レジスタで制限できるかどうか。
- **最大インクリメント/サイクル** - 当該イベントを各サイクルで増分する最大インクリメント/サイクルまたは最大値。
- **説明** - 当該イベントに関する簡単な説明。

11.3 基本イベント

表 11-1 は、2 つの基本実行モニタについてまとめたものである。CPU_CYCLES イベントを使用すると、現在実行中の命令セットに基づいて PMC/PMD を制限すると、Itanium アーキテクチャ・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できる。リタイアした Itanium 2 命令数 (IA64_INST_RETIRED) には、プレディケート付きの真命令と nop 命令が含まれるが、RSE の実行回数は含まれない。

表 11-1. 基本イベントのパフォーマンス・モニタ

シンボル名	イベント・コード	IAR	DAR	OPC	最大インクリメント/サイクル	説明
CPU_CYCLES	0x12	N	N	N	1	CPU サイクル
IA64_INST_RETIRED	0x08	Y	N	Y	6	リタイアした Itanium 命令
IA32_INST_RETIRED	0x59	N	N	N	2	リタイアした IA-32 命令
IA32_ISA_TRANSITIONS	0x07	N	N	N	1	Itanium 命令と IA-32 命令 ISA との間の遷移

表 11-2. 基本イベントの派生モニタ

シンボル名	説明	式
IA64_IPC	Itanium アーキテクチャ・ベースのコード・シーケンス中における、1 サイクル当たりの平均 Itanium 命令数	IA64_INST_RETIRED / CPU_CYCLES
IA32_IPC	IA-32 コード・シーケンス中における、1 サイクル当たりの平均 IA-32 命令数	IA32_INST_RETIRED / CPU_CYCLES
AVG_CPT	ISA 遷移当たりの平均サイクル数	CPU_CYCLES / (ISA_TRANSITIONS * 2)
AVG_IA32_IPT	ISA 遷移当たりの平均 IA-32 命令数	IA32_INST_RETIRED / (IA32_ISA_TRANSITIONS / 2)
AVG_IA64_IPT	ISA 遷移当たりの平均 Itanium 命令数	IA64_INST_RETIRED / (IA32_ISA_TRANSITIONS / 2)

11.4 命令ディスパーサル・イベント

命令キャッシュ・ラインは実行コアに送られ、Itanium 2 プロセッサの機能ユニットに配布される。Itanium 2 プロセッサは、1 クロック・サイクル当たり、6 つの命令を発行または配布できる。言い換えれば、Itanium 2 プロセッサは、6 つの命令スロット（またはシラブル）に対して発行を行うことができる。次に示すイベントは、命令がどれだけ効率的に配布されるか、命令がフル・キャパシティで配布されない理由を知るためのものである。命令がフル・キャパシティで配布されない理由は 5 つある。1 つは、DISP_STALLED で計測される。ディスパーサルがストールされるすべてのクロックで、ディスパーサルは 6 つのシラブルをヒットする。残りの 4 つの理由は、SYLL_NOT_DISPERSSED で計測される。ハードウェアの設計上の理由により、SYLL_NOT_DISPERSSED には、暗黙的および明示的なビットが原因でオーバーカウントが含まれるときがある。このオーバーカウントの数は少ないが、SYLL_OVERCOUNT によって、そのカウントが正確にわかる。

これらのイベント間の関係は、次のとおりである。

$$6*(CPU_CYCLES-DISP_STALLED) = INST_DISPERSSED + SYLL_NOT_DISPERSSED.ALL - SYLL_OVERCOUNT.ALL$$

表 11-3. 命令ディスパーサル・イベントのパフォーマンス・モニタ

シンボル名	イベント・コード	I A R	D A R	O P C	最大 インクリ メント/ サイクル	説明
DISP_STALLED	0x49	N	N	N	1	ディスパーサルがストールされたサイクル数
INST_DISPERSSED	0x4d	Y	N	N	6	REN から REG ステージに配布されたシラブル数
SYLL_NOT_DISPERSSED	0x4e	Y	N	N	5	配布されなかったシラブル
SYLL_OVERCOUNT	0x4f	Y	N	N	2	オーバーカウントされたシラブル

11.5 命令実行イベント

リタイアした命令数 (IA64_TAGGED_INST_RETIRED と NOPS_RETIRED) は、アドレス範囲チェック機能とオペコード・マッチ機能によって規定されるタグ情報に基づいている。プレディケート・オフ命令をカウントするには、個別のイベントである PREDICATE_SQUASHED_RETIRED を使用する。

表に示す FP モニタは、浮動小数点演算が原因で発生したパイプライン・フラッシュとフラッシュ・ツー・ゼロの情報を動的に収集する。FP_OPS_RETIRED イベントは、リタイアした FP 演算の数をカウントする。

表 11-4 に示すように、コントロール・スペキュレーションやデータ・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収集する。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 (INST_CHKA_LDC_ALAT.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 (INST_FAILED_CHKA_LDC_ALAT.ALL) 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 2 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。そのため、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 (INST_CHKA_LDC_ALAT.ALL) が必要である。

表 11-4. 命令実行イベントのパフォーマンス・モニタ

シンボル名	イベント・コード	I A R	D A R	O P C	最大 インクリ メント/ サイクル	説明
ALAT_CAPACITY_MISS	0x58	Y	Y	Y	2	置き換えられた ALAT エントリ
FP_FAILED_FCHKF	0x06	Y	N	N	1	失敗した fchkf
FP_FALSE_SIRSTALL	0x05	Y	N	N	1	トラップなしの SIR ストール
FP_FLUSH_TO_ZERO	0x0b	Y	N	N	2	ゼロ・フラッシュされた FP の結果
FP_OPS_RETIRED	0x09	Y	N	N	8	リタイアした FP 演算
FP_TRUE_SIRSTALL	0x03	Y	N	N	1	アサートされトラップに至る SIR ストール
IA64_TAGGED_INST_RETIRED	0x08	Y	N	Y	6	リタイアしたタグ付き命令
INST_CHKA_LDC_ALAT	0x56	Y	Y	Y	2	アドバンスト・チェック・ロード
INST_FAILED_CHKA_LDC_ALAT	0x57	Y	Y	Y	1	失敗したアドバンスト・チェック・ロード
INST_FAILED_CHKS_RETIRED	0x55	N	N	N	1	失敗したスペキュレーティブ・チェック・ロード
LOADS_RETIRED	0xcd	Y	Y	Y	4	リタイアしたロード
MISALIGNED_LOADS_RETIRED	0xce	Y	Y	Y	4	リタイアしたアライメントの合っていないロード命令
MISALIGNED_STORES_RETIRED	0xd2	Y	Y	Y	2	リタイアしたアライメントの合っていないストア命令
NOPS_RETIRED	0x50	Y	N	Y	6	リタイアした NOP 命令
PREDICATE_SQUASHED_RETIRED	0x51	Y	N	Y	6	プレディケート・オフのために実行されなかった命令
STORES_RETIRED	0xd1	Y	Y	Y	2	リタイアしたストア
UC_LOADS_RETIRED	0xcf	Y	Y	Y	4	リタイアしたキャッシュ不可ロード
UC_STORES_RETIRED	0xd0	Y	Y	Y	2	リタイアしたキャッシュ不可ストア

表 11-5. 命令実行イベントの派生モニタ

シンボル名	説明	式
ALAT_EAR_EVENTS	EAR によって収集された ALAT イベントの数をカウントする。	DATA_EAR_EVENTS
CTRL_SPEC_MISS_RATIO	コントロール・スペキュレーションのミス率	INST_FAILED_CHKS_RETIRED.ALL / IA64_TAGGED_INST_RETIRED[chk.s]
DATA_SPEC_MISS_RATIO	データ・スペキュレーションのミス率	INST_FAILED_CHKA_LDC_ALAT.ALL / INST_CHKA_LDC_ALAT.ALL

11.6 ストール・イベント

Itanium 2 プロセッサのストール・アカウンティングは、フロントエンドのストール・アカウンティングとバックエンドのストール・アカウンティングに分類される。バックエンド・イベントとフロントエンド・イベントは、それぞれ別のパイプライン・ステージでカウントされるため、比較すべきではない。

バックエンドは、FPU/L1D、RSE、EXE、分岐/例外、またはフロントエンドの5つの異なる機構が原因でストールされる場合がある。BACK_END_BUBBLEでは、ストールを引き起こしている機構の概要が分かる。その他のバックエンド・カウンタは、情報をカテゴリ別に分類してさらに詳しく提供する。ストールが発生すると、そのたびにパイプライン内の1個所にバブルが挿入される。フラッシュが発生すると、そのたびにパイプライン内のすべての個所にバブルが挿入される。メイン・パイプの操作を模倣するために、BACK_END_BUBBLEを除くバックエンド・ストール・アカウンティング・イベントには、優先順位が付けられる。(優先順位は、高い順にBE_FLUSH_BUBBLE.XPN、BE_FLUSH_BUBBLE.BRU、L1D_FPUストール、EXEストール、RSEストール、フロントエンド・ストールとなる)。この優先順位によって、イベントが相互に排他的であることが保証され、最も重要な原因(パイプライン内の最新のイベント)のみが確実にカウントされる。

Itanium 2 プロセッサのフロントエンドは、FEFLUSH、TLBMISS、IMISS、分岐、FILL-RECIRC、BUBBLE、IBFULLの7つの異なる機構が原因でストールする場合がある(これらの機構は、優先順位の高い順に並べてある)。フロントエンドのストールがパイプラインに与える影響は全く同じであるため、そのアカウンティングはより単純になる。

すべてのクロックの間、バックエンド・パイプラインは、バブルを持つか、1つまたは複数の命令(CPU_CYCLES = BACK_END_BUBBLE.all + (IA64_INST_RETIRED >= 1))をリタイアする。パイプラインのバックエンドで発生したバブルをさらに詳しく調査するには、次の等式を使用する。

$$\text{BACK_END_BUBBLE.all} = \text{BE_RSE_BUBBLE.all} + \text{BE_EXE_BUBBLE.all} + \text{BE_L1D_FPU_BUBBLE.all} + \text{BE_FLUSH_BUBBLE.all} + \text{BACK_END_BUBBLE.fe}$$

各ストール・イベント(表 11-6 に示す)は、umask を用いて、いくつかの使用可能なサブイベントを選択する。使用可能なサブイベントとそれぞれの解説は、11.14 節に示す、イベントの詳細な説明を参照のこと。

表 11-6. ストール・イベントのパフォーマンス・モニタ

シンボル名	イベント・コード	I A R	D A R	O P C	最大 インクリ メント/ サイクル	説明
BACK_END_BUBBLE	0x00	N	N	N	1	メイン・パイプ内のフル・パイプ・バブル
BE_EXE_BUBBLE	0x02	N	N	N	1	実行ユニット・ストールが原因のメイン・パイプ内のフル・パイプ・バブル
BE_FLUSH_BUBBLE	0x04	N	N	N	1	フラッシュが原因のメイン・パイプ内のフル・パイプ・バブル
BE_L1D_FPU_BUBBLE	0xca	N	N	N	1	FP または L1D キャッシュが原因のメイン・パイプ内のフル・パイプ・バブル
BE_LOST_BW_DUE_TO_FE	0x72	N	N	N	2	BE がその他の理由でストールされていない場合の無効なバンドル
BE_RSE_BUBBLE	0x01	N	N	N	1	RSE ストールが原因のメイン・パイプ内のフル・パイプ・バブル
FE_BUBBLE	0x71	N	N	N	1	FE によって検出されたバブル
FE_LOST_BW	0x70	N	N	N	2	IB への入り口での無効なバンドル
IDEAL_BE_LOST_BW_DUE_TO_FE	0x73	N	N	N	2	IB からの出口での無効なバンドル

11.7 分岐イベント

分岐イベントの場合、リタイアメントとは、そのプレディケート値に関わらず、分岐が到達しコミットされたことを意味する。予測結果に関する詳細は、モニタのペアに含まれている。正確な予測ミスのカウントするには、次の式を使用する。

$$BR_MISPRED_DETAIL.[umask] - BR_MISPRED_DETAIL2.[umask]$$

すべての umask についてこの計算を実行すると、BR_MISPRED_DETAIL イベントの正確な値が得られる。

BR_PATH_PRED の正確な値を得る方法は、これと若干異なる。バンドル内に複数の分岐が存在し、その1つが実行されると予測されている場合、より大きな番号のポートはすべて、それらの正確な予測を実際に知らずに、実行されない予測モードに強制される。

OKPRED_NOTTAKEN の正確な予測パス情報は、次の式を計算すると得られる。

$$BR_PATH_PRED.[分岐タイプ].OKPRED_NOTTAKEN - BR_PATH_PRED2.[分岐タイプ].UNKNOWNPRED_NOTTAKEN.$$

「分岐タイプ」には、両方のイベントに指定されたのと同じものを指定する (ALL、IPREL、RETURN、NRETIND のいずれか)。

同様に、MISPRED_TAKEN の正確な予測パス情報は、次の式を計算すると得られる。

$$BR_PATH_PRED.[分岐タイプ].MISPRED_TAKEN - BR_PATH_PRED2.[分岐タイプ].UNKNOWNPRED_TAKEN.$$

「分岐タイプ」には、両方のイベントに選択されたのと同じものを指定する (ALL、IPREL、RETURN、NRETIND のいずれか)。

BRANCH_EVENT は、分岐トレース・バッファ (分岐 EAR と呼ばれる) によって収集されたイベントの数をカウントする。分岐 EAR の詳細は、10.3.9 項「分岐トレース・バッファ」を参照のこと。

表 11-7. 分岐イベントのパフォーマンス・モニタ

シンボル名	イベント・コード	I A R	D A R	O P C	最大 インクリ メント/ サイクル	説明
BE_BR_MISPRED_DETAIL	0x61	Y	N	Y	1	BE 分岐予測ミスの詳細
BRANCH_EVENT	0x11	Y	N	Y	1	収集された分岐イベント
BR_MISPRED_DETAIL	0x5b	Y	N	Y	3	FE 分岐予測ミスの詳細
BR_MISPRED_DETAIL2	0x68	Y	N	Y	2	FE 分岐予測ミスの詳細 (不明なパス・コンポーネント)
BR_PATH_PRED	0x54	Y	N	Y	3	FE 分岐パス予測の詳細
BR_PATH_PRED2	0x6a	Y	N	Y	2	FE 分岐パス予測の詳細 (不明な予測コンポーネント)
ENCBR_MISPRED_DETAIL	0x63	Y	N	Y	1	リタイアしたエンコード済み分岐の数

11.8 メモリ階層

本節では、Itanium 2 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、以下のように分類される。

- L1 命令キャッシュと命令プリフェッチ・イベント (11.8.1 項)
- L1 データ・キャッシュ・イベント (11.8.2 項)
- L2 ユニファイド・キャッシュ・イベント (11.8.3 項)
- L3 キャッシュ・イベント (11.8.4 項)

図 11-1 に、Itanium 2 プロセッサの 3 層のメモリ階層とそのイベント・モニタの概要を示す。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 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 L2 キャッシュの下位には、容量の大きいユニファイド 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 つの節で説明する。

11.8.1 L1 命令キャッシュと命令プリフェッチ・イベント

表 11-8 は、Itanium 2 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。この命令フェッチ・モニタは、デマンド・フェッチ動作 (L1I_READS) とプリフェッチ動作 (L1I_PREFETCHES) 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 と ISB_LINES_IN) によって監視される。L1I_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 L1ITLB ミスを補足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用してその適用範囲を絞れるが、オペコード・マッチャーを使用し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、分岐パ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、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に基づいている。そのため、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 (詳細は、10.3.5 項「命令アドレス範囲マッチング」を参照のこと)。

L1I_EAR_EVENTS は、Itanium 2 プロセッサの命令 EAR によって収集されたイベントの数をカウントする。命令 EAR の詳細は、10.3.7 項「イベント・アドレス・レジスタ (PMC10,11/PMD0,1,2,3,17)」を参照のこと。

表 11-8. L1 命令キャッシュ・イベントおよびプリフェッチ・イベントのパフォーマンス・モニタ

シンボル名	イベント・コード	I A R	D A R	O P C	最大 インクリ メント/ サイクル	説明
ISB_BUNPAIRS_IN	0x46	Y	N	N	1	L2 から FE に書き込まれたバンドル・ペア
L1I_EAR_EVENTS	0x43	Y	N	N	1	命令 EAR イベント
L1I_FETCH_ISB_HIT	0x66	Y	N	N	1	ISB でヒットし ISB からバイパスされている「ジャストインタイム」命令フェッチ
L1I_FETCH_RAB_HIT	0x65	Y	N	N	1	RAB 内でヒットしている命令フェッチ
L1I_FILLS	0x41	Y	N	N	1	L1 命令キャッシュ・フィル
L1I_PREFETCHES	0x44	Y	N	N	1	L1 命令プリフェッチ要求
L2_INST_DEMAND_READS	0x42	Y	N	N	1	L1 命令キャッシュおよび ISB ミス
L1I_PREFETCH_STALL	0x67	N	N	N	1	プリフェッチ・パイプラインがストールした理由
L1I_PURGE	0x4b	Y	N	N	1	L1 命令によって処理される L1ITLB パージ
L1I_PVAB_OVERFLOW	0x69	N	N	N	1	PVAB オーバーフロー
L1I_RAB_ALMOST_FULL	0x64	N	N	N	1	RAB はほぼいっぱいか?
L1I_RAB_FULL	0x60	N	N	N	1	RAB はいっぱいか?
L1I_READS	0x40	Y	N	N	1	L1 命令キャッシュ読み出し
L1I_SNOOP	0x4a	Y	Y	Y	1	L1 命令によって処理されるスヌープ要求
L1I_STRM_PREFETCHES	0x5f	Y	N	N	1	L1 命令キャッシュ・ライン・プリフェッチ要求
L2_INST_PREFETCHES	0x45	Y	N	N	1	L2 命令プリフェッチ要求

表 11-9. L1 命令キャッシュ・イベントおよびプリフェッチ・イベントの派生モニタ

シンボル名	説明	式
ISB_LINES_IN	L2 (およびそれ以上のキャッシュ) からフロントエンドに書き込まれたキャッシュ・ラインの数	ISB_BUNPAIRS_IN/4
L1I_DEMAND_MISS_RATIO	L1 命令デマンドのミス率	$L2_INST_DEMAND_READS / L1I_READS$
L1I_PREFETCH_MISS_RATIO	L1 命令プリフェッチのミス率	$L2_INST_PREFETCHES / L1I_PREFETCHES$
L1I_REFERENCES	L1 命令キャッシュ読み出しおよびフィルの数	$L1I_READS + L1I_PREFETCHES$

11.8.2 L1 データ・キャッシュ・イベント

表 11-10 に、Itanium 2 プロセッサの L1 データ・キャッシュ・モニタを示す。図 11-1 に示したとおり、ライトスルーである L1 データ・キャッシュは、キャッシュ可能ロード、整数ロード、RSE ロード、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用される。DATA_REFERENCES は、発行されたデータ・メモリ参照の回数である。

L1 データ・キャッシュ読み出し (L1D_READS) と L1 データ・キャッシュ・ミス (L1D_READ_MISSES) は、L1 データ・キャッシュの読み出しヒット / ミス率を監視する。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。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。データ EAR の詳細は、10.3.8 項「データ EAR (PMC11、PMD2,3,17)」を参照のこと。

L1D キャッシュ・イベントは、5 つのセットに分割されている。L1D キャッシュ・イベントの別のセットからのイベントを同時に計測はできない。各セットは、PMC5 に設定されたイベント・コードによって選択される (すなわち、このセット内のイベントを計測したい場合、そのうちの 1 つは PMD5 で計測しなければならない)。umask に関する制限はない。各セットに属するモニタは、次の節で詳しく説明する。

表 11-10. L1 データ・キャッシュ・イベントのパフォーマンス・モニタ

シンボル名	イベント・コード	I A R	D A R	O P C	最大 インクリ メント/ サイクル	説明
DATA_EAR_EVENTS	0xc8	Y	Y	Y	1	L1 データ・キャッシュ EAR イベント
L1D_READS_SET0	0xc2	Y	Y	Y	2	L1 データ・キャッシュ読み出し
DATA_REFERENCES_SET0	0xc3	Y	Y	Y	4	メモリ・パイプラインに発行されたデータ・メモリ参照
L1D_READS_SET1	0xc4	Y	Y	Y	2	L1 データ・キャッシュ読み出し
DATA_REFERENCES_SET1	0xc5	Y	Y	Y	4	メモリ・パイプラインに発行されたデータ・メモリ参照
L1D_READ_MISSES	0xc7	Y	Y	Y	2	L1 データ・キャッシュ読み出しミス

11.8.2.1 L1D キャッシュ・イベント (セット 0)

表 11-11. L1D キャッシュ・セット 0 のパフォーマンス・モニタ

シンボル名	イベント・コード	I A R	D A R	O P C	最大 インクリ メント/ サイクル	説明
L1DTLB_TRANSFER	0xc0	Y	Y	Y	1	L1D_READS でカウントされたアクセスで、L2DTLB でヒットした L1DTLB ミス
L2DTLB_MISSES	0xc1	Y	Y	Y	4	L2DTLB ミス
L1D_READS_SET0	0xc2	Y	Y	Y	2	L1 データ・キャッシュ読み出し
DATA_REFERENCES_SET0	0xc3	Y	Y	Y	4	メモリ・パイプラインに発行されたデータ・メモリ参照

11.8.2.2 L1D キャッシュ・イベント (セット 1)

表 11-12. L1D キャッシュ・セット 1 のパフォーマンス・モニタ

シンボル名	イベント・コード	I A R	D A R	O P C	最大 インクリ メント/ サイクル	説明
L1D_READS_SET1	0xc4	Y	Y	Y	2	L1 データ・キャッシュ読み出し
DATA_REFERENCES_SET1	0xc5	Y	Y	Y	4	メモリ・パイプラインに発行されたデータ・メモリ参照
L1D_READ_MISSES	0xc7	Y	Y	Y	2	L1 データ・キャッシュ読み出しミス

11.8.2.3 L1D キャッシュ・イベント (セット 2)

表 11-13. L1D キャッシュ・セット 2 のパフォーマンス・モニタ

シンボル名	イベント・コード	I A R	D A R	O P C	最大 インクリ メント/ サイクル	説明
BE_L1D_FPU_BUBBLE	0xca	N	N	N	1	FP または L1D キャッシュが原因のメイン・パイプ内のフル・パイプ・バブル

11.8.2.4 L1D キャッシュ・イベント (セット 3)

表 11-14. L1D キャッシュ・セット 3 のパフォーマンス・モニタ

シンボル名	イベント・コード	I A R	D A R	O P C	最大イン クリメン ト/ サイ クル	説明
LOADS_RETIRED	0xcd	Y	Y	Y	4	リタイアしたロード
MISALIGNED_LOADS_RETIRED	0xce	Y	Y	Y	4	リタイアしたアライメントの合っていないロード命令
UC_LOADS_RETIRED	0xcf	Y	Y	Y	4	リタイアしたキャッシュ不可ロード

11.8.2.5 L1D キャッシュ・イベント (セット 4)

表 11-15. L1D キャッシュ・セット 4 のパフォーマンス・モニタ

シンボル名	イベント・コード	I A R	D A R	O P C	最大 インクリ メント/ サイクル	説明
MISALIGNED_STORES_RETIRED	0xd2	Y	Y	Y	2	リタイアしたアライメントの合っていないストア命令
STORES_RETIRED	0xd1	Y	Y	Y	2	リタイアしたストア
UC_STORES_RETIRED	0xd0	Y	Y	Y	2	リタイアしたキャッシュ不可ストア

11.8.3 L2 ユニファイド・キャッシュ・イベント

表 11-16 は、Itanium 2 プロセッサの L2 キャッシュ監視用に提供されているイベントをまとめたものである。

L2 キャッシュ・イベントは、6 つのセットに分割されている。同時に計測できるのは、同じセット内のイベント (または L2 以外のイベント) のみである。各セットは、PMC4 に設定されたイベント・コードによって選択される (すなわち、このセット内のイベントを計測したい場合、そのうちの 1 つは PMD4 で計測しなければならない)。セット内には、PMD4 でしか計測できないイベントもある。また、特定の同類の L2 イベントの umask を 1 次イベント (PMD4 を使用する L2 イベント) が指示するような umask については、いくつかの制限が存在する可能性がある。これらの制限は、セットごとに示される。各セットに属するモニタについては、次の節で詳しく説明する。

表 11-16. L2 ユニファイド・キャッシュ・イベントのパフォーマンス・モニタ

シンボル名	イベント・コード	I A R	D A R	O P C	最大 インクリ メント/ サイクル	説明
L2_BAD_LINES_SELECTED	0xb9	Y	Y	Y	4	無効なラインが使用可能なときに置き換えられた有効なライン
L2_BYPASS	0xb8	Y	Y	Y	1	バイパスをカウントする。
L2_DATA_REFERENCES	0xb2	Y	Y	Y	4	L2 へのデータ読み出し / 書き込みアクセス
L2_FILLB_FULL	0xbf	N	N	N	1	L2D フィル・バッファがいっぱい
L2_FORCE_RECIRC	0xb4	Y	Y	Y	4	強制された再循環
L2_GOT_RECIRC_IFETCH	0xba	Y	Y	Y	1	L2D が受け取った命令フェッチ再循環
L2_GOT_RECIRC_OZQ_ACC	0xb6	Y	Y	Y	1	L1D に再循環された OZQ アクセスの数をカウントする。
L2_IFET_CANCELS	0xa1、 0xa5、 0xa9、 0xad	Y	Y	Y	1	L2 による命令フェッチのキャンセル
L2_ISSUED_RECIRC_IFETCH	0xb9	Y	Y	Y	1	L2D が発行した命令フェッチ再循環
L2_ISSUED_RECIRC_OZQ_ACC	0xb5	Y	Y	Y	1	再循環発行を試行したが、先読みされなかった回数をカウントする。
L2_L3ACCESS_CANCEL	0xb0	Y	Y	Y	1	キャンセルされた L3 アクセス
L2_MISSES	0xcb	Y	Y	Y	1	L2 ミス
L2_OPS_ISSUED	0xb8	Y	Y	Y	4	L2D が発行した別の演算
L2_OZDB_FULL	0xbd	N	N	N	1	L2D OZ データ・バッファがいっぱい

表 11-16. L2 ユニファイド・キャッシュ・イベントのパフォーマンス・モニタ (続き)

シンボル名	イベント・コード	I A R	D A R	O P C	最大 インクリ メント/ サイクル	説明
L2_OZQ_ACQUIRE	0xa2、 0xa6、 0xaa、 0xae	N	N	N	1	L2 OZQ 内に存在する獲得順序属性に関するクロック
L2_OZQ_CANCEL0	0xa0	Y	Y	Y	4	L2 OZQ のキャンセル
L2_OZQ_CANCEL1	0xac	Y	Y	Y	4	L2 OZQ のキャンセル
L2_OZQ_CANCEL2	0xa8	Y	Y	Y	4	L2 OZQ のキャンセル
L2_OZQ_FULL	0xbc	N	N	N	1	L2D OZQ がいっぱい
L2_OZQ_RELEASE	0xa3、 0xa7、 0xab、 0xaf	N	N	N	1	L2 OZQ 内に存在する解放順序属性に関するクロック
L2_REFERENCES	0xb1	Y	Y	Y	4	L2 から発行された要求
L2_STORE_HIT_SHARED	0xba	Y	Y	Y	2	ストアが共用ラインをヒットした。
L2_SYNTH_PROBE	0xb7	Y	Y	Y	1	合成されたプローブ
L2_VICTIMB_FULL	0xbe	N	N	N	1	L2D ビクティム・バッファがいっぱい

表 11-17. L2 ユニファイド・キャッシュ・イベントの派生モニタ

シンボル名	説明	式
L2_DATA_RATIO	L2 に発行されたデータ要求率	L2_DATA_REFERENCES.L2_ALL / L2_REFERENCES
L2_DATA_READS	L2 データ読み出し要求	L2_DATA_REFERENCES.L2_DATA_ READS
L2_DATA_WRITES	L2 データ書き込み要求	L2_DATA_REFERENCES.L2_DATA_ WRITES
L2_INST_REFERENCES	L2 に発行された命令要求	L2_INST_DEMAND_READS - L1_FETCH_ISB_HIT + L2_INST_PREFETCHES
L2_INST_FETCHES	デマンド命令フェッチが原因で L2 に発行された要求	L2_INST_DEMAND_READS + L2_INST_PREFETCHES
L2_MISS_RATIO	L2 ミスのパーセント	L2_MISSES/L2_REFERENCES
L2_RECIRC_ATTEMPTS	L2 発行ロジックが再循環の発行 を試行した回数	L2_ISSUED_RECIRC_OZQ_ACC + L2_OZQ_CANCEL2.DIDNT_RECIRC

L2_MISS_RATIO は、重要な評価基準である。セマフォは L2_REFERENCES で一度カウントされるが、スヌープされバスから再要求されているキャッシュ・ラインが原因で、L2_MISSES が複数発生するときがある。そのため、セマフォにより、この評価基準が 100% より大きくなる場合があることに注意する。これは、順方向の進行が再開するまで、何度も繰り返される可能性がある。L2_MISSES と L2_REFERENCES は 5 サイクル・パイパスのみで遅れずに整列するため、この評価基準では、ある程度のエラーが発生することになる (この問題を回避するには、mf.a 命令に続けて sync.i 命令、srlz.i 命令を使用し、その後でカウンタを読み取ればよい)。

11.8.3.1 L2 キャッシュ・イベント (セット 0)

L2_OZQ_CANCELS* イベント、または L2_IFET_CANCELS は、PMD4 で計測しなければならない。これらのイベントは、同じ umask を使用する。一度に計測できるのは、3 つある L2_OZQ_CANCELS* イベントの 1 つだけである。

表 11-18. L2 キャッシュ・セット 0 のパフォーマンス・モニタ

シンボル名	イベント・コード	I A R	D A R	O P C	最大 インクリ メント/ サイクル	説明
L2_IFET_CANCELS	0xa1、 0xa5、 0xa9、 0xad	Y	Y	Y	1	L2 による命令フェッチのキャンセル
L2_OZQ_ACQUIRE	0xa2、 0xa6、 0xaa、 0xae	N	N	N	1	L2 OZQ 内に存在する獲得順序属性に関するクロック
L2_OZQ_CANCELS0	0xa0	Y	Y	Y	4	L2 OZQ のキャンセル
L2_OZQ_CANCELS1	0xac	Y	Y	Y	4	L2 OZQ のキャンセル
L2_OZQ_CANCELS2	0xa8	Y	Y	Y	4	L2 OZQ のキャンセル
L2_OZQ_RELEASE	0xa3、 0xa7、 0xab、 0xaf	N	N	N	1	L2 OZQ 内に存在する解放順序属性に関するクロック

11.8.3.2 L2 キャッシュ・イベント (セット 1)

L2_L3ACCESS_CANCEL は、PMD4 で計測しなければならない。

表 11-19. L2 キャッシュ・セット 1 のパフォーマンス・モニタ

シンボル名	イベント・コード	I A R	D A R	O P C	最大 インクリ メント/ サイクル	説明
L2_DATA_REFERENCES	0xb2	Y	Y	Y	4	L2 へのデータ読み出し / 書き込みアクセス
L2_L3ACCESS_CANCEL	0xb0	Y	Y	Y	1	キャンセルされた L3 アクセス
L2_REFERENCES	0xb1	Y	Y	Y	4	L2 から発行された要求

11.8.3.3 L2 キャッシュ・イベント (セット 2)

L2_FORCE_RECIRC は、PMD4 で計測しなければならない。

表 11-20. L2 キャッシュ・セット 2 のパフォーマンス・モニタ

シンボル名	イベント・コード	I A R	D A R	O P C	最大 インクリ メント/ サイクル	説明
L2_FORCE_RECIRC	0xb4	Y	Y	Y	4	強制された再循環
L2_ISSUED_RECIRC_OZQ_ACC	0xb5	Y	Y	Y	1	再循環発行を試行したが、先読みされなかった回数をカウントする。

表 11-20. L2 キャッシュ・セット 2 のパフォーマンス・モニタ (続き)

シンボル名	イベント・コード	I A R	D A R	O P C	最大 インクリ メント/ サイクル	説明
L2_GOT_RECIRC_OZQ_ACC	0xb6	Y	Y	Y	1	L1D に再循環された OZQ アクセスの数をカウントする。
L2_SYNTH_PROBE	0xb7	Y	Y	Y	1	合成されたプローブ

11.8.3.4 L2 キャッシュ・イベント (セット 3)

L2_BAD_LINES_SELECTED、L2_BYPASS、L2_STORE_HIT_SHARED は、同じ umask を共有する。

表 11-21. L2 キャッシュ・セット 3 のパフォーマンス・モニタ

シンボル名	イベント・コード	I A R	D A R	O P C	最大 インクリ メント/ サイクル	説明
L2_BAD_LINES_SELECTED	0xb9	Y	Y	Y	4	無効なラインが使用可能ときに置き換えられた有効なライン
L2_BYPASS	0xb8	Y	Y	Y	1	バイパスをカウントする。
L2_STORE_HIT_SHARED	0xba	Y	Y	Y	2	ストアが共有ラインをヒットした。

11.8.3.5 L2 キャッシュ・イベント (セット 4)

L2_OPS_ISSUED、L2_ISSUED_RECIRC_IFETCH、または L2_GOT_RECIRC_IFETCH は、PMD4 で計測しなければならない。これらの 3 つのイベントは、同じ umask を共有する。

表 11-22. L2 キャッシュ・セット 4 のパフォーマンス・モニタ

シンボル名	イベント・コード	I A R	D A R	O P C	最大 インクリ メント/ サイクル	説明
L2_GOT_RECIRC_IFETCH	0xba	Y	Y	Y	1	L2D が受け取った命令フェッチ再循環
L2_ISSUED_RECIRC_IFETCH	0xb9	Y	Y	Y	1	L2D が発行した命令フェッチ再循環
L2_OPS_ISSUED	0xb8	Y	Y	Y	4	L2D が発行した別の演算

11.8.3.6 L2 キャッシュ・イベント (セット 5)

L2_OZQ_FULL、L2_OZDB_FULL、L2_VICTIMB_FULL、または L2_FILLB_FULL は、PMD4 で計測しなければならない。これらの 4 つのイベントは、同じ umask を共有する。

表 11-23. L2 キャッシュ・セット 5 のパフォーマンス・モニタ

シンボル名	イベント・コード	I A R	D A R	O P C	最大 インクリ メント/ サイクル	説明
L2_OZQ_FULL	0xbc	N	N	N	1	L2D OZQ がいっぱい
L2_OZDB_FULL	0xbd	N	N	N	1	L2D OZ データ・バッファがいっぱい
L2_VICTIMB_FULL	0xbe	N	N	N	1	L2D ビクティム・バッファがいっぱい
L2_FILLB_FULL	0xbf	N	N	N	1	L2D フィル・バッファがいっぱい

11.8.4 L3 キャッシュ・イベント

表 11-24 は、直接計測される L3 キャッシュ・イベントについてまとめたものである。派生イベントの詳細は、表 11-25 に示している。

表 11-24. L3 ユニファイド・キャッシュ・イベントのパフォーマンス・モニタ

シンボル名	イベント・コード	I A R	D A R	O P C	最大 インクリ メント/ サイクル	説明
L3_LINES_REPLACED	0xdf	N	N	N	1	置き換えられた L3 キャッシュ・ライン
L3_MISSES	0xdc	Y	Y	Y	1	L3 ミス
L3_READS	0xdd	Y	Y	Y	1	L3 読み出し
L3_REFERENCES	0xdb	Y	Y	Y	1	L3 参照
L3_WRITES	0xde	Y	Y	Y	1	L3 書き込み

表 11-25. L3 ユニファイド・キャッシュ・イベントの派生モニタ

シンボル名	説明	式
L3_DATA_HITS	L3 データ読み出しヒット	L3_READS.DATA_READ.HIT
L3_DATA_MISS_RATIO	L3 データのミス率	$(L3_READS.DATA_READ.MISS + L3_WRITES.DATA_WRITE.MISS) / (L3_READS.DATA_READ.ALL + L3_WRITES.DATA_WRITE.ALL)$
L3_DATA_READ_MISSES	L3 データ読み出しミス	L3_READS.DATA_READ.MISS
L3_DATA_READ_RATIO	データ読み出し参照である L3 参照率	$L3_READS.DATA_READ.ALL / L3_REFERENCES$
L3_DATA_READ_REFERENCES	L3 データ読み出し参照	L3_READS.DATA_READ.ALL
L3_INST_HITS	L3 命令ヒット	L3_READS.INST_FETCH.HIT
L3_INST_MISSES	L3 命令ミス	L3_READS.INST_FETCH.MISS
L3_INST_MISS_RATIO		$L3_READS.INST_FETCH.MISS / L3_READS.INST_FETCH.ALL$
L3_INST_RATIO	命令参照である L3 参照率	$L3_READS.INST_FETCH.ALL / L3_REFERENCES$
L3_INST_REFERENCES	L3 命令参照	L3_READS.INST_FETCH.ALL
L3_MISS_RATIO	L3 ミスのパーセント	$L3_MISSES / L3_REFERENCES$
L3_READ_HITS	L3 読み出しヒット	L3_READS.READS.HIT
L3_READ_MISSES	L3 読み出しミス	L3_READS.READS.MISS
L3_READ_REFERENCES	L3 読み出し参照	L3_READS.READS.ALL
L3_STORE_HITS	L3 ストア・ヒット	L3_WRITES.DATA_WRITE.HIT
L3_STORE_MISSES	L3 ストア・ミス	L3_WRITES.DATA_WRITE.MISS
L3_STORE_REFERENCES	L3 ストア参照	L3_WRITES.DATA_WRITE.ALL
L2_WB_HITS	L2 ライトバック・ヒット	L3_WRITES.L2_WB.HIT
L2_WB_MISSES	L2 ライトバック・ミス	L3_WRITES.L2_WB.MISS
L2_WB_REFERENCES	L2 ライトバック参照	L3_WRITES.L2_WB.ALL
L3_WRITE_HITS	L3 書き込みヒット	L3_WRITES.ALL.HIT
L3_WRITE_MISSES	L3 書き込みミス	L3_WRITES.ALL.MISS
L3_WRITE_REFERENCES	L3 書き込み参照	L3_WRITES.ALL.ALL

11.9 システム・イベント

デバッグ・レジスタ・マッチ・イベントでは、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) 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) をカウントする。

CPU_CPL_CHANGES では、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。

表 11-26. システム・イベントのパフォーマンス・モニタ

シンボル名	イベント・コード	I A R	D A R	O P C	最大 インクリ メント/ サイクル	説明
CPU_CPL_CHANGES	0x13	N	N	N	1	特権レベルの変更
DATA_DEBUG_REGISTER_FAULT	0x52	N	N	N	1	データ・デバッグ・レジスタが原因のフォルト。ロード/ストア命令の一致
DATA_DEBUG_REGISTER_MATCHES	0xc6	Y	Y	Y	1	データ・デバッグ・レジスタが、メモリ参照のデータ・アドレスに一致
EXTERN_DP_PINS_0_TO_3	0x9e	N	N	N	1	アサートされた DP ピン 0 ~ 3
EXTERN_DP_PINS_4_TO_5	0x9f	N	N	N	1	アサートされた DP ピン 4 ~ 5
SERIALIZATION_EVENTS	0x53	N	N	N	1	sriz.l 命令の数

表 11-27. システム・イベントの派生モニタ

シンボル名	説明	式
CODE_DEBUG_REGISTER_MATCHES	コード・デバッグ・レジスタ・マッチ	IA64_TAGGED_INST_RETIRED

11.10 TLB イベント

Itanium 2 プロセッサの命令 TLB とデータ TLB、および仮想ハッシュ・ページ・テーブル・ウォーカは、表 11-28 の各種イベントによって監視される。

L1ITLB_REFERENCES は命令キャッシュ・アクセス・イベントから、L1DTLB_REFERENCES はデータ・キャッシュ・アクセス・イベントから求められる。ITLB_REFERENCES には 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は含まれ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 に目的のものがみつからないとそのプリフェッチはキャンセルされ、VHPT ウォークもソフトウェア TLB ミス・ハンドリングも実行されないからである。ITLB_MISSES_FETCH と L2DTLB_MISSES は、TLB ミスをカウントする。ITLB_INSERTS_HPW と DTLB_INSERTS_HPW は、仮想ハッシュ・ページ・テーブル・ウォーカが命令 / データ TLB へ挿入した回数をカウントする。

表 11-28. TLB イベントのパフォーマンス・モニタ

シンボル名	イベント・コード	I A R	D A R	O P C	最大 インクリ メント/ サイクル	説明
DTLB_INSERTS_HPWW	0xc9	Y	Y	Y	4	ハードウェア・ページ・ウォークによる DTLB への挿入
DTLB_INSERTS_HPWW_RETIRED	0x2c	Y	Y	Y	4	ハードウェア・ページ・ウォークにより DTLB に挿入された VHPT エントリ
HPWW_DATA_REFERENCES	0x2d	Y	Y	Y	4	VHPT へのデータ・メモリ参照
L2DTLB_MISSES	0xc1	Y	Y	Y	4	L2DTLB ミス
L1ITLB_INSERTS_HPWW	0x48	Y	N	N	1	L1ITLB ハードウェア・ページ・ウォークの挿入
ITLB_MISSES_FETCH	0x47	Y	N	N	1	ITLB ミス・デマンド・フェッチ
L1DTLB_TRANSFER	0xc0	Y	Y	Y	1	L1D_READS でカウントされたアクセスで、L2DTLB でヒットした L1DTLB ミス

表 11-29. TLB イベントの派生モニタ

シンボル名	説明	式
L1DTLB_EAR_EVENTS	EAR によって収集された L1DTLB イベントの数をカウントする。	DATA_EAR_EVENTS
L2DTLB_MISS_RATIO	L2DTLB ミス率	$L2DTLB_MISSES / DATA_REFERENCES_SET0$ または $L2DTLB_MISSES / DATA_REFERENCES_SET1$
L1DTLB_REFERENCES	L1DTLB 参照	$DATA_REFERENCES_SET0$ または $DATA_REFERENCES_SET1$
L1ITLB_EAR_EVENTS	EAR によって収集された L1ITLB イベントの数に関する情報を提供する。これは、L1I_EAR_EVENTS のサブセットである。	L1I_EAR_EVENTS
L1ITLB_MISS_RATIO	L1ITLB ミス率	$ITLB_MISSES_FETCH.L1ITLB / L1I_READS$
L1ITLB_REFERENCES	L1ITLB 参照	L1I_READS
L1DTLB_FOR_L1D_MISS_RATIO	L1D を補助する L1DTLB のミス率	$L1DTLB_TRANSFER / L1D_READS_SET0$ または $L1DTLB_TRANSFER / L1D_READS_SET1$

Itanium 2 プロセッサには、L1DTLB および L2DTLB と呼ばれる 2 つのデータ TLB がある（これらはそれぞれ、DTLB、L2 DTLB と呼ばれる）。これらの TLB は並列に配置されるが、L2DTLB のほうが容量が大きく、低速である。これらの TLB のヒットとミスの組み合わせに対するアクションを次に示す。

- L1DTLB_hit=0、L2DTLB_hit=0: イネーブルにされた場合、HPW はキックインし、一方または両方の TLB に変換を挿入する。
- L1DTLB_hit=0、L2DTLB_hit=1: 浮動小数点の場合、アクションは実行されない。それ以外の場合は、L2DTLB から L1DTLB へ転送が実行される。

- L1DTLB_hit=1、L2DTLB_hit=0: イネーブルにされた場合、HPW はキックインし、一方または両方の TLB に変換を挿入する。
- L1DTLB_hit=1、L2DTLB_hit=1: アクションは実行されない。

メモリ操作がメモリ・パイプラインに送られると、DATA_REFERENCES によってカウントされる。L2DTLB に変換が存在しない場合は、L2DTLB_MISSES によってカウントされる。HPW がイネーブルの場合は、HPW_DATA_REFERENCES によってカウントされる。HPW は、VHPT 内にデータを見つけると、そのデータを L1DTLB と (必要に応じて) L2DTLB に挿入する。L2DTLB 内に変換が存在する場合は、L1DTLB 内に変換が存在しない場合に限って処理が実行される。操作が L1D (L1D_READS の説明を参照) によって処理される場合は、L1DTLB_TRANSFER によってカウントされる。TLB ミス率を計算するため、VHPT メモリ参照は DATA_REFERENCES イベントから除外されているが、それを追加したい場合のために VHPT_REFERENCES が用意されている。

TLB ハードウェア設計上の理由により、動作を引き起こしている命令がリタイアメントに達していないにも関わらず (そのようにマークされている)、それらのイベントのいくつかは動作を示すときがいくつかある。そうした場合であってもプロセッサはストールされるため、それらはカウントに含まれる。また、この問題に関して評価基準の計算に使用されるすべてのイベントに矛盾がない限り、非常に正確な数が期待される。

11.11 システム・バス・イベント

表 11-30 は、システム・バス・トランザクション・モニタについてまとめたものである。表に示したバス・イベントの多くは、インシエータによってイベントを制限する umask を使用する。特に断りのない限り、すべてのバス・イベントで「1 サイクル当たり」と表現するときは、バス・クロック・サイクルではなく、CPU クロック・サイクルを意味するものとする。表 11-31 には、多数の派生イベントが含まれている。

表 11-30. システム・バス・イベントのパフォーマンス・モニタ

シンボル名	イベント・コード	I A R	D A R	O P C	最大 インクリ メント/ サイクル	説明
BUS_ALL	0x87	N	N	N	1	バス・トランザクション
BUS_BRQ_LIVE_REQ_HI	0x9c	N	N	N	2	BRQ ライブ要求 (5 ビットの未処理 BRQ 要求カウンットの最上位 2 ビット)
BUS_BRQ_LIVE_REQ_LO	0x9b	N	N	N	7	BRQ ライブ要求 (5 ビットの未処理 BRQ 要求カウンットの最下位 3 ビット)
BUS_BRQ_REQ_INSERTED	0x9d	N	N	N	1	挿入された BRQ 要求
BUS_DATA_CYCLE	0x88	N	N	N	1	バス上の有効なデータ・サイクル
BUS_HITM	0x84	N	N	N	1	バス・ヒット・モディファイド・ライン・トランザクション
BUS_IO	0x90	N	N	N	1	IA-32 互換 IO バス・トランザクション
BUS_IOQ_LIVE_REQ_HI	0x98	N	N	N	2	インオーダー・バス・キュー要求 (4 ビットの未処理 IOQ 要求カウンットの最上位 2 ビット)
BUS_IOQ_LIVE_REQ_LO	0x97	N	N	N	3	インオーダー・バス・キュー要求 (4 ビットの未処理 IOQ 要求カウンットの最下位 2 ビット)
BUS_LOCK	0x93	N	N	N	1	IA-32 互換 IO バス・ロック・トランザクション
BUS_BACKSNP_REQ	0x8e	N	N	N	1	バス・バック・スヌープ要求

表 11-30. システム・バス・イベントのパフォーマンス・モニタ (続き)

シンボル名	イベント・コード	I A R	D A R	O P C	最大 インクリ メント/ サイクル	説明
BUS_MEMORY	0x8a	N	N	N	1	バス・メモリ・トランザクション
BUS_MEM_READ	0x8b	N	N	N	1	フル・キャッシュ・ライン D/I メモリ RD、RD 無効化、BRIL
BUS_MEM_READ_OUT_HI	0x94	N	N	N	2	未処理のメモリ RD トランザクション
BUS_MEM_READ_OUT_LO	0x95	N	N	N	7	未処理のメモリ RD トランザクション
BUS_OOQ_LIVE_REQ_HI	0x9a	N	N	N	2	アウト・オブ・オーダー・バス・キュー要求 (4 ビットの未処理 OOO 要求カウンットの最上位 2 ビット)
BUS_OOQ_LIVE_REQ_LO	0x99	N	N	N	7	アウト・オブ・オーダー・バス・キュー要求 (4 ビットの未処理 OOO 要求カウンットの最下位 3 ビット)
BUS_RD_DATA	0x8c	N	N	N	1	バス読み出しデータ・トランザクション
BUS_RD_HIT	0x80	N	N	N	1	バス読み出しヒット・クリーン・ノンローカル・キャッシュ・トランザクション
BUS_RD_HITM	0x81	N	N	N	1	バス読み出しヒット・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
BUS_RD_INVALID_ALL_HITM	0x83	N	N	N	1	HITM におけるバス BRIL パースト・トランザクションの結果
BUS_RD_INVALID_HITM	0x82	N	N	N	1	HITM におけるバス BIL トランザクションの結果
BUS_RD_IO	0x91	N	N	N	1	IA-32 互換 IO 読み出しトランザクション
BUS_RD_PRTL	0x8d	N	N	N	1	バス読み出しパーシャル・トランザクション
BUS_SNOOPQ_REQ	0x96	N	N	N	7	バス・スヌープ・キュー要求
BUS_SNOOPS	0x86	N	N	N	1	バス・スヌープ・トータル
BUS_SNOOPS_HITM	0x85	N	N	N	1	バス・スヌープ・ヒット・モディファイド・キャッシュ・ライン
BUS_SNOOP_STALL_CYCLES	0x8f	N	N	N	1	バス・スヌープ・ストール・サイクル (任意のエージェントから)
BUS_WR_WB	0x92	N	N	N	1	バス・ライトバック・トランザクション
MEM_READ_CURRENT	0x89	N	N	N	1	バス上の現在のメモリの読み出しトランザクション

表 11-31. システム・バス・イベントの派生モニタ

シンボル名	説明	式
BIL_HITM_LINE_RATIO	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
BIL_RATIO	BIL 率	BUS_RD_INVALID / BUS_MEMORY
BRIL_HITM_LINE_RATIO	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
BUS_ADDR_BPRI	IO エージェントによって使用されるバス・トランザクション	BUS_MEMORY.*.IO
BUS_BRQ_LIVE_REQ	BRQ ライブ要求	BUS_BRQ_LIVE_REQ_HI * 8 + BUS_BRQ_LIVE_REQ_LO
BUS_BURST	フル・キャッシュ・ライン・メモリ・トランザクション (BRL、BRIL、BWL)	BUS_MEMORY.EQ_128BYTE.*
BUS_HITM_RATIO	バス・モディファイド・ライン・ヒット率	BUS_HITM / BUS_MEMORY または BUS_HITM / BUS_BURST
BUS_HITS_RATIO	共用ラインに対するバス読み出しヒット率	BUS_RD_HIT / BUS_RD_ALL または BUS_RD_HIT / BUS_MEMORY
BUS_IOQ_LIVE_REQ	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_IO_CYCLE_RATIO	バス I/O サイクル率	BUS_IO / BUS_ALL
BUS_IO_RD_RATIO	バス I/O 読み出し率	BUS_RD_IO / BUS_IO
BUS_MEM_READ_OUTSTANDING	未処理のメモリ RD トランザクションの数	BUS_MEM_READ_OUT_HI * 8 + BUS_MEM_READ_OUT_LO
BUS_OOQ_LIVE_REQ	アウト・オブ・オーダー・バス・キュー要求	BUS_OOQ_LIVE_REQ_HI * 8 + BUS_OOQ_LIVE_REQ_LO
BUS_PARTIAL	キャッシュ・ライン・メモリ・トランザクション未満 (BRP、BWP)	BUS_MEMORY.LT_128BYTE.*
BUS_PARTIAL_RATIO	バス・パーシャル・アクセス率	BUS_MEMORY.LT_128BYTE / BUS_MEMORY
BUS_RD_ALL	フル・キャッシュ・ライン・メモリ読み出しトランザクション (BRL)	BUS_MEM_READ.BRL.*
BUS_RD_DATA_RATIO	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
BUS_RD_HITM_RATIO	モディファイド・ラインに対するバス読み出しヒット率	BUS_RD_HITM / BUS_RD_ALL または BUS_RD_HITM / BUS_MEMORY
BUS_RD_INSTRUCTIONS	フル・キャッシュ・ライン命令メモリ読み出しトランザクション (BRP)	BUS_RD_ALL - BUS_RD_DATA
BUS_RD_INVALID	0 バイトのメモリ読み出し無効化トランザクション (BIL)	BUS_MEM_READ.BIL.*
BUS_RD_INVALID_BST	フル・キャッシュ・ライン読み出し無効化トランザクション (BRIL)	BUS_MEM_READ.BRIL.*

表 11-31. システム・バス・イベントの派生モニタ (続き)

シンボル名	説明	式
BUS_RD_INVALID_BST_MEMORY	メモリにより満たされたバースト・トランザクション内のバス読み出し無効ライン (BRIL)	BUS_RD_INVALID_BST - BUS_RD_INVALID_BST_HITM
BUS_RD_INVALID_MEMORY	メモリから満たされたバス読み出し無効化ライン・トランザクション (BIL)	BUS_RD_INVALID - BUS_RD_INVALID_HITM
BUS_RD_INVALID_BST_HITM	HITM における、バースト・トランザクション内のバス読み出し無効化ライン (BRIL)	BUS_RD_INVALID_ALL_HITM - BUS_RD_INVALID_HITM
BUS_RD_PRTL_RATIO	バス読み出しパーシャル・アクセス率	BUS_RD_PRTL / BUS_MEMORY
BUS_WB_RATIO	ライトバック率	BUS_WR_WB / BUS_MEMORY または BUS_WR_WB / BUS_BURST
CACHEABLE_READ_RATIO	キャッシュ可能読み出し率	(BUS_RD_ALL + BUS_MEM_READ.BRIL) / BUS_MEMORY

表 11-32 は、本節で Itanium 2 プロセッサのシステム・バス・トランザクション・モニタを説明する際に使用する表記規則と、11.14 節「パフォーマンス監視イベント・リスト」に示す個々のモニタの説明をまとめたものである。

表 11-32. システム・バス・トランザクションの表記規則

名前	説明
BRC	現在のメモリ読み出し (128 バイト・トランザクション)。ステートの変更なしの読み出し
BRL	メモリ読み出し (64 バイト・バースト)。WB メモリからのデータ・ロードおよびコード・フェッチを含む。
BRIL	メモリ読み出しおよび無効化 (64 バイト・バースト)。RFO (read for ownership) とも呼ばれる。
BIL	メモリ読み出しおよび無効化 (0 バイト・サイズ・トランザクション)。フラッシュ・キャッシュ (fc) 命令によってのみ実行される。
BWL	メモリ書き込み (64 バイト・バースト)。明示的なライトバック / コアレスリング・ライト
BRP	パーシャル・メモリ読み出し (<64 バイト・トランザクション)。通常は、キャッシュ不可読み出し
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、キャッシュ不可書き込み
IORD	パーシャル IO 読み出し (<64 バイト・トランザクション)。IO ポート・スペースに対するキャッシュ不可読み出し
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・スペースに対するキャッシュ不可書き込み

表 11-32 に示されていないその他のトランザクションには、遅延応答、特別トランザクション、割り込み、割り込み通知、ページ TC などがある。プライオリティ・エージェントからの再試行応答をトランザクションが取得した場合はモニタがカウントすることに注意する。

マルチプロセッサ・システム内のスヌープ・トラフィックの分析をサポートするため、Itanium 2 プロセッサは、ローカル・プロセッサとリモート応答モニタを備えている。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、インバウンド・スヌープ・トラフィックを監視する。リモート応答イベント (BUS_RD_HIT、BUS_RD_HITM、BUS_RD_INVALID_HITM、BUS_RD_INVALID_ALL_HITM) は、監視しているプロセッサが発行したバス・トランザクションに対する、その他のプロセッサのスヌープ応答を監視

する。表 11-33 は、バス・トランザクションによるリモート・スヌープ・イベントをまとめたものである。

表 11-33. スヌープ応答によるバス・イベント

リモート・プロセッサ応答	BRL	BIL	BRIL
HIT	BUS_RD_HIT	NA	NA
HITM	BUS_RD_HITM	BUS_RD_INVAL_HITM	BUS_RD_INVAL_BST_HITM
ALL	BUS_RD_ALL	BUS_RD_INVAL	BUS_RD_INVAL

11.12 RSE イベント

表 11-34 は、レジスタ・スタック・エンジン・イベントをまとめたものである。Itanium 2 プロセッサには物理レジスタが 96 個あるため、現在のレジスタとダーティなレジスタの数は、3 つのモニタ間で分配される。

表 11-34. RSE イベントのパフォーマンス・モニタ

シンボル名	イベント・コード	I A R	D A R	O P C	最大 インクリ メント/ サイクル	説明
RSE_CURRENT_REGS_2_TO_0	0x2b	N	N	N	7	現在の RSE レジスタ
RSE_CURRENT_REGS_5_TO_3	0x2a	N	N	N	7	現在の RSE レジスタ
RSE_CURRENT_REGS_6	0x26	N	N	N	1	現在の RSE レジスタ
RSE_DIRTY_REGS_2_TO_0	0x29	N	N	N	7	ダーティな RSE レジスタ
RSE_DIRTY_REGS_5_TO_3	0x28	N	N	N	7	ダーティな RSE レジスタ
RSE_DIRTY_REGS_6	0x24	N	N	N	1	ダーティな RSE レジスタ
RSE_EVENT_RETIRED	0x32	N	N	N	1	リタイアした RSE による操作
RSE_REFERENCES_RETIRED	0x20	Y	Y	Y	2	RSE アクセス

表 11-35. RSE イベントの派生モニタ

シンボル名	説明	式
RSE_CURRENT_REGS	RSE_EVENT_RETIRED が発生する前の現在の RSE レジスタ	$RSE_CURRENT_REGS_6 * 64 + RSE_CURRENT_REGS_5_TO_3 * 8 + RSE_CURRENT_REGS_2_TO_0$
RSE_DIRTY_REGS	RSE_EVENT_RETIRED が発生する前のダーティな RSE レジスタ	$RSE_DIRTY_REGS_6 * 64 + RSE_DIRTY_REGS_5_TO_3 * 8 + RSE_DIRTY_REGS_2_TO_0$
RSE_LOAD_LATENCY_PENALTY	リタイアした RSE ロードが原因でストールしたサイクル数をカウントする (RSE.BOF が RSE.storereg に到達し、フィルに必要なロードのすべてを RSE が発行していないとき)。	BE_RSE_BUBBLE.UNDERFLOW
RSE_AVG_LOAD_LATENCY	RSE ロードの平均レイテンシ	$RSE_LOAD_LATENCY_PENALTY / RSE_REFERENCES_RETIRED.LOAD$
RSE_AVG_CURRENT_REGS	現在のレジスタの平均個数	$RSE_CURRENT_REGS / RSE_EVENT_RETIRED$

表 11-35. RSE イベントの派生モニタ (続き)

シンボル名	説明	式
RSE_AVG_DIRTY_REGS	ダーティなレジスタの平均個数	$RSE_DIRTY_REGS / RSE_EVENT_RETIRED$
RSE_AVG_INVALID_REGS	無効なレジスタの平均個数。クリーンなレジスタの数が常にゼロであると想定する。	$96 - (RSE_DIRTY_REGS + RSE_CURRENT_REGS) / RSE_EVENT_RETIRED$

11.13 イベント・コードによって並べ替えられるパフォーマンス・モニタ

表 11-36 は、イベント・コードによって並べ替えられる Itanium 2 プロセッサのパフォーマンス・モニタをすべてまとめたものである。

表 11-36. コードによって並べ替えられるすべてのパフォーマンス・モニタ

シンボル名	イベント・コード	I A R	D A R	O P C	最大 インクリ メント/ サイクル	説明
BACK_END_BUBBLE	0x00	N	N	N	1	メイン・パイプ内のフル・パイプ・バブル
BE_RSE_BUBBLE	0x01	N	N	N	1	RSE ストールが原因のメイン・パイプ内のフル・パイプ・バブル
BE_EXE_BUBBLE	0x02	N	N	N	1	実行ユニット・ストールが原因のメイン・パイプ内のフル・パイプ・バブル
FP_TRUE_SIRSTALL	0x03	Y	N	N	1	アサートされトラップに至る SIR ストール
BE_FLUSH_BUBBLE	0x04	N	N	N	1	フラッシュが原因のメイン・パイプ内のフル・パイプ・バブル
FP_FALSE_SIRSTALL	0x05	Y	N	N	1	トラップなしの SIR ストール
FP_FAILED_FCHKF	0x06	Y	N	N	1	失敗した fchkf
IA32_ISA_TRANSITIONS	0x07	N	N	N	1	Itanium アーキテクチャと IA-32 ISA との間の遷移
IA64_INST_RETIRED	0x08	Y	N	Y	6	リタイアした Itanium 命令
IA64_TAGGED_INST_RETIRED	0x08	Y	N	Y	6	リタイアしたタグ付き命令
FP_OPS_RETIRED	0x09	Y	N	N	4	リタイアした FP 演算
FP_FLUSH_TO_ZERO	0x0b	Y	N	N	2	ゼロ・フラッシュされた FP の結果
BRANCH_EVENT	0x11	Y	N	Y	1	収集された分岐イベント
CPU_CYCLES	0x12	N	N	N	1	CPU サイクル
CPU_CPL_CHANGES	0x13	N	N	N	1	特権レベルの変更
RSE_REFERENCES_RETIRED	0x20	Y	Y	Y	2	RSE アクセス
RSE_DIRTY_REGS_6	0x24	N	N	N	1	ダーティな RSE レジスタ
RSE_CURRENT_REGS_6	0x26	N	N	N	1	現在の RSE レジスタ
RSE_DIRTY_REGS_5_TO_3	0x28	N	N	N	7	ダーティな RSE レジスタ
RSE_DIRTY_REGS_2_TO_0	0x29	N	N	N	7	ダーティな RSE レジスタ
RSE_CURRENT_REGS_5_TO_3	0x2a	N	N	N	7	現在の RSE レジスタ
RSE_CURRENT_REGS_2_TO_0	0x2b	N	N	N	7	現在の RSE レジスタ

表 11-36. コードによって並べ替えられるすべてのパフォーマンス・モニタ (続き)

シンボル名	イベント・コード	I A R	D A R	O P C	最大 インクリ メント/ サイクル	説明
DTLB_INSERTS_HPW_RETIRED	0x2c	Y	Y	Y	4	HW PW により DTLB に挿入された VHPT エントリ
HPW_DATA_REFERENCES	0x2d	Y	Y	Y	4	VHPT へのデータ・メモリ参照
RSE_EVENT_RETIRED	0x32	N	N	N	1	リタイアした RSE による操作
L11_READS	0x40	Y	N	N	1	L1 命令キャッシュ読み出し
L11_FILLS	0x41	Y	N	N	1	L1 命令キャッシュ・フィル
L2_INST_DEMAND_READS	0x42	Y	N	N	1	L1 命令キャッシュおよび ISB ミス
L11_EAR_EVENTS	0x43	Y	N	N	1	命令 EAR イベント
L11_PREFETCHES	0x44	Y	N	N	1	L1 命令プリフェッチ要求
L2_INST_PREFETCHES	0x45	Y	N	N	1	L2 命令プリフェッチ要求
ISB_BUNPAIRS_IN	0x46	Y	N	N	1	L2 から FE に書き込まれたバンドル・ペア
ITLB_MISSES_FETCH	0x47	Y	N	N	1	ITLB ミス・デマンド・フェッチ
L1ITLB_INSERTS_HPW	0x48	Y	N	N	1	L1ITLB ハードウェア・ページ・ウォーカの挿入
DISP_STALLED	0x49	N	N	N	1	ディスパースルがストールされたサイクル数
L11_SNOOP	0x4a	Y	Y	Y	1	L1 命令によって処理されるスヌープ要求
L11_PURGE	0x4b	Y	N	N	1	L1 命令によって処理される L1ITLB パージ
INST_DISPERSED	0x4d	Y	N	N	6	REN から REG ステージに配布されたシラブル数
SYLL_NOT_DISPERSED	0x4e	Y	N	N	5	配布されなかったシラブル数
SYLL_OVERCOUNT	0x4f	Y	N	N	2	オーバーカウントされたシラブル数
NOPS_RETIRED	0x50	Y	N	Y	6	リタイアした NOP 命令
PREDICATE_SQUASHED_RETIRED	0x51	Y	N	Y	6	プレディケート・オフのために実行されなかった命令
DATA_DEBUG_REGISTER_FAULT	0x52	N	N	N	1	データ・デバッグ・レジスタが原因のフォルト。ロード/ストア命令の一致
SERIALIZATION_EVENTS	0x53	N	N	N	1	srlz.1 命令の数
BR_PATH_PRED	0x54	Y	N	Y	3	FE 分岐パス予測の詳細
INST_FAILED_CHKS_RETIRED	0x55	N	N	N	1	失敗したスペキュレーティブ・チェック・ロード
INST_CHKA_LDC_ALAT	0x56	Y	Y	Y	2	アドバンスト・チェック・ロード
INST_FAILED_CHKA_LDC_ALAT	0x57	Y	Y	Y	1	失敗したアドバンスト・チェック・ロード
ALAT_CAPACITY_MISS	0x58	Y	Y	Y	2	置き換えられた ALAT エントリ
IA32_INST_RETIRED	0x59	N	N	N	2	リタイアした IA-32 命令
BR_MISPRED_DETAIL	0x5b	Y	N	Y	3	FE 分岐予測ミスの詳細
L11_STRM_PREFETCHES	0x5f	Y	N	N	1	L1 命令キャッシュ・ライン・プリフェッチ要求
L11_RAB_FULL	0x60	N	N	N	1	RAB はいっぱいか？

表 11-36. コードによって並べ替えられるすべてのパフォーマンス・モニタ (続き)

シンボル名	イベント・コード	I A R	D A R	O P C	最大 インクリ メント/ サイクル	説明
BE_BR_MISPRED_DETAIL	0x61	Y	N	Y	1	BE 分岐予測ミスの詳細
ENCBR_MISPRED_DETAIL	0x63	Y	N	Y	1	リタイアしたエンコード済み分岐の数
L1I_RAB_ALMOST_FULL	0x64	N	N	N	1	RAB はほぼいっぱいか?
L1I_FETCH_RAB_HIT	0x65	Y	N	N	1	RAB 内でヒットしている命令フェッチ
L1I_FETCH_ISB_HIT	0x66	Y	N	N	1	ISB でヒットし ISB からバイパスされている「ジャストインタイム」命令フェッチ
L1I_PREFETCH_STALL	0x67	N	N	N	1	プリフェッチ・パイプラインがストールした理由
BR_MISPRED_DETAIL2	0x68	Y	N	Y	2	FE 分岐予測ミスの詳細 (不明なバス・コンポーネント)
L1I_PVAB_OVERFLOW	0x69	N	N	N	1	PVAB オーバーフロー
BR_PATH_PRED2	0x6a	Y	N	Y	2	FE 分岐バス予測の詳細 (不明な予測コンポーネント)
FE_LOST_BW	0x70	N	N	N	2	IB への入り口での無効なバンドル
FE_BUBBLE	0x71	N	N	N	1	FE によって検出されたバブル
BE_LOST_BW_DUE_TO_FE	0x72	N	N	N	2	BE がその他の理由でストールされていない場合の無効なバンドル
IDEAL_BE_LOST_BW_DUE_TO_FE	0x73	N	N	N	2	IB からの出口での無効なバンドル
BUS_RD_HIT	0x80	N	N	N	1	バス読み出しヒット・クリーン・ノンローカル・キャッシュ・トランザクション
BUS_RD_HITM	0x81	N	N	N	1	バス読み出しヒット・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
BUS_RD_INVALID_HITM	0x82	N	N	N	1	HITM におけるバス BIL トランザクションの結果
BUS_RD_INVALID_ALL_HITM	0x83	N	N	N	1	HITM におけるバス BIL または BRIL トランザクションの結果
BUS_HITM	0x84	N	N	N	1	バス・ヒット・モディファイド・ライン・トランザクション
BUS_SNOOPS_HITM	0x85	N	N	N	1	バス・スヌープ・ヒット・モディファイド・キャッシュ・ライン
BUS_SNOOPS	0x86	N	N	N	1	バス・スヌープ・トータル
BUS_ALL	0x87	N	N	N	1	バス・トランザクション
BUS_DATA_CYCLE	0x88	N	N	N	1	バス上の有効なデータ・サイクル
MEM_READ_CURRENT	0x89	N	N	N	1	バス上の現在のメモリの読み出しトランザクション
BUS_MEMORY	0x8a	N	N	N	1	バス・メモリ・トランザクション
BUS_MEM_READ	0x8b	N	N	N	1	フル・キャッシュ・ライン D/I メモリ RD、RD 無効化、および BRIL
BUS_RD_DATA	0x8c	N	N	N	1	バス読み出しデータ・トランザクション
BUS_RD_PRTL	0x8d	N	N	N	1	バス読み出しパーシャル・トランザクション

表 11-36. コードによって並べ替えられるすべてのパフォーマンス・モニタ (続き)

シンボル名	イベント・コード	I A R	D A R	O P C	最大 インクリ メント/ サイクル	説明
BUS_BACKSNP_REQ	0x8e	N	N	N	1	バス・バック・スヌープ要求
BUS_SNOOP_STALL_CYCLES	0x8f	N	N	N	1	バス・スヌープ・ストール・サイクル (任意のエージェントから)
BUS_IO	0x90	N	N	N	1	IA-32 互換 IO バス・トランザクション
BUS_RD_IO	0x91	N	N	N	1	IA-32 互換 IO 読み出しトランザクション
BUS_WR_WB	0x92	N	N	N	1	バス・ライトバック・トランザクション
BUS_LOCK	0x93	N	N	N	1	IA-32 互換 IO バス・ロック・トランザクション
BUS_MEM_READ_OUT_HI	0x94	N	N	N	2	未処理のメモリ RD トランザクション
BUS_MEM_READ_OUT_LO	0x95	N	N	N	7	未処理のメモリ RD トランザクション
BUS_SNOOPQ_REQ	0x96	N	N	N	7	バス・スヌープ・キュー要求
BUS_IOQ_LIVE_REQ_LO	0x97	N	N	N	3	インオーダー・バス・キュー要求 (4 ビットの未処理 IOQ 要求カウンットの最下位 2 ビット)
BUS_IOQ_LIVE_REQ_HI	0x98	N	N	N	2	インオーダー・バス・キュー要求 (4 ビットの未処理 IOQ 要求カウンットの最上位 2 ビット)
BUS_OOQ_LIVE_REQ_LO	0x99	N	N	N	7	アウト・オブ・オーダー・バス・キュー要求 (4 ビットの未処理 OOO 要求カウンットの最下位 3 ビット)
BUS_OOQ_LIVE_REQ_HI	0x9a	N	N	N	2	アウト・オブ・オーダー・バス・キュー要求 (4 ビットの未処理 OOO 要求カウンットの最上位 2 ビット)
BUS_BRQ_LIVE_REQ_LO	0x9b	N	N	N	7	BRQ ライブ要求 (5 ビットの未処理 BRQ 要求カウンットの最下位 3 ビット)
BUS_BRQ_LIVE_REQ_HI	0x9c	N	N	N	2	BRQ ライブ要求 (5 ビットの未処理 BRQ 要求カウンットの最上位 2 ビット)
BUS_BRQ_REQ_INSERTED	0x9d	N	N	N	1	挿入された BRQ 要求
EXTERN_DP_PINS_0_TO_3	0x9e	N	N	N	1	アサートされた DP ピン 0 ~ 3
EXTERN_DP_PINS_4_TO_5	0x9f	N	N	N	1	アサートされた DP ピン 4 ~ 5
L2_OZQ_CANCEL0	0xa0	Y	Y	Y	4	L2 OZQ のキャンセル
L2_IFET_CANCEL0	0xa1、 0xa5、 0xa9、 0xad	Y	Y	Y	1	L2 による命令フェッチのキャンセル
L2_OZQ_ACQUIRE	0xa2、 0xa6、 0xaa、 0xae	N	N	N	1	L2 OZQ 内に存在する獲得順序属性に関するクロック
L2_OZQ_RELEASE	0xa3、 0xa7、 0xab、 0xaf	N	N	N	1	L2 OZQ 内に存在する解放順序属性に関するクロック

表 11-36. コードによって並べ替えられるすべてのパフォーマンス・モニタ (続き)

シンボル名	イベント・コード	I A R	D A R	O P C	最大 インクリ メント/ サイクル	説明
L2_OZQ_CANCELS2	0xa8	Y	Y	Y	4	L2 OZQ のキャンセル
L2_OZQ_CANCELS1	0xac	Y	Y	Y	4	L2 OZQ のキャンセル
L2_L3ACCESS_CANCEL	0xb0	Y	Y	Y	1	キャンセルされた L3 アクセス
L2_REFERENCES	0xb1	Y	Y	Y	4	L2 から発行された要求
L2_DATA_REFERENCES	0xb2	Y	Y	Y	4	L2 へのデータ読み出し / 書き込みアクセス
L2_FORCE_RECIRC	0xb4	Y	Y	Y	4	強制された再循環
L2_ISSUED_RECIRC_OZQ_ACC	0xb5	Y	Y	Y	1	再循環発行を試行したが、先読みされなかった回数をカウントする。
L2_GOT_RECIRC_OZQ_ACC	0xb6	Y	Y	Y	1	L1D に再循環された OZQ アクセスの数をカウントする。
L2_SYNTH_PROBE	0xb7	Y	Y	Y	1	合成されたプローブ
L2_BYPASS	0xb8	Y	Y	Y	1	バイパスをカウントする。
L2_OPS_ISSUED	0xb8	Y	Y	Y	4	L2D が発行した別の演算
L2_ISSUED_RECIRC_IFETCH	0xb9	Y	Y	Y	1	L2D が発行した命令フェッチ再循環
L2_BAD_LINES_SELECTED	0xb9	Y	Y	Y	4	無効なラインが使用可能なときに置き換えられた有効なライン
L2_GOT_RECIRC_IFETCH	0xba	Y	Y	Y	1	L2D が受け取った命令フェッチ再循環
L2_STORE_HIT_SHARED	0xba	Y	Y	Y	2	ストアが共用ラインをヒットした。
TAGGED_L2_DATA_RETURN_PORT	0xbb	Y	Y	Y	1	タグ付き L2 データ・リターン・ポート 0/1
L2_OZQ_FULL	0xbc	N	N	N	1	L2D OZQ がいっぱい
L2_OZDB_FULL	0xbd	N	N	N	1	L2D OZ データ・バッファがいっぱい
L2_VICTIMB_FULL	0xbe	N	N	N	1	L2D ビクティム・バッファがいっぱい
L2_FILLB_FULL	0xbf	N	N	N	1	L2D フィル・バッファがいっぱい
L1DTLB_TRANSFER	0xc0	Y	Y	Y	1	L1D_READS でカウントされたアクセスにおいて、L2DTLB でヒットした L1DTLB ミス
L2DTLB_MISSES	0xc1	Y	Y	Y	4	L2DTLB ミス
L1D_READS_SET0	0xc2	Y	Y	Y	2	L1 データ・キャッシュ読み出し
DATA_REFERENCES_SET0	0xc3	Y	Y	Y	4	メモリ・パイプラインに発行されたデータ・メモリ参照
L1D_READS_SET1	0xc4	Y	Y	Y	2	L1 データ・キャッシュ読み出し
DATA_REFERENCES_SET1	0xc5	Y	Y	Y	4	メモリ・パイプラインに発行されたデータ・メモリ参照
DATA_DEBUG_REGISTER_MATCHES	0xc6	Y	Y	Y	1	データ・デバッグ・レジスタが、メモリ参照のデータ・アドレスに一致
L1D_READ_MISSES	0xc7	Y	Y	Y	2	L1 データ・キャッシュ読み出しミス
DATA_EAR_EVENTS	0xc8	Y	Y	Y	1	L1 データ・キャッシュ EAR イベント

表 11-36. コードによって並べ替えられるすべてのパフォーマンス・モニタ (続き)

シンボル名	イベント・コード	I A R	D A R	O P C	最大 インクリ メント/ サイクル	説明
DTLB_INSERTS_HPW	0xc9	Y	Y	Y	4	ハードウェア・ページ・ウォークによる DTLB への挿入
BE_L1D_FPU_BUBBLE	0xca	N	N	N	1	FP または L1D キャッシュが原因のメイン・パイプ内のフル・パイプ・バブル
L2_MISSES	0xcb	Y	Y	Y	1	L2 ミス
LOADS_RETIRED	0xcd	Y	Y	Y	4	リタイアしたロード
MISALIGNED_LOADS_RETIRED	0xce	Y	Y	Y	4	リタイアしたアライメントの合わないロード命令
UC_LOADS_RETIRED	0xcf	Y	Y	Y	4	リタイアしたキャッシュ不可ロード
UC_STORES_RETIRED	0xd0	Y	Y	Y	2	リタイアしたキャッシュ不可ストア
STORES_RETIRED	0xd1	Y	Y	Y	2	リタイアしたストア
MISALIGNED_STORES_RETIRED	0xd2	Y	Y	Y	2	リタイアしたアライメントの合わないストア命令
L3_REFERENCES	0xdb	Y	Y	Y	1	L3 参照
L3_MISSES	0xdc	Y	Y	Y	1	L3 ミス
L3_READS	0xdd	Y	Y	Y	1	L3 読み出し
L3_WRITES	0xde	Y	Y	Y	1	L3 書き込み
L3_LINES_REPLACED	0xdf	N	N	N	1	置き換えられた L3 キャッシュ・ライン

11.14 パフォーマンス監視イベント・リスト

この節では、Itanium 2 プロセッサのパフォーマンス監視イベントを列挙する。

注: 特に断りのない限り、命令アドレス範囲で制限できるイベントは、IBRP0 でしか制限できない。

ALAT_CAPACITY_MISS

- タイトル: 置き換えられた ALAT エントリ
- カテゴリ: 命令実行 IAR/DAR/OPC: Y/Y/Y
- イベント・コード: 0x58、最大 Inc/Cyc: 2
- 定義: アドバンスト・ロード (ld.a、ld.as、ldfp.a、または ldfp.as)、または使用されなくなった ld.c.nc が、同じレジスタ id を持たない ALAT 内の有効なエントリを強制的に追い出した回数、あるいは最後の 1 ~ 2 つの無効なエントリを置き換えた回数について情報を提供する。

表 11-37. ALAT_CAPACITY_MISS のユニット・マスク

拡張子	PMC.umask [19:16]	説明
---	bxx00	(* 何もカウントされない *)
INT	bxx01	整数命令のみ
FP	bxx10	浮動小数点命令のみ
ALL	bxx11	整数命令と浮動小数点命令の両方

BACK_END_BUBBLE

- タイトル: メイン・パイプ内のフル・パイプ・バブル
- カテゴリ: ストール・イベント IAR/DAR/OPC: N/N/N
- イベント・コード: 0x00、最大 Inc/Cyc: 1
- 定義: 5 つのイベント (FPU/L1D、RSE、EXE、分岐 / 例外、またはフロントエンド) のいずれかが原因でストールしたメイン・パイプ内のフル・パイプ・バブルの数をカウントする。このイベントは、1 つのイベント・ユニット・マスクによりさらに制限される。また、4 つのカウンタを使用してすべての情報を容易に収集するために、いくつかの詳細情報が提供される。

表 11-38. BACK_END_BUBBLE のユニット・マスク

拡張子	PMC.umask [19:16]	説明
ALL	bxx00	例外 / 分岐の予測ミスが原因である、フロントエンド、RSE、EXE、FPU/L1D ストール、またはパイプライン・フラッシュ
FE	bxx01	フロントエンド
L1D_FPU_RSE	bxx10	
---	bxx11	(* 何もカウントされない *)

BE_BR_MISPRED_DETAIL

- タイトル: バックエンド分岐予測ミスの詳細
- カテゴリ: 分岐イベント IAR/DAR/OPC: Y/N/Y
- イベント・コード: 0x61、最大 Inc/Cyc: 1
- 定義: 予測結果、stg、rot、または pfs のバックエンド予測ミスに基づいて、リタイアした分岐の数をカウントする。これらの予測は、分岐当たりではなく、バンドル当たりのものである。
- 注: これらのイベントがカウントされるのは、分岐に関連するパス予測ミスがない場合に限られる。パス予測ミスは、stg/rot/pfs の予測ミスを保証するからである。

表 11-39. BE_BR_MISPREDICT_DETAIL のユニット・マスク

拡張子	PMC.umask [19:16]	説明
ANY	bxx00	任意のバックエンド予測ミス
STG	bxx01	バックエンド・ステージ予測ミスのみ
ROT	bxx10	バックエンド・ローテート予測ミスのみ
PFS	bxx11	実行された分岐のバックエンド pfs 予測ミスのみ

BE_EXE_BUBBLE

- タイトル: 実行ユニット・ストールが原因のメイン・パイプ内のフル・パイプ・バブル
- カテゴリ: ストール・イベント IAR/DAR/OPC: N/N/N
- イベント・コード: 0x02、最大 Inc/Cyc: 1
- 定義: 実行ユニットにより引き起こされたストールが原因である、メイン・パイプ内のフル・パイプ・バブルの数をカウントする。
- 注: このイベントの別の原因には優先順位が付けられない。なぜなら、その必要がないからである (原因は独立しており、そのいくつかは同時に発生する。原因はすべてカウントする)。

表 11-40. BE_EXE_BUBBLE のユニット・マスク

拡張子	PMC.umask [19:16]	説明
ALL	b0000	exe によってストールされた。
GRALL	b0001	GR/GR または GR/ ロードの依存関係が原因で、バックエンドが exe によってストールされた。
FRALL	b0010	FR/FR または FR/ ロードの依存関係が原因で、バックエンドが exe によってストールされた。
PR	b0011	PR の依存関係が原因で、バックエンドが exe によってストールされた。
ARCR	b0100	AR または CR の依存関係が原因で、バックエンドが exe によってストールされた。
GRGR	b0101	GR/GR の依存関係が原因で、バックエンドが exe によってストールされた。
CANCEL	b0110	キャンセルされたロードが原因で、バックエンドが exe によってストールされた。
BANK_SWITCH	b0111	バンク切り替えが原因で、バックエンドが exe によってストールされた。

表 11-40. BE_EXE_BUBBLE のユニット・マスク (続き)

拡張子	PMC.umask [19:16]	説明
ARCR_PR_CANCEL_BANK	b1000	ARCR、PR、CANCEL、または BANK_SWITCH
---	b1001-b1111	(* 何もカウントされない *)

BE_FLUSH_BUBBLE

- タイトル: フラッシュが原因のメイン・パイプ内のフル・パイプ・バブル
- カテゴリ: ストール・イベント IAR/DAR/OPC: N/N/N
- イベント・コード: 0x04、最大 Inc/Cyc: 1
- 定義: フラッシュが原因であるメイン・パイプ内のフル・パイプ・バブルの数をカウントする。
- 注: XPN は、BRU よりも優先順位が高い。

表 11-41. BE_FLUSH_BUBBLE のユニット・マスク

拡張子	PMC.umask [19:16]	説明
ALL	bxx00	例外 / 割り込みまたは分岐予測ミス・フラッシュが原因で、バックエンドがストールされた。
BRU	bxx01	分岐予測ミス・フラッシュが原因で、バックエンドがストールされた。
XPN	bxx10	例外 / 割り込みフラッシュが原因で、バックエンドがストールされた。
---	bxx11	(* 何もカウントされない *)

BE_L1D_FPU_BUBBLE

- タイトル: FP または L1D キャッシュが原因のメイン・パイプ内のフル・パイプ・バブル
- カテゴリ: ストール・イベント / L1D キャッシュ・セット 2 IAR/DAR/OPC: N/N/N
- イベント・コード: 0xca、最大 Inc/Cyc: 1
- 定義: 浮動小数点ユニットまたは L1D キャッシュにより引き起こされたストールが原因である、メイン・パイプ内のフル・パイプ・バブルの数をカウントする。
- 注: 制限付きのセット 2 L1D キャッシュ・イベントである。このイベントを計測するには、このセット内のイベントの 1 つを PMD5 で計測しなければならない。このイベントの別の原因には優先順位が付けられない。なぜなら、その必要がないからである (原因は独立しており、そのいくつかは同時に発生する。原因はすべてカウントする)。

表 11-42. BE_L1D_FPU_BUBBLE のユニット・マスク

拡張子	PMC.umask [19:16]	説明
ALL	b0000	L1D または FPU によって、バックエンドがストールされた。
FPU	b0001	FPU によって、バックエンドがストールされた。
L1D	b0010	L1D によって、バックエンドがストールされた。これには、L1 パイプラインによって引き起こされたストールがすべて含まれる (これは、メイン・パイプの DET ステージに相当する、L1 パイプラインの L1D ステージで生成される)。

表 11-42. BE_L1D_FPU_BUBBLE のユニット・マスク (続き)

拡張子	PMC.umask [19:16]	説明
L1D_FULLSTBUF	b0011	ストア・バッファがいっぱいであるために、バックエンドが L1D によってストールされた。
L1D_DCURECIR	b0100	DCU 再循環が原因で、バックエンドが L1D によってストールされた。
L1D_HPWF	b0101	ハードウェア・ページ・ウォークが原因で、バックエンドが L1D によってストールされた。
---	b0110	(* カウントは未定義 *)
L1D_FILLCONF	b0111	リターン・フィルと競合するストアが原因で、バックエンドが L1D によってストールされた。
L1D_DCS	b1000	ストールを要求している dcs が原因で、バックエンドが L1D によってストールされた。
L1D_L2BPRESS	b1001	L2 バック・プレッシャが原因で、バックエンドが L1D によってストールされた。
L1D_TLB	b1010	L2DTLB から L1DTLB への転送が原因で、バックエンドが L1D によってストールされた。
L1D_LDCONF	b1011	アーキテクチャ上の並べ替えの競合が原因で、バックエンドが L1D によってストールされた。
L1D_LDCHK	b1100	ロード・チェックの並べ替えの競合が原因で、バックエンドが L1D によってストールされた。
L1D_NAT	b1101	再循環 NaT 生成が必要な L1D データ・リターンが原因で、バックエンドが L1D によってストールされた。
L1D_STBUFRECIR	b1110	再循環が必要なストア・バッファ・キャンセルが原因で、バックエンドが L1D によってストールされた。
L1D_NATCONF	b1111	unat に書き込まれていない st8.spill と ld8.fill の競合が原因で、バックエンドが L1D によってストールされた。

BE_LOST_BW_DUE_TO_FE

- タイトル: BE がその他の理由でストールされていない場合の無効なバンドル
- カテゴリ: ストール・イベント IAR/DAR/OPC: N/N/N
- イベント・コード: 0x72、最大 Inc/Cyc: 2
- 定義: バックエンドがその他の理由でストールしていない場合に限って、命令バッファからの無効なバンドルの数を出口でカウントする。
- 注: このイベントで、帯域幅が失われた原因には、次の順序で優先順位が付けられる: FEFLUSH、TLBMISS、IMISS、PLP、BR_ILOCK、BRQ、BI、FILL_RECIRC、BUBBLE、IBFULL、UNREACHED。この優先順位付けでは、いくつかのストール条件が同時に存在する場合、最も優先順位の高いものだけがカウントされる。バンドルが「到達不可」と見なされる場合は 2 つある。実行された分岐がバンドル 0 に含まれる場合、またはバンドル 0 は無効であるが IP[4] に 1 が設定されている場合、バンドル 1 は到達されない。

表 11-43. BE_LOST_BW_DUE_TO_FE のユニット・マスク

拡張子	PMC.umask [19:16]	説明
ALL	b0000	原因に関わらずカウントする。
FEFLUSH	b0001	フロントエンド・フラッシュが発生原因である場合のみ
---	b0010	(* カウントは未定義 *)
---	b0011	(* 不正な選択 *)

表 11-43. BE_LOST_BW_DUE_TO_FE のユニット・マスク (続き)

拡張子	PMC.umask [19:16]	説明
UNREACHED	b0100	到達不可バンドルが発生原因である場合のみ
IBFULL	b0101	(* このイベントでは無効 *)
IMISS	b0110	命令キャッシュ・ミスストールが発生原因である場合のみ
TLBMISS	b0111	TLB ストールが発生原因である場合のみ
FILL_RECIRC	b1000	キャッシュ・ライン・フィル操作の再循環が発生原因である場合のみ
BI	b1001	分岐初期設定のストールが発生原因である場合のみ
BRQ	b1010	分岐リタイアメント・キューのストールが発生原因である場合のみ
PLP	b1011	完全なループ予測のストールが発生原因である場合のみ
BR_ILOCK	b1100	分岐インターロックのストールが発生原因である場合のみ
BUBBLE	b1101	分岐リステア・バブルのストールが発生原因である場合のみ
---	b1110-b1111	(* 不正な選択 *)

BE_RSE_BUBBLE

- タイトル: RSE ストールが原因であるメイン・パイプ内のフル・パイプ・バブル
- カテゴリ: ストール・イベント IAR/DAR/OPC: N/N/N
- イベント・コード: 0x01、最大 Inc/Cyc: 1
- 定義: レジスタ・スタック・エンジンにより引き起こされたストールが原因である、メイン・パイプ内のフル・パイプ・バブルの数をカウントする。
- 注: AR_DEP の優先順位は、OVERFLOW、UNDERFLOW、LOADRS よりも高い。ただし、これは、実装された唯一の優先順位付けである。OVERFLOW、UNDERFLOW、または LOADRS をカウントするには、AR_DEP を偽にする必要がある。

表 11-44. BE_RSE_BUBBLE のユニット・マスク

拡張子	PMC.umask [19:16]	説明
ALL	bx000	RSE によって、バックエンドがストールされた。
BANK_SWITCH	bx001	バンク切り替えが原因で、バックエンドが RSE によってストールされた。
AR_DEP	bx010	AR の依存関係が原因で、バックエンドが RSE によってストールされた。
OVERFLOW	bx011	スピルが必要であるため、バックエンドが RSE によってストールされた。
UNDERFLOW	bx100	フィルが必要であるため、バックエンドが RSE によってストールされた。
LOADRS	bx101	loadrs 計算が原因で、バックエンドが RSE によってストールされた。
---	bx110-bx111	(* 何もカウントされない *)

BRANCH_EVENT

- タイトル: 収集された分岐イベント
- カテゴリ: 分岐イベント IAR/DAR/OPC: Y/N/Y
- イベント・コード: 0x11、最大 Inc/Cyc: 1
- 定義: PMC12 の制限 (「パフォーマンス監視制御レジスタ」の元で定義される) に一致する、リタイアした分岐バンドルの数をカウントする。

BR_MISPRED_DETAIL

- タイトル: FE 分岐予測ミスの詳細
- カテゴリ: 分岐イベント IAR/DAR/OPC: Y/N/Y
- イベント・コード: 0x5b、最大 Inc/Cyc: 3
- 定義: リタイアした分岐の数をカウントする。予測結果 (予測ミスされたパス、またはフロントエンドによるターゲット・アドレス)、分岐タイプに基づいて情報を提供するために、PMC.umask の 16 個の値はすべて有効である。

表 11-45. BR_MISPRED_DETAIL のユニット・マスク

拡張子	PMC.umask [19:16]	説明
ALL.ALL_PRED	b0000	予測結果に関わらず、すべての分岐タイプ
ALL.CORRECT_PRED	b0001	すべての分岐タイプ、正確に予測された分岐 (結果およびターゲット)
ALL.WRONG_PATH	b0010	すべての分岐タイプ、誤った分岐予測が原因で予測ミスされた分岐
ALL.WRONG_TARGET	b0011	すべての分岐タイプ、実行された分岐のターゲットの誤りによって予測ミスされた分岐
IPREL.ALL_PRED	b0100	予測結果に関わらず、IP 相対分岐のみ
IPREL.CORRECT_PRED	b0101	IP 相対分岐、正確に予測された分岐のみ (結果およびターゲット)
IPREL.WRONG_PATH	b0110	IP 相対分岐、分岐予測の誤りによって予測ミスされた分岐のみ
IPREL.WRONG_TARGET	b0111	IP 相対分岐、実行された分岐のターゲットの誤りによって予測ミスされた分岐のみ
RETURN.ALL_PRED	b1000	予測結果に関わらず、リターン・タイプの分岐のみ
RETURN.CORRECT_PRED	b1001	リターン・タイプ分岐、正確に予測された分岐のみ (結果およびターゲット)
RETURN.WRONG_PATH	b1010	リターン・タイプ分岐、分岐予測の誤りによって予測ミスされた分岐のみ
RETURN.WRONG_TARGET	b1011	リターン・タイプ分岐、実行された分岐のターゲットの誤りによって予測ミスされた分岐のみ
NRETIND.ALL_PRED	b1100	予測結果に関わらず、非リターン間接分岐のみ
NRETIND.CORRECT_PRED	b1101	非リターン間接分岐、正確に予測された分岐のみ (結果およびターゲット)
NRETIND.WRONG_PATH	b1110	非リターン間接分岐、分岐予測の誤りによって予測ミスされた分岐のみ
NRETIND.WRONG_TARGET	b1111	非リターン間接分岐、実行された分岐のターゲットの誤りによって予測ミスされた分岐のみ

BR_MISPRED_DETAIL2

- タイトル: FE 分岐予測ミスの詳細 (不明なパス・コンポーネント)
- カテゴリ: 分岐イベント IAR/DAR/OPC: Y/N/Y
- イベント・コード: 0x68、最大 Inc/Cyc: 2
- 定義: このイベントは、予測結果と分岐タイプに基づいて BR_MISPRED_DETAIL イベントとともに使用する。
- 注: 正確な予測ミス・カウントを得るには、次の式を使用する。

$$\text{BR_MISPRED_DETAIL}.\text{[umask]} - \text{BR_MISPRED_DETAIL2}.\text{[umask]}$$
 すべての umask についてこの計算を実行すると、BR_MISPRED_DETAIL イベントの正確な値が得られる。

表 11-46. BR_MISPREDICT_DETAIL2 のユニット・マスク

拡張子	PMC.umask [19:16]	説明
ALL.ALL_UNKNOWN_PRED	b0000	すべての分岐タイプ、パス予測が不明な分岐
ALL.UNKNOWN_PATH_CORRECT_PRED	b0001	すべての分岐タイプ、パス予測が不明な分岐、正確に予測された分岐 (結果およびターゲット)
ALL.UNKNOWN_PATH_WRONG_PATH	b0010	すべての分岐タイプ、パス予測が不明であり分岐予測が誤りである分岐
---	b0011	(* 何もカウントされない *)
IPREL.ALL_UNKNOWN_PRED	b0100	IP 相対分岐、パス予測が不明な分岐のみ
IPREL.UNKNOWN_PATH_CORRECT_PRED	b0101	IP 相対分岐、パス予測が不明な分岐、正確に予測された分岐のみ (結果およびターゲット)
IPREL.UNKNOWN_PATH_WRONG_PATH	b0110	IP 相対分岐、パス予測が不明であり分岐予測が誤りである分岐のみ
---	b0111	(* 何もカウントされない *)
RETURN.ALL_UNKNOWN_PRED	b1000	リターン・タイプ分岐、パス予測が不明な分岐のみ
RETURN.UNKNOWN_PATH_CORRECT_PRED	b1001	リターン・タイプ分岐、パス予測が不明な分岐、正確に予測された分岐のみ (結果およびターゲット)
RETURN.UNKNOWN_PATH_WRONG_PATH	b1010	リターン・タイプ分岐、パス予測が不明であり分岐予測が誤りである分岐のみ
---	b1011	(* 何もカウントされない *)
NRETIND.ALL_UNKNOWN_PRED	b1100	非リターン間接分岐、パス予測が不明な分岐のみ
NRETIND.UNKNOWN_PATH_CORRECT_PRED	b1101	非リターン間接分岐、パス予測が不明な分岐、正確に予測された分岐のみ (結果およびターゲット)
NRETIND.UNKNOWN_PATH_WRONG_PATH	b1110	非リターン間接分岐、パス予測が不明であり分岐予測が誤りである分岐のみ
---	b1111	(* 何もカウントされない *)

BR_PATH_PRED

- タイトル: FE 分岐パス予測の詳細
- カテゴリ: 分岐イベント IAR/DAR/OPC: Y/N/Y
- イベント・コード: 0x54、最大 Inc/Cyc: 3
- 定義: 分岐予測 (実行された / 実行されない)、分岐プレディケーション、分岐タイプに基づいて、リタイアした分岐の数をカウントする。16 個ある PMC.umask の値はすべて有効である。

表 11-47. BR_PATH_PRED のユニット・マスク

拡張子	PMC.umask [19:16]	説明
ALL.MISPRED_NOTTAKEN	b0000	すべての分岐タイプ、不正確に予測されたパス、実行されなかった分岐
ALL.MISPRED_TAKEN	b0001	すべての分岐タイプ、不正確に予測されたパス、実行された分岐
ALL.OKPRED_NOTTAKEN	b0010	すべての分岐タイプ、正確に予測されたパス、実行されなかった分岐
ALL.OKPRED_TAKEN	b0011	すべての分岐タイプ、正確に予測されたパス、実行された分岐
IPREL.MISPRED_NOTTAKEN	b0100	IP 相対分岐、不正確に予測されたパス、実行されなかった分岐のみ
IPREL.MISPRED_TAKEN	b0101	IP 相対分岐、不正確に予測されたパス、実行された分岐のみ
IPREL.OKPRED_NOTTAKEN	b0110	IP 相対分岐、正確に予測されたパス、実行されなかった分岐のみ
IPREL.OKPRED_TAKEN	b0111	IP 相対分岐、正確に予測されたパス、実行された分岐のみ
RETURN.MISPRED_NOTTAKEN	b1000	リターン・タイプ分岐、不正確に予測されたパス、実行されなかった分岐のみ
RETURN.MISPRED_TAKEN	b1001	リターン・タイプ分岐、不正確に予測されたパス、実行された分岐のみ
RETURN.OKPRED_NOTTAKEN	b1010	リターン・タイプ分岐、正確に予測されたパス、実行されなかった分岐のみ
RETURN.OKPRED_TAKEN	b1011	リターン・タイプ分岐、正確に予測されたパス、実行された分岐のみ
NRETIND.MISPRED_NOTTAKEN	b1100	非リターン間接分岐、不正確に予測されたパス、実行されなかった分岐のみ
NRETIND.MISPRED_TAKEN	b1101	非リターン間接分岐、不正確に予測されたパス、実行された分岐のみ
NRETIND.OKPRED_NOTTAKEN	b1110	非リターン間接分岐、正確に予測されたパス、実行されなかった分岐のみ
NRETIND.OKPRED_TAKEN	b1111	非リターン間接分岐、正確に予測されたパス、実行された分岐のみ

BR_PATH_PRED2

- タイトル: FE 分岐パス予測の詳細 (不明な予測コンポーネント)
- カテゴリ: 分岐イベント IAR/DAR/OPC: Y/N/Y
- イベント・コード: 0x6a、最大 Inc/Cyc: 2
- 定義: このイベントは、BR_PATH_PREDICTION イベントとともに使用する。
- 注: バンドル内に複数の分岐が存在し、その 1 つが実行されるとして予測されている場合、より大きな番号のポートはすべて、それらの正確な予測を実際に行わずに、実行されないと予測されるモードに強制される。

OKPRED_NOTTAKEN の正確な予測パス情報は、次の式を計算すると得られる。

$BR_PATH_PRED.[分岐タイプ].OKPRED_NOTTAKEN - BR_PATH_PRED2.[分岐タイプ].$

UNKNOWNPRED_NOTTAKEN。「分岐タイプ」には、両方のイベントに指定されたのと同じものを指定する (ALL、IPREL、RETURN、NRETIND のいずれか)。

同様に、MISPRED_TAKEN の正確な予測パス情報は、次の式を計算すると得られる。

$BR_PATH_PRED.[分岐タイプ].MISPRED_TAKEN - BR_PATH_PRED2.[分岐タイプ].$

UNKNOWNPRED_TAKEN。「分岐タイプ」には、両方のイベントに選択されたのと同じものを指定する (ALL、IPREL、RETURN、NRETIND のいずれか)。

表 11-48. BR_PATH_PRED2 のユニット・マスク

拡張子	PMC.umask [19:16]	説明
ALL.UNKNOWNPRED_NOTTAKEN	b00x0	すべての分岐タイプ、不明な予測パス、実行されなかった分岐 (これは OKPRED_NOTTAKEN に影響を与える)
ALL.UNKNOWNPRED_TAKEN	b00x1	すべての分岐タイプ、不明な予測パス、実行された分岐 (これは MISPRED_TAKEN に影響を与える)
IPREL.UNKNOWNPRED_NOTTAKEN	b01x0	IP 相対分岐、不明な予測パス、実行されなかった分岐 (これは OKPRED_NOTTAKEN に影響を与える) のみ
IPREL.UNKNOWNPRED_TAKEN	b01x1	IP 相対分岐、不明な予測パス、実行された分岐 (これは MISPRED_TAKEN に影響を与える) のみ
RETURN.UNKNOWNPRED_NOTTAKEN	b10x0	リターン・タイプ分岐、不明な予測パス、実行されなかった分岐 (これは OKPRED_NOTTAKEN に影響を与える) のみ
RETURN.UNKNOWNPRED_TAKEN	b10x1	リターン・タイプ分岐、不明な予測パス、実行された分岐 (これは MISPRED_TAKEN に影響を与える) のみ
NRETIND.UNKNOWNPRED_NOTTAKEN	b11x0	非リターン間接分岐、不明な予測パス、実行されなかった分岐 (これは OKPRED_NOTTAKEN に影響を与える) のみ
NRETIND.UNKNOWNPRED_TAKEN	b11x1	非リターン間接分岐、不明な予測パス、実行された分岐 (これは MISPRED_TAKEN に影響を与える) のみ

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 IAR/DAR/OPC: N/N/N
- イベント・コード: 0x87、最大 Inc/Cyc: 1
- 定義: バス・トランザクションの数をカウントする。

表 11-49. BUS_ALL のユニット・マスク

拡張子	PMC.umask [19:16]	説明
---	bxx00	(* 何もカウントされない *)
IO	bxx01	非 CPU プライオリティ・エージェント
SELF	bxx10	ローカル・プロセッサ
ANY	bxx11	CPU または非 CPU (すべてのトランザクション)

BUS_BACKSNP_REQ

- タイトル: バス・バック・スヌープ要求
- カテゴリ: フロントサイド・バス IAR/DAR/OPC: N/N/N
- イベント・コード: 0x8e、最大 Inc/Cyc: 1
- 定義: バス・ユニットにより受け入れられたバス・バック・スヌープ me 要求の数をカウントする。

表 11-50. BUS_BACKSNP_REQ のユニット・マスク

拡張子	PMC.umask [19:16]	説明
---	bxxx0	(* 何もカウントされない *)
THIS	bxxx1	バス・バック・スヌープ me 要求の数をカウントする。

BUS_BRQ_LIVE_REQ_HI

- タイトル: BRQ ライブ要求 (上位 2 ビット)
- カテゴリ: フロントサイド・バス IAR/DAR/OPC: N/N/N
- イベント・コード: 0x9c、最大 Inc/Cyc: 2
- 定義: BRQ 内のライブ読み出し要求の数をカウントする。Itanium 2 プロセッサでは、このイベントを 1 サイクル当たり合計 16 個含むことができる。上位 2 ビットは、このカウンタに格納される (ビット 4:3)。
- 注: 読み出し要求がピクティムを持つ場合、その読み出し要求は (ライトバックとして) BRQ にも入力される。読み出しまたはそのピクティムが BRQ 内にある限り、このイベントは 1 をカウントする (最終的な影響は、ピクティムのために、BRQ 内の読み出しの寿命が長くなる)。

BUS_BRQ_LIVE_REQ_LO

- タイトル: BRQ ライブ要求 (下位 3 ビット)
- カテゴリ: フロントサイド・バス IAR/DAR/OPC: N/N/N
- イベント・コード: 0x9b、最大 Inc/Cyc: 7
- 定義: BRQ 内のライブ読み出し要求の数をカウントする。Itanium 2 プロセッサでは、このイベントを 1 サイクル当たり合計 16 個含むことができる。下位 3 ビットは、このカウンタに格納される (ビット 2:0)。
- 注: 読み出し要求がピクティムを持つ場合、その読み出し要求は (ライトバックとして) BRQ にも入力される。読み出しまたはそのピクティムが BRQ 内にある限り、このイベントは 1 をカウントする (最終的な影響は、ピクティムのために、BRQ 内の読み出しの寿命が長くなる)。

BUS_BRQ_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 IAR/DAR/OPC: N/N/N
- イベント・コード: 0x9d、最大 Inc/Cyc: 1
- 定義: BRQ に挿入された要求の数をカウントする。
- 注: L2 ピクティム (読み出し、fc、cc により引き起こされる) が原因で BRQ 内に生成されるエントリは、カウントされない。

BUS_DATA_CYCLE

- タイトル: バス上の有効なデータ・サイクル
- カテゴリ: フロントサイド・バス IAR/DAR/OPC: N/N/N
- イベント・コード: 0x88、最大 Inc/Cyc: 1
- 定義: バス上の有効なデータ・サイクルを持っていた BUS クロックの数をカウントする。

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 IAR/DAR/OPC: N/N/N
- イベント・コード: 0x84、最大 Inc/Cyc: 1
- 定義: HITM がアサートされたトランザクションの数をカウントする (すなわち、トランザクションは、その他のプロセッサのモディファイド・ラインによって満たされている)。
- 注: これは、BUS_RD_INVALID_ALL_HITM + BUS_RD_HITM と同じ働きをする。

BUS_IO

- タイトル: IA-32 互換 IO バス・トランザクション
- カテゴリ: フロントサイド・バス IAR/DAR/OPC: N/N/N
- イベント・コード: 0x90、最大 Inc/Cyc: 1
- 定義: IA-32 I/O トランザクションの数をカウントする。

表 11-51. BUS_IO のユニット・マスク

拡張子	PMC.umask [19:16]	説明
---	bxx00	(* 何もカウントされない *)
IO	bxx01	非 CPU プライオリティ・エージェント
SELF	bxx10	ローカル・プロセッサ
ANY	bxx11	CPU または非 CPU (すべてのトランザクション)

BUS_IOQ_LIVE_REQ_HI

- タイトル: インオーダー・バス・キュー要求 (上位 2 ビット)
- カテゴリ: フロントサイド・バス IAR/DAR/OPC: N/N/N
- イベント・コード: 0x98、最大 Inc/Cyc: 2
- 定義: ライブ・インオーダー・バス要求の数をカウントする。Itanium 2 プロセッサでは、このイベントを 1 サイクル当たり合計 8 個含むことができる。上位 2 ビットは、このカウンタに格納される。

BUS_IOQ_LIVE_REQ_LO

- タイトル: インオーダー・バス・キュー要求 (下位 2 ビット)
- カテゴリ: フロントサイド・バス IAR/DAR/OPC: N/N/N
- イベント・コード: 0x97、最大 Inc/Cyc: 3
- 定義: ライブ・インオーダー・バス要求の数をカウントする。Itanium 2 プロセッサでは、このイベントを 1 サイクル当たり合計 8 個含むことができる。下位 2 ビットは、このカウンタに格納される。

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 IAR/DAR/OPC: N/N/N
- イベント・コード: 0x93、最大 Inc/Cyc: 1
- 定義: IA-32 バス・ロック・トランザクションの数をカウントする。

表 11-52. BUS_LOCK のユニット・マスク

拡張子	PMC.umask [19:16]	説明
---	bxx00	(* 何もカウントされない *)
---	bxx01	(* 不正な選択 *)
SELF	bxx10	ローカル・プロセッサ
ANY	bxx11	CPU または非 CPU (すべてのトランザクション)

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 IAR/DAR/OPC: N/N/N
- イベント・コード: 0x8a、最大 Inc/Cyc: 1
- 定義: バス・メモリ・トランザクション (メモリ読み出し無効化、予約メモリ読み出し、メモリ読み出し、メモリ書き込みトランザクション) の数をカウントする。

表 11-53. BUS_MEMORY のユニット・マスク

拡張子	PMC.umask [19:16]	説明
---	b00xx	(* 何もカウントされない *)
---	b0100	(* 何もカウントされない *)
EQ_128BYTE.IO	b0101	非 CPU プライオリティ・エージェントからのフル・キャッシュ・ライン・トランザクション (BRL、BRIL、BWL) の数
EQ_128BYTE.SELF	b0110	ローカル・プロセッサからのフル・キャッシュ・ライン・トランザクション (BRL、BRIL、BWL) の数
EQ_128BYTE.ANY	b0111	CPU または非 CPU (すべてのトランザクション) からのフル・キャッシュ・ライン・トランザクション (BRL、BRIL、BWL) の数
---	b1000	(* 何もカウントされない *)
LT_128BYTE.IO	b1001	非 CPU プライオリティ・エージェントからのいっぱいになっていないキャッシュ・ライン・トランザクション (BRP、BWP) の数
LT_128BYTE.SELF	b1010	いっぱいになっていないキャッシュ・ライン・トランザクション (BRP、BWP) ローカル・プロセッサの数
LT_128BYTE.ANY	b1011	いっぱいになっていないキャッシュ・ライン・トランザクション (BRP、BWP) CPU または非 CPU (すべてのトランザクション) の数
---	b1100	(* 何もカウントされない *)
ALL.IO	b1101	非 CPU プライオリティ・エージェントからのすべてのバス・トランザクション
ALL.SELF	b1110	ローカル・プロセッサからのすべてのバス・トランザクション
ALL.ANY	b1111	CPU または非 CPU (すべてのトランザクション) からのすべてのバス・トランザクション

BUS_MEM_READ

- タイトル: フル・キャッシュ・ライン D/I メモリ RD、RD 無効化、BRIL
- カテゴリ: フロントサイド・バス IAR/DAR/OPC: N/N/N
- イベント・コード: 0x8b、最大 Inc/Cyc: 1
- 定義: フル・キャッシュ・ライン (128 バイト) のデータ/コード・メモリ読み出し (BRL)、フル・キャッシュ・ラインのメモリ読み出し無効化 (BRIL)、0 バイトのメモリ読み出し無効化 (BIL) トランザクションの数をカウントする。

表 11-54. BUS_MEM_READ のユニット・マスク

拡張子	PMC.umask [19:16]	説明
---	b0000	(* 何もカウントされない *)
BIL.IO	b0001	非 CPU プライオリティ・エージェントからの BIL 0 バイト・メモリ読み出し無効化トランザクションの数
BIL.SELF	b0010	ローカル・プロセッサからの BIL 0 バイト・メモリ読み出し無効化トランザクションの数
BIL.ANY	b0011	CPU または非 CPU (すべてのトランザクション) からの BIL 0 バイト・メモリ読み出し無効化トランザクションの数
---	b0100	(* 何もカウントされない *)
BRL.IO	b0101	非 CPU プライオリティ・エージェントからのフル・キャッシュ・ライン・メモリ読み出しトランザクションの数
BRL.SELF	b0110	ローカル・プロセッサからのフル・キャッシュ・ライン・メモリ読み出しトランザクションの数
BRL.ANY	b0111	CPU または非 CPU (すべてのトランザクション) からのフル・キャッシュ・ライン・メモリ読み出しトランザクションの数
---	b1000	(* 何もカウントされない *)
BRIL.IO	b1001	非 CPU プライオリティ・エージェントからのフル・キャッシュ・ライン・メモリ読み出し無効化トランザクションの数
BRIL.SELF	b1010	ローカル・プロセッサからのフル・キャッシュ・ライン・メモリ読み出し無効化トランザクションの数
BRIL.ANY	b1011	CPU または非 CPU (すべてのトランザクション) からのフル・キャッシュ・ライン・メモリ読み出し無効化トランザクションの数
---	b1100	(* 何もカウントされない *)
ALL.IO	b1101	非 CPU プライオリティ・エージェントからのすべてのメモリ読み出しトランザクション
ALL.SELF	b1110	ローカル・プロセッサからのすべてのメモリ読み出しトランザクション
ALL.ANY	b1111	CPU または非 CPU (すべてのトランザクション) からのすべてのメモリ読み出しトランザクション

BUS_MEM_READ_OUT_HI

- タイトル: 未処理のメモリ読み出しトランザクション (上位 2 ビット)
- カテゴリ: フロントサイド・バス IAR/DAR/OPC: N/N/N
- イベント・コード: 0x94, 最大 Inc/Cyc: 2
- 定義: 未処理のメモリ読み出しトランザクションの数をカウントする。Itanium 2 プロセッサでは、このイベントを 1 サイクル当たり合計 16 個含むことができる。上位 2 ビットは、このカウンタに格納される。このイベントの目的のため、読み出し要求が FSB 上に発効されてから、先頭の読み出しデータ群が L2 に返されるまで、メモリ読み出しアクセスは未処理であると想定される。
- 注: キャッシュ不可 (または L3 にアクセスしないその他のもの) は、トラックされない。このイベントは、平均システム・メモリ・レイテンシを得るための BUS_MEM_READ [all,self] とともに使用することを目的としている。

BUS_MEM_READ_OUT_LO

- タイトル: 未処理のメモリ読み出しトランザクション (下位 3 ビット)
- カテゴリ: フロントサイド・バス IAR/DAR/OPC: N/N/N
- イベント・コード: 0x95、最大 Inc/Cyc: 7
- 定義: 未処理のメモリ読み出しトランザクションの数をカウントする。Itanium 2 プロセッサでは、このイベントを 1 サイクル当たり合計 16 個含むことができる。下位 3 ビットは、このカウンタに格納される。このイベントの目的のため、読み出し要求が FSB 上に発効されてから、先頭の読み出しデータ群が L2 に返されるまで、メモリ読み出しアクセスは未処理であると想定される。
- 注: キャッシュ不可 (または L3 にアクセスしないその他のもの) は、トラックされない。このイベントは、平均システム・メモリ・レイテンシを得るための BUS_MEM_READ [all,self] とともに使用することを目的としている。

BUS_OOQ_LIVE_REQ_HI

- タイトル: アウト・オブ・オーダー・バス・キュー要求 (上位 2 ビット)
- カテゴリ: フロントサイド・バス IAR/DAR/OPC: N/N/N
- イベント・コード: 0x9a、最大 Inc/Cyc: 2
- 定義: ライブ遅延 (アウト・オブ・オーダー) バス要求の数をカウントする。Itanium 2 プロセッサでは、このイベントを 1 サイクル当たり合計 18 個含むことができる。上位 2 ビットは、このカウンタに格納される (ビット 4:3)。このイベントは、すべての CPU クロック・サイクルをインクリメントする。カウンタは、その時点でのライブ遅延トランザクションの数でインクリメントされる。
- 注: BUS_OOQ_LIVE_REQ/CPU_CYCLES は、1CPU クロック当たりの未処理の遅延トランザクションの平均個数を示す。

BUS_OOQ_LIVE_REQ_LO

- タイトル: アウト・オブ・オーダー・バス・キュー要求 (下位 3 ビット)
- カテゴリ: フロントサイド・バス IAR/DAR/OPC: N/N/N
- イベント・コード: 0x99、最大 Inc/Cyc: 7
- 定義: ライブ遅延 (アウト・オブ・オーダー) バス要求の数をカウントする。Itanium 2 プロセッサでは、このイベントを 1 サイクル当たり合計 18 個含むことができる。下位 3 ビットは、このカウンタに格納される (ビット 2:0)。このイベントは、すべての CPU クロック・サイクルをインクリメントする。カウンタは、その時点でのライブ遅延トランザクションの数でインクリメントされる。
- 注: BUS_OOQ_LIVE_REQ/CPU_CYCLES は、1CPU クロック当たりの未処理の遅延トランザクションの平均個数を示す。

BUS_RD_DATA

- タイトル: バス読み出しデータ・トランザクション
- カテゴリ: フロントサイド・バス IAR/DAR/OPC: N/N/N
- イベント・コード: 0x8c、最大 Inc/Cyc: 1
- 定義: フル・キャッシュ・ライン (128 バイト) データ・メモリ読み出しトランザクション (BRL) の数をカウントする。

表 11-55. BUS_RD_DATA のユニット・マスク

拡張子	PMC.umask [19:16]	説明
---	bxx00	(* 何もカウントされない *)
IO	bxx01	非 CPU プライオリティ・エージェント
SELF	bxx10	ローカル・プロセッサ
ANY	bxx11	CPU または非 CPU (すべてのトランザクション)

BUS_RD_HIT

- タイトル: バス読み出しヒット・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 IAR/DAR/OPC: N/N/N
- イベント・コード: 0x80、最大 Inc/Cyc: 1
- 定義: 別のプロセッサのキャッシュ内のクリーン・ラインをヒットしたバス読み出しの数をカウントする (HIT と BRL を意味する)。

BUS_RD_HITM

- タイトル: バス読み出しヒット・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 IAR/DAR/OPC: N/N/N
- イベント・コード: 0x81、最大 Inc/Cyc: 1
- 定義: 別のプロセッサのキャッシュ内のモディファイド・ラインをヒットしたバス読み出しの数をカウントする (HITM と BRL を意味する)。

BUS_RD_INVALID_ALL_HITM

- タイトル: HITM におけるバス BRIL トランザクションおよびバス 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 IAR/DAR/OPC: N/N/N
- イベント・コード: 0x83、最大 Inc/Cyc: 1
- 定義: リモート・プロセッサのみから満たされるバス読み出し無効化ライン・トランザクション (BRIL または BIL および HITM を意味する) の数をカウントする。

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 IAR/DAR/OPC: N/N/N
- イベント・コード: 0x82、最大 Inc/Cyc: 1
- 定義: HITM がアサートされ (BIL と HITM を意味する)、トランザクションが別のプロセッサのキャッシュから満たされた、バス読み出し無効化ライン・トランザクションの数をカウントする。

BUS_RD_IO

- タイトル: IA-32 互換 IO 読み出しトランザクション
- カテゴリ: フロントサイド・バス IAR/DAR/OPC: N/N/N
- イベント・コード: 0x91、最大 Inc/Cyc: 1
- 定義: IA-32 I/O 読み出しトランザクションの数をカウントする。

表 11-56. BUS_RD_IO のユニット・マスク

拡張子	PMC.umask [19:16]	説明
---	bxx00	(* 何もカウントされない *)
IO	bxx01	非 CPU プライオリティ・エージェント
SELF	bxx10	ローカル・プロセッサ
ANY	bxx11	CPU または非 CPU (すべてのトランザクション)

BUS_RD_PRTL

- タイトル: バス読み出しパーシャル・トランザクション
- カテゴリ: フロントサイド・バス IAR/DAR/OPC: N/N/N
- イベント・コード: 0x8d、最大 Inc/Cyc: 1
- 定義: いっぱいではないキャッシュ・ライン (0、8、16、32、64 バイト) メモリ読み出しトランザクション (BRP) の数をカウントする。

表 11-57. BUS_RD_PRTL のユニット・マスク

拡張子	PMC.umask [19:16]	説明
---	bxx00	(* 何もカウントされない *)
IO	bxx01	非 CPU プライオリティ・エージェント
SELF	bxx10	ローカル・プロセッサ
ANY	bxx11	CPU または非 CPU (すべてのトランザクション)

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 IAR/DAR/OPC: N/N/N
- イベント・コード: 0x96、最大 Inc/Cyc: 7
- 定義: ライブ・スヌープ応答の数をカウントする。このイベントは、すべての CPU クロック・サイクルをインクリメントする。カウンタがインクリメントされる量は、その時点での未処理のスヌープ応答の数である。

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 IAR/DAR/OPC: N/N/N
- イベント・コード: 0x86、最大 Inc/Cyc: 1
- 定義: バス上のバス・スヌープ要求の数をカウントする。

表 11-58. BUS_SNOOPS のユニット・マスク

拡張子	PMC.umask [19:16]	説明
---	bxx00	(* 何もカウントされない *)
IO	bxx01	非 CPU プライオリティ・エージェント

表 11-58. BUS_SNOOPS のユニット・マスク (続き)

拡張子	PMC.umask [19:16]	説明
SELF	bxx10	ローカル・プロセッサ
ANY	bxx11	CPU または非 CPU (すべてのトランザクション)

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 IAR/DAR/OPC: N/N/N
- イベント・コード: 0x85、最大 Inc/Cyc: 1
- 定義: ローカル・プロセッサ内のモディファイド・ラインをヒットしたりリモート・プロセッサからのバス・スヌープ要求の数をカウントする。

表 11-59. BUS_SNOOPS_HITM のユニット・マスク

拡張子	PMC.umask [19:16]	説明
---	bxx00	(* 何もカウントされない *)
---	bxx01	(* 不正な選択 *)
SELF	bxx10	ローカル・プロセッサ
ANY	bxx11	CPU または非 CPU (すべてのトランザクション)

BUS_SNOOP_STALL_CYCLES

- タイトル: バス・スヌープ・ストール・サイクル (任意のエージェントから)
- カテゴリ: フロントサイド・バス IAR/DAR/OPC: N/N/N
- イベント・コード: 0x8f、最大 Inc/Cyc: 1
- 定義: スヌープのために FSB がストールされたバス・クロック数をカウントする (これは、HIT および HITM が同時にアサートされたバス・クロック数の 2 倍である)。

表 11-60. BUS_SNOOP_STALL_CYCLES のユニット・マスク

拡張子	PMC.umask [19:16]	説明
---	bxx00	(* 何もカウントされない *)
---	bxx01	(* 不正な選択 *)
SELF	bxx10	ローカル・プロセッサ
ANY	bxx11	CPU または非 CPU (すべてのトランザクション)

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 IAR/DAR/OPC: N/N/N
- イベント・コード: 0x92、最大 Inc/Cyc: 1
- 定義: ライトバック・メモリ書き込みトランザクションの数をカウントする (BWL は、M ステート・ライン・ライトバックおよびコアレスリング・ライトが原因で、書き込みを行う)。

表 11-61. BUS_WR_WB のユニット・マスク

拡張子	PMC.umask [19:16]	説明
---	b00xx	(* 何もカウントされない *)
---	b0100	(* 何もカウントされない *)
EQ_128BYTE.IO	b0101	非 CPU プライオリティ・エージェント/ライトバック属性またはライト・コアレッシング属性を持つキャッシュ・ライン・トランザクションのみがカウントされる。
EQ_128BYTE.SELF	b0110	ローカル・プロセッサ/ライトバック属性またはライト・コアレッシング属性を持つキャッシュ・ライン・トランザクションのみがカウントされる。
EQ_128BYTE.ANY	b0111	CPU または非 CPU (すべてのトランザクション)/ライトバック属性またはライト・コアレッシング属性を持つキャッシュ・ライン・トランザクションのみがカウントされる。
---	b1000	(* 何もカウントされない *)
---	b1001	(* 不正な選択 *)
CCASTOUT.SELF	b1010	ローカル・プロセッサ/ライトバック属性を持つ 0 バイト・トランザクション (クリーン・キャスト・アウト) のみがカウントされる。
CCASTOUT.ANY	b1011	CPU または非 CPU (すべてのトランザクション)/ライトバック属性を持つ 0 バイト・トランザクションのみがカウントされる。
---	b1100	(* 何もカウントされない *)
ALL.IO	b1101	非 CPU プライオリティ・エージェント
ALL.SELF	b1110	ローカル・プロセッサ
ALL.ANY	b1111	CPU または非 CPU (すべてのトランザクション)

CPU_CPL_CHANGES

- タイトル: 特権レベルの変更
- カテゴリ: システム・イベント IAR/DAR/OPC: N/N/N
- イベント・コード: 0x13、最大 Inc/Cyc: 1
- 定義: 特権レベルの変更回数をカウントする。

CPU_CYCLES

- タイトル: CPU サイクル
- カテゴリ: 基本イベント IAR/DAR/OPC: N/N/N
- イベント・コード: 0x12、最大 Inc/Cyc: 1
- 定義: クロック・サイクル数をカウントする。

DATA_DEBUG_REGISTER_FAULT

- タイトル: データ・デバッグ・レジスタが原因のフォルト。ロード/ストア命令の一致
- カテゴリ: システム・イベント IAR/DAR/OPC: N/N/N
- イベント・コード: 0x52、最大 Inc/Cyc: 1
- 定義: データ・デバッグ・レジスタの 1 つがロード命令またはストア命令に一致しているのが原因でフォルトになった回数をカウントする。

DATA_DEBUG_REGISTER_MATCHES

- タイトル: データ・デバッグ・レジスタが、メモリ参照のデータ・アドレスに一致
- カテゴリ: システム・イベント IAR/DAR/OPC: Y/Y/Y
- イベント・コード: 0xc6、最大 Inc/Cyc: 1
- 定義: データ・デバッグ・レジスタがメモリ参照のデータ・アドレスに一致した回数をカウントする。これは、4 DBR が一致する OR 関数である。このイベントに影響を与えるのは、DBR0-7、PSR、DCR、PMC13 の各レジスタである。DATA_DEBUG_REGISTER_MATCHES はコミットを含まないため、ノイズを持つ可能性がある。

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 IAR/DAR/OPC: Y/Y/Y
- イベント・コード: 0xc8、最大 Inc/Cyc: 1
- 定義: EAR により収集された L1 データ・キャッシュ、または L1DTLB イベントか ALAT イベントの数をカウントする。

DATA_REFERENCES_SET0

- タイトル: メモリ・パイプラインに発行されたデータ・メモリ参照
- カテゴリ: L1 データ・キャッシュ /L1D キャッシュ・セット 0 IAR/DAR/OPC: Y/Y/Y
- イベント・コード: 0xc3、最大 Inc/Cyc: 4
- 定義: メモリ・パイプライン内に発行されたデータ・メモリ参照の数をカウントする (これには、チェック・ロード、キャッシュ不可アクセス、RSE 操作、セマフォ、浮動小数点メモリ参照が含まれる)。このカウントには、分岐バスの予測が外れた操作は含まれるが、プレディケータ・オフの操作は含まれない。このイベントには、VHPT メモリ参照は含まれない。
- 注: 制限付きのセット 0 L1D キャッシュ・イベントである。このイベントを計測するには、このセット内のイベントの 1 つを PMD5 で計測しなければならない。

DATA_REFERENCES_SET1

- タイトル: メモリ・パイプラインに発行されたデータ・メモリ参照
- カテゴリ: L1 データ・キャッシュ /L1D キャッシュ・セット 1 IAR/DAR/OPC: Y/Y/Y
- イベント・コード: 0xc5、最大 Inc/Cyc: 4
- 定義: メモリ・パイプライン内に発行されたデータ・メモリ参照の数をカウントする (これには、チェック・ロード、キャッシュ不可アクセス、RSE 操作、セマフォ、浮動小数点メモリ参照が含まれる)。このカウントには、分岐バスの予測が外れた操作は含まれるが、プレディケータ・オフの操作は含まれない。このイベントには、VHPT メモリ参照は含まれない。
- 注: 制限付きのセット 1 L1D キャッシュ・イベントである。このイベントを計測するには、このセット内のイベントの 1 つを PMD5 で計測しなければならない。

DISP_STALLED

- タイトル: ディスパーサルがストールされたサイクル数
- カテゴリ: 命令ディスパーサル・イベント IAR/DAR/OPC: N/N/N
- イベント・コード: 0x49、最大 Inc/Cyc: 1
- 定義: フラッシュまたはバックエンド・パイプライン・ストールが原因でディスパーサルがストールされたサイクル数をカウントする。

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: TLB IAR/DAR/OPC: Y/Y/Y
- イベント・コード: 0xc9、最大 Inc/Cyc: 4
- 定義: ハードウェア・ページ・ウォークにより DTLB に挿入された VHPT エントリの数をカウントする。
- 注: これには、ミスを引き起こしている命令がリタイアしていないのに DTLB が実行されたミスが含まれる。

DTLB_INSERTS_HPW_RETIRED

- タイトル: ハードウェア・ページ・ウォークにより DTLB に挿入された VHPT エントリ
- カテゴリ: TLB IAR/DAR/OPC: Y/Y/Y
- イベント・コード: 0x2c、最大 Inc/Cyc: 4
- 定義: ハードウェア・ページ・ウォークにより DTLB に挿入された VHPT エントリの数をカウントする。
- 注: これには、ミスを引き起こしている命令がリタイアしていないのに DTLB が実行されたミスは含まれない。このイベントと DTLB_INSERTS_HPW の違いは、DTLB への不要な潜在的挿入の総数である。

ENCBR_MISPRED_DETAIL

- タイトル: リタイアしたエンコード済み分岐の数
- カテゴリ: 分岐イベント IAR/DAR/OPC: Y/N/Y
- イベント・コード: 0x63、最大 Inc/Cyc: 3
- 定義: ポート B0 (エンコードされた分岐) 上に分岐が存在する場合に限って、リタイアした分岐の数をカウントする。

表 11-62. ENCBR_MISPRED_DETAIL のユニット・マスク

拡張子	PMC.umask [19:16]	説明
ALL.ALL_PRED	b0000	予測結果に関わらず、すべてのエンコード済み分岐タイプ
ALL.CORRECT_PRED	b0001	すべてのエンコード済み分岐、正確に予測された分岐 (結果およびターゲット)
ALL.WRONG_PATH	b0010	すべてのエンコード済み分岐、分岐予測の誤りによって予測ミスされた分岐
ALL.WRONG_TARGET	b0011	すべてのエンコード済み分岐、実行された分岐のターゲットの誤りによって予測ミスされた分岐
---	b0100	(* 何もカウントされない *)
---	b0101	(* 何もカウントされない *)
---	b0110	(* 何もカウントされない *)
---	b0111	(* 何もカウントされない *)
OVERSUB.ALL_PRED	b1000	予測結果に関わらず、オーバーサブスクリプションを引き起こすもののみ
OVERSUB.CORRECT_PRED	b1001	オーバーサブスクリプションを引き起こすもの、正確に予測された分岐のみ (結果およびターゲット)
OVERSUB.WRONG_PATH	b1010	オーバーサブスクリプションを引き起こすもの、分岐予測の誤りによって予測ミスされた分岐のみ

表 11-62. ENCBR_MISPRED_DETAIL のユニット・マスク (続き)

拡張子	PMC.umask [19:16]	説明
OVERSUB.WRONG_TARGET	b1011	オーバーサブスクリプションを引き起こすもの、実行された分岐のターゲットの誤りによって予測ミスされた分岐のみ
ALL2.ALL_PRED	b1100	予測結果に関わらず、すべてのエンコード済み分岐
ALL2.CORRECT_PRED	b1101	すべてのエンコード済み分岐、正確に予測された分岐 (結果およびターゲット)
ALL2.WRONG_PATH	b1110	すべてのエンコード済み分岐、分岐予測の誤りによって予測ミスされた分岐
ALL2.WRONG_TARGET	b1111	すべてのエンコード済み分岐、実行された分岐のターゲットの誤りによって予測ミスされた分岐

EXTERN_DP_PINS_0_TO_3

- タイトル: アサートされた DP ピン 0 ~ 3
- カテゴリ: システム・イベント IAR/DAR/OPC: N/N/N
- イベント・コード: 0x9e、最大 Inc/Cyc: 1
- 定義: 外部 DP ピンの 0 ~ 3 がアサートされたバス・クロック数をカウントする。

表 11-63. EXTERN_DP_PINS_0_TO_3 のユニット・マスク

拡張子	PMC.umask [19:16]	説明
---	b0000	(* 何もカウントされない *)
PIN0	bxxx1	ピン 0 のアサートを含む。
PIN1	bxx1x	ピン 1 のアサートを含む。
PIN2	bx1xx	ピン 2 のアサートを含む。
PIN3	b1xxx	ピン 3 のアサートを含む。

EXTERN_DP_PINS_4_TO_5

- タイトル: アサートされた DP ピン 4 ~ 5
- カテゴリ: システム・イベント IAR/DAR/OPC: N/N/N
- イベント・コード: 0x9f、最大 Inc/Cyc: 1
- 定義: 外部 DP ピンの 4 ~ 5 がアサートされたバス・クロック数をカウントする。

表 11-64. EXTERN_DP_PINS_4_TO_5 のユニット・マスク

拡張子	PMC.umask [19:16]	説明
---	bxx00	(* 何もカウントされない *)
PIN4	bxxx1	ピン 4 のアサートを含む。
PIN5	bxx1x	ピン 5 のアサートを含む。

FE_BUBBLE

- タイトル: FE によって検出されたバブル
- カテゴリ: ストール・イベント IAR/DAR/OPC: N/N/N
- イベント・コード: 0x71、最大 Inc/Cyc: 1
- 定義: フロントエンドによって検出されたバブルの数をカウントする。このイベントは、FE_LOST_BW イベントを確認するもう一つの方法である。
- 注: ストールの原因は、次の順序で優先順位が付けられる: FEFLUSH、TLBMISS、IMISS、BRANCH、FILL_RECIRC、BUBBLE、IBFULL。この優先順位付けでは、いくつかのストール条件が同時に存在する場合、最も優先順位の高いものだけがカウントされる。

表 11-65. FE_BUBBLE のユニット・マスク

拡張子	PMC.umask [19:16]	説明
ALL	b0000	原因に関わらずカウントする。
FEFLUSH	b0001	フロントエンド・フラッシュが発生原因である場合のみ
---	b0010	(* カウントは未定義 *)
GROUP1	b0011	BUBBLE または BRANCH
GROUP2	b0100	IMISS または TLBMISS
IBFULL	b0101	命令バッファ・フル・ストールが発生原因である場合のみ
IMISS	b0110	命令キャッシュ・ミスストールが発生原因である場合のみ
TLBMISS	b0111	TLB ストールが発生原因である場合のみ
FILL_RECIRC	b1000	フィル操作の再循環が発生原因である場合のみ
BRANCH	b1001	4 つの分岐再循環が発生原因である場合のみ
GROUP3	b1010	FILL_RECIRC または BRANCH
ALLBUT_FEFLUSH_BUBBLE	b1011	FEFLUSH および BUBBLE を除くすべて
ALLBUT_IBFULL	b1100	IBFULL を除くすべて
BUBBLE	b1101	分岐バブル・ストールが原因である場合のみ
---	b1110-b1111	(* 何もカウントされない *)

FE_LOST_BW

- タイトル: IB への入り口での無効なバンドル
- カテゴリ: ストール・イベント IAR/DAR/OPC: N/N/N
- イベント・コード: 0x70、最大 Inc/Cyc: 2
- 定義: 命令バッファへの入り口での無効なバンドルの数をカウントする。
- 注: 帯域幅が失われた原因には、次の順序で優先順位が付けられる: FEFLUSH、TLBMISS、IMISS、PLP、BR_ILOCK、BRQ、BI、FILL_RECIRC、BUBBLE、IBFULL、UNREACHED。この優先順位付けでは、いくつかのストール条件が同時に存在する場合、最も優先順位の高いものだけがカウントされる。バンドルが「到達不可」と見なされる場合は 2 つある。実行された分岐がバンドル 0 に含まれる場合、またはバンドル 0 は無効であるが IP[4] に 1 が設定されている場合、バンドル 1 は到達されない。

表 11-66. FE_LOST_BW のユニット・マスク

拡張子	PMC.umask [19:16]	説明
ALL	b0000	原因に関わらずカウントする。
FEFLUSH	b0001	フロントエンド・フラッシュが発生原因である場合のみ
---	b0010	(* カウントは未定義 *)
---	b0011	(* 不正な選択 *)
UNREACHED	b0100	到達不可バンドルが発生原因である場合のみ
IBFULL	b0101	命令バッファ・フル・ストールが発生原因である場合のみ
IMISS	b0110	命令キャッシュ・ミス・ストールが発生原因である場合のみ
TLBMISS	b0111	TLB ストールが発生原因である場合のみ
FILL_RECIRC	b1000	キャッシュ・ライン・フィル操作の再循環が発生原因である場合のみ
BI	b1001	分岐初期設定ストールが発生原因である場合のみ
BRQ	b1010	分岐リタイアメント・キュー・ストールが発生原因である場合のみ
PLP	b1011	完全なループ予測ストールが発生原因である場合のみ
BR_ILOCK	b1100	分岐インターロック・ストールが発生原因である場合のみ
BUBBLE	b1101	分岐リステア・バブル・ストールが発生原因である場合のみ
---	b1101-b1111	(* 不正な選択 *)

FP_FAILED_FCHKF

- タイトル: 失敗した fchkf
- カテゴリ: 命令実行 IAR/DAR/OPC: Y/N/N
- イベント・コード: 0x06、最大 Inc/Cyc: 1
- 定義: fchkf 命令が失敗した回数をカウントする。

FP_FALSE_SIRSTALL

- タイトル: トラップなしの SIR ストール
- カテゴリ: 命令実行 IAR/DAR/OPC: Y/N/N
- イベント・コード: 0x05、最大 Inc/Cyc: 1
- 定義: SIR ストールがアサートされたが、トラップには至らなかった回数をカウントする。

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: 命令実行 IAR/DAR/OPC: Y/N/N
- イベント・コード: 0x0b、最大 Inc/Cyc: 2
- 定義: 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。

FP_OPS_RETIRED

- タイトル: リタイアした FP 演算
- カテゴリ: 命令実行 IAR/DAR/OPC: Y/N/N
- イベント・コード: 0x09、最大 Inc/Cyc: 4
- 定義: リタイアした浮動小数点演算の回数に関して情報を提供する (プレディケート・オフ命令はすべて除く)。これは、基本浮動小数点演算の重み付けされた合計である。特定のオペコードがリタイアした回数をカウントするには、IA64_TAGGED_INST_RETIRED を使用する。
- 注: 以下の重み付けが使用される。
 4 op としてカウント: fpma、fpms、fpnma
 2 op としてカウント: fpma、fpnma (f2=f0)、fma、fms、fnma、fprcpa、fprsqrta、fmpy、fpmax、fpamin、fpamax、fpcmp、fpcvt
 1 op としてカウント: fms、fma、fnma (f2=f0 または f4=f1)、fmpy、fadd、fsub、frcpa、frsqrta、fmin、fmax、famin、famax、fpmmin、fcvt.fx、fcmp

FP_TRUE_SIRSTALL

- タイトル: アサートされトラップに至る SIR ストール
- カテゴリ: 命令実行 IAR/DAR/OPC: Y/N/N
- イベント・コード: 0x03、最大 Inc/Cyc: 1
- 定義: SIR ストールがアサートされトラップに至る回数をカウントする。

HPW_DATA_REFERENCES

- タイトル: VHPT へのデータ・メモリ参照
- カテゴリ: L1 データ・キャッシュ IAR/DAR/OPC: Y/Y/Y
- イベント・コード: 0x2d、最大 Inc/Cyc: 4
- 定義: VHPT へのデータ・メモリ参照の回数をカウントする。
- 注: HPW が常時イネーブルの場合、このイベントと L2DTLB_MISSES は同じである。HPW が常時ディセーブルの場合、このイベントは 0 をカウントする。これには、ミスを引き起こしている命令がリタイアしていないのに L2DTLB が実行されたミスが含まれる。

IA32_INST_RETIRED

- タイトル: リタイアした IA-32 命令
- カテゴリ: 基本イベント IAR/DAR/OPC: N/N/N
- イベント・コード: 0x59、最大 Inc/Cyc: 2
- 定義: リタイアした IA-32 命令の数をカウントする。

IA32_ISA_TRANSITIONS

- タイトル: Itanium アーキテクチャと IA-32 ISA との間の遷移
- カテゴリ: 基本イベント IAR/DAR/OPC: N/N/N
- イベント・コード: 0x07、最大 Inc/Cyc: 1
- 定義: Itanium アーキテクチャから IA-32 または IA-32 から Itanium アーキテクチャへの命令セットの遷移回数 (PSR.is ビット切り替えの回数) をカウントする。

IA64_INST_RETIRED

- タイトル: リタイアした Itanium 命令
- カテゴリ: 基本イベント IAR/DAR/OPC: Y/N/Y
- イベント・コード: 0x08、最大 Inc/Cyc: 6
- 定義: リタイアした命令の数と(ハードウェアにより生成された RSE 操作を除く)、プレディケート・オフされた命令の数をカウントする。このイベントには、真のプレディケートが指定されたリタイアメントに達した非分岐命令と分岐が、プレディケートに関わらずすべて含まれる。これは、IA64_TAGGED_INST_RETIRED のサブイベントである。
- 注: MLX バンドルは、2つを超えない命令としてカウントされる。対応するレジスタは、関係のある IBRP-PMC の組み合わせによって制限されないように設定する(電源オン時のデフォルトは、制限なしである)。

非デフォルト設定の例を次に示す。

PMD4 の IA64_INST_RETIRED の計測に IBRP2-PMC8 を使用すると仮定する。これを行うには、次のビットを設定する。

PMC4.umask = xx10

PMC14.IBRP2 = 1 (PMC14 は IPF_IBRC と呼ばれる)

PMC15.IBRP2_PMC8 = 1 (PMC15 は ISD_DEBUGTAG と呼ばれる)。PMC8 は、IBRP0_PMC8 umask 用に依然として使用できるのに注意する。

表 11-67. IA64_INST_RETIRED のユニット・マスク

拡張子	PMC.umask [19:16]	説明
THIS	bxx00	リタイアした Itanium 命令

IA64_TAGGED_INST_RETIRED

- タイトル: リタイアしたタグ付き命令
- カテゴリ: 命令実行 IAR/DAR/OPC: Y/N/Y
- イベント・コード: 0x08、最大 Inc/Cyc: 6
- 定義: リタイアした命令の数と(ハードウェアにより生成された RSE 操作を除く)、命令アドレス・ブレークポイント (IBR) およびオペコード・マッチ・レジスタの設定 (PMC8,9) に一致する、プレディケート・オフされた命令の数をカウントする。このイベントには、真のプレディケートが指定されたリタイアメントに達した非分岐命令と分岐が、プレディケートに関わらずすべて含まれる。各種のレジスタの設定方法の詳細は、10.3.5 項「命令アドレス範囲マッチング」を参照のこと。
- 注: MLX バンドルは、2つを超えない命令としてカウントされる。

表 11-68. IA64_TAGGED_INST_RETIRED のユニット・マスク

拡張子	PMC.umask [19:16]	説明
IBRP0_PMC8	bxx00	命令ブレークポイント・ペア 0 とオペコード・マッチャー PMC8 にタグを付けられた命令。PSR.is=1 を設定して実行されたコードが含まれる。
IBRP1_PMC9	bxx01	命令ブレークポイント・ペア 1 とオペコード・マッチャー PMC9 にタグを付けられた命令。PSR.is=1 を設定して実行されたコードが含まれる。

表 11-68. IA64_TAGGED_INST_RETIRED のユニット・マスク (続き)

拡張子	PMC.umask [19:16]	説明
IBRP2_PMC8	bxx10	命令ブレークポイント・ペア 2 とオペコード・マッチャー PMC8 にタグを付けられた命令。PSR.is=1 を設定して実行されたコードは含まれない。
IBRP3_PMC9	bxx11	命令ブレークポイント・ペア 3 とオペコード・マッチャー PMC9 にタグを付けられた命令。PSR.is=1 を設定して実行されたコードは含まれない。

IDEAL_BE_LOST_BW_DUE_TO_FE

- タイトル: IB からの出口での無効なバンドル
- カテゴリ: ストール・イベント IAR/DAR/OPC: N/N/N
- イベント・コード: 0x73、最大 Inc/Cyc: 2
- 定義: バックエンドがその他の理由でストールされたかどうかに関わらず、命令バッファからの出口での無効なバンドルの数をカウントする。
- 注: 帯域幅が失われた原因には、次の順序で優先順位が付けられる: FEFLUSH、TLBMISS、IMISS、PLP、BR_ILOCK、BRQ、BI、FILL_RECIRC、BUBBLE、IBFULL、UNREACHED。この優先順位付けでは、いくつかのストール条件が同時に存在する場合、最も優先順位の高いものだけがカウントされる。バンドルが「到達不可」と見なされる場合は 2 つある。実行された分岐がバンドル 0 に含まれる場合、またはバンドル 0 は無効であるが IP[4] に 1 が設定されている場合、バンドル 1 は到達されない。

表 11-69. IDEAL_BE_LOST_BW_DUE_TO_FE のユニット・マスク

拡張子	PMC.umask [19:16]	説明
ALL	b0000	原因に関わらずカウントする。
FEFLUSH	b0001	フロントエンド・フラッシュが発生原因である場合のみ
---	b0010	(* カウントは未定義 *)
---	b0011	(* 不正な選択 *)
UNREACHED	b0100	到達不可バンドルが発生原因である場合のみ
IBFULL	b0101	(* このイベントでは無効 *)
IMISS	b0110	命令キャッシュ・ミス・ストールが発生原因である場合のみ
TLBMISS	b0111	TLB ストールが発生原因である場合のみ
FILL_RECIRC	b1000	キャッシュ・ライン・フィル操作の再循環が発生原因である場合のみ
BI	b1001	分岐初期設定ストールが発生原因である場合のみ
BRQ	b1010	分岐リタイアメント・キュー・ストールが発生原因である場合のみ
PLP	b1011	完全なループ予測ストールが発生原因である場合のみ
BR_ILOCK	b1100	分岐インターロック・ストールが発生原因である場合のみ
BUBBLE	b1101	分岐リステア・バブル・ストールが発生原因である場合のみ
---	b1101-b1111	(* 不正な選択 *)

INST_CHKA_LDC_ALAT

- タイトル: リタイアした `chk.a` 命令および `ld.c` 命令
- カテゴリ: 命令実行 IAR/DAR/OPC: Y/Y/Y
- イベント・コード: 0x56、最大 Inc/Cyc: 2
- 定義: リタイアメントに達したすべてのアドバンスト・チェック・ロード (`chk.a`) 命令およびチェック・ロード (`ld.c`) 命令の数に関して情報を提供する。
- 注: 古いシプリングが失敗しても、失敗している `chk.a` はカウントされる。

表 11-70. INST_CHKA_LDC_ALAT のユニット・マスク

拡張子	PMC.umask [19:16]	説明
---	bxx00	(* 何もカウントされない *)
INT	bxx01	整数命令のみ
FP	bxx10	浮動小数点命令のみ
ALL	bxx11	整数命令と浮動小数点命令の両方

INST_DISPERSED

- タイトル: REN から REG に配布されたシラブルの数
- カテゴリ: 命令ディスパースル・イベント IAR/DAR/OPC: Y/N/N
- イベント・コード: 0x4d、最大 Inc/Cyc: 6
- 定義: ROTate から EXPand に配布されたシラブルの概算値を求めるため、REName から REGister パイプライン・ステージに配布されたシラブルの数をカウントする。

INST_FAILED_CHKA_LDC_ALAT

- タイトル: 失敗した `chk.a` 命令および `ld.c` 命令
- カテゴリ: 命令実行 IAR/DAR/OPC: Y/Y/Y
- イベント・コード: 0x57、最大 Inc/Cyc: 1
- 定義: リタイアメントに達した、失敗したアドバンスト・チェック・ロード (`chk.a`) 命令およびチェック・ロード (`ld.c`) 命令の数に関して情報を提供する。
- 注: 任意のある時点で、失敗した `chk.a` または `ld.c` が 2 つ存在する可能性があるが、そのときは先頭の命令のみがカウントされる。

表 11-71. INST_FAILED_CHKA_LDC_ALAT のユニット・マスク

拡張子	PMC.umask [19:16]	説明
---	bxx00	(* 何もカウントされない *)
INT	bxx01	整数命令のみ
FP	bxx10	浮動小数点命令のみ
ALL	bxx11	整数命令と浮動小数点命令の両方

INST_FAILED_CHKS_RETIRED

- タイトル: 失敗した `chk.s` 命令
- カテゴリ: 命令実行 IAR/DAR/OPC: N/N/N
- イベント・コード: 0x55、最大 Inc/Cyc: 1
- 定義: 失敗したスペキュレーティブ・チェック命令 (`chk.s`) の数に関して情報を提供する。

表 11-72. INST_FAILED_CHKS_RETIRED のユニット・マスク

拡張子	PMC.umask [19:16]	説明
---	bxx00	(* 何もカウントされない*)
INT	bxx01	整数命令のみ
FP	bxx10	浮動小数点命令のみ
ALL	bxx11	整数命令と浮動小数点命令の両方

ISB_BUNPAIRS_IN

- タイトル: L2 から FE に書き込まれたバンドル・ペア
- カテゴリ: L1 命令キャッシュおよびプリフェッチ IAR/DAR/OPC: Y/N/N
- イベント・コード: 0x46、最大 Inc/Cyc: 1
- 定義: L2 (およびそれ以降のキャッシュ) からフロントエンドに書き込まれたバンドル・ペア (32 バイト) の数に関して情報を提供する。
- 注: このイベントは、キャッシュ・ラインがデマンド・フェッチとしてタグ付けされていた場合は IBRP0 で修飾され、プリフェッチ・マッチとしてタグ付けされていた場合は IBRP1 で修飾される。

ITLB_MISSES_FETCH

- タイトル: 命令変換バッファ・ミス・デマンド・フェッチ
- カテゴリ: TLB IAR/DAR/OPC: Y/N/N
- イベント・コード: 0x47、最大 Inc/Cyc: 1
- 定義: デマンド・フェッチの ITLB ミスの数をカウントする。

表 11-73. ITLB_MISSES_FETCH のユニット・マスク

拡張子	PMC.umask [19:16]	説明
---	bxx00	(* 何もカウントされない*)
L1ITLB	bxx01	L1ITLB 内のミスがすべてカウントされる。L1ITLB は、アクセスでは更新されないが (キャッシュ不可 / nat ページ / 非表示ページ / 失敗 / いくつかフラッシュされた)、ここでカウントされる。
L2ITLB	bxx10	L1ITLB 内のすべてのミスの内、L2ITLB 内でもミスしているものがカウントされる。
ALL	bxx11	tlb ミスがすべてカウントされる。L1ITLB umask と L2ITLB umask の合計とは等しくないのに注意する。これは、L1ITLB と L2ITLB の中では、どのようなアクセスでもミスになり得るからである。

L1DTLB_TRANSFER

- タイトル: L1D_READS でカウントされたアクセスで、L2DTLB でヒットした L1DTLB ミス
- カテゴリ: TLB/L1D キャッシュ・セット 0 IAR/DAR/OPC: Y/Y/Y
- イベント・コード: 0xc0、最大 Inc/Cyc: 1
- 定義: L1D_READS でカウントされたアクセスで、L2DTLB でヒットした L1DTLB ミスの数をカウントする。
- 注: 制限付きのセット 0 L1D キャッシュ・イベントである。このイベントを計測するには、このセット内のイベントの 1 つを PMD5 で計測しなければならない。コード・シーケンス a;b で、"a" が例外を処理し、"b" が L2DTLB->L1DTLB への転送を要求する場合、この転送は実行されるが、このイベントではカウントされない。これは、到達していないために "b" をカウントしない L1D_READS との一貫性を保つためである。

L1D_READS_SET0

- タイトル: L1 データ・キャッシュ読み出し (セット 0)
- カテゴリ: L1 データ・キャッシュ /L1D キャッシュ・セット 0 IAR/DAR/OPC: Y/Y/Y
- イベント・コード: 0xc2、最大 Inc/Cyc: 2
- 定義: L1D (整数ロードのみ)、RSE ロード、L1 ヒント付きロード (L1D 内でヒットした場合、L1D はデータを返すがフィルは実行しない)、チェック・ロード (ld.c) によって処理される、メモリ・パイプラインに発行されたデータ・メモリ読み出し参照の数をカウントする。キャッシュ不可読み出し、VHPT ロード、セマフォ、浮動小数点ロード、lfetch 命令は、ここではカウントされない。L1D はこれらを処理しないからである。このカウントには、分岐パスの予測が外れた操作は含まれるが、プレディケート・オフの操作は含まれない。
- 注: 制限付きのセット 0 L1D キャッシュ・イベントである。このイベントを計測するには、このセット内のイベントの 1 つを PMD5 で計測しなければならない。計測されるのは、ポート 0 および 1 のみである。

L1D_READS_SET1

- タイトル: L1 データ・キャッシュ読み出し (セット 1)
- カテゴリ: L1 データ・キャッシュ /L1D キャッシュ・セット 1 IAR/DAR/OPC: Y/Y/Y
- イベント・コード: 0xc4、最大 Inc/Cyc: 2
- 定義: L1D (整数ロードのみ)、RSE ロード、L1 ヒント付きロード (L1D 内でヒットした場合、L1D はデータを返すがフィルは実行しない)、チェック・ロード (ld.c) によって処理される、メモリ・パイプラインに発行されたデータ・メモリ読み出し参照の数をカウントする。キャッシュ不可読み出し、VHPT ロード、セマフォ、浮動小数点ロード、lfetch 命令は、ここではカウントされない。L1D はこれらを処理しないからである。このカウントには、分岐パスの予測が外れた操作は含まれるが、プレディケート・オフの操作は含まれない。
- 注: 制限付きのセット 1 L1D キャッシュ・イベントである。このイベントを計測するには、このセット内のイベントの 1 つを PMD5 で計測しなければならない。計測されるのは、ポート 0 および 1 のみである。

L1D_READ_MISSES

- タイトル: L1 データ・キャッシュ読み出しミス
- カテゴリ: L1 データ・キャッシュ / L1D キャッシュ・セット 1 IAR/DAR/OPC: Y/Y/Y
- イベント・コード: 0xc7、最大 Inc/Cyc: 2
- 定義: L1 データ・キャッシュ読み出しミスの回数をカウントする。L1 データ・キャッシュはライトスルーである。したがって、書き込みミスはカウントされない。カウントには、L1D_READS イベントによりカウントされた参照に起因するミスのみが含まれる。これには、ALAT をミスした L1D ミスが含まれるが、ALAT でヒットした L1D ミスは含まれない。セマフォは L1D によって処理されず、このカウントには含まれない。
- 注: 制限付きのセット 1 L1D キャッシュ・イベントである。このイベントを計測するには、このセット内のイベントの 1 つを PMD5 で計測しなければならない。計測されるのは、ポート 0 および 1 のみである。

表 11-74. L1D_READ_MISSES のユニット・マスク

拡張子	PMC.umask [19:16]	説明
ALL	bxxx0	L1D 読み出しミスがすべてカウントされる。
RSE_FILL	bxxx1	RSE フィルが原因の L1D 読み出しミスのみがカウントされる。

L1ITLB_INSERTS_HPW

- タイトル: L1ITLB ハードウェア・ページ・ウォークの挿入
- カテゴリ: TLB IAR/DAR/OPC: Y/N/N
- イベント・コード: 0x48、最大 Inc/Cyc: 1
- 定義: ハードウェア・ページ・ウォークにより実行された L1ITLB の挿入回数をカウントする。

L1I_EAR_EVENTS

- タイトル: 命令 EAR イベント
- カテゴリ: L1 命令キャッシュおよびプリフェッチ IAR/DAR/OPC: Y/N/N
- イベント・コード: 0x43、最大 Inc/Cyc: 1
- 定義: EAR によって収集された L1 命令キャッシュまたは L1ITLB イベントの数をカウントする。

L1I_FETCH_ISB_HIT

- タイトル: ISB でヒットし ISB からバイパスされている「ジャストインタイム」命令フェッチ
- カテゴリ: L1 命令キャッシュおよびプリフェッチ IAR/DAR/OPC: Y/N/N
- イベント・コード: 0x66、最大 Inc/Cyc: 1
- 定義: ISB (命令ストリーミング・バッファ) でヒットし ISB からバイパスされている命令フェッチに関して情報を提供する。「クリティカル・バイパス」はカウントされない。すなわち、パイプラインは、データが L2 から提供されるのを待つ間、ストールする必要がある。L1I_FETCH_ISB_HIT は、フロントエンド・パイプをストールせずに命令を処理するために、命令データが L2 によって提供されるとき、「ジャストインタイム」バイパスをカウントする。
- 注: 命令キャッシュ (1 サイクル・ウィンドウ) に転送されるのと同時に ISB をヒットするデマンド・フェッチはカウントされない。これらのデマンド・フェッチが、分岐予測のためのキャッシュ・ヒットとして扱うからである。このイベントは、キャッシュ・ラインがデマンド・フェッチとしてタグ付けされていた場合は IBRP0 で修飾され、プリフェッチ・マッチとしてタグ付けされていた場合は IBRP1 で修飾される。

L1I_FETCH_RAB_HIT

- タイトル: RAB でヒットしている命令フェッチ
- カテゴリ: L1 命令キャッシュおよびプリフェッチ IAR/DAR/OPC: Y/N/N
- イベント・コード: 0x65、最大 Inc/Cyc: 1
- 定義: RAB でヒットしている命令フェッチに関して情報を提供する。
- 注: このイベントは、キャッシュ・ラインがデマンド・フェッチでタグ付けされていた場合は IBRP0 で修飾され、プリフェッチ・マッチでタグ付けされていた場合は IBRP1 で修飾される。

L1I_FILLS

- タイトル: L1 命令キャッシュ・フィル
- カテゴリ: L1 命令キャッシュおよびプリフェッチ IAR/DAR/OPC: Y/N/N
- イベント・コード: 0x41、最大 Inc/Cyc: 1
- 定義: ISB から L1 命令キャッシュ (64 バイト群) へのライン・フィルの数に関して情報を提供する。
- 注: このイベントは、キャッシュ・ラインがデマンド・フェッチでタグ付けされていた場合は IBRP0 で修飾され、プリフェッチ・マッチでタグ付けされていた場合は IBRP1 で修飾される。対応するエントリが L1ITLB 内にない場合、このイベントを発生させることはできない。

L1I_PREFETCHES

- タイトル: L1 命令プリフェッチ要求
- カテゴリ: L1 命令キャッシュおよびプリフェッチ IAR/DAR/OPC: Y/N/N
- イベント・コード: 0x44、最大 Inc/Cyc: 1
- 定義: 発行された L1 キャッシュ・ライン・プリフェッチ要求 (64 バイト / ライン) の数に関して情報を提供する。報告される回数には、ストリーミング・プリフェッチと非ストリーミング・プリフェッチが含まれる (L1 命令キャッシュ内のヒットとミスは両方含まれる)。
- 注: このイベントは、IBRP1 で修飾される。

L1I_PREFETCH_STALL

- タイトル: プリフェッチ・パイプライン・ストール
- カテゴリ: L1 命令キャッシュおよびプリフェッチ IAR/DAR/OPC: N/N/N
- イベント・コード: 0x67、最大 Inc/Cyc: 1
- 定義: プリフェッチ・パイプラインがストールされた原因について情報を提供する。

表 11-75. L1I_PREFETCH_STALL のユニット・マスク

拡張子	PMC.umask [19:16]	説明
---	bxx00-bxx01	(* 何もカウントされない *)
FLOW	bxx10	フローがアサートされないクロック数
ALL	bxx11	プリフェッチ・パイプラインがストールされるクロック数

L1I_PURGE

- タイトル: L1 命令によって処理される L1ITLB パージ
- カテゴリ: L1 命令キャッシュおよびプリフェッチ IAR/DAR/OPC: Y/N/N
- イベント・コード: 0x4b、最大 Inc/Cyc: 1
- 定義: L1 命令によって処理される L1ITLB パージの数に関して情報を提供する。このイベントは、パージ命令、バス・クラスタからのグローバル・パージ、L2ITLB への挿入が原因で発生する。これは、L1ITLB 上で実行される列の無効化と同じではない。

L1I_PVAB_OVERFLOW

- タイトル: PVAB オーバーフロー
- カテゴリ: L1 命令キャッシュおよびプリフェッチ IAR/DAR/OPC: N/N/N
- イベント・コード: 0x69、最大 Inc/Cyc: 1
- 定義: プリフェッチ仮想アドレス・バッファ・オーバーフローに関して情報を提供する。

L1I_RAB_ALMOST_FULL

- タイトル: RAB はほぼいっぱいか?
- カテゴリ: L1 命令キャッシュおよびプリフェッチ IAR/DAR/OPC: N/N/N
- イベント・コード: 0x64、最大 Inc/Cyc: 1
- 定義: ほぼいっぱいである読み出しアドレス・バッファに関して情報を提供する。

L1I_RAB_FULL

- タイトル: RAB はいっぱいか?
- カテゴリ: L1 命令キャッシュおよびプリフェッチ IAR/DAR/OPC: N/N/N
- イベント・コード: 0x60、最大 Inc/Cyc: 1
- 定義: いっぱいである読み出しアドレス・バッファに関して情報を提供する。

L1I_READS

- タイトル: L1 命令キャッシュ読み出し
- カテゴリ: L1 命令キャッシュおよびプリフェッチ IAR/DAR/OPC: Y/N/N
- イベント・コード: 0x40、最大 Inc/Cyc: 1
- 定義: L1 命令キャッシュ (32 バイト群) に対するデマンド・フェッチ読み出しの回数 (ヒット / ミスとは関係なく、すべてのアクセス) について情報を提供する。
- 注: L1ITLB ミスと L1 命令キャッシュ・ミスを持ち、命令キャッシュに対するフィル再循環と競合するデマンド・フェッチは、L2_INST_DEMAND_READS ではカウントされるが、このイベントではカウントされない。

L1I_SNOOP

- タイトル: L1 命令によって処理されるスヌープ要求
- カテゴリ: L1 命令キャッシュおよびプリフェッチ IAR/DAR/OPC: Y/Y/Y
- イベント・コード: 0x4a, 最大 Inc/Cyc: 1
- 定義: L1 命令によって処理されるスヌープ要求 (64 バイト単位) の数について情報を提供する。
- 注: 各 "fc" 命令は、バスに送出後、L1 命令に対するスヌープ要求を 1 つ生成する。各 IA32 ストアは、L1 命令に対してスヌープ要求を 1 つ生成するが、ここでは、L1D 内で再循環されるのと同じ回数だけカウントされる。これは、より重要な処理のためにビジー状態になっているからである。L1D が IFR にスヌープを送信したときに、IFR スヌープ・パイプラインがビジーである場合、このイベントは、同じスヌープについて複数カウントを行う。ビクティムにされたラインもスヌープを生成する。バス・トランザクションによっては、L1 命令スヌープを発生させるものもある。

L1I_STRM_PREFETCHES

- タイトル: L1 命令キャッシュ・ライン・プリフェッチ要求
- カテゴリ: L1 命令キャッシュおよびプリフェッチ IAR/DAR/OPC: Y/N/N
- イベント・コード: 0x5f, 最大 Inc/Cyc: 1
- 定義: ストリーミング・モードのみでプリフェッチ・パイプラインを通過する、64 バイト / ラインの L1 命令キャッシュ・ライン・プリフェッチ要求の数について情報を提供する (この場合、L1 命令キャッシュ内のヒットまたはミスは考慮されない。また、ストリーミング・モードは、br.many により開始される)。
- 注: このイベントは、IBRP1 で修飾される。

L2_BAD_LINES_SELECTED

- タイトル: 無効なラインが使用可能なときに置き換えられた有効なライン
- カテゴリ: L2 ユニファイド・キャッシュ / L2 キャッシュ・セット 3 IAR/DAR/OPC: Y/Y/Y
- イベント・コード: 0xb9, 最大 Inc/Cyc: 4
- 定義: 無効なラインが使用可能なときに、置き換えのために有効なラインが選択された回数をカウントする。
- 注: 制限付きのセット 3 L2 キャッシュ・イベントである。このイベントを計測するには、このセット内のイベントの 1 つを PMD4 で計測しなければならない。イベント・コードは、L2_ISSUED_RECIRC_IFETCH と共用する。

表 11-76. L2_BAD_LINES_SELECTED のユニット・マスク

拡張子	PMC.umask [19:16]	説明
ANY	b0xxx	無効なラインが使用可能なときに置き換えられた有効なライン

L2_BYPASS

- タイトル: L2 バイパスをカウントする
- カテゴリ: L2 ユニファイド・キャッシュ / L2 キャッシュ・セット 3 IAR/DAR/OPC: Y/Y/Y
- イベント・コード: 0xb8, 最大 Inc/Cyc: 1
- 定義: バイパスが発生した回数をカウントする。
- 注: 制限付きのセット 3 L2 キャッシュ・イベントである。このイベントを計測するには、このセット内のイベントの 1 つを PMD4 で計測しなければならない。イベント・コードは、L2_OPS_ISSUED と共用する。

表 11-77. L2_BYPASS のユニット・マスク

拡張子	PMC.umask [19:16]	説明
L2_DATA1	b0000	(L1D から L2A への) L2 データ・バイパスのみをカウントする。
L2_DATA2	b0001	(L1W から L2I への) L2 データ・バイパスのみをカウントする。
L3_DATA1	b0010	(L1D から L2A への) L3 データ・バイパスのみをカウントする。
---	b0011	(* 何もカウントされない *)
L2_INST1	b0100	(L1D から L2A への) L2 命令バイパスのみをカウントする。
L2_INST2	b0101	(L1W から L2I への) L2 命令バイパスのみをカウントする。
L3_INST1	b0110	(L1D から L2A への) L3 命令バイパスのみをカウントする。
---	b0111	(* 何もカウントされない *)

L2_DATA_REFERENCES

- タイトル: L2 へのデータ読み出し / 書き込みアクセス
- カテゴリ: L2 ユニファイド・キャッシュ / L2 キャッシュ・セット 1 IAR/DAR/OPC: Y/Y/Y
- イベント・コード: 0xb2、最大 Inc/Cyc: 4
- 定義: データ読み出しアクセスまたはデータ書き込みアクセスが原因で L2 に発行された要求の数をカウントする。報告されるカウントは、キャッシュ・ラインをマージする前の要求の数である。セマフォ操作は、1つの読み出しおよび1つの書き込みとしてカウントされる。
- 注: 制限付きのセット 1 L2 キャッシュ・イベントである。このイベントを計測するには、このセット内のイベントの1つを PMD4 で計測しなければならない。

表 11-78. L2_DATA_REFERENCES のユニット・マスク

拡張子	PMC.umask [19:16]	説明
---	bxx00	(* 何もカウントされない *)
L2_DATA_READS	bxx01	データ読み出し操作およびセマフォ操作のみカウントする。
L2_DATA_WRITES	bxx10	データ書き込み操作およびセマフォ操作のみカウントする。
L2_ALL	bxx11	読み出し操作と書き込み操作の両方をカウントする (セマフォは 2 としてカウントされる)。

L2DTLB_MISSES

- タイトル: L2DTLB ミス
- カテゴリ: TLB/L1D キャッシュ・セット 0 IAR/DAR/OPC: Y/Y/Y
- イベント・コード: 0xc1、最大 Inc/Cyc: 4
- 定義: デマンド要求に対する L2DTLB ミスの数をカウントする (これは、HPW への参照; DTLB_HIT=0 と同じである)。
- 注: 制限付きのセット 0 L1D キャッシュ・イベントである。このイベントを計測するには、このセット内のイベントの1つを PMD5 で計測しなければならない。HPW が常時イネーブルの場合、このイベントと HPW_DATA_REFERENCES は同じである。これには、ミスを引き起こしている命令がリタイアしていないのに L2DTLB が実行されたミスが含まれる。

L2_FILLB_FULL

- タイトル: L2 フィル・バッファがいっぱい
- カテゴリ: L2 ユニファイド・キャッシュ IAR/DAR/OPC: N/N/N
- イベント・コード: 0xbf, 最大 Inc/Cyc: 1
- 定義: L2 フィル・バッファがいっぱいになった回数をカウントする。
- 注: 制限付きのセット 5 L2 キャッシュ・イベントである。このイベントを計測するには、このセット内のイベントの 1 つを PMD4 で計測しなければならない。

表 11-79. L2_FILLB_FULL のユニット・マスク

拡張子	PMC.umask [19:16]	説明
THIS	b0000	L2 フィル・バッファがいっぱい
---	b0001-b1111	(* カウントは未定義 *)

L2_FORCE_RECIRC

- タイトル: 強制された再循環
- カテゴリ: L2 ユニファイド・キャッシュ / L2 キャッシュ・セット 2 IAR/DAR/OPC: Y/Y/Y
- イベント・コード: 0xb4, 最大 Inc/Cyc: 4
- 定義: 強制的に再循環された L2 op の数をカウントする (SNP_OR_L3 は除く)。SNP_OR_L3 は、L2 op が強制的に再循環される回数を計測する。0-32 op からのいずれの場所も、これにより影響を受ける可能性がある。SMC_HIT、TRAN_PERF、SNP_OR_L3 を除くカテゴリはすべて、OZQ への挿入時に発生する。SMC_HIT は、命令フェッチが IPFQ に書き込まれようとしているときであり、同じアドレスに対して未処理のストアが存在するために強制的に再循環される。SNP_OR_L3 は、既存の OZQ エントリが強制的に再循環させられるときである。これは、着信要求がそのアドレスに一致したか、またはこの OZQ_ENTRY が「ヒット」した同じウェイ/インデックスをフィルする L3/BC にアクセスが発行されるからである。TRAN_PREF は、既存の OZQ アクセスがプリフェッチに変換されるときである。
- 注: 制限付きのセット 2 L2 キャッシュ・イベントである。このイベントは、PMD4 で計測しなければならない。

表 11-80. L2_FORCE_RECIRC のユニット・マスク

拡張子	PMC.umask [19:16]	説明
ANY	b0000	原因に関わらず、強制された再循環をカウントする。ここでは、SMC_HIT、TRAN_PREF、SNP_OR_L3 は含まれない。
SMC_HIT	b0001	同じキャッシュ・ラインまたは未処理の WT ストアに対する命令フェッチおよびロードが原因の SMC ヒットにより生じたもののみをカウントする。
L1W	b0010	強制された limbo により生じたもののみをカウントする。
---	b0011	(* 何もカウントされない *)
TAG_NOTOK	b0100	同じインデックスに対するシプリング・ミス、同じラインに対するシプリング・プローブ、または未処理の sync.ia 命令を持つインフライト・スヌープ、ストアが原因で発生した L2 ヒットにより生じたもののみをカウントする。
TRAN_PREF	b0101	プリフェッチに対する変換により生じたもののみをカウントする。

表 11-80. L2_FORCE_RECIRC のユニット・マスク (続き)

拡張子	PMC.umask [19:16]	説明
SNP_OR_L3	b0110	スヌープまたは L3 発行により生じたもののみをカウントする。
---	b0111	(* 何もカウントされない *)
VIC_PEND	b1000	未処理のビクティムを持つ L2 ミスにより生じたもののみをカウントする。
FILL_HIT	b1001	フィル・バッファでヒットした L2 ミスにより生じたもののみをカウントする。
IPF_MISS	b1010	命令プリフェッチ・バッファ・ミスがすでに存在していたときに L2 ミスにより生じる。
VIC_BUF_FULL	b1011	ビクティム・バッファがいっぱいの L2 ミスにより生じたもののみをカウントする。
OZQ_MISS	b1100	OZQ ミスがすでに存在していたときに L2 ミスにより生じる。
SAME_INDEX	b1101	同じインデックスに対するミスがすでに存在していたときに L2 ミスにより生じる。
FRC_RECIRC	b1110	強制再循環がすでに存在していたときに L2 ミスにより生じる。
---	b1111	(* 何もカウントされない *)

L2_GOT_RECIRC_IFETCH

- タイトル: L2 が受け取る命令フェッチ再循環
- カテゴリ: L2 ユニファイド・キャッシュ /L2 キャッシュ・セット 4 IAR/DAR/OPC: Y/Y/Y
- イベント・コード: 0xba、最大 Inc/Cyc: 1
- 定義: L2 が受け取る命令フェッチ再循環の数をカウントする。
- 注: 制限付きのセット 4 L2 キャッシュ・イベントである。このイベントを計測するには、このセット内のイベントの 1 つを PMD4 で計測しなければならない。イベント・コードは、L2_STORE_HIT_SHARED と共用する。

表 11-81. L2_GOT_RECIRC_IFETCH のユニット・マスク

拡張子	PMC.umask [19:16]	説明
ANY	b1xxx	L2 が受け取った命令フェッチ再循環

L2_GOT_RECIRC_OZQ_ACC

- タイトル: L1D に再循環された OZQ アクセスをカウントする
- カテゴリ: L2 ユニファイド・キャッシュ IAR/DAR/OPC: Y/Y/Y
- イベント・コード: 0xb6、最大 Inc/Cyc: 1
- 定義: 正常に L1D に再循環された OZQ アクセスの数をカウントする。
- 注: 制限付きのセット 2 L2 キャッシュ・イベントである。このイベントを計測するには、このセット内のイベントの 1 つを PMD4 で計測しなければならない。

L2_IFET_CANCELS

- タイトル: L2 による命令フェッチ・キャンセル
- カテゴリ: L2 ユニファイド・キャッシュ / L2 キャッシュ・セット 0 IAR/DAR/OPC: Y/Y/Y
- イベント・コード: 0xa1、0xa5、0xa9、0xad、最大 Inc/Cyc: 1
- 定義: L2 による命令フェッチ・キャンセルの合計をカウントする。
- 注: 制限付きのセット 0 L2 キャッシュ・イベントである。このイベントを計測するには、L2_OZQ_CANCEL イベントまたはこのイベントの 1 つを PMD4 で計測しなければならない。

表 11-82. L2_IFET_CANCELS のユニット・マスク

拡張子	PMC.umask [19:16]	説明
ANY	b000x	L2 によりキャンセルされた命令フェッチの合計
BYPASS	b001x	バイパスが原因の命令フェッチのキャンセル
DIDNT_RECIR	b0100	再循環しなかったことが原因の命令フェッチのキャンセル
RECIR_OVER_SUB	b0101	再循環オーバーサブスクリプションが原因の命令フェッチのキャンセル
ST_FILL_WB	b0110	ストア、フィル、またはライトバックが原因の命令フェッチのキャンセル
DATA_RD	b0111	データ読み出しが原因の命令フェッチ / プリフェッチのキャンセル
PREEMPT	b10xx	先読みが原因の命令フェッチのキャンセル
CHG_PRIO	b1100	優先順位の変更が原因の命令フェッチのキャンセル
IFETCH_BYP	b1101	最終クロック時の命令フェッチ・バイパスが原因
---	b1110-b1111	(* 何もカウントされない *)

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: L1 命令キャッシュおよびプリフェッチ IAR/DAR/OPC: Y/N/N
- イベント・コード: 0x42、最大 Inc/Cyc: 1
- 定義: L1 命令のデマンド・フェッチ・ミスによる L2 命令要求の数をカウントする。このイベントは、L1 命令と ISB の両方でミスしたデマンド・フェッチの数をカウントする。この場合、デマンド・フェッチが RAB 内でヒットまたはミスしたかどうかは関係ない。
- 注: デマンド・フェッチが L1ITLB ミスを持たない場合、L2_INST_DEMAND_READS と L1I_READS は遅れずに整列する。デマンド・フェッチが L2ITLB ミスを持たない場合、L2_INST_DEMAND_READS は、3 ~ 4 クロックだけ L1I_READS より遅れる (フラッシュされた命令ウォークがその前で未処理でない場合に限る。未処理であると、未処理の命令ウォークが完了するまでの遅延が増加する)。デマンド・フェッチが L2ITLB ミスを持つ場合、L2_INST_DEMAND_READS と L1I_READS の間のスキューは決定できない。

L2_INST_PREFETCHES

- タイトル: L2 命令プリフェッチ要求
- カテゴリ: L1 命令キャッシュおよびプリフェッチ IAR/DAR/OPC: Y/N/N
- イベント・コード: 0x45、最大 Inc/Cyc: 1
- 定義: ユニファイド L2 キャッシュに発行されたプリフェッチ要求の数について情報を提供する。報告される数には、ストリーミング・プリフェッチおよび非ストリーミング・プリフェッチが含まれる。
- 注: このイベントは、IBRP1 で修飾される。

L2_ISSUED_RECIRC_IFETCH

- タイトル: L2 が発行する命令フェッチ再循環
- カテゴリ: L2 ユニファイド・キャッシュ /L2 キャッシュ・セット 4 IAR/DAR/OPC: Y/Y/Y
- イベント・コード: 0xb9、最大 Inc/Cyc: 1
- 定義: L2 が発行する命令フェッチ再循環の数をカウントする。
- 注: 制限付きのセット 4 L2 キャッシュ・イベントである。このイベントを計測するには、このセット内のイベントの 1 つを PMD4 で計測しなければならない。イベント・コードは、L2_BAD_LINES_SELECTED と共用する。

表 11-83. L2_ISSUED_RECIRC_IFETCH のユニット・マスク

拡張子	PMC.umask [19:16]	説明
ANY	b1xxx	L2 が発行した命令フェッチ再循環

L2_ISSUED_RECIRC_OZQ_ACC

- タイトル: 再循環発行を試行したが、先読みされなかった回数をカウントする
- カテゴリ: L2 ユニファイド・キャッシュ IAR/DAR/OPC: Y/Y/Y
- イベント・コード: 0xb5、最大 Inc/Cyc: 1
- 定義: 再循環を試行したが、fill/confirm/evervalid (fill/confirm タグ・アップデートはより高い優先順位を持つ) または再循環を発行している古いシプリング (1 クロックあたりに送信できる再循環は 1 つのみ) によって先読みされなかった回数をカウントする。この値は、L2 発行ロジックが再循環の発行を試行した合計回数の L2_OZQ_CANCELLED.DIDNT_RECIRC に追加できる。
- 注: 制限付きのセット 2 L2 キャッシュ・イベントである。このイベントを計測するには、このセット内のイベントの 1 つを PMD4 で計測しなければならない。

L2_L3ACCESS_CANCEL

- タイトル: キャンセルされた L3 アクセス
- カテゴリ: L2 ユニファイド・キャッシュ /L2 キャッシュ・セット 1 IAR/DAR/OPC: Y/Y/Y
- イベント・コード: 0xb0、最大 Inc/Cyc: 1
- 定義: キャンセルされた L3 アクセスの数をカウントする。以下の表に示すユニット・マスクにより、このイベントのキャンセルされた理由が特定される。
- 注: 制限付きのセット 1 L2 キャッシュ・イベントである。このイベントは、PMD4 で計測しなければならない。

表 11-84. L2_L3ACCESS_CANCEL のユニット・マスク

拡張子	PMC.umask [19:16]	説明
---	b0000	(* 何もカウントされない *)
SPEC_L3_BYP	b0001	スペキュレーティブ L3 バイパス
FILLD_FULLL	b0010	filld がいっぱい
---	b0011	(* カウントは未定義 *)
---	b0100	(* 何もカウントされない *)
UC_BLOCKED	b0101	キャッシュ不可のブロックされた L3 アクセス
INV_L3_BYP	b0110	無効な L3 バイパス
EBL_REJECT	b1000	ebl の拒否
ANY	b1001	理由のいかんに関わらず、キャンセルをすべてカウントする。この umask は、その他のすべての umask の合計より多くカウントする。この umask が、アクセスがコミットされていないものをカウントするのは、それらのアクセスが L1w に達したが、L2 が何らかの方法で (スペキュレーティブに) それらのアクセスを L3 にバイパスしようとしたときである。これには、メイン・パイプラインがストールされ、L1d がアクセスを L1d パイプラインに再循環しようとしているときに繰り返し実行されるアクセスが含まれる。したがって、L3 に実際にバイパスされる前に、アクセスが何度もカウントされる可能性がある。これは、要求を L3 にアサートしたがそれを確認しなかった回数である。
DFETCH	b1010	データ・フェッチ
IFETCH	b1011	命令フェッチ
---	b1100-b1111	(* 何もカウントされない *)

L2_MISSES

- タイトル: L2 ミス
- カテゴリ: L2 ユニファイド・キャッシュ IAR/DAR/OPC: Y/Y/Y
- イベント・コード: 0xcb, 最大 Inc/Cyc: 1
- 定義: L2 キャッシュ・ミスの数をカウントする (L3 に送信された L2 キャッシュ・ライン要求の数として)。これには、命令のフェッチ / プリフェッチおよびデータの読み出し / 書き込み操作に起因するミスが含まれる。キャッシュ不可またはライト・コアレッシング・アドレスに対する L1 ミスは含まれない。

L2_OPS_ISSUED

- タイトル: L2 により発行された操作
- カテゴリ: L2 ユニファイド・キャッシュ / L2 キャッシュ・セット 4 IAR/DAR/OPC: Y/Y/Y
- イベント・コード: 0xb8, 最大 Inc/Cyc: 4
- 定義: 操作タイプによって指定された、L2 により発行された操作の数をカウントする (すなわち、L2 パイプ・ステージで有効であった操作。そのため、これらの操作は、後でキャンセルされるとしてもカウントされる。これは、L2 でヒットする操作についてのみ発生する。つまり、OzQ がそれらを処理する)。
- 注: 制限付きのセット 4 L2 キャッシュ・イベントである。このイベントを計測するには、このセット内のイベントの 1 つを PMD4 で計測しなければならない。イベント・コードは、L2_BYPASS と共用する。

表 11-85. L2_OPS_ISSUED のユニット・マスク

拡張子	PMC.umask [19:16]	説明
INT_LOAD	b1000	有効な整数ロードのみをカウントする。
FP_LOAD	b1001	有効な浮動小数点ロードのみをカウントする。
RMW	b1010	有効な read_modify_write ストアのみをカウントする。
STORE	b1011	有効な非 read_modify_write ストアのみをカウントする。
NST_NLD	b1100	有効な非ロード、非ストア・アクセスのみをカウントする。
---	b1101-b1111	(* 何もカウントされない *)

L2_OZDB_FULL

- タイトル: L2 OZ データ・バッファがいっぱい
- カテゴリ: L2 ユニファイド・キャッシュ /L2 キャッシュ・セット 5 IAR/DAR/OPC: N/N/N
- イベント・コード: 0xbd、最大 Inc/Cyc: 1
- 定義: L2 Oz データ・バッファがいっぱいになった回数をカウントする。
- 注: 制限付きのセット 5 L2 キャッシュ・イベントである。このイベントを計測するには、このセット内のイベントの 1 つを PMD4 で計測しなければならない。

表 11-86. L2_OZDB_FULL のユニット・マスク

拡張子	PMC.umask [19:16]	説明
THIS	b0000	L2 OZ データ・バッファがいっぱい
---	b0001-b1111	(* カウントは未定義 *)

L2_OZQ_ACQUIRE

- タイトル: L2 OZQ 内に存在する獲得順序属性に関するクロック
- カテゴリ: L2 ユニファイド・キャッシュ /L2 キャッシュ・セット 0 IAR/DAR/OPC: N/N/N
- イベント・コード: 0xa2,0xa6,0xaa,0xae、最大 Inc/Cyc: 1
- 定義: L2 OZ キュー内に存在する「獲得」順序属性に関するクロック・エントリの数をカウントする。
- 注: 制限付きのセット 0 L2 キャッシュ・イベントである。このイベントを計測するには、このセット内のイベントの 1 つを PMD4 で計測しなければならない。

L2_OZQ_CANCELSD

- タイトル: L2 OZQ キャンセル (Late または Any)
- カテゴリ: L2 ユニファイド・キャッシュ /L2 キャッシュ・セット 0 IAR/DAR/OPC: Y/Y/Y
- イベント・コード: 0xa0、最大 Inc/Cyc: 4
- 定義: L2 OZ キューのキャンセル (理由を問わない) または特定の理由による L2 OZ キューのキャンセル (umask に基づく) の合計をカウントする。
- 注: 制限付きのセット 0 L2 キャッシュ・イベントである。一度に計測できるのは、3 つある L1_OZQ_CANCEL イベントの 1 つだけである。このイベントを計測するには、L2_IFET_CANCELSD またはこのイベントを PMD4 で計測しなければならない。

表 11-87. L2_OZQ_CANCEL0 のユニット・マスク

拡張子	PMC.umask [19:16]	説明
ANY	bx000	OZ キューのキャンセル数の合計をカウントする。
LATE_SPEC_BYP	bx001	スペキュレーティブなバイパスに起因する late キャンセル数をカウントする。
LATE_RELEASE	bx010	解放に起因する late キャンセル数をカウントする。
LATE_ACQUIRE	bx011	獲得に起因する late キャンセル数をカウントする。
LATE_BYP_EFFRELEASE	bx100	L1D から L2A バイパスへの効果的な解放に起因する late キャンセル数をカウントする。
---	bx101-bx111	(* 何もカウントされない *)

L2_OZQ_CANCEL1

- タイトル: L2 OZQ のキャンセル (特定の理由セット 1)
- カテゴリ: L2 ユニファイド・キャッシュ / L2 キャッシュ・セット 0 IAR/DAR/OPC: Y/Y/Y
- イベント・コード: 0xac、最大 Inc/Cyc: 4
- 定義: (umask に基づく) 特定の理由が原因である L2 OZ キューのキャンセル数の合計をカウントする。
- 注: 制限付きのセット 0 L2 キャッシュ・イベントである。一度に計測できるのは、3 つある L1_OZQ_CANCEL イベントの 1 つだけである。このイベントを計測するには、L2_IFET_CANCEL またはこのイベントを PMD4 で計測しなければならない。

表 11-88. L2_OZQ_CANCEL1 のユニット・マスク

拡張子	PMC.umask [19:16]	説明
REL	b0000	解放により発生する。
BANK_CONF	b0001	バンクの競合
L2D_ST_MAT	b0010	L2D 内でのストア・マッチ
---	b0011	(* 何もカウントされない *)
SYNC	b0100	sync.i により発生する。
HPW_IFETCH_CONF	b0101	命令フェッチの競合 (HPW をキャンセルしたか?)
CANC_L2M_ST	b0110	L2M 内のキャンセルされたストアにより発生する。
L1_FILL_CONF	b0111	L1 フィルの競合
ST_FILL_CONF	b1000	ストア・フィルの競合
CCV	b1001	ccv
SEM	b1010	セマフォ
L2M_ST_MAT	b1011	L2M 内でのストア・マッチ
MFA	b1100	メモリ・フェンス命令
L2A_ST_MAT	b1101	L2A 内でのストア・マッチ
L1DF_L2M	b1110	L2M 内の L1D フィル
ECC	b1111	問題を検出している ECC ハードウェア

L2_OZQ_CANCELS2

- タイトル: L2 OZQ のキャンセル (特定の理由セット 2)
- カテゴリ: L2 ユニファイド・キャッシュ /L2 キャッシュ・セット 0 IAR/DAR/OPC: Y/Y/Y
- イベント・コード: 0xa8、最大 Inc/Cyc: 4
- 定義: (umask に基づく) 特定の理由が原因である L2 OZ キューの合計をカウントする。
- 注: 制限付きのセット 0 L2 キャッシュ・イベントである。一度に計測できるのは、3 つある L1_OZQ_CANCEL イベントの 1 つだけである。このイベントを計測するには、L2_IFET_CANCELS またはこのイベントを PMD4 で計測しなければならない。

表 11-89. L2_OZQ_CANCELS2 のユニット・マスク

拡張子	PMC.umask [19:16]	説明
RECIRC_OVER_SUB	b0000	再循環オーバーサブスクリプションにより発生する。
CANC_L2C_ST	b0001	L2C 内のキャンセルされたストアにより発生する。
L2C_ST_MAT	b0010	L2C 内でのストア・マッチ
SCRUB	b0011	スクラビングが必要な 32/64 バイト HPW/L2D フィル
ACQ	b0100	獲得により発生する。
READ_WB_CONF	b0101	ライトバックの競合 (読み出しはキャンセルしたか ?)
OZ_DATA_CONF	b0110	OZ データの競合
---	b0111	(* 何もカウントされない *)
L2FILL_ST_CONF	b1000	L2C 内の L2fill およびストアの競合
DIDNT_RECIRC	b1001	再循環しなかったことが原因で発生する。
WEIRD	b1010	再循環を長時間ブロックしているアライメントの合っていないアクセスおよびバイパスに対して 5 サイクルのバイパスの試行により発生したキャンセルをカウントする。
---	b1011	(* 何もカウントされない *)
OVER_SUB	b1100	オーバーサブスクリプション
CANC_L2D_ST	b1101	L2D 内のキャンセルされたストアにより発生する。
---	b1110	(* 何もカウントされない *)
D_IFET	b1111	デマンド命令フェッチ

L2_OZQ_FULL

- タイトル: L2 OZQ がいっぱい
- カテゴリ: L2 ユニファイド・キャッシュ /L2 キャッシュ・セット 5 IAR/DAR/OPC: N/N/N
- イベント・コード: 0xbc、最大 Inc/Cyc: 1
- 定義: L2D Oz キューがいっぱいになった回数をカウントする。
- 注: 制限付きのセット 5 L2 キャッシュ・イベントである。このイベントを計測するには、このセット内のイベントの 1 つを PMD4 で計測しなければならない。

表 11-90. L2_OZQ_FULL のユニット・マスク

拡張子	PMC.umask [19:16]	説明
THIS	b0000	L2D OZQ がいっぱい
---	b0001-b1111	(* カウントは未定義 *)

L2_OZQ_RELEASE

- タイトル: L2 OZQ 内に存在する解放順序属性に関するクロック
- カテゴリ: L2 ユニファイド・キャッシュ / L2 キャッシュ・セット 0 IAR/DAR/OPC: N/N/N
- イベント・コード: 0xa3、0xa7、0xab、0xaf、最大 Inc/Cyc: 1
- 定義: L2 OZ キュー内に存在する「解放」順序属性に関するクロック・エントリの数をカウントする。
- 注: 制限付きのセット 0 L2 キャッシュ・イベントである。このイベントを計測するには、このセット内のイベントの 1 つを PMD4 で計測しなければならない。

L2_REFERENCES

- タイトル: L2 に発行された要求
- カテゴリ: L2 ユニファイド・キャッシュ / L2 キャッシュ・セット 1 IAR/DAR/OPC: Y/Y/Y
- イベント・コード: 0xb1、最大 Inc/Cyc: 4
- 定義: L2 から発行された要求 (データ読み出し、データ書き込み、命令フェッチ、命令プリフェッチ) の数をカウントする。
- 注: 制限付きのセット 1 L2 キャッシュ・イベントである。このイベントを計測するには、このセット内のイベントの 1 つを PMD4 で計測しなければならない。フェッチにプロモートされるプリフェッチは、1 度だけカウントされる。前半のフェッチがすでにカウントされている場合、ラインの後半の命令フェッチはカウントされない。データ参照のため、2 次的なミスはすべてカウントされる。セマフォ操作は、ここで一度だけカウントされる。ここでは OZQ に入力される要求のみがカウントされる。すなわち、再循環操作は、再カウントされない。キャッシュ不可 / WC アクセスはカウントされない。FROM_CCV、SETF、CCV、PTC_G、PTC_GA、FWB、MF、MFA、SYNCI、SYNCIA、PTCM、FC、CC の各操作は除く。

L2_STORE_HIT_SHARED

- タイトル: ストアが共用ラインをヒットした
- カテゴリ: L2 ユニファイド・キャッシュ / L2 キャッシュ・セット 3 IAR/DAR/OPC: Y/Y/Y
- イベント・コード: 0xba、最大 Inc/Cyc: 2
- 定義: ストアが共用ラインをヒットした回数をカウントする。
- 注: 制限付きのセット 3 L2 キャッシュ・イベントである。このイベントを計測するには、このセット内のイベントの 1 つを PMD4 で計測しなければならない。イベント・コードは、L2_GOT_RECIRC_IFETCH と共用する。

表 11-91. L2_STORE_HIT_SHARED のユニット・マスク

拡張子	PMC.umask [19:16]	説明
ANY	b0xxx	ストアが共用ラインをヒットした。

L2_SYNTH_PROBE

- タイトル: 合成されたプローブ
- カテゴリ: L2 ユニファイド・キャッシュ IAR/DAR/OPC: Y/Y/Y
- イベント・コード: 0xb7、最大 Inc/Cyc: 1
- 定義: 合成されたプローブの数をカウントする。合成されたプローブは、インフライト・スヌープによりフィルがヒットされたことを示すIまたはPのMESIステートを持つバス・クラスタから、L2がフィルを受け取ったことを示す。そのようなものとして、L2は、ラインが「一度使用」されたら、バス・クラスタにプローブ応答を「合成」する必要がある。順方向の進行について、L2はラインを一度使用するまでは、応答を送信しない。
- 注: 制限付きのセット2 L2 キャッシュ・イベントである。このイベントを計測するには、このセット内のイベントの1つをPMD4で計測しなければならない。

L2_VICTIMB_FULL

- タイトル: L2D ビクティム・バッファがいっぱい
- カテゴリ: L2 ユニファイド・キャッシュ /L2 キャッシュ・セット5 IAR/DAR/OPC: N/N/N
- イベント・コード: 0xbe、最大 Inc/Cyc: 1
- 定義: L2D ビクティム・バッファがいっぱいになった回数をカウントする。
- 注: 制限付きのセット5 L2 キャッシュ・イベントである。このイベントを計測するには、このセット内のイベントの1つをPMD4で計測しなければならない。

表 11-92. L2_VICTIMB_FULL のユニット・マスク

拡張子	PMC.umask [19:16]	説明
THIS	b0000	L2D ビクティム・バッファがいっぱい
---	b0001-b1111	(* カウントは未定義 *)

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 ユニファイド・キャッシュ IAR/DAR/OPC: N/N/N
- イベント・コード: 0xdf、最大 Inc/Cyc: 1
- 定義: 置き換えられた有効な L3 ライン (ダーティ・ビクティム) の数をカウントする。プラットフォーム固有の設定に従って、排他的なクリーン / 共用やクリーン・キャストアウトもカウントされる場合がある。

L3_MISSES

- タイトル: L3 ミス
- カテゴリ: L3 ユニファイド・キャッシュ IAR/DAR/OPC: Y/Y/Y
- イベント・コード: 0xdc、最大 Inc/Cyc: 1
- 定義: L3 キャッシュ・ミスの回数をカウントする。命令フェッチ、データの読み出し / 書き込み、L2 ライトバックに起因するミスが含まれる。

L3_READS

- タイトル: L3 読み出し
- カテゴリ: L3 ユニファイド・キャッシュ IAR/DAR/OPC: Y/Y/Y
- イベント・コード: 0xdd、最大 Inc/Cyc: 1
- 定義: L3 キャッシュ読み出しアクセスの数をカウントする。

表 11-93. L3_READS のユニット・マスク

拡張子	PMC.umask [19:16]	説明
---	b0000	(* 何もカウントされない *)
DINST_FETCH.HIT	b0001	L3 デマンド命令フェッチ・ヒット
DINST_FETCH.MISS	b0010	L3 デマンド命令フェッチ・ミス
DINST_FETCH.ALL	b0011	L3 デマンド命令参照
---	b0100	(* 何もカウントされない *)
INST_FETCH.HIT	b0101	L3 命令フェッチおよびプリフェッチ・ヒット
INST_FETCH.MISS	b0110	L3 命令フェッチおよびプリフェッチ・ミス
INST_FETCH.ALL	b0111	L3 命令フェッチおよびプリフェッチ参照
---	b1000	(* 何もカウントされない *)
DATA_READ.HIT	b1001	L3 ロード・ヒット (ストアを満たすために使用する RFO を除く)
DATA_READ.MISS	b1010	L3 ロード・ミス (ストアを満たすために使用する RFO を除く)
DATA_READ.ALL	b1011	L3 ロード参照 (ストアを満たすために使用する RFO を除く)
---	b1100	(* 何もカウントされない *)
ALL.HIT	b1101	L3 読み出しヒット
ALL.MISS	b1110	L3 読み出しミス
ALL.ALL	b1111	L3 読み出し参照

L3_REFERENCES

- タイトル: L3 参照
- カテゴリ: L3 ユニファイド・キャッシュ IAR/DAR/OPC: Y/Y/Y
- イベント・コード: 0xdb、最大 Inc/Cyc: 1
- 定義: L3 アクセスの数をカウントする。命令のフェッチ / プリフェッチ、データの読み出し / 書き込み、L2 ライトバックが含まれる。

L3_WRITES

- タイトル: L3 書き込み
- カテゴリ: L3 ユニファイド・キャッシュ IAR/DAR/OPC: Y/Y/Y
- イベント・コード: 0xde、最大 Inc/Cyc: 1
- 定義: L3 キャッシュ書き込みアクセスの数をカウントする。

表 11-94. L3_WRITES のユニット・マスク

拡張子	PMC.umask [19:16]	説明
---	b00xx	(* 何もカウントされない *)
---	b0100	(* 何もカウントされない *)
DATA_WRITE.HIT	b0101	L3 ストア・ヒット (L2 ライトバックは除き、ストアを満たす L3 RFO 要求は含む)
DATA_WRITE.MISS	b0110	L3 ストア・ミス (L2 ライトバックは除き、ストアを満たす L3 RFO 要求は含む)
DATA_WRITE.ALL	b0111	L3 ストア参照 (L2 ライトバックは除き、ストアを満たす L3 RFO 要求は含む)
---	b1000	(* 何もカウントされない *)
L2_WB.HIT	b1001	L2 ライトバック・ヒット
L2_WB.MISS	b1010	L2 ライトバック・ミス
L2_WB.ALL	b1011	L2 ライトバック参照
---	b1100	(* 何もカウントされない *)
ALL.HIT	b1101	L3 書き込みヒット
ALL.MISS	b1110	L3 書き込みミス
ALL.ALL	b1111	L3 書き込み参照

LOADS_RETIRED

- タイトル: リタイアしたロード
- カテゴリ: 命令実行 / L1D キャッシュ・セット 3 IAR/DAR/OPC: Y/Y/Y
- イベント・コード: 0xcd、最大 Inc/Cyc: 4
- 定義: プレディケート・オフ・ロードを除き、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、浮動小数点、RSE、セマフォ、VHPT、キャッシュ不可の各ロードと、ALAT と L1D でミスしたチェック・ロード (ld.c) が含まれる (これは、その他のロードのように見える唯一のときだからである)。
- 注: 制限付きのセット 3 L1D キャッシュ・イベントである。このイベントを計測するには、このセット内のイベントの 1 つを PMD5 で計測しなければならない。

MEM_READ_CURRENT

- タイトル: バス上の現在のメモリの読み出しトランザクション
- カテゴリ: フロントサイド・バス IAR/DAR/OPC: N/N/N
- イベント・コード: 0x89、最大 Inc/Cyc: 1
- 定義: バス上の現在のメモリ読み出しトランザクション (BRC) の数をカウントする。

表 11-95. MEM_READ_CURRENT のユニット・マスク

拡張子	PMC.umask [19:16]	説明
---	bxx00	(* 何もカウントされない *)
IO	bxx01	非 CPU プライオリティ・エージェント
---	bxx10	(* 不正な選択 *)
ANY	bxx11	CPU または非 CPU (すべてのトランザクション)

MISALIGNED_LOADS_RETIRED

- タイトル: リタイアしたアライメントの合っていないロード命令
- カテゴリ: 命令実行/L1D キャッシュ・セット 3 IAR/DAR/OPC: Y/Y/Y
- イベント・コード: 0x0c、最大 Inc/Cyc: 4
- 定義: プレディケート・オフされたものを除き、リタイアしたアライメントの合っていないロード命令の数をカウントする。これには、整数、浮動小数点ロード、セマフォ、ALAT と L1D でミスしたチェック・ロード (ld.c) が含まれる (これがその他のロードのように見える唯一のとき)。
- 注: アライメントの合っていないロードがトラップを処理する場合は、リタイアしたロードのみがカウントされるため、このアライメントの合っていないロードはここではカウントされない。PSR.ac=0 であり、0 ~ 7 バイト境界または 8 ~ 15 バイト境界を横切らないときだけが、トラップしないときである。これは、制限付きのセット 3 L1D キャッシュ・イベントである。このイベントを計測するには、このセット内のイベントの 1 つを PMD5 で計測しなければならない。

MISALIGNED_STORES_RETIRED

- タイトル: リタイアしたアライメントの合っていないストア命令
- カテゴリ: 命令実行/L1D キャッシュ・セット 4 IAR/DAR/OPC: Y/Y/Y
- イベント・コード: 0xd2、最大 Inc/Cyc: 2
- 定義: プレディケート・オフされたものを除き、リタイアしたアライメントの合っていないストア命令の数をカウントする。これには、整数、浮動小数点、セマフォ、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。
- 注: アライメントの合っていないストアがトラップを処理する場合は、リタイアしたストアのみがカウントされるため、このアライメントの合っていないストアはここではカウントされない。PSR.ac=0 であり、WB ページの 0 ~ 15 バイト境界を横切らないときだけが、トラップしないときである。これは、制限付きのセット 4 L1D キャッシュ・イベントである。このイベントを計測するには、このセット内のイベントの 1 つを PMD5 で計測しなければならない。カウントされるのは、ポート 2 および 3 のみである。

NOPS_RETIRED

- タイトル: リタイアした NOP 命令
- カテゴリ: 命令実行 IAR/DAR/OPC: Y/N/Y
- イベント・コード: 0x50、最大 Inc/Cyc: 6
- 定義: リタイアした nop.i、nop.m、および nop.b、nop.f 命令の数について情報を提供する (プレディケート・オフされた nop 命令は除く)。

PREDICATE_SQUASHED_RETIRED

- タイトル: プレディケート・オフのために実行されなかった命令
- カテゴリ: 命令実行 IAR/DAR/OPC: Y/N/Y
- イベント・コード: 0x51、最大 Inc/Cyc: 6
- 定義: 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について情報を提供する。これには、偽のプレディケートを持つリタイアメントに達した非 B シラブル命令がすべて含まれる。

RSE_CURRENT_REGS_2_TO_0

- タイトル: 現在の RSE レジスタ (ビット 2:0)
- カテゴリ: RSE イベント IAR/DAR/OPC: N/N/N
- イベント・コード: 0x2b、最大 Inc/Cyc: 7
- 定義: RSE_EVENT_RETIRED が発生する前の現在の RSE レジスタの数をカウントする。Itanium 2 プロセッサでは、このイベントを 1 サイクル当たり合計 96 個含むことができる。最下位の 3 ビットは、このカウンタに格納される (ビット 2:0)。

RSE_CURRENT_REGS_5_TO_3

- タイトル: 現在の RSE レジスタ (ビット 5:3)
- カテゴリ: RSE イベント IAR/DAR/OPC: N/N/N
- イベント・コード: 0x2a、最大 Inc/Cyc: 7
- 定義: RSE_EVENT_RETIRED が発生する前の現在の RSE レジスタの数をカウントする。Itanium 2 プロセッサでは、このイベントを 1 サイクル当たり合計 96 個含むことができる。中間の 3 ビットは、このカウンタに格納される (ビット 5:3)。

RSE_CURRENT_REGS_6

- タイトル: 現在の RSE レジスタ (ビット 6)
- カテゴリ: RSE イベント IAR/DAR/OPC: N/N/N
- イベント・コード: 0x26、最大 Inc/Cyc: 1
- 定義: RSE_EVENT_RETIRED が発生する前の現在の RSE レジスタの数をカウントする。Itanium 2 プロセッサでは、このイベントを 1 サイクル当たり合計 96 個含むことができる。最上位 1 ビットは、このカウンタに格納される (ビット 6)。

RSE_DIRTY_REGS_2_TO_0

- タイトル: ダーティな RSE レジスタ (ビット 2:0)
- カテゴリ: RSE イベント IAR/DAR/OPC: N/N/N
- イベント・コード: 0x29、最大 Inc/Cyc: 7
- 定義: RSE_EVENT_RETIRED が発生する前のダーティな RSE レジスタの数をカウントする。Itanium 2 プロセッサでは、このイベントを 1 サイクル当たり合計 96 個含むことができる。最下位の 3 ビットは、このカウンタに格納される (ビット 2:0)。

RSE_DIRTY_REGS_5_TO_3

- タイトル: ダーティな RSE レジスタ (ビット 5:3)
- カテゴリ: RSE イベント IAR/DAR/OPC: N/N/N
- イベント・コード: 0x28、最大 Inc/Cyc: 7
- 定義: RSE_EVENT_RETIRED が発生する前のダーティな RSE レジスタの数をカウントする。Itanium 2 プロセッサでは、このイベントを 1 サイクル当たり合計 96 個含むことができる。中間の 3 ビットは、このカウンタに格納される (ビット 5:3)。

RSE_DIRTY_REGS_6

- タイトル: ダーティな RSE レジスタ (ビット 6)
- カテゴリ: RSE イベント IAR/DAR/OPC: N/N/N
- イベント・コード: 0x24、最大 Inc/Cyc: 1
- 定義: RSE_EVENT_RETIRED が発生する前のダーティな RSE レジスタの数をカウントする。Itanium 2 プロセッサでは、このイベントを 1 サイクル当たり合計 96 個含むことができる。最上位 1 ビットは、このカウンタに格納される (ビット 6)。

RSE_EVENT_RETIRED

- タイトル: リタイアした RSE 操作
- カテゴリ: RSE イベント IAR/DAR/OPC: N/N/N
- イベント・コード: 0x32、最大 Inc/Cyc: 1
- 定義: リタイアした RSE 操作 (alloc、br.ret、br.call、loadrs、flushrs、cover、rfi。注を参照) の数をカウントする。このイベントは、RSE に影響を与える命令がいつリタイアするのかを示す (これは、メモリ・サブシステムの動作を引き起こす場合もあれば、引き起こさない場合もある)。
- 注: 2 つの RSE イベントを 1 クロックでリタイアできるのは、flushrs/call バンドルまたは flushrs/return バンドルのときである。これらのコーナー・ケースは、2 つのイベントではなく、1 つのイベントとしてカウントされる。これは、このイベントは、現在 / ダーティ / 無効レジスタの数の平均を計算するために使用されるからである。rfi 命令は、ifsvaid=1 の場合にのみ含まれる。これを設定するには、rfi の前に cover 命令を使用するか、有効なビットを明示的に設定する。

RSE_REFERENCES_RETIRED

- タイトル: RSE アクセス
- カテゴリ: RSE イベント IAR/DAR/OPC: Y/Y/Y
- イベント・コード: 0x20、最大 Inc/Cyc: 2
- 定義: 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 (RSE.bof が RSE.storereg に到達するごと。これは、rmat フィルおよびスピルを含む強制イベントとも呼ばれる)。このイベントは、いつ RSE がメモリ・サブシステムの動作を引き起こすのかを示す。
- 注: DBR タグの特権レベルは、RSC レジスタによって決定される。しかし、IBR タグの特権レベルは、PSR.cpl によって決定される。rfi によって引き起こされる RSE トラフィックは、rfi のターゲットによってタグが付けられる。

表 11-96. RSE_REFERENCES_RETIRED のユニット・マスク

拡張子	PMC.umask [19:16]	説明
---	bxx00	(* 何もカウントされない *)
LOAD	bxx01	RSE ロードのみがカウントされる
STORE	bxx10	RSE ストアのみがカウントされる
ALL	bxx11	RSE ロードおよびストアの両方がカウントされる

SERIALIZATION_EVENTS

- タイトル: srlz.i 命令の数
- カテゴリ: システム・イベント IAR/DAR/OPC: N/N/N
- イベント・コード: 0x53、最大 Inc/Cyc: 1
- 定義: srlz.i 命令の数をカウントする (これはマイクロラップと xpnflush の発生を引き起こすからである)。

STORES_RETIRED

- タイトル: リタイアしたストア
- カテゴリ: 命令実行/L1D キャッシュ・セット 4 IAR/DAR/OPC: Y/Y/Y
- イベント・コード: 0xd1、最大 Inc/Cyc: 2
- 定義: プレディケート・オフされたものを除き、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、浮動小数点、セマフォ、RSE、VHPT、キャッシュ不可ストアが含まれる。
- 注: 制限付きのセット 4 L1D キャッシュ・イベントである。このイベントを計測するには、このセット内のイベントの 1 つを PMD5 で計測しなければならない。カウントされるのは、ポート 2 および 3 のみである。

SYLL_NOT_DISPERSED

- タイトル: 配布されなかったシラブル
- カテゴリ: 命令ディスパースル・イベント IAR/DAR/OPC: Y/N/N
- イベント・コード: 0x4e、最大 Inc/Cyc: 5
- 定義: 配布されなかったシラブルの数をカウントする (ただし、ストールが原因で配布されなかったものは除く)。ユニット・マスクを使用すると、これを 4 つのコンポーネントの 1 つに分類できる。

表 11-97. SYLL_NOT_DISPERSED のユニット・マスク

拡張子	PMC.umask [19:16]	説明
EXPL	bxxx1	明示的なストップ・ビットが原因で配布されなかったシラブルの数をカウントする。これらは、プログラマが指定した設定済み S ビットとテンプレート 1 および 5 からなる。バンドル 0(1) 内のすべてのテンプレート 1/5 に対して、ディスパースルは 6 シラブル (3 シラブル) ヒットを処理する。バンドル 0(1) 内のすべての S ビットに対して、ディスパースルは 3 シラブル (0 シラブル) ヒットを処理する。
IMPL	bxx1x	暗黙的なストップ・ビットが原因で配布されなかったシラブルの数をカウントする。これらは、すべての未設定のストップ・ビット (非対称、オーバーサブスクリプション、暗黙) からなる。バンドル 0(1) 内のすべての暗黙ストップ・ビットに対して、ディスパースルは 6 シラブル (3 シラブル) ヒットを処理する。
FE	bx1xx	有効なバンドルを提供していないか、または有効な不正テンプレートを提供しているフロントエンドが原因で配布されなかったシラブルをカウントする。フロントエンドからのすべての無効なバンドルまたは有効な不正テンプレートに対して、ディスパースルは 3 シラブル・ヒットを処理する。ここでは、フロントエンド・フォルトを持つバンドル 1 がカウントされる (3 シラブル・ヒット)。

表 11-97. SYLL_NOT_DISPERSED のユニット・マスク (続き)

拡張子	PMC.umask [19:16]	説明
MLI	b1xxx	MLI バンドルおよび非 0 シラブルへのリステアが原因で配布されなかったシラブルをカウントする。各 MLI バンドルに対して、ディスパースルは 1 シラブル・ヒットを処理する。ディスパースルは、リステア先のシラブルに応じて、0 ~ 2 シラブル・ヒットを処理できる。ここでは、分割されたフロントエンド・フォルトを持つバンドル 1 がカウントされる (0 ~ 2 シラブル・ヒット)。
ALL	b1111	配布されなかったシラブルをすべてカウントする。 注: b0000 ~ b1111 のすべての組み合わせが有効である。

SYLL_OVERCOUNT

- タイトル: オーバーカウントされたシラブルの数
- カテゴリ: 命令ディスパースル・イベント IAR/DAR/OPC: Y/N/N
- イベント・コード: 0x4f, 最大 Inc/Cyc: 2
- 定義: SYLL_NOT_DISPERSED の明示的または暗黙的なストップ・ビットの部分でオーバーカウントされたシラブルの数をカウントする。

表 11-98. SYLL_OVERCOUNT のユニット・マスク

拡張子	PMC.umask [19:16]	説明
---	bxx00	(* 何もカウントされない *)
EXPL	bxx01	明示的なバケット内でオーバーカウントされたシラブルのみ
IMPL	bxx10	暗黙的なバケット内でオーバーカウントされたシラブルのみ
ALL	bxx11	暗黙的バケットおよび明示的バケット内でオーバーカウントされたシラブル

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: 命令実行/L1D キャッシュ・セット 3 IAR/DAR/OPC: Y/Y/Y
- イベント・コード: 0xcf, 最大 Inc/Cyc: 4
- 定義: プレディケート・オフされたものを除き、リタイアしたキャッシュ不可ロード命令の数をカウントする。これには、整数、浮動小数点、セマフォ、RSE、VHPT の各ロードと、ALAT および L1D でミスしたチェック・ロード (ld.c) が含まれる (これがその他のロードのように見える唯一のとき)。
- 注: 制限付きのセット 3 L1D キャッシュ・イベントである。このイベントを計測するには、このセット内のイベントの 1 つを PMD5 で計測しなければならない。

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: 命令実行 /L1D キャッシュ・セット 4 IAR/DAR/OPC: Y/Y/Y
- イベント・コード: 0xd0、最大 Inc/Cyc: 2
- 定義: リタイアしたキャッシュ不可ストア命令の数をカウントする。これには、整数、浮動小数点、RSE、キャッシュ不可ストアが含まれる (ポート 2 および 3 上のみ)。
- 注: 制限付きのセット 4 L1D キャッシュ・イベントである。このイベントを計測するには、このセット内のイベントの 1 つを PMD5 で計測しなければならない。

本章では、Itanium® 2 プロセッサに関するモデル固有の機能とオプションの機能について説明する。

12.1 メモリ属性

UCE (Uncacheable exported) は、Itanium アーキテクチャのオプションの機能である。Itanium 2 プロセッサは、メモリ属性として、WB、UC[E]、WC をサポートしている。

UCE は、fetchadd 命令の実行時を除いて、UC のように動作する。UCE では、fetchadd 命令をプロセッサの外にエクスポートできる。

UCE の動作の詳細は、『インテル® Itanium® 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：システム・アーキテクチャ』のアドレス指定に関する説明を参照のこと。

12.2 ptc.e のページ動作

ページ動作は、モデルごとに異なる。Itanium 2 プロセッサは、ページまたは挿入で、次のページ・サイズをサポートしている。

4K、8K、16K、64K、256K、1M、4M、16M、64M、256M、1G、4G。

ptc.e では、すべてのデータおよび命令 TLB レベルのトランザクション・キャッシュが、1 回の反復ですべてフラッシュされる。

12.3 CPUID の戻り値

12.3.1 Itanium® 命令 CPUID の戻り値

Itanium 2 プロセッサでは、CPUID レジスタからの Itanium 命令の move によって、次の値が返される。プロセッサの改訂に関する最新の情報は、Itanium 2 プロセッサの仕様を参照のこと。

表 12-1. Itanium® 2 プロセッサ CPUID の戻り値

CPUID[x]	ビット・フィールド	値	説明
3	7:0	0x04	サポートされている CPUID レジスタの数
3	15:8	-	プロセッサの改訂
3	23:16	0x00	プロセッサ・モデル
3	31:24	0x1f	プロセッサ・ファミリ
3	39:32	0x00	アーキテクチャの改訂
3	63:40	0x00	予約済み
4	63:0	0x01	brl 命令の実装を示す。OS はこれをエミュレートする必要はない。

12.3.2 IA-32 CPUID の戻り値

次の表は、Itanium 2 プロセッサのキャッシュで、IA-32 CPUID 命令の戻り値をデコードする方法をまとめたものである。

表 12-2. キャッシュ戻り値のエンコード

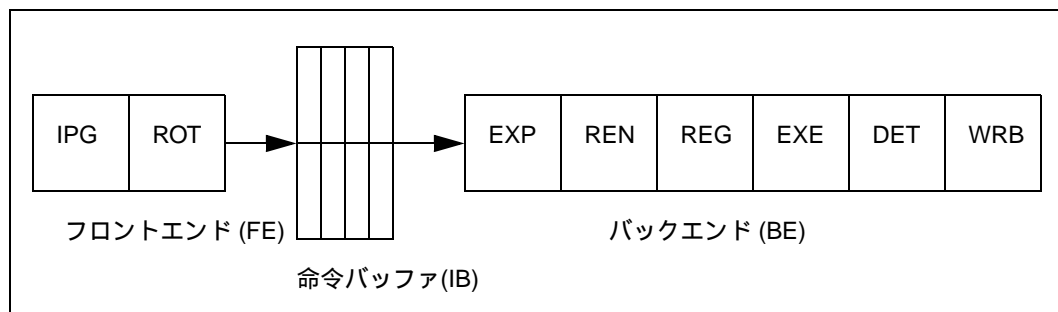
戻り値	キャッシュの説明
0x67	L1D: 16KB 4 ウェイ、64B ライン・サイズ
0x77	L1I: 16KB 4 ウェイ、64B ライン・サイズ
0x7e	L2: 256KB 8 ウェイ、128B ライン・サイズ
0x8d	L3: 3MB 12 ウェイ、128B ライン・サイズ

コア・パイプラインの深さは 8 ステージである。その他のマイクロパイプラインのいくつかは、コア・パイプラインに対して非同期的に動作する。

A.1 コア・パイプライン

コア・パイプラインは、フロントエンド (FE) とバックエンド (BE) に分かれている。FE と BE は、命令バッファ (IB) によって分離されている。

図 A-1. Itanium® 2 プロセッサのコア・パイプライン



コア・パイプラインは、以下の 8 つのステージからなる。

IPG: 命令ポインタの生成

ROT: 命令のローテーション

EXP: 命令テンプレートのデコード、拡張、配布

REN: リネーム (レジスタ・スタックおよびローテート・レジスタの)、デコード

REG: レジスタ・ファイル読み出し

EXE: ALU 実行

DET: 例外検出用の最終ステージ

WRB: ライトバック

A.2 パイプライン・ステージ

A.2.1 IPG STAGE

IPG (命令ポインタ生成) ステージは、命令ポインタを L1 命令に提供する。命令ポインタの値は、訂正されたターゲットまたはフォール・スルー・アドレス (分岐予測ミスを訂正するための)、例外ハンドラのアドレス (例外が発生した場合)、静的および動的に予測されたアドレス、または次のシーケンシャル・アドレスから提供される。このステージでは、L1 命令、ISB、L1 ITLB がアクセスされる。

L1 命令から IPG へのインターフェイスでは、バンドル・ペアのアライメントが常に偶数バンドル (32 バイト) アドレス境界に合わせられる。奇数境界バンドル上のバンドルをターゲットにする分岐は、低い方の偶数バンドル・アドレスからバンドル・ペアをフェッチする。結果として、有用なバンドルの 1 つのみが、(2 つのうちの最大のものの代わりとして) そのような分岐ターゲットでの IPG に提供される。

A.2.2 ROT ステージ

ROT (ローテート) ステージは、4 ウェイの命令キャッシュ配列および ISB データを読み出し、正確なウェイを選択する。フェッチされた命令は、アライメントを合わせて EXP ステージで使用するためにローテートされる。これが必要である理由は、EXP ステージの命令配布およびデコード・ユニットが、ローテーション・バッファから発行するバンドル数 (0 個、1 個、または 2 個) を決定するために、2 つのバンドルを参照するからである。したがって、ローテーション・バッファ内の命令のアライメントを適切に合わせなければならない。これは、デコード・ユニットでは常に、バンドル 0 に先頭の命令が含まれるものと想定しているからである。

A.2.3 EXP ステージ

EXP (拡張) ステージは、命令テンプレートをデコードし、最大 6 つの命令を機能ユニットに配布する。リソースの制限のため (リソースの制限については次の節を参照)、フェッチされたバンドルによっては、完全にディスパッチされないものがある。こうしたフェッチされたがディスパッチされないバンドルは、IB に押し戻される。EXP ステージで消費されたバンドルの数は、次のサイクルで EXP ステージに提供するバンドルを決定するために、ローテート・バッファにフィードバックされる。

A.2.4 REN ステージ

REN (リネーム) ステージは、仮想レジスタを物理レジスタに変換するために、それらの値をスタック・フレーム・ベースとローテート・レジスタ・ベースに追加する。このステージでは、命令のデコードも実行される。

A.2.5 REG ステージ

REG (レジスタ読み取り) ステージは、すべての実行ユニットにオペランドを提供する。データ読み出しは、一連のバイパス mux に送り込まれる。バイパス mux には、2 つのレベルがある。レジスタ・ファイルからのデータ読み出し、および DET ステージと WRB ステージからの結果は、初段のバイパスに送られる。現在の EXE ステージから生成されたデータとロード・データは、後段のバイパス mux に送り込まれる。レジスタや動的リソース・ハザードが検出されると、必要に応じて発行がストールされる。RSE ロードとストアは、REG ステージのパイプラインに送り込まれる。

A.2.6 EXE ステージ

EXE (実行) ステージは、ALU 実行ステージである。シングル・サイクル・レイテンシ演算によって、このステージの終わりの後段のバイパス mux に結果が送られ、それが後続の整数 ALU 演算で使用される。

A.2.7 DET ステージ

DET (検出) ステージは、例外検出を実行できる最終ステージである。アーキテクチャ・ステートのライトバックを kill するために、潜在的な例外はすべて、DET ステージの終わりまでに明らかにされる。不適切な分岐予測がないかどうかを調べる分岐の検証も、このステージで処理される。プレディケートの予測が誤り、または予測されたターゲット・アドレスが誤りであるために分岐が誤って予測されると、次の IP アドレスの実際のリステアがこのステージで発生する。そのため、分岐予測ミスにより、6 サイクルのパイプライン・バブルが発生する可能性がある。

A.2.8 WRB ステージ

WRB (ライトバック) ステージでは、結果がレジスタ・ファイルに書き戻される。命令が WRB ステージを完了すると、アーキテクチャ・プロセッサ・ステートの更新が保証される。

A.3 命令バッファ (IB)

メイン・パイプラインは、ROTパイプライン・ステージとEXPパイプライン・ステージとの間で分離される。先頭の2つのステージはフロントエンド (FE) に属し、残りの6つのステージはバックエンド (BE) である。命令バッファ (IB) (デカップリング・キュー) は、FEとBEをリンクする。このIBは、バンドル・ペアを4つ (24命令) 保持できる。IBによって、FEとBEは独立して機能できる。したがって、命令キャッシュ・ミスの遅延と処理済み分岐のペナルティは、BE内で発生する実行ストールによって隠蔽できる。

A.4 マイクロパイプライン

A.4.1 FPU マイクロパイプライン

FPUパイプラインの深さは4ステージ (FP1 ~ FP4) である。ライトバックは第5ステージ (FWB) で実行される。FPUは、全面的にパイプライン化されている。FP1ステージでは、数値オペランドの検査が早い段階で実行され、命令に数値例外がないかどうかを確認される。

表 A-1. FPUパイプライン

コア・パイプライン	REG	EXE	DET	WRB		
FPUパイプライン		FP1	FP2	FP3	FP4	FWB

A.4.2 L1D マイクロパイプライン

L1Mステージでは、L1データ、タグ、L1DTLBが同時にアクセスされ、実行ユニットにデータが提供される。

表 A-2. L1D マイクロパイプライン

コア・パイプライン	REN	REG	EXE	DET	WRB
L1Dパイプライン		L1I	L1M	L1D	WRB

A.4.3 L2 マイクロパイプライン

先頭のステージがL2TLBアクセスのために使用される。L2Aステージは、データ・アレイ・アクセスについてアービトレーションを行う。最高の優先順位を持つのは命令のデマンド・フェッチであり、その後にロード、プリフェッチが続く。L2Mステージでは、データ・アレイ・アクセスが発生する。L2Dステージは、ウェイの選択、データ配布のために使用される。L2Cステージは、ECCエラーの訂正とエラー検出のために使用される。

表 A-3. L2 マイクロパイプライン

コア・パイプライン	REG	EXE	DET	WRB			
L2パイプライン		L2L	L2A	L2M	L2D	L2C	L2W

