SPARC64™ IXfx Extensions

Fujitsu Limited

Ver 12, 2 Dec. 2013
## History

<table>
<thead>
<tr>
<th>Date</th>
<th>Descriptions</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2013/12/02</td>
<td>第 12 版発行</td>
<td></td>
</tr>
</tbody>
</table>
Contents

1. Overview 1
   1.1 Navigating the SPARC64™ IXfx Extensions 1
   1.2 Fonts and Notational Conventions 1

2. Definitions 3

3. Architectural Overview 7
   3.1 SPARC64 IXfx プロセッサ 7
       3.1.1 コア内部のコンポーネント 9
       3.1.2 命令制御ユニット (IU) 11
       3.1.3 命令実行ユニット (EU) 11
       3.1.4 ストレージ制御ユニット (SU) 12
       3.1.5 二次キャッシュユニット (SXU) 12
   3.2 プロセッサパイプライン 13
       3.2.1 命令フェッチステージ 13
       3.2.2 命令発行ステージ 15
       3.2.3 実行ステージ 15
       3.2.4 命令完了ステージ 16

4. Data Formats 17

5. Registers 19
   5.1 Nonprivileged Registers 20
       5.1.1 General-Purpose r Registers 20
       5.1.4 Floating-Point Registers 20
5.1.7 Floating-Point State Register (FSR) 22
5.1.9 Tick (TICK) Register 25
5.2 Privileged Registers 25
5.2.6 Trap State (TSTATE) Register 25
5.2.9 Version (VER) Register 26
5.2.11 Ancillary State Registers (ASRs) 26
5.2.12 Registers Referenced Through ASIs 34
5.2.13 Floating-Point Deferred-Trap Queue (FQ) 38
5.2.14 IU Deferred-Trap Queue 38

6. Instructions 39
6.1 Instruction Execution 39
6.1.1 Data Prefetch 39
6.1.2 Instruction Prefetch 40
6.1.3 Syncing Instructions 40
6.2 Instruction Formats and Fields 40
6.3 Instruction Categories 41
6.3.3 Control-Transfer Instructions (CTIs) 41
6.3.7 Floating-Point Operate (FPop) Instructions 42
6.3.8 Implementation-Dependent Instructions 42

7. Traps 43
7.1 Processor States, Normal and Special Traps 43
7.1.1 RED_state 43
7.1.2 error_state 44
7.2 Trap Categories 44
7.2.2 Deferred Traps 44
7.2.4 Reset Traps 44
7.2.5 Uses of the Trap Categories 45
7.3 Trap Control 45
7.3.1 PIL Control 45
7.4 Trap-Table Entry Addresses 45
7.4.2 Trap Type (TT) 45
7.4.3 Trap Priorities 49
7.5 Trap Processing 49
7.6 Exception and Interrupt Descriptions 50
7.6.1 Traps Defined by SPARC V9 As Mandatory 50
7.6.2 SPARC V9 Optional Traps That Are Mandatory in SPARC JPS1 50
7.6.4 SPARC V9 Implementation-Dependent, Optional Traps That Are Mandatory in SPARC JPS1 50
7.6.5 SPARC JPS1 Implementation-Dependent Traps 51

8. Memory Models 53
8.1 Overview 54
8.4 SPARC V9 Memory Model 54
8.4.5 Mode Control 54
8.4.7 Synchronizing Instruction and Data Memory 54

A. Instruction Definitions 57
A.4 Block Load and Store Instructions (VIS I) 66
A.9 Call and Link 68
A.24 Implementation-Dependent Instructions 69
A.24.1 Floating-Point Multiply-Add/Subtract 70
A.24.2 Suspend 76
A.24.3 Sleep 77
A.24.4 Integer Multiply-Add 78
A.25 Jump and Link 80
A.26 Load Floating-Point 81
A.27 Load Floating-Point from Alternate Space 85
A.30 Load Quadword, Atomic [Physical] 88
A.35 Memory Barrier 90
A.41 No Operation 92
A.42 Partial Store (VIS I) 93
A.48 Population Count 94
A.49 Prefetch Data 95
A.51 Read State Register 97
A.59 SHUTDOWN (VIS I) 99
A.61 Store Floating-Point 100
A.62 Store Floating-Point into Alternate Space 104
A.68 Trap on Integer Condition Codes (Tcc) 107
A.69 Write Privileged Register 108
A.70 Write State Register 111
A.71 Deprecated Instructions 113  
A.71.10 Store Barrier 113  
A.72 Floating-Point Conditional Compare to Register 114  
A.73 Floating-Point Minimum and Maximum 116  
A.74 Floating-Point Reciprocal Approximation 118  
A.75 Move Selected Floating-Point Register on Floating-Point Register's Condition 122  
A.76 Floating-Point Trigonometric Functions 123  
A.77 Store Floating-Point Register on Register Condition 128  
A.78 Set XAR (SXAR) 131  
A.79 Cache Line Fill with Undetermined Values 133  

B. IEEE Std. 754-1985 Requirements for SPARC-V9 139  
B.1 Traps Inhibiting Results 139  
B.6 Floating-Point Nonstandard Mode 139  
B.6.1 fp_exception_other Exception (ftt=unfinished_FPop) 140  
B.6.2 Behavior when FSR.NS = 1 143  

C. Implementation Dependencies 147  
C.4 List of Implementation Dependencies 147  

D. Formal Specification of the Memory Models 159  

E. Opcode Maps 161  

F. Memory Management Unit 173  
F.1 Virtual Address Translation 173  
F.2 Translation Table Entry (TTE) 174  
F.4 Hardware Support for TSB Access 177  
F.5 Faults and Traps 178  
F.5.1 Trap Conditions for SIMD Load/Store 180  
F.5.2 Behavior on TLB Error 180  
F.8 Reset, Disable, and RED_state Behavior 182  
F.10 Internal Registers and ASI Operations 183  
F.10.1 Accessing MMU Registers 183  
F.10.2 Context Registers 187  
F.10.3 Instruction/Data MMU TLB Tag Access Registers 191
F.10.4 I/D TLB Data In, Data Access, and Tag Read Registers 192
F.10.6 I/D TSB Base Registers 193
F.10.7 I/D TSB Extension Registers 194
F.10.8 I/D TSB 8-Kbyte and 64-Kbyte Pointer and Direct Pointer Registers 194
F.10.9 I/D Synchronous Fault Status Registers (I-SFSR, D-SFSR) 194
F.10.10 Synchronous Fault Addresses 201
F.10.11 I/D MMU Demap 201
F.10.12 Synchronous Fault Physical Addresses 202
F.11 MMU Bypass 202
F.12 Translation Lookaside Buffer Hardware 203
F.12.2 TLB Replacement Policy 203

G. Assembly Language Syntax 205
G.1 Notation Used 205
G.1.5 Other Operand Syntax 205
G.4 HPC-ACE 拡張機能の表記法 206
G.4.1 HPC-ACE 拡張のサフィックス 206

H. Software Considerations 209

I. Extending the SPARC V9 Architecture 210

J. Changes from SPARC V8 to SPARC V9 211

K. Programming with the Memory Models 212

L. Address Space Identifiers 213
L.2 ASI Values 213
L.3 SPARC64 IXfx ASI Assignments 214
L.3.1 Supported ASIs 214
L.3.2 Special Memory Access ASIs 220
L.3.3 ASI と命令の組み合わせと例外 221
L.3.4 内部レジスタの更新タイミング 222
L.4 ハードウェアバリア 222
L.4.1 バリア資源の初期化と状態獲得 224
L.4.2 BST ビット位置の取得 226
L.4.3 バリア資源の割り付け 227
L.4.4 バリア操作用 ASI 228
M. Cache Organization 231
M.1 キャッシュタイプ 231
  M.1.1 L1 命令キャッシュ (L1I キャッシュ) 232
  M.1.2 L1 データキャッシュ (L1D キャッシュ) 233
  M.1.3 L2 キャッシュ 233
M.2 キャッシュコヒーレンシープロトコル 234
M.3 キャッシュ制御 ASI 235
  M.3.1 命令キャッシュフラッシュ (ASI_FLUSH_L1I) 235
  M.3.2 キャッシュデータの無効化 (ASI_CACHE_INV) 235
  M.3.3 セクタキャッシュ設定 (SCCR) 236
M.4 ハードウェアプリフェッチ 239

N. Interrupt Handling 241
N.1 Interrupt Vector Dispatch 241
N.2 Interrupt Vector Receive 243
N.4 Interrupt ASI Registers 244
  N.4.1 Outgoing Interrupt Vector Data<7:0> Register 244
  N.4.2 Interrupt Vector Dispatch Register 244
  N.4.3 Interrupt Vector Dispatch Status Register 244
  N.4.4 Incoming Interrupt Vector Data Registers 244
  N.4.5 Interrupt Vector Receive Register 245
N.6 インタラプト配送先の識別法 245

O. Reset, RED_state, and error_state 247
O.1 リセット種類 247
  O.1.1 パワーオンリセット (POR) 247
  O.1.2 ウォッチドッグリセット (WDR) 248
  O.1.3 外部指示リセット (XIR) 248
  O.1.4 ソフトウェア指示リセット (SIR) 248
O.2 RED_state と error_state 249
  O.2.1 RED_state 250
  O.2.2 error_state 250
  O.2.3 CPU Fatal Error 250
O.3 リセット、RED_state 後のプロセッサ状態 251
  O.3.1 Operating Status Register (OPSR) 255
P. Error Handling  257

P.1 エラーの分類  257
  P.1.1 致命的エラー  258
  P.1.2 error_state 遷移エラー  258
  P.1.3 緊急エラー  259
  P.1.4 抑止可能エラー  261
  P.1.5 instruction_access_error  262
  P.1.6 data_access_error  262

P.2 エラー処理とエラー制御  263
  P.2.1 エラー処理に必要なレジスタ  263
  P.2.2 エラー検出時の動作  263
  P.2.3 CE 発見時に元データを自動で訂正できる限界  268
  P.2.4 キャッシュデータのエラーマーキング  269
  P.2.5 ASI_EIDR  272
  P.2.6 エラー検出の制御 (ASI_ERROR_CONTROL)  272

P.3 致命的エラーとerror_state 遷移エラー  274
  P.3.1 ASI_STCHG_ERROR_INFO  274
  P.3.2 サスペンド中の中スレッドでのerror_state 遷移エラー  276

P.4 緊急エラー  276
  P.4.1 緊急エラーステータス (ASI_UGESR)  277
  P.4.2 async_data_error (ADE)  トラブル発生時の処理  280
  P.4.3 ADE  トラブル発生した時の命令の実行状況  283
  P.4.4 ADE  トラブルハンドラの処理例  284

P.5 Instruction Access Errors  286

P.6 Data Access Errors  286

P.7 抑止可能エラー  287
  P.7.1 ASI_ASYNC_FAULT_STATUS (ASI_AFSR)  287
  P.7.2 抑止可能エラーに対するソフトウェア処理  288

P.8 レジスタで起きたエラーの処理方法  288
  P.8.1 特権・非特権レジスタの処理方法  289
  P.8.2 ASR レジスタの処理方法  290
  P.8.3 ASI レジスタの処理方法  291

P.9 キャッシュで起きたエラーの処理方法  294
  P.9.1 キャッシュタグで起きたエラーの処理方法  295
  P.9.2 L1I キャッシュデータで起きたエラーの処理方法  295
  P.9.3 L1D キャッシュデータで起きたエラーの処理方法  296
P.9.4 L2 キャッシュデータで起きたエラーの処理方法 297
P.9.5 L1I,L1D,L2 キャッシュの自動ウェイ縮退 298
P.10 TLB で発生したエラー 300
P.10.1 TLB エラーの処理 300

Q. Performance Instrumentation 303

Q.1 PA 概要 303
  Q.1.1 サンプル擬似コード 303
Q.2 PA イベントの説明 305
  Q.2.1 命令種類、トラップ種類毎の統計情報 308
  Q.2.2 MMU と L1 キャッシュ関連のイベント計測 316
  Q.2.3 L2 キャッシュ関連のイベント計測 317
  Q.2.4 バストランザクションの計測 320
Q.3 サイクルアカウンティング 321

R. System Programmer’s Model 325

R.1 System Config Register 325
R.2 STICK Control Register 326

S. Summary of Specification Differences 329
Overview

1.1 Navigating the SPARC64™ IXfx Extensions

SPARC64 IXfx は、JPS1 に準拠したアーキテクチャのプロセッサである。JPS1 では、仕様を共通仕様と実装依存仕様に分けて記述するようになっており、本仕様書は実装依存仕様を定義する。本仕様書では原則として、共通仕様で定義済の事項については再掲しない。

本書の章・節番号と見出し名は、基本的に JPS1 Commonality に合わせてあり、英語のままにしてある。各章・節では JPS1 Commonality の実装依存仕様や未定義仕様、あるいは SPARC64 IXfx で変更された仕様について説明している。JPS1 Commonality にない章・節には日本語の見出しがついており、SPARC64 IXfx 独自に追加された仕様を説明している。本書は JPS1 Commonality の定義を前提としているので、必要に応じて “SPARC Joint Programming Specification 1 (JPS1): Commonality” (JPS1 Commonality) を参照されたい。

1.2 Fonts and Notational Conventions

本仕様書の表記法は JPS1 Commonality に準ずる。
Reservedフィールドの扱い
命令語およびレジスタ中で使用されていないビットは、将来のために予約されている。そのようなフィールドをReservedフィールドと呼び、Reservedまたは—と表記している。

JPS1CommonalityではChapter2で、命令語およびレジスタのReservedフィールドの扱いを以下のように定義している。
- 命令語のReservedフィールドは0であるべき。0以外の値が入っていた場合の動作は未定義(Chapter2)。
- レジスタのReservedフィールドにソフトウェアが書く値は、そのフィールドの読み出しで得られた値か0。ハードウェアは0を出力すべき。

なお、命令語のReservedに関しては、JPS1CommonalityのSection6.3.9とAppendixI.2にも記述がある。

SPARC64IXfxでは、Reservedを以下のように扱う。
- 命令語のReservedフィールドは、仕様に動作と値が明記されているものはその通り。Reservedフィールドが0以外の値のときの動作が明記されていない命令は、そのフィールドを無視して実行する。
- レジスタのReservedフィールドは、仕様に動作と値が明記されているものはその通り。動作と値が明記されていないフィールドは、書き込みは無視され、読み出しには不定値が返る。書き込んだ場合の副作用が書かれていないものの動作は未定義である。

レジスタフィールドの読み書き属性
この論理仕様書では、レジスタのフィールドの読み書き属性を以下のように記述する。

TABLE1-1レジスタフィールドの読み書き属性

<table>
<thead>
<tr>
<th>Type</th>
<th>属性</th>
<th>Description</th>
<th>意味</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>読み出しにはフィールドの値が返され、書き込みは無視される。</td>
<td></td>
<td></td>
</tr>
<tr>
<td>W</td>
<td>読み出しには不定値が返され、書き込みにはその値が書き込まれる。</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RW</td>
<td>読み出しにはフィールドの値が返され、書き込みにはその値が書き込まれる。</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RW1C</td>
<td>読み出しにはフィールドの値が返され、1を書き込むとフィールドに0が書かれる。</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
 Definitions

この章では SPARC64 Ix fx に固有の概念・用語を定義する。JPS1 に共通の用語の定義は、JPS1 Commonality の Chapter 2 を参照。

**basic floating-point register**
HPC-ACE で拡張されたレジスタのうち、SIMD の basic 側演算で使われる $f[0] - f[254]$ レジスタ。

**(SIMD の basic 側演算）**
SIMD 動作時の 2 演算のうちのひとつ。命令で指定したレジスタ番号を使う。

**extended floating-point register**
HPC-ACE で拡張されたレジスタのうち、SIMD の extended 側演算で使われる $f[256] - f[512]$ レジスタ。

**(SIMD の extended 側演算）**
SIMD 動作時の 2 演算のうちのひとつ。命令で指定したレジスタ番号 +256 を使う。

**HPC-ACE**
High Performance Computing - Arithmetic Computational Extensions の略で、SPARC64 IxFx 独自に拡張された論理仕様の総称。レジスタ本数の増加、HPC 用途の命令、浮動小数点演算の SIMD 拡張などを含む。

**mTLB**
メイン TLB。命令用 (I) とデータアクセス用 (D) に分かれており、それぞれ mITLB, mDTLB と呼ばれることもある。uITLB, uDTLB に載せるアドレス変換情報を保持している。

**syncing 命令**
マシン sync を起こす命令。syncing 命令の発行は、プログラム順で syncing 命令よりも前の命令がすべて完了してから行われ、syncing 命令が完了してから後続命令が発行される。つまり syncing 命令は単独で発行、完了、完了される命令である。
uTLB マイクロ TLB。命令用（I）とデータアクセス用（D）に分かれており、それぞれ uTLB, uDTLB と呼ばれることもある。ハードウェアは uTLB に載っているアドレス変換情報に基づいてアドレス変換を行う。uTLB にアドレス変換情報がなければ、mTLB から補充される。

XAR 対象命令 XAR レジスタで指定される、レジスタフィールドの上位ビット情報を結合して実行される命令。

アウトオブオーダ実行 命令の実行が、プログラム順序を追い越して行われるマイクロアーキテクチャ。
入力ソースデータが揃っていない命令を、後続の入力ソースデータが揃った命令が追い抜いて実行する。

演算器 (functional unit) 演算命令を処理する資源。

(資源の) 解放 命令実行のために割り当てられた資源が、次の命令により使用可能になること。

コア コアとは、実行パイプラインと命令実行に関連する資源 (演算器や L1 キャッシュなど) を含んだ実体。一つまたはそれ以上のスレッドを持つことがあるが、SPARC64 IXf では 1 コアは 1 スレッドからなる。

コミット（完了）ある命令について、その命令より前のすべての命令実行が正常に完了した後、当該命令の実行結果を確定すること。ある命令がコミットすると、その命令の実行結果はソフトウェアから見える資源に反映される。それ以前の状態は失われる。

コンプリート（終了）ある命令の実行が終了し、正常終了したことが命令発行ユニットに通知される命令の終了により実行結果が一時的に反映されるが、完了するまではまだ永続的な状態にあっており、元の状態に復旧することが可能である。

実行 命令が実行ユニットに送られ、演算が行われること。命令が演算器にある間は実行中である。

実行終了 演算器での命令の実行が終了し、結果が出力バスに出力されること。出力バスの結果はレジスタファイルや他の演算器へと送られる。

サイクル アカウンティング性能阻害要因を分析する手法。

サスペンド スレッドが命令実行を一時中止している状態。サスペンド中は命令は実行されないが、キャッシュ上のデータに矛盾が起きないようメンテナンスされている。

スーパースキャン コア内のすべてのルーチンをスキャンする方法。スキャン可能なように設計されたルーチンをスキャンリングで読み出し・書き込みができる。

ストール 命令発行ができないこと。資源の空き状態やプログラム上の制約から、必ずしも毎サイクル命令が発行できるとは限らない。
ストロングプリフェッチ プロセッサ内部の資源が不足していても捨てられずに実行されることが保証されたデータプリフェッチ命令。

スレッド ソフトウェア命令列を実行するハードウェアの単位。ソフトウェアから見える資源（PC、レジスタなど）や、直接見えないが命令実行に必要なマイクロアーキテクチャを含んでいる。

投機的命令実行 命令実行が投機的であるとは、条件分岐の分岐方向が未確定な時点で、または割り込みやトラップなどの発生有無が未確定な時点で、後続の命令を実行すること。また、投機的な実行により得られた結果を使い、さらに後続の命令を実行すること。

プロセッサモジュール 情報処理用の独立した実体で、外部との接続バスを共有するひとつまたはそれ以上のコアが入った部品。

命令ディスパッチ ある命令の実行に必要な資源が全部使用可能になり、演算器に送られること。

命令発行 命令がリザービションステーションに送られること。

命令フェッチ 命令が命令キャッシュまたはコア内部のバッファから読み出され、命令発行ユニットに送られること。

メモリマネージメントユニット (MMU) コア内にある、64 ビットの論理アドレスを物理アドレスに変換するハードウェア機構。MMU には mTLB、mDTLB、uITLB、uDTLB およびアドレス変換を制御するための ASI レジスタがある。

リザービションステーション 発行された命令が、実行パイプラインに投入されるまで格納されるキュー（バッファ）。演算器が使用可能であった場合に、入力データが揃った命令は、演算器が使用可能になると、リザービションステーションから取り出され演算器に投入される。アウトオブオーダ実行の制御はこのキューで行う。

リネーミングレジスタ 命令の実行結果を、コミットしてレジスタに格納されるまで一時的に保持するバッファ。ユーザが直接操作することはできない。
この章では、SPARC64 IXfx プロセッサの概要について説明する。この章は JPS1 Commonality の章立てには沿っていない。

3.1 SPARC64 IXfx プロセッサ

SPARC64 IXfx プロセッサは、1つのCPUチップに16個のコアとL2キャッシュ、さらにメモリコントローラ (MAC) を統合した、マルチコア・プロセッサである。アーキテクチャはSPARC V9に準拠しながら独自の拡張を行い、サーバ用途の高性能・高信頼性に加え、HPC分野でも高性能を追求したプロセッサである。

高性能を実現するマイクロアーキテクチャ

SPARC64 IXfx は、1サイクルに最大4命令まで発行するアウトオブオーダ実行のスーパースカラプロセッサである。アウトオブオーダの処理では、命令フェッチ部は実行パスを予測しながら命令をフェッチし、その命令列をその順番に従い発行する。発行された命令は、実行可能になるまで一時的にリザーバーションステーションに格納される。命令が実行可能になると、元の命令列の順序とは関係なく実行ユニットへ送られ、実行される。実行が終了 (complete) した命令は、元の命令列の順序に従い完了 (commit) 処理される。つまり、ある命令の完了処理は、その命令に先行する命令がすべて完了してから行われる。完了処理が終わると命令の実行結果がレジスタ・メモリ等に反映されるプログラムから見えるようになる。アウトオブオーダ実行はSPARC64 IXfxの高性能に大きく貢献している。

SPARC64 IXfx は、分岐命令の実行方向を予測するためにブランチヒストリパッファを実装している。ブランチヒストリパッファは、DBMSのような大規模アプリケーションで高い予測の精度を維持するよう、また SPARC64 IXfx の高度な命令フェッチ機構をサポートするよう、十分な大きさを備えている。この命令フェッチ機構は、分
岐履歴に従い、複数の条件分岐を含む命令列の実行方向を予測し、命令をフェッチする。これにより、命令キャッシュミスによって生じる性能ペナルティの影響を可能な限り小さくできる。

SPARC64 IXfx はまた、HPC (High Performance Computing) 用途にふさわしい機能を備えている。それはたとえば、SPARC V9 の命令仕様を独自に拡張した HPC-ACE 拡張や、チップ内コア間で高速な同期を実現するハードウェアバリア機能などである。HPC-ACE 拡張では整数レジスタを 192 本、浮動小数点レジスタを 256 本に拡張し、浮動小数点演算命令を中心に 7 つの新規命令を追加したほか、2 並列の SIMD(Single Instruction Multiple Data) 演算も可能にした。SIMD の採用により、1 サイクルで最大 8 つの浮動小数点演算を実行することができ、HPC 分野の高性能を実現している。

チップ内に集約された高機能

SPARC64 IXfx は、CPU チップ上に L2 キャッシュを持っている。L2 キャッシュは、命令やデータ共用で、16 コアに共有されている。SPARC64 IXfx では、L2 キャッシュが最外層のキャッシュである。L2 キャッシュを CPU チップに統合したことで、SPARC64 IXfx の性能と信頼性は向上し、アクセス時間が短くウェイ数の多いキャッシュを実装することが可能である。また、L2 キャッシュに対する外部接続が不要なため、高信頼性にも貢献している。

さらに SPARC64 IXfx は、メモリコントローラも CPU チップに内蔵している。DIMM は CPU に直接され、メモリアクセスのレイテンシは劇的に短縮された。

HPC の高性能を約束する重要な機能の一つがハードウェアバリアである。マルチスレッドのジョブを高速に処理するためには、スレッド間の同期にかかる時間を可能な限り減らすことが重要である。この問題に対して SPARC64 IXfx は、ハードウェアによるバリア同期機構という解決策を提供する。SPARC64 IXfx のハードウェアバリアは、複数コアでのバリア同期用に加え、全コアによるマスター/ワーカーモデルを可能にする、post/wait 型のプリミティブも用意している。

高信頼性

SPARC64 IXfx は以下の高度な RAS 機能を実装している。

1. キャッシュの RAS 機能

   ■ 強固なキャッシュエラー保護:
       ■ LID キャッシュデータ、L2 キャッシュデータ、L2 キャッシュタグに対する ECC 保護。
       ■ L1I キャッシュデータに対するパリティ保護。
       ■ L1I キャッシュタグと LID キャッシュタグに対するパリティ保護と二重化。
   ■ すべての種類の 1 ビットエラーの自動訂正:
       ■ ECC 保護されたデータに対する自動 1 ビットエラー訂正。
       ■ L1I キャッシュデータパリティエラーに対する L1I キャッシュデータの無効化と再読み込み。
キャッシュタグおよびL1Dキャッシュタグのバーティティエラーに対する二重化されたタグからのコピー。

キャッシュの一貫性を維持したまま行われる動的なウェイ縮退。

キャッシュデータの訂正不能エラーのマーキング：
- 最初に訂正不能エラーを発見したモジュールが、特殊なパターンでマークする。
- 一つの故障で何度もエラーが報告されないよう、エラーマークで故障モジュールを識別して隔離する。

2. 命令実行ユニットのRAS機能

- 強固なエラー保護：
  - 全てのデータパスのバーティティ保護。
  - ほとんどのソフトウェア可視レジスタ、内部、一時レジスタに対するバーティティ保護。
  - 演算結果のバリティまたは剰余チェック。

- ハードウェアによる命令再実行。
  - (ハードウェア命令再実行の失敗の後の) ソフトウェア命令再実行のサポート。

- ソフトウェアリカバリのためのエラー分離：
  - どのレジスタがエラーしたか(被疑レジスタ)の明示。
  - エラーした命令が再実行可能かどうかを示す。
  - エラーの度合いによりトラップを使い分ける。

3. 充実したソフトウェアインターフェイス

- プログラム実行への影響の重大度によるエラー分類：
  - 重大なエラー(マスク不可): OSの介入無しでの実行続行は不可能なエラー。例外で報告される。
  - 軽微なエラー(マスク可): エラーが例外で報告されるかどうかをOSが制御する。
  - エラーは、プログラム実行には直接作用しない。
- ソフトウェアへの影響を判断するための、識別したエラーの表示。
- 付加的なエラーに対する非同期データエラー(ADE)例外
  - 例外通知により命令実行が中断されるが、その状態は発見されたエラーによって異なるので、完了方法を表示する。
  - 遅延報告(deferred) だがリトライ可能なADE例外。
  - エラー分離と命令再実行が正しく行えるよう、同時に起きた複数のエラーをすべて表示する。

3.1.1 コア内部のコンポーネント

FIGURE 3-1はSPARC64 IXfxのブロック図である。SPARC64 IXfxは16個のコアとメモリコントローラ、バスインタフェースをチップ内に統合している。コア内には以下のコンポーネントがある。
- 命令制御ユニット(IU)
- 実行ユニット(EU)
■ 記憶ユニット (SU)
■ L2 キャッシュと外部アクセスユニット (SXU)

FIGURE 3-1  SPARC64 IXfx ブロック図
3.1.2 命令制御ユニット (IU)

IU (Instruction control Unit) は、命令実行パスを予測し、予測したパスの命令をフェッチする。その後、フェッチした命令を適切なリザベーションステーションに配送し、そして、実行パイプラインに命令を発送する。発送された命令は、アウトオブオーダで実行され、インオーダで完了される。主要なブロックを TABLE 3-1 に示す。

<table>
<thead>
<tr>
<th>ブロック</th>
<th>説明</th>
</tr>
</thead>
<tbody>
<tr>
<td>命令フェッチパイプライン</td>
<td>5 ステージ：フェッチアドレス生成、iTLB と L1I キャッシュアクセス、iTLB と L1I キャッシュのタグ比較、命令バッファへの書き込み、結果の格納。</td>
</tr>
<tr>
<td>ブランチヒストリ</td>
<td>分岐先と分岐方向を予測するためのテーブル。</td>
</tr>
<tr>
<td>命令バッファ</td>
<td>フェッチした命令を保持するためのバッファ。</td>
</tr>
<tr>
<td>リザベーションステーション</td>
<td>命令が実行可能になるまで保持するための 5 つのリザベーションステーション：分岐とその他の制御転送命令用の RSBR、load/store 命令用の RSA、整数演算命令用の RSE、浮動小数点演算命令用の RSFA と RSFB。</td>
</tr>
<tr>
<td>コミットスタックエントリ (CSE)</td>
<td>実行中の（発行されてまだ完了していない）命令の情報を保持するためのバッファ。</td>
</tr>
<tr>
<td>PC, nPC, CCR, FSR</td>
<td>命令実行制御のためのプログラム可視レジスタ。</td>
</tr>
</tbody>
</table>

3.1.3 命令実行ユニット (EU)

EU (Execution Unit) は全ての整数（算術、論理、シフト）命令と全ての浮動小数点及び VIS 命令の実行を行う。TABLE 3-2 に EU の主要なブロックを記述する。

<table>
<thead>
<tr>
<th>ブロック</th>
<th>説明</th>
</tr>
</thead>
<tbody>
<tr>
<td>GUB</td>
<td>汎用整数レジスタ (gr) 用のリネーミングレジスタファイル。</td>
</tr>
<tr>
<td>GPR</td>
<td>汎用整数レジスタファイル。</td>
</tr>
<tr>
<td>FUB</td>
<td>浮動小数点レジスタ (fr) 用のリネーミングレジスタファイル。</td>
</tr>
<tr>
<td>FPR</td>
<td>浮動小数点レジスタファイル。</td>
</tr>
<tr>
<td>EU 制御論理</td>
<td>命令実行ステージを制御：実行する命令の選択、レジスタリード、命令実行。</td>
</tr>
<tr>
<td>インタフェースレジスタ</td>
<td>他ユニットへの入出力レジスタ。</td>
</tr>
<tr>
<td>2 つの整数実行パイプライン (EXA, EXB)</td>
<td>64 ビット ALU とシフタ。</td>
</tr>
</tbody>
</table>
3.1.4 ストレージ制御ユニット (SU)

SU (Storage Unit) は、ロード及びストア命令のために全てのデータの供給と受給を扱う。TABLE 3-3 に、SU の主要なブロックを記述する。

<table>
<thead>
<tr>
<th>ブロック</th>
<th>説明</th>
</tr>
</thead>
<tbody>
<tr>
<td>レベル 1 命令キャッシュ</td>
<td>32KB, 2 ウェイセットアソシエティブ、1ラインは 128 バイト。低レイテンシ、命令供給用。</td>
</tr>
<tr>
<td>レベル 1 データキャッシュ</td>
<td>32KB, 2 ウェイセットアソシエティブ、1ラインは 128 バイト。低レイテンシ、ロード・ストアデータ供給用。</td>
</tr>
<tr>
<td>命令用 TLB</td>
<td>128 エントリ、2 ウェイセットアソシエティブ TLB (sITLB) が 2つ。</td>
</tr>
<tr>
<td></td>
<td>16 エントリ、フルアソシエティブ TLB (HTLB)。</td>
</tr>
<tr>
<td>データ用 TLB</td>
<td>256 エントリ、2 ウェイセットアソシエティブ TLB (sDTLB) が 2つ。</td>
</tr>
<tr>
<td></td>
<td>16 エントリ、フルアソシエティブ TLB (fDTLB)。</td>
</tr>
<tr>
<td>ストアバッファとライトバッファ</td>
<td>ストア操作のレイテンシからパイプラインを分離する。ストアがデータを持っている間、パイプラインが流れ続ける事を可能にする。最終的には、レベル 1 データキャッシュに書き込む。</td>
</tr>
</tbody>
</table>

3.1.5 二次キャッシュユニット (SXU)

SXU (Secondary Cache and External Access Unit) は、レベル 2 キャッシュの操作と外部データアクセスインタフェースを制御する。TABLE 3-4 に、SXU の主要なブロックを記述する。

<table>
<thead>
<tr>
<th>ブロック</th>
<th>説明</th>
</tr>
</thead>
<tbody>
<tr>
<td>レベル 2 キャッシュ</td>
<td>サイズは 12MB であり、24 ウェイセットアソシエティブ構成、1ラインは 128 バイト。ライトバック方式のキャッシュ。</td>
</tr>
</tbody>
</table>
3.2 プロセッサパイプライン

SPARC64 IXfx のパイプラインは 16 ステージからなる。FIGURE 3-2 にパイプラインの各ステージを、FIGURE 3-3 にパイプライン図を示す。

<table>
<thead>
<tr>
<th>IA</th>
<th>IT</th>
<th>IM</th>
<th>IB</th>
<th>IR</th>
<th>E</th>
<th>PD</th>
<th>D</th>
<th>P</th>
<th>B</th>
<th>X</th>
<th>Ps</th>
<th>Ts</th>
<th>Ms</th>
<th>Bs</th>
<th>U</th>
<th>C</th>
<th>W</th>
</tr>
</thead>
</table>

FIGURE 3-2 SPARC64 IXfx のパイプラインステージ

3.2.1 命令フェッチステージ

- **IA**: 命令フェッチアドレス生成
- **IT**: iTLB, 命令キャッシュタグアクセス
- **IM**: 命令キャッシュタグ比較
- **IB**: 命令キャッシュからバッファへ読み込み
- **IR**: 命令フェッチ最終ステージ

IA から IR までのステージでは、キャッシュアクセスユニット (SU) と連携して命令を読み込み、パイプラインの後続ステージに供給する。メモリまたはキャッシュから読み出された命令は、命令バッファ (Instruction Buffer) に蓄えられる。

SPARC64 IXfx は分岐予測用の BRHIS (BRanch HIStory) と RAS (Return Address Stack) 資源を備えている。命令フェッチアドレスステージでは、分岐予測を行いフェッチアドレスを決定する。

SPARC64 IXfx の命令フェッチステージは、実行ステージがストールしていても命令フェッチができるよう、後続ステージとは可能な限り独立したデザインになってい る。命令フェッチ部は、命令バッファがフルになるまでフェッチを続け、その後もブ リフェッチを出して命令列を L1 キャッシュに持ってくることが可能である。
FIGURE 3-3 SPARC64 IXfx のパイプライン図
3.2.2 命令発行ステージ

■ E: エントリ
■ PD: プリデコード
■ D: デコード

SPARC64 IXfx はアウトオブオーダ実行のプロセッサである。SPARC64 IXfx には 6 個の演算器があり（整数用 2 つ、浮動小数点用 2 つ、メモリアクセス用 2 つ）、整数演算器とメモリアクセス演算器には演算器 2 つに対して 1 個の、浮動小数点演算器には独立した 2 個のリザベーション・ステーションがある。E, PD, D ステージでは命令をデコード・発行し、リザベーション・ステーションに登録する。SPARC64 IXfx は 1 サイクルで最大 4 命令まで発行できる。

命令実行に必要な以下の資源は、このステージで割り当てられる。

■ コミットスタックエントリ (CSE)
■ 整数用と浮動小数点用のリネーミングレジスタ (GUB, FUB)
■ リザベーション・ステーションのエントリ
■ メモリアクセス用のポート

命令実行に必要な資源は命令ごとに異なるが、どの命令でも、実行に必要なすべての資源の割り当てはこのステージで行われる。通常の実行では、リザベーション・ステーションの解放は X ステージで、その他の資源の解放はパイプラインの最後の W ステージで行われる。E ステージから W ステージまでにある命令が実行中 (in-flight) の命令となる。例外が通知されると、実行中の命令は中止され、それらの命令が使用していた資源は直ちに解放される。これにより、デコーダは新しい命令の発行がすぐに行える。

3.2.3 実行ステージ

■ P: プライオリティ
■ B: バッファ読み出し
■ X: 実行
■ U: 更新

リザベーション・ステーションに格納された命令は、各々の実行条件が満たされると演算器に送られる。実行条件とは、入力データが確定する、演算器が空く、などであ る。実行にかかるサイクル数は命令によって異なる。
キャッシュアクセスの実行ステージ

メモリアクセス要求は、アドレス計算が完了すると、キャッシュアクセスユニットに送られる。キャッシュアクセスのステージは、分岐予測を除けば命令フェッチの各ステージと同じである。詳細はSection 3.2.1を参照。命令フェッチのステージ名とキャッシュアクセスのステージ名の対応を以下に示す。

<table>
<thead>
<tr>
<th>命令フェッチ</th>
<th>データアクセス</th>
</tr>
</thead>
<tbody>
<tr>
<td>IA</td>
<td>Ps</td>
</tr>
<tr>
<td>IT</td>
<td>Ts</td>
</tr>
<tr>
<td>IM</td>
<td>Ms</td>
</tr>
<tr>
<td>IB</td>
<td>Bs</td>
</tr>
<tr>
<td>IR</td>
<td>Rs</td>
</tr>
</tbody>
</table>

例外が通知されると、パイプラインにデータを送るための資源は解放されるが、キャッシュアクセスのパイプライン自体は、システムに要求したメモリアクセスを完了させるまでは動き続ける。データが返ってくると、キャッシュにストアされる。

3.2.4 命令完了ステージ

■ C: コミット
■ W: 書き込み

アウトオブオーダで実行された命令は、最後に命令順序通りに完了処理が行われる。例外の処理はこのステージで行われる。実行ステージで起きた例外は、すぐに処理されないのでなく、先行するすべての命令の完了処理が終わった後で通知される。

1. RAS関連の例外は完了を待たずに報告される。
Data Formats

JPS1 Commonality の Chapter 4 を参照。
JPS1 Commonality の Chapter 5 では、汎用レジスタ、ASR レジスタ、ASI レジスタの 3 種類を定義している。章立ては大きく、非特権レジスタ、特権レジスタの順番になっており、ASR, ASI レジスタは特権レジスタとして扱われているが、実際は非特権アクセス可能なものを混ざっているので適切さを欠いている。さらに、ASI レジスタについては、Chapter 5 の他、用途に応じていくつかの Appendix でも定義されており、この点でもまとまりを欠いている。

SPARC64™ IXf Extensions の章立てではできる限り JPS1 Commonality に合わせることにしているので、JPS1 Commonality の Chapter 5 で定義されているレジスタに関する実装依存仕様はこの章に記述することにする。ASI, ASR レジスタは特権・非特権が混ざるが、便宜上 Section 5.2, “Privileged Registers” に配置した。

ASI レジスタについては、以下のセクションも参照されたい。

- Appendix F.10, “Internal Registers and ASI Operations”
- Appendix L.3.2, “Special Memory Access ASIs”
- Appendix L.4, “ハードウェアバリア”
- Appendix M.3, “キャッシュ制御ASI”
- Appendix N.4, “Interrupt ASI Registers”
- Appendix P.2.5, “ASI_EIDR”
- Appendix P.2.6, “エラー検出の制御 (ASI_ERROR_CONTROL)”
- Appendix P.3.1, “ASI_STC Flag ERROR_INFO”
- Appendix P.4.1, “緊急エラーステータス (ASI_UGESR)”
- Appendix P.7.1, “ASI_ASYNC_FAULT_STATUS (ASI_AFSR)”
- Appendix R.1, “System Config Register”
- Appendix R.2, “STICK Control Register”

また、電源オン時やリセット時のレジスタ値については Appendix O.3, “リセット、RED_state 後のプロセッサ状態” (page 251) に、レジスタでエラーが発生した際の通知方法やエラー修復方法については Appendix P.8, “レジスタで起きたエラーの処理方法” (page 288) にまとめられている。
5.1 Nonprivileged Registers

5.1.1 General-Purpose r Registers

$r[32] - r[63] (xg[0] - xg[31])$ を追加する。

レジスタ番号は既存命令フィールドには入らないので、上位 1 ビットを
XAR.urs1, XAR.urs2, XAR.urs3, XAR.urd で指示する。拡張されるレジスタ数は
32 本なので、各フィールドの bit<2:1> は 0 になっていなければならない。
bit<2:1> が 0 でないときは、illegal_action 例外が通知される。

拡張数レジスタは大部分の命令で使用可能。使用できない命令の実行時に
XAR.v = 1 になっていると、illegal_action 例外が通知される。

$xg[0] - xg[31]$ は PSTATE.AG, PSTATE.MG, PSTATE.IG の値によらず同じものが見
える。

拡張レジスタに対する書き込みを行うと、XASR.xgd = 1 になる。

Programming Note – Context switch 時に save する必要があるかどうかが判断でき
る。

5.1.4 Floating-Point Registers

レジスタを追加し、全部で 256 本の倍精度レジスタが使えるようにする。追加するの
は倍精度レジスタで、$f[64] - f[510]$ (偶数番号のみ) で指定する。レジスタの追
加に伴い、レジスタの状態を表わす XASR も追加される。詳細は “Extended Arithmetic
Register Status Register (XASR) (ASR 30)” (page 32) 参照。

$f[0] - f[254]$ を Basic Floating-Point Registers、$f[256] - f[510]$ を Extended
Floating-Point Registers と呼ぶことになる。また $f[0] - f[62]$ を V9 Floating-Point
 Registers とも呼ぶ。

Floating-Point Register Number Encoding

JPS1 Commonality の同名の節で、命令で倍精度レジスタを指示する際のエンコー
ディングが定義されている。
拡張された256本の倍精度レジスタを指示するには、命令語のレジスタフィールド5ビットでは足りない。このため HPC-ACE では、レジスタフィールドの上位ビットを XAR レジスタに置き、命令実行時にそれらを結合してレジスタ番号を生成することにした。したがって、命令語だけではレジスタ番号は特定できない。

HPC-ACE 拡張レジスタ番号の生成は、XAR に置かれた上位 3 ビットと、FIGURE 5-1 に示したデコード後の 6 ビットを結合し 9 ビットの番号とする。最下位ビットは常に 0 になるので、\( f[0] - f[510] \) の偶数番号のみ、256 本のレジスタが指示できる。

倍精度レジスタの単精度利用

レジスタの追加に合わせて、倍精度レジスタを単精度浮動小数点演算でも使えるようにする。SPARC64 1Xfx で追加されたレジスタだけでなく、SPARC V9 で定義された倍精度レジスタも単精度浮動小数点演算に使える。単精度浮動小数点演算命令で倍精度レジスタを指定するには、その命令の実行時点で XAR.v = 1 であればよい。したがって、単精度でSIMD 演算を行うときは、倍精度レジスタを使うことになる。

倍精度レジスタを単精度浮動小数点演算に使用する際は、以下の点が SPARC V9 仕様と異なる。

- 命令のレジスタフィールドの解釈は、JPS1 Commonality の TABLE 5-5 の倍精度と同じになる。したがって偶数番号レジスタのみが使用できる。\( f[2n] \ (n = 0–255) \)
- レジスタの値は、bit<63:32> の上位 4 バイトを単精度値と解釈し、bit<31:0> の下位 4 バイトは無視される。
- 演算結果やロード結果はレジスタの上位 4 バイトに書き込まれ、下位 4 バイトには 0 が書き込まれる。
编程注意 – XAR.v = 1 かつ XAR.urs1 = 0 の場合、rs1 では SPARC V9 の
倍精度レジスタを使って単精度演算を行う。rs2, rs3, rd も同様である。このとき
bit<31:0> すなわち奇数番号レジスタの内容は破壊されることに注意すること。

エンディアン変換は単精度数、つまり 4 バイトを単位として行われる。

SIMD 拡張命令のレジスタ指定方法

浮動小数点レジスタを使う命令の大部分は、XAR.v = 1 かつ XAR.SIMD = 1 のときに
SIMD 拡張命令となり、1 命令で 2 演算を実行する。SIMD 演算で使われるレジスタの
組は、f [2n] に対して f [2n+256] (n = 0–127) に固定されている。命令では f [2n] 側を指定する。使えないレジスタを指定した場合は illegal_action が通知される。

SIMD FMADD 命令だけは例外で、rs1, rs2 に f[2n+256] を指定できる。詳細は
Appendix A.24.1, “Floating-Point Multiply-Add/Subtract” (page 70) を参照。

编程注意 – 単精度浮動小数点命令も SIMD 拡張できるが、その際は、倍
精度レジスタを単精度で使う。詳細は “倍精度レジスタの単精度利用” (page 21) を参
照。

既存浮動小数点命令のうち SIMD 拡張されない命令には以下の命令が含まれる。
SIMD 拡張可能な命令の一覧は TABLE A-2 を参照。

- FDIV(S,D), FSQRT(S,D)
- VIS のうち論理演算 (FOR, FAND, etc.) でないもの
- FBfcc, FBFfcc, FCMP, FCMPF, FMOVcc など、fcc, icc, xcc を参照、更新する命
令
- FMOVr

なお、SIMD の 2 演算を区別するときは、f [2n] に結果が格納される側の演算を
basic 側、f [2n+256] に結果が格納される側の演算を extended 側と呼ぶ。

エンディアン変換は basic, extended 毎に行われる。

5.1.7 Floating-Point State Register (FSR)

FSR_nonstandard_fp (NS)

SPARC V9 仕様では FSR.NS ビットを定義している。このビットを 1 にセットするこ
とで、浮動小数点演算器が IEEE Std 754-1985 に準拠しない演算処理を行っても良い
ことになっている。SPARC64 IXfx は FSR.NS ビットをサポートしている。
FSR.NS に 1 がセットされていると、演算の入力が非正規化数の場合や結果が非正規化数の場合に fp_exception_other が ftt = unfinished_FPop 通知されるのではなく、同符号の 0 に置き換えられて、fp_exception_ieee_754 が fsr.cexc.nxc = 1 で通知される (この例外は FSR.TEM.NXM でマスクすることができる)。詳細は Appendix B.6, "Floating-Point Nonstandard Mode" (page 139) を参照。

FSR.NS に 0 がセットされていると、IEEE Std 754-1985 に準拠して動作する。

FSR_version (ver)

個々の SPARC V9 IU 実装 (VER.impl で識別される) は、1 個以上の FPU を実装してもよいし実装しなくてもよい。このフィールドはこの CPU に実装されている FPU のバージョンを示す。SPARC64 1Xfx の最初の版では FSR.ver = 0 である (impl.dep. #19)。しかし、将来の版では別の値になっているかもしれない。詳細はデータシートを参照。

FSR_floating-point_trap_type (ftt)

SPARC64 1Xfx が fp_exception_other を ftt = unfinished_FPop で通知する条件は Appendix B.6.1, “fp_exception_other Exception (ftt=unfinished_FPop)” (page 140) を参照 (impl.dep. #248)。

FSR_current_exception (cexc)

ビット 4 から 0 は、IEEE 754 で定義された例外のうちどれが発生したかを示す。例外が発生していなければ、対応するビットには 0 がセットされる。

SPARC64 1Xfx が cexc ビットをどのようにセットするかを擬似コードで示す。

if (<LDFSR or LDXFSR commits>)
  <update using data from LDFSR or LDXFSR>;
else if (<FPop commits with ftt = 0>)
  <update using value from FPU>
else if (<FPop commits with IEEE_754_exception>)
  <set one bit1 in the CEXC field as supplied by FPU>;
else if (<FPop commits with unfinished_FPop error>)
  <no change>;
else if (<FPop commits with unimplemented_FPop error>)
  <no change>;
else
  <no change>;

1. non-SIMD 時は 1 ビットだが、SIMD では複数ビットセットされることがある。

Ver 12, 2 Dec. 2013
FSR Conformance

SPARC V9 では TEM, cexc, aexc のハードウェアでの実装方法を 2 種類許容している（どちらも IEEE Std 754-1985 に準拠している）。SPARC64 IXfx は (1), すなわちこれら 3 つを全部実装している。別の実装方法については JPS1 Commonality の Section 5.1.7 を参照。

SIMD 演算における cexc, aexc 更新

SIMD 演算では basic, extended 両方の演算が同時に行われる。入力データは basic, extended で異なるので、両方で例外が起きることもある。両方で例外が起きた場合、SPARC64 IXfx の SIMD 演算での例外通知と ftt の更新は以下のようになる。

1. basic, extended 両方で fp_exception_ieee_754 例外が検出された場合

説明のために、basic で起きた例外を保持する basic.cexc レジスタ、extended で起きた例外を保持する extend.cexc レジスタを仮定し、それぞれが, uf/of/dz/nx/nv のビットを持つものとする。

a. 両方とも例外通知がマスクされており、例外を通知しない場合

    fsr.cexc には basic.cexc と extend.cexc の論理和が表示され、fsr.aexc には basic.cexc と extend.cexc の論理和が反映される。

    fsr.cexc ← basic.cexc | extend.cexc
    fsr.aexc ← fsr.aexc | basic.cexc | extend.cexc

b. basic か extended のどちらかの例外が通知される場合

    fsr.cexc には basic.cexc と extend.cexc の論理和が表示される。
    fsr.aexc は更新されない。

    fsr.cexc ← basic.cexc | extend.cexc

   c. basic, extended 両方とも例外が通知される場合

      fsr.cexc には basic.cexc と extend.cexc の論理和が表示される。
      fsr.aexc は更新されない。

           fsr.cexc ← basic.cexc | extend.cexc

2. 一方で fp_exception_ieee_754, 他方で fp_exception_other が検出された場合

例外の優先順位と異なり、fp_exception_other 例外が
FSR.ftt = unfinished_FPop で通知される。FSR.aexc, FSR.cexc は更新されない。
Programming Note – *fp_exception_other* が通知された場合、同時に
*fp_exception_ieee_754* 例外が起きたかどうかを判断する方法はないので、システムソフトウェアのエミュレーションルーチンは IEEE に準拠した例外の検出と、関連するレジスタの更新も行うこと。

3. basic, extended 両方で *fp_exception_other* が検出された場合

*fp_exception_other* 例外が FSR.ftt = unfinished_FPop で通知される。
FSR.aexc, FSR.cexc は更新されない。

Note – non-SIMD 演算の場合、*fp_exception_ieee_754* が通知されるとときは fsr.cexc には 1 要因のみが表示される。SIMD 演算の場合、basic と extended の要因の論理和が表示されるので、複数要因が表示されることがある。

5.1.9 Tick (TICK) Register

SPARC64 IXfx は TICK.counter を 63 ビット幅で実装している (Impl. Dep. #105).

Implementation Note – SPARC64 IXfx では、TICK の読み出し時に counter に表示される値は、RDTICK 命令が実行されたときのものであり、完了したときのものではない (SPARC64 IXfx はアウトオブオーダ実行なので両者は明らかに異なる)。TICK の読み出しを 2 回行ったとき、counter の差分は、2 回の RDTICK 命令の実行の間の CPU サイクル数である。間に挟まる命令列が十分長ければ、実行の間の CPU サイクル数と完了の間の CPU サイクル数の差は小さいだろう。

5.2 Privileged Registers

5.2.6 Trap State (TSTATE) Register

SPARC64 IXfx は TSTATE.CWP のビット <2:0> のみを実装している。ビット 3 とビット 4 の書き込みは無視され、読み出しには 0 が返る。

Note – ソフトウェアで PSTATE.RED に 1 を設定すべきではない。RED_state に遷移する際に必要な、ハードウェア内部の処理をせずに、遷移する可能性があるためである。
5.2.9 Version (VER) Register

TABLE 5-1 に SPARC64 IXfx の VER の各フィールドと値を示す。

<table>
<thead>
<tr>
<th>ビット番号</th>
<th>フィールド</th>
<th>説明</th>
</tr>
</thead>
<tbody>
<tr>
<td>63-48</td>
<td>manuf</td>
<td>000416 (Impl. Dep. #104)</td>
</tr>
<tr>
<td>47:32</td>
<td>impl</td>
<td>9</td>
</tr>
<tr>
<td>31:24</td>
<td>mask</td>
<td>0 (n の値はプロセッサチップの版数による)</td>
</tr>
<tr>
<td>15:8</td>
<td>maxtl</td>
<td>5</td>
</tr>
<tr>
<td>4:0</td>
<td>maxwin</td>
<td>7</td>
</tr>
</tbody>
</table>

manuf フィールドには、JEDEC で定められた富士通を示す 8 ビットコードが表示され、上位 8 ビットは 0 となる。manuf, impl, mask フィールドの値はその性質上、将来のプロセッサでは変更されることがある。mask フィールドは論理回路が変更されるたびに大きい数字が入ることになるが、それが連続した数字である必要はない。

5.2.11 Ancillary State Registers (ASRs)

ASR の詳細については、JPS1 Commonality の Section 5.2.11 を参照。

Performance Control Register (PCR) (ASR 16)

SPARC64 IXfx の PCR 仕様は JPS1 Commonality 仕様と一部が異なる。FIGURE 5-2 と TABLE 5-2 で、JPS1 Commonality で定義された impl.dep. #207, #250 の SPARC64 IXfx での仕様と、PCR.SU, PCR.SL の JPS1 Commonality 仕様からの変更分を含む、PCR の仕様を説明する。PCR<2:1> は JPS1 Commonality に準拠している。

PA イベントカウンタの詳細は Appendix Q を参照。

FIGURE 5-2 PA 設定レジスタ (Performance Control Register, PCR) (ASR 16)
### TABLE 5-2  PCR のフィールド

<table>
<thead>
<tr>
<th>ビット</th>
<th>フィールド</th>
<th>説明</th>
</tr>
</thead>
</table>
| 47:32  | OVF        | オーバーフロー。RDPCRでカウンタのオーバーフローの情報が読み出せ、WRPCRでオーバーフロー情報をセット・クリアすることができる。
|        |            | PCR.OVF は SPARC64 IXfx 独自のフィールドである (impl. dep. #207)。
|        |            | OVF のビットとカウンタの対応は以下の通りである。OVF のビットに 0 を書き込むことで、対応するカウンタのオーバーフロー情報をクリアできる。
| 15     |            |ノードウェアが 1 を書き込んでも、オーバーフローの例外は通知されない。
| 26     | OVRO      | オーバーフロー情報の変更禁止。書き込みの際、書き込もうとするデータの OVRO が 0 だと、データの OVF で PCR.OVF が更新される。データの OVRO が 1 だと、データの OVF の値は無視され、PCR.OVF は書き込みによって更新されない。読み出し時には 0 が返る。
| 24:22  | NC        | カウンタペアの数。このフィールドはリードオンリーで、SPARC64 IXfx では 3 (カウンタペア 4 つ) である。PIC にマッピングするカウンタペアの指定。書き込みでは PIC のマッピングを更新し、読み出しでは現在のマッピング情報を返される。
| 20:18  | SC        | PIC にマッピングするカウンタペアの指定。書き込みでは PIC のマッピングを更新し、読み出しでは現在のマッピング情報が返される。 |

<table>
<thead>
<tr>
<th></th>
<th>L0</th>
<th>U0</th>
<th>L1</th>
<th>U1</th>
<th>L2</th>
<th>U2</th>
<th>L3</th>
<th>U3</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
PIC は JPS1 Commonality に準拠する。

SPIR64 IXfx では PIC は 4 組実装される。PCR.SC で指定された PIC が ASR 17 でアクセスできる。PIC のアクセスは PICU, PICL カウンタのペアにアクセスすることになる。PICU, PICL のエンコーディングについては Appendix Q を参照。

オーバーフロー発生時、カウンタは 0 にラップアラウンドし、SOFTINT レジスタのビット 15 に 1 がセットされ、interrupt level-15 例外が通知される。カウンタオーバーフローのトリップは、カウンタ値が FFFF FFFF から 0 に更新される際に通知される。同時に複数のオーバーラップが起きたときは、複数のオーバーラップビットが 1 にセットされる。すでに 1 にセットされているオーバーフロービットは 1 のまま変更されない。

オーバーフロービットは、ソフトウェアが対応する PCR.OVF に 0 を書き込むことでクリアされる。ソフトウェアは 1 を書き込むことができ、この書き込みによるオーバーフロートラップは発生しない。
Dispatch Control Register (DCR) (ASR 18)

SPARC64 IXfx は DCR を実装しない。DCR の読み出しには 0 が返り、書き込みは無視される。DCR は特権レジスタである。非特権モードでのアクセスには privileged_opcode 例外が通知される。

Extended Arithmetic Register (XAR) (ASR 29)

新規に追加された非特権レジスタである。命令フィールドを拡張するためのレジスタである。命令のレジスタ指定フィールド (rs1, rs2, rs3, rd) の上位 3 ビットと、SIMD 演算するかどうかを指定する。

XAR は 2 命令分のレジスタ指示フィールドを持つ。1 命令目、2 命令目用の各フィールドには V (valid) ビットがあり、その他のフィールドは v = 1 のときに有効である。レジスタフィールドには整数 / 浮動小数点数の区別はなく、任意のレジスタと結合する。

トラップ時には、XAR の内容は TXAR [TL] にセーブされ、XAR は all0 になる。セーブされるのは、実行中の命令の実行直前の XAR の値である。

Note – Tcc の条件が成立してトラップした時も、Tcc 実行直前の XAR の内容がセーブされる。

### TABLE 5-3 XAR のフィールド

<table>
<thead>
<tr>
<th>ビット</th>
<th>フィールド</th>
<th>説明</th>
</tr>
</thead>
<tbody>
<tr>
<td>63:32</td>
<td>— Reserved. このフィールドに 0 以外の値を書くと、illegal_instruction 例外が通知される。</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>f_v</td>
<td>f_ で始まるフィールドの内容が有効かどうかを示す。f_v = 1 なら、1 番目に実行される命令に f_ が適用される。1 番目の命令が完了すると、f_ フィールドはすべて 0 クリアされる。</td>
</tr>
<tr>
<td>30:29</td>
<td>— Reserved. このフィールドに 0 以外の値を書くと、illegal_instruction 例外が通知される。</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>f_simd</td>
<td>f_simd = 1 なら、1 番目に実行される命令は SIMD 命令として実行される。f_simd = 0 なら non-SIMD で実行される。</td>
</tr>
<tr>
<td>27:25</td>
<td>f_urd</td>
<td>1 番目の命令の rd の拡張フィールド。</td>
</tr>
<tr>
<td>24:22</td>
<td>f_urs1</td>
<td>1 番目の命令の rs1 の拡張フィールド。</td>
</tr>
</tbody>
</table>
本仕様書における XAR の記述について

Table 5-3 のフィールド名以外に以下の別名表記も用いる。

### メモリアクセス用

<table>
<thead>
<tr>
<th>別名</th>
<th>実際のフィールド</th>
</tr>
</thead>
<tbody>
<tr>
<td>XAR.f_dis_hw_pf</td>
<td>XAR.f_urs3&lt;1&gt;</td>
</tr>
<tr>
<td>XAR.s_dis_hw_pf</td>
<td>XAR.s_urs3&lt;1&gt;</td>
</tr>
<tr>
<td>XAR.f_sector</td>
<td>XAR.f_urs3&lt;0&gt;</td>
</tr>
<tr>
<td>XAR.s_sector</td>
<td>XAR.s_urs3&lt;0&gt;</td>
</tr>
</tbody>
</table>

### SIMD FMA 用

<table>
<thead>
<tr>
<th>別名</th>
<th>実際のフィールド</th>
</tr>
</thead>
<tbody>
<tr>
<td>XAR.f_negate_mul</td>
<td>XAR.f_urd&lt;2&gt;</td>
</tr>
<tr>
<td>XAR.s_negate_mul</td>
<td>XAR.s_urd&lt;2&gt;</td>
</tr>
<tr>
<td>XAR.f_rs1_copy</td>
<td>XAR.f_urs3&lt;2&gt;</td>
</tr>
<tr>
<td>XAR.s_rs1_copy</td>
<td>XAR.s_urs3&lt;2&gt;</td>
</tr>
</tbody>
</table>
総称
1 命令目、2 命令目の区別なしに記述する場合は XAR.f_v, XAR.s_v の値により 1 命令目、2 命令目のどちらかを選択したものとする。

<table>
<thead>
<tr>
<th>記述法</th>
<th>XAR.f_v = 1 のとき</th>
<th>XAR.f_v = 0 かつ XAR.s_v = 1 のとき</th>
</tr>
</thead>
<tbody>
<tr>
<td>XAR.urd</td>
<td>XAR.f_urd</td>
<td>XAR.s_urd</td>
</tr>
<tr>
<td>XAR.urs1</td>
<td>XAR.f_urs1</td>
<td>XAR.s_urs1</td>
</tr>
<tr>
<td>XAR.urs2</td>
<td>XAR.f_urs2</td>
<td>XAR.s_urs2</td>
</tr>
<tr>
<td>XAR.urs3</td>
<td>XAR.f_urs3</td>
<td>XAR.s_urs3</td>
</tr>
<tr>
<td>XAR.dis_hw_pf</td>
<td>XAR.f_dis_hw_pf</td>
<td>XAR.s_dis_hw_pf</td>
</tr>
<tr>
<td>XAR.sector</td>
<td>XAR.f_sector</td>
<td>XAR.s_sector</td>
</tr>
<tr>
<td>XAR.negate_mul</td>
<td>XAR.f_negate_mul</td>
<td>XAR.s_negate_mul</td>
</tr>
<tr>
<td>XAR.rs1_copy</td>
<td>XAR.f_rs1_copy</td>
<td>XAR.s_rs1_copy</td>
</tr>
</tbody>
</table>

XAR 動作
XAR を参照する命令と参照しない命令がある。
ここでは XAR を参照する命令のことを、「XAR 対象命令」と呼ぶ。どの命令が XAR 対象命令かは、TABLE A-2 (page 59) を参照。

- XAR 対象でない命令は、その命令の実行時に XAR.v = 1 だと illegal_action 例外が通知される。
- XAR 対象命令の動作は、
  - XAR.v = 1 のとき、XAR.urs1, XAR.urs2, XAR.urs3, XAR.urd がそれぞれ命令フィールドの rs1, rs2, rs3, rd と結合する。
    整数レジスタでは、XAR のフィールドを上位 3 ビット、命令フィールドを下位 5 ビットとする計 8 ビットで指定されるレジスタを使用する。
    浮動小数点レジスタでは、XAR のフィールドを上位 3 ビットとし、命令フィールドの 5 ビットを倍精度レジスタのエンコーディング方式にしたがってコードした 6 ビットを下位とする計 9 ビットで指定されるレジスタを使用する。浮動小数点レジスタのエンコーディングの詳細は “Floating-Point Register Number Encoding” (page 20) を参照。
  - XAR.f_v = 1 なら XAR.f_urs1, XAR.f_urs2, XAR.f_urs3, XAR.f_urd が使われる。
  - XAR.f_v = 0 かつ XAR.s_v = 1 なら XAR.s_urs1, XAR.s_urs2, XAR.s_urs3, XAR.s_urd が使われる。
XAR の値は 1 回のみ有効。XAR を参照した命令が完了すると、XAR の関連フィールドは all0 になる。

XAR 対象命令であっても、以下のどれかにあたる場合 illegal_action が通知される。
- xg[32] 以上の整数レジスタを指定した場合。
- rs1 を使わない命令に対して、urs1 ≠ 0 を指定した場合。
- rs2, rs3, rd についても同様である。
  rs2 フィールドを simm13 や fcn などの即値として使う命令で、urs2 ≠ 0 の場合も illegal_action が通知される。
- FDIV(S,D), FSQRT(S,D) で rd に f[256] 以上を指定した場合。
- SIMD 拡張されない命令（整数演算を含む）に対して XAR.simd = 1 を指示した場合。
- XAR.simd = 1 で f[256] 以上を指定した場合。
  ただし FMADD は、rs1, rs2 に f[256] 以上を指定可能。
  XAR.urs3<2>, XAR.urd<2> も 1 になってもよいが、これは f[256] 以上の指定ではなく、別の意味に解釈される。詳細は "FMADD の SIMD 拡張" (page 73) を参照。
- ld/st/atomic 命令で XAR.urs3<2> ≠ 0 のとき。

後続 1 命令のレジスタ番号を指示する場合、1st を使っても 2nd を使ってもよい。

Programming Note – WRXAR 命令を使えば XAR.f_v, XAR.s_v どちらでも任意に 1 を設定できる。sxar1 命令では XAR.f_v = 1 になる。

XAR.f_v = 0 のときは、f_simd, f_urs1, f_urs2, f_urs3, f_urd の各フィールドに 0 でない値があっても無視される。命令実行後、各フィールドの値がどうなるかは未定義である。
XAR.s_v = 0 のときは、s_simd, s_urs1, s_urs2, s_urs3, s_urd の各フィールドに 0 でない値があっても無視される。命令実行後各フィールドの値がどうなるかは未定義である。

Extended Arithmetic Register Status Register (XASR) (ASR 30)
新規追加非特権レジスタ。

<table>
<thead>
<tr>
<th></th>
<th>xgd</th>
<th>xfd&lt;7:0&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>63</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>0</td>
</tr>
</tbody>
</table>
コンテキストスイッチ時に、レジスタを保存する必要があるかどうかを判断するために使用する。拡張レジスタを更新すると、対応するビットが 1 になる。

■ V9 定義の整数レジスタの更新を示すフラグはない。
■ xg[0] 〜 xg[31] を更新すると XASR.xgd = 1 がセットされる。
■ 拡張小数点レジスタを更新すると、対応する xf<i> = 1 がセットされる。レジスタと xf<i> のビット対応は以下の通り。

<table>
<thead>
<tr>
<th>ビット</th>
<th>フィールド</th>
<th>属性</th>
<th>説明</th>
</tr>
</thead>
<tbody>
<tr>
<td>63:9</td>
<td>—</td>
<td>R</td>
<td>Reserved.</td>
</tr>
<tr>
<td>8</td>
<td>xgd</td>
<td>RW</td>
<td>xg[0] 〜 xg[31] が更新されると、xgd = 1 になる。</td>
</tr>
<tr>
<td>7:0</td>
<td>xf&lt;i&gt;</td>
<td>RW</td>
<td>浮動小数点レジスタが更新されると、対応するビットが 1 がセットされる。</td>
</tr>
</tbody>
</table>

浮動小数点レジスタ | 対応する XASR のビット
---|---
 f[0] 〜 f[62] | xf<i> |
 f[64] 〜 f[126] | xf<i> |
 f[128] 〜 f[190] | xf<i> |
 f[192] 〜 f[254] | xf<i> |
 f[256] 〜 f[318] | xf<i> |
 f[320] 〜 f[382] | xf<i> |
 f[384] 〜 f[446] | xf<i> |
 f[448] 〜 f[510] | xf<i> |

**Programming Note** — xf<i> は V9 FPR を更新したときにセットされる。このとき、V9 の FPRS も同時に更新される。例えば f[15] を更新した場合は、FPRS.d1 = 1 と XASR.xfd<i> = 1 がセットされる。

**Implementation Note** — MOVr, MOVcc, FMOVr, FMOVcc 命令で条件が不成立のとき、対応する XASR のビットが 1 になるかどうかは実装に依存する。

**Trap XAR Registers (TXAR) (ASR 31)**
新規追加特権レジスタ。フィールドは XAR と同じである。

TL = 0 で read/write すると illegal_instruction が通知される。Reserved フィールドに 0 でない値を書くと illegal_instruction が通知される。

5.2.12 Registers Referenced Through ASIs

本節では JPS1 Commonality の 5.2.12 で定義されている ASI レジスタについてのみ記述する。その他の ASI レジスタについては Appendix L を参照。

Data Cache Unit Control Register (DCUCR)

ASI 4516 (ASI_DCU_CONTROL_REGISTER), VA = 0016。

DCUCR は、命令、プリフェッチ、書き込みとデータキャッシュ、MMU、ウォッチポイントなど、メモリアクセスに関連するハードウェアの機能を制御するためのレジスタである。SPARC64 IXfx の DCUCR は JPS1 Commonality で定義された仕様をほぼ実装している。DCUCR のビット配置を FIGURE 5-3 に、ビットの説明を TABLE 5-4 に示す。

| 63 | 50 | 49 | 48 | 47 | 42 | 41 | 40 | 33 | 32 | 25 | 24 | 23 | 22 | 21 | 20 | 4 | 3 | 2 | 1 | 0 |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
|    | 0  | 0  | 0  | WEAK_SPCA |    | VM | PR | PW | VR | VW |    | DM | IM | 0  | 0  |

FIGURE 5-3 DCUCR (ASI 4516)
Implementation Note – DCUCR.WEAK_SPCA のときに、CTI 以降の命令プリフェッチは、最大でもその CTI から 1KB を越えることはない。

<table>
<thead>
<tr>
<th>ビット</th>
<th>フィールド</th>
<th>属性</th>
<th>説明</th>
</tr>
</thead>
<tbody>
<tr>
<td>63:50</td>
<td>—</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>49:48</td>
<td>CP, CV</td>
<td>R</td>
<td>SPARC64 Ixf では実装されない。読み出しには 0 が返り、書き込みは無視される。</td>
</tr>
<tr>
<td>47:42</td>
<td>impl. dep.</td>
<td>R</td>
<td>読み出しには 0 が返り、書き込みは無視される。</td>
</tr>
<tr>
<td>41</td>
<td>WEAK_SPCA</td>
<td>RW</td>
<td>投機的なメモリアクセスの抑止（impl. dep. #240）。 WEAK_SPCA に 1 がセットされると、分岐予測が行われなくなる。命令のプリフェッチは常に分岐しない方向に行われる。分岐命令以降のロード、ストアは、分岐方向が確定するまで行われない。ハードウェアプリフェッチ機構は停止され、プリフェッチ命令はストロングプリフェッチも含めてすべて無効となる。</td>
</tr>
</tbody>
</table>

命令プリフェッチのアクセス範囲は CPU 内部リソースによって決まる一定範囲内に留まるので、weak_spca = 1 ではメモリアクセスの範囲は推定可能である。

| 40:33 | PM<7:0>   | RW   | Data Watchpoint Register のマスクを指定する。 SPARC64 Ixf では Data Watchpoint Register は物理アドレス、論理アドレス共用。 |
| 32:25 | VM<7:0>   | RW   | Data Watchpoint Register の値を物理アドレスと解釈して VM で指定されたアドレスに read または write アクセスすると PA_watchpoint 例外が通知される。 |
| 24, 23 | PR, FW    | RW   | Data Watchpoint Register の値を物理アドレスと解釈して VM で指定されたアドレスに read または write アクセスすると PA_watchpoint 例外が通知される。 |
| 22, 21 | VR, VW    | RW   | Data Watchpoint Register の値を論理アドレスと解釈して VM で指定されたアドレスに read または write アクセスすると VA_watchpoint 例外が通知される。 |
| 20:4  | —         | RW   | Reserved |
| 3     | DM        | RW   | Data MMU 有効化。DM を 0 にセットすると、データアクセス時にアドレス変換が行われない。このとき、論理アドレスはそのまま物理アドレスとして使われる。 |
| 2     | IM        | RW   | 命令 MMU 有効化。IM を 0 にセットすると、データアクセス時にアドレス変換が行われない。このとき、論理アドレスはそのまま物理アドレスとして使われる。 |
| 1     | DC        | R    | SPARC64 Ixf では実装されていない。読み出しには 0 が返り、書き込みは無視される。 |
| 0     | IC        | R    | SPARC64 Ixf では実装されていない。読み出しには 0 が返り、書き込みは無視される。 |

Implementation Note – DCUCR.WEAK_SPCA = 1 のとき、CTI 以降の命令プリフェッチは、最大でもその CTI から 1KB を越えることはない。
Programming Note – 投機メモリアクセス抑止を確実に行うために、システムソフトウェアは DCUCR.WEAK_SPCA = 1 に設定後直ちに membar #Sync を発行すること。

Programming Note – SPARC64 IXfx では DCUCR の IM (IMMU enable), DM (DMMU Enable) 変更は以下の命令列で行う必要がある。

# DCUCR.IM update
stxa DCUCR
flush

# DCUDR.DM update
stxa DCUCR
membar #sync

Data Watchpoint Registers

<table>
<thead>
<tr>
<th>レジスタ名</th>
<th>ASI_WATCHPOINT</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASI</td>
<td>58_16</td>
</tr>
<tr>
<td>VA</td>
<td>38_16</td>
</tr>
<tr>
<td>アクセス種別</td>
<td>Supervisor Read/Write</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ビット</th>
<th>DB</th>
<th>属性</th>
<th>説明</th>
</tr>
</thead>
<tbody>
<tr>
<td>63:3</td>
<td>3</td>
<td>RW</td>
<td>ウォッチポイントアドレス (VA, PA 共用)</td>
</tr>
</tbody>
</table>

JPS1 Commonality の TABLE 5-18 で、ウォッチポイントの対象は translating ASI, bypass ASI、であると定義している。しかしこの表の定義に従うと、実装依存 ASI や未定義 ASI もウォッチポイントの対象となってしまうので、SPARC64 IXfx では translating, bypass, nontranslating を定義し直した (TABLE L-1 (page 214) を参照)。この表で translating, bypass と定義された ASI がウォッチポイントの対象となる。

JPS1 Commonality ではウォッチポイントを、論理アドレス・物理アドレス独立に設定できるが、SPARC64 IXfx では、アドレスの指定は一つだけで、そのアドレスを論理アドレス・物理アドレスと解釈した場合にマッチするかどうかを監視するよう、仕
様を変更する。JPS1 Commonality の ASI_VA_WATCHPOINT (ASI = 58_{16}, VA = 38_{16}) を ASI_WATCHPOINT という名前にし、ASI_PA_WATCHPOINT (ASI = 58_{16}, VA = 40_{16}) は削除する。

Compatibility Note – この変更は SPARC JPS1 非互換である。

ウォッチポイントの有効・無効の指定方法は、SPARC JPS1 仕様に準拠し、DCUCR. VR, DCUCR. VN, DCUCR. PR, DCUCR. PW で指定する。DCUCR. VR または DCUCR. VN に 1 がセットされていると、DB を論理アドレスと解釈してマッチした場合に VA_watchpoint が通知され、DCUCR. PR または DCUCR. PW に 1 がセットされていると、DB を物理アドレスと解釈してマッチした場合に PA_watchpoint が通知される。論理アドレス・物理アドレスどちらでもマッチした場合は、VA_watchpoint が通知される。

ウォッチポイントが有効である ASI 空間の一覧が JPS1 Commonality の TABLE 5-18 で定義されているが、この表は SPARC64 IXfx で未実装である ASI も含まれている。SPARC64 IXfx は、未実装である translating ASI ではウォッチポイントの検出は行われず、アクセスすると data_access_exception が通知される。

物理アドレスとしてマッチしているかどうかを判定する際、DB<63:41> は無視される。

SIMD ロード、SIMD ストア命令では、basic 側、extended 側のどちらでもウォッチポイントを検出する。アドレスとマスクが basic 側のアドレスとアクセス長にマッチすれば、basic 側の VA_watchpoint, PA_watchpoint が通知され、アドレスとマスクが extended 側のアドレスとアクセス長にマッチすれば、extended 側の VA_watchpoint, PA_watchpoint が通知される。

ウォッチポイントの信頼性を下げる機能は SPARC64 IXfx には実装されていない（impl. dep. #244）。

以下の命令については、ウォッチポイントの設定と検出に関する特例がある。詳細はそれぞれの命令を参照。

- Appendix A.4, “Block Load and Store Instructions (VIS I)”
- Appendix A.30, “Load Quadword, Atomic [Physical]”
- Appendix A.42, “Partial Store (VIS I)”
- Appendix A.77, “Store Floating-Point Register on Register Condition”
- Appendix A.79, “Cache Line Fill with Undetermined Values”
- Appendix F.5.1, “Trap Conditions for SIMD Load/Store”

Instruction Trap Register

SPARC64 IXfx は命令トラップレジスタを実装する（impl. dep. #205）。

Ver 12, 2 Dec. 2013  F. Chapter 5  Registers  37
5.2.13 Floating-Point Deferred-Trap Queue (FQ)

SPARC64 IXfx は浮動小数点演算の遅延トラップキューを持たない (impl. dep. #24). RDPR 命令で FQ を読み出そうとすると、illegal_instruction 例外を通知する (impl. dep. #25).

5.2.14 IU Deferred-Trap Queue

SPARC64 IXfx は整数ユニットの遅延トラップキーを持たない (impl. dep. #16).
6.1 Instruction Execution

SPARC64 IXfx is a processor that implements advanced single-core architecture. SPARC V9 processors can execute multiple instructions per cycle. SPARC64 IXfx's instruction set is a next-in-line sequential model, so the instructions in the pipeline must be committed in order. If an instruction sequence is not committed, the instruction sequence is flushed. SPARC64 IXfx's instruction set is as follows:

1. When an operation x is volatile, SPARC64 IXfx does not perform a prefetch at location[x]. Location[x] is committed to be executed when x is committed.
2. When an operation x is nonvolatile, SPARC64 IXfx may perform a prefetch at location[x]. A prefetch is performed under the following conditions:
   a. If the operation x is an atomic operation, the operation is flushed from the processor.
   b. If the operation x is not an atomic operation, the operation is flushed from the processor.

SPARC64 IXfx is designed to prevent speculative load instructions.
1. ある種のページや I/O 空間に対しては投機アクセスを行わない。投機的なアクセスを禁止する場合は、PTE の E (side-effect) ビットに 1 をセットする。E ビットがセットされているページへのアクセスは、そのアクセスが投機的でなくなるまで待機させられる。詳細は Appendix F を参照。

2. ロードを強制的にプログラム順に行わせる ASI_PHYS_BYPASS_WITH_EBIT[_L] (ASI = 15_16, 1D16) は投機的には実行されない。

6.1.2 Instruction Prefetch

SPARC64 IXfx は、命令の供給が間に合わず実行が止まるということが極力起こらないよう、命令プリフェッチを行う。分岐予測の結果によっては、実際には実行しない命令列をプリフェッチすることがある。さらに、投機実行した命令がメモリアクセスをすることもある。命令プリフェッチや投機メモリアクセスによる例外を検出してもすぐに通知せず、その命令の直前の命令が完了してから通知される。

6.1.3 Syncing Instructions

SPARC64 IXfx はある種の命令を sync 命令として扱う。sync 命令は実行するとパイプラインを数サイクル間停止させるような命令である。sync 命令には pre sync と post sync の 2 種類がある。pre sync 命令は、プログラム順でその命令より前にある命令の実行が完了してから、その命令だけを実行完了し、その後で、後続命令を実行するような副次的効果を持つ命令である。post sync 命令は、プログラム順でその命令より後にある命令の発行を、その命令が実行完了するまで停止する副次的効果を持つ命令である。pre sync かつ post sync 属性を持つ命令も存在する。

SPARC64 IXfx では、ストアを除くすべての命令がプログラム順に完了する。ストアはその結果が global visible になる前に完了する（ストアの突き放し完了）。

6.2 Instruction Formats and Fields

SPARC64 IXfx の命令フォーマットは 5 種類に大別される。そのうち 4 種類の命令フォーマットは JPS1 Commonality の Section 6.2 を参照。5 番目の命令フォーマットは SPARC64 IXfx 独自のフォーマットである。

---

1. ハードウェアのエラーやその他命令実行とは非同期に起きるエラーは、そのエラーを起こした命令がの完了を待たずに通知される。
**Instructions**

6.3 Instruction Categories

6.3.3 Control-Transfer Instructions (CTIs)

実行制御命令には以下の種類がある。

- 条件分岐 (Bicc, BPr, FBfcc, FBPrfcc)
- 無条件分岐
- Call and link (CALL)
- Jump and link (JMPL, RETURN)

---

*Format 5 (op = 2, op3 = 3716): FMADD, FPMADDX, FSELMOV, and FTRIMADD (in place of IMPDEP2A and IMPDEP2B)*

<table>
<thead>
<tr>
<th>op</th>
<th>rd</th>
<th>op3</th>
<th>rs1</th>
<th>rs3</th>
<th>var</th>
<th>size</th>
<th>rs2</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>25</td>
<td>24</td>
<td>19</td>
<td>18</td>
<td>14</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>op</th>
<th>rd</th>
<th>op3</th>
<th>rs1</th>
<th>index</th>
<th>var</th>
<th>size</th>
<th>rs2</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>25</td>
<td>24</td>
<td>19</td>
<td>18</td>
<td>14</td>
</tr>
</tbody>
</table>

**TABLE 6-1**  Format 5 の命令フィールドの説明

<table>
<thead>
<tr>
<th>フィールド</th>
<th>説明</th>
</tr>
</thead>
<tbody>
<tr>
<td>rs3</td>
<td>この 5 ビットフィールドは、浮動小数点演算の 3 番目の入力浮動小数点レジスタを示す。</td>
</tr>
<tr>
<td>var</td>
<td>この 2 ビットフィールドは、浮動小数点積加演算の種類を指示するために使われるほか、Impdep2 に定義された命令を選択するために使われる。</td>
</tr>
<tr>
<td>size</td>
<td>この 2 ビットフィールドは、浮動小数点積加演算のサイズを指示するために使われるほか、Impdep2 に定義された命令を選択するために使われる。</td>
</tr>
<tr>
<td>index</td>
<td>FTRIMADD4 で係数テーブルを指示するために使われる。</td>
</tr>
</tbody>
</table>

**FIGURE 6-1**  Summary of Instruction Formats: Format 5
Return from trap (DONE, RETRY)
Traps (Tcc)

この仕様書では SPARC64 IXfx の CALL と JMPL 命令について説明する。それ以外の実行制御命令は JPS1 Commonality を参照。

CALL and JMPL Instructions

SPARC64 IXfx は PSTATE.AM = 0 のとき、PC の 64 ビットすべてをデスティネーションレジスタに書き込む。PSTATE.AM = 1 のときは上位 32 ビットをゼロにした値をデスティネーションレジスタに書き込む。

6.3.7 Floating-Point Operate (FPop) Instructions

FPop が fp_exception_other 例外を FSR.ftt = unfinished_FPop で通知する正確な条件は、Appendix B.6, "Floating-Point Nonstandard Mode" (page 139) で定義する。

6.3.8 Implementation-Dependent Instructions

SPARC64 IXfx は IMPDEP1、IMPDEP2 に、浮動小数点演算命令を定義している。JPS1 Commonality では FPop を "オペコードが FPop1 または FPop2 である命令" と定義しているので、IMPDEP の命令は FPop ではないことになる。

IMPDEP2 で定義された浮動小数点積和演算命令のうち FMADD, FMSUB, FNMSUB には 4 倍精度演算が定義されているが、SPARC64 IXfx では 4 倍精度演算は実装されていないので、これらの命令を実行すると illegal_instruction 例外が通知される。FMADD だけは 4 倍精度演算が定義されていない。4 倍精度浮動小数点積和演算は SPARC V9 で必須の命令ではないので、システムソフトウェアはエミュレーションしなくてもよい。

SPARC64 IXfx で IMPDEP1, IMPDEP2 に追加された命令のうち、浮動小数点レジスタを使う以下の命令は、PSTATE.PEF = 0 または FPRS.FEF = 0 のときに実行すると、fp_disabled 例外が通知される。

FCMP(<GT, LE, EQ, NE, GE, LE>)E(s,d), FCMP(EQ,NE)(s,d), FMAX(s,d), FMIN(s,d),
FRCPA(s,d), FRSQRTA(s,d), FTRISSELd, FTRISMULd, FTRIMADDd,
FSELMOV(s,d), F{N}M(ADD, SUB)(s,d), FPMADDX{HI}, ST{D}FR

これらの FPop ではないので、Reserved であるオペコードを実行しようとするとき、JPS1 Commonality 6.3.9 で定義されている通り、illegal_instruction 例外が通知される。また、これらの命令のうち FPMADDX{HI}, ST{D}FR 以外は、JPS1 Commonality 6.3.7 の FPop と同様に FSR を更新する。FTRISSELd, FSELMOV(s,d) は fp_exception_ieee_754 例外を通知しない命令だが、実行が完了すると FSR.cexc の全フィールドに 0 をセットし、FSR.aexc は変更しない。
7.1 Processor States, Normal and Special Traps

JPS1 Commonality ではこの節で CPU のステートと遷移を定義しているが、SPARC64™ IXfx Extensions では Appendix O.1, “リセット種類” (page 247) で定義しているので、そちらを参照されたい。

7.1.1 RED_state

Appendix O.2.1, “RED_state” (page 250) も参照。

RED_state Trap Table

RED_state 用のトラップテーブルは実装依存のアドレス RSTVaddr に配置される。RSTVaddr の値はある実装に対しては固定となっており、SPARC64 IXfx では論理アドレス FFFF FFFF F000 0000_{16}、物理アドレス 0000 01FF F000 0000_{16} である (impl. dep. #114)。

RED_state Execution Environment

RED_state では CPU はある制約下で動作するが、このためにいくつかの CPU 制御用レジスタの値は固定値で上書きされる。

Note – 遷移をアトミックに行うため、値は上書きであって代入ではない。

RED_state での SPARC64 IXfx の動作は以下の通りである (impl. dep. #115)
RED_state である間は、ITLB によるアドレス変換機能は無効となる。DTLB によるアドレス変換は RED_state 遷移直後は無効だが、RED_state 中にソフトウェアによって有効にすることができる。RED_state 中でも、ASI レジスタ経由で TLB を参照・変更する機能は有効である。

- TLB によるアドレス変換が無効である間、すべてのメモリアクセスはノンキャッシュ、ストロングオーダとして扱われる。

- XIR はマスクされないので、XIR を受けると例外を通知する。

Note – CPU 内のエラーにより RED_state に入ったら場合、ソフトウェアは致命的なエラーからの復旧やエラーを起こしたコンポーネントを無効にするなどの処理を行うべきである。リセット後に RED_state に入った場合、ソフトウェアはシステムを実行状態にするためのセットアップを行うこと。

7.1.2 error_state

プロセッサは、トラップレベルが最大 (TL = MAXTL) のときにトラップが起きると error_state に遷移する (impl. dep. #39)。

CPU は、CPU 内部で生成される watchdog reset (WDR) によって error_state に遷移するが、WDR を抑止し、error_state で留まるように設定することもできる (impl. dep #40, #254)。

7.2 Trap Categories

7.2.2 Deferred Traps

SPARC64 IXfx ではエラー処理の例外が deferred で通知されることがある。詳細は Appendix P.2.2, “エラー検出時の動作” (page 263) または Appendix P.4.3, “ADE トラップ発生した時の命令の実行状況” (page 283) を参照。

7.2.4 Reset Traps

SPARC64 IXfx は命令完了が約 6.7 秒の間行われない場合、watchdog reset (WDR) 例外を通知する。
7.2.5 Uses of the Trap Categories

SPARC64 IXfx では、命令実行に同期して起きる例外はすべて precise である（impl. dep. #33）。

複数のメモリアクセスとして実行される命令（LDD(A), STD(A), LDSTUB, CASA, CASXA, SWAP など）が、最初のアクセスでの致命的なエラーを起こした場合、precise で通知される。

7.3 Trap Control

7.3.1 PIL Control

SPARC64 IXfx はシステムからのインタラプト要求を受信すると、interrupt_vector_trap (TT = 6016) を通知する。トラップハンドラはインタラプト情報を読んだ後、SPARC V9 準拠のインタラプトで通知します。この通知はソフトウェアが SOFTINT を書き込むことで行われる。詳細は JPS1 Commonality の Section 5.2.11 を参照。

SPARC64 IXfx は SPARC V9 準拠のインタラプトを受けると、PIL レジスタの値をチェックする。優先順位の条件を満たすインタラプトがある場合、SPARC64 IXfx は新しい命令の発行をやめ、未完了の命令をすべてキャンセルし、トラップハンドラに処理を移す。ただし、実行中の命令列が、より優先順位が高いトラップハンドラのものである場合はこの限りではない。

SPARC64 IXfx はインタラプト要求を disrupting トラックとして処理する。

7.4 Trap-Table Entry Addresses

7.4.2 Trap Type (TT)

SPARC64 IXfx では以下のトラップを定義している（impl. dep. #35, impl. dep. #36）。

- async_data_error
- illegal_action
- SIMD_load_across_pages
JPS1 Commonality のトラップ一覧にこれらを含めたものを TABLE 7-1, TABLE 7-2 に示す。TABLE 7-1 の網掛け部分は、SPARC64 1Xfx では起こらないトラップである。

<table>
<thead>
<tr>
<th>SPARC V9 M/O</th>
<th>JPS1 M/O</th>
<th>トラップ名</th>
<th>TT</th>
<th>% e 種類</th>
<th>優先順位</th>
</tr>
</thead>
<tbody>
<tr>
<td>● ●</td>
<td>Reserved</td>
<td>00016</td>
<td>-NA-</td>
<td>-NA-</td>
<td></td>
</tr>
<tr>
<td>● ●</td>
<td>power_on_reset</td>
<td>00116</td>
<td>AG</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>○ ●</td>
<td>watchdog_reset</td>
<td>00216</td>
<td>AG</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>○ ●</td>
<td>externally_initiated_reset</td>
<td>00316</td>
<td>AG</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>● ●</td>
<td>software_initiated_reset</td>
<td>00416</td>
<td>AG</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>● ●</td>
<td>RED_state_exception</td>
<td>00516</td>
<td>AG</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>● ●</td>
<td>Reserved</td>
<td>00616-00716</td>
<td>-NA-</td>
<td>-NA-</td>
<td></td>
</tr>
<tr>
<td>● ●</td>
<td>instruction_access_exception</td>
<td>00816</td>
<td>MG</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>○ ○</td>
<td>instruction_access_MMU_miss</td>
<td>00916</td>
<td>MG (impl. dep.)</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>○ ●</td>
<td>instruction_access_error</td>
<td>00A16</td>
<td>AG</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>○ ●</td>
<td>Reserved</td>
<td>00B16-00F16</td>
<td>-NA-</td>
<td>-NA-</td>
<td></td>
</tr>
<tr>
<td>● ●</td>
<td>illegal_instruction</td>
<td>01016</td>
<td>AG</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>● ●</td>
<td>privileged_opcode</td>
<td>01116</td>
<td>AG</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>○ ○</td>
<td>unimplemented_LDD</td>
<td>01216</td>
<td>AG</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>○ ○</td>
<td>unimplemented_STD</td>
<td>01316</td>
<td>AG</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>● ●</td>
<td>Reserved</td>
<td>01416-01F16</td>
<td>-NA-</td>
<td>-NA-</td>
<td></td>
</tr>
<tr>
<td>● ●</td>
<td>fp_disabled</td>
<td>02016</td>
<td>AG</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>○ ●</td>
<td>fp_exception_ieee_754</td>
<td>02116</td>
<td>AG</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>○ ●</td>
<td>fp_exception_other</td>
<td>02216</td>
<td>AG</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>(fit = unimplemented_FPop のみ)</td>
<td>02216</td>
<td>AG</td>
<td>8.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>● ●</td>
<td>tag_overflow</td>
<td>02316</td>
<td>AG</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>○ ●</td>
<td>clean_window</td>
<td>02416-02716</td>
<td>AG</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>● ●</td>
<td>division_by_zero</td>
<td>02816</td>
<td>AG</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>○ ○</td>
<td>internal_processor_error</td>
<td>02916</td>
<td>impl. dep.</td>
<td>impl. dep</td>
<td></td>
</tr>
<tr>
<td>● ●</td>
<td>Reserved</td>
<td>02A16-02F16</td>
<td>-NA-</td>
<td>-NA-</td>
<td></td>
</tr>
<tr>
<td>● ●</td>
<td>data_access_exception</td>
<td>03016</td>
<td>MG</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>○ ○</td>
<td>data_access_MMU_miss</td>
<td>03116</td>
<td>MG (impl. dep.)</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>○ ●</td>
<td>data_access_error</td>
<td>03216</td>
<td>AG</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>○ ○</td>
<td>data_access_protection</td>
<td>03316</td>
<td>MG (impl. dep.)</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>● ●</td>
<td>mem_address_not_aligned</td>
<td>03416</td>
<td>AG</td>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>
### TABLE 7-1 SPARC64 IXfx のトラップ一覧 (TT 番号順) (2 of 2)

<table>
<thead>
<tr>
<th>SPARC V9 M/O</th>
<th>JPS1 M/O</th>
<th>トラブル名</th>
<th>TT</th>
<th>% % 種類</th>
<th>優先順位</th>
</tr>
</thead>
<tbody>
<tr>
<td>o</td>
<td>●</td>
<td>LDDF_mem_address_not_aligned (impl. dep. #109)</td>
<td>03516</td>
<td>AG</td>
<td>10</td>
</tr>
<tr>
<td>o</td>
<td>●</td>
<td>STDF_mem_address_not_aligned (impl. dep. #110)</td>
<td>03616</td>
<td>AG</td>
<td>10</td>
</tr>
<tr>
<td>●</td>
<td>●</td>
<td>privileged_action</td>
<td>03716</td>
<td>AG</td>
<td>11</td>
</tr>
<tr>
<td>o</td>
<td>○</td>
<td>LDQF_mem_address_not_aligned (impl. dep. #111)</td>
<td>03816</td>
<td>AG</td>
<td>10</td>
</tr>
<tr>
<td>o</td>
<td>○</td>
<td>STQF_mem_address_not_aligned (impl. dep. #112)</td>
<td>03916</td>
<td>AG</td>
<td>10</td>
</tr>
<tr>
<td>●</td>
<td>●</td>
<td>Reserved</td>
<td>03A16–03F16</td>
<td>-NA-</td>
<td>-NA-</td>
</tr>
<tr>
<td>o</td>
<td>○</td>
<td>async_data_error</td>
<td>04016</td>
<td>AG</td>
<td>2</td>
</tr>
<tr>
<td>●</td>
<td>●</td>
<td>interrupt_level_n (n = 1–15)</td>
<td>04116–04F16</td>
<td>AG</td>
<td>32-n</td>
</tr>
<tr>
<td>●</td>
<td>●</td>
<td>Reserved</td>
<td>05016–05F16</td>
<td>-NA-</td>
<td>-NA-</td>
</tr>
<tr>
<td>o</td>
<td>●</td>
<td>interrupt_vector</td>
<td>06016</td>
<td>IG</td>
<td>16</td>
</tr>
<tr>
<td>o</td>
<td>●</td>
<td>PA_watchpoint</td>
<td>06116</td>
<td>AG</td>
<td>12</td>
</tr>
<tr>
<td>o</td>
<td>●</td>
<td>VA_watchpoint</td>
<td>06216</td>
<td>AG</td>
<td>11</td>
</tr>
<tr>
<td>o</td>
<td>●</td>
<td>ECC_error</td>
<td>06316</td>
<td>AG</td>
<td>33</td>
</tr>
<tr>
<td>o</td>
<td>●</td>
<td>fast_instruction_access_MMU_miss</td>
<td>06416–06716</td>
<td>MG</td>
<td>2</td>
</tr>
<tr>
<td>o</td>
<td>●</td>
<td>fast_data_access_MMU_miss</td>
<td>06816–06B16</td>
<td>MG</td>
<td>12</td>
</tr>
<tr>
<td>o</td>
<td>●</td>
<td>fast_data_access_protection</td>
<td>06C16–06F16</td>
<td>MG</td>
<td>12</td>
</tr>
<tr>
<td>o</td>
<td>○</td>
<td>implementation_dependent_exception_n (impl. dep. #35)</td>
<td>07016–072</td>
<td>impl. dep.</td>
<td>impl. dep.</td>
</tr>
<tr>
<td>o</td>
<td>○</td>
<td>illegal_action</td>
<td>07316</td>
<td>AG</td>
<td>8.5</td>
</tr>
<tr>
<td>o</td>
<td>○</td>
<td>implementation_dependent_exception_n (impl. dep. #35)</td>
<td>07416–076</td>
<td>impl. dep.</td>
<td>impl. dep.</td>
</tr>
<tr>
<td>o</td>
<td>○</td>
<td>SIMD_load_across_pages</td>
<td>07716</td>
<td>AG</td>
<td>12</td>
</tr>
<tr>
<td>o</td>
<td>○</td>
<td>implementation_dependent_exception_n (impl. dep. #35)</td>
<td>07816–07F</td>
<td>impl. dep.</td>
<td>impl. dep.</td>
</tr>
<tr>
<td>●</td>
<td>●</td>
<td>spill_n_normal (n = 0–7)</td>
<td>08016–09F16</td>
<td>AG</td>
<td>9</td>
</tr>
<tr>
<td>●</td>
<td>●</td>
<td>spill_n_other (n = 0–7)</td>
<td>0A016–0BF16</td>
<td>AG</td>
<td>9</td>
</tr>
<tr>
<td>●</td>
<td>●</td>
<td>fill_n_normal (n = 0–7)</td>
<td>0C016–0DF16</td>
<td>AG</td>
<td>9</td>
</tr>
<tr>
<td>●</td>
<td>●</td>
<td>fill_n_other (n = 0–7)</td>
<td>0E016–0FF16</td>
<td>AG</td>
<td>9</td>
</tr>
<tr>
<td>●</td>
<td>●</td>
<td>trap_instruction</td>
<td>10016–17F16</td>
<td>AG</td>
<td>16</td>
</tr>
<tr>
<td>●</td>
<td>●</td>
<td>Reserved</td>
<td>18016–1FF16</td>
<td>-NA-</td>
<td>-NA-</td>
</tr>
</tbody>
</table>
### TABLE 7-2 SPARC64 IXfx のトラップ一覧 (優先順位順、0 が高く番号が大きいほど低い) (1 of 2)

<table>
<thead>
<tr>
<th>SPARC V9 M/O</th>
<th>JPS1 M/O</th>
<th>ブラップ名</th>
<th>TT</th>
<th>% 種類</th>
<th>優先順位</th>
</tr>
</thead>
<tbody>
<tr>
<td>● ●</td>
<td></td>
<td>power_on_reset (POR)</td>
<td>00116</td>
<td>AG</td>
<td>()</td>
</tr>
<tr>
<td>● ●</td>
<td></td>
<td>externally_initiated_reset (XIR)</td>
<td>00316</td>
<td>AG</td>
<td>1</td>
</tr>
<tr>
<td>●</td>
<td></td>
<td>watchdog_reset (WDR)</td>
<td>00216</td>
<td>AG</td>
<td>1</td>
</tr>
<tr>
<td>● ●</td>
<td></td>
<td>software_initiated_reset (SIR)</td>
<td>00416</td>
<td>AG</td>
<td>1</td>
</tr>
<tr>
<td>● ●</td>
<td></td>
<td>RED_state_exception</td>
<td>00516</td>
<td>AG</td>
<td>1</td>
</tr>
<tr>
<td>● ○</td>
<td></td>
<td>async_data_error</td>
<td>04016</td>
<td>AG</td>
<td>2</td>
</tr>
<tr>
<td>●</td>
<td></td>
<td>fast_instruction_access_MMU_miss</td>
<td>06416-06716</td>
<td>MG</td>
<td>2</td>
</tr>
<tr>
<td>● ●</td>
<td></td>
<td>instruction_access_error</td>
<td>00A16</td>
<td>AG</td>
<td>3</td>
</tr>
<tr>
<td>● ●</td>
<td></td>
<td>instruction_access_exception</td>
<td>00816</td>
<td>MG</td>
<td>5</td>
</tr>
<tr>
<td>● ●</td>
<td></td>
<td>privileged_opcode</td>
<td>01116</td>
<td>AG</td>
<td>6</td>
</tr>
<tr>
<td>● ●</td>
<td></td>
<td>illegal_instruction</td>
<td>01016</td>
<td>AG</td>
<td>7</td>
</tr>
<tr>
<td>● ●</td>
<td></td>
<td>fp_disabled</td>
<td>02016</td>
<td>AG</td>
<td>8</td>
</tr>
<tr>
<td>● ○</td>
<td></td>
<td>fp_exception_other (ftt = unimplemented_FPop のみ)</td>
<td>02216</td>
<td>AG</td>
<td>8.2</td>
</tr>
<tr>
<td>● ●</td>
<td></td>
<td>clean_window</td>
<td>02416-02716</td>
<td>AG</td>
<td>9</td>
</tr>
<tr>
<td>● ●</td>
<td></td>
<td>spill_n_normal (n = 0–7)</td>
<td>08016-09F16</td>
<td>AG</td>
<td>9</td>
</tr>
<tr>
<td>● ●</td>
<td></td>
<td>spill_n_other (n = 0–7)</td>
<td>0A016-0BF16</td>
<td>AG</td>
<td>9</td>
</tr>
<tr>
<td>● ●</td>
<td></td>
<td>fill_n_normal (n = 0–7)</td>
<td>0C016-0DF16</td>
<td>AG</td>
<td>9</td>
</tr>
<tr>
<td>● ●</td>
<td></td>
<td>fill_n_other (n = 0–7)</td>
<td>0E016-0FF16</td>
<td>AG</td>
<td>9</td>
</tr>
<tr>
<td>○ ●</td>
<td></td>
<td>mem_address_not_aligned</td>
<td>02416-02716</td>
<td>AG</td>
<td>10</td>
</tr>
<tr>
<td>○ ●</td>
<td></td>
<td>LDDF_mem_address_not_aligned (impl. dep. #109)</td>
<td>03516</td>
<td>AG</td>
<td>10</td>
</tr>
<tr>
<td>○ ●</td>
<td></td>
<td>STDF_mem_address_not_aligned (impl. dep. #110)</td>
<td>03616</td>
<td>AG</td>
<td>10</td>
</tr>
<tr>
<td>● ●</td>
<td></td>
<td>mem_address_not_aligned</td>
<td>03416</td>
<td>AG</td>
<td>10</td>
</tr>
<tr>
<td>○ ●</td>
<td></td>
<td>dp_exception_ieee_754</td>
<td>02116</td>
<td>AG</td>
<td>11</td>
</tr>
<tr>
<td>○ ●</td>
<td></td>
<td>dp_exception_other (ftt = unimplemented_FPop 以外)</td>
<td>02216</td>
<td>AG</td>
<td>11</td>
</tr>
<tr>
<td>● ●</td>
<td></td>
<td>privileged_action</td>
<td>03716</td>
<td>AG</td>
<td>11</td>
</tr>
<tr>
<td>●</td>
<td></td>
<td>VA_watchpoint</td>
<td>06216</td>
<td>AG</td>
<td>11</td>
</tr>
<tr>
<td>● ●</td>
<td></td>
<td>data_access_exception</td>
<td>03016</td>
<td>MG</td>
<td>12</td>
</tr>
<tr>
<td>●</td>
<td></td>
<td>fast_data_access_MMU_miss</td>
<td>06816-06B16</td>
<td>MG</td>
<td>12</td>
</tr>
<tr>
<td>○ ●</td>
<td></td>
<td>data_access_error</td>
<td>03216</td>
<td>AG</td>
<td>12</td>
</tr>
<tr>
<td>● ●</td>
<td></td>
<td>PA_watchpoint</td>
<td>06116</td>
<td>AG</td>
<td>12</td>
</tr>
<tr>
<td>● ●</td>
<td></td>
<td>fast_data_access_protection</td>
<td>06C16-06F16</td>
<td>MG</td>
<td>12</td>
</tr>
<tr>
<td>○ ●</td>
<td></td>
<td>SIMD_load_across_pages</td>
<td>07716</td>
<td>AG</td>
<td>12</td>
</tr>
</tbody>
</table>
7.4.3 Trap Priorities

SPARC64 Ixx では一部のトラップの優先順位を JPS1 Commonality 定義から変更している。

- *fp_exception_other* の優先順位は JPS1 Commonality では 11 だが、SPARC64 Ixx では FSR.flt = 3 (unimplemented FPop) のときに限り 8.2 である。
- *VA_watchpoint* は優先順位は 11 だが、SIMD ロード、SIMD ストア命令では優先順位 12 の例外とどちらが通知されるかは状況に依存する。詳細は Appendix F.5.1, “Trap Conditions for SIMD Load/Store” (page 180) を参照。
- *illegal_action* は SPARC64 Ixx 新規トラップで優先順位は 8.5 だが、優先順位 7 の illegal_instruction より優先して通知される場合がある。詳細は Chapter 7.6.1 を参照。
- TLB 多重ヒットが検出されると、TTE の内容に依存する例外は検出されない。詳細は Appendix F.5.2, “Behavior on TLB Error” (page 180) を参照。

7.5 Trap Processing

JPS1 Commonality ではトラップ時のレジスタの変化を、いくつかの場合に分けて説明しているが、SPARC64 Ixx で追加されたレジスタについてはどの場合でも同じなので、場合分けず説明する。

トラップ時には、以下のレジスタの値が更新される。

- HPC-ACE 機能の使用状況を保存し、トラップハンドラの先頭命令からは HPC-ACE 機能を使わずに実行できるようにする。

---

**TABLE 7-2**  SPARC64 Ixx のトラップ一覧 (優先順位順、0 が高く番号が大きいほど低い) (2 of 2)

<table>
<thead>
<tr>
<th>SPARC V9 M/O</th>
<th>JPS1 M/O</th>
<th>トラブル名</th>
<th>TT</th>
<th>%g 種類</th>
<th>優先順位</th>
</tr>
</thead>
<tbody>
<tr>
<td>● ●</td>
<td></td>
<td>tag_overflow</td>
<td>023₁₆</td>
<td>AG</td>
<td>14</td>
</tr>
<tr>
<td>● ●</td>
<td></td>
<td>division_by_zero</td>
<td>028₁₆</td>
<td>AG</td>
<td>15</td>
</tr>
<tr>
<td>● ●</td>
<td></td>
<td>trap_instruction</td>
<td>100₁₆-17F₁₆</td>
<td>AG</td>
<td>16</td>
</tr>
<tr>
<td>○ ●</td>
<td></td>
<td>interrupt_vector</td>
<td>060₁₆</td>
<td>IG</td>
<td>16</td>
</tr>
<tr>
<td>● ●</td>
<td></td>
<td>interrupt_level_n (n = 1–15)</td>
<td>041₁₆-04F₁₆</td>
<td>AG</td>
<td>32-n</td>
</tr>
<tr>
<td>○ ●</td>
<td></td>
<td>ECC_error</td>
<td>063₁₆</td>
<td>AG</td>
<td>33</td>
</tr>
</tbody>
</table>
対象命令で例外が発生した場合、0 になる前の値が TXAR[TL] にセーブされる。
Tcc の場合、taken のとき TXAR[TL] には Tcc 実行前の XAR がセーブされる。

DONE, RETRY を実行したときのレジスタの変化は以下のようにになる。

\[
\begin{align*}
\text{TXAR[TL]} & \quad \leftarrow \text{XAR} \\
\text{XAR} & \quad \leftarrow 0
\end{align*}
\]

---

**Programming Note** — エミュレーションルーチンが HPC-ACE 拡張命令をエミュレーションする場合は、XAR が消費されたように見えるように TXAR[TL] を調整してから DONE すること。
7.6.5 SPARC JPS1 Implementation-Dependent Traps

SPARC64 IXfx 固有のトラップは以下の通りである (impl. dep. #35)。

- **async_data_error** [tt = 04016] (Preemptive or disrupting) (impl. dep. #218) —
  SPARC64 IXfx では緊急エラー (Urgent Error) を報告するために使われる。詳細は Appendix P.4, “緊急エラー” (page 276) を参照。

- **illegal_action** [tt = 07316] (Precise) — 実行しようとする命令が XAR 対象外の命令だが XAR.v = 1 が指定されている場合、または、実行しようとする命令が XAR 対象命令だが XAR の指定が間違っている場合に通知される。SXAR で XAR をセットする場合は、後続の命令を実行しようとした時点で通知される。WRXAR, WRTXAR, WRPR %pstate では優先順位の高い illegal_instruction ではなく illegal_action が通知されることがある。詳細は各命令の説明を参照。

- **SIMD_load_across_pages** [tt = 07716] (Precise) — SIMD ロードが複数ページにまたがり、extended 側が TLB ミスした場合に通知される。システムソフトウェアはこの例外が通知された場合、エミュレーションにより basic, extended のロードを個別に処理すること。

**Note** — **SIMD_load_across_pages** で TLB の更新処理を行うと、basic と extended が交互に TLB から落ちる無限ループに陥る可能性がある。
Memory Models

SPARC V9 アーキテクチャ仕様は、SPARC V9 システム上で動作するソフトウェアが観測可能な振る舞いを記述したモデルである。したがって、メモリアクセスの方法も、それが JPS1 Commonality の Chapter 8 および Appendix D で定義されるメモリモデルに準拠している限りはどのように実装していてもよい。

SPARC V9 では Total Store Order (TSO), Partial Store Order (PSO), Relaxed Memory Order (RMO) の3種類のメモリモデルを定義している。SPARC V9 に準拠するプロセッサは、SPARC V8 との互換性を保証するため、TSO またはそれ以上に制約の強いメモリモデル (たとえば逐次一貫性 Sequential Consistency モデルなど) を実装する必要がある。

これに対し、PSO, RMO モデルをサポートするかどうかは SPARC V9 システムでは実装依存である。SPARC64 IXfx はどのメモリモデルでも、その定義通りに動作する。
8.1 Overview

**Note** — 以下の文で“ハードウェアメモリモデル”は“SPARC V9メモリモデル”との対比で使われる。SPARC V9メモリモデルはPSTATE.MMで選択されるメモリモデルを意味する。

SPARC64 IXfxはSPARC V9で定義された3種類のメモリモデルのどの定義にも準拠するメモリモデル一種類だけを実装する (impl. dep. #113)。

■ **Total Store Order** — すべてのロードは、先行するロードとの順序を守り、すべてのストアは、先行するロードとストアとの順序を守る。この動作はSPARC V9のメモリモデルTSO, PSO, RMOの仕様に準拠する。PSTATE.MMでPSOまたはRMOが指示された場合でも、SPARC64 IXfxはこのメモリモデルで動作する。定義から、PSOまたはRMOで動作するよう書かれたプログラムはTSOでも動作するので、SPARC64 IXfxのこの動作はPSO, RMOの弱い制約のメリットを享受することはできないが、安全である。

8.4 SPARC V9 Memory Model

8.4.5 Mode Control

SPARC64 IXfxはPSTATE.MMのすべての設定においてTSOで動作する。
PSTATE.MMに112をセットした場合もTSOで動作する (impl. dep. #119) が、将来のバージョンのSPARC64 IXfxでは112を別のメモリモデルに割り当てるかもしれないので、使うべきではない。

8.4.7 Synchronizing Instruction and Data Memory

SPARC64 IXfxではハードウェアがすべてのキャッシュのデータの同一性を保証する。データキャッシュの書き込みにより、対応するデータが命令キャッシュに載っているばそれを無効化し、命令フェッチによる命令キャッシュの読み出しでは、データキャッシュに変更済みデータがあればそれが充当される。

SPARC64 IXfxのこの仕様はFLUSH命令が不要であることを意味しない。FLUSHはキャッシュとパイプライン内データの同一性を保証したい場合には実行する必要がある。
SPARC64 IXfx はマルチプロセッサをサポートしないので、マルチプロセッサでの FLUSH 命令のレイテンシは定義されない。CPU チップ内のコア間での FLUSH 命令のレイテンシは、CPU 内部の状態に依存するが、最短で 30 サイクルである (impl. dep. #122).
この章では、JPS1 Commonality で実装依存仕様が定義された命令の SPARC64 IXfx での仕様と、SPARC64 IXfx 独自に拡張された命令について記述する。JPS1 Commonality 仕様に準拠している命令については記述していないので、JPS1 Commonality を参照のこと。なお、セクション番号は JPS1 Commonality に一致させている。

SPARC64 IXfx 固有に拡張された命令は、Section A.24 のサブセクションと、Section A.72 以降に記載されている。それ以外のセクションは、JPS1 Commonality の仕様通りである。

実装依存仕様を記述している命令は、必要な情報のみを記述している。SPARC64 IXfx 固有命令の仕様には以下の記述が含まれている。

1. オペコード表。そのオペコードに固有のフィールド値と、HPC-ACE 拡張機能に対応しているかどうかの情報を含む。

2. 命令フォーマットの図。図中、ダッシュ（—）で表示されているフィールドは将来の拡張用に予約されており（Reserved）、SPARC64 IXfx 用のプログラムでは 0 が入っているべきである。SPARC V9 のプロセッサでは、Reserved が 0 でない命令を実行したときの挙動は未定義である。SPARC64 IXfx の動作は Section 1.2, “Fonts and Notational Conventions” (page 1) を参照。

3. アセンブリ言語での表記法。詳細は Appendix G を参照。

4. 命令の詳細、制限事項、例外発生条件などの説明。

5. その命令を実行したときに起こりうる例外の一覧を、優先順位に従って記載している。
   ただし以下の例外は記載していない。

   a. すべての命令で起こり得る instruction_access_error、
      instruction_access_exception, fast_instruction_access_MMU_miss, 
      async_data_error, ECC_error, およびインタラプト。

   b. 実装されていない命令で起こる illegal_instruction (浮動小数点演算では 
      fp_exception_other で ftt = unimplemented_FPop)。
c. **IIU_INST_TRAP (ASI = 60, VA = 0)** を指定すればどの命令でも発生する illegal_instruction。

なお、illegal_action 例外の発生条件を記述する際は、XAR のフィールド名には f_、s_ をつけず表記する。

以下の例外は SPARC64 IXfx では起こらない。

- instruction_access_MMU_miss
- data_access_MMU_miss
- data_access_protection
- unimplemented_LDD
- unimplemented_STD
- LDQF_mem_address_not_aligned
- STQF_mem_address_not_aligned
- internal_processor_error
- fp_exception_other (ftt = invalid_fp_register)

命令仕様には、タイミングに関する情報は記載していない。

**JPS1 Commonality** の命令と SPARC64 IXfx で拡張された命令の一覧を TABLE A-2 に示す。この表、およびこの章と Appendix E ではオペコードの右肩に文字列が付されている命令がある。その文字列の意味は下表の通りである。

<table>
<thead>
<tr>
<th>文字</th>
<th>意味</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>非推奨命令</td>
</tr>
<tr>
<td>P</td>
<td>特権命令</td>
</tr>
<tr>
<td>P_ASI</td>
<td>ASI の 7 ビット目が 0 だと特権動作</td>
</tr>
<tr>
<td>P_ASR</td>
<td>ASR 番号によっては特権動作</td>
</tr>
<tr>
<td>P_NPT</td>
<td>PSTATE.PRIV = 0 かつ (S)TICK.NPT = 1 のとき特権動作</td>
</tr>
<tr>
<td>P_NOTIFY</td>
<td>PCR.PRIV = 1 のとき特権アクセス</td>
</tr>
<tr>
<td>P_COMPARE</td>
<td>PCR.PRIV = 1 のとき特権アクセス</td>
</tr>
</tbody>
</table>

TABLE A-2 および各命令のオペコード表にある HPC-ACE 拡張の列は、その命令で SPARC64 IXfx のどの拡張機能が有効かを示している。

- **Inst. JPS1 Commonality** では定義されていない、SPARC64 IXfx 固有の命令。
- **Regs. XAR 対象命令**。拡張された整数および浮動小数点レジスタが使用できるほか、メモリアクセス命令ではセクタキャッシュ指定もできる。この列に ♠ がついている命令は、rd に指定できるのは basic FPR のみ。
- **SIMD SIMD 演算可能**。
  この列に ♠ がついている命令は、4 倍精度命令では SIMD 演算ができない。

この 3 つの欄のどれにも ✓ がつっていない命令は XAR 非対象命令である。XAR 非対象命令の詳細は "XAR 動作 " (page 31) を参照。
<table>
<thead>
<tr>
<th>命令</th>
<th>処理内容</th>
<th>HPC-ACE拡張</th>
<th>Inst. Regs. SIMD</th>
<th>ページ</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD (ADDcc)</td>
<td>Add (and modify condition codes)</td>
<td>✓</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>ADDC (ADDCCc)</td>
<td>Add with carry (and modify condition codes)</td>
<td>✓</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>ALIGNADDRESS(_LITTLE)</td>
<td>Calculate address for misaligned data</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>AND (ANDcc)</td>
<td>And (and modify condition codes)</td>
<td>✓</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>ANDN (ANDNcc)</td>
<td>And not (and modify condition codes)</td>
<td>✓</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>ARRAY(8,16,32)</td>
<td>3-D array addressing instructions</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>BPcc</td>
<td>Branch on integer condition codes with prediction</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>BiccD</td>
<td>Branch on integer condition codes</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>BMASK</td>
<td>Set the GSR.MASK</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>BPr</td>
<td>Branch on contents of integer register with prediction</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>BSHUFFLE</td>
<td>Permute bytes as specified by GSR.MASK</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>CALL</td>
<td>Call and link</td>
<td>68</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CASAPASM</td>
<td>Compare and swap word in alternate space</td>
<td>✓</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>CASXAPASM</td>
<td>Compare and swap doubleword in alternate space</td>
<td>✓</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>DONEP</td>
<td>Return from trap</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>EDGE(8,16,32){L}</td>
<td>Edge handling instructions</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>FABS(s,d,q)</td>
<td>Floating-point absolute value</td>
<td>✓ ✤</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>FADD(s,d,q)</td>
<td>Floating-point add</td>
<td>✓ ✤</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>FALIGNDATA</td>
<td>Perform data alignment for misaligned data</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>FAND[S]</td>
<td>Logical AND operation</td>
<td>✓ ✓</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>FANDNOT(1,2){S}</td>
<td>Logical AND operation with one inverted source</td>
<td>✓ ✓</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>FBfccD</td>
<td>Branch on floating-point condition codes</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>FBfpcc</td>
<td>Branch on floating-point condition codes</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>FCMPF(s,d,q)</td>
<td>Floating-point compare</td>
<td>✓</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>FMADD(s,d,q)</td>
<td>Floating-point Multiply-and-Add</td>
<td>✓ ✓ ✓</td>
<td>70</td>
<td></td>
</tr>
<tr>
<td>FCMPF(GT,LE,NE,GE)</td>
<td>Floating-point conditional compare (exception if unordered)</td>
<td>✓ ✓ ✓</td>
<td>114</td>
<td></td>
</tr>
<tr>
<td>FCMPF(GT,LT,NE,GE,LE)</td>
<td>Floating-point conditional compare to register</td>
<td>✓ ✓ ✓</td>
<td>114</td>
<td></td>
</tr>
<tr>
<td>FDIV(s,d,q)</td>
<td>Floating-point divide</td>
<td>✤</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>FdMULq</td>
<td>Floating-point multiply double to quad</td>
<td>✓</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>FEXPAND</td>
<td>Pixel expansion</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>FiTO(s,d,q)</td>
<td>Convert integer to floating-point</td>
<td>✓ ✤</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>FLUSH</td>
<td>Flush instruction memory</td>
<td>✓</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>FLUSHW</td>
<td>Flush register windows</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>FMADD(s,d)</td>
<td>Floating-point Multiply-and-Add</td>
<td>✓ ✓ ✓</td>
<td>70</td>
<td></td>
</tr>
<tr>
<td>命令</td>
<td>处理内容</td>
<td>HPC-ACE 拡張</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>--------------------------------------</td>
<td>---------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FMAX(s,d)</td>
<td>Floating-point maximum</td>
<td>✓ ✓ ✓ 116</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FMIN(s,d)</td>
<td>Floating-point minimum</td>
<td>✓ ✓ ✓ 116</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FMSUB(s,d)</td>
<td>Floating-point Multiply-and-Subtract</td>
<td>✓ ✓ ✓ 70</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FMOV(s,d,q)</td>
<td>Floating-point move</td>
<td>✓ ✓ ✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FMOV(s,d,q)cc</td>
<td>Move f-p reg. if condition is satisfied</td>
<td>—</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FMUL(s,d,q)</td>
<td>Floating-point multiply</td>
<td>✓ ✓ ✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FMOV(s,d,q)r</td>
<td>Move f-p reg. if integer reg. contents satisfy condition</td>
<td>—</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FMUL8x16</td>
<td>8x16 partitioned product</td>
<td>—</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FMUL8x16(AU,AL)</td>
<td>8x16 upper/lower α partitioned product</td>
<td>—</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FMULB8(SU,UL)x16</td>
<td>8x16 upper/lower partitioned product</td>
<td>—</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FNAND(s)</td>
<td>Logical NAND operation</td>
<td>✓ ✓ ✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FNEG(s,d,q)</td>
<td>Floating-point negate</td>
<td>✓ ✓ ✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FNMSADD(s,d)</td>
<td>Floating-point Multiply-and-Add and negate</td>
<td>✓ ✓ ✓ 70</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FNMSUB(s,d)</td>
<td>Floating-point Multiply-and-Subtract and negate</td>
<td>✓ ✓ ✓ 70</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FNOR(s)</td>
<td>Logical NOR operation</td>
<td>✓ ✓ ✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FNEG(s,d,q)</td>
<td>Floating-point negate</td>
<td>✓ ✓ ✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FPADDX(HI)</td>
<td>Integer Multiply-and-Add</td>
<td>✓ ✓ ✓ 78</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FPACK(16,32, FIX)</td>
<td>Pixel packing</td>
<td>—</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FPADD(16,32){S}</td>
<td>Pixel add (single) 16- or 32-bit</td>
<td>—</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FPADDX[HI]</td>
<td>Integer Multiply-and-Add</td>
<td>✓ ✓ ✓ 78</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FPCRADD(s,d)</td>
<td>Floating-point reciprocal approximation</td>
<td>✓ ✓ ✓ 118</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FFSQRTA(s,d)</td>
<td>Floating-point reciprocal square root approximation</td>
<td>✓ ✓ ✓ 118</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FONE(s)</td>
<td>One fill</td>
<td>✓ ✓ ✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FOR(s)</td>
<td>Logical OR operation</td>
<td>✓ ✓ ✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FNOT(1,2){S}</td>
<td>Logical OR operation with one inverted source</td>
<td>✓ ✓ ✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FFSUB(16,32){S}</td>
<td>Pixel subtract (single) 16- or 32-bit</td>
<td>—</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FSADD(s,d)</td>
<td>Float-point multiply single to double</td>
<td>✓ ✓ ✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FSQRT(s,d,q)</td>
<td>Floating-point square root</td>
<td>☆ —</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FSC(1,2){S}</td>
<td>Copy source</td>
<td>✓ ✓ ✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FSELMOV(s,d)</td>
<td>Move selected floating-point register</td>
<td>✓ ✓ ✓ 122</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F(s,d,q)TOi</td>
<td>Convert floating point to integer</td>
<td>✓ ✓ ✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F(s,d,q)TOe</td>
<td>Convert between floating-point formats</td>
<td>✓ ✓ ✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F(s,d,q)TOx</td>
<td>Convert floating point to 64-bit integer</td>
<td>✓ ✓ ✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FSUBU(s,d,q)</td>
<td>Floating-point subtract</td>
<td>✓ ✓ ✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FTRIMADd</td>
<td>Floating-point trigonometric function</td>
<td>✓ ✓ ✓ 123</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### TABLE A-2  SPARC64 Ix fx の命令セット  (3 of 7)

<table>
<thead>
<tr>
<th>命令</th>
<th>処理内容</th>
<th>HPC-ACE 拡張</th>
<th>Inst. Regs. SIMD</th>
<th>ページ</th>
</tr>
</thead>
<tbody>
<tr>
<td>FTRIS(MUL,SEL)d</td>
<td>Floating-point trigonometric functions</td>
<td>✓</td>
<td>✓</td>
<td>123</td>
</tr>
<tr>
<td>FXNOR{s}</td>
<td>Logical XNOR operation</td>
<td>✓</td>
<td>✓</td>
<td>—</td>
</tr>
<tr>
<td>FXOR{s}</td>
<td>Logical XOR operation</td>
<td>✓</td>
<td>✓</td>
<td>—</td>
</tr>
<tr>
<td>FxTO(s,d,q)</td>
<td>Convert 64-bit integer to floating-point</td>
<td>✓</td>
<td>✤</td>
<td>—</td>
</tr>
<tr>
<td>FZERO{s}</td>
<td>Zero fill</td>
<td>✓</td>
<td>✓</td>
<td>—</td>
</tr>
<tr>
<td>ILLTRAP</td>
<td>Illegal instruction</td>
<td>—</td>
<td></td>
<td>—</td>
</tr>
<tr>
<td>JMPL</td>
<td>Jump and link</td>
<td>80</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDDD</td>
<td>Load integer doubleword</td>
<td>✓</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>LDDA,D, PASI</td>
<td>Load integer doubleword from alternate space</td>
<td>✓</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>LDDA ASI_NUCLEUS_QUAD*</td>
<td>Load integer quadword, atomic</td>
<td>✓</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>LDDA ASI_QUAD_PHYS*</td>
<td>Load integer quadword, atomic (physical address)</td>
<td>✓</td>
<td>88</td>
<td></td>
</tr>
<tr>
<td>LDDF</td>
<td>Load double floating-point</td>
<td>✓</td>
<td>81</td>
<td></td>
</tr>
<tr>
<td>LDDFX_PASI</td>
<td>Load double floating-point from alternate space</td>
<td>✓</td>
<td>85</td>
<td></td>
</tr>
<tr>
<td>LDDFA ASI_BLK*</td>
<td>Block loads</td>
<td>✓</td>
<td>66</td>
<td></td>
</tr>
<tr>
<td>LDDFA ASI_FL*</td>
<td>Short floating point loads</td>
<td>—</td>
<td></td>
<td>—</td>
</tr>
<tr>
<td>LDF</td>
<td>Load floating-point</td>
<td>✓</td>
<td>✓</td>
<td>81</td>
</tr>
<tr>
<td>LDFPAS*</td>
<td>Load floating-point from alternate space</td>
<td>✓</td>
<td>✓</td>
<td>85</td>
</tr>
<tr>
<td>LDFFR</td>
<td>Load floating-point state register lower</td>
<td>✓</td>
<td>81</td>
<td></td>
</tr>
<tr>
<td>LDQF</td>
<td>Load quad floating-point</td>
<td>✓</td>
<td>81</td>
<td></td>
</tr>
<tr>
<td>LDQFPAS*</td>
<td>Load quad floating-point from alternate space</td>
<td>✓</td>
<td>85</td>
<td></td>
</tr>
<tr>
<td>LDSB</td>
<td>Load signed byte</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>LDSBA_PASI</td>
<td>Load signed byte from alternate space</td>
<td>✓</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>LDSH</td>
<td>Load signed halfword</td>
<td>—</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDSHA_PASI</td>
<td>Load signed halfword from alternate space</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>LDSTUB</td>
<td>Load-store unsigned byte</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>LDSTUBA_PASI</td>
<td>Load-store unsigned byte in alternate space</td>
<td>✓</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>LDSW</td>
<td>Load signed word</td>
<td>✓</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>LDSWA_PASI</td>
<td>Load signed word from alternate space</td>
<td>✓</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>LDUB</td>
<td>Load unsigned byte</td>
<td>—</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDUBA_PASI</td>
<td>Load unsigned byte from alternate space</td>
<td>✓</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>LDUN</td>
<td>Load unsigned halfword</td>
<td>✓</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>LDUNA_PASI</td>
<td>Load unsigned halfword from alternate space</td>
<td>✓</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>LDUN</td>
<td>Load unsigned word</td>
<td>—</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDUNA_PASI</td>
<td>Load unsigned word from alternate space</td>
<td>✓</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>LDX</td>
<td>Load extended</td>
<td>✓</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>LDXPA_PASI</td>
<td>Load extended from alternate space</td>
<td>✓</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>命令</td>
<td>処理内容</td>
<td>HPC-ACE 拡張</td>
<td>Inst. Regs. SIMD</td>
<td>ページ</td>
</tr>
<tr>
<td>---------</td>
<td>-----------------------------------</td>
<td>-------------</td>
<td>-----------------</td>
<td>-------</td>
</tr>
<tr>
<td>LDXFSR</td>
<td>Load floating-point state register</td>
<td>✓</td>
<td>81</td>
<td></td>
</tr>
<tr>
<td>MEMBAR</td>
<td>Memory barrier</td>
<td></td>
<td>90</td>
<td></td>
</tr>
<tr>
<td>MOVcc</td>
<td>Move integer register if condition is satisfied</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MOVr</td>
<td>Move integer register on contents of integer register</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MULSccD</td>
<td>Multiply step (and modify condition codes)</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MULX</td>
<td>Multiply 64-bit integers</td>
<td>✓</td>
<td>92</td>
<td></td>
</tr>
<tr>
<td>NOP</td>
<td>No operation</td>
<td>✓</td>
<td>95</td>
<td></td>
</tr>
<tr>
<td>OR (ORcc)</td>
<td>Inclusive-or (and modify condition codes)</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ORN (ORNcc)</td>
<td>Inclusive-or not (and modify condition codes)</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PDIST</td>
<td>Pixel component distance</td>
<td></td>
<td>94</td>
<td></td>
</tr>
<tr>
<td>POPC</td>
<td>Population count</td>
<td>✓</td>
<td>95</td>
<td></td>
</tr>
<tr>
<td>PREFETCH</td>
<td>Prefetch data</td>
<td>✓</td>
<td>97</td>
<td></td>
</tr>
<tr>
<td>PREFETCHAos</td>
<td>Prefetch data from alternate space</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RDASI</td>
<td>Read ASI register</td>
<td>✓</td>
<td>95</td>
<td></td>
</tr>
<tr>
<td>RDASRPasr</td>
<td>Read ancillary state register</td>
<td>✓</td>
<td>97</td>
<td></td>
</tr>
<tr>
<td>RDCCRP</td>
<td>Read condition codes register</td>
<td>✓</td>
<td>97</td>
<td></td>
</tr>
<tr>
<td>RDCCRPc</td>
<td>Read dispatch control register</td>
<td>✓</td>
<td>97</td>
<td></td>
</tr>
<tr>
<td>RDPPRS</td>
<td>Read floating-point registers state register</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RDGSR</td>
<td>Read graphic status register</td>
<td>✓</td>
<td>97</td>
<td></td>
</tr>
<tr>
<td>RDPC</td>
<td>Read program counter</td>
<td>✓</td>
<td>97</td>
<td></td>
</tr>
<tr>
<td>RDPCPRpc</td>
<td>Read performance control register</td>
<td>✓</td>
<td>97</td>
<td></td>
</tr>
<tr>
<td>RDPCPRpcx</td>
<td>Read performance instrumentation counters</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RDPRP</td>
<td>Read privileged register</td>
<td>✓</td>
<td>97</td>
<td></td>
</tr>
<tr>
<td>RDSOFTINTp</td>
<td>Read per-processor soft interrupt register</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RDSYCKpnt</td>
<td>Read system TICK register</td>
<td>✓</td>
<td>97</td>
<td></td>
</tr>
<tr>
<td>RDSYCK_CMPRP</td>
<td>Read system TICK compare register</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RDTICKpnt</td>
<td>Read TICK register</td>
<td>✓</td>
<td>97</td>
<td></td>
</tr>
<tr>
<td>RDTICK_CMPRP</td>
<td>Read TICK compare register</td>
<td>✓</td>
<td>97</td>
<td></td>
</tr>
<tr>
<td>RDTXARP</td>
<td>Read TXAR register</td>
<td>✓</td>
<td>97</td>
<td></td>
</tr>
<tr>
<td>RXASRP</td>
<td>Read XASR register</td>
<td>✓</td>
<td>97</td>
<td></td>
</tr>
<tr>
<td>RDYD</td>
<td>Read Y register</td>
<td>✓</td>
<td>97</td>
<td></td>
</tr>
<tr>
<td>RESTORE</td>
<td>Restore caller’s window</td>
<td>✓</td>
<td>97</td>
<td></td>
</tr>
<tr>
<td>RESTOREDp</td>
<td>Window has been restored</td>
<td></td>
<td>97</td>
<td></td>
</tr>
<tr>
<td>RETRYp</td>
<td>Return from trap and retry</td>
<td></td>
<td>97</td>
<td></td>
</tr>
<tr>
<td>RETURN</td>
<td>Return</td>
<td></td>
<td>97</td>
<td></td>
</tr>
<tr>
<td>SAVE</td>
<td>Save caller’s window</td>
<td>✓</td>
<td>97</td>
<td></td>
</tr>
</tbody>
</table>
### TABLE A-2 SPARC64 IXfx の命令セット (5 of 7)

<table>
<thead>
<tr>
<th>命令</th>
<th>处理内容</th>
<th>HPC-ACE 拡張</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAVED</td>
<td>Window has been saved</td>
<td>—</td>
</tr>
<tr>
<td>SDIVD (SDIVccD)</td>
<td>32-bit signed integer divide (and modify condition codes)</td>
<td>✓</td>
</tr>
<tr>
<td>SDIVX</td>
<td>64-bit signed integer divide</td>
<td>✓</td>
</tr>
<tr>
<td>SETHI</td>
<td>Set high 22 bits of low word of integer register</td>
<td>✓</td>
</tr>
<tr>
<td>SHUTDOWN</td>
<td>Shut down the processor</td>
<td>99</td>
</tr>
<tr>
<td>SIAM</td>
<td>Set Interval Arithmetic Mode</td>
<td>—</td>
</tr>
<tr>
<td>SIR</td>
<td>Software-initiated reset</td>
<td>—</td>
</tr>
<tr>
<td>SLEEP</td>
<td>Sleep this thread</td>
<td>77</td>
</tr>
<tr>
<td>SLL</td>
<td>Shift left logical</td>
<td>✓</td>
</tr>
<tr>
<td>SLLX</td>
<td>Shift left logical, extended</td>
<td>✓</td>
</tr>
<tr>
<td>SMULD (SMULccD)</td>
<td>Signed integer multiply (and modify condition codes)</td>
<td>✓</td>
</tr>
<tr>
<td>SRA</td>
<td>Shift right arithmetic</td>
<td>✓</td>
</tr>
<tr>
<td>SRAX</td>
<td>Shift right arithmetic, extended</td>
<td>✓</td>
</tr>
<tr>
<td>SRL</td>
<td>Shift right logical</td>
<td>✓</td>
</tr>
<tr>
<td>SRLX</td>
<td>Shift right logical, extended</td>
<td>✓</td>
</tr>
<tr>
<td>STB</td>
<td>Store byte</td>
<td>✓</td>
</tr>
<tr>
<td>STBA PASI</td>
<td>Store byte into alternate space</td>
<td>✓</td>
</tr>
<tr>
<td>STBARD</td>
<td>Store barrier</td>
<td>113</td>
</tr>
<tr>
<td>STD</td>
<td>Store doubleword</td>
<td>✓</td>
</tr>
<tr>
<td>STDAD, PASI</td>
<td>Store doubleword into alternate space</td>
<td>✓</td>
</tr>
<tr>
<td>STD (D,DF,X)A ASI_XFILL*</td>
<td>Cache line fill</td>
<td>✓ ✓</td>
</tr>
<tr>
<td>STDF</td>
<td>Store double floating-point</td>
<td>✓ ✓</td>
</tr>
<tr>
<td>STDFAPASI</td>
<td>Store double floating-point into alternate space</td>
<td>✓ ✓</td>
</tr>
<tr>
<td>STDFA ASI_BLK*</td>
<td>Block stores</td>
<td>✓</td>
</tr>
<tr>
<td>STDFA ASI_FL*</td>
<td>Short floating point stores</td>
<td>—</td>
</tr>
<tr>
<td>STDFA ASI_PST*</td>
<td>Partial Store instructions</td>
<td>—</td>
</tr>
<tr>
<td>STDPR</td>
<td>Store double floating-point on register’s condition</td>
<td>✓ ✓ ✓</td>
</tr>
<tr>
<td>STF</td>
<td>Store floating-point</td>
<td>✓ ✓</td>
</tr>
<tr>
<td>STFAPASI</td>
<td>Store floating-point into alternate space</td>
<td>✓ ✓</td>
</tr>
<tr>
<td>STFR</td>
<td>Store floating-point on register condition</td>
<td>✓ ✓ ✓</td>
</tr>
<tr>
<td>STFSRD</td>
<td>Store floating-point state register</td>
<td>✓</td>
</tr>
<tr>
<td>STH</td>
<td>Store halfword</td>
<td>✓</td>
</tr>
<tr>
<td>STHPASI</td>
<td>Store halfword into alternate space</td>
<td>✓</td>
</tr>
<tr>
<td>STQF</td>
<td>Store quad floating-point</td>
<td>✓</td>
</tr>
<tr>
<td>STQFAPASI</td>
<td>Store quad floating-point into alternate space</td>
<td>✓</td>
</tr>
<tr>
<td>STW</td>
<td>Store word</td>
<td>✓</td>
</tr>
<tr>
<td>命令</td>
<td>処理内容</td>
<td>HPC-ACE拡張</td>
</tr>
<tr>
<td>------</td>
<td>-----------------------------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>STWA</td>
<td>Store word into alternate space</td>
<td>✓</td>
</tr>
<tr>
<td>STX</td>
<td>Store extended</td>
<td>✓</td>
</tr>
<tr>
<td>STXAP</td>
<td>Store extended into alternate space</td>
<td>✓</td>
</tr>
<tr>
<td>STXFSR</td>
<td>Store extended floating-point state register</td>
<td>✓ 100</td>
</tr>
<tr>
<td>SUB</td>
<td>Subtract (and modify condition codes)</td>
<td>✓</td>
</tr>
<tr>
<td>SUBCC</td>
<td>Subtract with carry (and modify condition codes)</td>
<td>✓ 76</td>
</tr>
<tr>
<td>SUSPEND</td>
<td>Suspend this thread</td>
<td>✓</td>
</tr>
<tr>
<td>SWAP</td>
<td>Swap integer register with memory</td>
<td>✓</td>
</tr>
<tr>
<td>SWAPA</td>
<td>Swap integer register with memory in alternate space</td>
<td>✓ 131</td>
</tr>
<tr>
<td>SXAR(1,2)</td>
<td>Set XAR</td>
<td>✓</td>
</tr>
<tr>
<td>TADDcc (TADDccTV&lt;sup&gt;D&lt;/sup&gt;)</td>
<td>Tagged add and modify condition codes (trap on overflow)</td>
<td>✓ 107</td>
</tr>
<tr>
<td>Tcc</td>
<td>Trap on integer condition codes</td>
<td>✓</td>
</tr>
<tr>
<td>TSUBcc (TSUBccTV&lt;sup&gt;D&lt;/sup&gt;)</td>
<td>Tagged subtract and modify condition codes (trap on overflow)</td>
<td>✓ 107</td>
</tr>
<tr>
<td>UDIV&lt;sup&gt;D&lt;/sup&gt; (UDIVcc&lt;sup&gt;D&lt;/sup&gt;)</td>
<td>Unsigned integer divide (and modify condition codes)</td>
<td>✓ 107</td>
</tr>
<tr>
<td>UDIVX</td>
<td>64-bit unsigned integer divide</td>
<td>✓</td>
</tr>
<tr>
<td>UMUL&lt;sup&gt;D&lt;/sup&gt; (UMULcc&lt;sup&gt;D&lt;/sup&gt;)</td>
<td>Unsigned integer multiply (and modify condition codes)</td>
<td>✓ 111</td>
</tr>
<tr>
<td>WRASI</td>
<td>Write ASI register</td>
<td>✓</td>
</tr>
<tr>
<td>WRASRP</td>
<td>Write ancillary state register</td>
<td>✓</td>
</tr>
<tr>
<td>WCRCR</td>
<td>Write condition codes register</td>
<td>✓</td>
</tr>
<tr>
<td>WRDCR</td>
<td>Write dispatch control register</td>
<td>✓</td>
</tr>
<tr>
<td>WRFPRS</td>
<td>Write floating-point registers state register</td>
<td>✓</td>
</tr>
<tr>
<td>WRGSR</td>
<td>Write graphic status register</td>
<td>✓</td>
</tr>
<tr>
<td>WRPCR</td>
<td>Write performance control register</td>
<td>✓</td>
</tr>
<tr>
<td>WRPIC</td>
<td>Write performance instrumentation counters register</td>
<td>✓ 111</td>
</tr>
<tr>
<td>WRP</td>
<td>Write privileged register</td>
<td>✓</td>
</tr>
<tr>
<td>WRSOFTINT</td>
<td>Write per-processor soft interrupt register</td>
<td>✓</td>
</tr>
<tr>
<td>WRSOFTINT_CLR</td>
<td>Clear bits of per-processor soft interrupt register</td>
<td>✓ 111</td>
</tr>
<tr>
<td>WRSOFTINT_SET</td>
<td>Set bits of per-processor soft interrupt register</td>
<td>✓ 111</td>
</tr>
<tr>
<td>WRTICK_CMPR</td>
<td>Write TICK compare register</td>
<td>✓</td>
</tr>
<tr>
<td>WRTICK</td>
<td>Write System TICK register</td>
<td>✓</td>
</tr>
<tr>
<td>WRSTICK</td>
<td>Write System TICK compare register</td>
<td>✓</td>
</tr>
<tr>
<td>命令</td>
<td>処理内容</td>
<td>HPC-ACE 拡張</td>
</tr>
<tr>
<td>-------</td>
<td>-----------------------------------------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>WRTXAR'</td>
<td>Write TXAR register</td>
<td>✓</td>
</tr>
<tr>
<td>WRXAR</td>
<td>Write XAR register</td>
<td>✓</td>
</tr>
<tr>
<td>WRXASR</td>
<td>Write XASR register</td>
<td>✓</td>
</tr>
<tr>
<td>WRY^D</td>
<td>Write Y register</td>
<td>✓</td>
</tr>
<tr>
<td>XNOR (XNORcc)</td>
<td>Exclusive-nor (and modify condition codes)</td>
<td>✓</td>
</tr>
<tr>
<td>XOR (XORcc)</td>
<td>Exclusive-or (and modify condition codes)</td>
<td>✓</td>
</tr>
</tbody>
</table>
A.4 Block Load and Store Instructions (VIS I)

Deprecated – block load/store は SPARC64 IXfx では過去の互換性のために定義されており、新規プログラムでの使用は推奨されない。高速なメモリコピーには Section A.79, “Cache Line Fill with Undetermined Values” を用いること。

SPARC64 IXfx の block load/store 仕様は、SPARC64 V…SPARC64 VII とは異なる。新仕様は従来の仕様よりも制約が強いか、一部非互換なところがある。以下に SPARC64 IXfx の block load/store の動作と、従来仕様との相違点を示す。

1. block load/store はアトミックなメモリ操作ではなく、内部的に 8 バイトの load/store に分割して実行される。block load/store の前後にある MEMBAR またはアトミック命令との間でオーダリングを遵守する。

2. block load/store 命令は TSO を遵守する。前後の load/store/atomic 命令、および block load/store 内の 8 バイトに分割された各 load/store 間でも TSO に準拠して動作する。

Compatibility Note – 従来仕様では、命令間では SPARC V9 メモリモデルに準拠せず、命令内の 8 バイト単位では RMO で動作することになっていた。

3. レジスタ間のアクセス順序は、他の命令と同様に保証される。つまり、block load/store 他の命令の間でのレジスタの read-after-write, write-after-write はプログラム順序になる。

4. block load/store のキャッシュ上の動作は、通常の load/store と同じである。block load は L1 キャッシュ上にデータがあれば L1 キャッシュから読み、L1 キャッシュ上になければメモリから L1 キャッシュに読み込まれる。block store は L1 キャッシュ上にデータがあれば L1 キャッシュに書き、L1 キャッシュ上になければ L1 キャッシュに読み込んだ上で当該データを更新する。

Compatibility Note – block load/store のキャッシュに対する副作用は従来仕様とは大きく異なる。従来仕様では、block load はキャッシュにデータがあれば読み、キャッシュにない場合の動作は未定義。block store は、ダーティデータを持つキャッシュがあれば、そのキャッシュに書き込むとともにそれより上位（パイプラインに近い）キャッシュからは消し、ダーティーデータを持つキャッシュがないかまたはキャッシュに乗っていないときは、メモリに書く仕様だった。

5. SPARC64 IXfx では block store と block store with commit は完全に同じ動作をする。
Compatibility Note – block store with commit のキャッシュに対する副作用は従来仕様とは大きく異なる。従来仕様では、キャッシュにデータがあれば全部消して、データをメモリに書き込む仕様だった。

6. TTE.E = 0 であるページに対する block load/store は、64 バイト中の任意の 8 バイトの処理中に fast_data_access_MMU_miss 例外が通知されることがある。block load の途中で例外が通知されたときは、レジスタには block load 実行前の、実行後どちらの値も見えることもある。block store の途中で例外が通知されたときは、メモリの状態は block store 実行前のままである。

Programming Note – ノンキャッシュプル空間の一部に、block store は正常に終了したように見えるが実際に書き込みが行われない領域がある。詳細はシステム仕様書を参照。

Note – JPS1 Commonality で定義されている通り、block load/store 命令では、LDDF_mem_address_not_aligned, STDF_mem_address_not_aligned 例外は通知されない (Appendix L.3.3, “ASI と命令の組み合わせと例外” (page 221) も参照)。LDDFA 命令で ASI_BLK_COMMIT_{PS} を指定した場合は、block load/store 命令ではないので、4 バイト境界にアクセスすると LDDF_mem_address_not_aligned 例外が通知される (“Block Load and Store ASIs” (page 221) も参照)。

Exceptions
  illegal_instruction (misaligned rd)
  fp_disabled
  illegal_action (XAR.v = 1 and (XAR.urs1 > 1 or (i = 0 and XAR.urs2 > 1) or (i = 1 and XAR.urs2 ≠ 0) or XAR.urs3<2> ≠ 0);)
  mem_address_not_aligned (see “Block Load and Store ASIs” (page 221))
  LDDF_mem_address_not_aligned (see “Block Load and Store ASIs” (page 221))
  VA_watchpoint (only detected on the first 8 bytes of a transfer)
  fast_data_access_MMU_miss
  data_access_exception (see “Block Load and Store ASIs” (page 221))
  fast_data_access_protection
  PA_watchpoint (only detected on the first 8 bytes of a transfer)
  data_access_error
A.9 Call and Link

PSTATE.AM がセットされているとき、r[15] には PC の上位 32 ビットを 0 にした値が入る (impl. dep. #125)。r[15] の更新は直ちに行われ、ディレイスロットの命令には新しい値が見える。

Exceptions  illegal_action (XAR.v = 1)
A.24 Implementation-Dependent Instructions

IMPDEP1 と IMPDEP2 は実装依存命令である。実装依存の意味は、命令の動作、bit<29:25>, bit<18:0> の解釈、通知される例外の種類などである。

SPARC64 IXfx は IMPDEP1 に VIS, SUSPEND, SLEEP, FCMPcond{d,s}, FMIN{d,s}, FMAX{d,s}, FRCPA{d,s}, FRSQRTA{d,s}, FTRISELd, FTRISMULd 命令を実装している（impl. dep. #106）。IMPDEP2A には FPMADDX, FPMADDXHI, FTRIMADDd, FSELMOV{d,s} を、IMPDEP2B には浮動小数点積和演算命令を実装している（impl. dep. #106）。

SPARC V9 命令の実装依存命令の拡張方法については、JPS1 Commonality の I.1.2, “Implementation-Dependent and Reserved Opcodes” を参照。

Compatibility Note – SPARC V8 ではこの命令は CPopl 命令だった。

なお、SPARC64 IXfx で新規追加された IMPDEP1, IMPDEP2 命令については、Section A.24 のサブセクションではなく、他の新規追加命令と同様 Section A.71 より後に配置である。

<table>
<thead>
<tr>
<th>オペコード</th>
<th>sp3</th>
<th>動作</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMPDEP1</td>
<td>11 0110</td>
<td>実装依存命令 1</td>
</tr>
<tr>
<td>IMPDEP2</td>
<td>11 0111</td>
<td>実装依存命令 2</td>
</tr>
</tbody>
</table>

Exceptions 実装に依存する
A.24.1 Floating-Point Multiply-Add/Subtract

SPARC64 IXfxはIMPDEP2Bに浮動小数点積和演算(Floating-Point Multiply and Add: FMA)を実装している。FMAはHPC-ACEでSIMD拡張されているが、他のSIMD拡張よりも自由度が高いものとなっている。この節ではまずnon-SIMDの動作を説明し、次にHPC-ACEの拡張を説明する。

<table>
<thead>
<tr>
<th>HPC-ACE Ext.</th>
<th>オペコード</th>
<th>Var</th>
<th>Size</th>
<th>変数</th>
</tr>
</thead>
<tbody>
<tr>
<td>FMADDs</td>
<td>00 01</td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>FMADDd</td>
<td>00 10</td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>FMSUBs</td>
<td>01 01</td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>FMSUBd</td>
<td>01 10</td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>FNMSUBs</td>
<td>10 01</td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>FNMSUBd</td>
<td>10 10</td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>FNMADDs</td>
<td>11 01</td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>FNMADDd</td>
<td>11 10</td>
<td></td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

1.size = 00についてはSection A.24.4, Section A.75, Section A.76を参照。
2.size = 11は4倍精度に予約されているが、一部はSection A.75, "Move Selected Floating-Point Register on Floating-Point Register's Condition"で使われている。

Format (5)

<table>
<thead>
<tr>
<th>10</th>
<th>rd</th>
<th>110111</th>
<th>rs1</th>
<th>rs3</th>
<th>var</th>
<th>size</th>
<th>rs2</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>25</td>
<td>24</td>
<td>19</td>
<td>18</td>
<td>14</td>
</tr>
</tbody>
</table>

処理          | 演算           |
------------|----------------|
乗算と加算    | rd ← rs1 × rs2 + rs3 |
乗算と減算    | rd ← rs1 × rs2 − rs3 |
乗算と減算後符号反転 | rd ← − rs1 × rs2 + rs3 |
乗算と加算後符号反転 | rd ← − rs1 × rs2 − rs3 |
FMADD は、rs1 で指定される浮動小数点レジスタに rs2 で指定される浮動小数点レジスタを乗じ、その結果に rs3 で指定される浮動小数点レジスタの値を加え、rd で指定される浮動小数点レジスタに格納する。

FMSUB は、rs1 で指定される浮動小数点レジスタに rs2 で指定される浮動小数点レジスタを乗じ、その結果から rs3 で指定される浮動小数点レジスタの値を引き、rd で指定される浮動小数点レジスタに格納する。

FNMAADD は、rs1 で指定される浮動小数点レジスタに rs2 で指定される浮動小数点レジスタを乗じ、その結果の符号を反転させた値から rs3 で指定される浮動小数点レジスタの値を引き、rd で指定される浮動小数点レジスタに格納する。

FNMSUB は、rs1 で指定される浮動小数点レジスタに rs2 で指定される浮動小数点レジスタを乗じ、その結果の符号を反転させた値に rs3 で指定される浮動小数点レジスタの値を加え、rd で指定される浮動小数点レジスタに格納する。

浮動小数点積和演算命令は一つの連続した (fused) 命令として処理される。つまり、乗算の結果は内部的に丸められることなく無限の精度を持つとして扱われ、加減算ののちに丸め処理が行われる。したがって、丸め処理による誤差が発生するのは一回だけである。

### Programming Note – SPARC64 V
SPARC64 V では浮動小数点積和演算は、乗算と加算を独立して処理していた。つまり、乗算の結果は内部的に丸められることなく無限の精度を持つとして扱われ、加減算ののちに丸め処理が行われた。したがって、丸め処理による誤差が発生するのは一回だけである。

また、FNMAADD と FNMSUB の rs1 または rs2 が NaN のときの動作も SPARC64 V と SPARC64 Ixfx で異なる。SPARC64 Ixfx は結果としてどちらかの NaN をそのまま出力するが、SPARC64 V は符号を反転させた NaN を結果として出力する。

### TABLE A-3
SPARC64 Ixfx が浮動小数点積和演算でトラップをどのように処理するかをまとめた。乗算部分でトラップが生じる無効処理例 (NV) が生じるか、FSR.NS = 1 で乗算の入力が非正規化数のとき、命令の実行は中断されトラップが発生する。
生する。このとき FSR.cexc には例外の情報が表示され、FSR.aexc は更新されない。加減算の部分は乗算部でトラブルする無効処理例外 (NV) が生じなかったときのみ実行される。

加減算部分でトラップを生じる IEEE754 例外条件が発生したときは、トラップを生じる例外条件のみが FSR.cexc に表示され、FSR.aexc は更新されない。トラブルを生じる IEEE754 例外条件が発生していない場合、トラップを生じない例外条件の情報が FSR.cexc に表示され、FSR.aexc にはそれまでの FSR.aexc と FSR.cexc の論理和が表示される。unfinished_FPpop 例外を通知する境界条件は、乗算部分は rs1, rs2 について FMUL と同じ、加減算部分は乗算結果と rs3 について FADD と同じである。

<table>
<thead>
<tr>
<th>TABLE A-3</th>
<th>浮動小数点積和演算命令の IEEE754 例外発生条件</th>
</tr>
</thead>
<tbody>
<tr>
<td>FMUL</td>
<td>FADD</td>
</tr>
<tr>
<td>IEEE754</td>
<td>トラブル (NV または NX のみ)</td>
</tr>
<tr>
<td>FADD</td>
<td>トラブルを生じない</td>
</tr>
<tr>
<td></td>
<td>トラブルを生じない</td>
</tr>
<tr>
<td>cexc</td>
<td>FMUL の例外検出条件と同じ</td>
</tr>
<tr>
<td></td>
<td>FADD の例外検出条件と同じ</td>
</tr>
<tr>
<td></td>
<td>FADD の例外検出条件 (トラップしない)</td>
</tr>
<tr>
<td></td>
<td>FADD の例外検出条件 (トラップしない)</td>
</tr>
<tr>
<td>aexc</td>
<td>更新されない</td>
</tr>
<tr>
<td></td>
<td>cexc (上記) と aexc の論理和</td>
</tr>
</tbody>
</table>

cexc に表示される値は各種条件に依存する。詳細を TABLE A-4, TABLE A-5 にまとめた。この表で uf, of, nv, nx はそれぞれ IEEE754 で定義された例外 (uf: アンダーフロー, of: オーバーフロー, nv: 無効演算, nx: 不正確な演算) でトラップしない場合を意味する。

<table>
<thead>
<tr>
<th>TABLE A-4</th>
<th>トラブルしない場合の cexc の値 (FSR.NS = 0 の時)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FMUL</td>
<td>FADD</td>
</tr>
<tr>
<td>none</td>
<td>none</td>
</tr>
<tr>
<td>nv</td>
<td>none</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE A-5</th>
<th>トラブルしない場合の cexc の値 (FSR.NS = 1 の時)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FMUL</td>
<td>FADD</td>
</tr>
<tr>
<td>none</td>
<td>none</td>
</tr>
<tr>
<td>nv</td>
<td>none</td>
</tr>
</tbody>
</table>

表中 “—” のケースは起こりえない。


Programming Note – 浮動小数点積和演算は SPARC V9 の IMPDEP2 に定義されている。この命令は SPARC64 IXfx 固有の命令であり、他の SPARC V9 プロセッサで実行されるかもしれないプログラムでは使用ができない。

FMADD の SIMD 拡張

SPARC64 IXfx の SIMD 拡張では、basic 側の演算と extended 側の演算は独立して行われることになっている。命令で指定するのは basic 側のレジスタ f[0]～f[254] なので、rs1, rs2, rs3, rd の最上位ビットは必ず 0 になる (page 22)。これに対し、FMADD の SIMD 拡張では、制限つきではあるが basic と extended の間での演算が可能となっている。

Note – ここに書かれているのは XAR.simd = 1 のときのことである。XAR.simd = 0 のときは rs1, rs2, rs3, rd にすべての浮動小数点レジスタが使える。

FMADD の SIMD 拡張では、rs1, rs2 については basic 側、extended 側どちらにも全てのレジスタ f[2n] (n=0...255) を指定することができる。basic 側演算に extended 側のレジスタを指定すると、extended 側にはベースになる basic 側のレジスタが使われる。つまり、basic 側では f[2n] (n=0...255) が、extended 側では f[(2n+256) mod 512] (n=0...255) が使われる。

これに対し rs3 と rd は他の SIMD 拡張と同じで、basic 側演算では f[0]～f[254], extended 側演算では f[256]～f[510] という制限のままである。したがって、urs1<2> と urd<2> はレジスタ指示には使われない。他の SIMD 拡張ではこれらのビットは 0 である必要があるが、FMADD ではこのビットを、演算を変化させるために使用する。urs1<2> = 1 のとき、extended 側演算の rs1 には basic 側と同じもののが使われる。urd<2> = 1 のとき、extended 側演算の積演算の符号を反転させる。

FMADD の SIMD 演算での XAR.urs1, XAR.urs2, XAR.urs3, XAR.urd の意味をまとめると下記のようになる。

- XAR.urs1<2> : basic 側演算の rs1<8>, extended 側演算の ¬rs1<8>
- XAR.urs2<2> : basic 側演算の rs2<8>, extended 側演算の ¬rs2<8>
- XAR.urs3<2> : extended 側演算に rs1<8> と ¬rs1<8> のどちらを使うか
- XAR.urd<2> : extended 側演算の積演算の符号を反転させるかどうか

なお、上の rs1<8> は、倍精度レジスタのビットテコード後のビットを意味する。詳細は FIGURE 5-1 (page 21) 参照。
例 1: 複素数の乗算
\[(a_1 + i b_1) (a_2 + i b_2) = (a_1 a_2 - b_1 b_2) + i (a_1 b_2 + a_2 b_1)\]

/*
* X: 入力の複素数のアドレス
* Y: 入力の複素数のアドレス
* Z: 出力の複素数のアドレス
*/

/* 前準備 */
sxar2
ldd,s [X], %f0 /* %f0: a1, %f256: b1 */
ldd,s [Y], %f2 /* %f2: a2, %f258: b2 */
sxar1
fzero,s %f4 /* 結果レジスタをクリアしておく */

/* 計算 */
sxar2
fmadd, snc %f256, %f258, %f4, %f4
/* %f4 := -%f256 * %f258 - %f4 */
/* %f260 := %f256 * %f2 - %f260 */
fmadd, sc %f0, %f2, %f4, %f4
/* %f4 := %f0 * %f2 + %f4 */
/* %f260 := %f0 * %f258 + %f260 */

/* 結果を格納する */
sxar1
std,s %f4, [Z]

例 2: 2x2 行列の乗算

<table>
<thead>
<tr>
<th>命令</th>
<th>basic 側演算</th>
<th>extend 側演算</th>
</tr>
</thead>
<tbody>
<tr>
<td>fmadd</td>
<td>%frd_h ← %frs1 × %frs2 + %frs3_b</td>
<td>%frd_e ← (-1)^n × (c ? %frs1 : %frs1) × %frs2 + %frs3_e</td>
</tr>
<tr>
<td>fmsub</td>
<td>%frd_h ← %frs1 × %frs2 - %frs3_b</td>
<td>%frd_e ← (-1)^n × (c ? %frs1 : %frs1) × %frs2 - %frs3_e</td>
</tr>
<tr>
<td>fnmsub</td>
<td>%frd_h ← - %frs1 × %frs2 + %frs3_b</td>
<td>%frd_e ← (-1)^n × (c ? %frs1 : %frs1) × %frs2 + %frs3_e</td>
</tr>
<tr>
<td>fnmadd</td>
<td>%frd_h ← - %frs1 × %frs2 - %frs3_b</td>
<td>%frd_e ← (-1)^n × (c ? %frs1 : %frs1) × %frs2 - %frs3_e</td>
</tr>
</tbody>
</table>
/* A: 入力の行列のアドレス: a11, a12, a21, a22
   * B: 入力の行列のアドレス: b11, b12, b21, b22
   * C: 出力の行列のアドレス: c11, c12, c21, c22 */

/* 前準備 */
sxar2
ldd, s [A], %f0 /* %f0: a11, %f256: a12 */
ldd, s [A+16], %f2/* %f2: a21, %f258: a22 */
sxar2
ldd, s [B], %f4 /* %f4: b11, %f260: b12 */
ldd, s [B+16], %f6/* %f6: b21, %f262: b22 */
sxar2
fzero, s %f8 /* %f8: c11, %f264: c12 */
fzero, s %f10 /* %f10: c21, %f266: c22 */

/* 計算 */
sxar2
fmadd, sc %f0, %f4, %f8, %f8
   /* %f8 := %f0 * %f4 + %f8 */
   /* %f264 := %f0 * %f260 + %f264 */
fmadd, sc %f256, %f6, %f8, %f8
   /* %f8 := %f256 * %f6 + %f8 */
   /* %f264 := %f256 * %f262 + %f264 */
sxar2
fmadd, sc %f2, %f4, %f10, %f10
   /* %f10 := %f2 * %f4 + %f10 */
   /* %f266 := %f2 * %f260 + %f266 */
fmadd, sc %f258, %f6, %f10, %f10
   /* %f10 := %f258 * %f6 + %f10 */
   /* %f266 := %f258 * %f262 + %f266 */

/* 結果を格納する */
sxar2
std, s %f8, [Z]
std, s %f10, [Z+16]

Exceptions
illegal_instruction (size = 112 and var ≠ 112)
   (このとき fp_disabled かどうかのチェックは行わない)
fp_disabled
fp_exception_ieee_754 (NV, NX, OF, UF)
fp_exception_other (FSR.f1t = unfinished_FPop)
A.24.2 Suspend

**Description**

SUSPEND 指令是，将 PSTATE.IE = 1 位设置，使执行线程进入 SUSPENDED 状态。以下条件允许从 SUSPENDED 状态恢复到执行状态。

- POR, WDR, XIR
- interrupt_vector
- interrupt_level_n

**Exceptions**

privilegedOpcode
illegal_action (XAR.v = 1)

**Format (3)**

<table>
<thead>
<tr>
<th>10</th>
<th>—</th>
<th>110110</th>
<th>—</th>
<th>opf</th>
<th>—</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>25</td>
<td>24</td>
<td>19</td>
</tr>
</tbody>
</table>

HPC-ACE Ext.
Regs. SIMD オペコード opf 処理
SUSPEND 0 1000 0010 スレッドのサスペンド

アセンブリ言語表現

suspend

SUSPEND 指令是，将 PSTATE.IE = 1 位设置，使执行线程进入 SUSPENDED 状态。以下条件允许从 SUSPENDED 状态恢复到执行状态。

- POR, WDR, XIR
- interrupt_vector
- interrupt_level_n

**Exceptions**

privilegedOpcode
illegal_action (XAR.v = 1)
A.24.3 Sleep

**Format (3)**

<table>
<thead>
<tr>
<th>10</th>
<th>—</th>
<th>110110</th>
<th>—</th>
<th>opf</th>
<th>—</th>
</tr>
</thead>
<tbody>
<tr>
<td>31 30 29 25 24 19 18 14 13 5 4 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Description**

SLEEP は実行スレッドをスリープさせる。実行状態に復帰する条件は

- POR, WDR, XIR
- interrupt_vector 例外
- interrupt_level_n 例外
- 実装により決まる一定時間経過後
  SPARC64 IXfx では 1.6 マイクロ秒。STICK で計測される。
- バリアの窓 ASI に割り当てられている LBSY が更新されたとき。
  窓 ASI が割り当てられていないければ LBSY が更新されても実行状態に復帰しない。

**Note**

PSTATE.IE = 0 で SLEEP 命令を実行すると、インタラプトがあっても実行状態に復帰しない。

**Programming Note**

スレッドがスリープしていない状態で LBSY が更新されると、次の SLEEP 命令の実行でスリープしないことがある。

**Exceptions**

illegal_action (XAR.v = 1)
### A.24.4 Integer Multiply-Add

SPARC64 IXfx は IMPDEP2A に整数積和演算を定義している。

<table>
<thead>
<tr>
<th>Regs.</th>
<th>SIMD</th>
<th>オペコード</th>
<th>Var</th>
<th>Size</th>
<th>处理</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>✓</td>
<td>FPMADDX</td>
<td>00</td>
<td>00</td>
<td>符号なし整数の積和演算の下位 8 バイト</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>FPMADDXHI</td>
<td>01</td>
<td>00</td>
<td>符号なし整数の積和演算の上位 8 バイト</td>
</tr>
</tbody>
</table>

Table (5)

<table>
<thead>
<tr>
<th>10</th>
<th>rd</th>
<th>110111</th>
<th>rs1</th>
<th>rs3</th>
<th>var</th>
<th>size</th>
<th>rs2</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>25</td>
<td>24</td>
<td>19</td>
<td>18</td>
<td>14</td>
</tr>
</tbody>
</table>

アセンブリ言語表記

- FPMADDX: `freg_r1, freg_r2, freg_r3, freg_d`
- FPMADDXHI: `freg_r1, freg_r2, freg_r3, freg_d`

#### Description

整数積和演算は、浮動小数点レジスタに格納された符号なし 8 バイト整数の乗算と加算を行う。

FPMADDX は rs1 で指定される倍精度レジスタと rs2 で指定される倍精度レジスタを乗じ、その結果に rs3 で指定される倍精度レジスタを加算し、結果の下位 8 バイトを rd で指定される倍精度レジスタに格納する。rs1, rs2, rs3 はすべて符号なし 8 バイト整数として扱われる。

FPMADDXHI は rs1 で指定される倍精度レジスタと rs2 で指定される倍精度レジスタを乗じ、その結果に rs3 で指定される倍精度レジスタを加算し、結果の上位 8 バイトを rd で指定される倍精度レジスタに格納する。rs1, rs2, rs3 はすべて符号なし 8 バイト整数として扱われる。

FPMADDX, FPMADDXHI は FSR のどのフィールドも更新しない。

FPMADDX, FPMADDXHI は IMPDEP2 で定義された命令だが、PA の `Impdep2_instruction` イベントでは計測されない。詳細は Appendix Q.2.1, “命令種類、トラップ種類毎の統計情報” (page 308) を参照。
Exceptions

*fp_disabled*

*illegal_action* (XAR.\texttt{v} = 1 and XAR.\texttt{simd} = 1 and
\(XAR.\texttt{urs1}<2> \neq 0 \) or \(XAR.\texttt{urs2}<2> \neq 0 \)
or \(XAR.\texttt{urs3}<2> \neq 0 \) or \(XAR.\texttt{urd}<2> \neq 0 \))
A.25 Jump and Link

PSTATE.AM がセットされているとき、r[rd] には PC の上位 32 ビットを 0 にした値が入る (impl. dep. #125)。r[rd] の更新は直ちに行われ、ディレイスロットの命令には新しい値が見える。

飛び先アドレスの下位 2 ビットが 0 でない場合、mem_address_not_aligned 例外が通知される。このとき、DSFSR および DSFAR は更新されない (impl. dep. #237)。

Exceptions  illegal_action (XAR.v = 1)
A.26 Load Floating-Point

### Format (3)

<table>
<thead>
<tr>
<th>rd</th>
<th>op3</th>
<th>rs1</th>
<th>i=0</th>
<th>—</th>
<th>rs2</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Description

まず non-SIMD での動作仕様を説明する。

単精度浮動小数点ロード命令 (LDF) は、メモリの4バイト境界にある4バイトデータを \( f[rd] \) にコピーする。
倍精度浮動小数点ロード命令 (LDDF) は、メモリの 4 バイト境界にある 8 バイトデータを倍精度浮動小数点レジスタにコピーする。

4 倍精度浮動小数点ロード命令 (LDQF) は、メモリの4 バイト境界にある16 バイトデータを4 倍精度浮動小数点レジスタにコピーする。

浮動小数点ステータスレジスタロード命令 (LDXFSR) は、未完了のすべての浮動小数点演算の完了を持ってから、8 バイトデータを FSR にコピーする。

これらの浮動小数点ロード命令は、プライマリアドレス空間 (ASI = 8016) にアクセスする。メモリアドレスは i = 0 のとき "r[rs1]+r[rs2]", i = 1 のとき "r[rs1]+sign_ext(simm13)" である。

LDF は、アクセスするアドレスが 4 バイト境界にない場合、mem_address_not_aligned を通知する。LDFXSR は、アクセスするアドレスが 8 バイト境界にない場合、mem_address_not_aligned を通知する。浮動小数点演算器が無効なとき (FPRS.FEF, PSTATE.PEF の値による）、浮動小数点ロード命令は fp_disabled を通知する。

SPARC64 IXfx では、SIMD ではない LDDF は、アクセスするアドレスが 8 バイト境界でない 4 バイト境界の場合、LDDF_mem_address_not_aligned 例外を通知する。システムソフトウェアはエミュレーションすること、(impl.dep. #109(1))。

SPARC64 IXfx は LDQF を実装していないので、illegal_instruction 例外を通知する。fp_disabled は検出しない。システムソフトウェアは LDQF をエミュレーションすること、(impl.dep. #111(1))。

Programming Note – SPARC V8 では、倍精度または4 倍精度データが適切にアラインされている保証がないとき、複数の単精度ロード命令で処理するコードを生成するコンパイラがあった。SPARC V9 ではアラインされていない命令のエミュレーションが高速になると期待されるので、複数の単精度ロード命令で処理するのは、倍精度または4 倍精度データが適切にアラインされていないことが確実なときのみにすることを推奨する。

Programming Note – SPARC64 IXfx では、SIMD ではない浮動小数点ロードがアクセスエラーを起こした時、デスティネーションレジスタは無変更のままである (impl.dep. #44(1))。SIMD のときは下記を参照。

Programming Note – 単精度浮動小数点ロード命令 LDF のアドレス指定 (rs1, rs2) に HPC-ACE 拡張整数レジスタ xg[0] - xg[31] を使う場合、ロード先のレジスタは倍精度レジスタとなる。これは XAR.v = 1 のときの rd のデコーデ定義から来る制約であり (page 21)。rs1, rs2 に拡張整数レジスタ、rd に SPARC V9 単精度レジスタ (奇数番号レジスタ) を指定する方法はない。
SIMD 拡張

SPARC64 IX fx では浮動小数点ロード命令は SIMD 拡張される。SIMD 拡張されたロード命令は、basic 側の単精度・倍精度ロードと extended 側の単精度・倍精度ロードを同時に実行する。レジスタの使用方法は "SIMD 拡張命令のレジスタ指定方法 " (page 22) を参照。

単精度 SIMD ロードは 4 バイト境界にある 2 つの単精度数データをロードする。アクセス境界違反には mem_address_not_aligned が通知される。

倍精度 SIMD ロードは 8 バイト境界にある 2 つの倍精度数データをロードする。アクセス境界違反には mem_address_not_aligned が通知される。

Note – 倍精度 SIMD ロードでは、8 バイト境界ではない 4 バイト境界に対するアクセスでも LDDF_mem_address_not_aligned は通知されない。

SIMD ロードは、単精度・倍精度とも、basic 側でロードされるデータと extended 側でロードされるデータが、異なるページに属することがある。このようなアクセスに対し SPARC64 IX fx では、SIMD_load_across_pages 例外を通知する。例外を通知する条件は、basic 側の TLB 検索が成功し、extended 側の TLB 検索が失敗したときにである。

SIMD ロードはキャッシュ領域からのみロードできる。ノンキャッシュ領域および nontranslating ASI に対して SIMD ロードを実行しようとすると、data_access_exception が通知される。bypass ASI に対する SIMD ロードは、AST_PHYS_USE_EC{ Little } については可能で、その場合ページサイズは 8KB と解釈される (Appendix F.11, "MMU Bypass" (page 202) 参照)。

メモリアクセスのセマンティクスは通常のロード命令と同じく、TSO を遵守する。SIMD ロードは 1 命令で basic と extended の両方を同時にロードするが、basic と extended の間でも TSO が遵守されるものとする。

SPARC64 IX fx では、SIMD 浮動小数点ロードがアクセスエラーを起こした時、デスティネーションレジスタは無変更のままである (impl.dep. #44(1))。

SIMD ロードにおけるエンディアン変換は、basic, extended 個別に行われる。basic, extended が異なるページに属し、それぞれのページのエンディアンが異なる場合も、一方のみエンディアン変換が行われる。

SIMD ロードは basic, extended どちらでもウォッチポイントを検出する。

Note – PSTATE.AM=1 のとき、論理アドレス FFFF FFFF FFFC16 対する単精度 SIMD ロード、および論理アドレス FFFF FFFF FFF816 対する倍精度 SIMD ロードの extended 側は、論理アドレス 0 番地にアクセスする。

SIMD ロードの例外条件と優先順位に関しては Appendix F.5.1, "Trap Conditions for SIMD Load/Store" (page 180) を参照。
Exceptions

illegal_instruction (LDQF; LDXFSR with rd = 2–31)
fp_disabled

illegal_action (LDF, LDDF with XAR.v = 1 and (XAR.urs1 > 1 or (i = 0 and XAR.urs2 > 1)) or (i = 1 and XAR.urs2 ≠ 0) or XAR.urs3<2> ≠ 0);
LDF, LDDF with XAR.v = 1 and XAR.simd = 1 and XAR.urd<2> ≠ 0;
LDXFSR with XAR.v = 1 and (XAR.urs1 > 1 or (i = 0 and XAR.urs2 > 1)) or (i = 1 and XAR.urs2 ≠ 0) or XAR.urs3<2> ≠ 0 or XAR.urd ≠ 0 or XAR.simd = 1))

LDDF_mem_address_not_aligned (LDDF and (XAR.v = 0 or XAR.simd = 0))
mem_address_not_aligned
VA_watchpoint
fast_data_access_MMU_miss
SIMD_load_across_pages
data_access_exception
PA_watchpoint
data_access_error
fast_data_access_protection
A.27 Load Floating-Point from Alternate Space

**HPC-ACE Ext.**

<table>
<thead>
<tr>
<th>Regs.</th>
<th>SIMD</th>
<th>オペコード</th>
<th>op3</th>
<th>rd</th>
<th>urd</th>
<th>处理</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>LDFA&lt;sub&gt;PAS&lt;/sub&gt;</td>
<td>11 0000</td>
<td>0–31</td>
<td>—</td>
<td>别空间から単精度浮点数点レジスタへの読みだし</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>LDFA</td>
<td>11 0000</td>
<td>†</td>
<td>0-7</td>
<td>別空間から単精度浮点数点レジスタへの読みだし</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>LDDFA&lt;sub&gt;PAS&lt;/sub&gt;</td>
<td>11 0011</td>
<td>†</td>
<td>0-7</td>
<td>別空間から倍精度浮点数点レジスタへの読みだし</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>LDQFA&lt;sub&gt;PAS&lt;/sub&gt;</td>
<td>11 0010</td>
<td>†</td>
<td>0-7</td>
<td>別空間から4倍精度浮点数点レジスタへの読みだし</td>
</tr>
</tbody>
</table>

† JPSI Commonality のSection 5.1.4 で定義される浮点数点レジスタエンコードに従う。
‡ XAR.<sub>v</sub> = 0 のとき

**Format (3)**

<table>
<thead>
<tr>
<th>rd</th>
<th>op3</th>
<th>rs1</th>
<th>imm&lt;sub&gt;asi&lt;/sub&gt;</th>
<th>rs2</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31 30 29 25 24 19 18 14 13 12 5 4 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

アセンブリ言語表現

| lda | [regaddr] imm<sub>asi</sub>, freg<sub>rd</sub> |
| da | [reg_′plus_imm] %asi, freg<sub>rd</sub> |
| ldqa | [regaddr] imm<sub>asi</sub>, freg<sub>rd</sub> |
| ldda | [reg_′plus_imm] %asi, freg<sub>rd</sub> |
| lddq | [reg_′plus_imm] %asi, freg<sub>rd</sub> |

**Description**

まず non-SIMD での動作仕様を説明する。

別空間からの単精度浮点数点ロード命令 (LDFA) は、メモリの 4 バイト境界にある 4 バイトデータを f[rd] にコピーする。

別空間からの倍精度浮点数点ロード命令 (LDDFA) は、メモリの 4 バイト境界にある 8 バイトデータを倍精度浮点数点レジスタにコピーする。
別空間からの4倍精度浮動小数点ロード命令 (LDQFA) は、メモリの4バイト境界にある16バイトデータを4倍精度浮動小数点レジスタにコピーする。

これら別空間からの浮動小数点ロード命令は、空間を示す識別者 (ASI) を必要とする。ASIは、i = 0 のとき imm_asl フィールドで指示され、i = 1 のとき ASI レジスタの値が使われる。ASIのビット7が0のときは特権アクセス、1のときは非特権アクセスである。メモリアドレスはi = 0 のとき “r[rs1] + r[rs2]”、i = 1 のとき “r[rs1] + sign_ext(simm13)” である。

LDFA は、アクセスするアドレスが4バイト境界でない場合、mem_address_not_aligned を通知する。浮動小数点演算器が無効なとき (FPRS.FEF, PSTATE.PEF の値による)、別空間からの浮動小数点ロード命令はfp_disabled を通知する。

SPARC64 IXfx では、SIMD でない LDFA は、アクセスするアドレスが8バイト境界でない4バイト境界の場合、LDDF_mem_address_not_aligned 例外を通知する。システムソフトウェアはエミュレーションすること (impl.dep. #109(2))。

SPARC64 IXfx は LDQFA を実装していないので、illegal_instruction 例外を通知する。fp_disabled は検出しない。システムソフトウェアは LDQFA をエミュレーションすること (impl.dep. #111(2))。

ASI番号によっては、8バイト以外のメモリアクセスを定義しているものがある。詳細は Appendix A の他の節を参照。

Implementation Note – LDFA, LDDFA は PSTATE.PRIV = 0 で ASI のビット7が0のとき privileged_action を通知する。

Programming Note – SPARC V8 では、倍精度または4倍精度データが適切にアラインされている保証がないとき、複数の単精度ロード命令で処理するコードを生成するコンパイラがあった。SPARC V9 ではアラインされていない命令のエミュレーションが高速になること期待されるので、コンパイラは倍精度または4倍精度データが適切にアラインされていないことが確実なときのみ、複数の単精度ロード命令を生成すること。

SPARC64 IXfx では、SIMD ではない浮動小数点ロードがアクセスエラーを起こした時、デスティネーションレジスタは無変更のままである (impl.dep. #44(2))。

Programming Note – 単精度浮動小数点ロード命令 LDFA のアドレス指定 (rs1, rs2) に HPC-ACE 拡張整数レジスタ xg[0] - xg[31] を使う場合、ロード先のレジスタは倍精度レジスタとなる。これは XAR.v=1 のときの rd のデコード定義から来ることである (page 21), rs1, rs2 に拡張整数レジスタ、rd に SPARC V9 単精度レジスタ (奇数番号レジスタ) を指定する方法はない。
SIMD拡張

Section A.26, “Load Floating-Point” の SIMD 拡張の項を参照。

Exceptions

illegal_instruction (LDQFA only)
fp_disabled
illegal_action (LDFA, LDDFA with XAR.v = 1 and (XAR.urs1 > 1 or
(i = 0 and XAR.urs2 > 1) or
(i = 1 and XAR.urs2 ≠ 0) or
XAR.urs3<2> ≠ 0);
LDFA, LDDFA with XAR.v = 1 and XAR.simd = 1 and XAR.urd<2> ≠ 0)
LDDF_mem_address_not_aligned (LDDFA and (XAR.v = 0 or XAR.simd = 0))
mem_address_not_aligned
privileged_action
VA_watchpoint
fast_data_access_MMU_miss
SIMD_load_across_pages
data_access_exception
fast_data_access_protection
PA_watchpoint
data_access_error
A.30 Load Quadword, Atomic [Physical]

16 バイトロード ASI は SPARC64 IXfx 固有の ASI である。

<table>
<thead>
<tr>
<th>HPC-ACE.EK</th>
<th>Regs.</th>
<th>SIMD</th>
<th>オペコード</th>
<th>imm_asi</th>
<th>ASI value</th>
<th>处理</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>LDDA</td>
<td></td>
<td>ASI_QUAD_LDD_PHYS</td>
<td>34_{16}</td>
<td>物理アドレス指定で 128 ビット一括読みだし</td>
<td></td>
</tr>
<tr>
<td>✓</td>
<td>LDDA</td>
<td></td>
<td>ASI_QUAD_LDD_PHYS_L</td>
<td>3C_{16}</td>
<td>物理アドレス指定で 128 ビット一括読みだし、エンディアン変換あり</td>
<td></td>
</tr>
</tbody>
</table>

Format (3) LDDA

| 11 | rd | 010011 | rs1 | i=0 | imm_asi | rs2 |
| 11 | rd | 010011 | rs1 | i=1 | simm_13 |

アセンブリ言語表記

| ldda | [reg_addr] imm_asi, reg,r
| ldda | [reg_plus_imm] %asi, reg,r

Description

ASI 34_{16}, 3C_{16} は LDDA 命令で使われ、物理アドレスで指定された領域から 128 ビット長のデータを一度に (atomically) コピーする。データは偶数・奇数の 64 ビットレジスタの組にロードされる。アドレスの低位にある 64 ビットが偶数番号のレジスタに、高位にある 64 ビットが奇数番号のレジスタに格納される。

ASI 34_{16}, 3C_{16} は論理アドレスの 16 バイトロード (ASI 24_{16}, 2C_{16}) の物理アドレス版で、SPARC64 IXfx 固有の ASI である。16 バイト境界にないアドレスにアクセスすると mem_address_not_aligned が通知される。

ASI_QUAD_LDD_PHYS{{−L}} を使ったアクセスは、TTE の設定が以下であるかのように振舞う。

- TTE.NFO = 0
- TTE.CP = 1
- TTE.CV = 0
- TTE.E = 0
- TTE.P = 1
\[ TTE.W = 0 \]

**Note** – \( TTE.IE \) の値は ASI によって異なる。03416 の場合は \( TTE.IE = 0 \) で、03C16 の場合は \( TTE.IE = 1 \) である。

このため、この ASI はキャッシュプル空間にのみ使用できる。意味的には、ASI_QUAD_LDD_PHYS\{ _L \} は ASI_NUCLEUS_QUAD_LDD と ASI_PHYS_USE_EC を組み合わせたものである。

エンディアンの扱いは、64 ビット毎に行われる。各 64 ビットがバイト単位でエンディアン変換され、結果レジスタに書き込まれる。

**Exceptions**

- **illegal_instruction** (misaligned rd)
- **illegal_action** (XAR.\textit{v} = 1 and (XAR.urs1 > 1 or (i = 0 and XAR.urs2 > 1) or (i = 1 and XAR.urs2 ≠ 0) or XAR.urs3 < 2) ≠ 0 and XAR.urd > 1);
- **privileged_action**
- **mem_address_notAligned**
- **fast_data_access_MMU_miss**
- **data_access_exception**
- **fast_data_access_protection**
- **PA_watchpoint** (recognized on only the first 8 bytes of a transfer)
- **data_access_error**
A.35 Memory Barrier

**Format (3)**

<table>
<thead>
<tr>
<th>10</th>
<th>0</th>
<th>op3</th>
<th>0 1111</th>
<th>i=1</th>
<th>—</th>
<th>cmask</th>
<th>mmask</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>25</td>
<td>24</td>
<td>19</td>
<td>18</td>
<td>14</td>
</tr>
</tbody>
</table>

**Description**

メモリバリア命令 MEMBAR は、メモリ参照の順序制御と完了の明示的な制御という2つの相異なる機能を持つ。アセンブリ言語の文法における membar_mask フィールドで、命令フィールドの cmask と mmask を指示する。

membar は命令のビット3から0にあるフィールドである。TABLE A-6 で mmask の各ビットについて説明する。これはメモリ参照に関して MEMBAR を入れることで付加されるメモリ順序制御を意味する。

**TABLE A-6**

<table>
<thead>
<tr>
<th>mmask ビット</th>
<th>mmask 仮名</th>
<th>説明</th>
</tr>
</thead>
<tbody>
<tr>
<td>mmask&lt;3&gt;</td>
<td>#StoreStore</td>
<td>MEMBAR の前に現れるストアの結果が、すべてのプロセッサにおいて、MEMBAR の後に現れるストアの結果より前に観測可能であることを保証する。非推奨の STBAR 命令と同じ役割を果たす。SPARC64 IXfx ではすべてのストア命令がプログラム順序で実行されるので、このビットは意味を持たない。</td>
</tr>
<tr>
<td>mmask&lt;2&gt;</td>
<td>#LoadStore</td>
<td>MEMBAR の前に現れるロード命令が、MEMBAR の後に現れるストアがいずれかのプロセッサで観測可能になる前に、実行されることを保証する。SPARC64 IXfx ではすべてのストア命令がプログラム順序で実行され、ロード命令との順序は保証されているので、このビットは意味を持たない。</td>
</tr>
<tr>
<td>mmask&lt;1&gt;</td>
<td>#StoreLoad</td>
<td>MEMBAR の前に現れるストアの結果が、すべてのプロセッサにおいて、MEMBAR の後に現れるロード命令の実行前に観測可能であることを保証する。</td>
</tr>
<tr>
<td>mmask&lt;0&gt;</td>
<td>#LoadLoad</td>
<td>MEMBAR の前に現れるすべてのロードの実行が、MEMBAR の後に現れるロードの実行より前に完了していることを保証する。SPARC64 IXfx ではすべてのロード命令がプログラム順序で実行されるので、このビットは意味を持たない。</td>
</tr>
</tbody>
</table>
cmask は命令のビット 6 から 4 にあるフィールドである。TABLE A-7 (page 91) で cmask の各ビットについて説明する。これはメモリ参照と命令の実行に関する制限を付加するためのものである。もし cmask がゼロなら、MEMBAR は mmask フィールドで指定される部分的な順序制御を指示する。もし cmask がゼロでなければ、部分的な順序制御と完了制御が適用される。

<table>
<thead>
<tr>
<th>マスクビット</th>
<th>機能</th>
<th>名前</th>
<th>説明</th>
</tr>
</thead>
<tbody>
<tr>
<td>cmask&lt;2&gt;</td>
<td>同期バリア</td>
<td>#Sync</td>
<td>MEMBAR より前に実行されるべきすべての命令 (メモリアクセス以外の命令も) が実行され、例外があれば、MEMBAR の後の命令の実行が開始される前に通知される。</td>
</tr>
<tr>
<td>cmask&lt;1&gt;</td>
<td>メモリ参照バリア</td>
<td>#MemIssue</td>
<td>MEMBAR の前に現れるすべてのメモリ参照命令が、MEMBAR の後に現れるどのメモリ参照命令の実行開始よりも前に実行されることを保証する。SPARC64 IXfx では #Sync 同じ。</td>
</tr>
<tr>
<td>cmask&lt;0&gt;</td>
<td>ルックアサイドバリア</td>
<td>#Lookaside</td>
<td>MEMBAR の前のストアが MEMBAR の後の同一アドレスを参照するロードより前に完了していることを保証する。SPARC64 IXfx では #Sync 同じ。</td>
</tr>
</tbody>
</table>

Exceptions  illegal_action (XAR.v = 1)
A.41 No Operation

Format (2)

<table>
<thead>
<tr>
<th>00</th>
<th>00000</th>
<th>op2</th>
<th>00000000000000000000000000000000</th>
</tr>
</thead>
<tbody>
<tr>
<td>31 30 29</td>
<td>25 24 22 21</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

アセンブリ言語表現

nop

Description

NOP は SETHI 命令で imm22 = 0 かつ rd = 0 を指定した場合と等価である。

NOP は PC と nPC 以外のレジスタ・メモリの状態を変更しない。ただし、NOP 命令を実行する時点で xar.urd = 1 の場合、SETHI 命令で rd に r[32] が指示されたものと解釈されるので、r[32] が更新される。

Exceptions

illegal_action (XAR.v = 1 and 
(XAR.simd = 1 or XAR.urs1 ≠ 0 or XAR.urs2 ≠ 0 or 
XAR.urs3 ≠ 0 or XAR.urd > 1))
A.42 Partial Store (VIS I)

SPARC64 IXfx ではパーシャルストアのウォッチポイント検出は保守的に行われる。DCUCR のマスクビットはゼロかゼロでないかのみがチェックされる。パーシャルストア命令のバイトストアマスク ($t[rs2]$) は無視され、バイトストアマスクがゼロ（つまり何もストアされない）場合でもウォッチポイント例外が通知される（impl. dep. #249）。

Implementation Note – ノンキャッシュ領域に対するパーシャルストア領域でストアマスクが 0 のとき、SPARC64 IXfx はマスク 0 のバストランザクションを生成する。

Exceptions

- illegal_instruction ($i = 1$)
- fp_disabled
- illegal_action ($XAR.\ \nu = 1$)
- LDDF_mem_address_not_aligned (see “Partial Store ASIs” (page 221))
- mem_address_not_aligned (see “Partial Store ASIs” (page 221))
- VA_watchpoint
- fast_data_access_MMU_miss
- data_access_exception (see “Partial Store ASIs” (page 221))
- fast_data_access_protection
- PA_watchpoint
- data_access_error
### A.48 Population Count

#### Format (3)

<table>
<thead>
<tr>
<th>Regs.</th>
<th>SIMD</th>
<th>ムーコード</th>
<th>op3</th>
<th>处理</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>POPC</td>
<td>10 1110</td>
<td></td>
<td>1 であるビット数を数える</td>
</tr>
</tbody>
</table>

#### Description

POPc は i = 0 のとき r[rs2] のビットが 1 である数を、i = 1 のとき sign_ext(simm13) のビットが 1 である数を返す。この命令は CCR を更新しない。

#### Note

SPARC64 V と異なり、SPARC64 IXfx はこの命令を実装している。

#### Exceptions

- **illegal_instruction** (instruction<18:14> ≠ 0)
- **illegal_action** (XAR. v = 1 and (XAR. urs1 ≠ 0 or (i = 0 and XAR. urs2 > 1)) or (i = 1 and XAR. urs2 ≠ 0) or XAR. urs3 ≠ 0 or XAR. urd > 1 or XAR. simd = 1)
A.49 Prefetch Data

SPARC64 IXfx では PREFETCHA は以下の ASI でのみ有効である。

- ASI_PRIMARY (08016), ASI_PRIMARY_LITTLE (08816)
- ASI_SECONDARY (08116), ASI_SECONDARY_LITTLE (08916)
- ASI_NUCLEUS (0416), ASI_NUCLEUS_LITTLE (0C16)
- ASI_PRIMARY_AS_IF_USER (01016), ASI_PRIMARY_AS_IF_USER_LITTLE (01816)
- ASI_SECONDARY_AS_IF_USER (01116), ASI_SECONDARY_AS_IF_USER_LITTLE (01916)

その他の ASI については PREFETCHA は NOP として扱われる。

SPARC64 IXfx では、データブロックのサイズは 128 バイトで、アラインメントも 128 バイト境界である (impl. dep. #103(3))。PREFETCH/PREFETCHA 命令はアラインメント制約はなく、データブロック内の任意のアドレスが指定でき、1 データブロックをプリフェッチする。

SPARC64 IXfx は、TTE.CP = 0 である領域をプリフェッチできない領域と認識し、TTE.CP = 0 に対するプリフェッチは NOP として扱われる。

TABLE A-8 に SPARC64 IXfx のプリフェッチの動作を示す。

TABLE A-8 プリフェッチの種類

<table>
<thead>
<tr>
<th>fcn</th>
<th>どのキャッシュに データを載せるか 説明</th>
<th>説明</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>L1D 読み出し用にプリフェッチする。</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>L2 読み出し用にプリフェッチする。</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>L1D 書き込み用にプリフェッチする。</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>L2 書き込み用にプリフェッチする。</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>— NOP</td>
<td></td>
</tr>
<tr>
<td>5-15</td>
<td>— Reserved (SPARC V9)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>illegal_instruction 例外が通知される。</td>
<td></td>
</tr>
<tr>
<td>16-19</td>
<td>— NOP</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>L1D ストロングプリフェッチ。読み出し用にプリフェッチする。</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>L2 ストロングプリフェッチ。読み出し用にプリフェッチする。</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>L1D ストロングプリフェッチ。書き込み用にプリフェッチする。</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>L2 ストロングプリフェッチ。書き込み用にプリフェッチする。</td>
<td></td>
</tr>
<tr>
<td>24-28</td>
<td>— NOP</td>
<td></td>
</tr>
</tbody>
</table>
ストロングプリフェッチ

fcn が 20-23, 29, 31 のいずれかであるプリフェッチ命令はストロングプリフェッチである。SPARC64 IXfx ではストロングプリフェッチは、TLB ミスと DCUCR.weak_spca = 1 の場合を除いてロストしないことを保証されたプリフェッチである。

Programming Note – ストロングでないプリフェッチは、CPU の内部資源が枯渇している場合などは実行されない（ロスト）ことがあるが、ストロングプリフェッチはそのような状況でも実行される。ストロングプリフェッチは後続のロード、ストア命令の実行を阻害するかもしれないのので、濫用は避けるべきである。

SPARC64 IXfx は fcn が 20-23, 29, 31 のときに fast_data_access_MMU_miss を通知しない (impl. dep. #103(2))。

ハードウェアプリフェッチ

PREFETCH, PREFETCHA ではハードウェアプリフェッチの on/off 指定は意味を持たない。XAR.dis_hw_pf の値は無視される。

Exceptions

illegal_instruction (fcn = 5–15)
illegal_action (XAR.v = 1 and (XAR.simd = 1 or XAR.urs1 > 1 or (i = 0 and XAR.urs2 > 1) or (i = 1 and XAR.urs2 ≠ 0) or XAR.urs3 < 2 ≠ 0 or XAR.urd ≠ 0))
A.51  Read State Register

上表の網かけ部分については、JPS1 Commonality の Section A.51, “Read State Register”を参照。

SPARC64 IIXf では、RDPCR は PSTATE.PRIV = 0 かつ PCR.PRIV = 1 のとき privileged_action 例外を通知する。PSTATE.PRIV = 0 かつ PCR.PRIV = 0 ならば、RDPCR は例外を通知しない。(impl. dep. #250)

RDTXAR は、PSTATE.PRIV = 0 のとき privileged_opcode 例外を通知する。
Exceptions

privileged_opcode (RDDCR, RDSOPTINT, RDTICK_CMPR, RDSTICK, RDSTICK_CMPR, and RDTXAR)

illegal_instruction (RDASR with rs1 = 1 or 7–14;
RDASR with rs1 = 15 and rd ≠ 0;
RDASR with rs1 = 20–21, 26–29;
RDTXAR with TL = 0)

fp_disabled (RDGSR with PSTATE.PEF = 0 or PPRS.PEF = 0)

illegal_action (XAR.v = 1 and
(XAR.simd = 1 or XAR.urs1 ≠ 0 or XAR.urs2 ≠ 0 or
XAR.urs3 ≠ 0 or XAR.urd > 1))

privileged_action (RDTICK with PSTATE.PRIV = 0 and TICK.NPT = 1;
RDPIC with PSTATE.PRIV = 0 and PCR.PRIV = 1;
RDSTICK with PSTATE.PRIV = 0 and STICK.NPT = 1;
RDPCR with PSTATE.PRIV = 0 and PCR.PRIV = 1)
A.59 SHUTDOWN (VIS I)

SPARC64 IXfx では SHUTDOWN は特権モードで NOP として動作する (impl. dep. #206)。

Exceptions

privileged_opcode
illegal_action (XAR.v = 1)
A.61 Store Floating-Point

### HPC-ACE Ext.

<table>
<thead>
<tr>
<th>Regs.</th>
<th>SIMD</th>
<th>オペコード</th>
<th>op3</th>
<th>rd</th>
<th>urd</th>
<th>处理</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>STF</td>
<td>10 0100</td>
<td>0–31</td>
<td>—</td>
<td>単精度浮動小数点レジスタのメモリ書き込み</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>STF</td>
<td>10 0100</td>
<td>†</td>
<td>0–7</td>
<td>単精度浮動小数点レジスタのメモリ書き込み</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>STDF</td>
<td>10 0111</td>
<td>†</td>
<td>0–7</td>
<td>倍精度浮動小数点レジスタのメモリ書き込み</td>
</tr>
<tr>
<td>✓</td>
<td></td>
<td>STQF</td>
<td>10 0110</td>
<td>†</td>
<td>0–7</td>
<td>4倍精度浮動小数点レジスタのメモリ書き込み</td>
</tr>
<tr>
<td>✓</td>
<td></td>
<td>STFSRD</td>
<td>10 0101</td>
<td>0</td>
<td>—</td>
<td>(A.71.11 of JPS1 Commonality 参照)</td>
</tr>
<tr>
<td>✓</td>
<td></td>
<td>STXFSR</td>
<td>10 0101</td>
<td>1</td>
<td>—</td>
<td>FSR レジスタのメモリ書き込み</td>
</tr>
<tr>
<td></td>
<td></td>
<td>STXFSR</td>
<td>—</td>
<td>10 0101</td>
<td>2–31</td>
<td>0</td>
</tr>
</tbody>
</table>

† JPSI Commonality の Section 5.1.4 で定義される浮動小数点レジスタエンコードに従う。

### Format (3)

<table>
<thead>
<tr>
<th>11</th>
<th>rd</th>
<th>op3</th>
<th>rs1</th>
<th>i=0</th>
<th>—</th>
<th>rs2</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>rd</td>
<td>op3</td>
<td>rs1</td>
<td>i=1</td>
<td>simm13</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>25</td>
<td>24</td>
<td>19</td>
<td>18</td>
</tr>
</tbody>
</table>

アセンブリ言語表記

- st frg<sub>rd</sub>, [address]
- std frg<sub>rd</sub>, [address]
- stq frg<sub>rd</sub>, [address]
- stx ∀fsx, [address]

### Description

まず non-SIMD での動作仕様を説明する。

単精度浮動小数点ストア命令 (STF) は、f[rd] の内容を 4 バイト境界にある 4 バイト領域にコピーする。
倍精度浮動小数点ストア命令 (STDF) は、倍精度浮動小数点レジスタの内容を 4 バイト境界にある 8 バイト領域にコピーする。

4 倍精度浮動小数点ストア命令 (STQF) は、4 倍精度浮動小数点レジスタの内容を 4 バイト境界にある 16 バイト領域にコピーする。

浮動小数点ステータスレジスタストア命令 (STXFSR) は、未完了のすべての浮動小数点演算の完了を待ってから、FSR の全 64 ビットをメモリにコピーする。書き込み後 FSR.ftt はゼロクリアされる。

Implementation Note – ストア命令がトラップしないことが確実になるまで、FSR.ftt をゼロクリアしてはいけない。

これらの浮動小数点ストア命令は、プライマリアドレス空間 (ASI = 8016) にアクセスする。メモリアドレスは i = 0 のとき “r[rs1]+r[rs2]”, i = 1 のとき “r[rs1]+sign_ext(simm13)” である。

STF は、アクセスするアドレスが 4 バイト境界にないとき、mem_address_not_aligned を通知する。STXFSR は、アクセスするアドレスが 8 バイト境界にないとき、mem_address_not_aligned を通知する。浮動小数点演算器が無効なとき (FPRS.FEF, PSTATE.PEF の値による）、浮動小数点ストア命令は fp_disabled を通知する。

SPARC64 IXfx では、SIMD でない STDF は、アクセスするアドレスが 8 バイト境界でない 4 バイト境界の場合、STDF_mem_address_not_aligned 例外を通知する。システムソフトウェアはエミュレーションすること (impl.dep. #110(1))。

SPARC64 IXfx は STQF を実装していないので、illegal_instruction 例外を通知する。fp_disabled は検出しない。システムソフトウェアは STQF をエミュレーションすること (impl.dep. #112(1))。

Programming Note – SPARC V8 では、倍精度または 4 倍精度データが適切にアラインされている保証がないとき、複数の単精度ストア命令で処理するコードを生成するコンバイラがあった。SPARC V9 ではアラインされていない命令のエミュレーションが高速になることから期待されるので、複数の単精度ストア命令で処理するのは、倍精度または 4 倍精度データが適切にアラインされていないことが確実なときのみにすることを推奨する。

Programming Note – 単精度浮動小数点ストア命令 STF のアドレス指定 (rs1, rs2) に HPC-ACE 拡張整数レジスタ xq[0] - xq[31] を使う場合、ストア元のレジスタは倍精度レジスタとなる。これは XAR.v=1 のときの rd の解釈定義から来る制約であり (page 21), rs1, rs2 に拡張整数レジスタ, rd に SPARC V9 単精度レジスタ (奇数番号レジスタ ) を指定する方法はない。
SIMD 拡張

SPARC64 IXfx では浮動小数点ストア命令は SIMD 拡張される。SIMD 拡張されたストア命令は、basic 側の単精度・倍精度ストアと extended 側の単精度・倍精度ストアを同時に実行する。レジスタの使用方法は “SIMD 拡張命令のレジスタ指定方法” (page 22) を参照。

単精度 SIMD ストアは 8 バイト境界に 2 つの単精度数データをストアする。アクセス境界違反には mem_address_not_aligned が通知される。

倍精度 SIMD ストアは 16 バイト境界に 2 つの倍精度数データをストアする。アクセス境界違反には mem_address_not_aligned が通知される。

Note – 倍精度 SIMD ストアでは、8 バイト境界ではない 4 バイト境界に対するアクセスでも STDF_mem_address_not_aligned は通知されない。さらに、倍精度 SIMD ストアは、倍精度 SIMD ロードと異なり 8 バイト境界に対するアクセスでも mem_address_not_aligned が通知されることに注意。

SIMD ストアはキャッシュプル領域に対してのみストアできる。ノンキャッシュプル領域および nontranslating ASI に対して SIMD ストアを実行しようとするとき、data_access_exception が通知される。bypass ASI に対する SIMD ストアは、ASI_PHYS_USE_EC_{LITTLE} については可能である。

メモリアクセスのセマンティクスは通常のストア命令と同じく、TSO を遵守する。SIMD ストアは 1 命令で basic と extended の両方を同時にストアするが、basic と extended の間でも TSO が遵守される。

SIMD ストアは basic, extended どちらでもウォッチポイントを検出する。

SIMD ストアの例外条件と優先順位に関しては Appendix F.5.1, “Trap Conditions for SIMD Load/Store” (page 180) を参照。

Exceptions

illegal_instruction (STXFSR with rd = 2–31)

fp_disabled

illegal_action (STF, STDF with XAR.v = 1 and (XAR.urs1 > 1 or (i = 0 and XAR.urs2 > 1) or (i = 1 and XAR.urs2 ≠ 0) or XAR.urs3<2> ≠ 0);

STF, STDF with XAR.v = 1 and XAR.simd = 1 and XAR.urd<2> ≠ 0;

STXFSR with XAR.v = 1 and (XAR.urs1 > 1 or (i = 0 and XAR.urs2 > 1) or (i = 1 and XAR.urs2 ≠ 0) or XAR.urs3<2> ≠ 0 or XAR.urd ≠ 0 or XAR.simd = 1));

mem_address_not_aligned

STDF_mem_address_not_aligned (STDF and (XAR.v = 0 or XAR.simd = 0))

VA_watchpoint
fast_data_access_MMU_miss
data_access_exception
fast_data_access_protection
PA_watchpoint
data_access_error
A.62 Store Floating-Point into Alternate Space

### Format (3)

<table>
<thead>
<tr>
<th>op3</th>
<th>rd</th>
<th>urd</th>
<th>rd11 op3</th>
<th>rs1</th>
<th>rs2</th>
<th>imm asi</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>29</td>
<td>24</td>
<td>19 18</td>
<td>14 13 12</td>
<td>5 4 0</td>
<td></td>
</tr>
</tbody>
</table>

### Description

まず non-SIMD での動作仕様を説明する。

別空間への単精度浮動小数点ストア命令 (STFA) は、f [rd] の内容を 4 バイト境界にある 4 バイト領域にコピーする。
別空間への倍精度浮動小数点ストア命令 (STDFA) は、倍精度浮動小数点レジスタの内容を 4 バイト境界にある 8 バイト領域にコピーする。

別空間への 4 倍精度浮動小数点ストア命令 (STQFA) は、4 倍精度浮動小数点レジスタの内容を 4 バイト境界にある 16 バイト領域にコピーする。

これら別空間への浮動小数点ストア命令は、空間を示す識別子 (ASI) を必要とする。ASI は、i = 0 のとき imm_asl フィールドで指示され、i = 1 のとき ASI レジスタの値が使われる。ASI のビット 7 が 0 のときは特権アクセス、1 のときは非特権アクセスである。メモリアドレスは i = 0 のとき "r[rs1] + r[rs2]"、i = 1 のとき "r[rs1] + sign_ext(simm13)" である。

STDFA は、アクセスするアドレスが 4 バイト境界にない場合、mem_address_not_aligned を通知する。浮動小数点演算器が無効なとき (FPRS.FEF、PSTATE.PEF の値による)、別空間への浮動小数点ストア命令は fp_disabled を通知する。

Implementation Note – STQFA ではこの検査は行われない。STFA と STDFA は PSTATE.PRIV = 0 で ASI のビット 7 が 0 のとき privileged_action を通知する。

ASI 番号によっては、8 バイト以外のメモリアクセスを定義しているものがある。詳細は Appendix A の他の節を参照。

SPARC64 IXfx では、SIMD でない STDFA は、アクセスするアドレスが 8 バイト境界でない 4 バイト境界の場合、STDF_mem_address_not_aligned 例外を通知する。システムソフトウェアはエミュレーションすること (impl.dep. #110(2))。

SPARC64 IXfx は STQFA を実装していないので、illegal_instruction 例外を通知する。fp_disabled は検出しない。システムソフトウェアは STQFA をエミュレーションすること (impl.dep. #112(2))。

Programming Note – SPARC V8 では、倍精度または 4 倍精度データが適切にアラインされている保証がないとき、複数の単精度ストア命令で処理するコードを生成するコンパイラがあった。SPARC V9 ではアラインされているない命令のエミュレーションが高速になること期待されるので、複数の単精度ストア命令で処理するのは、倍精度または 4 倍精度データが適切にアラインされていないことが確実なときのみにすることを推奨する。

Programming Note – 単精度浮動小数点ストア命令 STFA のアドレス指定 (rs1, rs2) に HPC-ACE 拡張整数レジスタ xg[0] - xg[31] を使う場合、ストア元のレジスタは倍精度レジスタとなる。これは XAR.v = 1 のときの rd のデコード定義から来る制約であり (page 21)、rs1, rs2 に拡張整数レジスタ、rd に SPARC V9 単精度レジスタ (奇数番号レジスタ) を指定する方法はない。
SIMD拡張 Section A.61, “Store Floating-Point”のSIMD拡張の項を参照。

Exceptions
fp_disabled
illegal_action (STFA, STDFA with XAR.v = 1 and (XAR.urs1 > 1 or
  (i = 0 and XAR.urs2 > 1) or
  (i = 1 and XAR.urs2 ≠ 0) or
  XAR.urs3<2> ≠ 0);
  STFA, STDFA with XAR.v = 1 and XAR.simd = 1 and XAR.urd<2> ≠ 0)
mem_address_not_aligned
STDF_mem_address_not_aligned (STDFA and (XAR.v = 0 or XAR.simd = 0))
privileged_action
VA_watchpoint
fast_data_access_MMU_miss
data_access_exception
fast_data_access_protection
PA_watchpoint
data_access_error
A.68 Trap on Integer Condition Codes (Tcc)

Tcc命令はXARの値によらず、JPS1 Commonalityで定義された通りに動作する。
illegal_action例外は起きない。

例外条件が成立してtrap_instructionが通知されるときは、Tcc命令実行直前のXAR
の内容がTXARにコピーされる。条件不成立のときは、XAR.f_v = 1ならば
XAR.f_*が0クリアされ、XAR.f_y = 0かつXAR.s_v = 1ならばXAR.s_*が0クリア
される。詳細は“XAR動作”(page 31)参照。

Programming Note – Tccはデッガのブレークポイント用に使われるため、任意
の位置に挿入できるようXARを無視する。

Exceptions
illegal_instruction (cc1 [ ] cc0 = 012 or 112, or reserved fields nonzero)
trap_instruction
A.69 Write Privileged Register

Format (3)

<table>
<thead>
<tr>
<th>rd</th>
<th>op3</th>
<th>rs1</th>
<th>i=0</th>
<th>—</th>
<th>rs2</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>rd</th>
<th>Privileged Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>TPC</td>
</tr>
<tr>
<td>1</td>
<td>TNPC</td>
</tr>
<tr>
<td>2</td>
<td>TSTATE</td>
</tr>
<tr>
<td>3</td>
<td>TT</td>
</tr>
<tr>
<td>4</td>
<td>TICK</td>
</tr>
<tr>
<td>5</td>
<td>TBA</td>
</tr>
<tr>
<td>6</td>
<td>PSTATE</td>
</tr>
<tr>
<td>7</td>
<td>TL</td>
</tr>
<tr>
<td>8</td>
<td>PIL</td>
</tr>
<tr>
<td>9</td>
<td>CWP</td>
</tr>
<tr>
<td>10</td>
<td>CANSAVE</td>
</tr>
<tr>
<td>11</td>
<td>CANRESTORE</td>
</tr>
<tr>
<td>12</td>
<td>CLEANWIN</td>
</tr>
<tr>
<td>13</td>
<td>OTHERWIN</td>
</tr>
<tr>
<td>14</td>
<td>WSTATE</td>
</tr>
<tr>
<td>15–31</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
この命令は、i=0 のとき "r[rs1] xor r[rs2]" を、i=1 のとき "r[rs1] xor sign_ext(simm13)" を、特権レジスタの書き込み可能フィールドに書き込む。排他論理和であることに注意。

命令の rd フィールドは書き込む特権レジスタを指定するために使われる。TPC, TNPC, TT, TSTATE レジスタはトラップレベル毎にレジスタがあるが、現在の TL に対応したレジスタに書き込まれる。TL = 0 で TPC, TNPC, TT, TSTATE に書き込むと illegal_instruction が通知される。

TL に対する書き込みは、トラップを生成したりトラップから復帰したりはしない。TL 以外の CPU の状態は変更されない。

Programming Note – TL に対する書き込みは、任意のトラップレベルでの TPC, TNPC, TT, TSTATE を読み出すために使われる。TL が変更されている間にトラップが起きないよう注意すること。

WRPR は直ちに実行される。特権レジスタを更新することによるマシンステートの変更の影響は、WRPR の直後で命令から観測できる。

rd が 15–31 の範囲の WRPR は、将来のアーキテクチャのために予約されている。rd にこれらの値を指定して WRPR を実行すると、illegal_instruction が通知される。
WRPR で PSTATE レジスタを更新する際、AG, IG, MG の Reserved な組み合わせを指定すると illegal_instruction が通知されるが、この例外通知は illegal_action よりも優先順位が低い。

Exceptions

privilegedOpcode

illegal_instruction ((rd = 15–31) or ((rd ≤ 3) and (TL = 0));
(rd = 6 and reserved combination of AG, IG, and MG))

illegal_action (XAR.v = 1 and (XAR.simd = 1 or
XAR.urs1 > 1 or
(i = 0 and XAR.urs2 > 1) or
(i = 1 and XAR.urs2 ≠ 0) or
XAR.urs3 ≠ 0 or
XAR.urd ≠ 0))
A.70 Write State Register

<table>
<thead>
<tr>
<th>Regs.</th>
<th>SIMD</th>
<th>オペコード</th>
<th>op3</th>
<th>rd</th>
<th>处理</th>
</tr>
</thead>
</table>
| ✓     | WRYD | 11 0000 0 | Y   | 0  | Y レジスタに書き込む。使用を推奨しない命令 (JPS1 Commonality の A.71.18 参照)。
|       |      | 11 0000 1 | Reserved |
| ✓     | WRCCR | 11 0000 2 | CCR | 2  | CCR レジスタに書き込む。
| ✓     | WRASI | 11 0000 3 | ASI | 3  | ASI レジスタに書き込む。
|       |      | 11 0000 4, 5 | Reserved |
| ✓     | WRFPRS | 11 0000 6 | FPRS | 6  | FPRS レジスタに書き込む。
|       |      | 11 0000 7–14 | Reserved |
|       |      | 11 0000 15 | SIR 命令。 (JPS1 Commonality の A.60 参照)。
| ✓     | WRASR | 11 0000 16–31 | SPARC V9 で定義されていない ASR に書き込む。
| ✓     | WRPCRPCR | 16 | PCR | 16 | PCR レジスタに書き込む。
| ✓     | WRPICPCR | 17 | PIC | 17 | PIC レジスタに書き込む。
| ✓     | WRDCR | 18 | DCR | 18 | DCR レジスタに書き込む。
| ✓     | WRGSR | 19 | GSR | 19 | GSR レジスタに書き込む。
| ✓     | WRSOFTINT_SETPCR | 20 | SOFTINT レジスタの特定ビットを 1 にする。
| ✓     | WRSOFTINT_CLRPCR | 21 | SOFTINT レジスタの特定ビットを 0 にする。
| ✓     | WRSOFTINT | 22 | SOFTINT レジスタに書き込む。
| ✓     | WRTICK_CMPRP | 23 | TICK_CMPR レジスタに書き込む。
| ✓     | WRSTICK | 24 | STICK レジスタに書き込む。
| ✓     | WRSTICK_CMPRP | 25 | STICK_CMPR レジスタに書き込む。
|       |      | 26–28 | Reserved |
| ✓     | WRXAR | 29 | XAR | 29 | XAR レジスタに書き込む。
| ✓     | WRXASR | 30 | XASR | 30 | XASR レジスタに書き込む。
| ✓     | WRTXARP | 31 | TXAR | 31 | TXAR レジスタに書き込む。

上表の網かけ部分については、JPS1 Commonality の Section A.70、“Write State Register” を参照。

SPARC64 Ix fx では、WRPCR は PSTATE.PRIV = 0 かつ PCR.PRIV = 1 のとき privileged_action 例外が通知される。PSTATE.PRIV = 0 かつ PCR.PRIV = 0 のときは、PCR.PRIV を変更しようとすると(すなわち 1 を書こうとすると) privileged_action 例外が通知される (impl. dep. #250)。
WRXAR, WRTXAR で XAR の Reserved フィールドに 0 以外の値を書くと、illegal_instruction 例外が通知される。ただしこの理由による illegal_instruction と illegal_action 例外では、illegal_action 例外が優先して通知される。

Note – TL = 0 で WRTXAR を実行すると、XAR の値によらず illegal_instruction 例外が通知される。

XAR.v = 0 の値を書くとき、また、WRTXAR で TXAR.v = 0 の値を書くとき、関連するフィールドの値は書かれる値によらず不定になる。すなわち、

- XAR.f_v = 0 の値を書くときは、XAR.f_urs1, XAR.f_urs2, XAR.f_urs3, XAR.f_urd, XAR.f_simd の値は何を書くかによらず不定となる。
- XAR.s_v = 0 の値を書くときは、XAR.s_urs1, XAR.s_urs2, XAR.s_urs3, XAR.s_urd, XAR.s_simd の値は何を書くかによらず不定となる。
- TXAR.f_v = 0 の値を書くときは、TXAR.f_urs1, TXAR.f_urs2, TXAR.f_urs3, TXAR.f_urd, TXAR.f_simd の値は何を書くかによらず不定となる。
- TXAR.s_v = 0 の値を書くときは、TXAR.s_urs1, TXAR.s_urs2, TXAR.s_urs3, TXAR.s_urd, TXAR.s_simd の値は何を書くかによらず不定となる。

となる。

Implementation Note – XAR.v = 0 な値を書くときは関連フィールドを強制的に 0 クリアするよう実装してもよい。

Exceptions

- software_initiated_reset (rd = 15, rs1 = 0, and i = 1 only)
- privileged_opcode (WRDCR, WRSOFTINT_SET, WRSOFTINT_CLR, WRSOFTINT, WRTICK_CMPR, WRESTICK, WRESTICK_CMPR, and WRTXAR)
- illegal_instruction (WRASR with rd = 1, 4, 5, 7–14, 26–28;
  WRASR with rd = 15 and rs1 ≠ 0 or i ≠ 1,
  WRTXAR with TL = 0;
  WRXAR with reserved fields to nonzero)
- fp_disabled (WRGSR with PSTATE.PEF = 0 or FPRS.FEF = 0)
- illegal_action (XAR.v = 1 and (XAR.simd = 1 or
  XAR.urs1 > 1 or
  (i = 0 and XAR.urs2 > 1) or
  (i = 1 and XAR.urs2 ≠ 0) or
  XAR.urs3 ≠ 0 or
  XAR.urd ≠ 0))
- privileged_action (WRPIC with PSTATE.PRIV = 0 and PCR.PRIV = 1,
  WRPCR with PSTATE.PRIV = 0 and PCR.PRIV = 1;
  WRPCR to modify PCR.PRIV
  with PSTATE.PRIV = 0 and PCR.PRIV = 0)
A.71 Deprecated Instructions

JPS1 Commonality の Appendix A.71 で定義される非推奨命令は、過去のアーキテクチャとの互換性のために提供される。新しいソフトウェアでこれらの命令を使うことは推奨されない。

A.71.10 Store Barrier

SPARC64 IXfx では STBAR は NOP として動作する。これは SPARC64 IXfx ハードウェアのメモリモデルが、すべてのメモリアクセスにこの命令を挟んでいるのと等価なためである。

Exceptions illeagal_action (XAR.v = 1)
## A.72 Floating-Point Conditional Compare to Register

<table>
<thead>
<tr>
<th>HPC-ACE Ext.</th>
<th>オペコード</th>
<th>op3</th>
<th>opf</th>
<th>处理</th>
<th>レジスタ比較</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓ ✓</td>
<td>FCMPEQd</td>
<td>11 0110 1 0110 0000</td>
<td></td>
<td>位精度レジスタ比較 (EQ)</td>
<td>f[rs1] = f[rs2]</td>
</tr>
<tr>
<td>✓ ✓</td>
<td>FCMPEQEd</td>
<td>11 0110 1 0110 0010</td>
<td></td>
<td>位精度レジスタ比較 (EQ)、比較不能なら例外生成</td>
<td>f[rs1] = f[rs2]</td>
</tr>
<tr>
<td>✓ ✓</td>
<td>FCMPLEEd</td>
<td>11 0110 1 0110 0100</td>
<td></td>
<td>位精度レジスタ比較 (LE)、比較不能なら例外生成</td>
<td>f[rs1] ≤ f[rs2]</td>
</tr>
<tr>
<td>✓ ✓</td>
<td>CMPLTEEd</td>
<td>11 0110 1 0110 0110</td>
<td></td>
<td>位精度レジスタ比較 (LT)、比較不能なら例外生成</td>
<td>f[rs1] &lt; f[rs2]</td>
</tr>
<tr>
<td>✓ ✓</td>
<td>FCMPNEd</td>
<td>11 0110 1 0110 1000</td>
<td></td>
<td>位精度レジスタ比較 (NE)</td>
<td>f[rs1] ≠ f[rs2]</td>
</tr>
<tr>
<td>✓ ✓</td>
<td>FCMPNEEd</td>
<td>11 0110 1 0110 1010</td>
<td></td>
<td>位精度レジスタ比較 (NE)、比較不能なら例外生成</td>
<td>f[rs1] ≠ f[rs2]</td>
</tr>
<tr>
<td>✓ ✓</td>
<td>FCMPGTEd</td>
<td>11 0110 1 0110 1100</td>
<td></td>
<td>位精度レジスタ比較 (GT)、比較不能なら例外生成</td>
<td>f[rs1] &gt; f[rs2]</td>
</tr>
<tr>
<td>✓ ✓</td>
<td>FCMPGTEd</td>
<td>11 0110 1 0110 1110</td>
<td></td>
<td>位精度レジスタ比較 (GE)、比較不能なら例外生成</td>
<td>f[rs1] ≥ f[rs2]</td>
</tr>
<tr>
<td>✓ ✓</td>
<td>FCMPGQs</td>
<td>11 0110 1 0110 0001</td>
<td></td>
<td>単精度レジスタ比較 (EQ)</td>
<td>f[rs1] = f[rs2]</td>
</tr>
<tr>
<td>✓ ✓</td>
<td>FCMPQEs</td>
<td>11 0110 1 0110 0011</td>
<td></td>
<td>単精度レジスタ比較 (EQ)、比較不能なら例外生成</td>
<td>f[rs1] = f[rs2]</td>
</tr>
<tr>
<td>✓ ✓</td>
<td>FCMPLEEs</td>
<td>11 0110 1 0110 0101</td>
<td></td>
<td>単精度レジスタ比較 (LE)、比較不能なら例外生成</td>
<td>f[rs1] ≤ f[rs2]</td>
</tr>
<tr>
<td>✓ ✓</td>
<td>CMPLTEEs</td>
<td>11 0110 1 0110 0111</td>
<td></td>
<td>単精度レジスタ比較 (LT)、比較不能なら例外生成</td>
<td>f[rs1] &lt; f[rs2]</td>
</tr>
<tr>
<td>✓ ✓</td>
<td>FCMPNEEs</td>
<td>11 0110 1 0110 1001</td>
<td></td>
<td>単精度レジスタ比較 (NE)</td>
<td>f[rs1] ≠ f[rs2]</td>
</tr>
<tr>
<td>✓ ✓</td>
<td>FCMPNEEs</td>
<td>11 0110 1 0110 1011</td>
<td></td>
<td>単精度レジスタ比較 (NE)、比較不能なら例外生成</td>
<td>f[rs1] ≠ f[rs2]</td>
</tr>
<tr>
<td>✓ ✓</td>
<td>FCMPGTEEs</td>
<td>11 0110 1 0110 1101</td>
<td></td>
<td>単精度レジスタ比較 (GT)、比較不能なら例外生成</td>
<td>f[rs1] &gt; f[rs2]</td>
</tr>
<tr>
<td>✓ ✓</td>
<td>FCMPGGEEs</td>
<td>11 0110 1 0110 1111</td>
<td></td>
<td>単精度レジスタ比較 (GE)、比較不能なら例外生成</td>
<td>f[rs1] ≥ f[rs2]</td>
</tr>
</tbody>
</table>

### Format (3)

<table>
<thead>
<tr>
<th>10</th>
<th>rd</th>
<th>op3</th>
<th>11 0110</th>
<th>rs1</th>
<th>14</th>
<th>13</th>
<th>5</th>
<th>4</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>25 24</td>
<td>19 18</td>
<td>14</td>
<td>13</td>
<td>5</td>
<td>4</td>
<td>0</td>
</tr>
</tbody>
</table>

SPARC™ IX fx Extensions Ver 12, 2 Dec. 2013
これらの命令は、rs1 で指定された浮動小数点レジスタの値と rs2 で指定された浮動小数点レジスタの値を比較し、条件が成立しているれば all1, 不成立なら all0 を、rd で指定された浮動小数点レジスタに格納する。

入力値のどちらかが SNaN または QNaN のときの例外と結果は以下のようになる。exception の列は fp_exception_ieee_754 例外通知時 FSR.cexc にセットされる値を、rd の列は例外が通知されない場合に結果として格納される値を示す。

<table>
<thead>
<tr>
<th>命令</th>
<th>SNan</th>
<th>QNaN</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCMPGTE{s,d}, FCMPLTE{s,d}, FCMPGEE{s,d}, FCMPLEE{s,d}</td>
<td>NV all0</td>
<td>NV all0</td>
</tr>
<tr>
<td>FCMPGE{s,d}</td>
<td>NV all0</td>
<td>NV all0</td>
</tr>
<tr>
<td>FCMPNEE{s,d}</td>
<td>NV all1</td>
<td>NV all1</td>
</tr>
<tr>
<td>FCMPEQ{s,d}</td>
<td>NV all0</td>
<td>—</td>
</tr>
<tr>
<td>FCMPNE{s,d}</td>
<td>NV all1</td>
<td>—</td>
</tr>
</tbody>
</table>

Programming Note – この命令は、FSELMOV{s,d}, STFR, STDFR, VIS の論理演算命令と組み合わせて使うことを想定している。

Exceptions

- fp_disabled
- illegal_action (XAR.v = 1 and XAR.urs3 ≠ 0; XAR.v = 1 and XAR.simd = 1 and (XAR.urs1<2> ≠ 0 or XAR.urs2<2> ≠ 0 or XAR.urd<2> ≠ 0))
- fp_exception_ieee_754 (NV if unordered)
A.73 Floating-Point Minimum and Maximum

**Format (3)**

<table>
<thead>
<tr>
<th>HPC-ACE Ext.</th>
<th>Regs.</th>
<th>SIMD</th>
<th>Opcode</th>
<th>op3</th>
<th>opf</th>
<th>Processing</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓ ✓</td>
<td>FMAXd</td>
<td>11</td>
<td>0110 1</td>
<td>0111 0000</td>
<td>倍精度最大值</td>
<td></td>
</tr>
<tr>
<td>✓ ✓</td>
<td>FMAXs</td>
<td>11</td>
<td>0110 1</td>
<td>0111 0001</td>
<td>単精度最大値</td>
<td></td>
</tr>
<tr>
<td>✓ ✓</td>
<td>FMINd</td>
<td>11</td>
<td>0110 1</td>
<td>0111 0010</td>
<td>倍精度最小値</td>
<td></td>
</tr>
<tr>
<td>✓ ✓</td>
<td>FMINs</td>
<td>11</td>
<td>0110 1</td>
<td>0111 0011</td>
<td>単精度最小値</td>
<td></td>
</tr>
</tbody>
</table>

**Description**

FMAX\{s,d\} は、rs1 で指定された浮動小数点レジスタの値と rs2 で指定された浮動小数点レジスタの値を比較し、\( f[rs1] > f[rs2] \) なら \( f[rs1] \) を、そうでなければ \( f[rs2] \) を、rd で指定された浮動小数点レジスタに格納する。

FMIN\{s,d\} は、rs1 で指定された浮動小数点レジスタの値と rs2 で指定された浮動小数点レジスタの値を比較し、\( f[rs1] < f[rs2] \) なら \( f[rs1] \) を、そうでなければ \( f[rs2] \) を、rd で指定された浮動小数点レジスタに格納する。

FMIN, FMAX はゼロの符号は無視する。\( f[rs1], f[rs2] \) が +0, -0 または -0, +0 のとき は、\( f[rs2] \) の値が出力される。

---

**アセンプリ言語表記**

| fmax\{s,d\} | fregrs1, fregrs2, fregrd |
| fmin\{s,d\} | fregrs1, fregrs2, fregrd |

**HPC-ACE Ext.**

<table>
<thead>
<tr>
<th>10</th>
<th>rd</th>
<th>op3</th>
<th>11 0110</th>
<th>rs1</th>
<th>14 13</th>
<th>opf</th>
<th>5 4 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>25 24</td>
<td>19</td>
<td>18</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**Description**

FMAX\{s,d\} は、rs1 で指定された浮動小数点レジスタの値と rs2 で指定された浮動小数点レジスタの値を比較し、\( f[rs1] > f[rs2] \) なら \( f[rs1] \) を、そうでなければ \( f[rs2] \) を、rd で指定された浮動小数点レジスタに格納する。

FMIN\{s,d\} は、rs1 で指定された浮動小数点レジスタの値と rs2 で指定された浮動小数点レジスタの値を比較し、\( f[rs1] < f[rs2] \) なら \( f[rs1] \) を、そうでなければ \( f[rs2] \) を、rd で指定された浮動小数点レジスタに格納する。

FMIN, FMAX はゼロの符号は無視する。\( f[rs1], f[rs2] \) が +0, -0 または -0, +0 のとき は、\( f[rs2] \) の値が出力される。
入力値の一方がQNaN、もう一方がQNaN, SNaN以外の数のときは、rdにはQNaN以外の数が出力される。他の命令と異なり、QNaNは伝播しない。入力値の一方または両方がSNaN、または入力値の両方がQNaNのときは、rdにはJPS1 CommonalityのTABLE B-1で定義された値が格納される。また、入力値の少なくとも一方にQNaN, SNaNが含まれる場合、fp_exception_ieee_754例外を検出する。

### TABLE A-9 FMIN, FMAXの入力と出力

<table>
<thead>
<tr>
<th>rs1</th>
<th>rs2</th>
<th>rd</th>
<th>例外</th>
</tr>
</thead>
<tbody>
<tr>
<td>NaN 以外</td>
<td>NaN 以外</td>
<td>min(rs1,rs2), or max(rs1,rs2)</td>
<td>—</td>
</tr>
<tr>
<td>NaN 以外</td>
<td>QNaN</td>
<td>rs1</td>
<td>NV</td>
</tr>
<tr>
<td>NaN 以外</td>
<td>SNaN</td>
<td>QNaN2</td>
<td>NV</td>
</tr>
<tr>
<td>QNaN</td>
<td>NaN 以外</td>
<td>rs2</td>
<td>NV</td>
</tr>
<tr>
<td>QNaN</td>
<td>QNaN</td>
<td>rs2 (QNaN)</td>
<td>NV</td>
</tr>
<tr>
<td>QNaN</td>
<td>SNaN</td>
<td>QNaN2</td>
<td>NV</td>
</tr>
<tr>
<td>SNaN</td>
<td>NaN 以外</td>
<td>QNaN1</td>
<td>NV</td>
</tr>
<tr>
<td>SNaN</td>
<td>QNaN</td>
<td>QNaN1</td>
<td>NV</td>
</tr>
<tr>
<td>SNaN</td>
<td>SNaN</td>
<td>QNaN2</td>
<td>NV</td>
</tr>
</tbody>
</table>

### Exceptions

**fp_disabled illegal_action**

(XAR.v = 1 and XAR.urs3 ≠ 0;
XAR.v = 1 and XAR.simd = 1 and
(XAR.urs1<2> ≠ 0 or XAR.urs2<2> ≠ 0 or XAR.urd<2> ≠ 0))

**fp_exception_ieee_754** (NV if unordered)
A.74 Floating-Point Reciprocal Approximation

HPC-ACE Ext.

<table>
<thead>
<tr>
<th>Regs.</th>
<th>SIMD</th>
<th>オペコード</th>
<th>op3</th>
<th>opf</th>
<th>処理</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓ ✓</td>
<td>FRCPA</td>
<td>11 0110</td>
<td>1</td>
<td>0111</td>
<td>0100</td>
</tr>
<tr>
<td>✓ ✓</td>
<td>FRCPA</td>
<td>11 0110</td>
<td>1</td>
<td>0111</td>
<td>0101</td>
</tr>
<tr>
<td>✓ ✓</td>
<td>FRSQRTA</td>
<td>11 0110</td>
<td>1</td>
<td>0111</td>
<td>0110</td>
</tr>
<tr>
<td>✓ ✓</td>
<td>FRsqrtA</td>
<td>11 0110</td>
<td>1</td>
<td>0111</td>
<td>0111</td>
</tr>
</tbody>
</table>

Format (3)

<table>
<thead>
<tr>
<th>10</th>
<th>rd</th>
<th>11 0110</th>
<th>0000</th>
<th>opf</th>
<th>rs2</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>25</td>
<td>24</td>
<td>19</td>
</tr>
</tbody>
</table>

アセンブリ言語表記

- \texttt{frcpa\{s,d\}} \ \texttt{fregr2,fregd}
- \texttt{frsqrt\{s,d\}} \ \texttt{fregr2,fregd}

Description

\begin{equation}
\frac{|frcpa(x) - 1/x|}{1/x} \leq \frac{1}{256}
\end{equation}

となる。
FRCPA{s,d} の演算結果と例外条件を TABLE A-10 に示す。結果の欄の上段が例外、下段が例外が通知されないときの値である。fp_exception_ieee_754 例外の要因が複数あるものの優先順位は本書および JPS1 Commonality の Appendix B を参照。

TABLE A-10  FRCPA{s,d} の演算結果

<table>
<thead>
<tr>
<th>rs2</th>
<th>例外と演算結果</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FSR.NS = 0</td>
</tr>
<tr>
<td>+∞</td>
<td>—</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>+N ( N ≥ 2^{126} for single, N ≥ 2^{1022} for double)</td>
<td>UF</td>
</tr>
<tr>
<td></td>
<td>+1/N の近似値 (非正規化数)¹</td>
</tr>
<tr>
<td>+N ( +Nmin ≤ N &lt; 2^{126} for single, +Nmin ≤ N &lt; 2^{1022} for double)</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>+1/N の近似値</td>
</tr>
<tr>
<td>+D</td>
<td>unfinished_FPop</td>
</tr>
<tr>
<td></td>
<td>—</td>
</tr>
<tr>
<td>+0</td>
<td>DZ</td>
</tr>
<tr>
<td></td>
<td>+∞</td>
</tr>
<tr>
<td>-0</td>
<td>DZ</td>
</tr>
<tr>
<td></td>
<td>—</td>
</tr>
<tr>
<td>-D</td>
<td>unfinished_FPop</td>
</tr>
<tr>
<td></td>
<td>—</td>
</tr>
<tr>
<td>-N ( -Nmin ≤ N &lt; 2^{126} for single, -Nmin ≤ N &lt; 2^{1022} for double)</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>-1/N の近似値</td>
</tr>
<tr>
<td>-N ( N ≥ 2^{126} for single, N ≥ 2^{1022} for double)</td>
<td>UF</td>
</tr>
<tr>
<td></td>
<td>-1/N の近似値 (非正規化数)¹</td>
</tr>
<tr>
<td>-∞</td>
<td>—</td>
</tr>
<tr>
<td>-0</td>
<td>-0</td>
</tr>
<tr>
<td>SNaN</td>
<td>NV</td>
</tr>
<tr>
<td>QNaN</td>
<td>—</td>
</tr>
<tr>
<td>rs2</td>
<td>rs2</td>
</tr>
</tbody>
</table>

¹結果が非正規化数のとき、精度は 1/256 より大きくなり得る。

N 正の正規化数（ゼロ、NaN、無限大を除く）
D 正の非正規化数
FRSQRTA{s, d} は、rs2 で指定された浮動小数点レジスタの値の平方根の逆数近似値を求め、rd で指定された浮動小数点レジスタに格納する。得られる結果は近似値だが、FSR.RD の影響を受けない。結果が正規化数のとき、近似値の精度は 1/256 未満、つまり

\[
\left| \frac{frsqrta(x) - 1/\sqrt{x}}{1/\sqrt{x}} \right| < \frac{1}{256}
\]

となる。

FRSQRTA{s, d} の演算結果と例外条件を TABLE A-11 に示す。結果の欄の上段が例外、下段が例外が通知されないときの値である。fp_exception_ieee_754 例外の要因が複数あるものの優先順位は本書および JPS1 Commonality の Appendix B を参照。

### TABLE A-11 FRSQRTA{s,d} の演算結果

<table>
<thead>
<tr>
<th>rs2</th>
<th>FSR.NS = 0</th>
<th>FSR.NS = 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>+∞</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>+N</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>1/{\sqrt{3}}</td>
<td>1/{\sqrt{3}}</td>
</tr>
<tr>
<td>+D</td>
<td>unfinished_FPop</td>
<td>DZ</td>
</tr>
<tr>
<td></td>
<td>—</td>
<td>+0</td>
</tr>
<tr>
<td>+0</td>
<td>DZ</td>
<td>DZ</td>
</tr>
<tr>
<td></td>
<td>+0</td>
<td>+0</td>
</tr>
<tr>
<td>-0</td>
<td>DZ</td>
<td>DZ</td>
</tr>
<tr>
<td></td>
<td>+0</td>
<td>+0</td>
</tr>
<tr>
<td>-D</td>
<td>NV</td>
<td>NV</td>
</tr>
<tr>
<td></td>
<td>dNaN</td>
<td>dNaN</td>
</tr>
<tr>
<td>-N</td>
<td>NV</td>
<td>NV</td>
</tr>
<tr>
<td></td>
<td>dNaN</td>
<td>dNaN</td>
</tr>
<tr>
<td>-∞</td>
<td>NV</td>
<td>NV</td>
</tr>
<tr>
<td></td>
<td>dNaN</td>
<td>dNaN</td>
</tr>
<tr>
<td>SNaN</td>
<td>NV</td>
<td>NV</td>
</tr>
<tr>
<td></td>
<td>QSNaN2</td>
<td>QSNaN2</td>
</tr>
</tbody>
</table>

Nmin
正の正規化数の最小値
dNaN
符号は 0。指数部と仮数部が全ビット 1 の QNaN
QSNaN2
JPS1 Commonality の TABLE B-1 参照。
### Exceptions

- **illegal_instruction** (instruction<18:14> ≠ 0)
- **fp_disabled**
- **illegal_action** (XAR.v = 1 and (XAR.urs1 ≠ 0 or XAR.urs3 ≠ 0); XAR.v = 1 and XAR.simd = 1 and (XAR.urs2<2> ≠ 0 or XAR.urd<2> ≠ 0))
- **fp_exception_ieee_754** (NV, DZ, UF, NX for FRCPA{s, d}; NV, DZ for FRSQRTA{s, d})
- **fp_exception_other** (ftt = unfinished_FPop)

### TABLE A-11  FRSQRTA{s,d} の演算結果

<table>
<thead>
<tr>
<th>rs2</th>
<th>FSR.NS = 0</th>
<th>FSR.NS = 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>QNaN</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>rs2</td>
<td>rs2</td>
<td></td>
</tr>
</tbody>
</table>
A.75 Move Selected Floating-Point Register on Floating-Point Register's Condition

Description

FSELMOV{s, d} は、rs3 で指定された浮動小数点レジスタの値の最上位ビットにより rs1 か rs2 を選択し、指定された浮動小数点レジスタの値を rd で指定された浮動小数点レジスタに格納する。rs3 で指定された浮動小数点レジスタの bit<63> が 1 なら rs1 を、0 なら rs2 を選択する。

Exceptions

- fp_disabled
- illegal_action (XAR.v = 1 and XAR.simd = 1 and
  (XAR.urs1<2> ≠ 0 or XAR.urs2<2> ≠ 0 or
   XAR.urs3<2> ≠ 0 or XAR.urd<2> ≠ 0))
A.76 Floating-Point Trigonometric Functions

<table>
<thead>
<tr>
<th>HPC-ACE Ext.</th>
<th>Regs.</th>
<th>SIMD</th>
<th>オペコード</th>
<th>op3</th>
<th>opf</th>
<th>处理</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓ ✓</td>
<td></td>
<td>FTRIMADDd</td>
<td>11 0111</td>
<td>—</td>
<td></td>
<td>三角関数補助演算</td>
</tr>
<tr>
<td>✓ ✓</td>
<td></td>
<td>FTRISMLd</td>
<td>11 0110 1 0111 1010</td>
<td></td>
<td></td>
<td>FTRIMADDdの初期価出</td>
</tr>
<tr>
<td>✓ ✓</td>
<td></td>
<td>FTRISSELd</td>
<td>11 0110 1 0111 1000</td>
<td></td>
<td></td>
<td>係数テーブルの選択と最終段の係数算出</td>
</tr>
</tbody>
</table>

Format (5 and 3)

| 10 | rd | 11 0111 | op3 | rs1 | index | var 10 | size 00 | 31 24 02530 29 19 18 14 13 5 4 |
|----|----|---------|-----|-----|-------|--------|--------|                              |
| 25 24 | 19 18 | 14 13 | 9 8 | 7 6 | 5 4 | 0 | |

アセンブリ言語表記

- FTRIMADDd: fregs1, fregs2, index, fregd
- FTRISMLd: fregs1, fregs2, fregd
- FTRISSELd: fregs1, fregs2, fregd

処理 計算

<table>
<thead>
<tr>
<th>处理</th>
<th>計算</th>
</tr>
</thead>
<tbody>
<tr>
<td>FTRIMADDd</td>
<td>rd ← rs1×abs(rs2)+T[rs2&lt;63][index]</td>
</tr>
<tr>
<td>FTRISMLd</td>
<td>rd ← (rs2&lt;0&gt; &lt;&lt; 63) ^ (rs1 × rs1)</td>
</tr>
<tr>
<td>FTRISSELd</td>
<td>rd ← (rs2&lt;1&gt; &lt;&lt; 63) ^ (rs2&lt;0&gt; ? 1.0 : rs1)</td>
</tr>
</tbody>
</table>

Description

これらの3命令はsin(x)を求めるための級数計算の補助命令である。FTRIMADDdは、
-π/4 < x ≤ π/4の範囲でテーラー級数の級数部分の計算を行い、FTRISMLdと
FTRISSELdは、任意のxについてsin(x)を求めるためFTRIMADDdの前処理を行う。こ
れらの命令は倍精度命令のみが定義されている。FIGURE A-1に、これら3命令が行う
計算式を示す。
FTRIMADDd は、rs1 で指定される倍精度浮動小数点レジスタの値と rs2 で指定される倍精度浮動小数点レジスタの値の絶対値を乗じ、さらに演算器内にあるテーブルから index で指定される倍精度数を取り出して加算し、結果を rd で指定される倍精度浮動小数点レジスタに格納する。FTRIMADDd は \( \sin(x), \cos(x) \) の級数部分の計算を行う。

FTRISMULd は、rs1 で指定される倍精度浮動小数点レジスタの値を二乗し、rs2 で指定される倍精度浮動小数点レジスタの値の最下位ビットを符号とする結果を、rd で指定される倍精度浮動小数点レジスタに格納する。FTRISMULd は FTRIMADDd の初期値を求めるために使われる。

FTRISSELd は、rs1 で指定される倍精度浮動小数点レジスタの値か 1.0 を、rs2 で指定される倍精度浮動小数点レジスタの値の最下位ビットを符号として排他に論理和 (exclusive or) をとる、rd で指定される倍精度浮動小数点レジスタに格納する。FTRISSELd は級数部分の計算結果に乗る最後の項を求めるために使われる。

FTRIMADDd は \( \sin(x), \cos(x) \) の級数部分の計算を行う。初期値として \( f[rs1] \) にゼロ、\( f[rs2] \) に \(-\pi/4 < x \leq \pi/4\) の \( x^2 \) をセットし、FTRIMADDd を 8 回実行すると、級数部分が倍精度浮動小数点数として十分な精度で計算できる。FIGURE A-1 の式からわかる通り、\( \sin(x) \) と \( \cos(x) \) の級数部分は係数が異なるので、FTRIMADDd は、rs2 の符号をどちらの係数を使うかの識別に利用する。

- \( f[rs2]<63> = 0 \) のとき、\( \sin(x) \) の係数テーブルを使う
- \( f[rs2]<63> = 1 \) のとき、\( \cos(x) \) の係数テーブルを使う

\[
\sin x \equiv \frac{1}{3!} x^3 + \frac{1}{5!} x^5 - \frac{1}{7!} x^7 + \frac{1}{9!} x^9 - \frac{1}{11!} x^{11} + \frac{1}{13!} x^{13} - \frac{1}{15!} x^{15}
\]

\[
= x \left( \frac{1}{3!} x^2 + \frac{1}{5!} x^4 - \frac{1}{7!} x^6 + \frac{1}{9!} x^8 - \frac{1}{11!} x^{10} + \frac{1}{13!} x^{12} - \frac{1}{15!} x^{14} \right)
\]

\[
= x \left( \left( \frac{0 \cdot x^2 - \frac{1}{15!}}{1} \right) x^2 + \frac{1}{13!} x^2 - \frac{1}{11!} x^2 + \frac{1}{9!} x^2 - \frac{1}{7!} x^2 + \frac{1}{5!} x^2 - \frac{1}{3!} x^2 \right)
\]

FTRISSELd

\[
\cos x \equiv 1 - \frac{1}{2!} x^2 + \frac{1}{4!} x^4 - \frac{1}{6!} x^6 + \frac{1}{8!} x^8 - \frac{1}{10!} x^{10} + \frac{1}{12!} x^{12} - \frac{1}{14!} x^{14}
\]

\[
= 1 \cdot \left( \left( \frac{0 \cdot x^2 - \frac{1}{14!}}{1} \right) x^2 + \frac{1}{12!} x^2 - \frac{1}{10!} x^2 + \frac{1}{8!} x^2 - \frac{1}{6!} x^2 + \frac{1}{4!} x^2 - \frac{1}{2!} x^2 \right)
\]

FTRISSELd

**FIGURE A-1** SPARC64 IIXfx の三角関数補助演算
FTRIMADDd は下の例の使い方を想定しており、この使い方のときに精度誤差が小さくなるような係数を採用しているため、係数が数学的に正しいものとは異なっている。

TABLE A-12, TABLE A-13 に SPARC64 IXfx の FTRIMADDd の係数テーブルの内容を示す。

**TABLE A-12 sin(x) (f[rs2] <63> = 0) の係数テーブル**

<table>
<thead>
<tr>
<th>Index</th>
<th>16進表記</th>
<th>10進表記</th>
<th>数学的に正しい係数</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3ff0 0000 0000 0000_16</td>
<td>1.0</td>
<td>= 1/1!</td>
</tr>
<tr>
<td>1</td>
<td>bfc5 5555 5555 5543_16</td>
<td>-0.1666666666666661</td>
<td>&gt; -1/3!</td>
</tr>
<tr>
<td>2</td>
<td>3f81 1111 1110 f30c_16</td>
<td>0.83333333333320002e-02</td>
<td>&lt; 1/5!</td>
</tr>
<tr>
<td>3</td>
<td>bf2a 01a0 19b9 2fc6_16</td>
<td>-0.1984126982840213e-03</td>
<td>&gt; -1/7!</td>
</tr>
<tr>
<td>4</td>
<td>3ec7 1de3 51f3 d22b_16</td>
<td>0.2755731329901505e-05</td>
<td>&lt; 1/9!</td>
</tr>
<tr>
<td>5</td>
<td>be5a e5e2 b60f 7b91_16</td>
<td>-0.2505070584637887e-07</td>
<td>&gt; -1/11!</td>
</tr>
<tr>
<td>6</td>
<td>3de5 d840 8868 552f_16</td>
<td>0.1589413637195215e-09</td>
<td>&lt; 1/13!</td>
</tr>
<tr>
<td>7</td>
<td>0000 0000 0000 0000_16</td>
<td>0</td>
<td>&gt; -1/15!</td>
</tr>
</tbody>
</table>

**TABLE A-13 cos(x) (f[rs2] <63> = 1) の係数テーブル**

<table>
<thead>
<tr>
<th>Index</th>
<th>16進表記</th>
<th>10進表記</th>
<th>数学的に正しい係数</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3ff0 0000 0000 0000_16</td>
<td>1.0</td>
<td>= 1/0!</td>
</tr>
<tr>
<td>1</td>
<td>bfe0 0000 0000 0000_16</td>
<td>-0.5000000000000000</td>
<td>= -1/2!</td>
</tr>
<tr>
<td>2</td>
<td>3fa5 5555 5555 5536_16</td>
<td>0.4166666666666645e-01</td>
<td>&lt; 1/4!</td>
</tr>
<tr>
<td>3</td>
<td>bf56 c16c 16c1 3a0b_16</td>
<td>-0.138888888888611e-02</td>
<td>&lt; -1/6!</td>
</tr>
<tr>
<td>4</td>
<td>3efa 01a0 19b1 ee8d_16</td>
<td>0.2480158728388683e-04</td>
<td>&lt; 1/8!</td>
</tr>
<tr>
<td>5</td>
<td>be92 7e4f 7282 f468_16</td>
<td>-0.2755731309913950e-06</td>
<td>&lt; -1/10!</td>
</tr>
<tr>
<td>6</td>
<td>3e21 ee9e d264 1b13_16</td>
<td>0.2087558253975872e-08</td>
<td>&lt; 1/12!</td>
</tr>
<tr>
<td>7</td>
<td>bda8 f763 80fb b401_16</td>
<td>-0.1135338700720054e-10</td>
<td>&gt; -1/14!</td>
</tr>
</tbody>
</table>

FTRISMULd は FTRIMADDd の初期値を計算する。f[rs1] に x, f[rs2] に FIGURE A-2 の Q を与えると、x² を計算し、符号のところに係数テーブル選択するための情報を付加した値を返す。Q は浮動小数点数ではなく整数で与える。

f[rs2] <63:1> は使われない。f[rs2] が NaN であっても例外を検出しない。
FTRISSELd は FTRIMADDd による級数計算の後、最後に乗ずる値を計算する。
f[rs1] に x, f[rs2] に FIGURE A-2 の Q を与えると、x または 1.0 のどちらかに適切な符号をつけた値を返す。Q は浮動小数点数ではなく整数で与える。f[rs2]<63:2> は使われない。f[rs2] が NaN であっても例外を検出しない。

\[
q: (2q-1) \cdot \frac{\pi}{4} < x \leq (2q+1) \cdot \frac{\pi}{4} \
Q: q \mod 4 \
R: x - q \cdot \frac{\pi}{2} \left( \frac{\pi}{4} < R \leq \frac{\pi}{4} \right)
\]

\[
\begin{align*}
Q = 1 & \quad \sin(x) = \cos(R) \\
Q = 2 & \quad \sin(x) = -\sin(R) \\
Q = 3 & \quad \sin(x) = -\cos(R)
\end{align*}
\]

例：\( \sin(x) \) を求める
/*
  * 入力値：x
  * q: (2q-1)*π/4 < x <= (2q+1)*π/4 なる q
  * Q: q mod 4
  * R: x - q * π/2
  */

ftrismuld R, Q, M
ftrisseld R, Q, N

/*
  * M ← R^2[63]=table_type, R^2[62:0]=R^2
  * (R^2 は必ず正となるので sign ビット (bit63) は常に 0 となる。
  * この sign ビットを ftrimaddd の table_type として使う)
  * N ← 最後に掛ける値 (1.0 or R) * sign
*/
* S ← 0
*/

ftrimaddd S, M, 7, S
ftrimaddd S, M, 6, S
ftrimaddd S, M, 5, S
ftrimaddd S, M, 4, S
ftrimaddd S, M, 3, S
ftrimaddd S, M, 2, S
ftrimaddd S, M, 1, S
ftrimaddd S, M, 0, S
fmuld S, N, S

/*
* S ← 結果
*/

Exceptions
illegal_instruction (FTRIMADdd with index > 7)
fp_disabled
illegal_action (XAR.v = 1 and XAR.urs3 ≠ 0;
    XAR.v = 1 and XAR.simd = 1 and
    (XAR.urs1<2> ≠ 0 or XAR.urs2<2> ≠ 0 or XAR.urd<2> ≠ 0))
fp_exception_ieee_754 (FTRIMADdd with NV, NX, OF, UF;
    FTRISMD with NX, OF, UF;
    FTRISMD with NV (rs1 only))
fp_exception_other (FTRIMADdd, FTRISMD with ftt = unfinished_FPop)
### A.77 Store Floating-Point Register on Register Condition

<table>
<thead>
<tr>
<th>HPC-ACE Ext.</th>
<th>Regs.</th>
<th>SIMD</th>
<th>op3</th>
<th>rd</th>
<th>urd</th>
<th>处理</th>
</tr>
</thead>
<tbody>
<tr>
<td>STFR</td>
<td>10 1100</td>
<td>0-31</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>STDFR</td>
<td>10 1111</td>
<td>0-7</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

† JPS1 CommonalityのSection 5.1.4で定義される浮動小数点レジスタエンコードに従う。

XAR.v = 0 のとき

**Format (3)**

<table>
<thead>
<tr>
<th></th>
<th>rd</th>
<th>op3</th>
<th>rs1</th>
<th>i = 1</th>
<th>simm8</th>
<th>rs2</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>25</td>
<td>24</td>
<td>19</td>
<td>18</td>
</tr>
<tr>
<td>14</td>
<td>13</td>
<td>12</td>
<td>5</td>
<td>4</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

アセンブリ言語表記

- STFR: `stfr fregrd, fregrs2, [address]`
- STDFR: `stdfr fregrd, fregrs2, [address]`

**Description**

STFR は、f [rs2] の最上位ビットが 1 のとき、f [rd] で指示される単精度浮動小数点レジスタの内容を、4 バイト境界の 4 バイト領域に書き込む。

STDFR は、f [rs2] の最上位ビットが 1 のとき、f [rd] で指示される倍精度浮動小数点レジスタの内容を、8 バイト境界の 8 バイト領域に書き込む。

書き込みアドレスは "r[rs1] + sign_ext(simm8 << 2)" で計算される。

STFR は、4 バイト境界にないアドレスにアクセスすると mem_address_not_aligned例外を通知する。

STDFR は、8 バイト境界にないアドレスにアクセスすると mem_address_not_aligned例外を通知する。

SIMD でない STDFR は、4 バイト境界だが 8 バイト境界にないアドレスにアクセスすると STDF_mem_address_not_aligned例外を通知する。
浮動小数点演算器が使えないとき（FPRS.FEF, PSTATE.PEF の設定による）、STFR, STDFR は fp_disabled を通知する。

STFR, STDFR のウォッチポイント検出は、実際にストアをする / しないに関係なく、アドレスがマッチしていれば例外を通知する。また、MMU が有効ならば、実際にストアをする / しないに関係らず TLB が検索される。変換エントリが存在しなければ fast_data_access_MMU_miss が通知され、変換エントリが存在すればその設定が検査され、不正なアクセスには data_access_exception, data_access_error, fast_data_access_protection が通知される。

Implementation Note – f[r[rs2]] の最上位ビットが 0 で指定されたアドレスがノンキャッシュプル領域のときは、マスク 0 のバストランザクションを生成する。

Programming Note – 単精度浮動小数点ストア命令 STFR のアドレス指定(r[rs1, rs2]) に HPC-ACE 拡張整数レジスタ xg[0] - xg[31] を使う場合、ストア元のレジスタは倍精度レジスタとなる。これは XAR.v=1 のときの rd のデコード定義から来る制約であり (page 21), rs1, rs2 に拡張整数レジスタ, rd に SPARC V9 単精度レジスタ (奇数番号レジスタ) を指定する方法はない。

SIMD 拡張

SPARC64 IIXfx では STFR, STDFR は SIMD 拡張される。SIMD 拡張された STFR, STDFR は、basic 側の単精度・倍精度ストアと extended 側の単精度・倍精度ストアを同時に実行する。レジスタの使用方法は “SIMD 拡張命令のレジスタ指定方法” (page 22) を参照。

SIMD STFR は 8 バイト境界に 2 つの単精度数データをストアする。アクセス境界違反には mem_address_not_aligned が通知される。

SIMD STDFR は 16 バイト境界に 2 つの倍精度数データをストアする。アクセス境界違反には mem_address_not_aligned が通知される。SIMD STDFR では 4 バイト境界に対するアクセスでも STDF_mem_address_not_aligned は通知されない。

SIMD STFR, SIMD STDFR はキャッシュプル領域に対してのみストアできる。ノンキャッシュプル領域に対して SIMD STFR, SIMD STDFR を実行しようとすると、data_access_exception が通知される。bypass ASI に対する SIMD ストアは、ASI_PHYS_USE_EC{LITTTLE} については可能である。

メモリアクセスのセマンティクスは通常のストア命令と同じく、TSO を遵守する。

SIMD STFR, SIMD STDFR は 1 命令で basic と extended の両方を同時にストアするが、basic と extended の間でも TSO が遵守される。

SIMD STFR, SIMD STDFR は basic, extended どちらでもウォッチポイントを検出する。

SIMD STFR, SIMD STDFR の例外条件と優先順位に関しては Appendix F.5.1, “Trap Conditions for SIMD Load/Store” (page 180) を参照。
Exceptions

illegal_instruction \((i = 0)\)

fp_disabled

illegal_action \((\text{XAR}.v = 1 \text{ and } (\text{XAR}.urs1 > 1 \text{ or } \text{XAR}.urs3[2] \neq 0))\)

\[(\text{XAR}.v = 1 \text{ and } \text{XAR}.simd = 1 \text{ and } ((\text{XAR}.urs2[2] \neq 0 \text{ or } \text{XAR}.urd[2] \neq 0))\]

mem_address_not_aligned

STDF_mem_address_not_aligned \((\text{STDFR} \text{ and } ((\text{XAR}.v = 0 \text{ or } \text{XAR}.simd = 0))\)

VA_watchpoint

data_access_exception

fast_data_access_MMU_miss

fast_data_access_protection

PA_watchpoint

data_access_error
A.78  Set XAR (SXAR)

HPC-ACE Ext.

<table>
<thead>
<tr>
<th>Regs.</th>
<th>SIMD</th>
<th>オペコード</th>
<th>op2</th>
<th>cmb</th>
<th>処理</th>
</tr>
</thead>
<tbody>
<tr>
<td>SXAR1</td>
<td>111</td>
<td>0</td>
<td></td>
<td></td>
<td>後続1命令のXAR指定</td>
</tr>
<tr>
<td>SXAR2</td>
<td>111</td>
<td>1</td>
<td></td>
<td></td>
<td>後続2命令のXAR指定</td>
</tr>
</tbody>
</table>

Format (2)

Description

SXAR命令はXARを更新する命令である。XARが2命令までの値を保持できるのに対応し、1命令分指定できるSXAR1と2命令分指定できるSXAR2がある。f_で始まるフィールドはSXARの直後に実行される命令に、s_で始まるフィールドは2命令目に適用される。SXAR1のs_フィールドは無視される。

Compatibility Note – SXAR1のs_で始まるフィールドが0以外でもillegal_instruction例外は通知されないが、将来の互換性を考慮したプログラムでは、SXAR1のs_で始まるフィールドに0以外の値を書くべきではない。

SXAR命令は後続の最大2命令で、SPARC64 IXfxで追加された整数・浮動小数点レジスタを使う場合や、SIMD拡張動作を指示するために使われる。

Implementation Note – 命令列上連続する命令を高速に実行できるようにハードウェアを実装してよい。

SXARと後続命令がメモリ上連続していない場合、たとえばSXARがディレイスロットに置かれていたり、Tccを挟んでいると、性能は低下する可能性がある。

SXARの実行はIIU_INST_TRAPで検出できない場合がある。

SXAR自身はXAR対象命令ではないので、実行時にXAR.v=1だとillegal_actionが通知される。

Ver 12, 2 Dec. 2013
Compatibility Note – op = 002, op2 = 1112 は SPARC V9 では Reserved だが、SPARC V8 ではは FBcc 命令が定義されていた。SPARC V8 のアプリケーションを SPARC64 IXfx で動かすと、動作が異なる可能性がある。

Programming Note – SXAR 命令は命令語中に XAR にセットする値を抱えているが、アセンブリ言語の表記上は引数はない。HPC-ACE 拡張動作は後続命令のニーモニックにサフィックスで記述し、アセンブラが SXAR に落とし込むこと。

```
sxar1
    fadd3, s %f0, %f2, %f4 /* SIMD */
```

Exceptions  

`illegal_action` (XAR.v = 1)
A.79 Cache Line Fill with Undetermined Values

<table>
<thead>
<tr>
<th>HPC-ACE Ext.</th>
<th>Regs.</th>
<th>SIMD</th>
<th>オペコード</th>
<th>immasi</th>
<th>ASI Value</th>
<th>处理</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>STXA</td>
<td>STDAD</td>
<td>STDFA</td>
<td>ASI_XFILL_AIUP</td>
<td>7216</td>
<td>プライマリ空間の指定されたアドレスをキャッシュにのせ、不定値で埋める。</td>
</tr>
<tr>
<td>✓</td>
<td>STXA</td>
<td>STDAD</td>
<td>STDFA</td>
<td>ASI_XFILL_AIUS</td>
<td>7316</td>
<td>セカンダリ空間の指定されたアドレスをキャッシュにのせ、不定値で埋める。</td>
</tr>
<tr>
<td>✓</td>
<td>STXA</td>
<td>STDAD</td>
<td>STDFA</td>
<td>ASI_XFILL_P</td>
<td>f216</td>
<td>プライマリ空間の指定されたアドレスをキャッシュにのせ、不定値で埋める。</td>
</tr>
<tr>
<td>✓</td>
<td>STXA</td>
<td>STDAD</td>
<td>STDFA</td>
<td>ASI_XFILL_S</td>
<td>f316</td>
<td>セカンダリ空間の指定されたアドレスをキャッシュにのせ、不定値で埋める。</td>
</tr>
</tbody>
</table>

Description

STXA, STDA, STDFA にこれらのASIを指定すると、指定されたアドレスに対応するキャッシュラインをキャッシュ上に書き込み用に確保し、そのアドレスを含むキャッシュラインを不定値で埋める。メモリからCPUへのデータ転送は発生しない。命令で指定するアドレスは、8バイト境界にある論理アドレスならば、キャッシュライン上の任意のアドレスを指定してよい。

STXA, STDA は、8バイト境界でないアドレスを指定すると mem_address_not_aligned を通知する。

STDFA は、4バイト境界だが8バイト境界でないアドレスを指定すると STDF_mem_address_not_aligned を通知し、8バイト境界でも4バイト境界でもないアドレスを指定すると mem_address_not_aligned を通知する。

XFILL_{AIUPAIUS,SP} ではハードウェアアブリファッチのon/off指定は意味を持たない。XAR.dis_hw_pfの値は無視される。

XFILL_{AIUPAIUS,SP} と後続のメモリアクセスの間のメモリオーダはTSOが遵守される。

TTE.CP = 0 であるページに対するXFILL_{AIUPAIUS,SP} は、ウォッチポイント、アラインメント、プロテクション違反は検出されるが、キャッシュラインの不定値フィルは行われない。

XFILL_{AIUPAIUS,SP} ではバスエラー、バスタイムアウトによるECC_error例外は通知されない。また、XFILL_{AIUPAIUS,SP} は、指定されたアドレスがL1キャッシュまたはL2キャッシュにあり、そのキャッシュライン上にUEがある場合でも、data_access_error例外は通知しない。
XFILL {AIUP, AIUS, S, P} はキャッシュライン 128 バイト全てでウォッチポイントの一致を検出する。

キャッシュラインフィル中に、当該キャッシュラインに対する後続アクセスが発生すると、後続アクセスはキャッシュラインフィルが完了するまで保留される。

Programming Notes – XFILL と後続アクセスの間に MEMBAR は不要。

後続アクセスが保留されると性能が低下するので、高速なプログラムを作成しようとするとときは XFILL は実際のストアに十分先立って実行する必要がある。しかし、XFILL が完了するまでにかかる時間はシステムに依存するので、あるシステムで十分先立って実行されていたものが、将来的のプロセッサでは不十分になる可能性がある。

本 ASI は memset(), memcpy() の高速化を意図して設けられた。以下に本 ASI を用いた memset()/memcpy() のサンプルコードを示す。命令ニーモニックは HPC-ACE 拡張サフィックスを使用している。詳細は Appendix G.4, ”HPC-ACE 拡張機能の表記法” (page 206) を参照。

これらのサンプルコードはいずれも、セクタ 0 に再利用性の低いデータがキャッシュされる場合のコードであることに注意。セクタ 0 とセクタ 1 に何という用途に使うかはアプリケーションが決めることなので、セクタ 1 に再利用性の低いデータをキャッシュするアプリケーションでこのサンプルコードを使うと、性能低下を引き起こす可能性がある。

```c
[memset(0) の pseudo-code]
/*
 *  %i0: dst
 */
ahead = 4 * 128; // 適時調整
for (i = 0 ; i < size; i += 128) {
    stxa      %g0, [%i0+ahead] #ASI_XFILL
    sxar2
    stx,d     %g0, [%i0]
    stx,d     %g0, [%i0+8]
    sxar2
    stx,d     %g0, [%i0+16]
    stx,d     %g0, [%i0+24]
    sxar2
    stx,d     %g0, [%i0+32]
    stx,d     %g0, [%i0+40]
    sxar2
    stx,d     %g0, [%i0+48]
    stx,d     %g0, [%i0+56]
sxar2
```
stx, d  %g0, [%i0+64]
stx, d  %g0, [%i0+72]
sxar2
stx, d  %g0, [%i0+80]
stx, d  %g0, [%i0+88]
sxar2
stx, d  %g0, [%i0+96]
stx, d  %g0, [%i0+104]
sxar2
stx, d  %g0, [%i0+112]
stx, d  %g0, [%i0+120]

add  %i0, 128, %i0

[memcpy() の pseudo-code]

/*
  *  %i0: dst
  *  %i1: src
*/

ahead = 4 * 128; // 適時調整

for (i = 0; i < size; i += 128) {
  prefetch [%i1+128], #n_reads
  ldx  [%i1], %l2
  ldx  [%i1+8], %l3
  ldx  [%i1+16], %l4
  ldx  [%i1+24], %l5
  ldx  [%i1+32], %l6
  ldx  [%i1+40], %l7
  ldx  [%i1+48], %l0
  ldx  [%i1+56], %l1
  ldx  [%i1+64], %l2
  ldx  [%i1+72], %l3
  ldx  [%i1+80], %l4
  ldx  [%i1+88], %l5
  ldx  [%i1+96], %l6
  ldx  [%i1+104], %l7
  ldx  [%i1+112], %l8
  ldx  [%i1+120], %l9

  stxa %g0, [%i0+ahead] #ASI_XFILL

  prefetch [%i0+128], #nWrites
  sxar2
  stx, d  %l2, [%i0]
stx, d  %l3, [%i0+8]
sxar2
Exceptions

fp_disabled (STDFA)
illegal_action (STXA, STDA with XAR.v = 1 and (XAR.urs1 > 1 or
   (i = 0 and XAR.urs2 > 1) or
   (i = 1 and XAR.urs2 ≠ 0) or
   XAR.urs3<2> ≠ 0 or
   XAR.urd > 1);

STDFA with XAR.v = 1 and(XAR.urs1 > 1 or
   (i = 0 and XAR.urs2 > 1) or
   (i = 1 and XAR.urs2 ≠ 0) or
   XAR.urs3<2> ≠ 0);

XAR.v = 1 and XAR.simd = 1)
mem_address_not_aligned
STDF_mem_address_not_aligned
privileged_action (ASI_XFILL_AIUP, ASI_XFILL_AIUS)
VA_watchpoint
fast_data_access_MMU_miss
data_access_exception
fast_data_access_protection
PA_watchpoint
data_access_error
IEEE Std. 754-1985 Requirements for SPARC-V9

IEEE Std. 754-1985 浮動小数点数規格は実装依存仕様を許容している。JPS1 Commonality の Appendix B では SPARC V9 に準拠したプロセッサ間での処理が統一されるよう、IEEE 754-1985 実装依存仕様の扱いを定義している。その詳細については JPS1 Commonality を参照。

この章で規定するのは、

■ unfinished_FPop が起きる条件
■ IEEE 非準拠モードの動作仕様

である。なお、前者は JPS1 Commonality の Section 5.1.7, “Floating-Point State Register (FSR)” (page 22) 内の “FSR_floating-point_trap_type (ftt)” の実装依存部を明確に再定義したもので、便宜上この章に配置してある。

B.1 Traps Inhibiting Results

詳細は JPS1 Commonality の Section B.1 を参照。SPARC64 IXfx はハードウェアとシステムソフトウェアが協調して、この節で定義されている状態を実現する。

B.6 Floating-Point Nonstandard Mode

この節では、fp_exception_other 例外を FSR.ftt = unfinished_FPop で通知する境界条件と、SPARC64 IXfx の IEEE 754-1985 非準拠モードの動作について説明する。前者は SPARC64 IXfx では IEEE 準拠モード (FSR.NS = 0) のときにのみ起きる例外だが、読者の便宜を考えこの章に配置してある。
SPARC64 IXfx の浮動小数点演算器は一定範囲の数のみを扱うことができる。演算の入力になる数や中間結果から、演算結果がその範囲から外れるだろうと予想される場合、SPARC64 IXfx は演算を中断し $fp_{\text{exception\_other}}$ 例外を $FSR.f\text{tt} = 0216$ ($unfinished\_FPop$) で通知し、後の処理をソフトウェアに委ねる。そしてエミュレーションルーチンは、IEEE 754-1985 に準拠する形で演算を完了させる (impl. dep. #3)。

SPARC64 IXfx は IEEE 754-1985 非準拠モードで動作することもできる。これは $FSR.\text{NS} = 1$ により指示する (“$FSR\_nonstandard\_fp$ (NS)” (page 22) を参照)。$FSR.\text{NS}$ の値により SPARC64 IXfx の動作は変わってくる。

B.6.1 $fp_{\text{exception\_other}}$ Exception ($f\text{tt}=unfinished\_FPop$)

SPARC64 IXfx のほとんどの浮動小数点演算命令は、$fp_{\text{exception\_other}}$ 例外を $FSR.f\text{tt} = unfinished\_FPop$ で通知する可能性がある (詳細は各命令の仕様を参照)。例外発生の境界条件は以下の通りである。

1. 入力オペランドの 1 つが非正規化数で、それ以外がゼロでない正規化数 (NaN と無限大を除く) のとき、$fp_{\text{exception\_other}}$ 例外が $unfinished\_FPop$ で通知される。ただし、結果がゼロかオーバーフローのときはこの限りではない。

2. すべての入力オペランドが非正規化数で、結果がゼロかオーバーフロー以外のとき、$fp_{\text{exception\_other}}$ 例外が $unfinished\_FPop$ で通知される。

3. すべての入力オペランドが正規化数、丸める前の演算結果が非正規化数で、$TEM.\text{UFM} = 0$、かつ結果がゼロでない場合、$fp_{\text{exception\_other}}$ 例外が $unfinished\_FPop$ で通知される。

結果が定数、たとえばゼロや無限大になると予想され、そのための演算が簡単な場合、$unfinished\_FPop$ を通知せずに演算を行う。
Implementation Note – 境界条件を精確に検出するためには大量の論理回路が必要になる。このようなハードウェアのコストを避けるため、SPARC64 IXfxは中間結果の指数部（丸める前の値）を計算することで境界条件にあてはまるかどうかを大まかに判定する。このような判定方法のため、精確にはゼロやオーバフローにならない値でも、そうであると判定し、unfinished_FPop例外を通知することがある。

境界条件検出のための、結果の指数部を求める式を TABLE B-1 に示す。ここで Er は丸める前の計算値の指数部（バイアス込み）で、入力データの指数部 (esrc1, esrc2) だけから計算される。

TABLE B-1  unfinished_FPop 例外検出のための結果の指数部の計算式

<table>
<thead>
<tr>
<th>处理</th>
<th>計算式</th>
</tr>
</thead>
<tbody>
<tr>
<td>fmuls</td>
<td>$Er = esrc1 + esrc2 - 126$</td>
</tr>
<tr>
<td>fmuld</td>
<td>$Er = esrc1 + esrc2 - 1022$</td>
</tr>
<tr>
<td>fdivs</td>
<td>$Er = esrc1 - esrc2 + 126$</td>
</tr>
<tr>
<td>fdivd</td>
<td>$Er = esrc1 - esrc2 + 1022$</td>
</tr>
</tbody>
</table>

esrc1, esrc2 はバイアス込みの入力データの指数部である。非正規化数の場合、この値は 0 である。

Er から eres が求められる。eres は仮数部の桁合わせ後、丸める前の指数部である。すなわち、演算後の仮数部を暗黙の 1 が小数点の左になるまで右または左にシフトし、そのシフト量を Er に加算または減算する操作後の値である。

TABLE B-2 に、全浮動小数点演算命令の unfinished_FPop 検出条件をまとめた。

TABLE B-2  unfinished_FPop 検出条件

<table>
<thead>
<tr>
<th>处理</th>
<th>検出条件</th>
</tr>
</thead>
<tbody>
<tr>
<td>FdToS</td>
<td>$-25 &lt; eres &lt; 1$ かつ TEM.UFM = 0 のとき。</td>
</tr>
<tr>
<td>FsToD</td>
<td>第二オペランド (rs2) が非正規化数のとき。</td>
</tr>
<tr>
<td>FADDs, FSUBs, FADDd, FSUBd</td>
<td>1. 入力のひとつが非正規化数、もうひとつがゼロ以外の正規化数 (無限大と NaN を除く) のとき。</td>
</tr>
<tr>
<td></td>
<td>2. 入力が両方とも非正規化数。</td>
</tr>
<tr>
<td></td>
<td>3. 入力が両方ともゼロ以外の正規化数 (無限大と NaN を除く) で、eres &lt; 1 かつ TEM.UFM = 0 のとき。</td>
</tr>
<tr>
<td>FMULs, FMULd</td>
<td>1. 入力のひとつが非正規化数、もうひとつがゼロ以外の正規化数 (無限大と NaN を除く) のとき、</td>
</tr>
<tr>
<td></td>
<td>単精度演算: $-25 &lt; Er$</td>
</tr>
<tr>
<td></td>
<td>倍精度演算: $-54 &lt; Er$</td>
</tr>
<tr>
<td></td>
<td>2. 入力が両方ともゼロ以外の正規化数 (無限大と NaN を除く) で</td>
</tr>
<tr>
<td></td>
<td>TEM.UFM = 0 のとき、</td>
</tr>
<tr>
<td></td>
<td>単精度演算: $-25 &lt; eres &lt; 1$</td>
</tr>
<tr>
<td></td>
<td>倍精度演算: $-54 &lt; eres &lt; 1$</td>
</tr>
</tbody>
</table>
1. 入力のひとつが非正規化数、もうひとつがゼロ以外の正規化数（無限大とNaNを除く）のとき。
2. 入力が両方とも非正規化数のとき。

FDIVs, FDIVd
1. 被除数(rs1)がゼロ以外の正規化数（無限大とNaNを除く）、除数(rs2)が非正規化数のとき、
   単精度演算: $E_r < 255$
   倍精度演算: $E_r < 2047$
2. 被除数(rs1)が非正規化数、除数(rs2)がゼロ以外の正規化数（無限大とNaNを除く）のとき、
   単精度演算: $-25 < E_r$
   倍精度演算: $-54 < E_r$
3. 被除数、除数とも非正規化数のとき。
4. 被除数、除数ともゼロ以外の正規化数（無限大とNaNを除く）で
   $TEM.UFM = 0$ のとき、
   単精度演算: $-25 < eres < 1$
   倍精度演算: $-54 < eres < 1$

FSQRTs, FSQRTd
入力(rs2)が正のゼロでない非正規化数のとき。

FMADD{s,d}, FMSUB{s,d}, FNADD{s,d}, FNMSUB{s,d}
乗算部分は FMUL{s,d} と同じ。加算部分は FADD{s,d} と同じ。

FTRIMADDd
乗算部分は FMUL{s,d} と同じ。加算部分では起きない。

FTRISMULd
1. rs1 がゼロ以外の正規化数（無限大とNaNを除く）で $TEM.UFM = 0$ のとき、
   倍精度演算: $-54 < eres < 1$

FRCPA{s,d}
入力が非正規化数のとき。

FRSQRTA{s,d}
入力が正のゼロでない非正規化数のとき。

1. 入力が非正規化数とゼロのときは、IEEE754-1985 で規定されている通りの結果を生成する。

<table>
<thead>
<tr>
<th>処理</th>
<th>検出条件</th>
</tr>
</thead>
<tbody>
<tr>
<td>FSMULd</td>
<td>1. 入力のひとつが非正規化数、もうひとつがゼロ以外の正規化数（無限大とNaNを除く）のとき。</td>
</tr>
<tr>
<td></td>
<td>2. 入力が両方とも非正規化数のとき。</td>
</tr>
<tr>
<td>FDIVs, FDIVd</td>
<td>1. 被除数(rs1)がゼロ以外の正規化数（無限大とNaNを除く）、除数(rs2)が非正規化数のとき、</td>
</tr>
<tr>
<td></td>
<td>単精度演算: $E_r &lt; 255$</td>
</tr>
<tr>
<td></td>
<td>倍精度演算: $E_r &lt; 2047$</td>
</tr>
<tr>
<td></td>
<td>2. 被除数(rs1)が非正規化数、除数(rs2)がゼロ以外の正規化数（無限大とNaNを除く）のとき、</td>
</tr>
<tr>
<td></td>
<td>単精度演算: $-25 &lt; E_r$</td>
</tr>
<tr>
<td></td>
<td>倍精度演算: $-54 &lt; E_r$</td>
</tr>
<tr>
<td></td>
<td>3. 被除数、除数とも非正規化数のとき。</td>
</tr>
<tr>
<td></td>
<td>4. 被除数、除数ともゼロ以外の正規化数（無限大とNaNを除く）で</td>
</tr>
<tr>
<td></td>
<td>$TEM.UFM = 0$ のとき、</td>
</tr>
<tr>
<td></td>
<td>単精度演算: $-25 &lt; eres &lt; 1$</td>
</tr>
<tr>
<td></td>
<td>倍精度演算: $-54 &lt; eres &lt; 1$</td>
</tr>
<tr>
<td>FSQRTs, FSQRTd</td>
<td>入力(rs2)が正のゼロでない非正規化数のとき。</td>
</tr>
<tr>
<td>FMADD{s,d}, FMSUB{s,d}, FNADD{s,d}, FNMSUB{s,d}</td>
<td>乗算部分は FMUL{s,d} と同じ。加算部分は FADD{s,d} と同じ。</td>
</tr>
<tr>
<td>FTRIMADDd</td>
<td>乗算部分は FMUL{s,d} と同じ。加算部分では起きない。</td>
</tr>
<tr>
<td>FTRISMULd</td>
<td>1. rs1 がゼロ以外の正規化数（無限大とNaNを除く）で $TEM.UFM = 0$ のとき、</td>
</tr>
<tr>
<td></td>
<td>倍精度演算: $-54 &lt; eres &lt; 1$</td>
</tr>
<tr>
<td>FRCPA{s,d}</td>
<td>入力が非正規化数のとき。</td>
</tr>
<tr>
<td>FRSQRTA{s,d}</td>
<td>入力が正のゼロでない非正規化数のとき。</td>
</tr>
</tbody>
</table>
Conditions for a Zero Result

TABLE B-3 の条件が成立するとき、SPARC64 IXfx は演算結果として、FSR.RD に応じて 0 または最小の非正規化数を返す。この値を pessimistic zero と呼ぶ。

TABLE B-3 　Pessimistic Zero 発生条件

<table>
<thead>
<tr>
<th>处理</th>
<th>条件</th>
<th>左方のオペランドが非正規化数</th>
<th>裏方のオペランドが非正規化数</th>
<th>脱い方が正規化数</th>
</tr>
</thead>
<tbody>
<tr>
<td>FdTOs</td>
<td>常に</td>
<td>-</td>
<td>-</td>
<td>erez ≤ -25</td>
</tr>
<tr>
<td>FMULs、</td>
<td>単精度、</td>
<td>Er ≤ -25</td>
<td>常に</td>
<td>帯度、</td>
</tr>
<tr>
<td>FMULd</td>
<td>除精度</td>
<td>Er ≤ -54</td>
<td>常に</td>
<td>除精度</td>
</tr>
<tr>
<td>FDIVs、</td>
<td>除精度、</td>
<td>Er ≤ -25</td>
<td>脱こらない</td>
<td>単精度</td>
</tr>
<tr>
<td>FDIVd</td>
<td>除精度</td>
<td>Er ≤ -54</td>
<td>脱こらない</td>
<td>帯度、</td>
</tr>
</tbody>
</table>

1.両方ともゼロ、NaN、無限大以外。
2.両方がゼロでもよいが、どちらも無限大、NaN ではない。

Conditions for an Overflow Result

TABLE B-4 の条件が成立するとき、SPARC64 IXfx は演算でオーバーフローが起きたものとみなす。

TABLE B-4 　Pessimistic Overflow 発生条件

<table>
<thead>
<tr>
<th>处理</th>
<th>条件</th>
</tr>
</thead>
<tbody>
<tr>
<td>FDIVs</td>
<td>除数 (rs2) が非正規化数で Er ≥ 255 のとき。</td>
</tr>
<tr>
<td>FDIVd</td>
<td>除数 (rs2) が非正規化数で Er ≥ 2047 のとき。</td>
</tr>
</tbody>
</table>

B.6.2 Behavior when FSR.NS = 1

FSR.NS = 1 (IEEE 非準拠モード) の場合、SPARC64 IXfx は入力あるいは演算結果が非正規化数になった場合、それをゼロに置き換えて処理を続けようとする。この置き換えにより動作がどのように変わるかを以下に示す。

- 入力の一つが非正規化数で、すべてがゼロ、NaN、無限大以外のとき、非正規化数は同符号のゼロで置き換えて処理が行われる。演算後、cexc.nxc = 1 がセットされる。ただし、以下の場合に該当する場合は cexc.nxc = 0 になる。
  - 演算が FDIV{s,d} で、ゼロ除算や無効演算が検出されたとき
  - 演算が FSQRT{s,d} で、無効演算が検出されたとき
  - 演算が FRPCA{s,d}、FRSQRTA{s,d} のとき

  cexc.nxc = 1 かつ FSR の TEM,NXM = 1 のとき、fp_exception_ieee_754例外が通知される。

- 丸める前の演算結果が非正規化数のときは、同符号のゼロで置き換えられる。
FSR の TEM.UFM = 1 ならば cexc.ufc = 1 が、TEM.UFM = 0 で TEM.NXM = 1 なら cexc.nxc = 1 がセットされ、fp_exception_ieee_754 例外が通知される。TEM.UFM = 0 かつ TEM.NXM = 0 のときは、cexc.ufc = 1, cexc.nxc = 1 がセットされる。

上に述べたとおり、FSR.NS = 1 のとき SPARC64 IXfx は unfinished_FPop 例外を起こさないし、演算結果として非正規化数を返すこともない。

TABLE B-5 は、TABLE B-2 であげた浮動小数点演算命令1 について、入力と結果の数の種類と FSR.TEM によりどの例外が通知されるかを、FSR.NS = 0, 1 それぞれの場合についてまとめたものである。条件にしたがって表を左から右へとたどると、Result の列に起こりうる例外と例外がマスクされている場合の演算結果が得られる。

FSR.NS = 1 で入力が非正規化数の場合は TABLE B-6 を参照。TABLE B-5 で Result が網掛けの部分は、IEEE754-1985 に準拠していることを示す。

Note – TABLE B-5, TABLE B-6 を通じて。英小文字の例外条件、nx, uf, of, dv, nv は、FSR.TEM のそれぞれに対応するビットが 0 のため fp_exception_ieee_754 例外が通知されないこと、大文字の NX, UF, OF, DV, NV は fp_exception_ieee_754 例外が通知されることを示す。

### TABLE B-5 浮動小数点例外の発生条件と結果 (1 of 2)

<table>
<thead>
<tr>
<th>FSR.NS</th>
<th>入力が非正規化数1</th>
<th>結果が非正規化数2</th>
<th>Zero Result</th>
<th>Overflow Result</th>
<th>UFM</th>
<th>OFM</th>
<th>NXM</th>
<th>結果</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No</td>
<td>Yes</td>
<td>—</td>
<td>—</td>
<td>1</td>
<td>—</td>
<td>—</td>
<td>UF</td>
</tr>
<tr>
<td></td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>1</td>
<td>—</td>
<td>—</td>
<td>UF</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>—</td>
<td>0</td>
<td>—</td>
<td>0</td>
<td>1</td>
<td>—</td>
<td>NF</td>
</tr>
<tr>
<td></td>
<td>No</td>
<td>—</td>
<td>0</td>
<td>—</td>
<td>0</td>
<td>—</td>
<td>—</td>
<td>unfinished_FPop4</td>
</tr>
<tr>
<td></td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>IEEE754-1985 準拠</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>—</td>
<td>1</td>
<td>—</td>
<td>1</td>
<td>—</td>
<td>—</td>
<td>UF</td>
</tr>
<tr>
<td></td>
<td>No</td>
<td>Yes</td>
<td>—</td>
<td>0</td>
<td>—</td>
<td>1</td>
<td>—</td>
<td>NF</td>
</tr>
<tr>
<td></td>
<td>—</td>
<td>No</td>
<td>—</td>
<td>—</td>
<td>1</td>
<td>—</td>
<td>—</td>
<td>unfinished_FPop4</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>—</td>
<td>—</td>
<td>1</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>NF</td>
</tr>
<tr>
<td></td>
<td>No</td>
<td>—</td>
<td>—</td>
<td>0</td>
<td>1</td>
<td>—</td>
<td>—</td>
<td>unfinished_FPop4</td>
</tr>
</tbody>
</table>

1. FTRISmuld の m2 は浮動小数点数ではないので、Source Denormal の対象外。
TABLE B-6 は、TABLE B-2 であげた浮動小数点演算命令について、FSR.NS = 1 で入力が非正規化数のときの SPARC64 IXfx の動作をまとめたものである。TABLE B-6 で Result が網掛けの部分は、IEEE754-1985 に準拠していることを示す。

### TABLE B-6  FSR.NS = 1 における非正規化数の演算 (1 of 2)

<table>
<thead>
<tr>
<th>命令</th>
<th>入力値</th>
<th>FSR.TEM</th>
<th>結果</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>op1</td>
<td>op2</td>
<td>op3</td>
</tr>
<tr>
<td>FsTod</td>
<td>—</td>
<td>Denorm</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FdTos</td>
<td>—</td>
<td>Denorm</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FADD{s,d}</td>
<td>Denorm</td>
<td>Normal</td>
<td>—</td>
</tr>
<tr>
<td>FSUB{s,d}</td>
<td>Normal</td>
<td>Denorm</td>
<td>—</td>
</tr>
<tr>
<td>FMUL{s,d}</td>
<td>Denorm</td>
<td>Denorm</td>
<td>—</td>
</tr>
<tr>
<td>FsMULd</td>
<td>—</td>
<td>Denorm</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. 入力のひとつが非正規化数。他は正規化数 (0, NaN, 無限大を除く) でも非正規化数でもよい。
2. 丸め前の演算結果が非正規化数になった場合。
3. Dmin = 最小の非正規化数。
4. 演算が FADD{s,d}, FSUB{s,d} で入力がゼロと非正規化数の場合、unfinished_FPpop 例外を通知せず IEEE754-1985 に準拠した演算を行う。
5. Nmax = 最大の正規化数。
<table>
<thead>
<tr>
<th>命令</th>
<th>入力値</th>
<th>FSR.TEM</th>
<th>結果</th>
</tr>
</thead>
<tbody>
<tr>
<td>FDIV{s,d}</td>
<td>Denorm</td>
<td>Normal</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>Normal</td>
<td>Denorm</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>Denorm</td>
<td>Denorm</td>
<td>—</td>
</tr>
<tr>
<td>FSQRT{s,d}</td>
<td>Denorm and op2 &gt; 0</td>
<td>—</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Denorm and op2 &lt; 0</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>FMADD{s,d}</td>
<td>Denorm</td>
<td>Normal</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>Denorm</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>FNSADD{s,d}</td>
<td>Denorm</td>
<td>Denorm</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>Denorm</td>
<td>Normal</td>
<td>—</td>
</tr>
<tr>
<td>FNSUB{s,d}</td>
<td>Denorm</td>
<td>Denorm</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>Denorm</td>
<td>Denorm</td>
<td>—</td>
</tr>
<tr>
<td>FTRIMADDd2</td>
<td>Denorm</td>
<td>Denorm</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>Normal</td>
<td>Normal</td>
<td>—</td>
</tr>
<tr>
<td>FTRISMULd</td>
<td>Denorm</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>FRCPA{s,d}</td>
<td>Denorm</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>FRSQRTA{s,d}</td>
<td>Denorm</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

1. 単精度の dNaN は 7FFF.FFFF16, 信頼度の dNaN は 7FFF.FFFF.FFFF.FFFF16
2. op3 は算術器のテーブルから引いてくる。常に正規化数。
3. op1 × op2 が Denorm のときは, op1 × op2 と同符号のゼロ。
Implementation Dependencies

この章では、JPS1 Commonalityで実装依存仕様とされたものについて、
SPARC64 IXfxがどのように実装しているかをまとめている。SPARC V9とSPARC
JPS1で、実装依存仕様は“IMPL. DEP. #nn:”という形式で記述されており、この仕様
書で実装依存仕様の実装を定義した部分には“(impl. dep. #nn)”という形式で、対応す
る実装依存仕様が示されている。この実装依存仕様の一覧をTABLE C-1にまとめた。

Note – SPARC Internationalは、SPARC V9に準拠したすべてのプロセッサの実装依存
について書かれたドキュメントImplementation Characteristics of Current SPARC-V9-
based Products, Revision 9.xを提供している。このドキュメントに関する問い合わせは
下記へ。
home page: www.sparc.org
email: info@sparc.org

C.4 List of Implementation Dependencies

TABLE C-1は、JPS1実装依存仕様がSPARC64 IXfxでどのように実装されているかを
まとめた表である。

<table>
<thead>
<tr>
<th>番号</th>
<th>命令のソフトウェアエミュレーション</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>命令のソフトウェアエミュレーション</td>
</tr>
<tr>
<td></td>
<td>ソフトウェアはillegal_instructionまたはunimplemented_FPop例外を通知す</td>
</tr>
<tr>
<td></td>
<td>るすべての4倍精度命令をエミュレーションすること。</td>
</tr>
</tbody>
</table>

---

F. Appendix C  Implementation Dependencies  147

Ver 12, 2 Dec. 2013
汎用整数レジスタの本数
SPARC64 IXfx は 8 レジスタウインドウを実装する \((\text{NWINDOWS} = 8)\)。
SPARC64 IXfx はさらに 2 フレームのグローバルレジスタ（インタラプト用と MMU 用）を実装するほか、HPC-ACE でも拡張され、合計 192 本の汎用整数レジスタを実装する。

IEEE 754-1985 仕様に準拠しない演算結果
Appendix B.6, “Floating-Point Nonstandard Mode” を参照。

I/O レジスタのアクセス権限
この実装依存項目は本仕様書の規定範囲外である。SPARC64 IXfx を搭載するシステムの仕様書を参照。

I/O レジスタ定義
この実装依存項目は本仕様書の規定範囲外である。SPARC64 IXfx を搭載するシステムの仕様書を参照。

RDASR/WRASR 対象レジスタ
SPARC64 IXfx では RDASR で XAR, XASR, TXAR の読み出しができ、WRASR で XASR, TXAR の書き込みができる。

RDASR/WRASR のアクセス権限
SPARC64 IXfx では TXAR は特権レジスタである。

Reserved.

VER.impl
SPARC64 IXfx では VER.impl = 9 である。

Reserved.

IU deferred トラブルキー
SPARC64 IXfx では IU deferred トラブルキーは不要なので実装していない。

Reserved.

IEEE 754-1985 非準拠時の演算結果
SPARC64 IXfx は FSR.NS = 1 のとき、演算結果が非正規化数になるとゼロで置き換える。詳細は Appendix B.6, “Floating-Point Nonstandard Mode” を参照。

FPU バージョン、FSR.ver
SPARC64 IXfx では FSR.ver = 0。

Reserved.

FPU の TEM, cexc, aexc
SPARC64 IXfx は TEM, cexc, and aexc のすべてのビットを実装している。

浮動小数点演算のトラブル
SPARC64 IXfx では、浮動小数点演算で検出される例外のトラブルはすべて precise である。よって FQ は必要ない。

<table>
<thead>
<tr>
<th>番号</th>
<th>SPARC64 IXfx での実装</th>
<th>ページ</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>汎用整数レジスタの本数</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>SPARC64 IXfx は 8 レジスタウインドウを実装する ((\text{NWINDOWS} = 8))。</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SPARC64 IXfx はさらに 2 フレームのグローバルレジスタ（インタラプト用と MMU 用）を実装するほか、HPC-ACE でも拡張され、合計 192 本の汎用整数レジスタを実装する。</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>IEEE 754-1985 仕様に準拠しない演算結果</td>
<td>139</td>
</tr>
<tr>
<td></td>
<td>Appendix B.6, “Floating-Point Nonstandard Mode” を参照。</td>
<td></td>
</tr>
<tr>
<td>4-5</td>
<td>Reserved.</td>
<td>—</td>
</tr>
<tr>
<td>6</td>
<td>I/O レジスタのアクセス権限</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>この実装依存項目は本仕様書の規定範囲外である。SPARC64 IXfx を搭載するシステムの仕様書を参照。</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>I/O レジスタ定義</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>この実装依存項目は本仕様書の規定範囲外である。SPARC64 IXfx を搭載するシステムの仕様書を参照。</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>RDASR/WRASR 対象レジスタ</td>
<td>97, 111</td>
</tr>
<tr>
<td></td>
<td>SPARC64 IXfx では RDASR で XAR, XASR, TXAR の読み出しができ、WRASR で XASR, TXAR の書き込みができる。</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>RDASR/WRASR のアクセス権限</td>
<td>97, 111</td>
</tr>
<tr>
<td></td>
<td>SPARC64 IXfx では TXAR は特権レジスタである。</td>
<td></td>
</tr>
<tr>
<td>10–12</td>
<td>Reserved.</td>
<td>—</td>
</tr>
<tr>
<td>13</td>
<td>VER.impl</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>SPARC64 IXfx では VER.impl = 9 である。</td>
<td></td>
</tr>
<tr>
<td>14–15</td>
<td>Reserved.</td>
<td>—</td>
</tr>
<tr>
<td>16</td>
<td>IU deferred トラブルキー</td>
<td>38</td>
</tr>
<tr>
<td></td>
<td>SPARC64 IXfx では IU deferred トラブルキーは不要なので実装していない。</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Reserved.</td>
<td>—</td>
</tr>
<tr>
<td>18</td>
<td>IEEE 754-1985 非準拠時の演算結果</td>
<td>139</td>
</tr>
<tr>
<td></td>
<td>SPARC64 IXfx は FSR.NS = 1 のとき、演算結果が非正規化数になるとゼロで置き換える。詳細は Appendix B.6, “Floating-Point Nonstandard Mode” を参照。</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>FPU バージョン、FSR.ver</td>
<td>23</td>
</tr>
<tr>
<td></td>
<td>SPARC64 IXfx では FSR.ver = 0。</td>
<td></td>
</tr>
<tr>
<td>20–21</td>
<td>Reserved.</td>
<td>—</td>
</tr>
<tr>
<td>22</td>
<td>FPU の TEM, cexc, aexc</td>
<td>23</td>
</tr>
<tr>
<td></td>
<td>SPARC64 IXfx は TEM, cexc, and aexc のすべてのビットを実装している。</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>浮動小数点演算のトラブル</td>
<td>38</td>
</tr>
<tr>
<td></td>
<td>SPARC64 IXfx では、浮動小数点演算で検出される例外のトラブルはすべて precise である。よって FQ は必要ない。</td>
<td></td>
</tr>
</tbody>
</table>
TABLE C-1  JPS1 実装依存仕様の SPARC64 IXfx での実装 (3 of 12)

<table>
<thead>
<tr>
<th>番号</th>
<th>SPARC64 IXfx での実装</th>
<th>ページ</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>FPUG トレップキュー (FQ)</td>
<td>38</td>
</tr>
<tr>
<td></td>
<td>SPARC64 IXfx では FQ deferred トレップキューは不要なので実装していない。</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>FQ が存在しない場合の RDPR での読み出し</td>
<td>38</td>
</tr>
<tr>
<td></td>
<td>FQ を RDPR で読み出すとすると、illegal_instruction が通知される。</td>
<td></td>
</tr>
<tr>
<td>26-28</td>
<td>Reserved.</td>
<td>—</td>
</tr>
<tr>
<td>29</td>
<td>空間識別子 (ASI) の定義</td>
<td>213</td>
</tr>
<tr>
<td></td>
<td>SPARC64 IXfx の定義は Appendix L を参照。</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>ASI 番号の有効ビット数</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>SPARC64 IXfx は 8 ビットすべてをデコードする。</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>致命的なエラーのトラップ</td>
<td>248</td>
</tr>
<tr>
<td></td>
<td>SPARC64 IXfx は、命令コミットを監視するウォッチドッグタイマーを実装しており、ある CPU サイクル以上命令がコミットされないと、async_data_error 例外を通知しようとする。约 6.7 秒経過すると、プロセッサは error_state に遷移する。error_state に入る際、error_state からの復旧のために WDR リセットを発行するように設定することもできる。</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>Deferred トラップ</td>
<td>44, 257</td>
</tr>
<tr>
<td></td>
<td>SPARC64 IXfx は深刻なエラーを報告するトラップを deferred で通知することがある。SPARC64 IXfx は deferred トラブルキューを実装しない。</td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>トラップの精度</td>
<td>44</td>
</tr>
<tr>
<td></td>
<td>深刻なエラーを報告するトラップ以外で deferred なトラップはない。</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SPARC64 IXfx では、命令実行の結果発生するトラップはすべて precise である。</td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>インタラプトのクリア</td>
<td>241</td>
</tr>
<tr>
<td></td>
<td>インタラプトの詳細は Appendix N を参照。</td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>実装依存なトラップ</td>
<td>51</td>
</tr>
<tr>
<td></td>
<td>SPARC64 IXfx は実装依存と定義された以下のトラップを実装している。</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• interrupt_vector_trap (tt = 06016)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• PA_watchpoint (tt = 06116)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• VA_watchpoint (tt = 06216)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• ECC_error (tt = 06316)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• fast_instruction_access_MMU_miss (tt = 06416 から 06716)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• fast_data_access_MMU_miss (tt = 06816 から 06B16)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• fast_data_access_protection (tt = 06C16 から 06F16)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• async_data_error (tt = 04016)</td>
<td></td>
</tr>
</tbody>
</table>
トラップの優先順位
SPARC64 IXfx の実装依存トラップの優先順位は以下の通りである。
- interrupt_vector_trap (priority=16)
- PA_watchpoint (priority=12)
- VA_watchpoint (priority=1)
- ECC_error (priority=33)
- fast_instruction_access_MMU_miss (priority = 2)
- fast_data_access_MMU_miss (priority = 12)
- fast_data_access_protection (priority = 12)
- async_data_error (priority = 2)

リセットトラップ
SPARC64 IXfx はパワーオンリセット (POR) とウォッチドッグリセットを実装している。

リセットトラップ時の実装依存レジスタへの作用
Appendix O.2, “RED_state と error_state”を参照。

error_state に遷移する固有の条件
ウォッチドッグタイムアウトで約 6.7 秒経過すると、または TL=MAXTL で通常のトラップか SIR が発生すると、error_state に遷移する。

error_state
SPARC64 IXfx は error_state に遷移後、ウォッチドッグリセットを発行することもできる。ほとんどのエラーログ用レジスタの状態は保存される (impl. dep. #254 も参照)。

Reserved.

データアクセスの FPU トラブル
アクセスエラーを起こしたとき、レジスタの値は更新されない。

Reserved.

RDASR
SPARC64 IXfx では rd = 29–31 でそれぞれ XAR, XASR, TXAR が読み出せる。このとき、
- 命令フィールドの bit<18:0> の扱いは他の RDASR と同じである。すなわち、
  bit<18:14> は rs1, bit<13> は i, i=0 のとき、bit<12:5> は Reserved,
  bit<4:0> は rs2 である。i=1 のとき、bit<12:0> は simm13 である。
- 特権レジスタは TXAR のみ。

Reserved が 0 以外でも illegal_instruction 例外を検出しない。

<table>
<thead>
<tr>
<th>番号</th>
<th>SPARC64 IXfx での実装</th>
<th>ページ</th>
</tr>
</thead>
<tbody>
<tr>
<td>36</td>
<td>トラップの優先順位</td>
<td>49</td>
</tr>
<tr>
<td></td>
<td>SPARC64 IXfx の実装依存トラップの優先順位は以下の通りである。</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• interrupt_vector_trap (priority=16)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• PA_watchpoint (priority=12)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• VA_watchpoint (priority=1)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• ECC_error (priority=33)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• fast_instruction_access_MMU_miss (priority = 2)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• fast_data_access_MMU_miss (priority = 12)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• fast_data_access_protection (priority = 12)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• async_data_error (priority = 2)</td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>リセットトラップ</td>
<td>44</td>
</tr>
<tr>
<td></td>
<td>SPARC64 IXfx はパワーオンリセット (POR) とウォッチドッグリセットを実装している。</td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>リセットトラップ時の実装依存レジスタへの作用</td>
<td>249</td>
</tr>
<tr>
<td></td>
<td>Appendix O.2, “RED_state と error_state”を参照。</td>
<td></td>
</tr>
<tr>
<td>39</td>
<td>error_state に遷移する固有の条件</td>
<td>44</td>
</tr>
<tr>
<td></td>
<td>ウォッチドッグタイムアウトで約 6.7 秒経過すると、または TL=MAXTL で通常のトラップか SIR が発生すると、error_state に遷移する。</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>error_state</td>
<td>44</td>
</tr>
<tr>
<td></td>
<td>SPARC64 IXfx は error_state に遷移後、ウォッチドッグリセットを発行することもできる。ほとんどのエラーログ用レジスタの状態は保存される (impl. dep. #254 も参照)。</td>
<td></td>
</tr>
<tr>
<td>41</td>
<td>Reserved.</td>
<td>44</td>
</tr>
<tr>
<td>42</td>
<td>FLUSH 命令</td>
<td>81</td>
</tr>
<tr>
<td></td>
<td>SPARC64 IXfx は FLUSH 命令をハードウェアで実行する。</td>
<td></td>
</tr>
<tr>
<td>43</td>
<td>Reserved.</td>
<td>81</td>
</tr>
<tr>
<td>44</td>
<td>データアクセスの FPU トラブル</td>
<td>81</td>
</tr>
<tr>
<td></td>
<td>アクセスエラーを起こしたとき、レジスタの値は更新されない。</td>
<td></td>
</tr>
<tr>
<td>45–46</td>
<td>Reserved.</td>
<td>81</td>
</tr>
<tr>
<td>47</td>
<td>RDASR</td>
<td>97</td>
</tr>
<tr>
<td></td>
<td>SPARC64 IXfx では rd = 29–31 でそれぞれ XAR, XASR, TXAR が読み出せる。このとき、</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 命令フィールドの bit&lt;18:0&gt; の扱いは他の RDASR と同じである。すなわち、</td>
<td></td>
</tr>
</tbody>
</table>
|      |  bit<18:14> は rs1, bit<13> は i, i=0 のとき、bit<12:5> は Reserved,
|      |  bit<4:0> は rs2 である。i=1 のとき、bit<12:0> は simm13 である。 |
|      | • 特権レジスタは TXAR のみ。 |
|      | Reserved が 0 以外でも illegal_instruction 例外を検出しない。 |
48  WRASR
SPARC64 IXfx では rd = 29–31 でそれぞれ XAR, XASR, TXAR に書き込むことができ。このとき、
- 命令フィールドの bit<18:0> の扱いは他の WRASR と同じである。すなわち、
  bit<18:14> は rs1, bit<13> は i, i=0 のとき、bit<12:5> は Reserved,
  bit<4:0> は rs2 である。i=1 のとき、bit<12:0> は simm13 である。
- rs1 と rs2 または simm13 の間の演算は xor である。
- 特権レジスタは TXAR のみ。
Reserved が 0 以外でも illegal_instruction 例外を検出しない。

55  浮動小数点演算のアンダーフロー検出
SPARC64 IXfx は JPS1 の規定通り、丸め前にアンダーフローを検出する。

56–100  Reserved.

101  最大トラップレベル
MAXTL = 5.

102  クリーンウィンドウトラップ
SPARC64 IXfx は clean_window トラブルを通知する。レジスタウィンドウはソフトウェアがクリアする。

103  プリファッチ命令
SPARC64 IXfx では PREFETCH 命令の fcn 0–3 と 20–23 は以下のように動作する。
- 特権モードでも PREFETCH 命令は有効に作用する。
- どの fcn も fast_data_access/MMU_miss トラブルを発生しない。
- プリファッチはすべてキャッシュライン単位で行われるので、128 バイトアラインされた 128 バイトがプリファッチされる。
- fcn の特性の説明は Appendix A.49, "Prefetch Data" を参照。
- プリファッチが作用するのは ASI_PRIMARY, ASI_SECONDARY, ASI_NUCLEUS, ASI_PRIMARY_AS_IF_USER, ASI_SECONDARY_AS_IF_USER およびこれらのリトルエンディアン用の ASI に対してである。

104  VER.manuf
VER.manuf = 000416 である。最下位 8 ビットは富士通を示す JEDEC の製造者コードである。

105  TICK レジスタ
SPARC64 IXfx は TICK レジスタを 63 ビット幅で実装しており、各 CPU サイクルカウントアップする。

106  IMPDEPn 命令
SPARC64 IXfx は VIS1, VIS2 命令を実装するほか、独自の拡張命令を多数実装している。

107  Unimplemented LDD trap
SPARC64 IXfx は LDD をハードウェアで実装する。
### TABLE C-1  JPS1 実装依存仕様の SPARC64 IXfx での実装 (6 of 12)

<table>
<thead>
<tr>
<th>番号</th>
<th>SPARC64 IXfx での実装</th>
<th>ページ</th>
</tr>
</thead>
<tbody>
<tr>
<td>108</td>
<td>Unimplemented STD trap</td>
<td>—</td>
</tr>
<tr>
<td>109</td>
<td>LDDF_mem_address_not_aligned</td>
<td>81, 85</td>
</tr>
<tr>
<td></td>
<td>SPARC64 IXfx では、SIMD でない LDDF は、アクセスするアドレスが 8 バイト境界でない 4 バイト境界の場合、LDDF_mem_address_not_aligned 例外を通知する。システムソフトウェアはエミュレーションすること。SIMD の LDDF では LDDF_mem_address_not_aligned ではなく mem_address_not_aligned が通知される。</td>
<td></td>
</tr>
<tr>
<td>110</td>
<td>STDF_mem_address_not_aligned</td>
<td>100, 104</td>
</tr>
<tr>
<td></td>
<td>SPARC64 IXfx では、SIMD でない STDF は、アクセスするアドレスが 8 バイト境界でない 4 バイト境界の場合、STDF_mem_address_not_aligned 例外を通知する。システムソフトウェアはエミュレーションすること。SIMD の STDF では STDF_mem_address_not_aligned ではなく mem_address_not_aligned が通知される。</td>
<td></td>
</tr>
<tr>
<td>111</td>
<td>LDQF_mem_address_not_aligned</td>
<td>81, 85</td>
</tr>
<tr>
<td></td>
<td>SPARC64 IXfx は LDQF を実装していないので、illegal_instruction 例外を通知する。fp_disabled は検出しない。システムソフトウェアは LDQF をエミュレーションすること。</td>
<td></td>
</tr>
<tr>
<td>112</td>
<td>STQF_mem_address_not_aligned</td>
<td>100, 104</td>
</tr>
<tr>
<td></td>
<td>SPARC64 IXfx は STQF を実装していないので、illegal_instruction 例外を通知する。fp_disabled は検出しない。システムソフトウェアは STQF をエミュレーションすること。</td>
<td></td>
</tr>
<tr>
<td>113</td>
<td>実装しているメモリモデル</td>
<td>53</td>
</tr>
<tr>
<td></td>
<td>SPARC64 IXfx は PSTATE.MM のすべての値に対し Total Store Order (TSO) で動作する。</td>
<td></td>
</tr>
<tr>
<td>114</td>
<td>RED_state トラブルベクタアドレス (RSTVaddr)</td>
<td>43</td>
</tr>
<tr>
<td></td>
<td>SPARC64 IXfx では RSTVaddr は固定であり、 VA=FFFF FFFF F000 000016 PA=01FF F000 000016 である。</td>
<td></td>
</tr>
<tr>
<td>115</td>
<td>RED_state プロセッサステート</td>
<td>43</td>
</tr>
<tr>
<td></td>
<td>RED_state 時の動作は Section 7.1.1, “RED_state” を参照。</td>
<td></td>
</tr>
<tr>
<td>116</td>
<td>SIR_enable 制御フラグ</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>JPS1 の規定通り、SPARC64 IXfx では SIR_enable は存在せず、非特権モードでは NOP となる。</td>
<td></td>
</tr>
<tr>
<td>117</td>
<td>MMU 無効時のプリフェッチの動作</td>
<td>182</td>
</tr>
<tr>
<td></td>
<td>SPARC64 IXfx では、DMMU が無効のとき PREFETCH はメモリアクセスをせずに実行完了する。ノンフォールティングロードは、JPS1 Commonality の Section F.5 の定義通り data_access_exception が通知される。</td>
<td></td>
</tr>
<tr>
<td>118</td>
<td>I/O 領域の識別方法</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>この項目は本仕様書の記載範囲外である。SPARC64 IXfx を搭載するシステムの仕様書を参照。</td>
<td></td>
</tr>
</tbody>
</table>
### TABLE C-1  JPS1 実装依存仕様の SPARC64 IXfx での実装 (7 of 12)

<table>
<thead>
<tr>
<th>番号</th>
<th>SPARC64 IXfx での実装</th>
<th>ページ</th>
</tr>
</thead>
<tbody>
<tr>
<td>119</td>
<td>PSTATE.MM の未定義値</td>
<td>54</td>
</tr>
<tr>
<td></td>
<td>PSTATE.MM に 112 を指定すると、ITSO で動作する。しかしながら、112 は SPARC64 IXfx の将来のバージョンでは別の意味になるかもしれないため、使うべきではない。</td>
<td></td>
</tr>
<tr>
<td>120</td>
<td>プロセッサと I/O DMA 間のメモリ操作の同一性・原子性</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>この項目は本仕様書の記載範囲外である。SPARC64 IXfx を搭載するシステムの仕様書を参照。</td>
<td></td>
</tr>
<tr>
<td>121</td>
<td>実装依存メモリモデル</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>SPARC64 IXfx は E ビット（Volatile）がセットされているページに対するアクセスをプログラムオーダで処理する。</td>
<td></td>
</tr>
<tr>
<td>122</td>
<td>FLUSH レイテンシ</td>
<td>54</td>
</tr>
<tr>
<td></td>
<td>FLUSH 命令はプロセッサ内のキャッシュの状態を同期させるので、処理にかかる時間はプロセッサの状態に依存する。先行する命令がすべて完了しているとき、FLUSH 命令のレイテンシは 30CPU サイクルである。</td>
<td></td>
</tr>
<tr>
<td>123</td>
<td>I/O のセマンティクス</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>この項目は本仕様書の記載範囲外である。SPARC64 IXfx を搭載するシステムの仕様書を参照。</td>
<td></td>
</tr>
<tr>
<td>124</td>
<td>TL &gt; 0 のとき暗黙に使用される ASI</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>JPS1 の規定通り、TL &gt; 0 では PSTATE.CLE に応じて ASI_NUCLEUS または ASI_NUCLEUS_LITTLE が使われる。</td>
<td></td>
</tr>
<tr>
<td>125</td>
<td>アドレスマスク</td>
<td>42, 68, 80</td>
</tr>
<tr>
<td></td>
<td>PSTATE.AM = 1 のとき、SPARC64 IXfx は PC の上位 32 ビットをマスクしてデスティネーションレジスタに転送する。</td>
<td></td>
</tr>
<tr>
<td>126</td>
<td>レジスタウィンドウ関連レジスタのビット幅</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>SPARC64 IXfx では NWINDOWS は 8 である。よって、CWE, CANSAVE, CANRESTORE, OTHERWIN の有効ビットは 3 ビットである。これらのレジスタに NWINDOWS − 1 より大きな値を設定しようとすると、下 3 ビット以外を無視した値がセットされる。CLEANWIN レジスタは 3 ビットを保持する。</td>
<td></td>
</tr>
<tr>
<td>127–201</td>
<td>Reserved.</td>
<td></td>
</tr>
<tr>
<td>202</td>
<td>fast_ECC_error トラップ</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>SPARC64 IXfx は fast_ECC_error トラップを生成しない。</td>
<td></td>
</tr>
<tr>
<td>203</td>
<td>DCR のビット 13:6 とビット 1</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>SPARC64 IXfx は DCR を実装しない。</td>
<td></td>
</tr>
<tr>
<td>204</td>
<td>DCR のビット 5:3 とビット 0</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>SPARC64 IXfx は DCR を実装しない。</td>
<td></td>
</tr>
<tr>
<td>205</td>
<td>命令トラップレジスター</td>
<td>37</td>
</tr>
<tr>
<td></td>
<td>SPARC64 IXfx は命令トラップレジスターを JPS1 の定義通りに実装する。</td>
<td></td>
</tr>
<tr>
<td>206</td>
<td>SHUTDOWN 命令</td>
<td>99</td>
</tr>
<tr>
<td></td>
<td>SPARC64 IXfx は特権モードで SHUTDOWN 命令を NOP として実行する。</td>
<td></td>
</tr>
</tbody>
</table>
PCR のビット 47:32、ビット 26:17、ビット 3
SPARC64 IXfx ではこれらのビットには以下の機能が実装されている。
- ビット 47:32 は、オーバーフローの表示と設定 (OVF)。
- ビット 26 は OVF の書き込み制御 (OVRO)。
- ビット 24:22 はカウンタ数 (NC)。
- ビット 20:18 はカウンタ選択 (SC)。
- ビット 3 は SU/SL の書き込み制御 (ULRO)。
他の実装依存ビットは読み出しには 0 が返り、書き込みは無視される。

命令実行とエラーの通知順序
SPARC64 IXfx ではプログラム順にエラーが通知される。

ソフトウェアが介入する必要があるエラーのトラップ
SPARC64 IXfx では命令実行に同期して起こるエラーは precise で通知される。

ERROR 出力信号
この項目は本仕様書の記載範囲外である。SPARC64 IXfx システム制御仕様書を参照。

エラー情報表示レジスタの値
SPARC64 IXfx では致命的エラーの要因は ASI_STCHG_ERR_INFO レジスタに表示されない。

致命的エラーのトラップ
SPARC64 IXfx では致命的エラーはトラップを発生しない。

AFSR.PRIV
SPARC64 IXfx では AFSR.PRIV ビットを実装しない。

deferred トラブルの有効・無効制御
SPARC64 IXfx には deferred トラブルを制御する機能はない。

エラーバリア

data_access_error トラブルの精度
SPARC64 IXfx では data_access_error は常に precise である。

instruction_access_error トラブルの精度
SPARC64 IXfx では instruction_access_error は常に precise である。

async_data_error
SPARC64 IXfx は async_data_error トラブルを TT = 4016 で生成する。

AFSR
SPARC64 IXfx は AFAR を実装しない。

その他のエラー表示・制御レジスタ
SPARC64 IXfx は RAS 機能を充実させ高い信頼性を保証している。詳細は Appendix P を参照。

Special/signalling ECCs

<table>
<thead>
<tr>
<th>番号</th>
<th>SPARC64 IXfx での実装</th>
<th>ページ</th>
</tr>
</thead>
<tbody>
<tr>
<td>207</td>
<td>PCR のビット 47:32、ビット 26:17、ビット 3</td>
<td>26</td>
</tr>
<tr>
<td>208</td>
<td>命令実行とエラーの通知順序</td>
<td>257</td>
</tr>
<tr>
<td>209</td>
<td>ソフトウェアが介入する必要があるエラーのトラップ</td>
<td>—</td>
</tr>
<tr>
<td>210</td>
<td>ERROR 出力信号</td>
<td>—</td>
</tr>
<tr>
<td>211</td>
<td>エラー情報表示レジスタの値</td>
<td>274</td>
</tr>
<tr>
<td>212</td>
<td>致命的エラーのトラップ</td>
<td>274</td>
</tr>
<tr>
<td>213</td>
<td>AFSR.PRIV</td>
<td>287</td>
</tr>
<tr>
<td>214</td>
<td>deferred トラブルの有効・無効制御</td>
<td>—</td>
</tr>
<tr>
<td>215</td>
<td>エラーバリア</td>
<td>—</td>
</tr>
<tr>
<td>216</td>
<td>data_access_error トラブルの精度</td>
<td>—</td>
</tr>
<tr>
<td>217</td>
<td>instruction_access_error トラブルの精度</td>
<td>—</td>
</tr>
<tr>
<td>218</td>
<td>async_data_error</td>
<td>45, 257</td>
</tr>
<tr>
<td>219</td>
<td>AFSR</td>
<td>—</td>
</tr>
<tr>
<td>220</td>
<td>その他のエラー表示・制御レジスタ</td>
<td>257</td>
</tr>
<tr>
<td>221</td>
<td>Special/signalling ECCs</td>
<td>—</td>
</tr>
</tbody>
</table>
TLB構成
SPARC64 1XfxのTLB構成は以下のようになっている。
- レベル1マイクロiTLB(uITLB)、フルアソシエイティブ
- レベル1マイクロdTLB(uDTLB)、フルアソシエイティブ
- レベル2IMMU TLB—sITLB(セットアソシエイティブ)とfITLB(フルアソシエイティブ)からなる
- レベル2DMMU TLB—sDTLB(セットアソシエイティブ)とfDTLB(フルアソシエイティブ)からなる

TLBマルチヒット検出
SPARC64 1Xfxでは、マイクロTLBをミスしメインTLBにアクセスした場合のみマルチヒットを検出する。

MMU物理アドレスビット幅
SPARC64 1Xfxでは物理メモリのアドレス空間は41ビットである。TTEのPAフィールドは41ビットのみを格納し、PA<46:41>の読み出しは0が読み出され、書き込みは無視される。

TLBロックエントリ
ロックビットがセットされたTTEがTLB Data Inで書かれる場合、SPARC64 1XfxはそのエントリをfTLBに書き込みロックする。それ以外では、ページサイズに応じてsTLBかfTLBに書かれる。

TTE.CVビット
SPARC64 1XfxではどのTLBもCVを実装していない。SPARC64 1Xfxはハードウェアでキャッシュエイリアスを解消する機構を備えている。#232も参照。

TSBエントリ数
SPARC64 1Xfx仕様ではTSBをサポートしないので、この実装依存仕様は意味を持たない。

TSB、コンテキストIDどちらがTSB_Hashに使われるか
SPARC64 1Xfx仕様ではTSBをサポートしないので、この実装依存仕様は意味を持たない。

TSB_Baseアドレス生成
SPARC64 1Xfx仕様ではTSBをサポートしないので、この実装依存仕様は意味を持たない。

data_access_exceptionトラップ
SPARC64 1XfxはJPSI CommonalityのAppendix F.5で挙げられた要因でdata_access_exception例外を通知する。

MMU物理アドレスビット幅の変動
SPARC64 1Xfxでは物理アドレスのビット幅は41ビットで一定である。

DCUCRのCP,CVビット
SPARC64 1XfxはDCUCRのCP,CVビットを実装しない。impl.dep.#226も参照。
TABLE C-1  JPS1 実装依存仕様の SPARC64 IXfx での実装 (10 of 12)

<table>
<thead>
<tr>
<th>番号</th>
<th>SPARC64 IXfx での実装</th>
<th>ページ</th>
</tr>
</thead>
<tbody>
<tr>
<td>233</td>
<td>TSB_Hash フィールド</td>
<td>183</td>
</tr>
<tr>
<td></td>
<td>SPARC64 IXfx では TSB Extension レジスタを定義していないので、この実装依存仕様は意味を持たない。</td>
<td></td>
</tr>
<tr>
<td>234</td>
<td>TLB 入れ替えアルゴリズム</td>
<td>192</td>
</tr>
<tr>
<td></td>
<td>fTLB は擬似 LRU、sTLB は LRU である。</td>
<td></td>
</tr>
<tr>
<td>235</td>
<td>TLB データアクセスのアドレス割り当て</td>
<td>192</td>
</tr>
<tr>
<td></td>
<td>Appendix F.10.4 参照。</td>
<td></td>
</tr>
<tr>
<td>236</td>
<td>TSB_Size フィールド幅</td>
<td>193</td>
</tr>
<tr>
<td></td>
<td>SPARC64 IXfx では TSB_Size は 4 ビットで、bit&lt;3:0&gt; を占める。TSB_Size に書いた値は、読みだしによりその値のまま読みだされる。SPARC64 IXfx は値を保持するだけで、この値を何の計算にも使わない。</td>
<td></td>
</tr>
<tr>
<td>237</td>
<td>JMPL/RETURN で mem_address_not_aligned が起きたときの DSFAR/DSFSR</td>
<td>80, 178, 194</td>
</tr>
<tr>
<td></td>
<td>JMPL または RETURN で mem_address_not_aligned が起きた場合、D-SFAR, D-SFSR は更新されない。</td>
<td></td>
</tr>
<tr>
<td>238</td>
<td>TLB はラージページのページ内オフセットを保存するかどうか</td>
<td>176</td>
</tr>
<tr>
<td></td>
<td>SPARC64 IXfx ではページオフセットは書き込み時に捨てられ、読み出しでは不定データが読み出される。</td>
<td></td>
</tr>
<tr>
<td>239</td>
<td>ASI レジスタ 5516 または 5D16 のアクセス</td>
<td>183</td>
</tr>
<tr>
<td></td>
<td>SPARC64 IXfx では、IMMU ASI 5516, DMMU ASI 5D16 の V A&lt;63:18&gt; は無視される。</td>
<td></td>
</tr>
<tr>
<td>240</td>
<td>DCCSR のビット 47:41</td>
<td>34</td>
</tr>
<tr>
<td></td>
<td>SPARC64 IXfx はビット 41 を、メモリの投機アクセスを制御する WEAK_SPCA として実装する。</td>
<td></td>
</tr>
<tr>
<td>241</td>
<td>アドレスマスクと DSFAR</td>
<td>42, 68, 80</td>
</tr>
<tr>
<td></td>
<td>PSTATE.AM = 1 のとき、SPARC64 IXfx は DSFAR の上位 32 ビットに 0 を書く。</td>
<td></td>
</tr>
<tr>
<td>242</td>
<td>TLB ロックビット</td>
<td>176</td>
</tr>
<tr>
<td></td>
<td>fTTLB と sTTLB はロックビットをサポートする。sTTLB と sDTLB では 0 が読み出される。</td>
<td></td>
</tr>
<tr>
<td>243</td>
<td>インタラプト送信ステータスレジスタの BUSY/NACK の組</td>
<td>244</td>
</tr>
<tr>
<td></td>
<td>SPARC64 IXfx では 8 組の BUSY/NACK が実装される。</td>
<td></td>
</tr>
<tr>
<td>244</td>
<td>データウォッチポイントの信頼性</td>
<td>36</td>
</tr>
<tr>
<td></td>
<td>データウォッチポイントの信頼性を下げような機構は実装されていない。</td>
<td></td>
</tr>
<tr>
<td>245</td>
<td>命令キャッシュ内での CALL、分岐命令の飛び先の保存形式</td>
<td>37</td>
</tr>
<tr>
<td></td>
<td>SPARC64 IXfx では CALL、分岐 (BFcc, FBPfcc, Bicc, BPr) 命令の低位 11 ビットはメモリ内の形式通りに保存する。</td>
<td></td>
</tr>
<tr>
<td>246</td>
<td>インタラプト送信レジスタアクセス時の VA&lt;38:29&gt;</td>
<td>244</td>
</tr>
<tr>
<td></td>
<td>インタラプト送信レジスタにアクセスする際、SPARC64 IXfx は VA&lt;38:29&gt; の全 10 ビットを無視する。</td>
<td></td>
</tr>
<tr>
<td>247</td>
<td>インタラプト受信レジスタの SID フィールド</td>
<td>245</td>
</tr>
<tr>
<td></td>
<td>SID_H と SID_L の値は未定義である。</td>
<td></td>
</tr>
</tbody>
</table>
### TABLE C-1  JPS1 実装依存仕様の SPARC64 IXfx での実装 (11 of 12)

<table>
<thead>
<tr>
<th>番号</th>
<th>SPARC64 IXfx での実装</th>
<th>ページ</th>
</tr>
</thead>
<tbody>
<tr>
<td>248</td>
<td><strong>fp_exception_other が unfinished_FPop で通知される条件</strong>&lt;br&gt;SPARC64 IXfx は JPS1 Commonality Section 5.1.7 で既定された条件で unfinished_FPop の fp_exception_other を通知する。</td>
<td>23</td>
</tr>
<tr>
<td>249</td>
<td>パーシャルストアのデータウォッチポイント検出&lt;br&gt;SPARC64 IXfx ではウォッチポイントの検出は保守的に行われる。DCUCR のデータマスクは 0 かどうかだけが検査される。バイトマスク (r[rs2]) は無視され、マスクが 0 であっても (ストアが行われない場合でも) ウォッチポイントを検出する。</td>
<td>93</td>
</tr>
<tr>
<td>250</td>
<td><strong>PSTATE.PRIV = 0 で PCR のアクセスが可能かどうか</strong>&lt;br&gt;SPARC64 IXfx では、PSTATE.PRIV = 0 のとき PCR にアクセスできるかどうかは PCR.PRIV によって決まる。PSTATE.PRIV = 0 かつ PCR.PRIV = 1 のとき、RDFCR や WRPCR 命令を実行しようとすると、privileged_action が通知される。PSTATE.PRIV = 0 かつ PCR.PRIV = 0 のときは、RDFCR は正常に実行され、WRPCR は PCR.PRIV を変更しようとすると (つまり 1 を書き込むと) 場合のみ、privileged_action を通知する。</td>
<td>26, 28, 97</td>
</tr>
<tr>
<td>251</td>
<td>Reserved.</td>
<td>—</td>
</tr>
<tr>
<td>252</td>
<td><strong>DCUCR.DC (データキャッシュ有効/無効設定)</strong>&lt;br&gt;SPARC64 IXfx は DCUCR.DC を実装していない。</td>
<td>34</td>
</tr>
<tr>
<td>253</td>
<td><strong>DCUCR.IC (命令キャッシュ有効/無効設定)</strong>&lt;br&gt;SPARC64 IXfx は DCUCR.IC を実装していない。</td>
<td>34</td>
</tr>
<tr>
<td>254</td>
<td><strong>error_state から復旧する手段</strong>&lt;br&gt;通常の SPARC64 IXfx の動作では、error_state に遷移後 watchdog_reset (WDR) を発生しリセットする。しかし、OSPR による設定で、error_state に遷移後の状態にとどまるようにすることもできる。</td>
<td>44, 255</td>
</tr>
<tr>
<td>255</td>
<td><strong>LDDFA で ASI E016 または E116 を指定し、レジスタ番号の条件が間違っている場合</strong>&lt;br&gt;デスティネーションレジスタ番号を間違っていてもトラップは通知されない。</td>
<td>221</td>
</tr>
<tr>
<td>256</td>
<td><strong>LDDFA で ASI E016 または E116 を指定し、メモリアドレスのアラインメントが間違っている場合</strong>&lt;br&gt;SPARC64 IXfx は以下のよう動作する。&lt;br&gt;• 8 バイト境界なら、アラインメント例外は通知されず、&lt;br&gt;  data_access_exception が通知される。&lt;br&gt;• 4 バイト境界なら、LDDF_mem_address_not_aligned が通知される。&lt;br&gt;それ以外なら、mem_address_not_aligned が通知される。</td>
<td>221</td>
</tr>
<tr>
<td>257</td>
<td><strong>LDDFA で ASI C016–C516 または C816–CD16 を指定し、メモリアドレスのアラインメントが間違っている場合</strong>&lt;br&gt;SPARC64 IXfx は以下のよう動作する。&lt;br&gt;• 8 バイト境界なら、アラインメント例外は通知されず、&lt;br&gt;  data_access_exception が通知される。&lt;br&gt;• 4 バイト境界なら、LDDF_mem_address_not_aligned が通知される。&lt;br&gt;それ以外なら、mem_address_not_aligned が通知される。</td>
<td>221</td>
</tr>
<tr>
<td>番号</td>
<td>ASI_SERIAL_ID</td>
<td>SPARC64 IXfx での実装</td>
</tr>
<tr>
<td>------</td>
<td>---------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>258</td>
<td></td>
<td>SPARC64 IXfx では個々のプロセッサの識別コードを返す。</td>
</tr>
</tbody>
</table>
Formal Specification of the Memory Models

JPS1 Commonality の Appendix D を参照。
Appendix E は、JPS1 Commonality と HPC-ACE 拡張命令の定義を含む完全な情報を提供する。

表中、横棒（—）になっているオペコードは予約 Reserved されたオペコードである。予約されたオペコードを実行しようとすると、例外が通知される。詳細は JPS1 Commonality の Section 6.3.9, Reserved Opcodes and Instruction Fields を参照。

この章および Appendix A のオペコードには、右肩に記号が付されたものがある。この記号の意味は TABLE A-1 (58 ページ) に示してある。非推奨命令 (deprecated) については JPS1 Commonality の Section A.71, “Deprecated Instructions” を参照。

この章の表において、予約オペコード（—）と網掛けのエントリは SPARC64 IXfx では実装されていない。

**TABLE E-1**

<table>
<thead>
<tr>
<th>op 1:0</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
</table>
| 分岐と SETHI | CALL | 算術演算その他 | メモリアルアクセス | TABLE E-2 参照。
| 分岐と SETHI | CALL | 算術演算その他 | メモリアルアクセス | TABLE E-2 参照。

**TABLE E-2**

<table>
<thead>
<tr>
<th>op2 2:0</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>ILLTRAP</td>
<td>BPcc 参照</td>
<td>Bicc 参照</td>
<td>BPr 参照</td>
<td>SETHI</td>
<td>FBfcc 参照</td>
<td>FBfcc 参照</td>
<td>SXAR</td>
<td></td>
</tr>
<tr>
<td>ILLTRAP</td>
<td>BPcc 参照</td>
<td>Bicc 参照</td>
<td>BPr 参照</td>
<td>SETHI</td>
<td>FBfcc 参照</td>
<td>FBfcc 参照</td>
<td>SXAR</td>
<td></td>
</tr>
</tbody>
</table>

Ver 12, 2 Dec. 2013
F. Appendix E Opcode Maps 161
ILLTRAP を実行すると illegal_instruction 例外が通知される。

TABLE E-3  \( \text{op3}<5:0> \) (op = 2)

<table>
<thead>
<tr>
<th>op3&lt;3:0&gt;</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD</td>
<td>ADDcc</td>
<td>TADDcc</td>
<td>WRY(^D) (rd = 0)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(rd = 1)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>WRCCR (rd = 2)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>WRASI (rd = 3)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(rd = 4, 5)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>WRPPS (rd = 6)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>WRPCSE(^P) (rd = 16)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>WRPCSE(^P) (rd = 17)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>WRDCP(^P) (rd = 18)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>WRGR (rd = 19)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>WRSOFTINT_SET(^P) (rd = 20)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>WRSOFTINT_CLR(^P) (rd = 21)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>WRSOFTINT(^P) (rd = 22)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>WRTICK_CMPR(^P) (rd = 23)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>WRTICK_CMPR(^P) (rd = 24)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>WRSTICK(^P) (rd = 25)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>WRXAR (rd = 29)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>WRXASR (rd = 30)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>WRTXAR(^P) (rd = 31)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SIR (rd = 15, rs1 = 1, i = 1)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>op3&lt;3:0&gt;</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>AND</td>
<td>ANDcc</td>
<td>TSUBcc</td>
<td>SAVED(^P) (fcn = 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RESTORED(^P) (fcn = 1)</td>
</tr>
<tr>
<td>OR</td>
<td>ORcc</td>
<td>TADDccTV(^P)</td>
<td>( \text{FFop1 - TABLE E-5 参照。} )</td>
</tr>
<tr>
<td>XOR</td>
<td>XORcc</td>
<td>TSUBccTV(^P)</td>
<td>( \text{FFop2 - TABLE E-6 参照。} )</td>
</tr>
<tr>
<td>SUB</td>
<td>SUBcc</td>
<td>MULSc(^D)</td>
<td>( \text{IMPDEP1 (VIS) - TABLE E-12 および TABLE E-13 参照。} )</td>
</tr>
<tr>
<td>ANDN</td>
<td>ANDNcc</td>
<td>SLL (x = 0), SLLX (x = 1)</td>
<td>( \text{FFop2 - TABLE E-6 参照。} )</td>
</tr>
<tr>
<td>ORN</td>
<td>ORNcc</td>
<td>SRL (x = 0), SRLX (x = 1)</td>
<td>( \text{IMPDEP1 (VIS) - TABLE E-12 および TABLE E-13 参照。} )</td>
</tr>
<tr>
<td>XNOR</td>
<td>XNORcc</td>
<td>SRA (x = 0), SRAX (x = 1)</td>
<td>( \text{IMPDEP2 (FMADD/SUB, etc.) - TABLE E-14 参照。} )</td>
</tr>
</tbody>
</table>
### TABLE E-3

<table>
<thead>
<tr>
<th>op3&lt;5:0&gt;</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDC</td>
<td>ADDCc</td>
<td>RDI(^D) (rs1 = 0)</td>
<td>— (rs1 = 1)</td>
<td>JMPL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RDCCR (rs1 = 2)</td>
<td>RDASI (rs1 = 3)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>RDTICK(_{rs1}) (rs1 = 4)</td>
<td>RDPC (rs1 = 5)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>RDPFRS (rs1 = 6)</td>
<td>RDPCCR(_{rs1}) (rs1 = 16)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>RDPIC(_{rs1}) (rs1 = 17)</td>
<td>RDDCR(^P) (rs1 = 18)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>RDGSR (rs1 = 19)</td>
<td>RDSOFTINT(^P) (rs1 = 22)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>RDTICK(_{cmpr}) (rs1 = 23)</td>
<td>RDTICK(_{rep}) (rs1 = 24)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>RDXASP (rs1 = 30)</td>
<td>RDTXARP (rs1 = 31)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>MEMBAR (rs1 = 15, rd = 0, i = 1)</td>
<td>STBAR(^D) (rs1 = 15, rd = 0, i = 0)</td>
<td></td>
</tr>
</tbody>
</table>

| 8 | MULX | — | — | RETURN |
|   | UMUL\(^D\) | UMULCc\(^D\) | RDPR\(^P\) | Tcc — TABLE E-7 参照。 |
|   | SMUL\(^D\) | SMULCc\(^D\) | FLUSHW | FLUSH |
|   | SUBC | SUBCc | MOVcc | SAVE |
|   | UDIVX | — | SDIVX | RETRY\(^D\) (fcn = 1) |
|   | UDIV\(^D\) | UDIVCc\(^D\) | POPC (rs1 = 0) | DONE\(^P\) (fcn = 0) |
|   | — (rs1 > 0) | — | — | |

### TABLE E-4

<table>
<thead>
<tr>
<th>op3&lt;5:0&gt;</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>LDW</td>
<td>LDUNAP(^AS)</td>
<td>LDF</td>
<td>LDFA(^P)P(^AS)</td>
</tr>
<tr>
<td>1</td>
<td>LDUB</td>
<td>LDUBAP(^AS)</td>
<td>LDFFS(^D), LDXXSR</td>
<td>—</td>
</tr>
<tr>
<td>2</td>
<td>LDUN</td>
<td>LDUNAP(^AS)</td>
<td>LDSP</td>
<td>LDQF (^P)P(^AS)</td>
</tr>
<tr>
<td>3</td>
<td>LDD(^D)</td>
<td>LDDAP(^AS)</td>
<td>LDDF</td>
<td>LDDFP(^AS)</td>
</tr>
</tbody>
</table>

Ver 12, 2 Dec. 2013

F. Appendix E Opcode Maps 163
SPARC64™ IXfx では LDQF, LDQFA, STQF, STQFA と Reserved (—) オペコードを実行すると illegal_instruction 例外を通知する。

<table>
<thead>
<tr>
<th>opf&lt;8:3&gt;</th>
<th>opf&lt;2:0&gt;</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>016</td>
<td>FMOVd</td>
<td>FMOVq</td>
<td>—</td>
<td>—</td>
<td>FNEGd</td>
<td>FNEGq</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>026</td>
<td>—</td>
<td>FABSd</td>
<td>FABSq</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>036</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>046</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>056</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>066</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>076</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>086</td>
<td>—</td>
<td>FADDq</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>FSUBq</td>
<td>FSUBd</td>
<td>FSUBq</td>
<td></td>
</tr>
<tr>
<td>096</td>
<td>—</td>
<td>FMULq</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>FDIVq</td>
<td>FDIVd</td>
<td>FDIVq</td>
<td></td>
</tr>
</tbody>
</table>
TABLE E-6  opf<8:0> (op = 2, op3 = 3416 = FPop2)  (Continued)

<table>
<thead>
<tr>
<th>opf&lt;8:3&gt;</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0A16</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>0B16</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>0C16</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>0D16</td>
<td>—</td>
<td>FsMULd</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>0E16</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>0F16</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>1016</td>
<td>—</td>
<td>FsTOx</td>
<td>FdTOx</td>
<td>FqTOx</td>
<td>FxTOx</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>1116</td>
<td>FxTOd</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>FxTOq</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>1216</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>1316</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>1416</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>1516</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>1616</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>1716</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>1816</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>1916</td>
<td>FiTOd</td>
<td>FsTOd</td>
<td>—</td>
<td>FqTOd</td>
<td>FiTOq</td>
<td>FsTOq</td>
<td>FdTOq</td>
<td>—</td>
</tr>
<tr>
<td>1A16</td>
<td>—</td>
<td>FsTOi</td>
<td>FdTOi</td>
<td>FqTOi</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>1B16-3F16</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

網掛けの部分と Reserved（—）は、SPARC64 IIXf では fp_exception_other 例外を ftt = unimplemented_FPop で通知する。

TABLE E-5  opf<8:0> (op = 2, op3 = 3416 = FPop1)  (Continued)

<table>
<thead>
<tr>
<th>opf&lt;8:4&gt;</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8-F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0016</td>
<td>—</td>
<td>FMOVs (fcc0)</td>
<td>FMOVd (fcc0)</td>
<td>FMOVq (fcc0)</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>0116</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>0216</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>0316</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>0416</td>
<td>—</td>
<td>FMOVs (fcc1)</td>
<td>FMOVd (fcc1)</td>
<td>FMOVq (fcc1)</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

Ver 12, 2 Dec. 2013  F. Appendix E  Opcode Maps  165
### TABLE E-6  \(\text{opf}<8:0>\) (op = 2, op3 = 35) = FPop2)  (Continued)

| \(\text{opf}<8:4>\) | \(\text{opf}<3:0>\) | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8–F |
|---|---|---|---|---|---|---|---|---|---|
| 0516 | FCMPs | FCMPd | FCMPq |  |  |  |  |  |  |
| 0616 |  |  |  |  |  |  |  |  |  |
| 0716 |  |  |  |  |  |  |  |  |  |
| 0816 | FMOVa(fcc2) | FMOVd(fcc2) | FMOVq(fcc2) |  |  |  |  |  |  |
| 0916 |  |  |  |  |  |  |  |  |  |
| 0A16 |  |  |  |  |  |  |  |  |  |
| 0B16 |  |  |  |  |  |  |  |  |  |
| 0C16 | FMOVa(fcc3) | FMOVd(fcc3) | FMOVq(fcc3) |  |  |  |  |  |  |
| 0D16 |  |  |  |  |  |  |  |  |  |
| 0E16 |  |  |  |  |  |  |  |  |  |
| 0F16 |  |  |  |  |  |  |  |  |  |
| 1016 | FMOVa(icc) | FMOVd(icc) | FMOVq(icc) |  |  |  |  |  |  |
| 1116–1716 |  |  |  |  |  |  |  |  |  |
| 1816 | FMOVa(xcc) | FMOVd(xcc) | FMOVq(xcc) |  |  |  |  |  |  |
| 1916–1F16 |  |  |  |  |  |  |  |  |  |

†FMOVRに予約されたオペコード

網掛けの部分と Reserved (—) は、SPARC64 IXfX では fp_exception_other 例外を ftt = unimplemented_FPop で通知する。

### TABLE E-7  \(\text{cond}<3:0>\)

<table>
<thead>
<tr>
<th>(\text{cond}&lt;3:0&gt;)</th>
<th>BPcc</th>
<th>Bicc&lt;3:0&gt;</th>
<th>FBPFcc</th>
<th>FBfcc&lt;3:0&gt;</th>
<th>Tcc</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>BPN</td>
<td>BN&lt;3:0&gt;</td>
<td>FBPN</td>
<td>FBN&lt;3:0&gt;</td>
<td>TN</td>
</tr>
<tr>
<td>1</td>
<td>BPE</td>
<td>BE&lt;3:0&gt;</td>
<td>FBPN</td>
<td>FBNS&lt;3:0&gt;</td>
<td>TE</td>
</tr>
<tr>
<td>2</td>
<td>BPLE</td>
<td>BLE&lt;3:0&gt;</td>
<td>FBPLG</td>
<td>FBLG&lt;3:0&gt;</td>
<td>TLH</td>
</tr>
<tr>
<td>3</td>
<td>BPL</td>
<td>BL&lt;3:0&gt;</td>
<td>FBPU</td>
<td>FBUL&lt;3:0&gt;</td>
<td>TL</td>
</tr>
<tr>
<td>cond&lt;3:0&gt;</td>
<td>BPcc</td>
<td>Bicc&lt;0&gt;</td>
<td>FBPlc</td>
<td>FBicc&lt;0&gt;</td>
<td>Tcc</td>
</tr>
<tr>
<td>----------</td>
<td>------</td>
<td>--------</td>
<td>-------</td>
<td>----------</td>
<td>-----</td>
</tr>
<tr>
<td></td>
<td>op = 0</td>
<td>op = 0</td>
<td>op = 0</td>
<td>op = 0</td>
<td>op = 2</td>
</tr>
<tr>
<td></td>
<td>op2 = 1</td>
<td>op2 = 2</td>
<td>op2 = 5</td>
<td>op2 = 6</td>
<td>op3 = 3A16</td>
</tr>
<tr>
<td>4</td>
<td>BPLEU</td>
<td>BLEU&lt;0&gt;</td>
<td>FBPL</td>
<td>FBLEU&lt;0&gt;</td>
<td>TLEU</td>
</tr>
<tr>
<td>5</td>
<td>BPCS</td>
<td>BCS&lt;0&gt;</td>
<td>FBPUG</td>
<td>FBUCS&lt;0&gt;</td>
<td>TCS</td>
</tr>
<tr>
<td>6</td>
<td>BPNEG</td>
<td>BNEG&lt;0&gt;</td>
<td>FBPG</td>
<td>FBPG&lt;0&gt;</td>
<td>TNEG</td>
</tr>
<tr>
<td>7</td>
<td>BPVS</td>
<td>BVS&lt;0&gt;</td>
<td>FBPUG</td>
<td>FBUG&lt;0&gt;</td>
<td>TVS</td>
</tr>
<tr>
<td>8</td>
<td>BPA</td>
<td>BA&lt;0&gt;</td>
<td>FBPA</td>
<td>FBPA&lt;0&gt;</td>
<td>TA</td>
</tr>
<tr>
<td>9</td>
<td>BPNE</td>
<td>BNE&lt;0&gt;</td>
<td>FBPE</td>
<td>FBPE&lt;0&gt;</td>
<td>TNE</td>
</tr>
<tr>
<td>A</td>
<td>BPG</td>
<td>BG&lt;0&gt;</td>
<td>FBPU</td>
<td>FBPU&lt;0&gt;</td>
<td>TG</td>
</tr>
<tr>
<td>B</td>
<td>BPGE</td>
<td>BGE&lt;0&gt;</td>
<td>FBPGE</td>
<td>FPBGE&lt;0&gt;</td>
<td>TGE</td>
</tr>
<tr>
<td>C</td>
<td>BPGU</td>
<td>BGU&lt;0&gt;</td>
<td>FBPUGE</td>
<td>FBPGU&lt;0&gt;</td>
<td>TGU</td>
</tr>
<tr>
<td>D</td>
<td>BFCC</td>
<td>BCC&lt;0&gt;</td>
<td>FBPL</td>
<td>FBPL&lt;0&gt;</td>
<td>TCC</td>
</tr>
<tr>
<td>E</td>
<td>BPPOS</td>
<td>BPOS&lt;0&gt;</td>
<td>FBPlE</td>
<td>FBPUL&lt;0&gt;</td>
<td>TPOS</td>
</tr>
<tr>
<td>F</td>
<td>BPVC</td>
<td>BVc&lt;0&gt;</td>
<td>FBPO</td>
<td>FBPO&lt;0&gt;</td>
<td>TVC</td>
</tr>
</tbody>
</table>
TABLE E-8 命令ワードの rcond<2:0> のエンコーディング

<table>
<thead>
<tr>
<th>rcond &lt;2:0&gt;</th>
<th>BPr</th>
<th>MOVR</th>
<th>FMOVR</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>1</td>
<td>BRZ</td>
<td>MOVRZ</td>
<td>FMUVRZ</td>
</tr>
<tr>
<td>2</td>
<td>BRLZ</td>
<td>MOVRLZ</td>
<td>FMUVRLEZ</td>
</tr>
<tr>
<td>3</td>
<td>BRLZ</td>
<td>MOVRLZ</td>
<td>FMUVRLEZ</td>
</tr>
<tr>
<td>4</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>5</td>
<td>BRNZ</td>
<td>MOVNZ</td>
<td>FMUVRNZ</td>
</tr>
<tr>
<td>6</td>
<td>BRGZ</td>
<td>MOVGZ</td>
<td>FMUVRGZ</td>
</tr>
<tr>
<td>7</td>
<td>BRGEZ</td>
<td>MOVGEZ</td>
<td>FMUVRGEZ</td>
</tr>
</tbody>
</table>

TABLE E-9 cc / opf_cc フィールド (MOVcc および FMOVcc)

<table>
<thead>
<tr>
<th>cc2</th>
<th>cc1</th>
<th>cc0</th>
<th>選択される条件フィールド</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>fcc0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>fcc1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>fcc2</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>fcc3</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>icc</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>—</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>xcc</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>—</td>
</tr>
</tbody>
</table>
TABLE E-10  ccフィールド (FBFcc, FCMP および FCMPE)  

<table>
<thead>
<tr>
<th>cc1</th>
<th>cc0</th>
<th>選択される条件フィールド</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>fcc0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>fcc1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>fcc2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>fcc3</td>
</tr>
</tbody>
</table>

TABLE E-11  ccフィールド (BPcc および Tcc)  

<table>
<thead>
<tr>
<th>cc1</th>
<th>cc0</th>
<th>選択される条件フィールド</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>icc</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>—</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>xcc</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>—</td>
</tr>
</tbody>
</table>

TABLE E-12  IMPDEP1: VIS 命令 (op = 2, op3 = 3616) の opf<8:0>, ただし 0 ≤ opf<8:4> ≤ 7  

<table>
<thead>
<tr>
<th>opf&lt;3:0&gt;</th>
<th>00&lt;16&gt;</th>
<th>01&lt;16&gt;</th>
<th>02&lt;16&gt;</th>
<th>03&lt;16&gt;</th>
<th>04&lt;16&gt;</th>
<th>05&lt;16&gt;</th>
<th>06&lt;16&gt;</th>
<th>07&lt;16&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>0&lt;16</td>
<td>EDGE8</td>
<td>ARRAY8</td>
<td>FCMPLE16</td>
<td>—</td>
<td>—</td>
<td>FPADD16</td>
<td>FZERO</td>
<td>FAND</td>
</tr>
<tr>
<td>1&lt;16</td>
<td>EDGE8N</td>
<td>—</td>
<td>—</td>
<td>FMUL 8x16</td>
<td>—</td>
<td>FPADD16S</td>
<td>FZEROS</td>
<td>FANDS</td>
</tr>
<tr>
<td>2&lt;16</td>
<td>EDGE8L</td>
<td>ARRAY16</td>
<td>FCMPNE16</td>
<td>—</td>
<td>—</td>
<td>FPADD32</td>
<td>FNOR</td>
<td>FXNOR</td>
</tr>
<tr>
<td>3&lt;16</td>
<td>EDGE8LN</td>
<td>—</td>
<td>—</td>
<td>FMUL 8x16AU</td>
<td>—</td>
<td>FPADD32S</td>
<td>FNORS</td>
<td>FXNORS</td>
</tr>
<tr>
<td>4&lt;16</td>
<td>EDGE16</td>
<td>ARRAY32</td>
<td>FCMPLE32</td>
<td>—</td>
<td>—</td>
<td>FPSUB16</td>
<td>FANDNOT2</td>
<td>FSRC1</td>
</tr>
<tr>
<td>5&lt;16</td>
<td>EDGE16N</td>
<td>—</td>
<td>—</td>
<td>FMUL 8x16AL</td>
<td>—</td>
<td>FPSUB16S</td>
<td>FANDNOT2S</td>
<td>FSRC1S</td>
</tr>
<tr>
<td>6&lt;16</td>
<td>EDGE16L</td>
<td>—</td>
<td>FCMPNE32</td>
<td>FMUL 8SUx16</td>
<td>—</td>
<td>FPSUB32</td>
<td>FNOT2</td>
<td>FORNOT2</td>
</tr>
<tr>
<td>7&lt;16</td>
<td>EDGE16LN</td>
<td>—</td>
<td>—</td>
<td>FMUL 8ULx16</td>
<td>—</td>
<td>FPSUB32S</td>
<td>FNOT2S</td>
<td>FORNOT2S</td>
</tr>
</tbody>
</table>
### TABLE E-12  IMPDEP1: VIS 命令 (op = 2, op3 = 36_16) の opf<8:0>, ただし 0 ≤ opf<8:4> ≤ 7

<table>
<thead>
<tr>
<th>opf&lt;3:0&gt;</th>
<th>00_16</th>
<th>01_16</th>
<th>02_16</th>
<th>03_16</th>
<th>04_16</th>
<th>05_16</th>
<th>06_16</th>
<th>07_16</th>
</tr>
</thead>
<tbody>
<tr>
<td>8_16</td>
<td>EDGE32</td>
<td>ALIGN ADDRESS</td>
<td>FCMPT16</td>
<td>FMULD 8SUx16</td>
<td>FALIGNDATA</td>
<td>—</td>
<td>FANDNOT1</td>
<td>FSRC2</td>
</tr>
<tr>
<td>9_16</td>
<td>EDGE32N</td>
<td>BMASK</td>
<td>—</td>
<td>FMULD 8ULx16</td>
<td>—</td>
<td>—</td>
<td>FANDNOT1S</td>
<td>FSRC2S</td>
</tr>
<tr>
<td>A_16</td>
<td>EDGE32L</td>
<td>ALIGN ADDRESS</td>
<td>FCMPEQ16</td>
<td>FPACK32</td>
<td>—</td>
<td>—</td>
<td>FNOT1</td>
<td>FORNOT1</td>
</tr>
<tr>
<td>B_16</td>
<td>EDGE32LN</td>
<td>—</td>
<td>—</td>
<td>FPACK16</td>
<td>FPMERGE</td>
<td>—</td>
<td>FNOT1S</td>
<td>FORNOR1S</td>
</tr>
<tr>
<td>C_16</td>
<td>—</td>
<td>—</td>
<td>FCMPPGT12</td>
<td>—</td>
<td>BSHUFFLE</td>
<td>—</td>
<td>FXOR</td>
<td>FOR</td>
</tr>
<tr>
<td>D_16</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>FPACKFIX</td>
<td>FEXPAND</td>
<td>—</td>
<td>FXORS</td>
<td>FORS</td>
</tr>
<tr>
<td>E_16</td>
<td>—</td>
<td>—</td>
<td>FCMPEQ32</td>
<td>PDIST</td>
<td>—</td>
<td>—</td>
<td>FNAND</td>
<td>FONE</td>
</tr>
<tr>
<td>F_16</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>FNANDS</td>
<td>FONES</td>
</tr>
</tbody>
</table>

### TABLE E-13  IMPDEP1: VIS 命令 (op = 2, op3 = 36_16) の opf<8:0>, ただし 08_16 ≤ opf<8:4> ≤ 1F_16

<table>
<thead>
<tr>
<th>opf&lt;3:0&gt;</th>
<th>08_16</th>
<th>09_16至15_16</th>
<th>16_16</th>
<th>17_16</th>
<th>18_16至1F_16</th>
</tr>
</thead>
<tbody>
<tr>
<td>0_16</td>
<td>SHUTDOWN</td>
<td>—</td>
<td>FCMPEQd</td>
<td>FMAXd</td>
<td>—</td>
</tr>
<tr>
<td>1_16</td>
<td>SIAM</td>
<td>—</td>
<td>FCMPEQs</td>
<td>FMAXs</td>
<td>—</td>
</tr>
<tr>
<td>2_16</td>
<td>SUSPEND</td>
<td>—</td>
<td>FCMPEQEd</td>
<td>FMINd</td>
<td>—</td>
</tr>
<tr>
<td>3_16</td>
<td>SLEEP</td>
<td>—</td>
<td>FCMPEQEs</td>
<td>FMINs</td>
<td>—</td>
</tr>
<tr>
<td>4_16</td>
<td>—</td>
<td>—</td>
<td>FCMPLEEd</td>
<td>FRCPAD</td>
<td>—</td>
</tr>
<tr>
<td>5_16</td>
<td>—</td>
<td>—</td>
<td>FCMPLEEEd</td>
<td>FRCPAs</td>
<td>—</td>
</tr>
<tr>
<td>6_16</td>
<td>—</td>
<td>—</td>
<td>FCMLTED</td>
<td>FRSQRTAd</td>
<td>—</td>
</tr>
<tr>
<td>7_16</td>
<td>—</td>
<td>—</td>
<td>FCMLTEEs</td>
<td>FRSQRTAs</td>
<td>—</td>
</tr>
<tr>
<td>8_16</td>
<td>—</td>
<td>—</td>
<td>FCMPNED</td>
<td>FTRISSELD</td>
<td>—</td>
</tr>
<tr>
<td>9_16</td>
<td>—</td>
<td>—</td>
<td>FCMPNEs</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>A_16</td>
<td>—</td>
<td>—</td>
<td>FCMPNEEd</td>
<td>FTRISMULd</td>
<td>—</td>
</tr>
<tr>
<td>B_16</td>
<td>—</td>
<td>—</td>
<td>FCMPNEEs</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>C_16</td>
<td>—</td>
<td>—</td>
<td>FCMPGTEd</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>D_16</td>
<td>—</td>
<td>—</td>
<td>FCMPGTEs</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>opf&lt;3:0&gt;</td>
<td>08&lt;16</td>
<td>09&lt;16~15&lt;16</td>
<td>16&lt;16</td>
<td>17&lt;16</td>
<td>18&lt;16~1F&lt;16</td>
</tr>
<tr>
<td>-----------</td>
<td>--------</td>
<td>-------------</td>
<td>--------</td>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>E&lt;16</td>
<td>—</td>
<td>—</td>
<td>FCMPGEEd</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>r&lt;16</td>
<td>—</td>
<td>—</td>
<td>FCMPGEEs</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

**TABLE E-13** IMPDEP1: VIS 命令 (op = 2, op3 = 36<16>) の opf<8:0>，ただし 08<16 ≤ opf<8:4> ≤ 1F<16

<table>
<thead>
<tr>
<th>size</th>
<th>00&lt;02</th>
<th>01&lt;02</th>
<th>10&lt;02</th>
<th>11&lt;02</th>
</tr>
</thead>
<tbody>
<tr>
<td>00&lt;02</td>
<td>FPMADDX</td>
<td>FPMADDXHI</td>
<td>FTRIMADDd</td>
<td>PSELMOVd</td>
</tr>
<tr>
<td>01&lt;02</td>
<td>FMADDs</td>
<td>PMSUBs</td>
<td>FNMSUBs</td>
<td>FNADDs</td>
</tr>
<tr>
<td>10&lt;02</td>
<td>FMADDd</td>
<td>PMSUBd</td>
<td>FNMSUBd</td>
<td>FNADDd</td>
</tr>
<tr>
<td>11&lt;02</td>
<td></td>
<td></td>
<td></td>
<td>(4 倍精度演算に予約されている )</td>
</tr>
</tbody>
</table>

**TABLE E-14** IMPDEP2 (op = 2, op3 = 37<16)
本章では、SPARC64 IXfx の MMU の実装依存部の定義と、SPARC64 IXfx で拡張された機能について説明する。SPARC64 IXfx の MMU には JPS1 仕様と非互換な部分がある。詳細は以下を参照。

- Section F.4, “Hardware Support for TSB Access”
- Section F.10, “Internal Registers and ASI Operations”

### F.1 Virtual Address Translation

**IMPL. DEP. #222**: TLB organization is JPS1 implementation dependent.

SPARC64 IXfx の TLB は 2 階層構成となっている。

- レベル 1 マイクロ ITLB (uITLB)、フルアソシエイティブ
- レベル 1 マイクロ DTLB (uDTLB)、フルアソシエイティブ
- レベル 2 IMMU-TLB、これはセットウェイアソシエイティブの sITLB と、フルアソシエイティブの fITLB からなる。
- レベル 2 DMMU-TLB、これはセットウェイアソシエイティブの sDTLB と、フルアソシエイティブの fDTLB からなる。

**TABLE F-1** に SPARC64 IXfx の各 TLB の構成を示す。

uITLB と uDTLB は、それぞれ IMMU-TLB、DMMU-TLB の一時的な記憶領域として使われる。マイクロ TLB に対し、IMMU-TLB、DMMU-TLB をメイン TLB と呼ぶこともある。マイクロ TLB の内容はメイン TLB のサブセットとなっており、ハードウェアが同一性を保証する。
マイクロ TLB はソフトウェアから直接操作することはできず、また、ソフトウェアの動作が変わるような影響を与えることも、TLB の複数エントリがヒットした場合を除いて、ない。本仕様書ではマイクロ TLB についてはこれ以上触れらない。

**TABLE F-1** SPARC64 IXfx の TLB 構成

<table>
<thead>
<tr>
<th>項目</th>
<th>sITLB と sDTLB</th>
<th>dITLB と dDTLB</th>
</tr>
</thead>
<tbody>
<tr>
<td>エントリ数</td>
<td>2x128 (sITLB), 2x256 (sDTLB)</td>
<td>16</td>
</tr>
<tr>
<td>構成</td>
<td>2 ウェイセットアソシエイティブ</td>
<td>フルアソシエイティブ</td>
</tr>
<tr>
<td>エントリをロックできるか</td>
<td>できない</td>
<td>できる</td>
</tr>
<tr>
<td>ページサイズ</td>
<td>2 種類</td>
<td>すべてのページサイズ</td>
</tr>
</tbody>
</table>

**IMPL. DEP. #223**: Whether TLB multiple-hit detection is supported in a JPS1 processor is implementation dependent.

SPARC64 IXfx の MMU は多重ヒットを検出するが、検出できるのはメイン TLB でも fTLB 内で起こった多重ヒットのみである。fTLB に多重ヒットエントリがあるときでも、マイクロ TLB にヒットしている間は検出できない。詳細は Appendix F.5.2, “Behavior on TLB Error” (page 180) を参照。

**F.2 Translation Table Entry (TTE)**

変換テーブルエントリ (Translation Table Entry: TTE) は、論理アドレスである連続した領域 (ページ) から物理アドレスへの対応と、そのページの属性を保持するエントリである。TTE は、タグ部とデータ部という 2 つの 64 ビットのデータからなる。タグ部は検索用の情報が入っており、一致している場合はデータ部にあるページ情報がアドレス変換に使われる。

SPARC JPS1 では、TTE は TSB のエントリであり、TLB のエントリの操作も同じフォーマットで操作するという定義になっている。SPARC64 IXfx は TSB をハードウェアでサポートしていないが、TLB の操作はこのフォーマットで行うので、JPS1 Commonality の定義を FIGURE F-1, TABLE F-2 に示す。

**FIGURE F-1** 変換テーブルエントリ (TTE)
<table>
<thead>
<tr>
<th>ビット位置</th>
<th>フィールド名</th>
<th>説明</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag – 63</td>
<td>G</td>
<td>グローバル。G に 1 がセットされていると、TLB 検索時に Context フィールドは使われない。ある CPU 上で動作している、ユーザ・スーパバイザを問わずすべてのコンテキストが、G = 1 のエントリを共有することになる。ソフトウェアが TLB ミスを効率的に処理できるよう、G ビットは TTE のタグとデータの両方にある。</td>
</tr>
<tr>
<td>Tag – 60:48</td>
<td>Context</td>
<td>この TTE のコンテキスト番号。</td>
</tr>
<tr>
<td>Tag – 41:0</td>
<td>VA_tag</td>
<td>論理アドレスタグ。論理アドレスのページ番号。</td>
</tr>
<tr>
<td>Data – 63</td>
<td>V</td>
<td>バリッド。V に 1 がセットされていると、TTE の他のフィールドの値が意味を持つ。Note: V ビットがなくても、ソフトウェアのコンベンションとして使わない TTE を設定することで、無効なエントリを作ることができる。しかし、V ビットを持たせるほうが TLB ミスハンドラの処理が簡単になる。</td>
</tr>
<tr>
<td>Data – 62:61</td>
<td>Size</td>
<td>size2 と size をピット結合した値でページサイズを指示する。</td>
</tr>
<tr>
<td>Data – 58:50</td>
<td>Soft2</td>
<td>ソフトウェアが使用するフィールド。ハードウェアはこのフィールドを TLB 内で保持する必要はなく、TLB から読み出すと 0 が読めるかもしれない。</td>
</tr>
<tr>
<td>Data – 49</td>
<td>Reserved</td>
<td>Reserved。書き込みは無視され、0 が読み出される。</td>
</tr>
<tr>
<td>Data – 48</td>
<td>Size2</td>
<td>size の項目参照。</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Size2</th>
<th>Size&lt;1:0&gt;</th>
<th>Page Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>00000000</td>
<td>8 Kbyte</td>
</tr>
<tr>
<td>0001</td>
<td>01000000</td>
<td>64 Kbyte</td>
</tr>
<tr>
<td>0010</td>
<td>01000000</td>
<td>512 Kbyte</td>
</tr>
<tr>
<td>0011</td>
<td>10000000</td>
<td>4 Mbyte</td>
</tr>
<tr>
<td>0100</td>
<td>10000000</td>
<td>16 Mbyte</td>
</tr>
<tr>
<td>0101</td>
<td>10000000</td>
<td>256 Mbyte</td>
</tr>
<tr>
<td>1000</td>
<td>10000000</td>
<td>2 Gbyte</td>
</tr>
<tr>
<td>1001</td>
<td>11000000</td>
<td>4 Gbyte</td>
</tr>
</tbody>
</table>

Data – 60 | NFO | NFO (No Fault Only)。NFO に 1 がセットされているエントリは、ASI_PRIMARY_NO_FAULT, ASI_SECONDARY_NO_FAULT およびそれに _LITTLE がついた ASI を使ったときのみ変換される。それ以外のアクセスでは data_access_exception 例外が通知され、DSFSR_FT = 1010 がセットされる。IMMU の NFO ビットは、読み出し時には 0 が返り、書き込みは無視される。

iTLB のミスハンドラは、TLB にエントリを書き込む際、NFO = 1 ならばエラーを報告すべきである。 |

Data – 59 | IE | エンディアン反転。IE に 1 がセットされていると、そのページに対するアクセスでは、命令で指定したのは逆のエンティア ン (big に対し little、little に対し big) が使われる。詳細は JPS1 Commonality の Section F.7 を参照。IMMU の IE ビットは、読み出し時には 0 が返り、書き込みは無視される。Note: このビットは主に、ノンキャッシュプール領域に使われる。キャッシュプールで IE = 1 であるページへのアクセスが L1D キャッシュミスすると、性能に悪影響を与える可能性がある。 |

Data - 48:49 | Reserved | Reserved。書き込みは無視され、0 が読み出される。 |

| Size2 | size の項目参照。 |
TABLE F-2  TTE のビット説明（2 of 3）

<table>
<thead>
<tr>
<th>ビット位置</th>
<th>フィールド名</th>
<th>説明</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data – 47:41</td>
<td>Reserved</td>
<td><em>Reserved</em> - 書き込みは無視され、0 が読み出される。</td>
</tr>
<tr>
<td>Data – 40:13</td>
<td>PA</td>
<td>物理ページ番号。8KB より大きいページサイズのページ内オフセットにあたるビット (64KB ページなら PA&lt;15:13&gt;、512KB ページなら PA&lt;18:13&gt; など) は通常の変換では無視される。</td>
</tr>
<tr>
<td>SPARC64 IXfx は、JPS1 Commonality の定義と異なり、物理アドレスのビット数は 41 ビットまでをサポートする。（impl.dep.#224）</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TLB からエントリを読み出した際、ページ内オフセットにあたるビットに読み出される値は不定である。PA だけでなく VA についても、8KB より大きいページでページ内オフセットに当たるビットに読み出される値は不定である。（impl.dep.#238）</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data – 12:7</td>
<td>Soft</td>
<td>ソフトウェアが使用するフィールド。ハードウェアはこのフィールドを TLB 内で保持する必要はなく、TLB から読み出すと 0 が読めるかもしれません。</td>
</tr>
<tr>
<td>Data – 6</td>
<td>L</td>
<td>ロック。L に 1 がセットされているエントリが TLB に書き込まれると、そのエントリは TLB 上にロックされる。エントリが有効ならば、Data In レジスタによる TLB 書き込みによるリプレース対象とならない。無効なエントリでは L ビットは意味を持たない。ソフトウェアはロックしていないエントリを最低 1 つ残すようにしなければならない。</td>
</tr>
<tr>
<td>SPARC64 IXfx は TLB Data In を通じた書き込みではロックするかどうかを自動で判断する。書き込みが行われているエントリが TTE.L = 1 であるならば TLB に、そうでなければページサイズに応じて FTLB か sTLB に書き込まれる。（impl.dep.#225）</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPARC64 IXfx では、FTLB と sFTLB が lock ビットをサポートする。sFTLB, sDTLB では lock ビットは実装されておらず、lock ビットの書き込みは無視され、読み出しされは 0 が返る。（impl.dep.#242）</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data – 5</td>
<td>CP</td>
<td>物理アドレス・インデックスキャッシュでキャッシュプルかどうか (CP) と、論理アドレス・インデックスキャッシュでキャッシュプルかどうか (CV) を指示する。CP に 1 がセットされていると、そのページのデータは L1I, L1D, L2 キャッシュにキャッシュされる。</td>
</tr>
<tr>
<td>SPARC64 IXfx はハードウェアでキャッシュエイリアスを解消する機構を備えているので CV を実装していない。CV ビットの書き込みは無視され、読み出しされは 0 が返る。（impl.dep.#226）</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
TABLE F-2  TTE のビット説明 (3 of 3)

<table>
<thead>
<tr>
<th>ビット位置</th>
<th>フィールド名</th>
<th>説明</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data – 3</td>
<td>E</td>
<td>TTE. 副作用があることを示す。Eに1がセットされていると：</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ノンフォールティングロードには例外が通知される。</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ノンキャッシュ領域へのアクセスはストロングオーダで処理される。</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ノンキャッシュ領域へのストアはマージされない。</td>
</tr>
<tr>
<td></td>
<td></td>
<td>このビットは、IOデバイスの副作用があるレジスタをマッピングするときなどはセットする必要がある。</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IMMUではこのビットは、書き込みは無視され、読み出しには0が返る。</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Note: Eビットはノンキャッシュを意味しない。Eに1がセットされているときは、CPには通常0がセットされているはずだが、強制はされていない。CPにもEにも1がセットされているときの動作は未定義である。</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Note: EビットとNFOビットは同時に1に設定してはいけない。</td>
</tr>
<tr>
<td>Data – 2</td>
<td>P</td>
<td>特権アクセスビット。Pに1がセットされていると、そのTTEでマップされたページに特権モードでのみアクセス可能である。</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PSTATE.PRIV = 0 でアクセスすると、instruction_access_exception や data_access_exception 例外を通知し、そのときのISFSR.FTまたはDSFSR.FTには16がセットされる。</td>
</tr>
<tr>
<td>Data – 1</td>
<td>W</td>
<td>書き込み可能。Wに1がセットされていると、そのTTEでマップされたページのデータは書き換え可能である。0がセットされている場合、書き込むとするとfast_data_access_protection 例外が通知される。IMMUではこのビットは、書き込みは無視され、読み出しには0が返る。</td>
</tr>
<tr>
<td>Data – 0</td>
<td>G</td>
<td>グローバル。このビットの値はTTEタグのGビットと一致しなければならない。</td>
</tr>
</tbody>
</table>

F.4 Hardware Support for TSB Access

JPS1 Commonality では TSB を検索するのソフトウェアであり、ハードウェアは TLB ミス時に、ミスを起こしたVAがあると思われる TSB 上のポインタを計算するのみである。しかし、このポインタ計算は簡単な整数演算命令で行える簡単なものである。また、JPS1 Commonality の仕様では TSB のハードウェアサポートは、8KB ベージと 64KB ベージに対するサポートは一切ないことから、SPARC64 IXf では TSB のハードウェアサポートは実装していない。
しかしながら、SPARC64 IXfx では TSB Base レジスタは仕様に残してある。 TLB ミス発生時、システムソフトウェアは TSB の先頭アドレスをメモリから読むのではなく、TSB Base レジスタから取得することができるので、TLB ミス時処理オーバーヘッドはポイント計算数命令程度にとどまり、従来仕様の CPU とはほとんど変わらない。 TSB Base レジスタの詳細は Section F.10.6 を参照。

F.5 Faults and Traps

IMPL. DEP. #230: The cause of a data_access_exception trap is implementation dependent in JPS1, but there are several mandatory causes of a data_access_exception trap.

SPARC64 IXfx では data_access_exception を JPS1 Commonality の Section F.5 で定義された条件で通知する。ただし invalid ASI の場合は注意が必要である。詳細は Section F.10.9, “I/D Synchronous Fault Status Registers (I-SFSR, D-SFSR)” を参照。

IMPL. DEP. #237: Whether the fault status and/or address (DSFSR/DSFAR) are captured when mem_address_not_aligned is generated during a JMPL or RETURN instruction is implementation dependent.

SPARC64 IXfx では、JMPL または RETURN 命令で mem_address_not_aligned が通知される際、DSFSR/DSFAR は更新されない。

SPARC64 IXfx では、JPS1 Commonality の定義に加え、instruction_access_error と data_access_error, SIMD_load_across_pages が MMU により記録される。JPS1 Commonality の TABLE F-2 にこれらを加えたものを TABLE F-3 に示す。

<table>
<thead>
<tr>
<th>番号</th>
<th>トラップ名</th>
<th>トラップ要因</th>
<th>更新されるレジスタ</th>
<th>トラブル番号</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>fast_instruction_access_MMU_miss</td>
<td>I-TLB ミス</td>
<td>I-MMU Tag Access, D-MMU Tag Access</td>
<td>64_{16}–67_{16}</td>
</tr>
<tr>
<td>2.</td>
<td>instruction_access_exception</td>
<td>下記参照</td>
<td>I-MMU Tag Access, D-MMU Tag Access</td>
<td>08_{16}</td>
</tr>
<tr>
<td>3.</td>
<td>fast_data_access_MMU_miss</td>
<td>D-TLB ミス</td>
<td>X^3 X, X^4 X</td>
<td>68_{16}–6B_{16}</td>
</tr>
<tr>
<td>4.</td>
<td>data_access_exception</td>
<td>下記参照</td>
<td>D-MMU Tag Access</td>
<td>30_{16}</td>
</tr>
<tr>
<td>5.</td>
<td>fast_data_access_protection</td>
<td>書き込み保護違反</td>
<td>X^3 X</td>
<td>6C_{16}–6F_{16}</td>
</tr>
<tr>
<td>6.</td>
<td>privileged_action</td>
<td>特権 ASI 使用</td>
<td>X^3 X^4 X, X^4</td>
<td>37_{16}</td>
</tr>
<tr>
<td>7.</td>
<td>breakpoint</td>
<td>ウォッチポイント検出</td>
<td>X^3 X</td>
<td>61_{16}–62_{16}</td>
</tr>
<tr>
<td>8.</td>
<td>mem_address_not_aligned, *mem_address_not_aligned</td>
<td>アラインメント違反</td>
<td>(impl. dep #237)</td>
<td>35_{16}–36_{16}, 38_{16}–39_{16}</td>
</tr>
</tbody>
</table>

TABLE F-3 MMU 例外と ASI レジスタ更新
バスエラー、バスタイムアウトによる data_access_error は、優先順位 12 のトラップの中では一番優先順位が低い。

TABLE F-3

<table>
<thead>
<tr>
<th>番号</th>
<th>トラブル名</th>
<th>トラブル要因</th>
<th>I-MMU</th>
<th>D-MMU</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>instruction_access_error</td>
<td>下記参照</td>
<td>X^2</td>
<td>0A^16</td>
</tr>
<tr>
<td>10</td>
<td>data_access_error</td>
<td>下記参照</td>
<td>X^3</td>
<td>32^16</td>
</tr>
<tr>
<td>11</td>
<td>SIMD_load_across_pages</td>
<td>SIMD ロードのextend</td>
<td>X^3</td>
<td>77^16</td>
</tr>
</tbody>
</table>

---

1. TAG_ACCESS_EXT_REG を含む。
2. I-SFSR の詳細は Section F.10.9 を参照。
3. D-SFSR, D-SFAR の詳細は Section F.10.9 を参照。
4. data_access_exception 通知後、D-MMU Tag Access Register の context フィールドの内容は未定義。

パラエラー、バスタイムアウトによる data_access_error は、優先順位 12 のトラップの中では一番優先順位が低い。

TABLE F-3 の Ref #1~8 は JPS1 Commonality の Section F.5 の定義に準拠している。Ref #9, #10, #11 は以下の通り。

9. instruction_access_error — 以下の条件のうちどれか一つでも成立していれば例外が通知される。
   - 命令フェッチで訂正不可能なエラーが発見された場合。
   - 命令フェッチのメモリ参照にバスタイムアウトが通知された場合。
   - fITLB で多重ヒットが検出された場合。

10. data_access_error — 以下の条件のうちどれか一つでも成立していれば例外が通知される。
    - データアクセスで訂正不可能なエラーが発見された場合。
    - データアクセスのメモリ参照にバスタイムアウトが通知された場合。
    - fDTLB で多重ヒットが検出された場合。

Note — SPARC64 IXfx はストアバッファを実装しているので、あるアドレスの読み出しに対して data_access_error が通知されない場合がある。詳細は Appendix P.7.1, “ASI_ASYNC_FAULT_STATUS (ASI_AFSR)” (page 287) を参照。

11. SIMD_load_across_pages — SIMD ロードの extended 側で TLB ミスが起きた場合、この例外が通知される。DSPAR には extended 側のアドレスが表示される。

Ver 12, 2 Dec. 2013  F. Appendix F  Memory Management Unit  179
Programming Note – SIMD_load_across_pages が通知された場合、システムソフトウェアは TLB を更新するのではなくエミュレーションすべきである。TLB 更新処理は行う必要ないので、TAG_ACCESS_REG は更新されない。Section 7.6.5, “SPARC JPS1 Implementation-Dependent Traps” (page 51) も参照。

F.5.1 Trap Conditions for SIMD Load/Store

SIMD ロード/ストアの例外は、basic, extended を個々にこの順序で処理した場合に起こりうる例外が、TABLE 7-2 (page 48) の優先順位に従って通知される。DSFSR, DSFAR に表示される値は、basic 側で起きた例外では basic 側の情報が、extended 側で起きた例外では extended 側の情報が表示される。

Note – SIMD_load_across_pages は extended 側で起る例外である。

ただし VA_watchpoint については、basic 側の優先順位 12 の例外 (fast_data_MMU_miss, data_access_exception, fast_data_access_protection, data_access_error, data_access_protection) より、extended 側の VA_watchpoint が優先して通知される場合がある。

F.5.2 Behavior on TLB Error

SPARC64 IXfx は、fTLB の多重ヒットを検出すると data_access_error で通知するが、sTLB の多重ヒットは消去されソフトウェアには通知されない。TLB 検索時にバーティエラーが発見されると、そのエントリは消去 (sTLB) または自動訂正 (fTLB) され、ソフトウェアには通知されない。トラップはかならずプログラム順序で起こるが、消去または自動訂正は検出された時点で行われるので、投機実行されたメモリアクセスでも行われる。
SPARC64 IXfx で TLB でパリティエラー、多重ヒットが起きた場合の動作を TABLE F-4 に示す。

### TABLE F-4 パリティエラー、多重ヒット検出時の動作

<table>
<thead>
<tr>
<th>パリティエラー</th>
<th>多重ヒット</th>
<th>動作</th>
</tr>
</thead>
<tbody>
<tr>
<td>sTLB</td>
<td>fTLB</td>
<td>sTLB</td>
</tr>
<tr>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

1. fTLB は二重化されているので訂正可能。訂正できない場合は致命的エラー。
2. fTLB の多重ヒットは、常に検出できるとは限らない。
3. sTLB と fTLB の間で多重ヒットしている場合。

sTLB のパリティエラー、多重ヒットでは、エラーを起こしたエントリが消去される。この消去はソフトウェアには見えないが、基本側の TLB 検索により extended 側で必要な sTLB エントリのパリティエラー、多重ヒットが消去された場合、SIMD_load_across_pages 例外という形でソフトウェアに見える。

パリティエラー、多重ヒットの検出は、TABLE F-3 に示すすべての例外検出と同時に起こりうる。TLB エントリの消去は、他の例外を検出したかどうかとは無関係に行われる。つまり、投機実行によりパリティエラーや、多重ヒットが検出された場合でも、そのエントリは消去される。
F.8 Reset, Disable, and RED_state Behavior

**IMPL. DEP. #231**: The variability of the width of physical address is implementation dependent in JPS1, and if variable, the initial width of the physical address after reset is also implementation dependent in JPS1.

Table F-2 of the Data 部の PA 項を参照。SPARC64 IXfx の物理アドレスは 41 ビットである。

**IMPL. DEP. #232**: Whether CP and CV bits exist in the DCU Control Register is implementation dependent in JPS1.

SPIARC64 IXfx は DCU を実装していない (page 29)。CP, CV ビットは存在しない。

DMMU が無効なとき、MMU は TTE が以下の設定であるとみなして動作する。

- TTE.IE ← 0
- TTE.P ← 0
- TTE.W ← 1
- TTE.NFO ← 0
- TTE.CV ← 0
- TTE.CP ← 0
- TTE.E ← 1

**IMPL. DEP. #117**: Whether prefetch and nonfaulting loads always succeed when the MMU is disabled is implementation dependent.

SPARC64 IXfx では、DMMU が無効のとき PREFETCH はメモリアクセスをせずに実行完了する。ノンフォールティングロードは、JPS1 Commonality の Section F.5 の定義通り data_access_exception が通知される。
F.10 Internal Registers and ASI Operations

SPARC64 IXfx 仕様では TSB のハードウェアサポートは実装しない。このため JPS1 Commonality で定義されているレジスタのうち以下のものは実装されない。

<table>
<thead>
<tr>
<th>IMMU ASI</th>
<th>DMMU ASI</th>
<th>VA</th>
<th>レジスタ</th>
</tr>
</thead>
<tbody>
<tr>
<td>50_{16}</td>
<td>58_{16}</td>
<td>48_{16}</td>
<td>Instruction/Data TSB Primary Extension Registers</td>
</tr>
<tr>
<td>_</td>
<td>58_{16}</td>
<td>50_{16}</td>
<td>DATA TSB Secondary Extension Register</td>
</tr>
<tr>
<td>50_{16}</td>
<td>58_{16}</td>
<td>58_{16}</td>
<td>I/D TSB Nucleus Extension Registers</td>
</tr>
<tr>
<td>51_{16}</td>
<td>59_{16}</td>
<td>00_{16}</td>
<td>I/D TSB 8KB Pointer Registers</td>
</tr>
<tr>
<td>52_{16}</td>
<td>5A_{16}</td>
<td>00_{16}</td>
<td>I/D TSB 64KB Pointer Registers</td>
</tr>
<tr>
<td>_</td>
<td>5B_{16}</td>
<td>00_{16}</td>
<td>DATA TSB Direct Pointer Register</td>
</tr>
</tbody>
</table>

これらの ASI, VA にアクセスすると、data_access_exception 例外が通知される。

F.10.1 Accessing MMU Registers

**IMPL. DEP. #233:** Whether the TSB_Hash field is implemented in I/D Primary/Secondary/Nucleus TSB Extension Register is implementation dependent in JPS1.

SPARC64 IXfx では TSB Extension レジスタを定義していないので、この実装依存仕様は意味を持たない。

**IMPL. DEP. #239:** The register(s) accessed by IMMU ASI 55_{16} and DMMU ASI 5D_{16} at virtual addresses 40000_{16} to 60FF8_{16} are implementation dependent.

“/I.D TLB Data In, Data Access, and Tag Read Registers” (page 192) を参照。

SPARC64 IXfx では JPS1 Commonality の TABLE F-9 記載のレジスタ以外にも、ASI_DCUCR (page 34) と ASI_MCNTL (page 183) に MMU の制御機能が割り当てられて いる。
### ASI_MCNTL (Memory Control Register)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>レジスタ名</td>
<td>ASI_MCNTL</td>
</tr>
<tr>
<td>ASI</td>
<td>$45_{16}$</td>
</tr>
<tr>
<td>VA</td>
<td>$08_{16}$</td>
</tr>
<tr>
<td>アクセス種別</td>
<td>Supervisor read/write</td>
</tr>
<tr>
<td>ビット</td>
<td>フィールド名</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>63:20</td>
<td>Reserved</td>
</tr>
<tr>
<td>19</td>
<td>Reserved</td>
</tr>
<tr>
<td>18:17</td>
<td>hpf</td>
</tr>
<tr>
<td>16</td>
<td>NC_Cache</td>
</tr>
<tr>
<td>15</td>
<td>fw_fITLB</td>
</tr>
<tr>
<td>ビット</td>
<td>フィールド名</td>
</tr>
<tr>
<td>-------</td>
<td>---------------</td>
</tr>
</tbody>
</table>
| 14    | fw_fDTLB     | RW      | DTLB 更新時、必ずfDTLBに書き込む。fw_fDTLBに1がセットされていると、DTLB Data Inレジスタ経由のTLB書き込みは、必ずfDTLBに書かれる。fw_fDTLBはOBP向けの機能である。
| 13:12 | RMD          | R       | このフィールドは2に固定されており、書き込みは無視される。
| 11:8  | Reserved     |         | sITLBでマルチプルページサイズ機能を有効にする。
| 7     | mpg_sITLB1   | RW      | mpg_sITLB1に1がセットされていると、sITLBにはコ
| 6     | mpg_sDTLB1   | RW      | コンテキストごとに異なるページサイズのTTEを登録できる。
| 5:0   | Reserved     |         | mpg_sDTLB1に1がセットされていると、sDTLBにはコンテキストごとに異なるページサイズのTTEを登録できる。

1. mpg_sITLB = 1かつmpg_sDTLB = 0という設定をしないでください。この設定のときの動作は未定義。
F.10.2 Context Registers

sTLB は 2 つのセットアソシエイティブ TLB で構成されている。1st TLB, 2nd TLB ともに、ITLB は 128 エントリ、DTLB は 256 エントリである。デフォルトでは 1st sTLB は 8KB ページの TTE のみ、2nd sTLB は 4MB ページの TTE のみが登録されるが、MCNTL.mpg_sITLB, MCNTL.mpg_sDTLB を 1 にセットすることで、8 KB, 64 KB, 512 KB, 4 MB, 32 MB, 256 MB, 2 GB のうち任意の一つのページサイズの TTE が登録できる。ページサイズの設定は 1st sTLB, 2nd sTLB で独自に設定でき、両方を同じページサイズに設定することもできる。

ページサイズはコンテキストレジスタのフィールドで指定する。

ASI_PRIMARY_CONTEXT_REG では sITLB, sDTLB のページ設定が、ASI_SECONDARY_CONTEXT_REG では sDTLB の設定ができる。1st sTLB, 2nd sTLB のページサイズを同じものを設定した場合、sTLB は一つの 4 ウェイセットアソシエイティブ TLB と同じように動作する。

ページサイズ指定は以下のエンコードデータで行う。

- 00002 = 8 KB
- 00102 = 64 KB
- 01002 = 512 KB
- 01102 = 4 MB
- 10002 = 32 MB
- 10102 = 256 MB
- 11002 = 2 GB

Note – 11102 を指定したときの挙動は未定義。

JPS1 Commonality で定義されたコンテキストレジスタに加えて、SPARC64 IXfx では共有コンテキスト (Shared Context) が定義されている。共有コンテキストは、ある論理空間を複数のプロセスで共有するための仕組みで、命令列や共有データを置くのに使われる。あるコンテキストから別のコンテキストをアクセスするという点ではセカンドリコンテキストと似ているが、以下の点が異なる。

セカンドリコンテキスト空間にアクセスするためには、ASI 付きロード/ストア命令を使う必要があるが、共有コンテキスト空間はプライマリコンテキストと同じく、暗黙のうちにアクセス可能である。

セカンドリコンテキストがデータアクセスのみなので、共有コンテキスト空間は命令フェッチとデータアクセス両方で使われる。

以下の説明では実効コンテキストという用語が使われる。複数あるコンテキストレジスタのどの値を使うかは命令種類やプロセッサの状態により決まるので、実際に使用されるコンテキスト番号を指す用語として用いる。

- TL = 0 における命令フェッチやASI 付きではないロード/ストア命令によるアクセスでは、ASI_PRIMARY_CONTEXT の値を指す。
- TL > 0 における命令フェッチやASI 付きではないロード/ストア命令によるアクセスでは、ASI_NUCLEUS_CONTEXT の値を指す。
ASI 付きのロード/ストア命令によるアクセスでは、その ASI によって決まるコンテキストレジスタの値を指す。

ASI_PRIMARY_CONTEXT

<table>
<thead>
<tr>
<th>レジスタ名</th>
<th>ASI_PRIMARY_CONTEXT</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASI</td>
<td>5816</td>
</tr>
<tr>
<td>VA</td>
<td>0816</td>
</tr>
<tr>
<td>アクセス種別</td>
<td>Supervisor read/write</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ビット</th>
<th>フィールド名</th>
<th>アクセス</th>
<th>説明</th>
</tr>
</thead>
<tbody>
<tr>
<td>63:61</td>
<td>N_pgsz0</td>
<td>RW</td>
<td>Nucleus コンテキストが 1st sDTLB で使うページサイズ。</td>
</tr>
<tr>
<td>60:58</td>
<td>N_pgsz1</td>
<td>RW</td>
<td>Nucleus コンテキストが 2nd sDTLB で使うページサイズ。</td>
</tr>
<tr>
<td>55:53</td>
<td>N_lpsz0</td>
<td>RW</td>
<td>Nucleus コンテキストが 1st sITLB で使うページサイズ。</td>
</tr>
<tr>
<td>52:50</td>
<td>N_lpsz1</td>
<td>RW</td>
<td>Nucleus コンテキストが 2nd sITLB で使うページサイズ。</td>
</tr>
<tr>
<td>29:27</td>
<td>P_lpsz1</td>
<td>RW</td>
<td>プライマリコンテキストが 2nd sITLB で使うページサイズ。</td>
</tr>
<tr>
<td>26:24</td>
<td>P_lpsz0</td>
<td>RW</td>
<td>プライマリコンテキストが 1st sITLB で使うページサイズ。</td>
</tr>
<tr>
<td>21:19</td>
<td>P_pgsz1</td>
<td>RW</td>
<td>プライマリコンテキストが 2nd sDTLB で使うページサイズ。</td>
</tr>
<tr>
<td>18:16</td>
<td>P_pgsz0</td>
<td>RW</td>
<td>プライマリコンテキストが 1st sDTLB で使うページサイズ。</td>
</tr>
<tr>
<td>12:0</td>
<td>PContext</td>
<td>RW</td>
<td>プライマリコンテキスト番号。</td>
</tr>
</tbody>
</table>

ASI_PRIMARY_CONTEXT の各ページサイズフィールドは、ASI_MCNTL.mpg_sITLB,ASI_MCNTL.mpg_sDTLB の設定に係わらず読み出すことができる。

ASI_SECONDARY_CONTEXT

<table>
<thead>
<tr>
<th>レジスタ名</th>
<th>ASI_SECONDARY_CONTEXT</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASI</td>
<td>5816</td>
</tr>
<tr>
<td>VA</td>
<td>1016</td>
</tr>
<tr>
<td>アクセス種別</td>
<td>Supervisor read/write</td>
</tr>
</tbody>
</table>
各ページサイズフィールドは、ASI_MCNTL.mpg_sITLB, ASI_MCNTL.mpg_sDTLB の設定に係わらず読み出すことができる。

ASI_SHARED_CONTEXT

レジスタ名 ASI_SHARED_CONTEXT
ASI $58_{16}$
VA $68_{16}$
アクセス種別 Supervisor read/write
共有コンテキストは、実効コンテキストによるMMU変換時に、共有コンテキストによる変換を行うかどうか、またそのときのコンテキスト番号を指示するレジスタである。共有コンテキスト番号として0以外の値を指定し、IVまたはDVを1にセットしたときに有効になる。実効コンテキストが0のときは、IVまたはDVの値によらず共有コンテキストは使われない。例えば、TL > 0でASI_SHARED_CONTEXTを指定したロード命令を実行すると、SContextが実効コンテキストになるので、共有コンテキストが使われるかどうかはSContextが0かどうかで決まる。

共有コンテキストの機能は、ページサイズ指定を除き実効コンテキストと同じである。SPARC64 IxFxには2つのsITLBと2つのsDTLBがあり、格納されるTTEのページサイズを個々に設定することができる。しかし、共有コンテキストは独自のページサイズを持たず、実効コンテキストのページサイズが使われる。このため、ASI_MCNTRL.mpq_sI/DTLBが0のときは、1st sTLBは8KBページ、2nd sTLBは4MBページのエントリが書き込まれ、ASI_MCNTRL.mpq_sI/DTLBが1のときは、1st sTLBにはp_pgsz0/s_pgsz0/p_Ipgsz0で指定されたページサイズが、2nd sTLBにはp_pgsz0/s_pgsz0/p_Ipgsz0で指定されたページサイズが使われる。

Note – 実効コンテキストが0のときは共有コンテキストが無効なので、n_pgsz0/1は使われない。

Programming Note – 共有コンテキストでsTLBを効率よく使うためには、ある共有コンテキストを使う全てのコンテキストでp_pgsz(0,1)/p_Ipgsz(0,1)/s_pgsz(0,1)に同じページサイズを指定するとよい。
F.10.3 Instruction/Data MMU TLB Tag Access Registers

MMU ミスやアクセス権違反による例外が通知されるとき、共有コンテキストが有効な場合、I/D TLB Tag Access レジスタには実効コンテキストのコンテキスト番号が表示される。

**Programming Note –** TLB に共有コンテキストの TTE を書き込む場合、I/D TLB Tag Access レジスタに共有コンテキスト番号を設定してから、I/D TLB Data In/Data Access レジスタによる書き込みを行うこと。

ASI_I/DMMU_TAG_ACCESS_EXT

<table>
<thead>
<tr>
<th>レジスタ名</th>
<th>ASI_IMMU_TAG_ACCESS_EXT, ASI_DMMU_TAG_ACCESS_EXT</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASI</td>
<td>5016 (IMMU), 5816 (IMMU)</td>
</tr>
<tr>
<td>VA</td>
<td>6016</td>
</tr>
<tr>
<td>アクセス種別</td>
<td>Supervisor read/write</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>—</th>
<th>pgsz1</th>
<th>pgsz0</th>
<th>—</th>
</tr>
</thead>
<tbody>
<tr>
<td>63</td>
<td>21</td>
<td>19</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>15</td>
<td>0</td>
</tr>
</tbody>
</table>

MMU の例外によるトラップが発生する際、トラップの種類によっては、ハードウェア例を起こした論理アドレスとコンテキスト番号を Tag Access レジスタ (ASI_I/DMMU_TAG_ACCESS) にセーブする。詳細は TABLE F-3 (page 178) を参照。I/DTLB Data In レジスタで TLB に TTE を書き込む際の sTLB のインデックス計算を簡単にするため、SPARC64 IXfx は Tag Access レジスタで欠けている実効コンテキストのページサイズ情報を、ASI_I/DMMU_TAG_ACCESS_EXT レジスタにセーブする。

**Note –** 書き込まれる TTE のページサイズと ASI_I/DMMU_TAG_EXT.pgsz0/1 が異なる場合、その TTE は sTLB ではなく fTLB に書き込まれる。

*instruction_access_exception, data_access_exception* が通知された場合、ASI_I/DMMU_TAG_ACCESS_EXT レジスタは無効で、値は未定義である。また、ASI_MCNTL.mpg_sITLB が 0 のときの ASI_IMMU_TAG_ACCESS_EXT レジスタ、ASI_MCNTL.mpg_sDTLB が 0 のときの ASI_DMMU_TAG_ACCESS_EXT レジスタは無効で、値は未定義である。
F.10.4 I/D TLB Data In, Data Access, and Tag Read Registers

**IMPL. DEP. #234**: The replacement algorithm of a TLB entry is implementation dependent in JPS1.

fTLB is pseudo-LRU, sTLB is LRU by being evicted out.

**IMPL. DEP. #235**: The MMU TLB data access address assignment and the purpose of the address are implementation dependent in JPS1.

SPARC64 IXfx の I/D TLB Data Access レジスタのアドレス割り当てを TABLE F-6 に示す。

### TABLE F-6 TLB Data Access レジスタのアドレス割り当て

<table>
<thead>
<tr>
<th>ビット</th>
<th>フィールド名</th>
<th>アクセス</th>
<th>説明</th>
</tr>
</thead>
<tbody>
<tr>
<td>17:16</td>
<td>TLB#</td>
<td>RW</td>
<td>アクセスする TLB を指定する。</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>00b2: fTLB (16 エントリ)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>01b2: Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>10b2: sTLB (IMMU では 256 エントリ、DMMU では 512 エントリ)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>11b2: Reserved</td>
</tr>
<tr>
<td>15</td>
<td></td>
<td>Reserved</td>
<td>TLB のインデクス番号。</td>
</tr>
<tr>
<td>13:3</td>
<td>TLB index</td>
<td>RW</td>
<td>fTLB の場合は、下位 4 ビットがインデクス番号で、上位 7 ビットは無視される。下位 4 ビットの値と TLB インデクスの関係は：</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0-15: fTLB のインデクス番号</td>
</tr>
</tbody>
</table>

- sTLB の場合は、bit<13:12> でウェイ、bit<8:3> でインデクスを表わし、bit<11:9> は無視される。値と TLB インデクスの関係は：
  - 0-63: 1st sTLB のウェイ 0 のインデクス番号
  - 512-575: 1st sTLB のウェイ 1 のインデクス番号
  - 1024-1087: 2nd sTLB のウェイ 0 のインデクス番号
  - 1536-1599: 2nd sTLB のウェイ 1 のインデクス番号

- sDTLB の場合は、bit<13:12> でウェイ、bit<9:3> でインデクスを表わし、bit<11:10> は無視される。値と TLB インデクスの関係は：
  - 0-127: 1st sDTLB のウェイ 0 のインデクス番号
  - 512-639: 1st sDTLB のウェイ 1 のインデクス番号
  - 1024-1151: 2nd sDTLB のウェイ 0 のインデクス番号
  - 1536-1663: 2nd sDTLB のウェイ 1 のインデクス番号
Note – I/D TLB Data In による TLB 書き込みでは、TTE.G = 1 のエントリは fTLB に書かれる。

I/D MMU TLB Tag Read Register

IMPL. DEP. #238: When read, an implementation will return either 0 or the value previously written to them.

TABLE F-2 (page 175) の PA の項参照。

TLB Tag Read レジスタの VA フォーマットは、TLB Data Access レジスタと同じである。詳細は TABLE F-6 を参照。

I/D MMU TLB Tag Access Register

しかし、TTE.V = 0 であるエントリを I/D TLB Data Access で書き込むときは、整合性の検査は行われず書き込まれる。これにより TLB の特定エントリだけを削除することができる。この機能は、ソフトウェアによりエラーを起こしている TLB エントリのエラーを消去をする際に使うことができる。

I/D TLB Data Access レジスタ、I/D TLB Data In レジスタで TLB に TTE を書き込む際、ハードウェアは I/D TLB Tag Access レジスタの内容との整合性を検査し、不整合な場合は TLB は更新されない。

Implementation Note – TTE.V = 0 であるエントリを書き込み、それを読み出すと、全ビット 0 のデータが返る。

F.10.6 I/D TSB Base Registers

<table>
<thead>
<tr>
<th>TSB_Base&lt;63:13&gt;</th>
<th>Reserved</th>
<th>TSB size</th>
</tr>
</thead>
<tbody>
<tr>
<td>63 13 12 4 3 0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

SPARC64 IXfx は TSB のハードウェアサポートはないが、システムソフトウェアが TSB を扱えるよう TSB Base レジスタ仕様は残してある。JPS1 Commonality では TSB Base レジスタには以下のフィールドがある。

- TSB_Base
- Split
- TSB_Size
SPARC64 IXfx の TSB Base レジスタはこのうち TSB_Base と TSB_Size を実装し、
Split は Reserved とする。

TSB_size は bit<3:0> の 4 ビットとする (impl.dep. #236)。TSB_Size に書いた値は、読み
だしによりその値のまま読みだされる。ハードウェアは値を保持するだけで、この値
を何の計算にも使わない。

F.10.7 I/D TSB Extension Registers

SPARC64 IXfx は TSB Extension レジスタをサポートしない。読み出し、書き込みを
しようとするとき data_access_exception が通知される。

F.10.8 I/D TSB 8-Kbyte and 64-Kbyte Pointer and Direct
Pointer Registers

SPARC64 IXfx はこれらのレジスタをサポートしない。読み出し、書き込みをしよう
とするとき data_access_exception が通知される。

F.10.9 I/D Synchronous Fault Status Registers (I-SFSR, D-
SFSR)

<table>
<thead>
<tr>
<th>TLB #</th>
<th>Reserved</th>
<th>index</th>
<th>Reserved</th>
<th>MK</th>
<th>EID</th>
<th>UE</th>
<th>BERR</th>
<th>Reserved</th>
<th>mTLB</th>
<th>NC</th>
</tr>
</thead>
<tbody>
<tr>
<td>63</td>
<td>62</td>
<td>61</td>
<td>60</td>
<td>59</td>
<td>49</td>
<td>48</td>
<td>47</td>
<td>46</td>
<td>45</td>
<td>32</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NF</th>
<th>ASI</th>
<th>TM</th>
<th>Reserved</th>
<th>FT</th>
<th>E</th>
<th>CT</th>
<th>PR</th>
<th>W</th>
<th>OW</th>
<th>FV</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>23</td>
<td>16</td>
<td>15</td>
<td>14</td>
<td>13</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
</tr>
</tbody>
</table>

FIGURE F-2 MMU I/D Synchronous Fault Status Registers (I-SFSR, D-SFSR)
JPS1 Commonality では、I/D-SFSR のビット 63–25 は実装依存となっている。SPARC64 IXf の I-SFSR の実装依存ビットの定義を TABLE F-7 に D-SFSR の実装依存ビットの定義を TABLE F-10 に示す。ビット 24–0 は JPS1 Commonality に準拠する。

### TABLE F-7  I-SFSR のフィールドの説明 (1 of 2)

<table>
<thead>
<tr>
<th>ビット</th>
<th>フィールド名</th>
<th>アクセス</th>
<th>説明</th>
</tr>
</thead>
<tbody>
<tr>
<td>63:62</td>
<td>TLB#</td>
<td>RW</td>
<td>mITLB でエラーが起こったことを示す。SPARC64 IXfx では 0002 が表示される。</td>
</tr>
<tr>
<td>59:49</td>
<td>index</td>
<td>RW</td>
<td>mITLB でエラーが起きた場合のインデックス番号を示す。</td>
</tr>
<tr>
<td>46</td>
<td>MK</td>
<td>RW</td>
<td>マーク済の訂正不能エラー。SPARC64 IXfx では、すべての訂正不能エラーはエラーマークして報告される。I-SFSR.UE が 1 のとき、MK には常に 1 がセットされる。詳細は Appendix P.2.4, “キャッシュブルデータのエラーマーキング” (page 269) 参照。</td>
</tr>
<tr>
<td>45:32</td>
<td>EID</td>
<td>RW</td>
<td>エラーマーク ID。このフィールドは MK が 1 のとき有効。詳細は Appendix P.2.4, “キャッシュブルデータのエラーマーキング” (page 269) 参照。</td>
</tr>
<tr>
<td>31</td>
<td>UE</td>
<td>RW</td>
<td>訂正不能エラー (Uncorrectable Error:UE)。UE に 1 がセットされているとき、フェッチした命令列の中に訂正不能エラーがあったことを示す。このフィールドは instruction_access_error が通知されたときのみ有効。</td>
</tr>
<tr>
<td>30</td>
<td>BERR</td>
<td>RW</td>
<td>命令フェッチにメモリバスエラーが返されたことを示す。このフィールドは instruction_access_error が通知されたときのみ有効。</td>
</tr>
<tr>
<td>29</td>
<td>BRTO</td>
<td>RW</td>
<td>命令フェッチにバスタイムアウトが返されたことを示す。このフィールドは instruction_access_error が通知されたときのみ有効。</td>
</tr>
<tr>
<td>27:26</td>
<td>mITLB&lt;1:0&gt;</td>
<td>RW</td>
<td>mITLB のエラーステータス。mITLB の検索時に多重ヒットが起きたときは mITLB&lt;1&gt; に 1 がセットされる。mITLB&lt;0&gt; は常に 0。このフィールドは instruction_access_error が通知されたときのみ有効。</td>
</tr>
<tr>
<td>25</td>
<td>NC</td>
<td>RW</td>
<td>ノンキャッシュブル領域を参照したことを示す。このフィールドは訂正不能エラー、バスタイムアウトのいずれかによる instruction_access_error が通知されたときのみ有効。それ以外の例外通知時は、このフィールドの値は不定である。</td>
</tr>
<tr>
<td>23:16</td>
<td>ASI&lt;7:0&gt;</td>
<td>RW</td>
<td>例外が起きた際のアクセスに使われた ASI 番号が表示される。このフィールドは ISFSR.FV に 1 がセットされているときのみ有効。表示される ASI 番号は、TL = 0 のときは 8016(ASI_PRIMARY), TL &gt; 0 のときは 0416 (ASI_NUCLEUS) である。</td>
</tr>
<tr>
<td>15</td>
<td>TM</td>
<td>RW</td>
<td>命令フェッチ中に TLB ミスが起きたことを示す。</td>
</tr>
</tbody>
</table>
TABLE F-7  1-SFSR のフィールド説明 (2 of 2)
<table>
<thead>
<tr>
<th>ビット</th>
<th>フィールド名</th>
<th>アクセス</th>
<th>説明</th>
</tr>
</thead>
<tbody>
<tr>
<td>13:7</td>
<td>FT&lt;6:0&gt;</td>
<td>RW</td>
<td>例外発生の原因をエンコード情報で示す。エンコードの意味は TABLE F-8 を参照。このフィールドは instruction_access_exception が通知されたときのみ有効。fast_instruction_access_MMU_miss では常に 0 が読み出され、instruction_access_exception では常に 0116 が読み出される。</td>
</tr>
<tr>
<td>5:4</td>
<td>CT&lt;1:0&gt;</td>
<td>RW</td>
<td>例外を起こした命令フェッチのコンテキストに関する情報が表示される。</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0002: Primary</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0102: Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1002: Nucleus</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1102: Reserved</td>
</tr>
</tbody>
</table>

Translated ASI でないか、または無効な ASI の場合、1102 が表示される。

**Note:** 共有コンテキストで起きたことを示す手段は定義されていない。共有コンテキストが関係する多重ヒットのときは、実効コンテキストの情報が表示される。

<table>
<thead>
<tr>
<th>ビット</th>
<th>PR</th>
<th>RW</th>
<th>特権モードで命令フェッチ中に例外が通知されたことを示す。このフィールドは FV が 1 のときにのみ有効。</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>OW</td>
<td>RW</td>
<td>ISFSR.FV = 1 のときに例外が通知されたことを示す。例外通知時点で ISFSR.FV = 1 だと 1 がセットされ、ISFSR.FV = 0 だと 0 がセットされる。</td>
</tr>
<tr>
<td>0</td>
<td>FV</td>
<td>RW</td>
<td>IMMU で TLB ミス以外の例外が発生したときに 1 がセットされる。このフィールドが 0 のとき、他のフィールドは意味を持たない（ただし MMU ミスの場合を除く）。</td>
</tr>
</tbody>
</table>

ISFSR.FT のエンコーディングを TABLE F-8 に示す。

TABLE F-8  1-SFSR.FT のエンコーディング
<table>
<thead>
<tr>
<th>FT&lt;6:0&gt;</th>
<th>例外発生日由</th>
</tr>
</thead>
<tbody>
<tr>
<td>0116</td>
<td>特権アクセス違反。命令フェッチ時、TTE.F = 1 かつ PSTATE.PRIV = 0 だったことを示す。特権アクセス違反は instruction_access_exception 例外で通知される。</td>
</tr>
<tr>
<td>0216</td>
<td>Reserved</td>
</tr>
<tr>
<td>0416</td>
<td>Reserved</td>
</tr>
<tr>
<td>0816</td>
<td>Reserved</td>
</tr>
<tr>
<td>1016</td>
<td>Reserved</td>
</tr>
<tr>
<td>2016</td>
<td>Reserved</td>
</tr>
<tr>
<td>4016</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
I-SFSR は fast_instruction_access_MMU_miss, instruction_access_exception, instruction_access_error のいずれかが通知された際に更新される。TABLE F-9 に各例外によりどのフィールドが更新されるかを示す。

<table>
<thead>
<tr>
<th>フィールド</th>
<th>TLB#, index</th>
<th>FV</th>
<th>OW</th>
<th>PR, CT</th>
<th>FT</th>
<th>TM</th>
<th>ASI</th>
<th>UE, BRR, BRTOS, mITLB, NC</th>
</tr>
</thead>
<tbody>
<tr>
<td>I-SFSR.OW = 0 のとき</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Miss:</td>
<td>fast_instruction_access_MMU_miss</td>
<td>—</td>
<td>0</td>
<td>0</td>
<td>V</td>
<td>—</td>
<td>1</td>
<td>—</td>
</tr>
<tr>
<td>Exception:</td>
<td>instruction_access_exception</td>
<td>—</td>
<td>1</td>
<td>0</td>
<td>V</td>
<td>V</td>
<td>0</td>
<td>V</td>
</tr>
<tr>
<td>Error:</td>
<td>instruction_access_error</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I-SFSR.OW = 1 のとき</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Miss:</td>
<td>fast_instruction_access_MMU_miss</td>
<td>—</td>
<td>0</td>
<td>0</td>
<td>V</td>
<td>—</td>
<td>1</td>
<td>—</td>
</tr>
<tr>
<td>Exception:</td>
<td>instruction_access_exception</td>
<td>—</td>
<td>1</td>
<td>0</td>
<td>V</td>
<td>V</td>
<td>0</td>
<td>V</td>
</tr>
<tr>
<td>Error:</td>
<td>instruction_access_error</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TABLE F-10  D-SFSR フィールドの説明 (1 of 3)

ビット | フィールド名 | アクセス | 説明
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>63:62</td>
<td>TLB#</td>
<td>RW</td>
<td>mDTLB でエラーが起きたことを示す。SPARC64 Ix fx では 0002 が表示される。</td>
</tr>
<tr>
<td>59:49</td>
<td>index</td>
<td>RW</td>
<td>mDTLB でエラーが起きた場合のインデックス番号を示す。</td>
</tr>
</tbody>
</table>
| 46 | MK | RW | マーク済の訂正不能エラー。SPARC64 Ix fx では、すべての訂正不能エラーはエラーマークして報告される。DSFSR.UE が 1 のとき、MK 内に常 に 1 がセットされる。詳細は Appendix P.2.4, “キャッシュ済データのエラーマーク” (page 269) 参照。
### TABLE F-10  D-SFSR フィールドの説明 (2 of 3)

<table>
<thead>
<tr>
<th>ビット</th>
<th>フィールド名</th>
<th>アクセス</th>
<th>説明</th>
</tr>
</thead>
<tbody>
<tr>
<td>45:32</td>
<td>EID</td>
<td>RW</td>
<td>エラーマーケット ID。このフィールドは MK が 1 のとき有効。詳細は Appendix P.2.4, &quot;キャッシュブルデータのエラーマーキング&quot; (page 269) 参照。</td>
</tr>
<tr>
<td>31</td>
<td>UE</td>
<td>RW</td>
<td>訂正不能エラー (Uncorrectable Error:UE)。UE に 1 がセットされていると、アクセスデータの中に訂正不能エラーがあったことを示す。このフィールドは data_access_error が通知されたときのみ有効。</td>
</tr>
<tr>
<td>30</td>
<td>BERR</td>
<td>RW</td>
<td>データアクセスにメモリベースエラーが返されたことを示す。このフィールドは data_access_error が通知されたときのみ有効。</td>
</tr>
<tr>
<td>29</td>
<td>BRTO</td>
<td>RW</td>
<td>データアクセスにバスタイムアウトが返されたことを示す。このフィールドは data_access_error が通知されたときのみ有効。</td>
</tr>
<tr>
<td>27:26</td>
<td>mDTLB&lt;1:0&gt;</td>
<td>RW</td>
<td>mDTLB のエラーステータス。mDTLB の検索時に多重ヒットが起きたときは mDTLB&lt;1&gt; がセットされる。mDTLB&lt;0&gt; は常に 0。このフィールドは data_access_error が通知されたときのみ有効。</td>
</tr>
<tr>
<td>25</td>
<td>NC</td>
<td>RW</td>
<td>ノンキャッシュブル領域を参照したことを示す。このフィールドは訂正不能エラー、バスエラー、バスタイムアウトのいずれかによる data_access_error が通知されたときのみ有効。それ以外の例外通知時は、このフィールドの値は不定である。</td>
</tr>
<tr>
<td>24</td>
<td>NF</td>
<td>RW</td>
<td>ノンフォールティングロード命令で例外が起きたことを示す。</td>
</tr>
<tr>
<td>23:16</td>
<td>ASI&lt;7:0&gt;</td>
<td>RW</td>
<td>例外が起きた際のアクセスに使われた ASI 番号が表示される。このフィールドは D_SFSR.FV に 1 がセットされているときのみ有効。データアクセス時に明示的に ASI が使われていないければ、暗黙の ASI が使われているので、以下の値がセットされる。</td>
</tr>
<tr>
<td>15</td>
<td>TM</td>
<td>RW</td>
<td>データアクセス中に TLB ミスが起きたことを示す。</td>
</tr>
<tr>
<td>13:7</td>
<td>FT&lt;6:0&gt;</td>
<td>RW</td>
<td>例外発生の原因をエンコード情報で示す。エンコードの意味は TABLE F-11 を参照。</td>
</tr>
<tr>
<td>6</td>
<td>E</td>
<td>RW</td>
<td>副作用があるページにアクセスしたことを示す。TTE.E = 1 のページか、ASI 15_16 か 1D_16 でアクセスして例外が通知されたときに、E に 1 がセットされる。このフィールドは訂正不能エラー、バスエラー、バスタイムアウトのいずれかによる data_access_error が通知されたときのみ有効。それ以外の例外通知時は、このフィールドの値は不定である。</td>
</tr>
</tbody>
</table>

TABLE F-10 D-SFSR フィールドの説明 (2 of 3)
### TABLE F-10 D-SFSR フィールドの説明 (3 of 3)

<table>
<thead>
<tr>
<th>ビット</th>
<th>フィールド名</th>
<th>アクセス</th>
<th>説明</th>
</tr>
</thead>
<tbody>
<tr>
<td>5:4</td>
<td>CT&lt;1:0&gt;</td>
<td>RW</td>
<td>例外を起こしたデータアクセスのコンテキストに関する情報が表示される。</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>00_{02}: Primary</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>01_{02}: Secondary</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>10_{02}: Nucleus</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>11_{02}: Reserved</td>
</tr>
</tbody>
</table>

Translating ASI でないか、または無効な ASI の場合、11_{02} が表示される。無効な組み合わせの ASI と命令により data_access_exception が通知された場合 (atomic quad load, block load/store, block commit store, partial store, short floating-point load/store, xfill の各 ASI は、特定のメモリアクセス命令でのみ使える）、CT には命令で指定された ASI が表示される。

### Note
共有コンテキストで起きたことを示す手段は定義されていない。共有コンテキストが関係する多重ヒットのときは、実効コンテキストの情報が表示される。

<table>
<thead>
<tr>
<th>ビット</th>
<th>フィールド名</th>
<th>アクセス</th>
<th>説明</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>PR</td>
<td>RW</td>
<td>特権モードでデータアクセス中に例外が通知されたことを示す。このフィールドは FV が 1 のときのみ有効。</td>
</tr>
<tr>
<td>2</td>
<td>W</td>
<td>RW</td>
<td>書き込み命令 (ストア命令かアトミック命令) で例外が発生したことを示す。</td>
</tr>
<tr>
<td>1</td>
<td>OW</td>
<td>RW</td>
<td>DSFSR.FV = 1 のときに例外が通知されたことを示す。例外通知時点で DSFSR.FV = 1 がセットされ、DSFSR.FV = 0 がセットされる。</td>
</tr>
<tr>
<td>0</td>
<td>FV</td>
<td>RW</td>
<td>DMMU で TLB ミス以外の例外が発生したときに 1 がセットされる。このフィールドが 0 のとき、他のフィールドは意味を持たない（ただし MMU ミスの場合を除く）。</td>
</tr>
</tbody>
</table>

DSFSR.FT のエンコーディングを TABLE F-11 に示す。

### TABLE F-11 D-SFSR.FT のエンコーディング

<table>
<thead>
<tr>
<th>FT&lt;6:0&gt;</th>
<th>例外発生理由</th>
</tr>
</thead>
<tbody>
<tr>
<td>01_{16}</td>
<td>特権アクセス違反。TTE.P = 1 のページにアクセスした時、PSTATE.PRI = 0 かつ ASI_PRIMARY/SECONDARY_AS_IF_USER (_LITTLE) が使われたことを示す。特権アクセス違反は data_access_exception 例外で通知される。</td>
</tr>
<tr>
<td>02_{16}</td>
<td>ノンフォールディングロードで TTE.E = 1 であるページにアクセスしたとき、1がセットされる。</td>
</tr>
<tr>
<td>04_{16}</td>
<td>TTE.CP = 0 であるページに、アトミック命令 (CASA, CASXA, SWAP, SWAPA, LDSTUB, LDSTUBA), atomic quad load 命令 (LDDA with ASI = 24_{16}, 2C_{16}, 34_{16}, 3C_{16}) または SIMD ロード/ストアでアクセスしたとき、1がセットされる。</td>
</tr>
</tbody>
</table>
複数の要因であるひとつの例外が通知されると、DSFSR.FTの複数ビットがセットされる。

D-SFSRはfast_data_access_MMU_miss, data_access_exception, fast_data_access_protection, PA_watchpoint, VA_watchpoint, privileged_action, mem_address_not_aligned, data_access_errorのいずれかが通知された際に更新される。

TABLE F-12 D-SFSR 更新方針のまとめ

<table>
<thead>
<tr>
<th>フィールド</th>
<th>TLB, index</th>
<th>FV</th>
<th>OW</th>
<th>W, PR, NF, CT1</th>
<th>FT</th>
<th>TM</th>
<th>ASI</th>
<th>UE, BERR, BRT0, mDTLB, NC2, E2</th>
<th>DSFAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Miss:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>fast_data_access_MMU_miss</td>
<td>V</td>
</tr>
<tr>
<td>Exception:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>data_access_protection</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PA_watchpoint</td>
<td>V</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>VA_watchpoint</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>privileged_action1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>mem_address_not_aligned,</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>*_mem_address_not_aligned</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>data_access_error</td>
<td>V</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SIMD_load_across_pages</td>
<td>0</td>
</tr>
</tbody>
</table>

複数の要因であるひとつの例外が通知されると、DSFSR.FTの複数ビットがセットされる。
F.10.10 Synchronous Fault Addresses

VA\_watchpoint, PA\_watchpoint 例外が通知されたとき、D-SFAR には命令で指示されたアドレスが表示される。

ただし、SIMD 拡張されたロード・ストア命令の extend 側だけでウォッチポイント例外を検出したときは、extend 側のアドレス、つまり、命令で指示されたアドレスに、単精度ならば 4、倍精度ならば 8 を加算したアドレスが表示される。

F.10.11 I/D MMU Demap

sTLB 上のページデマップ時、インデクス計算に使われるページサイズは、ASI_I/DMMU\_DEMAP アクセスアドレスの context フィールドの情報から、TLB 検索と同じ手順で求められる。すなわち、ASI\_MCNTL.mp_g_sI/DTLB が 0 のときは、1st sTLB は 8KB ページで 2nd sTLB は 4MB ページ、ASI\_MCNTL.mp_g_sI/DTLB が 1 のときは、context フィールドで指示されたコンテキストのページサイズ情報が使われる。
ページサイズ情報はまた、ページデマップやコンテキストデマップにおける TTE の選択にも使われる。すなわち、ページサイズが TLB エントリのそれと一致しない場合、そのエントリはデマップされない。

**Note** – ページデマップ、コンテキストデマップでは、有効なコンテキスト番号を指定すること。IMMU で 012 または 112、DMMU で 112 を指定した場合は、無関係な sTLB のエントリをデマップするかもしれない。

sTLB の全デマップでは、ページ情報に係らずすべてのエントリがデマップされる。共有コンテキストを直接指示してデマップする方法はない。

**Programming Note** – 共有コンテキストの TTE のデマップは、セカンドリコンテキストを一時的に変更して行うことができる。

### F.10.12 Synchronous Fault Physical Addresses

**JPS1 Commonality** では、IMMU、DMMU で例外が発生した際に論理アドレス情報を記録するレジスタを定義している。SPARC64 IXfx ではこれらに加え、物理アドレスを記録するレジスタを定義する。

| レジスタ名 | ASI.IMMU_SFPAR, ASI_DMMU_SFPAR |
| ASI | 016 (IMMU), 5816 (DMMU) |
| VA | 7816 |
| アクセス種別 | Supervisor read/write |

<table>
<thead>
<tr>
<th>63</th>
<th>41</th>
<th>40</th>
<th>3</th>
<th>2</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fault Address (PA&lt;40:3&gt;)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

I/D-SFPAR は、例外を起こしたメモリアクセスの物理アドレス情報を表示するレジスタである。instruction/data_access_error が通知され、I/D-SFSR の MK, UE, BERR, BRTO のどれかに 1 がセットされるときに更新される。

### F.11 MMU Bypass

SPARC64 IXfx では、以下の 2 つの MMU バイパス ASI が定義されている。
F.12  Translation Lookaside Buffer Hardware

F.12.2  TLB Replacement Policy

Automatic TLB Replacement

I/D MMU Data In レジスタによる TLB 書き込みでは、ハードウェアが入れ替える TLB 種類とエントリを選択する。その選択基準は以下の通りである。

1. 以下のすべての条件に当てはまる場合は sTLB が、そうでなければ fTLB が選択される。
   - 書き込むエントリは TTE.L = 0 かつ TTE.G = 0
   - ページサイズが、ASI_MCNTRL.mpg_sITLB/mpg_sDTLB = 0 なら、8KB か 4MB
     ASI_MCNTRL.mpg_sITLB/mpg_sDTLB = 1 なら、I/
     DTLB_TAG_ACCESS_EXT コンテキストレジスタのページサイズに一致する
     とき
   - ASI_MCNTRL.fw_fITLB/fDTLB = 0

2. sTLB が選択された場合、TLB Tag Access の VA からページサイズに応じたビット位置を切り出して、インデクス番号とする。そのインデクスのエントリの LRU により追い出すエントリを決定する。
3. fTLB が選択された場合、以下の順で決定する。
   a. エントリ 0 から探索して最初に見つかった空きエントリが使われる。空きエントリがなければ、
   b. エントリ 0 から探索して最初に見つかった、ロックされておらず、used ビット1が 0 のエントリが使われる。
   c. そのようなエントリが見つからない場合、全エントリの used ビットを 0 クリアして、b をやりなおす。
   全エントリがロックされている場合、TLB の書き込みは行われず、例外も通知されない。
4. fTLB に書き込む際は、多重ヒットの確認が行われる。すでに fTLB にある TTE と書き込もうとする TTE が比較され、多重ヒットになる場合、新規 TTE の書き込みは行われない。

Restrictions on Direct Replacement of sTLB Entries

I/D MMU Data Access レジスタによる sTLB 曖み込みでは、書き込む TTE に制約はない。TTE のページサイズと sTLB のページサイズが一致するかどうかの検査は行われない。
ソフトウェアトラップの定義を以下のように修正する。

software_trap_number
以下のいずれかの形式である

- \( reg_{rs1} \) (\( reg_{rs1} + %g0 \) と等価）
- \( reg_{rs1} + imm7 \)
- \( reg_{rs1} - imm7 \)
- \( imm7 \) (\( %g0 + imm7 \) と等価）
- \( imm7 + reg_{rs1} \) (\( reg_{rs1} + imm7 \) と等価）
- \( reg_{rs1} + reg_{rs2} \)

ここで、\( imm7 \) は 7 ビットで表現可能な符号なし整数である。オペランドの値 (ソフトウェアトラップ番号) は 0–127 の間に値でなければならない。
HPC-ACE 拡張機能の表記法

いくつかの命令については HPC-ACE で機能が拡張されているが、この拡張が使われるのは、命令実行時点の XAR の値により決まる。一般的には SXAR と演算命令を組み合わせて指示する。この節ではアセンブリ言語で HPC-ACE 拡張機能を指示するための表記法を定義する。

命令仕様の HPC-ACE 拡張には、拡張されたレジスタの使用、SIMD 演算、セクタキャッシュ指定、ハードウェアリフェッチのオン / オフなどがある。仕様上はこれらはすべて SXAR で指示するが、アセンブリ言語上の表記は見やすさを考慮して以下のようになる。

1. SXAR は sxar1 または sxar2 と表記する。引数はなし。
2. 拡張レジスタは対象命令の中で直接指定する。
3. それ以外の拡張機能は、対象命令のニーモニックにサフィックスをつけて指示する。
4. ニーモニックで指示された機能は、その命令から遡って最初に出現する SXAR で指示されているものとする。対象命令と SXAR の間にラベルがあったとしても、そのラベルへと分岐してくる命令列の方向へは遡らない。

一方、SXAR と対象命令の間にラベルを許すかどうかは規定しない。4 によりどの SXAR で HPC-ACE 拡張機能を指示するかは明確なためである。

HPC-ACE 拡張のサフィックス

HPC-ACE 拡張は、ニーモニックの後にコンマ (,) を置き、その後に英数字 1 文字で指定する。TABLE G-1 に拡張指示サフィックスを示す。

<table>
<thead>
<tr>
<th>XAR 表記</th>
<th>サフィックス</th>
<th>備考</th>
</tr>
</thead>
<tbody>
<tr>
<td>XAR.simd</td>
<td>s</td>
<td></td>
</tr>
<tr>
<td>XAR.diss_hw_pf</td>
<td>d</td>
<td></td>
</tr>
</tbody>
</table>
サフィックスは大文字・小文字の区別はなく、複数のサフィックスを指定する場合は任意の順序で指定可能とする。

例1: 拡張レジスタ使用とSIMD演算
sxar2
  fmaddd  %f0, %f2, %f510  /*拡張レジスタ使用、non-SIMD */
  fmaddd,s %f0, %f2, %f4  /*SIMD, 拡張側で拡張レジスタ使用 */

例2: sector 1 にSIMD load
sxarl
  ldd,s1  [%xg24], %f0  /*サフィックスはlsでも可 */

<table>
<thead>
<tr>
<th>XAR 表記</th>
<th>サフィックス</th>
<th>備考</th>
</tr>
</thead>
<tbody>
<tr>
<td>XAR.sector</td>
<td>l</td>
<td>0 でセクタ0指定 (デフォルト)</td>
</tr>
<tr>
<td>XAR.negate_mul</td>
<td>n</td>
<td></td>
</tr>
<tr>
<td>XAR.rsl_copy</td>
<td>c</td>
<td></td>
</tr>
</tbody>
</table>

TABLE G-1  HPC-ACE 拡張命令で使用できるサフィックス
Software Considerations

JPS1 Commonality の Appendix H を参照。
Extending the SPARC V9 Architecture

JPS1 Commonality の Appendix I を参照。
Changes from SPARC V8 to SPARC V9

JPS1 Commonality の Appendix J を参照。
Programming with the Memory Models

JPS1 Commonality の Appendix K を参照。
Address Space Identifiers

この章では SPARC64 IXfx でサポートされている ASI の一覧を示し、特殊な ASI について解説する。

L.2 ASI Values

SPARC V9 の空間識別子 (Address Space Identifier:ASI) は、制限ありのものと制限なしの 2 つにわけられる。00_{16}–7F_{16} は制限ありで、80_{16}–FF_{16} は制限なしである。制限ありの ASI を非特権ソフトウェアが使おうとすると、privileged_action 例外が通知される。

さらに ASI は、MMU により変換されるもの (translating)、MMU をバイパスするもの (bypass)、CPU 内部の資源にアクセスするもの (nontranslating) の 3 つに分けられる。SPARC64 IXfx の定義を TABLE L-1 に示す。

Compatibility Note – JPSI Commonality ではこの 3 種類に、実装依存や未定義 ASI が含まれていたが、SPARC64 IXfx 仕様では定義された ASI のみを含むように分類されている。
ASI の種類は Data Watchpoint とも関連している。詳細は、JPS1 Commonality および本論理仕様書の "Data Watchpoint Registers" (page 36) を参照。

### L.3 SPARC64 IXfx ASI Assignments

SPARC V9 プロセッサでは、メモリアクセス命令で指示するアドレスは、8 ビットの空間識別子 (Address Space Identifier) と論理アドレス (VA) の組で一意に定まるアドレスを使用する。命令フェッチや ASI を明示しないメモリアクセス命令は、暗黙の ASI がハードウェアによって付加される。load from alternate, store from alternate 命令等の ASI を明示する命令の場合、%asi レジスタまたは命令により直に指定される ASI が使われる。その他、メモリ空間をアクセスするのではなく、MMU やハードウェアバリアなど CPU レジスタ内のレジスタにアクセスするために使われる ASI もある。

Section L.3.1 の情報は JPS1 Commonality と SPARC64 IXfx 拡張の両方を網羅している。

### L.3.1 Supported ASIs

TABLE L-2 には、SPARC V9 で定義された ASI、SPARC V9 では定義されていないが JPS1 プロセッサで必須の ASI、およびSPARC64 IXfx で定義された ASI のリストである。網掛けされた部分は、SPARC V9 または JPS1 では定義されていたが SPARC64 IXfx では定義されていない ASI である。
黒丸（●）がつっているのは SPARC V9 定義の ASI である。任意のサイズのメモリアクセスで使用できる。

白丸（○）がつっているのは SPARC V9 ではなく定義されていないが JPS1 プロセッサで必須の ASI である。これらは特に記述がない限り LDAX, STXA, LDDFA, STDFA 命令でのみ使用できる。

星印（★）がつているのは SPARC64 IXfx で定義された ASI である。これらは特に記述がない限り LDAX, STXA, LDDFA, STDFA 命令でのみ使用できる。

ASI の有効な論理アドレスは、TABLE L-2 の VA, 有効ビット, アラインの列により規定される。

- VA の列は論理アドレスを示す。"—" と表示されている場合は、任意のアドレスが指定できる。"encode" と表示されている場合は各 ASI の説明を参照。
- 有効ビットで、VA のどのビットが有効かを示す。有効でないビットは無視される。
  1. "full" は 64 ビットすべてが有効。
  2. "physical" は物理アドレスのビット幅までが有効。
  3. bit<a:b> はビット位置 a から b までが有効。
- アラインの列には、アラインメントに制約がある場合はその条件が示され、アラインメントに制約がない場合は "—" となっている。アラインメント違反に対して通知される例外の種類は、各命令の説明を参照。

"共有" の列は、nontranslating ASI が CPU チップで共有されているのか、コアごとに独立しているかを示す。translating および bypass ASI は "—" となっている。未定義のASI および許されていない命令との組み合わせで起こる例外については、Appendix L.3.3 を参照。

<table>
<thead>
<tr>
<th>ASI</th>
<th>VA</th>
<th>有効ビット</th>
<th>アライン</th>
<th>ASI 名（と省略表現）</th>
<th>共有</th>
<th>アクセス</th>
<th>ページ</th>
</tr>
</thead>
<tbody>
<tr>
<td>● 0416</td>
<td>—</td>
<td>full</td>
<td>—</td>
<td>ASI_NUCLEUS (ASI_N)</td>
<td>—</td>
<td>RW</td>
<td></td>
</tr>
<tr>
<td>● 0C16</td>
<td>—</td>
<td>full</td>
<td>—</td>
<td>ASI_NUCLEUS_LITTLE (ASI_NL)</td>
<td>—</td>
<td>RW</td>
<td></td>
</tr>
<tr>
<td>● 1016</td>
<td>—</td>
<td>full</td>
<td>—</td>
<td>ASI_AS_IF_USER_PRIMARY (ASI_AIUP)</td>
<td>—</td>
<td>RW</td>
<td></td>
</tr>
<tr>
<td>● 1116</td>
<td>—</td>
<td>full</td>
<td>—</td>
<td>ASI_AS_IF_USER_SECONDARY (ASI_AIUS)</td>
<td>—</td>
<td>RW</td>
<td></td>
</tr>
<tr>
<td>○ 1416</td>
<td>—</td>
<td>physical</td>
<td>—</td>
<td>ASI_PHYS_USE_EC</td>
<td>■</td>
<td>RW</td>
<td></td>
</tr>
<tr>
<td>○ 1516</td>
<td>—</td>
<td>physical</td>
<td>—</td>
<td>ASI_PHYS_BYPASS_EC_WITH_EBIT</td>
<td>—</td>
<td>RW</td>
<td></td>
</tr>
<tr>
<td>● 1816</td>
<td>—</td>
<td>full</td>
<td>—</td>
<td>ASI_AS_IF_USER_PRIMARY_LITTLE (ASI_AIUPL)</td>
<td>—</td>
<td>RW</td>
<td></td>
</tr>
<tr>
<td>● 1916</td>
<td>—</td>
<td>full</td>
<td>—</td>
<td>ASI_AS_IF_USER_SECONDARY_LITTLE (ASI_AIUSL)</td>
<td>—</td>
<td>RW</td>
<td></td>
</tr>
<tr>
<td>○ 1C16</td>
<td>—</td>
<td>physical</td>
<td>—</td>
<td>ASI_PHYS_USE_EC_LITTLE</td>
<td>—</td>
<td>RW</td>
<td></td>
</tr>
</tbody>
</table>

Ver 12, 2 Dec. 2013 F. Appendix L Address Space Identifiers 215
<table>
<thead>
<tr>
<th>ASI</th>
<th>VA</th>
<th>有効ビット</th>
<th>アライ線</th>
<th>ASI名 (と略称表記)</th>
<th>共有</th>
<th>アクセス ページ</th>
</tr>
</thead>
<tbody>
<tr>
<td>⊗ 1D16</td>
<td>—</td>
<td>physical</td>
<td>—</td>
<td>ASI_PHYS_BYPASS_BC_WITH_EBIT_LITTLE (ASI_PHYS_BYPASS_BC_WITH_EBIT_L)</td>
<td>—</td>
<td>RW</td>
</tr>
<tr>
<td>⊗ 2416</td>
<td>—</td>
<td>full</td>
<td>16byte</td>
<td>ASI_NUCLEUS_QUAD_LDD</td>
<td>—</td>
<td>R</td>
</tr>
<tr>
<td>⊗ 2C16</td>
<td>—</td>
<td>full</td>
<td>16byte</td>
<td>ASI_NUCLEUS_QUAD_LDD_LITTLE (ASI_NUCLEUS_QUAD_LDD_L)</td>
<td>—</td>
<td>R</td>
</tr>
<tr>
<td>★ 3416</td>
<td>—</td>
<td>physical</td>
<td>16byte</td>
<td>ASI_ATOMIC_QUAD_LDD_PHYS</td>
<td>—</td>
<td>R 88</td>
</tr>
<tr>
<td>★ 3C16</td>
<td>—</td>
<td>physical</td>
<td>16byte</td>
<td>ASI_ATOMIC_QUAD_LDD_PHYS_LITTLE</td>
<td>—</td>
<td>R 88</td>
</tr>
<tr>
<td>⊗ 4516</td>
<td>0016</td>
<td>bit&lt;7:0&gt;</td>
<td>8byte</td>
<td>ASI_DCU_CONTROL_REGISTER (ASI_DCUCR)</td>
<td>Core</td>
<td>RW 34</td>
</tr>
<tr>
<td>⊗ 4516</td>
<td>0816</td>
<td>bit&lt;7:0&gt;</td>
<td>8byte</td>
<td>ASI_MEMORY_CONTROL_REG (ASI_MCNTR)</td>
<td>Core</td>
<td>RW 184</td>
</tr>
<tr>
<td>★ 4616</td>
<td>0016</td>
<td>bit&lt;7:0&gt;</td>
<td>8byte</td>
<td>Reserved</td>
<td>Core</td>
<td>R</td>
</tr>
<tr>
<td>★ 4716</td>
<td>0016</td>
<td>bit&lt;7:0&gt;</td>
<td>8byte</td>
<td>Reserved</td>
<td>Core</td>
<td>R</td>
</tr>
<tr>
<td>⊗ 4816</td>
<td>0016</td>
<td>bit&lt;7:0&gt;</td>
<td>8byte</td>
<td>ASI_INTR_DISPATCH_STATUS (ASI_MONDO_SEND_CTRL)</td>
<td>Core</td>
<td>R 244</td>
</tr>
<tr>
<td>⊗ 4916</td>
<td>0016</td>
<td>bit&lt;7:0&gt;</td>
<td>8byte</td>
<td>ASI_INTR.Receive (ASI_MONDO_RECEIVE_CTRL)</td>
<td>Core</td>
<td>RW 245</td>
</tr>
<tr>
<td>★ 4A16</td>
<td>—</td>
<td>bit&lt;7:0&gt;</td>
<td>8byte</td>
<td>ASI_SYS_CONFIG</td>
<td>Core</td>
<td>R 325</td>
</tr>
<tr>
<td>★ 4B16</td>
<td>0016</td>
<td>bit&lt;7:0&gt;</td>
<td>8byte</td>
<td>ASI_STICK_CNTL</td>
<td>Chip</td>
<td>RW 326</td>
</tr>
<tr>
<td>⊗ 4C16</td>
<td>0016</td>
<td>bit&lt;7:0&gt;</td>
<td>8byte</td>
<td>ASI_ASYNC_FAULT_STATUS (ASI_AFSR)</td>
<td>Core</td>
<td>RW 287</td>
</tr>
<tr>
<td>★ 4C16</td>
<td>0816</td>
<td>bit&lt;7:0&gt;</td>
<td>8byte</td>
<td>ASI_URGENT_ERROR_STATUS (ASI_UGESR)</td>
<td>Core</td>
<td>R 277</td>
</tr>
<tr>
<td>★ 4C16</td>
<td>1016</td>
<td>bit&lt;7:0&gt;</td>
<td>8byte</td>
<td>ASI_ERROR_CONTROL (ASI_ECR)</td>
<td>Core</td>
<td>RW 272</td>
</tr>
<tr>
<td>★ 4C16</td>
<td>1816</td>
<td>bit&lt;7:0&gt;</td>
<td>8byte</td>
<td>ASI_STATE_CHANGE_ERROR_INFO (ASI_STCHG_ERR_INFO)</td>
<td>Core</td>
<td>RW 274</td>
</tr>
<tr>
<td>⊗ 4D16</td>
<td>0016</td>
<td>bit&lt;7:0&gt;</td>
<td>8byte</td>
<td>ASI_ASYNC_FAULT_ADDR (ASI_AFAR)</td>
<td>Core</td>
<td>RW 220</td>
</tr>
<tr>
<td>★ 4F16</td>
<td>0016</td>
<td>bit&lt;7:0&gt;</td>
<td>8byte</td>
<td>ASI_SCRATCH_REG0</td>
<td>Core</td>
<td>RW 220</td>
</tr>
<tr>
<td>★ 4F16</td>
<td>0816</td>
<td>bit&lt;7:0&gt;</td>
<td>8byte</td>
<td>ASI_SCRATCH_REG1</td>
<td>Core</td>
<td>RW 220</td>
</tr>
<tr>
<td>★ 4F16</td>
<td>1016</td>
<td>bit&lt;7:0&gt;</td>
<td>8byte</td>
<td>ASI_SCRATCH_REG2</td>
<td>Core</td>
<td>RW 220</td>
</tr>
<tr>
<td>★ 4F16</td>
<td>1816</td>
<td>bit&lt;7:0&gt;</td>
<td>8byte</td>
<td>ASI_SCRATCH_REG3</td>
<td>Core</td>
<td>RW 220</td>
</tr>
<tr>
<td>★ 4F16</td>
<td>2016</td>
<td>bit&lt;7:0&gt;</td>
<td>8byte</td>
<td>ASI_SCRATCH_REG4</td>
<td>Core</td>
<td>RW 220</td>
</tr>
<tr>
<td>★ 4F16</td>
<td>2816</td>
<td>bit&lt;7:0&gt;</td>
<td>8byte</td>
<td>ASI_SCRATCH_REG5</td>
<td>Core</td>
<td>RW 220</td>
</tr>
<tr>
<td>★ 4F16</td>
<td>3016</td>
<td>bit&lt;7:0&gt;</td>
<td>8byte</td>
<td>ASI_SCRATCH_REG6</td>
<td>Core</td>
<td>RW 220</td>
</tr>
<tr>
<td>★ 4F16</td>
<td>3816</td>
<td>bit&lt;7:0&gt;</td>
<td>8byte</td>
<td>ASI_SCRATCH_REG7</td>
<td>Core</td>
<td>RW 220</td>
</tr>
<tr>
<td>⊗ 5016</td>
<td>0016</td>
<td>bit&lt;7:0&gt;</td>
<td>8byte</td>
<td>ASI_IMMU_TAG_TARGET</td>
<td>Core</td>
<td>R</td>
</tr>
<tr>
<td>⊗ 5016</td>
<td>1816</td>
<td>bit&lt;7:0&gt;</td>
<td>8byte</td>
<td>ASI_IMMU_SFSR</td>
<td>Core</td>
<td>RW 194</td>
</tr>
<tr>
<td>⊗ 5016</td>
<td>2816</td>
<td>bit&lt;7:0&gt;</td>
<td>8byte</td>
<td>ASI_IMMU_TSB_BASE</td>
<td>Core</td>
<td>RW 193</td>
</tr>
<tr>
<td>⊗ 5016</td>
<td>3016</td>
<td>bit&lt;7:0&gt;</td>
<td>8byte</td>
<td>ASI_IMMU_TAG_ACCESS</td>
<td>Core</td>
<td>RW 193</td>
</tr>
<tr>
<td>⊗ 5016</td>
<td>4816</td>
<td>bit&lt;7:0&gt;</td>
<td>8byte</td>
<td>ASI_IMMU_TSB_PEXT_REG</td>
<td>Core</td>
<td>RW 191</td>
</tr>
<tr>
<td>⊗ 5016</td>
<td>5816</td>
<td>bit&lt;7:0&gt;</td>
<td>8byte</td>
<td>ASI_IMMU_TSB_NEXT_REG</td>
<td>Core</td>
<td>RW 202</td>
</tr>
<tr>
<td>★ 5016</td>
<td>6016</td>
<td>bit&lt;7:0&gt;</td>
<td>8byte</td>
<td>ASI_IMMU_TAG_ACCESS_EXT</td>
<td>Core</td>
<td>RW 191</td>
</tr>
<tr>
<td>★ 5016</td>
<td>7816</td>
<td>bit&lt;7:0&gt;</td>
<td>8byte</td>
<td>ASI_IMMU_SFPAR</td>
<td>Core</td>
<td>RW 202</td>
</tr>
<tr>
<td>ASI</td>
<td>VA</td>
<td>有效ビット</td>
<td>アライン</td>
<td>ASI 名 (と省略表記)</td>
<td>共有</td>
<td>アクセス</td>
</tr>
<tr>
<td>-------</td>
<td>------</td>
<td>-----------</td>
<td>--------</td>
<td>----------------------------------------------------------------------------------</td>
<td>------</td>
<td>--------</td>
</tr>
<tr>
<td>☒ 51₁₆</td>
<td>00₁₆</td>
<td>bit&lt;7:0&gt; 8byte</td>
<td>ASI_IMMU_TSB_8KB_PTR_REG</td>
<td>Chip R 220</td>
<td></td>
<td></td>
</tr>
<tr>
<td>☒ 52₁₆</td>
<td>00₁₆</td>
<td>bit&lt;7:0&gt; 8byte</td>
<td>ASI_IMMU_TSB_64KB_PTR_REG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>★ 53₁₆</td>
<td>—</td>
<td>bit&lt;7:0&gt; 8byte</td>
<td>ASI_Serial_ID</td>
<td>Chip W 192</td>
<td></td>
<td></td>
</tr>
<tr>
<td>★ 54₁₆</td>
<td>—</td>
<td>bit&lt;7:0&gt; 8byte</td>
<td>ASI_ITLB_DATA_IN_REG</td>
<td>Core RW 192</td>
<td></td>
<td></td>
</tr>
<tr>
<td>★ 55₁₆</td>
<td>encode</td>
<td>bit&lt;17:0&gt; 8byte</td>
<td>ASI_ITLB_DATA_ACCESS_REG</td>
<td>Core RW 192</td>
<td></td>
<td></td>
</tr>
<tr>
<td>★ 56₁₆</td>
<td>encode</td>
<td>bit&lt;17:0&gt; 8byte</td>
<td>ASI_ITLB_TAG_READ_REG</td>
<td>Core R 193</td>
<td></td>
<td></td>
</tr>
<tr>
<td>★ 57₁₆</td>
<td>encode</td>
<td>full</td>
<td>8byte</td>
<td>ASI_IMMU_DEMAP</td>
<td>Core W 201</td>
<td></td>
</tr>
<tr>
<td>★ 58₁₆</td>
<td>00₁₆</td>
<td>bit&lt;7:0&gt; 8byte</td>
<td>ASI_DMMU_TAG_TARGET_REG</td>
<td>Core R</td>
<td></td>
<td></td>
</tr>
<tr>
<td>★ 58₁₆</td>
<td>08₁₆</td>
<td>bit&lt;7:0&gt; 8byte</td>
<td>ASIPRIMARY_CONTEXT_REG</td>
<td>Core RW 188</td>
<td></td>
<td></td>
</tr>
<tr>
<td>★ 58₁₆</td>
<td>10₁₆</td>
<td>bit&lt;7:0&gt; 8byte</td>
<td>ASISECONDARY_CONTEXT_REG</td>
<td>Core RW 188</td>
<td></td>
<td></td>
</tr>
<tr>
<td>★ 58₁₆</td>
<td>18₁₆</td>
<td>bit&lt;7:0&gt; 8byte</td>
<td>ASI_DMMU_SFSR</td>
<td>Core RW 194</td>
<td></td>
<td></td>
</tr>
<tr>
<td>★ 58₁₆</td>
<td>20₁₆</td>
<td>bit&lt;7:0&gt; 8byte</td>
<td>ASI_DMMU_SFAR</td>
<td>Core RW</td>
<td></td>
<td></td>
</tr>
<tr>
<td>★ 58₁₆</td>
<td>28₁₆</td>
<td>bit&lt;7:0&gt; 8byte</td>
<td>ASI_DMMU_TSB_BASE</td>
<td>Core RW 193</td>
<td></td>
<td></td>
</tr>
<tr>
<td>★ 58₁₆</td>
<td>30₁₆</td>
<td>bit&lt;7:0&gt; 8byte</td>
<td>ASI_DMMU_TAG_ACCESS</td>
<td>Core RW 193</td>
<td></td>
<td></td>
</tr>
<tr>
<td>★ 58₁₆</td>
<td>38₁₆</td>
<td>bit&lt;7:0&gt; 8byte</td>
<td>ASI_DMMU_WATCHPOINT_REG</td>
<td>Core RW 36</td>
<td></td>
<td></td>
</tr>
<tr>
<td>★ 58₁₆</td>
<td>40₁₆</td>
<td></td>
<td>ASI_DMMU_PA_WATCHPOINT_REG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>★ 58₁₆</td>
<td>48₁₆</td>
<td></td>
<td>ASI_DMMU_TSB_PEXT_REG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>★ 58₁₆</td>
<td>50₁₆</td>
<td></td>
<td>ASI_DMMU_TSB_SEXT_REG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>★ 58₁₆</td>
<td>58₁₆</td>
<td></td>
<td>ASI_DMMU_TSB_NEXT_REG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>★ 58₁₆</td>
<td>60₁₆</td>
<td>bit&lt;7:0&gt; 8byte</td>
<td>ASI_DMMU_TAG_ACCESS_EXT</td>
<td>Core RW 191</td>
<td></td>
<td></td>
</tr>
<tr>
<td>★ 58₁₆</td>
<td>68₁₆</td>
<td>bit&lt;7:0&gt; 8byte</td>
<td>ASI_SHARED_CONTEXT_REG</td>
<td>Core RW 189</td>
<td></td>
<td></td>
</tr>
<tr>
<td>★ 58₁₆</td>
<td>78₁₆</td>
<td>bit&lt;7:0&gt; 8byte</td>
<td>ASI_DMMU_SFPAR</td>
<td>Core RW 202</td>
<td></td>
<td></td>
</tr>
<tr>
<td>☒ 59₁₆</td>
<td>00₁₆</td>
<td></td>
<td>ASI_DMMU_TSB_8KB_PTR_REG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>☒ 5A₁₆</td>
<td>00₁₆</td>
<td></td>
<td>ASI_DMMU_TSB_64KB_PTR_REG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>☒ 5B₁₆</td>
<td>00₁₆</td>
<td></td>
<td>ASI_DMMU_TSB_DIRECT_PTR_REG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>☒ 5C₁₆</td>
<td>—</td>
<td>bit&lt;7:0&gt; 8byte</td>
<td>ASI_DTLB_DATA_IN_REG</td>
<td>Core W 192</td>
<td></td>
<td></td>
</tr>
<tr>
<td>☒ 5D₁₆</td>
<td>encode</td>
<td>bit&lt;17:0&gt; 8byte</td>
<td>ASI_DTLB_DATA_ACCESS_REG</td>
<td>Core RW 192</td>
<td></td>
<td></td>
</tr>
<tr>
<td>☒ 5E₁₆</td>
<td>encode</td>
<td>bit&lt;17:0&gt; 8byte</td>
<td>ASI_DTLB_TAG_READ_REG</td>
<td>Core R 193</td>
<td></td>
<td></td>
</tr>
<tr>
<td>☒ 5F₁₆</td>
<td>encode</td>
<td>full</td>
<td>8byte</td>
<td>ASI_DMMU_DEMAP</td>
<td>Core W 201</td>
<td></td>
</tr>
<tr>
<td>★ 60₁₆</td>
<td>—</td>
<td>bit&lt;7:0&gt; 8byte</td>
<td>ASI_IIU_INST_TRAP</td>
<td>Core RW 37</td>
<td></td>
<td></td>
</tr>
<tr>
<td>★ 67₁₆</td>
<td>—</td>
<td>bit&lt;7:0&gt; 8byte</td>
<td>ASI_FLUSH_L1I</td>
<td>Core W 235</td>
<td></td>
<td></td>
</tr>
<tr>
<td>★ 6D₁₆</td>
<td>00₁₆–58₁₆</td>
<td>bit&lt;7:0&gt; 8byte</td>
<td>ASI_BARRIER_INIT</td>
<td>Chip RW 224</td>
<td></td>
<td></td>
</tr>
<tr>
<td>★ 6D₁₆</td>
<td>80₁₆–D8₁₆</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>★ 6E₁₆</td>
<td>00₁₆</td>
<td>bit&lt;7:0&gt; 8byte</td>
<td>ASI_ERROR_IDENT (ASI_EIDR)</td>
<td>Core RW 272</td>
<td></td>
<td></td>
</tr>
<tr>
<td>★ 6E₁₆</td>
<td>08₁₆</td>
<td>bit&lt;7:0&gt; 8byte</td>
<td>ASI_BARRIER_INIT</td>
<td>Core R 226</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### TABLE L-2  SPARC64 IXfx ASIs  (4 of 6)

<table>
<thead>
<tr>
<th>ASI VA</th>
<th>有効ビット</th>
<th>アライン</th>
<th>ASI 名 (と省略表記)</th>
<th>共有</th>
<th>アクセス ページ</th>
</tr>
</thead>
<tbody>
<tr>
<td>★ 6F16</td>
<td>0016–9816</td>
<td>bit&lt;7:0&gt;</td>
<td>8byte ASI_BARRIER_ASSIGN</td>
<td>Core</td>
<td>RW 227</td>
</tr>
<tr>
<td>○ 7016</td>
<td>full</td>
<td>64byte</td>
<td>ASI_BLOCK_AS_IP_USER_PRIMARY (ASI_BLK_AIUP)</td>
<td>—</td>
<td>RW</td>
</tr>
<tr>
<td>○ 7116</td>
<td>full</td>
<td>64byte</td>
<td>ASI_BLOCK_AS_IP_USER_SECONDARY (ASI_BLK_AIUS)</td>
<td>—</td>
<td>RW</td>
</tr>
<tr>
<td>★ 7216</td>
<td>full</td>
<td>8byte</td>
<td>ASI_XFILL_AIUP</td>
<td>—</td>
<td>W 133</td>
</tr>
<tr>
<td>★ 7316</td>
<td>full</td>
<td>8byte</td>
<td>ASI_XFILL_AIUS</td>
<td>—</td>
<td>W 133</td>
</tr>
<tr>
<td>★ 7416</td>
<td>physical</td>
<td>8byte</td>
<td>ASI_CACHE_INV</td>
<td>Chip</td>
<td>W 235</td>
</tr>
<tr>
<td>○ 7F16</td>
<td>4016</td>
<td>bit&lt;7:0&gt;</td>
<td>8byte ASI_INTR_DATA0_W</td>
<td>Core</td>
<td>W 244</td>
</tr>
<tr>
<td>○ 7F16</td>
<td>4816</td>
<td>bit&lt;7:0&gt;</td>
<td>8byte ASI_INTR_DATA1_W</td>
<td>Core</td>
<td>W 244</td>
</tr>
<tr>
<td>○ 7F16</td>
<td>5016</td>
<td>bit&lt;7:0&gt;</td>
<td>8byte ASI_INTR_DATA2_W</td>
<td>Core</td>
<td>W 244</td>
</tr>
<tr>
<td>○ 7716</td>
<td>5816</td>
<td></td>
<td>ASI_INTR_DATA3_W</td>
<td>Core</td>
<td>W 244</td>
</tr>
<tr>
<td>○ 7716</td>
<td>6016</td>
<td></td>
<td>ASI_INTR_DATA4_W</td>
<td></td>
<td></td>
</tr>
<tr>
<td>○ 7716</td>
<td>6816</td>
<td></td>
<td>ASI_INTR_DATA5_W</td>
<td></td>
<td></td>
</tr>
<tr>
<td>○ 7716</td>
<td>8016</td>
<td></td>
<td>ASI_INTR_DATA6_W</td>
<td></td>
<td></td>
</tr>
<tr>
<td>○ 7716</td>
<td>8816</td>
<td></td>
<td>ASI_INTR_DATA7_W</td>
<td></td>
<td></td>
</tr>
<tr>
<td>○ 7716</td>
<td>encode</td>
<td></td>
<td>ASI_INTR DISPATCH_W</td>
<td>Core</td>
<td>W 244</td>
</tr>
<tr>
<td>○ 7716</td>
<td>7016</td>
<td>bit&lt;26:24&gt;, bit&lt;16:14&gt;, bit&lt;13:0&gt;</td>
<td>8byte</td>
<td>Core</td>
<td>W 244</td>
</tr>
<tr>
<td>○ 7816</td>
<td>full</td>
<td>64byte</td>
<td>ASI_BLOCK_AS_IP_USER_PRIMARY_LITTLE (ASI_BLK_AIUPL)</td>
<td>—</td>
<td>RW</td>
</tr>
<tr>
<td>○ 7916</td>
<td>full</td>
<td>64byte</td>
<td>ASI_BLOCK_AS_IP_USER_SECONDARY_LITTLE (ASI_BLK_AIUSL)</td>
<td>—</td>
<td>RW</td>
</tr>
<tr>
<td>○ 7F16</td>
<td>4016</td>
<td>bit&lt;7:0&gt;</td>
<td>8byte ASI_INTR_DATA0_R</td>
<td>Core</td>
<td>R 244</td>
</tr>
<tr>
<td>○ 7F16</td>
<td>4816</td>
<td>bit&lt;7:0&gt;</td>
<td>8byte ASI_INTR_DATA1_R</td>
<td>Core</td>
<td>R 244</td>
</tr>
<tr>
<td>○ 7F16</td>
<td>5016</td>
<td>bit&lt;7:0&gt;</td>
<td>8byte ASI_INTR_DATA2_R</td>
<td>Core</td>
<td>R 244</td>
</tr>
<tr>
<td>○ 7F16</td>
<td>5816</td>
<td></td>
<td>ASI_INTR_DATA3_R</td>
<td></td>
<td></td>
</tr>
<tr>
<td>○ 7F16</td>
<td>6016</td>
<td></td>
<td>ASI_INTR_DATA4_R</td>
<td></td>
<td></td>
</tr>
<tr>
<td>○ 7F16</td>
<td>6816</td>
<td></td>
<td>ASI_INTR_DATA5_R</td>
<td></td>
<td></td>
</tr>
<tr>
<td>○ 7F16</td>
<td>8016</td>
<td></td>
<td>ASI_INTR_DATA6_R</td>
<td></td>
<td></td>
</tr>
<tr>
<td>○ 7F16</td>
<td>8816</td>
<td></td>
<td>ASI_INTR_DATA7_R</td>
<td></td>
<td></td>
</tr>
<tr>
<td>● 8016</td>
<td>full</td>
<td></td>
<td>ASI_PRIMARY (ASI_P)</td>
<td>—</td>
<td>RW</td>
</tr>
<tr>
<td>● 8116</td>
<td>full</td>
<td></td>
<td>ASI_SECONDARY (ASI_S)</td>
<td>—</td>
<td>RW</td>
</tr>
<tr>
<td>● 8216</td>
<td>full</td>
<td></td>
<td>ASI_PRIMARY_NO_FAULT (ASI_PNF)</td>
<td>—</td>
<td>R</td>
</tr>
<tr>
<td>● 8316</td>
<td>full</td>
<td></td>
<td>ASI_SECONDARY_NO_FAULT (ASI_SNF)</td>
<td>—</td>
<td>R</td>
</tr>
<tr>
<td>● 8816</td>
<td>full</td>
<td></td>
<td>ASI_PRIMARY_LITTLE (ASI_FL)</td>
<td>—</td>
<td>RW</td>
</tr>
<tr>
<td>● 8916</td>
<td>full</td>
<td></td>
<td>ASI_SECONDARY_LITTLE (ASI_SL)</td>
<td>—</td>
<td>RW</td>
</tr>
</tbody>
</table>
### TABLE L-2 SPARC64 IXfix ASIs (5 of 6)

<table>
<thead>
<tr>
<th>ASI</th>
<th>VA</th>
<th>有効ビット</th>
<th>アライン</th>
<th>ASI 名（と省略表記）</th>
<th>共有</th>
<th>アクセス ページ</th>
</tr>
</thead>
<tbody>
<tr>
<td>8A16</td>
<td>—</td>
<td>full</td>
<td>—</td>
<td>ASI_PRIMARY_NO_FAULT_LITTLE (ASI_PNFL)</td>
<td>—</td>
<td>R</td>
</tr>
<tr>
<td>8B16</td>
<td>—</td>
<td>full</td>
<td>—</td>
<td>ASI_SECONDARY_NO_FAULT_LITTLE (ASI_SNFL)</td>
<td>—</td>
<td>R</td>
</tr>
<tr>
<td>8A16</td>
<td>—</td>
<td>full</td>
<td>—</td>
<td>ASI_PRIMARY_NO_FAULT_LITTLE (ASI_PNFL)</td>
<td>—</td>
<td>R</td>
</tr>
<tr>
<td>8B16</td>
<td>—</td>
<td>full</td>
<td>—</td>
<td>ASI_SECONDARY_NO_FAULT_LITTLE (ASI_SNFL)</td>
<td>—</td>
<td>R</td>
</tr>
<tr>
<td>C016</td>
<td>—</td>
<td>full</td>
<td>—</td>
<td>ASI_PST8_PRIMARY (ASI_PST8_P)</td>
<td>—</td>
<td>W 221</td>
</tr>
<tr>
<td>C116</td>
<td>—</td>
<td>full</td>
<td>—</td>
<td>ASI_PST8_SECONDARY (ASI_PST8_S)</td>
<td>—</td>
<td>W 221</td>
</tr>
<tr>
<td>C216</td>
<td>—</td>
<td>full</td>
<td>—</td>
<td>ASI_PST16_PRIMARY (ASI_PST16_P)</td>
<td>—</td>
<td>W 221</td>
</tr>
<tr>
<td>C316</td>
<td>—</td>
<td>full</td>
<td>—</td>
<td>ASI_PST16_SECONDARY (ASI_PST16_S)</td>
<td>—</td>
<td>W 221</td>
</tr>
<tr>
<td>C416</td>
<td>—</td>
<td>full</td>
<td>—</td>
<td>ASI_PST32_PRIMARY (ASI_PST32_P)</td>
<td>—</td>
<td>W 221</td>
</tr>
<tr>
<td>C516</td>
<td>—</td>
<td>full</td>
<td>—</td>
<td>ASI_PST32_SECONDARY (ASI_PST32_S)</td>
<td>—</td>
<td>W 221</td>
</tr>
<tr>
<td>C616</td>
<td>—</td>
<td>full</td>
<td>—</td>
<td>ASI_PST8_PRIMARY_LITTLE (ASI_PST8_PL)</td>
<td>—</td>
<td>W 221</td>
</tr>
<tr>
<td>C716</td>
<td>—</td>
<td>full</td>
<td>—</td>
<td>ASI_PST8_SECONDARY_LITTLE (ASI_PST8_SL)</td>
<td>—</td>
<td>W 221</td>
</tr>
<tr>
<td>C816</td>
<td>—</td>
<td>full</td>
<td>—</td>
<td>ASI_PST16_PRIMARY_LITTLE (ASI_PST16_PL)</td>
<td>—</td>
<td>W 221</td>
</tr>
<tr>
<td>C916</td>
<td>—</td>
<td>full</td>
<td>—</td>
<td>ASI_PST16_SECONDARY_LITTLE (ASI_PST16_SL)</td>
<td>—</td>
<td>W 221</td>
</tr>
<tr>
<td>CA16</td>
<td>—</td>
<td>full</td>
<td>—</td>
<td>ASI_PST16_PRIMARY_LITTLE (ASI_PST16_PL)</td>
<td>—</td>
<td>W 221</td>
</tr>
<tr>
<td>CB16</td>
<td>—</td>
<td>full</td>
<td>—</td>
<td>ASI_PST16_SECONDARY_LITTLE (ASI_PST16_SL)</td>
<td>—</td>
<td>W 221</td>
</tr>
<tr>
<td>CC16</td>
<td>—</td>
<td>full</td>
<td>—</td>
<td>ASI_PST32_PRIMARY_LITTLE (ASI_PST32_PL)</td>
<td>—</td>
<td>W 221</td>
</tr>
<tr>
<td>CD16</td>
<td>—</td>
<td>full</td>
<td>—</td>
<td>ASI_PST32_SECONDARY_LITTLE (ASI_PST32_SL)</td>
<td>—</td>
<td>W 221</td>
</tr>
<tr>
<td>D016</td>
<td>—</td>
<td>full</td>
<td>—</td>
<td>ASI_FL8_PRIMARY (ASI_FL8_P)</td>
<td>—</td>
<td>RW</td>
</tr>
<tr>
<td>D116</td>
<td>—</td>
<td>full</td>
<td>—</td>
<td>ASI_FL8_SECONDARY (ASI_FL8_S)</td>
<td>—</td>
<td>RW</td>
</tr>
<tr>
<td>D216</td>
<td>—</td>
<td>full</td>
<td>—</td>
<td>ASI_FL16_PRIMARY (ASI_FL16_P)</td>
<td>—</td>
<td>RW</td>
</tr>
<tr>
<td>D316</td>
<td>—</td>
<td>full</td>
<td>—</td>
<td>ASI_FL16_SECONDARY (ASI_FL16_S)</td>
<td>—</td>
<td>RW</td>
</tr>
<tr>
<td>D816</td>
<td>—</td>
<td>full</td>
<td>—</td>
<td>ASI_FL8_PRIMARY_LITTLE (ASI_FL8_PL)</td>
<td>—</td>
<td>RW</td>
</tr>
<tr>
<td>D916</td>
<td>—</td>
<td>full</td>
<td>—</td>
<td>ASI_FL8_SECONDARY_LITTLE (ASI_FL8_SL)</td>
<td>—</td>
<td>RW</td>
</tr>
<tr>
<td>DA16</td>
<td>—</td>
<td>full</td>
<td>—</td>
<td>ASI_FL16_PRIMARY_LITTLE (ASI_FL16_PL)</td>
<td>—</td>
<td>RW</td>
</tr>
<tr>
<td>DB16</td>
<td>—</td>
<td>full</td>
<td>—</td>
<td>ASI_FL16_SECONDARY_LITTLE (ASI_FL16_SL)</td>
<td>—</td>
<td>RW</td>
</tr>
<tr>
<td>E016</td>
<td>—</td>
<td>full</td>
<td>—</td>
<td>ASI_BLOCK_COMMIT_PRIMARY (ASI_BLK_COMMIT_P)</td>
<td>—</td>
<td>W 221</td>
</tr>
<tr>
<td>E116</td>
<td>—</td>
<td>full</td>
<td>—</td>
<td>ASI_BLOCK_COMMIT_SECONDARY (ASI_BLK_COMMIT_S)</td>
<td>—</td>
<td>W 221</td>
</tr>
<tr>
<td>E716</td>
<td>0016</td>
<td>bit&lt;7:0&gt;</td>
<td>8byte</td>
<td>ASI_SCCR</td>
<td>Chip RW 236</td>
<td></td>
</tr>
<tr>
<td>EF16</td>
<td>0016— 9816</td>
<td>bit&lt;7:0&gt;</td>
<td>8byte</td>
<td>ASI_LBSY, ASI_BST</td>
<td>Core RW 228</td>
<td></td>
</tr>
<tr>
<td>F016</td>
<td>—</td>
<td>full</td>
<td>—</td>
<td>ASI_BLOCK_PRIMARY (ASI_BLK_P)</td>
<td>—</td>
<td>RW</td>
</tr>
<tr>
<td>F116</td>
<td>—</td>
<td>full</td>
<td>—</td>
<td>ASI_BLOCK_SECONDARY (ASI_BLK_S)</td>
<td>—</td>
<td>RW</td>
</tr>
<tr>
<td>F216</td>
<td>—</td>
<td>full</td>
<td>—</td>
<td>ASI_XFILL_P</td>
<td>—</td>
<td>W 133</td>
</tr>
</tbody>
</table>
L.3.2 Special Memory Access ASIs

この節では SPARC64 IXfx で定義されている ASI について説明する。JPS1 Commonality の Section L.3.2 で定義されている ASI はここでは説明しないので、対応する節を参照。

ASI 53₁₆ (ASI_SERIAL_ID)

SPARC64 IXfx は個々の CPU チップ毎にユニークな ID コードを持っている。この ID コードと VER の情報から、完全に一意な CPU 識別を生成することができる。

このレジスタはリードオンリーで、書き込むとすると data_access_exception 例外が通知される。

<table>
<thead>
<tr>
<th>ASI VA</th>
<th>有効ビット</th>
<th>アライン</th>
<th>ASI 名 (と省略表記)</th>
<th>共有</th>
<th>アクセス ページ</th>
</tr>
</thead>
<tbody>
<tr>
<td>★ F₃₁₆</td>
<td>full</td>
<td>8byte</td>
<td>ASI_XFILL_S</td>
<td>—</td>
<td>W 133</td>
</tr>
<tr>
<td>○ F₈₁₆</td>
<td>full</td>
<td>64byte</td>
<td>ASI_BLOCK_PRIMARY_LITTLE (ASI_BLK_PL)</td>
<td>—</td>
<td>RW</td>
</tr>
<tr>
<td>○ F₉₁₆</td>
<td>full</td>
<td>64byte</td>
<td>ASI_BLOCK_SECONDARY_LITTLE (ASI_BLK_SL)</td>
<td>—</td>
<td>RW</td>
</tr>
</tbody>
</table>

![Chip_ID<63:0>](image)

ASI 4F₁₆ (ASI_SCRATCH_REGx)

SPARC64 IXfx はシステムソフトウェア用に 8 本の 64 ビットレジスタを用意している。

![Data<63:0>](image)

レジスタ名  | ASI_SCRATCH_REGx (x = 0–7)  |
-------------|-----------------------------|
ASI          | 4F₁₆                        |
VA          | VA<5:3> = レジスタ番号      |
             | 他のビットは 0 でなければならない。|
アクセス種別  | Supervisor read/write       |
Block Load and Store ASIs

ASI 番号 E016 と E116 は Block Store with Commit (page 66) と定義されており、STDF 命令でのみ使える ASI である。この ASI を LDDFA 命令で指定することはできない。指定した場合の動作は以下の通りである。

- デスティネーションレジスタ rd に関する例外は検出されない（impl. dep. #255）。
- メモリアドレスの境界条件によって、以下の例外が通知される（impl. dep. #256）。
  - 8 バイト境界ならば、data_access_exception で、DSFSR.FTYPE = 0816 (invalid ASI)
  - 4 バイト境界ならば、LDDF_mem_address_not_aligned
  - それ以外なら mem_address_not_aligned

Partial Store ASIs

ASI 番号 C016–C516, C816–CD16 は Partial Store (page 93) と定義されており、STDFA 命令でのみ使える ASI である。この ASI を LDDFA 命令で指定することはできない。指定した場合の動作は以下の通りである。

- メモリアドレスの境界条件によって、以下の例外が通知される（impl. dep. #257）。
  - 8 バイト境界ならば、data_access_exception で、DSFSR.FTYPE = 0816 (invalid ASI)
  - 4 バイト境界ならば、LDDF_mem_address_not_aligned
  - それ以外なら mem_address_not_aligned

L.3.3 ASI と命令の組み合わせと例外

SPARC64 IXfx では、未定義の ASI や無効な命令と ASI の組み合わせにより起こる例外が、JPSI Commonality の定義と一部異なる。この節では SPARC64 IXfx での定義を、実際に通知される優先順位に沿って説明する。

1. Block Load/Store, Partial Store 命令では illegal_instruction が通知される場合がある。
   詳細は各命令の定義を参照。LDDA, STDA の rd に奇数番号のレジスタを指定した場合も illegal_instruction が通知される。

2. 命令によって決まるアラインメント条件が検査され、違反していると
   mem_address_not_aligned, *_mem_address_not_aligned が通知される。

a. block load, block store 命令は 64 バイトアラインメントを要求する命令なので、64 バイト境界にないアドレスをアクセスすると mem_address_not_aligned が通知される。LDDF_mem_address_not_aligned, STDF_mem_address_not_aligned は通知されない。

LDDFA 命令で block store with commit ASI を指定した場合は、block load, block store 命令ではないので、この項には当てはまらない。
b. 16 ビット short load, short store 命令は 2 パイトアラインメントを要求する命令なので、2 パイト境界にないアドレスをアクセスすると mem_address_not_aligned が通知される。LDDF_mem_address_not_aligned, STDF_mem_address_not_aligned は通知されない。

c. 8 ビット short load, short store 命令は 1 パイトアラインメントを要求する命令なので、アラインメント違反は起きない。

d. partial store 命令は 8 パイトアラインメントを要求する命令なので、8 パイト境界にないアドレスをアクセスすると mem_address_not_aligned が通知される。

LDDF_mem_address_not_aligned, STDF_mem_address_not_aligned は通知されない。

LDDFA 命令で partial store ASI を指定した場合は、partial store 命令ではないので、この項には当てはまらない。

e. LDDFA, ST DFA で上記以外の ASI を指定し、4 パイト境界にアクセスすると、それぞれ LDDF_mem_address_not_aligned, STDF_mem_address_not_aligned が通知される。

f. 上記以外のアラインメント違反には、mem_address_not_aligned が通知される。

上記 e および f は、ASI が定義済みか未定義か、ASI と命令の組み合わせが正しいかどうかより優先されるので、data_access_exception （FT = 0816）は通知されない。

3. ASI と命令の組み合わせが正しくない場合、data_access_exception が通知される。

ただし PREFETCHA では data_access_exception が通知されず、nop として処理される。

L.3.4 内部レジスタの更新タイミング

SPARC64 IXfx では、nontranslating ASI のほとんどが、CPU 内部レジスタにマップされている。CPU 内部レジスタは MMU 関連やハードウェアバリアなど、副作用を伴うもののがほとんどだが、nontranslating ASI のアクセスがプログラム順序になることを保証しない。CPU 内部レジスタの更新による効果（副作用）を、レジスタ変更直後の命令に見せるためには、ソフトウェアによる明示的な membar #Sync が必要である。

L.4 ハードウェアバリア

SPARC64 IXfx は CPU チップ内で高速に同期を取るためのハードウェアバリア機構を提供する。バリア機構は CPU チップ内にあり、全コアで共有されている。FIGURE L-1 は SPARC64 IXfx のバリア資源の構成図である
主要な資源は Barrier Blade (BB) と呼ばれるもので、SPARC64 IXfx には 24 個ある。各 BB は BST (Barrier Sstatus bit) と BST のマスクビット、それに前回同期したときの値を記憶しておく LBSY (Last Barrier Synchronization status) を持っている。24 個ある BB のうち 8 個は、BST, BST_mask は 16 ビット長で、それぞれがチップ内のコアに固定的に対応している。残り 16 個では BST が 1 ビット長で BST_mask は存在しない。前者は複数スレッド間のバリア同期用、後者は 2 スレッド間での post-wait 同期用を図っている。BB の番号は、バリア同期用が 0-3 および 16-19, post-wait 用が 4-11 および 20-27 となっている。12-15 は欠番となっている。

同期が成立するのは、BST の選択されたビット (BST_mask で選択する) がすべて 0 または 1 のどちらかに揃ったときである。同期が成立すると、その値 (0 か 1) が LBSY にコピーされる。同期成立と LBSY へのコピーは单一の操作として行われるの
で、同期成立前に LBSY を読み出すと必ず古い値が読め、同期成立後は必ず新しい値が読める。したがって、ソフトウェアが同期を取る手順は、まず LBSY を読み出し、BST を更新した後で LBSY が変化するのを待つという順序になる。LBSY の変化を監視するためには普通スピンループが使われるが、複数コア・スレッドで資源を共有する CPU では、スピンループは CPU 資源を浪費し、他コア・スレッドの実行を阻害するおそれがある。SPARC64 IXfx では LBSY の値が変化したとき、SLEEP 命令による休止状態にあるコア・スレッドを実行状態に復帰させる。これにより、高速な同期と CPU 資源の効率的な利用の両立を可能にしている。

LBSY は最後に同期したときの値を覚えているので、ソフトウェアは、次の同期で BST にセットする値を容易に決することができる。すなわち、LBSY から読み出した値が 0 なら 1 を、1 なら 0 を BST に書き込めばよい。

各コア・スレッドは、また、BB にアクセスするための窓 ASI を 20 個持っている。ユーザプログラムはバリア資源に直接アクセスするのではなく、窓を通じてアクセスすることになる。窓を設けることで、BST ビットマップを隠蔽し、単一の操作を可能にし、他のユーザプログラムから同期を破壊するような操作を防ぐことができる。窓 ASI はバリア用、post-wait 用に分かれており、バリア用窓 ASI にはバリア用 BB だけを、post-wait 用窓 ASI には post-wait 用 BB だけを割りつけることができる。

バリア資源のメモリモデルは JPS1 Commonality の Chapter 8 で定義されている TSO に準拠する。これは BB 同士、BB とメモリ間いずれにおいても成立する。つまり、store の後の load 以外は命令列に現れた順序で実行される。窓 ASI に対する store の後でメモリまたは LBSY を読み出す場合には、間に membar #storeload を挟む必要がある。

兼容性注意 — SPARC64 VIIIfx では窓 ASI に制限なく、任意の BB を任意の窓に割りつけることができた。SPARC64 VIIIfx と SPARC64 IXfx のどちらでも動くソフトウェアを制作するためには、SPARC64 IXfx の窓 ASI の制約にあわせること。

ノート — SPARC64 IXfx は CPU チップ間のバリア同期はサポートしない。

L.4.1 バリア資源の初期化と状態獲得

<table>
<thead>
<tr>
<th>レジスタ名</th>
<th>ASI_BARRIER_INIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASI</td>
<td>6D16</td>
</tr>
<tr>
<td>VA</td>
<td>バリア用 0016 0816 1016 1816 8016 8816 9016 9816</td>
</tr>
<tr>
<td></td>
<td>post-wait 用 2016 2816 3016 3816 4016 4816 5016 5816</td>
</tr>
<tr>
<td></td>
<td>A016 A816 B016 B816 C016 C816 D016 D816</td>
</tr>
</tbody>
</table>

アクセス種別 Supervisor read/write
Note – UBST_maskとBST_maskは連結してひとつのビットマップフィールドを構成している。以下の説明では、曖昧さがない場合、UBST_mask, BST_maskを区別せずBST_maskと表記することがある。BST_mask<8>にはUBST_mask<0>を意味する。UBST_value, BST_valueも同様。

VAで指定されるBarrier Bladeの値の取得および初期化を行う。readで現在の設定が読み出され、writeで新しい設定を書くことができる。

BST_mask, BST_valueがバリアのグループ構成とバリアの状態を表わす。各ビットは各コアに対応している。BST_maskには、Barrier Bladeを使うコアのビットを1、使わないコアのビットを0に設定する。

### ビット フィールド名 アクセス 説明

<table>
<thead>
<tr>
<th>ビット</th>
<th>フィールド名</th>
<th>アクセス</th>
<th>説明</th>
</tr>
</thead>
<tbody>
<tr>
<td>47:40</td>
<td>UBST_mask</td>
<td>RW</td>
<td>UBST_maskとBST_maskでBSTのマスクを指示・読み出す。各ビットとコアの対応は、BB#0–BB#3, BB#16–BB#19では、ASI_BST_BITで読み出された値がそのコアのビット位置となる。BST_mask&lt;i&gt;コアiのマスク。(0≤i&lt;8)UBST_mask&lt;i&gt;コアiのマスク。(8≤i&lt;16)BB#4–BB#11, BB#20–BB#27にはBST_maskは存在しない。</td>
</tr>
<tr>
<td>39:32</td>
<td>UBST_value</td>
<td>RW</td>
<td>UBST_valueとBST_valueでBSTの値を指示・読み出す。各ビットとコアの対応は、BB#0–BB#3, BB#16–BB#19では、ASI_BST_BITで読み出された値がそのコアのビット位置となる。BST_value&lt;i&gt;コアiのBST。(0≤i&lt;8)UBST_value&lt;i&gt;コアiのBST。(8≤i&lt;16)BB#4–BB#11, BB#20–BB#27では、BST_value&lt;0&gt;コア0–15のBST。</td>
</tr>
<tr>
<td>16</td>
<td>LBSY</td>
<td>RW</td>
<td>最後に同期したときのBSTの値。</td>
</tr>
<tr>
<td>15:8</td>
<td>BST_mask</td>
<td>RW</td>
<td>UBST_value参照。</td>
</tr>
<tr>
<td>7:0</td>
<td>BST_value</td>
<td>RW</td>
<td>UBST_value参照。</td>
</tr>
</tbody>
</table>

読み出し時は、VAで指定されるBarrier BladeのBST_value, BST_mask, LBSYが読み出される。

BB#0–BB#3およびBB#16–BB#19では、BST_value, BST_maskの各ビットはそれぞれが固有のコアに対応している。BST_maskのあるビットが0のとき、BST_valueの対応するビットに読み出される値は不定である。

Ver 12, 2 Dec. 2013
post/wait 用 BB では、LBSY, BST_value<0> のみが意味を持ち、他のビットは 0
が読み出される。

書き込み時は、VA で指定される Barrier Blade の BST_value, BST_mask, LBSY が更
新される。

BB#0–BB#3 および BB#16-BB#19 では、BST_mask と BST_value の各ビットは
それぞれが固有のコアに対応している。BST_mask のあるビットが 0 で、
BST_value の対応するビットが 1 であるような値を書こうとした場合、その値が
書かれるかどうかは未定義である。

post/wait 用 BB では、LBSY, BST_value<0> のみが意味を持ち、他のビットの書
き込みは無視される。

書き込み後、ハードウェアはバリア同期が成立しているかどうかを確認し、それに応
じて LBSY を更新する。BST_value, BST_mask にすべて 1 を、LBSY に 0 を書き込
むと、LBSY は直ちに 1 に更新される。

bst_mask = 0 の場合バリア同期が成立しているかどうかの確認は行われず、LBSY
に書き込んだ値がそのまま保持される。

L.4.2 BST ビット位置の取得

<table>
<thead>
<tr>
<th>レジスタ名</th>
<th>ASI_BST_BIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>VA</td>
<td>0816</td>
</tr>
<tr>
<td>アクセス種別</td>
<td>Supervisor readonly</td>
</tr>
</tbody>
</table>

ASI_BST_BIT は BST_mask および BST_value のビット位置を取得する ASI であ
る。コアごとに 0-15 のユニークな値が読み出される。
バリア資源の割り付け

ASI_BARRIER_ASSIGN は、窓ASI（ASI_BST, ASI_LBSY）の割付け状態の取得および変更を行うASIである。VAはASI_BST, ASI_LBSYのそれに対応しており、VAで指定された窓に、BB_numで指定されたBBを割りつける、あるいはVAで指定された窓の割りつけを解除することができる。

読み出しに対しては、どのBBが割付けられているかが返される。VAで指定された窓がBBに割り付けられているなら、valid=1となりBB_numにはBB番号が表示される。VAで指定された窓がBBに割り付けられていないときは、valid=0となりBB_numの値は不定である。

書き込みに対しては、

- valid=1の場合は、指定されたBB_numのLBSY,BSTを窓に割りつける。この書き込みの完了以降、ASI_BSTへの書き込みがBBのBSTに反映されるようになり、ASI_LBSYの読み出しによりBB_numにあるLBSYが読み出すようになる。
- valid=0の場合は、指定された窓の割りつけを解除する。この書き込みの完了以降、ASI_BSTへの書き込みは無視され、ASI_LBSYの読み出しには不定値が返る。

窓ASIはバリア用とpost-wait用に用途が分かれている。バリア用ASIにはバリア用BB(#0-#3および#16-#19)が割り付け可能であり、post-wait用ASIにはpost-wait用BB(#4-#11および#20-#27)が割り付け可能である。バリア用ASI
プログラミングノート – 業 ASI と BB の対応を間違えて指定しても例外は通知されない。正しく設定されたことを確認するためには、ASI_BARRIER_ASSIGN に書き込み後に読み出して、valid = 1 で指定した BB_num が設定されていることを確かめるとよい。

ASI_BARRIER_INIT と ASI_BARRIER_ASSIGN で矛盾するような設定をした場合の動作は不定である。ハードウェアでは矛盾を検出しないので、ソフトウェアは、バリア資源の初期化・割当てに際して矛盾を起こさないようにすること。使用中の Barrier Blade に ASI_BARRIER_INIT を発行する、BST_mask<i> = 0 である BST<i> を ASI_BARRIER_ASSIGN で割りつける、などの場合の同期処理は保証されない。

プログラミングノート – システムソフトウェアは初期化済の BB を割り当てる。未初期化の BB を割り当たると意図しない結果が起きたことがある。

L.4.4 バリア操作用 ASI

<table>
<thead>
<tr>
<th>レジスタ名</th>
<th>ASI_LBSY (read), ASI_BST (write)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASI</td>
<td>EF16</td>
</tr>
<tr>
<td>VA</td>
<td>バリア用 0016, 0816, 1016, 1816</td>
</tr>
<tr>
<td></td>
<td>post-wait 用 2016, 2816, 3016, 3816, 4016, 4816, 5016, 5816, 6016, 6816, 7016, 7816, 8016, 8816, 9016, 9816</td>
</tr>
</tbody>
</table>

アクセス種別 Read/Write

<table>
<thead>
<tr>
<th>—</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>63</td>
<td>1</td>
</tr>
</tbody>
</table>
ASI_LBSY, ASI_BST はバリア資源にアクセスするための窓 ASI で、各コア・スレッドに 20 個用意されている。ユーザモードのプログラムが直接アクセスできる。

<table>
<thead>
<tr>
<th>ビット</th>
<th>フィールド名</th>
<th>アクセス</th>
<th>説明</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Value</td>
<td>RW</td>
<td>読み出すと LBSY の値が読み、書き込むと BST が更新される。</td>
</tr>
</tbody>
</table>

割り付けられていない窓 ASI に対する読み出しは不定値が返り、書き込みは無視される (例外は通知されない)。

Sample Code for Barrier Synchronization

```c
/*
 * %r1: VA of a window ASI
 * %r2, %r3: work registers
 */

ldxa [%r1]ASI_LBSY, %r2          // 現在の LBSY を読み出す
not %r2                           // LBSY を反転させる
and %r2, 1, %r2                   // Reserved フィールドを捨ててる
stxa %r2, [%r1]ASI_BST            // BST を更新する
membar #storeload                 // stxa が完了するのを待つ
loop:
   ldxa [%r1]ASI_LBSY, %r3        // LBSY を読み出す
   and %r3, 1, %r3                // Reserved フィールドを捨ててる
   subcc %r3, %r2, %g0            // 値が変化したか?
   bne,a loop                     // 変化していなければ sleep する
sleep
```
M.1 キャッシュタイプ

SPARC64 IXfx はチップ上に 2 階層のキャッシュを持つ。

- L1 キャッシュは命令とデータに分かれており、L2 キャッシュはユニファイドキャッシュである。
- L1 キャッシュは論理アドレスインデックス・物理アドレスタグ (VIPT) で、L2 キャッシュは物理アドレスインデックス・物理アドレスタグ (PIPT) である。
- L1 キャッシュ、L2 キャッシュともラインサイズは 128 バイトである。
- L1 キャッシュと L2 キャッシュは包含関係にあり、L1 キャッシュのすべてのデータは L2 キャッシュ上にも載っている。
- L1 命令キャッシュと L1 データキャッシュ間、L1 キャッシュと L2 キャッシュ間のデータの同一性はハードウェアによって維持される。すなわち、
  - L2 キャッシュのあるキャッシュライン上のデータを無効化する場合、L1 キャッシュ上にデータがあればそれでも無効化する。
  - 命令列を変更すると、L1 データキャッシュ上のデータが更新され、L1 命令キャッシュの命令列は無効化される。
- L2 キャッシュはプロセッサモジュール上のすべてのコアで共有される。
M.1.1 L1 命令キャッシュ (L1I キャッシュ)

L1 命令キャッシュの諸元を以下に示す。

<table>
<thead>
<tr>
<th>諸元</th>
<th>値</th>
</tr>
</thead>
<tbody>
<tr>
<td>総容量</td>
<td>32 KB</td>
</tr>
<tr>
<td>ウェイ数</td>
<td>2 ウェイ</td>
</tr>
<tr>
<td>ラインサイズ</td>
<td>128 バイト</td>
</tr>
<tr>
<td>インデックス方式</td>
<td>論理アドレスインデックス・物理アドレスタグ (VIPT)</td>
</tr>
<tr>
<td>エラー保護方式 (タグ)</td>
<td>二重化とパリティ</td>
</tr>
<tr>
<td>エラー保護方式 (データ)</td>
<td>パリティ</td>
</tr>
<tr>
<td>その他</td>
<td>—</td>
</tr>
</tbody>
</table>

SPARC64 IXfx はハードウェアがエイリアス処理を行うため、L1 命令キャッシュは VIPT であるが TTE.CV は意味を持たない。

ノンキャッシャブル領域から命令をフェッチする場合、L1 命令キャッシュには命令列は載らない。ノンキャッシャブルアクセスが起きるのは、以下の 3 つの場合である。

■ PSTATE.RED = 1
■ DCUCR.IM = 0
■ TTE.CP = 0

MCNTL.NC_CACHE = 1 の場合、上の条件に係わらず SPARC64 IXfx はすべての命令列をキャッシャブル領域にあるものとして扱う。詳細は “ASI_MCNTL (Memory Control Register)” (page 184) を参照。

Programming Note – この機能は OBP のために用意されている。OBP がこの機能を使うときは、OBP から抜ける前に MCNTL.NC_CACHE に 0 をセットし、ASI_FLUSH_L1I で L1 命令キャッシュの内容を無効にすること。
M.1.2 L1 データキャッシュ (L1D キャッシュ)

L1 データキャッシュはライトバックキャッシュである。以下に諸元を示す。

<table>
<thead>
<tr>
<th>諸元</th>
<th>値</th>
</tr>
</thead>
<tbody>
<tr>
<td>総容量</td>
<td>32 KB</td>
</tr>
<tr>
<td>ウェイ数</td>
<td>2 ウェイ</td>
</tr>
<tr>
<td>ラインサイズ</td>
<td>128 バイト</td>
</tr>
<tr>
<td>インデックス方式</td>
<td>論理アドレスインデックス・物理アドレスタグ (VIPT)</td>
</tr>
<tr>
<td>エラー保護方式 (タグ)</td>
<td>二重化とパリティ</td>
</tr>
<tr>
<td>エラー保護方式 (データ)</td>
<td>ECC</td>
</tr>
<tr>
<td>その他</td>
<td>セクタキャッシュ</td>
</tr>
</tbody>
</table>

SPARC64 IIXfx はハードウェアがエイリアス処理を行うため、L1 データキャッシュは VIPT であるが TTE.CV は意味を持たない。

ノンキャッシャブル領域上のデータにアクセスする場合、そのデータは L1 データキャッシュには載らない。ノンキャッシャブルアクセスが起きるのは、以下の 3 つの場合である。

- ASI_PHYS_BYPASS_EC_WITH_B_BIT (1516) または ASI_PHYS_BYPASS_EC_WITH_B_BIT_LITTLE (1D16) 経由のアクセス。
- DCUCR.DM = 0
- TTE.CP = 0

L1 データキャッシュは、MCNTL.NC_CACHE の値に係わらずノンキャッシャブル領域のデータはキャッシュに載せない。

M.1.3 L2 キャッシュ

L2 キャッシュはライトバックキャッシュである。以下に諸元を示す。

<table>
<thead>
<tr>
<th>諸元</th>
<th>値</th>
</tr>
</thead>
<tbody>
<tr>
<td>総容量</td>
<td>12MB</td>
</tr>
<tr>
<td>ウェイ数</td>
<td>24 ウェイ</td>
</tr>
<tr>
<td>ラインサイズ</td>
<td>128 バイト</td>
</tr>
<tr>
<td>インデックス方式</td>
<td>物理アドレスインデックス・物理アドレスタグ (PIPT)</td>
</tr>
<tr>
<td>エラー保護方式 (タグ)</td>
<td>ECC</td>
</tr>
<tr>
<td>エラー保護方式 (データ)</td>
<td>ECC</td>
</tr>
<tr>
<td>その他</td>
<td>インデックスハッシュ、セクタキャッシュ</td>
</tr>
</tbody>
</table>

L2 キャッシュは、MCNTL.NC_CACHE の値に係わらずノンキャッシャブル領域のデータはキャッシュに載せない。
インデックスハッシュ

SPARC64 IXfxはL2キャッシュのインデックスにハッシュをかける。ハッシュの計算式は以下の通り。

- \( \text{index}_{11:9} = \text{PA}_{33:31} \oplus \text{PA}_{30:28} \oplus \text{PA}_{27:25} \oplus \text{PA}_{24:22} \oplus \text{PA}_{21:19} \oplus \text{PA}_{18:16} \)
- \( \text{index}_{8:0} = \text{PA}_{15:7} \)

M.2 キャッシュコヒーレンシープロトコル

Note – SPARC64 IXfxはマルチプロセッサ構成をサポートしないので、この節は削除した。
M.3 キャッシュ制御 ASI

M.3.1 命令キャッシュフラッシュ (ASI_FLUSH_L1I)

ASI_FLUSH_L1I は、それを実行したコアの L1 命令キャッシュの内容を全部無効にする。起動するには 8 バイト境界にある任意の VA に任意の値を書き込む。

ASI_FLUSH_L1I は書き込みだけが可能であり、読み出しには data_access_exception 例外が通知される。

M.3.2 キャッシュデータの無効化 (ASI_CACHE_INV)

ASI_CACHE_INV は、CPU チップ内の全コアの L1 キャッシュおよび L2 キャッシュの特定のキャッシュラインについて、必要ならばその内容をメモリに書き出した上で無効にする。キャッシュラインは VA に物理アドレスを指定する。

ASI_CACHE_INV は書き込みのみ可能であり、読み出しには data_access_exception 例外が通知される。

Note – DCUCR.WEAK_SPCA = 0 だと、ASI_CACHE_INV を発行した時点では無効化されるが、その後の投機実行やハードウェアプリフェッチによりキャッシュに載るかもしれないので、キャッシュ上にデータが残らないことを保証したい場合は、この命令を実行する前に DCUCR.WEAK_SPCA を 1 にセットすること。
M.3.3 セクタキャッシュ設定 (SCCR)

ASI_SCCR はセクタキャッシュの設定を制御するレジスタである。SCCR は CPU チップでひとつの資源であり、全コアで共有されている。

<table>
<thead>
<tr>
<th>ビット</th>
<th>フィールド名</th>
<th>アクセス</th>
<th>説明</th>
</tr>
</thead>
</table>
| 63    | NPT          | RW      | 特権アクセス。
|       |              |         | NPT = 1 のとき、PSTATE.priv=0 で SCCR にアクセスすると privileged_action 例外が通知される。
|       |              |         | NPT = 0 のときは PSTATE.priv の値によらず、読み出し・書き込みができる。ただし、PSTATE.priv=0 の書き込みでは、NPT は更新されない。 |
| 62:21 | —            |         | Reserved. |
| 20:16 | L2_sector0_max | RW  | L2 キャッシュのセクタ 0 で使用する最大ウェイ数 |
| 15:13 | —            |         | Reserved. |
| 12:8  | L2_sector1_max | RW  | L2 キャッシュのセクタ 1 で使用する最大ウェイ数 |
| 7:6   | —            |         | Reserved. |
| 5:4   | L1_sector0_max | RW  | L1 キャッシュのセクタ 0 で使用する最大ウェイ数。
|       |              |         | あるコアで設定を変更すると、全コアの L1 キャッシュの設定が変更される。 |
| 3:2   | —            |         | Reserved. |
| 1:0   | L1_sector1_max | RW  | L1 キャッシュのセクタ 1 で使用する最大ウェイ数。
|       |              |         | あるコアで設定を変更すると、全コアの L1 キャッシュの設定が変更される。 |

Warning – SCCR は全コアで共有されているので、あるプロセスがセクタキャッシュを使っているとき、別のコアから SCCR を変更すると、そのプロセスの実行性能に影響する。
Compatibility Note – SPARC64 IXfxでは、ユーザ権限のプロセスがSCCR.NPTに1をセットすることはできないので、ユーザプロセスが、セクタキャッシュを使っているユーザプロセスを異常終了させることはできない。

SPARC64 IXfxはキャッシュを2つのセクタと呼ばれる部分に分けて更新管理をする仕組みを導入する。この機構をセクタキャッシュと呼ぶ。セクタの指示はメモリアクセス命令単位で指定することができ、アクセスされたデータがキャッシュの指定されたセクタに格納される。SPARC64 IXfxではL1キャッシュ、L2キャッシュとともにセクタキャッシュ機構を実装しており、セクタキャッシュ機能を有効にするかどうかは、L1キャッシュ、L2キャッシュともに独立に設定できる。

セクタの容量はウェイ数で指定する。セットアソシエイティブキャッシュでは、あるインデックスには複数のウェイが存在するが、このうちセクタ0に属する最大ウェイ数と、セクタ1に属する最大ウェイ数を指定する。セクタ容量の指定は全インデックスで共通であり、インデックス毎に個別の指定はできない。

セクタキャッシュ機能が有効になるのは、セクタ0の容量、セクタ1の容量とも1以上の値を指定したときである。キャッシュウェイ数より大きな値を指定した場合は、キャッシュウェイ数が指定されたものとみなす。セクタ0の最大ウェイ数とセクタ1の最大ウェイ数の和は、キャッシュのウェイ数と等しくなくてもよい。どちらかのセクタのウェイ数が0の場合はセクタキャッシュ機能は無効となる。

セクタキャッシュ機能は、キャッシュデータの入れ替え時の動作の違いとしてだけ見える。セクタキャッシュ機能が有効のときは、追い出すエントリは全ウェイから選択される。セクタキャッシュ機能が有効のときは、各セクタ毎に設定された最大値を超えないように、追い出すエントリを選ぶ。すなわち、あるインデックスにおいて当該セクタのデータ数が最大値よりも小さければ、当該セクタでない側から追い出すエントリを選ぶ。セクタキャッシュ機能が有効であるかどうかと、キャッシュ上に載っているデータへのアクセスも無関係で、ソフトウェアは常に全ウェイのデータにアクセスできる。そのデータのセクタと異なるセクタを指定してアクセスした場合、そのデータが所属するセクタが変更される。

Notes – データ読み出しやプリフェッチでも、セクタ情報は変更される。セクタ情報はキャッシュライン単位なので、ライン内のデータ毎に異なるセクタ情報を指定すると、最後にアクセスされたときのセクタが指定されたことになる。

メモリアクセス命令(load/store/atomic/prefetch)でキャッシュセクタを指定するには、XAR.sector(XAR.urs3<0>)を使用する。XAR.sector=0ならセクタ0が、XAR.sector=1ならセクタ1が指定される。

セクタ指定情報とセクタキャッシュ機能は独立した概念である。セクタ指定情報はデータの属性であり、セクタキャッシュ機能はキャッシュ入れ替え時の動作である。セクタキャッシュ機能が無効であっても、セクタ指定情報は常に保存されている。
とえば L1 キャッシュのセクタキャッシュ機能が無効、L2 キャッシュのセクタキャッシュ機能が有効であっても、L1 キャッシュのデータが L2 キャッシュにライトバックされる際は、L2 キャッシュのセクタ情報は適切に更新される。

**Implementation Note** – L1 キャッシュ上のセクタ情報の変化を L2 キャッシュに伝達する手段とタイミングは実装依存とする。

各セクタの最大ウェイ数は、キャッシュ上のデータ更新時に参照される情報であり、ウェイ数設定時点で最大ウェイ数を越えるウェイが当該セクタに割り当てられていたとしても、制約的に無効にはしない。例えばあるインデックスでセクタ 0 が 5 ウェイ使用しているときに、セクタ 0 の最大ウェイ数に 2 が指定されたとしても、最大数を越えた 3 ウェイはすぐに無効にされるわけではない。この意味で最大ウェイ数は制御の目標値の役割をしているといえる。

各セクタの用途は本仕様では規定しない。

セクタキャッシュの動作アルゴリズムを説明する。このアルゴリズムは L1 キャッシュ、L2 キャッシュとも同じである。なお、以下の説明ではフィールド名の L1_、L2_ は省略し、キャッシュのウェイ数を nway と表記する。

**SCCR 設定値**

- sector0_max > 0 かつ sector1_max > 0 のときセクタキャッシュ有効
- sector0_max = 0 または sector1_max = 0 のときはセクタキャッシュ無効
- sector0_max + sector1_max = nway は成立する必要なし。

**セクタキャッシュ管理動作**

セクタ 0 として使用されているウェイ数を sector0_use、セクタ 1 として使用されているウェイ数を sector1_use と表す。常に以下が成立する。

\[
\text{sector0_use} + \text{sector1_use} \leq \text{nway}
\]
\[
0 \leq \text{sector0_use} \leq \text{nway}, 0 \leq \text{sector1_use} \leq \text{nway}
\]

セクタ番号 S に対するメモリアクセスが要求された場合の動作

- キャッシュにヒットした場合
  - ヒットしたウェイのセクタが S と異なる場合はウェイ数管理を調整する。
    
    \[
    \text{sector}_S\text{use}++ \text{, sector}_T\text{use}-- (\text{セクタ } T \text{ でないセクタ})
    \]
    
    \[
    \text{sector}_S\text{use} > \text{sector}_S\text{max} \text{ になり得る(ただし sector}_S\text{max < nwayのとき)}
    \]

- キャッシュミスした場合
  - 空のウェイがある場合、そのウェイをセクタ S とする。
    
    \[
    \text{sector}_S\text{use}++
    \]

238  SPARC64™ IXfx Extensions Ver 12, 2 Dec. 2013
空のウェイがある場合、sector\$use は sector\$max より大きな値になり得る。

- sector\$use < min(nwway, sector\$max) の場合、セクタ T の最も古いウェイを置き換え、セクタ S とする。
  
  \[\text{sector\$use}++\,\text{sector\$use}--\]

- sector\$use ≥ min(nwway, sector\$max) の場合、セクタ S の最も古いウェイを置き換え、セクタ S とする。
  
  \[\text{sector\$use}, \text{sector\$use} は変化しない}\]

  \[\text{sector\$use} > \text{min(nwway, sector\$max)} でも sector\$use は減少しない。\]

セクタキャッシュ無効時の動作

- キャッシュミスして空ウェイがない場合、リプレースウェイの選択に
  
  sector\$use, sector\$use を使わず、全ウェイから最も古いウェイを選択する。

セクタキャッシュ機能が無効でも、セクタ指定情報は保存される。

---

**Note** – SPARC64 IXfx はメモリアクセスをアウトオブオーダで処理するので、ユーザプログラムの意図通りにセクタ情報が更新されない可能性がある。

XAR.ssector はすべてのメモリアクセス命令で指定可能であるが、意味があるのは
TTE.CP = 1 である空間に対するアクセスのみである。TTE.CP = 0 である空間や nontranslating ASI に対するアクセスでは XAR.sccs の値は無視され、例外は通知されない。

---

**M.4 ハードウェアプリフェッチ**

SPARC64 IXfx では、連続するキャッシュブルアドレスに対するアクセスを検出する
とハードウェアがプリフェッチを生成する機構が実装されている。対象はキャッシュブル空間へのロード命令、ストア命令であり、PREFETCH、PREFETCHA、LDSTUB、
LDSTUBA、SWAP、SWAPA、CASA、CASXA、block load/store、partial store、short load/store、
xfill は対象外。

---

1. ここで言う連続アドレスは、キャッシュライン (128 バイト) 単位での連続を意味する。
ハードウェアプリフェッチ機構の動作概要は以下の通り。

1. ld/st 命令が L1 キャッシュをミスすると（アドレス A），そのアドレスの隣接ライン（A+128, A-128）へのアクセス監視を始める。

2. 監視しているアドレスへのアクセスがあると（たとえば A+128）、さらに隣接するライン（A+256）のプリフェッチを生成し、同時にそのライン（A+256）への ld/st アクセスを監視する。

3. A+256 への ld/st アクセスで、A+384 のプリフェッチを生成する。アクセス監視はキャッシュミスを契機とし、連続アクセスかどうかはキャッシュアクセス（ヒットかミスかによらない）で判断する。

連続回数が増えると、より遠くをプリフェッチするようになり、また、L1キャッシュへのプリフェッチも併用するようになる。（最初は L2キャッシュプリフェッチのみ）。

ソフトウェアからハードウェアプリフェッチ機構を制御する方法は 2 つある。

1. ASI_MCNTL.hpf でハードウェアプリフェッチ機構全体の on/off を制御できる。詳細は "ASI_MCNTL (Memory Control Register)" (page 184) 参照。

2. XAR.dis_hw_pf で命令単位での on/off を制御できる。
   XAR.dis_hw_pf = 1 のとき、ld/st 命令が L1 キャッシュミスしても、隣接アドレスのキャッシュミスを監視しない。XAR.dis_hw_pf = 0 のときは、ld/st 命令の L1キャッシュミスで、隣接アドレスのキャッシュミス監視を始める
   (ASI_MCNTL.hpf = 1 のとき)。

Note – SPARC64 IXfx 仕様では、ハードウェアプリフェッチ機構が生成するプリフェッチの種類が何になるかは定義しない。

XAR.dis_hw_pf はすべてのメモリアクセス命令で指定可能であるが、意味があるのは TTE.CP = 1 である空間に対するロード、ストア命令のみである。TTE.CP = 0 である空間や nontranslating ASI に対するアクセスおよび、PREFETCH, PREFETCHA, LDSTUB, LDSTUBA, SWAR, SWAPA, CASA, CASXA, block load/store, short load/store, xfill では XAR.dis_hw_pf の値は無視され、例外は通知されない。

1. partial store は XAR 非対象命令なので、ハードウェアプリフェッチの指定はできない。
Interrupt Handling

N.1 Interrupt Vector Dispatch

あるプロセッサ1から別のプロセッサに対してインタラプトを発行するとき、ソフトウェアはまずインタラプトのデータを ASI_INTR_DATA_{0-2}Wにセットし、次に ASI_INTR_DISPATCH_Wに書き込みを行いインタラプトを送出する。送り側のプロセッサは INTR_DISPATCH_STATUS の BUSY ビットをポーリングし、送信が成功したかどうかを確認する。FIGURE N-1 にインタラプト送信の手順を示す。

1. ここではハードウェアの命令実行の主体。 SPARC64 IXfs ではコアと同義。
read ASI_INTR_DISPATCH_STATUS

Error

Busy? Y N

PSTATE.IE ← 0
(begin atomic sequence)

Write ASI_INTR_W (data 0)

Write ASI_INTR_W (data 2)

Write ASI_INTR_W (interrupt MEMBAR dispatch)

read ASI_INTR_DISPATCH_STATUS

Busy? Y N

PSTATE.IE ← 1
(end atomic sequence)

Nack? Y

dispatch complete

FIGURE N-1 Dispatching an Interrupt
インタラプトパケットを受信すると、受信データが ASI_INTR_DATA_[0-2]R に格納され、ASI_INTR_RECEIVE レジスタの BUSY ビットに 1 がセットされる。受信したプロセッサがインタラプトを許可している (PSTATE.IE = 1) とトラップが起きる。ソフトウェアはデータを読み出し、その内容に応じてハンドラを選択する。ハンドラによっては、処理を委譲するため、より優先度の低いトラップを起こすこともある。

受信パケットにエラーがあった場合、ASI_INTR_RECEIVE レジスタの BUSY ビットに 1 がセットされない。この場合、ASI_INTR_DATA_[0-2]R もエラーしているあるかも知れないので、読み出してはいけない。詳細は Appendix P.8.3, “ASI レジスタの処理方法” (page 291) を参照。

FIGURE N-2 にインタラプト受信の手順を示す。
N.4 Interrupt ASI Registers

N.4.1 Outgoing Interrupt Vector Data<7:0> Register

JPS1 Commonality ではインタラプト送信レジスタを 8 つ定義しているが、
SPARC64 IXfx ではこれを 3 つにする。ASI_INTR_DATA_[3–7]W に対する書き込み
は未定義の ASI に準拠する。

Compatibility Note –  この変更は SPARC JPS1 非互換である。

N.4.2 Interrupt Vector Dispatch Register

SPARC64 IXfx では、ASI_INTR_DISPATCH_W レジスタの書き込みの際、SID<9:0>
(= VA<38:29>) の全 10 ビットが無視される (impl.dep.#246)。

SPARC64 IXfx は BUSY/NACK ビットは 16 組までを実装する。
ASI_INTR_DISPATCH_W レジスタの書き込みの際、BN<4> (= VA<28>) は無視される。

SPARC64 IXfx では、ITID<9:4> (= VA<23:18>) は無視される。

N.4.3 Interrupt Vector Dispatch Status Register

SPARC64 IXfx では、BUSY/NACK ペアを 16 組実装する。同時に通信できるのは 16
箇所までとなる。

bit<63:32> は読み出しに対して 0 が読みだされる。

N.4.4 Incoming Interrupt Vector Data Registers

JPS1 Commonality ではインタラプト受信レジスタを 8 つ定義しているが、
SPARC64 IXfx ではこれを 3 つにする。ASI_INTR_DATA_[3–7]R に対する読み出し
は未定義の ASI に準拠する。

Compatibility Note –  この変更は SPARC JPS1 非互換である。
N.4.5 Interrupt Vector Receive Register

SPARC64 IXfx は 10 ビットの値を SID_H および SID_L フィールドに表示するが、その値は不定である (impl. dep. #247)。

N.6 インタラプト配送先の識別法

SPARC64 IXfx は CPU チップ内に複数のコアがある。よって、CPU チップが受信したインタラプトを適切なコアに配送するための判断基準が必要になる。CPU コアの識別子としては ASI_SYS_CONFIG.ITID と ASI_EIDR があり、ファームウェアによって正しく初期化された後は ASI_EIDR が識別の手段となる。

インタラプトパケットが正しく配送されるためには、すべてのコアの ASI_EIDR が初期化済で、ASI_EIDR<3:0> がユニークである必要がある。そのような状態では、インタラプトパケットが正しく配送されるかどうかは保証されない。
Reset, RED_state, and error_state

この章では、電源投入時およびリセット後の動作について説明する。JPS1 Commonality では Section 7.1, “Processor States, Normal and Special Traps” (page 43) でリセットの説明をしているが、リセット時の動作はハードウェア実装に強く依存するので、SPARC64™ IXfx Extensions ではこの章で説明する。RED_state 遷移時のレジスタ値や命令シーケンス開始位置など、ソフトウェアから観測可能な動作については、Section 7.1 を参照。

この章では以下の項目について説明する。

- リセット種類 on page 247
- RED_state と error_state on page 249
- リセット、RED_state 後のプロセッサ状態 on page 251

この章は JPS1 Commonality の章立てとは一致していない。

O.1 リセット種類

この節ではパワーオンリセット (POR)、ウォッチドッグリセット (WDR)、外部指示リセット (XIR)、ソフトウェア指示リセット (SIR) の 4 種類のリセットについて説明する。

POR と XIR は CPU チップ内の全コアに作用する。言い換えれば、全コアで同じトラブル処理を行う。一方 WDR と SIR はそれを起こしたコアにのみ作用する。他のコアはリセットの影響を受けず実行を継続する。

O.1.1 パワーオンリセット (POR)

SPARC64 IXfx で POR を起こすには、外部機構から JTAG で一連の操作を行う必要がある。

Ver 12, 2 Dec. 2013 F. Appendix O Reset, RED_state, and error_state 247
リセットピンがアサートされているか、パワーレディ信号がアサートされていないとき、プロセッサの動作は停止し JTAG コマンドのみが実行可能な状態になっている。プロセッサは、JTAG コマンドによる変更以外、ソフトウェアから見える状態を変更せず、メモリシステムの状態も変更しない。

POR を受けると、プロセッサは RED_state に遷移し、power_on_reset 例外 (TT = 1) が通知され、RSTVaddr + 2016 から命令実行を開始する。

O.1.2 ウォッチドッグリセット (WDR)
ウォッチドッグリセットは以下のときに生成される。

- TL < MAXTL で 2 回目のウォッチドッグタイムアウトを検出したとき。
- TL = MAXTL で 1 回目のウォッチドッグタイムアウトを検出したとき。
- TL = MAXTL でトラップが発生したとき。

ウォッチドッグタイムアウトが発生すると、watchdog_reset 例外 (TT = 2) が通知され、RSTVaddr + 4016 から命令実行を開始する。それ以外では TT は変更されず、CPU は error_state に遷移する。

O.1.3 外部指示リセット (XIR)
システムから XIR 要求を受けると、プロセッサは RED_state に遷移し、externally_initiated_reset 例外 (TT = 3) が通知され、RSTVaddr + 6016 から命令実行を開始する。

O.1.4 ソフトウェア指示リセット (SIR)
CPU チップ内のどのコアでも、SIR 命令によりソフトウェア指示リセットをかけることができる。

TL < MAXTL (5) で SIR 命令を実行すると、RED_state に遷移し、software_initiated_reset 例外 (TT = 4) が通知され、RSTVaddr + 8016 から命令実行を開始する。TL = 5 で SIR 命令を実行すると、error_state に遷移し、最終的にはウォッチドッグリセットによる例外が通知される。
O.2 RED_state と error_state

JPS1 Commonality の定義に、CPU Fatal Error ステートと suspended ステートが追加されている。

* WDT1 は 1 回目のウォッチドッグタイムアウト。
** WDT2 は 2 回目のウォッチドッグタイムアウト。WDT2 により CPU は error_state に遷移する。通常の設定では、error_state に入ると直ちにウォッチドッグリセットがかかり、CPU は RED_state に遷移するので、error_state にあるのは一時的である。OPSR (Operation Status Register) 設定により、error_state に遷移後ウォッチドッグリセットがかからず、CPU が error_state に留まるようにすることも可能である。
*** CPU_fatal_error_state では、CPU で致命的エラーが検出されたことを示す P_FERR をシステムに通知し、CPU は CPU_fatal_error_state ステートに遷移して停止する。

** FIGURE O-1 プロセッサの状態遷移図 **
O.2.1 RED_state

Section 7.1.1, “RED_state” (page 43) に参照。

プロセッサが POR 以外の要因で RED_state に遷移した場合、ソフトウェアは execute_state に遷移しようとしてはいけない。遷移した場合、プロセッサは予期せぬ状態になる。

プロセッサがリセットや RED_state に遷移する例外を通知されると、RED_state 用のトラップテーブル相対のトラップベクタ (RSTVaddr) から命令を実行する。
SPARC64 IXfx では RSTVaddr は VA = FFFF FFFF F000 0000 16,
PA = 0000 01FF F000 0000 16 である。

プロセッサはまた、PSTATE.RED = 1 に設定することでも RED_state に遷移する。この場合、RED_state 用のトラップベクタトラップへの分岐は起こらない。
RED_state に遷移する際にも、RED_state 中での動作は以下の通りである。

■ もう一つの要因で RED_state に遷移した場合、ハードウェアによりいくつかの機能が無効化され、ASI_DCUCR が更新される。ソフトウェアは必要なレジスタの値を再設定すること。

■ 他の要因で RED_state に遷移した場合、DCUCR のビットは更新されず、ペアリングが正常な状態を保つ。

■ また、RED_state が RED_state である間は、DCUCR.IM の値によらず IMMU は無効化される。

■ RED_state でもキャッシュの同一性はハードウェアによって保たれる。

O.2.2 error_state

プロセッサは TL = MAXTL (5) でトラップが起きるか、2 回目のウォッ치ドッグタイムアウトが起きると、error_state に遷移する。

通常の設定では、error_state に入ると直ちにウォッチュドッグリセットがかかり、CPU は RED_state に遷移する。OPSR (Operation Status Register) 設定により、error_state に遷移後ウォッチドッグリセットがかかります、CPU が error_state に留まるようにすることも可能である。

O.2.3 CPU Fatal Error

プロセッサは致命的なエラーを検出すると、CPU Fatal Error ステートに遷移する。プロセッサは致命的なエラーがおきたことをシステムに通知し、停止する。
リセット、RED_state 後のプロセッサ状態

TABLE O-1, TABLE O-2, TABLE O-3 にリセット、RED_state 時のプロセッサの状態を示す。

Programming Note — SPARC64 IXfXは、error_stateから復帰するためにWDRを発生させることができる。ソフトウェアからは、error_stateへの遷移要因によりWDRが発生するように見えるが、ハードウェアの内部では2回の遷移が起きている。TABLE O-1, TABLE O-2, TABLE O-3 各表のWDRの列が表しているのは、WDRの前後でレジスタの状態であって、error_stateに遷移する際のレジスタの状態の変化は含まれていないことに注意。

TABLE O-1 は、トラップまたはリセットによりRED_stateに遷移する場合の特権・非特権レジスタの値である。WRPR命令でPSTATE.REDビットに1をセットした場合、PSTATE.REDビット以外の特権・非特権レジスタは変化しない。

<table>
<thead>
<tr>
<th>名前</th>
<th>POR</th>
<th>WDR</th>
<th>XIR</th>
<th>SIR</th>
<th>RED_state</th>
</tr>
</thead>
<tbody>
<tr>
<td>整数レジスタ</td>
<td>不定/無変更</td>
<td>無変更</td>
<td>無変更</td>
<td></td>
<td></td>
</tr>
<tr>
<td>浮動小数点レジスタ</td>
<td>不定/無変更</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RSTV値</td>
<td>VA = FFFF FFFF F000 000016</td>
<td>PA = 01FF F000 000016</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NPC</td>
<td>RSTV</td>
<td>2016</td>
<td>RSTV</td>
<td>4016</td>
<td>RSTV</td>
</tr>
<tr>
<td>nPC</td>
<td>RSTV</td>
<td>2416</td>
<td>RSTV</td>
<td>4416</td>
<td>RSTV</td>
</tr>
<tr>
<td>PSTATE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AG</td>
<td>1 (AGが使われる)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MG</td>
<td>0 (MGは使われない)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IG</td>
<td>0 (IGは使われない)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IE</td>
<td>0 (割り込み禁止)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PRIV</td>
<td>1 (特権モード)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AM</td>
<td>0 (64ビットアドレスモード)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PEF</td>
<td>1 (FPU使用可能)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RED</td>
<td>1 (Red_state)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MM</td>
<td>002 (TSO)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TLE</td>
<td>0</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLE</td>
<td>0</td>
<td>TLEがコピーされる</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TBA&lt;63:15&gt;</td>
<td>不定/無変更</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Y</td>
<td>不定/無変更</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PIL</td>
<td>不定/無変更</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### TABLE O-1 RED_state およびリセット後の特権・非特権レジスタの値 (2 of 2)

<table>
<thead>
<tr>
<th>名前</th>
<th>POR¹</th>
<th>WDR²</th>
<th>XIR</th>
<th>SIR</th>
<th>RED_state</th>
</tr>
</thead>
<tbody>
<tr>
<td>CWP</td>
<td>不定 / 無変更</td>
<td>無変更 (レジスタウィンドウトラップ以外)</td>
<td>無変更</td>
<td>無変更</td>
<td>無変更 (レジスタウィンドウトラップ以外)</td>
</tr>
<tr>
<td>TT[TL]</td>
<td>1</td>
<td>トラブルタイプか2</td>
<td>3</td>
<td>4</td>
<td>トラブルタイプ</td>
</tr>
<tr>
<td>CCR</td>
<td>不定 / 無変更</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ASI</td>
<td>不定 / 無変更</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TL</td>
<td>NAXTL</td>
<td>min (TL + 1, MAXTL)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TPC[TL]</td>
<td>不定 / 無変更</td>
<td>PC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TNPC[TL]</td>
<td>不定 / 無変更</td>
<td>nPC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TSTATE</td>
<td>不定 / 無変更</td>
<td>CCR</td>
<td>ASI</td>
<td>PSTATE</td>
<td>CWP</td>
</tr>
<tr>
<td>TICK</td>
<td>Counter</td>
<td>1</td>
<td>0 からカウント開始</td>
<td>無変更</td>
<td>カウント継続</td>
</tr>
<tr>
<td>CANSAVE</td>
<td>不定 / 無変更</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CANRESTORE</td>
<td>不定 / 無変更</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLEARWIN</td>
<td>不定 / 無変更</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WSTATE</td>
<td>OTHER</td>
<td>NORMAL</td>
<td>不定 / 無変更</td>
<td>無変更</td>
<td></td>
</tr>
<tr>
<td>VER</td>
<td>MANUP</td>
<td>0004₁₆</td>
<td>IMPL</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MASK</td>
<td>マスク値 (定数) で固定</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>MAXTL</td>
<td>₅₁₆</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>MAXWIN</td>
<td>₇₁₆</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FSR</td>
<td>0</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FFRS</td>
<td>不定 / 無変更</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. ハードパワーオンリセットは電源投入時、ソフトパワーオンリセットはリセット信号がアサートされた時。
2. 同日のウォッチドッグタイムアウトリセットが execute_state (PSTATE.RED = 0) で起きると、次のウォッチドッグタイムアウトリセットと TL=MAXTL でのウォッチドッグトラップで RED_state に入る。詳細は Appendix O.1.2 参照。

---

SPARC64™ IXfx Extensions Ver 12, 2 Dec. 2013
TABLE O-2 は、トラップまたはリセットにより RED_state に遷移した際の ASR レジスタの値である。WRPR 命令で PSTATE.RED ビットに 1 をセットした場合、ASR レジスタの値は変化しない。

TABLE O-2  RED_state およびリセット後の ASR レジスタの値

<table>
<thead>
<tr>
<th>ASR</th>
<th>名前</th>
<th>POR1</th>
<th>WDR2</th>
<th>XIR</th>
<th>SIR</th>
<th>RED_state</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>PCR</td>
<td>0</td>
<td>0</td>
<td>無変更</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>UT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ST</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Others</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>PIC</td>
<td>不定 / 無変更</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>DCR</td>
<td>常に 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>GSR</td>
<td>0</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>IM</td>
<td>0</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>IRND</td>
<td>0</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>その他</td>
<td>不定 / 無変更</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>SOFTINT</td>
<td>不定 / 無変更</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>TICK_COMPARE</td>
<td>1</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>INT_DIS</td>
<td>0</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TICK_CMPR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>STICK</td>
<td>1</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NPT</td>
<td>0 からカウント開始</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Counter</td>
<td>0 からカウント開始</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>カウント継続</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>STICK_COMPARE</td>
<td>1</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>INT_DIS</td>
<td>0</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TICK_CMPR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>XAR</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>XASR</td>
<td>不定 / 無変更</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>TXAR [TL]</td>
<td>不定 / 無変更</td>
<td>XAR</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. ハードパワーオンリセットは電源投入時。ソフトパワーオンリセットはリセット信号がアサートされた時。2.1 回目のウォッチドッグタイムアウトリセットが execute_state (PSTATE.RED = 0) で起きると、次のウォッチドッグタイムアウトリセットと TL = MAXTL でのウォッチドッグトラップで RED_state に入れる。詳細は Appendix O.1.2 参照。
TABLE O-3 は、トラップまたはリセットにより RED_state に遷移した際の ASI レジスタの値である。WRPR 命令で PSTATE.RED ビットに 1 をセットした場合、ASI レジスタの値は変化しない。

<table>
<thead>
<tr>
<th>ASI</th>
<th>VA</th>
<th>名前</th>
<th>POR1</th>
<th>WDR2</th>
<th>XIR</th>
<th>SIR</th>
<th>RED_state</th>
</tr>
</thead>
<tbody>
<tr>
<td>4516</td>
<td>0016</td>
<td>DCUCR</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4516</td>
<td>0816</td>
<td>MCNTL, RMD, その他</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4816</td>
<td>0016</td>
<td>INTR_DISPATCH_STATUS</td>
<td>0</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4916</td>
<td>0016</td>
<td>INTE_RECEIVE</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4A16</td>
<td>—</td>
<td>SYS_CONFIG, ITID</td>
<td>システム規定値 / 無変更</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4B16</td>
<td>0016</td>
<td>STICK_CNTL</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4C16</td>
<td>0016</td>
<td>AFSR</td>
<td>不定 / 無変更</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4C16</td>
<td>0816</td>
<td>UGESR</td>
<td>不定 / 無変更</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4C16</td>
<td>1016</td>
<td>ERROR_CONTROL, WEAK_ED, その他</td>
<td>1</td>
<td>不定 / 無変更</td>
<td>1</td>
<td>無変更</td>
<td></td>
</tr>
<tr>
<td>4C16</td>
<td>1816</td>
<td>STCHG_ERR_INFO</td>
<td>不定 / 無変更</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4F16</td>
<td>0016–3816</td>
<td>SCRATCH_REGS</td>
<td>不定 / 無変更</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5016</td>
<td>0016</td>
<td>IMMU_TAG_TARGET</td>
<td>不定 / 無変更</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5016</td>
<td>1816</td>
<td>IMMU_SFSR</td>
<td>不定 / 無変更</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5016</td>
<td>2816</td>
<td>IMMU_TSB_BASE</td>
<td>不定 / 無変更</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5016</td>
<td>3016</td>
<td>IMMU_TAG_ACCESS</td>
<td>不定 / 無変更</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5016</td>
<td>6016</td>
<td>IMMU_TAG_ACCESS_EXT</td>
<td>不定 / 無変更</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5016</td>
<td>7816</td>
<td>IMMU_SFPAR</td>
<td>不定 / 無変更</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5316</td>
<td>—</td>
<td>SERIAL_ID</td>
<td>定数値</td>
<td>定数値</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5416</td>
<td>—</td>
<td>ITLB_DATA_IN</td>
<td>不定 / 無変更</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5516</td>
<td>—</td>
<td>ITLB_DATA_ACCESS</td>
<td>不定 / 無変更</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5616</td>
<td>—</td>
<td>ITLB_TAG_READ</td>
<td>不定 / 無変更</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5716</td>
<td>—</td>
<td>ITLB_DEMAP</td>
<td>不定 / 無変更</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5816</td>
<td>0016</td>
<td>DMNU_TAG_TARGET</td>
<td>不定 / 無変更</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5816</td>
<td>0816</td>
<td>PRIMARY_CONTEXT</td>
<td>不定 / 無変更</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5816</td>
<td>1016</td>
<td>SECONDARY_CONTEXT</td>
<td>不定 / 無変更</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5816</td>
<td>1816</td>
<td>DMNU_SFSR</td>
<td>不定 / 無変更</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5816</td>
<td>2016</td>
<td>DMNU_SFAR</td>
<td>不定 / 無変更</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5816</td>
<td>2816</td>
<td>DMNU_TSB_BASE</td>
<td>不定 / 無変更</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5816</td>
<td>3016</td>
<td>DMNU_TAG_ACCESS</td>
<td>不定 / 無変更</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
O.3.1 Operating Status Register (OPSR)

OPSR は CPU チップの制御レジスタである。OPSR の値は CPU 外部からセットされ、ソフトウェアから変更することはできない。ハードウェアパワーオンリセット時 CPU が動き出す前にスキャンインされ、その後は JTAG コマンドで変更できる。

<table>
<thead>
<tr>
<th>ASI</th>
<th>VA</th>
<th>名前</th>
<th>POR</th>
<th>WDR</th>
<th>XIR</th>
<th>SIR</th>
<th>RED_state</th>
</tr>
</thead>
<tbody>
<tr>
<td>58_16</td>
<td>38_16</td>
<td>DMMU_WATCHPOINT</td>
<td>不定/無変更</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>58_16</td>
<td>60_16</td>
<td>DMMU_TAG_ACCESS_EXT</td>
<td>不定/無変更</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>58_16</td>
<td>68_16</td>
<td>SHARED_CONTEXT</td>
<td>不定/無変更</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>58_16</td>
<td>78_16</td>
<td>DMMU_SPPAR</td>
<td>不定/無変更</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5C_16</td>
<td></td>
<td>DTLB_DATA_IN</td>
<td>不定/無変更</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5D_16</td>
<td></td>
<td>DTLB_DATA_ACCESS</td>
<td>不定/無変更</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5E_16</td>
<td></td>
<td>DTLB_TAG_READ</td>
<td>不定/無変更</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5E_16</td>
<td></td>
<td>DMMU_DEMAP</td>
<td>不定/無変更</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>60_16</td>
<td></td>
<td>IIU_INST_TRAP</td>
<td>0</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6D_16</td>
<td>00_16-58_16</td>
<td>BARRIER_INIT</td>
<td>0</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6F_16</td>
<td>00_16</td>
<td>EIDR</td>
<td>0/無変更</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6F_16</td>
<td>08_16</td>
<td>BST_BIT</td>
<td>規定値/無変更</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6F_16</td>
<td>00_16-58_16</td>
<td>BARRIER_ASSIGN</td>
<td>0</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>77_16</td>
<td>40_16-50_16</td>
<td>INTR_DATA0:2_W</td>
<td>不定/無変更</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>77_16</td>
<td>70_16</td>
<td>INTR_DISPATCH_W</td>
<td>不定/無変更</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7F_16</td>
<td>40_16-50_16</td>
<td>INTR_DATA0:2_R</td>
<td>不定/無変更</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E7_16</td>
<td>00_16</td>
<td>SCCR</td>
<td>規定値</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E7_16</td>
<td>00_16</td>
<td>NPT</td>
<td>1</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E7_16</td>
<td>00_16</td>
<td>その他</td>
<td>0</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EF_16</td>
<td>00_16-58_16</td>
<td>LBSY, BST</td>
<td>0</td>
<td>無変更</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. ハードパワーオンリセットは電源投入時、ソフトパワーオンリセットはリセット信号がアサートされた時。
2. 回目のウォッチドッグタイムアウトリセットが execute_state \((PSTATE.RED = 0)\) で起きると、次のウォッチドッグタイムアウトリセットと TL = MAXTL でのウォッチドッグトラップで RED_state に入る。詳細は Appendix O.1.2 参照。

TABLE O-3 RED_state およびリセット後の ASI レジスタの値 (2 of 2)
 Error Handling

この章では SPARC64 IXfx のエラー時における SPARC64 IXfx の振舞いと、エラー復旧のために OS やファームウェアが行うべきことを説明する。

P.1 エラーの分類

SPARC64 IXfx ではエラーは以下の 4 つに分類される。

- 致命的エラー (Fatal Error)
- error_state 遷移エラー (Error State Transition error)
- 緊急エラー (Urgent Error)
- 抑止可能エラー (Restrainable Error)

SPARC64 IXfx は 16 コア/チッププロセッサである。どのコアでどのようなエラーが起きたかを識別する方法は、上記のエラー種類によって異なる。

命令実行に起因するエラー、あるいはスレッド固有のリソースに起因するエラーは、命令実行に同期したエラーである。この種のエラーは、エラーを起こしたスレッドだけに報告される。instruction_access_error や data_access_error は、この種のエラーを通知する例外である。

命令実行に起因しないエラー、あるいはチップ内で共有されるリソースに起因するエラーは、命令実行に非同期のエラーである。この種のエラーは、関連するすべてのスレッドに報告される。

エラーマーキングは基本的に命令実行とは非同期に行われる。L1 キャッシュや L2 キャッシュでマークされていないエラー (Raw UE) が発見された場合、生きている (degraded でない) コア内で最小の EIDR でマークされる。

サスペンド状態のスレッドで起きたエラーをどのようにログし報告するかも重要な問題である。致命的エラーを除き、スレッドがサスペンドから抜けつまでエラー報告は延期される。
P.1.1 致命的エラー
致命的エラーはシステム全体に影響を及ぼすエラーである。

a. システム内のデータ一貫性が保証できなくなるようなエラー
キャッシュコヒーレントを壊すようなエラーがこれにあたる。

b. CPU チップ内が制御不能になるようなエラー
致命的エラーを発見すると、CPU は CPU Fatal Error ステートに遷移し、システムに致命的エラーが起きたことを通知した後に停止する。システムは CPU からの致命的エラー通知を受けた後、そのシステムで規定された動作を行う。

すべての致命的エラーは命令実行に非同期である。あるスレッドが致命的エラーを発見すると、CPU チップ内のすべてのスレッドにリセット (Power On Reset: POR) が通知される。これはサスペンド中のスレッドがいるかどうかによらない。

P.1.2 error_state 遷移エラー
error_state 遷移エラー (EE) とは、トラップを上げることができないほどの深刻なエラーである。しかし、そのエラーの影響範囲は CPU 内に限定される。

CPU が error_state 遷移エラーを検出すると、error_state に遷移する。
error_state からは、ウォッチドッグリセットで RED_state に遷移し、ウォッチドッグリセットトラップハンドラから命令実行を再開する。

命令実行に非同期な error_state 遷移エラー
命令実行に非同期の error_state 遷移エラーには以下のものがある。この種の EE がスレッドで起きると、コア内の全スレッドの ASI_STCHG_ERROR_INFO にエラー情報が記録され、ウォッチドッグリセット例外が通知される（ただしサスペンドしていいないとき）。他のコアのスレッドは影響を受けない。

- EE_TRAP_ADR_UOE
- EE_OTHER

命令実行に同期して起きる error_state 遷移エラー
命令実行に同期して起きる error_state 遷移エラーには以下のものがある。この種の EE がスレッドで起きると、当該スレッドの ASI_STCHG_ERROR_INFO にエラー情報が記録され、ウォッチドッグリセット例外が通知される。他のスレッドは影響を受けない。

- EE_SIR_IN_MAXTL
- EE_TRAP_IN_MAXTL
- EE_WDT_IN_MAXTL
EESECOND_WDT

Note – SPARC64 IXfx はマルチスレッド CPU ではないので、命令実行に同期して起きる error_state 遷移エラーでも非同期に起きる error_state 遷移エラーでも、エラー情報はそのコアの ASI_STCHG_ERROR_INFO だけに記録される。

P.1.3 緊急エラー

緊急エラー (UGE) とは、直ちにシステムソフトウェアが介入する必要のあるエラーである。以下の種類がある。

命令実行を阻害するエラー

- I_UGE: 命令の緊急エラー
- IAE: 命令アクセスエラー
- DAE: データアクセスエラー

命令実行とは無関係のエラー

- A_UGE: 自律性緊急エラー

命令実行を阻害するエラー

命令実行阻害エラーとは、命令実行中に検出される、命令実行を不可能にするようなエラーである。

命令実行阻害エラーを検出したときに ASI_ERROR_CONTROL.WEAK_ED = 0 だと（これは通常の実行状態でシステムソフトウェアによって設定される）、例外が通知される。このエラーはマスクすることができない。ASI_ERROR_CONTROL.WEAK_ED = 1 のときは（多重エラー時や POST/OBP によるリセット処理中）、以下のどれかが起きる。

- 可能ならば、実行を阻害された命令のデスティネーションレジスタに、不定値を書き込み、命令を完了させる。
- それができなければ、例外を通知する。エラーを起こした命令は、ASI_ERROR_CONTROL.WEAK_ED = 0 のときと同様に実行される。

この種のエラーは 3 種類ある。

- I_UGE (Instruction Urgent Error: 命令緊急エラー) — IAE (命令アクセスエラー)，DAE (データアクセスエラー) 以外のエラー。I_UGE はさらに 2 種類に分けられる。

- プログラムから見えるレジスタに復旧不可能なエラーが起き、命令実行ができなくなった

PSTATE, PC, NPC, CCR, ASI, FSR, GSR レジスタで復旧不可能なエラーが起きたときがこれにあたる。一回目のウォッチドッグタイムアウトも I_UGE として扱われる。
命令実行部でエラーが起きたとき

演算部や一時レジスタや内部のバスエラーがこれにあたる。

I_UGE は Appendix P.2.2 で言うところのプリエンプティブエラーに相当する。

IAE (Instruction Access Error: 命令アクセスエラー) — JPS1 Commonality で定義されている instruction_access_error のこと。SPARC64 IXfx では、キャッシュやメモリ上の UE を命令フェッチ中に検出すると、IAE が通知される。

IAE は precise な例外である。

DAE (Data Access Error: データアクセスエラー) — JPS1 Commonality で定義されている data_access_error のこと。SPARC64 IXfx では、キャッシュやメモリ上の UE をデータアクセス中に検出すると、DAE が通知される。

DAE は precise な例外である。

命令実行とは無関係に発生するエラー

A_UGE (Autonomous Urgent Error: 自律性緊急エラー) — 命令実行とは無関係に発生するエラーで、直ちに処理する必要があるものである。

通常の命令実行では、ASI_ERROR_CONTROL.WEAK_ED は 0 にセットされている。
この場合、緊急エラー処理中 (つまり async_data_error トリップハンドラ内) では、A_UGE による例外は通知されない。

そうでない場合、たとえばマルチエラーが起きた場合や POST/OBP のリセットルーチンでは、ソフトウェアによって ASI_ERROR_CONTROL.WEAK_ED に 1 がセットされる。この場合、A_UGE による例外は通知されない。

A_UGE には 2 種類ある。

■ 重要なリソースでエラーが起きていて、そのリソースを使うと致命的エラーや error_state 遷移エラーを起こすような場合。

■ 重要なリソースでエラーが起きていて、OS パニックさせたい場合。

エラーが起きているリソースを使うとこれ以上実行が続けられないので、OS パニックさせたいとき。

自律性緊急エラーは disrupting 例外で通知されるが、以下の点で V9 仕様と異なる。

■ PSTATE.IE = 0 でもマスクされず例外が通知される。

■ TPC が指定アドレスにある命令は終了させることができないかもしれません。終了方法はロックストレートレジスタに表示される。

緊急エラーによる例外通知

緊急エラーが起こり、そのエラーがマスクされていないとき、以下のいずれかの例外でシステムソフトウェアに通知される。

■ I_UGE, A_UGE:  async_data_error 例外

■ IAE:  instruction_access_error 例外
命令実行とは無関係に起きる緊急エラー

以下のエラーがこれに当たる。あるスレッドでこのエラーが起きると、コア内の全スレッドの ASI_UGESR にエラーが記録され、async_data_error 例外が通知される。ただしサスペンス状態を除く。他コアは影響を受けない。

- IAUG_CRE
- IAUG_TSBCTX
- IUG_TSBP
- IUG_PSTATE
- IUG_TSTATE
- IUG_%F (ただし f[n] のパリティエラー以外)
- IUR_%R (ただし r[n] と Y のパリティエラー以外)
- IUG_WDT
- IUG_DTLB
- IUG_ITLB
- IUG_COREERR

命令実行に同期して起きる緊急エラー

以下のエラーがこれに当たる。あるスレッドでこのエラーが起きると、そのスレッドの ASI_UGESR だけにエラーが記録され、ADE 例外が通知される。ただしサスペンス状態を除く。他スレッドは影響を受けない。

- IUG_%F (f[n] のパリティエラーのみ)
- IUR_%R (r[n] と Y のパリティエラーのみ)

**Note** – SPARC64 IAXf はマルチスレッド CPU ではないので、命令実行に同期して起きる緊急エラーでも非同期に起きる緊急エラーでも、エラー情報はそのコアの ASI_UGESR だけに記録される。

P.1.4 抑止可能エラー

抑止可能エラー (Restrainable Error) とは、実行中のプログラムに深刻な影響を与えないため、システムソフトウェアが直ちに処理する必要のないエラーである。優先度の低い disrupting 例外で通知される。

抑止可能エラーには 2 種類ある。

- 訂正不可能だが現在の命令列の実行に影響を与えないエラー
  - キャッシュのライトバックやコピーバック時に検出されたエラーがこれにあたる。
- 縮退 (Degradation)
頻繁にエラーを起こしているが、命令実行に深刻な影響を与えないリソースを隔離して使わないようにすることができる。しかしながら、性能は多少犠牲になる。

**Compatibility Note** – SPARC64 IXfx は訂正可能エラー（Correctable Error: CE）を検出した場合、自動で訂正し、ソフトウェアには通知しない。

抑止可能エラーは *ECC_error* で通知される。ただしこれは、PSTATE.IE = 1 で、抑止可能エラーの通知が許可されているときのみである。

**DG_U2$, UE_RAW_L2$INSN**

これらは命令実行に非同期なエラーである。これらのエラーが発見されると、CPU モジュール内の全スレッドの AFSR にエラーが記録され、*ECC_error* が通知される。ただしサスペンド中のスレッドには例外は通知されない。

**DG_D1$sTLB, UE_RAW_D1$INSN**

これらは命令実行に非同期なエラーである。これらのエラーが発見されると、コア内の全スレッドの AFSR にエラーが記録され、*ECC_error* が通知される。ただしサスペンド中のスレッドには例外は通知されない。

他のコアには影響を与えない。

**UE_DST_BETO**

これは命令実行に同期したエラーである。エラーが発見されると、エラーを起こしたスレッドの ASI_ISFSR, TPC, ASI_ISFPAR にエラーが記録され、*instruction_access_error* が通知される。他のスレッドには影響を与えない。

**P.1.5 instruction_access_error**

これは命令実行に同期したエラーである。エラーを起こしたスレッドの ASI_ISFSR, TPC, ASI_ISFPAR にエラーが記録され、*instruction_access_error* が通知される。他のスレッドには影響を与えない。

**P.1.6 data_access_error**

これは命令実行に同期したエラーである。エラーを起こしたスレッドの ASI_DSFSR, ASI_DSFAR, ASI_DSFPAR にエラーが記録され、*data_access_error* が通知される。他のスレッドには影響を与えない。
P.2 エラー処理とエラー制御

P.2.1 エラー処理に必要なレジスタ

TABLE P-1 はエラー処理に必要なレジスタの一覧である。これらのレジスタのうち、ASI_ERROR_CONTROL は、エラーを検出した際に例外として通知するかどうかを制御するレジスタで、ASI_EIDR はエラーマーキング用の識別 ID である。その他のレジスタにはエラーの詳細情報が表示される。

TABLE P-1 エラー処理に必要なレジスタ

<table>
<thead>
<tr>
<th>ASI</th>
<th>VA</th>
<th>名前</th>
<th>説明されている章</th>
</tr>
</thead>
<tbody>
<tr>
<td>4C16</td>
<td>0016</td>
<td>ASI_ASYNC_FAULT_STATUS</td>
<td>P.7.1</td>
</tr>
<tr>
<td>4C16</td>
<td>0816</td>
<td>ASI_URGENT_ERROR_STATUS</td>
<td>P.4.1</td>
</tr>
<tr>
<td>4C16</td>
<td>1016</td>
<td>ASI_ERROR_CONTROL</td>
<td>P.2.6</td>
</tr>
<tr>
<td>4C16</td>
<td>1816</td>
<td>ASI_STCHG_ERROR_INFO</td>
<td>P.3.1</td>
</tr>
<tr>
<td>5016</td>
<td>1816</td>
<td>ASI_IMMU_SFSR</td>
<td>F.10.9</td>
</tr>
<tr>
<td>5016</td>
<td>7816</td>
<td>ASI_IMMU_SFPAR</td>
<td>F.10.12</td>
</tr>
<tr>
<td>5816</td>
<td>1816</td>
<td>ASI_DMMU_SFSR</td>
<td>F.10.9</td>
</tr>
<tr>
<td>5816</td>
<td>2016</td>
<td>ASI_DMMU_SPAR</td>
<td>F.10.10 of JPS1 Commonality</td>
</tr>
<tr>
<td>5816</td>
<td>7816</td>
<td>ASI_DMMU_SFPAR</td>
<td>F.10.12</td>
</tr>
<tr>
<td>6E16</td>
<td>0016</td>
<td>ASI_EIDR</td>
<td>P.2.5</td>
</tr>
</tbody>
</table>

P.2.2 エラー検出時の動作

ここではエラー検出時の挙動を説明する。
エラー検出を抑止する条件

<table>
<thead>
<tr>
<th>エラー種類</th>
<th>抑止条件</th>
</tr>
</thead>
<tbody>
<tr>
<td>致命的エラー</td>
<td>なし (すべて検出する)</td>
</tr>
<tr>
<td>error_state 遷移エラー</td>
<td>ASI_ECR.WEAK_ED = 1 で大部分のエラーを検出しないが、一部は検出する。</td>
</tr>
<tr>
<td>壊急エラー</td>
<td>なし</td>
</tr>
<tr>
<td>I_UGE, IAE, DAE:</td>
<td>なし</td>
</tr>
<tr>
<td>A_UGE:</td>
<td>なし</td>
</tr>
</tbody>
</table>

レジスタ使用以外のエラーは、ASI_ECR.WEAK_ED = 1 または個々のエラー独自のとき抑制される。レジスタ使用のエラーは、個々のエラー独自の条件のときに抑制される。個々のエラー独自の条件があるのはごく一部。
エラー検出時、例外の通知を抑止する条件

<table>
<thead>
<tr>
<th>エラー種類</th>
<th>通知抑止条件</th>
</tr>
</thead>
<tbody>
<tr>
<td>致命的エラー</td>
<td>なし（すべて検出する）。</td>
</tr>
<tr>
<td>error_state遅延エラー</td>
<td>なし（すべて検出する）。</td>
</tr>
<tr>
<td>緊急エラー</td>
<td>1. UGE, IAE, DAE: 合計</td>
</tr>
<tr>
<td></td>
<td>- SUSPENDED ステートのとき。</td>
</tr>
<tr>
<td></td>
<td>A UGE:</td>
</tr>
<tr>
<td></td>
<td>- ASI_ECR.UGE_HANLDER = 1 のとき。</td>
</tr>
<tr>
<td></td>
<td>- ASI_ECR.WEAK_ED = 1 のとき。</td>
</tr>
<tr>
<td></td>
<td>- トラブルマスク中に例外が検出されると、通知は延期される。</td>
</tr>
<tr>
<td></td>
<td>- マスクが開くと async_data_error が通知される。</td>
</tr>
<tr>
<td></td>
<td>- SUSPENDED ステートのとき。</td>
</tr>
<tr>
<td>抑止可能エラー</td>
<td>合計</td>
</tr>
<tr>
<td></td>
<td>- ASI_ECR.UGE_HANLDER = 1 のとき。</td>
</tr>
<tr>
<td></td>
<td>- ASI_ECR.WEAK_ED = 1 のとき。</td>
</tr>
<tr>
<td></td>
<td>- PSSTATE_IE = 0 のとき。</td>
</tr>
<tr>
<td></td>
<td>- エラーをマスクする設定になっているとき。</td>
</tr>
<tr>
<td></td>
<td>- マスクはエラー種類に応じて ASI_ECR.RTE_DG と</td>
</tr>
<tr>
<td></td>
<td>- ASI_ECR.RTE_UE がある。</td>
</tr>
<tr>
<td></td>
<td>- SUSPENDED ステートのとき。</td>
</tr>
</tbody>
</table>

エラー検出時の動作

<table>
<thead>
<tr>
<th>エラー種類</th>
<th>動作</th>
</tr>
</thead>
<tbody>
<tr>
<td>致命的エラー</td>
<td>1. CPU はフェイタルステートに遅延する。</td>
</tr>
<tr>
<td></td>
<td>2. CPU はシステムにフェイタルエラー検出を通知する。</td>
</tr>
<tr>
<td></td>
<td>3. システムが停止する。</td>
</tr>
<tr>
<td>error_state遅延エラー</td>
<td>1. CPU は error_state に遅延する。</td>
</tr>
<tr>
<td></td>
<td>2. WDR が CPU に通知される。</td>
</tr>
</tbody>
</table>
### エラー種類

<table>
<thead>
<tr>
<th></th>
<th>動作</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>緊急エラー</strong></td>
<td></td>
</tr>
</tbody>
</table>
| **I_UGE:** | • ASI_ECR.UGE_HANLDER = 0 のとき、単独 ADE 例外が通知される。  
  • ASI_ECR.UGE_HANLDER = 1 のとき、多重 ADE 例外が通知される。
| **A_UGE:** | • 例外通知がマスクされていないときは、単独 ADE 例外が通知される。  
  • 例外通知がマスクされているときは、例外通知はペンディングされる。
| **IAE:** | • ASI_ECR.UGE_HANLDER = 0 のとき、IAE 例外が通知される。  
  • ASI_ECR.UGE_HANLDER = 1 のとき、多重 ADE 例外が通知される。
| **DAE:** | • ASI_ECR.UGE_HANLDER = 0 のとき、DAE 例外が通知される。  
  • ASI_ECR.UGE_HANLDER = 1 のとき、多重 ADE 例外が通知される。
| **抑止可能エラー** | 例外通知がマスクされておらず、ASI_AFSR にもエラー情報が表示されていないにも係らず、ECC_error が通知されることがある。
| 1. | エラー通知がペンディングされている状態で、ASI_AFSR への書き込みを行い、エラー情報が消されたとき。  
  2. UE を検出して ECC_error が通知されたとき、ペンディングしている DG の情報が、ASI_AFSR への書き込みにより消されたとき。
| 3. DG を検出して ECC_error が通知されたとき、ペンディングしている UE の情報が、ASI_AFSR への書き込みにより消されたとき。  
  このような例外が通知された場合、システムソフトウェアは例外を無視して処理を続行すべきである。
TPC とエラーを起こした命令の関係

<table>
<thead>
<tr>
<th>エラー種類</th>
<th>動作</th>
</tr>
</thead>
<tbody>
<tr>
<td>致命的エラー</td>
<td>無関係。</td>
</tr>
<tr>
<td>error_state 遷移エラー</td>
<td>無関係。</td>
</tr>
<tr>
<td>緊急エラー</td>
<td></td>
</tr>
</tbody>
</table>

I_UGE:

- TLB 書き込みエラーでは、TLB を更新しようとした命令は TPC が指す命令か、命令フロー上それより前の命令のこともある。TLB 書き込みエラーは、書き込み後 DONE/RETRY が実行されるか例外が通知された時点検出される。
- TLB 書き込み以外のエラーでは、TPC が指す命令か、命令フロー上それより後の命令でエラーが起きている。

A_UGE:

- 無関係。
- IAE, DAE
  - TPC が指す命令がエラーを起こした命令である。

抑止可能エラー 無関係。

その他

複数種類のエラーが同時に検出された場合の優先順位

<table>
<thead>
<tr>
<th>致命的エラー</th>
<th>error_state 遷移エラー</th>
<th>緊急エラー</th>
<th>抑止可能エラー</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. フェイタルステートに遷移 (TT = 1)</td>
<td>2. error_state 遷移 (TT = 2)</td>
<td>3. ADE (TT = 4016)</td>
<td>4. DAE (TT = 3216)</td>
</tr>
<tr>
<td>5. IAE (TT = 0A16)</td>
<td>6. ECC_error_trap (TT = 6316)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

割り込まれた命令の完了方法

<table>
<thead>
<tr>
<th>致命的エラー</th>
<th>error_state 遷移エラー</th>
<th>緊急エラー</th>
<th>抑止可能エラー</th>
</tr>
</thead>
<tbody>
<tr>
<td>完了できない</td>
<td>完了できない</td>
<td>ADE:</td>
<td>JPS1 の Precise 定義通り。</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• P4.3 参照。</td>
<td></td>
</tr>
<tr>
<td>IAE, DAE:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• JPS1 の Precise 定義通り。</td>
<td></td>
</tr>
</tbody>
</table>

エラー表示レジスタ

<table>
<thead>
<tr>
<th>致命的エラー</th>
<th>error_state 遷移エラー</th>
<th>緊急エラー</th>
<th>抑止可能エラー</th>
</tr>
</thead>
</table>
発見時に元データを自動で訂正できる限界

CPUは入ってきたデータを訂正し、演算を続ける。しかし、元のデータを自動で訂正するのは限界がある。以下の箇所のデータは自動で訂正できない。

| ■ メモリの CE |
| ■ 外部からの非同期割込みに付属するデータ (INTR_DATA_R) |

その他の CEエラーについては元のデータを自動訂正することができる。

INTR_DATA_RのCEについては、次回の外部割込みによりエラーデータは上書きされるので、OSによるエラー処理は不要である。メモリのCEはOSによる訂正処理が必要である。

### 一回の例外で通知されるエラー数

<table>
<thead>
<tr>
<th>致命的エラー</th>
<th>error_state 遷移エラー</th>
<th>緊急エラー</th>
<th>抑止可能エラー</th>
</tr>
</thead>
<tbody>
<tr>
<td>すべての致命的エラーが検出される。</td>
<td>すべての error_state 遷移エラーが検出され、ASI_STCHG_ERROR_INFOに表示される。</td>
<td>ほかに発生した場合は、最初のADEのUGEが表示される。</td>
<td>すべての抑制可能エラーが検出され、ASI_AFSRに表示される。</td>
</tr>
<tr>
<td>IAE:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DAE:</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### P.2.3 CE 発見時に元データを自動で訂正できる限界

訂正可能エラー(CE)が発見されると、CPUは入ってきたデータを訂正し、演算を続ける。しかし、元のデータを自動で訂正するのは限界がある。以下の箇所のデータは自動で訂正できない。

<table>
<thead>
<tr>
<th>致命的エラー</th>
<th>error_state 遷移エラー</th>
<th>緊急エラー</th>
<th>抑止可能エラー</th>
</tr>
</thead>
<tbody>
<tr>
<td>すべての致命的エラーが検出される。</td>
<td>すべての error_state 遷移エラーが検出され、ASI_STCHG_ERROR_INFOに表示される。</td>
<td>ほかに発生した場合は、最初のADEのUGEが表示される。</td>
<td>すべての抑制可能エラーが検出され、ASI_AFSRに表示される。</td>
</tr>
<tr>
<td>IAE:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DAE:</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### T2.3 CE 発見時に元データを自動で訂正できる限界

訂正可能エラー(CE)が発見されると、CPUは入ってきたデータを訂正し、演算を続ける。しかし、元のデータを自動で訂正するのは限界がある。以下の箇所のデータは自動で訂正できない。

<table>
<thead>
<tr>
<th>致命的エラー</th>
<th>error_state 遷移エラー</th>
<th>緊急エラー</th>
<th>抑止可能エラー</th>
</tr>
</thead>
<tbody>
<tr>
<td>すべての致命的エラーが検出される。</td>
<td>すべての error_state 遷移エラーが検出され、ASI_STCHG_ERROR_INFOに表示される。</td>
<td>ほかに発生した場合は、最初のADEのUGEが表示される。</td>
<td>すべての抑制可能エラーが検出され、ASI_AFSRに表示される。</td>
</tr>
<tr>
<td>IAE:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DAE:</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
キャッシュプルデータのエラーマーキング

キャッシュプルデータのエラーマーキング

最初にキャッシュプルデータに訂正不能エラー（UE）を見つけたハードウェア内のユニットは、そのデータと ECC を特別な値にする。これにより、エラーが認識されていることと、エラー発生元が特定できる。これをエラーマーキングと言う。エラーマーキングはエラー発生源を特定し、一つのエラーにより何度もエラー報告が上がるように防ぐ。

システム内で ECC で保護されている箇所には以下のものがある。

■ メインメモリ
■ メモリと ICC からのデータパス
■ L2 キャッシュデータ
■ L1D キャッシュデータ

CPU がまだマークされていない UE を見つけると、エラーマーキングを行う。

エラーがマークされたかどうかは 8 バイト毎につけられるシンドロームで識別できる。

TABLE P-2 エラーマークに使われるシンドローム

<table>
<thead>
<tr>
<th>シンドローム</th>
<th>エラーマークの状態</th>
<th>訂正不能エラー（UE）の種類</th>
</tr>
</thead>
<tbody>
<tr>
<td>7F16</td>
<td>マーク済</td>
<td>訂正不能エラー (UE)</td>
</tr>
<tr>
<td>7F16 以外の複数ビットエラーパターン</td>
<td>まだマークされていない</td>
<td>マークされていないUE (Raw UE)</td>
</tr>
</tbody>
</table>

シンドローム 7F16 は、3 ビットエラーが起きていることを表わす。エラーマーキングでは、元のデータと ECC を次節で説明するものに置換える。エラー以外でシンドローム 7F16 が起きる確率はほぼ 0 とみなす。
エラーマーキングデータのフォーマット

キャッシュプールデータで UE が発見されると、エラーデータと ECC はエラーマーキングデータで置換えられる。

**TABLE P-3 エラーマーキングデータのフォーマット**

<table>
<thead>
<tr>
<th>ビット</th>
<th>値</th>
</tr>
</thead>
<tbody>
<tr>
<td>data 63</td>
<td>エラービット。値は不定。</td>
</tr>
<tr>
<td>62:56</td>
<td>0 (7 ビット)</td>
</tr>
<tr>
<td>55:42</td>
<td>ERROR_MARK_ID (14 ビット)</td>
</tr>
<tr>
<td>41:36</td>
<td>0 (6 ビット)</td>
</tr>
<tr>
<td>35</td>
<td>エラービット。値は不定。</td>
</tr>
<tr>
<td>34:23</td>
<td>0 (12 ビット)</td>
</tr>
<tr>
<td>22</td>
<td>エラービット。値は不定。</td>
</tr>
<tr>
<td>21:14</td>
<td>0 (8 ビット)</td>
</tr>
<tr>
<td>13:0</td>
<td>ERROR_MARK_ID (14 ビット)</td>
</tr>
</tbody>
</table>

ECC
ビット 63, 35, 22 に 3 ビットエラーが存在することを示すパターン。シンドロームが 7F16 になるパターンが設定される。

**TABLE P-4 ERROR_MARK_ID の各ビットの説明**

<table>
<thead>
<tr>
<th>ビット</th>
<th>値</th>
</tr>
</thead>
<tbody>
<tr>
<td>13:12</td>
<td>モジュール ID。エラーが起きたハードウェアを示す。</td>
</tr>
<tr>
<td>00:</td>
<td>メモリシステム (DIMM を含む)</td>
</tr>
<tr>
<td>01:</td>
<td>チャネル</td>
</tr>
<tr>
<td>10:</td>
<td>CPU</td>
</tr>
<tr>
<td>11:</td>
<td>Reserved</td>
</tr>
<tr>
<td>11:0</td>
<td>ソース ID。モジュール ID = 00 のとき、ソース ID は常に 0。それ以外では、エラーが発見したハードウェアの ID に入る。</td>
</tr>
</tbody>
</table>
CPU が付加する ERROR_MARK_ID

CPU が付加する ERROR_MARK_ID を TABLE P-5 に示す。

<table>
<thead>
<tr>
<th>マークされていないエラー (Raw UE)</th>
<th>Module_ID の値</th>
<th>Source_ID の値</th>
</tr>
</thead>
<tbody>
<tr>
<td>メモリバスからの入力データ</td>
<td>002 (メモリシステム)</td>
<td>0</td>
</tr>
<tr>
<td>メモリバスへの出力データ</td>
<td>102 (CPU)</td>
<td>00000000 000002</td>
</tr>
<tr>
<td>L2 キャッシュデータ</td>
<td>102 (CPU)</td>
<td>00000000 000002</td>
</tr>
<tr>
<td>L1D キャッシュデータ</td>
<td>102 (CPU)</td>
<td>00000000 000002 ASI_EIDR&lt;3:0&gt;</td>
</tr>
</tbody>
</table>
P.2.5 ASI_EIDR

ASI_EIDR レジスタは ERROR_MARK_ID の Source_ID に記録するための情報を保持する。ASI_EIDR はまた、インタラプトを受信する CPU の識別にも使われる (Appendix N.6、"インタラプト配送先の識別法" (page 245) 参照)。

<table>
<thead>
<tr>
<th>ビット</th>
<th>フィールド名</th>
<th>アクセス</th>
<th>説明</th>
</tr>
</thead>
<tbody>
<tr>
<td>63:4</td>
<td>Reserved</td>
<td>R</td>
<td>常に 0。</td>
</tr>
<tr>
<td>3:0</td>
<td>ERROR_MARK_ID</td>
<td>RW</td>
<td>CPU でエラーが発生したとき、エラーデータの ERROR_MARK_ID にコピーする。</td>
</tr>
</tbody>
</table>

Compatibility Note – SPARC64 VII 以前の仕様では、ソフトウェアが ASI_EIDR<13:12> に 10_2 を設定し、その値が ERROR_MARK_ID に反映されることになっていた。SPARC64 IXfx では Module_ID_Value はハードウェアが固定で持っているため、ソフトウェアによる設定は不要となった。

P.2.6 エラー検出の制御 (ASI_ERROR_CONTROL)

ASI_ERROR_CONTROL は、エラー検出のマスクおよび検出後の動作を設定するレジスタである。

<table>
<thead>
<tr>
<th>レジスタ名</th>
<th>ASI_ERROR_CONTROL (ASI_ECR)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASI</td>
<td>4C_16</td>
</tr>
<tr>
<td>VA</td>
<td>10_16</td>
</tr>
<tr>
<td>エラー検出</td>
<td>なし</td>
</tr>
<tr>
<td>フォーマット</td>
<td>TABLE P-7 参照。</td>
</tr>
</tbody>
</table>

リセット後の初期値

ハード POR 時は、WEAK_ED は 1、それ以外は 0 に設定される。
それ以外のリセットでは、UGE_HANDLER と WEAK_ED の値が ASI_STCHG_ERROR_INFO にコピーされ、全フィールドが 0 に設定される。
ASI_ERROR_CONTROL レジスタはエラーの検出時、例外通知時、多重エラー発生時の処理を制御する。レジスタフィールドは以下の通り。

<table>
<thead>
<tr>
<th>ビット</th>
<th>フィールド名</th>
<th>アクセス</th>
<th>説明</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>RTE_UA</td>
<td>RW</td>
<td>抑止可能エラーのうち UE, Raw UE を例外で通知するかどうかを指定する。Appendix P.2.2 に記述された処理を行う。</td>
</tr>
<tr>
<td>8</td>
<td>RTE_DG</td>
<td>RW</td>
<td>抑止可能エラーのうちデグレードエラーを例外で通知するかどうかを指定する。Appendix P.2.2 に記述された処理を行う。</td>
</tr>
<tr>
<td>1</td>
<td>WEAK_ED</td>
<td>RW</td>
<td>エラー検出を弱める。I_UGE, DAE の検出を抑止するかどうかを決める。</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0 エラー検出は行われる。</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 CPU が実行を続けられるならば、エラー検出は行われない。</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>WEAK_ED = 1 で命令実行中に I_UGE, DAE が検出されるとき、結果出力（レジスタやメモリ）には不定値が書き込まれる。</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>WEAK_ED = 1 であっても、I_UGE, DAE エラーを無視して実行を継続できないときは、エラーが通知される。</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>WEAK_ED は、P.2.2 に記述された、A_UGE と抑止可能エラーにおける例外通知マスクである。</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>多重 ADE が起きたときに、ハードウェアによって WEAK_ED = 1 がセットされる。</td>
</tr>
<tr>
<td>0</td>
<td>UGE_HANDLER</td>
<td>RW</td>
<td>UGE が起きた際に、OS が UGE 处理中かどうかをハードウェアが識別するために使われる。</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0 ハードウェアは、OS が UGE 处理中ではない、と認識する。</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 ハードウェアは、OS が UGE 处理中である、と認識する。</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>UGE_HANDLER は、P.2.2 で説明した、A_UGE と抑止可能エラーにおける例外通知マスクである。</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>I_UGE,IAE,DAE 発生時に多重エラーが起きたかどうかを識別するために使われる。</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ADE が起きたときに、UGE_HANDLER = 1 にセットされる。</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RETRY/DONE で 0 にリセットされる。</td>
</tr>
<tr>
<td>その他</td>
<td>Reserved</td>
<td>R</td>
<td>常に 0。</td>
</tr>
</tbody>
</table>

TABLE P-7  ASI_ERROR_CONTROL のフィールドの説明
P.3 致命的エラーと error_state 遷移エラー

P.3.1 ASI_STCHG_ERROR_INFO

ASI_STCHG_ERROR_INFO には、検出された error_state 遷移エラーの情報が表示される。これらの情報は主に OBP が利用する。

**Compatibility Note –** SPARC64 IXfx では致命的エラーの情報は ASI_STCHG_ERROR_INFO には表示されないので、システムソフトウェアが致命的エラーの詳細情報を知ることはできない。

<table>
<thead>
<tr>
<th>レジスタ名</th>
<th>ASI_STCHG_ERROR_INFO</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASI</td>
<td>4C</td>
</tr>
<tr>
<td>VA</td>
<td>18</td>
</tr>
<tr>
<td>エラー検出</td>
<td>なし</td>
</tr>
<tr>
<td>フォーマット</td>
<td>TABLE P-8 参照</td>
</tr>
<tr>
<td>リセット後の初期値</td>
<td>ハード POR 時は、全てのフィールドが初期化される。それ以外のリセットでは、値は変更されない。</td>
</tr>
<tr>
<td>更新ポリシー</td>
<td>エラーを検出した際、関連するフィールドを 1 に設定する。 ビット 0 に 1 を書きと、全ビットが 0 にリセットされる。</td>
</tr>
</tbody>
</table>
TABLE P-8 に ASI_STCHG_ERROR_INFO のフィールドの説明を示す。表中 sticky とは、いったんハードウェアがそのビットに 1 をセットすると、ソフトウェアが 0 を書くまでその値を保持し続けることを意味する。

TABLE P-8  ASI_STCHG_ERROR_INFO のフィールドの説明（1 of 2）

<table>
<thead>
<tr>
<th>ビット</th>
<th>フィールド名</th>
<th>アクセス</th>
<th>説明</th>
</tr>
</thead>
<tbody>
<tr>
<td>63:34</td>
<td>Reserved</td>
<td>R</td>
<td>常に 0。</td>
</tr>
<tr>
<td>33</td>
<td>ECR_WEAK_ED</td>
<td>R</td>
<td>POR またはウォッチドッグリセット時に ASI_ERROR_CONTROL.WEAK_ED がコピーされる。</td>
</tr>
<tr>
<td>32</td>
<td>ECR_UGE_HANDLER</td>
<td>R</td>
<td>POR またはウォッチドッグリセット時に ASI_ERROR_CONTROL.UGE_HANDLER がコピーされる。</td>
</tr>
<tr>
<td>31:24</td>
<td>Reserved</td>
<td>R</td>
<td>常に 0。</td>
</tr>
<tr>
<td>23</td>
<td>EE_MODULE</td>
<td>RW</td>
<td>error_state 遷移エラーにより、CPU モジュールの縮退が要求されたことを示す。sticky。</td>
</tr>
<tr>
<td>22</td>
<td>EE_CORE</td>
<td>RW</td>
<td>error_state 遷移エラーにより、コアの縮退が要求されたことを示す。sticky。</td>
</tr>
<tr>
<td>21</td>
<td>EE_THREAD</td>
<td>RW</td>
<td>error_state 遷移エラーにより、スレッドの縮退が要求されたことを示す。sticky。ハードウェアがこのビットを 1 にセットすることはない。</td>
</tr>
<tr>
<td>20</td>
<td>UGE_MODULE</td>
<td>RW</td>
<td>緊急エラーにより、CPU モジュールの縮退が要求されたことを示す。sticky。</td>
</tr>
<tr>
<td>19</td>
<td>UGE_CORE</td>
<td>RW</td>
<td>緊急エラーにより、コアの縮退が要求されたことを示す。sticky。</td>
</tr>
<tr>
<td>18</td>
<td>UGE_THREAD</td>
<td>RW</td>
<td>緊急エラーにより、スレッドの縮退が要求されたことを示す。sticky。ハードウェアがこのビットを 1 にセットすることはない。</td>
</tr>
<tr>
<td>17</td>
<td>rawUE_MODULE</td>
<td>RW</td>
<td>L2 キャッシュで rawUE が検出されたことを示す。sticky。</td>
</tr>
<tr>
<td>16</td>
<td>rawUE_CORE</td>
<td>RW</td>
<td>L1 キャッシュで rawUE が検出されたことを示す。sticky。</td>
</tr>
<tr>
<td>15</td>
<td>EE_DCUCR_MCNTL_ECR</td>
<td>R</td>
<td>以下のレジスタで UE が検出されたことを示す。 &lt;br&gt; (A) ASI_DCUCR &lt;br&gt; (A) ASI_MCNTL &lt;br&gt; (A) ASI_ECR</td>
</tr>
<tr>
<td>14</td>
<td>EE_OTHER</td>
<td>R</td>
<td>この表に記載されていない箇所でのエラー発生で 1 がセットされる。 SPARC64 IXfx では常に 0。</td>
</tr>
<tr>
<td>13</td>
<td>EE_TRAP_ADR_ue</td>
<td>R</td>
<td>例外発生時にアドレスを計算しようとした際、 TBA, T? あるいはアドレス計算回路で UE が発生しアドレス計算がことができなかったことを示す。</td>
</tr>
<tr>
<td>12</td>
<td>Reserved</td>
<td>R</td>
<td>常に 0。</td>
</tr>
</tbody>
</table>
サスペンド中のスレッドでの error_state 遷移エラー

SPARC64 IXfx は SUSPEND 命令により suspended 状態に遷移し、POR, WDR, XDR, interrupt_vector, interrupt_level_n により復旧する。例外処理に関連する回路でエラーが発生すると、suspend 状態から復旧できなくなる。このような状態に陥ることを防ぐため、以下のレジスタの緊急エラーは error_state 遷移エラーとしてサスペンド中でも報告される。

- ASI_EIDR
- STICK, STICK_CMPR
- TICK, TICK_CMPR

このような場合、UGESR の対応するビットと STCHG_ERROR_INFO.UGE_CORE に 1 がセットされる。

P.4 緊急エラー

この章では緊急エラーの詳細、モニタ方法や命令終了方法について説明する。
P.4.1 緊急エラーステータス (ASI_UGESR)

<table>
<thead>
<tr>
<th>レジスタ名</th>
<th>ASI_URGENT_ERROR_STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASI</td>
<td>4C₁₆</td>
</tr>
<tr>
<td>VA</td>
<td>08₁₆</td>
</tr>
<tr>
<td>エラー検出</td>
<td>なし</td>
</tr>
<tr>
<td>フォーマット</td>
<td>TABLE P-9 参照</td>
</tr>
<tr>
<td>リセット後の初期値</td>
<td>ハード POR 時は、全てのフィールドがで初期化される。それ以外のリセットでは、値は変更されない。</td>
</tr>
</tbody>
</table>

UGESR は async_data_error 発生時のエラー詳細と多重 async_data_error 時の 2 番目のエラーの詳細を表示する。

TABLE P-9 に UGESR のフィールドの意味を示す。フィールド名にはプリフィックスがついているが、その意味は以下の通りである。

- IUG_ 命令の緊急エラー
- IAG_ 自律発生の緊急エラー
- IAUG_ 1_UGE,A_UGE の両方のエラー
TABLE 9  ASI_UGESR のフィールドの説明 (1 of 3)

<table>
<thead>
<tr>
<th>ビット</th>
<th>フィールド名</th>
<th>アクセス</th>
<th>説明</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;22:8&gt;</td>
<td>それぞれのビットは単独 ADE 例外発生の要因を表わす。1 のとき、例外が発生していることを示す。bit &lt;22:16&gt; は CPU 内レジスタでのエラーを表わす。これらエラーの検出条件は Appendix P.8、“レジスタで起きたエラーの処理方法”を参照。</td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>IUG_CRE</td>
<td>R</td>
<td>（IA）ASI_EIDR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>（IA）ASI_WATCHPOINT (有効な場合)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>（I）ASI_INTR_R</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>（A）ASI_INTR_DISPATCH_W (書き込み時の UE)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>（IA）STICK</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>（IA）STICK_CMPR</td>
</tr>
<tr>
<td>21</td>
<td>IUG_TSBCTX</td>
<td>R</td>
<td>以下レジスタの UE。</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>（IA）ASI_DMMU_TSB_BASE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>（IA）ASI_PRIMARY_CONTEXT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>（IA）ASI_SECONDARY_CONTEXT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>（IA）ASI_SHARED_CONTEXT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>（IA）ASI_IMMU_TSB_BASE</td>
</tr>
<tr>
<td>20</td>
<td>IUG_TSBP</td>
<td>R</td>
<td>以下レジスタの UE。</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>（I）ASI_DMMU_TAG_TARGET</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>（I）ASI_DMMU_TAG_ACCESS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>（I）ASI_IMMU_TAG_TARGET</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>（I）ASI_IMMU_TAG_ACCESS</td>
</tr>
<tr>
<td>19</td>
<td>IUG_PSTATE</td>
<td>R</td>
<td>以下レジスタの UE。</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PSTATE, PC, NPC, CWP, CANSAVE, CANRESTORE, OTHERWIN, CLEANWIN, PIL, WSTATE</td>
</tr>
<tr>
<td>18</td>
<td>IUG_TSTATE</td>
<td>R</td>
<td>以下レジスタの UE。</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TSTATE, TPC, TNPC, TXAR</td>
</tr>
<tr>
<td>17</td>
<td>IUG_%F</td>
<td>R</td>
<td>浮動小数点レジスタ (拡張レジスタを含む)、または FPRS, FSR, GSR の UE。</td>
</tr>
<tr>
<td>16</td>
<td>IUG_%R</td>
<td>R</td>
<td>整数レジスタ (拡張レジスタを含む)、または y, CCR, ASI の UE。</td>
</tr>
<tr>
<td>14</td>
<td>IUG_WDT</td>
<td>R</td>
<td>一番目のウォッチドッグタイムアウト。単独 ADE 通知時に IUG_WDT = 1 がセットされていると、TPC が指す命令の実行は中断され、結果は不定である。</td>
</tr>
<tr>
<td>10</td>
<td>IUG_DTLB</td>
<td>R</td>
<td>load/store, demap 時に DTQB で UE が発生した場合、1 がセットされる。DTLB で以下の事象が起きていることを示す。</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DTQB_DATA_ACCESS, DTQB_TAG_ACCESS で DTLB を読み出そうとした際、DTLB の data または tag で UE が発生した場合。</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DTLB への書き込み、demap が失敗した場合。TPC はエラーを起こした命令かその次の命令を指す。</td>
</tr>
</tbody>
</table>

bit <22:8> の各ビットは単独 ADE 例外発生の要因を表わす。1 のとき、例外が発生していることを示す。bit <22:16> は CPU 内レジスタでのエラーを表わす。これらエラーの検出条件は Appendix P.8、“レジスタで起きたエラーの処理方法”を参照。
IUG_ITLB

• load/store, demap 時に ITLB で UE 発生した場合、1 がセットされる。ITLB で以下の事象が起きていることを示す。
  • ITLB_DATA_ACCESS, ITLB_TAG_ACCESS で ITLB を読み出そうとした際、ITLB の data または tag で UE 発生した場合。
  • ITLB への書き込み、demap が失敗した場合。TPC はエラーを起こした命令かその次の命令を指す。

IUG_COREERR

CPU core でエラーが起きたことを示す。命令実行に関するリソースで、ソフトウェアに見えないものでエラーが起きた場合に 1 がセットされる。
ソフトウェアから見えるレジスタでエラーが起き、そのレジスタを読み出す命令が実行されたとき、そのレジスタのエラーを示すビットがセットされるが、IUG_COREERR はセットされる場合もされない場合もある。

### TABLE P-9  ASI_UGESR のフィールドの説明 (2 of 3)

<table>
<thead>
<tr>
<th>ビット</th>
<th>フィールド名</th>
<th>アクセス</th>
<th>説明</th>
</tr>
</thead>
</table>
| 9     | IUG_ITLB    | R        | - load/store, demap 時に ITLB で UE 発生した場合、1 がセットされる。ITLB で以下の事象が起きていることを示す。
|       |             |          | - ITLB_DATA_ACCESS, ITLB_TAG_ACCESS で ITLB を読み出そうとした際、ITLB の data または tag で UE 発生した場合。
|       |             |          | - ITLB への書き込み、demap が失敗した場合。TPC はエラーを起こした命令かその次の命令を指す。 |
| 8     | IUG_COREERR | R        | CPU core でエラーが起きたことを示す。命令実行に関するリソースで、ソフトウェアに見えないものでエラーが起きた場合に 1 がセットされる。
|       |             |          |ソフトウェアから見えるレジスタでエラーが起き、そのレジスタを読み出す命令が実行されたとき、そのレジスタのエラーを示すビットがセットされるが、IUG_COREERR はセットされる場合もされない場合もある。 |
### P.4.2 async_data_error (ADE) トラブル発生時の処理

単独または多重 ADE が起きる条件は、P.2.2 で定義された通りである。この節では発生した際の処理方法を説明する。

1. ADE が発生する条件は、以下のいずれかである。
   - ASI_ERROR_CONTROL.UGE_HANDLER = 0 の場合、I_UGE と/または A_UGEs が検出された場合、単独 ADE 例外が通知される。
   - ASI_ERROR_CONTROL.UGE_HANDLER = 1 の場合、I_UGE と AUGE, DAE のひとつまたは複数が検出された場合、多重 ADE 例外が通知される。

2. 状態遷移、トラブルハンドラのアドレス計算、TL の処理は以下の順で行われる。
状態遷移

TL = MAXTL のときは、ADE 例外の動作を中断し、error_state に遷移する。
CPU が実行状態で TL = MAXTL - 1 のときは、RED_state に遷移する。

b. トランプハンドラのアドレス計算

CPU が実行ステートのときは、TBA, TT, TL からアドレスを計算する。
それ以外、つまり RED_state では、RSTVaddr + A016 がセットされる。

TL に 1 を加算する。

3. TSTATE, TPC, TNPC, TXAR の更新。

ADE 例外が通知される直前の PSTATE, PC, NPC, XAR が TSTATE, TPC, TNPC, TXAR にコピーされる。元のレジスタに UE が含まれていても、そのままコピーされる。

4. その他のレジスタ値の更新。

以下の 3 種類のレジスタが更新される。

a. レジスタの自動検証。

ハードウェアにより以下のレジスタが更新される。

<table>
<thead>
<tr>
<th>レジスタ</th>
<th>更新する条件</th>
<th>更新値</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSTATE</td>
<td>常に</td>
<td>AG = 1, MG = 0, IG = 0, IE = 0, PRIV = 1, AM = 0, PEF = 1, RED = 0, MM = 00, TLE = 0, CLE = 0.</td>
</tr>
<tr>
<td>PC</td>
<td>常に</td>
<td>ADE トランプのアドレス。</td>
</tr>
<tr>
<td>nPC</td>
<td>常に</td>
<td>ADE トランプのアドレス +4。</td>
</tr>
<tr>
<td>CCR</td>
<td>レジスタのデータが訂正不能エラーを起こしているとき</td>
<td>0。</td>
</tr>
</tbody>
</table>
書き込まれたレジスタに存在していたエラーは消去される。
上記以外のレジスタおよびTLBエントリに存在するエラーは保持されたままとなる。

b. ASI_UGESRの更新。

<table>
<thead>
<tr>
<th>ビットフィールド</th>
<th>単独ADEトラップでの更新</th>
<th>多重ADEトラップでの更新</th>
</tr>
</thead>
<tbody>
<tr>
<td>63:6エラー表示</td>
<td>すべてのビットが更新される。</td>
<td>変更されない。</td>
</tr>
<tr>
<td></td>
<td>検出されたすべてのI_UGE, A_UGEが同時に表示される。</td>
<td></td>
</tr>
<tr>
<td>5:4INSTEND</td>
<td>TPCが指す命令の終了方法が表示される。</td>
<td>変更されない。</td>
</tr>
<tr>
<td>2</td>
<td>MUGE_DAE 0がセットされる。</td>
<td>DAEで多重ADE例外が起きた場合、1がセットされる。それ以外では変更されない。</td>
</tr>
<tr>
<td>1</td>
<td>MUGE_IAE 0がセットされる。</td>
<td>IAEで多重ADE例外が起きた場合、1がセットされる。それ以外では変更されない。</td>
</tr>
<tr>
<td>0</td>
<td>MUGE_IUGE 0がセットされる。</td>
<td>IUGEで多重ADE例外が起きた場合、1がセットされる。それ以外では変更されない。</td>
</tr>
</tbody>
</table>

c. ASI_ERROR_CONTROLの更新。

単独ADE例外時、ASI_ERROR_CONTROL.UGE_HANDLERに1がセットされる。RETRYまたはDONEが実行されるまではUGE_HANDLERは1のままで、ハードウェアにてエラー処理中であることを伝える。
多重 ADE が起きると、ASI_ERROR_CONTROL.WEAK_ED が 1 にセットされ、CPU はエラー検出を弱めて動作する。

5. ASI_ERROR_CONTROL.UGE_HANDLER に 0 がセットされる。

RETRY または DONE 命令が実行されると、完了のときに UGE_HANDLER に 0 がセットされる。

### P.4.3 ADE トラブル発生した時の命令の実行状況

SPARC64 IXfx では async_data_error 通知により終了させられた命令、つまり TPC が指す命令の終了状態は以下の 3 種類のいずれかになる。

- Precise
- Retryable but not precise (JPS1 定義外)
- Not retryable (JPS1 定義外)

単独 ADE 発生時は、TPC が指す命令の終了方法は、ASI_UGESR.INSTEND に表示される。

各終了方法の相違点を TABLE P-10 に示す。

<table>
<thead>
<tr>
<th>TABLE P-10 async_data_error 通知時点の命令の実行状況</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TPC が指す命令</strong></td>
</tr>
<tr>
<td>前回の ADE,IAE,DAE 発生から TPC が指す命令の直前までの命令</td>
</tr>
<tr>
<td><strong>TPC の次以降の命令の実行</strong></td>
</tr>
</tbody>
</table>

Ver 12, 2 Dec. 2013

F. Appendix P Error Handling 283
ADE トラブルハンドラの処理例

ADE トラブルハンドラの例を擬似 C コードで示す。このコードでは以下のエラーの復旧を目的としている。

- CPU 内部の RAM やレジスタ
- 加算機
- CPU 内の一時レジスタやデータバス

```
void expected_software_handling_of_ADE_trap()
{
  /*
   * ここから #1 地点までは、レジスタウィンドウを制御するレジスタが壊れているかもしれないので、%r0-%r7 レジスタのみを使う。
   * 可能ならば、単独 ADE のハンドラ全体で %r0-%r7 だけを使うのが望ましい。
   */

  ASI_SCRATCH_REGp ← %rX; /* working register 1 */
  ASI_SCRATCH_REGq ← %rY; /* working register 2 */
  %rX ← ASI_UGESR;

  if (%rX & 0x07) ≠ 0) { /*多重 ADE 例外が起きている*/
    / System Dump を採取する*/
  }

  if (%r.IUG_%R == 1) {
    %rX と %rY 以外の r1-%r63 ← %r0;
    %y ← %r0;
    %tstate.pstate ← %r0;
    /* %tstate.pstate の asi がエラーしているかもしれないのに*/
  }

else {
```

<table>
<thead>
<tr>
<th>TABLE P-10 async_data_error 通知時点の命令の実行状況</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precise</td>
</tr>
<tr>
<td>-----------------</td>
</tr>
<tr>
<td>できる。</td>
</tr>
</tbody>
</table>

例外を通知されたプログラムが、単独 ADE で報告されたエラーでダメージを受けない場合の、実行を継続できるか。
%rX, %rY, ASI_SCRATCH_REGp, ASI_SCRATCH_REGq を使
作業に必要なレジスタを確保するため, %rX, %rY,
ASI_SCRATCH_REGp, ASI_SCRATCH_REGq を USE %r1-%r7 を
退避する;

/*
 * PSTATE.AG == 1 であるコンテキストでエラーが起き、
 * その処理に復旧するのならば、全 %r レジスタの退避と
 * 復元が必要である。
 */

if (ASI_UGESR.IUG_PSTATE == 1) {
  %tstate.pstate ← %r0;
  %tpc ← %r0;
  %pil ← %r0;
  %wstate ← %r0;
  レジスタウィンドウ中の全レジスタ ← %r0;
  レジスタウィンドウ制御レジスタ (CWP, CANSAVE,
   CANRESTORE, OTHERWIN, CLEANWIN) には適切な値を設定する；
}

/* Point#1
 * これまででレジスタウィンドウ制御レジスタのエラー検証が完了して
 * いるので、これ以降 %r0-%r7 以外のウィンドウレジスタを使用可。
 */

if (ASI_UGESR.IAUG_CRE == 1 || ASI_UGESR.IAUG_TSBCTXT == 1
 || ASI_UGESR.IUG_TSBP == 1
 || ASI_UGESR.IUG_TSTATE == 1
 || ASI_UGESR.IUG_%F==1) {
  これらのエラーを起こす可能性のあるレジスタをすべて検証する；
}

if (ASI_UGESR.IUG_DTLB == 1) {
  DTLB に対し demap_all を実行する；
  /*
   * fDTLB のロックされた TTE は demap_all では消去されない。
   */
}

if (ASI_UGESR.IUG_ITLB == 1) {
  ITLB に対し demap_all を実行する；
  /*
fITLB のロックされた TTE は demap_all では消去されない。
*/

if (ASI_UGESR.bits<22:14> == 0 &&
    ASI_UGESR.INSTEND == 0 || ASI_UGESR.INSTEND == 1) {
  ++ADE_trap_retry_per_unit_of_time;
  if (ADE_trap_retry_per_unit_of_time < threshold)
    RETRY で例外が起きる前のコンテキストに復帰する;
  else
    ADE 例外発生回数が閾値を超えたので、OS を停止する;
} else if (ASI_UGESR.bits<22:18> == 0 &&
        ASI_UGESR.bits<15:14> == 0 &&
        ASI_UGESR.PRIV == 0) {
  ++ADE_trap_kill_user_per_unit_of_time;
  if (ADE_trap_kill_user_per_unit_of_time
      < threshold) {
    ユーザプロセスの実行を停止し、OS は処理を継続する;
  } else {
    ADE 例外発生によるユーザプロセス中断回数が閾値を超えたので、OS を停止する;
  }
} else {
  復旧不可能な緊急エラーが起きたので、OS を停止する。
}

---

P.5 Instruction Access Errors

Appendix F.5, “Faults and Traps” (page 178) を参照。

---

P.6 Data Access Errors

Appendix F.5, “Faults and Traps” (page 178) を参照。
P.7 抑止可能エラー

P.7.1 ASI_ASYNC_FAULT_STATUS (ASI_AFSR)

ASI_ASYNC_FAULT_STATUS は、抑止可能エラーが発生した際その種類を表示するレジスタである。各ビットは 1 にセットされると、システムソフトウェアによって書き込まれるまでその値を保持しつづける。TABLE P-11 に AFSR のフィールドを示す。

ASI_ASYNC_FAULT_STATUS は、抑止可能エラーが発生した際その種類を表示するレジスタである。各ビットは 1 にセットされると、システムソフトウェアによって書き込まれるまでその値を保持しつづける。TABLE P-11 に AFSR のフィールドを示す。

<table>
<thead>
<tr>
<th>ビット</th>
<th>フィールド名</th>
<th>アクセス</th>
<th>説明</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>Reserved</td>
<td>RW1C</td>
<td>CPU の L2 キャッシュが縮退した場合、1がセットされる。</td>
</tr>
<tr>
<td>11</td>
<td>DG_U2$</td>
<td>RW1C</td>
<td>L1I キャッシュ、L1D キャッシュ、sTLB、sDTLB が縮退した場合、1がセットされる。</td>
</tr>
<tr>
<td>10</td>
<td>DG_D1$sTLB</td>
<td>RW1C</td>
<td>L1I キャッシュ、L1D キャッシュ、sTLB、sDTLB が縮退した場合、1がセットされる。</td>
</tr>
<tr>
<td>9</td>
<td>Reserved</td>
<td>R</td>
<td>読み出しには常に 0 が返り、書き込みは無視される。</td>
</tr>
<tr>
<td>3</td>
<td>UE_DST_BETO</td>
<td>RW1C</td>
<td>メモリ書き込みに対してバスエラーが返ってきた場合、1がセットされる。</td>
</tr>
<tr>
<td>2</td>
<td>Reserved</td>
<td>R</td>
<td>読み出しには常に 0 が返り、書き込みは無視される。</td>
</tr>
<tr>
<td>1</td>
<td>UE_RAW_L2SNSD</td>
<td>RW1C</td>
<td>L1D キャッシュでマークされていない UE が発見された場合、1がセットされる。</td>
</tr>
<tr>
<td>0</td>
<td>UE_RAW_D1SNSD</td>
<td>RW1C</td>
<td>L1D キャッシュでマークされていない UE が発見された場合、1がセットされる。</td>
</tr>
<tr>
<td>その他</td>
<td>Reserved</td>
<td>R</td>
<td>読み出しには常に 0 が返り、書き込みは無視される。</td>
</tr>
</tbody>
</table>
Note – disruptingなバスエラーまたはタイムアウトは、AFSR.UE_DST_BETO、DSFSR.BERR、DSFSR.RTOのいずれか1つで報告される。

Note – 参照するとAFSR.UE_DST_BETOがセットされるような領域に対する書き込みの直後に同一アドレスを読み出すと、ストアバッファにあるデータが読み出され、data_access_errorが発生しないことがある。AFSR.UE_DST_BETOは書き込みの実行後にセットされる。

P.7.2 抑止可能エラーに対するソフトウェア処理
すべての抑止可能エラーは、エラーを記録することが望ましい。この節では各抑止可能エラーに対して期待されるソフトウェアでの処理法を説明する。

■ DG_L1S, DG_U2S — 以下のようなCPUの状態が報告される。
  ■ L1Iキャッシュ、L1Dキャッシュ、L2キャッシュ、sITLB、sDTLBでウェイが縮退したので、性能低下の可能性があることを示す。
  ■ CPUの可用性が低下していることを示す。L1Iキャッシュ、L1Dキャッシュ、L2キャッシュ、sITLB、sDTLBが1ウェイで動作している状態で、最後のウェイでもエラーが発生した場合は、error_state遷移エラーとなる。
    必要ならば、ソフトウェアはエラーを起こしているCPUの使用を止める。

■ UE_DST_BETO — このエラーが起きるのは以下のどちらかの場合である。
  ■ DTLBに間違ったTTEが存在する。
  ■ 物理アドレスアクセスASIで、無効な領域にアクセスした。
    どちらの場合も、原因はシステムソフトウェアのバグである。エラー情報を元にシステムソフトウェアを修正する。

■ UE_RAW_L2SINS, and UE_RAW_D1SINS — このエラーの処理は、
  ■ 可能ならば、UEを含むキャッシュラインのエラーを消去する。これによりキャッシュラインに載っていたデータは失われることに注意。
  ■ ECC_error例外が通知されたが、ASI_AFSRにエラーが記録されていない場合 — ECC_error例外を無視する。
    詳細は“エラー検出時の動作”(page 265)を参照。

P.8 レジスタで起きたエラーの処理方法
この節では、以下のレジスタで起きたエラーの処理方法について説明する。
P.8.1
特権・非特権レジスタの処理方法

TABLE P-12 内で使われる用語の定義は以下の通りである。

<table>
<thead>
<tr>
<th>用語</th>
<th>意味</th>
</tr>
</thead>
<tbody>
<tr>
<td>エラー検出条件</td>
<td>InstrAccess 命令実行によりレジスタがアクセスされたときにエラーが検出される。</td>
</tr>
<tr>
<td>エラー訂正</td>
<td>W レジスタ全体への書き込みでエラーが訂正される。</td>
</tr>
<tr>
<td></td>
<td>ADE trap ハードウェアが、async_data_error 例外によるトラップ処理中 にレジスタ全体への書き込みを行い、エラーが訂正される。</td>
</tr>
</tbody>
</table>

TABLE P-12 に、特権、非特権レジスタのエラー処理方法を示す。PC、nPC、PSTATE、CWP、ASI、XAR で緊急エラーが起きた場合、async_data_error トラブルハンドラに処理が移った時点で、エラーも含めてそれぞれ TPC、TNPC、TSTATE、TXAR にコピーされていることに注意。

TABLE P-12 特権・非特権レジスタのエラーの扱い（1 of 2）

<table>
<thead>
<tr>
<th>レジスタ</th>
<th>RW</th>
<th>エラー保護</th>
<th>エラー検出条件</th>
<th>エラー種類</th>
<th>エラー訂正</th>
</tr>
</thead>
<tbody>
<tr>
<td>$rn$</td>
<td>RW</td>
<td>パリティ</td>
<td>InstrAccess</td>
<td>IUG_%R</td>
<td>W</td>
</tr>
<tr>
<td>$fn$</td>
<td>RW</td>
<td>パリティ</td>
<td>InstrAccess</td>
<td>IUG_%F</td>
<td>W</td>
</tr>
<tr>
<td>PC</td>
<td>R</td>
<td>パリティ</td>
<td>Always</td>
<td>IUG_PSTATE</td>
<td>ADE trap</td>
</tr>
<tr>
<td>nPC</td>
<td>R</td>
<td>パリティ</td>
<td>Always</td>
<td>IUG_PSTATE</td>
<td>ADE trap</td>
</tr>
<tr>
<td>PSTATE</td>
<td>RW</td>
<td>パリティ</td>
<td>Always</td>
<td>IUG_PSTATE</td>
<td>ADE trap, W</td>
</tr>
<tr>
<td>TBA</td>
<td>RW</td>
<td>パリティ</td>
<td>PSTATE.RED = 0</td>
<td>error_state</td>
<td>W(OBPが行う)</td>
</tr>
<tr>
<td>PIL</td>
<td>RW</td>
<td>パリティ</td>
<td>PSTATE.IE = 1</td>
<td>InstrAccess</td>
<td>IUG_PSTATE</td>
</tr>
<tr>
<td>CWP、CANSAVE、CARESTORE、OTHERWIN、CLEANWIN</td>
<td>RW</td>
<td>パリティ</td>
<td>Always</td>
<td>IUG_PSTATE</td>
<td>ADE trap, W</td>
</tr>
<tr>
<td>TT</td>
<td>RW</td>
<td>なし</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>TL</td>
<td>RW</td>
<td>パリティ</td>
<td>PSTATE.RED = 0</td>
<td>error_state</td>
<td>W(OBPが行う)</td>
</tr>
<tr>
<td>TPC</td>
<td>RW</td>
<td>パリティ</td>
<td>InstrAccess</td>
<td>IUG_TSTATE</td>
<td>W</td>
</tr>
<tr>
<td>TNPC</td>
<td>RW</td>
<td>パリティ</td>
<td>InstrAccess</td>
<td>IUG_TSTATE</td>
<td>W</td>
</tr>
<tr>
<td>TSTATE</td>
<td>RW</td>
<td>パリティ</td>
<td>InstrAccess</td>
<td>IUG_TSTATE</td>
<td>W</td>
</tr>
<tr>
<td>WSTATE</td>
<td>RW</td>
<td>パリティ</td>
<td>Always</td>
<td>IUG_PSTATE</td>
<td>ADE trap, W</td>
</tr>
<tr>
<td>VER</td>
<td>R</td>
<td>なし</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>
TABLE P-13 内で使われる用語の定義は以下の通りである。

<table>
<thead>
<tr>
<th>列</th>
<th>用語</th>
<th>意味</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>エラー検出条件</td>
<td>AUG always</td>
</tr>
<tr>
<td></td>
<td></td>
<td>エラーは、ASI_ERROR_CONTROL.UGE_HANDLER = 0かつASI_ERROR_CONTROL.WEAK_ED = 0のときに検出される。</td>
</tr>
<tr>
<td></td>
<td>InstrAccess</td>
<td>命令実行によりレジスタがアクセスされたときに検出される。</td>
</tr>
<tr>
<td></td>
<td>エラー種類</td>
<td>(I)AUG_xxx</td>
</tr>
<tr>
<td></td>
<td></td>
<td>自律エラーが起き、ASI_UGESR.IAUG_xxx = 1となった。</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(I)AUG_xxx</td>
</tr>
<tr>
<td></td>
<td></td>
<td>命令エラーが起き、ASI_UGESR.IAUG_xxx = 1となった。</td>
</tr>
<tr>
<td></td>
<td>エラー訂正</td>
<td>W</td>
</tr>
<tr>
<td></td>
<td></td>
<td>レジスタ全体への書き込みでエラーが訂正される。</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADE trap</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ハードウェアが、async_data_error例外によるトラップ処理中にレジスタ全体への書き込みを行い、エラーが訂正される。</td>
</tr>
</tbody>
</table>

P.8.2 ASR レジスタの処理方法

TABLE P-13 内で使われる用語の定義は以下の通りである。

<table>
<thead>
<tr>
<th>ASR Number</th>
<th>レジスタ</th>
<th>RW</th>
<th>エラー保護</th>
<th>エラー検出条件</th>
<th>エラー種類</th>
<th>エラー訂正</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>PCR</td>
<td>RW</td>
<td>なし</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>17</td>
<td>PIC</td>
<td>RW</td>
<td>なし</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>18</td>
<td>DCR</td>
<td>R</td>
<td>なし</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>19</td>
<td>GSR</td>
<td>RW</td>
<td>パリティ</td>
<td>Always</td>
<td>IUG_%F</td>
<td>ADE trap, W</td>
</tr>
<tr>
<td>20</td>
<td>SET_SOFTINT</td>
<td>W</td>
<td>なし</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

1.拡張レジスタを含む。
2.サスペンド中のスレッドにはerror_state遷移エラーとして通知される。
3.訂正のために書く値は0x8000_0000_0000_0000
### STICK Behavior on Error

STICK でエラーが起きた場合、TABLE P-13 の検出条件とは関係なくカウントアップは停止する。

### P.8.3 ASI レジスタの処理方法

TABLE P-14 内で使われる用語の定義は以下の通りである。

<table>
<thead>
<tr>
<th>列</th>
<th>用語</th>
<th>意味</th>
</tr>
</thead>
<tbody>
<tr>
<td>なし</td>
<td>バリティ</td>
<td>バリティで保護されている。</td>
</tr>
<tr>
<td>Triple</td>
<td>レジスタが三重化されている。</td>
<td></td>
</tr>
<tr>
<td>ECC</td>
<td>ECC で保護されている。</td>
<td></td>
</tr>
<tr>
<td>Gecc</td>
<td>生成された ECC で保護されている。</td>
<td></td>
</tr>
</tbody>
</table>

---

1. サスペンド中のスレッドには error_state 遷移エラーとして通知される。
エラー検出条件

<table>
<thead>
<tr>
<th>変数</th>
<th>意味</th>
</tr>
</thead>
<tbody>
<tr>
<td>Always</td>
<td>エラーは常に検出される。</td>
</tr>
<tr>
<td>AUG always</td>
<td>エラーは、ASI_ERROR_CONTROL.UGE_HANDLER = 0 かつ ASI_ERROR_CONTROL.WEAK_ED = 0 のときに検出される。</td>
</tr>
<tr>
<td>LDAX</td>
<td>エラーは、レジスタの読み出し時に検出される。</td>
</tr>
<tr>
<td>ITLB write</td>
<td>エラーは、ITLB 書き込みまたは demap による更新時に検出される。</td>
</tr>
<tr>
<td>DTLB write</td>
<td>エラーは、DTLB 書き込みまたは demap による更新時に検出される。</td>
</tr>
<tr>
<td>Used by TLB</td>
<td>エラーは、TLB 検索の際に参照されたときに検出される。</td>
</tr>
<tr>
<td>Enabled</td>
<td>エラーは、その機能が有効なときに検出される。</td>
</tr>
<tr>
<td>intr_receive</td>
<td>エラーは、割り込みパケットを受信したときに検出される。受信パケットの中に UE があると、vector_interrupt 例外が通知されるが ASI_INTR_RECEIVE.BUSY は 0 にセットされる。ASI_INTR_RECEIVE.BUSY に 0 を書き込むと、新たなパケットが受信可能となる。</td>
</tr>
</tbody>
</table>

エラー種類

<table>
<thead>
<tr>
<th>エラー種類</th>
<th>error_state</th>
</tr>
</thead>
<tbody>
<tr>
<td>error_stat</td>
<td>error_state 遷移エラー。</td>
</tr>
<tr>
<td>I(AUG_xxxx)</td>
<td>自律エラーが発生し、エラー情報が ASI_UGESR.IAUG_xxxx = 1 に表示される。</td>
</tr>
<tr>
<td>II(AUG_xxxx)</td>
<td>命令エラーが発生し、エラー情報が ASI_UGESR.IAUG_xxxx = 1 に表示される。</td>
</tr>
<tr>
<td>Other</td>
<td>ASI_UGESR の、エラーに対応するビットに 1 がセットされる。</td>
</tr>
</tbody>
</table>

エラー訂正

| RED trap | RED_state トラブルの発生時に値が更新されエラーが訂正される。 |
| W | ASI レジスタへの書き込みでエラーが訂正される。 |
| W_other_I | 以下のレジスタ全てを更新することでエラーが訂正される。 |
| W_other_D | 以下のレジスタ全てを更新することでエラーが訂正される。 |
| Interrupt receive | 割り込みパケットの受信でエラーが訂正される。 |
TABLE P-14 に、ASI レジスタのエラー処理方法を示す。

<table>
<thead>
<tr>
<th>ASI</th>
<th>VA</th>
<th>レジスタ</th>
<th>RW</th>
<th>エラー保護</th>
<th>エラー検出条件</th>
<th>エラー種類</th>
<th>エラー訂正</th>
</tr>
</thead>
<tbody>
<tr>
<td>4516</td>
<td>0016</td>
<td>DCU_CONTROL</td>
<td>RW</td>
<td>パリティ</td>
<td>Always</td>
<td>error_state</td>
<td>RED trap</td>
</tr>
<tr>
<td>0816</td>
<td></td>
<td>MEMORY_CONTROL</td>
<td>RW</td>
<td>パリティ</td>
<td>Always</td>
<td>error_state</td>
<td>RED trap</td>
</tr>
<tr>
<td>4816</td>
<td>0016</td>
<td>INTR_DISPATCH_STATUS</td>
<td>R</td>
<td>パリティ</td>
<td>LDXA またはレジスタ更新</td>
<td>I(U)G_CRE (UE)</td>
<td>None</td>
</tr>
<tr>
<td>4916</td>
<td>0016</td>
<td>INTR_RECEIVE</td>
<td>RW</td>
<td>パリティ</td>
<td>LDXA</td>
<td>I(U)G_CRE (UE)</td>
<td>None</td>
</tr>
<tr>
<td>4A16</td>
<td>—</td>
<td>SYS_CONFIG</td>
<td>R</td>
<td>なし</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4B16</td>
<td>0016</td>
<td>STICK_CNTL</td>
<td>RW</td>
<td>Triple</td>
<td>Always</td>
<td>—</td>
<td>Always</td>
</tr>
<tr>
<td>4C16</td>
<td>0816</td>
<td>ASYNC_FAULT_STATUS</td>
<td>RW1C</td>
<td>なし</td>
<td></td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>4C16</td>
<td>0816</td>
<td>URGENT_ERROR_STATUS</td>
<td>R</td>
<td>なし</td>
<td></td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>4C16</td>
<td>1016</td>
<td>ERROR_CONTROL</td>
<td>RW</td>
<td>パリティ</td>
<td>Always</td>
<td>error_state</td>
<td>RED trap</td>
</tr>
<tr>
<td>4C16</td>
<td>1816</td>
<td>STCHG_ERROR_INFO</td>
<td>R,</td>
<td>なし</td>
<td></td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>4F16</td>
<td>0016-3816</td>
<td>SCRATCH_REGS</td>
<td>RW</td>
<td>パリティ</td>
<td>LDXA</td>
<td>I(U)G_COREERR</td>
<td>W</td>
</tr>
<tr>
<td>5016</td>
<td>0016</td>
<td>IMMU_TAG_TARGET</td>
<td>R</td>
<td>パリティ</td>
<td>LDXA</td>
<td>I(U)G_TSBP</td>
<td>W_other_I</td>
</tr>
<tr>
<td>5016</td>
<td>1816</td>
<td>IMMU_SFSR</td>
<td>RW</td>
<td>なし</td>
<td></td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>5016</td>
<td>2816</td>
<td>IMMU_TSB_BASE</td>
<td>RW</td>
<td>パリティ</td>
<td>LDXA</td>
<td>I(U)G_TSBCTX</td>
<td>W</td>
</tr>
<tr>
<td>5016</td>
<td>3016</td>
<td>IMMU_TAG_ACCESS</td>
<td>RW</td>
<td>パリティ</td>
<td>LDXA</td>
<td>I(U)G_TSBP</td>
<td>W</td>
</tr>
<tr>
<td>5016</td>
<td>6016</td>
<td>IMMU_TAG_ACCESS_EXT</td>
<td>RW</td>
<td>パリティ</td>
<td>LDXA</td>
<td>I(U)G_TSBP</td>
<td>W</td>
</tr>
<tr>
<td>5016</td>
<td>7816</td>
<td>IMMU_SFPAR</td>
<td>RW</td>
<td>パリティ</td>
<td>LDXA</td>
<td>I(U)G_CRE</td>
<td>W</td>
</tr>
<tr>
<td>5316</td>
<td>—</td>
<td>SERIAL_ID</td>
<td>R</td>
<td>なし</td>
<td></td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>5416</td>
<td>—</td>
<td>ITLB_DATA_IN</td>
<td>W</td>
<td>パリティ</td>
<td>ITLB 書き込み</td>
<td>I(U)G_ITLB</td>
<td>DemapAll</td>
</tr>
<tr>
<td>5516</td>
<td>—</td>
<td>ITLB_DATA_ACCESS</td>
<td>RW</td>
<td>パリティ</td>
<td>LDXA</td>
<td>I(U)G_ITLB</td>
<td>DemapAll</td>
</tr>
<tr>
<td>5616</td>
<td>—</td>
<td>ITLB_TAG_READ</td>
<td>R</td>
<td>パリティ</td>
<td>LDXA</td>
<td>I(U)G_ITLB</td>
<td>DemapAll</td>
</tr>
<tr>
<td>5716</td>
<td>—</td>
<td>IMMU_DEMAP</td>
<td>W</td>
<td>パリティ</td>
<td>ITLB 書き込み</td>
<td>I(U)G_ITLB</td>
<td>DemapAll</td>
</tr>
<tr>
<td>5816</td>
<td>0016</td>
<td>DMMU_TAG_TARGET</td>
<td>R</td>
<td>パリティ</td>
<td>LDXA</td>
<td>I(U)G_TSBP</td>
<td>W_other_D</td>
</tr>
<tr>
<td>5816</td>
<td>0816</td>
<td>PRIMARY_CONTEXT</td>
<td>RW</td>
<td>パリティ</td>
<td>LDXA</td>
<td>I(U)G_TSBCTX</td>
<td>W</td>
</tr>
</tbody>
</table>

5816 1016 | SECONDARY_CONTEXT | RW | パリティ = P_CONTEXT | I(U)G_TSBCTX | W         |
| 5816 1816 | DMMU_SFSR         | RW | なし      |                | —            | —         |
| 5816 2016 | DMMU_SFAR         | RW | パリティ | LDXA            | I(U)G_CRE    | W         |
| 5816 2816 | DMMU_TSB_BASE     | RW | パリティ | LDXA            | I(U)G_TSBCTX | W         |
| 5816 3016 | DMMU_TAG_ACCESS   | RW | パリティ | LDXA            | I(U)G_TSBP   | W         |

Ver 12, 2 Dec. 2013  F. Appendix P  Error Handling  293
キャッシュで起きたエラーの処理方法

この節では、キャッシュタグ、キャッシュデータのエラー対処について説明する。

---

P.9

キャッシュで起きたエラーの処理方法

この節では、キャッシュタグ、キャッシュデータのエラー対処について説明する。
P.9.1 キャッシュタグで起きたエラーの処理方法

L1I キャッシュタグ、L1D キャッシュタグのエラー

L2 キャッシュは、L1I キャッシュタグと L1D キャッシュタグのコピーを保持している。L1I キャッシュタグとそのコピー、L1D キャッシュタグとそのコピーはいずれもパリティで保護されている。

L1D キャッシュタグまたは L1D キャッシュタグのコピーでエラーが検出されると、ハードウェアは自動的にエラーが起きていない方から起きている方へコピーを行う。この処理でエラーが修復されたときは、プログラムの実行には影響を与えない。

同様に、L1I キャッシュタグまたは L1I キャッシュタグのコピーでエラーが検出されると、ハードウェアは自動的にエラーが起きていない方から起きている方へコピーを行う。この処理でエラーが修復されたときは、プログラムの実行には影響を与えない。

上記の動作でエラーが修復されないときは、タグコピーが繰り返される。固定故障によるエラーの場合、最終的にウォッチドッグタイムアウトかフェイタルエラーが検出される。

L2 キャッシュタグのエラー

L2 キャッシュタグは ECC により保護され、1 ビットエラーの訂正と 2 ビットエラーの検出ができる。

L2 キャッシュタグで訂正可能なエラーが発見された場合、ハードウェアは自動的に正しいデータを上書きしデータを修復する。エラーはシステムソフトウェアには報告されない。

L2 キャッシュタグで訂正不可能なエラーが発見された場合、致命的エラーとして通知され、CPU はフェイタルエラーステートに遷移する。

P.9.2 L1I キャッシュデータで起きたエラーの処理方法

L1I キャッシュデータは、8 バイト単位でパリティ保護されている。

命令フェッチ中、L1I キャッシュデータにパリティエラーが発見されると、ハードウェアは以下の順序で処理を行う。

1. パリティエラーを含むキャッシュラインを L2 キャッシュから読み出す。

L2 キャッシュから読み出されたデータは、UE を含まないデータか、マーク済 UE を含むデータのどちらかである。なぜなら、エラーマークは L2 キャッシュから外に出て行く (outgoing) データにのみ行われるからである。
2. L2 キャッシュから読み出された 8 バイトごとに、

a. その 8 バイトに UE がなければ、L1I キャッシュに保存し、必要なら命令フェッチ部に供給する。

   L1I キャッシュ再コピーの際にエラー訂正が行われても、直接システムソフトウェアに報告は上がらない。

b. その 8 バイトにマーク済 UE があるときは、L1I キャッシュデータの当該 8 バイトに対するパリティビットを、パリティエラーを示すように設定する。必要なら命令フェッチ部にデータを供給する。

3. フェッチ部でのエラーのある命令の扱いは、

   パリティエラーしている命令をフェッチしたが、その命令は実行されず、ソフトウェアから見える状態が変わらないとき、その命令は単に捨てる。

   フェッチした命令が実行され完了したときは、instruction_access_error例外が通知され、ASI_ISFSRにマーク済 UEを検出したときと、そのERROR_MARK_IDが表示される。

P.9.3 L1D キャッシュデータで起きたエラーの処理方法

L1D キャッシュデータは 8 バイト単位で ECC により保護され、1 ビットエラーの訂正と 2 ビットエラーの検出ができる。

L1D キャッシュデータの訂正可能エラー

L1D キャッシュデータで訂正可能エラーが発見された場合、データはハードウェアにより自動的に訂正される。訂正可能エラーはシステムソフトウェアには報告しない。

L1D キャッシュデータのマーク済 UE

L1D キャッシュから L2 キャッシュへのライトバック中に、L1D キャッシュデータでマーク済 UE が発見されると、L1D キャッシュデータと ECC は変更なしで L2 キャッシュに書き戻される。つまり、L1D キャッシュのマーク済 UE は L2 キャッシュに引き継がれる。この書き戻しはシステムソフトウェアには報告されない。

L1D キャッシュのマーク済 UE に対するロード、ストア命令 (8 バイトストアを除く) には、data_access_error例外が通知される。この例外通知は precise で、ASI_DSFSRにはマークのERROR_MARK_IDが表示される。
LID キャッシュ上のマークしていない UE の L2 への書き戻し

LID キャッシュから L2 への書き戻し時にマークしていない UE が発見されると、エラーを含む 8 バイトにマーキングが行われる。このときの ERROR_MARK_ID には ASI_EIDR の値が使われる。L2 キャッシュに書き戻されることは、訂正済みかマーク済のデータのみである。

訂正が行われると ASI_AFSR.UE_RAW_D1$INSD に 1 がセットされる。

LID キャッシュ上のマークしていない UE の命令による読み出し

LID キャッシュ上のマークしていない UE の命令に対する、メモリアクセス命令の読み出しに対し、ハードウェアは以下の順に処理を行う。

1. 一旦 LID キャッシュから L2 に書き戻し、再度 L2 キャッシュから読み込む。

LID キャッシュ内のデータは、L2 のデータと同様であるか更新されているかに係わらず書き戻される。この書き戻しの際にエラーマーキングが行われる。このときの ERROR_MARK_ID には ASI_EIDR の値が使われる。LID キャッシュラインは L2 キャッシュから読み出され、ASI_AFSR.UE_RAW_D1$INSD に 1 がセットされる。

2. 通常は 1 でマークされていないエラーはマーク済になるが、この操作の途中に同一の 8 バイトに新たな UE が起きることがある。この場合 1 の処理が繰り返される。この動作は LID キャッシュウェイが縮退するまで行われる。

3. ここまでの操作で、ハードウェアは LID キャッシュデータのマークされていない UE をマーク済にする。この後メモリアクセス命令はマーク済の UE をアクセスするが、このときの動作は "LID キャッシュデータのマーク済 UE" (page 296) を参照。

P.9.4 L2 キャッシュデータで起きたエラーの処理方法

L2 キャッシュデータは 8 バイト単位で ECC により保護され、1 ビットエラーの訂正と 2 ビットエラーの検出ができる。

L2 キャッシュデータの訂正可能エラー

メモリから L2 キャッシュへ読み込まれたデータに訂正可能エラーが発見されると、ハードウェアが自動的にエラーを訂正する。例外は通知されない。
L2 キャッシュから L1I キャッシュ、L1D キャッシュへの読み込みや、メモリや他キャッシュへの書き出し時に、転送データや元データに訂正可能エラーが発見された場合、ハードウェアが自動的にエラーを訂正する。このエラーはシステムソフトウェアには報告されない。

L2 キャッシュデータのマーク済 UE

L2 キャッシュデータ上にある、マーク済 UE の 8 バイトデータは、訂正済みデータと同等に扱われる。マーク済 UE が L2 キャッシュデータ上で発見されても、エラー報告はあがらない。

メモリから L2 キャッシュへ読み込まれたデータにマーク済 UE が発見されても、変更されずそのまま L2 キャッシュに書き込まれる。

L1D キャッシュから L2 キャッシュへの書き戻しの際にマーク済 UE が発見されても、変更されずそのまま L2 キャッシュに書き込まれる。なお、マークしていない UE を書き戻すことはない。詳細は "LID キャッシュ上のマークしていない UE の L2 への書き戻し" (page 297) を参照。

L2 キャッシュから L1I キャッシュ、L1D キャッシュへの読み込みや、メモリや他キャッシュへの書き出し時に、転送データや元データにマーク済 UE が発見されても、変更されずそのまま転送される。

L2 キャッシュデータのマークされていない UE

メモリから L2 キャッシュへ読み込まれたデータにマークされていない UE が発見されると、エラーを含む 8 バイト単位でエラーマーキングが行われる。このとき ERROR_MARK_ID は 0 でマークリングされる。当該 8 バイトと ECC がマーク済データに置き換えられ、L2 キャッシュに書き込まれる。例外は通知されない。

L2 キャッシュからの読み出し（L1I キャッシュや L1D キャッシュへの読み込み）時に、マークされていない UE が発見されると、エラーを含む 8 バイト単位でエラーマーキングが行われる。このとき ERROR_MARK_ID には ASI_EIDR が使われる。ASI_AFSR.UE_RAW_L2$INSD に 1 がセットされる。

P.9.5 L1I,L1D,L2 キャッシュの自動ウェイ縮退

L1I,L1D,L2 キャッシュでエラーが発生すると、ハードウェアはキャッシュデータの一貫性を保ちながら、ウェイを縮退させる。

ウェイ縮退の条件

ハードウェアは、各キャッシュのウェイ毎に、以下のエラー発生回数の合計を計測している。
■ L1I キャッシュの各ウェイについて
  - L1I キャッシュタグと L1I キャッシュタグコピーのパリティエラー
  - L1I キャッシュデータのパリティエラー
■ L1D キャッシュの各ウェイについて
  - L1D キャッシュタグと L1D キャッシュタグコピーのパリティエラー
  - L1D キャッシュデータの訂正可能エラー
  - L1D キャッシュデータのマークされていない UE
■ L2 キャッシュの各ウェイについて
  - L2 キャッシュタグの訂正可能エラーと UE
  - L2 キャッシュデータの訂正可能エラー
  - L2 キャッシュデータのマークされていない UE
あるキャッシュウェイのカウンタが、一定時間内に定められた閾値を越えると、ハードウェアはそのキャッシュウェイを縮退させる。その手順は以下の通り。

L1I キャッシュのウェイ縮退
L1I キャッシュのウェイ w を縮退させる手順は、
1. すでに 1 ウェイ縮退している場合は、エラー箇所のみを無効化する。
2. それ以外では、
  - ウェイ w の全データが無効化され、これ以降ウェイ w への読み込みは行われない。
  - ASI_AFSR.DG_D1$STLB に 1 がセットされ、抑止可能エラーが通知される。

L1D キャッシュのウェイ縮退
L1D キャッシュのウェイ w を縮退させる手順は、
1. すでに 1 ウェイ縮退している場合は、エラー箇所のみ L2 にライトバックし、エラー箇所のみを無効化する。
2. それ以外では、
  - ウェイ w の全データが無効化され、これ以降ウェイ w への読み込みは行われない。
  - L2 キャッシュデータから変更されているデータについては、L2 キャッシュへライトバックされる。
  - ASI_AFSR.DG_D1$STLB に 1 がセットされ、抑止可能エラーが通知される。

L2 キャッシュのウェイ縮退
L2 キャッシュの縮退は、DCUCR.WEAK_SPCA = 0 のときはすぐに行われるが、DCUCR.WEAK_SPCA = 1 のときはペンディングされ、DCUCR.WEAK_SPCA = 0 になったときに開始される。
L2キャッシュのウェイを縮退させる手順は、

1. すでに他のウェイは縮退して、1ウェイしか残っていない場合は、
   ■ L2キャッシュのウェイの全データが一斉に無効化されるが、ウェイwは引き続き使用される。システム全体のデータ一意性を保つため、L2キャッシュ上のデータは破棄される。
   ■ ASI_AFSR.DG_U2に1がセットされ、抑制可能エラーが通知される。エラー通知は、キャッシュ構成の変化によらず行われる。

2. それ以外では、
   ■ システム全体のデータ一意性を保つため、ウェイwを含む全ウェイの全データが無効化される。
   ■ これ以降ウェイwは使用されない。
   ■ ASI_AFSR.DG_U2に1がセットされ、抑制可能エラーが通知される。

---

P.10 TLBで発生したエラー

この節では、TLBエントリのエラー処理方法とsTLBのウェイ縮退について説明する。

P.10.1 TLBエラーの処理

SPARC64 IxTxの各TLBのエラー保護をTABLE P-15に示す。

<p>| TABLE P-15 TLBエントリのエラー保護と検出方法 |
|-----------------|-----------------|-----------------|</p>
<table>
<thead>
<tr>
<th>TLB種類</th>
<th>フィールド</th>
<th>エラー保護</th>
<th>検出できるエラー</th>
</tr>
</thead>
<tbody>
<tr>
<td>sITLB, sDTLB</td>
<td>タグ</td>
<td>パリティ</td>
<td>パリティエラー (訂正不可能)</td>
</tr>
<tr>
<td></td>
<td>データ</td>
<td>パリティ</td>
<td>パリティエラー (訂正不可能)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fITLB, fDTLB</td>
<td>ロックビット</td>
<td>三重化</td>
<td>なし。値は多数決で決定される。</td>
</tr>
<tr>
<td></td>
<td>ロックビット以外</td>
<td>パリティ</td>
<td>パリティエラー (二重化しており訂正可能)</td>
</tr>
<tr>
<td></td>
<td>データ</td>
<td>パリティ</td>
<td>パリティエラー (二重化しており訂正可能)</td>
</tr>
</tbody>
</table>

TLBのエラーは、メモリアクセス時のアドレス変換とASIレジスタ経由で直接アクセスする際に発見される。
ASI レジスタ経由でアクセスする際に発見されたエラー

DTLB に ASI_DTLB_DATA_ACCESS または ASI_DTLB_TAG_ACCESS でエラーが発見されると、ASI_UGESR.IUG_DTLB に 1 をセットし、命令緊急エラーを通知する。

ITLB に ASI_ITLB_DATA_ACCESS または ASI_ITLB_TAG_ACCESS でエラーが発見されると、ASI_UGESR.IUG_ITLB に 1 をセットし、命令緊急エラーを通知する。

アドレス変換の際に sTLB で発見されたエラー

アドレス変換の際に sTLB でエラーが発見されると、そのエントリを無効にする。システムソフトウェアには報告されない。

アドレス変換の際に fTLB で発見されたエラー

fTLB はタグもデータも二重化されているので、アドレス変換の際に fTLB でパリティエラーが発見されると、正しいほうからコピーして自動訂正する。システムソフトウェアには報告されない。両方のエントリでパリティエラーが発見されると、致命的エラーとなる。
この章では、SPARC64 IXfx の性能計測カウンタ (Performance Counter:PA) について説明する。

Q.1 PA 概要

PA カウンタの定義については、"Performance Control Register (PCR) (ASR 16)" (page 26) および "Performance Instrumentation Counter (PIC) Register (ASR 17)" (page 28) を参照。

Q.1.1 サンプル擬似コード

カウンタのセットとクリア

PIC は読み書き可能なレジスタである。0 を書くとカウンタがクリアされ、それ以外の値を書けばそれがカウンタにセットされる。以下の擬似コードは、すべての PIC カウンタをクリアする例である。

```c
/* SU/SL の設定を変更せずに PIC をクリアする */
pic_init = 0x0;
pcr = rd_pcr();
pcr.ulro = 0x1; /* 書き込み時に su/sl を変更しない */
pcr.ovf = 0x0; /* オーバーフロービットをクリアする */
pcr.ut = 0x0;
pcr.st = 0x0; /* 正確な計測のためにカウンタを停止する */
for (i=0; i<=pcr.nc; i++) {
    /* 書き込む PIC を選択する */
    pcr.sc = i;
    wr_pcr(pcr);
```

Ver 12, 2 Dec. 2013
計測項目の選択と計測開始

計測項目は PCR.SC と PCR.SU/PCR.SL で選択する。以下のコードは項目の選択と計測開始のサンプルである（特権アクセス可能な場合）

```c
pcr.ut = 0x0; /* ユーザイベント計測を停止 */
pcr.st = 0x0; /* システムイベント計測も停止 */
pcr.ulro = 0x0; /* SU/SLを変更可能にする */
pcr.ovro = 0x1; /* オーバーフロービットは変更しない */
/* 計測せずにイベントを選択する */
for(i=0; i<pcr.nc; i++) {
    pcr.sc = i;
    pcr.sl = select an event;
    pcr.su = select an event;
    wr_pcr(pcr);
}
/* 計測開始 */
pcr.ut = 0x1;
pcr.st = 0x1;
pcr.ulro = 0x1; /* SU/SLビットを変更しない */
/* 必要ならオーバーフロービットをここでクリアする */
wr_pcr(pcr);
```

計測停止と読みだし

以下のコードは計測停止と読みだしのサンプルである（特権アクセス可能な場合）

```c
pcr.ut = 0x0; /* ユーザイベント計測を停止 */
pcr.st = 0x0; /* システムイベント計測も停止 */
pcr.ulro = 0x1; /* SU/SLをリードオンリーにする */
pcr.ovro = 0x0; /* オーバーフロービットは変更しない */
for(i=0; i<pcr.nc; i++) {
    pcr.sc = i;
    wr_pcr(pcr);
    pic = rd_pic();
    picle[i] = pic.picl;
    picu[i] = pic.picu;
}
```
Q.2 PA イベントの説明

PA カウンタは以下のグループに大別される。
1. 命令種類、トラップ種類毎の統計情報
2. MMU と L1 キャッシュ関連のイベント計測
3. L2 キャッシュ関連のイベント計測
4. バストランザクションの計測

PA カウンタで計測できるイベントには公開、準公開の 2 種類がある。
公開イベントは、正確に動作することが検証されており、SPARC64 IXfx の将来の版でも互換性を保証するイベントである。
準公開イベントは、主としてハードウェアのデバッグ用のイベントである。
■ 準公開イベント必ずしも検証されているとは限らないので、このドキュメントで書かれている通りに動作しないかもしれない。
■ 定義は予告なしに変更されることがある。SPARC64 IXfx の将来の版での互換性は保証しない。

SPARC64 IXfx で定義されている全 PA イベントを TABLE Q-1 に示す。網掛けのイベントは準公開イベントである。各イベントの計測内容は次節以降で解説する。特に断りない限り、投機命令による事象も PA イベントの計測対象となる。

1. デザイン変更により該当する機能がなくなった場合はこの限りではない。
<table>
<thead>
<tr>
<th>Encoding</th>
<th>Counter</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000000</td>
<td>cycle_counts</td>
</tr>
<tr>
<td>0000001</td>
<td>instruction_counts</td>
</tr>
<tr>
<td>0000010</td>
<td>instruction_flow_counts</td>
</tr>
<tr>
<td>0000011</td>
<td>lwrc_empty</td>
</tr>
<tr>
<td>0001000</td>
<td>Reserved</td>
</tr>
<tr>
<td>0001010</td>
<td>op_stv_wait</td>
</tr>
<tr>
<td>0001011</td>
<td>effective_instruction_counts</td>
</tr>
<tr>
<td>0010000</td>
<td>load_store_instructions</td>
</tr>
<tr>
<td>0010010</td>
<td>branch_instructions</td>
</tr>
<tr>
<td>0010100</td>
<td>floating_instructions</td>
</tr>
<tr>
<td>0010101</td>
<td>fma_instructions</td>
</tr>
<tr>
<td>0011000</td>
<td>prefetch_instructions</td>
</tr>
<tr>
<td>0011010</td>
<td>Reserved</td>
</tr>
<tr>
<td>0011100</td>
<td>Reserved</td>
</tr>
<tr>
<td>0100000</td>
<td>Reserved</td>
</tr>
<tr>
<td>0100010</td>
<td>rs1</td>
</tr>
<tr>
<td>0100100</td>
<td>Reserved</td>
</tr>
<tr>
<td>0101010</td>
<td>toq_rsbr_phantom</td>
</tr>
<tr>
<td>0101100</td>
<td>trap_all</td>
</tr>
<tr>
<td>0101111</td>
<td>trap_SIMD_load_across_pages</td>
</tr>
<tr>
<td>0110000</td>
<td>Reserved</td>
</tr>
<tr>
<td>0110010</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

TABLE Q-1  PA カウンタのイベント番号と PIC
### TABLE Q-1  
PA カウンタのイベント番号と PIC (Continued)

<table>
<thead>
<tr>
<th>Encoding</th>
<th>Counter</th>
<th>Single_SXAR_Commit</th>
<th>Qid_use</th>
<th>Flush_RS</th>
<th>Reserved</th>
<th>Decall_INTKL</th>
<th>Suspend_Cycle</th>
<th>L1D_miss_dm</th>
<th>L1D_miss_pf</th>
<th>L1D_miss_qpF</th>
</tr>
</thead>
<tbody>
<tr>
<td>0011010</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>L1D_thrashing</td>
<td></td>
<td>3endop</td>
<td></td>
</tr>
<tr>
<td>0011100</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>L1D_miss_dm</td>
<td></td>
<td>3endop</td>
<td></td>
</tr>
<tr>
<td>0011101</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>L1D_miss_pf</td>
<td></td>
<td>3endop</td>
<td></td>
</tr>
<tr>
<td>0100000</td>
<td>uITLB_miss2</td>
<td>uDTLB_miss2</td>
<td>uITLB_miss</td>
<td>uDTLB_miss</td>
<td>L1I_miss</td>
<td>L1D_miss</td>
<td>L1L_wait_all</td>
<td>L1D_wait_all</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0100001</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>L1D_miss_qpF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0100101</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0100111</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Translation:**

<table>
<thead>
<tr>
<th>Encoding</th>
<th>Counter</th>
<th>Single_SXAR_Commit</th>
<th>Qid_use</th>
<th>Flush_RS</th>
<th>Reserved</th>
<th>Decall_INTKL</th>
<th>Suspend_Cycle</th>
<th>L1D_miss_dm</th>
<th>L1D_miss_pf</th>
<th>L1D_miss_qpF</th>
</tr>
</thead>
<tbody>
<tr>
<td>0011010</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>L1D_thrashing</td>
<td></td>
<td>3endop</td>
<td></td>
</tr>
<tr>
<td>0011100</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>L1D_miss_dm</td>
<td></td>
<td>3endop</td>
<td></td>
</tr>
<tr>
<td>0011101</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>L1D_miss_pf</td>
<td></td>
<td>3endop</td>
<td></td>
</tr>
<tr>
<td>0100000</td>
<td>uITLB_miss2</td>
<td>uDTLB_miss2</td>
<td>uITLB_miss</td>
<td>uDTLB_miss</td>
<td>L1I_miss</td>
<td>L1D_miss</td>
<td>L1L_wait_all</td>
<td>L1D_wait_all</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0100001</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>L1D_miss_qpF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0100101</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0100111</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Notes:**

- **Disabled (No PIC is counted up)**
Q.2.1 命令種類、トラップ種類毎の統計情報

公開 PA 項目

1 cycle_counts
CPU サイクル数を計測する。TICK レジスタと同じサイクルを計測するが、
cycle_counts は PCR.UT, PCR.ST の設定によりユーザ、システム個別に計測することができる。

2 instruction_counts (Non-Speculative)
完了した命令数を計測する。この命令数には SXAR1, SXAR2 が含まれる。
SPARC64 IXfx は 1 サイクルで最大 4 命令ずつ实行完了することができるので、
SXAR1, SXAR2 は通常、この完了命令数には含まれない。このため、
instruction_counts / cycle_counts は 4 より大きな値になることがある。

3 effective_instruction_counts (Non-Speculative)
完了した実効命令数を計測する。この命令数には SXAR1, SXAR2 は含まれない。
cycle_counts と組み合わせることで、サイクルあたりの実行命令数 IPC が導出される。

IPC = effective_instruction_counts / cycle_counts
ユーザ、システムそれぞれの effective_instruction_counts と cycle_counts からは、
ユーザ、システム毎の IPC が得られる。

4 load_store_instructions (Non-Speculative)
完了した整数、浮動小数点数のロード、ストア命令数を計測する。アトミック命令もカウントされる。
SIMD load, SIMD store は計測対象ではなく、別のイベントで計測される。

5 branch_instructions (Non-Speculative)
完了した分岐命令数を計測する。CALL, JMPL, RETURN 命令もカウントされる。

6 floating_instructions (Non-Speculative)
完了した浮動小数点命令数を計測する。計測対象は FPop1 (TABLE E-5), FPop2 (TABLE E-6) のほか、IMPDEP1 の opf<8:4> = 1616 または 1716 の命令である。SIMD 演算は計測対象ではなく、別のイベントでカウントされる。
Compatibility Note – SPARC64 VII までの CPU では、このイベントの計測対象はFPop1, FPop2 のみだった。

7 fma_instructions (Non-Speculative)
完了した浮動小数点積和演算命令の命令数を計測する。計測対象命令はFM{ADD, SUB}{s, d}, FNM{ADD, SUB}{s, d}, FTRIMADDd である。SIMD 演算は計測対象ではなく、別のイベントでカウントされる。

Compatibility Note – SPARC64 VII までの CPU では、このイベントはimpdep2_instructions という名前で、計測対象は浮動小数点積和演算のみだった。

計測対象命令の命令当たりの演算数は 2 なので、演算数を求める際は 2 倍する。

8 prefetch_instructions (Non-Speculative)
完了したプリフェッチ命令数を計測する。

9 SIMD_load_store_instructions (Non-Speculative)
完了した浮動小数点数のSIMD ロード、SIMD ストア命令数を計測する。

10 SIMD_floating_instructions (Non-Speculative)
計測対象はfloating_instructions 同じだが、SIMD 実行し完了した命令数を計測する。
計測対象命令のSIMD 実行時の命令当たりの演算数は 2 なので、演算数を求める際は 2 倍する。

11 SIMD_fma_instructions (Non-Speculative)
計測対象はfma_instructions 同じだが、SIMD 実行し完了した命令数を計測する。
計測対象命令のSIMD 実行時の命令当たりの演算数は 4 なので、演算数を求める際は 4 倍する。

12 sxar1_instructions (Non-Speculative)
完了した SXAR1 命令数を計測する。

13 sxar2_instructions (Non-Speculative)
完了した SXAR2 命令数を計測する。
14 trap_all (Non-Speculative)
すべてのトラップの回数を計測する。これは他のトラップイベントの合計に等しい。

15 trap_int_vector (Non-Speculative)
interrupt_vector_trap が通知された回数を計測する。

16 trap_int_level (Non-Speculative)
interrupt_level_n が通知された回数を計測する。

17 trap_spill (Non-Speculative)
spill_n_normal, spill_n_other が通知された回数を計測する。

18 trap_fill (Non-Speculative)
fill_n_normal, fill_n_other が通知された回数を計測する。

19 trap_trap_inst (Non-Speculative)
trap_instruction が通知された回数を計測する。

20 trap_IMMU_miss (Non-Speculative)
fast_instruction_access_MMU_miss が通知された回数を計測する。

21 trap_DMMU_miss (Non-Speculative)
fast_data_instruction_access_MMU_miss が通知された回数を計測する。

22 trap_SIMD_load_across_pages (Non-Speculative)
SIMD_load_across_pages が通知された回数を計測する。

準公開 PA 項目

23 xma_inst (Non-Speculative)
完了した FPMADDX, FPMADDXHI 命令数を計測する。

24 unpack_sxar1 (Non-Speculative)
完了した SXAR1 のうち、パックされなかった命令数を計測する。
25 **unpack_sxar2** (Non-Speculative)
完了した SXAR2 のうち、パックされなかった命令数を計測する。

26 **instruction_flow_counts** (Non-Speculative)
完了した命令数を計測する。SPARC64 IXfx では、いくつかの命令は内部的に複数の命令に分けて実行する。**instruction_flow_counts** が計測するのはこの内部的な命令数である。この命令数にはパックされた SXAR1, SXAR2 は含まれない。

27 **ex_load_instructions** (Non-Speculative)
完了した整数ロード命令の命令数を計測する。対象となる命令は \(LD(S,U)B(A), LD(S,U)H(A), LD(S,U)W(A), LDD(A), LDX(A)\) である。

28 **ex_store_instructions** (Non-Speculative)
完了した整数ストア命令およびアトミック命令の命令数を計測する。対象となる命令は \(STB(A), STH(A), STW(A), STD(A), STX(A), LDSTUB(A), SWAP(A), CAS(X)A\) である。

29 **fl_load_instructions** (Non-Speculative)
完了した浮動小数点ロード命令の命令数を計測する。対象となる命令は \(LDF(A), LDDF(A), LD(X)FSR\) である。
SIMD ロード命令および \(LDQF(A)\) はこのイベントでは計測されない。

30 **fl_store_instructions** (Non-Speculative)
完了した浮動小数点ストア命令の命令数を計測する。対象となる命令は \(STF(A), STDF(A), STFR, STDFR, ST(\times)FSR\) である。
SIMD ストア命令および \(STQF(A)\) はこのイベントでは計測されない。

31 **SIMD_fl_load_instructions** (Non-Speculative)
完了した SIMD 浮動小数点ロード命令の命令数を計測する。対象となる命令は SIMD で実行された \(LDF(A), LDDF(A)\) である。

32 **SIMD_fl_store_instructions** (Non-Speculative)
完了した SIMD 浮動小数点ストア命令の命令数を計測する。対象となる命令は SIMD で実行された \(STF(A), STDF(A), STFR, STDFR\) である。

33 **iwr_empty**
IWR (Issue Word Register) が空であるサイクル数を計測する。IWR はデコード時に命令を保持する 4 エントリのバッファで、IWR が空という状態は、命令フェッチがキャッシュミスによりできないときなどに起こりうる。
以下の要因で命令実行が停止したサイクル数を計測する。

- トラップ、インタラプト
- 特権レジスタの更新
- メモリオーダリング保証
- ハードウェア内部での再実行 (RAS 起因)

分岐予測ミスによるパイプラインフラッシュが起きた回数を計測する。
SPARC64 IXfx は投機実行を行うので、分岐予測ミスにより実際には実行されない命令をパイプラインに投入することがある。このような命令は、分岐方向がはっきりと間違ったパスであることが確定した時点でキャンセルされる。パイプランフラッシュはこのときに起きる。

misprediction rate = flush_rs / branch_instructions

ある CPU サイクルの命令発行数が 0 である回数を計測する。SPARC64 IXfx は最大 4 命令を発行することができるが、発行命令数が 0 のとき、0iid_use が 1 加算される。

SPARC64 IXfx では、いくつかの命令は内部的に複数の命令に分けて実行するが、0iid_use はこの個々の命令に対して計測される。SXAR は計測しない。

ある CPU サイクルの命令発行数が 1 である回数を計測する。計測条件は 0iid_use を参照。

ある CPU サイクルの命令発行数が 2 である回数を計測する。計測条件は 0iid_use を参照。

ある CPU サイクルの命令発行数が 3 である回数を計測する。計測条件は 0iid_use を参照。

ある CPU サイクルの命令発行数が 4 である回数を計測する。計測条件は 0iid_use を参照。
41 **sync_intlk**
パイプラインの **sync** により命令発行ができなかったサイクル数を計測する。

42 **regwin_intlk**
レジスタウィンドウの切り替えにより命令発行ができなかったサイクル数を計測する。

43 **decall_intlk**
命令デコード時のインタロックにより命令発行ができなかったサイクル数を計測する。**decall_intlk** は **sync_intlk** と **regwin_intlk** を含むが、動的に変化する要因（リザーブーションステーションの枯渇など）によるインタロックは計測対象ではない。

44 **rsf_pmmi** (Non-Speculative)
単精度の浮動小数点演算と倍精度の浮動小数点演算が混ざったため、命令発行ができなかったサイクル数を計測する。

45 **toq_rsbr_phantom**
分岐すると予測した命令が実際には分岐命令ではなくなかった回数を計測する。
SPARC64 1Xfx の分岐予測機構は命令デコードより前に分岐予測を行うので、分岐命令かどうかは無関係に分岐するかしないかを予測する。**toq_rsbr_phantom** は、このような間違った命令に対する予測のうち、分岐すると予測したものの回数を計測する。

46 **op_stv_wait** (Non-Speculative)
実行中の命令の中で一番古い命令が、メモリアクセスによるデータ待ちのため、完了命令数が 0 であるサイクル数を計測する。**op_stv_wait** はストア命令によるデータ待ちは計測しない（アトミック命令は計測する）。
**op_stv_wait** はすべてのキャッシュミスレイテンシを計測しているのではないことに注意。先行の未完了命令がある場合、**op_stv_wait** は計測されない。

47 **op_stv_wait_nc_pend** (Non-Speculative)
ノンキャッシュブルアクセスによる **op_stv_wait** を計測する。

48 **op_stv_wait_ex** (Non-Speculative)
整数のメモリアクセス命令による **op_stv_wait** を計測する。L1 キャッシュか L2 キャッシュかは区別しない。
49  *op_stv_wait_sxmiss* (Non-Speculative)

L2 キャッシュミスによる *op_stv_wait* を計測する。整数ロードか浮動小数点ロードかは区別しない。

50  *op_stv_wait_sxmiss_ex* (Non-Speculative)

整数ロードの L2 キャッシュミスによる *op_stv_wait* を計測する。

51  *op_stv_wait_pfp_busy* (Non-Speculative)

プリフェッチ用の空きポートがないため、メモリアクセス命令およびストロングプリフェッチ命令が実行できないことによる *op_stv_wait* を計測する。

52  *op_stv_wait_pfp_busy_ex* (Non-Speculative)

プリフェッチ用の空きポートがないため、整数メモリアクセス命令が実行できないことによる *op_stv_wait* を計測する。

53  *op_stv_wait_swpf* (Non-Speculative)

プリフェッチ用の空きポートがないため、プリフェッチ命令が実行できないことによる *op_stv_wait* を計測する。ストロングプリフェッチ命令の場合、このカウンタだけでなく *op_stv_wait_pfp_busy* でも計測される。

54  *cse_window_empty_sp_full* (Non-Speculative)

CSE が空でストアポートがフルのため、完了命令数が 0 であるサイクル数を計測する。

55  *cse_window_empty* (Non-Speculative)

CSE が空のため、完了命令数が 0 であるサイクル数を計測する。

56  *branch_comp_wait* (Non-Speculative)

実行中の命令の中で一番古い命令が、実行中の分岐命令で、完了命令数が 0 であるサイクル数を計測する。*branch_comp_wait* の計測は *eu_comp_wait* より優先度が低い。

57  *eu_comp_wait* (Non-Speculative)

実行中の命令の中で一番古い命令が、整数または浮動小数点数の演算実行中で、完了命令数が 0 であるサイクル数を計測する。*eu_comp_wait* の計測は *branch_comp_wait* より優先度が高い。
58 $fl\_comp\_wait$ (Non-Speculative)

実行中の命令の中で一番古い命令が、浮動小数点数の演算実行中で、完了命令数が0であるサイクル数を計測する。

59 $0endop$ (Non-Speculative)

完了命令数が0であるサイクル数を計測する。$sxar$ 命令だけがコミットした場合も$0endop$ で計測される。

60 $1endop$ (Non-Speculative)

完了命令数が1であるサイクル数を計測する。

61 $2endop$ (Non-Speculative)

完了命令数が2であるサイクル数を計測する。

62 $3endop$ (Non-Speculative)

完了命令数が3であるサイクル数を計測する。

63 $inh\_cmiit\_gpr\_2write$ (Non-Speculative)

整数レジスタを2個更新しているため、4 命令完了できなかったサイクル数を計測する。

64 $suspend\_cycle$ (Non-Speculative)

$SUSPEND$ 命令、$SLEEP$ 命令により命令制御部が停止しているサイクル数。

65 $sleep\_cycle$ (Non-Speculative)

$SLEEP$ 命令により命令制御部が停止しているサイクル数。

66 $single\_sxar\_commit$ (Non-Speculative)

パックされなかった $sxar$ 命令だけがコミットしたサイクル数を計測する。$0endop$ にも計測されている。

67 $op\_stv\_wait\_swpf\_pfp\_busy$ (Non-Speculative)

プリフェッチ用の空きポートがないため、ストロングプリフェッチ命令が実行できないことによる $op\_stv\_wait$ を計測する。この事象は $op\_stv\_wait\_swpf$ と $op\_stv\_wait\_pfp\_busy$ の両方で計測されている。
Q.2.2 MMU と L1 キャッシュ関連のイベント計測

公開 PA 項目

1. `uITLB_miss`
   命令フェッチの uTLB ミスが起きた回数を計測する。

2. `uDTLB_miss`
   データアクセスの uTLB ミスが起きた回数を計測する。

Note – メイン TLB のミス回数は `trap_IMMU_miss, trap_DMMU_miss` で計測できる。

3. `L1I_miss`
   L1 命令キャッシュのミス回数を計測する。

4. `L1D_miss`
   L1 データキャッシュのミス回数を計測する。

5. `L1I_wait_all`
   L1 命令キャッシュミスの処理にかかる時間 (ミスレイテンシ) を計測する。

   SPARC64 IXfx の L1 命令キャッシュはノンプロッキングキャッシュなので、同時に複数のキャッシュミスを処理することができるが、`L1I_wait_all` が計測するのは一つのミスレイテンシだけである。

6. `L1D_wait_all`
   L1 データキャッシュミスの処理にかかる時間 (ミスレイテンシ) を計測する。

   SPARC64 IXfx の L1 データキャッシュはノンプロッキングキャッシュなので、同時に複数のキャッシュミスを処理することができるが、`L1D_wait_all` が計測するのは一つのミスレイテンシだけである。

準公開 PA 項目

7. `uITLB_miss2`
   命令フェッチの uTLB ミスが起き、fITLB から読み出した回数を計測する。

8. `uDTLB_miss2`
   データアクセスの uTLB ミスが起き、fDTLB から読み出した回数を計測する。
9 \textit{swpf\_success\_all}  
SU でロストせず SX に送られた PREFETCH 命令の数。

10 \textit{swpf\_fail\_all}  
SU でロストした PREFETCH 命令の数。

11 \textit{swpf\_lbs\_hit}  
L1 キャッシュにヒットした PREFETCH 命令の数。  
SU に送られた PREFETCH 命令の総数

12 \textit{L1\_thrashing}  
リードポートが獲得されてから解放されるまでの間に、L2 キャッシュの読み出し要求が 2 回発行された回数。命令フェッチが L1L キャッシュをミスし、そのデータが L1L キャッシュに書き込まれたが、読み出されること前にキャッシュから追い出された場合、このカウンタで計測される。

13 \textit{L1D\_thrashing}  
リードポートが獲得されてから解放されるまでの間に、L2 キャッシュの読み出し要求が 2 回発行された回数。メモリアクセス命令が L1D キャッシュをミスし、そのデータが L1D キャッシュに書き込まれたが、読み出されること前にキャッシュから追い出された場合、このカウンタで計測される。

14 \textit{L1D\_miss\_dm}  
ロード・ストア命令による L1 データキャッシュのミス回数を計測する。

15 \textit{L1D\_miss\_pf}  
プリフェッチ命令による L1 データキャッシュのミス回数を計測する。

16 \textit{L1D\_miss\_qpf}  
ハードウェアプリフェッチによる L1 データキャッシュのミス回数を計測する。

Q.2.3 L2 キャッシュ関連のイベント計測

L2 キャッシュ関連のイベントには、CPU コアの動作により起きる事象と、CPU チップ外部からの要求により起きる事象がある。前者はコア毎に個別に計測され、後者はすべてのコアで計測される。
大部分の L2 キャッシュ関連のイベントには、dm (デマンド), pf (プリフェッチ) の区別がある。しかしかれは、それぞれロード・ストア・アトミック命令とプリフェッチ命令に対応しているわけではない。その理由は、

■ L1 キャッシュにデータを読み込めるための資源が不足していてロード・ストア命令が実行できないとき、先行して L2 キャッシュにデータを読みこんでおき、資源が確保できたら元のロード・ストア命令を行う。この先行読み込みがプリフェッチとして処理されるため。
■ ハードウェアプリフェッチ機構がプリフェッチを生成するため。
■ L1 キャッシュに対するプリフェッチ命令はデマンド要求として処理しようとするため。

したがって、L2 キャッシュ関連の PA イベントのデマンド (dm), プリフェッチ (pf) で計測される事象は以下のようになる。

■ L2 キャッシュをデマンド (dm) でアクセスするのは、命令フェッチ、ロード・ストア・アトミック命令、L1 プリフェッチ命令のうち、メモリアクセスに必要な資源が獲得できたもの。
■ L2 キャッシュをプリフェッチ (pf) でアクセスするのは、命令フェッチ、ロード・ストア命令、L1 プリフェッチ命令のうち、メモリアクセスに必要な資源が獲得できなかったもの、およびハードウェアプリフェッチ。

公開 PA 項目

1 L2_read_dm
デマンド要求の L2 キャッシュ参照回数を計測する。ブロックロード、ブロックストア命令は一命令で8回の参照回数として計測される。
CPUチップ外部からのキャッシュ参照要求は計測されない。

2 L2_read_pf
プリフェッチ要求の L2 キャッシュ参照回数を計測する。ブロックロード、ブロックストア命令は一命令で8回の参照回数として計測される。

3 L2_miss_dm
デマンド要求の L2 キャッシュミス回数を計測する。
このカウンタは、L2_miss_count_dm_bank{0,1,2,3} の合計を計測する。

4 L2_miss_pf
プリフェッチ要求の L2 キャッシュミス回数を計測する。
このカウンタは、L2_miss_count_pf_bank{0,1,2,3} の合計を計測する。
5  L2_miss_count_dm_bank{0,1,2,3}
デマンド要求が L2 キャッシュをミスした回数を計測する。L2 キャッシュのバンク毎に個別に計測する。

Note – あるアドレスに対して先行してプリフェッチが出され、L2 キャッシュをミスしてメモリアクセス要求を発行し、そのデータが戻ってくる前にデマンド要求がきた場合、後続のデマンド要求はミス回数には計上されない。

6  L2_miss_count_pf_bank{0,1,2,3}
プリフェッチ要求が L2 キャッシュの各バンクをミスした回数をバンク毎に計測する。

7  L2_miss_wait_dm_bank{0,1,2,3}
デマンド要求が L2 キャッシュをミスしたときの処理にかかる時間（ミスレイテンシ）を、バンク毎に計測する。計測は個々のメモリアクセス要求について行われる。

Note – あるアドレスに対して先行してプリフェッチが出され、L2 キャッシュをミスしてメモリアクセス要求を発行し、そのデータが戻ってくる前にデマンド要求がきた場合、それ以降データが戻ってくるまでのサイクル数は、L2_miss_wait_dm_bank{0,1,2,3} で計測される。

8  L2_wb_dm
デマンド要求の L2 キャッシュミスによるライトバック回数を計測する。

9  L2_wb_pf
プリフェッチ要求の L2 キャッシュミスによるライトバック回数を計測する。

準公開 PA 項目

10 lost_pf_pfp_full
PF ボートフルによりプリフェッチ要求がロストした回数を計測する。

11 lost_pf_by_abort
SX パイプラインアボートによりプリフェッチ要求がロストした回数を計測する。
Q.2.4 バストランザクションの計測

公開 PA 項目

1. **cpu_mem_read_count**
   CPU からの要求により、メモリ読み出し要求を発行した回数を計測する。

2. **cpu_mem_write_count**
   CPU からの要求により、メモリ書き込み要求を発行した回数を計測する。

3. **IO_mem_read_count**
   I/O からの要求により、メモリ読み出し要求を発行した回数を計測する。

4. **IO_mem_write_count**
   I/O からの要求により、メモリ書き込み要求を発行した回数を計測する。
   この項目で計測するのは、ICC-FST のみ。ICC-PST は IO_pst_count で計測できる。

5. **bi_count**
   CPU チップが外部からキャッシュ無効化要求により、キャッシュ上のデータを無効化した回数を計測する。
   この項目は、すべてのコアで同じ値が計測される。

Compatibility Note – SPARC64 VIIIfx では、無効化要求を受けた回数を計測していた。SPARC64 IXfx では無効化要求を受けて、キャッシュ上にデータがあった場合に計測する。

6. **cpi_count**
   CPU チップが外部からキャッシュ書き出しと無効化要求を受けた回数を計測する。
   計測対象は、キャッシュ上のデータが更新されていればメモリに書き出して無効化し、キャッシュ上のデータとメモリの内容が一致していれば無効化する要求である。
   この項目は、すべてのコアで同じ値が計測される。

Implementation Note – SPARC64 IXfx ではこの PA イベントは存在しない。項目は互換性のために残してある。
7 cpb_count

CPU チップが外部からキャッシュ書き出しと要求を受けた回数を計測する。計測対象は、キャッシュ上の更新されているデータをメモリに書き出す要求である。この項目は、すべてのコアで同じ値が計測される。

Implementation Note – SPARC64 IXfx ではこの PA イベントは存在しない。項目は互換性のために残している。

8 cpd_count

CPU チップが外部からキャッシュ内容の読み出し要求を受け、データを読み出しした回数を計測する。計測対象は、DMA による読み出しなど、キャッシュ上の更新されているデータを読み出し、メモリに書き出さない要求である。この項目は、すべてのコアで同じ値が計測される。

Compatibility Note – SPARC64 VIIIfx では、読み出し要求を受けた回数を計測していた。SPARC64 IXfx では読み出し要求を受けて、キャッシュ上にデータがあった場合に計測する。

9 IO_pst_count

I/O からの要求により、メモリ書き込み要求 (ICC-PST) を発行した回数を計測する。

Q.3 サイクルアカウンティング

サイクルアカウンティングとは、性能ボトルネック要因分析の手法である。ある命令列を実行するためにかかった総時間 (CPU サイクル数) を、CPU の動作状態（命令実行中、メモリアクセス待ちである、演算完了待ちである、など）で分類し、命令列を実行した時、CPU 内のどの部分にボトルネックがあるかを把握することで、性能分析や改善を行うことができる。SPARC64 IXfx は豊富な PA イベントを備えており、CPU の動作状態に関する詳細な情報が取得できるので、詳細なサイクルアカウンティングが作成でき、効率的なボトルネック分析や性能向上チューニングに役立てることができる。
SPARC64™ IXfx は複数の演算器を持ち、アウトオブオーダで実行する CPU である。このため、ある命令がメモリアルからのデータ待ち状態であっても、別の命令は浮動小数点乗算の実行中で、さらに別の命令は分岐方向の確定待ちにあるなど、実行中と実行待ちの命令が複数入り乱れた状態にある。このような状態で個々の命令の待ち要因を分析するのは意味がないので、インオーダで行われる命令完了に着目し、計測期間の総サイクル数を、1 サイクルあたりの命令完了数で大きく分類し、命令完了がないサイクルについてはさらにその要因により細分類したものをサイクルアカウンティングと呼んでいる。

SPARC64 IXfx は 1 サイクルに最大 4 命令まで完了することができるので、4 命令完了しているサイクル数の割合が高いほど実行効率が高いことになる。命令完了できなかったサイクルは、性能に与える悪影響が極めて大きいので、詳細に分析する必要がある。主な原因として、
- メモリアクセスのデータが帰ってくるのを待っている
- 演算の完了を待っている
- 命令フェッチが間に合わず、パイプラインに命令が供給されていない

が考えられる。サイクルアカウンティングに使える PA イベントの一覧とその計算方法を TABLE Q-2 に示す。

また、FIGURE Q-1 はメモリアクセス待ちの PA イベント群 op_stv_wait_* 間の関係を図示したものである。なお、表や図中で † がつくた PA イベントは、計算によって求められる架空の PA イベントである。

![FIGURE Q-1 op_stv_wait の内訳](image-url)
### TABLE Q-2 サイクルアカウンティングに有用な PA イベント

<table>
<thead>
<tr>
<th>CPU サイクルあたりのコミット数</th>
<th>総 CPU サイクル数</th>
<th>備考</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>cycle_counts</td>
<td>N/A（最大コミット数に達している）</td>
</tr>
<tr>
<td></td>
<td>- 3endop - 2endop</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- 1endop - 0endop</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3endop</td>
<td>次の命令をコミットできなかった要因のひとつは inh_cmit_gpr_2write で計測される。</td>
</tr>
<tr>
<td>2</td>
<td>2endop</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1endop</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>命令実行待ち:</td>
<td>eu_comp_wait = ex_comp_wait + fl_comp_wait</td>
</tr>
<tr>
<td></td>
<td>eu_comp_wait</td>
<td></td>
</tr>
<tr>
<td></td>
<td>+ branch_comp_wait</td>
<td></td>
</tr>
<tr>
<td>命令フェッチ待ち:</td>
<td>cse_window_empty = cse_window_empty_sp_full + sleep_state + misc.†</td>
<td></td>
</tr>
<tr>
<td></td>
<td>cse_window_empty</td>
<td></td>
</tr>
<tr>
<td>L1D キャッシュアクセス待ち:</td>
<td>op_stv_wait</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- L2 キャッシュミス（下記参照）</td>
<td></td>
</tr>
<tr>
<td>L2 キャッシュミス:</td>
<td>op_stv_wait_sxmiss + op_stv_wait_nc_pend</td>
<td></td>
</tr>
<tr>
<td>その他:</td>
<td>0endop</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- op_stv_wait</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- cse_window_empty</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- eu_comp_wait</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- branch_comp_wait</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(instruction_flow_counts - instruction_counts)</td>
<td></td>
</tr>
</tbody>
</table>
この章では、これまでの章で触れられていない CPU 内のコンポーネントにあるレジスタについて定義する。
なお、サービスプロセッサから CPU を制御するための仕様は本仕様書の範囲外である。

R.1 System Config Register

<table>
<thead>
<tr>
<th>レジスタ名</th>
<th>ASI_SYS_CONFIG</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASI</td>
<td>4A16</td>
</tr>
<tr>
<td>VA</td>
<td>—</td>
</tr>
</tbody>
</table>

アクセス種別 | Supervisor read/write( 書き込みは無視される )

<table>
<thead>
<tr>
<th>ビット</th>
<th>フィールド名</th>
<th>アクセス</th>
<th>説明</th>
</tr>
</thead>
<tbody>
<tr>
<td>63:10</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>9:0</td>
<td>ITID</td>
<td>R</td>
<td>スレッドの ITID (Interrupt Target ID).</td>
</tr>
</tbody>
</table>

Reserved | ITID
---|---
63 | 10 9 0
R.2 STICK Control Register

レジスタ名   ASI_STICK_CNTL  
ASI         4B16  
VA          0016  
アクセス種別 Supervisor read/write

<table>
<thead>
<tr>
<th>ビット</th>
<th>フィールド名</th>
<th>アクセス</th>
<th>説明</th>
</tr>
</thead>
<tbody>
<tr>
<td>63:1</td>
<td>—</td>
<td>RW</td>
<td>stop</td>
</tr>
</tbody>
</table>

```
stop に 1 を設定すると STICK のカウントアップは停止し、stop に 0 を設定すると STICK のカウントアップが再開する。
```

STICK_CNTL は、STICK のカウントアップを有効/無効を制御するレジスタである。STICK_CNTL は全コアで共有されており、どのコアから操作しても、全コアの STICK に対して同時に作用する。

STICK.stop = 1 の間は STICK のカウントアップが停止している。その影響は以下の通り。

- STICK_CMPR をセットしていてもタイマ割り込みがつかならない。
  ただし、STICK_CNTL.stop = 1 の状態で
  - STICK_CMPR.INT_DIS = 0
  - STICK_CMPR.STICK_CMPR = STICK.counter

と設定した場合は、SOFTINT.SM = 1 がセットされる。このとき PSTATE.IE = 1 かつ PIL < 14 ならば、レベル 14 の割り込みがかかる。

- SLEEP 命令を実行してスリープ中のコアは、命令実行状態に復帰しない。

複数のコアがほぼ同時に STICK_CNTL を読み書きした場合、それらの要求はひとつずつ処理される。処理順序はハードウェアの実装に依存する。

Programming Note – STICK_CNTL の操作はあるひとつのコアから行うこと。
STICK_CNTL の書き込み後の STICK の書き込み/読み出しは、STICK_CNTL の書き込み処理の完了を待って FLUSH 命令を実行してから行うこと。完了までの時間は未定義。完了確認は、STICK_CNTL への書き込みを行ったコアから STICK_CNTL を読み出すことで行える。STICK_CNTL の書き込みが完了する前に STICK を書き込み/読み出した場合、STICK の値は保証されない。
Summary of Specification Differences

この章では、SPARC V9, SPARC JPS1, SPARC64 VII, SPARC64 VIIIfx 仕様と SPARC64 IXfx 仕様の相違点を総括する。この章は読者の便宜を図る目的で書かれており、論理仕様の定義ではない。正確な定義は各項目を参照されたい。

TABLE S-1 に、SPARC V9, SPARC JPS1, SPARC64 VII, SPARC64 VIIIIfx 各仕様と SPARC64 IXfx 仕様との相違点をまとめた。前方互換性の欄は、SPARC V9, SPARC JPS1, SPARC64 VII 各仕様に準拠したソフトウェアが、SPARC64 IXfx 仕様の CPU で動作するかどうかを示している。なお、SPARC64 VIIIIfx 仕様に準拠したソフトウェアは、バリアを別にすれば SPARC64 IXfx 上でも無変更で動作するので、互換性の欄は設けていない。バリアの非互換仕様については Appendix L.4, “ハードウェアバリア” (page 222) を参照。

1. SPARC V9, SPARC JPS1, SPARC64 VII 各仕様で Reserved の部分を使っているソフトウェアは動作非互換になるが、TABLE S-1 は Reserved については言及しない。
### TABLE S-1 仕様差分サマリ (1 of 4)

<table>
<thead>
<tr>
<th>項目</th>
<th>仕様</th>
<th>前方互換性</th>
<th>ページ</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>V9</td>
<td>JPS1</td>
<td>SPARC64 VII</td>
</tr>
<tr>
<td>アーキテクチャ</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>コア、スレッド</td>
<td>未定義</td>
<td>4 コア、コア当たり 2スレッド</td>
<td>16 コア、コア当たり 1スレッド</td>
</tr>
<tr>
<td>整数レジスタ</td>
<td>160 本</td>
<td>192 本</td>
<td>20</td>
</tr>
<tr>
<td>浮動小数点レジスタ</td>
<td>倍精度 32 本</td>
<td>倍精度 256 本</td>
<td>20</td>
</tr>
<tr>
<td>ASR</td>
<td>未定義</td>
<td>$pcr, ppic, pcdr, psgr, $softint, $tick_cmpr, $sys_tick, $sys_tick_cmpr, $xar, $xasr, $txar</td>
<td>26</td>
</tr>
<tr>
<td>物理アドレス</td>
<td>未定義</td>
<td>43 ビット以上</td>
<td>41 ビット</td>
</tr>
<tr>
<td>RSTVaddr</td>
<td>実装依存</td>
<td>PA = 7fffffff00000000_{16}</td>
<td>PA = 1fffffff00000000_{16}</td>
</tr>
<tr>
<td>キャッシュ</td>
<td>未定義</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SXflush</td>
<td>未定義</td>
<td>あり</td>
<td>なし</td>
</tr>
<tr>
<td>TLB</td>
<td>未定義</td>
<td>32(TLB)+2x1024/2way(sTLB), 1D とも。</td>
<td>I: 16 (ITLB) 2x128/2way (sITLB) D: 16 (DTLB) 2x256/2way (sDTLB) victim cache の機能なし。エラー注入機能削除。</td>
</tr>
<tr>
<td>ベージサイズ</td>
<td>未定義</td>
<td>8KB, 64KB, 512KB, 4MB</td>
<td>8KB, 64KB, 512KB, 4MB, 32MB, 256MB, 2GB</td>
</tr>
</tbody>
</table>

330 SPARC64™ IXfx Extensions Ver 12, 2 Dec. 2013
<table>
<thead>
<tr>
<th>項目</th>
<th>仕様</th>
<th>前方互換性</th>
<th>ページ</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSB</td>
<td>未定義</td>
<td>なし</td>
<td>222</td>
</tr>
<tr>
<td>プリヤ</td>
<td>未定義</td>
<td>なし</td>
<td>239</td>
</tr>
<tr>
<td>ハードウェアプリヤフェッチ</td>
<td>未定義</td>
<td>なし</td>
<td>244</td>
</tr>
<tr>
<td>インタラプストレジスタ</td>
<td>未定義</td>
<td>なし</td>
<td></td>
</tr>
<tr>
<td>命令</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>impdep1</td>
<td>未定義</td>
<td>VIS</td>
<td>76, 77, 77, 114, 116, 118, 123</td>
</tr>
<tr>
<td>impdep2</td>
<td>未定義</td>
<td>F[N]MADD, SUB(s,d), FPMADDX{HI}</td>
<td>70, 78, 122</td>
</tr>
<tr>
<td>load/store</td>
<td>QUAD_LDD_PHYS</td>
<td>QUAD_LDD_PHYS, ST[D]FR, XFILL</td>
<td>88, 122, 133</td>
</tr>
<tr>
<td>その他</td>
<td>POPC</td>
<td>POPC, SXAR</td>
<td>94, 131</td>
</tr>
<tr>
<td>SIMD</td>
<td>なし</td>
<td>あり</td>
<td></td>
</tr>
<tr>
<td>prefetch fcn</td>
<td>0-4  , 0-20-23</td>
<td>0-0-3, 20-23</td>
<td>95</td>
</tr>
</tbody>
</table>
### TABLE 5-1
仕様差分サマリ（3 of 4）

<table>
<thead>
<tr>
<th>項目</th>
<th>仕様</th>
<th>前方互換性</th>
<th>ページ</th>
</tr>
</thead>
<tbody>
<tr>
<td>V9</td>
<td>JPS1</td>
<td>SPARC64 VII</td>
<td>SPARC64 VIIfx</td>
</tr>
<tr>
<td>block load, block store 動作</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• bst commit はキャッシュ上のデータを無効化し、メモリに書く。</td>
<td>• bst commit はキャッシュ上のデータを無効化し、メモリに書く。</td>
<td>• bst commit はキャッシュにストアする。</td>
<td>非互換</td>
</tr>
<tr>
<td>• レジスタ依存は検出する。</td>
<td>• メモリモデルは、bld/bst 内部は RMO, 前後命令とは V9 非準拠。</td>
<td>• レジスタ依存は検出する。</td>
<td></td>
</tr>
<tr>
<td>• bst commit はキャッシュ上のデータを無効化し、メモリに書く。</td>
<td>• bld/bst 中に TTE が無効化されると fast_data_access_MU_miss 発生。</td>
<td>• bst commit はキャッシュを無効化し、メモリに書く。</td>
<td></td>
</tr>
<tr>
<td>LDDF/STDF_mem _address_not_aligned</td>
<td>impdep. #109, #110</td>
<td>例外通知する。</td>
<td>non-SIMD では例外通知する。</td>
</tr>
<tr>
<td>命令属性</td>
<td>なし</td>
<td>SIMD, セクタキャッシュおよびハードウェアリフェッチ無効が指定できる。</td>
<td></td>
</tr>
<tr>
<td>優先順位</td>
<td>async_data_error</td>
<td>async_data_error, illegal_action, SIMD_load_across_pages</td>
<td></td>
</tr>
<tr>
<td>優先順位</td>
<td>async_data_error は 2, illegal_action が 8.5, SIMD_load_across_pages が 12, fp_exception_other (ft = unimplemented_FPPop) が 8.2, SIMD で fp_exception_ieee754 と fp_exception_other (ft = unfinished_FPPop) が同時に起きたときは、fp_exception_other が優先される。</td>
<td></td>
<td></td>
</tr>
<tr>
<td>レジスタセーブ</td>
<td>async_data_error が 2, illegal_action が 8.5, SIMD_load_across_pages が 12, fp_exception_other (ft = unimplemented_FPPop) が 8.2, SIMD で fp_exception_ieee754 と fp_exception_other (ft = unfinished_FPPop) が同時に起きたときは、fp_exception_other が優先される。</td>
<td></td>
<td></td>
</tr>
<tr>
<td>追加レジスタに関して、</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• トラブル時 TXAR[TL] ← XAR XAR ← 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• DONE/RETRY 時 XAR ← TXAR[TL] TXAR[TL] は変化しない</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

レジスタ機能
### TABLE S-1 仕様差分サマリ (4 of 4)

<table>
<thead>
<tr>
<th>項目</th>
<th>仕様</th>
<th>前方互換性</th>
<th>ページ</th>
</tr>
</thead>
<tbody>
<tr>
<td>%ver.imp</td>
<td>V9 7 SPARC64 VII</td>
<td>8 SPARC64 VIIIfx</td>
<td>9 SPARC64 IXfx</td>
</tr>
<tr>
<td>%fsr.cex</td>
<td>更新</td>
<td>高々 1 ビットがセットされる。</td>
<td>SIMD 演算では 2 ビットセットされることがある。</td>
</tr>
<tr>
<td>PAイベント</td>
<td>種類</td>
<td>6 ビット</td>
<td>7 ビット</td>
</tr>
<tr>
<td>watchpoint</td>
<td>VA, PA 独立に指定可能。</td>
<td>VA, PA でレジスタ共有。</td>
<td>非互換 36</td>
</tr>
<tr>
<td>AFAR</td>
<td>オプション</td>
<td>0 固定で読み出し可。</td>
<td>削除。</td>
</tr>
<tr>
<td>EIDR</td>
<td>bit&lt;13:0&gt; が有効。</td>
<td>bit&lt;2:0&gt; が有効。</td>
<td>bit&lt;3:0&gt; が有効。</td>
</tr>
<tr>
<td></td>
<td>bit&lt;13:12&gt; にはソフトウェアが 1002 をセットする。エラーマーク ID に使われる。</td>
<td>bits &lt;13:12&gt; はハード固定で 1002。</td>
<td>bits &lt;13:12&gt; はハード固定で 1002。</td>
</tr>
<tr>
<td>SYS_CONFIG</td>
<td>JB_CONFIG_REGIS</td>
<td>SYS_CONFIG</td>
<td>ITID のみ定義されている。</td>
</tr>
<tr>
<td></td>
<td>TER, UC_S, UC_SW, CLK_MODE, ITID が定義されている。</td>
<td>ITID のみ定義されている。</td>
<td></td>
</tr>
<tr>
<td>SCCR</td>
<td>なし</td>
<td>L1: 2bit</td>
<td>L2: 4bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>L1: 2bit</td>
<td>L2: 5bit</td>
</tr>
<tr>
<td>その他</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>致命的エラーの要因表示</td>
<td>STCHG_ERROR_INF</td>
<td>致命的エラー要因は表示されない。</td>
<td>非互換 274</td>
</tr>
<tr>
<td>STICK start/stop</td>
<td>なし (SC で制御)。</td>
<td>あり。</td>
<td>非互換 326</td>
</tr>
</tbody>
</table>

1. SXAR は V8 仕様と非互換。
Index

A
A_UGE
  specification of, 260
address mask (AM) field of PSTATE register, 68, 80
address space identifier (ASI)
  complete list, 214
  load floating-point instructions, 82, 101
ADE
  async_data_error を参照
ASI
  Bypass, 214
  Nontranslating, 214
  Translating, 214
ASI_AFAR, 216
ASI_AFSR, 216
ASI_AFSR, see ASI_ASYNC_FAULT_STATUS
ASI_AIUP, 215
ASI_AIUPL, 215
ASI_AIUS, 215
ASI_AIUSL, 215
ASI_AS_IF_USER_PRIMARY, 215
ASI_AS_IF_USER_PRIMARY_LITTLE, 215
ASI_AS_IF_USER_SECONDARY, 215
ASI_AS_IF_USER_SECONDARY_LITTLE, 215
ASI_ASYNC_FAULT_ADDR, 216
ASI_ASYNC_FAULT_STATUS, 216, 263, 287, 287,
  293
ASI_ATOMIC_QUAD_LDD_PHYS, 88, 203, 216
ASI_ATOMIC_QUAD_LDD_PHYS_LITTLE, 88, 203,
  216
ASI_BLOCK_COMM (ASI_BLK_COMMIT_P, 219
ASI_BLK_COMMIT_S, 219
ASI_BLK_P, 219
ASI_BLK_PL, 220
ASI_BLK_S, 219
ASI_BLK_SL, 220
ASI_BLOCK AS_IF USER PRIMARY, 218
ASI_BLOCK AS_IF_USER_PRIMARY_LITTLE, 218
ASI_BLOCK AS_IF_USER_SECONDARY, 218
ASI_BLOCK AS_IF_USER_SECONDARY_LITTLE, 218
ASI_BLOCK_COMMIT_PRIMARY, 219
ASI_BLOCK_COMMIT_SECONDARY, 219
ASI_BLOCK_PRIMARY, 219
ASI_BLOCK_PRIMARY_LITTLE, 220
ASI_BLOCK_SECONDARY, 219
ASI_BLOCK_SECONDARY_LITTLE, 220
ASI_BST, 219
ASI_BST_BIT, 217
ASI_BST_BIT, 294
ASI_CACHE_INV, 218
ASI_DCUCR, 216, 250
ASI_DMMU_DEMAP, 217
ASI_DMMU_PA_WATCHPOINT_REG, 217
ASI_DMMU_SFAR, 217, 263
ASI_DMMU_SFPAR, 217
ASI_DMMU_SFSR, 217, 263
ASI_DMMU_TAG_ACCESS, 217, 278
ASI_DMMU_TAG_ACCESS_EXT, 217
ASI_DMMU_TAG_TARGET, 278
ASI_DMMU_TAG_TARGET_REG, 217
ASI_DMMU_TSB_64KB_PTR_REG, 217
ASI_DMMU_TSB_8KB_PTR_REG, 217
ASI_DMMU_TSB_BASE, 217, 278
ASI_DMMU_TSB_DIRECT_PTR_REG, 217
ASI_DMMU_TSB_NEXT_REG, 217
ASI_DMMU_TSB_PEXT_REG, 217
ASI_DMMU_TSB_SEXT_REG, 217
ASI_DMMU_VA_WATCHPOINT_REG, 217
ASI_DMMU_WATCHPOINT_REG, 217
ASI_DTLB_DATA_ACCESS, 301
ASI_DTLB_DATA_ACCESS_REG, 217
ASI_DTLB_DATA_IN_REG, 217
ASI_DTLB_TAG_ACCESS, 301
ASI_DTLB_TAG_READ_REG, 217
ASI_ECR, 216, 272
ASI_EIDR, 217, 263, 272, 275, 278, 294
ASI_ERROR_CONTROL, 216, 263, 272, 273
ASI_INTR_DATA1_W, 218
ASI_INTR_DATA2_R, 218
ASI_INTR_DATA2_W, 218
ASI_INTR_DATA3_R, 218
ASI_INTR_DATA3_W, 218
ASI_INTR_DATA4_R, 218
ASI_INTR_DATA4_W, 218
ASI_INTR_DATA5_R, 218
ASI_INTR_DATA5_W, 218
ASI_INTR_DATA6_R, 218
ASI_INTR_DATA6_W, 218
ASI_INTR_DATA7_R, 218
ASI_INTR_DATA7_W, 218
ASI_INTR_DISPATCH_STATUS, 242
ASI_INTR_DISPATCH_W, 278
ASI_INTR_R, 243, 278
ASI_INTR_RECEIVE, 216, 243
ASI_INTR_W, 241, 242, 243
ASI_ITLB_DATA_ACCESS_REG, 217
ASI_ITLB_DATA_IN_REG, 217
ASI_ITLB_TAG_ACCESS, 278
ASI_ITLB_TAG_READ_REG, 217
 Asi_L2_CTRL, 184, 188, 189, 191, 202, 224, 226, 227, 228, 235, 236, 237
ASI_LBSY, 219
ASI_MCNTL, 183, 216
ASI_MEMORY_CONTROL_REG, 216
ASI_MONDO_RECEIVE_CTRL, 216
ASI_MONDO_SEND_CTRL, 216
ASI_N, 215
ASI_NL, 215
ASI_NUCLEUS, 95, 195, 215
ASI_NUCLEUS_LITTLE, 95, 215
ASI_NUCLEUS_QUAD_LDD_L, 216
ASI_NUCLEUS_QUAD_LDD_LITTLE, 216
ASI_P, 218
ASI_PA_WATCHPOINT, 275
ASI_PHYS_BYPASS_EC_WITH_E_BIT, 233
ASI_PHYS_BYPASS_EC_WITH_E_BIT_LITTLE, 23
ASI_PHYS_BYPASS_EC_WITH_EBIT, 215
ASI_PHYS_BYPASS_EC_WITH_EBIT_L, 216
ASI_PHYS_BYPASS_EC_WITH_EBIT_LITTLE, 216
ASI_PHYS_BYPASS_WITH_EBIT, 40
ASI_PHYS_USE_EC, 215
ASI_PHYS_USE_EC_L, 215
ASI_PHYS_USE_EC_LITTLE, 215
ASI_PL, 218
ASI_PNF, 218
ASI_PNFL, 219
|ASI PRIMARY| 95, 195, 198, 218|
|ASI PRIMARY AS IF USER| 95|
|ASI PRIMARY AS IF USER LITTLE| 95|
|ASI PRIMARY CONTEXT| 278|
|ASI PRIMARY CONTEXT REG| 217|
|ASI PRIMARY LITTLE| 95, 218|
|ASI PRIMARY NO FAULT| 218|
|ASI PRIMARY NO FAULT LITTLE| 219|
|ASI PST16 P| 219|
|ASI PST16 PL| 219|
|ASI PST16 PRIMARY| 219|
|ASI PST16 PRIMARY LITTLE| 219|
|ASI PST16 S| 219|
|ASI PST16 SECONDARY| 219|
|ASI PST16 SECONDARY LITTLE| 219|
|ASI PST32 P| 219|
|ASI PST32 PL| 219|
|ASI PST32 PRIMARY| 219|
|ASI PST32 PRIMARY LITTLE| 219|
|ASI PST32 S| 219|
|ASI PST32 SECONDARY| 219|
|ASI PST32 SECONDARY LITTLE| 219|
|ASI PST8 P| 219|
|ASI PST8 PL| 219|
|ASI PST8 PRIMARY| 219|
|ASI PST8 PRIMARY LITTLE| 219|
|ASI PST8 S| 219|
|ASI PST8 SECONDARY| 219|
|ASI PST8 SECONDARY LITTLE| 219|
|ASI PST8 SL| 219|
|ASI XFILL P| 218, 219|
|ASI XFILL S| 218, 220|

**async data error** exception, 45, 51, 57, 58, 84, 149, 150, 154, 260, 261, 276, 277, 280

**atomic**
- load quadword, 88

**B**
- BA instruction, 167
- BCC instruction, 167
- BCS instruction, 167
- BE instruction, 167
- BG instruction, 167
- BGE instruction, 167
- BGU instruction, 167
- Bicc instructions, 161, 166
- BL instruction, 167
- BLE instruction, 167
- BLEU instruction, 167

**block**
- block store with commit, 221
- load instructions, 221
- store instructions, 221

- BN instruction, 167
- BNE instruction, 167
- BNEG instruction, 167
- BP instructions, 168
- BPA instruction, 167
- BPE instruction, 166
- BPG instruction, 167
- BPGE instruction, 167
- BPGU instruction, 167
- BPL instruction, 166
- BPLE instruction, 166
- BPLEU instruction, 167
- BPN instruction, 166
BPNE instruction, 167
BPNEG instruction, 167
BPOS instruction, 167
BPOPS instruction, 167
BPPr instructions, 166
BPVC instruction, 167
BPVS instruction, 167
branch history buffer, 13
BRHIS, see branch history buffer, 13
BVC instruction, 167
BVS instruction, 167

C
cache
data
cache characteristics, 233
synchronizing, 54
CALL instruction, 38
CARESTORE register, 278
CANSAVE register, 278
CASA instruction, 39, 45, 199
CASXA instruction, 39, 45, 199
cc0 field of instructions, 168
cc1 field of instructions, 168
cc2 field of instructions, 168
CE
in L1D cache data, 296
clean windows (CLEANWIN) register, 108
CLEANWIN register, 153, 278
CLEAR_SOFTINT register, 291
clock-tick register (TICK), 108
cmask field, 91
Commit Stack Entry, 15
Context field of TTE, 175
current exception (cexc) field of FSR register, 23
current window pointer (CWP) register
writing CWP with WRPR instruction, 108
CWP register, 153, 278

D
d superscript on instruction name, 58
dae
error detection action, 273
reporting, 260
data_access_error exception, 84, 89, 103, 106, 130, 179, 198, 200, 261
data_access_exception exception, 84, 87, 103, 106,
E

E bit of PTE, 40

ECC_error exception, 57, 288
ee_second_watch_dog_timeout, 276
ee_sir_in_maxtl, 276
ee_trap_addr_uncorrected_error, 275
ee_trap_in_maxtl, 276
ee_watch_dog_timeout_in_maxtl, 276
enable floating-point (FEF) field of FPRS register, 82, 86, 101, 105, 129
enable floating-point (PEF) field of PSTATE register, 82, 86, 101, 105, 129
error
  handling
    ASI errors, 293
    ASR errors, 290
  restrainable, 261
  uncorrectable
    without direct damage, 261
urgent, 259
ERROR_CONTROL register, 293
ERROR_MARK_ID, 270
error_state, 150, 248, 250
exceptions
  async_data_error, 84
  catastrophic, 45
  data_access_error, 84, 89, 103, 106, 130
  data_access_exception, 84, 87, 103, 106, 130
  data_access_protection, 89
  data_breakpoint, 150
fp_disabled, 84, 87, 102, 106, 130
fp_exception_ieee_754, 75, 143, 144
fp_exception_other, 140, 157
illegal_instruction, 75, 84, 93, 102, 107, 147, 149
LDDF_mem_address_not_aligned, 84, 87, 157, 221
mem_address_not_aligned, 84, 87, 102, 106, 130, 157, 221
privileged_action, 87, 98, 106, 157
privileged_opcode, 110
STDF_mem_address_not_aligned, 102, 106
trap_instruction, 107
unfinished_FPop, 140, 144
execute_state, 250

F

FABSd instruction, 164, 165
FABSq instruction, 164, 165

fast_data_access_MMU_miss exception, 178, 200
fast_data_access_protection exception, 178, 200
fast_data_instruction_access_MMU_miss exception, 310
fast_instruction_access_MMU_miss exception, 57, 178, 196, 197, 310
Fatal error, 265, 267
FBA instruction, 167
FBE instruction, 167
FBfc instructions, 161, 166
FBG instruction, 167
FBGE instruction, 167
FBGE instruction, 167
FBL instruction, 167
FBLE instruction, 167
FBLG instruction, 167
FBN instruction, 167
FBNE instruction, 167
FDO instruction, 167
FBPA instruction, 167
FBPE instruction, 167
FBPfc instructions, 161, 166, 169
FBPG instruction, 167
FBPGE instruction, 167
FBPL instruction, 167
FBPLE instruction, 167
FBPLG instruction, 166
FBPN instruction, 166
FBPNE instruction, 166
FBPO instruction, 167
FBPU instruction, 167
FBPUE instruction, 167
FBPUG instruction, 167
FBPUGE instruction, 167
FBPU instruction, 166
FBPUL instruction, 166
FBPULE instruction, 167
FBU instruction, 167
FBUG instruction, 167
FBUGE instruction, 167
FBU instruction, 167
FBU instruction, 167
FCMP instructions, 169
FCMPd instruction, 165
FCMPE instructions, 169
FCMPed instruction, 165
FCMPEq instruction, 165
FCMPEs instruction, 165
FCMPq instruction, 165
FCMPs instruction, 165


fDTLB, 173
FdTOx instruction, 164, 165
fill_n_normal exception, 310
fill_n_other exception, 310
fITLB, 155, 173, 179
floating-point
defered-trap queue (FQ), 38
trap types
fp_disabled, 67, 75, 93
floating-point trap type (fit) field of FSR register, 101
FLUSH instruction, 150
FMADD instruction
SIMD 演算のレジスタ指定法の特例，73
FMOVcc instructions, 168
FMOVCd instruction, 165
FMOVCq instruction, 165
FMOVcs instruction, 165
FMOVd instruction, 164, 165
FMOVq instruction, 164, 165
FMOVr instructions, 168
FNEGd instruction, 164, 165
FNEGq instruction, 164, 165
fp_disabled exception, 67, 75, 84, 93, 102, 106, 130
fp_exception_ieee_754 exception, 75, 143, 144
fp_exception_other exception, 50, 58, 140, 157
FQ deferred トラブルキュー，149
FqTOx instruction, 164, 165
FSR
aexc field, 24
cexc field, 23, 24
conformance, 24
NS field, 140
TEM field, 24
VER field, 23
FxTOx instruction, 164, 165
fITLB, 156, 203, 301
PTRIMADDd instruction, 41, 42, 60, 69, 123, 124, 142, 146, 309, 331
FxTOd instruction, 164, 165
FxTOq instruction, 164, 165
FxTOs instruction, 164, 165

G
GSR register, 290

H
hardware barrier, 77
HPC-ACE, 3, 3, 8, 21, 49, 50, 57, 58, 70, 82, 86, 101, 105, 129, 132, 148, 161, 206
アセンブリ言語の表記法，206
アセンブリ言語表記，206

I
i field of instructions, 81, 85
I UGE
definition, 259
error detection action, 273
type, 259
IAE
reporting, 260
IE, Invert Endianness bit, 175
IEEE Std 754-1985, 139
IIU_INST_TRAP register, 58, 294
illegal_action exception, 45, 51
illegal_instruction exception, 38, 50, 75, 84, 93, 96, 102, 107, 110, 149
imm Asi field of instructions, 81, 85
IMMU
registers accessed, 183
IMMU_DEMAP register, 293
IMMU_SFSR register, 293
IMMU_TAG_ACCESS register, 293, 294
IMMU_TAG_TARGET register, 293
IMMU_TSB_BASE register, 293, 294
IMPDEP1 instruction, 42, 69
IMPDEP1 instructions, 169, 170, 171
IMPDEP2 instruction, 42, 69, 73
IMPDEP2A instruction, 78
IMPDEP2B instruction, 70
IMPDEPn instructions, 69, 70
implementation number (impl) field of VER
register, 148
instruction fields
i, 81, 85
imm asi, 81, 85
op3, 81, 85
rd, 81, 85
rs1, 81, 85
rs2, 81, 85
simm13, 81, 85
instruction fields, reserved, 57
instruction_access_error exception, 57, 179, 195, 197, 260
instruction_access_exception exception, 57, 177, 178, 196, 197


instruction_access_MMU_miss exception, 58
instructions
  cacheable, 232
  FLUSH, 150
implementation-dependent (IMPDEP2), 42
implementation-dependent (IMPDEPn), 69, 70
prefetch, 152, 182
store floating-point into alternate space, 104
timing, 58
write privileged register, 108
integer unit (IU) deferred-trap queue, 38
interrupt
  dispatch, 241
  level 15, 28
Interrupt Vector Receive Register, 245
interrupt_level_n exception, 310
interrupt_level_n exception, 77
interrupt_vector_trap exception, 45, 77, 310
INTR_DATA0
  3_W register, error handling, 294
INTR_DATA0:7_R register, error handling, 294
INTR_DISPATCH_STATUS register, 241, 293
INTR_DISPATCH_W register, 294
INTR_RECEIVE register, 293
I-SFSR
  update during MMU trap, 179
ISFSR
  FT field, 196
  update policy, 197
ITLB_DATA_ACCESS register, 293
ITLB_DATA_IN register, 293
ITLB_TAG_READ register, 293
IU deferred トランプキー , 148

J
JEDEC manufacturer code, 26

L
LDD instruction, 45
LDDA instruction, 45, 88, 199
LDDF instruction, 81
LDFF_mem_address_not_aligned exception, 84, 87, 157, 221
LDFFA instruction, 85, 221
LDF instruction, 81
LDFA instruction, 85
LDQF instruction, 81
LDQF_mem_address_not_aligned exception, 58
LDQFA instruction, 85
LDSTUB instruction, 39, 45, 199
LDSTUBA instruction, 199
LDXFSR instruction, 81
load quadword atomic, 88
LoadLoad MEMBAR relationship, 90
LoadStore MEMBAR relationship, 90
Lookaside MEMBAR relationship, 91

M
MAXTL, 44, 151, 248, 250
MCNTL_NC_CACHE, 232
mem_address_not_aligned exception, 84, 87, 88, 102, 106, 130, 157, 178, 200, 221
MEMBAR
  #LoadLoad, 90
  #LoadStore, 90
  #Lookaside, 91
  #MemIssue, 91
  #StoreLoad, 90
  #Sync, 91
blockload and blockstore, 66
in interrupt dispatch, 242
instruction, 90
partial ordering enforcement, 91
membar_mask field, 90
memory model
  TSO, 54
MEMORY_CONTROL register, 293
mmask field, 90
MMU
  disabled, 182
  exceptions recorded, 178
  registers accessed, 183
Synchronous Fault Address Registers, 249
TLB data access address assignment, 192
MOVcc instructions, 166, 168
MOVr instructions, 168

N
next program counter (nPC), 92
nonstandard floating-point (NS) field of FSR register, 22, 148
nonstandard floating-point mode, 22, 140
NOP instruction, 92
O

OBP
facilitating diagnostics, 232
validating register error handling, 289
with urgent error, 260
op3 field of instructions, 81, 85
opf_cc field of instructions, 168
other windows (OTHERWIN) register, 108
OTHERWIN register, 153, 278

P

P superscript on instruction name, 58
PA_watchpoint exception, 200
panic process, 260
parity error
  counting in L1D cache, 299
partial store instructions, 221
partial store order (PSO) memory model, 53
PASI superscript on instruction name, 58
PASR superscript on instruction name, 58
PC register, 281
PCR
  error handling, 290
NC field, 27
OVF field, 27
OVRO field, 27
PRIV field, 28, 97
SC field, 27, 304
ST field, 308
SU field, 304
UT field, 308
pessimistic zero, 143
PIC register
  clearing, 303
  error handling, 290
  nonprivileged access, 28
  OVF field, 28
PIL register, 45
P_superscript on instruction name, 58
power-on reset (POR)
  implementation dependency, 150
P_superscript on instruction name, 58
P_superscript on instruction name, 58
precise traps, 45
prefetch
  instruction, 152, 182
prefetcha instruction, 95

PRIM_A CONTEXT register, 293
privileged (PRIV) field of PSTATE register, 86, 105
privileged registers, 25
privileged_action exception, 28, 87, 98, 106, 157, 178, 200
  PCR access, 97, 111
privileged_opcode exception, 29, 110
  TXAR access, 97
processor interrupt level (PIL) register, 108
processor state (PSTATE) register, 108
processor states
  error_state, 44, 150, 250
  execute_state, 250
  RED_state, 44, 250
program counter (PC), 92
program counter (PC) register, 153
PSTATE register
  AM field, 42, 68, 80, 153
  IE field, 242, 243
  MM field, 54
  PRIV field, 97, 111
  RED field, 25, 232, 250, 251, 253, 254
PTE
  E field, 40

R

RAS, see Return Stack Address, 13
rcond field of instructions, 168
r/r field of instructions, 81, 85
RDASI instruction, 97
RDASR instruction, 97
RDCCCR instruction, 97
Rddcr instruction, 97
RDFPRS instruction, 97
RDGSR instruction, 97
RDPC instruction, 97
RDPCR instruction, 28, 97
RDPCIC instruction, 28, 97
RDPCPC instruction, 97
RDPTT instruction, 97
RDPCA instruction, 97
RED state, 281
entry after WDR, 250
processor states, 250, 251
setting of PSTATE.RED, 25
trap vector address (RSTVaddr), 152
 registers
clean windows (CLEANWIN), 108, 153
clock-tick (TICK), 151
current window pointer (CWP), 108, 153
Data Cache Unit Control (DCUCR), 34
other windows (OTHERWIN), 108, 153
privileged, 25
processor interrupt level (PIL), 108
processor state (PSTATE), 108
restorable windows (CANRESTORE), 108, 153
savable windows (CANSAVE), 108, 153
TICK, 108
trap base address (TBA), 108
trap level (TL), 108
trap next program counter (TNPC), 108
trap program counter (TPC), 108
trap state (TSTATE), 108
trap type (TT), 108
window state (WSTATE), 108
relaxed memory order (RMO) memory model, 53
reserved, 2
 reset
 externally_initiated_reset (XIR), 248
 power_on_reset (POR), 150
 software_initiated_reset (SIR), 248
 resets
 WDR, 265
 restorable windows (CANRESTORE) register, 108, 153
 Restrainable error, 266, 267
 restrainable error
definitions, 261
 handling
 ASI_AFSR.UE_DST_BETO, 288
 ASI_AFSR.UE_RAW_L2SFILL, 288
 UE_RAW_D1SINS, 288
 UE_RAW_L2SINS, 288
 Return Address Stack, 13
 rs1 field of instructions, 81, 85
 rs2 field of instructions, 81, 85
 rs3 field of instructions, 41
 RSTVaddr, 152, 250

S
 savable windows (CANSAVE) register, 108, 153
 sDTLB, 12, 173
 SECONDARY_CONTEXT register, 293
 SERIAL_ID register, 293
 SET_SOFTINT register, 290
 SETHI instruction, 131
 SHARED_CONTEXT register, 294
 SHUTDOWN instruction, 99
 SIMD, 8
 cexc, aexc update, 24
 load/store
   memory ordering, 129
 SXAR による指定, 131
 watchpoint 検出, 201
 アセンブリ言語での指示方法, 206
 レジスタ指定方法
 FMADD の特例, 73
 ロード
 メモリオーダリング, 83
 ロードストア
 watchpoint 検出, 37, 83, 102, 129
 エンディアン変換, 22, 83
 ノンキャッシュブル, 83, 102, 129
 メモリオーダリング, 102
 倍精度ロードと
 LDHF_mem_address_not_aligned, 83
例外検出条件, 180
 SIMD_load_across_pages exception, 45, 51, 83, 178,
 179, 180, 181, 182, 200, 310, 332
 simm13 field of instructions, 81, 85
 SIR instruction, 248
 sITLB, 12, 155, 173, 179
 size field of instructions, 41
 SLEEP instruction, 69, 77, 77, 315, 331
 wake on barrier synchronization, 77
 ハードウェアバリア同期による解除, 77
 SOFTINT register, 45, 243, 291
 software_trap_number, 205
 spill_n_normal exception, 310
 spill_n_other exception, 310
 STBAR instruction, 113
 STCHG_ERROR_INFO register, 293
 STD instruction, 45
 STDA instruction, 45
 STDF instruction, 100
 STDFR instruction, 102, 106
 STDFR_mem_address_not_aligned exception, 102, 413
 STDTA instruction, 104, 221
 STDF instruction, 128
 STF instruction, 100
 STFA instruction, 104
STFR instruction, 128
STICK, 77
STICK register, 97, 278, 291
STICK_COMP register, 278
STICK_COMPARE register, 97, 291
sTLB, 155, 156, 187, 192, 201, 203, 204, 301
store floating-point into alternate space instructions, 104
store order (STO) memory model, 153
StoreLoad MEMBAR relationship, 90
StoreStore MEMBAR relationship, 90
STQF instruction, 100
STQF_mem_address_not_aligned exception, 58
STQFA instruction, 104, 104
STXFSR instruction, 100
SUSPEND instruction, 64, 69, 76, 276, 315, 331
suspended state, 76, 257
SWAP instruction, 39, 45, 199
SWAPA instruction, 199
SXAR, 51
SXAR instruction, 131
Sync MEMBAR relationship, 91
synchronizing caches, 54
syncing 命令, 3

T
TA instruction, 167
tcc instructions, 163, 166, 169
tcs instruction, 167
tE instruction, 167
tG instruction, 167
tGE instruction, 167
tGU instruction, 167
tICK register, 25, 151
tICK_COMPARE register, 291
TL instruction, 167
TL register, 109, 248, 250
TLB
  CP field, 232, 233
  index, 192
  replacement algorithm, 192
tle instruction, 167
tLEU instruction, 167
TN instruction, 167
TNE instruction, 167
TNEG instruction, 167
tick register, 25, 151
tick_compare register, 291
TL instruction, 167
TL register, 109, 248, 250
tlb
  CP field, 232, 233
  index, 192
  replacement algorithm, 192
TLE instruction, 167
TLEU instruction, 167
TN instruction, 167
TNE instruction, 167
TNEG instruction, 167
total store order (TSO) memory model, 53, 54
TPOS instruction, 167
trap base address (TBA) register, 108
trap level (TL) register, 108
trap next program counter (TNPC) register, 108
trap program counter (TPC) register, 108
trap state (TSTATE) register, 108
trap type (TT) register, 108
trap Instruction (ISA) exception, 107
traps
devoted, 44
TSTATE register
  CP field, 25
  Context field, 175
  CV field, 175
  DA field, 175
  DVA field, 175
  E field, 177
  G field, 175, 177
  L field, 176
  NFO field, 175
  P field, 177
  PA field, 176
  Size field, 175
  Soft2 field, 175
  V field, 175
  V a_tag field, 175
  W field, 177
  Urgent error, 265
  Urgent error, 266, 267
  Urgent error
    definition, 259
    types
      A_UGE, 259
      DAE, 259
      IAE, 259
    Urgent_error_STATUS register, 293

U
uDTLB, 173
uITLB, 173, 179
uncorrectable error, 261
unimplemented_FPpop exception, 140, 144
unimplemented_FPpop floating-point trap type, 147
unimplemented_LDD exception, 58
unimplemented_STD exception, 58
Urgent Error, 265
Urgent error, 266, 267
urgent error
  definition, 259
  types
    A_UGE, 259
    DAE, 259
    IAE, 259
    Urgent_error_STATUS register, 293

x
VA_watchpoint exception, 200
var field of instructions, 41
VER register, 26
VIS instructions
  encoding, 169, 170

watchpoint exception
  on block load-store, 67
  quad-load physical instruction, 89
WDR reset, 265
window state (WSTATE) register
  writing WSTATE with WRPR instruction, 108
WRASI instruction, 111
WRASR instruction, 111
WRDCR instruction, 111
WRPCR instruction, 111
WRPIC instruction, 111
WRSOFTINT instruction, 111
WRSTICK instruction, 111
WRSTICK_CMPR instruction, 111
WRTICK_CMP instruction, 111
WRXAR instruction, 111
WRXASR instruction, 111
WRCCCR instruction, 111
WRFPRS instruction, 111
write privileged register instruction, 108
WRPCR instruction, 28, 111
WRPIC instruction, 28
WRY instruction, 111

XAR register, 291
XASR register, 291

ア行
アウトオブオーダ，4, 7, 11, 15, 16, 25, 39, 239, 322
エラー

I ビットエラーの訂正，8
種類，257
分離，9
分離，9

カ行
解放
メモリポート，16
リザーベーションステーション，15
資源の，4
カウンタ
オーバープロ (PIC の)，28
( 命令の ) 完了，4, 7, 11, 16
ECR, UGB_HANDLER をクリア，283
FSR の更新，42
RDTICK で取得される値，25
sync 命令，40
watchdog_reset 検出条件，44
XAR フィールドのクリア，29, 30, 32
エラーを起こした命令の，9
エラーを検出した命令の完了方法，259
ストアの突き放し，40
例外の通知，40
完了
MEMBAR によるメモリアクセス，90
XFILL 中のラインに対する後続アクセス，134
メモリアクセス，16
キャッシュ
エラー保護，8
命令
概要，12
緊急エラー，51, 257, 259, 264, 265, 266, 267, 275,
276, 276, 289
自律性緊急エラー，260
命令緊急エラー，259
コア，4, 7, 9, 55, 330
BST, BST_mask との対応，223, 225
L2 キャッシュを共有，231
L2 キャッシュ関連の PA イベント計測対象，317
SCCR を共有，236
STICK_CNTL を共有，326
インタラブトの配送，245
ハードウェアバリアを共有，222
リセット範囲，247
縮退，275
コミット，4, 16, 323
(命令の) 完了を参照
ウォッチドッグタイマーによる監視，149

サ行
サイクルアカウンティング，4, 321, 323
サスペンド，4, 257, 258, 261, 262
EE エラーの通知，276
SUSPEND instruction, 76
スーパースカラ，4, 7
スキャラン，4, 255
ストアバッファ，12, 179
エラー通知の制限，179, 288
ストア突き放し，40
ストール，4, 13
ストロング プリフェッチ，5
ストロングプリフェッチ，35, 95
スレッド，5, 8, 76, 77, 223, 224, 257, 258, 259, 261, 262, 275, 276

タ行
致命的エラー，154, 257, 258, 260, 264, 265, 266, 267, 273, 287, 288, 299, 300
ステータス表示，274
投機
メモリアクセス，36
TLB エラー検出，180, 181
メモリアクセスの抑止，35
投機的命令実行，5
ASI_FLUSH_L1I，235
PA 計測，305

ハ行
ハードウェアバリア，8, 77, 214, 222, 222, 222
資源，222, 224
全コアで共有，222
同期，224
フェッチ，5
フォーマット
命令，40
ブランチヒストリ，7, 7, 11

マ行
窓 ASI, 77, 224, 227, 229
命令実行，7
EU, 11
reserved が 0 でない命令，2
メモリアクセス
投機，36
投機アクセスの抑止，35

ヤ・ワ行
抑止可能エラー，257, 261, 264, 265, 266, 267, 273, 287, 288, 299, 300
ライトバッファ，12
リザーブーションステーション，5, 7, 11, 313
リネーミングレジスタ，5
レジスタ
フィールドの読み書き属性，2