

# V850E2M

ユーザーズマニュアル アーキテクチャ編

ルネサスマイクロコンピュータ  
V850E2M マイクロプロセッサ・コア

## ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器・システムの設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因して、お客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
2. 本資料に記載されている情報は、正確を期すため慎重に作成したのですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
3. 本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害に関し、当社は、何らの責任を負うものではありません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を改造、改変、複製等しないでください。かかる改造、改変、複製等により生じた損害に関し、当社は、一切その責任を負いません。
5. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。  
標準水準： コンピュータ、OA 機器、通信機器、計測機器、AV 機器、  
家電、工作機械、パーソナル機器、産業用ロボット等  
高品質水準： 輸送機器（自動車、電車、船舶等）、交通用信号機器、  
防災・防犯装置、各種安全装置等  
当社製品は、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（原子力制御システム、軍事機器等）に使用されることを意図しておらず、使用することはできません。たとえ、意図しない用途に当社製品を使用したことによりお客様または第三者に損害が生じて、当社は一切その責任を負いません。なお、ご不明点がある場合は、当社営業にお問い合わせください。
6. 当社製品をご使用の際は、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他の保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
8. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
9. 本資料に記載されている当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。また、当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途に使用しないでください。当社製品または技術を輸出する場合は、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。
10. お客様の転売等により、本ご注意書き記載の諸条件に抵触して当社製品が使用され、その使用から損害が生じた場合、当社は何らの責任も負わず、お客様にてご負担して頂きますのでご了承ください。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。

注 1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

## CMOSデバイスの一般的注意事項

### 入力端子の印加波形

入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。

CMOSデバイスの入力にノイズなどに起因して、 $V_{IL}$  (MAX.) から  $V_{IH}$  (MIN.) までの領域にとどまるような場合は、誤動作を引き起こす恐れがあります。入力レベルが固定な場合はもちろん、 $V_{IL}$  (MAX.) から  $V_{IH}$  (MIN.) までの領域を通過する遷移期間中にチャタリングノイズ等が入らないようご使用ください。

### 未使用入力の処理

CMOSデバイスの未使用端子の入力レベルは固定してください。

未使用端子入力については、CMOSデバイスの入力に何も接続しない状態で動作させるのではなく、プルアップかプルダウンによって入力レベルを固定してください。また、未使用の入出力端子が出力となる可能性（タイミングは規定しません）を考慮すると、個別に抵抗を介して  $V_{DD}$  または GND に接続することが有効です。

資料中に「未使用端子の処理」について記載のある製品については、その内容を守ってください。

### 静電気対策

MOSデバイス取り扱いの際は静電気防止を心がけてください。

MOSデバイスは強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には、当社が出荷梱包に使用している導電性のトレーやマガジン・ケース、または導電性の緩衝材、金属ケースなどを利用し、組み立て工程にはアースを施してください。プラスチック板上に放置したり、端子を触ったりしないでください。

また、MOSデバイスを実装したボードについても同様の扱いをしてください。

### 初期化以前の状態

電源投入時、MOSデバイスの初期状態は不定です。

電源投入時の端子の出力状態や入出力設定、レジスタ内容などは保証しておりません。ただし、リセット動作やモード設定で定義している項目については、これらの動作ののちに保証の対象となります。

リセット機能を持つデバイスの電源投入後は、まずリセット動作を実行してください。

### 電源投入切断順序

内部動作および外部インタフェースで異なる電源を使用するデバイスの場合、原則として内部電源を投入した後に外部電源を投入してください。切断の際には、原則として外部電源を切断した後に内部電源を切断してください。逆の電源投入切断順により、内部素子に過電圧が印加され、誤動作を引き起こしたり、異常電流が流れ内部素子を劣化させたりする場合があります。

資料中に「電源投入切断シーケンス」についての記載のある製品については、その内容を守ってください。

### 電源OFF時における入力信号

当該デバイスの電源がOFF状態の時に、入力信号や入出力プルアップ電源を入れないでください。入力信号や入出力プルアップ電源からの電流注入により、誤動作を引き起こしたり、異常電流が流れ内部素子を劣化させたりする場合があります。

資料中に「電源OFF時における入力信号」についての記載のある製品については、その内容を守ってください。

# このマニュアルの使い方

**対象者** このマニュアルは、V850E2M CPUコアの機能を理解し、それを用いたアプリケーション・システムを設計しようとするユーザを対象とします。

**目的** このマニュアルは、次の構成に示すV850E2M CPUコアのアーキテクチャをユーザに理解していただくことを目的としています。

**構成** このマニュアルは、おもに次の内容で構成しております。

- 基本機能
- プロセッサ保護機能
- 浮動小数点演算機能

**読み方** このマニュアルの読者には、電気、論理回路、およびマイクロコントローラに関する一般知識を必要とします。

ハードウェアの機能について知りたいとき

**各製品のユーザズ・マニュアル ハードウェア編**をお読みください。

特定の命令の機能を詳細に調べたいとき

**第2編 第5章 命令、第4編 第4章 命令**をお読みください。

**凡例** データ表記の重み：左が上位桁，右が下位桁

アクティブ・ロウの表記： $\overline{xxx}$ （端子，信号名称に上線）

メモリ・マップのアドレス：上部 - 上位，下部 - 下位

注：本文中に付けた注の説明

注意：気を付けて読んでいただきたい内容

備考：本文の補足説明

数の表記：2進数 ... xxxxまたはxxxxB

10進数 ... xxxx

16進数 ... xxxxH

2のべき数を示す接頭語（アドレス空間，メモリ容量）：

K（キロ）： $2^{10} = 1024$

M（メガ）： $2^{20} = 1024^2$

G（ギガ）： $2^{30} = 1024^3$

# 目 次

第1編 概 要.....	16
<b>第1章 特 徴.....</b>	<b>17</b>
1.1 基本機能.....	17
1.2 プロセッサ保護機能.....	18
1.3 浮動小数点演算機能.....	18
第2編 基本機能.....	19
<b>第1章 概 説.....</b>	<b>20</b>
1.1 特 徴.....	21
<b>第2章 レジスタ・セット .....</b>	<b>22</b>
2.1 プログラム・レジスタ .....	23
2.2 システム・レジスタ・バンク .....	25
2.2.1 BSEL - レジスタ・バンクの選択.....	27
2.3 CPU機能グループ/基本バンク .....	28
2.3.1 EIPC, EIPSW - EIレベル例外受け付け時の状態退避レジスタ.....	29
2.3.2 FEPC, FEPSW - FEレベル例外受け付け時の状態退避レジスタ.....	29
2.3.3 ECR - 例外要因.....	30
2.3.4 PSW - プログラム・ステータス・ワード .....	31
2.3.5 SCCFG - SYSCALLの動作設定.....	34
2.3.6 SCBP - SYSCALLベース・ポインタ.....	34
2.3.7 EIIC - EIレベル例外要因 .....	35
2.3.8 FEIC - FEレベル例外要因.....	35
2.3.9 CTPC, CTPSW - CALLT実行時の状態退避レジスタ .....	35
2.3.10 CTBP - CALLTベース・ポインタ .....	36
2.3.11 EIWR - EIレベル例外用作業レジスタ.....	36
2.3.12 FEWR - FEレベル例外用作業レジスタ .....	36
2.3.13 DBIC - DBレベル例外要因 .....	37
2.3.14 DBPC, DBPSW - DBレベル例外受け付け時の状態退避レジスタ .....	37
2.3.15 DBWR - DBレベル例外用作業レジスタ .....	37
2.3.16 DIR - デバッグ・インタフェース・レジスタ .....	37
2.4 CPU機能グループ/例外ハンドラ・アドレス切り替え機能バンク .....	38
2.4.1 SW_CTL - 例外ハンドラ・アドレス切り替えの制御.....	39
2.4.2 SW_CFG - 例外ハンドラ・アドレス切り替え設定.....	39
2.4.3 SW_BASE - 例外ハンドラ・アドレス切り替えベース・アドレス .....	39
2.4.4 EH_CFG - 例外ハンドラ設定 .....	40
2.4.5 EH_BASE - 例外ハンドラ・ベース・アドレス.....	40
2.4.6 EH_RESET - リセット・アドレス .....	41
2.5 ユーザ・グループ .....	42

<b>第3章</b>	<b>データ・タイプ</b> .....	44
3.1	データ形式 .....	44
3.1.1	バイト .....	44
3.1.2	ハーフワード .....	44
3.1.3	ワード .....	45
3.1.4	ビット .....	45
3.2	データ表現 .....	46
3.2.1	整数 .....	46
3.2.2	符号なし整数 .....	46
3.2.3	ビット .....	46
3.3	データ・アラインメント .....	47
<b>第4章</b>	<b>アドレス空間</b> .....	48
4.1	メモリ・マップ .....	49
4.2	アドレッシング・モード .....	51
4.2.1	命令アドレス .....	51
4.2.2	オペランド・アドレス .....	54
<b>第5章</b>	<b>命令</b> .....	57
5.1	オペコードと命令フォーマット .....	57
5.1.1	CPU命令 .....	57
5.1.2	コプロセッサ命令 .....	62
5.1.3	予約命令 .....	62
5.2	命令の概要 .....	63
5.3	命令セット .....	68
	ADD .....	71
	ADDI .....	72
	ADF .....	73
	AND .....	74
	ANDI .....	75
	Bcond .....	76
	BSH .....	78
	BSW .....	79
	CALLT .....	80
	CAXI .....	81
	CLR1 .....	82
	CMOV .....	84
	CMP .....	86
	CTRET .....	87
	DI .....	88
	DISPOSE .....	89
	DIV .....	91
	DIVH .....	92
	DIVHU .....	94
	DIVQ .....	95
	DIVQU .....	97
	DIVU .....	98
	EI .....	99
	EIRET .....	100

FERET .....	101
FETRAP .....	102
HALT .....	103
HSH .....	104
HSW .....	105
JARL .....	106
JMP .....	108
JR .....	109
LD.B .....	110
LD.BU .....	111
LD.H .....	112
LD.HU .....	113
LD.W .....	114
LDSR .....	115
MAC .....	116
MACU .....	117
MOV .....	118
MOVEA .....	119
MOVHI .....	120
MUL .....	121
MULH .....	122
MULHI .....	123
MULU .....	124
NOP .....	125
NOT .....	126
NOT1 .....	127
OR .....	129
ORI .....	130
PREPARE .....	131
RETI .....	133
RIE .....	135
SAR .....	136
SASF .....	138
SATADD .....	139
SATSUB .....	141
SATSUBI .....	142
SATSUBR .....	143
SBF .....	144
SCH0L .....	145
SCH0R .....	146
SCH1L .....	147
SCH1R .....	148
SET1 .....	149
SETF .....	151
SHL .....	153
SHR .....	155
SLD.B .....	157
SLD.BU .....	158
SLD.H .....	159
SLD.HU .....	160
SLD.W .....	161
SST.B .....	162

SST.H.....	163
SST.W.....	164
ST.B.....	165
ST.H.....	166
ST.W.....	167
STSR.....	168
SUB.....	169
SUBR.....	170
SWITCH.....	171
SXB.....	172
SXH.....	173
SYNCE.....	174
SYNCM.....	175
SYNCP.....	176
SYSCALL.....	177
TRAP.....	179
TST.....	180
TST1.....	181
XOR.....	182
XORI.....	183
ZXB.....	184
ZXH.....	185
<b>第6章 例外</b> .....	<b>186</b>
6.1 例外の仕組み.....	186
6.1.1 例外要因一覧.....	186
6.1.2 例外の種別.....	189
6.1.3 例外処理フロー.....	191
6.1.4 例外受け付けの優先順位と保留条件.....	192
6.1.5 例外の受け付け条件.....	192
6.1.6 再開と回復.....	193
6.1.7 例外レベルとコンテキスト退避.....	193
6.1.8 復帰命令.....	194
6.2 例外発生時の動作.....	197
6.2.1 受け付け条件のないEIレベル例外.....	197
6.2.2 受け付け条件のあるEIレベル例外.....	199
6.2.3 受け付け条件のないFEレベル例外.....	201
6.2.4 受け付け条件のあるFEレベル例外.....	203
6.2.5 特殊な動作.....	205
6.3 例外の管理.....	207
6.3.1 例外受け付け / 復帰時の例外同期.....	209
6.3.2 例外同期命令.....	209
6.3.3 保留中例外の確認と取り下げ.....	209
6.4 例外ハンドラ・アドレス切り替え機能.....	211
6.4.1 例外ハンドラ・アドレスの決定.....	211
6.4.2 例外ハンドラ・アドレスの切り替えの目的.....	212
6.4.3 例外ハンドラ・アドレス切り替え機能の設定方法.....	212
<b>第7章 コプロセッサ使用不可状態</b> .....	<b>213</b>



7.1	コプロセッサ使用不可例外 .....	213
7.2	システム・レジスタ .....	213
<b>第8章</b>	<b>リセット .....</b>	<b>214</b>
8.1	リセット後のレジスタの状態 .....	214
8.2	起 動 .....	214
<b>第3編</b>	<b>プロセッサ保護機能 .....</b>	<b>215</b>
<b>第1章</b>	<b>概 説 .....</b>	<b>216</b>
1.1	特 徴 .....	216
<b>第2章</b>	<b>レジスタ・セット .....</b>	<b>218</b>
2.1	システム・レジスタ・バンク .....	218
2.2	システム・レジスタ .....	220
2.2.1	PSW - プログラム・ステータス・ワード .....	223
2.2.2	MPM - プロセッサ保護動作モードの設定 .....	224
2.2.3	MPC - プロセッサ保護コマンドの指定 .....	226
2.2.4	TID - タスク識別子 .....	226
2.2.5	その他のシステム・レジスタ .....	226
<b>第3章</b>	<b>動作設定 .....</b>	<b>227</b>
3.1	プロセッサ保護機能の利用開始 .....	227
3.2	実行レベル自動遷移機能の設定 .....	227
3.3	プロセッサ保護機能の利用停止 .....	227
<b>第4章</b>	<b>実行レベル .....</b>	<b>228</b>
4.1	プログラムの性質 .....	228
4.2	PSW上の保護ビット .....	229
4.2.1	Tステート (信頼状態) .....	229
4.2.2	NTステート (非信頼状態) .....	229
4.3	実行レベルの定義 .....	229
4.4	実行レベルの遷移 .....	230
4.4.1	システム・レジスタへの書き込み命令の実行による遷移 .....	230
4.4.2	例外の発生による遷移 .....	231
4.4.3	復帰命令の実行による遷移 .....	231
4.5	プログラム・モデル .....	231
4.6	タスク識別子 .....	232
<b>第5章</b>	<b>システム・レジスタ保護 .....</b>	<b>233</b>
5.1	レジスタ・セット .....	234
5.1.1	VSECR - システム・レジスタ保護違反要因 .....	235
5.1.2	VSTID - システム・レジスタ保護違反タスク識別子 .....	235
5.1.3	VSADR - システム・レジスタ保護違反アドレス .....	235
5.2	アクセス制御 .....	236
5.3	対象レジスタ .....	236
5.4	違反の検出 .....	237

5.5	運用方法	237
<b>第6章</b>	<b>メモリ保護</b>	<b>238</b>
6.1	レジスタ・セット	238
6.1.1	IPAnL - 命令/定数保護領域n下限アドレス (n = 0-4)	240
6.1.2	IPAnU - 命令/定数保護領域n上限アドレス (n = 0-4)	241
6.1.3	DPAnL - データ保護領域n下限アドレス (n = 0-5)	242
6.1.4	DPAnU - データ保護領域n上限アドレス (n = 0-5)	243
6.1.5	VMECR - メモリ保護違反要因	244
6.1.6	VMTID - メモリ保護違反タスク識別子	245
6.1.7	VMADR - メモリ保護違反アドレス	245
6.2	アクセス制御	246
6.3	保護領域の設定	246
6.3.1	有効ビット (Eビット)	248
6.3.2	実行許可ビット (Xビット)	248
6.3.3	リード許可ビット (Rビット)	248
6.3.4	ライト許可ビット (Wビット)	248
6.3.5	sp相対アクセス許可ビット (Sビット)	249
6.3.6	保護領域指定方式ビット (Tビット)	249
6.3.7	保護領域下限アドレス (AL31-AL0ビット)	249
6.3.8	保護領域上限アドレス (AU31-AU0ビット)	249
6.4	保護領域設定時の注意事項	250
6.4.1	保護領域境界の交差	250
6.4.2	無効な保護領域の設定	250
6.5	スタック検査機能	251
6.6	特殊なメモリ・アクセス命令	253
6.6.1	ミスアライン・アクセスを行うロード/ストア命令	253
6.6.2	一部のビット操作命令とCAXI命令	253
6.6.3	スタック・フレーム操作命令	253
6.6.4	SYSCALL命令	253
6.7	保護違反と例外	254
<b>第7章</b>	<b>周辺装置保護</b>	<b>255</b>
7.1	レジスタ・セット	255
7.1.1	PPM - 周辺装置保護動作モードの設定	257
7.1.2	PPEC - 周辺装置保護例外の制御	258
7.1.3	VPNECR - 周辺装置保護NTステート違反要因	259
7.1.4	VPNADR - 周辺装置保護NTステート違反アドレス	260
7.1.5	VPNTID - 周辺装置保護NTステート違反タスクID	260
7.1.6	VPTECR - 周辺装置保護Tステート違反要因	261
7.1.7	VPTADR - 周辺装置保護Tステート違反アドレス	262
7.1.8	VPTTID - 周辺装置保護Tステート違反タスクID	262
7.1.9	PPSn - 特殊周辺装置の指定	263
7.1.10	PPPn - OS周辺装置の指定	263
7.1.11	PPVn - 一般周辺装置保護の有効指定	264
7.1.12	PPTn - 一般周辺装置の保護種別の指定	265
7.2	メモリ保護と周辺装置保護の位置付け	266
7.3	周辺装置の種類	267
7.3.1	周辺装置の種類の設定	268

7.3.2	一般周辺装置に対する詳細な保護設定	269
7.4	T状態における周辺装置保護違反	270
7.5	NT状態における周辺装置保護違反	270
7.5.1	後続アクセスの無効化	270
7.6	PPI例外の取り扱い	271
7.6.1	PPI例外の取り下げ	271
7.6.2	PPI例外を用いない運用方法	272
7.6.3	PPI例外処理で行うべき操作	272
7.7	周辺装置保護違反の検出結果一覧	272
7.8	特殊周辺装置へのアクセス方法	273
7.9	周辺装置保護設定レジスタに対する保護設定	273
7.10	特殊な周辺装置アクセス命令	273
7.10.1	SYSCALL命令	273
<b>第8章</b>	<b>タイミング監視機能</b>	<b>274</b>
8.1	レジスタ・セット	274
8.1.1	TSEC - タイミング監視の制御	276
8.1.2	TSECR - タイミング監視例外要因	277
8.1.3	TSCCFGn - タイミング監視機能カウンタnの設定 (n = 0-5)	278
8.1.4	TSCCNTn - タイミング監視カウンタnのカウント値 (n = 0-5)	281
8.1.5	TSCCMPn - タイミング監視カウンタnのコンペア値 (n = 0-5)	282
8.1.6	TSCRLDn - タイミング監視カウンタnのリロード値 (n = 0-5)	283
8.2	カウンタ機能	284
8.2.1	解像度	284
8.2.2	カウント方向	284
8.2.3	例外モード	285
8.2.4	オート・リロード	285
8.2.5	カウンタ・モード	285
8.3	カウンタとCPUの動作モード	287
8.4	違反の検出	287
8.4.1	違反要因の特定	287
8.5	TSI例外の取り扱い	288
8.5.1	TSI例外の通知	288
8.5.2	例外要因の特定	290
8.5.3	TSI例外の取り下げ	290
8.6	各監視機能に対応するカウンタの設定	290
8.6.1	グローバル割り込みロック監視	290
8.6.2	ランタイム監視	291
8.6.3	割り込み到着回数監視	291
8.6.4	経過時間の監視	292
<b>第9章</b>	<b>プロセッサ保護例外</b>	<b>293</b>
9.1	違反の種類	293
9.1.1	システム・レジスタ保護違反	293
9.1.2	実行保護違反	293
9.1.3	データ保護違反	293
9.1.4	周辺装置保護違反	293
9.1.5	タイミング監視違反	294
9.2	例外の種類	294

9.2.1	MIP例外	294
9.2.2	MDP例外	294
9.2.3	PPI例外	294
9.2.4	TSI例外	294
9.3	違反要因の特定	295
9.3.1	MIP例外	295
9.3.2	MDP例外	296
9.3.3	PPI例外	296
9.3.4	TSI例外	296
<b>第10章</b>	<b>メモリ保護設定チェック機能</b>	<b>297</b>
10.1	レジスタ・セット	298
10.1.1	MCA - メモリ保護設定チェック・アドレス	299
10.1.2	MCS - メモリ保護設定チェック・サイズ	299
10.1.3	MCC - メモリ保護設定チェック・コマンド	300
10.1.4	MCR - メモリ保護設定チェック結果	300
<b>第11章</b>	<b>特殊機能</b>	<b>302</b>
11.1	メモリ保護設定の一括クリア	302
<b>第4編</b>	<b>浮動小数点演算機能</b>	<b>303</b>
<b>第1章</b>	<b>概 説</b>	<b>304</b>
1.1	特 徴	304
1.2	浮動小数点演算機能の搭載 / 非搭載	305
<b>第2章</b>	<b>レジスタ・セット</b>	<b>306</b>
2.1	浮動小数点レジスタ	306
2.2	浮動小数点システム・レジスタ	306
2.2.1	FPSR - 浮動小数点演算の設定 / ステータス	308
2.2.2	FPEPC - 浮動小数点演算例外プログラム・カウンタ	310
2.2.3	FPST - 浮動小数点演算のステータス	311
2.2.4	FPCC - 浮動小数点演算の比較結果	311
2.2.5	FPCFG - 浮動小数点演算の設定	312
2.2.6	FPEC - 浮動小数点演算例外の制御	313
<b>第3章</b>	<b>データ・タイプ</b>	<b>314</b>
3.1	データ形式	314
3.1.1	浮動小数点の形式	314
3.1.2	整数の形式	316
<b>第4章</b>	<b>命 令</b>	<b>317</b>
4.1	命令フォーマット	317
4.2	浮動小数点演算命令の概要	318
4.3	比較命令のための条件	320
4.4	命令セット	322

ABSF.D .....	325
ABSF.S .....	326
ADDF.D .....	327
ADDF.S .....	328
CEILF.DL .....	329
CEILF.DUL .....	330
CEILF.DUW .....	331
CEILF.DW .....	332
CEILF.SL .....	333
CEILF.SUL .....	334
CEILF.SUW .....	335
CEILF.SW .....	336
CMOVF.D .....	337
CMOVF.S .....	338
CMPF.D .....	339
CMPF.S .....	342
CVTF.DL .....	345
CVTF.DS .....	346
CVTF.DUL .....	347
CVTF.DUW .....	348
CVTF.DW .....	349
CVTF.LD .....	350
CVTF.LS .....	351
CVTF.SD .....	352
CVTF.SL .....	353
CVTF.SUL .....	354
CVTF.SUW .....	355
CVTF.SW .....	356
CVTF.ULD .....	357
CVTF.ULS .....	358
CVTF.UWD .....	359
CVTF.UWS .....	360
CVTF.WD .....	361
CVTF.WS .....	362
DIVF.D .....	363
DIVF.S .....	364
FLOORF.DL .....	365
FLOORF.DUL .....	366
FLOORF.DUW .....	367
FLOORF.DW .....	368
FLOORF.SL .....	369
FLOORF.SUL .....	370
FLOORF.SUW .....	371
FLOORF.SW .....	372
MADDF.S .....	373
MAXF.D .....	375
MAXF.S .....	376
MINF.D .....	377
MINF.S .....	378
MSUBF.S .....	379
MULF.D .....	381
MULF.S .....	382

NEGF.D.....	383
NEGF.S.....	384
NMADDF.S.....	385
NMSUBF.S.....	387
RECIPF.D.....	389
RECIPF.S.....	390
RSQRTF.D.....	391
RSQRTF.S.....	392
SQRTF.D.....	393
SQRTF.S.....	394
SUBF.D.....	395
SUBF.S.....	396
TRFSR.....	397
TRNCF.DL.....	398
TRNCF.DUL.....	399
TRNCF.DUW.....	400
TRNCF.DW.....	401
TRNCF.SL.....	402
TRNCF.SUL.....	403
TRNCF.SUW.....	404
TRNCF.SW.....	405
<b>第5章 浮動小数点演算例外.....</b>	<b>406</b>
5.1 例外の種類.....	406
5.2 例外処理.....	407
5.2.1 ステータス・フラグ.....	407
5.3 例外の詳細.....	408
5.3.1 不正確演算例外 (I).....	408
5.3.2 無効演算例外 (V).....	409
5.3.3 ゼロ除算例外 (Z).....	409
5.3.4 オーパフロー例外 (O).....	410
5.3.5 アンダフロー例外 (U).....	410
5.3.6 未実装演算例外 (E).....	411
5.4 プレサイス例外とインプレサイス例外.....	412
5.4.1 プレサイス例外.....	412
5.4.2 インプレサイス例外.....	412
5.5 状態の退避と復帰.....	413
5.6 浮動小数点演算モデルの選択.....	415
5.6.1 正確な演算を必要とする場合.....	415
5.6.2 演算性能を優先する場合.....	416
<b>付録A 命令一覧.....</b>	<b>417</b>
A.1 基本命令.....	417
A.2 浮動小数点演算命令.....	421
<b>付録B 命令オペコード一覧.....</b>	<b>423</b>
B.1 基本命令オペコード一覧.....	423
B.2 浮動小数点演算命令オペコード一覧.....	428

付録C	パイプライン	430
C.1	特    徴	432
C.2	命令実行クロック数	434
C.2.1	基本命令の実行クロック数	434
C.2.2	浮動小数点演算命令の命令実行クロック数	439
C.3	基本命令のパイプライン	442
C.3.1	ロード命令	442
C.3.2	ストア命令	443
C.3.3	乗算命令	443
C.3.4	加算付き乗算命令	444
C.3.5	算術演算命令	444
C.3.6	条件付き演算命令	445
C.3.7	飽和演算命令	445
C.3.8	論理演算命令	446
C.3.9	データ操作命令	446
C.3.10	ビット・サーチ命令	447
C.3.11	除算命令	447
C.3.12	高速除算命令	448
C.3.13	分岐命令	449
C.3.14	ビット操作命令	451
C.3.15	特殊命令	452
付録D	周辺装置保護領域一覧	458
付録E	V850E2M CPUと他のCPUの相違点	463
E.1	V850E1, V850E2との相違点	463
付録F	命令索引	465
F.1	基本命令索引	465
F.2	浮動小数点演算命令索引	466

## 第 1 編 概 要



# 第1章 特 徴

V850E2M CPU は V850E2v3 アーキテクチャに準拠し、高性能、高機能、高信頼性をコンセプトに設計された、組み込みシステムにおける機器制御用マイコン向け CPU です。

V850E2M CPU は、次のような機能を提供します。

- (1) 基本機能
- (2) プロセッサ保護機能
- (3) 浮動小数点演算機能

V850E2M CPU は、7 段パイプライン制御によりアドレス計算、算術論理演算、データ転送などのほとんどの命令処理を 1 クロックで実行します。

また、V850E2M CPU は、V850 CPU, V850E1 CPU, V850E2 CPU に対して、オブジェクト・コード・レベルでの上位互換性を持たせているため、従来のシステムのソフトウェア資産をそのまま使用できます。

## 1.1 基本機能

一般的なデータ処理 / 制御プログラミングを可能にする基本的な整数演算命令、アプリケーション・プログラム最適化のための特殊命令を備えています。また、高信頼プログラミングを可能にする柔軟な例外処理機能や、マルチ・コア環境でのデータ共有を可能にするための排他制御命令、ディスプレイメント範囲を拡張したロード / ストア命令を備えています。

基本機能は主に、命令キュー、プログラム・カウンタ、実行ユニット、汎用レジスタ、システム・レジスタとその制御部から構成されています。実行ユニットは、ALU, LD/STユニット、乗算器 (32ビット×32ビット乗算)、パレル・シフタ (32ビット / 1クロック)、除算器などの専用ハードウェア内蔵し、複雑な処理を高速で実行できます。

基本機能の詳細は第2編 **基本機能**を参照してください。

## 1.2 プロセッサ保護機能

プログラムごとのメモリ，周辺装置，システム・レジスタ，CPU時間などのリソースを保護し，不正な使用からシステムを守るためのプロセッサ保護機能を備えています。

プロセッサ保護機能は，システム・レジスタ保護，メモリ保護，周辺装置保護，タイミング監視によって構成されるCPUの高信頼動作を保証するための機能です。

プロセッサ保護機能の詳細は**第3編 プロセッサ保護機能**を参照してください。

## 1.3 浮動小数点演算機能

V850E2M CPUでは，浮動小数点演算機能（FPU）をコプロセッサとして搭載可能です。

V850E2M CPUの浮動小数点演算機能（FPU）は，ANSI/IEEE標準規格754-1985「IEEE 2進浮動小数点演算規格」に準拠しており，倍精度 / 単精度浮動小数点演算命令が使用可能です。

また状況にあわせて高いスループットで実行することが可能な高性能演算と，正確な例外処理を行うことが可能な高信頼演算を切り替えることが可能です。

浮動小数点演算機能（FPU）の搭載 / 非搭載は製品によって異なります。詳細は，各製品のユーザーズ・マニュアルを参照してください。

浮動小数点機能の詳細は**第4編 浮動小数点機能**を参照してください。

## 第 2 編 基本機能

# 第1章 概 説

V850E2M CPU は V850E2v3 アーキテクチャに準拠し、OS やアプリケーション・プログラムを成立させるための基本的な演算操作、各種例外の管理などを行うための基本機能を提供します。

## (1) 整数演算命令群

一般的なデータ処理 / 制御プログラミングを可能にする基本的な整数演算命令を備えています。また、従来のロード / ストア命令を拡張し、23ビット・ディスプレイメント形式を追加しています。

## (2) 特殊命令群

スタック・フレーム操作命令や、共用関数呼び出し用命令など、アプリケーション・プログラムの最適化に有用な命令を備えています。

## (3) マルチコア対応

複数CPU間での排他制御を行うために必要な排他制御命令や、メモリ同期化命令を備えています。

## (4) 高機能OS支援

高機能OSの開発を支援するために特化した命令を備えています。

## (5) 柔軟で高性能な例外処理

高信頼プログラミングを可能にする様々な例外処理機能を備えています。

## 1.1 特 徴

### (1) 組み込み制御用高性能32ビット・アーキテクチャ

- 命令数：98
- 32ビット汎用レジスタ：32本
- 複数のディスプレースメント形式を持つロード/ストア命令
  - ロング (23ビット)
  - ミドル (16ビット)
  - ショート (8ビット)
- 3オペランド命令
- アドレス空間：プログラム領域 ... 4 Gバイト・リニア  
データ領域 ... 4 Gバイト・リニア

### (2) 各種応用分野に適した命令群

- 飽和演算命令
- ビット操作命令
- 乗算命令 (ハードウェア乗算器内蔵により、1クロックでの乗算処理が可能)
  - 16ビット×16ビット → 32ビット
  - 32ビット×32ビット → 32ビット、または64ビット
- MAC演算命令
  - 32ビット×32ビット + 64ビット → 64ビット
- 高速除算命令
  - 有効なビット長を検出して、必要最小の実行サイクル数に変化する除算命令です。
  - 32ビット ÷ 32ビット 32ビット (商)、32ビット (剰余)

### (3) 高機能/高性能プログラミングに適した命令群

- スタック・フレーム操作命令
- 排他制御命令
- システム・コール命令 (OSサービス呼び出し命令)
- 同期命令 (イベント制御)

## 第2章 レジスタ・セット

基本機能に関わるレジスタは、一般のプログラム用として使用するプログラム・レジスタと、実行環境の制御をするシステム・レジスタの2つに分類できます。すべて32ビット・レジスタです。

図2-1 レジスタ一覧



## 2.1 プログラム・レジスタ

プログラム・レジスタには、汎用レジスタ（r0-r31）とプログラム・カウンタ（PC）があります。

表2-1 プログラム・レジスタ一覧

プログラム・レジスタ	名称	機能	説明
汎用レジスタ	r0	ゼロ・レジスタ	常に0を保持
	r1	アセンブラ予約レジスタ	アドレス生成用のワーキング・レジスタとして使用
	r2	アドレス/データ変数用レジスタ(使用するリアルタイムOSがこのレジスタを使用していない場合)	
	r3	スタック・ポインタ (SP)	関数コール時のスタック・フレーム生成時に使用
	r4	グローバル・ポインタ (GP)	データ領域のグローバル変数をアクセスするときに使用
	r5	テキスト・ポインタ (TP)	テキスト領域 (プログラム・コードを配置する領域) の先頭を示すレジスタとして使用
	r6-r29	アドレス/データ変数用レジスタ	
	r30	エレメント・ポインタ (EP)	メモリをアクセスするときのアドレス生成用ベース・ポインタとして使用
	r31	リンク・ポインタ (LP)	コンパイラが関数コールをするときに使用
プログラム・カウンタ	PC	プログラム実行中の命令アドレスを保持	

**備考** アセンブラやCコンパイラで使用される r1, r3-r5, r31 の詳細な説明は、それぞれのソフトウェア開発環境のドキュメントを参照してください。

### (1) 汎用レジスタ (r0-r31)

汎用レジスタとして、r0-r31 の32本が用意されています。これらのレジスタは、すべてデータ変数用またはアドレス変数用として利用できます。

ただし、r0-r5, r30, r31は、ソフトウェア開発環境において特殊な用途に用いられることを想定しているため、使用する際には次のような注意が必要です。

#### (a) r0, r3, r30

命令により暗黙的に使用されます。

r0は常に0を保持しているレジスタであり、0を使用する演算やベース・アドレスが0のアドレッシングで使用されます。

r3はPREPARE命令、DISPOSE命令により、暗黙的に使用されます。

r30はSLD命令とSST命令により、メモリをアクセスするときのベース・ポインタとして使用されます。

#### (b) r1, r4, r5, r31

アセンブラとCコンパイラにより暗黙的に使用されます。

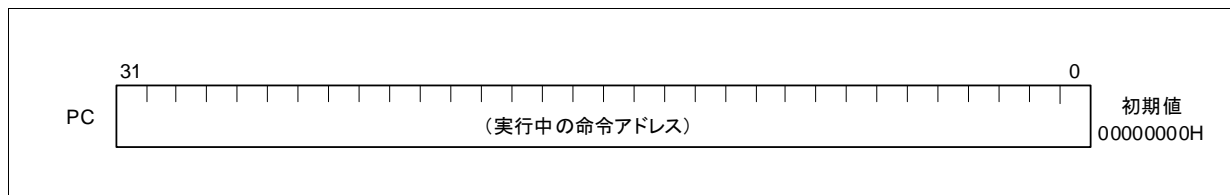
これらのレジスタを使用する際には、レジスタの内容を破壊しないように退避してから使用し、使用後に元に戻す必要があります。

#### (c) r2

リアルタイムOSが使用する場合があります。使用するリアルタイムOSがr2を使用していない場合は、アドレス変数用またはデータ変数用レジスタとして利用できます。

**(2) プログラム・カウンタ (PC)**

プログラム実行中の命令アドレスを保持しています。また、ビット0は0に固定されており、奇数番地への分岐はできません。



**注意** 製品仕様により、命令アドレッシング範囲が512 Mバイトに制限されたCPUでは、ビット31-29はビット28を符号拡張した値が自動的に設定されます。



## 2.2 システム・レジスタ・バンク

V850E2M CPUのシステム・レジスタは、システム・レジスタ・バンク上に用意されています。機能ごとに分類されたシステム・レジスタ群を「グループ」と定義し、さらに細かく用途ごとに分類したものを「バンク」と定義します。各バンクには、0から27まで最大28本のシステム・レジスタが定義可能です。

V850E2M CPUには次のようなグループとバンクがあります。

- CPU機能グループ
  - ・基本バンク：従来のシステム・レジスタ群です。
  - ・例外ハンドラ切り替え機能バンク0：例外ハンドラ切り替えを行うシステム・レジスタ群です。
  - ・例外ハンドラ切り替え機能バンク1：例外ハンドラ切り替えを行うシステム・レジスタ群です。
- プロセッサ保護機能グループ
  - ・プロセッサ保護違反バンク：プロセッサ保護違反に関するシステム・レジスタ群です。
  - ・プロセッサ保護設定バンク：プロセッサ保護機能に関するシステム・レジスタ群です。
  - ・ソフトウェア・ページング・バンク：メモリ保護をページング方式で利用する場合に使用するシステム・レジスタ群です。
- PMU機能グループ
  - ・PMU機能バンク：パフォーマンス測定機能を設定するシステム・レジスタ群です<sup>注</sup>。
- FPU機能グループ
  - ・FPUステータス・バンク：浮動小数点演算に関するシステム・レジスタ群です。
- ユーザ・グループ
  - ・ユーザ0バンク：ユーザ・アプリケーションで使用されるシステム・レジスタだけにアクセスできるバンクです。
  - ・ユーザ互換バンク：互換性のために、ユーザ・アプリケーションで使用されるシステム・レジスタに加えて、例外関連のシステム・レジスタにもアクセスできるバンクです。

**注** PMU機能バンクは開発ツール向けのデバッグ機能です。

図2-3 システム・レジスタ・バンク

CPU機能グループ			プロセッサ保護機能グループ			PMU機能グループ	FPU機能グループ	ユーザ・グループ	
基本バンク									
システム・レジスタ00									
システム・レジスタ01									
システム・レジスタ02									
システム・レジスタ03									
システム・レジスタ04	例外ハンドラ切り替え機能バンク0	例外ハンドラ切り替え機能バンク1	プロセッサ保護違反バンク	プロセッサ保護設定バンク	ソフトウェア・ページング・バンク	注 PMU機能バンク	FPUステータス・バンク	ユーザ0バンク	ユーザ互換バンク
システム・レジスタ05									
システム・レジスタ06									
システム・レジスタ07									
...									
システム・レジスタ21									
システム・レジスタ22									
システム・レジスタ23									
システム・レジスタ24									
システム・レジスタ25									
システム・レジスタ26									
システム・レジスタ27									
システム・レジスタ28 (EIWR - EIレベル用作業レジスタ)									
システム・レジスタ29 (FEWR - FEレベル用作業レジスタ)									
システム・レジスタ30 (DBWR - DBレベル用作業レジスタ)									
システム・レジスタ31 (BSEL - レジスタ・バンクの選択)									

BSELレジスタの設定により、バンクが選択されるコアクセスできるシステム・レジスタ

BSELレジスタの設定にかかわらず、常にコアクセスできるシステム・レジスタ

注 開発ツール向けのデバッグ機能です。

## 2.2.1 BSEL - レジスタ・バンクの選択

レジスタ・バンク選択レジスタは、LDSR命令およびSTSR命令でアクセスされるシステム・レジスタ群を選択します。BSELレジスタは、どのバンクが選択されているときでも常に参照可能です。

ビット31-16には必ず0を設定してください。

31	16 15	8 7	0
BSEL			
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
GRP		BNK	
初期値 00000000H			

ビット位置	ビット名	意味																															
15-8	GRP	システム・レジスタ・バンクのグループ番号を指定します（初期値：0）。 00H：CPU機能グループ 10H：プロセッサ保護機能グループ 11H：PMU機能グループ <sup>注</sup> 20H：FPU機能グループ FFH：ユーザ・グループ 上記以外の設定は禁止です。																															
7-0	BNK	システム・レジスタ・バンクのバンク番号を指定します（初期値：0）。 <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>GRP</th> <th>BNK</th> <th>対応する実行レベル</th> </tr> </thead> <tbody> <tr> <td rowspan="3" style="text-align: center;">00H</td> <td style="text-align: center;">00H</td> <td>基本バンク</td> </tr> <tr> <td style="text-align: center;">10H</td> <td>例外ハンドラ切り替え機能バンク0</td> </tr> <tr> <td style="text-align: center;">11H</td> <td>例外ハンドラ切り替え機能バンク1</td> </tr> <tr> <td rowspan="3" style="text-align: center;">10H</td> <td style="text-align: center;">00H</td> <td>プロセッサ保護違反バンク</td> </tr> <tr> <td style="text-align: center;">01H</td> <td>プロセッサ保護設定バンク</td> </tr> <tr> <td style="text-align: center;">10H</td> <td>ソフトウェア・ページング・バンク</td> </tr> <tr> <td style="text-align: center;">11H</td> <td style="text-align: center;">00H</td> <td>PMU機能バンク<sup>注</sup></td> </tr> <tr> <td style="text-align: center;">20H</td> <td style="text-align: center;">00H</td> <td>FPUステータス・バンク</td> </tr> <tr> <td rowspan="2" style="text-align: center;">FFH</td> <td style="text-align: center;">00H</td> <td>ユーザ0バンク</td> </tr> <tr> <td style="text-align: center;">FFH</td> <td>ユーザ互換バンク</td> </tr> <tr> <td colspan="2" style="text-align: center;">上記以外</td> <td style="text-align: center;">設定禁止</td> </tr> </tbody> </table>	GRP	BNK	対応する実行レベル	00H	00H	基本バンク	10H	例外ハンドラ切り替え機能バンク0	11H	例外ハンドラ切り替え機能バンク1	10H	00H	プロセッサ保護違反バンク	01H	プロセッサ保護設定バンク	10H	ソフトウェア・ページング・バンク	11H	00H	PMU機能バンク <sup>注</sup>	20H	00H	FPUステータス・バンク	FFH	00H	ユーザ0バンク	FFH	ユーザ互換バンク	上記以外		設定禁止
GRP	BNK	対応する実行レベル																															
00H	00H	基本バンク																															
	10H	例外ハンドラ切り替え機能バンク0																															
	11H	例外ハンドラ切り替え機能バンク1																															
10H	00H	プロセッサ保護違反バンク																															
	01H	プロセッサ保護設定バンク																															
	10H	ソフトウェア・ページング・バンク																															
11H	00H	PMU機能バンク <sup>注</sup>																															
20H	00H	FPUステータス・バンク																															
FFH	00H	ユーザ0バンク																															
	FFH	ユーザ互換バンク																															
上記以外		設定禁止																															

注 PMU機能バンクは開発ツール向けのデバッグ機能です。

第2編では、CPU機能グループのシステム・レジスタ、ユーザ・バンクについて説明します。プロセッサ保護機能グループのシステム・レジスタについては第3編 **プロセッサ保護機能**を、FPU機能グループのシステム・レジスタについては第4編 **浮動小数点演算機能**をそれぞれ参照してください。

## 2.3 CPU機能グループ/基本バンク

基本バンクのシステム・レジスタは、CPUの状態制御、例外情報保持などを行います。

システム・レジスタへのリード/ライトは、LDSR命令、STSR命令により、次に示すシステム・レジスタ番号を指定することで行います。

表2-2 システム・レジスタ一覧(基本バンク)

システム・レジスタ番号	名称	機能	オペランド指定の可否		システム・レジスタ保護
			LDSR命令	STSR命令	
0	EIPC	EIレベル例外受け付け時の状態退避レジスタ			
1	EIPSW	EIレベル例外受け付け時の状態退避レジスタ			
2	FEPC	FEレベル例外受け付け時の状態退避レジスタ			
3	FEPSW	FEレベル例外受け付け時の状態退避レジスタ			
4	ECR	例外要因	x		
5	PSW	プログラム・ステータス・ワード			注1
6-10		(将来の機能拡張のための予約番号 (アクセスした場合の動作は保証しません))	x	x	
11	SCCFG	SYSCALの動作設定			
12	SCBP	SYSCALLベース・ポインタ			
13	EIIC	EIレベル例外要因			
14	FEIC	FEレベル例外要因			
15	DBIC <sup>注2</sup>	DBレベル例外要因	-	-	-
16	CTPC	CALLT実行時の状態退避レジスタ			
17	CTPSW	CALLT実行時の状態退避レジスタ			
18	DBPC <sup>注2</sup>	DBレベル例外受け付け時の状態退避レジスタ	-	-	-
19	DBPSW <sup>注2</sup>	DBレベル例外受け付け時の状態退避レジスタ	-	-	-
20	CTBP	CALLTベース・ポインタ			x
21	DIR <sup>注2</sup>	デバッグ・インタフェース・レジスタ	-	-	-
22-27		デバッグ機能レジスタ	-	-	-
28	EIWR	EIレベル例外用作業レジスタ			
29	FEWR	FEレベル例外用作業レジスタ			
30	DBWR <sup>注2</sup>	DBレベル例外用作業レジスタ			
31	BSEL	レジスタ・バンクの選択			

注1. ビット31-6のみ保護。保護されている場合に書き込みがあっても、システム・レジスタ保護違反として検出されません。詳細は、第3編 5章 システム・レジスタ保護を参照してください。

2. 開発ツール向けのデバッグ機能のレジスタです。

備考 : オペランド指定の可否の欄では指定可能であることを示します。システム・レジスタ保護の欄では、保護対象であることを示します。

x : オペランド指定の可否の欄では指定不可能であることを示します。システム・レジスタ保護の欄では、保護対象ではないことを示します。

### 2.3.1 EIPC, EIPSW - EIレベル例外受け付け時の状態退避レジスタ

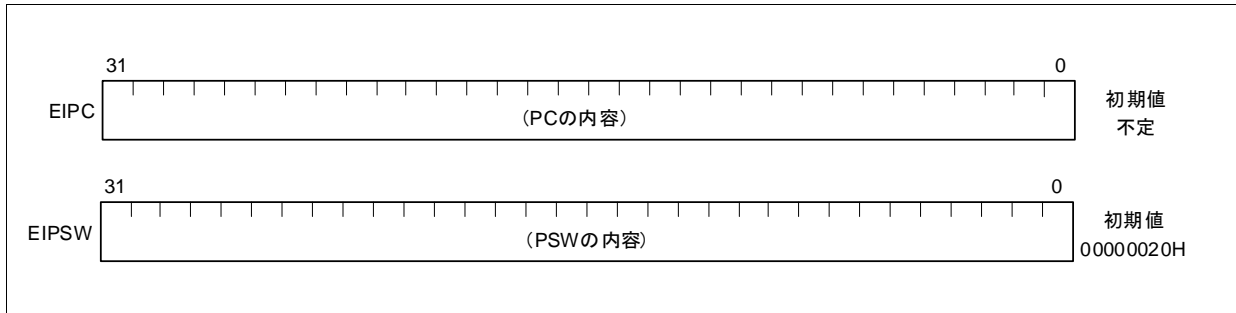
EIレベル例外時状態退避レジスタには、EIPCとEIPSWがあります。

EIレベル例外（EIレベル・ソフトウェア例外やEIレベル割り込み（INT）など）が発生した場合、EIPCには、EIレベル例外が発生したときに実行していた命令、あるいはその次の命令のアドレスが退避されます（表6-1 例外要因一覧参照）。EIPSWには、現在のPSWの内容が退避されます。

EIレベル例外時状態退避レジスタは、1組であるため、多重例外処理を行う場合はプログラムによってこれらのレジスタの内容を退避する必要があります。

EIPCレジスタには必ず偶数番地を設定してください。奇数番地の指定はできません。

なお、PSWで「0を設定してください」とされているビットは、EIPSWでも必ず0を設定してください。



**注意** 製品仕様により、命令アドレッシング範囲が512 Mバイトに制限されたCPUでは、EIPCのビット31-29はビット28を符号拡張した値が自動的に設定されます。

### 2.3.2 FEPC, FEPSW - FEレベル例外受け付け時の状態退避レジスタ

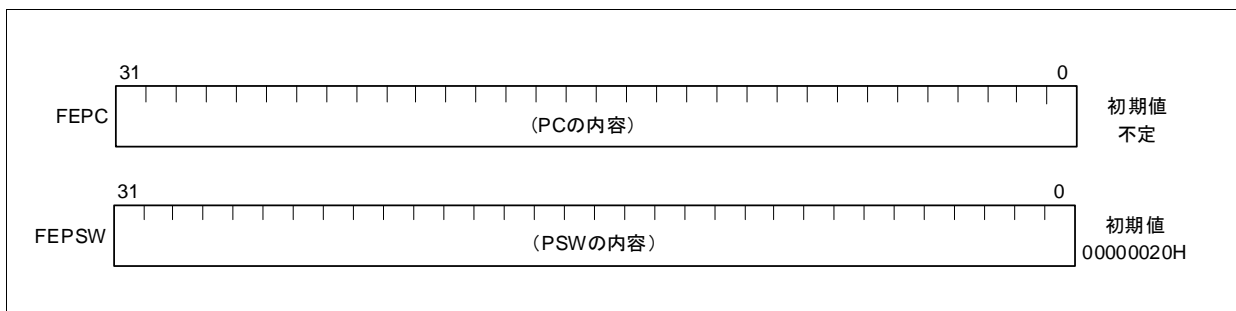
FEレベル例外時状態退避レジスタには、FEPCとFEPSWがあります。

FEレベル例外（FEレベル・ソフトウェア例外やFEレベル割り込み（FEINT, FENMI）など）が発生した場合、FEPCには、FEレベル例外が発生したときに実行していた命令、あるいはその次の命令のアドレスが退避されます（表6-1 例外要因一覧参照）。FEPSWには、現在のPSWの内容が退避されます。

FEレベル例外時状態退避レジスタは、1組であるため、多重例外処理を行う場合はプログラムによってこれらのレジスタの内容を退避する必要があります。

FEPCレジスタには必ず偶数番地を設定してください。奇数番地の指定はできません。

なお、PSWで「0を設定してください」とされているビットは、FEPSWでも必ず0を設定してください。



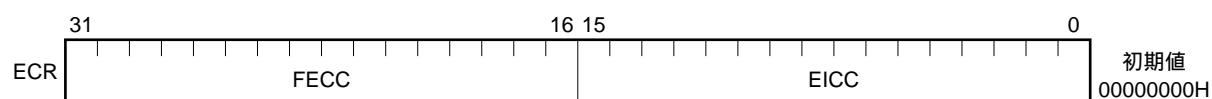
**注意** 製品仕様により、命令アドレッシング範囲が512 Mバイトに制限されたCPUでは、FEPCのビット31-29はビット28を符号拡張した値が自動的に設定されます。

### 2.3.3 ECR - 例外要因

ECRレジスタは、例外が発生した場合に、その要因を保持するレジスタです。ECRが保持する値は、例外要因ごとにコード化された例外要因コードです（表6-1 例外要因一覧参照）。なお、このレジスタは読み出し専用のため、LDSR命令を使ってこのレジスタにデータを書き込むことはできません。

**注意** ECR レジスタは V850E1, V850E2 CPU コア上位互換のための機能であり、原則として使用を禁止しています。

修正の不可能な既存プログラム以外では、ECR レジスタを使用していた部分すべてを、EIIC レジスタまたはFEIC レジスタを使用するプログラムで置き換えて使用してください。



ビット位置	ビット名	意味
31-16	FECC	FEレベル例外の例外要因コード（初期値：0）
15-0	EICC	EIレベル例外の例外要因コード（初期値：0）

### 2.3.4 PSW - プログラム・ステータス・ワード

PSW (プログラム・ステータス・ワード) は、プログラムの状態 (命令実行の結果) を示すフラグやCPUの動作状態を示すビットの集合です (フラグとは条件命令 (BcondやCMOVなど) によって参照されるPSW上のビットを示します)。

LDSR命令を使用してこのレジスタの各ビットの内容を変更した場合は、LDSR命令実行終了直後から変更内容が有効となります。

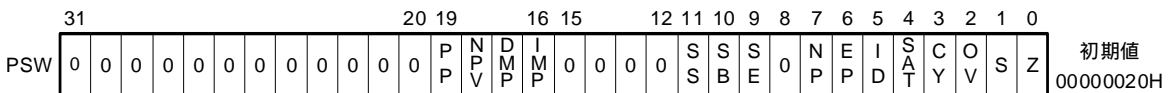
ビット31-6は、システム・レジスタ保護の対象です。システム・レジスタ保護が有効であるとき、LDSR命令を使用してビット31-6の内容を変更することはできません (第3編 第5章 システム・レジスタ保護参照)。

なお、ビット31-20, 15-12, 8は、将来の機能拡張のために予約されているため、必ず0を設定してください。また、読み出した場合の値は不定です<sup>注</sup>。

**注** ビット 19-16 は第3編 プロセッサ保護機能で使用するビットです。(第3編 プロセッサ保護機能参照)。

ビット 11-9 は開発ツール向けのデバッグ機能で使用するビットです。ユーザ・プログラムでは LDSR 命令によりこれらのビットの値を変更することはできません。

(1/3)



ビット位置	ビット / フラグ名	意味
19	PP	周辺装置保護の状態ビットです。 CPUが、現在実行中のプログラムによる周辺装置へのアクセスを信頼している状態であるかどうかを示します。 0: Tステート (CPUは、周辺装置へのアクセスを信頼しています) (初期値) 1: NTステート (CPUは、周辺装置へのアクセスを信頼していません) 周辺装置保護機能は、PPビットがTステートを示している場合、限定的なアクセス制限を行います。また、NTステートを示している場合、厳密なアクセス制限を行います。
18	NPV	システム・レジスタ保護の状態ビットです。 CPUが、現在実行中のプログラムによるシステム・レジスタへのアクセスを信頼している状態であるかどうかを示します。 0: Tステート (CPUは、システム・レジスタへのアクセスを信頼しています) (初期値) 1: NTステート (CPUは、システム・レジスタへのアクセスを信頼していません) システム・レジスタ保護機能は、NPVビットがTステートを示している場合、アクセス制限を行いません。また、NTステートを示している場合、アクセス制限を行います。
17	DMP	データ・アクセス (データ領域) に対するメモリ保護の状態ビットです。 CPUが、現在実行中のプログラムによるデータ・アクセスを信頼している状態であるかどうかを示します。 0: Tステート (CPUは、データ・アクセスを信頼しています) (初期値) 1: NTステート (CPUは、データ・アクセスを信頼していません) メモリ保護機能は、DMPビットがTステートを示している場合、データ・アクセスに対するアクセス制限を行いません。また、NTステートを示している場合、データ・アクセスに対するアクセス制限を行います。

ビット位置	フラグ名	意味
16	IMP	プログラム領域に対するメモリ保護の状態ビットです。CPUが、現在実行中のプログラムによるプログラム領域へのアクセスを信頼している状態であるかどうかを示します。 0：Tステート（CPUはプログラム領域へのアクセスを信頼しています）（初期値） 1：NTステート（CPUはプログラム領域へのアクセスを信頼していません） メモリ保護機能は、IMPビットがTステートを示している場合、プログラム領域に対するアクセス制限を行いません。また、NTステートを示している場合、プログラム領域に対するアクセス制限を行います。
11	SS	開発ツール向けのデバッグ機能で使います。
10	SB	開発ツール向けのデバッグ機能で使います。
9	SE	開発ツール向けのデバッグ機能で使います。
7	NP	FEレベル例外処理中であることを示します。FEレベル例外が受け付けられるとセット（1）され、多重例外の発生を禁止します。 0：FEレベル例外処理中でない（初期値） 1：FEレベル例外処理中である
6	EP	割り込み <sup>注1</sup> 以外の例外処理中であることを示します。該当する例外の発生でセット（1）されます。なお、このビットはセット（1）されても例外要求の受け付けには影響しません。 0：割り込みの処理中である（初期値） 1：割り込み以外の例外処理中である
5	ID	EIレベル例外処理中であることを示します。EIレベル例外が受け付けられるとセット（1）され、多重例外の発生を禁止します。また、通常のプログラムや、割り込み処理中にクリティカル・セクションとして、EIレベル例外の受け付けを禁止する場合にも使用されます。DI命令の実行によってセット（1）し、EI命令の実行によってクリア（0）します。 0：EIレベル例外処理中またはクリティカル・セクションでない（EI命令実行後） 1：EIレベル例外処理中またはクリティカル・セクションである（DI命令実行後）（初期値）
4	SAT <sup>注2</sup>	飽和演算命令の演算結果がオーバフローし、演算結果が飽和していることを示します。累積フラグのため、飽和演算命令で演算結果が飽和するとセット（1）され、以降の命令の演算結果が飽和しなくてもクリア（0）されません。クリア（0）する場合は、LDSR命令により行います。なお、算術演算命令の実行では、セット（1）もクリア（0）も行いません。 0：飽和していない（初期値） 1：飽和している
3	CY	演算結果にキャリー、またはボローがあったかどうかを示します。 0：キャリー、およびボローが発生していない（初期値） 1：キャリー、またはボローが発生した

注1. 割り込みについては、6.1.2 例外の種別を参照してください。

2. 飽和演算時のOVフラグとSフラグの内容で飽和处理した演算結果が決まります。また、飽和演算時にOVフラグがセット（1）された場合だけ、SATフラグはセット（1）されます。

演算結果の状態	フラグの状態			飽和处理をした演算結果
	SAT	OV	S	
正の最大値を越えた	1	1	0	7FFFFFFFH
負の最大値を越えた	1	1	1	80000000H
正（最大値を越えない）	演算前の値を	0	0	演算結果そのもの
負（最大値を越えない）	保持		1	



ビット位置	フラグ名	意 味
2	OV <sup>注</sup>	演算中にオーバーフローが発生したかどうかを示します。 0：オーバーフローが発生していない（初期値） 1：オーバーフローが発生した
1	S <sup>注</sup>	演算の結果が負かどうかを示します。 0：演算の結果は、正または0であった（初期値） 1：演算の結果は負であった
0	Z	演算の結果が0かどうかを示します。 0：演算の結果は0でなかった（初期値） 1：演算の結果は0であった

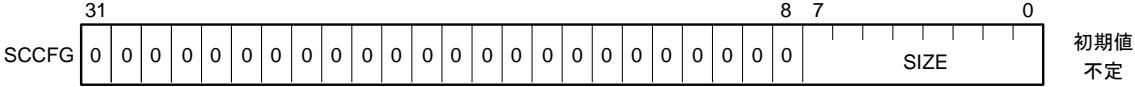
**注** 飽和演算時の OV フラグと S フラグの内容で飽和処理した演算結果が決まります。また、飽和演算時に OV フラグがセット（1）された場合だけ、SAT フラグはセット（1）されます。

演算結果の状態	フラグの状態			飽和処理をした演算結果
	SAT	OV	S	
正の最大値を越えた	1	1	0	7FFFFFFFH
負の最大値を越えた	1	1	1	80000000H
正（最大値を越えない）	演算前の値を 保持	0	0	演算結果そのもの
負（最大値を越えない）			1	

### 2.3.5 SCCFG - SYSCALLの動作設定

SYSCALL命令に関する動作設定を行います。SYSCALL命令の使用前に必ず適切な値を設定してください。  
ビット31-8には必ず0を設定してください。

**注意** SCCFG レジスタの変更を行う LDSR 命令の直後に、SYSCALL 命令を配置しないでください。



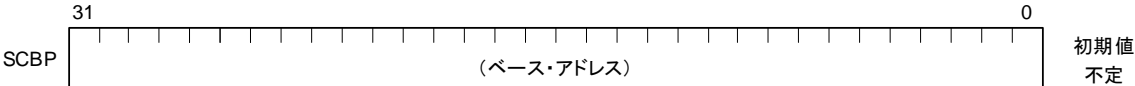
ビット位置	ビット名	意味
7-0	SIZE	SYSCALL命令が参照するテーブルの最大エントリ数を指定します。SYSCALLが参照する最大エントリ数は、SIZEが0の場合は1エントリ、255の場合は256エントリです。SYSCALL命令で分岐する関数の数に合わせて、最大エントリ数を適切に設定することで、メモリ領域を有効に活用できます。 最大エントリ数を越えるベクタがSYSCALL命令で指定された場合には、先頭のエントリが選択されます。先頭のエントリには、エラー処理ルーチンを配置してください。

### 2.3.6 SCBP - SYSCALLベース・ポインタ

SCBPレジスタは、SYSCALL命令のテーブル・アドレスの指定、ターゲット・アドレスの生成に使用されます。SYSCALL命令の使用前に、必ず適切な値を設定してください。

SCBPレジスタには必ずワード・アドレスを設定してください。

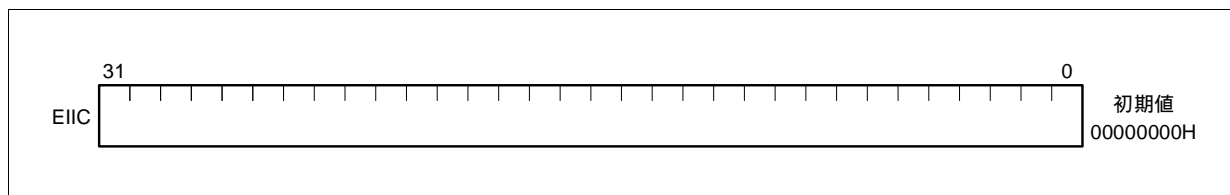
また、ビット1, 0は0に固定されています。



**注意** 製品仕様により、命令アドレッシング範囲が512 Mバイトに制限されたCPUでは、SCBPのビット31-29はビット28を符号拡張した値が自動的に設定されます。

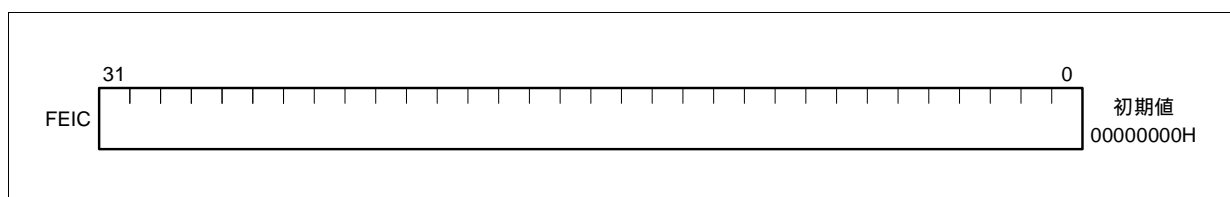
### 2.3.7 EIIC - EIレベル例外要因

EIICレジスタは、EIレベルの例外が発生した場合に、その要因を保持するレジスタです。EIICレジスタが保持する値は、例外要因ごとにコード化された例外要因コードです（表6-1 例外要因一覧参照）。



### 2.3.8 FEIC - FEレベル例外要因

FEICレジスタは、FEレベルの例外が発生した場合に、その要因を保持するレジスタです。FEICレジスタが保持する値は、例外要因ごとにコード化された例外要因コードです（表6-1 例外要因一覧参照）。



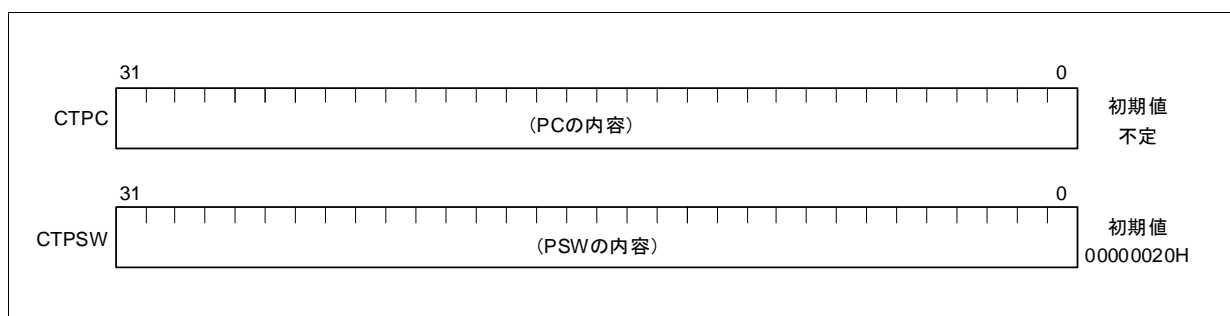
### 2.3.9 CTPC, CTPSW - CALLT実行時の状態退避レジスタ

CALLT実行時の状態退避レジスタには、CTPCとCTPSWがあります。

CALLT命令が実行されると、CALLT命令の次の命令のアドレスがCTPCに、PSW（プログラム・ステータス・ワード）の内容がCTPSWに退避されます。

CTPC レジスタのビット0は、必ず0を設定してください。

なお、PSWで「0を設定してください」とされているビットは、CTPSWでも必ず0を設定してください。



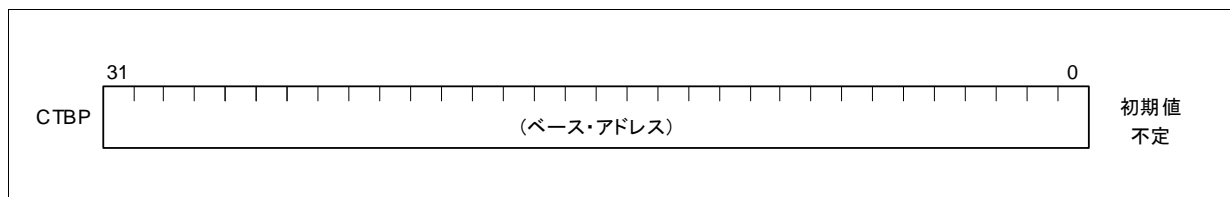
**注意** 製品仕様により、命令アドレッシング範囲が512 Mバイトに制限されたCPUでは、CTPCのビット31-29はビット28を符号拡張した値が自動的に設定されます。

### 2.3.10 CTBP - CALLTベース・ポインタ

CTBPレジスタは、CALLT命令のテーブル・アドレスの指定、ターゲット・アドレスの生成に使用されます。

CTBPレジスタには必ずハーフワード・アドレスを設定してください。

また、ビット0は、0に固定されています。

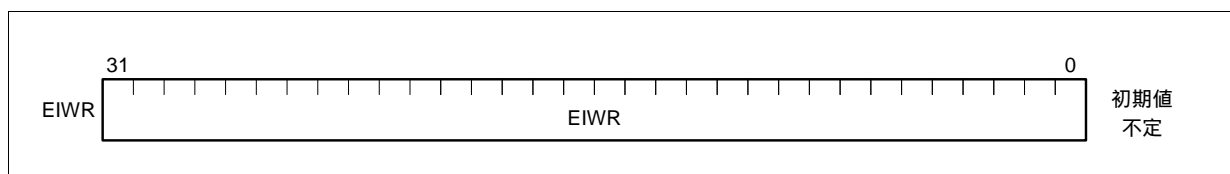


**注意** 製品仕様により、命令アドレッシング範囲が512 Mバイトに制限されたCPUでは、CTBPのビット31-29はビット28を符号拡張した値が自動的に設定されます。

### 2.3.11 EIWR - EIレベル例外用作業レジスタ

EIWRレジスタは、EIレベルの例外が発生したときの作業用レジスタです。

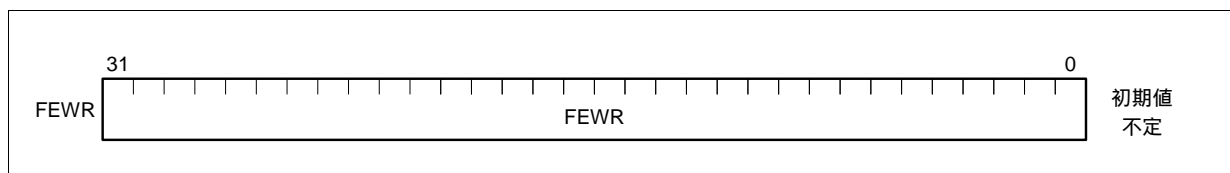
EIWRレジスタは、どのバンクが選択されているときでも常に参照可能です。



### 2.3.12 FEWR - FEレベル例外用作業レジスタ

FEWRレジスタは、FEレベルの例外が発生したときの作業用レジスタです。

FEWRレジスタは、どのバンクが選択されているときでも常に参照可能です。



### 2.3.13 DBIC - DBレベル例外要因

DBICレジスタは、デバッグ機能に関するレジスタです。  
このレジスタは開発ツール向けのデバッグ機能で使います。

### 2.3.14 DBPC, DBPSW - DBレベル例外受け付け時の状態退避レジスタ

DBレベル例外時状態退避レジスタとして、DBPCとDBPSWがあります。  
このレジスタは開発ツール向けのデバッグ機能で使います。

### 2.3.15 DBWR - DBレベル例外用作業レジスタ

DBWRレジスタは、デバッグ機能に関するレジスタです。  
このレジスタは開発ツール向けのデバッグ機能で使います。

### 2.3.16 DIR - デバッグ・インタフェース・レジスタ

DIRレジスタは、デバッグ機能の制御や状態を示します。  
DIRレジスタ、およびデバッグ機能レジスタ（システム・レジスタ22-27）は、開発ツール向けのデバッグ機能で使います。

## 2.4 CPU機能グループ / 例外ハンドラ・アドレス切り替え機能バンク

例外ハンドラ切り替え機能バンク0,1は、各LDSR命令でBSELレジスタ（2.2.1 BSEL - レジスタ・バンクの選択参照）に00000010Hおよび00000011Hを設定することにより選択されます。

システム・レジスタ番号28-31はバンク共通のシステム・レジスタで、BSELレジスタの設定値に関係なく、CPU機能バンクのEIWR, FEWR, DBWR, BSELレジスタが参照されます。

- ・ 例外ハンドラ切り替え機能バンク0  
（グループ番号00H, バンク番号10H, 略称EHSW0バンク）
- ・ 例外ハンドラ切り替え機能バンク1  
（グループ番号00H, バンク番号11H, 略称EHSW1バンク）

表2-3 システム・レジスタ・バンク

グループ	CPU機能 (00H)									
バンク	例外ハンドラ切り替え機能バンク0 (10H)				例外ハンドラ切り替え機能バンク1 (11H)					
バンク・ラベル	EHSW0				EHSW1					
レジスタ番号	名称	機能	オペランド指定の可否		システム・レジスタ保護	名称	機能	オペランド指定の可否		システム・レジスタ保護
			LDSR命令	STSR命令				LDSR命令	STSR命令	
0	SW_CTL	例外ハンドラ・アドレス切り替えの制御				機能拡張用に予約		×	×	
1	SW_CFG	例外ハンドラ・アドレス切り替え設定				EH_CFG	例外ハンドラ設定	×		
2	機能拡張用に予約		×	×		EH_RESET	リセット・アドレス・レジスタ	×		
3	SW_BASE	例外ハンドラ・アドレス切り替えベース・アドレス				EH_BASE	例外ハンドラ・ベース・アドレス	×		
4-27	機能拡張用に予約		×	×		機能拡張用に予約		×	×	
28	EIWR	EIレベル例外用作業レジスタ								
29	FEWR	FEレベル例外用作業レジスタ								
30	DBWR <sup>注</sup>	DBレベル例外用作業レジスタ								
31	BSEL	レジスタ・バンクの選択								

注 開発ツール向けのデバッグ機能のレジスタです。

備考 : オペランド指定の可否の欄では指定可能であることを示します。システム・レジスタ保護の欄では、保護対象であることを示します。

× : オペランド指定の可否の欄では指定不可能であることを示します。システム・レジスタ保護の欄では、保護対象ではないことを示します。

### 2.4.1 SW\_CTL - 例外ハンドラ・アドレス切り替えの制御

例外ハンドラ・アドレス切り替え機能の制御レジスタです。

ビット31-1には必ず0を設定してください。



### 2.4.2 SW\_CFG - 例外ハンドラ・アドレス切り替え設定

例外ハンドラ・アドレス切り替え機能を行う設定を指定するレジスタです。

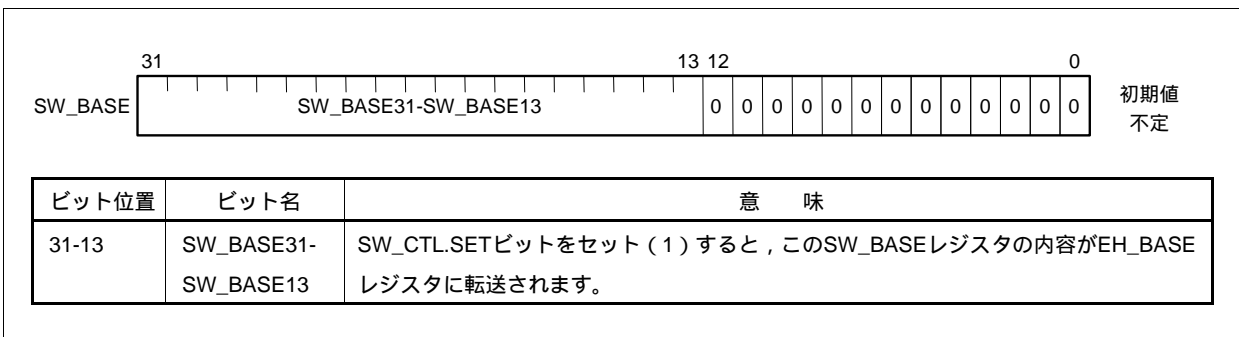
ビット31-1には必ず0を設定してください。



### 2.4.3 SW\_BASE - 例外ハンドラ・アドレス切り替えベース・アドレス

例外ハンドラ・アドレス切り替え機能の切り替えを行う例外ハンドラ・アドレスのベース・アドレスを指定するレジスタです。

ビット12-0には必ず0を設定してください。



**注意** 製品仕様により、命令アドレッシング範囲が512 Mバイトに制限されたCPUでは、SW\_BASEのビット31-29はビット28を符号拡張した値が自動的に設定されます。

## 2.4.4 EH\_CFG - 例外ハンドラ設定

例外ハンドラ・アドレス切り替え機能の現在の設定を示すレジスタです。

ビット31-1は0に固定されています。

EH_CFG	31	1 0	初期値 0000000xH
ビット位置	ビット名	意 味	
0	RINT	RINTビットがセット(1)された場合、INT0-INT255の例外ハンドラ・アドレスを1つのハンドラ・アドレス(INT0)に縮小します。クリア(0)されている場合、INT0-INT255は独立した例外ハンドラ・アドレスを持ちます。EH_CFGレジスタは製品仕様によって、リセット時に初期値が設定されます。またLDSR命令による直接書き換えは行えません。SW_CTL.SETビットをセット(1)することによって、SW_CFGの内容が転送されます。	

## 2.4.5 EH\_BASE - 例外ハンドラ・ベース・アドレス

例外ハンドラ・アドレス切り替え機能の現在の例外ハンドラ・アドレスのベース・アドレスを示すレジスタです。

ビット12-0は0に固定されています。

EH_BASE	31	13 12	0	初期値 不定
	EH_BASE31-EH_BASE13			
ビット位置	ビット名	意 味		
31-13	EH_BASE31- EH_BASE13	例外ハンドラ・ルーチンのアドレスがこのレジスタで指定されたベース・アドレスに、各例外のオフセット・アドレスを加えたアドレスに変更されます。  EH_BASEレジスタは製品仕様によって、リセット時に初期値が設定されます。またLDSR命令による直接書き換えは行えません。SW_CTL.SETビットをセット(1)することによって、SW_BASEの内容が転送されます。		

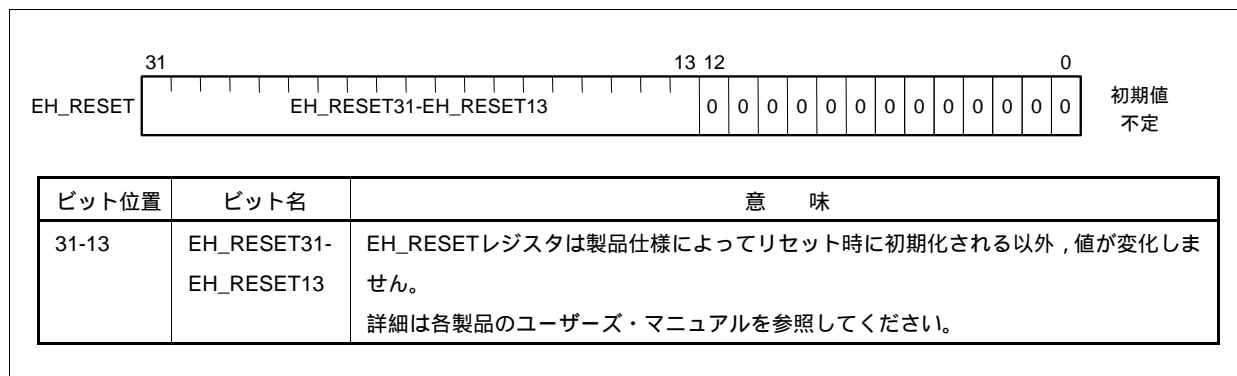
**注意** 製品仕様により、命令アドレッシング範囲が512 Mバイトに制限されたCPUでは、EH\_BASEのビット31-29はビット28を符号拡張した値が自動的に設定されます。



## 2.4.6 EH\_RESET - リセット・アドレス

現在のリセット入力時のリセット・アドレスを示します。

ビット12-0は0に固定されています。



**注意** 製品仕様により、命令アドレッシング範囲が 512 M バイトに制限された CPU では、EH\_RESET のビット 31-29 はビット 28 を符号拡張した値が自動的に設定されます。

## 2.5 ユーザ・グループ

ユーザ・グループは、LDSR命令でBSELレジスタに0000FF00Hまたは0000FFFFHを設定することにより選択されます（2.2.1 BSEL - レジスタ・バンクの選択参照）。ユーザ・グループにあるシステム・レジスタは基本バンク、FPUステータス・バンクにあるレジスタの写像となっています。

ユーザ・グループには次の2つバンクがあります。

- ・ユーザ0バンク（表2 - 4参照）
- ・ユーザ互換バンク（表2 - 5参照）

**注意** ユーザ互換バンクはV850E1, V850E2 アーキテクチャとの後方互換のために定義しており、原則として使用を禁止します。修正不可能な既存プログラムがユーザ0バンクに存在しないシステム・レジスタを操作している場合を除き、ユーザ0バンクを使用してください。

表2 - 4 システム・レジスタ一覧（ユーザ0バンク）

システム・レジスタ番号	名称	機能	オペランド指定の可否		システム・レジスタ保護
			LDSR	STSR	
0-4		（将来の機能拡張のための予約番号 （アクセスした場合の動作は保証しません））	×	×	
5	PSW	プログラム・ステータス・ワード			注1
6, 7		（将来の機能拡張のための予約番号 （アクセスした場合の動作は保証しません））	×	×	
8	FPST	浮動小数点演算のステータス			×
9	FPCC	浮動小数点演算の比較結果			×
10	FPCFG	浮動小数点機能の設定			×
11-15		（将来の機能拡張のための予約番号 （アクセスした場合の動作は保証しません））	×	×	
16	CTPC	CALLT実行時の状態回避レジスタ			
17	CTPSW	CALLT実行時の状態回避レジスタ			
18, 19		（将来の機能拡張のための予約番号 （アクセスした場合の動作は保証しません））	×	×	
20	CTBP	CALLTベース・ポインタ			×
21-27		（将来の機能拡張のための予約番号 （アクセスした場合の動作は保証しません））	×	×	
28	EIWR	EIレベル例外用作業レジスタ			
29	FEWR	FEレベル例外用作業レジスタ			
30	DBWR <sup>注2</sup>	DBレベル例外用作業レジスタ			
31	BSEL	レジスタ・バンクの選択			

注1. ビット 31-6 のみ保護

2. 開発ツール向けのデバッグ機能のレジスタです。

**備考** : オペランド指定の可否の欄では指定可能であることを示します。システム・レジスタ保護の欄では、保護対象であることを示します。

× : オペランド指定の可否の欄では指定不可能であることを示します。システム・レジスタ保護の欄では、保護対象ではないことを示します。

表2-5 システム・レジスタ一覧(ユーザ互換バンク)

システム・レジスタ番号	名称	機能	オペランド指定の可否		システム・レジスタ保護
			LDSR	STSR	
0	EIPC	EIレベル例外受け付け時の状態退避レジスタ			
1	EIPSW	EIレベル例外受け付け時の状態退避レジスタ			
2	FEPC	FEレベル例外受け付け時の状態退避レジスタ			
3	FEPSW	FEレベル例外受け付け時の状態退避レジスタ			
4	ECR	例外要因	×		
5	PSW	プログラム・ステータス・ワード			注1
6, 7		(将来の機能拡張のための予約番号 (アクセスした場合の動作は保証しません))	×	×	
8	FPST	浮動小数点演算のステータス			×
9	FPCC	浮動小数点演算の比較結果			×
10	FPCFG	浮動小数点機能の設定			×
11, 12		(将来の機能拡張のための予約番号 (アクセスした場合の動作は保証しません))	×	×	
13	EIIC	EIレベル例外要因			
14	FEIC	FEレベル例外要因			
15		(将来の機能拡張のための予約番号 (アクセスした場合の動作は保証しません))	×	×	
16	CTPC	CALLT実行時の状態退避レジスタ			
17	CTPSW	CALLT実行時の状態退避レジスタ			
18, 19		(将来の機能拡張のための予約番号 (アクセスした場合の動作は保証しません))	×	×	
20	CTBP	CALLTベース・ポインタ			×
21-27		(将来の機能拡張のための予約番号 (アクセスした場合の動作は保証しません))	×	×	
28	EIWR	EIレベル例外用作業レジスタ			
29	FEWR	FEレベル例外用作業レジスタ			
30	DBWR <sup>注2</sup>	DBレベル例外用作業レジスタ			
31	BSEL	レジスタ・バンクの選択			

注1. ビット 31-6 のみ保護

2. 開発ツール向けのデバッグ機能のレジスタです。

**備考** : オペランド指定の可否の欄では指定可能であることを示します。システム・レジスタ保護の欄では、保護対象であることを示します。

× : オペランド指定の可否の欄では指定不可能であることを示します。システム・レジスタ保護の欄では、保護対象ではないことを示します。

## 第3章 データ・タイプ

### 3.1 データ形式

V850E2M CPU では、データをリトル・エンディアン形式で取り扱います。つまり、ハーフワード、ワードではバイト0が常に最下位（最右端）バイトとなります。

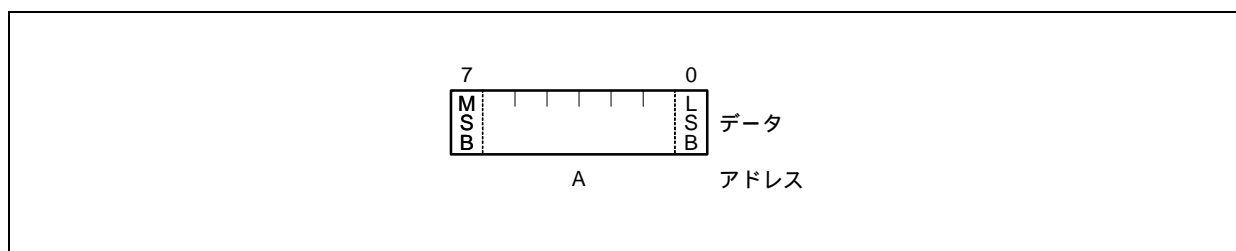
また、サポートしているデータ形式は次のとおりです。

- バイト（8ビット長データ）
- ハーフワード（16ビット長データ）
- ワード（32ビット長データ）
- ビット（1ビット長データ）

#### 3.1.1 バイト

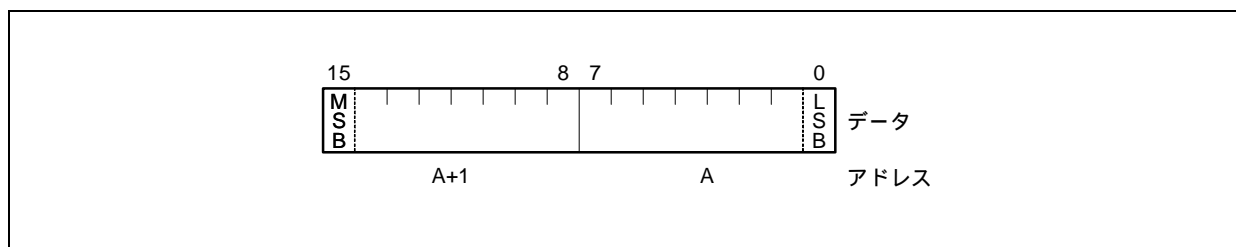
バイトは、任意のバイト境界から始まる連続した8ビットのデータです。各ビットには0から7までの番号が付けられており、LSB（Least significant bit）はビット0、MSB（Most significant bit）はビット7に対応します。

バイトは、そのアドレス「A」で指定されます。



#### 3.1.2 ハーフワード

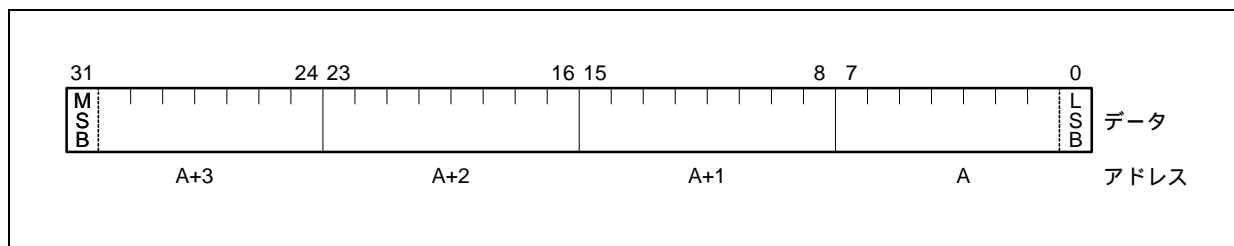
ハーフワードは任意のバイト境界<sup>※</sup>から始まる連続した2バイト（16ビット）のデータです。各ビットには、0から15までの番号が付けられており、LSBはビット0、MSBはビット15に対応します。ハーフワードはそのアドレス「A」で指定され、2つのアドレス「A」、「A+1」のバイト・データを占めます。



**注** ハーフワード・アクセスにおいても、すべてのバイト境界にアクセスできます。3.3 データ・アライメントを参照してください。

### 3.1.3 ワード

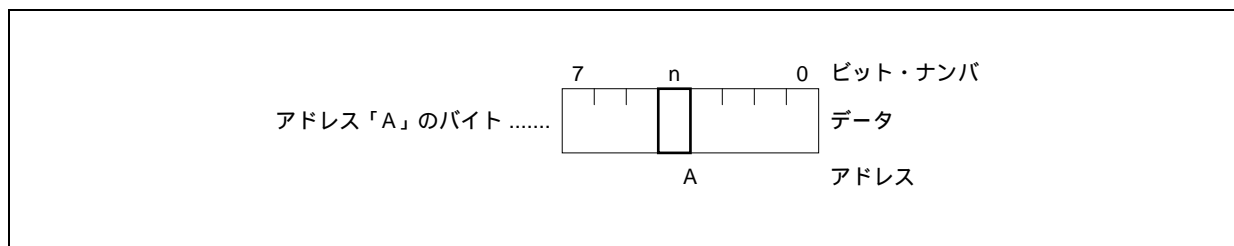
ワードは任意のバイト境界<sup>※</sup>から始まる連続した4バイト（32ビット）のデータです。各ビットには0から31までの番号が付けられており、LSBはビット0、MSBはビット31に対応します。ワードはそのアドレス「A」で指定され、4つのアドレス「A」、「A+1」、「A+2」、「A+3」のバイト・データを占めます。



**注** ワード・アクセスにおいても、すべてのバイト境界にアクセスできます。3.3 データ・アラインメントを参照してください。

### 3.1.4 ビット

ビットは、任意のバイト境界から始まる8ビット・データのnビット目の1ビット・データです。ビットはそのバイトのアドレス「A」と、ビット・ナンバ「n」で指定されます（n=0-7）。



## 3.2 データ表現

### 3.2.1 整数

整数は2の補数による2進数で表現し、32ビット、16ビット、8ビットの3通りの長さを持っています。整数の位取りは、その長さにかかわらずビット0を最下位ビットとし、ビット番号が増えるに従って位取りを高くします。2の補数表現であるため、最上位ビットを符号ビットとして使用します。

各データ長の整数の範囲は次のとおりです。

- ワード (32 ビット) : -2147483648 ~ +2147483647
- ハーフワード (16 ビット) : -32768 ~ +32767
- バイト (8 ビット) : -128 ~ +127

### 3.2.2 符号なし整数

「整数」が、正負両方の値を取るデータであるのに対して、「符号なし整数」は、負でない整数を意味します。整数と同様に、符号なし整数も2進数で表現し、32ビット、16ビット、8ビットの3通りの長さを持っています。符号なし整数の位取りは、整数と同様に、その長さにかかわらずビット0を最下位ビットとし、ビット番号が増えるに従って位取りを高くします。ただし符号ビットは存在しません。

各データ長の符号なし整数の範囲は次のとおりです。

- ワード (32 ビット) : 0 ~ 4294967295
- ハーフワード (16 ビット) : 0 ~ 65535
- バイト (8 ビット) : 0 ~ 255

### 3.2.3 ビット

ビット・データとして、クリア(0)またはセット(1)の2つの値をとる1ビットのデータを扱うことができます。ビットに関する操作は、メモリ空間の1バイト・データだけを対象とし、次の4種類の操作ができます。

- セット
- クリア
- 反転
- テスト

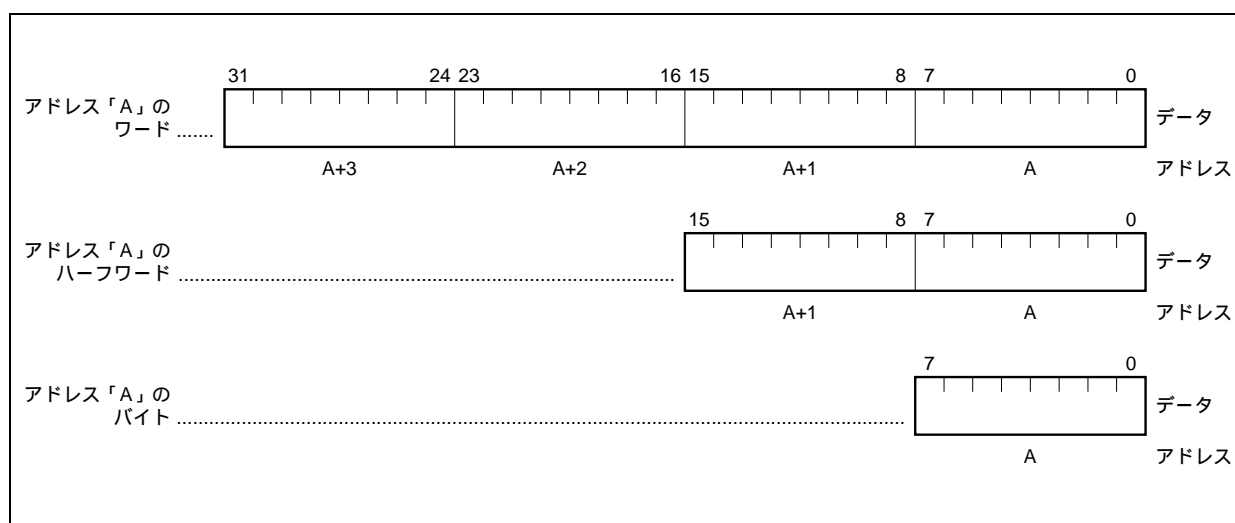


## 第4章 アドレス空間

V850E2M CPU は、4 G バイトのリニアなアドレス空間をサポートしています。このアドレス空間にはメモリと I/O の両方をマッピングします（メモリ・マップト I/O 方式）。CPU からメモリ、I/O に対して 32 ビットのアドレスが出力され、アドレス番地は最大「 $2^{32}-1$ 」となります。

各アドレスに配置されるバイト・データは、ビット 0 を LSB、ビット 7 を MSB と定義されています。また、複数バイト構成のデータでは特に注意しないかぎり、下位側アドレスのバイト・データが LSB、上位側アドレスのバイト・データが MSB を持つように定義されています（リトル・エンディアン形式）。

このマニュアルでは、複数バイト構成のデータを表現する場合、次のように右側を下位側アドレス、左側を上位側アドレスとして表現します。





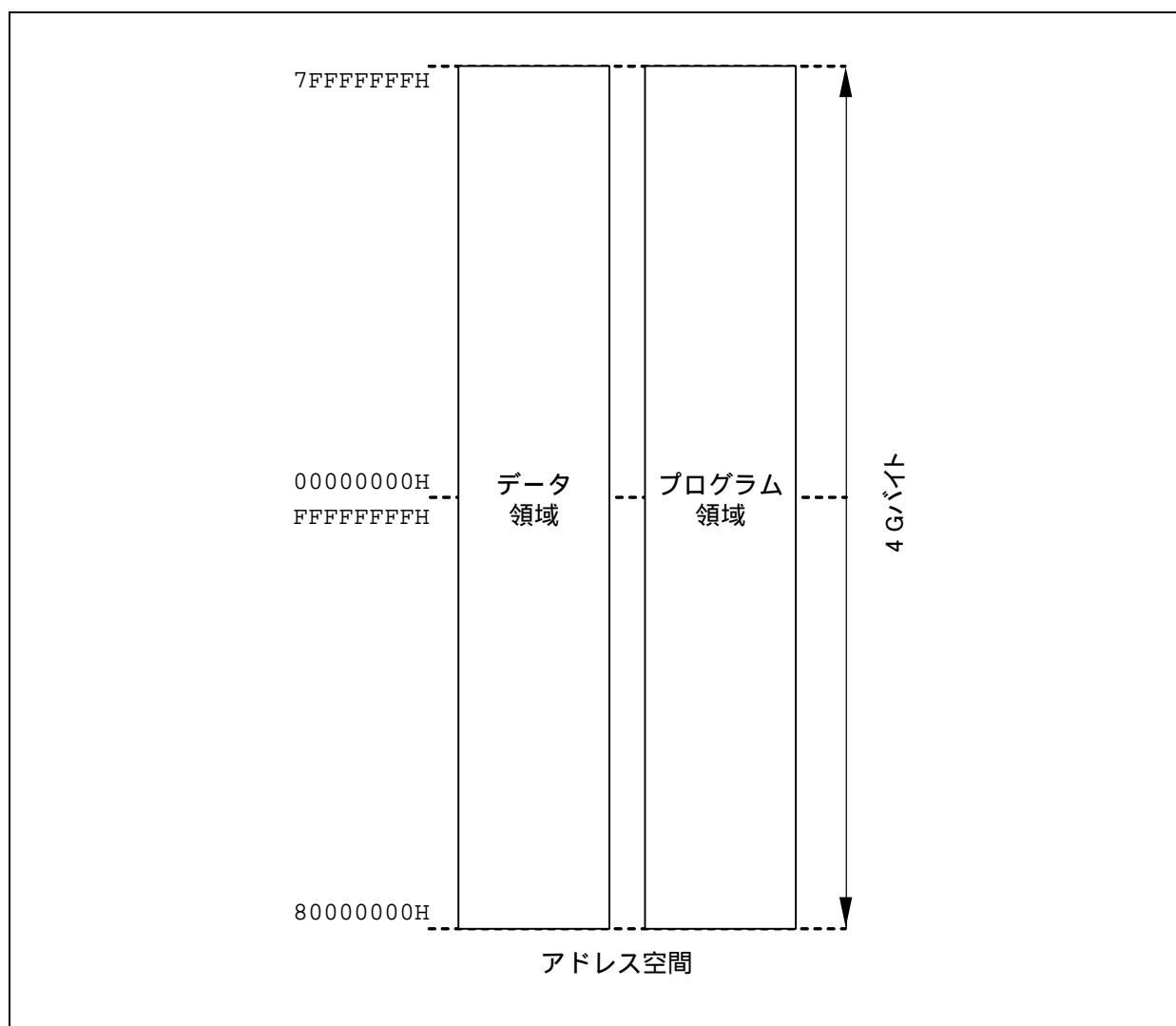
## 4.1 メモリ・マップ

V850E2M CPU は、32 ビット・アーキテクチャであり、最大 4 G バイトのリニア・アドレス空間をサポートします。命令アドレッシング（命令アクセス）、およびオペランド・アドレッシング（データ・アクセス）において、この 4 G バイトのアドレス空間内の全範囲を指定可能です。

**注意** 命令アドレッシングは、V850E1 CPU においては 64 M バイト、V850E2 CPU においては 512 M バイトの範囲内でのみ可能でした。

メモリ・マップを図 4 - 1 に示します。

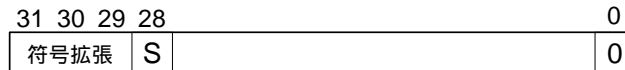
図4 - 1 メモリ・マップ（アドレス空間）



**注意** V850E2M CPUでは、命令アドレスを保持するレジスタ（プログラム・カウンタなど）の物理的な制約によって、4 Gバイトのプログラム領域のうち、実際にアドレッシング可能な範囲が限られる場合があります。命令アドレスを保持するレジスタとは、次のレジスタです。

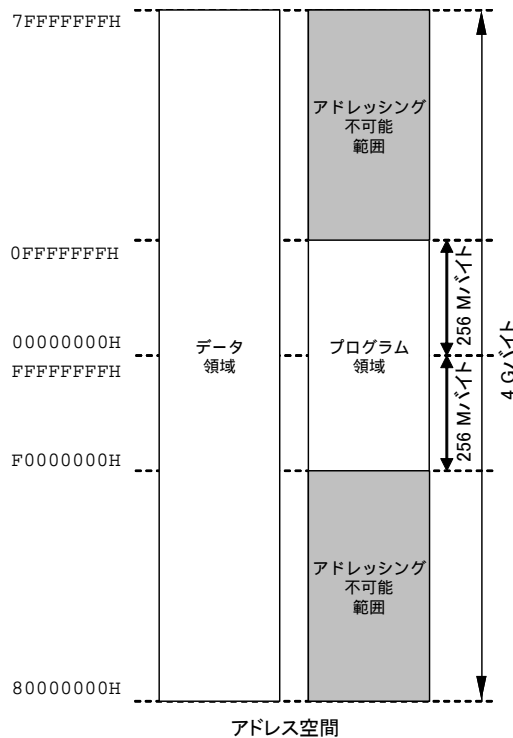
- PC（プログラム・カウンタ）
- EIPC, FEPC（例外コンテキスト）
- SCBP, CTBP, CTPC（テーブル分岐 / 例外命令）
- SW\_BASE, EH\_BASE, EH\_RESET（例外ハンドラ切り替え機能）
- FPEPC（浮動小数点演算機能）
- VSADR（プロセッサ保護機能）

たとえば、製品仕様により、プログラム領域のアドレッシング可能範囲が512 Mバイトに制限されたCPUにおいては、これらのレジスタの上位3ビットは28ビット目が符号拡張された値が自動的に設定されます。したがって、アドレッシング可能な範囲は、00000000H-0FFFFFFEHおよびF0000000H-FFFFFFEHとなります（最下位ビットは常に0）。



(512Mバイト制限時の命令アドレス・レジスタ)

この場合のメモリ・マップを次に示します。



(512 Mバイト制限時のメモリ・マップ)

512 Mバイトに制限されたメモリ・マップで用いる場合、命令およびSWITCH, CALLT, SYSCALL命令で参照するテーブルは、必ず命令アドレッシング可能なアドレス範囲に配置してください。それ以外のデータに関しては、4 Gバイト内のどこにでも配置することが可能です。

## 4.2 アドレッシング・モード

アドレス生成には、分岐をともなう命令が使用する命令アドレスと、データをアクセスする命令が使用するオペランド・アドレスの2種類があります。

### 4.2.1 命令アドレス

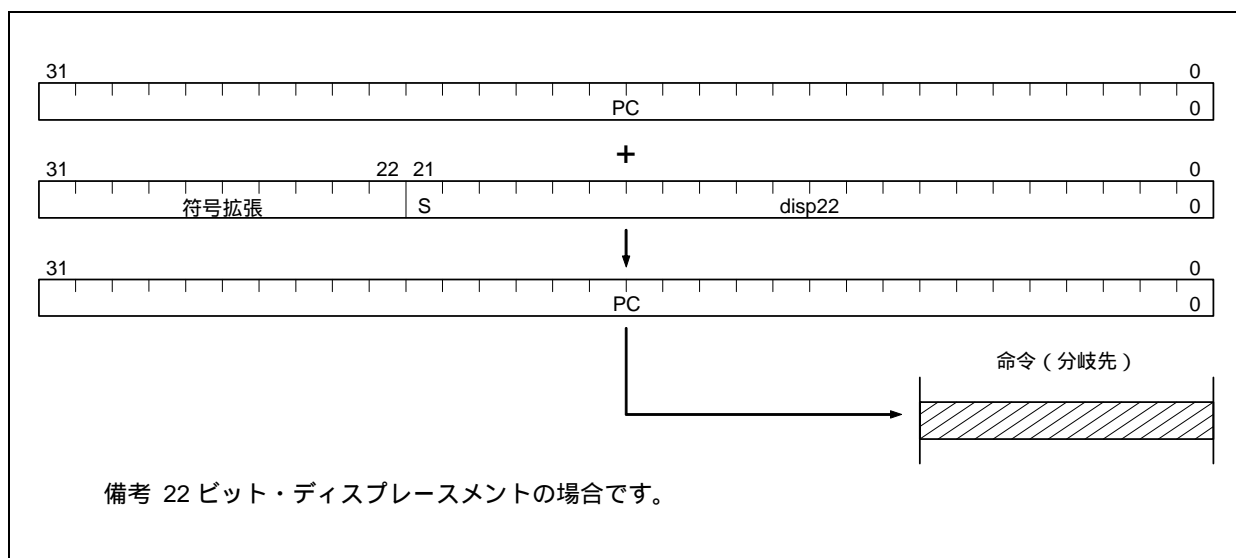
命令アドレスは、プログラム・カウンタ (PC) の内容によって決定され、実行した命令のバイト数に応じて自動的にインクリメントされます。また、分岐命令を実行する際には、次に示すアドレッシングにより、分岐先アドレスをPCにセットします。

#### (1) レラティブ・アドレッシング (PC相対)

プログラム・カウンタ (PC) に、命令コードの符号付き N ビット・データ (ディスプレースメント: disp N) を加算します。このとき、ディスプレースメントは、2 の補数データとして扱われ、それぞれ最上位ビットが符号ビット (S) となります。ディスプレースメントが 32 ビット未満の場合、上位ビットを符号拡張します (N は命令ごとに異なります)。

JARL 命令, JR 命令, Bcond 命令が、このアドレッシングの対象となります。

図4-2 レラティブ・アドレッシング

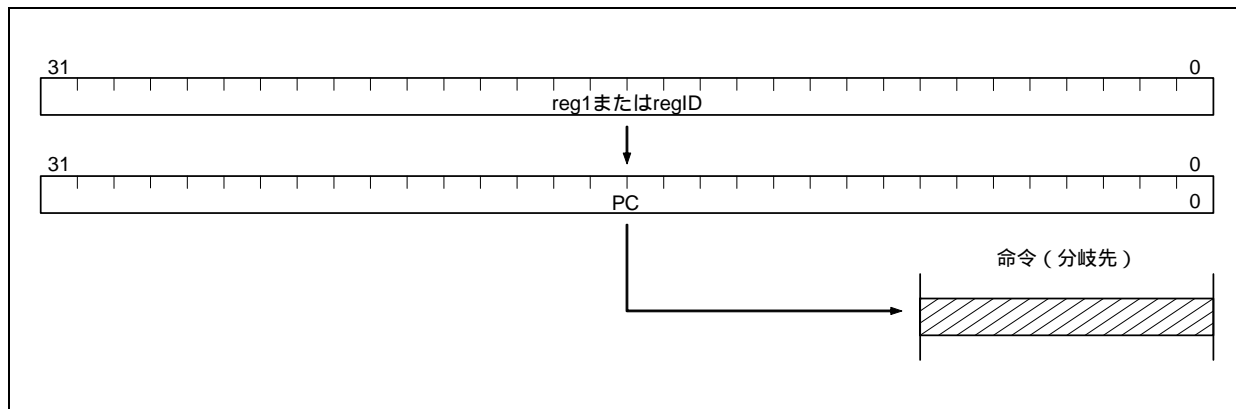


### (2) レジスタ・アドレッシング (レジスタ間接)

命令によって指定される汎用レジスタ (reg1) またはシステム・レジスタ (regID) の内容をプログラム・カウンタ (PC) に転送します。

JMP 命令, CTRET 命令, EIRET 命令, FERET 命令, RETI 命令, DISPOSE 命令が, このアドレッシングの対象となります。

図4-3 レジスタ・アドレッシング

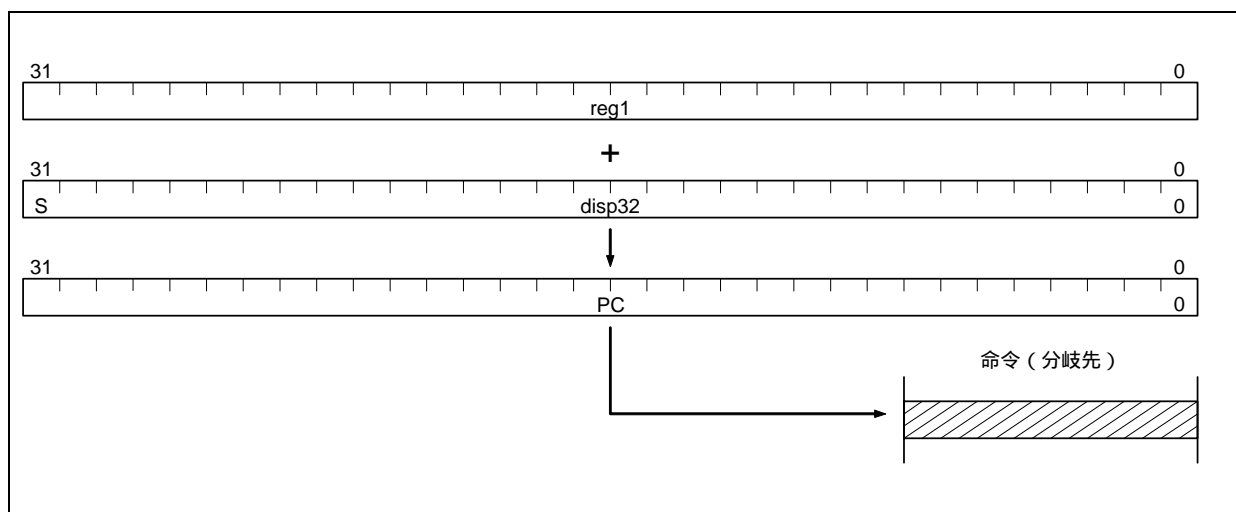


### (3) ベースト・アドレッシング

命令によって指定される汎用レジスタ (reg1) に, Nビット・ディスプレースメント (dispN) を加算した内容をプログラム・カウンタ (PC) に転送します。このとき, ディスプレースメントは, 2の補数データとして扱われ, それぞれ最上位ビットが符号ビット (S) となります。ディスプレースメントが32ビット未満の場合, 上位ビットを符号拡張します (Nは命令ごとに異なります)。

JMP命令が, このアドレッシングの対象となります。

図4-4 ベースト・アドレッシング



**(4) その他のアドレッシング**

命令によって指定される値をプログラム・カウンタ (PC) に転送します。値の指定方法は、それぞれの命令のオペレーション、または説明に記載されています。

CALLT命令、SYSCALL命令、TRAP命令、FETRAP命令、RIE命令、および例外発生時の分岐が、このアドレッシングの対象となります。

## 4.2.2 オペランド・アドレス

命令を実行する際に対象となるレジスタやメモリなどをアクセスするために、次に示す方法があります。

### (1) レジスタ・アドレッシング

汎用レジスタ指定フィールドにより指定される汎用レジスタ、またはシステム・レジスタをオペランドとしてアクセスするアドレッシングです。

オペランドに、reg1, reg2, reg3 または regID を含む命令が、このアドレッシングの対象となります。

### (2) イミディエト・アドレッシング

命令コード中に、操作対象となる任意の長さのデータを持つアドレッシングです。

オペランドに、imm5, imm16, vector, または cccc を含む命令が、このアドレッシングの対象となります。

**備考** vector：例外・ベクタ (00H-1FH) を指定するイミディエトであり、TRAP 命令、FETRAP 命令、SYSCALL 命令で使用されるオペランドです。データ幅は、各命令で異なります。

cccc：条件コード指定用の 4 ビット・データであり、CMOV 命令、SASF 命令、SETF 命令で使用されるオペランドです。0 の 1 ビットを上位に付加し、5 ビット・イミディエト・データとしてオペコード中に割り当てられます。

### (3) ベースト・アドレッシング

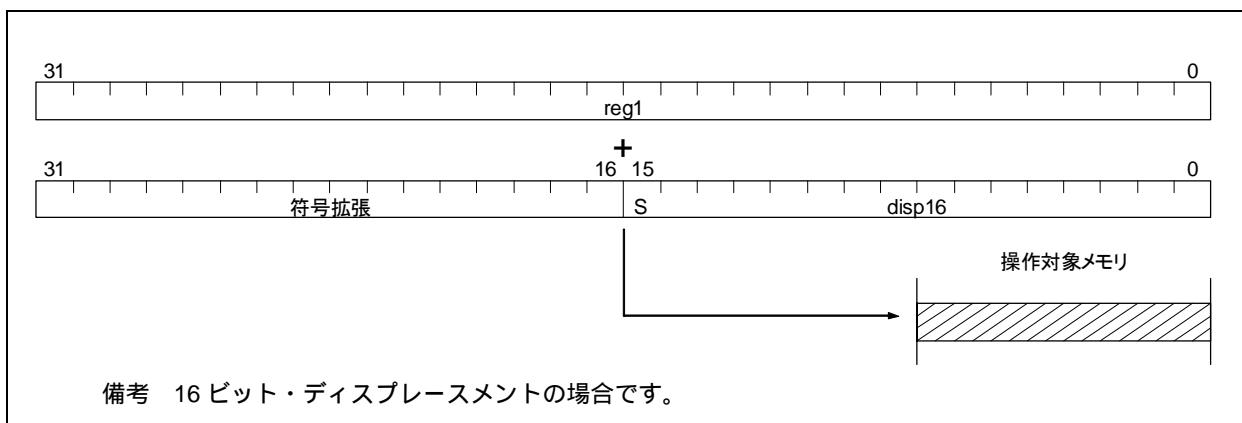
ベースト・アドレッシングには、次に示す 2 種類があります。

#### (a) タイプ1

命令コード中のアドレッシング指定フィールドで指定される汎用レジスタ (reg1) の内容とワード長まで符号拡張した N ビット・ディスプレイースメント (dispN) の和をオペランド・アドレスとして、操作対象となるメモリへのアクセスを行うアドレッシングです。このとき、ディスプレイースメントは、2 の補数データとして扱われ、それぞれ最上位ビットが符号ビット (S) となります。ディスプレイースメントが 32 ビット未満の場合、上位ビットを符号拡張します (N は命令ごとに異なります)。

LD 命令、ST 命令、CAXI 命令が、このアドレッシングの対象となります。

図4-5 ベースト・アドレッシング (タイプ1)

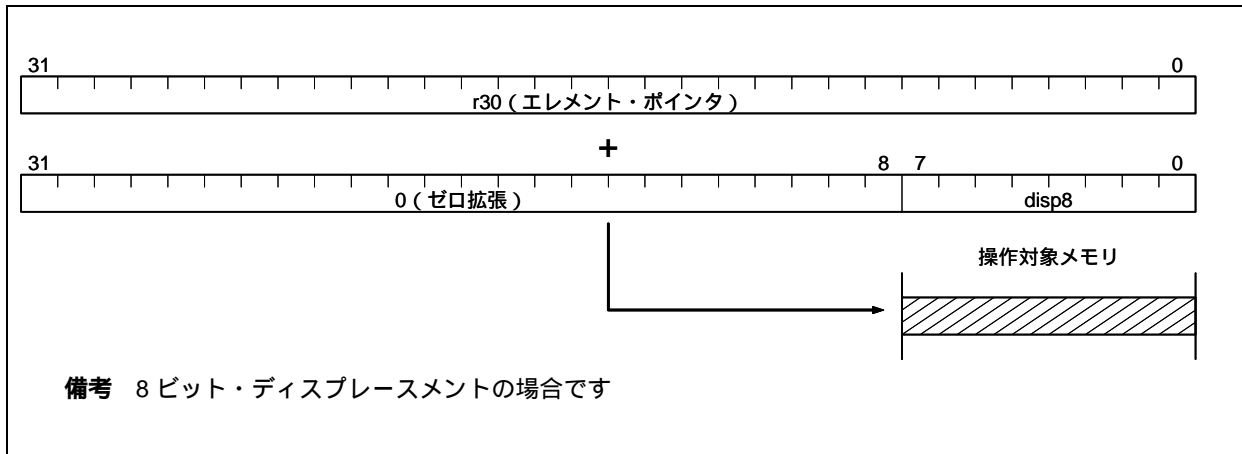


## (b) タイプ2

エレメント・ポインタ (r30) の内容とワード長までゼロ拡張した N ビット・ディスプレイメント・データ (dispN) の和をオペランド・アドレスとして、操作対象となるメモリへのアクセスを行うアドレッシングです。ディスプレイメントが 32 ビット未満の場合、上位ビットをゼロ拡張します (N は命令ごとに異なります)。

SLD 命令と SST 命令が、このアドレッシングの対象となります。

図4-6 ベースト・アドレッシング (タイプ2)

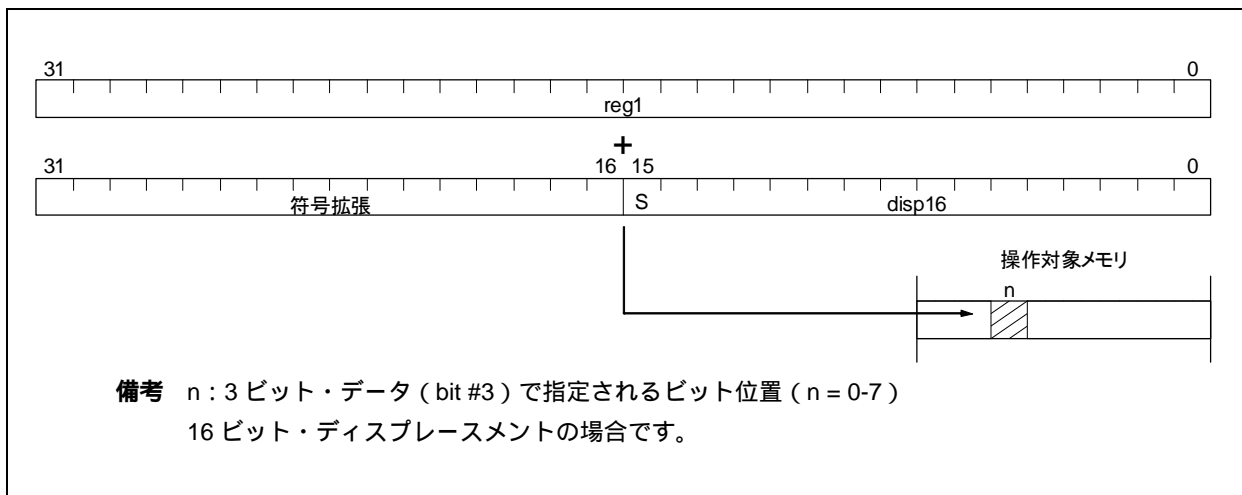


## (4) ビット・アドレッシング

汎用レジスタ (reg1) の内容とワード長まで符号拡張した N ビット・ディスプレイメント (dispN) の和をオペランド・アドレスとして、操作対象となるメモリ空間の 1 バイト中の 1 ビット (3 ビット・データ「bit #3」で指定) をアクセスするアドレッシングです。このとき、ディスプレイメントは、2 の補数データとして扱われ、それぞれ最上位ビットが符号ビット (S) となります。ディスプレイメントが 32 ビット未満の場合、上位ビットを符号拡張します (N は命令ごとに異なります)。

CLR1 命令, SET1 命令, NOT1 命令, TST1 命令が、このアドレッシングの対象となります。

図4-7 ビット・アドレッシング



**(5) その他のアドレッシング**

命令によって指定される値をオペランド・アドレスとして、操作対象となるメモリへのアクセスを行うアドレッシングです。値の指定方法は、それぞれの命令のオペレーション、または説明に記載されています。

SWITCH命令、CALLT命令、SYSCALL命令、PREPARE命令、DISPOSE命令が、このアドレッシングの対象となります。



## 第5章 命令

### 5.1 オペコードと命令フォーマット

V850E2M CPUの命令には、基本命令として定義される「CPU命令」と、用途ごとに定義される「コプロセッサ命令」の2種類があります。

#### 5.1.1 CPU命令

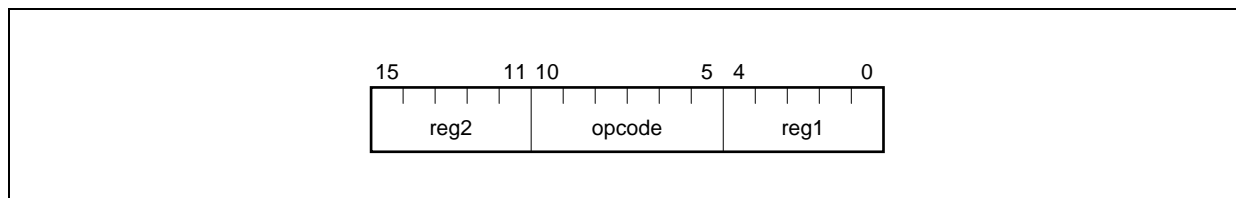
5.1.2 コプロセッサ命令で示されるコプロセッサ命令フォーマット以外のオペコード領域は、CPU命令に分類される命令が配置されます。

CPU命令は、基本的に16ビット長/32ビット長のフォーマットに従って表現されます。また、いくつかの命令は、これらのフォーマットに追加する形で、さらにオプション・データを利用し、48ビット長/64ビット長の命令を構成します。詳細は5.3 命令セットの各命令のオペコードを参照してください。

このオペコード領域中で、有意なCPU命令が定義されていないオペコードは、予約命令として将来の機能拡張のために予約されています。詳細は5.1.3 予約命令を参照してください。

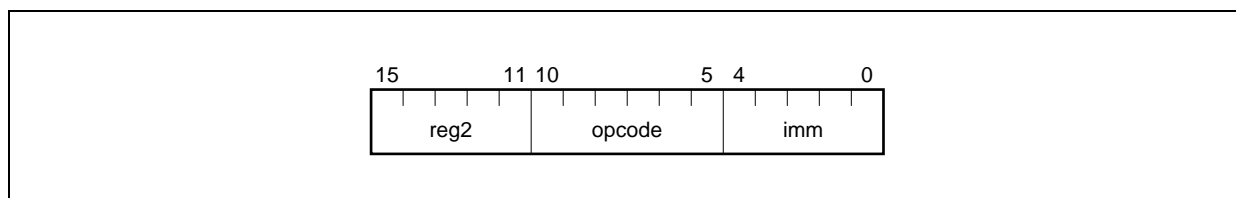
##### (1) reg-reg命令形式 (Format I)

6ビットのオペコード・フィールド、2つの汎用レジスタ指定フィールドを持つ16ビット長命令形式。



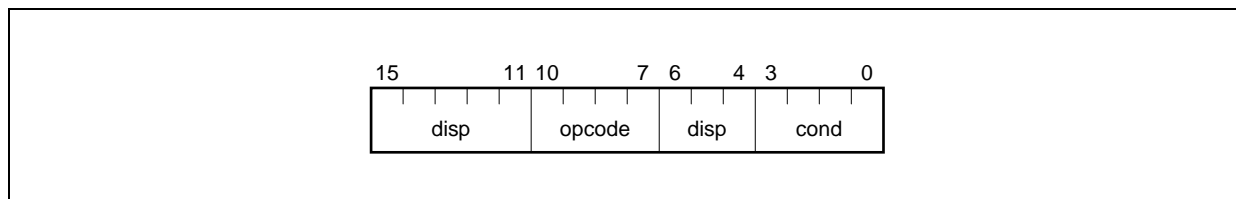
##### (2) imm-reg命令形式 (Format II)

6ビットのオペコード・フィールド、5ビットのイミディエイト・フィールド、1つの汎用レジスタ・フィールドを持つ16ビット長命令形式。

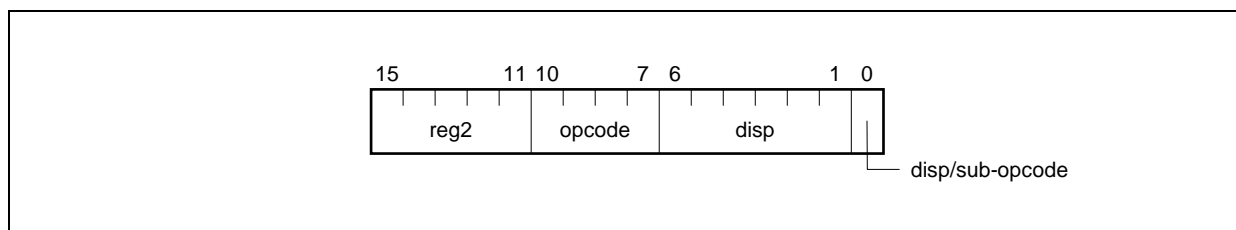


**(3) 条件分岐命令形式 (Format III)**

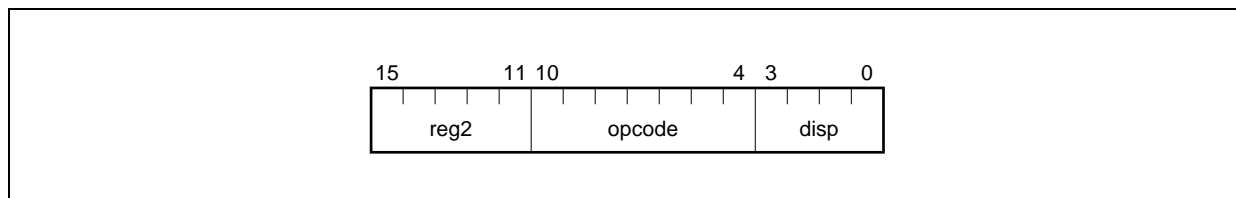
4 ビットのアペコード・フィールド, 4 ビットの条件コード・フィールド, 8 ビットのディスプレイメント・フィールドを持つ 16 ビット長命令形式。

**(4) ロード/ストア命令16ビット形式 (Format IV)**

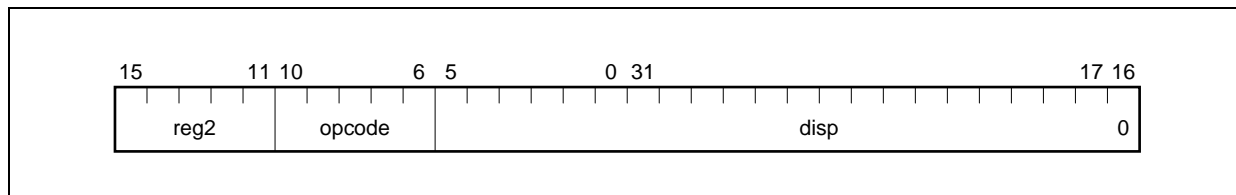
4 ビットのアペコード・フィールド, 1 つの汎用レジスタ指定フィールド, 7 ビットのディスプレイメント・フィールド (または 6 ビット・ディスプレイメント・フィールドと 1 ビット・サブアペコード・フィールド) を持つ 16 ビット長命令形式。



または, 7 ビットのアペコード・フィールドと 1 つの汎用レジスタ指定フィールド, 4 ビットのディスプレイメント・フィールドを持つ 16 ビット長命令形式。

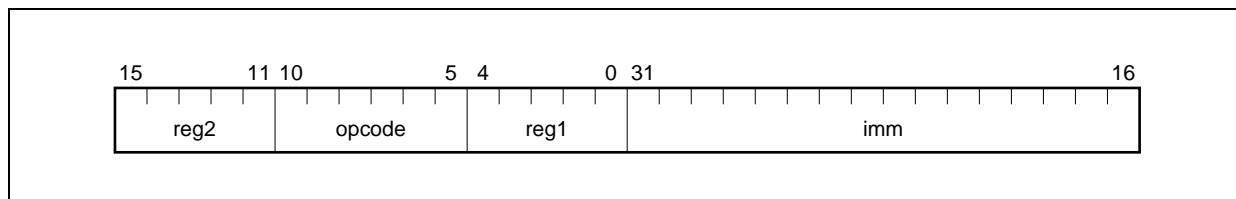
**(5) ジャンプ命令形式 (Format V)**

5 ビットのアペコード・フィールド, 1 つの汎用レジスタ指定フィールド, 22 ビットのディスプレイメント・フィールドを持つ 32 ビット長命令形式。

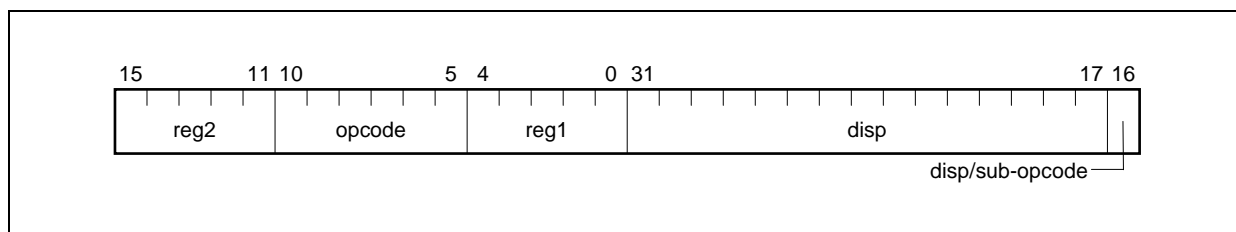


**(6) 3オペランド命令形式 (Format VI)**

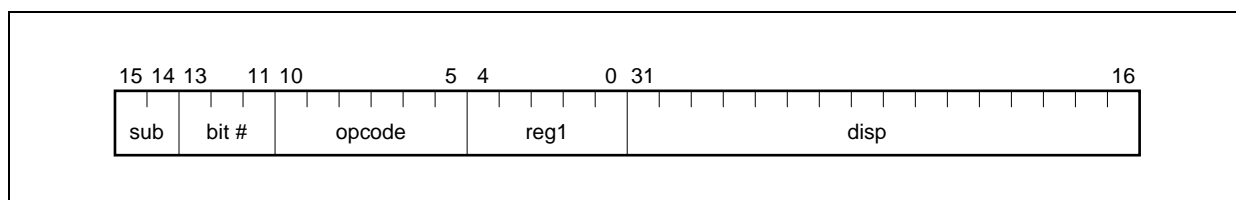
6 ビットのおペコード・フィールド, 2 つの汎用レジスタ指定フィールド, 16 ビットのエミーディエト・フィールドを持つ 32 ビット長命令形式。

**(7) ロード/ストア命令32ビット形式 (Format VII)**

6 ビットのおペコード・フィールド, 2 つの汎用レジスタ指定フィールド, 16 ビットのディスプレイースメント・フィールド(または 15 ビットのディスプレイースメント・フィールドと 1 ビット・サブオペコード・フィールド)を持つ 32 ビット長命令形式。

**(8) ビット操作命令形式 (Format VIII)**

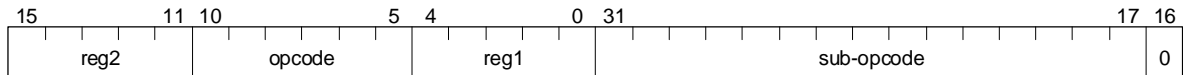
6 ビットのおペコード・フィールドと 2 ビットのサブオペコード・フィールド, 3 ビットのビット指定フィールド, 1 つの汎用レジスタ指定フィールド, 16 ビットのディスプレイースメント・フィールドを持つ 32 ビット長命令形式。



## (9) 拡張命令形式1 (Format IX)

6 ビットのアペコード・フィールドと 2 つの汎用レジスタ指定フィールドを持ち、それ以外のビットはサブアペコード・フィールドとして取り扱う 32 ビット長命令形式。

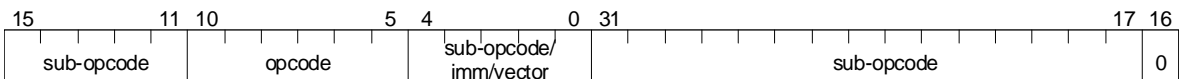
**注意** 拡張命令形式 1 では、汎用レジスタ指定フィールド、またはサブアペコード・フィールドの一部をシステム・レジスタ番号フィールド、条件コード・フィールド、イミディエト・フィールド、ディスプレイースメント・フィールドとして取り扱う場合があります。詳細は 5.3 命令セットの各命令の説明を参照してください。



## (10) 拡張命令形式2 (Format X)

6 ビットのアペコード・フィールドを持ち、それ以外のビットをサブアペコード・フィールドとして取り扱う 32 ビット長命令形式。

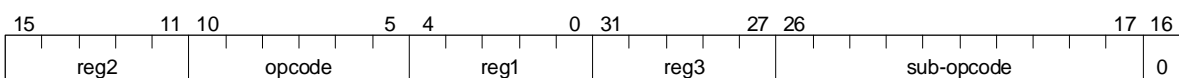
**注意** 拡張命令形式 2 では、汎用レジスタ指定フィールド、またはサブアペコード・フィールドの一部をシステム・レジスタ番号フィールド、条件コード・フィールド、イミディエト・フィールド、ディスプレイースメント・フィールドとして取り扱う場合があります。詳細は 5.3 命令セットの各命令の説明を参照してください。



## (11) 拡張命令形式3 (Format XI)

6 ビットのアペコード・フィールドと、3 つの汎用レジスタ指定フィールドを持ち、それ以外のビットをサブアペコード・フィールドとして取り扱う 32 ビット長命令形式。

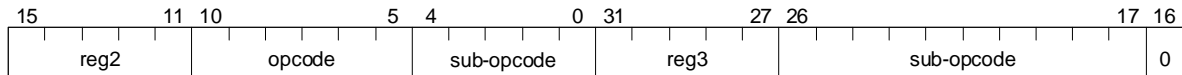
**注意** 拡張命令形式 3 では、汎用レジスタ指定フィールド、またはサブアペコード・フィールドの一部をシステム・レジスタ番号フィールド、条件コード・フィールド、イミディエト・フィールド、ディスプレイースメント・フィールドとして取り扱う場合があります。詳細は 5.3 命令セットの各命令の説明を参照してください。



## (12) 拡張命令形式4 (Format XII)

6 ビットのアペコード・フィールド, 2 つの汎用レジスタ指定フィールドを持ち, それ以外のビットをサブアペコード・フィールドとして取り扱う 32 ビット長命令形式。

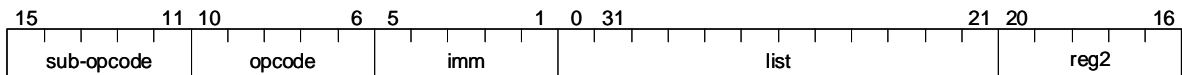
**注意** 拡張命令形式 4 では, 汎用レジスタ指定フィールド, またはサブアペコード・フィールドの一部をシステム・レジスタ番号フィールド, 条件コード・フィールド, イミューディエト・フィールド, ディスプレースメント・フィールドとして取り扱う場合があります。詳細は 5.3 命令セットの各命令の説明を参照してください。



## (13) スタック操作命令形式 (Format XIII)

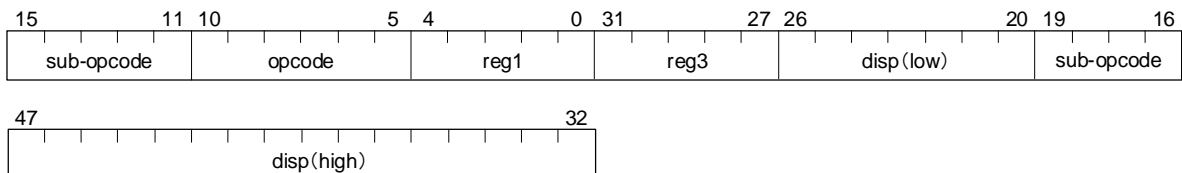
5 ビットのアペコード・フィールドと 5 ビットのアミューディエト・フィールド, 12 ビットのレジスタ・リスト・フィールド, 5 ビットのサブアペコード・フィールド, 1 つの汎用レジスタ指定フィールド (または 5 ビットのサブアペコード・フィールド) を持つ 32 ビット長命令形式。

汎用レジスタ指定フィールドは, 命令の形式によっては, サブアペコード・フィールドとして取り扱います。



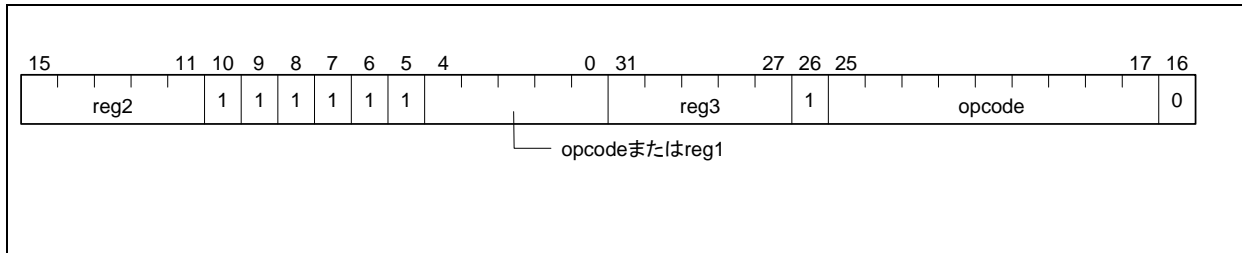
## (14) ロード/ストア命令48ビット形式 (Format XIV)

6 ビットのアペコード・フィールドと, 2 つの汎用レジスタ指定フィールド, 23 ビットのディスプレイースメント・フィールドを持ち, それ以外のビットをサブアペコード・フィールドとして取り扱う 48 ビット長命令形式。



### 5.1.2 コプロセッサ命令

次のフォーマットに従う命令は、コプロセッサ命令として定義されます。



コプロセッサ命令は、それぞれのコプロセッサ機能で定義されます。

V850E2M CPUでは、浮動小数点演算機能がコプロセッサとして定義されています。

浮動小数点演算命令の命令フォーマットは、**第4編 第4章 命 令**を参照してください。

#### (1) コプロセッサ使用不可例外

コプロセッサ命令と定義されたオペコードに対して、製品上で搭載されていない、あるいは動作状態によって、使用が許可されていない場合に、これらのコプロセッサ命令を実行しようとした場合、ただちにコプロセッサ使用不可例外（UCPOP）が発生します。

詳細は、**第7章 コプロセッサ使用不可状態**を参照してください。

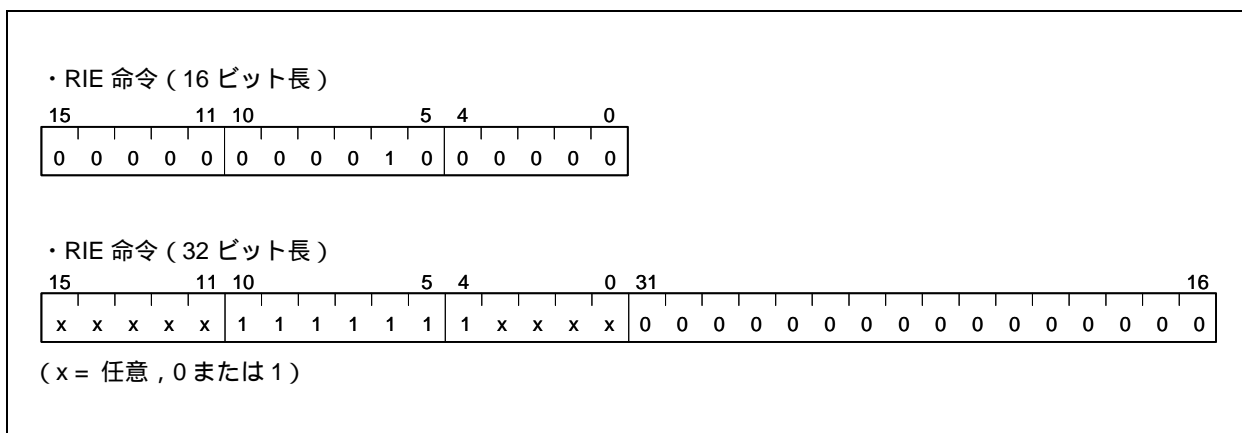
### 5.1.3 予約命令

将来の機能拡張のため予約され、命令が定義されていないオペコードは予約命令として定義されています。

予約命令のオペコードに対しては、次の2種類の動作のいずれを行うことが製品仕様によって定義されます。

- ・予約命令例外が発生する。
- ・いずれかの命令として動作する。

また、次のオペコードは V850E2M CPU において、常に予約命令例外が発生する RIE 命令として定義されています。



## 5.2 命令の概要

### (1) ロード命令

メモリからレジスタへのデータ転送を行います。次の命令（ニモニック）があります。

#### (a) LD命令

- LD.B : Load byte
- LD.BU : Load byte unsigned
- LD.H : Load half-word
- LD.HU : Load half-word unsigned
- LD.W : Load word

#### (b) SLD命令

- SLD.B : Short format load byte
- SLD.BU : Short format load byte unsigned
- SLD.H : Short format load half-word
- SLD.HU : Short format load half-word unsigned
- SLD.W : Short format load word

### (2) ストア命令

レジスタからメモリへのデータ転送を行います。次の命令（ニモニック）があります。

#### (a) ST命令

- ST.B : Store byte
- ST.H : Store half-word
- ST.W : Store word

#### (b) SST命令

- SST.B : Short format store byte
- SST.H : Short format store half-word
- SST.W : Short format store word

**(3) 乗算命令**

内蔵のハードウェア乗算器により、1クロックでの乗算処理を行います。次の命令（二モニック）があります。

- MUL : Multiply word
- MULH : Multiply half-word
- MULHI : Multiply half-word immediate
- MULU : Multiply word unsigned

**(4) 加算付き乗算命令**

乗算後、その結果に対する加算を行います。次の命令（二モニック）があります。

- MAC : Multiply and add word
- MACU : Multiply and add word unsigned

**(5) 算術演算命令**

加減算、レジスタ間のデータ転送、データ比較を行います。次の命令（二モニック）があります。

- ADD : Add
- ADDI : Add immediate
- CMP : Compare
- MOV : Move
- MOVEA : Move effective address
- MOVHI : Move high half-word
- SUB : Subtract
- SUBR : Subtract reverse

**(6) 条件付き演算命令**

指定された条件に応じた加減算を行います。次の命令（二モニック）があります。

- ADF : Add on condition flag
- SBF : Subtract on condition flag

**(7) 飽和演算命令**

飽和加減算を行います。なお、演算の結果が正の最大値(7FFFFFFFH)を越えたときは7FFFFFFFHを、負の最大値(80000000H)を越えたときは80000000Hを返します。次の命令（二モニック）があります。

- SATADD : Saturated add
- SATSUB : Saturated subtract
- SATSUBI : Saturated subtract immediate
- SATSUBR : Saturated subtract reverse



**(8) 論理演算命令**

論理演算を行います。次の命令（ニモニック）があります。

- AND : AND
- ANDI : AND immediate
- NOT : NOT
- OR : OR
- ORI : OR immediate
- TST : Test
- XOR : Exclusive OR
- XORI : Exclusive OR immediate

**(9) データ操作命令**

データ操作とシフト命令があります。シフト命令には、算術シフトと論理シフトがあります。内蔵のバレル・シフタにより、1 クロックで複数ビットのシフトを行います。次の命令（ニモニック）があります。

- BSH : Byte swap half-word
- BSW : Byte swap word
- CMOV : Conditional move
- HSH : Half-word swap half-word
- HSW : Half-word swap word
- SAR : Shift arithmetic right
- SASF : Shift and set flag condition
- SETF : Set flag condition
- SHL : Shift logical left
- SHR : Shift logical right
- SXB : Sign extend byte
- SXH : Sign extend half-word
- ZXB : Zero extend byte
- ZXH : Zero extend half-word

**(10) ビット・サーチ命令**

レジスタに格納されたデータから指定のビットを検索します。

- SCH0L : Search zero from left
- SCH0R : Search zero from right
- SCH1L : Search one from left
- SCH1R : Search one from right

**(11) 除算命令**

除算を行います。レジスタに格納された値にかかわらず、常に一定のステップ数で演算を実行します。次の命令（二モニック）があります。

- DIV : Divide word
- DIVH : Divide half-word
- DIVHU : Divide half-word unsigned
- DIVU : Divide word unsigned

**(12) 高速除算命令**

除算を行います。レジスタに格納された値から、あらかじめ商の有効桁数を判断し、必要最小なステップで演算を実行します。次の命令（二モニック）があります。

- DIVQ : Divide word quickly
- DIVQU : Divide word unsigned quickly

**(13) 分岐命令**

無条件分岐命令（JARL, JMP, JR）とフラグの状態により制御を変更する条件分岐命令（Bcond）があります。分岐命令により指定されたアドレスにプログラムの制御を移します。次の命令（二モニック）があります。

- Bcond ( BC, BE, BGE, BGT, BH, BL, BLE, BLT, BN, BNC, BNE, BNH, BNL, BNV, BNZ, BP, BR, BSA, BV, BZ ) : Branch on condition code
- JARL : Jump and register link
- JMP : Jump register
- JR : Jump relative

**(14) ビット操作命令**

メモリのビット・データに対して、論理演算を行います。指定されたビット以外は影響を受けません。次の命令（二モニック）があります。

- CLR1 : Clear bit
- NOT1 : Not bit
- SET1 : Set bit
- TST1 : Test bit

**(15) 特殊命令**

前項までのカテゴリに含まれない命令です。次の命令（二モニック）があります。

- CALLT : Call with table look up
- CAXI : Compare and exchange for interlock
- CTRET : Return from CALLT
- DI : Disable interrupt

- DISPOSE : Function dispose
- EI : Enable interrupt
- EIRET : Return from trap or interrupt
- FERET : Return from trap or interrupt
- FETRAP : Software Trap
- HALT : Halt
- LDSR : Load system register
- NOP : No operation
- PREPARE : Function prepare
- RETI : Return from trap or interrupt
- RIE : Reserved instruction exception
- STSR : Store system register
- SWITCH : Jump with table look up
- TRAP : Trap
- SYNCM : Synchronize memory
- SYNCP : Synchronize pipeline
- SYNCE : Synchronize exceptions
- SYSCALL : System call

## 5.3 命令セット

この節では、各命令のニモニクごとに（アルファベット順）、次の項目に分けて説明します。

- 命令形式 : 命令の記述方法，オペランドを示します（略号については，表5-1参照）。
- オペレーション : 命令の機能を示します（略号については，表5-2参照）。
- フォーマット : 命令形式を命令フォーマットで示します  
（5.1 オペコードと命令フォーマット参照）。
- オペコード : 命令のオペコードをビット・フィールドで示します（略号については，表5-3参照）。
- フラグ : 命令実行により変化する PSW（プログラム・ステータス・ワード）の各フラグの動作を示します。「0」はクリア（リセット）を，「1」はセットを，「-」は変化しないことを示します。
- 説明 : 命令の動作説明をします。
- 補足 : 命令の補足説明をします。
- 注意 : 注意事項を示します。

表5-1 命令形式の凡例

略号	意味
reg1	汎用レジスタ（ソース・レジスタとして使用）
reg2	汎用レジスタ（主にデスティネーション・レジスタとして使用。一部の命令で，ソース・レジスタとしても使用）
reg3	汎用レジスタ（主に除算結果の余り，乗算結果の上位32ビットを格納）
bit#3	ビット・ナンバ指定用3ビット・データ
imm x	x ビット・イミディエト・データ
disp x	x ビット・ディスプレイメント・データ
regID	システム・レジスタ番号
vector x	ベクタを指定するデータ（xはビット・サイズをあらわします）
cond	条件名を示します（表5-4 条件コード一覧参照）
cccc	条件コードを示す4ビット・データ（表5-4 条件コード一覧参照）
sp	スタック・ポインタ（r3）
ep	エレメント・ポインタ（r30）
list12	レジスタ・リスト

表5 - 2 オペレーションの凡例

略 号	意 味
←	代入
GR []	汎用レジスタ
SR []	システム・レジスタ
zero-extend (n)	nを, ワード長までゼロ拡張する。
sign-extend (n)	nを, ワード長まで符号拡張する。
load-memory (a, b)	アドレス「a」から, サイズ「b」のデータを読み出す。
store-memory (a, b, c)	アドレス「a」にデータ「b」をサイズ「c」で書き込む。
extract-bit (a, b)	データ「a」のビット・ナンバ「b」の値を取り出す。
set-bit (a, b)	データ「a」のビット・ナンバ「b」の値をセットする。
not-bit (a, b)	データ「a」のビット・ナンバ「b」の値を反転する。
clear-bit (a, b)	データ「a」のビット・ナンバ「b」の値をクリアする。
saturated (n)	nの飽和处理を行う。 計算の結果, n 7FFFFFFFHとなった場合, n = 7FFFFFFFHとする。 計算の結果, n 80000000Hとなった場合, n = 80000000Hとする。
result	結果をフラグに反映する。
Byte	バイト (8ビット)
Half-word	ハーフワード (16ビット)
Word	ワード (32ビット)
+	加算
-	減算
	ビット連結
×	乗算
÷	除算
%	除算結果の余り
AND	論理積
OR	論理和
XOR	排他的論理和
NOT	論理否定
logically shift left by	論理左シフト
logically shift right by	論理右シフト
arithmetically shift right by	算術右シフト

表5-3 オペコードの凡例

略号	意味
R	reg1またはregIDを指定するコードの1ビット分データ
r	reg2を指定するコードの1ビット分データ
w	reg3を指定するコードの1ビット分データ
D	ディスプレイメントの1ビット分データ(ディスプレイメントの上位ビットを示す)
d	ディスプレイメントの1ビット分データ
l	イミューディエットの1ビット分データ(イミューディエットの上位ビットを示す)
i	イミューディエットの1ビット分データ
V	vectorを指定するコードの1ビット分データ(vectorの上位ビットを示す)
v	vectorを指定するコードの1ビット分データ
cccc	条件コードを示す4ビット・データ(表5-4 条件コード一覧参照)
bbb	ビット・ナンバ指定用3ビット・データ
L	レジスタ・リスト中の汎用レジスタを指定する1ビット分データ
S	レジスタ・リスト中のEIPC/FEPC, EIPSW/FEPSWを指定する1ビット分データ
P	レジスタ・リスト中のPSWを指定する1ビット分データ

表5-4 条件コード一覧

条件コード(cccc)	条件名	条件式
0000	V	OV = 1
1000	NV	OV = 0
0001	C/L	CY = 1
1001	NC/NL	CY = 0
0010	Z	Z = 1
1010	NZ	Z = 0
0011	NH	(CY or Z) = 1
1011	H	(CY or Z) = 0
0100	S/N	S = 1
1100	NS/P	S = 0
0101	T	always (無条件)
1101	SA	SAT = 1
0110	LT	(S xor OV) = 1
1110	GE	(S xor OV) = 0
0111	LE	((S xor OV) or Z) = 1
1111	GT	((S xor OV) or Z) = 0

## &lt; 算術演算命令 &gt;

ADD	Add register/immediate
	加算

- [ 命令形式 ]           ( 1 ) ADD reg1, reg2  
                          ( 2 ) ADD imm5, reg2

- [ オペレーション ]   ( 1 ) GR [reg2] ← GR [reg2] + GR [reg1]  
                          ( 2 ) GR [reg2] ← GR [reg2] + sign-extend (imm5)

- [ フォーマット ]      ( 1 ) Format I  
                          ( 2 ) Format II

- [ オペコード ]
- |       |  |
|-------|--|
| ( 1 ) | <div style="display: flex; justify-content: space-between; width: 100%;"> <span>15</span> <span>0</span> </div> <div style="font-family: monospace; font-size: 1.2em;">rrrrrr001110RRRRR</div> |
| ( 2 ) | <div style="display: flex; justify-content: space-between; width: 100%;"> <span>15</span> <span>0</span> </div> <div style="font-family: monospace; font-size: 1.2em;">rrrrrr010010iiii</div>  |

- [ フラグ ]
- |     |                              |
|-----|------------------------------|
| CY  | MSB からのキャリーがあれば 1, そうでないとき 0 |
| OV  | オーバーフローが起こったとき 1, そうでないとき 0  |
| S   | 演算結果が負のとき 1, そうでないとき 0       |
| Z   | 演算結果が 0 のとき 1, そうでないとき 0     |
| SAT | –                            |

- [ 説 明 ]
- ( 1 ) 汎用レジスタ reg2 のワード・データに汎用レジスタ reg1 のワード・データを加算し, その結果を汎用レジスタ reg2 に格納します。汎用レジスタ reg1 は影響を受けません。
- ( 2 ) 汎用レジスタ reg2 のワード・データにワード長まで符号拡張した 5 ビット・イミュードを加算し, その結果を汎用レジスタ reg2 に格納します。

## &lt; 算術演算命令 &gt;

ADDI	Add immediate  加算
------	-------------------------

[ 命令形式 ]        ADDI imm16, reg1, reg2

[ オペレーション ]   GR [reg2] ← GR [reg1] + sign-extend (imm16)

[ フォーマット ]    Format VI

[ オペコード ]

15	0 31	16
rrrrrr110000RRRRR	iiiiiiiiiiiiiiiiiii	

[ フラ グ ]        CY    MSB からのキャリーがあれば 1, そうでないとき 0

OV    オーバフローが起こったとき 1, そうでないとき 0

S     演算結果が負のとき 1, そうでないとき 0

Z     演算結果が 0 のとき 1, そうでないとき 0

SAT   -

[ 説 明 ]        汎用レジスタ reg1 のワード・データにワード長まで符号拡張した 16 ビット・イミディエトを加算し, その結果を汎用レジスタ reg2 に格納します。汎用レジスタ reg1 は影響を受けません。



## &lt; 条件付き演算命令 &gt;

ADF	Add on condition flag  条件付き加算
-----	-------------------------------------

[ 命令形式 ]        ADF   cccc, reg1, reg2, reg3

[ オペレーション ]    if conditions are satisfied  
                           then GR [reg3] ← GR [reg1] + GR [reg2] +1  
                           else GR [reg3] ← GR [reg1] + GR [reg2] +0

[ フォーマット ]     Format XI

[ オペコード ]        15                            0 31                            16

rrrrrr111111RRRRR	wwwww011101cccc0
-------------------	------------------

[ フラ グ ]        CY        MSB からのキャリーがあれば 1, そうでないとき 0  
                       OV        オーバフローが起こったとき 1, そうでないとき 0  
                       S        演算結果が負のとき 1, そうでないとき 0  
                       Z        演算結果が 0 のとき 1, そうでないとき 0  
                       SAT       -

[ 説 明 ]        条件コード「cccc」で指定された条件が満たされた場合は、汎用レジスタ reg2 のワード・データに汎用レジスタ reg1 のワード・データを加算した結果に、1 を加算し、その結果を汎用レジスタ reg3 に格納します。

条件コード「cccc」で指定された条件が満たされなかった場合は、汎用レジスタ reg2 のワード・データに汎用レジスタ reg1 のワード・データを加算し、その結果を汎用レジスタ reg3 に格納します。

汎用レジスタ reg1, reg2 は影響を受けません。

次の表で示されている条件コードのうちの 1 つを「cccc」として指定してください(ただし, cccc ≠ 1101)。

条件コード	条件名	条件式	条件コード	条件名	条件式
0000	V	OV = 1	0100	S/N	S = 1
1000	NV	OV = 0	1100	NS/P	S = 0
0001	C/L	CY = 1	0101	T	always (無条件)
1001	NC/NL	CY = 0	0110	LT	(S xor OV) = 1
0010	Z	Z = 1	1110	GE	(S xor OV) = 0
1010	NZ	Z = 0	0111	LE	((S xor OV) or Z) = 1
0011	NH	(CY or Z) = 1	1111	GT	((S xor OV) or Z) = 0
1011	H	(CY or Z) = 0	(1101)	設定禁止	

## &lt; 論理演算命令 &gt;

AND	AND  論理積
-----	----------------

[ 命令形式 ]      AND reg1, reg2

[ オペレーション ]   GR [reg2] ← GR [reg2] AND GR [reg1]

[ フォーマット ]    Format I

[ オペコード ]      15                          0  
rrrrr001010RRRRR

[ フ ラ グ ]        CY    -

OV    0

S     演算結果のワード・データの MSB が 1 のとき 1, そうでないとき 0

Z     演算結果が 0 のとき 1, そうでないとき 0

SAT  -

[ 説 明 ]          汎用レジスタ reg2 のワード・データと汎用レジスタ reg1 のワード・データの論理積をとり, その結果を汎用レジスタ reg2 に格納します。汎用レジスタ reg1 は影響を受けません。

## &lt; 論理演算命令 &gt;

ANDI	AND immediate  論理積
------	--------------------------

[ 命令形式 ]        ANDI imm16, reg1, reg2

[ オペレーション ] GR [reg2] ← GR [reg1] AND zero-extend (imm16)

[ フォーマット ]    Format VI

[ オペコード ]

15	0 31	16
rrrrr110110RRRRR	iiiiiiiiiiiiiiiiiii	

[ フラ グ ]

CY    -

OV    0

S     演算結果のワード・データの MSB が 1 のとき 1, そうでないとき 0

Z     演算結果が 0 のとき 1, そうでないとき 0

SAT   -

[ 説 明 ]        汎用レジスタ reg1 のワード・データと 16 ビット・イミディエトをワード長までゼロ拡張した値の論理積をとり, その結果を汎用レジスタ reg2 に格納します。汎用レジスタ reg1 は影響を受けません。

## &lt; 分岐命令 &gt;

Bcond	Branch on condition code with 9-bit displacement
	条件分岐

[ 命令形式 ]      Bcond   disp9

[ オペレーション ]    if conditions are satisfied  
                          then PC ← PC + sign-extend (disp9)

[ フォーマット ]      Format III

[ オペコード ]        15                            0  
                          ┌───────────────────┐  
                          | dddddd1011dddcccc |

ただし、ddddddd は disp9 の上位 8 ビットです。

cccc は、cond で示される条件の条件コードです（表 5 - 5 Bcond 命令一覧参照）。

[ フラグ ]            CY    -  
                          OV    -  
                          S     -  
                          Z     -  
                          SAT  -

[ 説明 ]              命令が指定する PSW の各フラグをテストし、条件を満たしているときは分岐し、そうでないときは次の命令に進みます。分岐先 PC は、現在の PC と 8 ビット・イミューディエトを 1 ビット・シフトしてワード長まで符号拡張した 9 ビット・ディスプレースメントを加算した値です。

[ 補 足 ]              9 ビット・ディスプレースメントのビット 0 は 0 にマスクされます。なお、計算に使用される現在の PC とは、この命令自身の先頭バイトのアドレスであるためディスプレースメント値が 0 のときは、分岐先はこの命令自身になります。

表5 - 5 Bcond命令一覧

命 令	条件コード (cccc)	フラグの状態	分岐条件	
符号付き整数	BGE	1110	(S xor OV) = 0	Greater than or equal signed
	BGT	1111	((S xor OV) or Z) = 0	Greater than signed
	BLE	0111	((S xor OV) or Z) = 1	Less than or equal signed
	BLT	0110	(S xor OV) = 1	Less than signed
整数符号なし整数	BH	1011	(CY or Z) = 0	Higher (Greater than)
	BL	0001	CY = 1	Lower (Less than)
	BNH	0011	(CY or Z) = 1	Not higher (Less than or equal)
	BNL	1001	CY = 0	Not lower (Greater than or equal)
共通	BE	0010	Z = 1	Equal
	BNE	1010	Z = 0	Not equal
その他	BC	0001	CY = 1	Carry
	BF	1010	Z = 0	False
	BN	0100	S = 1	Negative
	BNC	1001	CY = 0	No carry
	BNV	1000	OV = 0	No overflow
	BNZ	1010	Z = 0	Not zero
	BP	1100	S = 0	Positive
	BR	0101	-	Always (無条件)
	BSA	1101	SAT = 1	Saturated
	BT	0010	Z = 1	True
	BV	0000	OV = 1	Overflow
	BZ	0010	Z = 1	Zero

**注意** 飽和演算命令の実行結果で SAT フラグがセット(1)された場合、符号付き整数の条件分岐(BGE, BGT, BLE, BLT)は、分岐条件に意味がなくなります。これは、次の理由によるものです。通常の演算では、結果が正の最大値を越えると負の値になり、負の最大値を越えたときは正の値になります。つまり、オーバーフローが生じると、S フラグが反転(0 1, 1 0)します。一方、飽和演算命令では、結果が正の最大値を越えたときは正の値で、負の最大値を越えたときは負の値で飽和します。通常の演算とは異なり、オーバーフローが生じても S フラグは反転しません。このように、演算結果が飽和したときの S フラグは通常の演算とは異なるので、OV フラグとの排他的論理和(XOR)をとる分岐条件に意味がなくなります。

## &lt; データ操作命令 &gt;

<p style="font-size: 24px; margin: 0;">BSH</p>	<p style="text-align: right; font-size: 12px;">Byte swap half-word</p> <p style="text-align: center; font-size: 12px;">ハーフワード・データのバイト・スワップ</p>
--	--

[ 命令形式 ]      BSH reg2, reg3

[ オペレーション ]   GR [reg3] ← GR [reg2] (23:16) || GR [reg2] (31:24) || GR [reg2] (7:0) || GR [reg2] (15:8)

[ フォーマット ]    Format XII

[ オペコード ]

15		0 31		16
rrrrrr111111100000   wwwwww01101000010				

[ フ ラ グ ]      CY    演算結果の下位ハーフワード・データ中に、0 のバイトが 1 つ以上含まれるとき 1 ,  
                              そうでないとき 0

OV    0

S    演算結果のワード・データの MSB が 1 のとき 1 , そうでないとき 0

Z    演算結果の下位ハーフワード・データが 0 のとき 1 , そうでないとき 0

SAT   -

[ 説 明 ]      エンディアン変換します。

## &lt;データ操作命令&gt;

BSW	Byte swap word  ワード・データのバイト・スワップ
-----	--

[ 命令形式 ]         BSW reg2, reg3

[ オペレーション ]   GR [reg3] ← GR [reg2] (7:0) || GR [reg2] (15:8) || GR [reg2] (23:16) || GR [reg2] (31:24)

[ フォーマット ]     Format XII

[ オペコード ]

15	0	31	16
rrrrrr	111111	100000	wwwww01101000000

[ フ ラ グ ]         CY    演算結果のワード・データ中に、0のバイトが1つ以上含まれるとき 1、  
                          そうでないとき 0

OV    0

S      演算結果のワード・データのMSBが1のとき 1、そうでないとき 0

Z      演算結果のワード・データが0のとき 1、そうでないとき 0

SAT   -

[ 説 明 ]         エンディアン変換します。

## &lt; 特殊命令 &gt;

CALLT	Call with table look up  テーブル参照によるサブルーチン・コール
-------	--

[ 命令形式 ]      CALLT imm6

[ オペレーション ]      CTPC ← PC + 2 (return PC)

                            CTPSW ← PSW

                            adr ← CTBP + zero-extend (imm6 logically shift left by 1)

                            PC ← CTBP + zero-extend (Load-memory (adr, Half-word) )

[ フォーマット ]      Format II

[ オペコード ]      15                      0  
                            0000001000iiiiiii

[ フラグ ]              CY    -

                            OV    -

                            S     -

                            Z     -

                            SAT -

[ 説 明 ]              次の順に処理を行います。

<1> 復帰 PC と PSW の内容を CTPC と CTPSW に転送

<2> CTBP の値と、1 ビット論理左シフトしワード長までゼロ拡張した 6 ビット・イミュー  
        ディエト・データを加算して 32 ビット・テーブル・エントリ・アドレスを生成

<3> <2>で生成されたアドレスのハーフワードをロードし、ワード長までゼロ拡張

<4> <3>のデータに CTBP の値を加算して 32 ビット・ターゲット・アドレスを生成

<5> <4>で生成されたターゲット・アドレスへ分岐

- |   |
|---|
| <p>注意 1. 命令実行中に例外が発生すると、リード・サイクルが終了したあとに命令の実行を中止する場合があります。</p> <p>2. CALLT 命令のテーブル読み出しのためのメモリからの読み出し操作では、プロセッサ保護が行われ<br/>      ません。</p> <p>3. メモリ保護 (PSW.DMP = 1) や周辺装置保護 (PSW.PP = 1) が有効である場合に、ユーザ・プログラ<br/>      ムからのアクセスが禁止されている領域に配置されているテーブルからターゲット・アドレスを生成す<br/>      るためのデータをロードすることはできません。</p> |
|---|



## &lt; 特殊命令 &gt;

CAXI	Compare and exchange for interlock  比較と交換
------	---

[ 命令形式 ]        CAXI [reg1], reg2, reg3

[ オペレーション ]    adr        GR[reg1]<sup>注</sup>  
                       token      Load-memory(ad, Word)  
                       result     GR[reg2] – token  
                       If result == 0  
                       then     Store-memory(ad, GR[reg3], Word)  
                                   GR[reg3]    token  
                       else     Store-memory(ad, token, Word)  
                                   GR[reg3]    token

注 GR[reg1]の下位 2 ビットは, 0 にマスクし adr とします。

[ フォーマット ]     Format XI

[ オペコード ]        15                            0 31                            16  

rrrrrr111111RRRRR	wwwww00011101110
-------------------	------------------

[ フラグ ]            CY    result の演算時に MSB へのポローがあれば 1, そうでないとき 0  
                       OV    result の演算時にオーバーフローが起こったとき 1, そうでないとき 0  
                       S     result が負のとき 1, そうでないとき 0  
                       Z     result が 0 のとき 1, そうでないとき 0  
                       SAT   -

[ 説 明 ]            まず, 汎用レジスタ reg1 のデータを読み出して, 下位 2 ビットを 0 にマスクし, ワード境界にアラインされた 32 ビット・アドレスを生成します。生成したアドレスからワード・データを読み出し, 汎用レジスタ reg2 のワード・データと比較し, 結果を PSW の各フラグに示します。比較は汎用レジスタ reg2 のワード・データから, 読み出したワード・データを減算することで行います。比較の結果が 0 であれば, 汎用レジスタ reg3 のワード・データを, そうでなければ, 読み出したワード・データを, 生成したアドレスに格納します。  
                       その後, 読み出したワード・データを汎用レジスタ reg3 へ格納します。汎用レジスタ reg1, reg2 は影響を受けません。

<p><b>注意</b> この命令は排他制御を目的としたアトミック性保証のため, 読み出しから書き込みまでの間, 対象のアドレスが他の要因によるアクセスによって操作されることはありません。</p>
--

## &lt; ビット操作命令 &gt;

CLR1	Clear bit  ビット・クリア
------	--------------------------

- [ 命令形式 ]           ( 1 ) CLR1 bit#3, disp16 [reg1]  
                          ( 2 ) CLR1 reg2, [reg1]

- [ オペレーション ]   ( 1 ) adr   GR [reg1] + sign-extend (disp16)  
                          token   Load-memory (adr, Byte)  
                          Z フラグ   Not (extract-bit (token, bit#3) )  
                          token   clear-bit (token, bit#3)  
                          Store-memory (adr, token, Byte)  
                          ( 2 ) adr   GR [reg1]  
                          token   Load-memory (adr, Byte)  
                          Z フラグ   Not (extract-bit (token, reg2) )  
                          token   clear-bit (token, reg2)  
                          Store-memory (adr, token, Byte)

- [ フォーマット ]      ( 1 ) Format VIII  
                          ( 2 ) Format IX

- [ オペコード ]
- |       |                  |                 |
|-------|------------------|-----------------|
| 15    | 0 31             | 16              |
| ( 1 ) | 10bbb111110RRRRR | ddddddddddddddd |
- |       |                   |                  |
|-------|-------------------|------------------|
| 15    | 0 31              | 16               |
| ( 2 ) | rrrrrr111111RRRRR | 0000000011100100 |

- [ フラ グ ]           CY    -  
                          OV    -  
                          S     -  
                          Z     指定したビットが 0 のとき 1, 指定したビットが 1 のとき 0  
                          SAT  -

- [ 説 明 ]           ( 1 ) まず, 汎用レジスタ reg1 のワード・データと, ワード長まで符号拡張した 16 ビット・ディスプレイメントを加算して 32 ビット・アドレスを生成します。生成したアドレスのバイト・データを読み出し, 3 ビットのビット・ナンバで指定されるビットをクリア (0) し, 元のアドレスに書き戻します。読み出したバイト・データの指定ビットが 0 のとき Z フラグをセット (1) し, 指定ビットが 1 のとき Z フラグをクリア (0) します。  
                          ( 2 ) まず, 汎用レジスタ reg1 のワード・データを読み出して 32 ビット・アドレスを生成します。生成したアドレスのバイト・データを読み出し, 汎用レジスタ reg2 の下位 3 ビットで指定されるビットをクリア (0) し, 元のアドレスに書き戻します。読み出したバイト・データの指定ビットが 0 のとき Z フラグをセット (1) し, 指定ビットが 1 のとき Z フラグをクリア (0) します。

[ 補 足 ] PSW の Z フラグはこの命令を実行する前に該当ビットが 0 か 1 だったかを示します。この命令実行後の該当ビットの内容を示すものではありません。

**注意** この命令は排他制御を目的としたアトミック性保証のため、読み出しから書き込みまでの間、対象のアドレスが他の要因によるアクセスによって操作されることはありません。

## &lt; データ操作命令 &gt;

CMOV	Conditional move  条件付き転送
------	--------------------------------

- [ 命令形式 ]           ( 1 ) CMOV   cccc, reg1, reg2, reg3  
                          ( 2 ) CMOV   cccc, imm5, reg2, reg3

- [ オペレーション ]   ( 1 ) if conditions are satisfied  
                          then GR [reg3] ← GR [reg1]  
                          else GR [reg3] ← GR [reg2]  
                          ( 2 ) if conditions are satisfied  
                          then GR [reg3] ← sign-extended (imm5)  
                          else GR [reg3] ← GR [reg2]

- [ フォーマット ]    ( 1 ) Format XI  
                          ( 2 ) Format XII

- [ オペコード ]
- |       |                   |                  |
|-------|-------------------|------------------|
| 15    | 0 31              | 16               |
| ( 1 ) | rrrrrr111111RRRRR | wwwww011001cccc0 |
- |       |                  |                  |
|-------|------------------|------------------|
| 15    | 0 31             | 16               |
| ( 2 ) | rrrrrr111111iiii | wwwww011000cccc0 |

- [ フラグ ]           CY    -  
                          OV    -  
                          S     -  
                          Z     -  
                          SAT  -

- [ 説 明 ]           ( 1 ) 条件コード「cccc」で指定された条件が満たされた場合は汎用レジスタ reg1 のデータを、満たされなかった場合は汎用レジスタ reg2 のデータを、汎用レジスタ reg3 に転送します。次の表で示されている条件コードのうちの 1 つを「cccc」として指定してください。

条件コード	条件名	条件式	条件コード	条件名	条件式
0000	V	OV = 1	0100	S/N	S = 1
1000	NV	OV = 0	1100	NS/P	S = 0
0001	C/L	CY = 1	0101	T	always ( 無条件 )
1001	NC/NL	CY = 0	1101	SA	SAT = 1
0010	Z	Z = 1	0110	LT	(S xor OV) = 1
1010	NZ	Z = 0	1110	GE	(S xor OV) = 0
0011	NH	(CY or Z) = 1	0111	LE	((S xor OV) or Z) = 1
1011	H	(CY or Z) = 0	1111	GT	((S xor OV) or Z) = 0

- ( 2 ) 条件コード「cccc」で指定された条件が満たされた場合はワード長まで符号拡張した

5 ビット・イミディエト・データを、満たされなかった場合は汎用レジスタ reg2 のデータを、汎用レジスタ reg3 に転送します。次の表で示されている条件コードのうちの 1 つを「cccc」として指定してください。

条件コード	条件名	条件式	条件コード	条件名	条件式
0000	V	$OV = 1$	0100	S/N	$S = 1$
1000	NV	$OV = 0$	1100	NS/P	$S = 0$
0001	C/L	$CY = 1$	0101	T	always (無条件)
1001	NC/NL	$CY = 0$	1101	SA	$SAT = 1$
0010	Z	$Z = 1$	0110	LT	$(S \text{ xor } OV) = 1$
1010	NZ	$Z = 0$	1110	GE	$(S \text{ xor } OV) = 0$
0011	NH	$(CY \text{ or } Z) = 1$	0111	LE	$((S \text{ xor } OV) \text{ or } Z) = 1$
1011	H	$(CY \text{ or } Z) = 0$	1111	GT	$((S \text{ xor } OV) \text{ or } Z) = 0$

[ 補 足 ]          SETF 命令を参照してください。

## &lt; 算術演算命令 &gt;

CMP	Compare register/immediate (5-bit)
	比較

- [ 命令形式 ]           ( 1 ) CMP reg1, reg2  
                          ( 2 ) CMP imm5, reg2

- [ オペレーション ]   ( 1 ) result ← GR [reg2] – GR [reg1]  
                          ( 2 ) result ← GR [reg2] – sign-extend (imm5)

- [ フォーマット ]      ( 1 ) Format I  
                          ( 2 ) Format II

- [ オペコード ]
- |       |  |
|-------|--|
| ( 1 ) | <div style="display: flex; justify-content: space-between; width: 100%;"> <span>15</span> <span>0</span> </div> <div style="font-family: monospace; font-size: 1.2em;">rrrrrr001111RRRRR</div> |
| ( 2 ) | <div style="display: flex; justify-content: space-between; width: 100%;"> <span>15</span> <span>0</span> </div> <div style="font-family: monospace; font-size: 1.2em;">rrrrrr010011iiii</div>  |

- [ フラグ ]           CY   MSB へのポローがあれば 1, そうでないとき 0  
                      OV   オーバフローが起こったとき 1, そうでないとき 0  
                      S    演算結果が負のとき 1, そうでないとき 0  
                      Z    演算結果が 0 のとき 1, そうでないとき 0  
                      SAT  –

- [ 説 明 ]           ( 1 ) 汎用レジスタ reg2 のワード・データと汎用レジスタ reg1 のワード・データを比較し, 結果を PSW の各フラグに示します。比較は汎用レジスタ reg2 のワード・データから汎用レジスタ reg1 の内容を減算することで行います。汎用レジスタ reg1, reg2 は影響を受けません。  
                      ( 2 ) 汎用レジスタ reg2 のワード・データとワード長まで符号拡張した 5 ビット・イミディエトを比較し, 結果を PSW の各フラグに示します。比較は汎用レジスタ reg2 のワード・データから符号拡張したイミディエトの内容を減算することで行います。汎用レジスタ reg2 は影響を受けません。

## &lt; 特殊命令 &gt;

CTRET	Return from CALLT  サブルーチン・コールからの復帰
-------	--

[ 命令形式 ]           CTRET

[ オペレーション ]   PC ← CTPC  
                          PSW ← CTPSW

[ フォーマット ]      Format X

[ オペコード ]        15                           0 31                           16  

00000111111100000	0000000101000100
-------------------	------------------

[ フラグ ]            CY   CTPSW から読み出した値が設定される  
                       OV   CTPSW から読み出した値が設定される  
                       S    CTPSW から読み出した値が設定される  
                       Z    CTPSW から読み出した値が設定される  
                       SAT  CTPSW から読み出した値が設定される

[ 説 明 ]            システム・レジスタから復帰 PC と PSW を取り出し，CALLT 命令により呼び出されたルーチンから復帰します。この命令の動作は次のとおりです。

<1> 復帰 PC と PSW を，CTPC と CTPSW から取り出します。

<2> 取り出した復帰 PC と PSW を PC と PSW に設定し，制御を移します。

## &lt; 特殊命令 &gt;

DI	Disable interrupt EI レベル・マスクブル例外の禁止
----	--

[ 命令形式 ]        DI

[ オペレーション ]   PSW.ID ← 1 ( EI レベル・マスクブル例外の禁止 )

[ フォーマット ]     Format X

[ オペコード ]        15                                  0 31                                  16

00000111111100000	00000001011100000
-------------------	-------------------

[ フ ラ グ ]        CY   -  
                      OV   -  
                      S    -  
                      Z    -  
                      SAT -  
                      ID   1

[ 説 明 ]        PSW の ID ビットをセット ( 1 ) し , この命令実行中から EI レベル・マスクブル例外の受け付けを禁止します。

[ 補 足 ]        この命令による PSW のフラグの書き換えが有効になるのは次の命令からとなります。



< 特殊命令 >

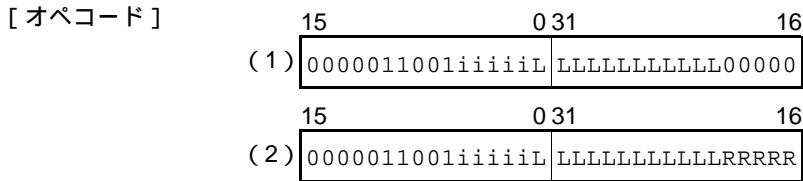
DISPOSE	Function dispose  スタック・フレームの削除
---------	--------------------------------------

- [ 命令形式 ]           ( 1 ) DISPOSE imm5, list12  
                          ( 2 ) DISPOSE imm5, list12, [reg1]

- [ オペレーション ]   ( 1 ) adr    sp + zero-extend (imm5 logically shift left by 2)  
                          foreach (all regs in list12) {  
                                  GR[reg in list12]   Load-memory (adr, Word) <sup>注</sup>  
                                  adr    adr + 4  
                          }  
                          sp    adr  
  
                          ( 2 ) adr    sp + zero-extend (imm5 logically shift left by 2)  
                          foreach (all regs in list12) {  
                                  GR[reg in list12]   Load-memory (adr, Word) <sup>注</sup>  
                                  adr    adr + 4  
                          }  
                          sp    adr  
                          PC    GR[reg1]

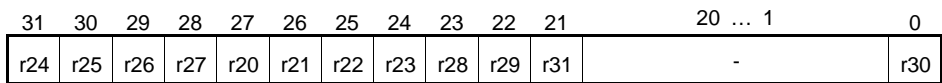
**注** Load-memory 時に adr の下位 2 ビットは 0 にマスクされます。

[ フォーマット ]    Format XIII



RRRRR    00000 ( reg1 には r0 を設定しないでください )  
また, LLLLLLLLLLLL は, レジスタ・リスト「list12」の中の対応するビットの値を示します (たとえば, オペコード中のビット 21 の「L」は list12 のビット 21 の値を示します)。

list12 は, 次のように定義される 32 ビットのレジスタ・リストです。



ビット 31-21 とビット 0 の各ビットに汎用レジスタ ( r20-r31 ) が対応しており, セット ( 1 )されたビットに対応するレジスタが操作の対象として指定されます。たとえば, r20, r30 を指定する場合, list12 の値は次のようになります ( レジスタが対応付けられていな

いビット 20-1 への設定値は任意です)。

- レジスタが対応付けられていないビットの値をすべて 0 とした場合 : 08000001H
- レジスタが対応付けられていないビットの値をすべて 1 とした場合 : 081FFFFFFH

[ フ ラ グ ]      CY    -  
                  OV    -  
                  S     -  
                  Z     -  
                  SAT  -

[ 説 明 ]           (1) 5 ビット・イミディエト・データを, 2 ビット論理左シフトし, ワード長までゼロ拡張したワード・データを, sp に加算します。そして, list12 で指定されている汎用レジスタに復帰 (sp で指定するアドレスからデータをロードし, sp に 4 を加算) します。  
                  (2) 5 ビット・イミディエト・データを, 2 ビット論理左シフトし, ワード長までゼロ拡張したワード・データを, sp に加算します。そして, list12 で指定されている汎用レジスタに復帰 (sp で指定するアドレスからデータをロードし, sp に 4 を加算) し, 汎用レジスタ reg1 で指定されたアドレスに制御を移します。

[ 補 足 ]           list12 の汎用レジスタは, 降順にロードされます (r31, r30, ..., r20)。  
                  imm5 は, 自動変数と一時データのためのスタック・フレームを復元します。  
                  sp で指定された下位 2 ビットのアドレスは, 0 でマスクされ, ワード境界にアラインされます。

**注意 1.** 命令実行中に例外が発生すると, リード・サイクルとレジスタ値の書き換えが終了したあとに, 命令の実行を中止する場合がありますが, sp は実行開始前の元の値を保持します。そのあと, 例外から復帰すると, 命令が再実行されます。

**2.** 命令形式 (2) の DISPOSE imm5, list12, [reg1] では, reg1 には r0 を指定しないでください。

## &lt; 除算命令 &gt;

DIV	Divide word  (符号付き)ワード・データの除算
-----	-------------------------------------

[ 命令形式 ]      DIV reg1, reg2, reg3

[ オペレーション ]   GR [reg2] ← GR [reg2] ÷ GR [reg1]  
                          GR [reg3] ← GR [reg2] % GR [reg1]

[ フォーマット ]     Format XI

[ オペコード ]      15                            0 31                            16  
                          rrrrrr111111RRRRR | wwww01011000000

[ フラグ ]            CY    -  
                          OV    オーバフローが起こったとき 1, そうでないとき 0  
                          S      演算結果の商が負のとき 1, そうでないとき 0  
                          Z      演算結果の商が 0 のとき 1, そうでないとき 0  
                          SAT   -

[ 説 明 ]            汎用レジスタ reg2 のワード・データを汎用レジスタ reg1 のワード・データで除算し, その商を汎用レジスタ reg2 に, 余りを汎用レジスタ reg3 に格納します。汎用レジスタ reg1 は影響を受けません。ゼロで割ったときは, オーバフローを生じ, OV フラグ以外の演算結果は不定となります。

[ 補 足 ]            オーバフローは負の最大値 (80000000H) を-1 で割ったとき (商が 80000000H) と, ゼロによる除算のとき (演算結果は不定) に生じます。  
                          汎用レジスタ reg2 と汎用レジスタ reg3 が同じレジスタの場合, そのレジスタには余りが格納されます。  
                          この命令実行中に例外が発生すると, 実行を中止し, 戻り番地をこの命令の先頭アドレスとして例外を処理してから, 例外処理完了後に再実行します。この場合, 汎用レジスタ reg1 と汎用レジスタ reg2 はこの命令実行前の値を保持します。

**注意**    汎用レジスタ reg2 と汎用レジスタ reg3 に同じレジスタを指定した場合, 汎用レジスタ reg2 に演算結果の商が格納されないため, フラグは不定となります。

## &lt; 除算命令 &gt;

DIVH	Divide half-word  (符号付き) ハーフワード・データの除算
------	--

- [ 命令形式 ]           (1) DIVH reg1, reg2  
                          (2) DIVH reg1, reg2, reg3

- [ オペレーション ]   (1) GR [reg2] ← GR [reg2] ÷ GR [reg1]  
                          (2) GR [reg2] ← GR [reg2] ÷ GR [reg1]  
                          GR [reg3] ← GR [reg2] % GR [reg1]

- [ フォーマット ]      (1) Format I  
                          (2) Format XI

- [ オペコード ]
- |   |                                |    |
|---|--------------------------------|----|
| 15  | 0                              |    |
| (1) <span style="border: 1px solid black; padding: 2px;">rrrrrr000010RRRRR</span>                   |                                |    |
| RRRRR   | 00000 (reg1 には r0 を設定しないでください) |    |
| rrrrrr  | 00000 (reg2 には r0 を設定しないでください) |    |
| 15  | 0 31                           | 16 |
| (2) <span style="border: 1px solid black; padding: 2px;">rrrrrr111111RRRRR wwwww010100000000</span> |                                |    |

- [ フラ グ ]
- CY    -
- OV    オーバーフローが起こったとき 1, そうでないとき 0
- S      演算結果の商が負のとき 1, そうでないとき 0
- Z      演算結果の商が 0 のとき 1, そうでないとき 0
- SAT   -

- [ 説 明 ]
- (1) 汎用レジスタ reg2 のワード・データを汎用レジスタ reg1 の下位ハーフワード・データで除算し、その商を汎用レジスタ reg2 に格納します。汎用レジスタ reg1 は影響を受けません。ゼロで割ったときは、オーバーフローを生じ、OV フラグ以外の演算結果は不定となります。
- (2) 汎用レジスタ reg2 のワード・データを汎用レジスタ reg1 の下位ハーフワード・データで除算し、その商を汎用レジスタ reg2 に、余りを汎用レジスタ reg3 に格納します。汎用レジスタ reg1 は影響を受けません。ゼロで割ったときは、オーバーフローを生じ、OV フラグ以外の演算結果は不定となります。

[ 補 足 ]

- (1) 除算結果の余りは格納されません。オーバーフローは負の最大値 (80000000H) を  $-1$  で割ったとき (商が 80000000H) と、ゼロによる除算のとき (演算結果は不定) に生じます。この命令実行中に例外が発生すると、実行を中止し、戻り番地をこの命令の先頭アドレスとして例外を処理してから、例外処理完了後に再実行します。この場合、汎用レジスタ reg1 と汎用レジスタ reg2 はこの命令実行前の値を保持します。
- (2) オーバーフローは負の最大値 (80000000H) を  $-1$  で割ったとき (商が 80000000H) と、ゼロによる除算のとき (演算結果は不定) に生じます。  
汎用レジスタ reg2 と汎用レジスタ reg3 が同じレジスタの場合、そのレジスタには余りが格納されます。  
この命令実行中に例外が発生すると、実行を中止し、戻り番地をこの命令の先頭アドレスとして例外を処理してから、例外処理完了後に再実行します。この場合、汎用レジスタ reg1 と汎用レジスタ reg2 はこの命令実行前の値を保持します。

**注意 1.** 汎用レジスタ reg2 と汎用レジスタ reg3 に同じレジスタを指定した場合、汎用レジスタ reg2 に演算結果の商が格納されないため、フラグは不定となります。

**2.** 命令形式 (1) の DIVH reg1, reg2 では、reg1, reg2 には r0 を指定しないでください

## &lt; 除算命令 &gt;

<p><b>DIVHU</b></p>	<p>Divide half-word unsigned</p> <p>(符号なし) ハーフワード・データの除算</p>
---------------------	--

[ 命令形式 ]            DIVHU reg1, reg2, reg3

[ オペレーション ]    GR [reg2] ← GR [reg2] ÷ GR [reg1]  
                           GR [reg3] ← GR [reg2] % GR [reg1]

[ フォーマット ]        Format XI

[ オペコード ]            15                            0 31                            16

rrrrrr111111RRRRR	wwwww01010000010
-------------------	------------------

[ フラグ ]                CY    -  
                           OV    オーバフローが起こったとき 1, そうでないとき 0  
                           S     演算結果の商のワード・データの MSB が 1 のとき 1, そうでないとき 0  
                           Z     演算結果の商が 0 のとき 1, そうでないとき 0  
                           SAT   -

[ 説 明 ]                汎用レジスタ reg2 のワード・データを汎用レジスタ reg1 の下位ハーフワード・データで除算し、その商を汎用レジスタ reg2 に、余りを汎用レジスタ reg3 に格納します。汎用レジスタ reg1 は影響を受けません。ゼロで割ったときは、オーバフローを生じ、OV フラグ以外の演算結果は不定となります。

[ 補 足 ]                オーバフローはゼロによる除算のとき (演算結果は不定) に生じます。  
                           汎用レジスタ reg2 と汎用レジスタ reg3 が同じレジスタの場合、そのレジスタには余りが格納されます。  
                           この命令実行中に例外が発生すると、実行を中止し、戻り番地をこの命令の先頭アドレスとして例外を処理してから、例外処理完了後に再実行します。この場合、汎用レジスタ reg1 と汎用レジスタ reg2 はこの命令実行前の値を保持します。

**注意**    汎用レジスタ reg2 と汎用レジスタ reg3 に同じレジスタを指定した場合、汎用レジスタ reg2 に演算結果の商が格納されないため、フラグは不定となります。

## &lt; 高速除算命令 &gt;

<div data-bbox="205 241 293 277" data-label="Text"> <p>DIVQ</p> </div>	<div data-bbox="1185 208 1394 235" data-label="Text"> <p>Divide word quickly</p> </div> <div data-bbox="887 295 1399 324" data-label="Text"> <p>(符号付き)ワード・データの除算(可変ステップ)</p> </div>
--	---

[ 命令形式 ]      DIVQ reg1, reg2, reg3

[ オペレーション ]    GR [reg2] ← GR [reg2] ÷ GR [reg1]  
                           GR [reg3] ← GR [reg2] % GR [reg1]

[ フォーマット ]      Format XI

[ オペコード ]        15                            0 31                            16

rrrrr111111RRRRR	wwwww01011111100
------------------	------------------

[ フラグ ]            CY    -  
                           OV    オーバフローが起こったとき 1, そうでないとき 0  
                           S     演算結果の商が負のとき 1, そうでないとき 0  
                           Z     演算結果の商が 0 のとき 1, そうでないとき 0  
                           SAT   -

[ 説 明 ]            汎用レジスタ reg2 のワード・データを汎用レジスタ reg1 のワード・データで除算し, その商を汎用レジスタ reg2 に, 余りを汎用レジスタ reg3 に格納します。汎用レジスタ reg1 は影響を受けません。

reg1, reg2 の値から除算に必要な最小なステップ数を判断して, 演算を実行します。ゼロで割ったときは, オーバフローを生じ, OV フラグ以外の演算結果は不定となります。

[ 補 足 ]            (1) オーバフローは負の最大値 (80000000H) を-1 で割ったとき (商が 80000000H) と, ゼロによる除算のとき (演算結果は不定) に生じます。  
                           汎用レジスタ reg2 と汎用レジスタ reg3 が同じレジスタの場合, そのレジスタには余りが格納されます。  
                           この命令実行中に例外が発生すると, 実行を中止し, 戻り番地をこの命令の先頭アドレスとして例外を処理してから, 例外処理完了後に再実行します。この場合, 汎用レジスタ reg1 と汎用レジスタ reg2 はこの命令実行前の値を保持します。

(2) 実行サイクル数は reg1, reg2 の有効ビット数の差が小さいほど少なく, ほとんどの場合に通常の除算命令より実行サイクル数が少なくなります。16 ビット整数型のデータ同士の除算の場合, 有効ビット数の差は 15 ビット以下であり, 20 サイクル以内で演算を完了します。

- 注意 1. 汎用レジスタ reg2 と汎用レジスタ reg3 に同じレジスタを指定した場合、汎用レジスタ reg2 に演算結果の商が格納されないため、フラグは不定となります。
2. 正確な実行サイクル数は、C. 2 命令実行クロック数を参照してください。
3. リアルタイム性の保証などのために、常に実行サイクル数が一定であることが必要な場合は、通常の除算命令を使用してください。



## &lt; 高速除算命令 &gt;

<div data-bbox="205 241 316 277" data-label="Text">DIVQU</div>	<div data-bbox="1099 208 1406 235" data-label="Text">Divide word unsigned quickly</div> <div data-bbox="887 295 1399 324" data-label="Text">(符号なし)ワード・データの除算(可変ステップ)</div>
--	--

[ 命令形式 ]            DIVQU reg1, reg2, reg3

[ オペレーション ]    GR [reg2] ← GR [reg2] ÷ GR [reg1]  
                           GR [reg3] ← GR [reg2] % GR [reg1]

[ フォーマット ]        Format XI

[ オペコード ]            15                            0 31                            16

rrrrrr111111RRRRR	wwwww010111111110
-------------------	-------------------

[ フラグ ]                CY    -  
                           OV    オーバフローが起こったとき 1, そうでないとき 0  
                           S     演算結果の商のワード・データの MSB が 1 のとき 1, そうでないとき 0  
                           Z     演算結果の商が 0 のとき 1, そうでないとき 0  
                           SAT   -

[ 説 明 ]                汎用レジスタ reg2 のワード・データを汎用レジスタ reg1 のワード・データで除算し, その商を汎用レジスタ reg2 に, 余りを汎用レジスタ reg3 に格納します。汎用レジスタ reg1 は影響を受けません。

reg1, reg2 の値から除算に必要な最小なステップ数を判断して, 演算を実行します。ゼロで割ったときは, オーバフローを生じ, OV フラグ以外の演算結果は不定となります。

[ 補 足 ]                (1) オーバフローはゼロによる除算のとき(演算結果は不定)に生じます。

                          汎用レジスタ reg2 と汎用レジスタ reg3 が同じレジスタの場合, そのレジスタには余りが格納されず。

                          この命令実行中に例外が発生すると, 実行を中止し, 戻り番地をこの命令の先頭アドレスとして例外を処理してから, 例外処理完了後に再実行します。この場合, 汎用レジスタ reg1 と汎用レジスタ reg2 はこの命令実行前の値を保持します。

                          (2) 実行サイクル数は reg1, reg2 の有効ビット数の差が小さいほど少なく, ほとんどの場合に通常の除算命令より実行サイクル数が少なくなります。16 ビット整数型のデータ同士の除算の場合, 有効ビット数の差は 15 ビット以下であり, 20 サイクル以内で演算を完了します。

**注意 1.** 汎用レジスタ reg2 と汎用レジスタ reg3 に同じレジスタを指定した場合, 汎用レジスタ reg2 に演算結果の商が格納されないため, フラグは不定となります。

**2.** 正確な実行サイクル数は, C. 2 命令実行クロック数を参照してください。

**3.** リアルタイム性の保証などのために, 常に実行サイクル数が一定であることが必要な場合は, 通常の除算命令を使用してください。

## &lt; 除算命令 &gt;

<div data-bbox="205 239 292 273" data-label="Text">DIVU</div>	<div data-bbox="1184 206 1406 230" data-label="Text">Divide word unsigned</div> <div data-bbox="1054 293 1406 320" data-label="Text">(符号なし)ワード・データの除算</div>
---	---

[ 命令形式 ]      DIVU reg1, reg2, reg3

[ オペレーション ]    GR [reg2] ← GR [reg2] ÷ GR [reg1]  
                           GR [reg3] ← GR [reg2] % GR [reg1]

[ フォーマット ]      Format XI

[ オペコード ]        15                            0 31                            16

rrrrrr111111RRRRR	wwwww01011000010
-------------------	------------------

[ フラグ ]            CY    -  
                           OV    オーバフローが起こったとき 1, そうでないとき 0  
                           S     演算結果の商のワード・データの MSB が 1 のとき 1, そうでないとき 0  
                           Z     演算結果の商が 0 のとき 1, そうでないとき 0  
                           SAT   -

[ 説 明 ]            汎用レジスタ reg2 のワード・データを汎用レジスタ reg1 のワード・データで除算し, その商を汎用レジスタ reg2 に, 余りを汎用レジスタ reg3 に格納します。汎用レジスタ reg1 は影響を受けません。  
                           ゼロで割ったときは, オーバフローを生じ, OV フラグ以外の演算結果は不定となります。

[ 補 足 ]            オーバフローはゼロによる除算のとき (演算結果は不定) に生じます。  
                           汎用レジスタ reg2 と汎用レジスタ reg3 が同じレジスタの場合, そのレジスタには余りが格納されません。  
                           この命令実行中に例外が発生すると, 実行を中止し, 戻り番地をこの命令の先頭アドレスとして例外を処理してから, 例外処理完了後に再実行します。この場合, 汎用レジスタ reg1 と汎用レジスタ reg2 はこの命令実行前の値を保持します。

**注意** 汎用レジスタ reg2 と汎用レジスタ reg3 に同じレジスタを指定した場合, 汎用レジスタ reg2 に演算結果の商が格納されないため, フラグは不定となります。

## &lt; 特殊命令 &gt;

EI	Enable interrupt  EI レベル・マスクブル例外の許可
----	---

[ 命令形式 ]      EI

[ オペレーション ]    PSW.ID ← 0 ( EI レベル・マスクブル例外の許可 )

[ フォーマット ]      Format X

[ オペコード ]

15	0 31	16
10000111111100000	0000000101100000	

[ フラ グ ]

CY	-
OV	-
S	-
Z	-
SAT	-
ID	0

[ 説 明 ]            PSW の ID ビットをクリア ( 0 ) し , 次の命令より EI レベル・マスクブル例外の受け付けを許可します。

## &lt; 特殊命令 &gt;

EIRET	Return from trap or interrupt  EI レベル例外からの復帰
-------	--

[ 命令形式 ]        EIRET

[ オペレーション ]    PC ← EIPC  
                          PSW ← EIPSW

[ フォーマット ]     Format X

[ オペコード ]        15                            0 31                            16

00000111111100000	0000000101001000
-------------------	------------------

[ フラグ ]            CY    EIPSW から読み出した値が設定される  
                          OV    EIPSW から読み出した値が設定される  
                          S     EIPSW から読み出した値が設定される  
                          Z     EIPSW から読み出した値が設定される  
                          SAT  EIPSW から読み出した値が設定される

[ 説 明 ]            EI レベル例外から復帰する命令です。EIPC, EIPSW から復帰 PC と PSW を取り出し, PC, PSW に設定し制御を移します。  
                          また, EP = 0 の場合, 例外ルーチンの実行を終了したことを外部 ( 割り込みコントローラなど ) に通知します。

## &lt; 特殊命令 &gt;

FERET	Return from trap or interrupt  FE レベル例外からの復帰
-------	--

[ 命令形式 ]            FERET

[ オペレーション ]    PC ← FEPC  
                          PSW ← FEPSW

[ フォーマット ]      Format X

[ オペコード ]        15                            0 31                            16

00000111111100000	0000000101001010
-------------------	------------------

[ フラグ ]            CY    FEPSW から読み出した値が設定される  
                          OV    FEPSW から読み出した値が設定される  
                          S     FEPSW から読み出した値が設定される  
                          Z     FEPSW から読み出した値が設定される  
                          SAT  FEPSW から読み出した値が設定される

[ 説 明 ]            FE レベル例外から復帰する命令です。FEPC, FEPSW から復帰 PC と PSW を取り出し, PC, PSW に設定し制御を移します。  
また, EP = 0 の場合, 例外ルーチンの実行を終了したことを外部 ( 割り込みコントローラなど ) に通知します。

< 特殊命令 >

FETRAP	FE-level Trap
FE レベル・ソフトウェア例外	

[ 命令形式 ]        FETRAP    vector4

[ オペレーション ]    FEPC ← PC + 2 (復帰 PC)  
                           FEPSW ← PSW  
                           ECR.FECC ← 例外要因コード ( 31H-3FH )  
                           FEIC ← 例外要因コード ( 31H-3FH )  
                           PSW.EP ← 1  
                           PSW.ID ← 1  
                           PSW.NP ← 1  
                           If (MPM.AUE==1) is satisfied  
                           then        PSW.IMP ← 0  
   PSW.DMP ← 0  
   PSW.NPV ← 0  
   PSW.PP ← 0  
                           PC ← 00000030H

[ フォーマット ]    Format I

[ オペコード ]        15                            0  
                           0vvvvv00001000000

ただし、vvvv は vector4 です。  
       また、vector4 には 0H を設定しないでください (vvvv    0000)。

[ フラ グ ]        CY    -  
                           OV    -  
                           S     -  
                           Z     -  
                           SAT   -

[ 説 明 ]        復帰 PC ( FETRAP 命令の次の命令のアドレス ) と現在の PSW の内容を、それぞれ FEPC と  
                           FEPSW に退避し、例外要因コードを FEIC レジスタと ECR.FECC ビットに格納、PSW.NP,  
                           EP, ID ビットをセット ( 1 ) します。また、MPM.AUE ビットがセット ( 1 ) されている場合、  
                           PSW.PP, NPV, DMP, IMP をクリア ( 0 ) します。  
                           続いて、例外ハンドラ・アドレス ( 00000030H ) に分岐し、例外処理を開始します。

## &lt; 特殊命令 &gt;

HALT	Halt
	停止

[ 命令形式 ]          HALT

[ オペレーション ]   HALT 状態の解除要求が発生するまで、後続命令の実行を停止する

[ フォーマット ]      Format X

[ オペコード ]        15                              0 31                              16

00000111111100000	0000000100100000
-------------------	------------------

[ フ ラ グ ]            CY    -  
                           OV    -  
                           S     -  
                           Z     -  
                           SAT  -

[ 説 明 ]              CPU を HALT 状態に移行します。  
 HALT 状態から通常の実行状態への復帰は、特定の例外要求の発生によって行われます。  
 なお、HALT 状態で例外を受け付けた場合、その例外の復帰 PC は、HALT 命令の次の命令の PC となります。

HALT 状態の解除要求の入力は、次の例外要求の発生時に行われます。

- リセット入力 (RESET)
- EI レベル・マスカブル割り込み入力 (INT0-INT255)
- FE レベル・マスカブル割り込み入力 (FEINT)
- FE レベル・ノン・マスカブル割り込み入力 (FENMI)
- 周辺装置保護例外 (PPI)
- タイミング監視例外 (TSI)
- システム・エラー例外 (SYSERR)

また、上記の例外の受け付け条件 (ID および NP の値) を満たしていない場合であっても、要求が存在する場合には HALT 状態の解除が行われます (例: ID=1 であっても、INT0 が発生した段階で HALT 状態が解除されます)。

[ 補 足 ]              割り込み入力 (INT0-INT255, FEINT, FENMI) が、割り込みコントローラの次に示すレジスタの機能によって、「割り込み処理を禁止」されている場合は、HALT 状態は解除されません。

- EI レベル割り込み制御レジスタ (EIC0-EIC255)
- EI レベル割り込みマスクレジスタ (IMR0-IMR15)

## &lt; データ操作命令 &gt;

HSH	Half-word swap half-word  ハーフワード・データのハーフワード・スワップ
-----	--

[ 命令形式 ]        HSH   reg2, reg3

[ オペレーション ]   GR [reg3] ← GR [reg2]

[ フォーマット ]     Format XII

[ オペコード ]        15                                  0 31                                  16  

rrrrrr111111100000	wwwww01101000110
--------------------	------------------

[ フラ グ ]        CY    演算結果の下位ハーフワードが 0 のとき 1, そうでないとき 0

OV    0

S     演算結果のワード・データの MSB が 1 のとき 1, そうでないとき 0

Z     演算結果の下位ハーフワードが 0 のとき 1, そうでないとき 0

SAT   -

[ 説 明 ]        汎用レジスタ reg2 の内容を汎用レジスタ reg3 に格納し, フラグの判定結果を PSW に格納します。



&lt; データ操作命令 &gt;

HSW

Half-word swap word

ワード・データのハーフワード・スワップ

[ 命令形式 ]        HSW reg2, reg3

[ オペレーション ] GR [reg3] ← GR [reg2] (15:0) || GR [reg2] (31:16)

[ フォーマット ]    Format XII

[ オペコード ]        15                    0 31                    16

rrrrrr11111100000	wwwww01101000100
-------------------	------------------

[ フラグ ]        CY    演算結果のワード・データ中に, 0 のハーフワードが 1 つ以上含まれるとき 1,  
                              そうでないとき 0

OV    0

S    演算結果のワード・データの MSB が 1 のとき 1, そうでないとき 0

Z    演算結果のワード・データが 0 のとき 1, そうでないとき 0

SAT -

[ 説 明 ]        エンディアン変換します。

## &lt; 分岐命令 &gt;

## JARL

Jump and register link

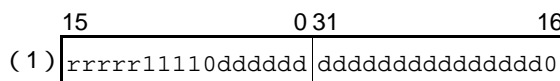
分岐とレジスタ・リンク

- [ 命令形式 ]           ( 1 ) JARL disp22, reg2  
                          ( 2 ) JARL disp32, reg1

- [ オペレーション ]   ( 1 ) GR [reg2] ← PC + 4  
                          PC ← PC + sign-extend (disp22)  
                          ( 2 ) GR [reg1] ← PC + 6  
                          PC ← PC + disp32

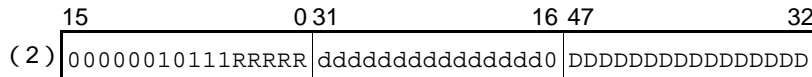
- [ フォーマット ]       ( 1 ) Format V  
                          ( 2 ) Format VI

## [ オペコード ]



ただし、dddddddddddddddddd は disp22 の上位 21 ビットです。

rrrrr   00000 ( reg2 には r0 を設定しないでください )



ただし、DDDDDDDDDDDDDDDDdddddddddd は disp32 の上位 31 ビットです。

RRRRR   00000 ( reg1 には r0 を設定しないでください )

- [ フラグ ]           CY   -  
                      OV   -  
                      S    -  
                      Z    -  
                      SAT -

- [ 説 明 ]           ( 1 ) 現在の PC に 4 を加算した値を汎用レジスタ reg2 に退避し、現在の PC とワード長まで符号拡張した 22 ビット・ディスプレースメントを加算した値を PC に設定し、制御を移します。22 ビット・ディスプレースメントのビット 0 は 0 にマスクされます。  
                      ( 2 ) 現在の PC に 6 を加算した値を汎用レジスタ reg1 に退避し、現在の PC と 32 ビット・ディスプレースメントを加算した値を PC に設定し、制御を移します。32 ビット・ディスプレースメントのビット 0 は 0 にマスクされます。

[ 補 足 ]          計算に使用される現在の PC とは、この命令自身の先頭バイトのアドレスであるためディスプレースメント値が 0 のときは、分岐先はこの命令自身になります。

                    この命令は、サブルーチン制御命令のコールに相当し、復帰 PC を汎用レジスタ reg1 または reg2 に格納します。一方、リターンに相当する JMP 命令では、復帰 PC を格納している汎用レジスタを汎用レジスタ reg1 として指定して、使用できます。

**注意**    命令形式 ( 1 ) JARL disp22, reg2 では、reg2 には r0 を指定しないでください。  
            命令形式 ( 2 ) JARL disp32, reg1 では、reg1 には r0 を指定しないでください。

## &lt; 分岐命令 &gt;

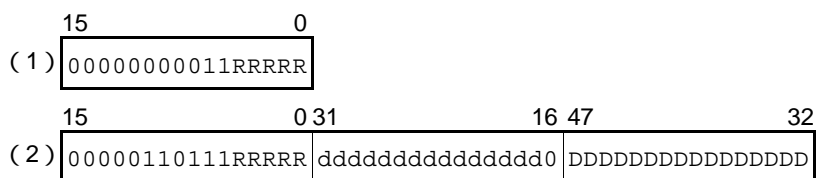
<p><b>JMP</b></p>	<p>Jump register</p> <p>無条件分岐 (レジスタ間接)</p>
-------------------	--

- [ 命令形式 ]           ( 1 ) JMP [reg1]  
                           ( 2 ) JMP disp32 [reg1]

- [ オペレーション ]   ( 1 ) PC ← GR [reg1]  
                           ( 2 ) PC ← GR [reg1] + disp32

- [ フォーマット ]      ( 1 ) Format I  
                           ( 2 ) Format VI

## [ オペコード ]



ただし, DDDDDDDDDDDDDDDdddddddddddddd は disp32 の上位 31 ビットです。

- [ フラグ ]           CY   -  
                           OV   -  
                           S    -  
                           Z    -  
                           SAT -

- [ 説 明 ]           ( 1 ) 汎用レジスタ reg1 で指定されるアドレスに制御を移します。アドレスのビット 0 は 0 にマスクされます。  
                           ( 2 ) 汎用レジスタ reg1 に 32 ビット・ディスプレースメントを加算したアドレスに制御を移します。アドレスのビット 0 は 0 にマスクされます。

- [ 補 足 ]           この命令をサブルーチン制御命令のリターンとして使用する場合は, 復帰 PC を汎用レジスタ reg1 で指定します。

## &lt; 分岐命令 &gt;

JR	Jump relative 無条件分岐 (PC 相対)
----	--------------------------------

[ 命令形式 ]           ( 1 ) JR   disp22  
                          ( 2 ) JR   disp32

[ オペレーション ]   ( 1 ) PC ← PC + sign-extend (disp22)  
                          ( 2 ) PC ← PC + disp32

[ フォーマット ]      ( 1 ) Format V  
                          ( 2 ) Format VI

[ オペコード ]

( 1 ) 

15	0 31	16
0000011110	dddddd	ddddddddddddddd0

ただし、dddddddddddddddddd は disp22 の上位 21 ビットです。

( 2 ) 

15	0 31	16 47	32
0000001011100000	ddddddddddddddd0	DDDDDDDDDDDDDDDD	DDDDDDDDDDDDDDDD

ただし、DDDDDDDDDDDDDDDDdddddddddddddd は disp32 の上位 31 ビットです。

[ フラグ ]           CY   -  
                      OV   -  
                      S    -  
                      Z    -  
                      SAT -

[ 説 明 ]           ( 1 ) 現在の PC とワード長まで符号拡張した 22 ビット・ディスプレースメントを加算した値を PC に設定し、制御を移します。22 ビット・ディスプレースメントのビット 0 は 0 にマスクされます。  
                      ( 2 ) 現在の PC と 32 ビット・ディスプレースメントを加算した値を PC に設定し、制御を移します。32 ビット・ディスプレースメントのビット 0 は 0 にマスクされます。

[ 補 足 ]           計算に使用される現在の PC とは、この命令自身の先頭バイトのアドレスであるため、ディスプレースメント値が 0 の場合の分岐先は、この命令自身になります。

## &lt; ロード命令 &gt;

LD.B	Load byte
( 符号付き ) バイト・データのロード	

- [ 命令形式 ]           ( 1 ) LD.B   disp16 [reg1] , reg2  
                           ( 2 ) LD.B   disp23 [reg1] , reg3

- [ オペレーション ]   ( 1 ) adr ← GR [reg1] + sign-extend (disp16)  
                           GR [reg2] ← sign-extend (Load-memory (adr, Byte) )  
                           ( 2 ) adr ← GR [reg1] + sign-extend (disp23)  
                           GR [reg3] ← sign-extend (Load-memory (adr, Byte) )

- [ フォーマット ]      ( 1 ) Format VII  
                           ( 2 ) Format XIV

- [ オペコード ]
- |                   |                 |    |
|-------------------|-----------------|----|
| 15                | 031             | 16 |
| rrrrrr111000RRRRR | ddddddddddddddd |    |
- ( 1 )
- |                  |                  |                  |    |
|------------------|------------------|------------------|----|
| 15               | 031              | 1647             | 32 |
| 00000111100RRRRR | wwwwwddddddd0101 | DDDDDDDDDDDDDDDD |    |
- ( 2 )
- ただし , RRRRR = reg1, wwwww = reg3 です。  
 ddddddd は , disp23 の下位 7 ビットです。  
 DDDDDDDDDDDDDDD は , disp23 の上位 16 ビットです。

- [ フラグ ]           CY   -  
                           OV   -  
                           S    -  
                           Z    -  
                           SAT -

- [ 説 明 ]           ( 1 ) 汎用レジスタ reg1 のワード・データとワード長まで符号拡張した 16 ビット・ディスプレイメントを加算して 32 ビット・アドレスを生成します。生成したアドレスからバイト・データを読み出し , ワード長まで符号拡張し , 汎用レジスタ reg2 に格納します。  
                           ( 2 ) 汎用レジスタ reg1 のワード・データとワード長まで符号拡張した 23 ビット・ディスプレイメントを加算して 32 ビット・アドレスを生成します。生成したアドレスからバイト・データを読み出し , ワード長まで符号拡張し , 汎用レジスタ reg3 に格納します。

## &lt; ロード命令 &gt;

LD.BU	Load byte unsigned  (符号なし)バイト・データのロード
-------	---

- [ 命令形式 ]           ( 1 ) LD.BU disp16 [reg1] , reg2  
                          ( 2 ) LD.BU disp23 [reg1] , reg3

- [ オペレーション ]   ( 1 ) adr ← GR [reg1] + sign-extend (disp16)  
                          GR [reg2] ← zero-extend (Load-memory (adr, Byte) )  
                          ( 2 ) adr ← GR [reg1] + sign-extend (disp23)  
                          GR [reg3] ← zero-extend (Load-memory (adr, Byte) )

- [ フォーマット ]      ( 1 ) Format VII  
                          ( 2 ) Format XIV

- [ オペコード ]
- |                   |                  |    |
|-------------------|------------------|----|
| 15                | 031              | 16 |
| rrrrrr11110bRRRRR | ddddddddddddddd1 |    |
- ただし、ddddddddddddddd は disp16 の上位 15 ビット、b は disp16 のビット 0 です。  
rrrrrr   00000 ( reg2 には r0 を設定しないでください )
- |                  |                   |                  |    |
|------------------|-------------------|------------------|----|
| 15               | 031               | 1647             | 32 |
| 00000111101RRRRR | wwwwwwddddddd0101 | DDDDDDDDDDDDDDDD |    |
- ただし、RRRRR = reg1, wwwww = reg3 です。  
ddddddd は、disp23 の下位 7 ビットです。  
DDDDDDDDDDDDDDDD は、disp23 の上位 16 ビットです。

- [ フラグ ]           CY   -  
                      OV   -  
                      S    -  
                      Z    -  
                      SAT -

- [ 説 明 ]           ( 1 ) 汎用レジスタ reg1 のワード・データとワード長まで符号拡張した 16 ビット・ディスプレイメントを加算して 32 ビット・アドレスを生成します。生成したアドレスからバイト・データを読み出し、ワード長までゼロ拡張し、汎用レジスタ reg2 に格納します。  
                      ( 2 ) 汎用レジスタ reg1 のワード・データとワード長まで符号拡張した 23 ビット・ディスプレイメントを加算して 32 ビット・アドレスを生成します。生成したアドレスからバイト・データを読み出し、ワード長までゼロ拡張し、汎用レジスタ reg3 に格納します。

**注意** reg2 には、r0 を指定しないでください。

## &lt; ロード命令 &gt;

LD.H	Load half-word
( 符号付き ) ハーフワード・データのロード	

- [ 命令形式 ]           ( 1 ) LD.H   disp16 [reg1] , reg2  
                          ( 2 ) LD.H   disp23 [reg1] , reg3

- [ オペレーション ]   ( 1 ) adr ← GR [reg1] + sign-extend (disp16)  
                          GR [reg2] ← sign-extend (Load-memory (adr, Half-word) )  
                          ( 2 ) adr ← GR [reg1] + sign-extend (disp23)  
                          GR [reg3] ← sign-extend (Load-memory (adr, Half-word) )

- [ フォーマット ]      ( 1 ) Format VII  
                          ( 2 ) Format XIV

- [ オペコード ]
- ( 1 ) 

15	031	16
rrrrrr1111001RRRRR	ddddddddddddddd0	

  
ただし、ddddddddddddddd は disp16 の上位 15 ビットです。
- ( 2 ) 

15	031	1647	32
00000111100RRRRR	wwwww	dddddd00111	DDDDDDDDDDDDDDDD

  
ただし、RRRRR = reg1, wwwww = reg3 です。  
dddddd は、disp23 の下位側ビット 6-1 です。  
DDDDDDDDDDDDDDDD は、disp23 の上位 16 ビットです。

- [ フラグ ]           CY   -  
                          OV   -  
                          S    -  
                          Z    -  
                          SAT -

- [ 説 明 ]           ( 1 ) 汎用レジスタ reg1 のワード・データとワード長まで符号拡張した 16 ビット・ディスプレイメントを加算して 32 ビット・アドレスを生成します。生成したアドレスからハーフワード・データを読み出し、ワード長まで符号拡張し、汎用レジスタ reg2 に格納します。
- ( 2 ) 汎用レジスタ reg1 のワード・データとワード長まで符号拡張した 23 ビット・ディスプレイメントを加算して 32 ビット・アドレスを生成します。生成したアドレスからハーフワード・データを読み出し、ワード長まで符号拡張し、汎用レジスタ reg3 に格納します。



## &lt; ロード命令 &gt;

LD.HU	Load half-word unsigned  (符号なし) ハーフワード・データのロード
-------	--

- [ 命令形式 ]           (1) LD.HU disp16 [reg1], reg2  
                          (2) LD.HU disp23 [reg1], reg3

- [ オペレーション ]   (1) adr ← GR [reg1] + sign-extend (disp16)  
                          GR [reg2] ← zero-extend (Load-memory (adr, Half-word))  
                          (2) adr ← GR [reg1] + sign-extend (disp23)  
                          GR [reg3] ← zero-extend (Load-memory (adr, Half-word))

- [ フォーマット ]      (1) Format VII  
                          (2) Format XIV

- [ オペコード ]
- (1) 

15		031		16
rrrrrr111111RRRRR		ddddddddddddddd1		

  
ただし、ddddddddddddddd は disp16 の上位 15 ビットです。  
rrrrrr 00000 (reg2 には r0 を設定しないでください)
- (2) 

15		031		1647		32
00000111101RRRRR		wwwwwdddddd00111		DDDDDDDDDDDDDDDD		

  
ただし、RRRRR = reg1, wwwww = reg3 です。  
dddddd は、disp23 の下位側ビット 6-1 です。  
DDDDDDDDDDDDDDDD は、disp23 の上位 16 ビットです。

- [ フラグ ]           CY   -  
                          OV   -  
                          S    -  
                          Z    -  
                          SAT -

- [ 説 明 ]           (1) 汎用レジスタ reg1 のワード・データとワード長まで符号拡張した 16 ビット・ディスプレイメントを加算して 32 ビット・アドレスを生成します。生成した 32 ビット・アドレスからハーフワード・データを読み出し、ワード長までゼロ拡張し、汎用レジスタ reg2 に格納します。  
                          (2) 汎用レジスタ reg1 のワード・データとワード長まで符号拡張した 23 ビット・ディスプレイメントを加算して 32 ビット・アドレスを生成します。生成したアドレスからハーフワード・データを読み出し、ワード長までゼロ拡張し、汎用レジスタ reg3 に格納します。

**注意** reg2 には、r0 を指定しないでください。

## &lt; ロード命令 &gt;

LD.W	Load word  ワード・データのロード
------	------------------------------

- [ 命令形式 ]           ( 1 ) LD.W   disp16 [reg1] , reg2  
                          ( 2 ) LD.W   disp23 [reg1] , reg3

- [ オペレーション ]   ( 1 ) adr ← GR [reg1] + sign-extend (disp16)  
                          GR [reg2] ← Load-memory (adr, Word)  
                          ( 2 ) adr ← GR [reg1] + sign-extend (disp23)  
                          GR [reg3] ← Load-memory (adr, Word)

- [ フォーマット ]      ( 1 ) Format VII  
                          ( 2 ) Format XIV

- [ オペコード ]
- |                    |                  |    |
|--------------------|------------------|----|
| 15                 | 031              | 16 |
| rrrrrr1111001RRRRR | ddddddddddddddd1 |    |
- ただし , ddddddddddddddd は disp16 の上位 15 ビットです。
- |                  |                  |                  |    |
|------------------|------------------|------------------|----|
| 15               | 031              | 1647             | 32 |
| 00000111100RRRRR | wwwwwdddddd01001 | DDDDDDDDDDDDDDDD |    |
- ただし , RRRRR = reg1, wwwww = reg3 です。  
dddddd は , disp23 の下位側ビット 6-1 です。  
DDDDDDDDDDDDDDDD は , disp23 の上位 16 ビットです。

- [ フラグ ]           CY   -  
                      OV   -  
                      S    -  
                      Z    -  
                      SAT -

- [ 説 明 ]           ( 1 ) 汎用レジスタ reg1 のワード・データとワード長まで符号拡張した 16 ビット・ディスプレイメントを加算して 32 ビット・アドレスを生成します。生成した 32 ビット・アドレスからワード・データを読み出し , 汎用レジスタ reg2 に格納します。  
                      ( 2 ) 汎用レジスタ reg1 のワード・データとワード長まで符号拡張した 23 ビット・ディスプレイメントを加算して 32 ビット・アドレスを生成します。生成したアドレスからワード・データを読み出し , 汎用レジスタ reg3 に格納します。

## &lt; 特殊命令 &gt;

LDSR	Load to system register  システム・レジスタへのロード
------	---

[ 命令形式 ]        LDSR reg2, regID

[ オペレーション ]    SR [regID] ← GR [reg2]

[ フォーマット ]     Format IX

[ オペコード ]        15                                  0 31                                  16

rrrrr111111RRRRR	0000000000100000
------------------	------------------

**注意** この命令では、ニモニク記述の都合上、ソース・レジスタを reg2 としていますが、オペコード上は reg1 のフィールドを使用しています。したがって、ニモニク記述とオペコードにおいて、レジスタ指定の意味付けがほかの命令と異なります。

rrrrr : regID 指定

RRRRR : reg2 指定

[ フラグ ]            CY    -  
                      OV    -  
                      S     -  
                      Z     -  
                      SAT  -

[ 説 明 ]            汎用レジスタ reg2 のワード・データをシステム・レジスタ番号 ( regID ) で指定されるシステム・レジスタに設定します。汎用レジスタ reg2 は影響を受けません。

**注意** システム・レジスタ番号は、システム・レジスタを一意に識別するための番号です。予約されているシステム・レジスタ、書き込み禁止のシステム・レジスタに対してこの命令を実行した場合の動作は保証しません。

## &lt; 加算付き乗算命令 &gt;

MAC	Multiply and add word  (符号付き)ワード・データの加算付き乗算
-----	---

[ 命令形式 ]           MAC reg1, reg2, reg3, reg4

[ オペレーション ]   GR [reg4+1] || GR [reg4] ← GR [reg2] × GR [reg1] + GR [reg3+1] || GR [reg3]

[ フォーマット ]       Format XI

[ オペコード ]         15                                 0 31                                 16

rrrrr111111RRRRR	www0011110mmmm0
------------------	-----------------

[ フラ グ ]           CY    -

                      OV    -

                      S     -

                      Z     -

                      SAT  -

[ 説 明 ]           汎用レジスタ reg2 のワード・データに汎用レジスタ reg1 のワード・データを乗算した結果 (64 ビット・データ) と、汎用レジスタ reg3 を下位 32 ビットとして、汎用レジスタ reg3+1 (たとえば、reg3 が r6 の場合、「reg3+1」は r7 となります) を上位 32 ビットとして結合した 64 ビット・データを加算し、その結果 (64 ビット・データ) の上位 32 ビットを汎用レジスタ reg4+1 に、下位 32 ビットを汎用レジスタ reg4 に格納します。

汎用レジスタ reg1, reg2 の内容を 32 ビットの符号付き整数として扱います。

汎用レジスタ reg1, reg2, reg3, reg3+1 は影響を受けません。

**注意** reg3, または reg4 に指定できる汎用レジスタは、偶数番号の付いたレジスタ (r0, r2, r4, ..., r30) だけです。奇数番号の付いたレジスタ (r1, r3, ..., r31) を指定した場合の結果は不定です。

## &lt; 加算付き乗算命令 &gt;

MACU	Multiply and add word unsigned  (符号なし)ワード・データの加算付き乗算
------	--

[ 命令形式 ]           MACU reg1, reg2, reg3, reg4

[ オペレーション ]   GR [reg4+1] || GR [reg4] ← GR [reg2] × GR [reg1] + GR [reg3+1] || GR [reg3]

[ フォーマット ]       Format XI

[ オペコード ]         15                                 0 31                                 16

rrrrr111111RRRRR	www00111111mmmm0
------------------	------------------

[ フラ グ ]           CY    -

                      OV    -

                      S     -

                      Z     -

                      SAT  -

[ 説 明 ]           汎用レジスタ reg2 のワード・データに汎用レジスタ reg1 のワード・データを乗算した結果 (64 ビット・データ) と、汎用レジスタ reg3 を下位 32 ビットとして、汎用レジスタ reg3+1 (たとえば、reg3 が r6 の場合、「reg3+1」は r7 となります) を上位 32 ビットとして結合した 64 ビット・データを加算し、その結果 (64 ビット・データ) の上位 32 ビットを汎用レジスタ reg4+1 に、下位 32 ビットを汎用レジスタ reg4 に格納します。

汎用レジスタ reg1, reg2 の内容を 32 ビットの符号なし整数として扱います。

汎用レジスタ reg1, reg2, reg3, reg3+1 は影響を受けません。

**注意** reg3, または reg4 に指定できる汎用レジスタは、偶数番号の付いたレジスタ (r0, r2, r4, ..., r30) だけです。奇数番号の付いたレジスタ (r1, r3, ..., r31) を指定した場合の結果は不定です。

## &lt; 算術演算命令 &gt;

MOV	Move register/immediate (5-bit) /immediate (32-bit)
	データの転送

- [ 命令形式 ]
- ( 1 ) MOV reg1, reg2
  - ( 2 ) MOV imm5, reg2
  - ( 3 ) MOV imm32, reg1

- [ オペレーション ]
- ( 1 ) GR [reg2] ← GR [reg1]
  - ( 2 ) GR [reg2] ← sign-extend (imm5)
  - ( 3 ) GR [reg1] ← imm32

- [ フォーマット ]
- ( 1 ) Format I
  - ( 2 ) Format II
  - ( 3 ) Format VI

- [ オペコード ]
- ( 1 )
- |                   |   |
|-------------------|---|
| 15                | 0 |
| rrrrrr000000RRRRR |   |
- rrrrrr 00000 ( reg2にはr0を設定しないでください )
- ( 2 )
- |                    |   |
|--------------------|---|
| 15                 | 0 |
| rrrrrr010000iiiiii |   |
- rrrrrr 00000 ( reg2にはr0を設定しないでください )
- ( 3 )
- |   |      |       |    |
|---|------|-------|----|
| 15  | 0 31 | 16 47 | 32 |
| 00000110001RRRRR   iiiiiiiiiiiiiiiiii   Iiiiiiiiiiiiiiiiiii |      |       |    |
- i ( ビット 31-16 ) は 32 ビット・イミディエト・データの下位 16 ビットです。  
I ( ビット 47-32 ) は 32 ビット・イミディエト・データの上位 16 ビットです。

- [ フラグ ]
- CY -
  - OV -
  - S -
  - Z -
  - SAT -

- [ 説 明 ]
- ( 1 ) 汎用レジスタ reg1 のワード・データを、汎用レジスタ reg2 にコピーし転送します。  
汎用レジスタ reg1 は影響を受けません。
  - ( 2 ) 5 ビット・イミディエトをワード長まで符号拡張した値を、汎用レジスタ reg2 にコピーし転送します。
  - ( 3 ) 32 ビット・イミディエトを、汎用レジスタ reg1 にコピーし転送します。

**注意** 命令形式 ( 1 ) の MOV reg1, reg2 と命令形式 ( 2 ) の MOV imm5, reg2 では、reg2 には r0 を指定しないでください。

## &lt; 算術演算命令 &gt;

MOVEA	Move effective address  実行アドレスの転送
-------	---

[ 命令形式 ]        MOVEA imm16, reg1, reg2

[ オペレーション ] GR [reg2] ← GR [reg1] + sign-extend (imm16)

[ フォーマット ]    Format VI

[ オペコード ]

15	0 31	16
rrrrr110001RRRRR   iiiiiiiiiiiiiiiiiii		

rrrrr    00000 ( reg2 には r0 を設定しないでください )

[ フラ グ ]

CY	-
OV	-
S	-
Z	-
SAT	-

[ 説 明 ]        汎用レジスタ reg1 のワード・データにワード長まで符号拡張した 16 ビット・イミディエトを加算し、その結果を汎用レジスタ reg2 に格納します。汎用レジスタ reg1 は影響を受けません。加算によってもフラグは変化しません。

[ 補 足 ]        32 ビット・アドレスを計算する際、フラグを変化させたくない場合に、この命令を使用します。

**注意**    reg2 には、r0 を指定しないでください。

## &lt; 算術演算命令 &gt;

MOVHI	Move high half-word  上位ハーフワードの転送
-------	--

[ 命令形式 ]        MOVHI imm16, reg1, reg2

[ オペレーション ] GR [reg2] ← GR [reg1] + (imm16 || 0<sup>16</sup>)

[ フォーマット ]    Format VI

[ オペコード ]

15	0 31	16
rrrrrr110010RRRRR   iiii		

rrrrrr    00000 ( reg2 には r0 を設定しないでください )

[ フラ グ ]

CY	-
OV	-
S	-
Z	-
SAT	-

[ 説 明 ]        汎用レジスタ reg1 のワード・データに、上位 16 ビットが 16 ビット・イミューディエト、下位 16 ビットが 0 であるワード・データを加算し、その結果を汎用レジスタ reg2 に格納します。汎用レジスタ reg1 は影響を受けません。加算によってもフラグは変化しません。

[ 補 足 ]        32 ビット・アドレスの上位 16 ビットの生成にこの命令を使用します。

<b>注意</b> reg2 には、r0 を指定しないでください。
-----------------------------------



## &lt; 乗算命令 &gt;

<p>MUL</p>	<p>Multiply word by register/immediate (9-bit)</p> <p>(符号付き)ワード・データの乗算</p>
------------	--

- [ 命令形式 ]           ( 1 ) MUL reg1, reg2, reg3  
                           ( 2 ) MUL imm9, reg2, reg3

- [ オペレーション ]   ( 1 ) GR [reg3] || GR [reg2] ← GR [reg2] × GR [reg1]  
                           ( 2 ) GR [reg3] || GR [reg2] ← GR [reg2] × sign-extend (imm9)

- [ フォーマット ]      ( 1 ) Format XI  
                           ( 2 ) Format XII

- [ オペコード ]
- |                   |  |    |      |    |                   |                  |  |
|-------------------|--|----|------|----|-------------------|------------------|--|
| ( 1 )             | <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="text-align: right; padding-right: 5px;">15</td> <td style="text-align: center; padding: 0 10px;">0 31</td> <td style="text-align: left; padding-left: 5px;">16</td> </tr> <tr> <td style="border-right: 1px solid black; padding: 2px;">rrrrrr111111RRRRR</td> <td style="border-right: 1px solid black; padding: 2px;">wwwww01000100000</td> <td></td> </tr> </table> | 15 | 0 31 | 16 | rrrrrr111111RRRRR | wwwww01000100000 |  |
| 15                | 0 31   | 16 |      |    |                   |                  |  |
| rrrrrr111111RRRRR | wwwww01000100000   |    |      |    |                   |                  |  |
| ( 2 )             | <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="text-align: right; padding-right: 5px;">15</td> <td style="text-align: center; padding: 0 10px;">0 31</td> <td style="text-align: left; padding-left: 5px;">16</td> </tr> <tr> <td style="border-right: 1px solid black; padding: 2px;">rrrrrr111111iiii</td> <td style="border-right: 1px solid black; padding: 2px;">wwwww01001IIII00</td> <td></td> </tr> </table>  | 15 | 0 31 | 16 | rrrrrr111111iiii  | wwwww01001IIII00 |  |
| 15                | 0 31   | 16 |      |    |                   |                  |  |
| rrrrrr111111iiii  | wwwww01001IIII00   |    |      |    |                   |                  |  |
- iiii は、9 ビット・イミディエト・データの低位 5 ビットです。  
 IIII は、9 ビット・イミディエト・データの上位 4 ビットです。

- [ フラグ ]           CY    -  
                           OV    -  
                           S     -  
                           Z     -  
                           SAT  -

- [ 説 明 ]           ( 1 ) 汎用レジスタ reg2 のワード・データに汎用レジスタ reg1 のワード・データを乗算し、その結果 (64 ビット・データ) の上位 32 ビットを汎用レジスタ reg3 に、下位 32 ビットを汎用レジスタ reg2 に格納します。  
                           reg1, reg2 の内容を 32 ビットの符号付き整数として扱います。汎用レジスタ reg1 は影響を受けません。  
                           ( 2 ) 汎用レジスタ reg2 のワード・データにワード長まで符号拡張した 9 ビット・イミディエト・データを乗算し、その結果 (64 ビット・データ) の上位 32 ビットを汎用レジスタ reg3 に、下位 32 ビットを汎用レジスタ reg2 に格納します。

- [ 補 足 ]           汎用レジスタ reg2 と汎用レジスタ reg3 が同じレジスタの場合、そのレジスタには乗算結果の上位 32 ビットが格納されます。

## &lt; 乗算命令 &gt;

<h1>MULH</h1>	Multiply half-word by register/immediate (5-bit)  (符号付き) ハーフワード・データの乗算
---------------	--

[ 命令形式 ]           ( 1 ) MULH reg1, reg2  
                          ( 2 ) MULH imm5, reg2

[ オペレーション ]   ( 1 ) GR [reg2] (32) ← GR [reg2] (16) × GR [reg1] (16)  
                          ( 2 ) GR [reg2] ← GR [reg2] × sign-extend (imm5)

[ フォーマット ]       ( 1 ) Format I  
                          ( 2 ) Format II

[ オペコード ]

( 1 ) 

15	0
rrrrrr	000111RRRRR

  
rrrrrr   00000 ( reg2 には r0 を設定しないでください )

( 2 ) 

15	0
rrrrrr	010111iiiiii

  
rrrrrr   00000 ( reg2 には r0 を設定しないでください )

[ フラグ ]           CY   -  
                      OV   -  
                      S    -  
                      Z    -  
                      SAT -

[ 説 明 ]           ( 1 ) 汎用レジスタ reg2 の下位ハーフワード・データに汎用レジスタ reg1 の下位ハーフワード・データを乗算し、その結果を汎用レジスタ reg2 に格納します。  
                      汎用レジスタ reg1 は影響を受けません。  
                      ( 2 ) 汎用レジスタ reg2 の下位ハーフワード・データにハーフワード長まで符号拡張した 5 ビット・イミューディエトを乗算し、その結果を汎用レジスタ reg2 に格納します。

[ 補 足 ]           乗数, 被乗数の場合, 汎用レジスタ reg1, reg2 の上位 16 ビットを無視します。

**注意** reg2 には, r0 を指定しないでください。

## &lt; 乗算命令 &gt;

<p><b>MULHI</b></p>	<p>Multiply half-word by immediate (16-bit)</p> <p>(符号付き) ハーフワード・イミディエトの乗算</p>
---------------------	--

[ 命令形式 ]      MULHI imm16, reg1, reg2

[ オペレーション ]   GR [reg2] ← GR [reg1] × imm16

[ フォーマット ]    Format VI

[ オペコード ]

15	0 31	16
rrrrrr110111RRRRR	iiiiiiiiiiiiiiiiiiii	

rrrrrr    00000 ( reg2 には r0 を設定しないでください )

[ フラグ ]

CY	-
OV	-
S	-
Z	-
SAT	-

[ 説 明 ]      汎用レジスタ reg1 の下位ハーフワード・データに、16 ビット・イミディエトを乗算し、その結果を汎用レジスタ reg2 に格納します。汎用レジスタ reg1 は影響を受けません。

[ 補 足 ]      被乗数の場合、汎用レジスタ reg1 の上位 16 ビットを無視します。

**注意**    reg2 には、r0 を指定しないでください。

## &lt; 乗算命令 &gt;

<p>MULU</p>	<p>Multiply word unsigned by register/immediate (9-bit)</p> <p>(符号なし)ワード・データの乗算</p>
-------------	---

- [ 命令形式 ]           ( 1 ) MULU reg1, reg2, reg3  
                           ( 2 ) MULU imm9, reg2, reg3

- [ オペレーション ]   ( 1 ) GR [reg3] || GR [reg2] ← GR [reg2] × GR [reg1]  
                           ( 2 ) GR [reg3] || GR [reg2] ← GR [reg2] × zero-extend (imm9)

- [ フォーマット ]      ( 1 ) Format XI  
                           ( 2 ) Format XII

- [ オペコード ]
- |                   |  |    |      |    |                   |                  |  |
|-------------------|--|----|------|----|-------------------|------------------|--|
| ( 1 )             | <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 15%; text-align: right;">15</td> <td style="width: 50%; text-align: center;">0 31</td> <td style="width: 15%; text-align: right;">16</td> </tr> <tr> <td style="border-right: 1px solid black; padding: 2px;">rrrrrr111111RRRRR</td> <td style="padding: 2px;">wwwww01000100010</td> <td></td> </tr> </table> | 15 | 0 31 | 16 | rrrrrr111111RRRRR | wwwww01000100010 |  |
| 15                | 0 31   | 16 |      |    |                   |                  |  |
| rrrrrr111111RRRRR | wwwww01000100010   |    |      |    |                   |                  |  |
| ( 2 )             | <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 15%; text-align: right;">15</td> <td style="width: 50%; text-align: center;">0 31</td> <td style="width: 15%; text-align: right;">16</td> </tr> <tr> <td style="border-right: 1px solid black; padding: 2px;">rrrrrr111111iiii</td> <td style="padding: 2px;">wwwww01001IIII10</td> <td></td> </tr> </table>  | 15 | 0 31 | 16 | rrrrrr111111iiii  | wwwww01001IIII10 |  |
| 15                | 0 31   | 16 |      |    |                   |                  |  |
| rrrrrr111111iiii  | wwwww01001IIII10   |    |      |    |                   |                  |  |
- iiii は、9ビット・イミディエト・データの下位5ビットです。  
 IIII は、9ビット・イミディエト・データの上位4ビットです。

- [ フラ グ ]           CY    -  
                           OV    -  
                           S     -  
                           Z     -  
                           SAT  -

- [ 説 明 ]           ( 1 ) 汎用レジスタ reg2 のワード・データに汎用レジスタ reg1 のワード・データを乗算し、その結果 (64 ビット・データ) の上位 32 ビットを汎用レジスタ reg3 に、下位 32 ビットを汎用レジスタ reg2 に格納します。  
                           汎用レジスタ reg1 は影響を受けません。  
                           ( 2 ) 汎用レジスタ reg2 のワード・データにワード長までゼロ拡張した 9 ビット・イミディエト・データを乗算し、その結果 (64 ビット・データ) の上位 32 ビットを汎用レジスタ reg3 に、下位 32 ビットを汎用レジスタ reg2 に格納します。

- [ 補 足 ]           汎用レジスタ reg2 と汎用レジスタ reg3 が同じレジスタの場合、そのレジスタには乗算結果の上位 32 ビットが格納されます。



## &lt; 論理演算命令 &gt;

NOT	NOT
論理否定 (1 の補数をとる)	

[ 命令形式 ]            NOT reg1, reg2

[ オペレーション ]    GR [reg2] ← NOT (GR [reg1])

[ フォーマット ]       Format I

[ オペコード ]         15                          0  
                         rrrrrr000001RRRRR

[ フラ グ ]            CY    -  
                         OV    0  
                         S     演算結果のワード・データの MSB が 1 のとき 1, そうでないとき 0  
                         Z     演算結果が 0 のとき 1, そうでないとき 0  
                         SAT   -

[ 説 明 ]               汎用レジスタ reg1 のワード・データの論理否定 (1 の補数) をとり, その結果を汎用レジスタ reg2 に格納します。汎用レジスタ reg1 は影響を受けません。

## &lt; ビット操作命令 &gt;

NOT1	NOT bit  ビット・ノット
------	------------------------

- [ 命令形式 ]           ( 1 ) NOT1 bit#3, disp16 [reg1]  
                          ( 2 ) NOT1 reg2, [reg1]

- [ オペレーション ]   ( 1 ) adr   GR[reg1] + sign-extend (disp16)  
                          token   Load-memory (adr, Byte)  
                          Zフラグ   Not(extract-bit (token, bit#3) )  
                          token   not-bit(token, bit#3)  
                          Store-memory (adr, token, Byte)  
                          ( 2 ) adr   GR [reg1]  
                          token   Load-memory (adr, Byte)  
                          Zフラグ   Not(extract-bit (token, reg2) )  
                          token   not-bit (token, reg2)  
                          Store-memory (adr, token, Byte)

- [ フォーマット ]      ( 1 ) Format VIII  
                          ( 2 ) Format IX

- [ オペコード ]
- |       |                  |                  |
|-------|------------------|------------------|
| 15    | 0 31             | 16               |
| ( 1 ) | 01bbb111110RRRRR | ddddddddddddddd  |
| 15    | 0 31             | 16               |
| ( 2 ) | rrrrr11111RRRRR  | 0000000011100010 |

- [ フラグ ]           CY    -  
                          OV    -  
                          S     -  
                          Z     指定したビットが 0 のとき 1, 指定したビットが 1 のとき 0  
                          SAT   -

- [ 説 明 ]           ( 1 ) まず、汎用レジスタ reg1 のワード・データと、ワード長まで符号拡張した 16 ビット・ディスプレイメントを加算して 32 ビット・アドレスを生成します。生成したアドレスのバイト・データを読み出し、3 ビットのビット・ナンバで指定されるビットを反転 (0 1, 1 0) し、元のアドレスに書き戻します。  
読み出したバイト・データの指定ビットが 0 のとき Z フラグをセット (1) し、指定ビットが 1 のとき Z フラグをクリア (0) します。  
( 2 ) まず、汎用レジスタ reg1 のワード・データを読み出して 32 ビット・アドレスを生成します。生成したアドレスのバイト・データを読み出し、汎用レジスタ reg2 の下位 3 ビットで指定されるビットを反転 (0 1, 1 0) し、元のアドレスに書き戻します。  
読み出したバイト・データの指定ビットが 0 のとき Z フラグをセット (1) し、指定ビットが 1 のとき Z フラグをクリア (0) します。

[ 補 足 ] PSW の Z フラグはこの命令を実行する前に該当ビットが 0 か 1 だったかを示します。この命令実行後の該当ビットの内容を示すものではありません。

**注意** この命令は排他制御を目的としたアトミック性保証のため、読み出しから書き込みまでの間、対象のアドレスが他の要因によるアクセスによって操作されることはありません。



## &lt; 論理演算命令 &gt;

OR	OR 論理和
----	-----------

[ 命令形式 ]        OR reg1, reg2

[ オペレーション ] GR [reg2] ← GR [reg2] OR GR [reg1]

[ フォーマット ]     Format I

[ オペコード ]       15                          0  
                         rrrrrr001000RRRRR

[ フラ グ ]        CY    -  
                      OV    0  
                      S     演算結果のワード・データの MSB が 1 のとき 1, そうでないとき 0  
                      Z     演算結果が 0 のとき 1, そうでないとき 0  
                      SAT   -

[ 説 明 ]        汎用レジスタ reg2 のワード・データと汎用レジスタ reg1 のワード・データの論理和をとり, その結果を汎用レジスタ reg2 に格納します。汎用レジスタ reg1 は影響を受けません。

## &lt; 論理演算命令 &gt;

ORI	OR immediate (16-bit)
	論理和

[ 命令形式 ]      ORI   imm16, reg1, reg2

[ オペレーション ]   GR [reg2] ← GR [reg1] OR zero-extend (imm16)

[ フォーマット ]    Format VI

[ オペコード ]

15	0 31	16
rrrrrr110100RRRRR	iiiiiiiiiiiiiiiiiii	

[ フラ グ ]

CY    -

OV    0

S     演算結果のワード・データの MSB が 1 のとき 1, そうでないとき 0

Z     演算結果が 0 のとき 1, そうでないとき 0

SAT   -

[ 説 明 ]            汎用レジスタ reg1 のワード・データと 16 ビット・イミディエトをワード長までゼロ拡張した値の論理和をとり, その結果を汎用レジスタ reg2 に格納します。汎用レジスタ reg1 は影響を受けません。

## &lt; 特殊命令 &gt;

PREPARE	Function prepare  スタック・フレームの生成
---------	--------------------------------------

- [ 命令形式 ]       ( 1 ) PREPARE list12, imm5  
                      ( 2 ) PREPARE list12, imm5, sp/imm<sup>注</sup>

**注** sp/imm の値は、サブオペコードのビット 19, ビット 20 で指定します。

- [ オペレーション ]   ( 1 ) adr    sp  
                      foreach (all regs in list12) {  
                          adr    adr - 4  
                          Store-memory (adr, GR[reg in list12], Word )<sup>注</sup>  
                      }  
                      sp    adr - zero-extend (imm5 logically shift left by 2)
- ( 2 ) adr    sp  
                      foreach (all regs in list12) {  
                          adr    adr - 4  
                          Store-memory (adr, GR[reg in list12], Word )<sup>注</sup>  
                      }  
                      sp    adr - zero-extend (imm5 logically shift left by 2)
- case  
                          ff = 00: ep    sp  
                          ff = 01: ep    sign-extend (imm16)  
                          ff = 10: ep    imm16 logically shift left by 16  
                          ff = 11: ep    imm32

**注** Store-memory 時に adr の下位 2 ビットは 0 にマスクされます。

- [ フォーマット ]    Format XIII

## [ オペコード ]

- |                                       |   |    |      |    |                                       |  |  |
|---------------------------------------|---|----|------|----|---------------------------------------|--|--|
| ( 1 )                                 | <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 15%; text-align: right;">15</td> <td style="width: 60%; text-align: center;">0 31</td> <td style="width: 25%; text-align: left;">16</td> </tr> <tr> <td colspan="3" style="border: 1px solid black; padding: 2px;">0000011110iiiiiiL LLLLLLLLLLLLLL00001</td> </tr> </table> | 15 | 0 31 | 16 | 0000011110iiiiiiL LLLLLLLLLLLLLL00001 |  |  |
| 15                                    | 0 31  | 16 |      |    |                                       |  |  |
| 0000011110iiiiiiL LLLLLLLLLLLLLL00001 |   |    |      |    |                                       |  |  |
- |                                       |   |    |      |    |                                       |  |  |                           |
|---------------------------------------|---|----|------|----|---------------------------------------|--|--|---------------------------|
| ( 2 )                                 | <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 15%; text-align: right;">15</td> <td style="width: 60%; text-align: center;">0 31</td> <td style="width: 25%; text-align: left;">16</td> </tr> <tr> <td colspan="3" style="border: 1px solid black; padding: 2px;">0000011110iiiiiiL LLLLLLLLLLLLLLff011</td> </tr> </table> | 15 | 0 31 | 16 | 0000011110iiiiiiL LLLLLLLLLLLLLLff011 |  |  | オプション ( 47-32 または、63-32 ) |
| 15                                    | 0 31  | 16 |      |    |                                       |  |  |                           |
| 0000011110iiiiiiL LLLLLLLLLLLLLLff011 |   |    |      |    |                                       |  |  |                           |
- 32 ビット・イミューディエト・データ ( imm32 ) の場合、ビット 47-32 が imm32 の下位 16 ビット、ビット 63-48 が imm32 の上位 16 ビットです。

ff = 00 : sp を ep にロード

ff = 01 : 符号拡張した 16 ビット・イミューディエト・データ ( ビット 47-32 ) を ep にロード

ff = 10 : 16 ビット論理左シフトした 16 ビット・イミューディエト・データ(ビット 47-32)  
を ep にロード

ff = 11 : 32 ビット・イミューディエト・データ(ビット 63-32) を ep にロード

また, LLLLLLLLLLLL は, レジスタ・リスト「list12」の中の対応するビットの値を示します(たとえば, オペコード中のビット 21 の「L」は list12 のビット 21 の値を示します)。

list12 は, 次のように定義される 32 ビットのレジスタ・リストです。

31	30	29	28	27	26	25	24	23	22	21	20 ... 1	0
r24	r25	r26	r27	r20	r21	r22	r23	r28	r29	r31	-	r30

ビット 31-21 とビット 0 の各ビットに汎用レジスタ (r20-r31) が対応しており, セット (1)されたビットに対応するレジスタが操作の対象として指定されます。たとえば, r20, r30 を指定する場合, list12 の値は次のようになります(レジスタが対応付けられていないビット 20-1 への設定値は任意です)。

- レジスタが対応付けられていないビットの値をすべて 0 とした場合 : 08000001H
- レジスタが対応付けられていないビットの値をすべて 1 とした場合 : 081FFFFFFH

[ フ ラ グ ]      CY    -  
                  OV    -  
                  S     -  
                  Z     -  
                  SAT  -

[ 説 明 ]            (1) list12 で指定されている汎用レジスタを退避 (sp から 4 を減算し, データをそのアドレスに格納) します。次に, 2 ビット論理左シフトしワード長までゼロ拡張した 5 ビット・イミューディエトを sp から減算します。  
                  (2) list12 で指定されている汎用レジスタを退避 (sp から 4 を減算し, データをそのアドレスに格納) します。次に, 2 ビット論理左シフトしワード長までゼロ拡張した 5 ビット・イミューディエトを sp から減算します。  
                  続いて, 第 3 オペランド (sp/imm) で指定されるデータを ep にロードします。

[ 補 足 ]            list12 の汎用レジスタは, 昇順に格納されます (r20, r21, ..., r31)。  
                  imm5 は, 自動変数と一時データ用のスタック・フレームを作るために使用されます。  
                  sp で指定された下位 2 ビットのアドレスは, 0 でマスクされワード境界にアラインされます。

**注意** 命令実行中に例外が発生すると, リード・サイクルとレジスタ値の書き換えが終了したあとに, 命令の実行を中止する場合がありますが, sp は実行開始前の元の値を保持します。そのあと, 例外から復帰すると, 命令が再実行されます。

## &lt; 特殊命令 &gt;

<p>RETI</p>	<p>Return from trap or interrupt</p> <p>EI レベル・ソフトウェア例外または割り込みからの復帰</p>
-------------	---

[ 命令形式 ]        RETI

[ オペレーション ]    if PSW.EP = 1  
                           then PC ← EIPC  
                               PSW ← EIPSW  
                           else if PSW.NP = 1  
                               then PC ← FEPC  
                               PSW ← FEPSW  
                           else PC ← EIPC  
                               PSW ← EIPSW

[ フォーマット ]     Format X

[ オペコード ]        15                            0 31                            16

00000111111100000	0000000101000000
-------------------	------------------

[ フラ グ ]        CY    FEPSW または EIPSW から読み出した値が設定される  
                       OV    FEPSW または EIPSW から読み出した値が設定される  
                       S    FEPSW または EIPSW から読み出した値が設定される  
                       Z    FEPSW または EIPSW から読み出した値が設定される  
                       SAT  FEPSW または EIPSW から読み出した値が設定される

[ 説 明 ]        システム・レジスタから、復帰 PC と PSW を取り出し、EI レベル・ソフトウェア例外または割り込みから復帰する命令です。この命令の動作は次のとおりです。

<1> EP ビットが 1 の場合、NP ビットの状態にかかわらず、EIPC, EIPSW から復帰 PC , PSW を取り出します。

EP ビットが 0 かつ NP ビットが 1 の場合、FEPC , FEPSW から復帰 PC, PSW を取り出します。

EP ビットが 0 かつ NP ビットが 0 の場合、EIPC , EIPSW から復帰 PC, PSW を取り出します。

<2> 取り出した復帰 PC と PSW を PC , PSW に設定し、制御を移します。

また、EP = 0 の場合、例外ルーチンの実行を終了したことを外部（割り込みコントローラなど）に通知します。

注意 1. RETI 命令は V850E1, V850E2 CPU との後方互換のために定義しており, 原則として, RETI 命令の使用を禁止しています。修正の不可能な既存プログラム以外の RETI 命令はすべて, EIRET または FERET 命令に置き換えて使用してください。

割り込み, EI レベル・ソフトウェア例外からの復帰以外に使用された場合の動作は不定です。

2. FE レベル・ノンマスクابل割り込み例外 (FENMI), FE レベル・マスクابل割り込み例外 (FEINT), または EI レベル・ソフトウェア例外 (TRAP) からの RETI 命令による復帰時は, PC, PSW を正常にリストアするために, RETI 命令の直前で NP ビット, EP ビットを次の状態にしておく必要があります。

- RETI 命令による FE レベル・ノンマスクابل割り込み例外 (FENMI) からの復帰時: NP = 1 かつ EP = 0
- RETI 命令による FE レベル・マスクابل割り込み例外 (FEINT) からの復帰時: NP = 1 かつ EP = 0
- RETI 命令による EI レベル・ソフトウェア例外 (EITRAP0/EITRAP1) からの復帰時: EP = 1

## &lt; 特殊命令 &gt;

RIE	Reserved Instruction Exception  予約命令例外
-----	--

- [ 命令形式 ]           ( 1 ) RIE  
                          ( 2 ) RIE imm5, imm4

[ オペレーション ] FEPC   PC ( 復帰 PC )  
                      FEPSW   PSW  
                      ECR.FECC   例外要因コード  
                      FEIC   例外コード  
                      PSW.NP   1  
                      PSW.EP   1  
                      PSW.ID   1

If (MPM.AUE==1) is satisfied  
          then    PSW.IMP ← 0  
                                  PSW.DMP ← 0  
                                  PSW.NPV ← 0  
                                  PSW.PP ← 0

PC    00000030H

- [ フォーマット ]       ( 1 ) Format I  
                          ( 2 ) Format X

[ オペコード ]

( 1 ) 

15	0
00000000001000000	

( 2 ) 

15	0 31	16
iiiiii11111111IIIII 000000000000000000		

  
ただし、iiiiii は imm5 です。  
IIIII は imm4 です。

[ フラグ ]           CY   -  
                      OV   -  
                      S    -  
                      Z    -  
                      SAT -

[ 説 明 ]           復帰 PC ( RIE 命令のアドレス ) と現在の PSW の内容を、それぞれ FEPC と FEPSW に退避し、例外要因コードを FEIC レジスタと ECR.FECC ビットに格納、PSW.NP, EP, ID ビットをセット ( 1 ) します。また、MPM.AUE ビットがセット ( 1 ) されている場合、PSW.PP, NPV, DMP, IMP をクリア ( 0 ) します。  
続いて、例外ハンドラ・アドレス ( 00000030H ) に分岐し、例外処理を開始します。

## &lt; データ操作命令 &gt;

SAR	Shift arithmetic right by register/immediate (5-bit)
	算術右シフト

- [ 命令形式 ]
- ( 1 ) SAR reg1, reg2
  - ( 2 ) SAR imm5, reg2
  - ( 3 ) SAR reg1, reg2, reg3

- [ オペレーション ]
- ( 1 ) GR [reg2] GR [reg2] arithmetically shift right by GR [reg1]
  - ( 2 ) GR [reg2] GR [reg2] arithmetically shift right by zero-extend (imm5)
  - ( 3 ) GR [reg3] GR [reg2] arithmetically shift right by GR [reg1]

- [ フォーマット ]
- ( 1 ) Format IX
  - ( 2 ) Format II
  - ( 3 ) Format XI

- [ オペコード ]
- ( 1 ) 

15	0 31	16
rrrrrr	111111RRRRR	0000000010100000
  - ( 2 ) 

15	0
rrrrrr	010101iiii
  - ( 3 ) 

15	0 31	16
rrrrrr	111111RRRRR	wwwww00010100010

- [ フラグ ]
- CY 最後にシフト・アウトしたビットが 1 のとき 1, そうでないとき 0, ただしシフト数が 0 のときは 0
  - OV 0
  - S 演算結果が負のとき 1, そうでないとき 0
  - Z 演算結果が 0 のとき 1, そうでないとき 0
  - SAT -

- [ 説 明 ]
- ( 1 ) 汎用レジスタ reg2 のワード・データを汎用レジスタ reg1 の下位 5 ビットで示されるシフト数分, 0 から+31 までを算術右シフトし (シフト以前の MSB の値を, シフトを実行したあとの MSB にコピーする), 汎用レジスタ reg2 に書き込みます。シフト数が 0 のときは, 汎用レジスタ reg2 は命令実行前の値を保持します。汎用レジスタ reg1 は影響を受けません。
  - ( 2 ) 汎用レジスタ reg2 のワード・データを, ワード長までゼロ拡張した 5 ビット・イミューディエイトで示されるシフト数分, 0 から+31 までを算術右シフトし (シフト以前の MSB の値を, シフトを実行したあとの MSB にコピーする), 汎用レジスタ reg2 に書き込みます。シフト数が 0 のときは, 汎用レジスタ reg2 は命令実行前の値を保持します。



- (3) 汎用レジスタ reg2 のワード・データを汎用レジスタ reg1 の下位 5 ビットで示されるシフト数分、0 から+31 までを算術右シフトし（シフト以前の MSB の値を、シフトを実行したあとの MSB にコピーする）、汎用レジスタ reg3 に書き込みます。シフト数が 0 のときは、汎用レジスタ reg3 は命令実行前の値を保持します。汎用レジスタ reg1, reg2 は影響を受けません。

## &lt; データ操作命令 &gt;

<b>SASF</b>	Shift and set flag condition  シフトとフラグ条件の設定
-------------	--

[ 命令形式 ] SASF cccc, reg2

[ オペレーション ] if conditions are satisfied  
                   then GR [reg2] (GR [reg2] Logically shift left by 1) OR 00000001H  
                   else GR [reg2] (GR [reg2] Logically shift left by 1) OR 00000000H

[ フォーマット ] Format IX

[ オペコード ]

15	0 31	16
rrrrrr1111110cccc	0000001000000000	

[ フラグ ]  
 CY -  
 OV -  
 S -  
 Z -  
 SAT -

[ 説 明 ] 条件コード「cccc」で指定された条件が満たされた場合は、汎用レジスタ reg2 のデータを 1 ビット論理左シフトし、LSB がセット (1) されます。満たされなかった場合は、汎用レジスタ reg2 のデータを 1 ビット論理左シフトし、LSB がクリア (0) されます。  
 次の表で示されている条件コードのうちの 1 つを「cccc」として指定してください。

条件コード	条件名	条件式	条件コード	条件名	条件式
0000	V	OV = 1	0100	S/N	S = 1
1000	NV	OV = 0	1100	NS/P	S = 0
0001	C/L	CY = 1	0101	T	always (無条件)
1001	NC/NL	CY = 0	1101	SA	SAT = 1
0010	Z	Z = 1	0110	LT	(S xor OV) = 1
1010	NZ	Z = 0	1110	GE	(S xor OV) = 0
0011	NH	(CY or Z) = 1	0111	LE	((S xor OV) or Z) = 1
1011	H	(CY or Z) = 0	1111	GT	((S xor OV) or Z) = 0

[ 補 足 ] SETF 命令を参照してください。

## &lt; 飽和演算命令 &gt;

SATADD	Saturated add register/immediate (5-bit)
	飽和加算

- [ 命令形式 ]
- ( 1 ) SATADD reg1, reg2
  - ( 2 ) SATADD imm5, reg2
  - ( 3 ) SATADD reg1, reg2, reg3

- [ オペレーション ]
- ( 1 ) GR [reg2]   saturated (GR [reg2] + GR [reg1])
  - ( 2 ) GR [reg2]   saturated (GR [reg2] + sign-extend (imm5))
  - ( 3 ) GR [reg3]   saturated (GR [reg2] + GR [reg1])

- [ フォーマット ]
- ( 1 ) Format I
  - ( 2 ) Format II
  - ( 3 ) Format XI

- [ オペコード ]
- ( 1 ) 

15	0
rrrrrr000110RRRRR	

  
rrrrrr   00000 ( reg2 には r0 を設定しないでください )
  - ( 2 ) 

15	0
rrrrrr010001iiii	

  
rrrrrr   00000 ( reg2 には r0 を設定しないでください )
  - ( 3 ) 

15	0 31	16
rrrrrr111111RRRRR	wwwww01110111010	

- [ フラグ ]
- CY   MSB からのキャリーがあれば 1, そうでないとき 0
  - OV   オーバーフローが起こったとき 1, そうでないとき 0
  - S    飽和演算結果が負のとき 1, そうでないとき 0
  - Z    飽和演算結果が 0 のとき 1, そうでないとき 0
  - SAT  OV = 1 であるとき 1, そうでないとき変化しない

- [ 説 明 ]
- ( 1 ) 汎用レジスタ reg2 のワード・データに汎用レジスタ reg1 のワード・データを加算し、その結果を汎用レジスタ reg2 に格納します。ただし、結果が正の最大値 7FFFFFFFH を越えたときは 7FFFFFFFH を、負の最大値 80000000H を越えたときは 80000000H を reg2 に格納し、SAT フラグをセット ( 1 ) します。汎用レジスタ reg1 は影響を受けません。
  - ( 2 ) 汎用レジスタ reg2 のワード・データにワード長まで符号拡張した 5 ビット・イミディエイトを加算し、その結果を汎用レジスタ reg2 に格納します。ただし、結果が正の最大値 7FFFFFFFH を越えたときは 7FFFFFFFH を、負の最大値 80000000H を越えたときは 80000000H を reg2 に格納し、SAT フラグをセット ( 1 ) します。

(3) 汎用レジスタ reg2 のワード・データに汎用レジスタ reg1 のワード・データを加算し、その結果を汎用レジスタ reg3 に格納します。ただし、結果が正の最大値 7FFFFFFFH を越えたときは 7FFFFFFFH を、負の最大値 80000000H を越えたときは 80000000H を reg3 に格納し、SAT フラグをセット (1) します。汎用レジスタ reg1 と reg2 は影響を受けません。

[ 補 足 ] SAT フラグは累積フラグであり、飽和演算命令で演算結果が飽和するとセット (1) され、以降の命令の演算結果が飽和しなくてもクリア (0) されません。  
SAT フラグがセット (1) されていても、飽和演算命令は正常に実行します。

**注意 1. SAT フラグをクリア (0) するときは、LDSR 命令によって PSW にデータをロードしてください。**  
**2. 命令形式 (1) SATADD reg1, reg2 と命令形式 (2) の SATADD imm5, reg2 では、reg2 には r0 を指定しないでください。**

## &lt; 飽和演算命令 &gt;

SATSUB	Saturated subtract  飽和減算
--------	--------------------------------

- [ 命令形式 ]           (1) SATSUB reg1, reg2  
                          (2) SATSUB reg1, reg2, reg3

- [ オペレーション ]   (1) GR [reg2]   saturated (GR [reg2] – GR [reg1])  
                          (2) GR [reg3]   saturated (GR [reg2] – GR [reg1])

- [ フォーマット ]    (1) Format I  
                          (2) Format XI

- [ オペコード ]
- |                   |   |
|-------------------|---|
| 15                | 0 |
| rrrrrr000101RRRRR |   |
- rrrrrr   00000 (reg2 には r0 を設定しないでください)
- |                   |                  |    |
|-------------------|------------------|----|
| 15                | 0 31             | 16 |
| rrrrrr111111RRRRR | wwwww01110011010 |    |

- [ フラグ ]
- CY   MSB へのポローがあれば 1, そうでないとき 0  
OV   オーバーフローが起こったとき 1, そうでないとき 0  
S    飽和演算結果が負のとき 1, そうでないとき 0  
Z    飽和演算結果が 0 のとき 1, そうでないとき 0  
SAT  OV = 1 であるとき 1, そうでないとき変化しない

- [ 説 明 ]
- (1) 汎用レジスタ reg2 のワード・データから汎用レジスタ reg1 のワード・データを減算し、その結果を汎用レジスタ reg2 に格納します。ただし、結果が正の最大値 7FFFFFFFH を越えたときは 7FFFFFFFH を、負の最大値 80000000H を越えたときは 80000000H を reg2 に格納し、SAT フラグをセット (1) します。汎用レジスタ reg1 は影響を受けません。
- (2) 汎用レジスタ reg2 のワード・データから汎用レジスタ reg1 のワード・データを減算し、その結果を汎用レジスタ reg3 に格納します。ただし、結果が正の最大値 7FFFFFFFH を越えたときは 7FFFFFFFH を、負の最大値 80000000H を越えたときは 80000000H を reg3 に格納し、SAT フラグをセット (1) します。汎用レジスタ reg1 と reg2 は影響を受けません。

- [ 補 足 ]           SAT フラグは累積フラグであり、飽和演算命令で演算結果が飽和するとセット (1) され、以降の命令の演算結果が飽和しなくてもクリア (0) されません。  
                      SAT フラグがセット (1) されていても、飽和演算命令は正常に実行します。

**注意 1. SAT フラグをクリア (0) するときは、LDSR 命令によって PSW にデータをロードしてください。**  
**2. 命令形式 (1) の SATSUB reg1, reg2 では、reg2 には r0 を指定しないでください。**

## &lt; 飽和演算命令 &gt;

SATSUBI	Saturated subtract immediate  飽和減算
---------	--

[ 命令形式 ]        SATSUBI imm16, reg1, reg2

[ オペレーション ]   GR [reg2]    saturated (GR [reg1] – sign-extend (imm16) )

[ フォーマット ]    Format VI

[ オペコード ]

15	0 31	16
rrrrrr110011RRRRR	iiiiiiiiiiiiiiiiiii	

rrrrrr    00000 ( reg2 には r0 を設定しないでください )

[ フラグ ]

CY    MSB へのボローがあれば 1, そうでないとき 0

OV    オーバフローが起こったとき 1, そうでないとき 0

S     飽和演算結果が負のとき 1, そうでないとき 0

Z     飽和演算結果が 0 のとき 1, そうでないとき 0

SAT   OV = 1 であるとき 1, そうでないとき変化しない

[ 説 明 ]        汎用レジスタ reg1 のワード・データからワード長まで符号拡張した 16 ビット・イミディエトを減算し、その結果を汎用レジスタ reg2 に格納します。ただし、結果が正の最大値 7FFFFFFFH を越えたときは 7FFFFFFFH を、負の最大値 80000000H を越えたときは 80000000H を reg2 に格納し、SAT フラグをセット (1) します。汎用レジスタ reg1 は影響を受けません。

[ 補 足 ]        SAT フラグは累積フラグであり、飽和演算命令で演算結果が飽和するとセット (1) され、以降の命令の演算結果が飽和しなくてもクリア (0) されません。  
SAT フラグがセット (1) されていても、飽和演算命令は正常に実行します。

**注意 1. SAT フラグをクリア (0) するときは、LDSR 命令によって PSW にデータをロードしてください。**  
**2. reg2 には、r0 を指定しないでください。**

## &lt; 飽和演算命令 &gt;

SATSUBR	Saturated subtract reverse  飽和逆減算
---------	---

[ 命令形式 ]            SATSUBR  reg1, reg2

[ オペレーション ]    GR [reg2]    saturated (GR [reg1] – GR [reg2])

[ フォーマット ]      Format I

[ オペコード ]        15                          0  
                         rrrrrr000100RRRRR  
rrrrrr    00000 ( reg2 には r0 を設定しないでください )

[ フ ラ グ ]            CY    MSB へのポローがあれば 1 , そうでないとき 0  
                         OV    オーバフローが起こったとき 1 , そうでないとき 0  
                         S     飽和演算結果が負のとき 1 , そうでないとき 0  
                         Z     飽和演算結果が 0 のとき 1 , そうでないとき 0  
                         SAT   OV = 1 であるとき 1 , そうでないとき変化しない

[ 説 明 ]                汎用レジスタ reg1 のワード・データから汎用レジスタ reg2 のワード・データを減算し、その結果を汎用レジスタ reg2 に格納します。ただし、結果が正の最大値 7FFFFFFFH を越えたときは 7FFFFFFFH を、負の最大値 80000000H を越えたときは 80000000H を reg2 に格納し、SAT フラグをセット ( 1 ) します。汎用レジスタ reg1 は影響を受けません。

[ 補 足 ]                SAT フラグは累積フラグであり、飽和演算命令で演算結果が飽和するとセット ( 1 ) され、以降の命令の演算結果が飽和しなくてもクリア ( 0 ) されません。  
                         SAT フラグがセット ( 1 ) されていても、飽和演算命令は正常に実行します。

- 注意 1.** SAT フラグをクリア ( 0 ) するときは、LDSR 命令によって PSW にデータをロードしてください。  
**2.** reg2 には、r0 を指定しないでください。

## &lt; 条件付き演算命令 &gt;

<b>SBF</b>	Subtract on condition flag  条件付き減算
------------	--

[ 命令形式 ]      SBF   cccc, reg1, reg2, reg3

[ オペレーション ]    if conditions are satisfied  
                           then GR [reg3] ← GR [reg2] – GR [reg1] –1  
                           else GR [reg3] ← GR [reg2] – GR [reg1] –0

[ フォーマット ]    Format XI

[ オペコード ]      15                                    0 31                                    16

rrrrrr111111RRRRR	wwwww011100cccc0
-------------------	------------------

[ フ ラ グ ]        CY      MSB へのボローがあれば 1, そうでないとき 0  
                       OV      オーバフローが起こったとき 1, そうでないとき 0  
                       S      演算結果が負のとき 1, そうでないとき 0  
                       Z      演算結果が 0 のとき 1, そうでないとき 0  
                       SAT     –

[ 説 明 ]        条件コード「cccc」で指定された条件が満たされた場合は、汎用レジスタ reg2 のワード・データから汎用レジスタ reg1 のワード・データを減算した結果から、1 を減算し、その結果を汎用レジスタ reg3 に格納します。  
                       条件コード「cccc」で指定された条件が満たされなかった場合は、汎用レジスタ reg2 のワード・データから汎用レジスタ reg1 のワード・データを減算し、その結果を汎用レジスタ reg3 に格納します。  
                       汎用レジスタ reg1, reg2 は影響を受けません。  
                       次の表で示されている条件コードのうちの 1 つを「cccc」として指定してください(ただし, cccc ≠ 1101)。

条件コード	条件名	条件式	条件コード	条件名	条件式
0000	V	OV = 1	0100	S/N	S = 1
1000	NV	OV = 0	1100	NS/P	S = 0
0001	C/L	CY = 1	0101	T	always (無条件)
1001	NC/NL	CY = 0	0110	LT	(S xor OV) = 1
0010	Z	Z = 1	1110	GE	(S xor OV) = 0
1010	NZ	Z = 0	0111	LE	((S xor OV) or Z) = 1
0011	NH	(CY or Z) = 1	1111	GT	((S xor OV) or Z) = 0
1011	H	(CY or Z) = 0	(1101)	設定禁止	



## &lt;ビット・サーチ命令&gt;

SCH0L	Search zero from left
	MSB 側からのビット (0) 検索

[ 命令形式 ]        SCH0L reg2, reg3

[ オペレーション ] GR [reg3]    search zero from left of GR [reg2]

[ フォーマット ]    Format IX

[ オペコード ]

15	0 31	16
rrrrrr11111100000	wwwww01101100100	

[ フラグ ]        CY    最後にビット (0) が見つかったとき 1, そうでないとき 0

OV    0

S     0

Z     ビット (0) が見つからなかったとき 1, そうでないとき 0

SAT   -

[ 説 明 ]        汎用レジスタ reg2 のワード・データを左側 (MSB 側) から検索し, 最初に 0 が見つかったビット位置 (0~31) までの 1 の個数 + 1 を汎用レジスタ reg3 に書き込みます (たとえば, reg2 のビット 31 が 0 の場合は, reg3 に 01H を書き込みます)。  
ビット (0) が見つからなかった場合は, reg3 に 0 を書き込み, 同時に Z フラグをセット (1) します。最後にビット (0) が見つかった場合は CY フラグをセット (1) します。

## &lt;ビット・サーチ命令&gt;

SCH0R	Search zero from right  LSB 側からのビット (0) 検索
-------	--

[ 命令形式 ]            SCH0R    reg2, reg3

[ オペレーション ]    GR [reg3]    search zero from right of GR [reg2]

[ フォーマット ]       Format IX

[ オペコード ]        15                            0 31                            16  
                          rrrrrr111111000000 | wwwwww011011000000

[ フラグ ]            CY    最後にビット (0) が見つかったとき 1, そうでないとき 0  
                          OV    0  
                          S    0  
                          Z    ビット (0) が見つからなかったとき 1, そうでないとき 0  
                          SAT   -

[ 説 明 ]            汎用レジスタ reg2 のワード・データを右側 (LSB 側) から検索し, 最初に 0 が見つかったビット位置 (0~31) までの 1 の個数 + 1 を汎用レジスタ reg3 に書き込みます (たとえば, reg2 のビット 0 が 0 の場合は, reg3 に 01H を書き込みます)。  
                          ビット (0) が見つからなかった場合は, reg3 に 0 を書き込み, 同時に Z フラグをセット (1) します。最後にビット (0) が見つかった場合は CY フラグをセット (1) します。

## &lt;ビット・サーチ命令&gt;

SCH1L	Search one from left  MSB 側からのビット (1) 検索
-------	--

[ 命令形式 ]          SCH1L reg2, reg3

[ オペレーション ] GR [reg3]    search one from left of GR [reg2]

[ フォーマット ]      Format IX

[ オペコード ]          15                                  0 31                                  16

rrrrrr11111100000	wwwww01101100110
-------------------	------------------

[ フ ラ グ ]          CY    最後にビット (1) が見つかったとき 1, そうでないとき 0

OV    0

S     0

Z     ビット (1) が見つからなかったとき 1, そうでないとき 0

SAT   -

[ 説 明 ]                  汎用レジスタ reg2 のワード・データを左側 (MSB 側) から検索し、最初に 1 が見つかったビット位置 (0~31) までの 0 の個数 + 1 を汎用レジスタ reg3 に書き込みます (たとえば, reg2 のビット 31 が 1 の場合は, reg3 に 01H を書き込みます)。  
ビット (1) が見つからなかった場合は, reg3 に 0 を書き込み, 同時に Z フラグをセット (1) します。最後にビット (1) が見つかった場合は CY フラグをセット (1) します。

## &lt; ビット・サーチ命令 &gt;

SCH1R	Search one from right  LSB 側からのビット (1) 検索
-------	---

[ 命令形式 ]            SCH1R   reg2, reg3

[ オペレーション ]   GR [reg3]    search one from right of GR [reg2]

[ フォーマット ]      Format IX

[ オペコード ]        15                            0 31                            16

rrrrrr11111100000	wwwww01101100010
-------------------	------------------

[ フ ラ グ ]            CY    最後にビット (1) が見つかったとき 1, そうでないとき 0

OV    0

S     0

Z     ビット (1) が見つからなかったとき 1, そうでないとき 0

SAT   -

[ 説 明 ]            汎用レジスタ reg2 のワード・データを右側 (LSB 側) から検索し, 最初に 1 が見つかったビット位置 (0~31) までの 0 の個数 + 1 を汎用レジスタ reg3 に書き込みます (たとえば, reg2 のビット 0 が 1 の場合は, reg3 に 01H を書き込みます)。  
ビット (1) が見つからなかった場合は, reg3 に 0 を書き込み, 同時に Z フラグをセット (1) します。最後にビット (1) が見つかった場合は CY フラグをセット (1) します。

## &lt; ビット操作命令 &gt;

SET1	Set bit  ビット・セット
------	------------------------

- [ 命令形式 ]           ( 1 ) SET1 bit#3, disp16 [reg1]  
                          ( 2 ) SET1 reg2, [reg1]

- [ オペレーション ]   ( 1 ) adr   GR [reg1] + sign-extend (disp16)  
                          token   Load-memory (adr, Byte)  
                          Z フラグ   Not (extract-bit (token, bit#3) )  
                          token   set-bit (token, bit#3)  
                          Store-memory (adr, token, Byte)  
                          ( 2 ) adr   GR [reg1]  
                          token   Load-memory (adr, Byte)  
                          Z フラグ   Not (extract-bit (token, reg2) )  
                          token   set-bit (token, reg2)  
                          Store-memory (adr, token, Byte)

- [ フォーマット ]   ( 1 ) Format VIII  
                          ( 2 ) Format IX

- [ オペコード ]
- |       |                   |                |
|-------|-------------------|----------------|
| 15    | 0 31              | 16             |
| ( 1 ) | 00bbb1111110RRRRR | dddddddddddddd |
- |       |                   |                  |
|-------|-------------------|------------------|
| 15    | 0 31              | 16               |
| ( 2 ) | rrrrrr111111RRRRR | 0000000011100000 |

- [ フラグ ]           CY   -  
                          OV   -  
                          S    -  
                          Z    指定したビットが 0 のとき 1, 指定したビットが 1 のとき 0  
                          SAT  -

- [ 説 明 ]           ( 1 ) まず、汎用レジスタ reg1 のワード・データと、ワード長まで符号拡張した 16 ビット・ディスプレイメントを加算して 32 ビット・アドレスを生成します。生成したアドレスのバイト・データを読み出し、3 ビットのビット・ナンバで指定されるビットをセット (1) し、元のアドレスに書き戻します。  
                          読み出したバイト・データの指定ビットが 0 のとき Z フラグをセット (1) し、指定ビットが 1 のとき Z フラグをクリア (0) します。  
                          ( 2 ) まず、汎用レジスタ reg1 のワード・データを読み出して 32 ビット・アドレスを生成します。生成したアドレスのバイト・データを読み出し、汎用レジスタ reg2 の下位 3 ビットで指定されるビットをセット (1) し、元のアドレスに書き戻します。  
                          読み出したバイト・データの指定ビットが 0 のとき Z フラグをセット (1) し、指定ビットが 1 のとき Z フラグをクリア (0) します。

[ 補 足 ] PSW の Z フラグはこの命令を実行する前に該当ビットが 0 か 1 だったかを示します。この命令実行後の該当ビットの内容を示すものではありません。

**注意** この命令は排他制御を目的としたアトミック性保証のため、読み出しから書き込みまでの間、対象のアドレスが他の要因によるアクセスによって操作されることはありません。

< データ操作命令 >

<b>SETF</b>	Set flag condition  フラグ条件の設定
-------------	------------------------------------

[ 命令形式 ]         SETF  cccc, reg2

[ オペレーション ]  if conditions are satisfied  
                         then GR [reg2]    00000001H  
                         else GR [reg2]    00000000H

[ フォーマット ]     Format IX

[ オペコード ]       15                            0 31                            16

rrrrrr	1111110cccc	000000000000000000
--------	-------------	--------------------

[ フラグ ]           CY    -  
                         OV    -  
                         S     -  
                         Z     -  
                         SAT  -

[ 説 明 ]            条件コード「cccc」の示す条件が満たされた場合、汎用レジスタ reg2 に 1 を、そうでない場合は 0 を格納します。  
                         次の表で示されている条件コードのうちの 1 つを「cccc」として指定してください。

条件コード	条件名	条件式	条件コード	条件名	条件式
0000	V	OV = 1	0100	S/N	S = 1
1000	NV	OV = 0	1100	NS/P	S = 0
0001	C/L	CY = 1	0101	T	always (無条件)
1001	NC/NL	CY = 0	1101	SA	SAT = 1
0010	Z	Z = 1	0110	LT	(S xor OV) = 1
1010	NZ	Z = 0	1110	GE	(S xor OV) = 0
0011	NH	(CY or Z) = 1	0111	LE	((S xor OV) or Z) = 1
1011	H	(CY or Z) = 0	1111	GT	((S xor OV) or Z) = 0

[ 補 足 ] この命令の利用方法の例を示します。

(1) 複数の条件節の翻訳

C 言語での if ( A ) という文において、A が複数の条件節 ( a1, a2, a3, ... ) から成り立つとき、通常は if ( a1 ) then, if ( a2 ) then というシーケンスに翻訳します。オブジェクト・コードでは an に相当する評価の結果を見て「条件分岐」をします。パイプライン・プロセッサでは「条件判断 + 分岐」は通常の演算に比べて遅いので、おのおのの条件節を評価した結果 if ( an ) の結果をレジスタ Ra に覚えておきます。すべての条件節を評価し終わったあとに Ran をまとめて論理演算することで、パイプラインによる遅れを回避できます。

(2) 倍長演算

Add with Carry のような倍長演算をするときに、CY フラグの結果を汎用レジスタ reg2 に格納できるため、下位からの桁上りを数値として表現できます。



## &lt; データ操作命令 &gt;

SHL	Shift logical left by register/immediate (5-bit)
	論理左シフト

- [ 命令形式 ]
- ( 1 ) SHL reg1, reg2
  - ( 2 ) SHL imm5, reg2
  - ( 3 ) SHL reg1, reg2, reg3

- [ オペレーション ]
- ( 1 ) GR [reg2] GR [reg2] logically shift left by GR [reg1]
  - ( 2 ) GR [reg2] GR [reg2] logically shift left by zero-extend (imm5)
  - ( 3 ) GR [reg3] GR [reg2] logically shift left by GR [reg1]

- [ フォーマット ]
- ( 1 ) Format IX
  - ( 2 ) Format II
  - ( 3 ) Format XI

- [ オペコード ]
- ( 1 )

15	0 31	16
rrrrrr111111RRRRR 0000000011000000		
  - ( 2 )

15	0
rrrrrr010110iiii	
  - ( 3 )

15	0 31	16
rrrrrr111111RRRRR wwwww00011000010		

- [ フラグ ]
- CY 最後にシフト・アウトしたビットが 1 のとき 1, そうでないとき 0, ただしシフト数が 0 のときは 0
  - OV 0
  - S 演算結果が負のとき 1, そうでないとき 0
  - Z 演算結果が 0 のとき 1, そうでないとき 0
  - SAT -

- [ 説 明 ]
- ( 1 ) 汎用レジスタ reg2 のワード・データを汎用レジスタ reg1 の下位 5 ビットで示されるシフト数分, 0 から+31 までを論理左シフトし (LSB 側に 0 を送り込む), 汎用レジスタ reg2 に書き込みます。シフト数が 0 のときは, 汎用レジスタ reg2 は命令実行前の値を保持します。汎用レジスタ reg1 は影響を受けません。
  - ( 2 ) 汎用レジスタ reg2 のワード・データを, ワード長までゼロ拡張した 5 ビット・イミューディオで示されるシフト数分, 0 から+31 までを論理左シフトし (LSB 側に 0 を送り込む), 汎用レジスタ reg2 に書き込みます。シフト数が 0 のときは, 汎用レジスタ reg2 は命令実行前の値を保持します。

- (3) 汎用レジスタ reg2 のワード・データを汎用レジスタ reg1 の下位 5 ビットで示されるシフト数分, 0 から+31 までを論理左シフトし (LSB 側に 0 を送り込む), 汎用レジスタ reg3 に書き込みます。シフト数が 0 のときは, 汎用レジスタ reg3 は命令実行前の値を保持します。汎用レジスタ reg1, reg2 は影響を受けません。

## &lt; データ操作命令 &gt;

SHR	Shift logical right by register/immediate (5-bit)  論理右シフト
-----	---

- [ 命令形式 ]
- ( 1 ) SHR reg1, reg2
  - ( 2 ) SHR imm5, reg2
  - ( 3 ) SHR reg1, reg2, reg3

- [ オペレーション ]
- ( 1 ) GR [reg2] GR [reg2] logically shift right by GR [reg1]
  - ( 2 ) GR [reg2] GR [reg2] logically shift right by zero-extend ( imm5 )
  - ( 3 ) GR [reg3] GR [reg2] logically shift right by GR [reg1]

- [ フォーマット ]
- ( 1 ) Format IX
  - ( 2 ) Format II
  - ( 3 ) Format XI

- [ オペコード ]
- ( 1 ) 

15	0 31	16
rrrrrr	111111RRRRR	0000000010000000
  - ( 2 ) 

15	0
rrrrrr	010100iiii
  - ( 3 ) 

15	0 31	16
rrrrrr	111111RRRRR	wwwwww00010000010

- [ フ ラ グ ]
- CY 最後にシフト・アウトしたビットが 1 のとき 1 , そうでないとき 0 ,  
ただしシフト数が 0 のときは 0
  - OV 0
  - S 演算結果が負のとき 1 , そうでないとき 0
  - Z 演算結果が 0 のとき 1 , そうでないとき 0
  - SAT -

- [ 説 明 ]
- ( 1 ) 汎用レジスタ reg2 のワード・データを汎用レジスタ reg1 の下位 5 ビットで示されるシフト数分, 0 から+31 までを論理右シフトし (MSB 側に 0 を送り込む) , 汎用レジスタ reg2 に書き込みます。シフト数が 0 のときは, 汎用レジスタ reg2 は命令実行前と同じ値を保持します。汎用レジスタ reg1 は影響を受けません。
  - ( 2 ) 汎用レジスタ reg2 のワード・データを, ワード長までゼロ拡張した 5 ビット・イミューディオで示されるシフト数分, 0 から+31 までを論理右シフトし (MSB 側に 0 を送り込む) , 汎用レジスタ reg2 に書き込みます。シフト数が 0 のときは, 汎用レジスタ reg2 は命令実行前の値を保持します。

- (3) 汎用レジスタ reg2 のワード・データを汎用レジスタ reg1 の下位 5 ビットで示されるシフト数分, 0 から+31 までを論理右シフトし (MSB 側に 0 を送り込む), 汎用レジスタ reg3 に書き込みます。シフト数が 0 のときは, 汎用レジスタ reg3 は命令実行前の値を保持します。汎用レジスタ reg1, reg2 は影響を受けません。



## &lt; ロード命令 &gt;

SLD.BU

Short format load byte unsigned

(符号なし) バイト・データのロード

[ 命令形式 ] SLD.BU disp4 [ep], reg2

[ オペレーション ] adr ep + zero-extend (disp4)  
GR [reg2] zero-extend (Load-memory (adr, Byte) )

[ フォーマット ] Format IV

[ オペコード ] 

15	0
rrrrrr	0000110dddd

  
rrrrrr 00000 ( reg2 には r0 を設定しないでください )[ フラグ ] CY -  
OV -  
S -  
Z -  
SAT -

[ 説 明 ] エレメント・ポインタと、ワード長までゼロ拡張した 4 ビット・ディスプレースメントを加算して 32 ビット・アドレスを生成します。生成したアドレスからバイト・データを読み出し、ワード長までゼロ拡張し、reg2 に格納します。

**注意** reg2 には、r0 を指定しないでください。

&lt;ロード命令&gt;

SLD.H	Short format load half-word  (符号付き) ハーフワード・データのロード
-------	--

[命令形式] SLD.H disp8 [ep], reg2

[オペレーション] adr ep + zero-extend (disp8)  
GR [reg2] sign-extend (Load-memory (adr, Half-word))

[フォーマット] Format IV

[オペコード]

15	0
rrrrrr1000ddddddd	

ただし、ddddddd は disp8 の上位7ビットです。

[フラグ]

CY -  
OV -  
S -  
Z -  
SAT -

[説明] エレメント・ポインタと、ワード長までゼロ拡張した8ビット・ディスプレイacementsを加算して32ビット・アドレスを生成します。生成した32ビット・アドレスからハーフワード・データを読み出し、ワード長まで符号拡張し、reg2に格納します。

&lt; ロード命令 &gt;

SLD.HU

Short format load half-word unsigned

(符号なし) ハーフワード・データのロード

[ 命令形式 ] SLD.HU disp5 [ep], reg2

[ オペレーション ] adr ep + zero-extend (disp5)  
GR [reg2] zero-extend (Load-memory (adr, Half-word))

[ フォーマット ] Format IV

[ オペコード ]  
15 0  
rrrrrr0000111dddd  
rrrrrr 00000 (reg2にはr0を設定しないでください)  
ただし, dddd は disp5 の上位 4 ビット[ フラグ ] CY -  
OV -  
S -  
Z -  
SAT -

[ 説 明 ] エレメント・ポインタと, ワード長までゼロ拡張した 5 ビット・ディスプレースメントを加算して 32 ビット・アドレスを生成します。生成した 32 ビット・アドレスからハーフワード・データを読み出し, ワード長までゼロ拡張し, reg2 に格納します。

**注意** reg2 には r0 を指定しないでください。



## &lt; ロード命令 &gt;

SLD.W	Short format load word  ワード・データのロード
-------	---

[ 命令形式 ]            SLD.W    disp8 [ep], reg2

[ オペレーション ]    adr     ep + zero-extend (disp8)  
                          GR [reg2]    Load-memory (adr, Word)

[ フォーマット ]        Format IV

[ オペコード ]         15                                 0  
                          rrrrrr1010dddddd0  
                          ただし、dddddd は disp8 の上位 6 ビットです。

[ フ ラ グ ]            CY    -  
                          OV    -  
                          S     -  
                          Z     -  
                          SAT  -

[ 説 明 ]                エlement・ポインタと、ワード長までゼロ拡張した 8 ビット・ディスプレイacementsを加算して 32 ビット・アドレスを生成します。生成した 32 ビット・アドレスからワード・データを読み出し、reg2 に格納します。

## &lt; ストア命令 &gt;

SST.B	Short format store byte
	バイト・データのストア

[ 命令形式 ] SST.B reg2, disp7 [ep]

[ オペレーション ] adr ep + zero-extend (disp7)  
Store-memory (adr, GR [reg2] , Byte)

[ フォーマット ] Format IV

[ オペコード ]       15                       0  
                  rrrrr01111ddddddd

[ フ ラ グ ]       CY   -  
                  OV   -  
                  S    -  
                  Z    -  
                  SAT -

[ 説 明 ]        エレメント・ポインタと、ワード長までゼロ拡張した7ビット・ディスプレースメントを加算して32ビット・アドレスを生成します。reg2の最下位バイト・データを生成したアドレスに格納します。

## &lt;ストア命令&gt;

SST.H	Short format store half-word  ハーフワード・データのストア
-------	--

[ 命令形式 ]         SST.H reg2, disp8 [ep]

[ オペレーション ]   adr     ep + zero-extend (disp8)  
                          Store-memory (adr, GR [reg2] , Half-word)

[ フォーマット ]     Format IV

[ オペコード ]       15                                 0  
                   rrrrrr1001ddddddd

ただし、dddddd は disp8 の上位 7 ビットです。

[ フラグ ]           CY     -  
                      OV     -  
                      S     -  
                      Z     -  
                      SAT  -

[ 説 明 ]            エレメント・ポインタと、ワード長までゼロ拡張した 8 ビット・ディスプレイースメントを加算して 32 ビット・アドレスを生成します。reg2 の下位ハーフワード・データを生成した 32 ビット・アドレスに格納します。



## &lt;ストア命令&gt;

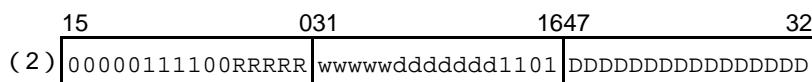
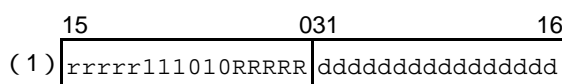
ST.B	Store byte  バイト・データのストア
------	-------------------------------

- [ 命令形式 ]           ( 1 ) ST.B reg2, disp16 [reg1]  
                          ( 2 ) ST.B reg3, disp23 [reg1]

- [ オペレーション ]   ( 1 ) adr   GR [reg1] + sign-extend (disp16)  
                          Store-memory (adr, GR [reg2], Byte)  
                          ( 2 ) adr   GR [reg1] + sign-extend (disp23)  
                          Store-memory (adr, GR [reg3], Byte)

- [ フォーマット ]      ( 1 ) Format VII  
                          ( 2 ) Format XIV

## [ オペコード ]



ただし、RRRRR = reg1, wwwww = reg3 です。

ddddddd は、disp23 の下位 7 ビットです。

DDDDDDDDDDDDDDDD は、disp23 の上位 16 ビットです。

- [ フラグ ]           CY   -  
                      OV   -  
                      S    -  
                      Z    -  
                      SAT -

- [ 説 明 ]           ( 1 ) 汎用レジスタ reg1 のデータと、ワード長まで符号拡張した 16 ビット・ディスプレイメントを加算して 32 ビット・アドレスを生成します。汎用レジスタ reg2 の最下位のバイト・データを生成したアドレスに格納します。  
                      ( 2 ) 汎用レジスタ reg1 のデータと、ワード長まで符号拡張した 23 ビット・ディスプレイメントを加算して 32 ビット・アドレスを生成します。汎用レジスタ reg3 の最下位のバイト・データを生成したアドレスに格納します。

## &lt;ストア命令&gt;

ST.H	Store half-word  ハーフワード・データのストア
------	---------------------------------------

- [ 命令形式 ]           ( 1 ) ST.H reg2, disp16 [reg1]  
                          ( 2 ) ST.H reg3, disp23 [reg1]

- [ オペレーション ]   ( 1 ) adr   GR [reg1] + sign-extend (disp16)  
                          Store-memory (adr, GR [reg2], Half-word)  
                          ( 2 ) adr   GR [reg1] + sign-extend (disp23)  
                          Store-memory (adr, GR [reg3], Half-word)

- [ フォーマット ]      ( 1 ) Format VII  
                          ( 2 ) Format XIV

- [ オペコード ]
- |  |  |                  |     |      |                    |                  |                  |                  |
|--|--|------------------|-----|------|--------------------|------------------|------------------|------------------|
| ( 1 )                                    | <table style="border-collapse: collapse; width: 100%;"> <tr> <td style="text-align: right; padding-right: 5px;">15</td> <td style="text-align: center; padding: 0 10px;">031</td> <td style="text-align: left; padding-left: 5px;">16</td> </tr> <tr> <td style="border-right: 1px solid black; padding: 2px;">rrrrrr1111011RRRRR</td> <td style="padding: 2px;">ddddddddddddddd0</td> </tr> </table>  | 15               | 031 | 16   | rrrrrr1111011RRRRR | ddddddddddddddd0 |                  |                  |
| 15                                       | 031  | 16               |     |      |                    |                  |                  |                  |
| rrrrrr1111011RRRRR                       | ddddddddddddddd0   |                  |     |      |                    |                  |                  |                  |
| ただし, dddddddddddddddはdisp16の上位15ビットです。   |  |                  |     |      |                    |                  |                  |                  |
| ( 2 )                                    | <table style="border-collapse: collapse; width: 100%;"> <tr> <td style="text-align: right; padding-right: 5px;">15</td> <td style="text-align: center; padding: 0 10px;">031</td> <td style="text-align: left; padding-left: 5px;">1647</td> <td style="text-align: right; padding-right: 5px;">32</td> </tr> <tr> <td style="border-right: 1px solid black; padding: 2px;">00000111101RRRRR</td> <td style="padding: 2px;">wwwwwdddddd01101</td> <td style="padding: 2px;">DDDDDDDDDDDDDDDD</td> </tr> </table> | 15               | 031 | 1647 | 32                 | 00000111101RRRRR | wwwwwdddddd01101 | DDDDDDDDDDDDDDDD |
| 15                                       | 031  | 1647             | 32  |      |                    |                  |                  |                  |
| 00000111101RRRRR                         | wwwwwdddddd01101   | DDDDDDDDDDDDDDDD |     |      |                    |                  |                  |                  |
| ただし, RRRRR = reg1, wwwww = reg3 です。      |  |                  |     |      |                    |                  |                  |                  |
| dddddd は, disp23 の下位側ビット 6-1 です。         |  |                  |     |      |                    |                  |                  |                  |
| DDDDDDDDDDDDDDDD は, disp23 の上位 16 ビットです。 |  |                  |     |      |                    |                  |                  |                  |

- [ フ ラ グ ]           CY   -  
                          OV   -  
                          S    -  
                          Z    -  
                          SAT -

- [ 説 明 ]           ( 1 ) 汎用レジスタ reg1 のデータと, ワード長まで符号拡張した 16 ビット・ディスプレイメントを加算して 32 ビット・アドレスを生成します。汎用レジスタ reg2 の下位ハーフワード・データを生成したアドレスに格納します。  
                          ( 2 ) 汎用レジスタ reg1 のデータと, ワード長まで符号拡張した 23 ビット・ディスプレイメントを加算して 32 ビット・アドレスを生成します。汎用レジスタ reg3 の最下位のハーフワード・データを生成したアドレスに格納します。

## &lt;ストア命令&gt;

ST.W	Store word  ワード・データのストア
------	-------------------------------

- [ 命令形式 ]           ( 1 ) ST.W reg2, disp16 [reg1]  
                          ( 2 ) ST.W reg3, disp23 [reg1]

- [ オペレーション ]   ( 1 ) adr   GR [reg1] + sign-extend (disp16)  
                          Store-memory (adr, GR [reg2], Word)  
                          ( 2 ) adr   GR [reg1] + sign-extend (disp23)  
                          Store-memory (adr, GR [reg3], Word)

- [ フォーマット ]      ( 1 ) Format VII  
                          ( 2 ) Format XIV

## [ オペコード ]

- ( 1 ) 

15	031	16
rrrrrr111011RRRRR	ddddddddddddddd1	

  
ただし、dddddddddddddddはdisp16の上位15ビットです。
- ( 2 ) 

15	031	1647	32
00000111100RRRRR	wwwwwdddddd01111	DDDDDDDDDDDDDDDD	

  
ただし、RRRRR = reg1, wwwww = reg3 です。  
dddddd は、disp23 の下位側ビット 6-1 です。  
DDDDDDDDDDDDDDDD は、disp23 の上位 16 ビットです。

- [ フ ラ グ ]          CY   -  
                          OV   -  
                          S    -  
                          Z    -  
                          SAT -

- [ 説 明 ]           ( 1 ) 汎用レジスタ reg1 のデータと、ワード長まで符号拡張した 16 ビット・ディスプレイメントを加算して 32 ビット・アドレスを生成します。汎用レジスタ reg2 のワード・データを生成したアドレスに格納します。  
                          ( 2 ) 汎用レジスタ reg1 のデータと、ワード長まで符号拡張した 23 ビット・ディスプレイメントを加算して 32 ビット・アドレスを生成します。汎用レジスタ reg3 の最下位のワード・データを生成したアドレスに格納します。

&lt; 特殊命令 &gt;

STSR	Store contents of system register
	システム・レジスタの内容のストア

[ 命令形式 ]          STSR  regID, reg2

[ オペレーション ]  GR [reg2]    SR [regID]

[ フォーマット ]    Format IX

[ オペコード ]	15	0 31	16
	rrrrrr111111RRRRR	0000000001000000	

[ フラグ ]	CY	-
	OV	-
	S	-
	Z	-
	SAT	-

[ 説 明 ]          システム・レジスタ番号( regID )で指定されるシステム・レジスタの内容を汎用レジスタ reg2 に設定します。システム・レジスタは影響を受け付けません。

**注意**    システム・レジスタ番号は、システム・レジスタを一意に識別するための番号です。予約されているシステム・レジスタ番号を指定した場合の動作は保証しません。



## &lt; 算術演算命令 &gt;

SUB	Subtract  減算
-----	--------------------

[ 命令形式 ]       SUB reg1, reg2

[ オペレーション ] GR [reg2]   GR [reg2] – GR [reg1]

[ フォーマット ]   Format I

[ オペコード ]      15                    0  
rrrrr001101RRRRR

[ フラ グ ]        CY   MSB へのポローがあれば 1 , そうでないとき 0  
 OV   オーバーフローが起こったとき 1 , そうでないとき 0  
 S     演算結果が負のとき 1 , そうでないとき 0  
 Z     演算結果が 0 のとき 1 , そうでないとき 0  
 SAT   -

[ 説 明 ]        汎用レジスタ reg2 のワード・データから汎用レジスタ reg1 のワード・データを減算し、その結果を汎用レジスタ reg2 に格納します。汎用レジスタ reg1 は影響を受けません。

## &lt; 算術演算命令 &gt;

<h1 style="margin: 0;">SUBR</h1>	<div style="font-size: small;">Subtract reverse</div> <div style="font-size: x-small; margin-top: 10px;">逆減算</div>
----------------------------------	--

[ 命令形式 ]      SUBR  reg1, reg2

[ オペレーション ]   GR [reg2]   GR [reg1] – GR [reg2]

[ フォーマット ]     Format I

[ オペコード ]      15                                  0  
                         rrrrr001100RRRR

[ フラグ ]            CY   MSB へのボローがあれば 1，そうでないとき 0  
                         OV   オーバフローが起こったとき 1，そうでないとき 0  
                         S    演算結果が負のとき 1，そうでないとき 0  
                         Z    演算結果が 0 のとき 1，そうでないとき 0  
                         SAT -

[ 説 明 ]             汎用レジスタ reg1 のワード・データから汎用レジスタ reg2 のワード・データを減算し，その結果を汎用レジスタ reg2 に格納します。汎用レジスタ reg1 は影響を受けません。

&lt;特殊命令&gt;

SWITCH	Jump with table look up  テーブル参照分岐
--------	---

[命令形式] SWITCH reg1

[オペレーション] adr (PC + 2) + (GR [reg1] logically shift left by 1)  
PC (PC + 2) + (sign-extend (Load-memory (adr, Half-word) )) logically shift left by 1

[フォーマット] Format I

[オペコード]      15                          0  
                          00000000010RRRRR  
RRRRR      00000 (reg1 には r0 を設定しないでください)

[フラグ]      CY     -  
              OV     -  
              S      -  
              Z      -  
              SAT    -

[説明]        次の順に処理を行います。

- <1> テーブルの先頭アドレス ( SWITCH 命令の次のアドレス ) と 1 ビット論理左シフトした汎用レジスタ reg1 のデータを加算し、32 ビット・テーブル・エン트리・アドレスを生成します。
- <2> <1>で生成されたアドレスが指し示すハーフワード・エン 트리・データをロードします。
- <3> ロードしたハーフワード・データをワード長まで符号拡張し、1 ビット論理左シフトしたあとテーブルの先頭アドレス ( SWITCH 命令の次のアドレス ) を加算し、32 ビット・ターゲット・アドレスを生成します。
- <4> <3>で生成されたターゲット・アドレスへ分岐します。

**注意 1.** reg1 には、r0 を指定しないでください。

- 2. SWITCH 命令のテーブル読み出しのためのメモリからの読み出し操作では、プロセッサ保護が行われません。
- 3. メモリ保護 (PSW.DMP = 1) や周辺装置保護 (PSW.PP = 1) が有効である場合に、ユーザ・プログラムからのアクセスが禁止されている領域に配置されているテーブルからターゲット・アドレスを生成するためのデータをロードすることはできません。

## &lt; データ操作命令 &gt;

SXB	Sign extend byte  バイト・データの符号拡張
-----	--------------------------------------

[ 命令形式 ]        SXB reg1

[ オペレーション ]   GR [reg1]    sign-extend (GR [reg1] (7:0) )

[ フォーマット ]    Format I

[ オペコード ]      15                                  0  
00000000101RRRRR

[ フ ラ グ ]        CY    -  
                       OV    -  
                       S     -  
                       Z     -  
                       SAT   -

[ 説 明 ]            汎用レジスタ reg1 の最下位バイトをワード長に符号拡張します。

## &lt; データ操作命令 &gt;

SXH	Sign extend half-word ハーフワード・データの符号拡張
-----	--

[ 命令形式 ]      SXH reg1

[ オペレーション ]   GR [reg1] ← sign-extend (GR [reg1] (15:0) )

[ フォーマット ]      Format I

[ オペコード ]      15                                  0  
00000000111RRRRR

[ フ ラ グ ]          CY    -

OV    -

S     -

Z     -

SAT  -

[ 説 明 ]              汎用レジスタ reg1 の下位ハーフワードをワード長に符号拡張します。

## &lt; 特殊命令 &gt;

SYNCE	Synchronize exceptions  例外同期化命令
-------	---------------------------------------

[ 命令形式 ]         SYNCE

[ オペレーション ]   例外の同期化を待ってから実行を開始し，何もせず PC を +2 します。

[ フォーマット ]     Format I

[ オペコード ]       15                   0  
                      0000000000011101

[ フ ラ グ ]         CY    -  
                      OV    -  
                      S     -  
                      Z     -  
                      SAT  -

[ 説 明 ]           この命令以前の例外が同期化( synchranaization )されるまで ,実行の開始を待ち合わせます。実行が開始された後，この命令は何も操作を行わず完了します。

“ 例外の同期化 ” とは，先行する命令に起因して発生する例外がすべて CPU に通知され，優先順位判定の対象となる時点まで待ち合わせることを指します。したがって，この命令の実行以前に例外の受け付け条件を満たしていた場合，先行する命令に起因して発生するすべてのインプレサイス例外 ( FPI 例外 / PPI 例外 ) は，必ずこの命令の実行完了前に受け付けられます。

この命令はマルチ・プロセッシング環境において，タスク切り替え / 終了前の先行タスクの例外処理の完了を保証するために，利用することが可能です。



## &lt; 特殊命令 &gt;

SYNCP	Synchronize pipeline
	パイプライン同期化命令

[ 命令形式 ]          SYNCP

[ オペレーション ]    パイプラインの同期化を待ってから実行を開始し、何もせず PC を +2 します。

[ フォーマット ]      Format I

[ オペコード ]        15                                  0  

00000000000011111
-------------------

[ フラ グ ]            CY    -  
                        OV    -  
                        S     -  
                        Z     -  
                        SAT  -

[ 説 明 ]              この命令よりも前に実行されるべき、すべての命令の実行が完了するまで待ってから実行されます。この命令が実行された結果 PC は +2 されます。



## &lt; 特殊命令 &gt;

SYSCALL	System call  システム・コール例外
---------	-------------------------------

[ 命令形式 ]        SYSCALL    vector8

[ オペレーション ]    EIPC ← PC + 4 (復帰 PC)  
                           EIPSW ← PSW  
                           EIIC    例外要因コード (8000H-80FFH)  
                           ECR.EICC ← 例外要因コード (8000H-80FFH)  
                           PSW.EP ← 1  
                           PSW.ID ← 1  
                           If (MPM.AUE==1) is satisfied  
                               then PSW.IMP ← 0  
                                       PSW.DMP ← 0  
                                       PSW.NPV ← 0  
                                       PSW.PP ← 0  
                           if (vector8 <= SCCFG.SIZE) is satisfied  
                               then adr ← SCBP + zero-extend (vector8 logically shift left by 2)  
                               else adr ← SCBP  
                           PC ← SCBP + Load-memory (adr, Word)

[ フォーマット ]     Format X

[ オペコード ]        15                            0 31                            16  
                           11010111111vvvvv    00VVV00101100000

ただし、vvv は vector8 の上位 3 ビット、vvvvv は vector8 の下位 5 ビットです。

[ フラグ ]            CY    -  
                           OV    -  
                           S     -  
                           Z     -  
                           SAT   -

[ 説 明 ]            OS のシステム・サービス呼び出しを行います。

- <1> 復帰 PC ( SYSCALL 命令の次の命令のアドレス ) と PSW の内容を EIPC と EIPSW に退避します。
- <2> vector8 に対応する例外要因コードを ,EIIC レジスタ ,ECR.EICC ビットに格納します。例外要因コードは、8000H に vector8 を加算した値です。
- <3> PSW.ID,EP ビットをセット ( 1 ) します。

- <4> MPM.AUE ビットが 1 のときは PSW.PP, NPV, DMP, IMP ビットをクリア(0)します。
- <5> SCBP レジスタの値と, 2 ビット論理左シフトしワード長までゼロ拡張した vector8 を加算して 32 ビット・テーブル・エントリ・アドレスを生成します。  
ただし, vector8 がシステム・レジスタの SCCFG.SIZE ビットで指定された値より大きい場合, 上記加算に用いる vector8 は 0 として扱います。
- <6> <5>で生成されたアドレスのワードをロードします。
- <7> <6>のデータに SCBP レジスタの値を加算した 32 ビット・ターゲット・アドレスを生成します。
- <8> <7>で生成されたターゲット・アドレスへ分岐します。

- 注意 1. OS のシステム・サービス呼び出しに使用する専用命令です。ユーザ・プログラム中での使用は, 各 OS の機能仕様に従ってください。**
- 2. SYSCALL 命令のテーブル読み出しのためのメモリからの読み出し操作では, プロセッサ保護が行われません。
  - 3. メモリ保護 (PSW.DMP = 1) や周辺装置保護 (PSW.PP = 1) が有効である場合に, ユーザ・プログラムからのアクセスが禁止されている領域に配置されているテーブルからも, ターゲット・アドレスを生成するためのデータをロードすることができます。ユーザ・プログラム中での使用は, 各 OS の機能仕様に従ってください。

## &lt; 特殊命令 &gt;

TRAP	Trap  ソフトウェア例外
------	----------------------

[ 命令形式 ]        TRAP   vector5

[ オペレーション ]    EIPC ← PC + 4 (復帰 PC)  
                           EIPSW ← PSW  
                           ECR.EICC ← 例外要因コード ( 40H5FH )  
                           EIIC ← 例外要因コード ( 40H-5FH )  
                           PSW.EP ← 1  
                           PSW.ID ← 1  
                           If (MPM.AUE==1) is satisfied  
                               then    PSW.IMP ← 0  
                                       PSW.DMP ← 0  
                                       PSW.NPV ← 0  
                                       PSW.PP ← 0  
                           PC ← 00000040H ( vector5 が 00H-0FH ( 例外要因コード : 40H-4FH ) のとき )  
                               00000050H ( vector5 が 10H-1FH ( 例外要因コード : 50H-5FH ) のとき )

[ フォーマット ]     Format X

[ オペコード ]        15                            0 31                            16  
                           00000111111vvvvv    0000000100000000

ただし、vvvvv は vector5 です。

[ フラグ ]            CY    -  
                           OV    -  
                           S      -  
                           Z      -  
                           SAT   -

[ 説 明 ]            復帰 PC ( TRAP 命令の次の命令のアドレス ) と現在の PSW の内容を、それぞれ EIPC と EIPSW に退避し、例外要因コードを EIIC レジスタと ECR.EICC ビットに格納、PSW.EP, ID ビットをセット ( 1 ) します。また、MPM.AUE ビットがセット ( 1 ) されている場合、PSW.PP, NPV, DMP, IMP ビットをクリア ( 0 ) します。  
                           続いて、「vector5」で指定されるベクタ ( 00H-1FH ) に対応する例外ハンドラ・アドレスに  
                           分岐し、例外処理を開始します。

## &lt; 論理演算命令 &gt;

TST	Test  テスト
-----	-----------------

[ 命令形式 ]           TST   reg1, reg2

[ オペレーション ]   result ← GR [reg2] AND GR [reg1]

[ フォーマット ]       Format I

[ オペコード ]         15                             0  
rrrrr001011RRRRR

[ フラグ ]             CY    -

OV   0

S     演算結果のワード・データの MSB が 1 のとき 1, そうでないとき 0

Z     演算結果が 0 のとき 1, そうでないとき 0

SAT   -

[ 説 明 ]             汎用レジスタ reg2 のワード・データと汎用レジスタ reg1 のワード・データの論理積をとります。結果は格納されず, フラグだけが影響を受けます。汎用レジスタ reg1, reg2 は影響を受けません。

## &lt; ビット操作命令 &gt;

TST1	Test bit  ビット・テスト
------	-------------------------

- [ 命令形式 ]           (1) TST1 bit#3, disp16 [reg1]  
                          (2) TST1 reg2, [reg1]

- [ オペレーション ]   (1) adr ← GR [reg1] + sign-extend (disp16)  
                          token    Load-memory (adr, Byte)  
                          Z フラグ   Not (extract-bit (token, bit#3))  
  
                          (2) adr ← GR [reg1]  
                          token    Load-memory (adr, Byte)  
                          Z フラグ   Not (extract-bit (token, reg2))

- [ フォーマット ]      (1) Format VIII  
                          (2) Format IX

- [ オペコード ]
- |     |                  |                 |
|-----|------------------|-----------------|
| 15  | 0 31             | 16              |
| (1) | 11bbb111110RRRRR | ddddddddddddddd |
- 
- |     |                 |                  |
|-----|-----------------|------------------|
| 15  | 0 31            | 16               |
| (2) | rrrrr11111RRRRR | 0000000011100110 |

- [ フ ラ グ ]       CY    -  
                      OV    -  
                      S     -  
                      Z     指定したビットが 0 のとき 1, 指定したビットが 1 のとき 0  
                      SAT  -

- [ 説 明 ]           (1) まず、汎用レジスタ reg1 のワード・データと、ワード長まで符号拡張した 16 ビット・ディスプレイメントを加算して 32 ビット・アドレスを生成します。生成したアドレスのバイト・データの、3 ビットのビット・ナンバで指定されるビットが 0 ならば Z フラグをセット (1) し、1 ならばクリア (0) します。指定されたビットも含め、バイト・データは影響を受けません。  
                      読み出したバイト・データの指定ビットが 0 のとき Z フラグをセット (1) し、指定ビットが 1 のとき Z フラグをクリア (0) します。  
  
                      (2) まず、汎用レジスタ reg1 のワード・データを読み出して 32 ビット・アドレスを生成します。  
                      生成したアドレスのバイト・データの、汎用レジスタ reg2 の下位 3 ビットで指定されるビットが 0 ならば Z フラグをセット (1) し、1 ならばクリア (0) します。指定されたビットも含め、バイト・データは影響を受けません。  
                      読み出したバイト・データの指定ビットが 0 のとき Z フラグをセット (1) し、指定ビットが 1 のとき Z フラグをクリア (0) します。

< 論理演算命令 >

XOR	Exclusive OR
	排他的論理和

[ 命令形式 ] XOR reg1, reg2

[ オペレーション ] GR [reg2] ← GR [reg2] XOR GR [reg1]

[ フォーマット ] Format I

[ オペコード ]

15	0
rrrrr001001RRRRR	

[ フ ラ グ ]

CY	-
OV	0
S	演算結果のワード・データの MSB が 1 のとき 1, そうでないとき 0
Z	演算結果が 0 のとき 1, そうでないとき 0
SAT	-

[ 説 明 ] 汎用レジスタ reg2 のワード・データと汎用レジスタ reg1 のワード・データとの排他的論理和をとり, その結果を汎用レジスタ reg2 に格納します。汎用レジスタ reg1 は影響を受けません。

## &lt; 論理演算命令 &gt;

XORI	Exclusive OR immediate (16-bit)  排他的論理和
------	---

[ 命令形式 ]        XORI imm16, reg1, reg2

[ オペレーション ] GR [reg2] ← GR [reg1] XOR zero-extend (imm16)

[ フォーマット ]    Format VI

[ オペコード ]

15	0 31	16
rrrrr110101RRRRR	iiiiiiiiiiiiiiiiiii	

[ フラ グ ]

CY    -

OV    0

S     演算結果のワード・データの MSB が 1 のとき 1, そうでないとき 0

Z     演算結果が 0 のとき 1, そうでないとき 0

SAT   -

[ 説 明 ]        汎用レジスタ reg1 のワード・データとワード長までゼロ拡張した 16 ビット・イミディエ  
トの排他的論理和をとり, その結果を汎用レジスタ reg2 に格納します。汎用レジスタ reg1  
は影響を受けません。

## &lt; データ操作命令 &gt;

ZXB	Zero extend byte バイト・データのゼロ拡張
-----	----------------------------------

[ 命令形式 ]        ZXB reg1

[ オペレーション ]   GR [reg1] ← zero-extend (GR [reg1] (7:0) )

[ フォーマット ]     Format I

[ オペコード ]       15                                     0  
00000000100RRRRR

[ フ ラ グ ]        CY   -  
                   OV   -  
                   S    -  
                   Z    -  
                   SAT -

[ 説 明 ]            汎用レジスタ reg1 の最下位バイトをワード長にゼロ拡張します。





## 第6章 例 外

例外とは、特定の要因によって実行中のプログラムから別のプログラムへの強制的な分岐動作を発生する事象です。

それぞれの例外ごとの分岐先のプログラムを“例外ハンドラ”と呼びます。この例外ハンドラの先頭アドレスは例外ハンドラ・アドレス切り替え機能によって設定されます（6.4 例外ハンドラ・アドレス切り替え機能参照）。

**注意** V850E2M CPU では、V850E1 CPU, V850E2 CPU における割り込みを例外の一種として扱います。

### 6.1 例外の仕組み

ここでは、各例外の性質を特徴付ける次の要素について説明し、例外の仕組みを示します。

- ・ 例外要因一覧
- ・ 例外の種別
- ・ 例外処理フロー
- ・ 割り込み
- ・ 例外受け付けの優先順位
- ・ 例外の受け付け条件
- ・ 再開と回復
- ・ 例外レベルとコンテキスト退避
- ・ 復帰命令

#### 6.1.1 例外要因一覧

V850E2M CPUでは、次のような例外をサポートしています。

表6-1 例外要因一覧 (1/2)

名称	略称	発生要因	優先順位	例外レベル	種別	再開	回復	受け付け条件 (x:0または1)			例外要因コード <sup>※1</sup>	復帰PC <sup>※1</sup>	レジスタ更新値 (s:保持)				復帰命令	
								ID	NP	PSW			ハンドラ・オフセット <sup>※2</sup>	実行レベル <sup>※3</sup>	NP	EP		ID
CPU初期化	RESET	リセット入力	1	-	非同期	不可	不可	x	x	なし	なし	+0000H	0	0	0	1	なし	
FEレベル・ノンマスカブル割り込み	FENMI	FENMI入力 <sup>※4</sup>	3	FE	割り込み	不可	不可	x	x	0000020H	currentPC	+0020H	注5	1	0	1	FERET	
システム・エラー例外	SYSERR	SYSERR入力 <sup>※7</sup>	4	FE	注6	不可	不可	x	x	00000230H 00000233H	currentPC	+0030H	注5	1	1	1	FERET	
周辺装置保護例外	PPI	周辺装置保護違反	5	FE	インプレサイズ	可能	不可	x	0	00000432H	currentPC	+0030H	0	1	1	1	FERET	
タイミング監視例外	TSI	タイミング監視違反	6	FE	非同期	可能	可能	x	0	00000433H	currentPC	+0030H	0	1	1	1	FERET	
FEレベル・マスカブル割り込み	FEINT	FEINT入力 <sup>※4</sup>	7	FE	割り込み	可能	可能	x	0	00000010H	currentPC	+0010H	注5	1	0	1	FERET	
浮動小数点演算例外 (インプレサイズ)	FPI	FPU命令	8	EI	インプレサイズ	可能	不可	0	0	00000072H	currentPC	+0070H	注5	s	1	1	EIRET	
EIレベル・マスカブル割り込み	INT	INTn入力 <sup>※4</sup> (n=0-255)	9	EI	割り込み	可能	可能	0	0	00000080H 00001070H	currentPC	+0080H +1070H	注5	s	0	1	EIRET	

注1. 復帰PCおよびPSW, 例外要因コードの格納先は, 例外レベル (EI, FE) によって指定されます (nextPC: 次の命令, currentPC: 現在の命令)。

2. ベース・アドレスは, 「例外ハンドラ切り替え機能」によって設定されます。

3. 実行レベルの詳細は, 第3編 第4章 実行レベルを参照してください。

4. INTnからの入力になります。

5. MPM.AUE = 1のとき, 実行レベルが0に遷移します。MPM.AUE = 0の場合は, 変化しません。

6. 要因ごとに非同期, あるいはインプレサイズになります。製品の実装に依存します。

7. SYSERRの要因については, 製品の实装に依存します。

原則是に非同期ですが, データ・エラーのような命令に起因して起きるエラーが定義されている場合は, インプレサイズである場合があります。

SYSERRの要因については, 製品の实装に依存します。

備考 優先順位とは, 同時に発生し, 受け付け条件が成立している例外が, 複数のときの受け付けの優先順位を示します。

表6 - 1 例外要因一覧 (2/2)

名称	略称	発生要因	優先順位	例外レベル	種別	再開	回復	受け付け条件 (×:0または1)		例外要因 コード <sup>※1</sup>	復帰PC <sup>※1</sup>	レジスタ更新値 (s:保持)			復帰命令		
								PSW				実行レベル <sup>※3</sup>	PSW				
								ID	NP				NP	EP		ID	
実行保護例外	MIP	実行保護違反	11	FE	ブレイク	可能 <sup>※4</sup>	可能 <sup>※4</sup>	×	×	00000430H	currentPC	+0030H	0	1	1	1	FERET
メモリ・エラー例外	MEP	命令アクセス・ エラー入力 <sup>※5</sup>	12	FE	ブレイク	不可能 <sup>※4</sup>	不可能 <sup>※4</sup>	×	×	00000330H 00000333H	currentPC	+0030H	注6	1	1	1	FERET
データ保護例外	MDP	データ保護違反	13 <sup>※7</sup>	FE	ブレイク	可能 <sup>※4</sup>	可能 <sup>※4</sup>	×	×	00000431H	currentPC	+0030H	0	1	1	1	FERET
浮動小数点演算例外 (ブレイク)	FPP	FPU命令		EI	ブレイク	可能 <sup>※4</sup>	可能 <sup>※4</sup>	×	×	00000071H	currentPC	+0070H	注6	s	1	1	EIRET
コプロセッサ 使用不可例外	UCPOP	コプロセッサ命令		FE	ブレイク	可能 <sup>※4</sup>	可能 <sup>※4</sup>	×	×	00000530H 00000537H	currentPC	+0030H	注6	1	1	1	FERET
予約命令例外	RIEX	予約命令		FE	ブレイク	可能 <sup>※4</sup>	可能 <sup>※4</sup>	×	×	00000130H	currentPC	+0030H	注6	1	1	1	FERET
FEレベル・ ソフトウエア例外	FETRAPEX	FETRAP命令 (vector = 1H-FH)		FE	ブレイク	可能 <sup>※4</sup>	可能 <sup>※4</sup>	×	×	00000031H 0000003FH	nextPC	+0030H	注6	1	1	1	FERET
EIレベル・ ソフトウエア例外	EITRAP0	TRAP0n命令 (vector = 00-0FH)		EI	ブレイク	可能 <sup>※4</sup>	可能 <sup>※4</sup>	×	×	00000040H 0000004FH	nextPC	+0040H	注6	s	1	1	EIRET
EIレベル・ ソフトウエア例外	EITRAP1	TRAP1n命令 (vector = 10H-1FH)		EI	ブレイク	可能 <sup>※4</sup>	可能 <sup>※4</sup>	×	×	00000050H 0000005FH	nextPC	+0050H	注6	s	1	1	EIRET
システム・コール例外	SYSCALLEX	SYSCALL命令 (vector = 00H-FFH)		EI	ブレイク	可能 <sup>※4</sup>	可能 <sup>※4</sup>	×	×	00008000H 000080FFH	nextPC	注8	注6	s	1	1	EIRET

注1. 復帰PCおよびPSW, 例外要因コードの格納先は, 例外レベル (EI, FE) によって指定されます (nextPC: 次の命令, currentPC: 現在の命令)。

2. ベース・アドレスは, 「例外ハンドラ切り替え機能」によって設定されます。

3. 実行レベルの詳細は, 第3編 第4章 実行レベルを参照してください。

4. これらの要因は, 各命令のオペレーション順序に依存して発生します。

5. 命令アクセス・エラー入力については各製品のユーザーズ・マニュアルを参照してください。

6. MPM.AUE = 1のとき, 実行レベルが0に遷移します。MPM.AUE = 0の場合は, 変化しません。

7. 同一例外レベルのクリティカル・セクション中に発生した場合, 元の復帰PC/PSWなどの値を破壊する可能性があります。

8. 分岐先は, 5.3 命令セットのSYSCALL命令を参照してください。

備考 優先順位とは, 同時に発生し, 受け付け条件が成立している例外が, 複数のときの受け付けの優先順位を示します。

## 6.1.2 例外の種別

V850E2M CPUでは、例外を発生タイミングや性質によって、次の4つの種類に分類します。

- ・ プレサイス例外
- ・ インプレサイス例外
- ・ 非同期例外
- ・ 割り込み

### (1) プレサイス例外

ソフトウェア例外のように命令の実行の結果として常に例外を発生するものや、実行の結果が不正である場合に即座に例外を発生する場合のように、原因となった命令に同期して発生する正確な例外です。プレサイス例外は、後続の命令が実行される前に例外処理に分岐することができるため、多くの場合、例外処理後元の処理を正常に実行することが可能です<sup>\*</sup>。

プレサイス例外として分類される例外は次のものがあります。

- ・ 実行保護例外
- ・ メモリ・エラー例外
- ・ データ保護例外
- ・ 浮動小数点演算例外（プレサイス）
- ・ コプロセッサ使用不可例外
- ・ 予約命令例外
- ・ FEレベル・ソフトウェア例外
- ・ EIレベル・ソフトウェア例外
- ・ システム・コール例外

**注** メモリ・エラー例外は発生タイミングを制御できないため、元の処理に復帰できません。

## (2) インプレサイズ例外

命令のオペレーションを実行する前に、その命令を中断して受け付けられる例外です。中断する命令以前の命令の実行の結果が不正である場合に、遅延して発生する不正な例外です。インプレサイズ例外は、その原因となった命令の後続の命令が既に実行を完了している場合があり、例外の原因となった時点のCPUの状態が保存されていないため、例外の処理後に元の処理を回復して再実行することができません。

インプレサイズ例外として分類される例外は次のものがあります。

- ・周辺装置保護例外
- ・浮動小数点演算例外（インプレサイズ）

## (3) 非同期例外

命令のオペレーションを実行する前に、その命令を中断して受け付けられる例外です。現在実行中の命令の実行結果によって発生するわけではなく、その命令と無関係に発生します。

非同期例外として分類される例外は次のものがあります。

- ・CPU初期化
- ・システム・エラー例外（要因ごとに実装依存）
- ・タイミング監視例外

## (4) 割り込み

命令のオペレーションを実行する前に、その命令を中断して受け付けられる例外です。現在実行中の命令の実行結果によって発生するわけではなく、その命令と無関係に発生します。割り込みは、割り込みコントローラを介して任意のユーザ・プログラムを実行するための例外です。

割り込みとして分類される例外は次のものがあります。

- ・FEレベル・ノンマスカブル割り込み
- ・FEレベル・マスカブル割り込み
- ・EIレベル・マスカブル割り込み

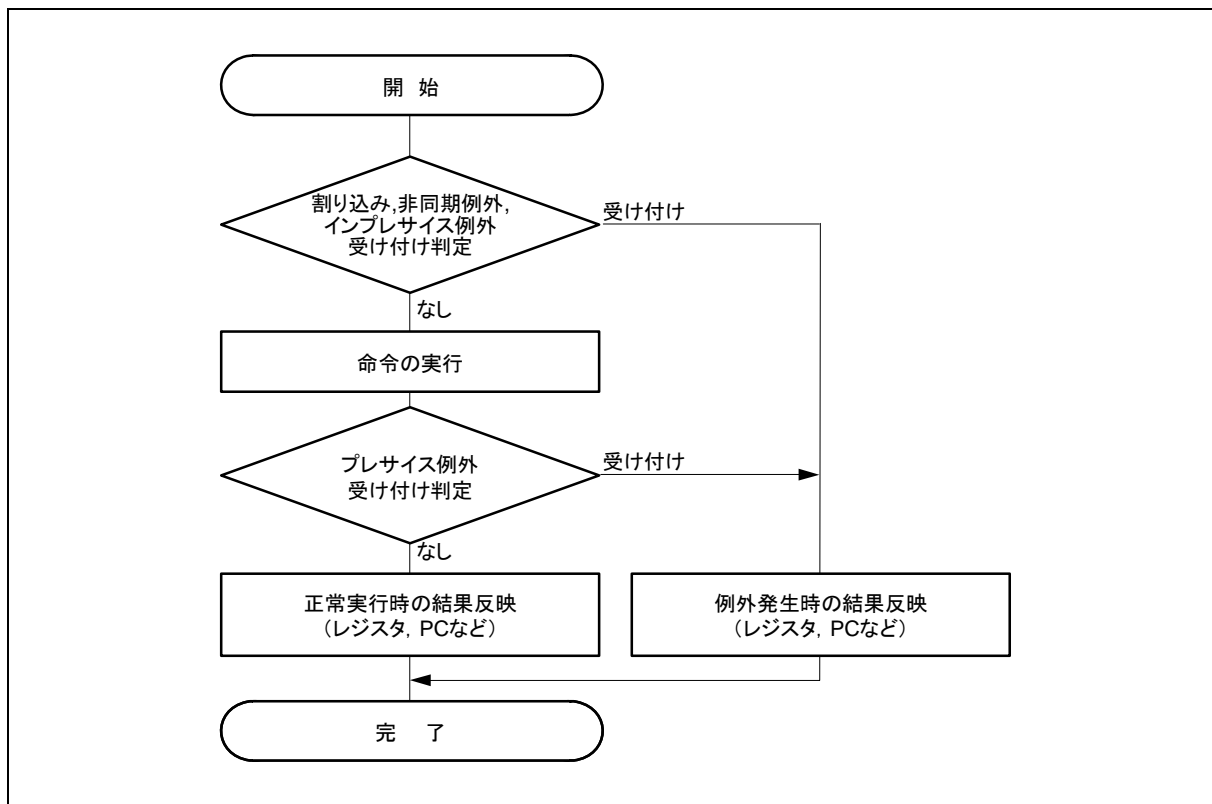
割り込み発生時には他の例外と異なり、PSW.EPビットがクリア（0）されます。このため、復帰命令実行時に、外部の割り込みコントローラに対して、例外処理ルーチンの終了を通知します。割り込みからの復帰命令実行時には必ずPSW.EPビットがクリア（0）されている状態で実行してください。

**注意** PSW.EPビットは、割り込み（INT0-INT255, FEINT, FENMI）の受け付け時にのみクリア（0）します。その他の例外ではPSW.EPビットをセット（1）します。

PSW.EPビットがセット（1）された状態で、割り込みによって起きた例外処理ルーチンからからの復帰命令を実行すると、外部の割り込みコントローラ上のリソースが解放されず、誤動作を引き起こす可能性があります。

### 6.1.3 例外処理フロー

例外と命令の実行と結果の反映（レジスタなどへの書き込み）の処理フローを次に示します。



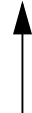
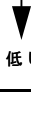
割り込みと非同期例外，インプレサイズ例外は命令の実行前に受け付けの可否が判定され，受け付け可能であれば例外処理に分岐します。このため，例外処理後は実行が中断された現在の命令を再実行する必要がありますため，復帰PCは現在の命令（Current PC）を格納します。

一方，プレサイズ例外は命令の実行の結果，例外が発生し無条件に例外処理に分岐します。このとき，プレサイズ例外となる要因が複数発生した場合，優先順位に従って最高優先順位のものを受け付けます。復帰PCはその例外の性質によって決定され，ソフトウェア・トラップなどのような，例外を発生した命令自身の再実行が必要のない場合は，次命令（Next PC）を格納し，再実行が必要なメモリ保護例外などの場合においては現在の命令（Current PC）を格納します。

### 6.1.4 例外受け付けの優先順位と保留条件

例外の受け付けとは、ある例外要因によって例外が発生し、その例外要因に対応した例外ハンドラへ分岐することを示します。CPUは、ある瞬間にひとつの例外のみを受け付け可能です。このとき受け付けする例外は次の優先順位に従って決定されます。同時に発生している例外が複数ある場合、受け付けられなかった例外は保留されます（CPU初期化を除きます。詳細は6.2.5 特殊な動作を参照してください）。

表6-2 例外優先順位

優先度	例外	発生タイミング	
高い   低い	CPU初期化（RESET）	命令の実行前	
	FEレベル・ノンマスカブル割り込み（FENMI）		
	システム・エラー例外（SYSERR）		
	周辺装置保護例外（PPI）		
	タイミング監視例外（TSI）		
	FEレベル・マスカブル割り込み（FEINT）		
	浮動小数点演算例外（インプレサイス）（FPI）		
	EIレベル・マスカブル割り込み（INT）		
	実行保護例外（MIP）		命令の実行後
	メモリ・エラー例外（MEP）		
データ保護例外（MDP） <sup>※</sup>			
浮動小数点演算例外（プレサイス）（FPP） <sup>※</sup>			
コプロセッサ使用不可例外（UCPOP） <sup>※</sup>			
予約命令例外（RIEX） <sup>※</sup>			
FEレベル・ソフトウェア例外（FETRAPEX） <sup>※</sup>			
EIレベル・ソフトウェア例外（EITRAP0/EITRAP1） <sup>※</sup>			
システム・コール例外（SYSCALLEX） <sup>※</sup>			

注 優先順位は同じです。命令のオペレーション内容に依存して発生します。

### 6.1.5 例外の受け付け条件

一部の例外は特定の条件によって、例外の受け付けが保留される場合があります。

表6-1において、受け付け条件の欄に“0”とある例外は、該当ビットが“0”であるときに例外の受け付けが可能となります。このような例外では、該当ビットが“1”であると例外の受け付けが保留されますが、該当ビットが“0”に変化して受け付け条件が成立すると、例外の受け付けが可能状態となります。



### 6.1.6 再開と回復

例外処理を行った場合、例外の受け付けによって中断した元のプログラムに対して影響を与える可能性があります。この影響は「再開」と「回復」という2つの観点で表現されます。

- ・再開：元のプログラムの中断した位置から実行再開が可能／不可能であることを示します。
- ・回復：元のプログラムを中断した時点のプロセッサ状態（汎用レジスタ、システム・レジスタなどのプロセッサ資源の状態）への回復が可能／不可能であることを示します。

### 6.1.7 例外レベルとコンテキスト退避

#### (1) 例外レベル

V850E2M CPUでは例外要因を3つの例外レベル（EIレベル、FEレベル、DBレベル）に階層化して管理します。例外発生時に例外要因、復帰PC、復帰PSWが自動的にレベルごとに対応する復帰レジスタへ格納されます（CPU初期化を除きます。詳細は6.2.5 特殊な動作を参照してください）。

表6-3 例外レベル

EIレベル例外	FEレベル例外	DBレベル例外 <sup>※</sup>
EIレベル・マスカブル割り込み	システム・エラー例外	デバッグ例外 <sup>※</sup>
EIレベル・ソフトウェア例外	FEレベル・マスカブル割り込み	
浮動小数点演算例外	FEレベル・ノンマスカブル割り込み	
システム・コール例外	FEレベル・ソフトウェア例外	
	予約命令例外	
	メモリ・エラー例外	
	実行保護例外	
	データ保護例外	
	周辺装置保護例外	
	タイミング監視例外	
	コプロセッサ使用不可例外	

注 DBレベル例外は開発ルール向けのデバッグ機能で使用します。

## (2) コンテキスト退避

受け付け条件が定められている一部の例外は、ほかの例外受け付け時に自動的にセットされる保留ビット (PSW.ID, NPビット) によって、例外処理の開始時点では受け付けられない状態となっています。

同一レベルの例外を再度受け付け可能な多重例外処理を可能にするためには、これらの復帰レジスタ、およびそれぞれの例外要因ごとに定められた特定の情報をスタックなどへ退避しておく必要があります。これらの退避が必要な情報を「コンテキスト」と呼びます。

原則として、コンテキストの退避前に同一のレベルへの例外を発生させないように注意する必要があります。

コンテキスト退避のための作業を行う際に利用できる作業用システム・レジスタと、多重例外処理を可能にするために最低限、退避が必要なシステム・レジスタを、基本コンテキスト・レジスタと呼びます。基本コンテキスト・レジスタはレベルごとに用意されています。

表6-4 基本コンテキスト・レジスタ

例外レベル	基本コンテキスト・レジスタ
EIレベル	EIPC, EIPSW, EIIC, EIWR
FEレベル	FEPC, FEPSW, FEIC, FEWR
DBレベル <sup>注</sup>	DBPC <sup>注</sup> , DBPSW <sup>注</sup> , DBIC <sup>注</sup> , DBWR <sup>注</sup>

注 DBレベル例外は開発ルール向けのデバッグ機能で使します。

## 6.1.8 復帰命令

例外処理からの復帰には、それぞれの例外レベルに対応した復帰命令 (EIRET, FERET) の実行によって行います。

スタックなどにコンテキストを退避している場合は、復帰命令の実行前にコンテキストの復帰を必ず行ってください。また、回復不可能な例外からの復帰時には、元のプログラムが例外を起こす直前の状態には回復できず、例外が起きなかった場合の実行結果と異なる可能性があることに注意してください。

### (1) EIRET命令

EIレベルの例外処理からの復帰は、EIRET命令により行われます。

EIRET命令の実行により、CPUは次の処理を行い復帰PCのアドレスへ制御を移します。

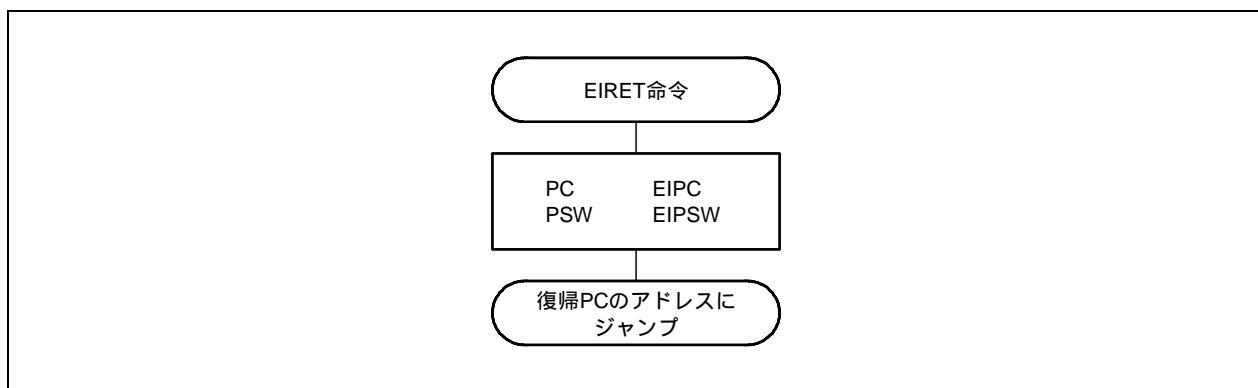
<1> EIPC, EIPSWレジスタから復帰PC, PSWを取り出します。

<2> 取り出した復帰PC, PSWのアドレスに制御を移します。

EP = 0の場合、例外ルーチンの実行を終了したことを外部(割り込みコントローラ)などに通知します。

EIレベルの例外処理からの復帰を次に示します。

図6 - 1 EIRET命令



## (2) FERET命令

FEレベルの例外処理からの復帰は、FERET命令により行われます。

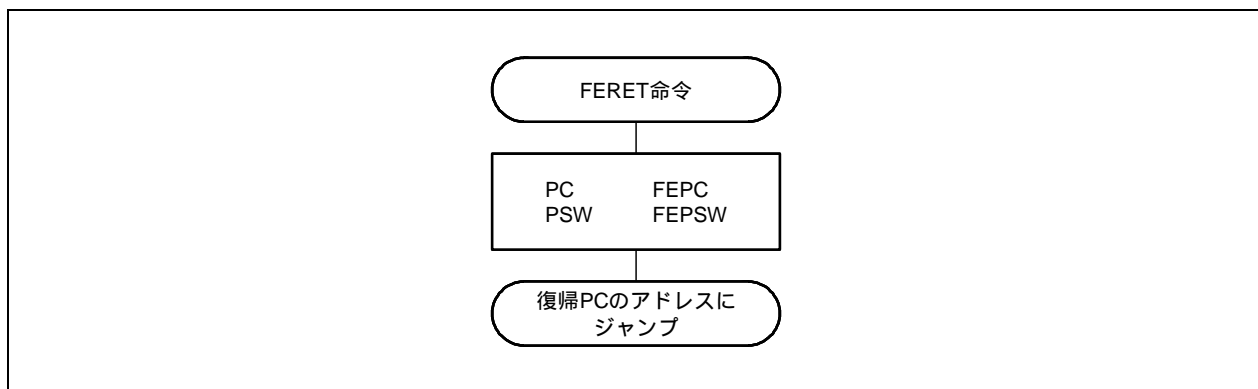
FERET命令の実行により、CPUは次の処理を行い復帰PCのアドレスへ制御を移します。

<1> FEPC, FEPSWレジスタから復帰PC, PSWを取り出します。

<2> 取り出した復帰PC, PSWのアドレスに制御を移します。

EP = 0の場合、例外ルーチンの実行を終了したことを外部(割り込みコントローラ)などに通知します。

図6 - 2 FERET命令



(3) RETI命令による割り込み，EIレベル・ソフトウェア例外 (EITRAP0/EITRAP1) からの復帰

**注意** RETI 命令は V850E1, V850E2 CPU との後方互換のために定義しており，原則として，RETI 命令の使用を禁止しています。修正の不可能な既存プログラム以外の RETI 命令はすべて，EIRET または FERET 命令に置き換えて使用してください。

割り込み，EI レベル・ソフトウェア例外 (EITRAP0/EITRAP1) からの復帰以外に使用された場合の動作は不定です。

割り込み，EIレベル・ソフトウェア例外 (EITRAP0/EITRAP1) からの復帰はRETI命令によっても行えます。RETI命令の実行により，CPUは次の処理を行い復帰PCのアドレスへ制御を移します。

<1> PSW.EPビットが0，かつ PSW.NPビットが1の場合，FEPC, FEPSWから復帰PC, PSWを取り出します。それ以外の場合，EIPC, EIPSWから復帰PC, PSWを取り出します。

<2> 取り出した復帰PC, PSWのアドレスに制御を移します。

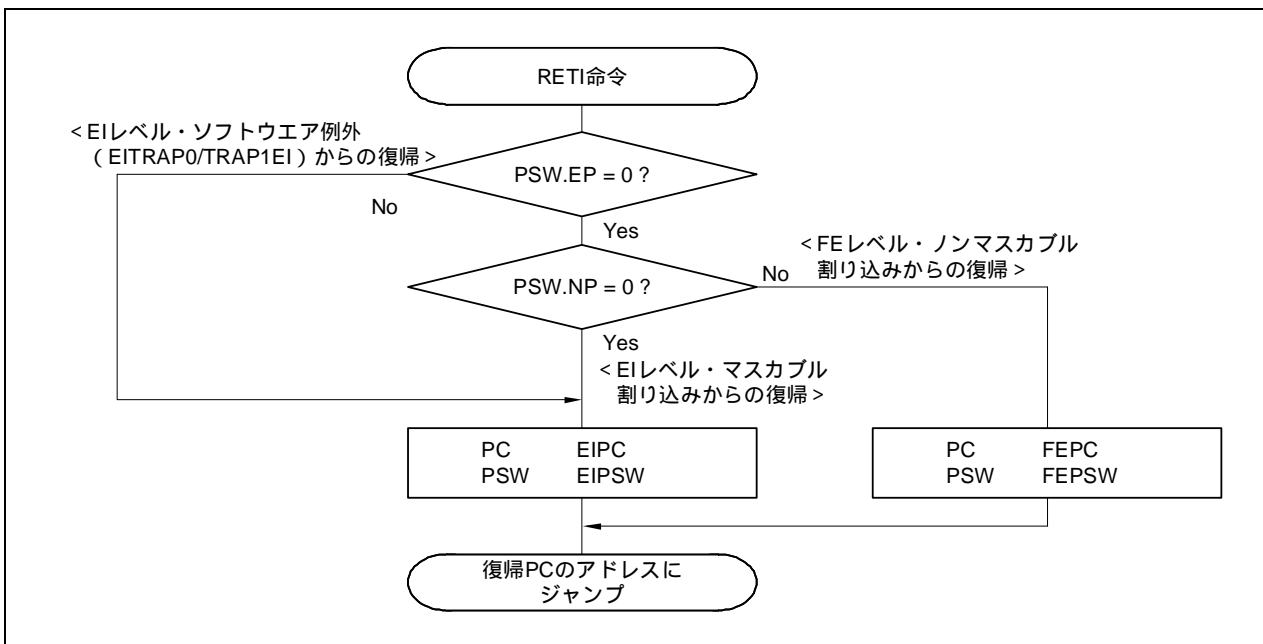
各例外処理からの復帰時は，PC, PSWを正常にリストアするために，RETI命令の直前で，LDSR命令を使用し，PSW.NPビット，PSW.EPビットの各フラグを次の状態にしておく必要があります。

- FEレベル・マスク可能割り込み処理からの復帰時<sup>※</sup> : PSW.NPビット=1，PSW.EPビット = 0
- EIレベル・マスク可能割り込み処理からの復帰時 : PSW.NPビット = 0 ,PSW.EPビット = 0
- EIレベル・ソフトウェア例外 (EITRAP0/EITRAP1) 処理からの復帰時 : PSW.EPビット = 1

**注** FENMIは，RETI命令による復帰はできません。例外処理後にシステム・リセットを行ってください。また，FENMIはPSW.NPビットがセット (1) されていても受け付けられません。

RETI命令による復帰の処理形態を次に示します。

図6 - 3 RETI命令



## 6.2 例外発生時の動作

### 6.2.1 受け付け条件のないEIレベル例外

命令やPSWの状態変更などによる受け付け禁止ができない常時受け付けが可能な例外です。

受け付け条件のないEIレベル例外が発生した場合、CPUは次の処理を行い例外ハンドラ・ルーチンへ制御を移します。

- <1> 復帰PCをEIPCに退避します。
- <2> 現在のPSWをEIPSWへ退避します。
- <3> EIICレジスタに例外要因コードを書き込みます<sup>※</sup>。
- <4> PSW.IDビットをセット(1)します。
- <5> PSW.EPビットをセット(1)します。
- <6> MPM.AUEビットがセット(1)されている場合は、PSW.PP, NPV, DMP, IMPビットをクリア(0)します。それ以外の場合は、PSW.PP, NPV, DMP, IMPは更新しません。
- <7> PCに例外ハンドラ・アドレスをセットし、制御を移します。

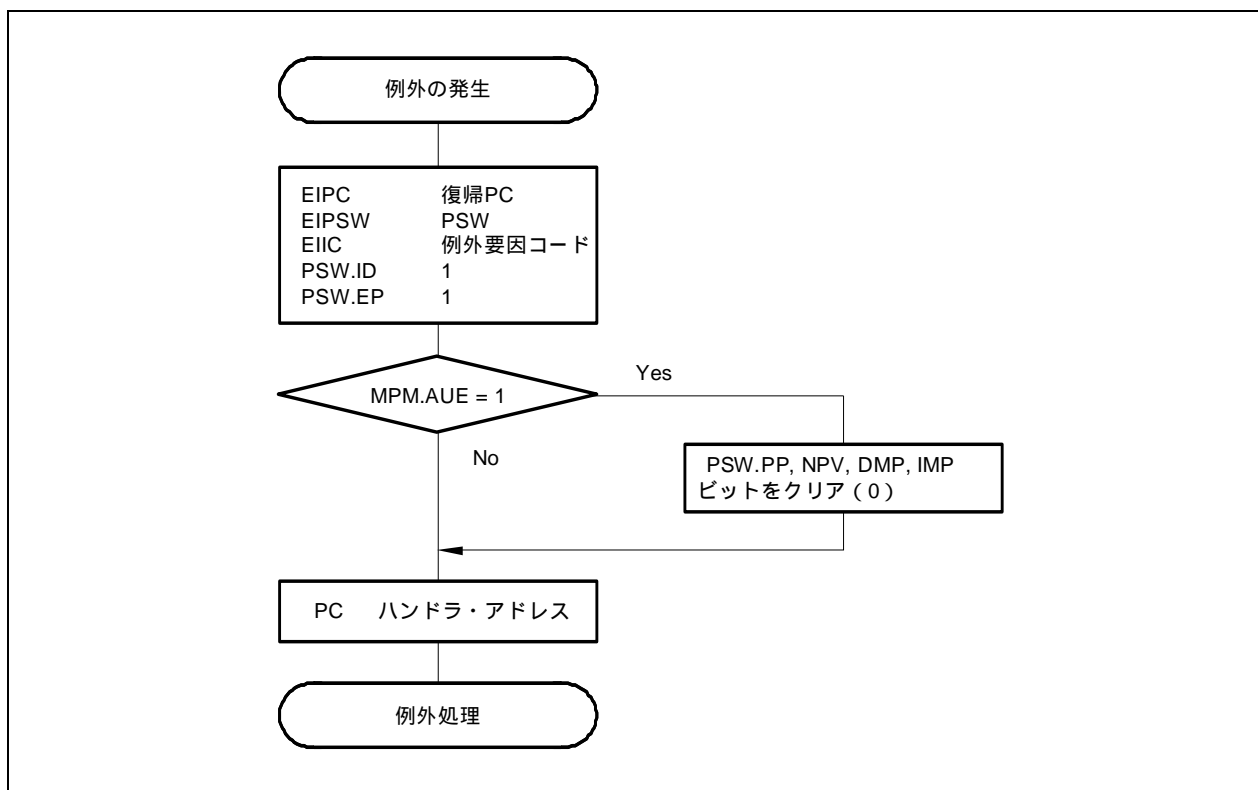
**注** ECRレジスタの下位16ビット(EICC)にも例外要因コードが書き込まれますが、修正の不可能な既存プログラム以外はEIICレジスタを使用してください。

状態退避レジスタには、EIPC, EIPSWを使用します。ほかのEIレベル例外処理中(PSW.NPビットが1、またはPSW.IDビットが1のとき)に発生しても、受け付け条件のないEIレベル例外は、受け付けられます。したがって、EI例外レベルのコンテキスト退避処理以前に発生した場合、元の復帰PC, PSWを破壊する可能性があります。

なお、EIPC, EIPSWは1組しかないので、多重例外を許可する場合には、事前にプログラムによってコンテキストを退避する必要があります。

受け付け条件のないEIレベル例外の処理形態を次に示します。

図6 - 4 受け付け条件のないのEIレベル例外の処理形態



## 6.2.2 受け付け条件のあるEIレベル例外

PSW.IDビット、NPビットにより受け付けを保留できる例外です。

受け付け条件ありのEIレベル例外が発生した場合、CPUは次の処理を行い、例外ハンドラ・ルーチンへ制御を移します。

- <1> PSW.NPビットがセット（1）されている場合は、受け付けを保留します。
- <2> PSW.IDビットがセット（1）されている場合は、受け付けを保留します。
- <3> 復帰PCをEIPCに退避します。
- <4> 現在のPSWをEIPSWへ退避します。
- <5> EIICに例外要因コードを書き込みます<sup>注</sup>。
- <6> PSW.IDビットをセット（1）します。
- <7> 割り込みの場合は、PSW.EPビットをクリア（0）、それ以外の例外の場合はPSW.EPビットをセット（1）します。
- <8> MPM.AUEビットがセット（1）されている場合は、PSW.PP, NPV, DMP, IMPビットをクリア（0）します。それ以外の場合は、PSW.PP, NPV, DMP, IMPIは更新しません。
- <9> PCに例外ハンドラ・アドレスをセットし、制御を移します。

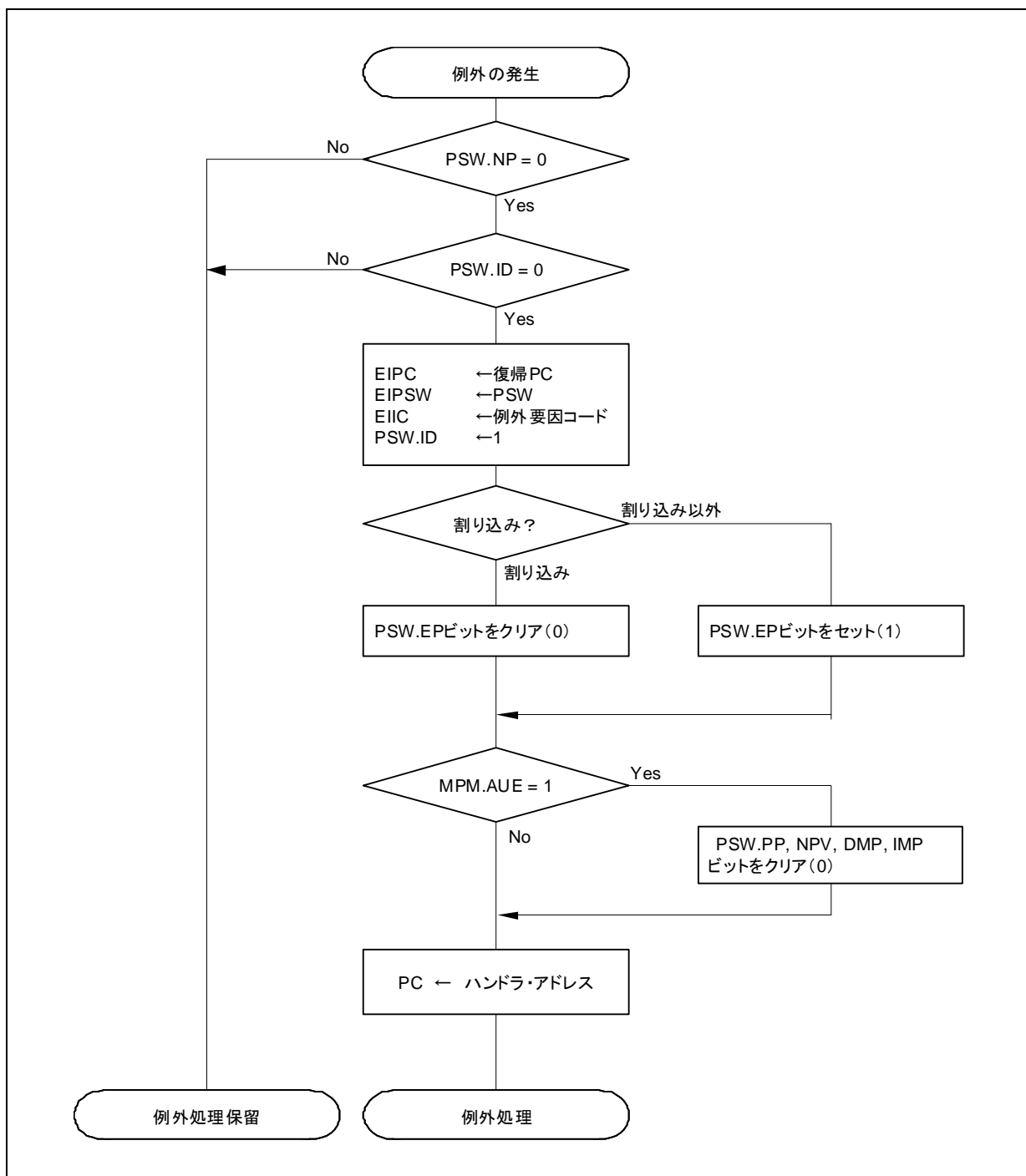
**注** ECRレジスタの下位16ビット（EICC）にも例外要因コードが書き込まれますが、修正の不可能な既存プログラム以外はEIICレジスタを使用してください。

状態退避レジスタにはEIPC、EIPSWを使用します。ほかのEIレベル例外処理中（PSW.NPビットが1、またはPSW.IDビットが1のとき）に発生した受け付け条件のあるEIレベル例外は保留されます。この場合、LDSR命令、EI命令などを使用してPSW.NPビットとIDビットをクリア（0）すると、保留していた受け付け条件のあるEIレベル例外処理が受け付けられます。

なお、EIPC、EIPSWは1組しかいないため、多重例外を許可する場合には、事前にプログラムによってコンテキストを退避する必要があります。

受け付け条件のあるEIレベル例外の処理形態を次に示します。

図6-5 受け付け条件のあるEIレベル例外の処理形態





### 6.2.3 受け付け条件のないIFEレベル例外

命令やPSWの状態変更などによる受け付け禁止ができない常時受け付けが可能な例外です。

受け付け条件のないIFEレベル例外が発生した場合、CPUは次の処理を行い例外ハンドラ・ルーチンへ制御を移します。

- <1> 復帰PCをFEPCへ退避します。
- <2> 現在のPSWをFEPSWへ退避します。
- <3> FEICに例外要因コードを書き込みます<sup>注1</sup>。
- <4> PSW.NP, IDビットをセット(1)します。
- <5> 割り込みの場合は、PSW.EPビットをクリア(0)、それ以外の例外の場合はPSW.EPビットをセット(1)します。
- <6> MPM.AUEビットがセット(1)されている場合は、PSW.PP, NPV, DMP, IMPビットをクリア(0)します。それ以外の場合は、PSW.PP, NPV, DMP, IMPは更新しません<sup>注2</sup>。
- <7> PCに例外ハンドラ・アドレスをセットし、制御を移します。

注1. ECRレジスタの上位16ビット(FECC)にも例外要因コードが書き込まれますが、修正の不可能な既存プログラム以外はFEICレジスタを使用してください。

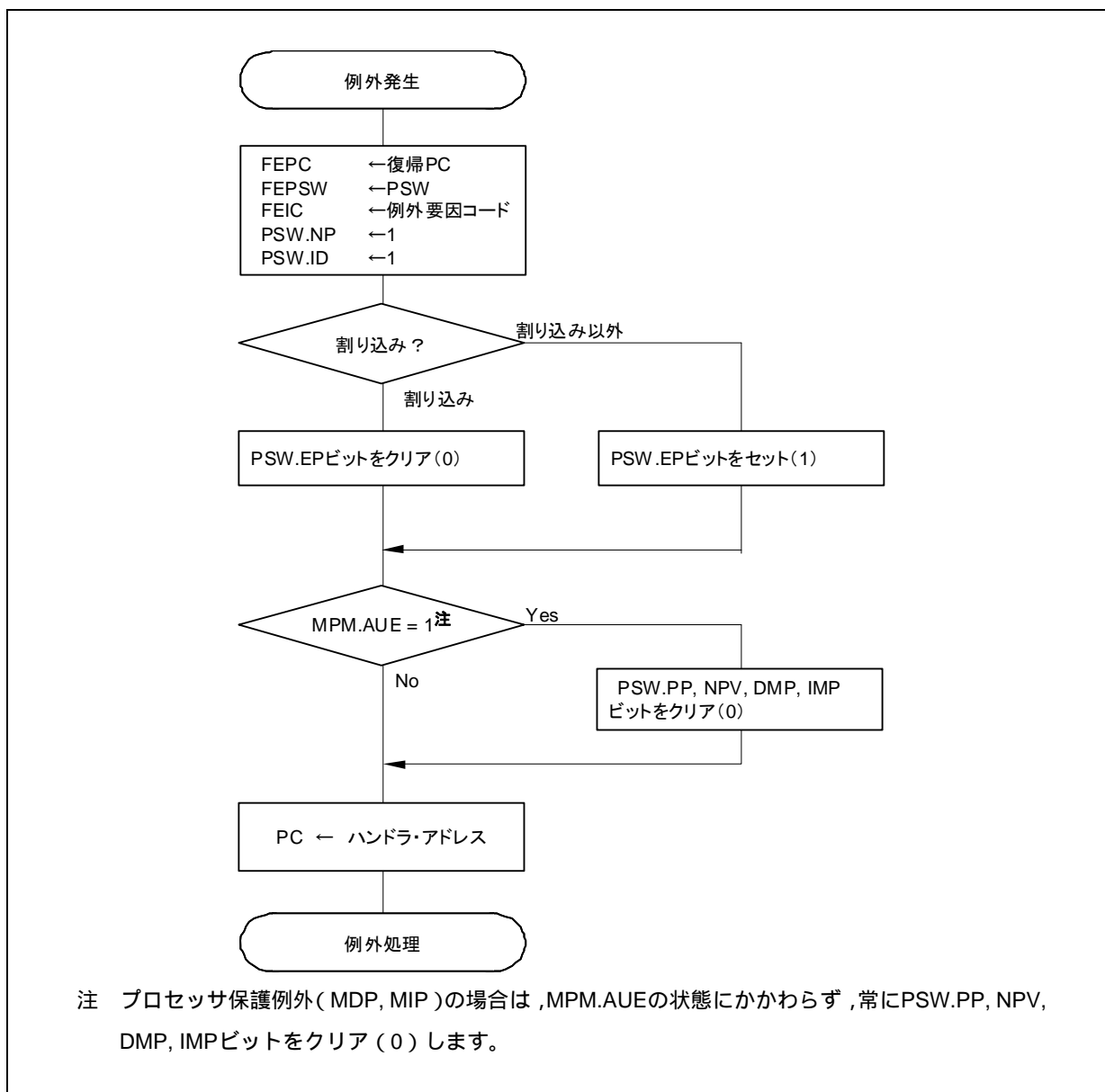
2. プロセッサ保護に関する例外(MDP例外, MIP例外)の場合は、PSW.PP, NPV, DMP, IMPは常にクリア(0)します。

状態退避レジスタには、FEPC, FEPSWを使用します。ほかのFEレベル例外処理中(PSW.NPビットが1のとき)に発生しても、受け付け条件のないIFEレベル例外は、受け付けられます。従って、FE例外レベルのコンテキスト退避処理以前に発生した場合、元の復帰PC, PSWを破壊する可能性があります。

なお、FEPC, FEPSWは1組しかないため、多重例外を許可する場合には、事前にプログラムによってコンテキストを退避する必要があります。

受け付け条件のないIFEレベル例外の処理形態を次に示します。

図6 - 6 受け付け条件のないFEレベル例外の処理形態



## 6.2.4 受け付け条件のあるFEレベル例外

PSW.NPビットにより受け付けを保留できる例外です。

受け付け条件のあるFEレベル例外が発生した場合、CPUは次の処理を行い、例外ハンドラ・ルーチンへ制御を移します。

- <1> PSW.NPビットがセット(1)されている場合は、受け付けを保留します。
- <2> 復帰PCをFEPCに退避します。
- <3> 現在のPSWをFEPSWへ退避します。
- <4> FEICに例外要因コードを書き込みます<sup>注1</sup>。
- <5> PSW.NP,IDビットをセット(1)します。
- <6> 割り込みの場合は、PSW.EPビットをクリア(0)、それ以外の例外の場合はPSW.EPビットをセット(1)します。
- <7> MPM.AUEビットがセット(1)されている場合は、PSW.PP, NPV, DMP, IMPビットをクリア(0)します。それ以外の場合は、PSW.PP, NPV, DMP, IMPは更新しません<sup>注2</sup>。
- <8> PCに例外ハンドラ・アドレスをセットし、制御を移します。

**注1** ECRレジスタの下位16ビット(FECC)にも例外要因コードが書き込まれますが、修正の不可能な既存プログラム以外はFEICレジスタを使用してください。

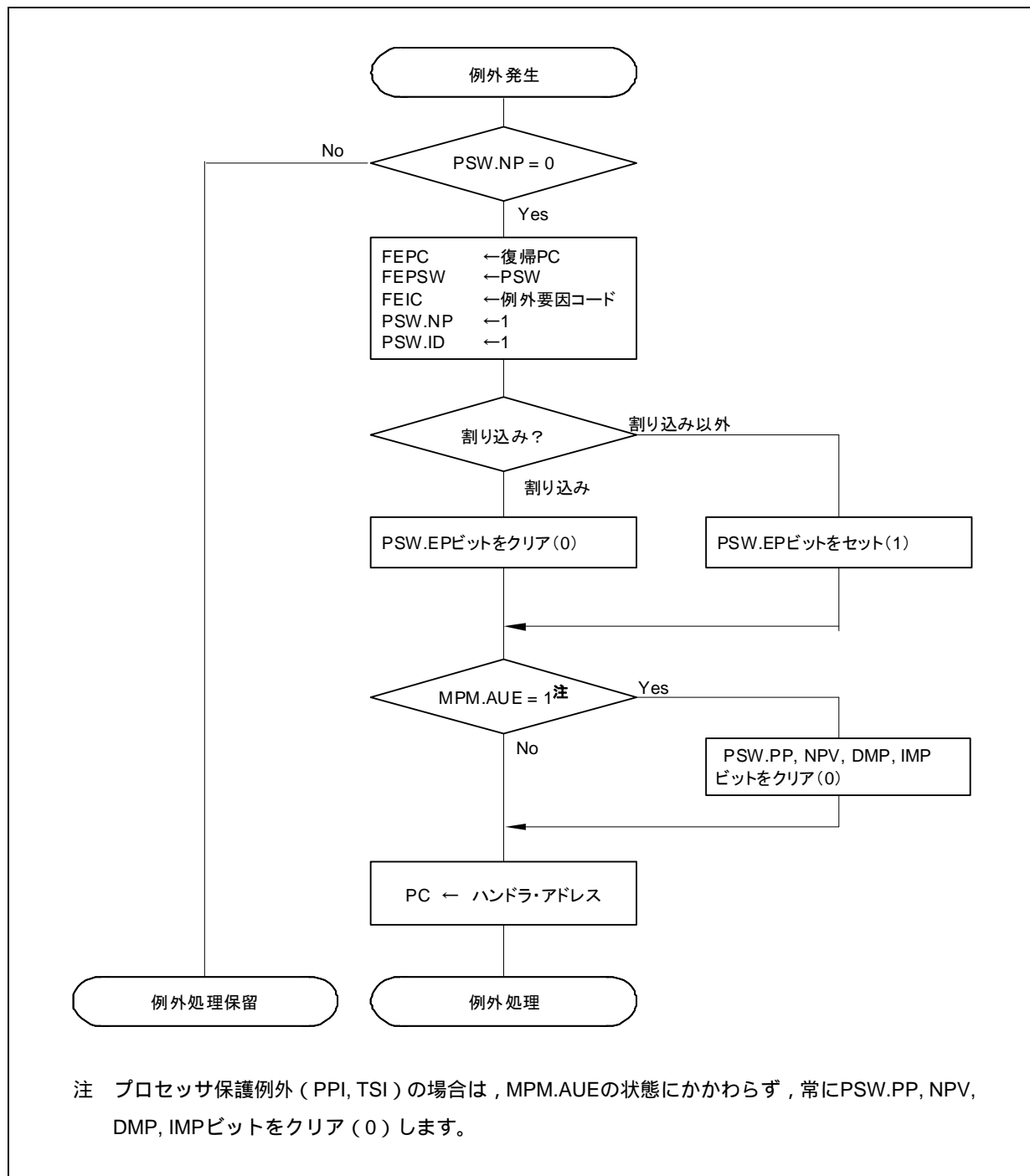
2. プロセッサ保護に関する例外(PPI例外, TSI例外)の場合は、PSW.PP, NPV, DMP, IMPは常にクリア(0)します。

状態退避レジスタにはFEPC, FEPSWを使用します。ほかのFEレベル例外処理中(PSW.NPビットが1のとき)に発生した受け付け条件のあるFEレベル例外は保留されます。この場合、LDSR命令を使用してPSW.NPビットをクリア(0)すると、保留していた受け付け条件のあるFEレベル例外処理が受け付けられます。

なお、FEPC, FEPSWは1組しかないので、多重例外を許可する場合には、事前にプログラムによってコンテキストを退避する必要があります。

受け付け条件のあるFEレベル例外の処理形態を次に示します。

図6-7 受け付け条件のあるFEレベル例外の処理形態



## 6.2.5 特殊な動作

### (1) PSWレジスタのEPビット

割り込みを受け付けた場合、PSW.EPビットをクリア(0)します。割り込み以外の例外を受け付けた場合、PSW.EPビットをセット(1)します。

EPビットの状態によって、EIRET, FERET, RETI命令の実行時の動作が変化します。EPビットがクリア(0)されている場合、外部の割り込みコントローラに対して、例外処理ルーチンの終了を通知します。これは、割り込みの受け付け/割り込みからの復帰によって、割り込みコントローラ上のリソースを適切に制御するために必要な機能です。

割り込みからの復帰する際には、必ずEPビットをクリア(0)した状態で復帰命令を実行してください。

### (2) PSWレジスタのPP, NPV, DMP, IMPビット

プロセッサ保護例外を受け付けた場合、PSW.PP, NPV, DMP, IMPを無条件にクリア(0)します。プロセッサ保護例外以外の例外を受け付けた場合は、MPM.AUEビットの設定により動作が異なります。MPM.AUEビットがセット(1)されている場合(実行レベル自動遷移機能が有効の場合)は、PSW.PP, NPV, DMP, IMPビットをクリア(0)します。MPM.AUEビットがクリア(0)されている場合(実行レベル自動遷移機能が無効の場合)は、PSW.PP, NPV, DMP, IMPビットの値を更新せず、前値を保持します。

### (3) コプロセッサ使用不可例外

コプロセッサ使用不可例外は、製品ごとの機能仕様によって、発生するオペコードが変化します。

コプロセッサ命令と定義されたオペコードに対して、製品上で搭載されていない、あるいは、動作状態によって使用が許可されていない場合に、これらのコプロセッサ命令を実行しようとした場合、ただちにコプロセッサ使用不可例外(UCPOP)が発生します。

詳細は、**第7章 コプロセッサ使用不可状態**を参照してください。

### (4) 予約命令例外

将来の機能拡張のために予約され、命令が定義されていないオペコードに対して実行を行う際、予約命令例外(RIEX)が発生します。

ただし、個々のオペコードごとに、次の2種類の動作のいずれかを行うことを製品仕様により定義することがあります。

- ・ 予約命令例外が発生する
- ・ いずれかの定義された命令として動作する

常に予約命令例外が発生するオペコードがRIE命令として定義されています。

**(5) システム・コール例外**

システム・コール例外は、オペコードによって指定されるベクタの値とSCCFG.SIZEビットの値によって、参照するテーブル・エントリが選択され、そのテーブル・エントリの内容とSCBPレジスタの値に従って例外ハンドラ・アドレスを計算します。

たとえば、SCCFG.SIZEによってテーブル・サイズ $n$ が指定された場合、次のようにテーブル・エントリを選択します。 $n < 255$ の場合には、ベクタ $n+1 \sim 255$ から参照されるテーブル・エントリは、テーブル・エントリ0であることに注意してください。

ベクタ	例外要因コード	参照するテーブル・エントリ
0	0000 8000H	テーブル・エントリ 0
1	0000 8001H	テーブル・エントリ 1
2	0000 8002H	テーブル・エントリ 2
(中略)	:	:
$n - 1$	0000 8000H + ( $n - 1$ ) H	テーブル・エントリ $n - 1$
$n$	0000 8000H + $n$ H	テーブル・エントリ $n$
$n + 1$	0000 8000H + ( $n + 1$ ) H	テーブル・エントリ 0
(中略)	:	:
254	0000 80FEH	テーブル・エントリ 0
255	0000 80FFH	テーブル・エントリ 0

**注意** テーブル・エントリ0は、SCCFG.SIZEで指定する $n$ を越えたベクタが指定された場合にも選択されるため、エラー処理ルーチンを配置してください。

**(6) リセット**

リセット入力によるCPU初期化も、例外と同様の動作を行いますが、EIレベル例外、FEレベル例外のいずれにも属しません。動作としては、受け付け条件のない例外と同様ですが、各レジスタは初期値に変化します。また、リセットからの復帰等も行えません。

また、CPU初期化と同等に発生していた例外は、すべて取り消され、CPU初期化後にも受け付けられることはありません。

## 6.3 例外の管理

V850E2M CPU は、マルチプログラミングにおけるタスク間の相互干渉を防ぐことを目的とした、例外を管理する次の機能を備えています。

- ・ 例外受け付け / 復帰時の例外同期機能
- ・ 例外同期命令 (SYNCE)
- ・ 保留中例外の確認機能
- ・ 保留中例外の取り下げ機能

V850E2M CPU で定義される例外には、例外の原因が発生したあと、例外処理が開始されるまでの間に遅延が生じる可能性のあるインプレサイス例外や、タスクに結び付いていながら命令実行と同期せずに発生する非同期例外が存在します。特定のタスクに対して発生するインプレサイス例外、非同期例外は、タスク切り替え時あるいはタスク終了時に処理を行い、その例外が処理されないまま、次のタスクに移行することを防ぐ必要があります。

V850E2M CPU では、例外管理機能を利用することで、タスク切り替えやタスク終了前にそのタスクに起因するすべての例外を待ち合わせ、順序よく処理することが可能です。これにより、あるタスクの不正な処理の影響が、他のタスクに及ぶことを防止します。また、例外を処理することなくタスクの終了処理を完了してしまうことを防止できます。

例外の管理が必要となるのは、次のような場合です。

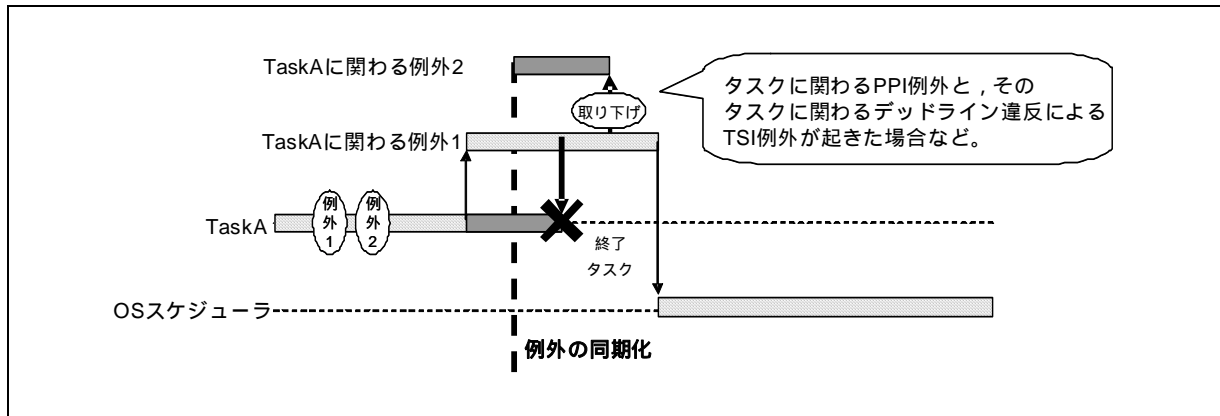
### (1) 例外の起因となったタスクを中断し終了させる場合

同一タスクに対し、複数の例外が同時に発生した場合、これらの例外の処理には注意が必要です。例外優先順位に従っていずれかの例外が先に受け付けられ例外処理を開始しますが、その例外処理においてはもう一方で保留されている例外を管理する必要があります。そのため、タスクに起因して発生する例外は、そのタスクを終了する前までに同期させなければなりません。

たとえば、浮動小数点例外 (FPI例外) と周辺装置保護例外 (PPI例外) が同時に発生した場合、例外優先順位によって、周辺装置保護違反が先に受け付けられ、もう一方の浮動小数点例外保留されます。周辺装置保護例外は、比較的深刻なエラーである場合があり、そのタスクを強制的に終了させる状況が想定されますが、その際、仮にそのままタスクの終了処理を行って周辺装置保護例外から復帰すると、別のタスクの実行を開始すると同時に、保留されていた浮動小数点例外が受け付けられてしまいます。しかし、この浮動小数点例外の起因となったタスクは、既に終了処理が行われ、OSの管理テーブルから削除されているので、別のタスクが例外を起こしたと誤認される懸念があります。

このため、あるタスクに関わる例外処理によって、そのタスクの終了処理を行う場合、例外同期処理を行ったあと、そのタスクに関わる保留されている他の例外を取り下げる必要があります。

図6-8 例外の起因となったタスクを終了させる場合

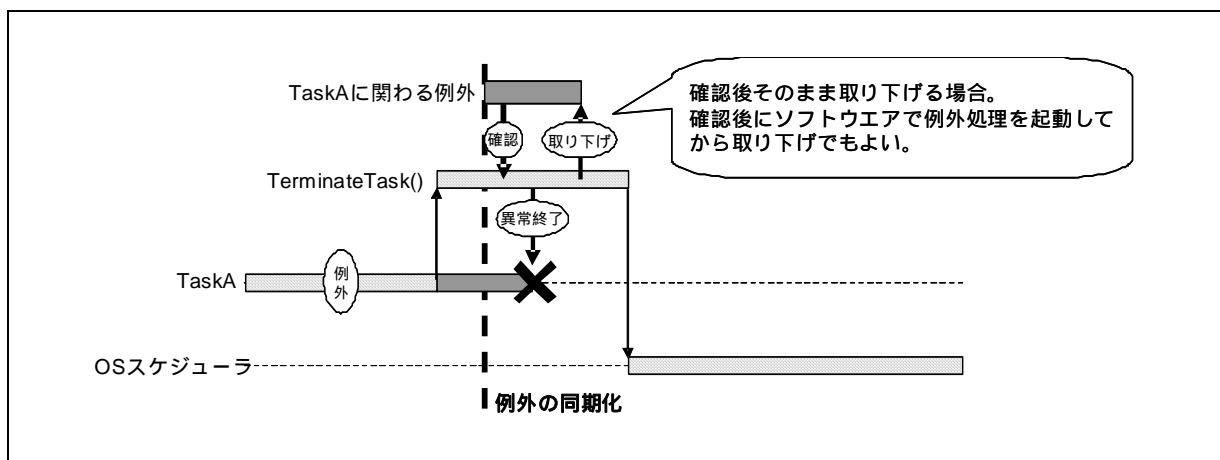


## (2) タスク終了間際に例外が発生した場合

あるタスクにおいて、タスクの終了処理の直前にインプレサイス例外や非同期例外が発生した場合、その例外が受け付けられる前に終了処理が開始され、その結果、本来例外が発生し異常を引き起こしたと判断されるべきタスクが、正常に終了したと誤認される懸念があります。

このため、タスクの終了処理を行う場合、例外同期処理を行い、例外処理を先に行うか、あるいは保留されている他の例外を取り下げる必要があります。

図6-9 タスク終了間際に例外が発生した場合





### 6.3.1 例外受け付け / 復帰時の例外同期

EIレベル例外 / FEレベル例外を受け付けるとき、およびEIレベル例外 / FEレベル例外から復帰するとき、CPUでは次のインプレサイス例外、非同期例外を待ち合わせて、受け付け可能なすべての例外の中から、優先度に従って例外を受け付けます。

- ・周辺装置保護例外（PPI例外）
- ・浮動小数点例外（FPI例外）
- ・ランタイム監視モードによるタイミング監視例外（TSI例外）

これらの例外は、特定のタスクに結び付いているため、CPUはタスク切り替えの契機となるEIレベル例外 / FEレベル例外受け付け時、およびEIレベル例外 / FEレベル例外からの復帰時に例外を待ち合わせ、同期を行います。CPUによる例外同期により、ソフトウェアは容易に例外を管理することができます。

### 6.3.2 例外同期命令

SYNCE命令により同期される例外は、周辺装置保護例外と浮動小数点演算例外です。これらのインプレサイス例外の受け付けを任意の時点で行いたい場合は、次の手順に従ってください。

- (1) 受け付けを行うインプレサイス例外の受け付け条件を満たすようなマスクを設定します（PSW.ID, NPのクリアなど）。
- (2) 例外同期化命令（SYNCE）を実行します。この時点で、SYNCE命令以前の命令によって発生するすべてのインプレサイス例外は、かならずCPUに通知が完了した状態になります。ただし、例外の受け付けは（1）で設定した受け付け条件によってマスクされ、保留される場合があります。
- (3) (2)の結果、マスクを行わなかった例外があった場合、例外が受け付けられます。このとき、複数の例外要因があった場合は、優先順位に従って順番に受け付けられます。

### 6.3.3 保留中例外の確認と取り下げ

保留されている例外があるかどうか確認したい場合は、次の手順に従ってください。

- (1) 確認を行うインプレサイス例外の受け付け条件を満たさないように、マスクを設定します（PSW.ID, NPのセットなど）。
- (2) 例外同期化命令（SYNCE）を実行します。この時点で、SYNCE命令以前の命令によって発生するすべてのインプレサイス例外は、かならずCPUに通知が完了した状態になります。確認を行う例外は（1）で設定したマスクによって、受け付けが行われず保留されます。但し、その他の例外が受け付けられる場合があります。
- (3) 確認を行う例外の例外通知ビットを読み出します。1であれば例外が保留されています。
- (4) (1)で設定したマスクを必要に応じて解除します。

保留中の例外を受け付けず、例外処理を行わずに取り下げたい場合は、次の手順に従ってください。

- (1) 取り下げを行うインプレサイス例外の受け付け条件を満たさないように、マスクを設定します(PSW.ID, NPのセットなど)。
- (2) 例外同期化命令(SYNCE)を実行します。この時点で、SYNCE命令以前の命令によって発生するすべてのインプレサイス例外は、かならずCPUに通知が完了した状態になります。取り下げを行う例外は(1)で設定したマスクによって、受け付けが行われず保留されます。ただし、その他の例外が受け付けられる場合があります。
- (3) 取り下げを行う例外の例外通知ビットをクリアします。
- (4) 取り下げが完了するまでの待ち合わせ処理を行ってください。待ち合わせ処理の命令シーケンスは製品仕様として定義されます。各製品のマニュアルを参照してください。
- (5) 取り下げが完了したら、(1)で設定したマスクを必要に応じて解除します。

各例外の取り下げ機能は、次のレジスタの機能によって提供されます。

例 外	原 因	取り下げ機能ビット	備 考
FPI例外	FPU命令	FPUバンク FPECレジスタ FPIVDビット	クリアすると後続FPU命令無効化も解除されます。
PPI例外	メモリ・アクセスをともなう命令	周辺I/O領域 PPECレジスタ PPVDビット	クリアすると後続アクセス命令無効化も解除されます。
TSI例外	タイミング監視	周辺I/O領域 TSUCFGnレジスタ (n = 0-5) VSビット	

## 6.4 例外ハンドラ・アドレス切り替え機能

V850E2M CPUでは、例外ハンドラ・アドレス切り替え機能を用いることで、例外ハンドラ・アドレスを変更することが可能です。各例外発生時に処理を移す例外ハンドラ・アドレスは、その時点での例外ハンドラ切り替え機能の設定値により決定されます。

例外ハンドラ・アドレス切り替え機能は、システム・レジスタ・バンク上に2つのバンクがあります。詳細は2.

### 4 CPU機能バンク / 例外ハンドラ・アドレス切り替え機能バンクを参照してください。

例外ハンドラ・アドレスは、次の3種類に分けられます。

- ・ CPU初期化 (RESET)
- ・ EIレベル・マスカブル割り込み (INT0-INT255)
- ・ 上記以外の各種例外

### 6.4.1 例外ハンドラ・アドレスの決定

現在の例外ハンドラ・アドレスは、例外ハンドラ切り替え機能バンク1 (ESWH1) に配置されたレジスタによって示されます。

#### (1) CPU初期化 (RESET) 時の開始アドレス

EH\_RESETレジスタによって示されます。

#### (2) EIレベル・マスカブル割り込み (INT0-INT255)

EH\_BASEレジスタと、EH\_CFGレジスタのRINTビットによって示されます。このレジスタの設定値は、実行中にソフトウェアによって変更することが可能です。

RINTビットがクリア (0) されている場合、INT0-INT255の例外ハンドラ・アドレスは、EH\_BASEレジスタにそれぞれのオフセット・アドレスを加えた256個の異なる例外ハンドラ・アドレスを使用します。

また、RINTビットがセット (1) されている場合、INT0-INT255の例外ハンドラ・アドレスは縮小され、EH\_BASEに0080Hを加えた一つの例外ハンドラ・アドレスを使用します。

INT0-INT255の例外ハンドラ・アドレスのために4096バイトのアドレス範囲が予約された状態から、RINTビットをセット (1) することで16バイトに縮小することが可能です。

**注意** 縮小した例外ハンドラ・アドレスを用いる場合でも、例外要因コードによってINT0-INT255までの例外要因の区別は可能です。

#### (3) 上記以外の各種例外の例外ハンドラ・アドレス

EH\_BASEレジスタによって示されます。EH\_BASEレジスタの示すアドレスに、各例外のオフセット・アドレスを加えたアドレスが、その例外の例外ハンドラ・アドレスとなります。このレジスタの設定値は、実行中にソフトウェアによって変更することが可能です。

## 6.4.2 例外ハンドラ・アドレスの切り替えの目的

例外ハンドラ・アドレスの切り替えは、起動後、ソフトウェアによって設定します。

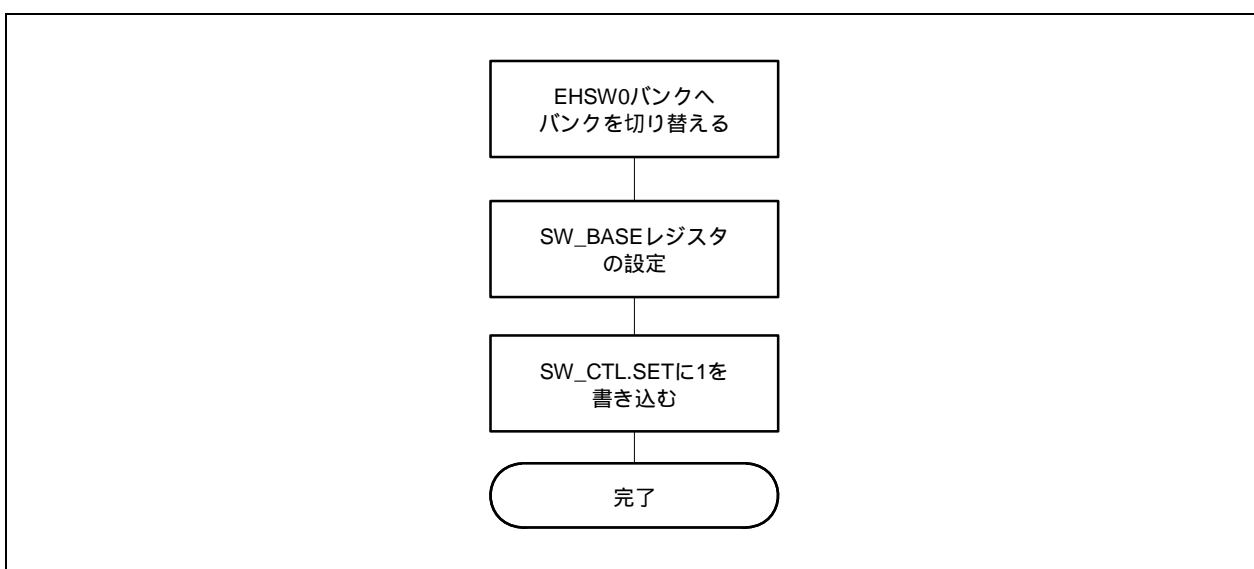
### (1) ソフトウェアによる切り替え

システム起動後に、何らかの理由（フラッシュ・メモリの書き換えなど）によって、例外ハンドラ・アドレス付近の命令の一貫性が保てない場合に、一時的に他の一貫性のとれた領域を例外ハンドラ・アドレスとして利用したい場合に利用します。

## 6.4.3 例外ハンドラ・アドレス切り替え機能の設定方法

### (1) ソフトウェアによる切り替え

CPUの動作中に下記のような手順によって、例外ハンドラ・アドレスを変更することができます。



例外ハンドラ・アドレスを切り替える場合は、切り替え手順の開始から完了までの間、例外が発生しない、または発生しても問題がないように考慮してください（例：例外を禁止する。システムの例外が発生しないように制御を行う。切り替え前後のいずれの例外ハンドラ・アドレスにも正しく動作をするプログラムを配置するなど）。

**注意** CPU 初期化 (RESET) による開始アドレスは、ソフトウェアによって変更することはできません。

## 第7章 コプロセッサ使用不可状態

V850E2M CPU は、特定の応用に限定された機能はコプロセッサとして定義しています。

V850E2M CPU は、コプロセッサとして浮動小数点演算機能（FPU）を搭載しています。

### 7.1 コプロセッサ使用不可例外

コプロセッサ命令と定義されたオペコードに対して実行を行おうとした際、次の場合にコプロセッサ使用不可例外（UCPOP）が発生します。

- ・コプロセッサ機能が未定義の場合
- ・コプロセッサ機能が製品に搭載されていない場合
- ・コプロセッサ機能が製品の機能によって、使用不可とされている場合

また、コプロセッサ使用不可例外は、コプロセッサ機能ごとに例外要因コードが割り当てられています。コプロセッサ機能と例外要因の対応関係は次の表で示されます。

コプロセッサ機能	発生する例外	例外要因コード
単精度FPU拡張機能	UCPOP0	530H
倍精度FPU拡張機能	UCPOP1	531H
未定義	UCPOP0-UCPOP7 <sup>注</sup>	530H-537H <sup>注</sup>

**注** 未定義のオペコードに対して、いずれの例外が発生するかについては、製品仕様で定義されます。

詳細は、各製品のマニュアルを参照してください。

### 7.2 システム・レジスタ

コプロセッサ機能によっては、その機能の一部としてシステム・レジスタが定義されるものがあります。次の場合において、該当するコプロセッサ機能のシステム・レジスタの動作はアーキテクチャ上不定です。

- ・コプロセッサ機能が製品に搭載されていない場合
- ・コプロセッサ機能が製品の機能によって、使用不可とされている場合

## 第8章 リセット

### 8.1 リセット後のレジスタの状態

製品仕様によって定義されたリセット入力方法によって、リセットが指示された場合、プログラム・レジスタとシステム・レジスタは、表8-1に示す状態になり、プログラムの実行を開始します。各レジスタの内容は、プログラムの中で必要に応じて適切な値に初期化を行ってください。

表8-1 リセット後のレジスタの状態

レジスタ		リセット後の状態（初期値）
プログラム・レジスタ	汎用レジスタ（r0）	00000000H（固定）
	汎用レジスタ（r1-r31）	不定
	プログラム・カウンタ（PC）	00000000H
システム・レジスタ	EIPC - EIレベル例外受け付け時の状態退避レジスタ	不定
	EIPSW - EIレベル例外受け付け時の状態退避レジスタ	00000020H
	FEPC - FEレベル例外受け付け時の状態退避レジスタ	不定
	FEPSW - FEレベル例外受け付け時の状態退避レジスタ	00000020H
	ECR - 例外要因	00000000H
	PSW - プログラム・ステータス・ワード	00000020H
	SCCFG - SYSCALLの動作設定	不定
	SCBP - SYSCALLベース・ポインタ	不定
	EIIC - EIレベル例外要因	00000000H
	FEIC - FEレベル例外要因	00000000H
	DBIC <sup>注</sup> - DBレベル例外要因	00000000H
	CTPC - CALLT実行時の状態退避レジスタ	不定
	CTPSW - CALLT実行時の状態退避レジスタ	00000020H
	CTBP - CALLTベース・ポインタ	不定
	EIWR - EIレベル例外作業レジスタ	
	FEWR - FEレベル例外作業レジスタ	
	BSEL - レジスタ・バンクの選択	00000000H

注 DBICレジスタは開発ルール向けのデバッグ機能で使します。

### 8.2 起 動

CPU はリセットにより、「例外ハンドラ・アドレス切り替え機能」によって定められたリセット・アドレスからプログラムの実行を開始します。

なお、リセット直後は、INT 例外は受け付けられません。INT 例外を使用する場合は、PSW.ID ビットをクリア（0）してください。

## 第 3 編 プロセッサ保護機能

# 第1章 概 説

V850E2M CPUではV850E2v3アーキテクチャに準拠し、信頼済みでないプログラムや暴走などによるシステム・リソースの不正使用、CPU実行時間の不当な占有などを検出/抑止し、システムの一貫性を維持するためのプロセッサ保護機能を提供しています。

**注意** プロセッサ保護機能は、各製品の実装に依存します。

## 1.1 特 徴

### (1) リソース・アクセス制御

V850E2M CPUは次の4種類のリソースに対するアクセス制御機能を提供します。

#### ・システム・レジスタ保護 (System Register Protection)

信頼済みでないプログラムによるシステム・レジスタ破壊を防ぐことができます。

#### ・メモリ保護 (Memory Protection)

アドレス空間上に命令/定数保護領域を最大5個、データ保護領域を最大6個まで配置可能です。これによって、ユーザ・プログラムに許可されていない実行、またはデータ操作を検出し、不正な実行、データ操作を防ぐことができます。各領域は上限アドレス/下限アドレスで指定するため、アドレス空間を効率よく細かい粒度で使用できます。

#### ・周辺装置保護 (Peripheral Device Protection)

周辺装置へのアクセスに対し、システムごとに固有に定義された不正アクセスを検出し、防ぐことができます。

#### ・タイミング監視 (Timing Supervision)

信頼済みでないプログラムの不当なCPU時間占有を防ぐことや、資源管理、割り込み禁止の時間の管理を行えます。

### (2) 実行レベルによる管理

V850E2M CPUでは、リソースへのアクセス制御を行うための状態ビットを複数持っており、これらの状態ビットの組み合わせを実行レベルとして定義しています。

ユーザは状況に応じた実行レベルを選択するほか、例外発生時、例外からの復帰時、いくつかの特殊命令の実行により自動的に変化する実行レベルの自動遷移機能を用いることで、状況に応じたアクセス制御を行うことが可能です。



**(3) 選択可能でスケーラブルな仕様**

実行レベルの自動遷移機能を利用するためには、OS（およびそれに準ずるプログラム）、共用ライブラリやユーザ・タスクがそれぞれ一定のプログラム・モデルに従う必要があります。

V850E2M CPUでは、実行レベルの自動遷移機能を選択しない場合にも、プロセッサ保護を利用できるスケーラブルな仕様をとっています。このため既存のソフトウェア資産に対しても容易にプロセッサ保護を導入することが可能です。また、従来どおり、プロセッサ保護機能のない状態で動作させることも可能です。

## 第2章 レジスタ・セット

### 2.1 システム・レジスタ・バンク

プロセッサ保護機能に関わるシステム・レジスタ・バンクを、表2-1に示します。

プロセッサ保護設定バンク、プロセッサ保護違反バンクおよびソフトウェア・ページング・バンクは、各LDSR命令でシステム・レジスタ (BSEL) に00001001H, 00001000Hおよび00001010Hを設定することにより選択されます。

システム・レジスタ番号28-31はバンク共通のシステム・レジスタで、BSELレジスタの設定値に関係なく、CPU機能バンクのEIWR, FEWR, DBWR<sup>注</sup>, BSELレジスタが参照されます。

**注** DBWR レジスタは、開発ツール向けのデバッグ機能で使います。

- ・ プロセッサ保護違反バンク  
(グループ番号10H, バンク番号00H, 略称MPV/PROT00バンク, プロセッサ保護違反レジスタを格納)
- ・ プロセッサ保護設定バンク  
(グループ番号10H, バンク番号01H, 略称MPU/PROT01バンク, プロセッサ保護設定レジスタを格納)
- ・ ソフトウェア・ページング・バンク  
(グループ番号10H, バンク番号10H, 略称PROT10バンク, プロセッサ保護設定 / 違反レジスタを格納)

また、次のCPU機能バンクのシステム・レジスタがプロセッサ保護機能に関わるレジスタとして使われます。

- ・ PSWレジスタ

表2-1 システム・レジスタ・バンク

グループ	プロセッサ保護機能 (10H)				
バンク	プロセッサ保護違反 (00H)		プロセッサ保護設定 (01H)		ソフトウェア・ページング (10H)
バンク・ラベル	MPV, PROT00		MPU, PROT01		PROT10
レジスタ番号	名称	機能	名称	機能	名称
0	VSECR	システム・レジスタ保護違反要因	MPM	プロセッサ保護動作モードの設定	MPM
1	VSTID	システム・レジスタ保護違反タスク識別子	MPC	プロセッサ保護コマンドの指定	MPC
2	VSADR	システム・レジスタ保護違反アドレス	TID	タスク識別子	TID
3	機能拡張用に予約		機能拡張用に予約		VMECR
4	VMECR	メモリ保護違反要因			VMTID
5	VMTID	メモリ保護違反タスク識別子			VMADR
6	VMADR	メモリ保護違反アドレス	IPA0L	命令/定数保護領域0下限アドレス	IPA0L
7	機能拡張用に予約		IPA0U	命令/定数保護領域0上限アドレス	IPA0U
8			IPA1L	命令/定数保護領域1下限アドレス	IPA1L
9			IPA1U	命令/定数保護領域1上限アドレス	IPA1U
10			IPA2L	命令/定数保護領域2下限アドレス	IPA2L
11			IPA2U	命令/定数保護領域2上限アドレス	IPA2U
12			IPA3L	命令/定数保護領域3下限アドレス	IPA3L
13			IPA3U	命令/定数保護領域3上限アドレス	IPA3U
14			IPA4L	命令/定数保護領域4下限アドレス	IPA4L
15			IPA4U	命令/定数保護領域4上限アドレス	IPA4U
16			DPA0L	データ保護領域0下限アドレス (スタック用)	DPA0L
17			DPA0U	データ保護領域0上限アドレス (スタック用)	DPA0U
18			DPA1L	データ保護領域1下限アドレス	DPA1L
19			DPA1U	データ保護領域1上限アドレス	DPA1U
20			DPA2L	データ保護領域2下限アドレス	DPA2L
21			DPA2U	データ保護領域2上限アドレス	DPA2U
22			DPA3L	データ保護領域3下限アドレス	DPA3L
23			DPA3U	データ保護領域3上限アドレス	DPA3U
24	MCA	メモリ保護設定チェック・アドレス	DPA4L	データ保護領域4下限アドレス	DPA4L
25	MCS	メモリ保護設定チェック・サイズ	DPA4U	データ保護領域4上限アドレス	DPA4U
26	MCC	メモリ保護設定チェック・コマンド	DPA5L	データ保護領域5下限アドレス (2 <sup>nd</sup> 指定のみ)	DPA5L
27	MCR	メモリ保護設定チェック結果	DPA5U	データ保護領域5上限アドレス (2 <sup>nd</sup> 指定のみ)	DPA5U
28	EIWR	EIレベル例外用作業レジスタ			
29	FEWR	FEレベル例外用作業レジスタ			
30	DBWR <sup>注</sup>	DBレベル例外用作業レジスタ			
31	BSEL	レジスタ・バンクの選択			

注 DBWR レジスタは、開発ツール向けのデバッグ機能で使います。

## 2.2 システム・レジスタ

### (1) プロセッサ保護設定レジスタ

プロセッサ保護設定レジスタは、プロセッサ保護モードの選択、保護対象の指定などを行います。システム・レジスタへのリード/ライトは、LDSR命令、STSR命令により、次に示すシステム・レジスタ番号を指定することで行います。

表2 - 2にプロセッサ保護設定レジスタ一覧を示します。

表2 - 2 プロセッサ保護設定レジスタ一覧

システム・レジスタ番号	名称	機能	オペランド指定の可否		システム・レジスタ保護 <sup>※</sup>
			LDSR	STSR	
0	MPM	プロセッサ保護動作モードの設定			
1	MPC	プロセッサ保護コマンドの指定			
2	TID	タスク識別子			
3-5	(将来の機能拡張のための予約番号(アクセスした場合の動作は保証しません))		×	×	
6	IPA0L	命令/定数保護領域0下限アドレス			
7	IPA0U	命令/定数保護領域0上限アドレス			
8	IPA1L	命令/定数保護領域1下限アドレス			
9	IPA1U	命令/定数保護領域1上限アドレス			
10	IPA2L	命令/定数保護領域2下限アドレス			
11	IPA2U	命令/定数保護領域2上限アドレス			
12	IPA3L	命令/定数保護領域3下限アドレス			
13	IPA3U	命令/定数保護領域3上限アドレス			
14	IPA4L	命令/定数保護領域4下限アドレス			
15	IPA4U	命令/定数保護領域4上限アドレス			
16	DPA0L	データ保護領域0下限アドレス			
17	DPA0U	データ保護領域0上限アドレス			
18	DPA1L	データ保護領域1下限アドレス			
19	DPA1U	データ保護領域1上限アドレス			
20	DPA2L	データ保護領域2下限アドレス			
21	DPA2U	データ保護領域2上限アドレス			
22	DPA3L	データ保護領域3下限アドレス			
23	DPA3U	データ保護領域3上限アドレス			
24	DPA4L	データ保護領域4下限アドレス			
25	DPA4U	データ保護領域4上限アドレス			
26	DPA5L	データ保護領域5下限アドレス			
27	DPA5U	データ保護領域5上限アドレス			

注 第5章 システム・レジスタ保護を参照してください。

備考 : オペランド指定の可否の欄では指定可能であることを示します。システム・レジスタ保護の欄では、保護対象であることを示します。

× : オペランド指定の可否の欄では指定不可能であることを示します。システム・レジスタ保護の欄では、保護対象ではないことを示します。

**(2) プロセッサ保護違反レジスタ**

プロセッサ保護違反レジスタ（メモリ保護違反通知レジスタ）は、違反要因、違反タスク識別子、違反アドレスの通知などを行います。システム・レジスタへのリード/ライトは、LDSR命令、STSR命令により、次に示すシステム・レジスタ番号を指定することで行います。

表2-3にプロセッサ保護違反レジスタ一覧を示します。

表2-3 プロセッサ保護違反レジスタ一覧

システム・レジスタ番号	名称	機能	オペランド指定の可否		システム・レジスタ保護 <sup>※</sup>
			LDSR	STSR	
0	VSECR	システム・レジスタ保護違反要因			
1	VSTID	システム・レジスタ保護違反タスク識別子			
2	VSADR	システム・レジスタ保護違反アドレス			
3	(将来の機能拡張のための予約番号(アクセスした場合の動作は保証しません))		×	×	
4	VMECR	メモリ保護違反要因			
5	VMTID	メモリ保護違反タスク識別子			
6	VMADR	メモリ保護違反アドレス			
7-23	(将来の機能拡張のための予約番号(アクセスした場合の動作は保証しません))		×	×	
24	MCA	メモリ保護設定チェック・アドレス			
25	MCS	メモリ保護設定チェック・サイズ			
26	MCC	メモリ保護設定チェック・コマンド			
27	MCR	メモリ保護設定チェック結果			

**注 第5章 システム・レジスタ保護を参照してください。**

**備考** : オペランド指定の可否の欄では指定可能であることを示します。システム・レジスタ保護の欄では、保護対象であることを示します。

× : オペランド指定の可否の欄では指定不可能であることを示します。システム・レジスタ保護の欄では、保護対象ではないことを示します。

**(3) ソフトウェア・ページング・レジスタ**

ソフトウェア・ページング・レジスタは、ソフトウェアによるメモリ保護のページング運用を実現するためのレジスタ群です。ソフトウェア・ページング・レジスタは、プロセッサ保護設定レジスタとプロセッサ保護違反レジスタにあるレジスタの写像になっています。

V850E2M CPUにおいて、メモリ保護の運用はタスクごとに固定的なメモリ保護領域設定を行うことが原則です。しかし、非常に大規模なソフトウェア・システムにおいてメモリ保護領域の数が足りない場合に備えて、プログラムがメモリ・アクセスを行う際に、そのメモリ・アクセス要求の際にプロセッサ保護例外によって起動された例外プログラムによって、保護設定を順次切り替える運用を想定しています。この運用方式をソフトウェア・ページングと呼び、またこの運用に最適なシステム・レジスタで構成されたバンクをソフトウェア・ページング・バンクとして定義しています。

このバンクを利用することで、プロセッサ保護例外処理中でのバンクの切替えを一度行うだけに抑えることができ、ソフトウェア・ページング時に必要な汎用レジスタ数を減らし、コンテキストの退避/復帰にかかる実行サイクルを低減することで、ソフトウェア・オーバヘッドを減少させます。

表2-4にソフトウェア・ページング・レジスタ一覧を示します。システム・レジスタへのリード/ライトは、LDSR命令、STSR命令により、次に示すシステム・レジスタ番号を指定することで行います。

表2-4 ソフトウェア・ページング・レジスタ一覧

システム・レジスタ番号	名称	機能	オペランド指定の可否		システム・レジスタ保護 <sup>※</sup>
			LDSR	STSR	
0	MPM	プロセッサ保護動作モードの設定			
1	MPC	プロセッサ保護コマンドの指定			
2	TID	タスク識別子			
3	VMECR	メモリ保護違反要因			
4	VMTID	メモリ保護違反タスク識別子			
5	VMADR	メモリ保護違反アドレス			
6	IPA0L	命令 / 定数保護領域0下限アドレス			
7	IPA0U	命令 / 定数保護領域0上限アドレス			
8	IPA1L	命令 / 定数保護領域1下限アドレス			
9	IPA1U	命令 / 定数保護領域1上限アドレス			
10	IPA2L	命令 / 定数保護領域2下限アドレス			
11	IPA2U	命令 / 定数保護領域2上限アドレス			
12	IPA3L	命令 / 定数保護領域3下限アドレス			
13	IPA3U	命令 / 定数保護領域3上限アドレス			
14	IPA4L	命令 / 定数保護領域4下限アドレス			
15	IPA4U	命令 / 定数保護領域4上限アドレス			
16	DPA0L	データ保護領域0下限アドレス			
17	DPA0U	データ保護領域0上限アドレス			
18	DPA1L	データ保護領域1下限アドレス			
19	DPA1U	データ保護領域1上限アドレス			
20	DPA2L	データ保護領域2下限アドレス			
21	DPA2U	データ保護領域2上限アドレス			
22	DPA3L	データ保護領域3下限アドレス			
23	DPA3U	データ保護領域3上限アドレス			
24	DPA4L	データ保護領域4下限アドレス			
25	DPA4U	データ保護領域4上限アドレス			
26	DPA5L	データ保護領域5下限アドレス			
27	DPA5U	データ保護領域5上限アドレス			

注 第5章 システム・レジスタ保護を参照してください。

備考 : オペランド指定の可否の欄では指定可能であることを示します。システム・レジスタ保護の欄では、保護対象であることを示します。

x : オペランド指定の可否の欄では指定不可能であることを示します。システム・レジスタ保護の欄では、保護対象ではないことを示します。







ビット位置	ビット名	意味
0	MPE	<p>プロセッサ保護機能の動作の有効 / 無効を設定します (初期値 : 0)。</p> <p>0 : プロセッサ保護機能無効</p> <ul style="list-style-type: none"> <li>PSW.PPビットを0に固定します。 周辺装置保護機能が限定的なアクセス制限のみを行い、特殊周辺装置のみ違反を検出します。また、PPI例外は発生しません<sup>※</sup>。</li> <li>PSW.NPVビットを0に固定します。 システム・レジスタ保護機能が無効となり、すべてのシステム・レジスタへのアクセスを許可します。</li> <li>PSW.DMP, IMPビットを0に固定します。 メモリ保護機能が無効となり、すべてのメモリ・アクセスを許可します。また、MIP例外、MDP例外は発生しません。</li> <li>TSCCFGn.TSCCFGnACTビット (n = 0-5) を0に固定します。 タイミング監視機能が無効となります。また、TSI例外は発生しません<sup>※</sup>。</li> </ul> <p>1 : プロセッサ保護機能有効</p> <ul style="list-style-type: none"> <li>PSW.PPビットの更新を許可します。 周辺装置保護が厳密なアクセス制限を行い、設定に従って違反を検出します。また、PPI例外が発生する可能性があります。</li> <li>PSW.NPVビットの更新を許可します。 システム・レジスタ保護機能が有効となり、設定に従って違反を検出します。</li> <li>PSW.DMP, IMPビットの更新を許可します。 メモリ保護機能が有効となり、設定に従って違反を検出します。また、MIP例外、MDP例外が発生する可能性があります。</li> <li>TSCCFGn.TSCCFGnACTビット (n = 0-5) の更新を許可します。 タイミング監視機能が有効となり、設定に従って動作します。また、TSI例外が発生する可能性があります。</li> </ul>

**注** ただし、MPEビットを0にする以前に検出されたPPI例外、TSI例外が保留されている可能性があります。この場合、MPEビットが0であっても、PSW.NPビットが0になった時点で、PPI例外、TSI例外が受け付けられる場合があります。

**備考** PP, NPV, IMP, DMPビットの詳細は2. 2. 1 PSW - プログラム・ステータス・ワードを参照してください。



## 第3章 動作設定

プロセッサ保護機能を使用する場合は、まずプロセッサ保護全体に関わる動作設定を行う必要があります。システム・レジスタ・バンクをプロセッサ保護設定バンク（グループ番号10H，バンク番号01H）に切り替えて、MPMレジスタに適切な値を設定してください。

### 3.1 プロセッサ保護機能の利用開始

プロセッサ保護機能を有効にするには、まずMPM.MPEビットをセット（1）する必要があります。MPEビットがクリア（0）されている場合、PSW.PP, NPV, DMP, IMPビットおよび、TSCCFGn.TSCCFGnACTビット（n = 0-5）が0に固定され、プロセッサ保護機能の各機能が動作しません。MPM.MPEビットをセット（1）すると、次のようにプロセッサ保護機能の利用を開始します。

- ・ PSW.PP, NPV, DMP, IMPビットが更新可能になります。  
（システム・レジスタ保護機能，メモリ保護機能，周辺装置保護機能の利用が可能になります）。
- ・ タイミング監視機能の各カウンタの設定レジスタ（TSCCFGn）のTSCCFGnACTビットが更新可能になります  
ます  
（タイミング監視機能の利用が可能になります）。

### 3.2 実行レベル自動遷移機能の設定

MPM.AUEビットをセット（1）することで、実行レベルの自動遷移機能が有効になります。自動遷移を行うプログラム・モデルでシステムを運用する場合には、プロセッサ保護機能を利用する前に必ずAUEビットをセット（1）してください。

詳細は第4章 実行レベルを参照してください。

### 3.3 プロセッサ保護機能の利用停止

プロセッサ保護機能を一度有効にしたあとに、再び無効にする場合はMPM.MPEビットをクリア（0）してください。この操作を行うことで、PSW.PP, NPV, DMP, IMPビットおよび、TSCCFGn.TSCCFGnACTビット（n = 0-5）が0に固定され、次のようにプロセッサ保護機能の利用を停止します。

- ・ PSW.PP, NPV, DMP, IMPビットを0に固定します  
（システム・レジスタ保護機能，メモリ保護機能，周辺装置保護機能を利用できません）。
- ・ タイミング監視機能の各カウンタの設定レジスタ（TSCCFGn）のTSCCFGnACTビットを0に固定します  
（タイミング監視機能を利用できません）。

**注意** プロセッサ保護機能の利用を停止した時点で、MPE ビットを 0 にする以前に検出された一部のプロセッサ保護例外（PPI 例外および TSI 例外）が保留されている可能性があります。この場合、MPE を 0 に変更した後であっても、PSW.NP ビットが 0 になった時点で、これらの例外が受け付けられます。

## 第4章 実行レベル

V850E2M CPUでは、PSW（プログラム・ステータス・ワード）上の次の4つのビットの制御によって、現在実行中のプログラムの信頼状態を示し、プログラムのリソースに対するアクセスの権限を示します。これらのビットを保護ビットと呼び、またこれらのビットの特定の組み合わせを「実行レベル」と呼びます。

- ・PPビット：  
CPUが、現在実行中のプログラムによる周辺装置へのアクセスを信頼している状態であるかどうかを示します。
- ・NPVビット：  
CPUが、現在実行中のプログラムによるシステム・レジスタへのアクセスを信頼している状態であるかどうかを示します。
- ・DMPビット：  
CPUが、現在実行中のプログラムによるデータ・アクセスを信頼している状態であるかどうかを示します。
- ・IMPビット：  
CPUが、現在実行中のプログラムによるプログラム領域へのアクセスを信頼している状態であるかどうかを示します。

### 4.1 プログラムの性質

動作中のプログラムは、その設計品質に従って「信頼済みプログラム（Trusted Program）」と「信頼済みでないプログラム（Non-trusted Program）」のいずれかに属します。一般的に「信頼済みプログラム」とは、OSなどやデバイス・ドライバなど、システムを脅かす恐れがないことが保証されたプログラムを指します。「信頼済みでないプログラム」とは、開発段階のユーザ・プログラム、あるいはサード・パーティ製のプログラム等のシステムを脅かす恐れがないことがまだ確認されていないプログラムを指します。

システム・レジスタ、データ領域、プログラム領域、周辺装置という4つの保護対象の各々について信頼済みプログラムの動作中と、信頼済みでないプログラムの動作中をハードウェアが区別するための情報として、PSW.PP、NPV、DMP、IMPビットを定義しています。それぞれのビットは、そのビットが関連するリソースに対して、次の表のような意味を持ち、それぞれのプログラムの実行開始前に、OSなどによって適切な値を設定します。

保護ビットの状態	状態名	プログラムの品質
0	Tステート	信頼済み（Trusted Program）
1	NTステート	信頼済みでない（Non-trusted Program）

## 4.2 PSW上の保護ビット

現在実行中のプログラムのプロセッサ保護の対象リソースに関する信頼状態を示す保護ビット(PP, NPV, DMP, IMPビット)を, PSW上に配置しています。PSW上のビット19, 18, 17, 16をそれぞれPPビット, NPVビット, DMPビット, IMPビットとして定義しており, プロセッサ保護機能を利用する場合は, これらのビットを適切に設定してください。

**注意** PSW.PP, NPV, DMP, IMP ビットは, MPM.MPE ビットが0の場合は0に固定されます。また, システム・レジスタ保護の対象となっているため, NPV ビットが1の場合にも書き込みが行えません。

### 4.2.1 Tステート (信頼状態)

現在実行中のプログラムが各ビットに対応するリソースへの操作に対して十分に信頼でき, 不正な操作をしない場合には, 0を設定してください。各ビットが0に設定された状態を, そのビットに対応するリソースへの操作に対して「信頼済み状態」とし, 「Tステート」と呼びます。

通常, プログラムがTステートである場合は, 違反を検出せず, 特権的な動作を行います。

**注意 1.** 周辺装置保護では, Tステートであっても特殊周辺装置に対する操作のみ, 違反を検出します。

**2.** タイミング監視機能に関しては, 信頼状態という概念はありません。

### 4.2.2 NTステート (非信頼状態)

現在実行中のプログラムが各ビットに対応するリソースへの操作に対して信頼できず, 不正な操作を行う可能性がある場合には, 1を設定してください。各ビットが1に設定された状態を, そのビットに対応するリソースへの操作に対して「信頼済みでない状態」とし, 「NTステート」と呼びます。

プログラムがNTステートである場合は, 設定に従って違反を検出し, 場合によっては例外が発生します。

## 4.3 実行レベルの定義

V850E2M CPUでは, PSW.PP, NPV, DMP, IMPビットの状態の組み合わせにおいて, 標準的に利用される一部の組み合わせを実行レベルとして定義し運用することを想定しています。また, これらの組み合わせ以外での使用も可能ですが, 推奨するものではありません。

実行レベルとその用途の例を表4-1に示します。

表4-1 実行レベル

実行レベル	PPビット (周辺装置保護)	NPVビット (システム・レジスタ保護)	DMPビット (メモリ保護 データ領域)	IMPビット (メモリ保護 プログラム領域)	実行レベルの用途例
0	0	0	0	0	例外ハンドラ, OSカーネルなど
1	1	0	0	0	デバイス・ドライバなど
2	1	0	1	1	共有ライブラリなど
3	1	1	1	1	ユーザ・タスク

## 4.4 実行レベルの遷移

V850E2M CPUにおいて実行レベルの遷移を行う方法には、大きく分けて次の3種類があります。

- ・システム・レジスタへの書き込み命令の実行
- ・例外の発生
- ・復帰命令の実行

MPM.MPEビットがクリア(0)されている場合は、上記のいずれの場合においても、PSW.PP, NPV, DMP, IMPビットは0に固定され、実行レベルは0以外のレベルへ遷移しません。

**注意** V850E2M CPU では CALLT 命令の実行により実行レベルが遷移することはありません。CALLT 命令によって起動される共用サブルーチン等は呼び出し元と同じプロセッサ保護状態で動作します。

### 4.4.1 システム・レジスタへの書き込み命令の実行による遷移

システム・レジスタへの書き込み命令(LDSR命令)の実行により、PSW.PP, NPV, DMP, IMPビットを書き換えることが可能です。これによってユーザは任意の実行レベルに遷移することが可能になります。書き換えられた実行レベルは、次命令実行時より有効になります。

- 注意 1.** PSW.NPV ビットがセット(1)されている場合は、システム・レジスタ保護により PSW.PP, NPV, DMP, IMP ビットは変更できません。このとき、実行レベルは遷移しません(第5章 システム・レジスタ保護参照)。
2. MPM.AUE ビットがクリア(0)されている場合は、PSW.NPV ビットは0に固定され、変更できません。
  3. LDSR 命令によって PSW.IMP ビットの設定を変更した場合、設定が反映されるまでに数命令かかる場合があります。このようなときは、EIRET 命令または FERET 命令の実行によって分岐を行うことで、設定を確実に反映できます。

#### 4.4.2 例外の発生による遷移

MPM.AUEビットをセット(1)することで、実行レベル自動遷移機能が有効になります。この状態でいずれかの例外が発生した場合、PSW.PP, NPV, DMP, IMPビットが自動的にクリア(0)され、実行レベル0に遷移します。

**注意** DB レベルへ遷移する例外および、メモリ保護例外(MDP, MIP, PPI, TSI)の発生時は、AUE ビットの設定にかかわらず、PSW.PP, NPV, DMP, IMP ビットが自動的にクリア(0)されます。

#### 4.4.3 復帰命令の実行による遷移

例外やCALLT命令からの復帰命令(RETI, EIRET, FERET, CTRET)実行時に、それぞれの命令に対応する復帰PSW(EIPSW, FEPSW, CTPSW)から、PSWへ値がコピーされます。このとき、MPM.MPEビットがセット(1)されている場合は、復帰PSWの値を格納したレジスタのPP, NPV, DMP, IMPビットに対応するビットの値が、PSW上のPP, NPV, DMP, IMPビットにコピーされます。MPEビットがクリア(0)されている場合は変化しません。また、MPM.AUEビットがクリア(0)されている場合は、NPVビットは、0に固定されます。

例外発生時に例外レベルごとに、例外発生前のPSWの値が保存されるため、例外処理中に復帰PSWの値を格納するレジスタの値を変更しなかった場合、例外復帰命令実行時にはPP, NPV, DMP, IMPビットは例外発生前の状態に復元されます。このため、例外によって中断されたプログラムからみて、例外処理前と例外処理後において実行レベルが変化することはありません。

**注意** 1. 復帰命令による PSW.PP, NPV, DMP, IMP ビットの更新は、実行レベル自動遷移機能が無効の場合にも行われます。

2. CALLT 命令により呼び出されたサブルーチンからの CTRET 命令による復帰も、この実行レベルの遷移を行いますが、CTPSW には CALLT 命令実行時の PSW の値が保存されており、また CTPSW はシステム・レジスタ保護の対象となっているため、ユーザが不正に CTPSW 上の保護ビットに対応したビットを変更し、T ステートに遷移してしまうことはありません。

### 4.5 プログラム・モデル

実行レベル自動遷移機能を用いることで、2つの異なる実行レベル管理ポリシーを持ったプログラム・モデルを選択することが可能です。ユーザはそれぞれのシステムに適合したプログラム・モデルを選択してください。

#### ・実行レベルを自動的に遷移するプログラム・モデル

実行レベル自動遷移機能を有効に設定し、例外発生により実行レベルの遷移を行います。OSなどや階層管理されたプログラムで使用する場合に適しています。

#### ・常に一定の実行レベルで動作するプログラム・モデル

実行レベル自動遷移機能を無効に設定し、例外受け付けによる実行レベルの遷移を行いません。実行レベルの遷移は、ユーザのプログラムによって特定命令の実行によって行われるのみとなり、既存ソフトウェア資産に対して容易にプロセッサ保護を適用できます。

## 4.6 タスク識別子

V850E2M CPUでは、複数の異なるプログラムの集合を実行する場合、現在実行中のプログラムが、どの集合に属しているかを判断するためのレジスタを備えています。このプログラムの集合をタスクと呼び、タスクごとの識別子をタスク識別子と定義して、TIDレジスタに設定できます。

プロセッサ保護機能では例外の発生時に、このタスク識別子を違反情報のひとつとして使用します。



## 第5章 システム・レジスタ保護

V850E2M CPUでは、信頼済みでない (Non-trusted) プログラムによる、システム設定の不当な変更からシステムを保護するために、特定のシステム・レジスタに対するアクセス制御を行うことが可能です。

システム・レジスタ保護違反が発生した場合には、違反情報がMPU違反バンクのシステム・レジスタに保存されます。

## 5.1 レジスタ・セット

システム・レジスタ保護機能に関わるシステム・レジスタを、表5-1に示します。

表5-1 システム・レジスタ・バンク

グループ	プロセッサ保護機能 (10H)				
バンク	プロセッサ保護違反 (00H)		プロセッサ保護設定 (01H)		ソフトウェア・ページング (10H)
バンク・ラベル	MPV, PROT00		MPU, PROT01		PROT10
レジスタ番号	名称	機能	名称	機能	名称
0	VSECR	システム・レジスタ保護違反要因	MPM	プロセッサ保護動作モードの設定	MPM
1	VSTID	システム・レジスタ保護違反タスク識別子	MPC	プロセッサ保護コマンドの指定	MPC
2	VSADR	システム・レジスタ保護違反アドレス	TID	タスク識別子	TID
3	機能拡張用に予約		機能拡張用に予約		VMECR
4	VMECR	メモリ保護違反要因			VMTID
5	VMTID	メモリ保護違反タスク識別子			VMADR
6	VMADR	メモリ保護違反アドレス	IPA0L	命令定数保護領域0下限アドレス	IPA0L
7	機能拡張用に予約		IPA0U	命令定数保護領域0上限アドレス	IPA0U
8			IPA1L	命令定数保護領域1下限アドレス	IPA1L
9			IPA1U	命令定数保護領域1上限アドレス	IPA1U
10			IPA2L	命令定数保護領域2下限アドレス	IPA2L
11			IPA2U	命令定数保護領域2上限アドレス	IPA2U
12			IPA3L	命令定数保護領域3下限アドレス	IPA3L
13			IPA3U	命令定数保護領域3上限アドレス	IPA3U
14			IPA4L	命令定数保護領域4下限アドレス	IPA4L
15			IPA4U	命令定数保護領域4上限アドレス	IPA4U
16			DPA0L	データ保護領域0下限アドレス (スタック用)	DPA0L
17			DPA0U	データ保護領域0上限アドレス (スタック用)	DPA0U
18			DPA1L	データ保護領域1下限アドレス	DPA1L
19			DPA1U	データ保護領域1上限アドレス	DPA1U
20			DPA2L	データ保護領域2下限アドレス	DPA2L
21			DPA2U	データ保護領域2上限アドレス	DPA2U
22			DPA3L	データ保護領域3下限アドレス	DPA3L
23			DPA3U	データ保護領域3上限アドレス	DPA3U
24	MCA	メモリ保護設定チェック・アドレス	DPA4L	データ保護領域4下限アドレス	DPA4L
25	MCS	メモリ保護設定チェック・サイズ	DPA4U	データ保護領域4上限アドレス	DPA4U
26	MCC	メモリ保護設定チェック・コマンド	DPA5L	データ保護領域5下限アドレス (2 <sup>nd</sup> 指定のみ)	DPA5L
27	MCR	メモリ保護設定チェック結果	DPA5U	データ保護領域5上限アドレス (2 <sup>nd</sup> 指定のみ)	DPA5U

### 5.1.1 VSECR - システム・レジスタ保護違反要因

システム・レジスタ保護によって違反が検出された回数を示します。

ビット31-8には必ず0を設定してください。

31	VSECR																												8 7	0	初期値 00000000H
0 0																												VSEC			
ビット位置	ビット名	意味																													
7-0	VSEC	システム・レジスタ保護違反を検出した回数を保存します。 VSECビットの値が255に達したとき、以降は増加せず、255のまま変化しません。違反処理を行った場合には、VSECRレジスタをプログラムによってクリアしてください。																													

### 5.1.2 VSTID - システム・レジスタ保護違反タスク識別子

システム・レジスタ保護によって違反を検出された、最初の命令の実行時点のタスク識別子 (TID) の内容を保存します。

31	VSTID																												0	初期値 不定
VSTID																														
ビット位置	ビット名	意味																												
31-0	VSTID	VSECRレジスタのVSECビットが0の状態、システム・レジスタ保護違反を検出した場合に、その時点のTIDレジスタ (プロセッサ保護設定バンク) の値を格納します。 VSTIDレジスタは違反処理後、最初にシステム・レジスタ保護違反を起こしたタスクの識別子を保存します。																												

### 5.1.3 VSADR - システム・レジスタ保護違反アドレス

システム・レジスタ保護によって違反を検出された、最初の命令のPCを保存します。

ビット0は、0に固定されています。

31	VSADR																												0	初期値 不定
VSADR																														
ビット位置	ビット名	意味																												
31-0	VSADR	VSECRレジスタのVSECビットが0の状態、システム・レジスタ保護違反を検出した場合に、その命令のPCを格納します。 VSADRレジスタは違反処理後、最初にシステム・レジスタ保護違反を起こした命令のPCを保存します。																												

**注意** 一部の命令アドレッシング範囲が 512 M バイトに制限された CPU では、VSADR レジスタのビット 31-29 はビット 28 を符号拡張した値が自動的に設定されます。

## 5.2 アクセス制御

PSW.NPVビットにより、システム・レジスタの書き込みに関するアクセス制御が適用されます<sup>※</sup>。

PSW.NPVビットがクリア(0)されている場合(Tステート)、LDSR命令で指定可能なすべてのシステム・レジスタに書き込み可能です。一方、NPVビットがセット(1)されている場合(NTステート)、LDSR命令による保護対象となる特定のシステム・レジスタ、およびシステム・レジスタ中の特定のビットへの書き込み操作を阻止し、レジスタ値に反映させません。

この書き込み制御により、信頼済みでない(Non-trusted)プログラムによる設定変更を防ぐことが可能です。

**注** LDSR命令による書き込みのみが、アクセス制御の対象となります。EI命令、DI命令、復帰命令(EIRET, FERET, RETI)、その他のシステム・レジスタ更新動作は、保護の対象外です。

**注意 1.** 実行レベル自動遷移機能を無効(AUE=0)に設定している場合、NPVビットは0に固定されます。このため、システム・レジスタ保護機能によって、書き込み操作が阻止されることはありません。

**2.** PSW.NPVビット自身も、システム・レジスタ保護の対象となることに注意してください。このため、一度PSW.NPVビットをセット(1)すると、いずれかの例外を発生させない限り、クリア(0)することはできません。

## 5.3 対象レジスタ

次のシステム・レジスタ一覧におけるシステム・レジスタ保護の項目を参照してください。

- ・基本バンク
  - 第2編 表2-2 システム・レジスタ一覧(基本バンク)
- ・例外ハンドラ切り替え機能0, 1
  - 第2編 表2-3 システム・レジスタ・バンク
- ・ユーザ0バンク
  - 第2編 表2-4 システム・レジスタ一覧
- ・プロセッサ保護設定バンク
  - 第3編 表2-2 プロセッサ保護設定レジスタ一覧
- ・プロセッサ保護違反バンク
  - 第3編 表2-3 プロセッサ保護違反レジスタ一覧
- ・ソフトウェア・ページング・バンク
  - 第3編 表2-4 ソフトウェア・ページング・レジスタ一覧
- ・FPUステータス・バンク
  - 第4編 表2-1 システム・レジスタ・バンク

**注意** 機能が定義されていないシステム・レジスタ番号はすべてシステム・レジスタ保護の対象とします。

## 5.4 違反の検出

現在実行中のプログラムが信頼済みでない(Non-trusted)プログラムである場合、PSW.NPVビットをセット(1)し、NT状態で動作させてください。CPUがNT状態で動作している間に、システム・レジスタ保護の対象となるレジスタへLDSRによる書き込み操作を行った場合、ただちにシステム・レジスタ保護違反を検出します。

システム・レジスタ保護違反が検出された場合、次の動作を行います。

- ・そのLDSR命令による書き込み操作を阻止します(レジスタ値に反映されません)。
- ・VSECRレジスタの値が0であった場合、次の動作を行います。
  - ・そのLDSR命令の実行時点のTIDレジスタの値を、VSTIDレジスタに格納します。
  - ・そのLDSR命令のPCを、VSADRレジスタに格納します。
- ・VSECRレジスタの値を1加算します。

**注意** PSW レジスタは、同一レジスタ内に保護を行うビットと、保護を行わないビットが混在しているため、書き込みが起きても違反を検出しません。ただし、保護を行うビットに対する書き込み操作は阻止を行い、ビットの値に反映しません。

## 5.5 運用方法

OSなどによるタスク切り替え操作ごと、あるいは任意の一定周期ごとにVSECR、VSTID、VSADRレジスタで、システム・レジスタ保護違反の検出状況を確認し、適切な処理を行ってください。このとき、システム・レジスタ保護違反を一回以上検出していた場合は、前回の確認から、今回の確認までの間にシステム・レジスタに不正なアクセスをした可能性があります。

また、通常処理に復帰する前に必ずVSECRレジスタをクリアしてください。

## 第6章 メモリ保護

V850E2M CPUは、命令実行の際に参照するプログラム領域（命令アクセス）、およびメモリ・アクセスを行う命令の実行によって参照するデータ領域（データ・アクセス）のアドレス空間上の2つの領域に対して、信頼済みでない（Non-trusted）プログラムによる不正なアクセスからシステムを保護するアクセス制御を行うことが可能です。

V850E2M CPUのメモリ保護では、メモリ保護領域を上限・下限アドレスで指定します。指定可能な領域粒度は16バイト単位です。このため、少ない領域数で適切な保護設定ができます。指定されたアドレスは、32ビットのシステム・レジスタで保持され、4 Gバイトの論理アドレス空間を完全にカバーします。

### 6.1 レジスタ・セット

メモリ保護機能に関わるシステム・レジスタを、表6 - 1に示します。

表6-1 システム・レジスタ・バンク

グループ	プロセッサ保護機能 (10H)				
バンク	プロセッサ保護違反 (00H)		プロセッサ保護設定 (01H)		ソフトウェア・ページング (10H)
バンク・ラベル	MPV, PROT00		MPU, PROT01		PROT10
レジスタ番号	名称	機能	名称	機能	名称
0	VSECR	システム・レジスタ保護違反要因	MPM	プロセッサ保護動作モードの設定	MPM
1	VSTID	システム・レジスタ保護違反タスク識別子	MPC	プロセッサ保護コマンドの指定	MPC
2	VSADR	システム・レジスタ保護違反アドレス	TID	タスク識別子	TID
3	機能拡張用に予約		機能拡張用に予約		VMECR
4	VMECR	メモリ保護違反要因			VMTID
5	VMTID	メモリ保護違反タスク識別子			VMADR
6	VMADR	メモリ保護違反アドレス	IPA0L	命令/定数保護領域0下限アドレス	IPA0L
7	機能拡張用に予約		IPA0U	命令/定数保護領域0上限アドレス	IPA0U
8			IPA1L	命令/定数保護領域1下限アドレス	IPA1L
9			IPA1U	命令/定数保護領域1上限アドレス	IPA1U
10			IPA2L	命令/定数保護領域2下限アドレス	IPA2L
11			IPA2U	命令/定数保護領域2上限アドレス	IPA2U
12			IPA3L	命令/定数保護領域3下限アドレス	IPA3L
13			IPA3U	命令/定数保護領域3上限アドレス	IPA3U
14			IPA4L	命令/定数保護領域4下限アドレス	IPA4L
15			IPA4U	命令/定数保護領域4上限アドレス	IPA4U
16			DPA0L	データ保護領域0下限アドレス (スタック用)	DPA0L
17			DPA0U	データ保護領域0上限アドレス (スタック用)	DPA0U
18			DPA1L	データ保護領域1下限アドレス	DPA1L
19			DPA1U	データ保護領域1上限アドレス	DPA1U
20			DPA2L	データ保護領域2下限アドレス	DPA2L
21			DPA2U	データ保護領域2上限アドレス	DPA2U
22			DPA3L	データ保護領域3下限アドレス	DPA3L
23			DPA3U	データ保護領域3上限アドレス	DPA3U
24	MCA	メモリ保護設定チェック・アドレス	DPA4L	データ保護領域4下限アドレス	DPA4L
25	MCS	メモリ保護設定チェック・サイズ	DPA4U	データ保護領域4上限アドレス	DPA4U
26	MCC	メモリ保護設定チェック・コマンド	DPA5L	データ保護領域5下限アドレス (2 <sup>nd</sup> 指定のみ)	DPA5L
27	MCR	メモリ保護設定チェック結果	DPA5U	データ保護領域5上限アドレス (2 <sup>nd</sup> 指定のみ)	DPA5U

### 6.1.1 IPAnL - 命令 / 定数保護領域n下限アドレス (n = 0-4)

命令 / 定数保護領域の下限アドレスと動作を設定するためのレジスタです。

ビット3には必ず0を設定してください。

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
IPAnL	AL	AL	AL	AL	AL	AL	AL	AL	AL	AL	AL	AL	AL	AL	AL	AL	
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	AL	AL	AL	AL	AL	AL	AL	AL	AL	AL	AL	AL	0	T	S	E	初期値 0000002H
	15	14	13	12	11	10	9	8	7	6	5	4					

ビット位置	ビット名	意味
31-4	AL31- AL4	保護領域指定方式 (Tビット) の設定によって、保護領域の下限アドレス、または保護領域のベース・アドレスを設定します。  また、IPAnLレジスタのビット3-0は保護領域の他の設定で利用されているため、下限アドレス / ベース・アドレスのビット3-0 (AL3-0) は暗黙的に0を使用します。
2	T	保護領域の範囲を指定する方法を選択します。  Tビットがクリア (0) されている場合は、上限 / 下限指定方式が選択されます。セット (1) されている場合は、マスク / ベース指定方式が選択されます。  0 : 上限 / 下限指定方式 (AU31-0を上限アドレス, AL31-0レジスタを下限アドレスとします。)  1 : ベース / マスク指定方式 (AU31-0をマスク, AL31-0レジスタをベース・アドレスとします。)
1	S <sup>注</sup>	保護領域に対するsp (r3) レジスタ相対によるデータ・アクセスの許可 / 禁止を設定します。  Sビットがクリア (0) されている場合、データ領域上の保護領域に置かれたデータに対するsp (r3) レジスタ相対によるデータ・アクセスは禁止されます。  禁止された領域に対するsp (r3) レジスタ相対によるデータ・アクセスを行う命令を実行しようとした場合、データ保護違反を検出し、MDP例外が直ちに受け付けられます。  0 : 保護領域に対するsp (r3) レジスタ相対によるデータ・アクセスの禁止 1 : 保護領域に対するsp (r3) レジスタ相対によるデータ・アクセスの許可
0	E	保護領域の設定の有効 / 無効を設定します。  Eビットがクリア (0) されている場合、その他の設定ビットの内容はすべて無効となり、保護領域は設定されません。  0 : 無効 (命令 / 定数保護領域nを使用しません) 1 : 有効 (命令 / 定数保護領域nを使用します)

注 Sビットは、MPM.SPSビットの設定値によって0または1に固定されます。詳細は2.2.2 MPM - プロセッサ保護動作モードの設定を参照してください。

備考 n = 0-4



## 6.1.2 IPAnU - 命令 / 定数保護領域n上限アドレス (n = 0-4)

命令 / 定数保護領域の上限アドレスを設定するためのレジスタです。

ビット3, 2には必ず0を設定してください。

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
IPAnU	AU31	AU30	AU29	AU28	AU27	AU26	AU25	AU24	AU23	AU22	AU21	AU20	AU19	AU18	AU17	AU16	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	AU15	AU14	AU13	AU12	AU11	AU10	AU9	AU8	AU7	AU6	AU5	AU4	0	0	R	X	初期値 00000000H

ビット位置	ビット名	意味
31-4	AU31- AU4	保護領域指定方式 (Tビット) の設定によって、保護領域の上限アドレス、または保護領域のマスク値を設定します。 また、IPAnUレジスタのビット3-0は保護領域の他の設定で利用されているため、上限アドレス / マスク値のビット3-0(AU3-0)は暗黙的に1を使用します。 マスク値を指定する場合は、必ず下位側から1を連続させた値を設定してください (000050FFHなどのように、1/0が交互に配置された場合の動作は保証しません)。
1	R	保護領域に対するリード・アクセスの許可 / 禁止を設定します。 Rビットがクリア (0) されている場合、データ領域上の保護領域に置かれたデータに対するリード・アクセスは禁止されます。 禁止された領域に対するリード・アクセスを行う命令を実行しようとした場合、データ保護違反を検出し、MDP例外が直ちに受け付けられます。 0 : 保護領域に対するリード・アクセスの禁止 1 : 保護領域に対するリード・アクセスの許可
0	X	保護領域に対する命令実行の許可 / 禁止を設定します。 Xビットがクリア (0) されている場合、プログラム上の保護領域に置かれたプログラムの実行は禁止されます。 禁止された領域の命令実行を行おうとした場合、命令保護違反を検出し、MIP例外が直ちに受け付けられます。 0 : 保護領域に対する命令実行の禁止 1 : 保護領域に対する命令実行の許可

備考 n = 0-4

### 6.1.3 DPAnL - データ保護領域n下限アドレス (n = 0-5)

データ保護領域の下限アドレスと動作を設定するためのレジスタです。

ビット3には必ず0を設定してください。

DPAnL	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	AL 31	AL 30	AL 29	AL 28	AL 27	AL 26	AL 25	AL 24	AL 23	AL 22	AL 21	AL 20	AL 19	AL 18	AL 17	AL 16
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	AL 15	AL 14	AL 13	AL 12	AL 11	AL 10	AL 9	AL 8	AL 7	AL 6	AL 5	AL 4	0	T	S	E

初期値  
注1

ビット位置	ビット名	意 味
31-4	AL31- AL4	保護領域指定方式 (Tビット) の設定によって、保護領域の下限アドレス、または保護領域のベース・アドレスを設定します。  また、DPAnLレジスタのビット3-0は保護領域の他の設定で利用されているため、下限アドレス/ベース・アドレスのビット3-0 (AL3-AL0) は暗黙的に0を使用します。
2	T	保護領域の範囲を指定する方法を選択します。  Tビットがクリア (0) されている場合は、上限/下限指定方式が選択されます。セット (1) されている場合は、マスク/ベース指定方式が選択されます。  0: 上限/下限指定方式 (AU31-AU0を上限アドレス, AL31-AL0を下限アドレスとします。)  1: ベース/マスク指定方式 (AU31-AU0をマスク, AL31-AL0をベース・アドレスとします。)
1	S <sup>注2</sup>	保護領域に対するsp (r3) レジスタ相対によるデータ・アクセスの許可/禁止を設定します。  Sビットがクリア (0) されている場合、データ領域上の保護領域に置かれたデータに対するsp (r3) レジスタ相対によるデータ・アクセスは禁止されます。  禁止された領域に対するsp (r3) レジスタ相対によるデータ・アクセスを行う命令を実行しようとした場合、データ保護違反を検出し、MDP例外が直ちに受け付けられます。  0: 保護領域に対するsp (r3) レジスタ相対によるデータ・アクセスの禁止 1: 保護領域に対するsp (r3) レジスタ相対によるデータ・アクセスの許可
0	E	保護領域の設定の有効/無効を設定します。  DPEビットがクリア (0) されている場合、その他の設定ビットの内容はすべて無効となり、保護領域は設定されません。  0: 無効 (データ保護領域nを使用しません) 1: 有効 (データ保護領域nを使用します)

注1. 初期値はチャンネルによって異なります。

DPA0L-DPA4L : 0000 0002H, DPA5L : 0000 0006H

2. Sビットは、MPM.SPSビットの設定値によって0または1に固定されます。

詳細は2.2.2 MPM - プロセッサ保護動作モードの設定を参照してください。

備考 n = 0-5

### 6.1.4 DPAnU - データ保護領域n上限アドレス (n = 0-5)

データ保護領域の上限アドレスを設定するレジスタです。

ビット3, 0には必ず0を設定してください。

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DPAnU	AU	AU	AU	AU	AU	AU	AU	AU	AU	AU	AU	AU	AU	AU	AU	AU
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	AU	AU	AU	AU	AU	AU	AU	AU	AU	AU	AU	AU	0	W	R	0
	15	14	13	12	11	10	9	8	7	6	5	4				

初期値  
注

ビット位置	ビット名	意 味
31-4	AU31- AU4	保護領域指定方式 (Tビット) の設定によって、保護領域の上限アドレス、または保護領域のマスク値を設定します。  また、DPAnUレジスタのビット3-0は保護領域の他の設定で利用されているため、上限アドレス/マスク値のビット3-0 (AU3-AU0) は暗黙的に1を使用します。  マスク値を指定する場合は、必ず下位側から1を連続させた値を設定してください (000050FFHなどのように、1/0が交互に配置された場合の動作は保証しません)。
2	W	保護領域に対するライト・アクセスの許可/禁止を設定します。  Wビットがクリア (0) されている場合、データ領域上の保護領域に置かれたデータに対するライト・アクセスは禁止されます。  禁止された領域に対するライト・アクセスを行う命令を実行しようとした場合、データ保護違反を検出し、MDP例外が直ちに受け付けられます。  0 : 保護領域に対するライト・アクセスの禁止 1 : 保護領域に対するライト・アクセスの許可
1	R	保護領域に対するリード・アクセスの許可/禁止を設定します。  Rビットがクリア (0) されている場合、データ領域上の保護領域に置かれたデータに対するリード・アクセスは禁止されます。  禁止された領域に対するリード・アクセスを行う命令を実行しようとした場合、データ保護違反を検出し、MDP例外が直ちに受け付けられます。  0 : 保護領域に対するリード・アクセスの禁止 1 : 保護領域に対するリード・アクセスの許可

注 初期値はチャンネルによって異なります。

DPA0U : 0000 0006H , DPA1U-DPA5U : 0000 0000H

備考 n = 0-5

### 6.1.5 VMECR - メモリ保護違反要因

VMECRレジスタはMIP例外、MDP例外発生時の保護違反要因を示します。

ビット31-7, 0には必ず0を設定してください。

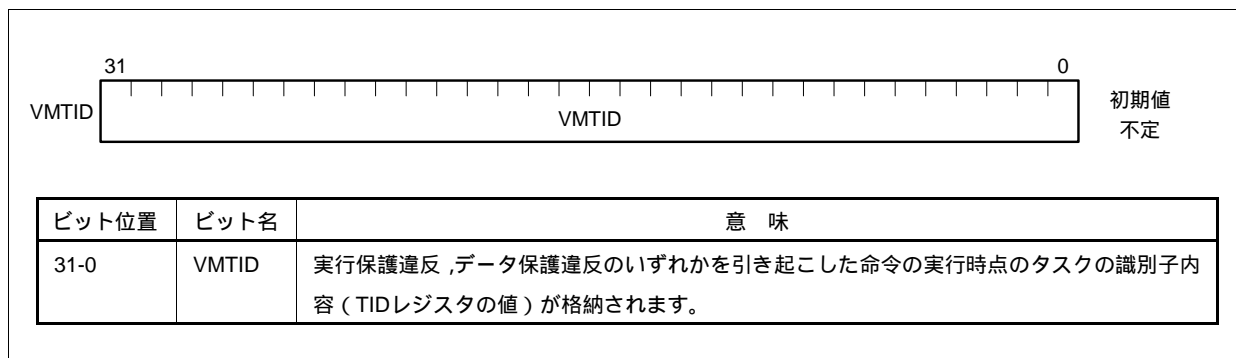
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
VMECR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	0	0	0	0	0	0	0	0	0	VMMS	VM RMW	VMS	VMW	VMR	VMX	0	初期値 00000000H

ビット位置	ビット名	意味
6	VMMS	データ保護例外の発生状況を示します。LD, ST, SLD, SST命令によるミスアライン・アクセス時にデータ保護例外が発生した場合、セット(1)されます。それ以外の場合はクリア(0)されます。
5	VMRMW	データ保護例外の発生状況を示します。SET1, CLR1, NOT1, CAXI命令によるアクセス時にデータ保護例外が発生した場合、セット(1)されます。それ以外の場合はクリア(0)されます。
4	VMS	sp相対アクセス違反によるMDP例外が発生した場合、セット(1)されます。それ以外の場合はクリア(0)されます。 このビットがセット(1)された場合、VMXビットは常にクリア(0)の状態となります。VMR, VMWビットはこのビットと同時にセット(1)される可能性があります。
3	VMW	ライト・アクセス違反によるMDP例外が発生した場合、セット(1)されます。それ以外の場合はクリア(0)されます。 このビットがセット(1)された場合、VMX, VMRビットは常にクリア(0)の状態となります。VMSビットは、このビットと同時にセット(1)される可能性があります。
2	VMR	リード・アクセス違反によるMDP例外が発生した場合、セット(1)されます。それ以外の場合はクリア(0)されます。 このビットがセット(1)された場合、VMX, VMWビットは常にクリア(0)の状態となります。VMSビットはこのビットと同時にセット(1)される可能性があります。
1	VMX	命令実行違反によるMIP例外が発生した場合、セット(1)されます。それ以外の場合はクリア(0)されます。 このビットがセット(1)された場合、VMR, VMW, VMSビットは常にクリア(0)の状態となります。

**備考** 例外発生時の各ビットの状態については、9.3 違反要因の特定を参照してください。

### 6.1.6 VMTID - メモリ保護違反タスク識別子

VMTIDレジスタはMIP例外，MDP例外発生時のタスクの識別子を格納します。



### 6.1.7 VMADR - メモリ保護違反アドレス

VMADRレジスタはMIP例外，MDP例外発生時のアドレスを格納します。



**注意** 実行保護違反を検出した命令のPC値は, 命令が複数のアドレスにまたがって配置されている場合などに, 実際に違反を起こした命令アドレスと一致しない場合があります。

## 6.2 アクセス制御

PSW.IMPビットにより、命令実行の際に参照するプログラム領域に関するアクセス制御が適用されます。IMPビットがクリア(0)されている場合(Tステート)、すべてのプログラム領域に対する命令実行が許可され、どの位置にある命令も自由に実行が可能です。一方、IMPビットがセット(1)されている場合(NTステート)、すべてのプログラム領域に対する命令の実行は原則的に禁止され、保護領域として命令実行を許可された範囲の命令しか実行できません。

PSW.DMPビットにより、メモリ・アクセスを行う命令の実行によって参照するデータ領域に関するアクセス制御が適用されます。DMPビットがクリア(0)されている場合(Tステート)、すべてのデータ領域に対するメモリ・アクセスが許可され、どの位置にあるデータも自由に読み出し、書き込み等が可能です。一方、DMPビットがセット(1)されている場合(NTステート)、すべてのデータ領域に対するメモリ・アクセスは原則的に禁止され、保護領域として許可された範囲のデータに対する操作しか行えません。また、操作内容に関しても書き込み、読み込み、スタック操作などを考慮したアクセス制御が行われます。

このアクセス制御により、信頼済みでない(Non-trusted)プログラムによる不正な命令実行や、不正なデータ・アクセスを防ぐことが可能です。

## 6.3 保護領域の設定

プログラム領域またはデータ領域上は、原則的にアクセスが禁止されます。メモリ保護を利用する場合、それぞれの信頼済みでない(Non-trusted)プログラムごとに、これらの領域上にアクセスを許可する保護領域を指定します。保護領域はアクセスの種類(実行/リード/ライト)ごとに許可/禁止等を選択可能です。

V850E2M CPUでは、これらの保護領域を設定するために次のような2つのレジスタを1組として、各保護領域を設定するレジスタを備えています。

- IPAnL/IPAnU (n = 0-4)
- DPAmL/DPAmU (m = 0-5)

各保護領域の設定は、上限レジスタと下限レジスタの2つの組み合わせで表現されます。V850E2M CPUは、最大で11個の保護領域を設定できるレジスタを定義しており、それによって、プログラム領域に5領域、データ領域に6領域の保護領域を配置することが可能です。

それぞれの保護領域に対して可能な設定を次の表6-2に示します。レジスタによって、一部のビットの値が固定されていることに注意してください。

表6-2 保護領域の設定

レジスタ		xPAnU					xPAnL				
		ビット31-4	ビット3	ビット2	ビット1	ビット0	ビット31-4	ビット3	ビット2	ビット1	ビット0
フィールド機能	上限アドレス (マスク値)	(RFU)	ライト許可	リード許可	実行許可	下限アドレス (ベース・アドレス)	(RFU)	領域指定方式	sp相対アクセス許可	領域イネーブル	
フィールド名称	AU		W	R	X	AL		T	S	E	
IPAnU/L	命令保護	上限アドレス	0	0	(0)	(0)	下限アドレス	0	0	0/1 <sup>注</sup>	(0)
IPAnL/L	領域設定	上限アドレス	0	0	(0)	(0)	下限アドレス	0	0	0/1 <sup>注</sup>	(0)
IPAn2U/L		上限アドレス	0	0	(0)	(0)	下限アドレス	0	0	0/1 <sup>注</sup>	(0)
IPAn3U/L		上限アドレス	0	0	(0)	(0)	下限アドレス	0	0	0/1 <sup>注</sup>	(0)
IPAn4U/L		上限アドレス	0	0	(0)	(0)	下限アドレス	0	0	0/1 <sup>注</sup>	(0)
DPA0U/L		データ保護 領域設定 (スタック用)	上限アドレス	0	1	1	0	下限アドレス	0	0	1
DPA1U/L	データ保護	上限アドレス	0	(0)	(0)	0	下限アドレス	0	0	0/1 <sup>注</sup>	(0)
DPA2U/L	領域設定	上限アドレス	0	(0)	(0)	0	下限アドレス	0	0	0/1 <sup>注</sup>	(0)
DPA3U/L		上限アドレス	0	(0)	(0)	0	下限アドレス	0	0	0/1 <sup>注</sup>	(0)
DPA4U/L		上限アドレス	0	(0)	(0)	0	下限アドレス	0	0	0/1 <sup>注</sup>	(0)
DPA5U/L		マスク値	0	(0)	(0)	0	ベース・アドレス	0	1	0/1 <sup>注</sup>	(0)

注 MPM.SPSビットが0の場合、S = 1になります。MPM.SPSビットが1の場合、S = 0となります。

備考 0 : 必ず0を設定してください。

1 : 必ず1を設定してください。

(0) : ユーザによる設定が可能なビットです。( )内は初期値を示します。

したがって、次のようにプログラム領域、データ領域と、設定レジスタが関連付けられます。

#### プログラム領域 (命令実行の許可 / 禁止)

- IPAnL/IPAnU (n = 0-4)

#### データ領域 (リードまたはライト実行の許可 / 禁止)

- IPAnL/IPAnU (n = 0-4) ... リード許可のみ
- DPA0L/DPA0U ... sp相対アクセスは常に許可, 常にリード許可, ライト許可
- DPAnL/DPAnU (n = 1-4)
- DPA5L/DPA5U ... マスク / ベース指定方式

IPAnL/IPAnUレジスタによって、命令実行許可かつ、リード・アクセス許可の保護領域が設定できます。SWITCH命令やCALLT命令によるテーブル・ジャンプで、命令コード中にテーブルが配置される場合、命令実行許可とリード・アクセス許可を設定する必要があります。

データ領域に対しては、DPA0L/DPA0Uレジスタのみ、常にsp相対アクセスが許可されています。実行中にプログラムが参照するスタックは単一で、sp相対アクセスを行う領域は1つだけであるためです。

また、各レジスタの初期値は次のようになっています。

レジスタ	初期値
IPA0L-IPA4L	0000 0002H
IPA0U-IPA4U	0000 0000H
DPA0L	0000 0002H
DPA0U	0000 0006H
DPA1L-DPA4L	0000 0002H
DPA1U-DPA4U	0000 0000H
DPA5L	0000 0006H
DPA5U	0000 0000H

次に各ビットの機能を示します。

### 6.3.1 有効ビット (Eビット)

保護領域の設定の有効 / 無効を示します。

Eビットがクリア (0) されている場合、その他の設定ビットの内容はすべて無効となり、保護領域は設定されません。

### 6.3.2 実行許可ビット (Xビット)

保護領域に対する命令実行の許可 / 禁止を示します。

Xビットがクリア (0) されている場合、プログラム領域上の保護領域に置かれたプログラムの実行は禁止されます。

禁止された領域の命令実行を行おうとした場合、**命令保護違反**を検出し、**MIP例外**が直ちに受け付けられます。

### 6.3.3 リード許可ビット (Rビット)

保護領域に対するリード・アクセスの許可 / 禁止を示します。

Rビットがクリア (0) されている場合、データ領域上の保護領域に置かれたデータに対するリード・アクセスは禁止されます。

禁止された領域に対するリード・アクセスを行う命令を実行しようとした場合、**データ保護違反**を検出し、**MDP例外**が直ちに受け付けられます。

### 6.3.4 ライト許可ビット (Wビット)

保護領域に対するライト・アクセスの許可 / 禁止を示します。

Wビットがクリア (0) されている場合、データ領域上の保護領域に置かれたデータに対するライト・アクセスは禁止されます。

禁止された領域に対するライト・アクセスを行う命令を実行しようとした場合、**データ保護違反**を検出し、**MDP例外**が直ちに受け付けられます。



### 6.3.5 sp相対アクセス許可ビット (Sビット)

保護領域に対するsp (r3) レジスタ相対によるデータ・アクセスの許可 / 禁止を示します。

Sビットがクリア(0)されている場合、データ・アドレス空間上の保護領域に置かれたデータに対するsp (r3) レジスタ相対によるデータ・アクセスは禁止されます。

禁止された領域に対するsp (r3) レジスタ相対によるデータ・アクセスを行う命令を実行しようとした場合、**データ保護違反**を検出し、**MDP例外**がただちに受け付けられます。

**注意** 各レジスタのSビットは、MPM.SPSビットの設定値によって0または1に固定されます。  
詳細は2.2.2 MPM - プロセッサ保護動作モードの設定を参照してください。

### 6.3.6 保護領域指定方式ビット (Tビット)

保護領域の範囲を指定する方法を選択します。

Tビットがクリア(0)されている場合は、**上限 / 下限指定方式**が選択されます。セット(1)されている場合は、**マスク / ベース指定方式**が選択されます。

#### (1) 上限 / 下限指定方式

AU31-AU0ビットを上限アドレス、AL31-AL0ビットを下限アドレスとして、保護領域の範囲を指定します。上限アドレスよりも大きな値を設定した場合、保護領域は無効となります。

#### (2) マスク / ベース指定方式

AU31-AU0ビットをマスク値、AL31-AL0ビットをベース・アドレスとして、保護領域の範囲を指定します。指定したベース・アドレスに対して、マスク値によってマスクしたアドレス範囲が、保護領域となります。

保護領域は、次のような上限アドレス、下限アドレスで表される2のべき乗サイズの領域となります。

下限アドレス = ( AL31-AL0 and ( not AU31-AU0 ) )

上限アドレス = ( AL31-AL0 or AU31-AU0 )

### 6.3.7 保護領域下限アドレス (AL31-AL0ビット)

保護領域指定方式 (Tビット) の設定によって、保護領域の下限アドレス、または保護領域のベース・アドレスを示します。

また、IPAnL, DPAmLレジスタのビット3-0は保護領域の他の設定で利用されているため、下限アドレス / ベース・アドレスのビット3-0 (AL3-AL0) は暗黙的に0を使用します (n = 0-4, m = 0-5)。

### 6.3.8 保護領域上限アドレス (AU31-AU0ビット)

保護領域指定方式 (Tビット) の設定によって、保護領域の上限アドレス、または保護領域のマスク値を示します。

また、IPAnU, DPAmUレジスタのビット3-0は保護領域の他の設定で利用されているため、上限アドレス / マスク値のビット3-0 (AU3-AU0) は暗黙的に1を使用します。

マスク値を指定する場合は、必ず下位側から1を連続させた値を設定してください (000050FFHなどのように、1/0が交互に配置された場合の動作は保証しません) (n = 0-4, m = 0-5)。

## 6.4 保護領域設定時の注意事項

### 6.4.1 保護領域境界の交差

命令/定数保護領域, データ保護領域のいずれにおいても, 保護領域の範囲が重複して設定されている場合は, 交差部分に対するアクセス制御の設定は, 許可優先となります。

#### (1) 命令/定数保護領域

あるアドレスにおいて, 複数の保護領域が設定されている場合は, いずれかの保護領域の設定で実行許可にされていれば, 許可として判断されます。また, いずれかの領域でリード許可と設定されていれば, リード許可として判断されます。

#### (2) データ保護領域

あるアドレスにおいて, 複数の保護領域が設定されている場合は, いずれかの保護領域でリード許可にされていれば, リード許可として判断されます。

ライト許可, sp相対アクセス許可の場合も同様です。

### 6.4.2 無効な保護領域の設定

次の場合に, 保護領域の設定は無効となります。

- ・ 下限アドレスを上限アドレスよりも大きな値に設定した場合

## 6.5 スタック検査機能

V850E2M CPUでは、スタック操作に関するプログラム・モデルとして、汎用レジスタr3をスタック・ポインタ(sp)として使用することを規定しています。また、暗黙的にspをスタック・ポインタとして利用するスタック・フレームの生成/削除命令を定義しています。このプログラム・モデルに対応し、かつより厳密なスタック管理を行うための機能として、スタック検査機能を備えています。

各保護領域設定レジスタのSビットによって、保護領域ごとにスタック相対アクセスの許可/禁止を設定することが可能です<sup>注</sup>。

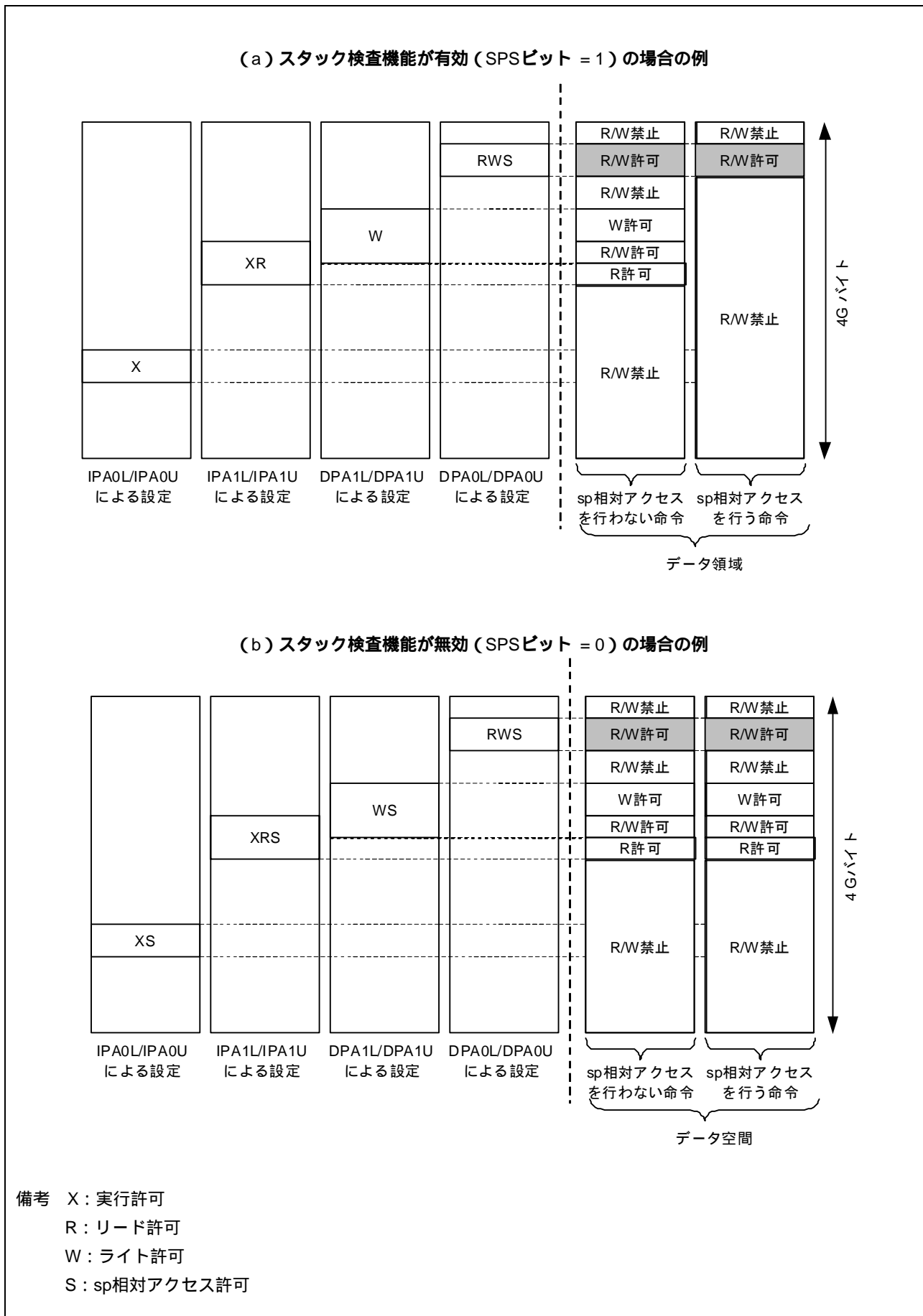
sp相対アクセスを禁止した保護領域(Sビットが0)に対して、spをベース・レジスタとした相対アクセスを行うメモリ・アクセス命令が実行された場合、そのアクセスを違反として検出します。同じ領域に対して、sp以外をベース・アドレスとした相対アクセスを行う場合は、リード許可ビット(Rビット)、ライト許可ビット(Wビット)に従ってアクセス制御を行います。

sp相対アクセスを許可した保護領域(Sビットが1)に対しては、いずれの場合でもリード許可ビット(Rビット)、ライト許可ビット(Wビット)に従ってアクセス制御を行います。

このスタック検査機能により、sp相対アクセスを許可する保護領域を1つだけ配置するプログラム・モデルにおいてspの値を誤った場合に、他の保護領域に対してはsp相対アクセス違反が検出されるため、より強固なメモリ保護が実現できます。

注 V850E2M CPUでは、各保護領域のSビットの値は、MPM.SPSレジスタの値によって一括制御します。

図6-1 スタック検査機能の例



## 6.6 特殊なメモリ・アクセス命令

V850E2M CPUでは、ひとつの命令実行中に複数回のメモリ・アクセスを行う命令があります。これらの命令に対するメモリ保護機能は特殊な動作を行います。特殊な保護動作の対象となる命令を次に示します。

- ・ ミスアライン・アクセスを行うロード/ストア命令 (LD, ST, SLD, SST)
- ・ 一部のビット操作命令 (SET1, NOT1, CLR1) とCAXI命令
- ・ スタック・フレーム操作命令 (PREPARE, DISPOSE)
- ・ SYSCALL命令

### 6.6.1 ミスアライン・アクセスを行うロード/ストア命令

V850E2M CPUは、データ形式(バイト/ハーフワード/ワード)にかかわらず、すべてのアドレスにデータの配置が可能です。ミスアライン・アクセスとは、処理対象のデータがハーフワード形式の場合は、ハーフワード境界(アドレスの最下位ビットが0)以外のアドレスへのアクセスを、処理対象のデータがワード形式の場合は、ワード境界(アドレスの下位2ビットが0)以外のアドレスへのアクセスを示します。

ミスアライン・アクセスの場合、アクセス対象のすべてのアドレスがいずれかの単一の保護領域内に収まっている、かつロード命令であればリード許可、ストア命令であればライト許可がされている場合、アクセスが許可されます。

**注意** 2つの保護領域が、重複せずに連続したアドレスに対して定義されている場合でも、2つの保護領域の境界をまたぐミスアライン・アクセスは保護違反と判定されます。

### 6.6.2 一部のビット操作命令とCAXI命令

一部のビット操作命令 (SET1, NOT1, CLR1) とCAXI命令は、アクセスを行うアドレスがリード許可かつライト許可でない場合、データ保護違反を検出します。

### 6.6.3 スタック・フレーム操作命令

スタック・フレーム操作命令 (PREPARE, DISPOSE) は、指定レジスタの数だけのメモリ・アクセスを発生させます。メモリ保護機能は、このそれぞれのアクセスに対して違反の検出を行い、違反を検出した時点でデータ保護例外の発生により、スタック・フレーム操作命令の実行を中断します。ただし、違反が検出されたメモリ・アクセス以前のメモリ・アクセスは実行され、またDISPOSE命令の場合は、実行されたメモリ・アクセスに対応する汎用レジスタへの書き込みは行われます。中断された場合は、spの更新は行われません。

また、例外からの復帰後に中断されたスタック・フレーム操作命令は、再び最初から実行されるため、中断前に一度実行したメモリ・アクセスと同じメモリ・アクセスが実行されます。

### 6.6.4 SYSCALL命令

SYSCALL命令は、OSなどの管理プログラムによって提供されるサービスの呼び出しに用いる命令です。サービスは信頼済みのプログラムであり、またサービスへ分岐するためのアドレス・テーブルもまた信頼済みであるため、SYSCALL命令によるメモリ・アクセスは、PSW.DMPビットがセット(1)されている状態であっても、メモリ保護が適用されません。

したがって、SYSCALL命令の実行時にMDP例外が検出されることはありません。

## 6.7 保護違反と例外

許可されていないアドレスに対して命令の実行，あるいはデータ・アクセスが行われた場合，それぞれ命令保護違反 / データ保護違反を検出します。違反を検出した場合，次のような動作を行います。

MIP例外，MDP例外に関する詳細は，**第9章 プロセッサ保護例外**を参照してください。

### (1) 命令保護違反の検出時

- ・ 命令保護違反を検出したアドレスに配置された命令は実行を開始しません。
- ・ 命令保護違反を検出したアドレスへのアクセスはCPUの外部に対して要求を行いません。
- ・ MIP例外を発生し，直ちに例外処理を開始します。

### (2) データ保護違反の検出時

- ・ データ保護違反を検出したアドレスへのアクセスを行う命令は実行を中断します。
- ・ データ保護違反を検出したアドレスへのアクセスはCPUの外部に対して要求を行いません。
- ・ MDP例外を発生し，ただちに例外処理を開始します。

## 第7章 周辺装置保護

周辺装置保護は、周辺装置（領域）を信頼済みでない（アントラステッドな）プログラムによる不正なアクセスから保護するためのアクセス制御を行う機能です。

V850E2M CPUは、応用システムごとに異なる周辺装置（I/O、小規模メモリなど）が接続されることを想定しています。応用システムごとに異なる周辺装置は、多くの場合メモリ保護が対象とする領域粒度に比べ、より細かく離散的なアドレスに配置されます。周辺装置保護では周辺装置ごとの保護設定が可能で、メモリ保護による限られた数の保護領域では実現できない細粒度の保護を実現できます。

### 7.1 レジスタ・セット

周辺装置保護機能に関わる周辺装置レジスタを、表7-1に示します。

V850E2M CPUでは、周辺装置保護機能に関わるレジスタ・セットは周辺装置レジスタ領域に配置します。周辺装置保護レジスタのベース・アドレスはハードウェア定義で、V850E2M CPUでは FFFF5100Hです。PPSn, PPPn, PPVn, PPTnレジスタは1セットで32個の周辺装置の保護を設定でき、1ビットが1つの周辺装置に対応します。製品の周辺装置の数、および周辺装置保護の方針によって、複数セット用意することができます。V850E2M CPUでは、合計9セットを備え、レジスタ名にサフィックス0-8を付けて指定します。

**注意** 周辺装置保護機能に関わるこれらのレジスタ自身も、周辺装置保護機能の対象となります。詳細は7.9 周辺装置保護設定レジスタに対する保護設定を参照してください。

表7-1 周辺装置保護機能レジスタ・セット

レジスタ名	オフセット・アドレス	アクセス可能サイズ				初期値
		1	8	16	32	
PPM	+00H	✓	✓	✓	✓	00000000H
PPEC	+04H		✓	✓	✓	00000000H
VPNECR	+10				✓	不定 <sup>注</sup>
VPNADR	+14				✓	不定 <sup>注</sup>
VPNTID	+18				✓	不定 <sup>注</sup>
VPTECR	+20				✓	不定 <sup>注</sup>
VPTADR	+24				✓	不定 <sup>注</sup>
VPTTID	+28				✓	不定 <sup>注</sup>
PPVn	+40H + (n*10h) + 0H	✓	✓	✓	✓	表7-2参照
PPTn	+40H + (n*10h) + 4H	✓	✓	✓	✓	表7-2参照
PPPn	+40H + (n*10h) + 8H	✓	✓	✓	✓	表7-2参照
PPSn	+40H + (n*10h) + CH	✓	✓	✓	✓	表7-2参照

注 リセットによる初期化はされず前置保持します。初期値は不定です。

備考 n = 0-8

V850E2M CPUでのPPSn, PPPn, PPVn, PPTnレジスタのレジスタ名, アドレス, 初期値, 保護対象アドレスを表7-2に示します。レジスタの各ビットと周辺装置の対応は、付録D 周辺装置保護領域一覧を参照してください。

表7-2 V850E2M CPUのPPSn, PPPn, PPVn, PPTnレジスタ

レジスタ名	アドレス	初期値	保護対象アドレス
PPV0	FFFF5140H	00030000H	CPU固有周辺装置 CPUシステム周辺装置
PPT0	FFFF5144H	00030000H	
PPP0	FFFF5148H	00030000H	
PPS0	FFFF514CH	00000000H	
PPV1	FFFF5150H	00000000H	FF400000-FF5FFFFFFH
PPT1	FFFF5154H	00000000H	
PPP1	FFFF5158H	00000000H	
PPS1	FFFF515CH	00000000H	
PPV2	FFFF5160H	00000000H	FF600000-FF7FFFFFFH
PPT2	FFFF5164H	00000000H	
PPP2	FFFF5168H	00000000H	
PPS2	FFFF516CH	00000000H	
PPV3	FFFF5170H	00000000H	FF800000-FF81FFFFFFH
PPT3	FFFF5174H	00000000H	
PPP3	FFFF5178H	00000000H	
PPS3	FFFF517CH	00000000H	
PPV4	FFFF5180H	00000000H	FF820000-FF83FFFFFFH
PPT4	FFFF5184H	00000000H	
PPP4	FFFF5188H	00000000H	
PPS4	FFFF518CH	00000000H	
PPV5	FFFF5190H	00000000H	FFFF8000-FFFF9FFFFH
PPT5	FFFF5194H	00000000H	
PPP5	FFFF5198H	00000000H	
PPS5	FFFF519CH	00000000H	
PPV6	FFFF51A0H	00000000H	FFFA000-FFFFBFFFFH
PPT6	FFFF51A4H	00000000H	
PPP6	FFFF51A8H	00000000H	
PPS6	FFFF51ACH	00000000H	
PPV7	FFFF51B0H	00000000H	FFFC000-FFFDFFFFH
PPT7	FFFF51B4H	00000000H	
PPP7	FFFF51B8H	00000000H	
PPS7	FFFF51BCH	00000000H	
PPV8	FFFF51C0H	00000000H	FFFE000-FFFFFFFHH
PPT8	FFFF51C4H	00000000H	
PPP8	FFFF51C8H	00000000H	
PPS8	FFFF51CCH	00000000H	



### 7.1.1 PPM - 周辺装置保護動作モードの設定

周辺装置保護機能に関する動作モードの設定を行うレジスタです。32ビット/16ビット/8ビット/1ビット単位でのアクセスが可能です。

ビット31-1には必ず0を設定してください。

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
PPM	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	PPM PP MSK
																	初期値 00000000H

ビット位置	ビット名	意味
0	PPMPP MSK	<p>周辺装置保護例外（PPI例外）の利用可否を選択します。</p> <p>0：PPI例外を使用します（初期値）。</p> <p>1：PPI例外を使用しません。</p> <p>PPMPPMSKビットがセット（1）されている場合、NT状態で周辺装置保護違反を検出しても、PPEC.PPECPPVDビットを変更しません。したがって、PPI例外の通知がされず、PPI例外が発生しません。</p> <p>PPMPPMSKビットの操作は必ずPSW.PPビットが0、かつPPEC.PPECPPVDビットが0の状態でのみ行ってください。</p>

### 7.1.2 PPEC - 周辺装置保護例外の制御

例外の通知状態を制御するレジスタです。32ビット/16ビット/8ビット単位でのアクセスが可能です。  
ビット31-1には必ず0を設定してください。

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
PPEC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	PPEC PPVD

初期値  
00000000H

ビット位置	ビット名	意味
0	PPEC PPVD <sup>注</sup>	<p>PPI例外の通知状況を示します。</p> <p>このビットがセット（1）されている場合はCPUへPPI例外を通知していて、かつPPI例外が受け付けられていない状態を示します。CPUがPPI例外を受け付けた時点で、このビットは自動的にクリア（0）されます。</p> <p>PPM.PPMPPMSKビットが0にセット（1）されている場合、NT状態で動作中に周辺装置保護違反を検出した時、このビットを自動的にセット（1）し、PPI例外を通知します。PPMPPMSKビットが1の場合は、自動的に値が変化することはありません。</p> <p>また、PPECPPVDビットがセットされていて、かつPSW.PP=1の間はすべてのメモリ・アクセスを行う命令のデータ・アクセスを無効化します（SYSCALL命令によるデータ・アクセスを除く）。</p> <p>PPECPPVDビットがセット（1）されている状態から、プログラムによってクリア（0）を行うことで、PPI例外の通知を取り下げることができます。PPI例外の通知を取り下げると、CPUがPPI例外を受け付けることはありません。</p> <p>0：PPI例外非通知状態（PPI例外の通知を行っていません）。（初期値）</p> <p>1：PPI例外通知状態（PPI例外の通知を行っています）。</p>

注 PPECPPVDビットに対する書き込み操作は、クリア（0）のみ可能です。セット（1）は行えません。

### 7.1.3 VPNECR - 周辺装置保護NTステート違反要因

NTステート中に周辺装置保護違反を検出した際に、違反となったアクセスの情報を保存する32ビットのレジスタです。32ビット単位でのアクセスが可能です。

ビット31-16, 11, 7, 6, 3-1には必ず0を設定してください。

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
VPNECR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	VPN ECR SP	VPN ECR OP	VPN ECR RWP	VPN ECR WP	0	VPN ECR WD	VPN ECR HW	VPN ECR BY	0	0	VPN ECR RD	VPN ECR WR	0	0	0	VPN ECR VD

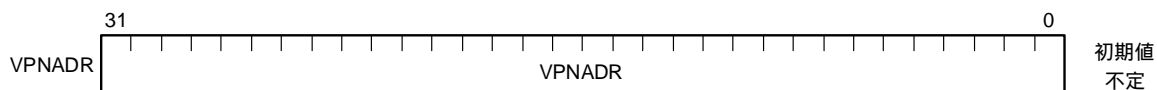
初期値  
不定

ビット位置	ビット名	意味
15	VPNECR SP	NTステートにおいて、周辺装置保護違反を検出したとき、違反を起こしたアクセスが、特殊周辺装置へのアクセスであった場合、セット(1)されます。それ以外の場合はクリア(0)されます。
14	VPNECR OP	NTステートにおいて、周辺装置保護違反を検出したとき、違反を起こしたアクセスが、OS周辺装置へのアクセスであった場合、セット(1)されます。それ以外の場合はクリア(0)されます。
13	VPNECR RWP	NTステートにおいて、周辺装置保護違反を検出したとき、違反を起こしたアクセスが、ライト・アクセスとリード・アクセスの保護が設定されている一般周辺装置へのアクセスであった場合、セット(1)されます。それ以外の場合はクリア(0)されます。
12	VPNECR WP	NTステートにおいて、周辺装置保護違反を検出したとき、違反を起こしたアクセスが、ライト・アクセスの保護が設定されている一般周辺装置へのアクセスであった場合、セット(1)されます。それ以外の場合はクリア(0)されます。
10	VPNECR WD	NTステートにおいて、周辺装置保護違反を検出したとき、違反を起こしたアクセスが、ワード・アクセスであった場合、セット(1)されます。それ以外の場合はクリア(0)されます。
9	VPNECR HW	NTステートにおいて、周辺装置保護違反を検出したとき、違反を起こしたアクセスが、ハーフワード・アクセスであった場合、セット(1)されます。それ以外の場合はクリア(0)されます。
8	VPNECR BY	NTステートにおいて、周辺装置保護違反を検出したとき、違反を起こしたアクセスが、バイト・アクセスであった場合、セット(1)されます。それ以外の場合はクリア(0)されます。
5	VPNECR RD	NTステートにおいて、周辺装置保護違反を検出したとき、違反を起こしたアクセスが、リード・アクセスであった場合、セット(1)されます。それ以外の場合はクリア(0)されます。
4	VPNECR WR	NTステートにおいて、周辺装置保護違反を検出したとき、違反を起こしたアクセスが、ライト・アクセスであった場合、セット(1)されます。それ以外の場合はクリア(0)されます。
0	VPNECR VD	NTステートにおいて、周辺装置保護違反を検出したことを示します。NTステート中に周辺装置へのアクセスで周辺装置保護違反を検出するとセット(1)されます。VPNECRVDビットがセットされている場合、新たにNTステートで周辺装置保護違反を検出しても、VPNECR, VPNADR, VPNTIDレジスタを更新せず、保持します。例外処理の完了後、VPNECRVDビットを必ずクリアしてください。 また、周辺装置保護を利用する以前に、VPNECRVDビットを必ずクリアしてください。

### 7.1.4 VPNADR - 周辺装置保護NTステート違反アドレス

NTステート中に周辺装置保護違反を検出した際に、違反となったアクセスのアドレスを保存する32ビットのレジスタです。32ビット単位でのアクセスが可能です。

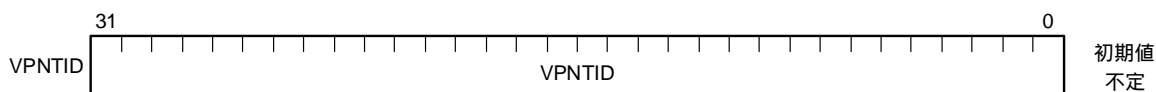
**注意** このレジスタに保存されるアドレスは、各製品のバス・システムに依存します。このレジスタに格納されたアドレスと、アーキテクチャ上の論理アドレスとの対応付けは製品ごとのマニュアルを参照してください。



ビット位置	ビット名	意味
31-0	VPNADR	NTステートにおいて、周辺装置保護違反を検出するとそのメモリ・アドレスを格納します。

### 7.1.5 VPNTID - 周辺装置保護NTステート違反タスクID

NTステート中に周辺装置保護違反を検出した時点に実行中のタスクIDを保存する32ビットのレジスタです。32ビット単位でのアクセスが可能です。



ビット位置	ビット名	意味
31-0	VPNTID	NTステートにおいて、周辺装置保護違反を引き起こした命令の実行時点のタスクの識別子内容（TIDレジスタの値）を格納します。

### 7.1.6 VPTECR - 周辺装置保護Tステート違反要因

Tステート中に周辺装置保護違反を検出した際に、違反となったアクセスの情報を保存する32ビットのレジスタです。32ビット単位でのアクセスが可能です。ビット31-16, 14-11, 7, 6, 3-1には必ず0を設定してください。

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
VPTECR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	VPT ECR SP	0	0	0	0	VPT ECR WD	VPT ECR HW	VPT ECR BY	0	0	VPT ECR RD	VPT ECR WR	0	0	0	VPT ECR VD
																初期値 不定

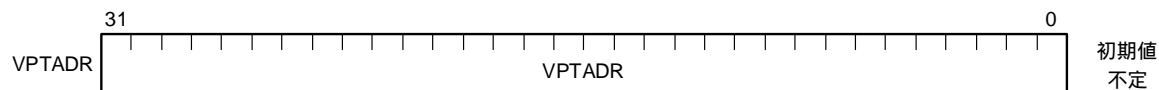
  

ビット位置	ビット名	意味
15	VPTECR SP	Tステートにおいて、周辺装置保護違反を検出したとき、違反を起こしたアクセスが、特殊周辺装置へのアクセスであった場合、セット(1)されます。それ以外の場合はクリア(0)されます。
10	VPTECR WD	Tステートにおいて、周辺装置保護違反を検出したとき、違反を起こしたアクセスが、ワード・アクセスであった場合、セット(1)されます。それ以外の場合はクリア(0)されます。
9	VPTECR HW	Tステートにおいて、周辺装置保護違反を検出したとき、違反を起こしたアクセスが、ハーフワード・アクセスであった場合、セット(1)されます。それ以外の場合はクリア(0)されます。
8	VPTECR BY	Tステートにおいて、周辺装置保護違反を検出したとき、違反を起こしたアクセスが、バイト・アクセスであった場合、セット(1)されます。それ以外の場合はクリア(0)されます。
5	VPTECR RD	Tステートにおいて、周辺装置保護違反を検出したとき、違反を起こしたアクセスが、リード・アクセスであった場合、セット(1)されます。それ以外の場合はクリア(0)されます。
4	VPTECR WR	Tステートにおいて、周辺装置保護違反を検出したとき、違反を起こしたアクセスが、ライト・アクセスであった場合、セット(1)されます。それ以外の場合はクリア(0)されます。
0	VPTECR VD	Tステートにおいて、周辺装置保護違反を検出したことを示します。Tステート中に周辺装置へのアクセスで周辺装置保護違反を検出するとセット(1)されます。VPTECRVDビットがセットされている場合、新たにTステートで周辺装置保護違反を検出しても、VPTECR、VPTADR、VPTTIDレジスタを更新せず、保持します。例外処理の完了後、VPTECRVDビットを必ずクリアしてください。 また、周辺装置保護を利用する以前に、VPTECRVDビットを必ずクリアしてください。

### 7.1.7 VPTADR - 周辺装置保護Tステート違反アドレス

Tステート中に周辺装置保護違反を検出した際に、違反となったアクセスのアドレスを保存する32ビットのレジスタです。32ビット単位でのアクセスが可能です。

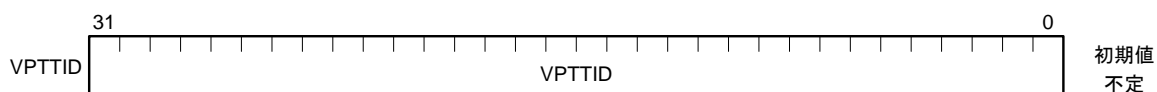
**注意** このレジスタに保存されるアドレスは、各製品のバス・システムに依存します。このレジスタに格納されたアドレスと、アーキテクチャ上の論理アドレスとの対応付けは製品ごとのマニュアルを参照してください。



ビット位置	ビット名	意味
31-0	VPTADR	Tステートにおいて、周辺装置保護違反を検出するとそのメモリ・アドレスを格納します。

### 7.1.8 VPTTID - 周辺装置保護Tステート違反タスクID

Tステート中に周辺装置保護違反を検出した時点に実行中のタスクIDを保存する32ビットのレジスタです。32ビット単位でのアクセスが可能です。



ビット位置	ビット名	意味
31-0	VPTTID	Tステートにおいて、周辺装置保護違反を引き起こした命令の実行時点のタスクの識別子内容 (TIDレジスタの値) を格納します。

### 7.1.9 PPSn - 特殊周辺装置の指定

各ビットに対応する周辺装置を特殊周辺装置に指定します。32ビット/16ビット/8ビット/1ビット単位でのアクセスが可能です。リセット時には製品ごとに定められた初期値を設定します。

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PPSn	PPSn	PPSn	PPSn	PPSn	PPSn	PPSn	PPSn	PPSn	PPSn	PPSn	PPSn	PPSn	PPSn	PPSn	PPSn	PPSn
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PPSn	PPSn	PPSn	PPSn	PPSn	PPSn	PPSn	PPSn	PPSn	PPSn	PPSn	PPSn	PPSn	PPSn	PPSn	PPSn
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

初期値  
表7-2参照

ビット位置	ビット名	意味
31-0	PPSn31-PPSn0	<p>各ビットに対応する周辺装置を特殊周辺装置に指定します。</p> <p>このビットを1にセットした場合は、このビットに対応するPPPn, PPVn, PPTnレジスタのビットを1にセットすることを推奨します。</p> <p>特殊周辺装置は信頼済みプログラムの実行中(Tステート)であっても、すべてのアクセスを周辺装置保護違反として検出します。</p> <p>0: このビットに対応する周辺装置は特殊周辺装置ではない。</p> <p>1: このビットに対応する周辺装置は特殊周辺装置である。</p> <p>初期値は各システム要件に合わせ製品仕様として定義され、特定の値に固定されている場合があります。</p>

### 7.1.10 PPPn - OS周辺装置の指定

各ビットに対応する領域の周辺装置をOS周辺装置に指定します。32ビット/16ビット/8ビット/1ビット単位でのアクセスが可能です。リセット時には製品ごとに定められた初期値を設定します。

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PPPn	PPPn	PPPn	PPPn	PPPn	PPPn	PPPn	PPPn	PPPn	PPPn	PPPn	PPPn	PPPn	PPPn	PPPn	PPPn	PPPn
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PPPn	PPPn	PPPn	PPPn	PPPn	PPPn	PPPn	PPPn	PPPn	PPPn	PPPn	PPPn	PPPn	PPPn	PPPn	PPPn
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

初期値  
表7-2参照

ビット位置	ビット名	意味
31-0	PPPn31-PPPn0	<p>各ビットに対応する周辺装置をOS周辺装置に指定します。</p> <p>このビットを1にセットした場合は、このビットに対応するPPVn, PPTnレジスタのビットを1にセットすることを推奨します。また、既にPPSnレジスタによって特殊周辺装置に指定されている場合は、このビットを1にセットすることを推奨します。</p> <p>OS周辺装置は信頼済みプログラム(Tステート)のみアクセスすることが可能です。信頼済みでないプログラム(NTステート)からのOS周辺装置へのアクセスは、すべて周辺装置保護違反を検出します。</p> <p>0: このビットに対応する周辺装置はOS周辺装置ではない。</p> <p>1: このビットに対応する周辺装置はOS周辺装置である。</p> <p>初期値は各システム要件に合わせ製品仕様として定義され、特定の値に固定されている場合があります。</p>

### 7.1.11 PPVn - 一般周辺装置保護の有効指定

一般周辺装置へのアクセスに対する違反検出の有無を指定するレジスタです。32ビット/16ビット/8ビット/1ビット単位でのアクセスが可能です。リセット時には製品ごとに定められた初期値を設定します。

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
PPVn	PPVn	PPVn	PPVn	PPVn	PPVn	PPVn	PPVn	PPVn	PPVn	PPVn	PPVn	PPVn	PPVn	PPVn	PPVn	PPVn	
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	PPVn	PPVn	PPVn	PPVn	PPVn	PPVn	PPVn	PPVn	PPVn	PPVn	PPVn	PPVn	PPVn	PPVn	PPVn	PPVn	初期値
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	表7-2参照

ビット位置	ビット名	意味
31-0	PPVn31-PPVn0	<p>各ビットに対応する周辺装置が一般周辺装置の場合に、実行中の信頼済みでないプログラム（NTステート）に対するアクセス制限の有効/無効を示します。</p> <p>既にPPSn, PPPnレジスタによって特殊周辺装置かOS周辺装置に指定されている場合は、このビットを1にセットすることを推奨します。</p> <p>一般周辺装置へのアクセスはこのビットがクリア（0）されている場合、アクセスが制限されず、違反を検出しません。このビットがセット（1）されている場合は、PPTnレジスタの対応するビットの指示に従って、違反が検出される可能性があります。</p> <p>0：このビットに対応する一般周辺装置へのアクセスは制限されません。</p> <p>1：このビットに対応する一般周辺装置へのアクセスは制限されます。</p> <p>初期値は各システム要件に合わせ製品仕様として定義され、特定の値に固定されている場合があります。</p>



### 7.1.12 PPTn - 一般周辺装置の保護種別の指定

一般周辺装置へのアクセスに対する違反検出の詳細を指定するレジスタです。32ビット/16ビット/8ビット/1ビット単位でのアクセスが可能です。リード・アクセスのみ可能です。

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PPTn	PPTn	PPTn	PPTn	PPTn	PPTn	PPTn	PPTn	PPTn	PPTn	PPTn	PPTn	PPTn	PPTn	PPTn	PPTn	PPTn
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PPTn	PPTn	PPTn	PPTn	PPTn	PPTn	PPTn	PPTn	PPTn	PPTn	PPTn	PPTn	PPTn	PPTn	PPTn	PPTn
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

初期値  
表7-2参照

ビット位置	ビット名	意味
31-0	PPTn31- PPTn0	<p>各ビットに対応する周辺装置が一般周辺装置の場合に、実行中の信頼済みでないプログラム（NTステート）に対するアクセス制限の内容を指示します。</p> <p>既にPPSn, PPPnレジスタによって特殊周辺装置かOS周辺装置に指定されている場合は、このビットを1にセットすることを推奨します。</p> <p>PPVnレジスタのビットでアクセス制限を行うと指定された周辺装置に対して、次のようなアクセス制限を指示します。</p> <p>0：このビットに対応する一般周辺装置へのリード・アクセスを制限しません。 このビットに対応する一般周辺装置へのライト・アクセスを違反とします。</p> <p>1：このビットに対応する一般周辺装置へのリード・アクセスを違反とします。 このビットに対応する一般周辺装置へのライト・アクセスを違反とします。</p> <p>初期値は各システム要件に合わせ製品仕様として定義され、特定の値に固定されている場合があります。</p>

## 7.2 メモリ保護と周辺装置保護の位置付け

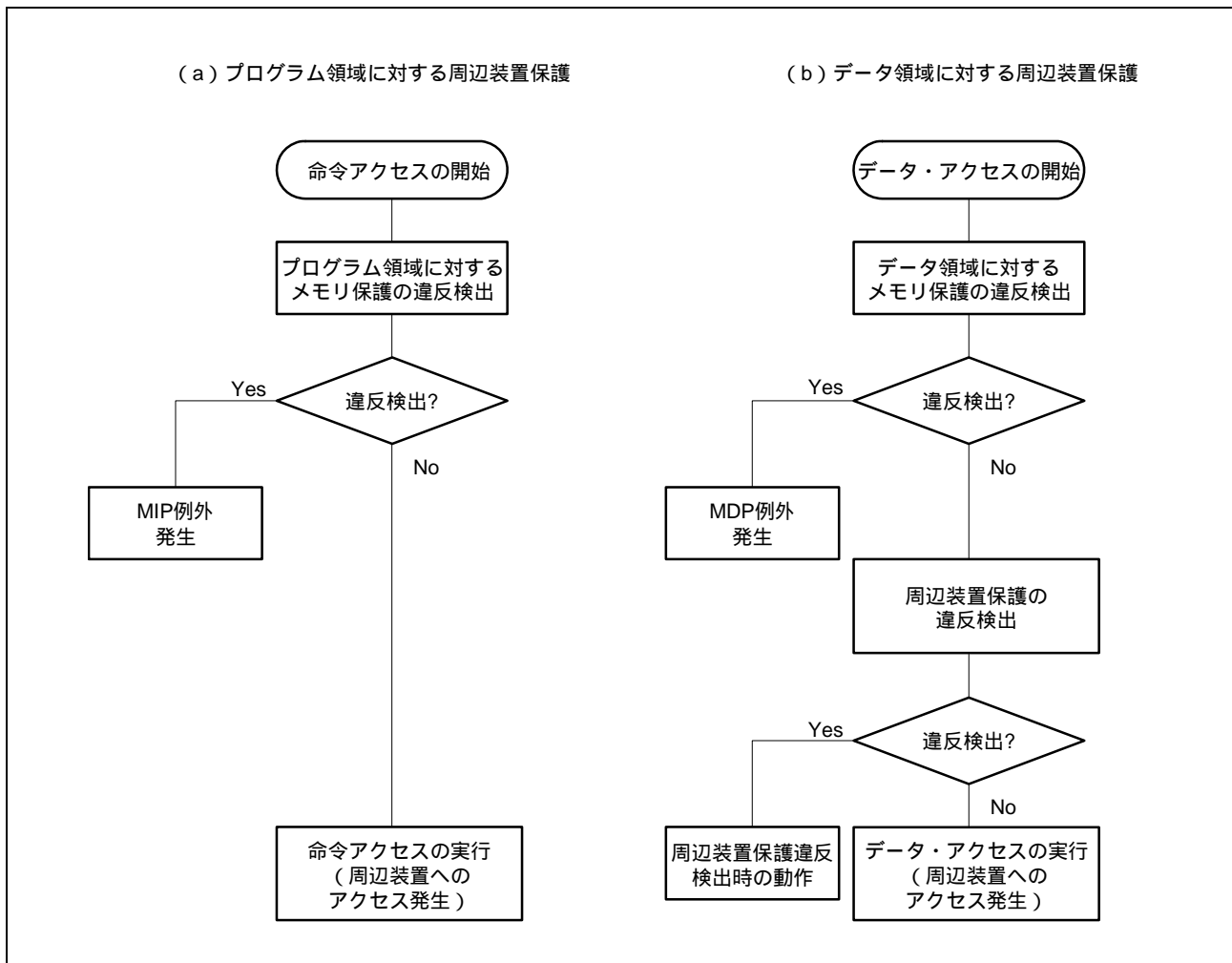
V850E2M CPUは、メモリ・マップトI/Oを採用しており、一方、メモリ保護はCPUの持つアドレス空間上のデータ領域に対して等しく適用されます。このため、周辺装置に対するいずれのデータ・アクセスに対しても、メモリ保護と周辺装置保護がそれぞれ適用されます。これは、V850E2M CPUが、メモリ・マップトI/Oアーキテクチャを採用しており、かつメモリ保護がCPUの持つアドレス空間上のデータ領域に対して等しく適用されるからです<sup>注</sup>。

CPUはまず、ある周辺装置へのデータ・アクセスに対して、メモリ保護によってアクセスの許可/禁止を判定し違反を検出します。禁止と判定し違反を検出した場合、周辺装置保護は動作せず直ちにMDP例外が発生します。許可と判定した場合、CPUはこのアクセスに対して、周辺装置保護によってアクセスの許可/禁止を判定し違反を検出します。禁止と判定し違反を検出した場合、周辺装置保護の動作設定/CPUの状態に従って、場合によってはPPI例外が発生します。許可と判定した場合は、周辺装置へのデータ・アクセスを行います。

注 プログラムの実行、分岐などによって発生する命令アクセスに対しては、周辺装置保護は適用されません。

メモリ保護と周辺装置保護の関係を図7-1に示します。

図7-1 メモリ保護と周辺装置保護の位置づけ



### 7.3 周辺装置の種類

周辺装置はシステムごとに考慮された利用方法に従って、それぞれ利用者を定めたり、利用するシチュエーションを限定したりすることが一般的です。周辺装置保護では、個々の周辺装置（あるいは周辺装置のグループ）ごとに次の3つの種類に分類して管理します。

- ・特殊周辺装置
- ・OS周辺装置
- ・一般周辺装置

### 7.3.1 周辺装置の種類の設定

PPSnレジスタ，PPPNレジスタの同一番号のビットの組合せが，個々の周辺装置の種類を示します。表7-2に，各ビットの組合せによって，どの種類の周辺装置保護が適用されるかを示します。

PPSn, PPPnレジスタは，システム初期化時，OSなどの初期化時を除いて，システムの運用中には変更しないでください。

表7-2 周辺装置の種類

PPSnのビット	PPPNのビット	種類
1	0または1(1推奨)	特殊周辺装置
0	1	OS周辺装置
0	0	一般周辺装置

#### (1) 特殊周辺装置

周辺装置の中でも特にプロセッサが動作するための最も基本的な環境を制御する周辺装置を「**特殊周辺装置**」として取り扱います。たとえば，クロック制御や，電源制御などを行うような周辺装置は，いかなる状況であっても，不用意にアクセスすることは避けるべきです。特殊周辺装置として指定した周辺装置は，このような背景を踏まえて，たとえ「信頼済みプログラム」であっても，特定の手順を踏まない限りアクセスができないように制御します。

#### (2) OS周辺装置

周辺装置の中で，「信頼済みプログラム」のみがアクセス可能な周辺装置を，信頼済みプログラムの代表となるOSの名前を冠して「**OS周辺装置**」として取り扱います。OSなどが最低限，自己の動作を保証するための重要な周辺装置を，このOS周辺装置として指定してください。たとえば，割り込みコントローラや，DMAコントローラなど，不用意に利用するとシステムを不安定な状態にするものは，このOS周辺装置として指定しておくべきです。OS周辺装置は「信頼済みでないプログラム」であるユーザ・アプリケーションからのアクセスが許可されません。このため，OSなどはユーザ・アプリケーションがいかなる操作を行おうとも，自己の動作を保証することができます。

#### (3) 一般周辺装置

周辺装置の中でも「信頼済みでないプログラム」からもアクセスが可能な周辺装置を，「**一般周辺装置**」として取り扱います。特殊周辺装置でも，OS周辺装置でもない周辺装置は，すべて，この一般周辺装置として取り扱います。一般周辺装置は汎用タイマや，A/Dコンバータ，通信系マクロなどの比較的緩やかな制限で取り扱っても，システム全体への影響が少ない周辺装置を割り当ててください。また，一般周辺装置については，ユーザ・アプリケーションごとに異なるアクセス制御を行うための仕組みを備えています。詳細は7.3.2 **一般周辺装置に対する詳細な保護設定**を参照してください。

### 7.3.2 一般周辺装置に対する詳細な保護設定

特殊周辺装置，OS周辺装置に対するアクセスは，信頼済みでないプログラムからはあらゆるアクセスを違反として検出しますが，一般周辺装置に対しては現在動作中のプログラムごとに，アクセスの権利を指定できます。このため一般周辺装置に対してはPPVnレジスタ，PPTnレジスタのビットの組み合わせによって，より詳細な保護方針を指定する必要があります。

OSなどは各プログラムの実行開始前に，それぞれのプログラムごとに適切な値をPPVnレジスタに設定することができます。これによって，実行中のプログラムごとに異なる一般周辺装置へのアクセス制御を可能にし，細粒度の周辺装置保護を実現します。

OSなどはPPVnレジスタの他に，PPTnレジスタを変更することができますが，タスク切り替え処理の性能低下を避けるため，運用中には頻繁にPPTnレジスタを変更しないことを推奨します。

表7-3 一般周辺装置のアクセス制御

PPVnのビット	PPTnのビット	一般周辺装置のアクセス制御
0	0または1(1推奨)	リード許可/ライト許可
1	0	リード許可/ライト禁止
1	1	リード禁止/ライト禁止

## 7.4 Tステートにおける周辺装置保護違反

CPUは信頼済みプログラムの実行時にはTステート (PSW.PPビット = 0) を示します。この状態で周辺装置保護違反として検出するのは、特殊周辺装置へアクセスをしようとした場合のみとなります。Tステートで違反を検出した場合、周辺装置保護は次のような動作を行います。

- ・違反を引き起こしたアクセスを阻止し、周辺装置へ出力させません。
- ・違反を引き起こしたアクセスの情報を、VPTECR, VPTTID, VPTADRレジスタへ格納します。

Tステート中は、クリティカルな処理を行っている可能性があるため、処理を中断させないために例外を発生させません。運用時には適切なタイミングで違反情報を格納するVPTECR, VPTTID, VPTADRレジスタを確認して、違反に対する処理を行ってください。

周辺装置のリード・アクセス違反検出に伴う実行阻止においては、ロード命令のデスティネーション・レジスタは更新されます。この更新において、デスティネーション・レジスタに格納される値は製品仕様として定義されます。

## 7.5 NTステートにおける周辺装置保護違反

CPUは信頼済みでないプログラムの実行中はNTステート (PSW.PP = 1) を示します。この状態では特殊周辺装置、OS周辺装置、一般周辺装置のすべての周辺装置へのアクセスに対して、周辺装置保護違反の検出を行います。NTステートで違反を検出した場合、周辺装置保護はTステートとは異なる次のような動作を行います。

- ・違反を引き起こしたアクセスを阻止し、周辺装置への影響を抑止します。
- ・違反を引き起こしたアクセスの情報を、VPNECR, VPNTID, VPNADRレジスタへ格納します。
- ・CPUへPPI例外を通知し、PPI例外処理の起動を待ち合わせます。
- ・PPI例外処理が起動されるまでの間、NTステートの場合に後続のメモリ・アクセスをすべて無効化します (周辺装置へのアクセスだけでなく、すべてのメモリ・アクセスが対象です)。

無効化に関しては7.5.1 後続アクセスの無効化を参照してください。

### 7.5.1 後続アクセスの無効化

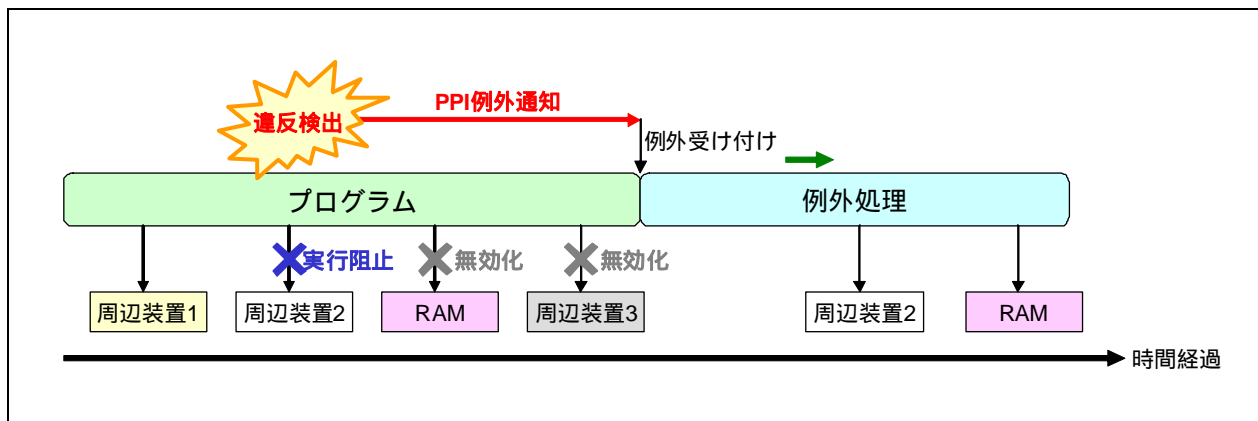
NTステートで動作する「信頼済みでないプログラム」で周辺装置保護違反が検出された場合、そのプログラムの実行継続は危険であると判断し、PPI例外を通じてOSなどを呼び出し、例外処理を起動することができます。

ただし、ハードウェア上の制約からPPI例外はただちに通知されず、例外処理の起動が遅延する可能性があります。一方、ソフトウェア上の都合により不可分の処理中で、例外処理を起動したくない場合もあります。このような場合には、周辺装置保護違反の検出からPPI例外処理の開始までの間に「信頼済みでないプログラム」が後続命令を実行する可能性があります。

これに対処するために、NTステートで周辺装置保護違反を検出してから、PPI例外処理が開始されるまでの間、後続のメモリ・アクセスをすべて無効化します。この後続アクセスの無効化によって、メモリ・アクセス要求は存在しなかったものとして扱われるので、周辺装置保護違反を検出した命令以降の後続命令によって、CPU外部の資源が更新されることを抑止できます。

無効化はPPI例外処理が開始されるまで継続しますが、PPI例外処理が開始される以前に、割り込みなどによって他のプログラムへ移行しそのプログラムがTステートであった場合は、Tステートであるうちは無効化を行いません。復帰処理等を行った後、NTステートへ再び移行した場合、メモリ・アクセスは再び無効化されます。

図7-2 後続アクセスの無効化



この仕様は「信頼済みでないプログラム」はOSなどによってスケジュールされ、「信頼済みでないプログラム」間の移行時には必ずOSなどが先にPPI例外処理を行った後、次の「信頼済みでないプログラム」の実行を開始することを想定しています。このような制御を行うことで、ある「信頼済みでないプログラム（NTステート）」で起きた無効化が、他の「信頼済みでないプログラム（NTステート）」には影響を与えません。また「信頼済みプログラム（Tステート）」は元々無効化が行われません。

周辺装置のリード・アクセス違反検出に伴う実行阻止、無効化においては、ロード命令のデスティネーション・レジスタは更新されます。

## 7.6 PPI例外の取り扱い

NTステートで動作する「信頼済みでないプログラム」で周辺装置保護違反を検出した場合、そのプログラムの実行継続は危険であると判断しPPI例外を通じてOSなどを呼び出し、速やかに例外処理を行います。

### 7.6.1 PPI例外の取り下げ

PPI例外は通知の遅延、及びプログラム上の不可分なアトミック操作などによって、受け付けが遅延される可能性があります。このため、PPI例外の処理が開始される前に、PPI例外を引き起こしたプログラムが終了処理を行ってしまう場合があります。このような問題に対処するために、終了処理の前には必ずSYNCE命令と、PPECレジスタのPPECPPVDビットを用いてPPI例外を取り下げ、無効化を解除してください。詳細は、[第2編 6.3 例外の管理](#)を参照してください。

### 7.6.2 PPI例外を用いない運用方法

応用によっては、動作中に例外を発生させることでプログラム・シーケンスを乱すことが不利益に繋がる場合があります。そのような場合には、周辺装置保護機能はPPI例外を起こさずに運用することが可能です。

PPI例外を起こさない運用を行う場合は、PPM.PPMPPMSKビットをセット(1)してください。これによって、PPEC.PPECPPVDビットが周辺装置保護違反の検出によってセット(1)されることがなくなり、PPI例外は発生しません。運用時には違反情報を格納するVPNECR, VPNTID, VPNADRレジスタを適切なタイミングで確認して、違反に対する処理を行ってください。

### 7.6.3 PPI例外処理で行うべき操作

例外処理を行った場合は、例外処理からの復帰後、次の違反検出時に違反情報が適切に保存されるようにVPNECR.VPNECRVDビットをクリアしてください。このビットをクリアしない場合、次の違反を検出しても、違反を引き起こしたメモリ・アクセスに対応する違反情報が格納されません。

## 7.7 周辺装置保護違反の検出結果一覧

NTステート、Tステート、及び各周辺装置保護の種類を加えた違反検出の結果は、表7-4のようにまとめられます。

表7-4 周辺装置保護違反の検出結果

周辺装置の種類	PPSn	PPPn	PPTn	PPVn	信頼済みプログラム (Tステート)		信頼済みでないプログラム (NTステート)	
					PSW.PP = 0		PSW.PP = 1	
					リード時	ライト時	リード時	ライト時
特殊周辺装置	1	0または1 (1推奨)	0または1 (1推奨)	0または1 (1推奨)	違反	違反	違反	違反
OS周辺装置	0	1	0または1 (1推奨)	0または1 (1推奨)	-	-	違反	違反
一般周辺装置	0	0	1	1	-	-	違反	違反
			0		-	-	-	違反
			0または1 (1推奨)	0	-	-	-	-

また、違反検出後の動作は、次のようにまとめられます。

表7-5 周辺装置保護違反の検出後の動作

違反検出時の状態	Tステート	NTステート	
PSW.PP	0	1	
PPM.PPMPPMSK	0または1	0	1
違反したアクセスの阻止	する	する	
違反情報の保存	VPTECR/VPTID/VPTADRに格納	VPNECR/VPNTID/VPNADRに格納	
PPI例外	発生しない	発生する	発生しない
後続の無効化	しない	する	



## 7.8 特殊周辺装置へのアクセス方法

通常、特殊周辺装置へのアクセスはNTステート、Tステートにかかわらず、すべての状態で違反を検出し、アクセスを阻止します。システムの挙動を制御するために、特殊周辺装置へのアクセスを行いたい場合は、次の手順でアクセスを行ってください。

- <1> Tステートで動作する信頼済みプログラムへ移行する。
- <2> アクセス対象となる特殊周辺装置に対応するPPSnレジスタのビットをクリアし、一時的にOS周辺装置に変更する。
- <3> 周辺装置へアクセスを行う。
- <4> <2>で変更したPPSnレジスタのビットを元の状態へ戻す（セットする）。

**注意** 上記手順以外での、特殊周辺装置へのアクセスを不可能にすることで、システムの安全性を高めています。特殊周辺装置をOS周辺装置へ変更することは、システムの安全性を下げることになるため、上記の手順による変更は限定的な短期間に限って行うことを推奨します。

## 7.9 周辺装置保護設定レジスタに対する保護設定

周辺装置保護の設定を行う周辺I/Oレジスタ自身も、周辺装置保護の対象となります。周辺装置保護設定レジスタは次のいずれかの方針で保護されます。

- ・ 周辺装置保護設定レジスタは、暗黙的にOS周辺装置である。
- ・ 周辺装置保護設定レジスタに対応するPPSn, PPPn, PPVn, PPTnビットを定義する。

**注意** 1. ただし、周辺装置保護設定レジスタに対応するPPSn上のビットは、必ず0固定ビットとしてください。  
2. 周辺装置保護設定レジスタに対応するPPPn, PPVn, PPTn上のビットは任意の値に固定、あるいは変更可能なビットとして構いませんが、OS周辺装置以外を示した場合、OSなどによる保護機能の提供が保証できなくなります。

## 7.10 特殊な周辺装置アクセス命令

### 7.10.1 SYSCALL命令

SYSCALL命令は、OSなどの管理プログラムによって提供されるサービスの呼び出しに用いる命令です。

サービスは信頼済みのプログラムであり、またサービスへ分岐するためのアドレス・テーブルもまた信頼済みであるため、SYSCALL命令による周辺装置アクセスは、PSW.PPビットがセット（1）されている状態であっても、周辺装置保護が適用されません。

したがって、SYSCALL命令の実行時にPPI例外が検出されることはありません。

これにより、デバッグなどに周辺装置領域にテスト・コードを配置した場合などでも、周辺装置保護とSYSCALL命令を使用することができます。

## 第8章 タイミング監視機能

タイミング監視機能は、プログラムの実行時間などの時間管理を行い、CPU 実行時間の不当な占有を防ぐための機能です。

タイミング監視機能は同一の機能を持った6つのカウンタを備えており、このカウンタを所定の運用方法に従って動作させることで、次の6種類の監視機能を提供します。

- ・ランタイム監視機能
- ・デッドライン監視機能
- ・リソース監視機能
- ・グローバル割り込みロック監視機能
- ・ソフトウェア割り込みロック監視機能
- ・割り込み到着回数監視

各カウンタは28ビットのカウンタ幅を持ち、CPUのクロック・サイクルごとにカウント動作を行います。また、1-1024までの分周カウント機能を持ちCPUクロック・サイクル精度で最大約1.34秒、CPUクロック・サイクルの1024倍の精度で最大約1374秒（約23分）のカウント・レンジを持ちます（CPU周波数が200MHz時）。

また、各カウンタが設定に従ってCPU状態を監視した結果として違反を検出した場合、TSI例外を発生させます。TSI例外はFEレベル例外として定義されており、通常のユーザ・アプリケーションのクリティカル・セクション（PSW.IDビットを1とし、割り込みを受け付けない期間）中であっても、TSI例外処理は強制的に例外処理を起動することが可能です。TSI例外については、8.5 TSI例外の取り扱いを参照してください。

### 8.1 レジスタ・セット

タイミング監視機能に関わる周辺装置レジスタを、表8-1に示します。

タイミング監視機能に関わるレジスタは、すべて周辺装置レジスタ領域に配置します。タイミング監視機能のレジスタ・セットのベース・アドレスはハードウェア定義で、V850E2M CPUでは FFFF5000Hです。

表8-1 タイミング監視機能レジスタ・セット

レジスタ名	アドレス	アクセス可能サイズ				初期値
		1	8	16	32	
TSEC	00H	✓	✓	✓	✓	00000000H
TSECR	04H		✓	✓	✓	00000000H
TSCCFGn	20H + (n*10H) + 0H		✓	✓	✓	00000000H
TSCCNTn	20H + (n*10H) + 4H				✓	表8-2参照
TSCCMPn	20H + (n*10H) + 8H				✓	表8-2参照
TSCRLDn	20H + (n*10H) + CH				✓	表8-2参照

備考 n=0-5

V850E2M CPUでのタイミング監視機能カウンタ・レジスタ ( TSCCFGn, TSCCNTn, TSCCMPn, TSCRLDnレジスタ ) のレジスタ名 , アドレス , 初期値を表8 - 2に示します。

表8 - 2 V850E2M CPUのタイミング監視機能カウンタ・レジスタ

レジスタ名	アドレス	初期値	TSUカウンタ
TSCCFG0	FFFF5020H	00000000H	TSUカウンタ0
TSCCNT0	FFFF5024H	00000000H	
TSCCMP0	FFFF5028H	00000000H	
TSCRLD0	FFFF502CH	00000000H	
TSCCFG1	FFFF5030H	00000000H	TSUカウンタ1
TSCCNT1	FFFF5034H	00000000H	
TSCCMP1	FFFF5038H	00000000H	
TSCRLD1	FFFF503CH	00000000H	
TSCCFG2	FFFF5040H	00000000H	TSUカウンタ2
TSCCNT2	FFFF5044H	00000000H	
TSCCMP2	FFFF5048H	00000000H	
TSCRLD2	FFFF504CH	00000000H	
TSCCFG3	FFFF5050H	00000000H	TSUカウンタ3
TSCCNT3	FFFF5054H	00000000H	
TSCCMP3	FFFF5058H	00000000H	
TSCRLD3	FFFF505CH	00000000H	
TSCCFG4	FFFF5060H	00000000H	TSUカウンタ4
TSCCNT4	FFFF5064H	00000000H	
TSCCMP4	FFFF5068H	00000000H	
TSCRLD4	FFFF506CH	00000000H	
TSCCFG5	FFFF5070H	00000000H	TSUカウンタ5
TSCCNT5	FFFF5074H	00000000H	
TSCCMP5	FFFF5078H	00000000H	
TSCRLD5	FFFF507CH	00000000H	

### 8.1.1 TSEC - タイミング監視の制御

タイミング監視機能の制御を行う機能ビットを配置したレジスタです。32ビット/16ビット/8ビット/1ビット単位でのアクセスが可能です。

ビット31-1には必ず0を設定してください。

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
TSEC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TSEC ESUP
																	初期値 00000000H

ビット位置	ビット名	意味
0	TSECESUP	<p>TSECESUPビットがセットされている間、タイミング監視カウンタが例外を要求することを抑制します。</p> <p>TSECESUPビットは、TSI例外が受け付けられた時点で自動的にセット(1)され、TSI例外の通知が制御されます。これにより、TSI例外処理中に、再度TSI例外へ突入することを防ぎます。</p> <p>また、TSECESUPビットがセットされている間に発生したTSI例外は保留され、TSECESUPビットのクリア後に、失われることなくTSI例外を再度要求します。</p> <p>TSECESUPビットは、TSI例外処理の終了前に、必ずクリア(0)してください。</p>

### 8.1.2 TSECR - タイミング監視例外要因

タイミング監視例外（TSI例外）の例外要因レジスタです。32ビット/16ビット/8ビット単位でのアクセスが可能です。

ビット31-6には必ず0を設定してください。

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
TSECR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	0	0	0	0	0	0	0	0	0	0	TSECR	TSECR	TSECR	TSECR	TSECR	TSECR	初期値 00000000H
											ES5	ES4	ES3	ES2	ES1	ES0	

ビット位置	ビット名	意味
5	TSECR ES5	タイミング監視カウンタ5の例外要因ビットです。 TSCCFG5.TSCCFG5ESビットの写像ビットです。
4	TSECR ES4	タイミング監視カウンタ4の例外要因ビットです。 TSCCFG4. TSCCFG4ESビットの写像ビットです。
3	TSECR ES3	タイミング監視カウンタ3の例外要因ビットです。 TSCCFG3. TSCCFG3ESビットの写像ビットです。
2	TSECR ES2	タイミング監視カウンタ2の例外要因ビットです。 TSCCFG2. TSCCFG2ESビットの写像ビットです。
1	TSECR ES1	タイミング監視カウンタ1の例外要因ビットです。 TSCCFG1. TSCCFG1ESビットの写像ビットです。
0	TSECR ES0	タイミング監視カウンタ0の例外要因ビットです。 TSCCFG0. TSCCFG0ESビットの写像ビットです。

### 8.1.3 TSCCFGn - タイミング監視機能カウンタnの設定 (n = 0-5)

タイミング監視用のカウンタnの設定レジスタです (n = 0-5)。32ビット / 16ビット / 8ビット単位でのアクセスが可能です。

ビット31-27, 23, 22, 19, 15-11, 7-4には必ず0を設定してください。

TSCCFGnレジスタは各フィールドを1バイト境界に配置しています。このため、いずれかのフィールドのみをアクセスしたい場合は、バイト・アクセスを行うことで部分的な操作が行えます (たとえば、ステータスのみをクリアしたい場合は、オフセット + 1Hにバイト書き込みを行う、など)。

( 1/3 )

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
TSCCFGn	0	0	0	0	0	TSCCFGnRES			0	0	TSCCFGnCTM		0	TSCCFGnARM	TSCCFGnEXM	TSCCFGnUDM	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	初期値
	0	0	0	0	0	TSCCFGnOS	TSCCFGnES	TSCCFGnVS	0	0	0	0	TSCCFGnRLD	TSCCFGnFEACT	TSCCFGnEIACT	TSCCFGnACT	00000000H

ビット位置	ビット名	意味
26-24	TSCCFGnRES	タイミング監視カウンタの解像度を選択します。 000 : 1クロック・サイクルごとに、カウントを行います。 001 : 4クロック・サイクルごとに、カウントを行います。 010 : 16クロック・サイクルごとに、カウントを行います。 011 : 64クロック・サイクルごとに、カウントを行います。 100 : 256クロック・サイクルごとに、カウントを行います。 101 : 1024クロック・サイクルごとに、カウントを行います。 110 : RFU 111 : RFU
21, 20	TSCCFGnCTM	動作モードを選択します。 00 : 通常モード 01 : グローバル割り込みロック監視モード 10 : ランタイム監視モード 11 : RFU <ul style="list-style-type: none"> <li>・通常モード 通常のカウンタとして動作します。自動的なカウンタの開始 / 停止を行いません。</li> <li>・グローバル割り込みロック監視モード PSW.IDビットが0から1に変化したときに、自動的にTSCRLDn.TSCRLDnVALビットの値をTSCCNTn.TSCCNTnVALビットに転送します。その後、TSCCFGnACT = 1に更新し、カウンタを開始します。 PSW.IDビットが1から0に変化したときに、自動的にTSCCFGnACT = 0に更新し、カウンタを停止します。</li> <li>・ランタイム監視モード EIレベル例外 / FEレベル例外の受け付け時に、自動的にTSCCFGnACTの値を、TSCCFGnEIACT / TSCCFGnFEACTに転送します。TSCCFGnACT = 0に更新し、カウンタを停止します。 EIRET / FERET命令が実行された場合、自動的にTSCCFGnEIACT / TSCCFGnFEACTビットの値をTSCCFGnACTに転送します。この結果、TSCCFGnACT = 1となった場合は、カウンタを再開します。</li> </ul>

ビット位置	ビット名	意味
18	TSCCFGn ARM	オート・リロードの有効/無効を選択します。 違反検出時 (TSCCNTn.TSCCNTnVALビットとTSCCMPn.TSCCMPnVALビットが一致した場合) に、TSCRLDn.TSCRLDnVALビットの値を、TSCCNTn.TSCCNTnVALビットに転送するかどうかを選択します。 0: オート・リロードを行いません (無効)。 1: オート・リロードを行います (有効)。
17	TSCCFGn EXM	例外モードを選択します。 違反検出時 (TSCCNTn.TSCCNTnVALビットとTSCCMPn.TSCCMPnVALビットが一致した場合) に、TSI例外を通知するかどうかを選択します。 0: TSI例外非通知モード。TSI例外を通知させません。 1: TSI例外通知モード。TSI例外を通知させます。
16	TSCCFGn UDM	カウント方向を選択します。 0: アップ・カウンタとして動作します (加算)。 1: ダウン・カウンタとして動作します (減算)。
10	TSCCFGn OS <sup>注</sup>	カウンタのオーバーフロー・ビットです。 TSCCNTn.TSCCNTnVALビットがオーバーフロー/またはアンダフローを起こした場合にセットされます。 0: このカウンタは、オーバーフロー/アンダフローを起こしていません。 1: このカウンタは、オーバーフロー/アンダフローを起こしました。
9	TSCCFGn ES <sup>注</sup>	例外要因ビットです。 例外モードが「TSI例外通知モード (TSCCFGnEXM = 1)」のとき、TSI例外が受け付けられた場合、セットされます。このビットはTSI例外処理の終了前に必ずクリア (0) してください。 0: このカウンタは、現在処理中のTSI例外の要因ではありません。 1: このカウンタは、現在処理中のTSI例外の要因です。
8	TSCCFGn VS <sup>注</sup>	違反検出ビットです。 0: 違反非検出状態です。 1: 違反検出状態です。 違反検出時 (TSCCNTn.TSCCNTnVALビットとTSCCMPn.TSCCMPnVALビットが一致した場合) にセットします。 さらに、例外モードが「TSI例外通知モード (TSCCFGnEXM = 1)」で、かつ、このビットがセットされている場合は、TSI例外を要求します。 このカウンタがTSI例外を要求している状態で、TSI例外が受け付けられた場合は、このビットをクリアします。 このビットをソフトウェア処理によってクリアした場合、このカウンタによるTSI例外の要求を取り消すことができます。
3	TSCCFGn RLD	リロード・コマンド・ビットです。 このビットをセット (1) した場合、TSCRLDn.TSCRLDnVALビットをTSCCNTn.TSCCNTnVALビットに転送します。 このビットは常に0が読み出されます。

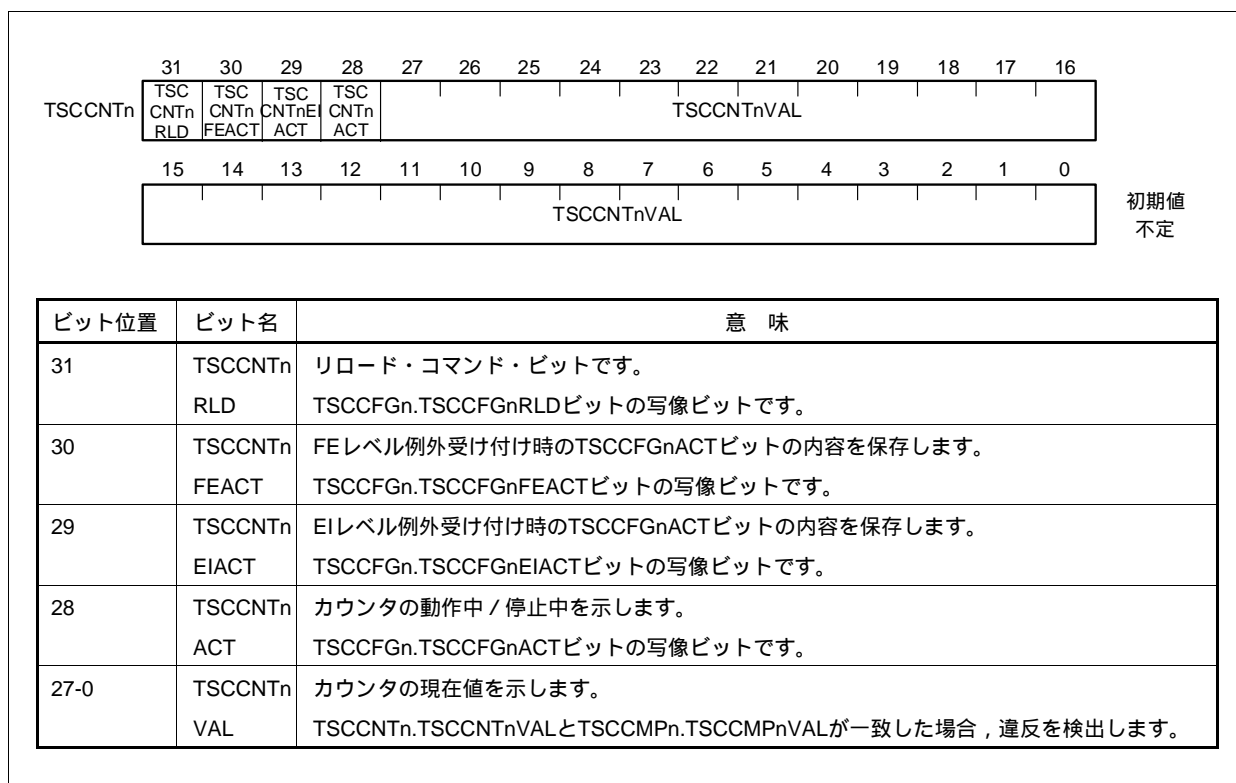
注 TSCCFGnOS, TSCCFGnVS, TSCCFGnESビットに対する書き込み操作は、クリア (0) のみ可能です。セット (1) することではできません。

ビット位置	ビット名	意味
2	TSCCFGn FEACT	FEレベル例外受け付け時のTSCCFGnACTビットの内容を保存します。  <ul style="list-style-type: none"> <li>・ランタイム監視モードの場合 FEレベル例外受け付け時に、TSCCFGnACTビットの内容をTSCCFGnFEACTビットに転送し、TSCCFGnACTビットをクリア(0)します。 FERET命令実行時に、TSCCFGnFEACTビットの内容をTSCCFGnACTビットに転送します。</li> <li>・ランタイム監視モード以外の場合 FERET命令の実行時にも、TSCCFGnFEACTビットの内容をTSCCFGnACTビットに転送しません。</li> </ul>
1	TSCCFGn EIACT	EIレベル例外受け付け時のTSCCFGnACTビットの内容を保存します。  <ul style="list-style-type: none"> <li>・ランタイム監視モードの場合 EIレベル例外受け付け時に、TSCCFGnACTビットの内容をTSCCFGnEIACTビットに転送し、TSCCFGnACTビットをクリア(0)します。 EIRET命令実行時に、TSCCFGnEIACTビットの内容をTSCCFGnACTビットに転送します。</li> <li>・ランタイム監視モード以外の場合 EIRET命令の実行時にも、TSCCFGnEIACTビットの内容をTSCCFGnACTビットに転送しません。</li> </ul>
0	TSCCFGn ACT	カウンタの動作中/停止中を示します。このビットをセット(1)することでカウント動作の開始、クリア(0)することで停止を行えます。 <ul style="list-style-type: none"> <li>0: 停止中</li> <li>1: 動作中</li> </ul> ランタイム監視モード時、グローバル割り込みロック監視モード時には、自動で値が変化します。詳細は、TSCCFGnCTMビットの説明を参照してください。 また、MPM.MPEビットが0の場合は、0に固定されます。この結果、タイミング監視カウンタは動作せず、違反の検出、TSI例外の要求を行わなくなります。



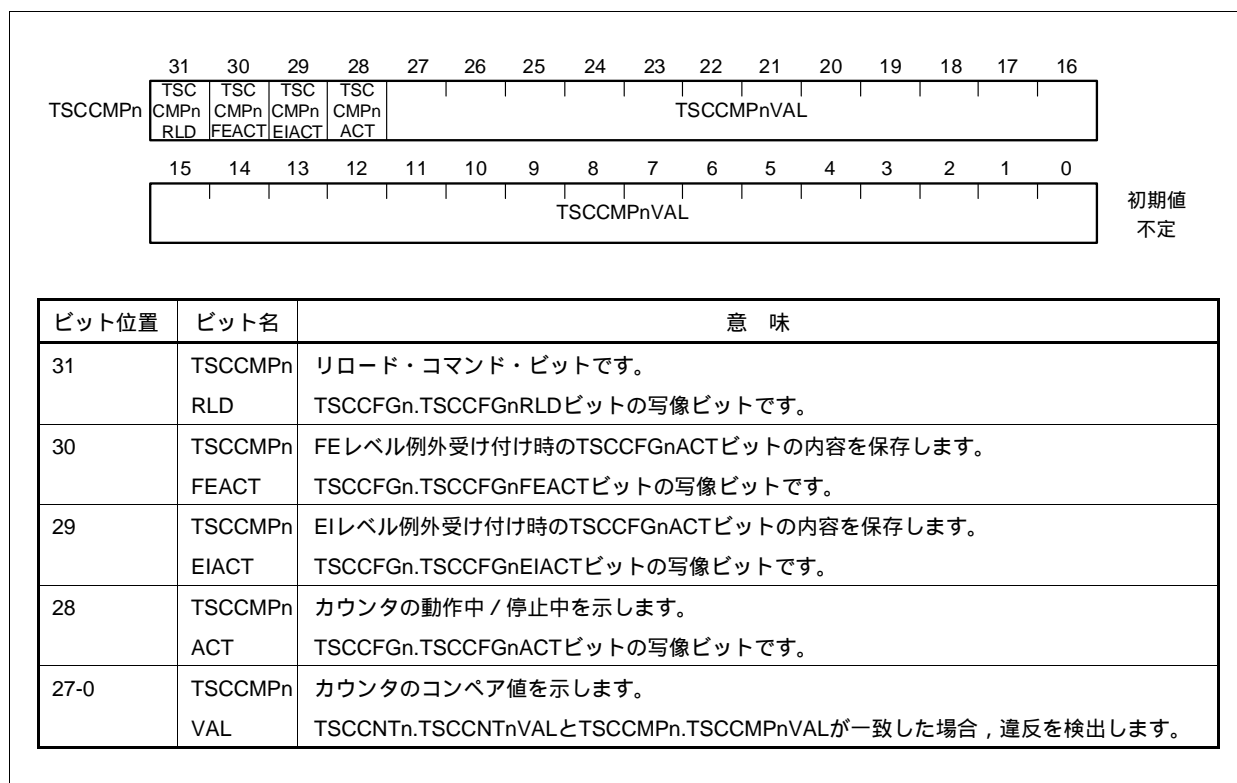
### 8.1.4 TSCCNTn - タイミング監視カウンタnのカウンタ値 (n = 0-5)

タイミング監視のカウンタnのカウンタ・レジスタです (n = 0-5)。32ビット単位でのアクセスが可能です。



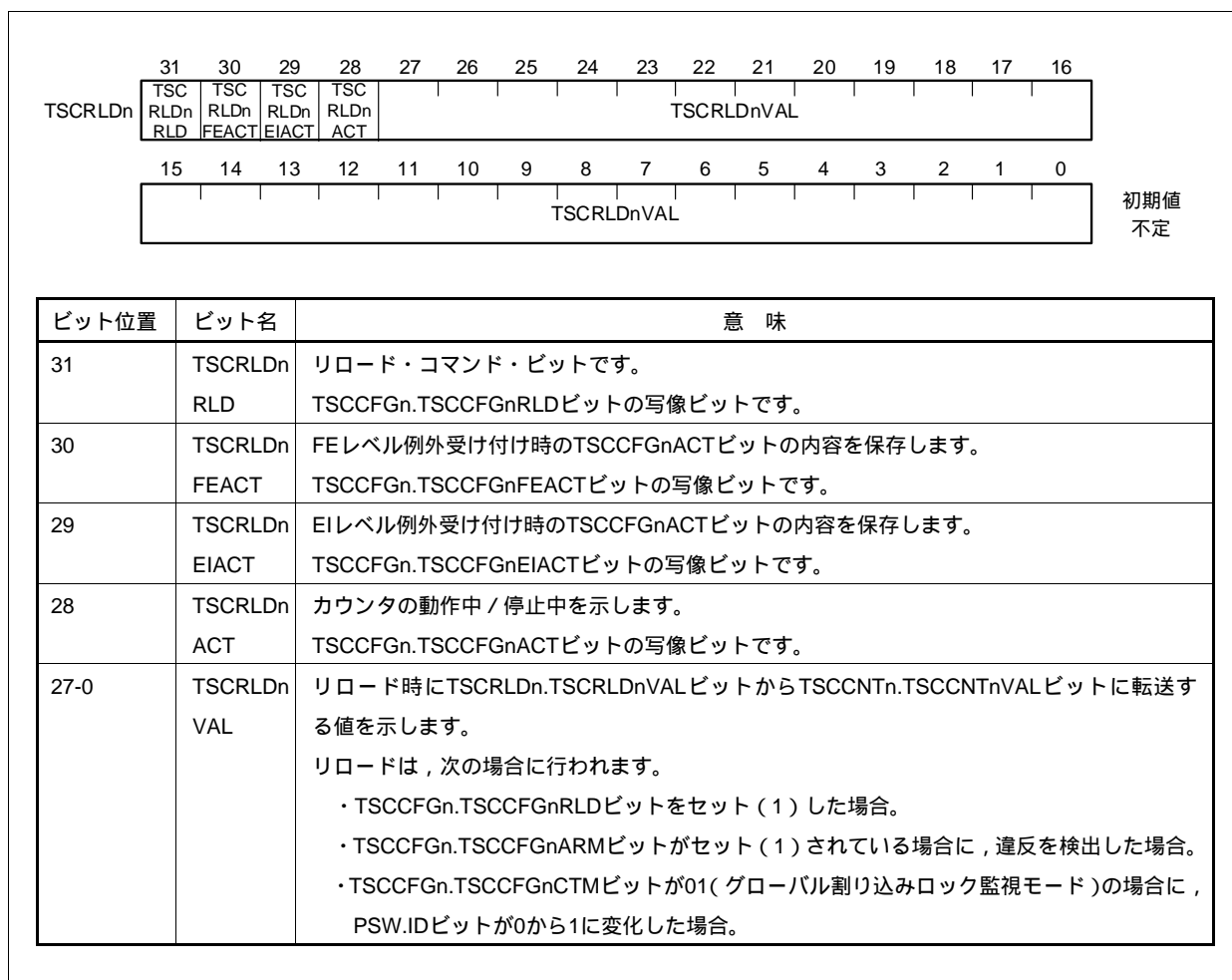
### 8.1.5 TSCCMPn - タイミング監視カウンタnのコンペア値 (n = 0-5)

タイミング監視のカウンタnのコンペア・レジスタです (n = 0-5)。32ビット単位でのアクセスが可能です。



### 8.1.6 TSCRLDn - タイミング監視カウンタnのリロード値 (n = 0-5)

タイミング監視のカウンタnのリロード・レジスタです (n = 0-5)。32ビット単位でのアクセスが可能です。



## 8.2 カウンタ機能

タイミング監視機能のため、同一の機能を持った6つのカウンタを備えています。各カウンタは28ビットのカウント幅を持ち、CPUのクロック・サイクルごとにカウント動作を行い、1-1024までの6段階の分解能設定が可能です。タイミング監視は、カウンタ値が予め設定したコンペア値と一致することで違反を検出します。タイミング監視違反を検出すると、設定に従ってCPUにTSI例外要求を通知します。

### 8.2.1 解像度

TSCCFGn.TSCCFGnRESビットによって、カウンタの1カウントごとの解像度を設定します。解像度とは、カウンタ（TSCCNTn.TSCCNTnVAL）の1カウントが、何クロック・サイクルにあたるかを示します。たとえば、解像度が1の場合、カウンタ値の1は、1クロックを示します。解像度が1024の場合は、カウンタ値の1は1024クロックを示します。

各カウンタはそれぞれ別々に、表8-2の解像度を選択可能です。

表8-2 解像度

TSCCFGn.TSCCFGnRES ビットの値	動作
000	1クロック・サイクルごとに、カウントを行います。
001	4クロック・サイクルごとに、カウントを行います。
010	16クロック・サイクルごとに、カウントを行います。
011	64クロック・サイクルごとに、カウントを行います。
100	256クロック・サイクルごとに、カウントを行います。
101	1024クロック・サイクルごとに、カウントを行います。
上記以外	RFU

**注意** 解像度1以外の場合、最大で解像度の2倍のクロック数のカウント誤差が生じます。低解像度で利用する場合には、この誤差に注意してください。

### 8.2.2 カウント方向

TSCCFGn.TSCCFGnUDMビットを設定することで、アップ・カウンタ・モード、ダウン・カウンタ・モードを選択可能です。それぞれ、カウント動作の1単位ごとにカウンタ値を増加させるか、減少させるかを意味します。

#### (1) アップ・カウンタ・モード

カウンタの1カウントごとに、カウンタ値（TSCCNTn.TSCCNTnVAL）を1増加させます。

#### (2) ダウン・カウンタ・モード

カウンタの1カウントごとに、カウンタ値（TSCCNTn.TSCCNTnVAL）を1減少させます。

### 8.2.3 例外モード

TSCCFGn.TSCCFGnEXMビットによって、カウンタ値 (TSCCNTn.TSCCNTnVAL) が規定のコンペア値 (TSCCMPn.TSCCMPnVAL) と一致し違反を検出した場合に、TSI例外を通知するか、しないかを選択可能です。通常のタイミング監視用途では、TSI例外を通知する設定で使用しますが、カウンタを他の用途で利用する場合には、目的に応じては、TSI例外を通知しない運用も可能です。

#### (1) TSI例外非通知モード

カウンタが違反を検出 (カウンタ値がコンペア値と一致) した場合に、TSI例外を通知しません。

#### (2) TSI例外通知モード

カウンタが違反を検出 (カウンタ値がコンペア値と一致) した場合に、TSI例外を通知します。

### 8.2.4 オート・リロード

TSCCFGn.TSCCFGnARMビットによって、カウンタ値 (TSCCNTn.TSCCNTnVAL) が規定のコンペア値 (TSCCMPn.TSCCMPnVAL) と一致し違反を検出した場合に、自動的にリロード値 (TSCRLDn.TSCRLDnVAL) をカウンタ値に転送するかどうかを選択可能です。周期的な時間を計測したい場合など、違反の検出と同時に次のカウントを即座に開始したい場合には、オート・リロードを有効に設定してください。

#### (1) オート・リロード無効

カウンタが違反を検出 (カウンタ値がコンペア値と一致) した場合に、自動的にリロードを行いません。カウントは、現在のカウンタ値からそのまま続きます。

#### (2) オート・リロード有効

カウンタが違反を検出 (カウンタ値がコンペア値と一致) した場合に、自動的にリロードを行います。カウントは、そのままリロード値から続きます。

### 8.2.5 カウンタ・モード

特定の目的のために利用するための特殊なカウンタ・モードを備えています。TSCCFGn.TSCCFGnCTMビットによって通常のカウンタとしての動作モードの他に、グローバル割り込みロック監視モード、ランタイム監視モードを選択可能です。

#### (1) 通常モード

通常のカウンタとして動作します。自動的にカウンタが開始/停止することではなく、ソフトウェアによるTSCCFGn.ACTビットへの操作によってのみ制御を行うモードです。グローバル割り込みロック監視やランタイム監視を行わない場合は、必ずこのモードに設定してください。

**注意** グローバル割り込みロック監視モード、ランタイム監視モードで利用していたカウンタの利用を停止する場合は、必ず通常モードへ変更を行ってください。グローバル割り込みロック監視モードやランタイム監視モードのままにしておいた場合、他のプログラムの実行によって自動的に動作が再開される可能性があります。

**(2) グローバル割り込みロック監視モード**

グローバル割り込みロック監視を行う場合に、このモードに設定してください。グローバル割り込みロック監視では、ユーザ・アプリケーションが不正にクリティカル・セクションを延長し、CPUの他のプログラムの動作を妨げないように監視を行います。グローバル割り込みロック期間として、PSW.IDビットがセット(1)されている間の時間を監視し、所定の時間を越えた場合に違反を検出します。

ユーザ・アプリケーションではDI命令、EI命令によって余分な手続きの必要なくクリティカル・セクションを開始/終了することができるため、PSW.IDビットの状態を監視し、自動的にカウントの開始、停止、リロードを行います。

次にグローバル割り込みロック監視モード時の動作を次に示します。

表8-3 グローバル割り込みロック監視モード時の動作

ビット	PSW.IDが0から1に変化したとき	PSW.IDが1から0に変化したとき
TSCCFGnACT	1に更新	0に更新
TSCCNTn.TSCCNTnVAL	TSCRLDn.VALの値を転送	更新なし

**(3) ランタイム監視モード**

ランタイム監視を行う場合に、このモードに設定してください。ランタイム監視モードでは、現在実行中のプログラムの時間予算を厳密に管理するために、例外等によって他のタスクへ遷移する場合に、自動的に停止します。この時、例外以前のカウンタの動作状態(TSCCFGn.TSCCFGnACT)ビットの内容を、それぞれの例外レベル毎に保存しておきます(TSCCFGn.TSCCFGnEIACT/TSCCFGnFEACT)。それぞれの復帰命令(EIRET/FERET)実行時には、この保存内容(TSCCFGnEIACT/TSCCFGnFEACT)から、TSCCFGnACTビットへ内容を復帰することで、復帰直後から自動的にカウント動作を再開します。カウンタ値に関しては、各例外の先頭/末尾において、ソフトウェアによって適切に退避/復帰を行ってください。

以下にランタイム監視モード時の動作を表8-4に示します。

表8-4 ランタイム監視モード時の動作

ビット	EIレベル例外 受け付け時	EIRET命令実行時	FEレベル例外 受け付け時	FERET命令実行時
TSCCFGnACT	0に更新	TSCCFGnEIACTの値 を転送	0に更新	TSCCFGnFEACTの値 を転送
TSCCFGnEIACT	TSCCFGnACTの値を 転送	更新なし	更新なし	更新なし
TSCCFGnFEACT	更新なし	更新なし	TSCCFGnACTの値を 転送	更新なし
TSCCNTn.TSCCNTnVAL	更新なし	更新なし	更新なし	更新なし

**注意** CALLT,CTRET 命令の実行時には、いずれも更新しません。

## 8.3 カウンタとCPUの動作モード

タイミング監視機能の各カウンタは、それぞれのCPUの動作状態で次のような動作を行います。

表8-5 カウンタとCPUの動作モード

V850E2M 動作モード	システム・ クロック	MPM.MPE	実行中プログラム	TSCCFGnACT ビット	TSUカウンタ の動作状態
通常/ HALT状態 <sup>注1</sup>	供給	0	任意	0固定	TSCCFGnACT = 0状態に従い、カウンタ停止
		1	通常(タスク)/ EIレベル例外/ FEレベル例外	0	TSCCFGnACT = 0状態に従い、カウンタ停止
				1	TSCCFGnACT = 1状態に従い、カウンタ動作
スタンバイ・ モード	停止	前値保持	任意	前値保持 <sup>注2</sup>	クロックの供給が停止しているためカウンタ停止

注1. TSI例外の発生によりHALT状態を解除し、受け付け条件に従って例外を受け付けます。

2. 事前にTSCCFGnACTビットをクリア(0)し、タイミング監視機能を停止させてから、クロックを停止してください。

## 8.4 違反の検出

タイミング監視機能に関する違反は、いずれのカウンタ・モードであっても、カウンタ値(TSCCNTn.TSCCNTnVAL)が、コンペア値(TSCCMPn.TSCCMPnVAL)と同値になった場合に検出します。違反を検出した場合、対応するカウンタの違反検出ビット(TSCCFGn.TSCCFGnVS)ビットをセット(1)します。

### 8.4.1 違反要因の特定

タイミング監視機能の各カウンタは、違反の要因を示す情報を持ちません。各カウンタの監視内容をソフトウェアで管理し、適切に違反処理を行ってください。

## 8.5 TSI例外の取り扱い

タイミング監視機能は、違反を検出した場合、設定に従ってTSI例外を通知します。

### 8.5.1 TSI例外の通知

違反を検出した場合、例外モードが「TSI例外通知モード (TSCCFGn.TSCCFGnEXM = 1)」である場合、直ちにTSI例外をCPUに通知します。通知されたTSI例外は、PSW.NPがクリア(0)されていれば直ちに受け付けます。

TSI例外の受け付けと同時に、TSI例外を通知しているカウンタに対応する違反検出ビット (TSCCFGn.TSCCFGnVS) が例外要因ビット (TSCCFGn.TSCCFGnES) に保存されます。すなわち、そのカウンタのTSCCFGnESビットがセット(1)され、TSCCFGnVSビットがクリア(0)されます。

このように、TSI例外受け付け時点で例外を通知しているカウンタに対応するTSCCFGnVSビットの内容がTSCCFGnESビットに保存されるため、TSI例外処理中にタイミング監視カウンタが動作し続けた結果、新たに検出したタイミング監視違反と、現在のTSI例外処理の要因となった違反を区別することが可能です。

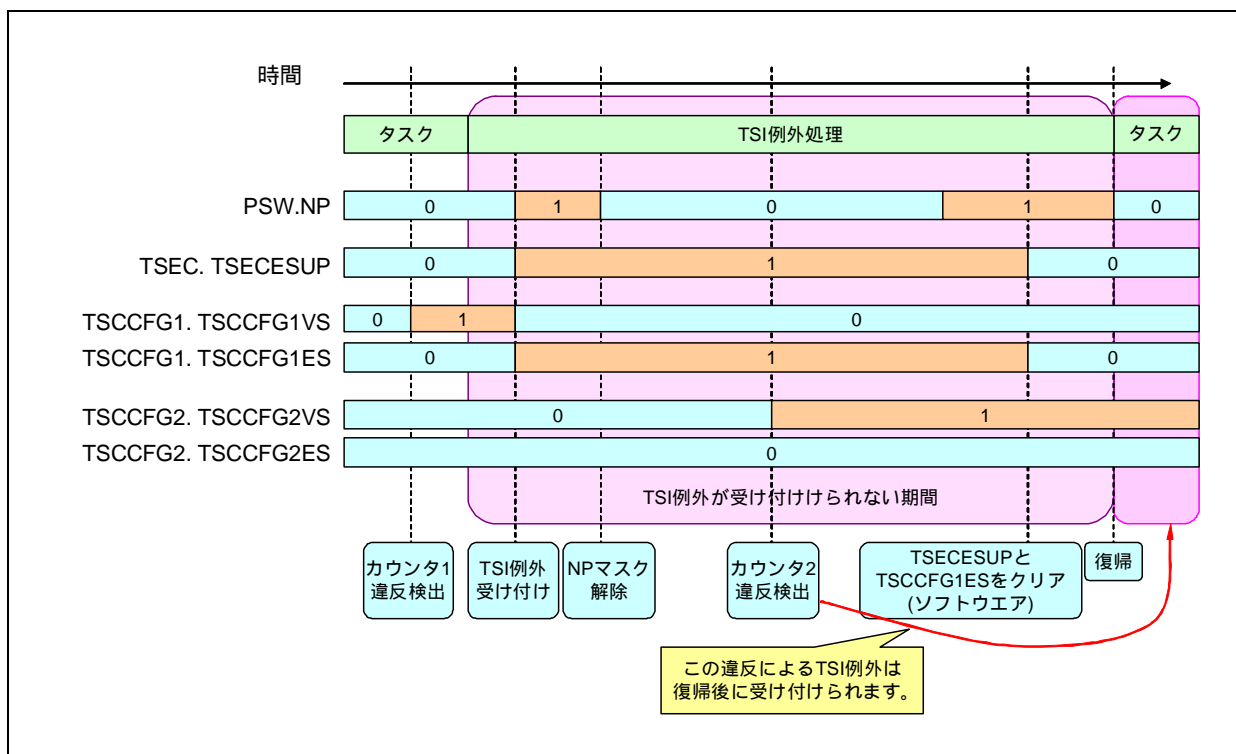
**注意** TSI例外が受け付けられた場合に、「TSI例外非通知モード」で動作しているカウンタのTSCCFGnVSビットは変更しません。

また、TSI例外の受け付けと同時にTSEC.TSECESUPビットがセット(1)され、TSI例外処理中のTSI例外の通知を抑制します。これによって、TSI例外処理中に他のカウンタが違反を検出したことで、再度TSI例外が通知され多重例外が発生することを防ぎます。

**注意** 次のTSI例外受け付けのために、TSI例外処理の終了以前に、必ず違反処理を行った各カウンタのTSCCFGn.TSCCFGnESビットと、TSEC.TSECESUPビットをクリア(0)してください。この処理を行わずに例外処理から復帰すると、次のTSI例外が受け付けられません。



図8-1 TSI例外の通知



### 8.5.2 例外要因の特定

タイミング監視機能の各カウンタは、例外要因となった理由を示す情報を持っていません。各カウンタがどのような内容についての監視を行っていたかはソフトウェアで管理し、適切に例外処理を行ってください。

### 8.5.3 TSI例外の取り下げ

他の例外処理の結果、あるカウンタの計測対象であるプログラムを終了させる場合などに、そのカウンタによるTSI例外が通知されていて、まだ受け付けられていない場合には下記の手順に従ってTSI例外を取り下げてください。このような処理が必要となる状況については、**第2編 6.3 例外の管理**を参照してください。

- <1> PSW.NPビットをセット(1)する。
- <2> 対象のカウンタのTSCCFGn.TSCCFGnACTビットをクリア(0)する。
- <3> 対象のカウンタのTSCCFGn.TSCCFGnCTMビットを、通常モードに変更する。
- <4> 対象のカウンタのTSCCFGn.TSCCFGnVSビットをクリア(0)する。

備考 TSCCFGnレジスタは32ビット・アクセス可能であり、<2>～<4>は同時に操作できます。

## 8.6 各監視機能に対応するカウンタの設定

それぞれの監視内容に合わせてカウンタの動作設定を行ってください。ここでは、各監視機能に対応するレジスタの推奨設定例を示します。実際の運用時にはそれぞれのOSなどあるいは目的に合わせて設定値を適切に調整してください。

### 8.6.1 グローバル割り込みロック監視

グローバル割り込みロック監視は、ユーザ・アプリケーションのクリティカル・セクションが所定の時間よりも長い時間継続しないように、PSW.IDビットを監視します。所定の時間を越えて、IDビットが1の状態を維持した場合、例外処理へ移行します。これによって、ユーザ・アプリケーションの設計ミスによるシステムのデッドロックを防止することが可能です。

表8-6 グローバル割り込みロック監視

レジスタ	ビット	設定例
TSCCFGn	TSCCFGnUDM	任意
	TSCCFGnEXM	1 (TSI例外を発生させる)
	TSCCFGnARM	0 (オート・リロードはしない)
	TSCCFGnCTM	1 (グローバル割り込みロック監視モード)
	TSCCFGnRES	任意
TSCCNTn	-	任意
TSCCMPn	-	アップ・カウンタの場合は、最大許容ロック時間。 ダウン・カウンタの場合は、0。
TSCRLDn	-	アップ・カウンタの場合は、0。 ダウン・カウンタの場合は、最大許容ロック時間。

### 8.6.2 ランタイム監視

ランタイム監視は、実行中のタスクに与えられた実行時間予算を監視し、予算を使い切った場合に例外処理を行います。また、割り込み等でプリエンブションが発生した際には、割り込み受け付け時/復帰時に自動的にカウンタの停止/再開を行う機能を備えており、CPUクロック・サイクル精度の正確できめ細やかな時間監視機能を提供します。

また、実行中のタスク内の一部の時間を監視する場合などは、それぞれの目的に応じて表8-7の設定値を適切に変更して使用してください。

たとえば、あるタスクがリソースを取得している時間を測る場合には、表8-7の設定を行った上で、リソースの取得時にソフトウェアにより開始を行い、リソースの解放時にソフトウェアにより停止を行うことで、そのタスクがリソースを取得中の間に、割り込みなどによるプリエンブションが発生した場合であっても、正確にタスクがリソースを取得している間の実行時間を監視できます。

表8-7 ランタイム監視

レジスタ	ビット	設定例
TSCCFGn	TSCCFGnUDM	任意
	TSCCFGnEXM	1 (TSI例外を発生させる)
	TSCCFGnARM	0 (オート・リロードはしない)
	TSCCFGnCTM	2 (ランタイム監視モード)
	TSCCFGnRES	任意
TSCCNTn	-	アップ・カウンタの場合は、0。 ダウン・カウンタの場合は、実行時間予算。
TSCCMPn	-	アップ・カウンタの場合は、実行時間予算。 ダウン・カウンタの場合は、0。
TSCRLDn	-	使用しません。

### 8.6.3 割り込み到着回数監視

割り込み到着回数監視は、一定のフレーム時間内に特定の割り込みの発生回数を監視し、所定の回数以上発生した場合はその割り込みの受け付けを禁止し、CPUの処理時間を他のプログラムに解放します。禁止された割り込みは、一定周期ごとに再び受け付けを許可されます。この周期動作のトリガとしてタイミング監視機能のカウンタを利用します。周期カウンタとして動作させるため、割り込み到着回数監視として使用するカウンタは、オート・リロードを使用し、カウンタが所定の時間に到達した瞬間に、例外を要求すると共に、次のタイム・フレームをカウントしはじめます。

また、割り込み到着率回数監視と同様に、一定周期で監視プログラムを起動する場合などは、それぞれの目的に応じて表8-8の設定値を適切に変更して使用してください。

表8 - 8 割り込み到着回数監視

レジスタ	ビット	推奨設定例
TSCCFGn	TSCCFGnUDM	任意
	TSCCFGnEXM	1 (TSI例外を発生させる)
	TSCCFGnARM	1 (オート・リロードをする)
	TSCCFGnCTM	0 (通常モード)
	TSCCFGnRES	任意
TSCCNTn	-	任意 (カウントの開始は、リロードと同時にTSCCFGnACTビットをセット)
TSCCMPn	-	アップ・カウンタの場合は、監視を行う単位フレーム時間。 ダウン・カウンタの場合は、0。
TSCRLDn	-	アップ・カウンタの場合は、0。 ダウン・カウンタの場合は、監視を行う単位フレーム時間。

#### 8.6.4 経過時間の監視

ある基準となるイベントの発生から、一定の時間を監視する場合には、基本的に次の表8 - 9のようにカウンタを設定してください。この設定を行うことで、ソフトウェアによってカウンタを開始し、ソフトウェアによって停止を行う、基本的なカウンタによる監視を行うことができます。

たとえば、デッドラインの監視や、特定のリソースの取得時間の監視、あるいは特定の割り込みソースがマスクされている時間などを監視する場合などは、それぞれの目的に応じて表8 - 9の設定値を適切に変更して使用してください。

表8 - 9 経過時間の監視

レジスタ	ビット	推奨設定例
TSCCFGn	TSCCFGnUDM	任意
	TSCCFGnEXM	1 (TSI例外を発生させる)
	TSCCFGnARM	0 (オート・リロードはしない)
	TSCCFGnCTM	0 (通常モード)
	TSCCFGnRES	任意
TSCCNTn	-	アップ・カウンタの場合は、0。 ダウン・カウンタの場合は、目標時間。
TSCCMPn	-	アップ・カウンタの場合は、目標時間。 ダウン・カウンタの場合は、0。
TSCRLDn	-	使用しません。

## 第9章 プロセッサ保護例外

この章では、プロセッサ保護違反と例外の種類について説明します。各例外の処理については、第2編 第6章 例外を参照してください。

### 9.1 違反の種類

V850E2M CPUでは保護機能ごとに設定に従って違反を検出し、場合によっては各保護機能で定められた例外が発生します。ここでは、この違反と例外の関係について詳細に説明します。

プロセッサ保護機能で定められた設定によって、次の5種類の違反が検出されます。

- ・システム・レジスタ保護違反
- ・実行保護違反
- ・データ保護違反
- ・周辺装置保護違反
- ・タイミング監視違反

#### 9.1.1 システム・レジスタ保護違反

システム・レジスタへの不正なアクセス時に検出される違反です。この違反に対しては、例外が発生しません。違反検出時の処理については、5.5 運用方法を参照してください。

#### 9.1.2 実行保護違反

命令の実行時に検出される違反です。プログラム領域上で実行が許可されていない領域に配置された命令を実行しようとした場合、実行保護違反が検出されます。

実行保護違反が検出された場合、常にMIP例外が発生します。

#### 9.1.3 データ保護違反

命令のデータ・アクセス時に検出される違反です。メモリ・アクセス命令が、データ領域上で許可されていない領域からデータをリード、またはライトしようとした際に検出されます。

データ保護違反が検出された場合、常にMDP例外が発生します。

#### 9.1.4 周辺装置保護違反

命令のデータ・アクセス時に検出される違反です。メモリ・アクセス命令が、メモリ保護によって許可されており、かつシステムごとに定義された周辺装置保護設定に基づいたアクセス制御で許可されていない操作を行った場合に検出されます。

周辺装置保護違反が検出された場合、設定に従ってPPI例外が発生します。

### 9.1.5 タイミング監視違反

タイミング監視機能により検出される違反です。タイミング監視機能の各カウンタが動作設定に従って、カウントを行い、指定された状態を満たした場合に、違反を検出します。

タイミング監視違反が検出された場合、設定に従ってTSI例外が発生します。

## 9.2 例外の種類

V850E2M CPUはプロセッサ保護機能で定められた4種類の例外が発生します。例外が発生すると、例外ハンドラ(00000030H)へ分岐動作を行い、要因ごとに定められた違反情報を違反レジスタなどに格納します。

### 9.2.1 MIP例外

実行保護違反を検出した場合に発生する例外です。許可されていないアドレスに配置された命令を実行しようとした場合に、ただちに発生するプレサイス例外であり、かつ例外処理後、違反が発生した命令から元の処理を正常に継続できる、「再開可能」かつ「回復可能」な例外です。

### 9.2.2 MDP例外

データ保護違反を検出した場合に発生する例外です。許可されていないアドレスに配置されたデータに対してリード、またはライトを実行した場合に、ただちに発生するプレサイス例外であり、かつ例外処理後、違反が発生した命令から元の処理を正常に継続できる、「再開可能」かつ「回復可能」な例外です。

### 9.2.3 PPI例外

周辺装置保護違反を検出した場合に発生する例外です。違反の原因となった事象から遅延して発生する可能性があり、またPSW.NPビットがセット(1)されている場合は例外の発生が保留されるインプレサイス例外です。違反が発生した命令が既に完了し、後続の命令を実行してしまってから例外処理が行われるため、違反を引き起こした命令からの正常に継続することが困難な「再開可能」かつ「回復不可能」な例外です。

PPI例外は、例外受け付け/復帰時の例外同期の対象となる例外であり、例外同期命令SYNCEによる例外同期の対象となる例外です。詳細は、**第2編 6.3.1 例外受け付け/復帰時の例外同期**、および**第2編 6.3.2 例外同期命令**を参照してください。

### 9.2.4 TSI例外

タイミング監視違反を検出した場合に発生する例外です。タイミング監視違反は、CPUのクロック入力で作るカウンタによって引き起こされるため、割り込みと同様に不特定のタイミングで発生します。このため、例外の受け付けが不可能な状態での受け付けを避けるため、PSW.NPビットがセット(1)されている場合は例外の発生が保留されます。命令や、レジスタ等に関連せずに発生するため、「再開可能」かつ「回復可能」な例外です。

TSI例外は、例外受け付け/復帰時の例外同期の対象となる例外です。詳細は、**第2編 6.3.1 例外受け付け/復帰時の例外同期**を参照してください。

## 9.3 違反要因の特定

保護違反が検出されると、CPUバンクのシステム・レジスタ上のFEICレジスタに、MIP例外、MDP例外、PPI例外、TSI例外のいずれかによって例外ハンドラへ分岐が発生したかを示す例外要因が格納されます。例外ハンドラ・アドレスが他の例外と共用されているため、例外要因による処理の分岐が必要です。表9-1に示すように、それぞれの例外の原因を示す補助情報がMPVバンクの特定のシステム・レジスタに格納されます。

表9-1 違反要因の特定

	FEIC	例外種別	違反内容	違反情報レジスタ
プロセッサ保護関連の違反	00000430H	MIP例外	実行保護違反	VMECR, VMADR, VMTID
	00000431H	MDP例外	データ保護違反	VMECR, VMADR, VMTID
	00000432H	PPI例外	周辺装置保護違反	VPNECR, VPNADR, VPNTID
	00000433H	TSI例外	タイミング監視違反	TSECR
その他の例外	-	-	-	-

### 9.3.1 MIP例外

FEICレジスタに00000430Hが格納されます。例外発生時のTIDレジスタの内容がVMTIDレジスタに、例外を発生した命令のPCがVMADRレジスタに格納されます。VMECRレジスタは、VMXビットがセット(1)され、そのほかのビットはクリア(0)されます。

MIP例外とMDP例外は同時に発生することがないため、違反情報レジスタ(VMECR, VMTID, VMADRレジスタ)はMDP例外と共用です。

表9-2 MIP例外発生時のVMECR設定値

レジスタ	VMECR						
	6	5	4	3	2	1	0
ビット番号	6	5	4	3	2	1	0
ビット名	VMMS	VMRMW	VMS	VMW	VMR	VMX	-
全命令	0	0	0	0	0	1	0

### 9.3.2 MDP例外

FEICレジスタに00000431Hが格納されます。例外発生時のTIDレジスタの内容がVMTIDレジスタに、例外を発生したメモリ・アクセスのアドレスがVMADRレジスタに格納されます。VMECRレジスタは、検出した違反の内容に応じて、VMR, VMW, VMSビットがセット(1)され、VMXビットはクリア(0)されます。

MIP例外とMDP例外は同時に発生することがないため、違反情報レジスタ(VMECR, VMTID, VMADRレジスタ)はMIP例外と共用です。

表9-3 MDP例外発生時のVMECR設定値

レジスタ	VMECR						
	6	5	4	3	2	1	0
ビット番号							
ビット名	VMMS	VMRMW	VMS	VMW	VMR	VMX	-
リード命令(アライン) <sup>注1</sup>	0	0	0/1 <sup>注5</sup>	0	1	0	0
ライト命令(アライン) <sup>注2</sup>	0	0	0/1 <sup>注5</sup>	1	0	0	0
リード命令(ミスアライン) <sup>注3</sup>	1	0	0/1 <sup>注5</sup>	0	1	0	0
ライト命令(ミスアライン) <sup>注2</sup>	1	0	0/1 <sup>注5</sup>	1	0	0	0
CAXI/SET1/NOT1/CLR1命令 <sup>注4</sup>	0	1	0/1 <sup>注5</sup>	0	1	0	0
PREPARE命令	0	0	1	1	0	0	0
DISPOSE命令	0	0	1	0	1	0	0

注1. LD/SLD/CALLT/SWITCH/TST1命令

2. ST/SST命令

3. LD/SLD命令

4. リード・モディファイ・ライトを行う命令は、リード・サイクル時にライト許可のチェックも行うため、リード時にのみ違反が発生します。

5. 命令のオペランド指定によってsp相対アクセスを行う場合1, それ以外は0

### 9.3.3 PPI例外

FEICレジスタに00000432Hが格納されます。また、違反検出時のタスクIDがVPNTIDレジスタに、違反を検出したメモリ・アクセスのアドレスがVPNADRレジスタに格納されます。またVPNECRレジスタには、例外要因となったアクセスや、対象の周辺装置に関する情報が格納されます。

注意1. Tステートにおける特殊周辺装置保護違反の検出時には、PPI例外は起きません。またその場合の違反情報は上記のVPNECR, VPNTID, VPNADRレジスタではなく、VPTECR, VPTTID, VPTADRレジスタに格納されます。詳細は7.4 Tステートにおける周辺装置保護違反を参照してください。

2. VPNADRレジスタに保存されるアドレスは、各製品のバス・システムに依存したアドレスです。VPNADRレジスタに格納されたアドレスと、アーキテクチャ上の論理アドレスとの対応付けは製品ごとのマニュアルを参照してください。

### 9.3.4 TSI例外

FEICレジスタに00000433Hが格納されます。また、TSI例外受け付け時に違反を検出していたカウンタに対応する例外要因ビット(TSECRレジスタの各ビット, またはTSCCFGn.TSCCFGnESビット)がセット(1)されます。



## 第10章 メモリ保護設定チェック機能

メモリ保護設定チェック機能は、OSなどで、ユーザ・アプリケーションから依頼された操作のためのデータ領域が、サービスの依頼元のアクセス権限内であるかどうかを事前に確認する、サービス・プロテクション機能を実現するための機能です。OSなどはこの機能を用いることで、ユーザから提供されるシステム・サービスに対するパラメータの正当性を検証することができます。また、ソフトウェアによる領域設定の読み出し、比較作業を繰り返すよりも短時間でこの検証処理を完了することができます。

## 10.1 レジスタ・セット

メモリ保護設定チェック機能のレジスタを次に示します。

表10-1 システム・レジスタ・バンク

グループ	プロセッサ保護機能 (10H)				
バンク	プロセッサ保護違反 (00H)		プロセッサ保護設定 (01H)		ソフトウェア・ページング (10H)
バンク・ラベル	MPV, PROT00		MPU, PROT01		PROT10
レジスタ番号	名称	機能	名称	機能	名称
0	VSECR	システム・レジスタ保護違反要因	MPM	プロセッサ保護動作モードの設定	MPM
1	VSTID	システム・レジスタ保護違反タスク識別子	MPC	プロセッサ保護コマンドの指定	MPC
2	VSADR	システム・レジスタ保護違反アドレス	TID	タスク識別子	TID
3	機能拡張用に予約		機能拡張用に予約		VMECR
4	VMECR	メモリ保護違反要因			VMTID
5	VMTID	メモリ保護違反タスク識別子			VMADR
6	VMADR	メモリ保護違反アドレス	IPA0L	命令 / 定数保護領域0下限アドレス	IPA0L
7	機能拡張用に予約		IPA0U	命令 / 定数保護領域0上限アドレス	IPA0U
8			IPA1L	命令 / 定数保護領域1下限アドレス	IPA1L
9			IPA1U	命令 / 定数保護領域1上限アドレス	IPA1U
10			IPA2L	命令 / 定数保護領域2下限アドレス	IPA2L
11			IPA2U	命令 / 定数保護領域2上限アドレス	IPA2U
12			IPA3L	命令 / 定数保護領域3下限アドレス	IPA3L
13			IPA3U	命令 / 定数保護領域3上限アドレス	IPA3U
14			IPA4L	命令 / 定数保護領域4下限アドレス	IPA4L
15			IPA4U	命令 / 定数保護領域4上限アドレス	IPA4U
16			DPA0L	データ保護領域0下限アドレス (スタック用)	DPA0L
17			DPA0U	データ保護領域0上限アドレス (スタック用)	DPA0U
18			DPA1L	データ保護領域1下限アドレス	DPA1L
19			DPA1U	データ保護領域1上限アドレス	DPA1U
20			DPA2L	データ保護領域2下限アドレス	DPA2L
21			DPA2U	データ保護領域2上限アドレス	DPA2U
22			DPA3L	データ保護領域3下限アドレス	DPA3L
23			DPA3U	データ保護領域3上限アドレス	DPA3U
24	MCA	メモリ保護設定チェック・アドレス	DPA4L	データ保護領域4下限アドレス	DPA4L
25	MCS	メモリ保護設定チェック・サイズ	DPA4U	データ保護領域4上限アドレス	DPA4U
26	MCC	メモリ保護設定チェック・コマンド	DPA5L	データ保護領域5下限アドレス (2 <sup>nd</sup> 指定のみ)	DPA5L
27	MCR	メモリ保護設定チェック結果	DPA5U	データ保護領域5上限アドレス (2 <sup>nd</sup> 指定のみ)	DPA5U

### 10.1.1 MCA - メモリ保護設定チェック・アドレス

メモリ保護設定のチェックを行う領域のベース・アドレスを指定します。

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MCA	MCA	MCA	MCA	MCA	MCA	MCA	MCA	MCA	MCA	MCA	MCA	MCA	MCA	MCA	MCA	MCA
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	MCA	MCA	MCA	MCA	MCA	MCA	MCA	MCA	MCA	MCA	MCA	MCA	MCA	MCA	MCA	MCA
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

初期値  
不定

ビット位置	ビット名	意味
31-0	MCA31- MCA0	メモリ保護設定のチェックを行う対象のメモリ領域の先頭アドレスをバイト単位で指定します。

### 10.1.2 MCS - メモリ保護設定チェック・サイズ

メモリ保護設定のチェックを行う領域のサイズを指定します。

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MCS	MCS	MCS	MCS	MCS	MCS	MCS	MCS	MCS	MCS	MCS	MCS	MCS	MCS	MCS	MCS	MCS
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	MCS	MCS	MCS	MCS	MCS	MCS	MCS	MCS	MCS	MCS	MCS	MCS	MCS	MCS	MCS	MCS
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

初期値  
不定

ビット位置	ビット名	意味
31-0	MCS31- MCS0	メモリ保護設定のチェックを行う対象のメモリ領域のサイズを指定する、対象領域のサイズをバイト単位で指定します。指定されたサイズは符号なしの整数として扱うため、MCAレジスタの値からアドレス値が減少する方向へ領域のチェックを行うことができません。 MCSレジスタには00000000Hを設定しないでください。

### 10.1.3 MCC - メモリ保護設定チェック・コマンド

メモリ保護設定のチェックを開始するためのコマンド・レジスタです。

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MCC	MCC	MCC	MCC	MCC	MCC	MCC	MCC	MCC	MCC	MCC	MCC	MCC	MCC	MCC	MCC	MCC
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	MCC	MCC	MCC	MCC	MCC	MCC	MCC	MCC	MCC	MCC	MCC	MCC	MCC	MCC	MCC	MCC
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

初期値 00000000H

ビット位置	ビット名	意味
31-0	MCC31-MCC0	MCCレジスタへの任意の値を書き込むと、メモリ保護設定のチェックが開始されます。事前にMCA/MCSレジスタを設定し、このレジスタへの書き込み操作を行うことで、MCRに結果が格納されます。  任意の書き込み値で、チェックを開始するため、r0をソース・レジスタとして、余分なレジスタを使用することなく、チェックを開始できます。また、チェックは、PSW.DMP、IMPビットの状態にかかわらず、各領域設定に従った結果を反映します。  MCCレジスタからの読み出し値は、常に00000000Hとなります。

### 10.1.4 MCR - メモリ保護設定チェック結果

メモリ保護設定のチェックの結果を格納するレジスタです。

ビット31-9, 7-5, 3には、必ず0を設定してください。

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MCR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	0	0	0	0	0	0	OV	0	0	0	SE	0	WE	RE	XE

初期値 00000000H

ビット位置	ビット名	意味
8	OV	指定された領域が00000000Hまたは、7FFFFFFFHをまたがる場合に、1が格納されます。それ以外の場合は、0が格納されます。
4	SE	指定された領域が、いずれか1つの保護領域の中に収まっており、かつその保護領域のスタック保護設定が有効であった場合に、1が格納されます。それ以外の場合は、0が格納されます。
2	WE	指定された領域が、いずれか1つの保護領域の中に収まっており、かつその保護領域がライト許可であった場合に、1が格納されます。それ以外の場合は、0が格納されます。
1	RE	指定された領域が、いずれか1つの保護領域の中に収まっており、かつその保護領域がリード許可であった場合に、1が格納されます。それ以外の場合は0が格納されます。
0	XE	指定された領域が、いずれか1つの保護領域の中に収まっており、かつその保護領域が実行許可であった場合に、1が格納されます。それ以外の場合は、0が格納されます。

メモリ保護設定チェック機能の使用例を次に示します。チェック対象の領域の指定が00000000Hをまたぐ場合、領域指定が誤っていると判断し、MCR.OVビットがセット(1)されます。このため、チェック結果を参照する場合には、必ずMCR.OVビットを確認し、結果が不正でないことを確認(OV = 0)であることを確認してから、その他のチェック結果を利用してください。

```
_service_protection:
...
ori    0x1000, r0, r12
ldsr   r12, BSEL          // MPV/PROT00バンクに切り替え
...
mov    ADDRESS, r10      // チェックする領域の先頭アドレスをr10に格納
mov    SIZE, r11         // チェックする領域のサイズをr11に格納
di
ldsr   r10, MCA          // アドレスをセット
ldsr   r11, MCS          // サイズをセット
ldsr   r0, MCC           // チェック開始
stsr   MCR, r12         // 結果の取得
ei
andi   0x0100, r12, r0
be     _overflow        // OV=1の場合は入力値不正として処理
br     _result_check    // それ以外の場合は結果を判定
```

## 第11章 特殊機能

この章では、プロセッサ保護機能に関する特殊機能について説明します。

### 11.1 メモリ保護設定の一括クリア

MPC.ALDSビットをセット(1)することで、セットした次のサイクルに、次のメモリ保護設定ビットが、一括でクリア(0)されます。

- IPAnL.Eビット (n = 0-4)
- DPAnL.Eビット (n = 0-5)

---

## 第 4 編 浮動小数点演算機能

# 第1章 概 説

V850E2M CPUの浮動小数点ユニット (FPU) はV850E2v3アーキテクチャに準拠し、CPUのコプロセッサとして動作し、浮動小数点演算命令を実行します。

単精度 (32ビット)、倍精度 (64ビット) のどちらのデータも使用できます。また、浮動小数点値と整数値の変換も可能です。

V850E2M CPUのFPUはANSI/IEEE標準規格754-1985「IEEE 2進浮動小数点演算規格」に準拠しています。

## 1.1 特 徴

### (1) 浮動小数点演算命令

- 命令数 : 73
- ANSI/IEEE標準規格754-1985「IEEE 2進浮動小数点演算規格」に準拠
- 単精度 (32ビット)、倍精度 (64ビット) をサポート
- 基本的な加減乗除算命令、積和命令、最大/最小命令、平方根命令をサポート
- 浮動小数点設定 / 状態レジスタの条件ビットをPSWレジスタのZフラグに転送する、フラグ転送命令をサポート
  - TRFSR
- 条件分岐の速度改善のために、条件付き転送命令をサポート
  - CMOVF.S, CMOVF.D
- 符号なし整数との型変換を効率よく実行する、符号なし変換命令をサポート
- 最近接整数への型変換を効率よく実行する、CEIL命令、FLOOR命令をサポート
- 浮動小数点の比較結果を格納する8ビットの条件ビットをサポート
- FPUの実行モードとしてプレサイス・モードとインプレサイス・モードをサポート

### (2) レジスタ・セット

- 浮動小数点レジスタ : 汎用レジスタを使用  
(浮動小数点演算専用のレジスタはありません)。
- 浮動小数点システム・レジスタ : FPSR - 浮動小数点演算の設定 / ステータス  
FPEPC - 浮動小数点演算例外プログラム・カウンタ  
FPST - 浮動小数点のステータス  
FPCC - 浮動小数点演算の比較結果  
FPCFG - 浮動小数点機能の設定  
FPEC - 浮動小数点演算例外の制御



## 1.2 浮動小数点演算機能の搭載 / 非搭載

V850E2M CPUでは、浮動小数点演算機能をコプロセッサとして位置づけているため、浮動小数点機能は製品により次のように搭載 / 非搭載が異なります。詳細は、各製品のマニュアルを参照してください。

### (1) 非搭載

浮動小数点演算機能を搭載しない場合は、すべての浮動小数点演算命令は使用不可となり、実行しようとした場合、コプロセッサ使用不可例外が発生します。また、すべての浮動小数点システム・レジスタも動作不定となるため、LDSR/STSRによる操作は行わないでください。

### (2) 単精度 / 倍精度搭載

単精度 / 倍精度の浮動小数点演算機能を搭載する場合は、すべての浮動小数点演算命令が使用可能です。また、すべての浮動小数点システム・レジスタは、**第2章 レジスタ・セット**で示される機能を提供します。

## 第2章 レジスタ・セット

### 2.1 浮動小数点レジスタ

FPUはCPUの汎用レジスタ（r0-r31）を使用します。浮動小数点演算専用のレジスタ・ファイルはありません。

・単精度浮動小数点演算命令：

32個の32ビット・レジスタを指定できます。これは汎用レジスタのr0-r31に相当します。

・倍精度浮動小数点演算命令：

16個の64ビット・レジスタを指定できます。これは汎用レジスタを1対ずつ使用するレジスタ・ペア（{r1, r0}, {r3, r2} ... {r31, r30}）に相当します。レジスタ・ペアは命令形式上、偶数レジスタで指定します。r0がゼロ・レジスタ（常に0を保持）であるので、原則として{r1, r0}は倍精度浮動小数点演算命令では使用するべきではありません。

### 2.2 浮動小数点システム・レジスタ

28個の制御レジスタがシステム・レジスタ・バンク上のFPUステータス・バンク（グループ#20-バンク番号00Hのバンク）にあります。FPUステータス・バンクはLDSR命令でBSELレジスタに0x2000を設定することにより選択されます。

FPUでは6個のシステム・レジスタが使用できます。

- ・FPSR： 例外の制御と監視を行います。また、比較演算の結果を保持し、FPUの動作モードを設定します。条件コード、例外モード、非正規化数フラッシュ許可、丸めモード制御、原因、例外許可、保存の各ビットがあります。
- ・FPEPC： 浮動小数点演算例外が発生した命令のプログラム・カウンタが格納されます。
- ・FPST： FPSR.XC, XP, PRビットと同一の内容を示します。
- ・FPCC： FPSR.CC(7:0)ビットと同一の内容を示します。
- ・FPCFG： FPSR.RM, XEビットと同一の内容を示します。
- ・FPEC： FPU例外の保留状態の確認、取り下げ等の制御を行います。

上記以外のFPUバンクのシステム・レジスタは、将来の拡張のために予約されています。書き込みは禁止です。また、読み出し値は不定です。システム・レジスタへのアクセスは、LDSR, STSR命令またはTRFSR命令によって可能です。

表2-1に、システム・レジスタ・バンクの構成を示します。システム・レジスタ番号28-31はバンク共通のシステム・レジスタで、BSELレジスタの設定値に関係なく、CPU機能バンクのEIWR, FEWR, DBWR<sup>注</sup>, BSELレジスタが参照されます。

注 DBWR レジスタは、開発ツール向けのデバッグ機能で使われます。

表2-1 システム・レジスタ・バンク

システム・レジスタ番号	システム・レジスタ名	オペランド指定の可否		システム・レジスタ保護
		LDSR命令	STSR命令	
0-5	(将来の機能拡張のための予約番号 (アクセスした場合の動作は保証しません))	×	×	
6	FPSR - 浮動小数点演算の設定 / ステータス			
7	FPEPC - 浮動小数点演算例外プログラム・カウンタ			
8	FPST - 浮動小数点のステータス			×
9	FPCC - 浮動小数点演算の比較結果			×
10	FPCFG - 浮動小数点機能の設定			×
11	FPEC - 浮動小数点演算例外の制御			
12-26	(将来の機能拡張のための予約番号 (アクセスした場合の動作は保証しません))	×	×	
28	EIWR - EIレベル例外用作業レジスタ			
29	FEWR - FEレベル例外用作業レジスタ			
30	DBWR <sup>#</sup> - DBレベル例外用作業レジスタ			
31	BSEL - レジスタ・バンクの選択			

注 DBWR レジスタは、開発ツール向けのデバッグ機能で使います。

備考 : オペランド指定の可否の場合は指定可能。システム・レジスタ保護の場合は、保護対象であることを示します。

× : オペランド指定の可否の場合は指定不可能。システム・レジスタ保護の場合は、保護対象ではないことを示します。

### 2.2.1 FPSR - 浮動小数点演算の設定 / ステータス

FPSRレジスタは、浮動小数点演算の実行状態や例外の発生を示します。

例外については、第2編 第6章 例外を参照してください。

ビット23, 22は、将来の機能拡張のために予約されており、0以外の書き込みを禁止します。0以外の値を書き込んだ場合の動作は不定です。また、読み出した場合の値は不定です。

(1/2)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
FPSR	CC7	CC6	CC5	CC4	CC3	CC2	CC1	CC0	0	0	DEM	SEM	RM		FS	PR	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	原因ビット(XC)						許可ビット(XE)						保存ビット(XP)				初期値 注
	E	V	Z	O	U	I	V	Z	O	U	I	V	Z	O	U	I	

ビット位置	ビット名	意 味														
31-24	CC(7:0)	CC (コンディション) ビットです。浮動小数点比較命令の結果がストアされます。CC(7:0) ビットは、比較命令とLDSR命令以外の影響を受けません。初期値は不定です。 0 : 比較結果が偽 1 : 比較結果が真														
21	DEM	倍精度演算例外モードです。DEMビットが1の場合、倍精度 (Double) 命令の実行により発生した例外は、プレサイス例外として扱います。倍精度命令については <b>4.2 浮動小数点演算命令の概要</b> を参照してください。初期値は0です。														
20	SEM	単精度演算例外モードです。SEMビットが1の場合、単精度 (Single) 命令の実行により発生した例外は、プレサイス例外として扱います。単精度命令については <b>4.2 浮動小数点演算命令の概要</b> を参照してください。初期値は0です。														
19, 18	RM	丸めモード制御ビットです。RMビットは、FPUがすべての浮動小数点演算命令で使用する丸めモードを規定します。初期値は0です。RMビットには必ず“00”を設定してください。 <table border="1" style="width:100%; border-collapse: collapse; margin-top: 10px;"> <thead> <tr> <th colspan="2">RMビット</th> <th rowspan="2">二モニック</th> <th rowspan="2">説 明</th> </tr> <tr> <th>ビット19</th> <th>ビット18</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">0</td> <td style="text-align: center;">RN</td> <td>表現可能な最も近い値に結果を丸めます。2つの表現可能な値の間である場合は、最下位ビットが0の方に結果を丸めます。</td> </tr> <tr> <td colspan="2" style="text-align: center;">上記以外</td> <td colspan="2" style="text-align: center;">設定禁止</td> </tr> </tbody> </table>	RMビット		二モニック	説 明	ビット19	ビット18	0	0	RN	表現可能な最も近い値に結果を丸めます。2つの表現可能な値の間である場合は、最下位ビットが0の方に結果を丸めます。	上記以外		設定禁止	
RMビット		二モニック	説 明													
ビット19	ビット18															
0	0	RN	表現可能な最も近い値に結果を丸めます。2つの表現可能な値の間である場合は、最下位ビットが0の方に結果を丸めます。													
上記以外		設定禁止														

**注** 各ビットの説明を参照してください。

17	FS	<p>正規化できない値（ディノーマル数）のフラッシュを許可するビットです。FSビットがセットされているとき、ディノーマル数の結果は未実装演算例外（E）を起こさず、フラッシュされます。フラッシュされたものが0になるか最小正規化値になるかは、丸めモードによって決まります。初期値は1です。FSビットには、必ず1に設定してください。</p> <table border="1"> <thead> <tr> <th>ディノーマル数の結果</th> <th>フラッシュされる結果の丸めモード（RN）</th> </tr> </thead> <tbody> <tr> <td>正</td> <td>+0</td> </tr> <tr> <td>負</td> <td>-0</td> </tr> </tbody> </table>	ディノーマル数の結果	フラッシュされる結果の丸めモード（RN）	正	+0	負	-0
ディノーマル数の結果	フラッシュされる結果の丸めモード（RN）							
正	+0							
負	-0							
16	PR	浮動小数点演算例外の発生要因となった命令の例外モードがインプレサイズ例外であればクリア（0）され、プレサイズ例外であればセット（1）されます。初期値は不定です。						
15-10	XC (E,V,Z,O,U,I)	原因ビットです。初期値は不定です。詳細は2.2.1(1)原因ビット(XC)を参照してください。						
9-5	XE (V,Z,O,U,I)	許可ビットです。初期値は0です。詳細は2.2.1(2)許可ビット(XE)を参照してください。						
4-0	XP (V,Z,O,U,I)	保存ビットです。初期値は不定です。詳細は2.2.1(3)保存ビット(XP)を参照してください。						

### (1) 原因ビット (XC)

FPSRレジスタのビット15-ビット10は原因ビットで、浮動小数点演算例外の発生とその要因を示します。IEEE754で定義されている例外が起き、その例外に対応する許可ビットがセット（1）されていた場合、原因ビットをセットし、例外が発生します。1つの命令で2つ以上の例外を検出した場合、それぞれのビットがセット（1）されます。

2つ以上の例外を検出した場合、いずれかの例外に対応する許可ビットがセット（1）されていれば、例外が発生します。この場合、許可ビットがクリア（0）されている例外を含め、検出したすべての例外の原因ビットがセット（1）されます。

原因ビットは、浮動小数点演算例外が発生した浮動小数点演算命令（TRFSR命令を除く）によって書き換えられます。Eビットは、ソフトウェアのエミュレートが必要な場合にセット（1）され、それ以外の場合はクリア（0）されます。そのほかのビットは、IEEE754で定義されている例外が発生したかどうかによりクリア（0）もしくはセット（1）されます。

浮動小数点演算例外が発生した場合、演算結果はストアされず、原因ビットだけが影響を受けます。

LDSR命令により原因ビットをセット（1）しても、浮動小数点演算例外は発生しません。

### (2) 許可ビット (XE)

FPSRレジスタのビット9-ビット5は許可ビットで、浮動小数点演算例外の発生を許可します。IEEE754で定義されている例外が起きたとき、例外に対応する許可ビットがセット（1）されていれば、浮動小数点演算例外が発生します。

未実装演算例外（E）に対応する許可ビットはありません。未実装演算例外（E）は、常に浮動小数点演算例外が発生します。

対応する許可ビットがセット（1）されていない場合、例外は発生せず、IEEE754によって定義されたデフォルトの結果がストアされます。

**(3) 保存ビット (XP)**

FPSRレジスタのビット4-ビット0は保存ビットで、リセット後、検出した例外を蓄積、表示します。IEEE754で定義されている例外が発生し、対応する許可ビットがセット(1)されていない場合に、保存ビットがセット(1)され、そのほかの場合は変化しません。保存ビットは、浮動小数点オペレーションではクリア(0)されません。しかし、LDSR命令を使用してFPSRレジスタに新たな値を書き込むことで、ソフトウェアによるセット/クリアができます。

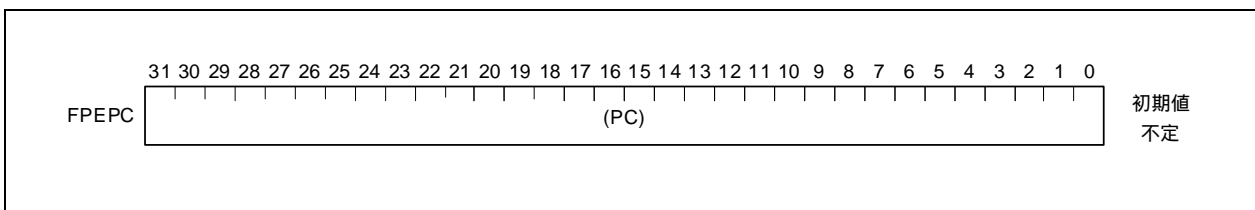
未実装演算例外(E)に対応する保存ビットはありません。未実装演算例外(E)は、常に浮動小数点演算例外が発生します。

**備考** 例外の種類ごとの各ビットの対応関係については、**図5-1 FPSRレジスタの原因/許可/保存ビット**を参照してください。

**2.2.2 FPEPC - 浮動小数点演算例外プログラム・カウンタ**

許可ビットによって許可されている例外が発生した場合、例外が発生した命令のプログラム・カウンタ(PC)が格納されます。

ビット0は、0に固定されています。



**注意** 製品仕様により、命令アドレッシング範囲が512 Mバイトに制限されたCPUでは、FPEPCのビット31-29はビット28を符号拡張した値が自動的に設定されます。

### 2.2.3 FPST - 浮動小数点演算のステータス

FPSR.PR, XC, XPビットと同一の内容を示します。

ビット31-16, 14, 7-5は, 0以外の書き込みを禁止します。0以外の値を書き込んだ場合の動作は不定です。

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FPST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PR	0			原因ビット(XC)					0	0	0		保存ビット(XP)			
		E	V	Z	O	U	I					V	Z	O	U	I

初期値  
不定

ビット位置	ビット名	意味
15	PR	浮動小数点演算例外の発生要因となった命令の例外モードを示します。初期値は不定です。 また、このビットへの書き込みは FPSR.PR へ反映されます。 0: インプレサイズ例外 1: プレサイズ例外
13-8	XC	原因ビットです。初期値は不定です。詳細は2.2.1(1)原因ビット(XC)を参照してください。また、このビットへの書き込みは FPSR.XCビットへ反映されます。
4-0	XP	保存ビットです。初期値は不定です。詳細は2.2.1(3)保存ビット(XP)を参照してください。また、このビットへの書き込みは FPSR.XPビットへ反映されます。

### 2.2.4 FPCC - 浮動小数点演算の比較結果

FPSR.CC(7:0)ビットと同一の内容を示します。

ビット31-8は, 0以外の書き込みを禁止します。0以外の値を書き込んだ場合の動作は不定です。

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FPCC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	0	0	0	0	0	0	0	CC7	CC6	CC5	CC4	CC3	CC2	CC1	CC0

初期値  
不定

ビット位置	ビット名	意味
7-0	CC(7:0)	CC (コンディション) ビットです。浮動小数点比較命令の結果がストアされます。CC(7:0)ビットは、比較命令とLDSR命令以外の影響を受けません。初期値は不定です。また、このビットへの書き込みは FPSR.CC(7:0) ビットへ反映されます。 0: 比較結果が偽 1: 比較結果が真

## 2.2.5 FPCFG - 浮動小数点演算の設定

FPSR.RM, XEビットと同一の内容を示します。

ビット31-10, 7-5は, 0以外の書き込みを禁止します。0以外の値を書き込んだ場合の動作は不定です。

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FPCFG	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	0	0	0	0	0	RM	0	0	0	許可ビット(XE)				初期値	
											V	Z	O	U	I	00000000H

ビット位置	ビット名	意味																
9, 8	RM	<p>丸めモード制御ビットです。RMビットは, FPUがすべての浮動小数点演算命令で使用する丸めモードを規定します。初期値は0です。RMビットには, 必ず“00”を設定してください。また, このビットへの書き込みは FPSR.RMビットへ反映されます。</p> <table border="1"> <thead> <tr> <th colspan="2">RMビット</th> <th>二モニック</th> <th>説明</th> </tr> <tr> <th>ビット9</th> <th>ビット8</th> <td></td> <td></td> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>RN</td> <td>表現可能な最も近い値に結果を丸めます。2つの表現可能な値の間である場合は, 最下位ビットが0の方に結果を丸めます。</td> </tr> <tr> <td colspan="2">上記以外</td> <td colspan="2">設定禁止</td> </tr> </tbody> </table>	RMビット		二モニック	説明	ビット9	ビット8			0	0	RN	表現可能な最も近い値に結果を丸めます。2つの表現可能な値の間である場合は, 最下位ビットが0の方に結果を丸めます。	上記以外		設定禁止	
RMビット		二モニック	説明															
ビット9	ビット8																	
0	0	RN	表現可能な最も近い値に結果を丸めます。2つの表現可能な値の間である場合は, 最下位ビットが0の方に結果を丸めます。															
上記以外		設定禁止																
4-0	XE (V,Z,O,U,I)	許可ビットです。初期値は0です。詳細は2.2.1(2)許可ビット(XE)を参照してください。また, このビットへの書き込みは FPSR.XEビットへ反映されます。																



## 2.2.6 FPEC - 浮動小数点演算例外制御

浮動小数点演算例外に関する制御を行うレジスタです。

ビット31-1は、0以外の書き込みを禁止します。0以外の値を書き込んだ場合の動作は不定です。

**注意** FPEC レジスタの取り扱いについては、第2編 6.3 例外の管理を参照してください。

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FPEC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	FPI VD

初期値  
00000000H

ビット位置	ビット名	意味
0	FPIVD <sup>注</sup>	<p>FPI例外の通知状況を示します。</p> <p>このビットがセット（1）されている場合、CPUに対してFPI例外を通知していて、かつFPI例外が受け付けられていない状態を示します。CPUがFPI例外を受け付けた時点で、このビットは自動的にクリア（0）されます。</p> <p>また、このビットがセット（1）されている間は、すべての浮動小数点演算命令を無効化します。</p> <p>このビットがセット（1）されている状態から、LDSR命令によってクリア（0）することで、FPI例外の通知を取り下げることができます。FPI例外の通知を取り下げると、CPUがFPI例外を受け付けることはありません。</p> <p>0：FPI例外非通知状態（FPI例外の通知を行っていません）。</p> <p>1：FPI例外通知状態（FPI例外の通知を行っています）。</p>

**注** FPIVDビットに対するLDSR命令による書き込み操作は、クリア（0）のみ可能です。セット（1）は行えません。

## 第3章 データ・タイプ

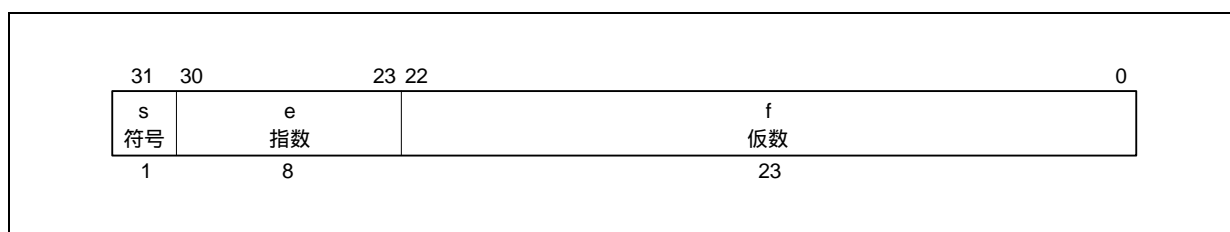
### 3.1 データ形式

#### 3.1.1 浮動小数点の形式

FPUは、32ビット（単精度）と64ビット（倍精度）のIEEE754浮動小数点演算をサポートします。

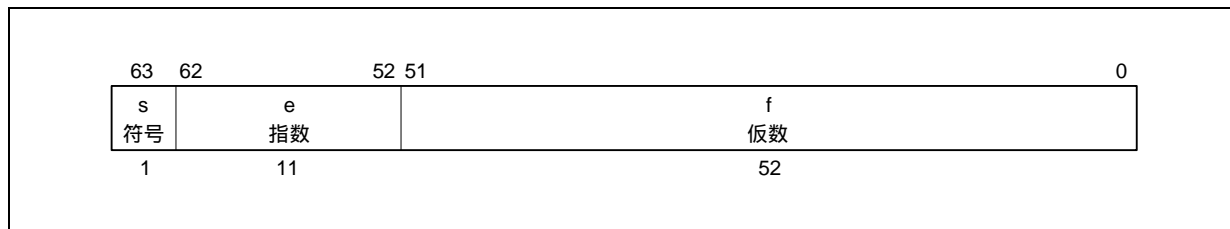
単精度浮動小数点形式は、図3 - 1に示すように24ビットの符号付き仮数部（s+f）と、8ビットの指数部（e）で構成されます。

図3 - 1 単精度浮動小数点形式



倍精度浮動小数点形式は、図3 - 2に示すように53ビットの符号付き仮数部（s+f）と、11ビットの指数部（e）で構成されます。

図3 - 2 倍精度浮動小数点形式



浮動小数点形式の数値は、次の3つの領域により構成されます。

- ・ 符号ビット : s
- ・ 指数部 :  $e = E + \text{バイアス値}$
- ・ 仮数部 :  $f = .b_1b_2\dots b_{P-1}$  (小数点第1位以下の値)

単精度形式の場合、バイアス値は127です。倍精度形式の場合、バイアス値は1023です。

バイアスしていない指数値Eの範囲は、 $E_{\min}$ から $E_{\max}$ までのすべての整数値と2つの予約値、 $E_{\min} - 1$  ( $\pm 0$ ,あるいはディノーマル数)と、 $E_{\max} + 1$  ( $\pm \infty$ ,あるいはNaN:非数)です。単精度と倍精度の形式によって、0以外の数値の表現は、1つの形式で表現されます。

この形式で表現される数値(v)は、表3 - 1に示す式によって求められます。

表3 - 1 浮動小数点値の計算式

種 類	計 算 式
NaN (非数)	$E = E_{\max} + 1$ かつ $f = 0$ ならば $v$ は $s$ にかかわらず NaN
$\pm$ (無限大数)	$E = E_{\max} + 1$ かつ $f = 0$ ならば $v = (-1)^s$
ノーマル数 (正規化数)	$E_{\min} \leq E \leq E_{\max}$ ならば $v = (-1)^s 2^E (1.f)$
ディノーマル数 (非正規化数)	$E = E_{\min} - 1$ かつ $f \neq 0$ ならば $v = (-1)^s 2^{E_{\min}} (0.f)$
$\pm 0$ (ゼロ)	$E = E_{\min} - 1$ かつ $f = 0$ ならば $v = (-1)^s 0$

・ NaN (非数)

IEEE754では、NaN (Not a Number) という浮動小数点値を規定しています。これは非数とも呼ばれ、数値ではないため大小関係もありません。

すべての浮動小数点形式において、 $v$ がNaNであった場合、 $f$ の最上位ビットの値によってSignalingNaN (S-NaN) か、QuietNaN (Q-NaN) のどちらかになります。 $f$ の最上位ビットがセットされている場合はQuietNaNで、クリアされている場合はSignalingNaNです。

浮動小数点の形式で定義されている各パラメータの値を表3 - 2に示します。

表3 - 2 浮動小数点形式とパラメータ値

パラメータ	形 式	
	単精度	倍精度
$E_{\max}$	+ 127	+ 1023
$E_{\min}$	- 126	- 1022
指数部のバイアス値	+ 127	+ 1023
指数部の長さ (ビット数)	8	11
整数ビット	見えない	見えない
仮数部の長さ (ビット数)	23	52
形式の長さ (ビット数)	32	64

この浮動小数点形式で表現できる最小値、最大値を表3 - 3に示します。

表3 - 3 浮動小数点の最大値、最小値

タイプ	値
単精度浮動小数点の最小値	1.40129846e - 45
単精度浮動小数点の最小値 (ノーマル)	1.17549435e - 38
単精度浮動小数点の最大値	3.40282347e + 38
倍精度浮動小数点の最小値	4.9406564584124654e - 324
倍精度浮動小数点の最小値 (ノーマル)	2.2250738585072014e - 308
倍精度浮動小数点の最大値	1.7976931348623157e + 308

### 3.1.2 整数の形式

整数の値は2の補数の形式で保持されます。図3 - 3に32ビット整数形式、図3 - 4に64ビット整数形式を示します。符号なし整数においては、符号ビットは存在せず、全ビットが整数値を表現します。

図3 - 3 32ビット整数形式

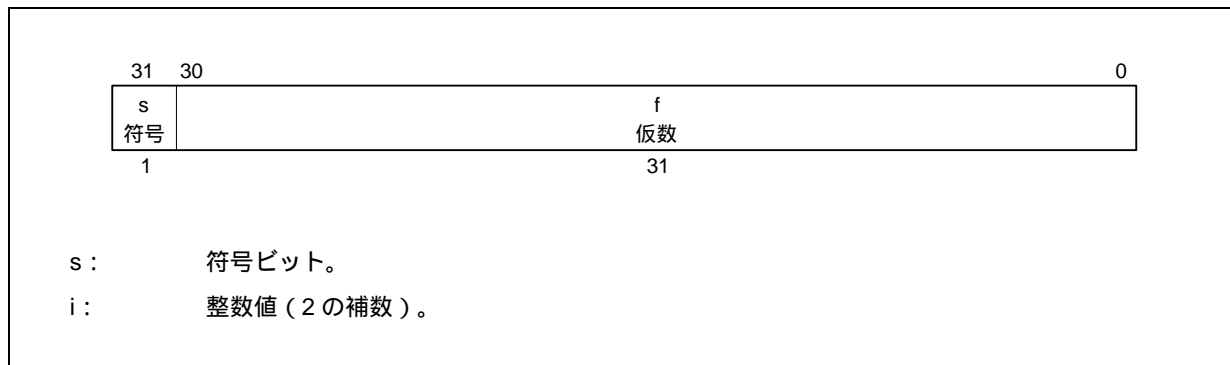
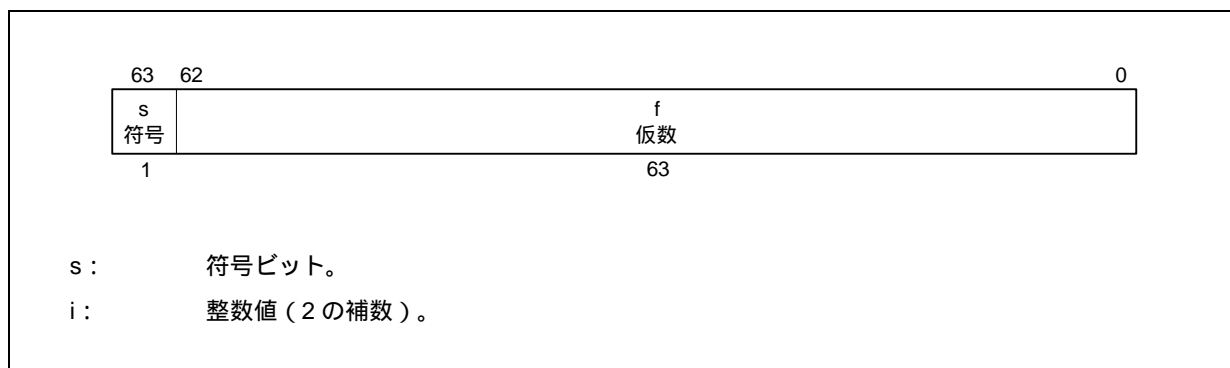


図3 - 4 64ビット整数形式



## 第4章 命令

### 4.1 命令フォーマット

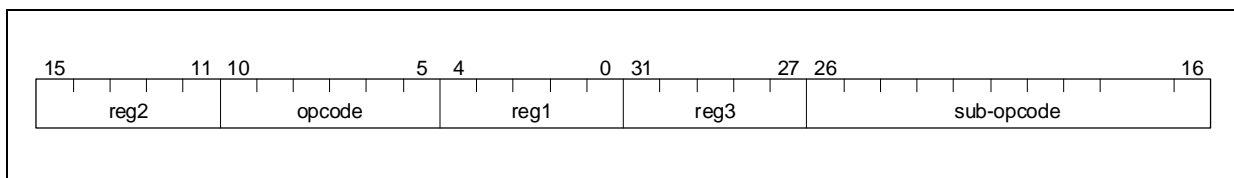
浮動小数点演算命令は、すべて32ビット・フォーマットです。

実際に命令がメモリに格納される時は、次のように配置されます。

- 各命令形式の下位部分（ビット0を含む） 下位アドレス側
- 各命令形式の上位部分（ビット15またはビット31を含む） 上位アドレス側

#### (1) Format F:l

6ビットのオペコード・フィールド、4ビットのサブ・オペコード・フィールド、3つの汎用レジスタ指定フィールド、3ビットのカテゴリ・フィールド、2ビットのタイプ・フィールドを持つ32ビット長浮動小数点演算命令形式。



## 4.2 浮動小数点演算命令の概要

浮動小数点演算命令は単精度命令 (Single) と倍精度命令 (Double) に分類され、次の命令 (ニモニック) があります。

### (1) 基本演算命令

- ABSF.D : Floating-point Absolute Value ( Double )
- ABSF.S : Floating-point Absolute Value ( Single )
- ADDF.D : Floating-point Add ( Double )
- ADDF.S : Floating-point Add ( Single )
- DIVF.D : Floating-point Divide ( Double )
- DIVF.S : Floating-point Divide ( Single )
- MAXF.D : Floating-point Maximum ( Double )
- MAXF.S : Floating-point Maximum ( Single )
- MINF.D : Floating-point Minimum ( Double )
- MINF.S : Floating-point Minimum ( Single )
- MULF.D : Floating-point Multiply ( Double )
- MULF.S : Floating-point Multiply ( Single )
- NEGF.D : Floating-point Negate ( Double )
- NEGF.S : Floating-point Negate ( Single )
- RECIPF.D : Reciprocal of a floating-point value ( Double )
- RECIPF.S : Reciprocal of a floating-point value ( Single )
- RSQRTF.D : Reciprocal of the square root of a floating-point value ( Double )
- RSQRTF.S : Reciprocal of the square root of a floating-point value ( Single )
- SQRTF.D : Floating-point Square Root ( Double )
- SQRTF.S : Floating-point Square Root ( Single )
- SUBF.D : Floating-point Subtract ( Double )
- SUBF.S : Floating-point Subtract ( Single )

### (2) 拡張基本演算命令

- MADDF.S : Floating-point Multiply-Add ( Single )
- MSUBF.S : Floating-point Multiply-Subtract ( Single )
- NMADDF.S : Floating-point Negate Multiply-Add ( Single )
- NMSUBF.S : Floating-point Negate Multiply-Subtract ( Single )

## (3) 変換命令

- CEILF.DL : Floating-point Truncate to Integer Format, rounded toward + ( Double )
- CEILF.DW : Floating-point Truncate to Integer Format, rounded toward + ( Double )
- CEILF.DUL : Floating-point Truncate to Unsigned Long, rounded toward + ( Double )
- CEILF.DUW : Floating-point Truncate to Unsigned Word, rounded toward + ( Double )
- CEILF.SL : Floating-point Truncate to Integer Format, rounded toward + ( Single )
- CEILF.SW : Floating-point Truncate to Integer Format, rounded toward + ( Single )
- CEILF.SUL : Floating-point Truncate to Unsigned Long, rounded toward + ( Single )
- CEILF.SUW : Floating-point Truncate to Unsigned Word, rounded toward + ( Single )
- CVTF.DL : Floating-point Convert to Integer Format ( Double )
- CVTF.DS : Floating-point Convert to Single Floating-point Format ( Double )
- CVTF.DUL : Floating-point Convert Double to Unsigned-Long ( Double )
- CVTF.DUW : Floating-point Convert Double to Unsigned-Word ( Double )
- CVTF.DW : Floating-point Convert to Integer Format ( Double )
- CVTF.LD : Floating-point Convert to Double Floating-point Format ( Double )
- CVTF.LS : Floating-point Convert to Single Floating-point Format ( Single )
- CVTF.SD : Floating-point Convert to Double Floating-point Format ( Double )
- CVTF.SL : Floating-point Convert to Integer Format ( Single )
- CVTF.SUL : Floating-point Convert Single to Unsigned-Long ( Single )
- CVTF.SUW : Floating-point Convert Single to Unsigned-Word ( Single )
- CVTF.SW : Floating-point Convert to Integer Format ( Single )
- CVTF.ULD : Floating-point Convert Unsigned-Long to Double ( Double )
- CVTF.ULS : Floating-point Convert Unsigned-Long to Single ( Single )
- CVTF.UWD : Floating-point Convert Unsigned-Word to Double ( Double )
- CVTF.UWS : Floating-point Convert Unsigned-Word to Single ( Single )
- CVTF.WD : Floating-point Convert to Double Floating-point Format ( Double )
- CVTF.WS : Floating-point Convert to Single Floating-point Format ( Single )
- FLOORF.DL : Floating-point Truncate to Integer Format, rounded toward - ( Double )
- FLOORF.DW : Floating-point Truncate to Integer Format, rounded toward - ( Double )
- FLOORF.DUL : Floating-point Truncate to Unsigned Long, rounded toward - ( Double )
- FLOORF.DUW : Floating-point Truncate to Unsigned Word, rounded toward - ( Double )
- FLOORF.SL : Floating-point Truncate to Integer Format, rounded toward - ( Single )
- FLOORF.SW : Floating-point Truncate to Integer Format, rounded toward - ( Single )
- FLOORF.SUL : Floating-point Truncate to Unsigned Long, rounded toward - ( Single )
- FLOORF.SUW : Floating-point Truncate to Unsigned Word, rounded toward - ( Single )
- TRNCF.DL : Floating-point Truncate to Integer Format, rounded to zero ( Double )
- TRNCF.DUL : Floating-point Truncate Double to Unsigned-Long ( Double )
- TRNCF.DUW : Floating-point Truncate Double to Unsigned-Word ( Double )
- TRNCF.DW : Floating-point Truncate to Integer Format, rounded to zero ( Double )
- TRNCF.SL : Floating-point Truncate to Integer Format, rounded to zero ( Single )
- TRNCF.SUL : Floating-point Truncate Single to Unsigned-Long ( Single )
- TRNCF.SUW : Floating-point Truncate Single to Unsigned-Word ( Single )

- TRNCF.SW : Floating-point Truncate to Integer Format, rounded to zero ( Single )

#### (4) 比較命令

- CMPF.S : Compares floating-point values ( Single )
- CMPF.D : Compares floating-point values ( Double )

#### (5) 条件付き転送命令

- CMOVF.S : Floating-point conditional move ( Single )
- CMOVF.D : Floating-point conditional move ( Double )

#### (6) 条件ビット転送命令

- TRFSR : transfers specified CC bit to Zero flag in PSW ( Single )

### 4.3 比較命令のための条件

浮動小数点の比較命令 (CMPF.D, CMPF.S) は、2つの浮動小数点データの比較演算を行います。結果は、データとコード中に含まれる比較条件に基づいて決定します。表4 - 1に比較命令で指定可能な条件のニモニックを示します。

比較命令の結果をTRFSR命令によりPSW (プログラム・ステータス・ワード) のZフラグに転送して、条件分岐を行う場合、条件の論理を反転して使用できます。表4 - 2は条件の真偽による論理反転を示しています。浮動小数点比較命令の4ビットの条件コードでは、表中の「真」欄の条件を指定します。条件分岐命令BTは比較の結果が真である場合に分岐し、BFは偽である場合に分岐します。



表4 - 1 比較命令のための条件一覧

二モニック	定 義	論理反転
F	常に偽	( T ) 常に真
UN	Unordered	( OR ) Ordered
EQ	等しい	( NEQ ) 等しくない
UEQ	Unorderedか等しい	( OLG ) Orderedで、より小さいか、より大きい
OLT	Orderedで、より小さい	( UGE ) Unorderedか、より大きいか、等しい
ULT	Unorderedか、より小さい	( OGE ) Orderedで、より大きいか、等しい
OLE	Orderedで、より小さいか、等しい	( UGT ) Unorderedか、より大きい
ULE	Unorderedか、より小さいか、等しい	( OGT ) Orderedで、より大きい
SF	Signalingで偽	( ST ) Signalingで真
NGLE	より大きくない、かつより小さくない、 かつ等しくない	( GLE ) より大きいか、より小さいか、等しい
SEQ	Signalingで等しい	( SNE ) Signalingで等しくない
NGL	より大きくない、かつより小さくない	( GL ) より大きいか、より小さい
LT	より小さい	( NLT ) より小さくない
NGE	より大きくない、かつ等しくない	( GE ) より大きいか、等しい
LE	より小さいか、等しい	( NLE ) より小さくない、かつ等しくない
NGT	より大きくない	( GT ) より大きい

表4 - 2 条件コードのビット定義と論理反転

二モ ニック ( 真 )	条件コード fcond		条件コードfcond[3-0]のビット定義				論理反転 ( 偽 )
			より小さい	等しい	Unordered	Unorderedのとき 無効演算例外の検出	
	10進	2進	fcond[2]	fcond[1]	fcond[0]	fcond[3]	
F	0	0b0000	F	F	F	しない	( T )
UN	1	0b0001	F	F	T	しない	( OR )
EQ	2	0b0010	F	T	F	しない	( NEQ )
UEQ	3	0b0011	F	T	T	しない	( OLG )
OLT	4	0b0100	T	F	F	しない	( UGE )
ULT	5	0b0101	T	F	T	しない	( OGE )
OLE	6	0b0110	T	T	F	しない	( UGT )
ULE	7	0b0111	T	T	T	しない	( OGT )
SF	8	0b1000	F	F	F	する	( ST )
NGLE	9	0b1001	F	F	T	する	( GLE )
SEQ	10	0b1010	F	T	F	する	( SNE )
NGL	11	0b1011	F	T	T	する	( GL )
LT	12	0b1100	T	F	F	する	( NLT )
NGE	13	0b1101	T	F	T	する	( GE )
LE	14	0b1110	T	T	F	する	( NLE )
NGT	15	0b1111	T	T	T	する	( GT )

## 4.4 命令セット

この節では、各命令の二モニックごとに（アルファベット順）、次の項目に分けて説明します。

- 命令形式 : 命令の記述方法、オペランドを示します（略号については、表4 - 3参照）。
- オペレーション : 命令の機能を示します（略号については、表4 - 4参照）。
- フォーマット : 命令形式を命令フォーマットで示します（4.1 命令フォーマット参照）。
- オペコード : 命令のオペコードをビット・フィールドで示します（略号については、表4 - 5参照）。
- 説明 : 命令の動作説明をします。
- 補足 : 命令の補足説明をします。

表4 - 3 命令形式の凡例

略 号	意 味
reg1	汎用レジスタ
reg2	汎用レジスタ
reg3	汎用レジスタ
reg4	汎用レジスタ
fcbit	浮動小数点比較命令の結果を格納するコンディション・ビットのビット・ナンバを指定
imm x	x ビット・イミディエト・データ
fcond	比較命令の比較条件の二モニックまたは条件コードを指定（詳細は4. 3 比較命令のための条件を参照してください）

表4-4 オペレーションの凡例

略号	意味
←	代入
GR []	汎用レジスタ
SR []	システム・レジスタ
result	結果をフラグに反映する。
+	加算
-	減算
	ビット連結
×	乗算
÷	除算
abs	絶対値
ceil	+ 方向への丸め
compare	比較
cvt	丸めモードに従う型変換
floor	- 方向への丸め
max	最大値
min	最小値
neg	符号反転
round	最も近い値への丸め
sqrt	平方根
trunc	ゼロ方向への丸め

表4-5 オペコードの凡例

略号	意味
R	reg1を指定するコードの1ビット分データ
r	reg2を指定するコードの1ビット分データ
w	reg3を指定するコードの1ビット分データ
W	reg4を指定するコードの1ビット分データ
l	イミューディアートの1ビット分データ(イミューディアートの上位ビットを示す)
i	イミューディアートの1ビット分データ
fff	浮動小数点比較命令の結果を格納するコンディション・ビットのビット・ナンバ(fcbits)を指定する3ビット・データ
FFFF	比較命令の比較条件の二モニクまたは条件コード(fcond)に対応する4ビット・データ

## &lt; 浮動小数点演算命令 &gt;

<b>ABSF.D</b>	Floating-point Absolute Value (Double)  浮動小数点絶対値 (倍精度)
---------------	--

[ 命令形式 ]            ABSF.D reg2, reg3

[ オペレーション ]        reg3 ← abs(reg2)

[ フォーマット ]        Format F: I

[ オペコード ]

15	11	10	5	4	0	31	27	26	25	23	22	21	20	17	16																
r	r	r	r	0	1	1	1	1	1	1	0	0	0	0	0	w	w	w	w	0	1	0	0	0	1	0	1	1	0	0	0
reg2										reg3					category	type	sub-op														

[ 説 明 ]            汎用レジスタ reg2 で指定されるレジスタ・ペアにある倍精度浮動小数点形式の内容の絶対値をとり、汎用レジスタ reg3 で指定されるレジスタ・ペアに格納します。  
絶対値演算は算術的に行います。すなわち、オペランドが S-NaN の場合は、IEEE754 の無効演算例外 (V) を検出します。

[ 浮動小数点演算例外 ] 無効演算例外 (V)  
不正確演算例外 (I)

[ 演算結果 ]            FPSR.FS = 1 の場合

reg2	+Normal	-Normal	+0	-0	+	-	Q-NaN	S-NaN
演算結果 [例外]	+Normal		+0		+		Q-NaN	Q-NaN[V]

備考 1. [ ] は必ず発生する例外です。

2. ディノーマル数はフラッシュされ“0”として扱われます。

## &lt; 浮動小数点演算命令 &gt;

<b>ABSF.S</b>	Floating-point Absolute Value (Single)  浮動小数点絶対値 (単精度)
---------------	--

[ 命令形式 ]            ABSF.S reg2, reg3

[ オペレーション ]        reg3 ← abs(reg2)

[ フォーマット ]         Format F: I

[ オペコード ]

	15		11	10		5	4		0	31		27	26	25		23	22	21	20		17	16										
	r	r	r	r	r	1	1	1	1	1	1	0	0	0	0	0	w	w	w	w	w	1	0	0	0	1	0	0	1	0	0	0
	reg2										reg3					category	type	sub-op														

[ 説 明 ]                汎用レジスタ reg2 にある単精度浮動小数点形式の内容の絶対値をとり、汎用レジスタ reg3 に格納します。

絶対値演算は算術的に行います。すなわち、オペランドが S-NaN の場合は、IEEE754 の無効演算例外 (V) を検出します。

[ 浮動小数点演算例外 ] 無効演算例外 (V)  
不正確演算例外 (I)

[ 演算結果 ]            FPSR.FS = 1 の場合

reg2	+Normal	-Normal	+0	-0	+	-	Q-NaN	S-NaN
演算結果 [例外]	+Normal		+0		+		Q-NaN	Q-NaN[V]

備考 1. [ ] は必ず発生する例外です。

2. ディノーマル数はフラッシュされ "0" として扱われます。

## &lt; 浮動小数点演算命令 &gt;

ADDF.D	Floating-point Add (Double)  浮動小数点加算 (倍精度)
--------	--

[ 命令形式 ]           ADDF.D reg1, reg2, reg3

[ オペレーション ]       reg3 ← reg2 + reg1

[ フォーマット ]       Format F: I

[ オペコード ]

15	11	10	5	4	0	31	27	26	25	23	22	21	20	17	16																
r	r	r	r	0	1	1	1	1	1	1	R	R	R	R	0	w	w	w	w	0	1	0	0	0	1	1	1	0	0	0	0
reg2					reg1					reg3					category		type		sub-op												

[ 説 明 ]               汎用レジスタ reg2 で指定されるレジスタ・ペアにある倍精度浮動小数点形式の内容と汎用レジスタ reg1 で指定されるレジスタ・ペアにある倍精度浮動小数点形式の内容を加算し、汎用レジスタ reg3 で指定されるレジスタ・ペアに格納します。演算は無限精度であるかのように実行し、結果を現在の丸めモードに従って丸めます。

[ 浮動小数点演算例外 ] 無効演算例外 (V)

不正確演算例外 (I)

オーバフロー例外 (O)

アンダフロー例外 (U)

[ 演算結果 ]           FPSR.FS = 1 の場合

reg2(B) reg1(A)	+Normal	-Normal	+0	-0	+	-	Q-NaN	S-NaN
+Normal	A + B				+	-	Q-NaN	S-NaN
-Normal								
+0								
-0								
+					+	Q-NaN[V]		
-						Q-NaN[V]	-	
Q-NaN							Q-NaN	
S-NaN							Q-NaN[V]	

備考 1. [ ] は必ず発生する例外です。

2. ディノーマル数はフラッシュされ“0”として扱われます。

## &lt; 浮動小数点演算命令 &gt;

ADDF.S	Floating-point Add (Single)  浮動小数点加算 (単精度)
--------	--

[ 命令形式 ]           ADDF.S reg1, reg2, reg3

[ オペレーション ]       reg3 ← reg2 + reg1

[ フォーマット ]       Format F: I

[ オペコード ]

	15		11	10		5	4		0	31		27	26	25		23	22	21	20		17	16										
	r	r	r	r	r	1	1	1	1	1	1	R	R	R	R	R	w	w	w	w	w	1	0	0	0	1	1	0	0	0	0	0
	reg2										reg1					reg3					category			type		sub-op						

[ 説 明 ]               汎用レジスタ reg2 にある単精度浮動小数点形式の内容と汎用レジスタ reg1 にある単精度浮動小数点形式の内容を加算し、汎用レジスタ reg3 に格納します。演算は無限精度であるかのように実行し、結果を現在の丸めモードに従って丸めます。

[ 浮動小数点演算例外 ] 無効演算例外 (V)

不正確演算例外 (I)

オーバフロー例外 (O)

アンダフロー例外 (U)

[ 演算結果 ]           FPSR.FS = 1 の場合

reg2(B) reg1(A)	+Normal	-Normal	+0	-0	+	-	Q-NaN	S-NaN
+Normal	A + B				+	-	Q-NaN	S-NaN
-Normal								
+0								
-0								
+					+	Q-NaN[V]		
-					Q-NaN[V]	-		
Q-NaN	Q-NaN							
S-NaN								Q-NaN[V]

備考 1. [ ] は必ず発生する例外です。

2. ディノーマル数はフラッシュされ“0”として扱われます。



## &lt; 浮動小数点演算命令 &gt;

CEILF.DL	Floating-point Ceiling to Integer Format (Double) 整数形式への変換 (倍精度)
----------	---

[ 命令形式 ]            CEILF.DL reg2, reg3

[ オペレーション ]        reg3 ← ceil reg2 (double    long-word)

[ フォーマット ]         Format F: I

15	11 10	5 4	0 31	27 26 25	23 22 21 20	17 16
r r r r 0	1 1 1 1 1 1	0 0 0 1 0	w w w w 0 1	0 0 0	1 0	1 0 1 0 0
reg2			reg3	category	type	sub-op

[ 説 明 ]                汎用レジスタ reg2 で指定されるレジスタ・ペアにある倍精度浮動小数点形式の内容を算術的に 64 ビットの整数形式に変換し、汎用レジスタ reg3 で指定されるレジスタ・ペアに格納します。

現在の丸めモードに関係なく、結果を + の方向へ丸めます。

ソース・オペランドが無限大か非数の場合、または丸めた結果が  $2^{63}-1 \sim -2^{63}$  の範囲外の場合は、IEEE754 の無効演算例外を検出します。

無効演算例外の発生が許可されていない場合は、例外を発生せず、無効演算としてと FPSR レジスタの保存ビット (ビット 4) がセットされます。ソースの違いにより返す値は次のように異なります。

- ・ソースが正数または+            :  $2^{63}-1$  を返します。
- ・ソースが負数、非数または-    :  $-2^{63}$  を返します。

[ 浮動小数点演算例外 ] 無効演算例外 (V)

不正確演算例外 (I)

[ 演算結果 ]            FPSR.FS = 1 の場合

reg2(A)	+Normal	-Normal	+0	-0	+	-	Q-NaN	S-NaN
演算結果 [例外]	A ( Integer )		0 ( Integer )		+Max Int[V]	-Max Int[V]		

備考 1. [ ] は必ず発生する例外です。

2. ディノーマル数はフラッシュされ "0" として扱われます。

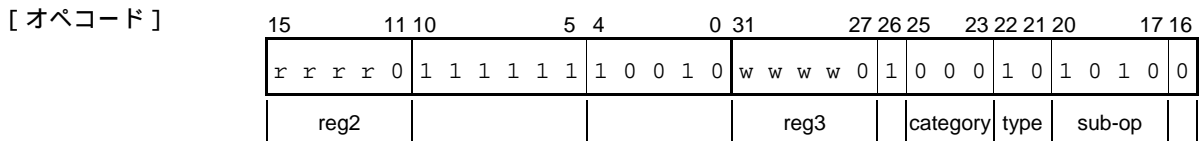
< 浮動小数点演算命令 >

<p style="font-size: 24pt; margin: 0;">CEILF.DUL</p>	<p style="font-size: 10pt; margin: 0;">Floating-point Ceiling to Unsigned Integer Format (Double)</p> <p style="font-size: 10pt; margin: 0;">符号なし整数形式への変換 (倍精度)</p>
--	---

[ 命令形式 ]            CEILF.DUL reg2, reg3

[ オペレーション ]      reg3 ← ceil reg2(double    Unsigned long-word)

[ フォーマット ]        Format F: I



[ 説 明 ]                汎用レジスタ reg2 で指定されるレジスタ・ペアにある倍精度浮動小数点形式の内容を算術的に符号のない 64 ビットの整数形式に変換し、汎用レジスタ reg3 で指定されるレジスタ・ペアに格納します。

現在の丸めモードに関係なく、結果を+ の方向へ丸めます。

ソース・オペランドが無限大か非数か負数の場合、または丸めた結果が  $2^{64}-1 \sim 0$  の範囲外の場合は、IEEE754 の無効演算例外を検出します。

無効演算例外の発生が許可されていない場合は、例外を発生せず、無効演算としてと FPSR レジスタの保存ビット (ビット 4) がセットされます。ソースの値により返す値は次のように異なります。

- ・ソースが  $2^{64}-1 \sim 0$  の範囲外の正数または+                    :  $2^{64}-1$  を返します。
- ・ソースが負数、非数または-                                        : 0 を返します。

[ 浮動小数点演算例外 ] 無効演算例外 (V)  
不正確演算例外 (I)

[ 演算結果 ]            FPSR.FS = 1 の場合

reg2(A)	+Normal	-Normal	+0	-0	+	-	Q-NaN	S-NaN
演算結果 [例外]	A (Integer)	0 [V]	0 (Integer)		Max U-Int [V]	0 [V]		

- 備考 1. [ ] は必ず発生する例外です。
2. ディノーマル数はフラッシュされ “0” として扱われます。

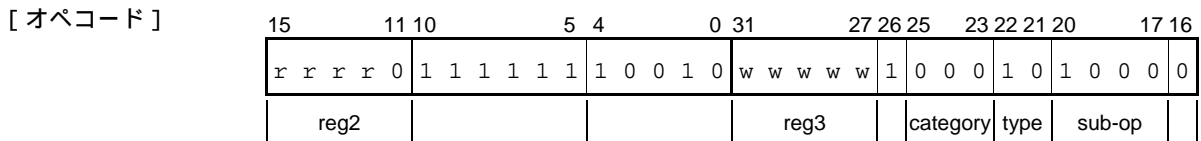
< 浮動小数点演算命令 >

<h2 style="margin: 0;">CEILF.DUW</h2>	Floating-point Ceiling to Unsigned Integer Format (Double)  符号なし整数形式への変換 (倍精度)
---------------------------------------	--

[ 命令形式 ]            CEILF.DUW reg2, reg3

[ オペレーション ]    reg3 ← ceil reg2(double    Unsigned word)

[ フォーマット ]      Format F: I



[ 説 明 ]            汎用レジスタ reg2 で指定されるレジスタ・ペアにある倍精度浮動小数点形式の内容を算術的に符号のない 32 ビットの整数形式に変換し、汎用レジスタ reg3 に格納します。

現在の丸めモードに関係なく、結果を+ の方向へ丸めます。

ソース・オペランドが無限大か非数か負数の場合、または丸めた結果が  $2^{32}-1 \sim 0$  の範囲外の場合は、IEEE754 の無効演算例外を検出します。

無効演算例外の発生が許可されていない場合は、例外を発生せず、無効演算としてと FPSR レジスタの保存ビット (ビット 4) がセットされます。ソースの値により返す値は次のように異なります。

- ・ ソースが  $2^{32}-1 \sim 0$  の範囲外の正数または+                    :  $2^{32}-1$  を返します。
- ・ ソースが負数、非数または-                                         : 0 を返します。

[ 浮動小数点演算例外 ] 無効演算例外 (V)  
不正確演算例外 (I)

[ 演算結果 ]            FPSR.FS = 1 の場合

reg2(A)	+Normal	-Normal	+0	-0	+	-	Q-NaN	S-NaN
演算結果 [例外]	A ( Integer )	0 [V]	0 ( Integer )		Max U-Int [V]	0 [V]		

- 備考 1. [ ] は必ず発生する例外です。
2. ディノーマル数はフラッシュされ " 0 " として扱われます。

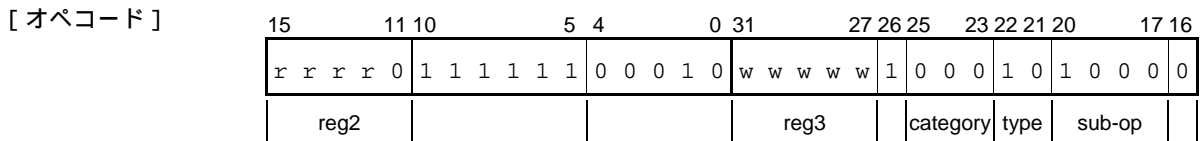
< 浮動小数点演算命令 >

<p style="font-size: 24px; margin: 0;">CEILF.DW</p>	<p style="font-size: 12px; margin: 0;">Floating-point Ceiling to Integer Format (Double)</p> <p style="font-size: 12px; margin: 0;">整数形式への変換 (倍精度)</p>
---	--

[ 命令形式 ]            CEILF.DW reg2, reg3

[ オペレーション ]        reg3 ← ceil reg2 (double    word)

[ フォーマット ]         Format F: I



[ 説 明 ]                汎用レジスタ reg2 で指定されるレジスタ・ペアにある倍精度浮動小数点形式の内容を算術的に 32 ビットの整数形式に変換し、汎用レジスタ reg3 に格納します。

現在の丸めモードに関係なく、結果を + の方向へ丸めます。

ソース・オペランドが無限大か非数の場合、または丸めた結果が  $2^{31} - 1 \sim -2^{31}$  の範囲外の場合は、IEEE754 の無効演算例外を検出します。

無効演算例外の発生が許可されていない場合は、例外を発生せず、無効演算としてと FPSR レジスタの保存ビット (ビット 4) がセットされます。ソースの違いにより返す値は次のように異なります。

- ・ソースが正数または+            :  $2^{31} - 1$  を返します。
- ・ソースが負数、非数または-    :  $-2^{31}$  を返します。

[ 浮動小数点演算例外 ] 無効演算例外 (V)  
不正確演算例外 (I)

[ 演算結果 ]             FPSR.FS = 1 の場合

reg2(A)	+Normal	-Normal	+0	-0	+	-	Q-NaN	S-NaN
演算結果 [例外]	A ( Integer )		0 ( Integer )		+Max Int[V]	-Max Int[V]		

備考 1. [ ] は必ず発生する例外です。  
2. ディノーマル数はフラッシュされ " 0 " として扱われます。

< 浮動小数点演算命令 >

CEILF.SL	Floating-point Ceiling to Integer Format (Single)  整数形式への変換 (単精度)
----------	---

[ 命令形式 ]           CEILF.SL reg2, reg3

[ オペレーション ]     reg3 ← ceil reg2 (single    long-word)

[ フォーマット ]       Format F: I

[ オペコード ]

15	11	10	5	4	0	31	27	26	25	23	22	21	20	17	16
r	r	r	r	r	1	1	1	1	1	1	0	0	0	1	0
reg2					reg3					category	type	sub-op			

[ 説 明 ]           汎用レジスタ reg2 にある単精度浮動小数点形式の内容を算術的に 64 ビットの整数形式に変換し、汎用レジスタ reg3 で指定されるレジスタ・ペアに格納します。

現在の丸めモードに関係なく、結果を + の方向へ丸めます。

ソース・オペランドが無大か非数の場合、または丸めた結果が  $2^{63}-1 \sim -2^{63}$  の範囲外の場合は、IEEE754 の無効演算例外を検出します。

無効演算例外の発生が許可されていない場合は、例外を発生せず、無効演算としてと FPSR レジスタの保存ビット (ビット 4) がセットされます。ソースの違いにより返す値は次のように異なります。

- ・ソースが正数または+           :  $2^{63}-1$  を返します。
- ・ソースが負数、非数または-   :  $-2^{63}$  を返します。

[ 浮動小数点演算例外 ] 無効演算例外 (V)  
不正確演算例外 (I)

[ 演算結果 ]           FPSR.FS = 1 の場合

reg2(A)	+Normal	-Normal	+0	-0	+	-	Q-NaN	S-NaN
演算結果 [例外]	A ( Integer )		0 ( Integer )		+Max Int[V]		-Max Int[V]	

- 備考 1. [ ] は必ず発生する例外です。
2. ディノーマル数はフラッシュされ “ 0 ” として扱われます。

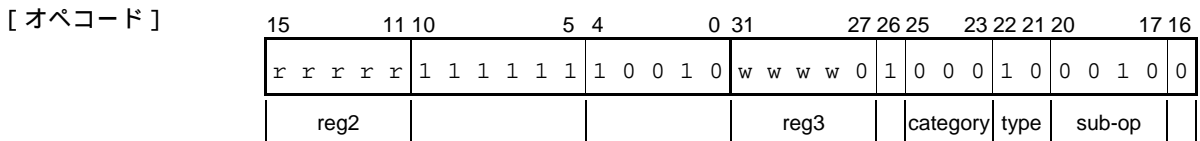
< 浮動小数点演算命令 >

CEILF.SUL	Floating-point Ceiling e to Unsigned Integer Format (Single)  符号なし整数形式への変換 (単精度)
-----------	--

[ 命令形式 ]            CEILF.SUL    reg2, reg3

[ オペレーション ]        reg3 ← ceil reg2(Single    Unsigned long-word)

[ フォーマット ]         Format F: I



[ 説 明 ]                 汎用レジスタ reg2 にある単精度浮動小数点形式の内容を算術的に符号のない 64 ビットの整数形式に変換し、汎用レジスタ reg3 で指定されるレジスタ・ペアに格納します。  
 現在の丸めモードに関係なく、結果を+ の方向へ丸めます。  
 ソース・オペランドが無大か非数か負数の場合、または丸めた結果が  $2^{64}-1 \sim 0$  の範囲外の場合は、IEEE754 の無効演算例外を検出します。  
 無効演算例外の発生が許可されていない場合は、例外を発生せず、無効演算としてと FPSR レジスタの保存ビット (ビット 4) がセットされます。ソースの値により返す値は次のように異なります。

- ・ ソースが  $2^{64}-1 \sim 0$  の範囲外の正数または+                                 :  $2^{64}-1$  を返します。
- ・ ソースが負数、非数または-     : 0 を返します。

[ 浮動小数点演算例外 ] 無効演算例外 (V)  
 不正確演算例外 (I)

[ 演算結果 ]                FPSR.FS = 1 の場合

reg2(A)	+Normal	-Normal	+0	-0	+	-	Q-NaN	S-NaN
演算結果 [例外]	A( Integer )	0 [V]	0 ( Integer )		Max U-Int [V]	0 [V]		

- 備考 1. [ ] は必ず発生する例外です。  
 2. ディノーマル数はフラッシュされ “0” として扱われます。

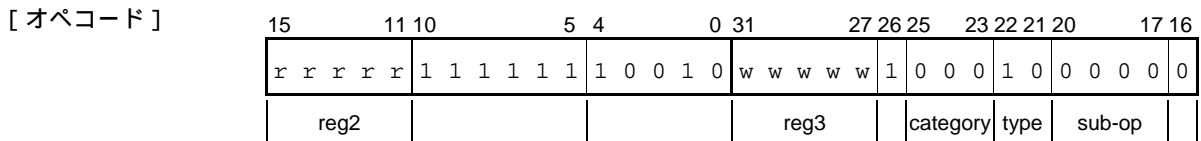
< 浮動小数点演算命令 >

<div style="display: flex; justify-content: space-between;"> <div style="font-size: 24pt; font-weight: bold;">CEILF.SUW</div> <div style="text-align: right; font-size: 10pt;">Floating-point Ceiling to Unsigned Integer Format (Single)</div> </div> <div style="text-align: right; margin-top: 10px; font-size: 10pt;">符号なし整数形式への変換 (単精度)</div>
--

[ 命令形式 ]            CEILF.SUW reg2, reg3

[ オペレーション ]        reg3 ← ceil reg2(Single    Unsigned word)

[ フォーマット ]         Format F: I



[ 説 明 ]                汎用レジスタ reg2 にある単精度浮動小数点数形式の内容を算術的に符号のない 32 ビットの整数形式に変換し、汎用レジスタ reg3 に格納します。

現在の丸めモードに関係なく、結果を+ の方向へ丸めます。

ソース・オペランドが無限大か非数か負数の場合、または丸めた結果が  $2^{32}-1 \sim 0$  の範囲外の場合は、IEEE754 の無効演算例外を検出します。

無効演算例外の発生が許可されていない場合は、例外を発生せず、無効演算としてと FPSR レジスタの保存ビット (ビット 4) がセットされます。ソースの値により返す値は次のように異なります。

- ・ ソースが  $2^{32}-1 \sim 0$  の範囲外の正数または+                                :  $2^{32}-1$  を返します。
- ・ ソースが負数、非数または-    : 0 を返します。

[ 浮動小数点演算例外 ] 無効演算例外 (V)  
不正確演算例外 (I)

[ 演算結果 ]             FPSR.FS = 1 の場合

reg2(A)	+Normal	-Normal	+0	-0	+	-	Q-NaN	S-NaN
演算結果 [例外]	A ( Integer )	0 [V]	0 ( Integer )		Max U-Int [V]	0 [V]		

- 備考 1. [ ] は必ず発生する例外です。
2. ディノーマル数はフラッシュされ “ 0 ” として扱われます。

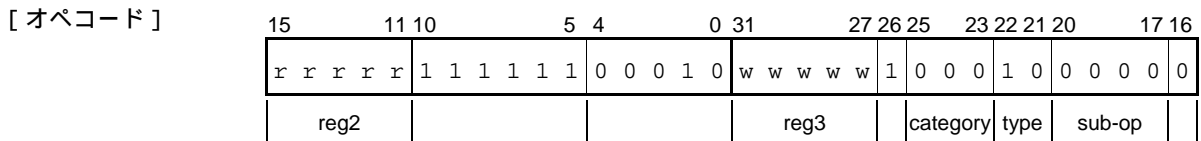
< 浮動小数点演算命令 >

CEILF.SW	Floating-point Ceiling to Integer Format (Single)  整数形式への変換 (単精度)
----------	---

[ 命令形式 ]           CEILF.SW reg2, reg3

[ オペレーション ]     reg3 ← ceil reg2 (single word)

[ フォーマット ]       Format F: I



[ 説 明 ]           汎用レジスタ reg2 にある単精度浮動小数点形式の内容を算術的に 32 ビットの整数形式に変換し、汎用レジスタ reg3 に格納します。

現在の丸めモードに関係なく、結果を + の方向へ丸めます。

ソース・オペランドが無大か非数の場合、または丸めた結果が  $2^{31} - 1 \sim -2^{31}$  の範囲外の場合は、IEEE754 の無効演算例外を検出します。

無効演算例外の発生が許可されていない場合は、例外を発生せず、無効演算としてと FPSR レジスタの保存ビット (ビット 4) がセットされます。ソースの違いにより返す値は次のように異なります。

- ・ソースが正数または+ :  $2^{31} - 1$  を返します。
- ・ソースが負数、非数または- :  $-2^{31}$  を返します。

[ 浮動小数点演算例外 ] 無効演算例外 (V)  
不正確演算例外 (I)

[ 演算結果 ]           FPSR.FS = 1 の場合

reg2(A)	+Normal	-Normal	+0	-0	+	-	Q-NaN	S-NaN
演算結果 [例外]	A ( Integer )		0 ( Integer )		+Max Int[V]		-Max Int[V]	

備考 1. [ ] は必ず発生する例外です。

2. ディノーマル数はフラッシュされ “ 0 ” として扱われます。



## &lt; 浮動小数点演算命令 &gt;

## CMOVF.D

Floating-point Conditional Move (Double)

条件付き転送 (倍精度)

[ 命令形式 ]            CMOVF.D fcbit, reg1, reg2, reg3

[ オペレーション ]    if FPSR.CCn == 1 then  
                           reg3    reg1  
                           else  
                           reg3    reg2  
                           endif

備考 n = fcbit

[ フォーマット ]        Format F: I

[ オペコード ]

15	11 10	5 4	0 31	27 26 25	23 22 21 20	17 16
r r r r 0	1 1 1 1 1 1	R R R R 0	w w w w 0 1	0 0 0	0 0 1 f f f 0	
			reg3 <sup>#</sup>	category	type	sub-op

注 reg3 : wwww! = 0

wwww    0000 (reg3 には r0 を設定しないでください)

備考 fcbit: fff

[ 説 明 ]            オペコードの fcbit によって指定された FPSR.CC(7:0)ビットが真 (1) の場合, reg1 で指定されるレジスタ・ペアのデータを reg3 で指定されるレジスタ・ペアに格納します。偽(0)の場合, reg2 で指定されるレジスタ・ペアのデータを reg3 で指定されるレジスタ・ペアに格納します。

[ 浮動小数点演算例外 ] なし

**注意**    reg3 には r0 を指定しないでください。

&lt; 浮動小数点条件命令 &gt;

CMOVF.S

Floating-point Conditional Move (Single)

条件付き転送 (単精度)

[ 命令形式 ]            CMOVF.S   fcbit, reg1, reg2, reg3

[ オペレーション ]    if   FPSR.CCn == 1 then  
                           reg3    reg1  
                           else  
                           reg3    reg2  
                           endif

**備考**   n = fcbit

[ フォーマット ]        Format F: I

[ オペコード ]

15	11 10	5 4	0 31	27 26 25	23 22 21 20	17 16
r r r r r	1 1 1 1 1 1	R R R R R	w w w w w	1	0 0 0	0 0 0 f f f 0
			reg3 <sup>#</sup>	category	type	sub-op

注    reg3 : wwwww! = 0

wwwww    00000 ( reg3 には r0 を設定しないでください )

**備考**   fcbit: fff

[ 説 明 ]            オペコードの fcbit によって指定された FPSR.CC(7:0)ビットが真 (1) の場合, reg1 のデータを reg3 に格納します。偽 (0) の場合, reg2 のデータを reg3 に格納します。

[ 浮動小数点演算例外 ] なし

**注意**    reg3 には r0 を指定しないでください。

## &lt; 浮動小数点演算命令 &gt;

CMPF.D

Floating-point Compare (Double)

浮動小数点比較 (倍精度)

[ 命令形式 ]            CMPF.D fcond, reg2, reg1, fcbit

CMPF.D fcond, reg2, reg1

[ オペレーション ]    if isNaN(reg1) or isNaN(reg2) then

result.less     0

result.equal    0

result.unordered 1

if fcond[3] == 1 then

無効演算例外を検出

endif

else

result.less     reg2 &lt; reg1

result.equal    reg2 == reg1

result.unordered 0

endif

FPSR.CCn     (fcond[2] &amp; result.less) | (fcond[1] &amp; result.equal) |

(fcond[0] &amp; result.unordered)

**備考** n : fcbit

[ フォーマット ]     Format F: I

[ オペコード ]

15	11	10	5	4	0	31	27	26	25	23	22	21	20	17	16
r	r	r	r	0	1	1	1	1	1	1	R	R	R	R	R
					0	F	F	F	F	1	0	0	0	0	1
											category	type	sub-op		
											1	f	f	f	0

**備考** fcond : FFFF

fcbit : fff

[ 説明 ] 汎用レジスタ reg2 で指定されるレジスタ・ペアにある倍精度浮動小数点形式の内容を、比較条件 fcond により、汎用レジスタ reg1 で指定されるレジスタ・ペアにある倍精度浮動小数点形式の内容と比較します。結果（真ならば1、偽ならば0）をオペコードの fcbits で指定される FPSR レジスタのコンディション・ビット（CC(7:0)ビット：ビット31-24）にセットします。fcbits が省略された場合は CC0 ビット（ビット24）にセットします。

比較条件 fcond のコードについては表4-6 比較条件を参照してください。

値の1つが非数で、比較条件 fcond の最上位ビットがセットされている場合は、IEEE754の無効演算例外を検出します。無効演算例外の発生が許可されている場合は、比較結果はセットされず、そのまま例外の処理に移ります。

許可ビットがセットされていない場合は、例外を発生せず、FPSR レジスタの保存ビット（ビット4）がセットされ、FPSR.CC(7:0)ビットに比較結果がセットされます。

比較も含め、浮動小数点演算命令ではオペランド値として SignalingNaN (S-NaN) を受け取ると、無効演算の条件と見なします。S-NaN だけでなく QuietNaN (Q-NaN) でも無効演算となる比較を使えば、NaN でエラーとなる場合のプログラムをより簡単なものにできます。つまり、結果が Unordered となるような Q-NaN を明確にチェックするためのコードが不要になります。代わりに、無効演算が検出されたときに例外を発生させ、エラーの処理を例外処理システムが行うようにします。次に、2つの数値が等しいかを調べるとともに、Unordered の場合はエラーとするような比較の場合を示します。

表4-6 比較条件

比較条件	fcond	定義	説明	Unorderedのとき
				無効演算例外の検出
F	0	FALSE	常に偽	しない
UN	1	Unordered	reg1, reg2の少なくとも一方が非数	しない
EQ	2	reg2 = reg1	Ordered (いずれも非数ではない) で、等しい	しない
UEQ	3	reg2 ? = reg1	Unordered (少なくとも一方が非数) か、等しい	しない
OLT	4	reg2 < reg1	Ordered (いずれも非数ではない) で、より小さい	しない
ULT	5	reg2 ? < reg1	Unordered (少なくとも一方が非数) か、より小さい	しない
OLE	6	reg2 reg1	Ordered (いずれも非数ではない) で、より小さいか、等しい	しない
ULE	7	reg2 ? reg1	Unordered (少なくとも一方が非数) か、より小さいか、等しい	しない
SF	8	FALSE	常に偽	する
NGLE	9	Unordered	reg1, reg2の少なくとも一方が非数	する
SEQ	10	reg2 = reg1	Ordered (いずれも非数ではない) で、等しい	する
NGL	11	reg2 ? = reg1	Unordered (少なくとも一方が非数) か、等しい	する
LT	12	reg2 < reg1	Ordered (いずれも非数ではない) で、より小さい	する
NGE	13	reg2 ? < reg1	Unordered (少なくとも一方が非数) か、より小さい	する
LE	14	reg2 reg1	Ordered (いずれも非数ではない) で、より小さいか、等しい	する
NGT	15	reg2 ? reg1	Unordered (少なくとも一方が非数) か、より小さいか、等しい	する

備考 ? : Unordered (比較不能)

# Q-NaNを明確にテストする場合

```

CMPF.D    OLT,r12,r14,0    # r12 < r14をチェック
CMPF.D    UN,r12,r14,1    # Unorderedかをチェック
TRFSR     0
BT        L2                # 真の場合はL2へ
TRFSR     1
BT        ERROR            # 真の場合はエラー処理へ
# Unorderedでなく、かつ、r12 < r14でない場合の処理コードを記述
L2:
# r12 < r14の場合の処理コードを記述
:

# Q-NaNを通知する比較を使用する場合
CMPF.D    LT,r12,r14,0    # r12 ?< r14をチェック
TRFSR     0
BT        L2                # 真の場合はL2へ
# Unorderedでなく、かつ、r12 < r14でない場合の処理コードを記述
L2:
# Unorderedか、r12 < r14の場合の処理コードを記述
:

```

[ 浮動小数点演算例外 ] 無効演算例外 (V)

[ 演算結果 ] FPSR.FS = 1 の場合

[ 条件コード (fcond) = 0-7 ]

reg1(B) reg2(A)	+Normal	-Normal	+0	-0	+	-	Q-NaN	S-NaN
± Normal	比較条件 (fcond) による比較結果の真偽を FPSR.CCnビットに格納 (n = fcbt)							
± 0								
±								
Q-NaN	Unorderd							
S-NaN								

[ 条件コード (fcond) = 8-15 ]

reg1(B) reg2(A)	+Normal	-Normal	+0	-0	+	-	Q-NaN	S-NaN
± Normal	比較条件 (fcond) による比較結果の真偽を FPSR.CCnビットに格納 (n = fcbt)							
± 0								
±								
Q-NaN	Unorderd[V]							
S-NaN								

## &lt; 浮動小数点演算命令 &gt;

**CMPF.S**

Floating-point Compare (Single)

浮動小数点比較 (単精度)

[ 命令形式 ]           CMPF.S fcond, reg2, reg1, fcbit  
                           CMPF.S fcond, reg2, reg1

[ オペレーション ]   if isNaN(reg1) or isNaN(reg2) then  
                           result.less     0  
                           result.equal    0  
                           result.unordered 1  
                           if fcond[3] == 1 then  
                               無効演算例外を検出  
                           endif  
                           else  
                               result.less    reg2 < reg1  
                               result.equal   reg2 == reg1  
                               result.unordered 0  
                           endif  
                           FPSR.CCn       (fcond[2] & result.less) | (fcond[1] & result.equal) |  
   (fcond[0] & result.unordered)

**備考** n : fcbit

[ フォーマット ]       Format F: I

[ オペコード ]

15	11	10	5	4	0	31	27	26	25	23	22	21	20	17	16
r	r	r	r	r	1	1	1	1	1	1	R	R	R	R	R
							0	F	F	F	F	1	0	0	0
											0	1	0	f	f
														f	f
														0	0
											category	type	sub-op		

**備考** fcond : FFFF

      fcbit : fff

[ 説明 ] 汎用レジスタ reg2 で指定されるレジスタ・ペアにある単精度浮動小数点形式の内容を、比較条件 fcond により、汎用レジスタ reg1 で指定されるレジスタ・ペアにある単精度浮動小数点形式の内容と比較します。結果（真ならば1、偽ならば0）をオペコードの fcbits で指定される FPSR レジスタのコンディション・ビット（CC(7:0)ビット：ビット31-24）にセットします。fcbits が省略された場合は CC0 ビット（ビット24）にセットします。

比較条件 fcond のコードについては表4-7 比較条件を参照してください。

値の1つが非数で、比較条件 fcond の最上位ビットがセットされている場合は、IEEE754の無効演算例外を検出します。無効演算例外の発生が許可されている場合は、比較結果はセットされず、そのまま例外の処理に移ります。

許可ビットがセットされていない場合は、例外を発生せず、FPSR レジスタの保存ビット（ビット4）がセットされ、FPSR.CC(7:0)ビットに比較結果がセットされます。

比較も含め、浮動小数点演算命令ではオペランド値として SignalingNaN (S-NaN) を受け取ると、無効演算の条件と見なします。S-NaN だけでなく QuietNaN (Q-NaN) でも無効演算となる比較を使えば、NaN でエラーとなる場合のプログラムをより簡単なものにできます。つまり、結果が Unordered となるような Q-NaN を明確にチェックするためのコードが不要になります。代わりに、無効演算が検出されたときに例外を発生させ、エラーの処理を例外処理システムが行うようにします。次に、2つの数値が等しいかを調べるとともに、Unordered の場合はエラーとするような比較の場合を示します。

表4-7 比較条件

比較条件	fcond	定義	説明	Unorderedのとき
				無効演算例外の検出
F	0	FALSE	常に偽	しない
UN	1	Unordered	reg1, reg2の少なくとも一方が非数	しない
EQ	2	reg2 = reg1	Ordered（いずれも非数ではない）で、等しい	しない
UEQ	3	reg2 ? = reg1	Unordered（少なくとも一方が非数）か、等しい	しない
OLT	4	reg2 < reg1	Ordered（いずれも非数ではない）で、より小さい	しない
ULT	5	reg2 ? < reg1	Unordered（少なくとも一方が非数）か、より小さい	しない
OLE	6	reg2 reg1	Ordered（いずれも非数ではない）で、より小さいか、等しい	しない
ULE	7	reg2 ? reg1	Unordered（少なくとも一方が非数）か、より小さいか、等しい	しない
SF	8	FALSE	常に偽	する
NGLE	9	Unordered	reg1, reg2の少なくとも一方が非数	する
SEQ	10	reg2 = reg1	Ordered（いずれも非数ではない）で、等しい	する
NGL	11	reg2 ? = reg1	Unordered（少なくとも一方が非数）か、等しい	する
LT	12	reg2 < reg1	Ordered（いずれも非数ではない）で、より小さい	する
NGE	13	reg2 ? < reg1	Unordered（少なくとも一方が非数）か、より小さい	する
LE	14	reg2 reg1	Ordered（いずれも非数ではない）で、より小さいか、等しい	する
NGT	15	reg2 ? reg1	Unordered（少なくとも一方が非数）か、より小さいか、等しい	する

備考 ? : Unordered（比較不能）

# Q-NaNを明確にテストする場合

```

CMPF.S    OLT,r12,r14,0    # r12 < r14をチェック
CMPF.S    UN,r12,r14,1    # Unorderedかをチェック
TRFSR     0
BT        L2                # 真の場合はL2へ
TRFSR     1
BT        ERROR            # 真の場合はエラー処理へ
# Unorderedでなく、かつ、r12 < r14でない場合の処理コードを記述
L2:
# r12 < r14の場合の処理コードを記述
:

# Q-NaNを通知する比較を使用する場合
CMPF.S    LT,r12,r14,0    # r12 ?< r14をチェック
TRFSR     0
BT        L2                # 真の場合はL2へ
# Unorderedでなく、かつ、r12 < r14でない場合の処理コードを記述
L2:
# Unorderedか、r12 < r14の場合の処理コードを記述
:

```

[ 浮動小数点演算例外 ] 無効演算例外 (V)

[ 演算結果 ]           FPSR.FS = 1 の場合

[ 条件コード (fcond) = 0-7 ]

reg1(B) reg2(A)	+Normal	-Normal	+0	-0	+	-	Q-NaN	S-NaN
± Normal	比較条件 (fcond) による比較結果の真偽を FPSR.CCnビットに格納 (n = fcbit)							
± 0								
±								
Q-NaN	Unorderd							
S-NaN								

[ 条件コード (fcond) = 8-15 ]

reg1(B) reg2(A)	+Normal	-Normal	+0	-0	+	-	Q-NaN	S-NaN
± Normal	比較条件 (fcond) による比較結果の真偽を FPSR.CCnビットに格納 (n = fcbit)							
± 0								
±								
Q-NaN	Unorderd[V]							
S-NaN								



## &lt; 浮動小数点演算命令 &gt;

CVTF.DL	Floating-point Convert to Integer Format (Double) 整数形式への変換 (倍精度)
---------	---

[ 命令形式 ]            CVTF.DL reg2, reg3

[ オペレーション ]       reg3 ← cvt reg2(double    long-word)

[ フォーマット ]        Format F: I

15	11 10	5 4	0 31	27 26 25	23 22 21 20	17 16
r r r r 0	1 1 1 1 1 1	0 0 1 0 0	w w w w 0 1	0 0 0	1 0	1 0 1 0 0
reg2			reg3	category	type	sub-op

[ 説 明 ]            汎用レジスタ reg2 で指定されるレジスタ・ペアにある倍精度浮動小数点形式の内容を現在の丸めモードに従って算術的に 64 ビットの整数形式に変換し、汎用レジスタ reg3 で指定されるレジスタ・ペアに格納します。

ソース・オペランドが無大か非数の場合、または丸めた結果が  $2^{63}-1 \sim -2^{63}$  の範囲外の場合は、IEEE754 の無効演算例外を検出します。

無効演算例外の発生が許可されていない場合は、例外を発生せず、無効演算としてと FPSR レジスタの保存ビット (ビット 4) がセットされます。ソースの違いにより返す値は次のように異なります。

- ・ ソースが正数または+            :  $2^{63}-1$  を返します。
- ・ ソースが負数、非数または-    :  $-2^{63}$  を返します。

[ 浮動小数点演算例外 ]            無効演算例外 (V)  
不正確演算例外 (I)

[ 演算結果 ]            FPSR.FS = 1 の場合

reg2(A)	+Normal	-Normal	+0	-0	+	-	Q-NaN	S-NaN
演算結果 [例外]	A ( Integer )		0 ( Integer )		+Max Int[V]		-Max Int[V]	

備考 1. [    ] は必ず発生する例外です。

2. ディノーマル数はフラッシュされ “ 0 ” として扱われます。

## &lt; 浮動小数点演算命令 &gt;

<b>CVTF.DS</b>	Floating-point Convert to Single Floating-point Format (Double)
浮動小数点形式への変換 (倍精度)	

[ 命令形式 ]            CVTF.DS reg2, reg3

[ オペレーション ]        reg3 ← cvt reg2(double     single)

[ フォーマット ]         Format F: I

[ オペコード ]

15	11	10	5	4	0	31	27	26	25	23	22	21	20	17	16																
r	r	r	r	0	1	1	1	1	1	1	0	0	0	1	1	w	w	w	w	w	1	0	0	0	1	0	1	0	0	1	0
reg2										reg3					category	type	sub-op														

[ 説 明 ]                 汎用レジスタ reg2 で指定されるレジスタ・ペアにある倍精度浮動小数点形式の内容を算術的に単精度浮動小数点形式に変換し、汎用レジスタ reg3 に格納します。結果は、現在の丸めモードに従って丸めます。

[ 浮動小数点演算例外 ] 無効演算例外 (V)

不正確演算例外 (I)

オーバフロー例外 (O)

アンダフロー例外 (U)

[ 演算結果 ]             FPSR.FS = 1 の場合

reg2(A)	+Normal	-Normal	+0	-0	+	-	Q-NaN	S-NaN
演算結果 [例外]	A (Single)		+0	-0	+	-	Q-NaN	Q-NaN[V]

備考 1. [ ] は必ず発生する例外です。

2. ディノーマル数はフラッシュされ“0”として扱われます。

<浮動小数点演算命令>

## CVTF.DUL

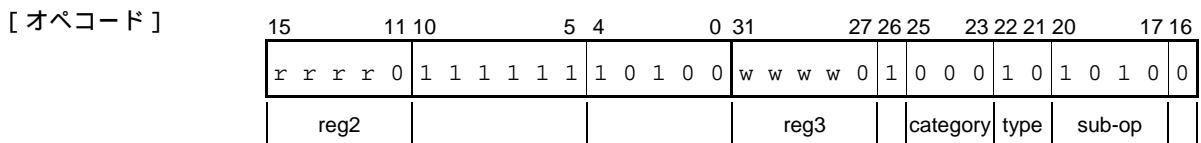
Floating-point Convert to Unsigned Integer Format (Double)

符号なし整数形式への変換（倍精度）

[ 命令形式 ]            CVTF.DUL reg2, reg3

[ オペレーション ]     reg3 ← cvt reg2(double     unsigned long-word)

[ フォーマット ]        Format F: I



[ 説 明 ]                汎用レジスタ reg2 で指定されるレジスタ・ペアにある倍精度浮動小数点形式の内容を現在の丸めモードに従って算術的に符号のない 64 ビットの整数形式に変換し、汎用レジスタ reg3 で指定されるレジスタ・ペアに格納します。

ソース・オペランドが無限大か非数か負数の場合、または丸めた結果が  $2^{64}-1 \sim 0$  の範囲外の場合は、IEEE754 の無効演算例外を検出します。

無効演算例外の発生が許可されていない場合は、例外を発生せず、無効演算としてと FPSR レジスタの保存ビット（ビット 4）がセットされます。ソースの値により返す値は次のように異なります。

- ・ ソースが  $2^{64}-1 \sim 0$  の範囲外の正数または+                    :  $2^{64}-1$  を返します。
- ・ ソースが負数、非数または-                                        : 0 を返します。

[ 浮動小数点演算例外 ] 無効演算例外 (V)  
不正確演算例外 (I)

[ 演算結果 ]             FPSR.FS = 1 の場合

reg2(A)	+Normal	-Normal	+0	-0	+	-	Q-NaN	S-NaN
演算結果 [例外]	A( Integer )	0 [V]	0 ( Integer )		Max U-Int [V]	0 [V]		

- 備考 1. [ ] は必ず発生する例外です。
2. ディノーマル数はフラッシュされ “0” として扱われます。

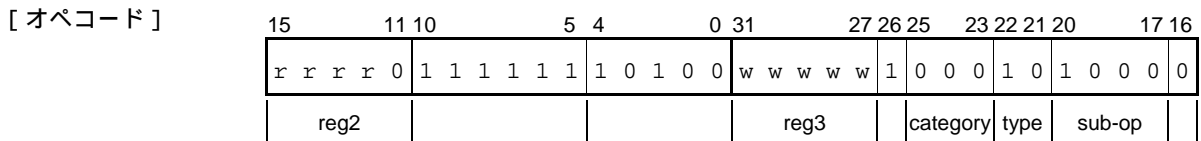
< 浮動小数点演算命令 >

<div style="display: flex; justify-content: space-between;"> <div style="text-align: center;">CVTF.DUW</div> <div style="text-align: right;">Floating-point Convert to Unsigned Integer Format (Double)</div> </div> <div style="text-align: right; margin-top: 10px;">符号なし整数形式への変換 (倍精度)</div>
---

[ 命令形式 ]            CVTF.DUW reg2, reg3

[ オペレーション ]        reg3 ← cvt reg2(double word)

[ フォーマット ]         Format F: I



[ 説 明 ]                汎用レジスタ reg2 で指定されるレジスタ・ペアにある倍精度浮動小数点形式の内容を算術的に符号のない 32 ビットの整数形式に変換し、汎用レジスタ reg3 に格納します。

ソース・オペランドが無限大か非数か負数の場合、または丸めた結果が  $2^{32}-1 \sim 0$  の範囲外の場合は、IEEE754 の無効演算例外を検出します。

無効演算例外の発生が許可されていない場合は、例外を発生せず、無効演算としてと FPSR レジスタの保存ビット (ビット 4) がセットされます。ソースの値により返す値は次のように異なります。

- ・ソースが  $2^{32}-1 \sim 0$  の範囲外の正数または+                                :  $2^{32}-1$  を返します。
- ・ソースが負数、非数または-    : 0 を返します。

[ 浮動小数点演算例外 ] 無効演算例外 (V)  
不正確演算例外 (I)

[ 演算結果 ]             FPSR.FS = 1 の場合

reg2(A)	+Normal	-Normal	+0	-0	+	-	Q-NaN	S-NaN
演算結果 [例外]	A( Integer )	0 [V]	0 ( Integer )		Max U-Int [V]	0 [V]		

備考 1. [ ] は必ず発生する例外です。  
2. ディノーマル数はフラッシュされ “0” として扱われます。

< 浮動小数点演算命令 >

CVTF.DW	Floating-point Convert to Integer Format (Double)  整数形式への変換 (倍精度)
---------	---

[ 命令形式 ]            CVTF.DW reg2, reg3

[ オペレーション ]      reg3 ← cvt reg2(double    word)

[ フォーマット ]        Format F: I

[ オペコード ]

15	11	10	5	4	0	31	27	26	25	23	22	21	20	17	16																
r	r	r	r	0	1	1	1	1	1	1	0	0	1	0	0	w	w	w	w	w	1	0	0	0	1	0	1	0	0	0	0
reg2										reg3					category	type	sub-op														

[ 説 明 ]            汎用レジスタ reg2 で指定されるレジスタ・ペアにある倍精度浮動小数点形式の内容を算術的に 32 ビットの整数形式に変換し、汎用レジスタ reg3 に格納します。

ソース・オペランドが無限大か非数の場合、または丸めた結果が  $2^{31}-1 \sim -2^{31}$  の範囲外の場合は、IEEE754 の無効演算例外を検出します。

無効演算例外の発生が許可されていない場合は、例外を発生せず、無効演算としてと FPSR レジスタの保存ビット (ビット 4) がセットされます。ソースの違いにより返す値は次のように異なります。

- ・ソースが正数または+            :  $2^{31}-1$  を返します。
- ・ソースが負数、非数または-    :  $-2^{31}$  を返します。

[ 浮動小数点演算例外 ] 無効演算例外 (V)  
不正確演算例外 (I)

[ 演算結果 ]            FPSR.FS = 1 の場合

reg2(A)	+Normal	-Normal	+0	-0	+	-	Q-NaN	S-NaN
演算結果 [例外]	A ( Integer )		0 ( Integer )		+Max Int[V]		-Max Int[V]	

- 備考 1. [ ] は必ず発生する例外です。
2. ディノーマル数はフラッシュされ “ 0 ” として扱われます。

## &lt; 浮動小数点演算命令 &gt;

CVTF.LD	Floating-point Convert to Double Floating-point Format (Double) 浮動小数点形式への変換 (倍精度)
---------	--

[ 命令形式 ]            CVTF.LD reg2, reg3

[ オペレーション ]        reg3 ← cvt reg2(long-word    double)

[ フォーマット ]         Format F: I

[ オペコード ]

15	11 10	5 4	0 31	27 26 25	23 22 21 20	17 16
r r r r 0	1 1 1 1 1 1	0 0 0 0 1	w w w w 0	1 0 0 0	1 0 1 0 0 1	0
reg2			reg3	category	type	sub-op

[ 説 明 ]                汎用レジスタ reg2 で指定されるレジスタ・ペアにある 64 ビットの整数形式の内容を現在の丸めモードに従って算術的に倍精度浮動小数点形式に変換し、汎用レジスタ reg3 で指定されるレジスタ・ペアに格納します。

[ 浮動小数点演算例外 ] 不正確演算例外 (I)

[ 演算結果 ]            FPSR.FS = 1 の場合

reg2(A)	+Integer	-Integer	0 ( Integer )
演算結果 [例外]	A ( Normal )		+0

備考 ディノーマル数はフラッシュされ“0”として扱われます。

## &lt; 浮動小数点演算命令 &gt;

<b>CVTF.LS</b>	Floating-point Convert to Single Floating-point Format (Single)  浮動小数点形式への変換 (単精度)
----------------	--

[ 命令形式 ]            CVTF.LS reg2, reg3

[ オペレーション ]      reg3 ← cvt reg2(long-word    single)

[ フォーマット ]        Format F: I

[ オペコード ]

15	11	10	5	4	0	31	27	26	25	23	22	21	20	17	16																
r	r	r	r	0	1	1	1	1	1	1	0	0	0	0	1	w	w	w	w	w	1	0	0	0	1	0	0	0	0	1	0
reg2										reg3					category		type		sub-op												

[ 説 明 ]                汎用レジスタ reg2 で指定されるレジスタ・ペアにある 64 ビットの整数形式の内容を算術的に単精度浮動小数点形式に変換し、汎用レジスタ reg3 に格納します。結果は、現在の丸めモードに従って丸めます。

[ 浮動小数点演算例外 ] 不正確演算例外 (I)

[ 演算結果 ]            FPSR.FS = 1 の場合

reg2(A)	+Integer	-Integer	0 ( Integer )
演算結果 [例外]	A ( Normal )		+0

備考 ディノーマル数はフラッシュされ“0”として扱われます。

## &lt; 浮動小数点演算命令 &gt;

CVTF.SD	Floating-point Convert to Double Floating-point Format (Double)
	浮動小数点形式への変換 (倍精度)

[ 命令形式 ]            CVTF.SD reg2, reg3

[ オペレーション ]        reg3 ← cvt reg2(single     double)

[ フォーマット ]         Format F: I

[ オペコード ]

15	11	10	5	4	0	31	27	26	25	23	22	21	20	17	16																
r	r	r	r	r	1	1	1	1	1	1	0	0	0	1	0	w	w	w	w	0	1	0	0	0	1	0	1	0	0	1	0
reg2										reg3					category		type		sub-op												

[ 説 明 ]                 汎用レジスタ reg2 にある単精度浮動小数点形式の内容を現在の丸めモードに従って算術的に倍精度浮動小数点形式に変換し、汎用レジスタ reg3 で指定されるレジスタ・ペアに格納します。

[ 浮動小数点演算例外 ] 無効演算例外 (V)  
不正確演算例外 (I)

[ 演算結果 ]             FPSR.FS = 1 の場合

reg2(A)	+Normal	-Normal	+0	-0	+	-	Q-NaN	S-NaN
演算結果 [例外]	A ( Double )		+0	-0	+	-	Q-NaN	Q-NaN[V]

備考 1. [ ] は必ず発生する例外です。

2. ディノーマル数はフラッシュされ "0" として扱われます。



< 浮動小数点演算命令 >

CVTF.SL	Floating-point Convert to Integer Format(Single)  整数形式への変換 (単精度)
---------	--

[ 命令形式 ]            CVTF.SL reg2, reg3

[ オペレーション ]      reg3 ← cvt reg2(single long-word)

[ フォーマット ]        Format F: I

[ オペコード ]

15	11	10	5	4	0	31	27	26	25	23	22	21	20	17	16
r	r	r	r	r	1	1	1	1	1	1	0	0	1	0	0
reg2					reg3					category	type	sub-op			

[ 説 明 ]                汎用レジスタ reg2 にある単精度浮動小数点形式の内容を現在の丸めモードに従って算術的に 64 ビットの整数形式に変換し、汎用レジスタ reg3 で指定されるレジスタ・ペアに格納します。

ソース・オペランドが無限大か非数の場合、または丸めた結果が  $2^{63}-1 \sim -2^{63}$  の範囲外の場合は、IEEE754 の無効演算例外を検出します。

無効演算例外の発生が許可されていない場合は、例外を発生せず、無効演算としてと FPSR レジスタの保存ビット (ビット 4) がセットされます。ソースの違いにより返す値は次のように異なります。

- ・ソースが正数または+            :  $2^{63}-1$  を返します。
- ・ソースが負数、非数または-    :  $-2^{63}$  を返します。

[ 浮動小数点演算例外 ] 無効演算例外 (V)  
不正確演算例外 (I)

[ 演算結果 ]            FPSR.FS = 1 の場合

reg2(A)	+Normal	-Normal	+0	-0	+	-	Q-NaN	S-NaN
演算結果 [例外]	A ( Integer )		0 ( Integer )		+Max Int[V]	-Max Int[V]		

- 備考 1. [ ] は必ず発生する例外です。
2. ディノーマル数はフラッシュされ “ 0 ” として扱われます。





< 浮動小数点演算命令 >

CVTF.SW	Floating-point Convert to Integer Format (Single)
	整数形式への変換 (単精度)

[ 命令形式 ]            CVTF.SW reg2, reg3

[ オペレーション ]     reg3 ← cvt reg2(single word)

[ フォーマット ]        Format F: I

[ オペコード ]

15	11	10	5	4	0	31	27	26	25	23	22	21	20	17	16																
r	r	r	r	r	1	1	1	1	1	1	0	0	1	0	0	w	w	w	w	w	1	0	0	0	1	0	0	0	0	0	0
reg2										reg3					category	type	sub-op														

[ 説 明 ]                汎用レジスタ reg2 にある単精度浮動小数点形式の内容を算術的に 32 ビットの整数形式に変換し、汎用レジスタ reg3 に格納します。

ソース・オペランドが無限大か非数の場合、または丸めた結果が  $2^{31}-1 \sim -2^{31}$  の範囲外の場合は、IEEE754 の無効演算例外を検出します。

無効演算例外の発生が許可されていない場合は、例外を発生せず、無効演算としてと FPSR レジスタの保存ビット (ビット 4) がセットされます。ソースの違いにより返す値は次のように異なります。

- ・ソースが正数または+            :  $2^{31}-1$  を返します。
- ・ソースが負数、非数または-    :  $-2^{31}$  を返します。

[ 浮動小数点演算例外 ] 無効演算例外 (V)  
不正確演算例外 (I)

[ 演算結果 ]            FPSR.FS = 1 の場合

reg2(A)	+Normal	-Normal	+0	-0	+	-	Q-NaN	S-NaN
演算結果 [例外]	A ( Integer )		0 ( Integer )		+Max Int[V]	-Max Int[V]		

- 備考 1. [ ] は必ず発生する例外です。
2. ディノーマル数はフラッシュされ “ 0 ” として扱われます。

## &lt; 浮動小数点演算命令 &gt;

<b>CVTF.ULD</b>	Floating-point Convert to Double Floating-point Format (Double)  浮動小数点形式への変換 (倍精度)
-----------------	--

[ 命令形式 ]            CVTF.ULD reg2, reg3

[ オペレーション ]    reg3 ← cvt reg2(unsigned long-word    Double)

[ フォーマット ]      Format F: I

[ オペコード ]

15	11	10	5	4	0	31	27	26	25	23	22	21	20	17	16																
r	r	r	r	0	1	1	1	1	1	1	1	0	0	0	1	w	w	w	w	0	1	0	0	0	1	0	1	0	0	1	0
reg2										reg3					category		type		sub-op												

[ 説 明 ]            汎用レジスタ reg2 で指定されるレジスタ・ペアにある符号のない 64 ビットの整数形式の内容を現在の丸めモードに従って算術的に倍精度浮動小数点形式に変換し、汎用レジスタ reg3 で指定されるレジスタ・ペアに格納します。

[ 浮動小数点演算例外 ] 不正確演算例外 (I)

[ 演算結果 ]            FPSR.FS = 1 の場合

reg2(A)	+Integer	-Integer	0 ( Integer )
演算結果 [例外]	A ( Normal )		+0

備考 ディノーマル数はフラッシュされ“0”として扱われます。

## &lt; 浮動小数点演算命令 &gt;

<b>CVTF.ULS</b>	Floating-point Convert to Single Floating-point Format (Single)  浮動小数点形式への変換 (単精度)
-----------------	--

[ 命令形式 ]            CVTF.ULS   reg2, reg3

[ オペレーション ]       reg3 ← cvt reg2(unsigned long-word    Single)

[ フォーマット ]        Format F: I

[ オペコード ]

15	11	10	5	4	0	31	27	26	25	23	22	21	20	17	16													
r	r	r	r	0	1	1	1	1	1	1	1	0	0	0	1	w	w	w	w	w	1	0	0	0	1	0	1	0
reg2										reg3					category	type	sub-op											

[ 説 明 ]                汎用レジスタ reg2 で指定されるレジスタ・ペアにある符号のない 64 ビットの整数形式の内容を算術的に単精度浮動小数点形式に変換し、汎用レジスタ reg3 に格納します。結果は、現在の丸めモードに従って丸めます。

[ 浮動小数点演算例外 ] 不正確演算例外 (I)

[ 演算結果 ]            FPSR.FS = 1 の場合

reg2(A)	+Integer	-Integer	0 ( Integer )
演算結果 [例外]	A ( Normal )		+0

備考    ディノーマル数はフラッシュされ“0”として扱われます。

## &lt; 浮動小数点演算命令 &gt;

<b>CVTF.UWD</b>	Floating-point Convert to Double Floating-point Format (Double)  浮動小数点形式への変換 (倍精度)
-----------------	--

[ 命令形式 ]            CVTF.UWD reg2, reg3

[ オペレーション ]     reg3 ← cvt reg2(unsigned word    Double)

[ フォーマット ]        Format F: I

[ オペコード ]

	15		11	10		5	4		0	31		27	26	25		23	22	21	20		17	16										
	r	r	r	r	r	1	1	1	1	1	1	1	0	0	0	0	w	w	w	w	0	1	0	0	0	1	0	1	0	0	1	0
	reg2										reg3					category	type	sub-op														

[ 説 明 ]                汎用レジスタ reg2 にある符号のない 32 ビットの整数形式の内容を現在の丸めモードに従って算術的に倍精度浮動小数点形式に変換し、汎用レジスタ reg3 で指定されるレジスタ・ペアに格納します。

この変換演算は、精度落ちなく正確に実行されます。

[ 浮動小数点演算例外 ] 不正確演算例外 (1)

[ 演算結果 ]            FPSR.FS = 1 の場合

reg2(A)	+Integer	-Integer	0 ( Integer )
演算結果 [例外]	A ( Normal )		+0

備考    ディノーマル数はフラッシュされ“0”として扱われます。

## &lt; 浮動小数点演算命令 &gt;

<b>CVTF.UWS</b>	Floating-point Convert to Double Floating-point Format (Single)  浮動小数点形式への変換 (単精度)
-----------------	--

[ 命令形式 ]            CVTF.UWS    reg2, reg3

[ オペレーション ]        reg3 ← cvt reg2(unsigned word    Single)

[ フォーマット ]         Format F: I

[ オペコード ]

15	11	10	5	4	0	31	27	26	25	23	22	21	20	17	16											
r	r	r	r	r	1	1	1	1	1	1	1	0	0	0	0	w	w	w	w	w	1	0	0	0	1	0
reg2										reg3					category	type	sub-op									

[ 説 明 ]                汎用レジスタ reg2 にある符号のない 32 ビットの整数形式の内容を算術的に単精度浮動小数点形式に変換し、汎用レジスタ reg3 に格納します。結果は、現在の丸めモードに従って丸めます。

[ 浮動小数点演算例外 ] 不正確演算例外 (I)

[ 演算結果 ]            FPSR.FS = 1 の場合

reg2(A)	+Integer	-Integer	0 ( Integer )
演算結果 [例外]	A ( Normal )		+0

備考    ディノーマル数はフラッシュされ“0”として扱われます。



## &lt; 浮動小数点演算命令 &gt;

<b>CVTF.WD</b>	Floating-point Convert to Double Floating-point Format (Double)  浮動小数点形式への変換 (倍精度)
----------------	--

[ 命令形式 ]            CVTF.WD reg2, reg3

[ オペレーション ]    reg3 ← cvt reg2(word    double)

[ フォーマット ]        Format F: I

[ オペコード ]

15	11	10	5	4	0	31	27	26	25	23	22	21	20	17	16																
r	r	r	r	r	1	1	1	1	1	1	0	0	0	0	0	w	w	w	w	0	1	0	0	0	1	0	1	0	0	1	0
reg2										reg3					category		type		sub-op												

[ 説 明 ]                汎用レジスタ reg2 にある 32 ビットの整数形式の内容を現在の丸めモードに従って算術的に倍精度浮動小数点形式に変換し、汎用レジスタ reg3 で指定されるレジスタ・ペアに格納します。

この変換演算は、精度落ちなく正確に実行されます。

[ 浮動小数点演算例外 ] 不正確演算例外 (I)

[ 演算結果 ]            FPSR.FS = 1 の場合

reg2(A)	+Integer	-Integer	0 ( Integer )
演算結果 [例外]	A ( Normal )		+0

備考    ディノーマル数はフラッシュされ“0”として扱われます。

## &lt; 浮動小数点演算命令 &gt;

CVTF.WS	Floating-point Convert to Single Floating-point Format (single)  浮動小数点形式への変換 (単精度)
---------	--

[ 命令形式 ]            CVTF.WS reg2, reg3

[ オペレーション ]    reg3 ← cvt reg2(word    single)

[ フォーマット ]      Format F: I

[ オペコード ]

15	11	10	5	4	0	31	27	26	25	23	22	21	20	17	16											
r	r	r	r	r	1	1	1	1	1	1	0	0	0	0	0	w	w	w	w	w	1	0	0	0	1	0
reg2										reg3					category	type	sub-op									

[ 説 明 ]            汎用レジスタ reg2 にある 32 ビットの整数形式の内容を算術的に単精度浮動小数点形式に変換し、汎用レジスタ reg3 に格納します。結果は、現在の丸めモードに従って丸めます。

[ 浮動小数点演算例外 ] 不正確演算例外 (I)

[ 演算結果 ]            FPSR.FS = 1 の場合

reg2(A)	+Integer	-Integer	0 ( Integer )
演算結果 [例外]	A ( Normal )		+0

備考 ディノーマル数はフラッシュされ“0”として扱われます。

< 浮動小数点演算命令 >

<p style="font-size: 24pt; margin: 0;">DIVF.D</p>	<p style="font-size: 10pt; margin: 0;">Floating-point Divide (Double)</p> <p style="font-size: 10pt; margin: 0;">浮動小数点除算（倍精度）</p>
---	---

[ 命令形式 ]            DIVF.D    reg1, reg2, reg3

[ オペレーション ]        reg3 ← reg2 ÷ reg1

[ フォーマット ]        Format F: I

[ オペコード ]

15	11	10	5	4	0	31	27	26	25	23	22	21	20	17	16															
r	r	r	r	0	1	1	1	1	1	1	R	R	R	R	0	w	w	w	w	0	1	0	0	0	1	1	1	1	1	0
reg2					reg1					reg3					category		type		sub-op											

[ 説 明 ]            汎用レジスタ reg2 で指定されるレジスタ・ペアにある倍精度浮動小数点形式の内容を汎用レジスタ reg1 で指定されるレジスタ・ペアにある倍精度浮動小数点形式の内容で除算し、汎用レジスタ reg3 で指定されるレジスタ・ペアに格納します。演算は無限精度であるかのように実行し、結果を現在の丸めモードに従って丸めます。

- [ 浮動小数点演算例外 ] 無効演算例外 (V)  
 不正確演算例外 (I)  
 ゼロ除算例外 (Z)  
 オーバフロー例外 (O)  
 アンダフロー例外 (U)

[ 演算結果 ]            FPSR.FS = 1 の場合

reg2(B) reg1(A)	Normal	-Normal	+0	-0	+	-	Q-NaN	S-NaN
Normal	B ÷ A				+	-	Q-NaN	S-NaN
-Normal					-	+		
+0	± [Z]		Q-NaN[V]		+	-		
-0					-	+		
+	+0	-0	+0	-0	Q-NaN[V]			
-	-0	+0	-0	+0				
Q-NaN	Q-NaN							
S-NaN	Q-NaN[V]							

- 備考 1. [ ] は必ず発生する例外です。  
 2. ディノーマル数はフラッシュされ “0” として扱われます。

## &lt; 浮動小数点演算命令 &gt;

DIVF.S	Floating-point Divide (Single)  浮動小数点除算 (単精度)
--------	---

[ 命令形式 ]            DIVF.S   reg1, reg2, reg3

[ オペレーション ]        reg3 ← reg2 ÷ reg1

[ フォーマット ]         Format F: I

[ オペコード ]

15	11	10	5	4	0	31	27	26	25	23	22	21	20	17	16																
r	r	r	r	r	1	1	1	1	1	1	R	R	R	R	R	w	w	w	w	w	1	0	0	0	1	1	0	1	1	1	0
reg2					reg1					reg3					category	type	sub-op														

[ 説 明 ]                 汎用レジスタ reg2 にある単精度浮動小数点形式の内容を汎用レジスタ reg1 にある単精度浮動小数点形式の内容で除算し、汎用レジスタ reg3 に格納します。演算は無限精度であるかのように実行し、結果を現在の丸めモードに従って丸めます。

[ 浮動小数点演算例外 ] 無効演算例外 (V)

不正確演算例外 (I)

ゼロ除算例外 (Z)

オーバフロー例外 (O)

アンダフロー例外 (U)

[ 演算結果 ]             FPSR.FS = 1 の場合

reg2(B) reg1(A)	Normal	-Normal	+0	-0	+	-	Q-NaN	S-NaN	
Normal	B ÷ A				+	-	Q-NaN	S-NaN	
-Normal	B ÷ A				-	+			
+0	± [Z]		Q-NaN[V]		+	-			
-0	± [Z]		Q-NaN[V]		-	+			
+	+0	-0	+0	-0	Q-NaN[V]		Q-NaN	Q-NaN[V]	
-	-0	+0	-0	+0	Q-NaN[V]				
Q-NaN	Q-NaN							Q-NaN	Q-NaN[V]
S-NaN	Q-NaN								

備考 1. [ ] は必ず発生する例外です。

2. ディノーマル数はフラッシュされ“0”として扱われます。

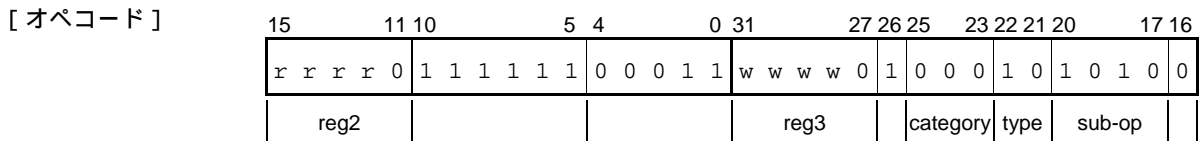
< 浮動小数点演算命令 >

<p style="font-size: 24pt; margin: 0;">FLOORF.DL</p>	<p style="font-size: 10pt; margin: 0;">Floating-point Truncate, rounded toward the direction of - (Double)</p> <p style="font-size: 10pt; margin: 0;">整数形式への変換 (倍精度)</p>
--	--

[ 命令形式 ]            FLOORF.DL reg2, reg3

[ オペレーション ]        reg3 ← floor reg2(double    long-word)

[ フォーマット ]        Format F: I



[ 説 明 ]            汎用レジスタ reg2 で指定されるレジスタ・ペアにある倍精度浮動小数点形式の内容を算術的に 64 ビットの整数形式に変換し、汎用レジスタ reg3 で指定されるレジスタ・ペアに格納します。

現在の丸めモードに関係なく、結果を - の方向へ丸めます。

ソース・オペランドが無限大か非数の場合、または丸めた結果が  $2^{63}-1 \sim -2^{63}$  の範囲外の場合は、IEEE754 の無効演算例外を検出します。

無効演算例外の発生が許可されていない場合は、例外を発生せず、無効演算としてと FPSR レジスタの保存ビット (ビット 4) がセットされます。ソースの違いにより返す値は次のように異なります。

- ・ソースが正数または+            :  $2^{63}-1$  を返します。
- ・ソースが負数、非数または-        :  $-2^{63}$  を返します。

[ 浮動小数点演算例外 ] 無効演算例外 (V)  
不正確演算例外 (I)

[ 演算結果 ]            FPSR.FS = 1 の場合

reg2(A)	+Normal	-Normal	+0	-0	+	-	Q-NaN	S-NaN
演算結果 [例外]	A ( Integer )		0 ( Integer )		+Max Int[V]	-Max Int[V]		

- 備考 1. [ ] は必ず発生する例外です。
2. ディノーマル数はフラッシュされ " 0 " として扱われます。

< 浮動小数点演算命令 >

<p style="margin: 0;">Floating-point Truncate to Unsigned, rounded toward the direction of - (Double)</p> <h2 style="margin: 0;">FLOORF.DUL</h2> <p style="margin: 0; text-align: right;">符号なし整数形式への変換 (倍精度)</p>
--

[ 命令形式 ]            FLOORF.DUL reg2, reg3

[ オペレーション ]        reg3 ← floor reg2(double    Unsigned long-word)

[ フォーマット ]         Format F: I

[ オペコード ]

15	11	10	5	4	0	31	27	26	25	23	22	21	20	17	16																
r	r	r	r	0	1	1	1	1	1	1	1	0	0	1	1	w	w	w	w	0	1	0	0	0	1	0	1	0	1	0	0
reg2										reg3					category	type	sub-op														

[ 説 明 ]                 汎用レジスタ reg2 で指定されるレジスタ・ペアにある倍精度浮動小数点形式の内容を算術的に符号のない 64 ビットの整数形式に変換し、汎用レジスタ reg3 で指定されるレジスタ・ペアに格納します。

現在の丸めモードに関係なく、結果を- の方向へ丸めます。

ソース・オペランドが無限大か非数か負数の場合、または丸めた結果が  $2^{64}-1 \sim 0$  の範囲外の場合は、IEEE754 の無効演算例外を検出します。

無効演算例外の発生が許可されていない場合は、例外を発生せず、無効演算としてと FPSR レジスタの保存ビット (ビット 4) がセットされます。ソースの値により返す値は次のように異なります。

- ・ ソースが  $2^{64}-1 \sim 0$  の範囲外の正数または+                    :  $2^{64}-1$  を返します。
- ・ ソースが負数、非数または-                                         : 0 を返します。

[ 浮動小数点演算例外 ] 無効演算例外 (V)  
不正確演算例外 (I)

[ 演算結果 ]                FPSR.FS = 1 の場合

reg2(A)	+Normal	-Normal	+0	-0	+	-	Q-NaN	S-NaN
演算結果 [例外]	A (Integer)	0 [V]	0 (Integer)		Max U-Int [V]	0 [V]		

- 備考 1. [ ] は必ず発生する例外です。
2. ディノーマル数はフラッシュされ "0" として扱われます。

< 浮動小数点演算命令 >

Floating-point Truncate to Unsigned, rounded toward the direction of - (Double)

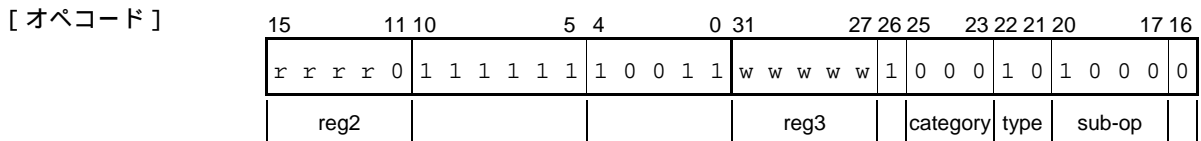
**FLOORF.DUW**

符号なし整数形式への変換 (倍精度)

[ 命令形式 ]            FLOORF.DUW reg2, reg3

[ オペレーション ]        reg3 ← floor reg2(double    Unsigned word)

[ フォーマット ]         Format F: I



[ 説 明 ]                汎用レジスタ reg2 で指定されるレジスタ・ペアにある倍精度浮動小数点形式の内容を算術的に符号のない 32 ビットの整数形式に変換し、汎用レジスタ reg3 に格納します。

現在の丸めモードに関係なく、結果を- の方向へ丸めます。

ソース・オペランドが無限大か非数か負数の場合、または丸めた結果が  $2^{32}-1 \sim 0$  の範囲外の場合は、IEEE754 の無効演算例外を検出します。

無効演算例外の発生が許可されていない場合は、例外を発生せず、無効演算としてと FPSR レジスタの保存ビット (ビット 4) がセットされます。ソースの値により返す値は次のように異なります。

- ・ ソースが  $2^{32}-1 \sim 0$  の範囲外の正数または+                                :  $2^{32}-1$  を返します。
- ・ ソースが負数、非数または-    : 0 を返します。

[ 浮動小数点演算例外 ] 無効演算例外 (V)  
不正確演算例外 (I)

[ 演算結果 ]            FPSR.FS = 1 の場合

reg2(A)	+Normal	-Normal	+0	-0	+	-	Q-NaN	S-NaN
演算結果 [例外]	A ( Integer )	0 [V]	0 ( Integer )		Max U-Int [V]	0 [V]		

備考 1. [ ] は必ず発生する例外です。  
2. ディノーマル数はフラッシュされ “ 0 ” として扱われます。

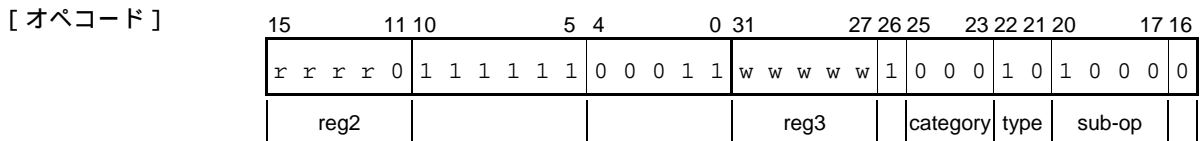
< 浮動小数点演算命令 >

<h2 style="margin: 0;">FLOORF.DW</h2>	Floating-point Truncate, rounded toward the direction of - (Double)  整数形式への変換 (倍精度)
---------------------------------------	---

[ 命令形式 ]            FLOORF.DW reg2, reg3

[ オペレーション ]        reg3 ← floor reg2(double word)

[ フォーマット ]         Format F: I



[ 説 明 ]            汎用レジスタ reg2 で指定されるレジスタ・ペアにある倍精度浮動小数点形式の内容を算術的に 32 ビットの整数形式に変換し、汎用レジスタ reg3 に格納します。

現在の丸めモードに関係なく、- の方向へ丸めます。

ソース・オペランドが無大か非数の場合、または丸めた結果が  $2^{31} - 1 \sim -2^{31}$  の範囲外の場合は、IEEE754 の無効演算例外を検出します。

無効演算例外の発生が許可されていない場合は、例外を発生せず、無効演算としてと FPSR レジスタの保存ビット (ビット 4) がセットされます。ソースの違いにより返す値は次のように異なります。

- ・ソースが正数または+ :  $2^{31} - 1$  を返します。
- ・ソースが負数、非数または- :  $-2^{31}$  を返します。

[ 浮動小数点演算例外 ] 無効演算例外 (V)  
不正確演算例外 (I)

[ 演算結果 ]            FPSR.FS = 1 の場合

reg2(A)	+Normal	-Normal	+0	-0	+	-	Q-NaN	S-NaN
演算結果 [例外]	A ( Integer )		0 ( Integer )		+Max Int[V]	-Max Int[V]		

- 備考 1. [ ] は必ず発生する例外です。
2. ディノーマル数はフラッシュされ " 0 " として扱われます。



< 浮動小数点演算命令 >

<p style="font-size: 24pt; margin: 0;">FLOORF.SL</p>	Floating-point Truncate, rounded toward the direction of - (Single)  整数形式への変換 (単精度)
--	---

[ 命令形式 ]            FLOORF.SL reg2, reg3

[ オペレーション ]    reg3 ← floor reg2(single    long-word)

[ フォーマット ]      Format F: I

[ オペコード ]

15	11	10	5	4	0	31	27	26	25	23	22	21	20	17	16																
r	r	r	r	r	1	1	1	1	1	1	0	0	0	1	1	w	w	w	w	0	1	0	0	0	1	0	0	0	1	0	0
reg2										reg3					category	type	sub-op														

[ 説 明 ]            汎用レジスタ reg2 にある単精度浮動小数点形式の内容を算術的に 64 ビットの整数形式に変換し、汎用レジスタ reg3 で指定されるレジスタ・ペアに格納します。

現在の丸めモードにかかわらず、結果を - の方向へ丸めます。

ソース・オペランドが無量大か非数の場合、または丸めた結果が  $2^{63}-1 \sim -2^{63}$  の範囲外の場合は、IEEE754 の無効演算例外を検出します。

無効演算例外の発生が許可されていない場合は、例外を発生せず、無効演算としてと FPSR レジスタの保存ビット (ビット 4) がセットされます。ソースの違いにより返す値は次のように異なります。

- ・ソースが正数または+            :  $2^{63}-1$  を返します。
- ・ソースが負数、非数または-    :  $-2^{63}$  を返します。

[ 浮動小数点演算例外 ] 無効演算例外 (V)  
不正確演算例外 (I)

[ 演算結果 ]            FPSR.FS = 1 の場合

reg2(A)	Normal	-Normal	+0	-0	+	-	Q-NaN	S-NaN
演算結果 [例外]	A ( Integer )		0 ( Integer )		Max Int[V]	-Max Int[V]		

備考 1. [ ] は必ず発生する例外です。  
2. ディノーマル数はフラッシュされ " 0 " として扱われます。

< 浮動小数点演算命令 >

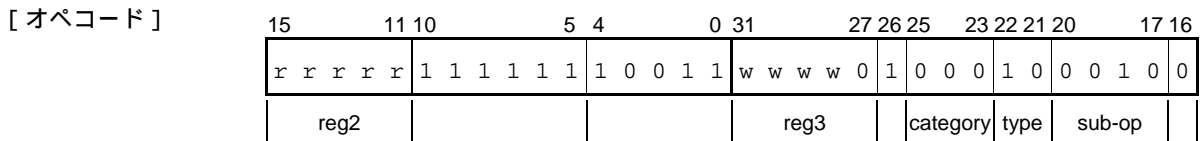
FLOORF.SUL
Floating-point Truncate to Unsigned, rounded toward the direction of - (Single)

符号なし整数形式への変換 (単精度)

[ 命令形式 ]            FLOORF.SUL reg2, reg3

[ オペレーション ]     reg3 ← floor reg2(Single Unsigned long-word)

[ フォーマット ]        Format F: I



[ 説 明 ]            汎用レジスタ reg2 にある単精度浮動小数点形式の内容を算術的に符号のない 64 ビットの整数形式に変換し、汎用レジスタ reg3 で指定されるレジスタ・ペアに格納します。現在の丸めモードに関係なく、結果を- の方向へ丸めます。ソース・オペランドが無限大か非数か負数の場合、または丸めた結果が  $2^{64}-1 \sim 0$  の範囲外の場合は、IEEE754 の無効演算例外を検出します。無効演算例外の発生が許可されていない場合は、例外を発生せず、無効演算としてと FPSR レジスタの保存ビット (ビット 4) がセットされます。ソースの値により返す値は次のように異なります。

- ・ ソースが  $2^{64}-1 \sim 0$  の範囲外の正数または+            :  $2^{64}-1$  を返します。
- ・ ソースが負数、非数または-                                : 0 を返します。

[ 浮動小数点演算例外 ] 無効演算例外 (V)  
不正確演算例外 (I)

[ 演算結果 ]            FPSR.FS = 1 の場合

reg2(A)	+Normal	-Normal	+0	-0	+	-	Q-NaN	S-NaN
演算結果 [例外]	A( Integer )	0 [V]	0 ( Integer )		Max U-Int [V]	0 [V]		

- 備考 1. [ ] は必ず発生する例外です。  
2. ディノーマル数はフラッシュされ "0" として扱われます。

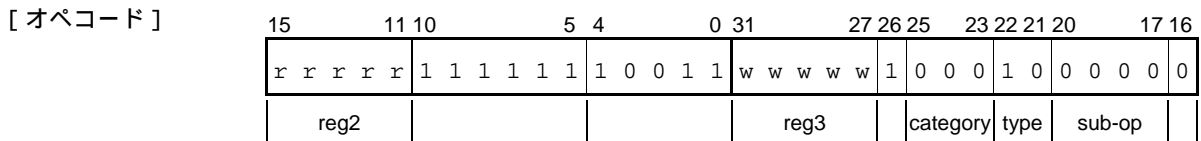
< 浮動小数点演算命令 >

<p style="font-size: 24pt; margin: 0;">FLOORF.SUW</p>	<p style="font-size: 10pt; margin: 0;">Floating-point Truncate to Unsigned, rounded toward the direction of - (Single)</p> <p style="font-size: 10pt; margin: 0;">符号なし整数形式への変換 (単精度)</p>
---	--

[ 命令形式 ]            FLOORF.SUW reg2, reg3

[ オペレーション ]        reg3 ← floor reg2(Single Unsigned word)

[ フォーマット ]         Format F: I



[ 説 明 ]                汎用レジスタ reg2 にある単精度浮動小数点数形式の内容を算術的に符号のない 32 ビットの整数形式に変換し、汎用レジスタ reg3 に格納します。

現在の丸めモードに関係なく、結果を- の方向へ丸めます。

ソース・オペランドが無限大か非数か負数の場合、または丸めた結果が  $2^{32}-1 \sim 0$  の範囲外の場合は、IEEE754 の無効演算例外を検出します。

無効演算例外の発生が許可されていない場合は、例外を発生せず、無効演算としてと FPSR レジスタの保存ビット (ビット 4) がセットされます。ソースの値により返す値は次のように異なります。

- ・ソースが  $2^{32}-1 \sim 0$  の範囲外の正数または+                                :  $2^{32}-1$  を返します。
- ・ソースが負数、非数または-    : 0 を返します。

[ 浮動小数点演算例外 ] 無効演算例外 (V)  
不正確演算例外 (I)

[ 演算結果 ]             FPSR.FS = 1 の場合

reg2(A)	+Normal	-Normal	+0	-0	+	-	Q-NaN	S-NaN
演算結果 [例外]	A( Integer )	0 [V]	0 ( Integer )		Max U-Int [V]	0 [V]		

備考 1. [ ] は必ず発生する例外です。  
2. ディノーマル数はフラッシュされ " 0 " として扱われます。

## &lt; 浮動小数点演算命令 &gt;

FLOORF.SW	Floating-point Truncate, rounded toward the direction of - (Single)
	整数形式への変換 (単精度)

[ 命令形式 ]            FLOORF.SW reg2, reg3

[ オペレーション ]        reg3 ← floor reg2(single word)

[ フォーマット ]         Format F: I

[ オペコード ]

15	11 10	5 4	0 31	27 26 25	23 22 21 20	17 16
r r r r r	1 1 1 1 1 1	0 0 0 1 1	w w w w w	1	0 0 0 1 0	0 0 0 0 0
reg2			reg3	category	type	sub-op

[ 説 明 ]                汎用レジスタ reg2 にある単精度浮動小数点形式の内容を算術的に 32 ビットの整数形式に変換し、汎用レジスタ reg3 に格納します。

現在の丸めモードに関係なく、- の方向へ丸めます。

ソース・オペランドが無大か非数の場合、または丸めた結果が  $2^{31}-1 \sim -2^{31}$  の範囲外の場合は、IEEE754 の無効演算例外を検出します。

無効演算例外の発生が許可されていない場合は、例外を発生せず、無効演算としてと FPSR レジスタの保存ビット (ビット 4) がセットされます。ソースの違いにより返す値は次のように異なります。

- ・ソースが正数または+ :  $2^{31}-1$  を返します。
- ・ソースが負数、非数または- :  $-2^{31}$  を返します。

[ 浮動小数点演算例外 ] 無効演算例外 (V)  
不正確演算例外 (I)

[ 演算結果 ]            FPSR.FS = 1 の場合

reg2(A)	+Normal	-Normal	+0	-0	+	-	Q-NaN	S-NaN
演算結果 [例外]	A ( Integer )		0 ( Integer )		+Max Int[V]		-Max Int[V]	

備考 1. [ ] は必ず発生する例外です。

2. ディノーマル数はフラッシュされ " 0 " として扱われます。

## &lt; 浮動小数点演算命令 &gt;

MADDF.S	Floating-point Multiply-add (Single)  浮動小数点積和算 (単精度)
---------	--

[ 命令形式 ]            MADDF.S reg1, reg2, reg3, reg4

[ オペレーション ]       $reg4 \leftarrow reg2 \times reg1 + reg3$

[ フォーマット ]        Format F: I

[ オペコード ]

15	11	10	5	4	0	31	27	26	25	23	22	21	20	17	16															
r	r	r	r	r	1	1	1	1	1	1	R	R	R	R	R	w	w	w	w	w	1	0	1	W	0	0	W	W	W	0
reg2					reg1					reg3					注1		注2		type		注2									

注 1. category

2. reg4 (最下位ビットはビット 23)

[ 説 明 ]            汎用レジスタ reg2 の内容と汎用レジスタ reg1 の内容を乗算した結果と、浮動小数点レジスタ reg3 の内容と加算し、結果を汎用レジスタ reg4 に格納します。演算は無限精度であるかのように実行し、結果を現在の丸めモードに従って丸めます。

[ 浮動小数点演算例外 ] 無効演算例外 (V)

不正確演算例外 (I)

オーバーフロー例外 (O)

アンダフロー例外 (U)

[ 演算結果 ] FPSR.FS = 1 の場合

reg3(C)	reg2(B)		+Normal	-Normal	+0	-0	+	-	Q-NaN	S-NaN	
	reg1(A)										
± Normal	+Normal	MADD(A, B, C)				+	-	Q-NaN[V]			
	-Normal	MADD(A, B, C)				-	+				
	± 0	MADD(A, B, C)				Q-NaN[V]					
	+	+	-	Q-NaN[V]		+	-				
	-	-	+	Q-NaN[V]		-	+				
± 0	+Normal	MADD(A, B, C)				+	-	Q-NaN[V]			
	-Normal	MADD(A, B, C)				-	+				
	± 0	MADD(A, B, C)				Q-NaN[V]					
	+	+	-	Q-NaN[V]		+	-				
	-	-	+	Q-NaN[V]		-	+				
+	+Normal	+				+	Q-NaN[V]	Q-NaN[V]			
	-Normal	+				Q-NaN[V]	+				
	± 0	+				Q-NaN[V]					
	+	+	Q-NaN[V]	Q-NaN[V]		+	Q-NaN[V]				
	-	Q-NaN[V]	+	Q-NaN[V]		Q-NaN[V]	+				
-	+Normal	-				Q-NaN[V]	-	Q-NaN[V]			
	-Normal	-				-	Q-NaN[V]				
	± 0	-				Q-NaN[V]					
	+	Q-NaN[V]	-	Q-NaN[V]		Q-NaN[V]	-				
	-	-	Q-NaN[V]	Q-NaN[V]		-	Q-NaN[V]				
Q-NaN	± Normal	Q-NaN									
	± 0	Q-NaN									
	±	Q-NaN									
S-NaN以外	Q-NaN	Q-NaN									
任意	S-NaN	Q-NaN									
S-NaN	任意	Q-NaN									Q-NaN[V]

備考 1. [ ] は必ず発生する例外です。

2. ディノーマル数はフラッシュされ“0”として扱われます。

[ 補 足 ] 乗算の結果による例外は、次の場合に発生します。

- ・乗算でオーバーフローし、加算が無効演算例外を起こさない場合

< 浮動小数点演算命令 >

<div style="display: flex; justify-content: space-between;"> <div style="font-size: 2em; font-weight: bold;">MAXF.D</div> <div style="text-align: right;">                     Floating-point Maximum (Double)                       浮動小数点最大値 (倍精度)                 </div> </div>
---

[ 命令形式 ]            MAXF.D reg1, reg2, reg3

[ オペレーション ]    reg3    max(reg2, reg1)

[ フォーマット ]      Format F: I

[ オペコード ]

15	11	10	5	4	0	31	27	26	25	23	22	21	20	17	16																
r	r	r	r	0	1	1	1	1	1	1	R	R	R	R	0	w	w	w	w	0	1	0	0	0	1	1	1	1	0	0	0
reg2					reg1					reg3			category	type	sub-op																

[ 説 明 ]            汎用レジスタ reg1 および reg2 で指定されるレジスタ・ペアにある倍精度浮動小数点形式のデータの中から最大値を汎用レジスタ reg3 で指定されるレジスタ・ペアに格納します。ソース・オペランドの 1 つが S-NaN の場合は、IEEE754 の無効演算例外を検出します。無効演算例外の発生が許可されていない場合は、例外を発生せず、Q-NaN を格納します。FPSR.FS ビットがセットされていて、reg1 および reg2 がともにディノーマル数、+0、-0 のいずれかである場合、reg3 に +0、または -0 を格納します。

[ 浮動小数点演算例外 ] 無効演算例外 (V)

[ 補 足 ]            FPSR.FS ビットがセットされていて、reg1 および reg2 がともにディノーマル数、+0、-0 のいずれかである場合、reg3 に +0、-0 のいずれかを格納するかは未定義です。

[ 演算結果 ]        FPSR.FS = 1 の場合

reg2(B) reg1(A) \	+Normal	-Normal	+0	-0	+	-	Q-NaN	S-NaN
+Normal	MAX ( A, B )						Q-NaN	S-NaN
-Normal								
+0								
-0								
+								
-								
Q-NaN							Q-NaN	S-NaN
S-NaN								Q-NaN[V]

- 備考 1. [ ] は必ず発生する例外です。  
 2. ディノーマル数はフラッシュされ“0”として扱われます。

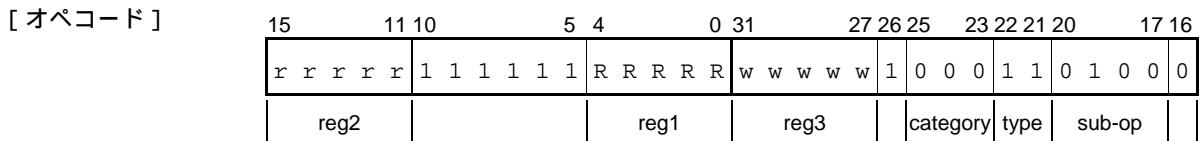
< 浮動小数点演算命令 >

MAXF.S	Floating-point Maximum (Single)  浮動小数点最大値 (単精度)
--------	---

[ 命令形式 ]            MAXF.S reg1, reg2, reg3

[ オペレーション ]    reg3    max(reg2, reg1)

[ フォーマット ]      Format F: I



[ 説 明 ]            汎用レジスタ reg1 および reg2 にある単精度浮動小数点形式のデータの中から最大値を汎用レジスタ reg3 に格納します。

ソース・オペランドの 1 つが S-NaN の場合は、IEEE754 の無効演算例外を検出します。無効演算例外の発生が許可されていない場合は、例外を発生せず、Q-NaN を格納します。FPSR.FS ビットがセットされていて、reg1 および reg2 がともにディノーマル数、+0、-0 のいずれかである場合、reg3 に +0、または -0 を格納します。

[ 浮動小数点演算例外 ] 無効演算例外 (V)

[ 補 足 ]            FPSR.FS ビットがセットされていて、reg1 および reg2 がともにディノーマル数、+0、-0 のいずれかである場合、reg3 に +0、-0 のいずれかを格納するかは未定義です。

[ 演算結果 ]        FPSR.FS = 1 の場合

reg2(B) reg1(A) \	Normal	-Normal	+0	-0	+	-	Q-NaN	S-NaN
Normal	MAX ( A, B )						Q-NaN	S-NaN
-Normal								
+0								
-0								
+								
-								
Q-NaN							Q-NaN	
S-NaN								Q-NaN[V]

- 備考 1. [ ] は必ず発生する例外です。  
 2. ディノーマル数はフラッシュされ “0” として扱われます。



< 浮動小数点演算命令 >

<h1 style="margin: 0;">MINF.D</h1>	Floating-point Minimum (Double)  浮動小数点最小値 (倍精度)
------------------------------------	---

[ 命令形式 ]            MINF.D    reg1, reg2, reg3

[ オペレーション ]    reg3    min(reg2, reg1)

[ フォーマット ]      Format F: I

[ オペコード ]

15	11	10	5	4	0	31	27	26	25	23	22	21	20	17	16																
r	r	r	r	0	1	1	1	1	1	1	R	R	R	R	0	w	w	w	w	0	1	0	0	0	1	1	1	1	0	1	0
reg2					reg1					reg3					category		type		sub-op												

[ 説 明 ]            汎用レジスタ reg1 および reg2 で指定されるレジスタ・ペアにある倍精度浮動小数点形式のデータの中から最小値を汎用レジスタ reg3 で指定されるレジスタ・ペアに格納します。ソース・オペランドの 1 つが S-NaN の場合は、IEEE754 の無効演算例外を検出します。無効演算例外の発生が許可されていない場合は、例外を発生せず、Q-NaN を格納します。FPSR.FS ビットがセットされていて、reg1 および reg2 がともにディノーマル数、+0、-0 のいずれかである場合、reg3 に +0、または -0 を格納します。

[ 浮動小数点演算例外 ] 無効演算例外 (V)

[ 補 足 ]            FPSR.FS ビットがセットされていて、reg1 および reg2 がともにディノーマル数、+0、-0 のいずれかである場合、reg3 に +0、-0 のいずれかを格納するかは未定義です。

[ 演算結果 ]        FPSR.FS = 1 の場合

reg2(B) reg1(A)	Normal	-Normal	+0	-0	+	-	Q-NaN	S-NaN
Normal	MIN ( A, B )						Q-NaN	S-NaN
-Normal								
+0								
-0								
+								
-								
Q-NaN							Q-NaN	
S-NaN								Q-NaN[V]

- 備考 1. [ ] は必ず発生する例外です。  
 2. ディノーマル数はフラッシュされ "0" として扱われます。

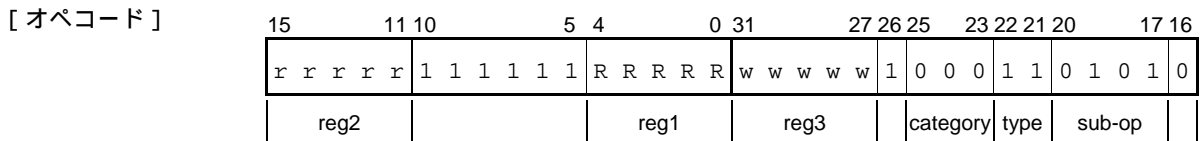
< 浮動小数点演算命令 >

<h1 style="margin: 0;">MINF.S</h1>	Floating-point Minimum (Single)  浮動小数点最小値 (単精度)
------------------------------------	---

[ 命令形式 ]            MINF.S    reg1, reg2, reg3

[ オペレーション ]    reg3    min(reg2, reg1)

[ フォーマット ]      Format F: I



[ 説 明 ]            汎用レジスタ reg1 および reg2 にある単精度浮動小数点形式のデータの中から最小値を汎用レジスタ reg3 に格納します。

ソース・オペランドの 1 つが S-NaN の場合は、IEEE754 の無効演算例外を検出します。無効演算例外の発生が許可されていない場合は、例外を発生せず、Q-NaN を格納します。FPSR.FS ビットがセットされていて、reg1 および reg2 がともにディノーマル数、+0、-0 のいずれかである場合、reg3 に +0、または -0 を格納します。

[ 浮動小数点演算例外 ] 無効演算例外 (V)

[ 補 足 ]            FPSR.FS ビットがセットされていて、reg1 および reg2 がともにディノーマル数、+0、-0 のいずれかである場合、reg3 に +0、-0 のいずれかを格納するかは未定義です。

[ 演算結果 ]        FPSR.FS = 1 の場合

reg2(B) reg1(A) \	Normal	-Normal	+0	-0	+	-	Q-NaN	S-NaN
Normal	MIN ( A, B )						Q-NaN	S-NaN
-Normal								
+0								
-0								
+								
-								
Q-NaN							Q-NaN	
S-NaN								Q-NaN[V]

- 備考 1. [ ] は必ず発生する例外です。
2. ディノーマル数はフラッシュされ “0” として扱われます。

## &lt; 浮動小数点演算命令 &gt;

MSUBF.S	Floating-point Multiply-subtract (Single)  浮動小数点積和算 (単精度)
---------	---

[ 命令形式 ]            MSUBF.S reg1, reg2, reg3, reg4

[ オペレーション ]         $reg4 \leftarrow reg2 \times reg1 - reg3$

[ フォーマット ]         Format F: I

[ オペコード ]

15	11	10	5	4	0	31	27	26	25	23	22	21	20	17	16																
r	r	r	r	r	1	1	1	1	1	1	R	R	R	R	R	w	w	w	w	w	1	0	1	W	0	1	W	W	W	W	0
reg2					reg1					reg3					注1		注2		type		注2										

注 1. category

2. reg4 (最下位ビットはビット 23)

[ 説 明 ]                汎用レジスタ reg2 にある単精度浮動小数点形式の内容と汎用レジスタ reg1 にある単精度浮動小数点形式の内容を乗算した結果から、汎用レジスタ reg3 にある単精度浮動小数点形式の内容を減算し、結果を汎用レジスタ reg4 に格納します。演算は無限精度であるかのように実行し、結果を現在の丸めモードに従って丸めます。

[ 浮動小数点演算例外 ] 無効演算例外 (V)

不正確演算例外 (I)

オーバフロー例外 (O)

アンダフロー例外 (U)

[ 演算結果 ]                  FPSR.FS = 1 の場合

reg3(C) \ reg2(B) reg1(A)		+Normal	-Normal	+0	-0	+	-	Q-NaN	S-NaN	
		± Normal	+Normal	MUSB ( A, B, C )				+	-	Q-NaN
-Normal	MUSB ( A, B, C )				-	+				
± 0	MUSB ( A, B, C )				Q-NaN[V]					
+	+		-	Q-NaN[V]		+	-			
-	-		+	Q-NaN[V]		-	+			
± 0	+Normal	MUSB ( A, B, C )				+	-	Q-NaN	S-NaN	
	-Normal	MUSB ( A, B, C )				-	+			
	± 0	MUSB ( A, B, C )				Q-NaN[V]				
	+	+	-	Q-NaN[V]		+	-			
	-	-	+	Q-NaN[V]		-	+			
+	+Normal	-				Q-NaN[V]	-	Q-NaN	S-NaN	
	-Normal	-				-	Q-NaN[V]			
	± 0	-				Q-NaN[V]				
	+	Q-NaN[V]	-	Q-NaN[V]		Q-NaN[V]	-			
	-	-	Q-NaN[V]	Q-NaN[V]		-	Q-NaN[V]			
-	+Normal	+				+	Q-NaN[V]	Q-NaN	S-NaN	
	-Normal	+				Q-NaN[V]	+			
	± 0	+				Q-NaN[V]				
	+	+	Q-NaN[V]	Q-NaN[V]		+	Q-NaN[V]			
	-	Q-NaN[V]	+	Q-NaN[V]		Q-NaN[V]	+			
Q-NaN	± Normal	Q-NaN							Q-NaN	S-NaN
	± 0	Q-NaN								
	±	Q-NaN								
S-NaN以外	Q-NaN	Q-NaN							Q-NaN	S-NaN
任意	S-NaN	Q-NaN							Q-NaN	任意
S-NaN	任意	Q-NaN							Q-NaN	任意

備考 1. [ ] は必ず発生する例外です。

2. ディノーマル数はフラッシュされ “ 0 ” として扱われます。

[ 補 足 ]                  乗算の結果による例外は，次の場合に発生します。

- ・乗算でオーバフローし，加算が無効演算例外を起こさない場合

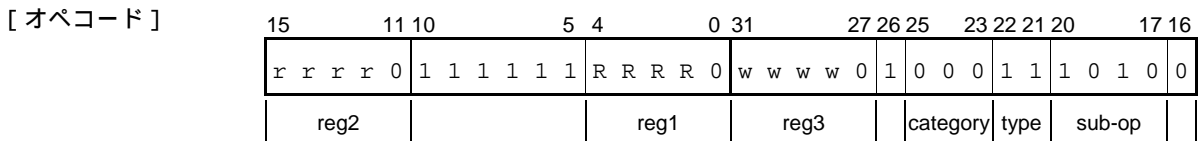
< 浮動小数点演算命令 >

<b>MULF.D</b>	Floating-point Multiply (Double)  浮動小数点乗算 (倍精度)
---------------	---

[ 命令形式 ]            MULF.D reg1, reg2, reg3

[ オペレーション ]    reg3 ← reg2 × reg1

[ フォーマット ]        Format F: I



[ 説 明 ]                汎用レジスタ reg2 で指定されるレジスタ・ペアにある倍精度浮動小数点形式の内容と汎用レジスタ reg1 で指定されるレジスタ・ペアにある倍精度浮動小数点形式の内容を乗算し、汎用レジスタ reg3 に格納します。

- [ 浮動小数点演算例外 ] 無効演算例外 (V)  
 不正確演算例外 (I)  
 オーバフロー例外 (O)  
 アンダフロー例外 (U)

[ 演算結果 ]            FPSR.FS = 1 の場合

reg2(B) reg1(A)	Normal	-Normal	+0	-0	+	-	Q-NaN	S-NaN
Normal	A × B				+	-	Q-NaN	Q-NaN[V]
-Normal					-	+		
+0					Q-NaN[V]			
-0					Q-NaN[V]			
+	+	-	Q-NaN[V]		+	-		
-	-	+	Q-NaN[V]		-	+		
Q-NaN	Q-NaN							
S-NaN	Q-NaN[V]							

- 備考 1. [ ] は必ず発生する例外です。  
 2. ディノーマル数はフラッシュされ “ 0 ” として扱われます。

## &lt; 浮動小数点演算命令 &gt;

MULF.S	Floating-point Multiply (Single)  浮動小数点乗算 (単精度)
--------	---

[ 命令形式 ]            MULF.S reg1, reg2, reg3

[ オペレーション ]        reg3 ← reg2 × reg1

[ フォーマット ]         Format F: I

[ オペコード ]

15	11	10	5	4	0	31	27	26	25	23	22	21	20	17	16			
r	r	r	r	r	1	1	1	1	1	1	R	R	R	R	R			
reg2					reg1					reg3					category	type	sub-op	

[ 説 明 ]                 汎用レジスタ reg2 にある単精度浮動小数点形式の内容と汎用レジスタ reg1 にある単精度浮動小数点形式の内容を乗算し、汎用レジスタ reg3 に格納します。

[ 浮動小数点演算例外 ] 無効演算例外 (V)

不正確演算例外 (I)

オーバーフロー例外 (O)

アンダフロー例外 (U)

[ 演算結果 ]             FPSR.FS = 1 の場合

reg2(B) reg1(A)	Normal	-Normal	+0	-0	+	-	Q-NaN	S-NaN
Normal	A × B				+	-	Q-NaN	Q-NaN[V]
-Normal					-	+		
+0					Q-NaN[V]			
-0					Q-NaN[V]			
+	+	-	Q-NaN[V]		+	-	Q-NaN	Q-NaN[V]
-	-	+	-	+				
Q-NaN	Q-NaN						Q-NaN	Q-NaN[V]
S-NaN	Q-NaN							

備考 1. [ ] は必ず発生する例外です。

2. ディノーマル数はフラッシュされ“0”として扱われます。

## &lt; 浮動小数点演算命令 &gt;

NEGF.D	Floating-point Negate (Double)  浮動小数点符号反転 (倍精度)
--------	---

[ 命令形式 ]            NEGF.D   reg2, reg3

[ オペレーション ]       reg3 ← neg reg2

[ フォーマット ]        Format F: I

[ オペコード ]

15	11	10	5	4	0	31	27	26	25	23	22	21	20	17	16																
r	r	r	r	0	1	1	1	1	1	1	0	0	0	0	1	w	w	w	w	0	1	0	0	0	1	0	1	1	0	0	0
reg2										reg3					category	type	sub-op														

[ 説 明 ]                汎用レジスタ reg2 で指定されるレジスタ・ペアにある倍精度浮動小数点形式の内容の符号を反転し、汎用レジスタ reg3 に格納します。  
符号の反転は算術的に行います。すなわち、オペランドが S-NaN の場合は、IEEE754 の無効演算例外を検出します。

[ 浮動小数点演算例外 ] 無効演算例外 (V)  
不正確演算例外 (I)

[ 演算結果 ]            FPSR.FS = 1 の場合

reg2	+Normal	-Normal	+0	-0	+	-	Q-NaN	S-NaN
演算結果 [例外]	-Normal	+Normal	-0	+0	-	+	Q-NaN	Q-NaN[V]

備考 1. [ ] は必ず発生する例外です。

2. ディノーマル数はフラッシュされ“0”として扱われます。

## &lt; 浮動小数点演算命令 &gt;

NEGF.S	Floating-point Negate (Single)  浮動小数点符号反転 (単精度)
--------	---

[ 命令形式 ]            NEGF.S reg2, reg3

[ オペレーション ]        reg3 ← neg reg2

[ フォーマット ]         Format F: I

[ オペコード ]

15	11	10	5	4	0	31	27	26	25	23	22	21	20	17	16																
r	r	r	r	r	1	1	1	1	1	1	0	0	0	0	1	w	w	w	w	w	1	0	0	0	1	0	0	1	0	0	0
reg2										reg3					category		type		sub-op												

[ 説 明 ]                 汎用レジスタ reg2 にある単精度浮動小数点形式の内容の符号を反転し、汎用レジスタ reg3 に格納します。

符号の反転は算術的に行います。すなわち、オペランドが S-NaN の場合は、IEEE754 の無効演算例外を検出します。

[ 浮動小数点演算例外 ] 無効演算例外 (V)

不正確演算例外 (I)

[ 演算結果 ]             FPSR.FS = 1 の場合

reg2	+Normal	-Normal	+0	-0	+	-	Q-NaN	S-NaN
演算結果 [例外]	-Normal	+Normal	-0	+0	-	+	Q-NaN	Q-NaN[V]

備考 1. [ ] は必ず発生する例外です。

2. ディノーマル数はフラッシュされ“0”として扱われます。



## &lt; 浮動小数点演算命令 &gt;

<b>NMADDF.S</b>	Floating-point Negate Multiply-add (Single)  浮動小数点積和算 (単精度)
-----------------	---

[ 命令形式 ]            NMADDF.S reg1, reg2, reg3, reg4

[ オペレーション ]         $reg4 \leftarrow \text{neg}(reg2 \times reg1 + reg3)$

[ フォーマット ]         Format F: I

[ オペコード ]

15	11	10	5	4	0	31	27	26	25	23	22	21	20	17	16																
r	r	r	r	r	1	1	1	1	1	1	R	R	R	R	R	w	w	w	w	w	1	0	1	W	1	0	W	W	W	W	0
reg2					reg1					reg3					注1		注2		type		注2										

注 1. category

2. reg4 (最下位ビットはビット 23)

[ 説 明 ]            汎用レジスタ reg2 にある単精度浮動小数点形式の内容と汎用レジスタ reg1 にある単精度浮動小数点形式の内容を乗算した結果に、汎用レジスタ reg3 にある単精度浮動小数点形式の内容を加算します。結果の符号を反転して汎用レジスタ reg4 に格納します。演算は無限精度であるかのように実行し、結果を現在の丸めモードに従って丸めます。符号の反転は算術的に行います。すなわち、オペランドが S-NaN の場合は、IEEE754 の無効演算例外を検出します。

[ 浮動小数点演算例外 ] 無効演算例外 (V)

不正確演算例外 (I)

オーバフロー例外 (O)

アンダフロー例外 (U)

[ 演算結果 ]          FPSR.FS = 1 の場合

reg3(C) \ reg2(B) reg1(A)		+Normal	-Normal	+0	-0	+	-	Q-NaN	S-NaN	
		± Normal	+Normal	NMADD ( A, B, C )				-	+	Q-NaN
-Normal	NMADD ( A, B, C )				+	-				
± 0	NMADD ( A, B, C )				Q-NaN[V]					
+	-		+	Q-NaN[V]		-	+			
-	+		-	Q-NaN[V]		+	-			
± 0	+Normal	NMADD ( A, B, C )				-	+	Q-NaN	S-NaN	
	-Normal	NMADD ( A, B, C )				+	-			
	± 0	NMADD ( A, B, C )				Q-NaN[V]				
	+	-	+	Q-NaN[V]		-	+			
	-	+	-	Q-NaN[V]		+	-			
+	+Normal	-				-	Q-NaN[V]	Q-NaN	S-NaN	
	-Normal	-				Q-NaN[V]	-			
	± 0	-				Q-NaN[V]				
	+	-	Q-NaN[V]	Q-NaN[V]		-	Q-NaN[V]			
	-	Q-NaN[V]	-	Q-NaN[V]		Q-NaN[V]	-			
-	+Normal	+				Q-NaN[V]	+	Q-NaN	S-NaN	
	-Normal	+				+	Q-NaN[V]			
	± 0	+				Q-NaN[V]				
	+	Q-NaN[V]	+	Q-NaN[V]		Q-NaN[V]	+			
	-	+	Q-NaN[V]	Q-NaN[V]		+	Q-NaN[V]			
Q-NaN	± Normal	Q-NaN							Q-NaN	S-NaN
	± 0	Q-NaN								
	±	Q-NaN								
S-NaN以外	Q-NaN	Q-NaN							Q-NaN	S-NaN
任意	S-NaN	Q-NaN							Q-NaN	S-NaN
S-NaN	任意	Q-NaN							Q-NaN	Q-NaN[V]

備考 1. [ ] は必ず発生する例外です。

2. ディノーマル数はフラッシュされ “ 0 ” として扱われます。

[ 補 足 ]          乗算の結果による例外は，次の場合に発生します。

- ・乗算でオーバフローし，加算が無効演算例外を起こさない場合

## &lt; 浮動小数点演算命令 &gt;

NMSUBF.S	Floating-point Negate Multiply-subtrat (Single)  浮動小数点積和算 (単精度)
----------	---

[ 命令形式 ]            NMSUBF.S reg1, reg2, reg3, reg4

[ オペレーション ]         $reg4 \leftarrow \text{neg}(reg2 \times reg1 - reg3)$

[ フォーマット ]         Format F: I

[ オペコード ]

15	11	10	5	4	0	31	27	26	25	23	22	21	20	17	16															
r	r	r	r	r	1	1	1	1	1	R	R	R	R	R	w	w	w	w	w	1	0	1	W	1	1	W	W	W	W	0
reg2					reg1					reg3					注1	注2	type			注2										

注 1. category

2. reg4 (最下位ビットはビット 23)

[ 説 明 ]            汎用レジスタ reg2 にある単精度浮動小数点形式の内容と汎用レジスタ reg1 にある単精度浮動小数点形式の内容を乗算した結果から、汎用レジスタ reg3 にある単精度浮動小数点形式の内容を減算します。結果の符号を反転して汎用レジスタ reg4 に格納します。演算は無限精度であるかのように実行し、結果を現在の丸めモードに従って丸めます。符号の反転は算術的に行います。すなわち、オペランドが S-NaN の場合は、IEEE754 の無効演算例外を検出します。

[ 浮動小数点演算例外 ] 無効演算例外 (V)

不正確演算例外 (I)

オーバフロー例外 (O)

アンダフロー例外 (U)

[ 演算結果 ]          FPSR.FS = 1 の場合

reg3(C) \ reg2(B) reg1(A)		+Normal	-Normal	+0	-0	+	-	Q-NaN	S-NaN
		± Normal	+Normal	NMSUB ( A, B, C )				-	+
-Normal	NMSUB ( A, B, C )				+	-			
± 0	Q-NaN[V]								
+	-		+	Q-NaN[V]		-	+		
-	+		-	Q-NaN[V]		+	-		
± 0	+Normal	NMSUB ( A, B, C )				-	+	Q-NaN	S-NaN
	-Normal	NMSUB ( A, B, C )				+	-		
	± 0	Q-NaN[V]							
	+	-	+	Q-NaN[V]		-	+		
	-	+	-	Q-NaN[V]		+	-		
+	+Normal	+				Q-NaN[V]	+	Q-NaN	S-NaN
	-Normal	+				+	Q-NaN[V]		
	± 0	Q-NaN[V]							
	+	Q-NaN[V]	+	Q-NaN[V]		Q-NaN[V]	+		
	-	+	Q-NaN[V]	Q-NaN[V]		+	Q-NaN[V]		
-	+Normal	-				-	Q-NaN[V]	Q-NaN	S-NaN
	-Normal	-				Q-NaN[V]	-		
	± 0	Q-NaN[V]							
	+	-	Q-NaN[V]	Q-NaN[V]		-	Q-NaN[V]		
	-	Q-NaN[V]	-	Q-NaN[V]		Q-NaN[V]	-		
Q-NaN	± Normal	Q-NaN						Q-NaN	S-NaN
	± 0	Q-NaN							
	±	Q-NaN							
S-NaN以外	Q-NaN							Q-NaN	S-NaN
任意	S-NaN							Q-NaN[V]	
S-NaN	任意							Q-NaN[V]	

備考 1. [ ] は必ず発生する例外です。

2. ディノーマル数はフラッシュされ “ 0 ” として扱われます。

[ 補 足 ]          乗算の結果による例外は，次の場合に発生します。

- ・乗算でオーバフローし，加算が無効演算例外を起こさない場合

## &lt; 浮動小数点演算命令 &gt;

<b>RECIPF.D</b>	Reciprocal of a Floating-point Value (Double)  逆数 (倍精度)
-----------------	---

[ 命令形式 ]            RECIPF.D reg2, reg3

[ オペレーション ]       $reg3 \leftarrow 1 \div reg2$

[ フォーマット ]        Format F: I

[ オペコード ]

15	11	10	5	4	0	31	27	26	25	23	22	21	20	17	16																
r	r	r	r	0	1	1	1	1	1	1	0	0	0	0	1	w	w	w	w	0	1	0	0	0	1	0	1	1	1	1	0
reg2										reg3					category	type	sub-op														

[ 説 明 ]                汎用レジスタ reg2 で指定されるレジスタ・ペアにある倍精度浮動小数点形式の内容の逆数を近似し、汎用レジスタ reg3 で指定されるレジスタ・ペアに格納します。演算は無限精度であるかのように実行し、結果を現在の丸めモードに従って丸めます。

[ 浮動小数点演算例外 ] 無効演算例外 (V)

不正確演算例外 (I)

ゼロ除算例外 (Z)

オーバフロー例外 (O)

アンダフロー例外 (U)

[ 演算結果 ]            FPSR.FS = 1 の場合

reg2(A)	Normal	-Normal	+0	-0	+	-	Q-NaN	S-NaN
演算結果 [例外]	1/A[ I ]		+ [Z]	- [Z]	+0	-0	Q-NaN	Q-NaN[V]

備考 1. [ ] は必ず発生する例外です。

2. ディノーマル数はフラッシュされ“0”として扱われます。

## &lt; 浮動小数点演算命令 &gt;

<p>RECIPF.S</p>	<p>Reciprocal of a Floating-point Value (Single)</p> <p>逆数 (単精度)</p>
-----------------	--

[ 命令形式 ]            RECIPF.S    reg2, reg3

[ オペレーション ]      reg3 ← 1 ÷ reg2

[ フォーマット ]        Format F: I

[ オペコード ]

15	11	10	5	4	0	31	27	26	25	23	22	21	20	17	16																
r	r	r	r	r	1	1	1	1	1	1	0	0	0	0	1	w	w	w	w	w	1	0	0	0	1	0	0	1	1	1	0
reg2					reg3					category	type	sub-op																			

[ 説 明 ]                汎用レジスタ reg2 にある単精度浮動小数点形式の内容の逆数を近似し、汎用レジスタ reg3 に格納します。演算は無限精度であるかのように実行し、結果を現在の丸めモードに従って丸めます。

[ 浮動小数点演算例外 ] 無効演算例外 (V)

不正確演算例外 (I)

ゼロ除算例外 (Z)

オーバフロー例外 (O)

アンダフロー例外 (U)

[ 演算結果 ]            FPSR.FS = 1 の場合

reg2(A)	Normal	-Normal	+0	-0	+	-	Q-NaN	S-NaN
演算結果 [例外]	1/A[ I ]		+ [Z]	- [Z]	+0	-0	Q-NaN	Q-NaN[V]

備考 1. [ ] は必ず発生する例外です。

2. ディノーマル数はフラッシュされ“0”として扱われます。

## &lt; 浮動小数点演算命令 &gt;

<b>RSQRTF.D</b>	Reciprocal of the Square Root of a Floating-point Value (Double)  平方根の逆数 (倍精度)
-----------------	--

[ 命令形式 ]           RSQRTF.D reg2, reg3

[ オペレーション ]       reg3 ← 1 ÷ (sqrt reg2)

[ フォーマット ]       Format F: I

[ オペコード ]

15	11	10	5	4	0	31	27	26	25	23	22	21	20	17	16																		
r	r	r	r	0	1	1	1	1	1	1	0	0	0	1	0	w	w	w	w	0	1	0	0	0	1	0	1	1	1	1	0		
reg2										reg3					category	type	sub-op																

[ 説 明 ]           汎用レジスタ reg2 で指定されるレジスタ・ペアにある倍精度浮動小数点形式の内容の正の算術平方根の逆数を近似し、汎用レジスタ reg3 で指定されるレジスタ・ペアに格納します。演算は無限精度であるかのように実行し、結果を現在の丸めモードに従って丸めます。

[ 浮動小数点演算例外 ] 無効演算例外 (V)

不正確演算例外 (I)

ゼロ除算例外 (Z)

オーバフロー例外 (O)

アンダフロー例外 (U)

[ 演算結果 ]           FPSR.FS = 1 の場合

reg2(A)	Normal	-Normal	+0	-0	+	-	Q-NaN	S-NaN
演算結果 [例外]	1/ A[I]	Q-NaN[V]	+ [Z]	- [Z]	+0	Q-NaN[V]	Q-NaN	Q-NaN[V]

備考 1. [ ] は必ず発生する例外です。

2. ディノーマル数はフラッシュされ“0”として扱われます。

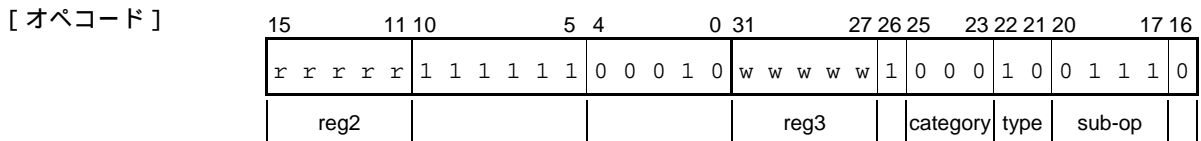
< 浮動小数点演算命令 >

<p style="font-size: 24px; margin: 0;">RSQRTF.S</p>	<p style="font-size: 12px; margin: 0;">Reciprocal of the Square Root of a Floating-point Value (Single)</p> <p style="font-size: 12px; margin: 0;">平方根の逆数 (単精度)</p>
---	---

[ 命令形式 ]            RSQRTF.S reg2, reg3

[ オペレーション ]        reg3 ← 1 ÷ (sqrt reg2)

[ フォーマット ]         Format F: I



[ 説 明 ]                 汎用レジスタ reg2 にある単精度浮動小数点形式の内容の正の算術平方根の逆数を近似し、汎用レジスタ reg3 に格納します。演算は無限精度であるかのように実行し、結果を現在の丸めモードに従って丸めます。

- [ 浮動小数点演算例外 ] 無効演算例外 (V)  
 不正確演算例外 (I)  
 ゼロ除算例外 (Z)  
 オーバフロー例外 (O)  
 アンダフロー例外 (U)

[ 演算結果 ]             FPSR.FS = 1 の場合

reg2(A)	Normal	-Normal	+0	-0	+	-	Q-NaN	S-NaN
演算結果 [例外]	1/ A[I]	Q-NaN[V]	+ [Z]	- [Z]	+0	Q-NaN[V]	Q-NaN	Q-NaN[V]

- 備考 1. [ ] は必ず発生する例外です。  
 2. ディノーマル数はフラッシュされ“0”として扱われます。



## &lt; 浮動小数点演算命令 &gt;

<p style="font-size: 24pt; margin: 0;">SQRTF.D</p>	Floating-point Square Root (Double)  平方根 (倍精度)
--	--

[ 命令形式 ]            SQRTF.D reg2, reg3

[ オペレーション ]        reg3 ← sqrt reg2

[ フォーマット ]         Format F: I

[ オペコード ]

	15		11	10		5	4		0	31		27	26	25		23	22	21	20		17	16										
	r	r	r	r	0	1	1	1	1	1	1	0	0	0	0	0	w	w	w	w	0	1	0	0	0	1	0	1	1	1	1	0
	reg2										reg3					category	type	sub-op														

[ 説 明 ]                 汎用レジスタ reg2 で指定されるレジスタ・ペアにある倍精度浮動小数点形式の内容の正の算術平方根を求め、汎用レジスタ reg3 で指定されるレジスタ・ペアに格納します。演算は無窮精度であるかのように実行し、結果を現在の丸めモードに従って丸めます。ソース・オペランドの値が - 0 の場合、結果は - 0 になります。

[ 浮動小数点演算例外 ] 無効演算例外 (V)  
不正確演算例外 (I)

[ 演算結果 ]             FPSR.FS = 1 の場合

reg2(A)	Normal	-Normal	+0	-0	+	-	Q-NaN	S-NaN
演算結果 [例外]	A	Q-NaN[V]	+0	-0	+	Q-NaN[V]	Q-NaN	Q-NaN[V]

備考 1. [ ] は必ず発生する例外です。

2. ディノーマル数はフラッシュされ "0" として扱われます。

## &lt; 浮動小数点演算命令 &gt;

SQRTF.S	Floating-point Square Root (Single)  平方根 (単精度)
---------	--

[ 命令形式 ]            SQRTF.S reg2, reg3

[ オペレーション ]        reg3 ← sqrt reg2

[ フォーマット ]         Format F: I

[ オペコード ]

15	11 10	5 4	0 31	27 26 25	23 22 21 20	17 16
r r r r r	1 1 1 1 1 1	0 0 0 0 0	w w w w w	1 0 0 0	1 0 0 1 1 1	0
reg2			reg3	category	type	sub-op

[ 説 明 ]                 汎用レジスタ reg2 にある単精度浮動小数点形式の内容の正の算術平方根を求め、汎用レジスタ reg3 に格納します。演算は無限精度であるかのように実行し、結果を現在の丸めモードに従って丸めます。ソース・オペランドの値が - 0 の場合、結果は - 0 になります。

[ 浮動小数点演算例外 ] 無効演算例外 (V)  
不正確演算例外 (I)

[ 演算結果 ]             FPSR.FS = 1 の場合

reg2(A)	Normal	-Normal	+0	-0	+	-	Q-NaN	S-NaN
演算結果 [例外]	A	Q-NaN[V]	+0	-0	+	Q-NaN[V]	Q-NaN	Q-NaN[V]

備考 1. [ ] は必ず発生する例外です。

2. ディノーマル数はフラッシュされ "0" として扱われます。

## &lt; 浮動小数点演算命令 &gt;

SUBF.D	Floating-point Subtract (Double)  浮動小数点減算 (倍精度)
--------	---

[ 命令形式 ]           SUBF.D reg1, reg2, reg3

[ オペレーション ]       reg3 ← reg2 - reg1

[ フォーマット ]       Format F: I

[ オペコード ]

15	11	10	5	4	0	31	27	26	25	23	22	21	20	17	16																
r	r	r	r	0	1	1	1	1	1	1	R	R	R	R	0	w	w	w	w	0	1	0	0	0	1	1	1	0	0	1	0
reg2					reg1					reg3					category		type		sub-op												

[ 説 明 ]               汎用レジスタ reg2 で指定されるレジスタ・ペアにある倍精度浮動小数点形式の内容から汎用レジスタ reg1 で指定されるレジスタ・ペアにある倍精度浮動小数点形式の内容を減算し、汎用レジスタ reg3 で指定されるレジスタ・ペアに格納します。演算は無限精度であるかのように実行し、結果を現在の丸めモードに従って丸めます。

[ 浮動小数点演算例外 ] 無効演算例外 (V)

不正確演算例外 (I)

オーバフロー例外 (O)

アンダフロー例外 (U)

[ 演算結果 ]           FPSR.FS = 1 の場合

reg2(B) reg1(A)	Normal	-Normal	+0	-0	+	-	Q-NaN	S-NaN
Normal	B-A				+	-	Q-NaN	S-NaN
-Normal								
+0								
-0								
+	-			Q-NaN[V]		Q-NaN[V]	Q-NaN	Q-NaN[V]
-	+		Q-NaN[V]					
Q-NaN							Q-NaN	
S-NaN								Q-NaN[V]

備考 1. [ ] は必ず発生する例外です。

2. ディノーマル数はフラッシュされ “0” として扱われます。

< 浮動小数点演算命令 >

SUBF.S	Floating-point Subtract (Single)  浮動小数点減算 (単精度)
--------	---

[ 命令形式 ]           SUBF.S reg1, reg2, reg3

[ オペレーション ]    reg3 ← reg2 - reg1

[ フォーマット ]      Format F: I

[ オペコード ]

15	11	10	5	4	0	31	27	26	25	23	22	21	20	17	16																
r	r	r	r	r	1	1	1	1	1	1	R	R	R	R	R	w	w	w	w	w	1	0	0	0	1	1	0	0	0	1	0
reg2					reg1					reg3					category		type		sub-op												

[ 説 明 ]           汎用レジスタ reg2 にある単精度浮動小数点形式の内容から汎用レジスタ reg1 にある単精度浮動小数点形式の内容を減算し、汎用レジスタ reg3 に格納します。演算は無限精度であるかのように実行し、結果を現在の丸めモードに従って丸めます。

- [ 浮動小数点演算例外 ] 無効演算例外 (V)  
 不正確演算例外 (I)  
 オーバフロー例外 (O)  
 アンダフロー例外 (U)

[ 演算結果 ]         FPSR.FS = 1 の場合

reg2(B) reg1(A)	Normal	-Normal	+0	-0	+	-	Q-NaN	S-NaN
Normal	B-A				+	-	Q-NaN	S-NaN
-Normal								
+0								
-0								
+	-			Q-NaN[V]				
-	+			Q-NaN[V]				
Q-NaN	Q-NaN							
S-NaN	Q-NaN[V]							

- 備考 1. [ ] は必ず発生する例外です。  
 2. ディノーマル数はフラッシュされ “ 0 ” として扱われます。

## &lt; 浮動小数点演算命令 &gt;

TRFSR	Transfer Floating Flags  フラグ転送
-------	--------------------------------------

[ 命令形式 ]            TRFSR fcbit

TRFSR

[ オペレーション ]    PSW.Z    fcbit

[ フォーマット ]      Format F: I

[ オペコード ]

15	11	10	5	4	0	31	27	26	25	23	22	21	20	17	16
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
				reg3				category		type		sub-op			

備考 fcbit : fff

[ 説 明 ]            fcbit によって指定された FPSR レジスタのコンディション・ビット ( CC(7:0)ビット : ビット 31-24 ) を , PSW 内の Z フラグへ転送します。 fcbit が省略された場合は CC0 ( ビット 24 ) を転送します。

[ 浮動小数点演算例外 ] なし

< 浮動小数点演算命令 >

<div style="display: flex; justify-content: space-between;"> <div style="font-size: 24pt; font-weight: bold;">TRNCF.DL</div> <div style="text-align: right; font-size: 10pt;">Floating-point Truncate to Integer Format, rounded to zero (Double)</div> </div> <div style="text-align: right; font-size: 10pt; margin-top: 5px;">整数形式への変換 (倍精度)</div>
---

[ 命令形式 ]            TRNCF.DL reg2, reg3

[ オペレーション ]        reg3 ← trunc reg2(double long-word)

[ フォーマット ]         Format F: I

[ オペコード ]

	15		11	10		5	4		0	31		27	26	25		23	22	21	20		17	16										
	r	r	r	r	0	1	1	1	1	1	1	0	0	0	0	1	w	w	w	w	0	1	0	0	0	1	0	1	0	1	0	0
	reg2								reg3				category	type	sub-op																	

[ 説 明 ]                汎用レジスタ reg2 で指定されるレジスタ・ペアにある倍精度浮動小数点形式の内容を算術的に 64 ビットの整数形式に変換し、汎用レジスタ reg3 で指定されるレジスタ・ペアに格納します。

現在の丸めモードに関係なく、結果を 0 の方向へ丸めます。

ソース・オペランドが無限大か非数の場合、または丸めた結果が  $2^{63}-1 \sim -2^{63}$  の範囲外の場合は、IEEE754 の無効演算例外を検出します。

無効演算例外の発生が許可されていない場合は、例外を発生せず、無効演算としてと FPSR レジスタの保存ビット (ビット 4) がセットされます。ソースの違いにより返す値は次のように異なります。

- ・ソースが正数または+            :  $2^{63}-1$  を返します。
- ・ソースが負数、非数または-    :  $-2^{63}$  を返します。

[ 浮動小数点演算例外 ] 無効演算例外 (V)  
不正確演算例外 (I)

[ 演算結果 ]             FPSR.FS = 1 の場合

reg2(A)	Normal	-Normal	+0	-0	+	-	Q-NaN	S-NaN
演算結果 [例外]	A ( Integer )		0 ( Integer )		Max Int[V]	-Max Int[V]		

- 備考 1. [ ] は必ず発生する例外です。
2. ディノーマル数はフラッシュされ “0” として扱われます。

## &lt; 浮動小数点演算命令 &gt;

<p>Floating-point Truncate to Unsigned Integer Format, rounded to zero (Double)</p> <p><b>TRNCF.DUL</b></p> <p style="text-align: right;">符号なし整数形式への変換 (倍精度)</p>
--

[ 命令形式 ]            TRNCF.DUL reg2, reg3

[ オペレーション ]        reg3 ← trunc reg2(double    Unsigned long-word)

[ フォーマット ]         Format F: I

[ オペコード ]

15	11	10	5	4	0	31	27	26	25	23	22	21	20	17	16																
r	r	r	r	0	1	1	1	1	1	1	1	0	0	0	1	w	w	w	w	0	1	0	0	0	1	0	1	0	1	0	0
reg2										reg3					category	type	sub-op														

[ 説 明 ]                汎用レジスタ reg2 で指定されるレジスタ・ペアにある倍精度浮動小数点形式の内容を算術的に符号のない 64 ビットの整数形式に変換し、汎用レジスタ reg3 で指定されるレジスタ・ペアに格納します。

現在の丸めモードに関係なく、結果を 0 の方向へ丸めます。

ソース・オペランドが無制限大か非数か負数の場合、または丸めた結果が  $2^{64}-1 \sim 0$  の範囲外の場合は、IEEE754 の無効演算例外を検出します。

無効演算例外の発生が許可されていない場合は、例外を発生せず、無効演算としてと FPSR レジスタの保存ビット (ビット 4) がセットされます。ソースの値により返す値は次のように異なります。

- ・ソースが  $2^{64}-1 \sim 0$  の範囲外の正数または+                    :  $2^{64}-1$  を返します。
- ・ソースが負数、非数または-                                         : 0 を返します。

[ 浮動小数点演算例外 ] 無効演算例外 (V)  
不正確演算例外 (I)

[ 演算結果 ]             FPSR.FS = 1 の場合

reg2(A)	+Normal	-Normal	+0	-0	+	-	Q-NaN	S-NaN
演算結果 [例外]	A (Integer)	0 [V]	0 (Integer)		Max U-Int [V]	0 [V]		

備考 1. [ ] は必ず発生する例外です。

2. ディノーマル数はフラッシュされ "0" として扱われます。

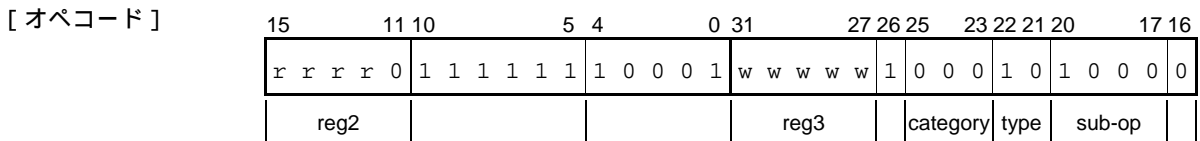
< 浮動小数点演算命令 >

<div style="display: flex; justify-content: space-between;"> <div style="font-size: 24pt; font-weight: bold;">TRNCF.DUW</div> <div style="font-size: 10pt;">Floating-point Truncate to Unsigned Integer Format, rounded to zero (Double)</div> </div> <div style="text-align: right; font-size: 10pt; margin-top: 10px;">符号なし整数形式への変換 (倍精度)</div>
---

[ 命令形式 ]            TRNCF.DUW    reg2, reg3

[ オペレーション ]        reg3 ← trunc reg2(double    Unsigned word)

[ フォーマット ]         Format F: I



[ 説 明 ]                汎用レジスタ reg2 で指定されるレジスタ・ペアにある倍精度浮動小数点形式の内容を算術的に符号のない 32 ビットの整数形式に変換し、汎用レジスタ reg3 に格納します。

現在の丸めモードに関係なく、結果を 0 の方向へ丸めます。

ソース・オペランドが無限大か非数か負数の場合、または丸めた結果が  $2^{32}-1 \sim 0$  の範囲外の場合は、IEEE754 の無効演算例外を検出します。

無効演算例外の発生が許可されていない場合は、例外を発生せず、無効演算としてと FPSR レジスタの保存ビット (ビット 4) がセットされます。ソースの値により返す値は次のように異なります。

- ・ソースが  $2^{32}-1 \sim 0$  の範囲外の正数または+                                :  $2^{32}-1$  を返します。
- ・ソースが負数、非数または-    : 0 を返します。

[ 浮動小数点演算例外 ] 無効演算例外 (V)  
不正確演算例外 (I)

[ 演算結果 ]             FPSR.FS = 1 の場合

reg2(A)	+Normal	-Normal	+0	-0	+	-	Q-NaN	S-NaN
演算結果 [例外]	A ( Integer )	0 [V]	0 ( Integer )		Max U-Int [V]	0 [V]		

- 備考 1. [ ] は必ず発生する例外です。
2. ディノーマル数はフラッシュされ “ 0 ” として扱われます。



< 浮動小数点演算命令 >

<p style="font-size: 24pt; margin: 0;">TRNCF.DW</p>	<p style="font-size: 10pt; margin: 0;">Floating-point Truncate to Integer Format, rounded to zero (Double)</p> <p style="font-size: 10pt; margin: 0;">整数形式への変換 (倍精度)</p>
---	--

[ 命令形式 ]            TRNCF.DW    reg2, reg3

[ オペレーション ]        reg3 ← trunc reg2(double    word)

[ フォーマット ]         Format F: I

[ オペコード ]

15	11	10	5	4	0	31	27	26	25	23	22	21	20	17	16																
r	r	r	r	0	1	1	1	1	1	1	0	0	0	0	1	w	w	w	w	w	1	0	0	0	1	0	1	0	0	0	0
reg2										reg3					category	type	sub-op														

[ 説 明 ]                汎用レジスタ reg2 で指定されるレジスタ・ペアにある倍精度浮動小数点形式の内容を算術的に 32 ビットの整数形式に変換し、汎用レジスタ reg3 に格納します。

現在の丸めモードに関係なく、結果を 0 の方向へ丸めます。

ソース・オペランドが無大か非数の場合、または丸めた結果が  $2^{31} - 1 \sim -2^{31}$  の範囲外の場合は、IEEE754 の無効演算例外を検出します。

無効演算例外の発生が許可されていない場合は、例外を発生せず、無効演算としてと FPSR レジスタの保存ビット (ビット 4) がセットされます。ソースの違いにより返す値は次のように異なります。

- ・ソースが正数または+            :  $2^{31} - 1$  を返します。
- ・ソースが負数、非数または-    :  $-2^{31}$  を返します。

[ 浮動小数点演算例外 ] 無効演算例外 (V)  
不正確演算例外 (I)

[ 演算結果 ]             FPSR.FS = 1 の場合

reg2(A)	Normal	-Normal	+0	-0	+	-	Q-NaN	S-NaN
演算結果 [例外]	A ( Integer )		0 ( Integer )		Max Int[V]	-Max Int[V]		

- 備考 1. [ ] は必ず発生する例外です。
2. ディノーマル数はフラッシュされ " 0 " として扱われます。

## &lt; 浮動小数点演算命令 &gt;

<b>TRNCF.SL</b>	Floating-point Truncate to Integer Format, rounded to zero (Single)  整数形式への変換 (単精度)
-----------------	---

[ 命令形式 ]            TRNCF.SL reg2, reg3

[ オペレーション ]    reg3 ← trunc reg2(single long-word)

[ フォーマット ]        Format F: I

[ オペコード ]

15	11	10	5	4	0	31	27	26	25	23	22	21	20	17	16												
r	r	r	r	r	1	1	1	1	1	1	0	0	0	0	1	w	w	w	w	0	1	0	0	0	1	0	0
reg2					reg3					category	type	sub-op															

[ 説 明 ]                汎用レジスタ reg2 にある単精度浮動小数点形式の内容を算術的に 64 ビットの整数形式に変換し、汎用レジスタ reg3 で指定されるレジスタ・ペアに格納します。

現在の丸めモードに関係なく、結果を 0 の方向へ丸めます。

ソース・オペランドが無限大か非数の場合、または丸めた結果が  $2^{63}-1 \sim -2^{63}$  の範囲外の場合は、IEEE754 の無効演算例外を検出します。

無効演算例外の発生が許可されていない場合は、例外を発生せず、無効演算としてと FPSR レジスタの保存ビット (ビット 4) がセットされます。ソースの違いにより返す値は次のように異なります。

- ・ソースが正数または+            :  $2^{63}-1$  を返します。
- ・ソースが負数、非数または-    :  $-2^{63}$  を返します。

[ 浮動小数点演算例外 ] 無効演算例外 (V)  
不正確演算例外 (I)

[ 演算結果 ]            FPSR.FS = 1 の場合

reg2(A)	Normal	-Normal	+0	-0	+	-	Q-NaN	S-NaN
演算結果 [例外]	A ( Integer )		0 ( Integer )		Max Int[V]		-Max Int[V]	

備考 1. [ ] は必ず発生する例外です。

2. ディノーマル数はフラッシュされ " 0 " として扱われます。

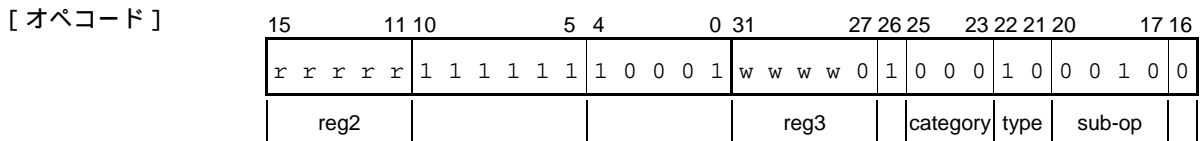
< 浮動小数点演算命令 >

<div style="display: flex; justify-content: space-between;"> <div style="flex: 1;"> <h2 style="margin: 0;">TRNCF.SUL</h2> </div> <div style="flex: 2; text-align: right;"> <p style="font-size: small; margin: 0;">Floating-point Truncate to Unsigned Integer Format, rounded to zero (Single)</p> <p style="margin: 0;">符号なし整数形式への変換 (単精度)</p> </div> </div>
--

[ 命令形式 ]            TRNCF.SUL reg2, reg3

[ オペレーション ]        reg3 ← trunc reg2(Single    Unsigned long-word)

[ フォーマット ]         Format F: I



[ 説 明 ]                汎用レジスタ reg2 にある単精度浮動小数点形式の内容を算術的に符号のない 64 ビットの整数形式に変換し、汎用レジスタ reg3 で指定されるレジスタ・ペアに格納します。

現在の丸めモードに関係なく、結果を 0 の方向へ丸めます。

ソース・オペランドが無限大か非数か負数の場合、または丸めた結果が  $2^{64}-1 \sim 0$  の範囲外の場合は、IEEE754 の無効演算例外を検出します。

無効演算例外の発生が許可されていない場合は、例外を発生せず、無効演算としてと FPSR レジスタの保存ビット (ビット 4) がセットされます。ソースの値により返す値は次のように異なります。

- ・ソースが  $2^{64}-1 \sim 0$  の範囲外の正数または+                                :  $2^{64}-1$  を返します。
- ・ソースが負数、非数または-    : 0 を返します。

[ 浮動小数点演算例外 ] 無効演算例外 (V)  
不正確演算例外 (I)

[ 演算結果 ]             FPSR.FS = 1 の場合

reg2(A)	+Normal	-Normal	+0	-0	+	-	Q-NaN	S-NaN
演算結果 [例外]	A( Integer )	0 [V]	0 ( Integer )		Max U-Int [V]	0 [V]		

- 備考 1. [ ] は必ず発生する例外です。
2. ディノーマル数はフラッシュされ “0” として扱われます。

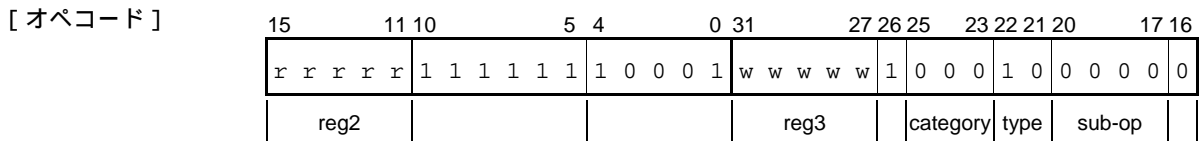
< 浮動小数点演算命令 >

<p style="font-size: 24pt; margin: 0;">TRNCF.SUW</p>	<p style="font-size: 10pt; margin: 0;">Floating-point Truncate to Unsigned Integer Format, rounded to zero (Single)</p> <p style="font-size: 10pt; margin: 0;">符号なし整数形式への変換 (単精度)</p>
--	---

[ 命令形式 ]            TRNCF.SUW    reg2, reg3

[ オペレーション ]        reg3 ← trunc reg2(Single    Unsigned word)

[ フォーマット ]         Format F: I



[ 説 明 ]                汎用レジスタ reg2 にある単精度浮動小数点数形式の内容を算術的に符号のない 32 ビットの整数形式に変換し、汎用レジスタ reg3 に格納します。

現在の丸めモードに関係なく、結果を 0 の方向へ丸めます。

ソース・オペランドが無限大か非数か負数の場合、または丸めた結果が  $2^{32}-1 \sim 0$  の範囲外の場合は、IEEE754 の無効演算例外を検出します。

無効演算例外の発生が許可されていない場合は、例外を発生せず、無効演算としてと FPSR レジスタの保存ビット (ビット 4) がセットされます。ソースの値により返す値は次のように異なります。

- ・ソースが  $2^{32}-1 \sim 0$  の範囲外の正数または+                                :  $2^{32}-1$  を返します。
- ・ソースが負数、非数または-    : 0 を返します。

[ 浮動小数点演算例外 ] 無効演算例外 (V)  
不正確演算例外 (I)

[ 演算結果 ]             FPSR.FS = 1 の場合

reg2(A)	+Normal	-Normal	+0	-0	+	-	Q-NaN	S-NaN
演算結果 [例外]	A( Integer )	0 [V]	0 ( Integer )		Max U-Int [V]	0 [V]		

- 備考 1. [ ] は必ず発生する例外です。
2. ディノーマル数はフラッシュされ “ 0 ” として扱われます。

< 浮動小数点演算命令 >

<p>TRNCF.SW</p>	<p>Floating-point Truncate to Integer Format, rounded to zero (Single)</p> <p style="text-align: center;">整数形式への変換 (単精度)</p>
-----------------	--

[ 命令形式 ]            TRNCF.SW reg2, reg3

[ オペレーション ]        reg3 ← trunc reg2(single word)

[ フォーマット ]         Format F: I

[ オペコード ]

15	11	10	5	4	0	31	27	26	25	23	22	21	20	17	16																
r	r	r	r	r	1	1	1	1	1	1	0	0	0	0	1	w	w	w	w	w	1	0	0	0	1	0	0	0	0	0	0
reg2										reg3					category	type	sub-op														

[ 説 明 ]                汎用レジスタ reg2 にある単精度浮動小数点数形式の内容を算術的に 32 ビットの整数形式に変換し、汎用レジスタ reg3 に格納します。

現在の丸めモードに関係なく、結果を 0 の方向へ丸めます。

ソース・オペランドが無大か非数の場合、または丸めた結果が  $2^{31} - 1 \sim -2^{31}$  の範囲外の場合は、IEEE754 の無効演算例外を検出します。

無効演算例外の発生が許可されていない場合は、例外を発生せず、無効演算としてと FPSR レジスタの保存ビット (ビット 4) がセットされます。ソースの違いにより返す値は次のように異なります。

- ・ソースが正数または+            :  $2^{31} - 1$  を返します。
- ・ソースが負数、非数または-    :  $-2^{31}$  を返します。

[ 浮動小数点演算例外 ] 無効演算例外 (V)  
不正確演算例外 (I)

[ 演算結果 ]             FPSR.FS = 1 の場合

reg2(A)	Normal	-Normal	+0	-0	+	-	Q-NaN	S-NaN
演算結果 [例外]	A ( Integer )		0 ( Integer )		Max Int[V]	-Max Int[V]		

- 備考 1. [ ] は必ず発生する例外です。
2. ディノーマル数はフラッシュされ “ 0 ” として扱われます。

## 第5章 浮動小数点演算例外

この章では、FPUが浮動小数点演算例外をどのように処理するかを説明します。

### 5.1 例外の種類

通常の方法で浮動小数点演算または演算結果を処理できなくなると、浮動小数点演算例外が発生します。浮動小数点演算例外発生時の動作には、次の2とおりがあります。

- ・ 例外許可の場合

浮動小数点設定 / 状態レジスタFPSRの原因ビットをセットし、例外ハンドラ・ルーチンに処理（ソフトウェア処理）を移行します。

- ・ 例外禁止の場合

浮動小数点設定 / 状態レジスタFPSRの保存ビットをセットし、FPUのデスティネーション・レジスタに適切な値（デフォルト値）を格納し、実行を継続します。

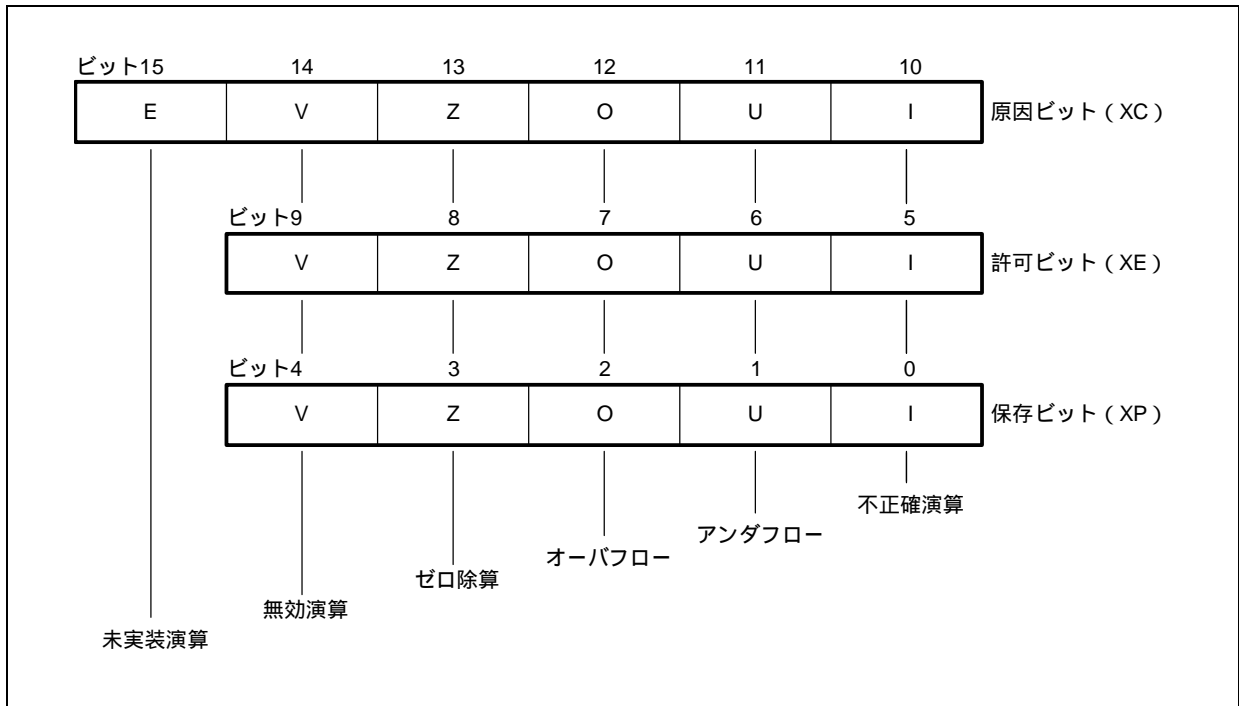
FPUは、次の5種類のIEEE754例外を、原因ビット、許可ビット、保存ビット（ステータス・フラグ）によってサポートします。

- ・ 不正確演算（I）
- ・ オーバフロー（O）
- ・ アンダフロー（U）
- ・ ゼロ除算（Z）
- ・ 無効演算（V）

また6番目の例外原因として、未実装演算（E）があり、浮動小数点演算を実行できないときに発生します。この例外は、ソフトウェアによる処理を必要とします。未実装演算例外（E）はその性質上、許可ビット、保存ビットはなく、常に例外が許可され、発生します。

図5 - 1に、例外をサポートするために使用するFPSRレジスタのビットを示します。

図5-1 FPSRレジスタの原因/許可/保存ビット



IEEE754の5つの例外 (V, Z, O, U, I) は、許可ビットをセットすることにより許可されます。例外が発生し、対応する許可ビットがセットされていれば、FPUは対応する原因ビットをセットし、例外が受け付け可能な状態であれば、例外ハンドラ・ルーチンに移行します。例外の発生が禁止されている場合、その例外に対応する保存ビットがセットされ例外ハンドラ・ルーチンには移行しません。

## 5.2 例外処理

浮動小数点演算例外が発生すると、FPSRレジスタの原因ビットは、浮動小数点演算例外の発生した原因を示します。

### 5.2.1 ステータス・フラグ

各IEEE754例外に対して、対応する保存ビットが用意されています。保存ビットは、対応する例外の発生が禁止されていて、かつ例外の条件が検出されたときにセットされます。保存ビットは、LDSR命令でFPSRレジスタに新しい値を書き込むことによってセット/リセットできます。

許可ビットによって例外が禁止されている場合、FPUにより既定の処理が行われます。この処理では、浮動小数点演算の結果の代わりに、デフォルト値を結果として与えます。このデフォルト値は例外の種類により決まっています。オーバフロー例外とアンダフロー例外の場合には、そのときの丸めモードにより異なります。表5-1に、それぞれのFPUのIEEE754例外によって与えられるデフォルト値を示します。

表5 - 1 FPUのIEEE754例外のデフォルト値

領域	説明	丸めモード	デフォルト値
V	無効演算	-	Quiet Not a Number (Q-NaN) を使用します。
Z	ゼロ除算	-	正しい符号付き を使用します。
O	オーバーフロー	RN	中間結果の符号を付けた
U	アンダフロー	RN	中間結果の符号を付けた0
I	不正確演算	-	丸められた結果を使用します。

## 5.3 例外の詳細

次に、FPUの各例外の発生条件とFPUでの対応について説明します。

### 5.3.1 不正確演算例外 (I)

次のような場合、FPUは不正確演算例外を検出します。

- ・丸めた結果が精度落ちした場合
- ・丸めた結果がオーバーフローし、かつ、オーバーフロー例外が禁止状態の場合
- ・丸めた結果がアンダフローし、かつ、アンダフロー例外が禁止状態の場合
- ・オペランドのディノーマル数がフラッシュされた場合で、無効演算例外 (V)、ゼロ除算例外 (Z) が検出されず、かつ他のオペランドがQ-NaNでない場合。

#### (1) 例外が許可されている場合

デスティネーション・レジスタの内容は変更せず、ソース・レジスタの内容は保存し、不正確演算例外が発生します。

#### (2) 例外が許可されていない場合

ほかの例外が発生しない場合、丸められた結果がアンダフロー / オーバフローした結果を、デスティネーション・レジスタに格納します。



### 5.3.2 無効演算例外 (V)

オペランドの片方または両方が無効の場合、無効演算例外を検出します。

- ・加減算・積和<sup>※</sup>：無限大同士の加減算（+ ） + （ - ）または（ - ） - （ - ）。
- ・乗算・積和： $\pm 0 \times \pm$  。
- ・除算： $\pm 0 \div \pm 0$ または $\pm \div \pm$  。
- ・比較：条件コード8-15で、オペランドがUnorderdの場合（表4-2 条件コードのビット定義と論理反転参照）
- ・オペランドにS-NaNを含む算術演算。条件付き転送命令（cmov）は算術演算として扱いませんが、絶対値（ABS）、算術否定（NEG）、最小値（MIN）、最大値（MAX）は算術演算として扱います。
- ・オペランドがS-NaNの場合の比較と浮動小数点への変換。
- ・ソースが整数範囲外の場合の整数への変換。
- ・平方根：オペランドが0より小さい。

**注** 乗算の結果が無限大に丸められ、無限大同士の加減算（+ ） + （ - ）または（ - ） - （ - ）となった場合。

#### (1) 例外が許可されている場合

デスティネーション・レジスタの内容は変更せず、ソース・レジスタの内容は保持し、無効演算例外が発生します。

#### (2) 例外が許可されていない場合

ほかの例外が発生しない場合、デスティネーションが浮動小数点形式であれば、Q-NaNがデスティネーション・レジスタに格納されます。デスティネーションが整数形式の場合に、デスティネーション・レジスタに格納される値については、各命令の演算結果の説明を参照してください。

### 5.3.3 ゼロ除算例外 (Z)

除数が0で被除数が0以外の有限数のとき、ゼロ除算例外を検出します。

#### (1) 例外が許可されている場合

デスティネーション・レジスタの内容は変更せず、ソース・レジスタの内容は保存し、ゼロ除算例外が発生します。

#### (2) 例外が許可されていない場合

ほかの例外が発生しない場合、正しい符号のついた無限大数（ $\pm$  ）がデスティネーション・レジスタに格納されます。

### 5.3.4 オーバフロー例外 (O)

オーバフロー例外は、指数範囲が無限のとき、丸められた浮動小数点の結果の大きさがデスティネーション形式の最大有限数よりも大きい場合に検出します。

#### (1) 例外が許可されている場合

デスティネーション・レジスタの内容は変更せず、ソース・レジスタの内容は保存し、オーバフロー例外が発生します。

#### (2) 例外が許可されていない場合

ほかの例外が発生しない場合、丸めモードと中間結果の符号によって決まるデフォルト値がデスティネーション・レジスタに格納されます(表5-1 FPUのIEEE754例外のデフォルト値参照)。

### 5.3.5 アンダフロー例外 (U)

次の2つの場合、アンダフロー例外を検出します。

- ・ 演算結果が  $-2^{E_{min}} \sim +2^{E_{min}}$  (ただし0以外) の場合
- ・ 正規化されていない小さな数同士の演算によって精度落ちが発生したとき

IEEE754では、アンダフローを検出する方法が多数用意されていますが、どの処理の場合にも、同じ方法で検出するように規定しています。

アンダフローを検出する方法として、次の2つがあります。

- ・ 丸め後、指数範囲を無限として計算された結果が、0以外で  $\pm 2^{E_{min}}$  内のとき
- ・ 丸め前、指数範囲と精度を無限として計算された結果が、0以外で  $\pm 2^{E_{min}}$  内のとき

本FPUでは、丸めの前にアンダフローを検出します。

精度落ちを検出する方法として、次の2つがあります。

- ・ ディノーマライズ・ロス(与えられた結果と、指数範囲が無限のとき計算された結果が異なる場合)
- ・ 不正確結果(与えられた結果と、指数範囲と精度が無限のとき計算された結果が異なる場合)

本FPUでは、精度落ちを不正確結果として検出します。

#### (1) 例外が許可されている場合

FPSR.FSビットがセットされていて、アンダフロー例外が許可されている場合、アンダフロー例外(U)が発生します。FPSR.FSビットがセットされていて、アンダフロー例外が許可されていない場合、不正確演算例外が許可されていれば、不正確演算例外(I)が発生します

**(2) 例外が許可されていない場合**

FPSR.FSビットがセットされている場合、丸めモードと中間結果の符号で決まるデフォルト値がデスティネーション・レジスタに格納されます（表5-1 FPUのIEEE754例外のデフォルト値参照）。

**5.3.6 未実装演算例外（E）**

将来拡張するために予約されているオペレーション・コードを実行しようとする、Eビットをセットし、未実装演算例外（E）が発生します。オペランドとデスティネーション・レジスタの内容は変更しません。通常、実装されていない命令はソフトウェアによってエミュレートされます。エミュレートされたオペレーションからIEEE754例外が発生した場合、今度はそれらの例外をシミュレートしてください。

FPSR.FSビットがセットされている場合、定義されているFPU命令においては、未実装演算例外（E）が発生することはありません。

## 5.4 プレサイス例外とインプレサイス例外

浮動小数点の例外において、プレサイスに例外を発生させるか、インプレサイスに例外を発生させるかを指定することができます。

デフォルトでは、単精度命令、倍精度命令ともにインプレサイスに例外を発生します。プレサイスな例外発生は、FPSR.SEMビットおよびDEMビットをセットすることにより指定します。SEMビットは単精度命令の例外モード、DEMビットは倍精度命令の例外モードを指定します。単精度命令と倍精度命令の分類については、**4.2 浮動小数点演算命令の概要**を参照してください。

### 5.4.1 プレサイス例外

プレサイス例外が指定されると、CPUは実行を開始した浮動小数点演算命令が完了するまで後続のすべての命令の実行を開始しません。したがって、例外が発生した場合に、ソフトウェアによるエミュレーションのあと、プログラムを続行することが可能です。

浮動小数点演算例外が発生した命令のプログラム・カウンタがEIPCレジスタおよびFPEPCレジスタに格納されます。エミュレーション処理からの復帰は、EIRET命令により行われます。プレサイス例外モードで発生した浮動小数点演算例外は、PSWのIDビットやNPビットの状態にかかわらず、ただちに受け付けられます。

### 5.4.2 インプレサイス例外

インプレサイス例外が指定されると、CPUは実行を開始した浮動小数点演算命令が完了する前に、後続の命令の実行を開始します。このため例外発生時には、後続の命令が投機的に実行されているので、例外が発生した場合、エミュレーションは困難となりますが、命令実行のスループットを大幅に引き上げることが可能になります。

インプレサイス例外で実行された浮動小数点演算命令が浮動小数点演算例外を発生すると、例外が受け付けられ例外ハンドラ・ルーチンへ移行するまでの間、後続の浮動小数点演算命令（TRFSR命令を除く）の結果は汎用レジスタに反映されず、また浮動小数点演算例外も発生しません。これを命令の無効化と呼びます。

後続命令を実行する前にインプレサイス浮動小数点演算例外を受け付けたい場合、SYNCE命令により例外を発生した命令の完了を待ち合わせることができます。

浮動小数点演算例外が発生した命令のプログラム・カウンタがFPEPCレジスタに、例外が受け付けられ中断された命令のプログラム・カウンタがEIPCレジスタに格納されます。

インプレサイス例外モードで発生した浮動小数点演算例外は、PSWのIDビットが1、またはNPビットが1のときに保留されます。この場合、LDSR命令を使用してPSW.NPビットとIDビットを0にすると、保留していた例外が受け付けられます。

## 5.5 状態の退避と復帰

浮動小数点演算例外が発生すると、PCおよびPSWはEIPC、EIPSWレジスタへ退避され、例外要因コードがEIICレジスタに格納されます。

浮動小数点演算例外の例外要因コードは、プレサイス例外では0x00000071、インプレサイス例外では0x00000072です。互換性のために、ECRレジスタの下位16ビットにも例外要因コードの下位16ビットが格納されます。

浮動小数点演算例外の処理中にEIレベル例外が受け付けられると、EIPCなどがオーバーライドされて浮動小数点演算例外発生命令へ復帰できなくなるため、EIレベル例外の受け付けを許可する必要がある場合は、必ず、先にEIPC、EIPSW、ECR、EIICレジスタの内容をスタックなどに退避させてください。

浮動小数点演算例外ハンドラ・ルーチン中で浮動小数点演算命令を使用する場合、さらに浮動小数点演算例外が発生することでFPSR、FPEPCレジスタがオーバーライドされることがあります。このような場合には、浮動小数点演算例外ハンドラの先頭でFPSR、FPEPCレジスタを退避させ、ハンドラの最後に復帰させてください。

浮動小数点演算例外ハンドラにおいては、FPSR.PRビットを調べることによって、プレサイス例外、インプレサイス例外のいずれであったかを判定できます。

FPSRレジスタの原因ビット、およびPRビットは許可された例外1回分の結果だけを保持します。いずれも次に許可された例外が発生するまで前の結果が保持されます。

FPSR、FPEPCレジスタをアクセスする場合、BSELレジスタでFPU機能システム・レジスタ・バンクを選択しておく必要があることに注意してください。例外処理中に例外が発生し、その例外処理においてBSELレジスタが書き換えられることがあります。このため、例外処理においては、BSELレジスタを、プログラム・コンテキストとして待避、復帰することを推奨します。

浮動小数点演算例外ハンドラのコード例を次に示します。

```

000  .offset  0x70          -- FPU exception
001  jr  _fpu_exception_handler
002  ...
003
004  _fpu_exception_handler:
005  -- 基本コンテキスト退避
006  addi    -100,sp,sp
007  st.w    r31,96[sp]
008  st.w    r1,92[sp]
009  stsr    31,r1          -- BSEL退避
010  st.w    r1,8[sp]
011  ldsr    zero,31       -- CPUバンク選択
012  stsr    13,r1        -- EIIC退避
013  st.w    r1,28[sp]
014  stsr    1,r1         -- EIPSW退避
015  st.w    r1,24[sp]
016  stsr    0,r1        -- EIPC退避
017  st.w    r1,20[sp]
018  ei                      -- EIレベル・マスカブル例外許可
019  stsr    17,r1       -- CTPSW退避
020  st.w    r1,16[sp]
021  stsr    16,r1       -- CTPC退避
022  st.w    r1,12[sp]
023  movea  0x2000,r0,r1 -- FPUステータス・バンク選択
024  ldsr    r1,31

```

```

025  stsr      6,r1          -- FPSR退避
026  st.w     r1,4[sp]
027  stsr     7,r1          -- FPEPC退避
028  st.w     r1,0[sp]
029
030  -- プレサイス例外/インプレサイス例外の判定
031  ld.w     28[sp],r1
032  addi    -0x72,r1,r0    -- EIICの例外要因コードが0x72か?
033  be      _imprecise_fpe
034
035  -- 浮動小数点演算例外処理本体
036
037  -- 基本コンテキスト回復
038  movea   0x2000,r0,r1   -- FPUステータス・バンク選択
039  ldsr    r1,31
040  ld.w    0[sp],r31
041  ld.w    4[sp],r1
042  ldsr    r31,7          -- FPEPC回復
043  ldsr    r1,6          -- FPSR回復
044  ldsr    zero,31       -- CPUバンク選択
045  ld.w    12[sp],r31
046  ld.w    16[sp],r1
047  ldsr    r31,16       -- CTPC回復
048  ldsr    r1,17       -- CTPSW回復
049  di
049      -- EIレベル・マスカブル例外禁止
050  ld.w    24[sp],r31
051  ld.w    28[sp],r1
052  ldsr    r31,1        -- EIPSW回復
053  ldsr    r1,1         -- EIIC回復
054  -- 復帰アドレスの設定 (プレサイス例外の場合)
055  ld.w    20[sp],r1     -- 退避していたEIPCを取り出す
056  add     4,r1          -- r1に4を加算 (FPU命令はすべて4バイト)
057  ldsr    r1,0          -- r1 (次の命令のPC) をEIPCに設定
058  ld.w    8[sp],r1     -- BSEL回復
059  ldsr    r1,31
060  ld.w    92[sp],r1
061  ld.w    96[sp],r31
062  addi   100,sp,sp
063  eiret
064
065  _imprecise_fpe:
066  -- 基本コンテキスト回復
067      ... 省略 ...
068
069  -- 復帰ポイントのアドレスをEIPCへ格納
070  ld.w    8[sp],r1     -- BSEL回復
071  ldsr    r1,31
072  ld.w    92[sp],r1
073  addi   100,sp,sp
074  eiret

```

## 5.6 浮動小数点演算モデルの選択

### 5.6.1 正確な演算を必要とする場合

FPUでは、将来拡張するために予約されているオペレーション・コードや無効な形式コードの命令を実行しようとする、Eビットをセットし、未実装演算例外（E）を発生します。オペランドとデスティネーション・レジスタの内容は変更しません。

厳密に正確な演算を必要とする場合、例外が発生した命令はソフトウェアによってエミュレートされます。エミュレートされたオペレーションからIEEE754例外が発生した場合、今度はそれらの例外をエミュレートしてください。エミュレーションのあと、プログラムを続行する必要がある場合は、FPSR.SEM, DEMビットをセットして、あらかじめプレサイス例外を指定してください。

IEEE754は、5つの標準例外のどれに対しても、代わりに計算した結果をデスティネーション・レジスタに格納できるような例外ハンドラを推奨しています。

FPEPCレジスタを使って命令を検索することによって、例外ハンドラは次のことを判別できます。

- ・実行中の命令
- ・デスティネーションの形式

オーバフロー例外、アンダフロー例外（変換命令を除く）、不正確演算例外が発生したときに正しく丸められた結果を得るには、ソース・レジスタを調べたり、命令をエミュレートするソフトウェアを例外ハンドラ中に用意したりしてください。

無効演算例外やゼロ除算例外が発生した場合や、オーバフロー例外またはアンダフロー例外が浮動小数点変換時に発生した場合、例外ハンドラ中に命令のソース・レジスタを調べることによってオペランドの値を得るようなソフトウェアを用意してください。

IEEE754においては、可能なら、オーバフロー例外およびアンダフロー例外を不正確演算例外に対して優先させることを推奨しています。この優先順位は、ソフトウェアによって設定します。ハードウェアは、オーバフロー例外、アンダフロー例外と不正確演算例外の両方のビットをセットします。

### 5.6.2 演算性能を優先する場合

FPUでは、浮動小数点演算実行による例外をできるだけ起こさないで、処理性能を優先させる演算モデルを提供します。リアルタイム性を必要とする用途において、厳密に正確な演算が必須でない場合には、例外発生によるエミュレーション処理のオーバーヘッドを排除することができます。

処理速度を優先する場合には、次の設定を推奨します。

- ・ FPSRレジスタの許可ビットをクリア(0)して例外発生を禁止
- ・ 精度を必要としない処理では単精度浮動小数点形式を使用
- ・ FPSR.SEMビットおよびDEMビットでインプレサイズ例外モードを指定

演算処理上、無視できる浮動小数点演算例外は例外発生を禁止することにより、デフォルトの値で演算を続行できます。単精度命令は、一般に、実行クロック数(latency)も少なくてすみます。

FPUでは、FPSR.FSビットをセット(1)して、ディノーマル数のフラッシュを許可しているとき、未実装演算例外(E)は発生しません。フラッシュを許可することにより、演算結果がアンダフローした場合でも、結果をディノーマルとして保持せずフラッシュして格納します。



## 付録A 命令一覧

### A.1 基本命令

アルファベット順の基本命令機能一覧を表A - 1に示します。

表A - 1 基本命令機能一覧（アルファベット順）（1/4）

ニモニック	オペランド	フォー マツト	フラグ					命令機能
			CY	OV	S	Z	SAT	
ADD	reg1, reg2	I	0/1	0/1	0/1	0/1	-	加算
ADD	imm5, reg2	II	0/1	0/1	0/1	0/1	-	加算
ADDI	imm16, reg1, reg2	VI	0/1	0/1	0/1	0/1	-	加算
ADF	cccc, reg1, reg2, reg3	XI	0/1	0/1	0/1	0/1	-	条件付き加算
AND	reg1, reg2	I	-	0	0/1	0/1	-	論理積
ANDI	imm16, reg1, reg2	VI	-	0	0/1	0/1	-	論理積
Bcond	disp9	III	-	-	-	-	-	条件分岐
BSH	reg2, reg3	XII	0/1	0	0/1	0/1	-	ハーフワード・データのバイト・スワップ
BSW	reg2, reg3	XII	0/1	0	0/1	0/1	-	ワード・データのバイト・スワップ
CALLT	imm6	II	-	-	-	-	-	テーブル参照によるサブルーチン・コール
CAXI	[reg1], reg2, reg3	IX	0/1	0/1	0/1	0/1	-	比較と交換
CLR1	bit#3, disp16 [reg1]	VIII	-	-	-	0/1	-	ビット・クリア
CLR1	reg2, [reg1]	IX	-	-	-	0/1	-	ビット・クリア
CMOV	cccc, reg1, reg2, reg3	XI	-	-	-	-	-	条件付き転送
CMOV	cccc, imm5, reg2, reg3	XII	-	-	-	-	-	条件付き転送
CMP	reg1, reg2	I	0/1	0/1	0/1	0/1	-	比較
CMP	imm5, reg2	II	0/1	0/1	0/1	0/1	-	比較
CTRET	(なし)	X	0/1	0/1	0/1	0/1	0/1	サブルーチン・コールからの復帰
DI	(なし)	X	-	-	-	-	-	EIレベル・マスク例外の禁止
DISPOSE	imm5, list12	XIII	-	-	-	-	-	スタック・フレームの削除
DISPOSE	imm5, list12, [reg1]	XIII	-	-	-	-	-	スタック・フレームの削除
DIV	reg1, reg2, reg3	XI	-	0/1	0/1	0/1	-	(符号付き)ワード・データの除算
DIVH	reg1, reg2	I	-	0/1	0/1	0/1	-	(符号付き)ハーフワード・データの除算
DIVH	reg1, reg2, reg3	XI	-	0/1	0/1	0/1	-	(符号付き)ハーフワード・データの除算
DIVHU	reg1, reg2, reg3	XI	-	0/1	0/1	0/1	-	(符号なし)ハーフワード・データの除算
DIVQ	reg1, reg2, reg3	XI	-	0/1	0/1	0/1	-	(符号付き)ワード・データの除算(可変ステップ)
DIVQU	reg1, reg2, reg3	XI	-	0/1	0/1	0/1	-	(符号なし)ワード・データの除算(可変ステップ)
DIVU	reg1, reg2, reg3	XI	-	0/1	0/1	0/1	-	(符号なし)ワード・データの除算
EI	(なし)	X	-	-	-	-	-	EIレベル・マスク例外の許可
EIRET	(なし)	X	0/1	0/1	0/1	0/1	0/1	EIレベル例外からの復帰
FERET	(なし)	X	0/1	0/1	0/1	0/1	0/1	FEレベル例外からの復帰
FETRAP	vector4	I	-	-	-	-	-	FEレベル・ソフトウェア例外命令

表A - 1 基本命令機能一覧（アルファベット順）（2/4）

二モニック	オペランド	フォー マット	フラグ					命令機能
			CY	OV	S	Z	SAT	
HALT	(なし)	X	-	-	-	-	-	停止
HSH	reg2, reg3	XII	0/1	0	0/1	0/1	-	ハーフワード・データのハーフワード・スワップ
HSW	reg2, reg3	XII	0/1	0	0/1	0/1	-	ワード・データのハーフワード・スワップ
JARL	disp22, reg2	V	-	-	-	-	-	分岐とレジスタ・リンク
JARL	disp32, reg1	VI	-	-	-	-	-	分岐とレジスタ・リンク
JMP	[reg1]	I	-	-	-	-	-	レジスタ間接無条件分岐
JMP	disp32 [reg1]	VI	-	-	-	-	-	レジスタ間接無条件分岐
JR	disp22	V	-	-	-	-	-	無条件分岐（PC相対）
JR	disp32	VI	-	-	-	-	-	無条件分岐（PC相対）
LD.B	disp16 [reg1], reg2	VII	-	-	-	-	-	（符号付き）バイト・データのロード
LD.B	disp23 [reg1], reg3	XIV	-	-	-	-	-	（符号付き）バイト・データのロード
LD.BU	disp16 [reg1], reg2	VII	-	-	-	-	-	（符号なし）バイト・データのロード
LD.BU	disp23 [reg1], reg3	XIV	-	-	-	-	-	（符号なし）バイト・データのロード
LD.H	disp16 [reg1], reg2	VII	-	-	-	-	-	（符号付き）ハーフワード・データのロード
LD.H	disp23 [reg1], reg3	XIV	-	-	-	-	-	（符号付き）ハーフワード・データのロード
LD.HU	disp16 [reg1], reg2	VII	-	-	-	-	-	（符号なし）ハーフワード・ロード
LD.HU	disp23 [reg1], reg3	XIV	-	-	-	-	-	（符号なし）ハーフワード・ロード
LD.W	disp16 [reg1], reg2	VII	-	-	-	-	-	ワード・データのロード
LD.W	disp23 [reg1], reg3	XIV	-	-	-	-	-	ワード・データのロード
LDSR	reg2, regID	IX	-	-	-	-	-	システム・レジスタへのロード
MAC	reg1, reg2, reg3, reg4	XI	-	-	-	-	-	（符号付き）ワード・データの加算付き乗算
MACU	reg1, reg2, reg3, reg4	XI	-	-	-	-	-	（符号なし）ワード・データの加算付き乗算
MOV	reg1, reg2	I	-	-	-	-	-	データの転送
MOV	imm5, reg2	II	-	-	-	-	-	データの転送
MOV	imm32, reg1	VI	-	-	-	-	-	データの転送
MOVEA	imm16, reg1, reg2	VI	-	-	-	-	-	実行アドレスの転送
MOVHI	imm16, reg1, reg2	VI	-	-	-	-	-	上位ハーフワードの転送
MUL	reg1, reg2, reg3	XI	-	-	-	-	-	（符号付き）ワード・データの乗算
MUL	imm9, reg2, reg3	XII	-	-	-	-	-	（符号付き）ワード・データの乗算
MULH	reg1, reg2	I	-	-	-	-	-	（符号付き）ハーフワード・データの乗算
MULH	imm5, reg2	II	-	-	-	-	-	（符号付き）ハーフワード・データの乗算
MULHI	imm16, reg1, reg2	VI	-	-	-	-	-	（符号付き）ハーフワード・イミディエイトの乗算
MULU	reg1, reg2, reg3	XI	-	-	-	-	-	（符号なし）ワード・データの乗算
MULU	imm9, reg2, reg3	XII	-	-	-	-	-	（符号なし）ワード・データの乗算
NOP	(なし)	I	-	-	-	-	-	オペレーションなし
NOT	reg1, reg2	I	-	0	0/1	0/1	-	論理否定（1の補数をとる）
NOT1	bit#3, disp16 [reg1]	VIII	-	-	-	0/1	-	ビット・ノット
NOT1	reg2, [reg1]	IX	-	-	-	0/1	-	ビット・ノット
OR	reg1, reg2	I	-	0	0/1	0/1	-	論理和
ORI	imm16, reg1, reg2	VI	-	0	0/1	0/1	-	論理和

表A - 1 基本命令機能一覧(アルファベット順)(3/4)

ニモニック	オペランド	フォー マット	フラグ					命令機能
			CY	OV	S	Z	SAT	
PREPARE	list12, imm5	XIII	-	-	-	-	-	スタック・フレームの生成
PREPARE	list12, imm5, sp/imm	XIII	-	-	-	-	-	スタック・フレームの生成
RETI	(なし)	X	0/1	0/1	0/1	0/1	0/1	EI レベル・ソフトウェア例外または割り込みからの復帰
RIE	(なし)	I/X	-	-	-	-	-	予約命令例外
SAR	reg1, reg2	IX	0/1	0	0/1	0/1	-	算術右シフト
SAR	imm5, reg2	II	0/1	0	0/1	0/1	-	算術右シフト
SAR	reg1, reg2, reg3	XI	0/1	0	0/1	0/1	-	算術右シフト
SASF	cccc, reg2	IX	-	-	-	-	-	シフトとフラグ条件の設定
SATADD	reg1, reg2	I	0/1	0/1	0/1	0/1	0/1	飽和加算
SATADD	imm5, reg2	II	0/1	0/1	0/1	0/1	0/1	飽和加算
SATADD	reg1, reg2, reg3	XI	0/1	0/1	0/1	0/1	0/1	飽和加算
SATSUB	reg1, reg2	I	0/1	0/1	0/1	0/1	0/1	飽和減算
SATSUB	reg1, reg2, reg3	XI	0/1	0/1	0/1	0/1	0/1	飽和減算
SATSUBI	imm16, reg1, reg2	VI	0/1	0/1	0/1	0/1	0/1	飽和減算
SATSUBR	reg1, reg2	I	0/1	0/1	0/1	0/1	0/1	飽和逆減算
SBF	cccc, reg1, reg2, reg3	XI	0/1	0/1	0/1	0/1	-	条件付き減算
SCH0L	reg2, reg3	IX	0/1	0	0	0/1	-	MSB側からのビット(0)検索
SCH0R	reg2, reg3	IX	0/1	0	0	0/1	-	LSB側からのビット(0)検索
SCH1L	reg2, reg3	IX	0/1	0	0	0/1	-	MSB側からのビット(1)検索
SCH1R	reg2, reg3	IX	0/1	0	0	0/1	-	LSB側からのビット(1)検索
SET1	bit#3, disp16 [reg1]	VIII	-	-	-	0/1	-	ビット・セット
SET1	reg2, [reg1]	IX	-	-	-	0/1	-	ビット・セット
SETF	cccc, reg2	IX	-	-	-	-	-	フラグ条件の設定
SHL	reg1, reg2	IX	0/1	0	0/1	0/1	-	論理左シフト
SHL	imm5, reg2	II	0/1	0	0/1	0/1	-	論理左シフト
SHL	reg1, reg2, reg3	XI	0/1	0	0/1	0/1	-	論理左シフト
SHR	reg1, reg2	IX	0/1	0	0/1	0/1	-	論理右シフト
SHR	imm5, reg2	II	0/1	0	0/1	0/1	-	論理右シフト
SHR	reg1, reg2, reg3	XI	0/1	0	0/1	0/1	-	論理右シフト
SLD.B	disp7 [ep], reg2	IV	-	-	-	-	-	(符号付き)バイト・ロード
SLD.BU	disp4 [ep], reg2	IV	-	-	-	-	-	(符号なし)バイト・データのロード
SLD.H	disp8 [ep], reg2	IV	-	-	-	-	-	(符号付き)ハーフワード・データのロード
SLD.HU	disp5 [ep], reg2	IV	-	-	-	-	-	(符号なし)ハーフワード・データのロード
SLD.W	disp8 [ep], reg2	IV	-	-	-	-	-	ワード・データのロード
SST.B	reg2, disp7 [ep]	IV	-	-	-	-	-	バイト・データのストア
SST.H	reg2, disp8 [ep]	IV	-	-	-	-	-	ハーフワード・データのストア
SST.W	reg2, disp8 [ep]	IV	-	-	-	-	-	ワード・データのストア
ST.B	reg2, disp16 [reg1]	VII	-	-	-	-	-	バイト・データのストア
ST.B	reg3, disp23 [reg1]	XIV	-	-	-	-	-	バイト・データのストア
ST.H	reg2, disp16 [reg1]	VII	-	-	-	-	-	ハーフワード・データのストア
ST.H	reg3, disp23 [reg1]	XIV	-	-	-	-	-	ハーフワード・データのストア

表A - 1 基本命令機能一覧(アルファベット順)(4/4)

ニモニック	オペランド	フォー マット	フラグ					命令機能
			CY	OV	S	Z	SAT	
ST.W	reg2, disp16 [reg1]	VII	-	-	-	-	-	ワード・データのストア
ST.W	reg3, disp23 [reg1]	XIV	-	-	-	-	-	ワード・データのストア
STSR	regID, reg2	IX	-	-	-	-	-	システム・レジスタの内容のストア
SUB	reg1, reg2	I	0/1	0/1	0/1	0/1	-	減算
SUBR	reg1, reg2	I	0/1	0/1	0/1	0/1	-	逆減算
SWITCH	reg1	I	-	-	-	-	-	テーブル参照分岐
SXB	reg1	I	-	-	-	-	-	バイト・データの符号拡張
SXH	reg1	I	-	-	-	-	-	ハーフワード・データの符号拡張
SYNCE	(なし)	I	-	-	-	-	-	例外同期化命令
SYNCM	(なし)	I	-	-	-	-	-	メモリ同期化命令
SYNCP	(なし)	I	-	-	-	-	-	パイプライン同期化命令
SYSCALL	vector8	X	-	-	-	-	-	システム・コール例外
TRAP	vector5	X	-	-	-	-	-	ソフトウェア例外
TST	reg1, reg2	I	-	0	0/1	0/1	-	テスト
TST1	bit#3, disp16 [reg1]	VIII	-	-	-	0/1	-	ビット・テスト
TST1	reg2, [reg1]	IX	-	-	-	0/1	-	ビット・テスト
XOR	reg1, reg2	I	-	0	0/1	0/1	-	排他的論理和
XORI	imm16, reg1, reg2	VI	-	0	0/1	0/1	-	排他的論理和
ZXB	reg1	I	-	-	-	-	-	バイト・データのゼロ拡張
ZXH	reg1	I	-	-	-	-	-	ハーフワード・データのゼロ拡張

## A.2 浮動小数点演算命令

浮動小数点演算命令機能一覧を表A - 2に示します。

表A - 2 浮動小数点演算命令機能一覧 (1/2)

二モニック	オペランド	形式	命令機能
ABSF.D	reg2, reg3	倍精度	浮動小数点絶対値
ABSF.S	reg2, reg3	単精度	浮動小数点絶対値
ADDF.D	reg1, reg2, reg3	倍精度	浮動小数点加算
ADDF.S	reg1, reg2, reg3	単精度	浮動小数点加算
CEILF.DL	reg2, reg3	倍精度	整数形式への変換
CEILF.DUL	reg2, reg3	倍精度	符号なし整数形式への変換
CEILF.DUW	reg2, reg3	倍精度	符号なし整数形式への変換
CEILF.DW	reg2, reg3	倍精度	整数形式への変換
CEILF.SL	reg2, reg3	単精度	整数形式への変換
CEILF.SUL	reg2, reg3	倍精度	符号なし整数形式への変換
CEILF.SUW	reg2, reg3	倍精度	符号なし整数形式への変換
CEILF.SW	reg2, reg3	単精度	整数形式への変換
CMOVF.D	cc, reg1, reg2, reg3	倍精度	条件付き転送
CMOVF.S	cc, reg1, reg2, reg3	単精度	条件付き転送
CMPF.D	cond, reg2, reg1, fcbt cond, reg1, reg2	倍精度	浮動小数点比較
CMPF.S	cond, reg2, reg1, fcbt cond, reg2, reg1	単精度	浮動小数点比較
CVTF.DL	reg2, reg3	倍精度	整数形式への変換
CVTF.DS	reg2, reg3	倍精度	浮動小数点形式への変換
CVTF.DUL	reg2, reg3	倍精度	符号なし整数形式への変換
CVTF.DUW	reg2, reg3	倍精度	符号なし整数形式への変換
CVTF.DW	reg2, reg3	倍精度	整数形式への変換
CVTF.LD	reg2, reg3	倍精度	浮動小数点形式への変換
CVTF.LS	reg2, reg3	単精度	浮動小数点形式への変換
CVTF.SD	reg2, reg3	倍精度	浮動小数点形式への変換
CVTF.SL	reg2, reg3	単精度	整数形式への変換
CVTF.SUL	reg2, reg3	単精度	符号なし整数形式への変換
CVTF.SUW	reg2, reg3	単精度	符号なし整数形式への変換
CVTF.SW	reg2, reg3	単精度	整数形式への変換
CVTF.ULD	reg2, reg3	倍精度	浮動小数点形式への変換
CVTF.ULS	reg2, reg3	単精度	浮動小数点形式への変換
CVTF.UWD	reg2, reg3	倍精度	浮動小数点形式への変換
CVTF.UWS	reg2, reg3	単精度	浮動小数点形式への変換
CVTF.WD	reg2, reg3	倍精度	浮動小数点形式への変換
CVTF.WS	reg2, reg3	単精度	浮動小数点形式への変換
DIVF.D	reg1, reg2, reg3	倍精度	浮動小数点除算

表A - 2 浮動小数点演算命令機能一覧 (2/2)

二モニック	オペランド	形式	命令機能
DIVF.S	reg1, reg2, reg3	単精度	浮動小数点除算
FLOORF.DL	reg2, reg3	倍精度	整数形式への変換
FLOORF.DUL	reg2, reg3	倍精度	符号なし整数形式への変換
FLOORF.DUW	reg2, reg3	倍精度	符号なし整数形式への変換
FLOORF.DW	reg2, reg3	倍精度	整数形式への変換
FLOORF.SL	reg2, reg3	単精度	整数形式への変換
FLOORF.SUL	reg2, reg3	単精度	符号なし整数形式への変換
FLOORF.SUW	reg2, reg3	単精度	符号なし整数形式への変換
FLOORF.SW	reg2, reg3	単精度	整数形式への変換
MADDF.S	reg1, reg2, reg3, reg4	単精度	浮動小数点積和算
MAXF.D	reg1, reg2, reg3	倍精度	浮動小数点最大値
MAXF.S	reg1, reg2, reg3	単精度	浮動小数点最大値
MINF.D	reg1, reg2, reg3	倍精度	浮動小数点最小値
MINF.S	reg1, reg2, reg3	単精度	浮動小数点最小値
MSUBF.S	reg1, reg2, reg3, reg4	単精度	浮動小数点積和算
MULF.D	reg1, reg2, reg3	倍精度	浮動小数点乗算
MULF.S	reg1, reg2, reg3	単精度	浮動小数点乗算
NEGF.D	reg2, reg3	倍精度	浮動小数点符号反転
NEGF.S	reg2, reg3	単精度	浮動小数点符号反転
NMADDF.S	reg1, reg2, reg3, reg4	単精度	浮動小数点積和算
NMSUBF.S	reg1, reg2, reg3, reg4	単精度	浮動小数点積和算
RECIPF.D	reg2, reg3	倍精度	逆数
RECIPF.S	reg2, reg3	単精度	逆数
RSQRTF.D	reg2, reg3	倍精度	平方根の逆数
RSQRTF.S	reg2, reg3	単精度	平方根の逆数
SQRTF.D	reg2, reg3	倍精度	平方根
SQRTF.S	reg2, reg3	単精度	平方根
SUBF.D	reg1, reg2, reg3	倍精度	浮動小数点減算
SUBF.S	reg1, reg2, reg3	単精度	浮動小数点減算
TRFSR	cc#3	単精度	フラグ転送
TRNCF.DL	reg2, reg3	倍精度	整数形式への変換
TRNCF.DUL	reg2, reg3	倍精度	符号なし整数形式への変換
TRNCF.DUW	reg2, reg3	倍精度	符号なし整数形式への変換
TRNCF.DW	reg2, reg3	倍精度	整数形式への変換
TRNCF.SL	reg2, reg3	単精度	整数形式への変換
TRNCF.SUL	reg2, reg3	単精度	符号なし整数形式への変換
TRNCF.SUW	reg2, reg3	単精度	符号なし整数形式への変換
TRNCF.SW	reg2, reg3	単精度	整数形式への変換

## 付録B 命令オペコード一覧

### B.1 基本命令オペコード一覧

基本命令コードに対応したオペコード・マップを次に示します。

表B-1 基本命令オペコード一覧(16, 32ビット命令)(1/4)

二モニック	オペランド	フォー マツト	オペコード										備 考				
			15	11	10	5	4	0	31	27	26	21		20	16		
NOP			00000	000000	000000												
SYNCE			00000	000000	11101												
SYNCM			00000	000000	11110												
SYNCP			00000	000000	11111												
MOV	reg1, reg2		r r r r r	000000	R R R R R											r r r r r	00000
NOT	reg1, reg2		r r r r r	000001	R R R R R												
RIE			00000	00010	00000												
SWITCH	reg1		00000	000010	R R R R R											R R R R R	00000
FETRAP	vector4		0 i i i i	000010	00000											i i i i	0000
DIVH	reg1, reg2		r r r r r	000010	R R R R R											r r r r r	00000,
																R R R R R	00000
JMP	[reg1]		000000	000011	R R R R R												
SLD.BU	disp4 [ep], reg2	IV	r r r r r	000011	0 d d d d											r r r r r	00000
SLD.HU	disp5 [ep], reg2	IV	r r r r r	000011	1 d d d d											r r r r r	00000
ZXB	reg1		00000	000100	R R R R R												
SXB	reg1		00000	000101	R R R R R												
ZXH	reg1		00000	000110	R R R R R												
SXH	reg1		00000	000111	R R R R R												
SATSUBR	reg1, reg2		r r r r r	000100	R R R R R											r r r r r	00000
SATSUB	reg1, reg2		r r r r r	000101	R R R R R											r r r r r	00000
SATADD	reg1, reg2		r r r r r	000110	R R R R R											r r r r r	00000
MULH	reg1, reg2		r r r r r	000111	R R R R R											r r r r r	00000
OR	reg1, reg2		r r r r r	001000	R R R R R												
XOR	reg1, reg2		r r r r r	001001	R R R R R												
AND	reg1, reg2		r r r r r	001010	R R R R R												
TST	reg1, reg2		r r r r r	001011	R R R R R												
SUBR	reg1, reg2		r r r r r	001100	R R R R R												
SUB	reg1, reg2		r r r r r	001101	R R R R R												
ADD	reg1, reg2		r r r r r	001110	R R R R R												
CMP	reg1, reg2		r r r r r	001111	R R R R R												
MOV	imm5, reg2		r r r r r	010000	i i i i i											r r r r r	00000
SATADD	imm5, reg2		r r r r r	010001	i i i i i											r r r r r	00000

表B - 1 基本命令オペコード一覧(16, 32ビット命令)(2/4)

二モニック	オペランド	フォー マツト	オペコード												備 考		
			15	11	10	5	4	0	31	27	26	21	20	16			
ADD	imm5, reg2	I	r r r r r		0 1 0 0 1 0			i i i i i									
CMP	imm5, reg2	I	r r r r r		0 1 0 0 1 1			i i i i i									
CALLT	imm6	II	0 0 0 0 0		0 1 0 0 0 i			i i i i i									
SHR	imm5, reg2	II	r r r r r		0 1 0 1 0 0			i i i i i									
SAR	imm5, reg2	II	r r r r r		0 1 0 1 0 1			i i i i i									
SHL	imm5, reg2	II	r r r r r		0 1 0 1 1 0			i i i i i									
MULH	imm5, reg2	II	r r r r r		0 1 0 1 1 1			i i i i i									rrrrr 00000
JR	disp32	VI	0 0 0 0 0		0 1 0 1 1 1	0 0 0 0 0			d d d d d	d d d d d d	d d d d d	0					表B - 2参照
JARL	disp32, reg1	VI	0 0 0 0 0		0 1 0 1 1 1	R R R R R			d d d d d	d d d d d d	d d d d d	0					表B - 2参照 R R R R R 00000
SLD.B	disp7 [ep], reg2	IV	r r r r r		0 1 1 0 d d	d d d d d											
SST.B	reg2, disp7 [ep]	IV	r r r r r		0 1 1 1 d d	d d d d d											
SLD.H	disp8 [ep], reg2	IV	r r r r r		1 0 0 0 d d	d d d d d											
SST.H	reg2, disp8 [ep]	IV	r r r r r		1 0 0 1 d d	d d d d d											
SLD.W	disp8 [ep], reg2	IV	r r r r r		1 0 1 0 d d	d d d d 0											
SST.W	reg2, disp8 [ep]	IV	r r r r r		1 0 1 0 d d	d d d d 1											
Bcond	disp9	III	d d d d d		1 0 1 1 d d	d C C C C											
ADDI	imm16, reg1, reg2	VI	r r r r r		1 1 0 0 0 0	R R R R R			i i i i i	i i i i i i	i i i i i						
MOV	imm32, reg1	VI	0 0 0 0 0		1 1 0 0 0 1	R R R R R			I I I I I	I I I I I I	I I I I I						表B - 2参照
MOVEA	imm16, reg1, reg2	VI	r r r r r		1 1 0 0 0 1	R R R R R			i i i i i	i i i i i i	i i i i i						rrrrr 00000
MOVHI	imm16, reg1, reg2	VI	r r r r r		1 1 0 0 1 0	R R R R R			i i i i i	i i i i i i	i i i i i						rrrrr 00000
SATSUBI	imm16, reg1, reg2	VI	r r r r r		1 1 0 0 1 1	R R R R R			i i i i i	i i i i i i	i i i i i						rrrrr 00000
DISPOSE	imm5, list12	XIII	0 0 0 0 0		1 1 0 0 1 i	i i i i i L	L L L L L	L L L L L L	0 0 0 0 0								
DISPOSE	imm5, list12, [reg1]	XIII	0 0 0 0 0		1 1 0 0 1 i	i i i i i L	L L L L L	L L L L L L	R R R R R	R R R R R	0 0 0 0 0						R R R R R 00000
ORI	imm16, reg1, reg2	VI	r r r r r		1 1 0 1 0 0	R R R R R			i i i i i	i i i i i i	i i i i i						
XORI	imm16, reg1, reg2	VI	r r r r r		1 1 0 1 0 1	R R R R R			i i i i i	i i i i i i	i i i i i						
ANDI	imm16, reg1, reg2	VI	r r r r r		1 1 0 1 1 0	R R R R R			i i i i i	i i i i i i	i i i i i						
MULHI	imm16, reg1, reg2	VI	r r r r r		1 1 0 1 1 1	R R R R R			i i i i i	i i i i i i	i i i i i						
JMP	imm32 [reg1]	VI	0 0 0 0 0		1 1 0 1 1 1	R R R R R			d d d d d	d d d d d d	d d d d d	0					表B - 2参照
LD.B	disp16 [reg1], reg2	VII	r r r r r		1 1 1 0 0 0	R R R R R			d d d d d	d d d d d d	d d d d d						
LD.H	disp16 [reg1], reg2	VII	r r r r r		1 1 1 0 0 1	R R R R R			d d d d d	d d d d d d	d d d d d	0					
LD.W	disp16 [reg1], reg2	VII	r r r r r		1 1 1 0 0 1	R R R R R			d d d d d	d d d d d d	d d d d d	1					
ST.B	reg2, disp16 [reg1]	VII	r r r r r		1 1 1 0 1 0	R R R R R			d d d d d	d d d d d d	d d d d d						
ST.H	reg2, disp16 [reg1]	VII	r r r r r		1 1 1 0 1 1	R R R R R			d d d d d	d d d d d d	d d d d d	0					
ST.W	reg2, disp16 [reg1]	VII	r r r r r		1 1 1 0 1 1	R R R R R			d d d d d	d d d d d d	d d d d d	1					
PREPARE	list12, imm5	XIII	0 0 0 0 0		1 1 1 1 0 i	i i i i i L	L L L L L	L L L L L L	0 0 0 0 1								
PREPARE	list12, imm5, sp/imm	XIII	0 0 0 0 0		1 1 1 1 0 i	i i i i i L	L L L L L	L L L L L L	f f 0 1 1								注
LD.B	disp23[reg1], reg3	XIV	0 0 0 0 0		1 1 1 1 0 0	R R R R R R	w w w w w	d d d d d d	d 0 1 0 1								表B - 2参照
LD.H	disp23[reg1], reg3	XIV	0 0 0 0 0		1 1 1 1 0 0	R R R R R R	w w w w w	d d d d d d	0 0 1 1 1								表B - 2参照
LD.W	disp23[reg1], reg3	XIV	0 0 0 0 0		1 1 1 1 0 0	R R R R R R	w w w w w	d d d d d d	0 1 0 0 1								表B - 2参照

注 ff = 01, 10の場合表B - 2参照。 ff = 11の場合表B - 3参照。



表B-1 基本命令オペコード一覧(16, 32ビット命令)(3/4)

ニモニック	オペランド	フォー マツト	オペコード										備 考			
			15	11	10	5	4	0	31	27	26	21		20	16	
ST.B	reg3, disp23[reg1]	XIV	0 0 0 0 0	1 1 1 1 0 0	RRRRRR	wwwww	dddddd	d 1 1 0 1								表B-2参照
ST.W	reg3, disp23[reg1]	XIV	0 0 0 0 0	1 1 1 1 0 0	RRRRRR	wwwww	dddddd	0 1 1 1 1								表B-2参照
LD.BU	disp23[reg1], reg3	XIV	0 0 0 0 0	1 1 1 1 0 1	RRRRRR	wwwww	dddddd	d 0 1 0 1								表B-2参照
LD.HU	disp23[reg1], reg3	XIV	0 0 0 0 0	1 1 1 1 0 1	RRRRRR	wwwww	dddddd	0 0 1 1 1								表B-2参照
ST.H	reg3, disp23[reg1]	XIV	0 0 0 0 0	1 1 1 1 0 1	RRRRRR	wwwww	dddddd	0 1 1 0 1								表B-2参照
JR	disp22	V	0 0 0 0 0	1 1 1 1 0 D	DDDDDD	dddddd	dddddd	dddddd								
JARL	disp22, reg2	V	r r r r r	1 1 1 1 0 D	DDDDDD	dddddd	dddddd	dddddd								rrrrrr 00000
LD.BU	disp16 [reg1], reg2	VII	r r r r r	1 1 1 1 0 b	RRRRR	dddddd	dddddd	dddddd								
SET1	bit3#, disp16 [reg1]	VIII	0 0 b b b	1 1 1 1 1 0	RRRRR	dddddd	dddddd	dddddd								
NOT1	bit#3, disp16 [reg1]	VIII	0 1 b b b	1 1 1 1 1 0	RRRRR	dddddd	dddddd	dddddd								
CLR1	bit3#, disp16 [reg1]	VIII	1 0 b b b	1 1 1 1 1 0	RRRRR	dddddd	dddddd	dddddd								
TST1	bit3#, disp16 [reg1]	VIII	1 1 b b b	1 1 1 1 1 0	RRRRR	dddddd	dddddd	dddddd								
LD.HU	disp16 [reg1], reg2	VII	r r r r r	1 1 1 1 1 1	RRRRR	dddddd	dddddd	dddddd								rrrrrr 00000
SETF	cond, reg2	IX	r r r r r	1 1 1 1 1 1	0CCCC	00000	000000	000000								
RIE		X	x x x x x	1 1 1 1 1 1	1 x x x x	00000	000000	000000								
LDSR	reg2, regID	IX	r r r r r	1 1 1 1 1 1	RRRRR	00000	000001	000000								
STSR	sr1, reg2	IX	r r r r r	1 1 1 1 1 1	RRRRR	00000	000010	000000								
SHR	reg1, reg2	IX	r r r r r	1 1 1 1 1 1	RRRRR	00000	000100	000000								
SHR	reg1, reg2, reg3	IX	r r r r r	1 1 1 1 1 1	RRRRR	wwwww	000100	000010								
SAR	reg1, reg2	IX	r r r r r	1 1 1 1 1 1	RRRRR	00000	000101	000000								
SAR	reg1, reg2, reg3	XI	r r r r r	1 1 1 1 1 1	RRRRR	wwwww	000101	000010								
SHL	reg1, reg2	IX	r r r r r	1 1 1 1 1 1	RRRRR	00000	000110	000000								
SHL	reg1, reg2, reg3	XI	r r r r r	1 1 1 1 1 1	RRRRR	wwwww	000110	000010								
SET1	reg2, [reg1]	IX	r r r r r	1 1 1 1 1 1	RRRRR	00000	000111	000000								
NOT1	reg2, [reg1]	IX	r r r r r	1 1 1 1 1 1	RRRRR	00000	000111	000010								
CLR1	reg2, [reg1]	IX	r r r r r	1 1 1 1 1 1	RRRRR	00000	000111	001000								
TST1	reg2, [reg1]	IX	r r r r r	1 1 1 1 1 1	RRRRR	00000	000111	001100								
CAXI	[reg1], reg2, reg3	XI	r r r r r	1 1 1 1 1 1	RRRRR	wwwww	000111	0 1 1 1 0								
TRAP	imm5	X	0 0 0 0 0	1 1 1 1 1 1	i i i i i	00000	001000	000000								
HALT		X	0 0 0 0 0	1 1 1 1 1 1	0 0 0 0 0	00000	001001	000000								
RETI		X	0 0 0 0 0	1 1 1 1 1 1	0 0 0 0 0	00000	001010	000000								
CTRET		X	0 0 0 0 0	1 1 1 1 1 1	0 0 0 0 0	00000	001010	001000								
EIRET		X	0 0 0 0 0	1 1 1 1 1 1	0 0 0 0 0	00000	001010	0 1 0 0 0								
FERET		X	0 0 0 0 0	1 1 1 1 1 1	0 0 0 0 0	00000	001010	0 1 0 1 0								
DI		X	0 0 0 0 0	1 1 1 1 1 1	0 0 0 0 0	00000	001011	000000								
EI		X	1 0 0 0 0	1 1 1 1 1 1	0 0 0 0 0	00000	001011	000000								
SYSCALL	vector8	X	1 1 0 1 0	1 1 1 1 1 1	v v v v v	00VVV	001011	000000								
SASF	cccc, reg2	IX	r r r r r	1 1 1 1 1 1	0 c c c c	00000	010000	000000								
MUL	reg1, reg2, reg3	XI	r r r r r	1 1 1 1 1 1	RRRRR	wwwww	010001	000000								
MULU	reg1, reg2, reg3	XI	r r r r r	1 1 1 1 1 1	RRRRR	wwwww	010001	000010								
MUL	imm9, reg2, reg3	XII	r r r r r	1 1 1 1 1 1	i i i i i	wwwww	01001I	IIII00								
MULU	imm9, reg2, reg3	XII	r r r r r	1 1 1 1 1 1	i i i i i	wwwww	01001I	IIII10								

表B - 1 基本命令オペコード一覧 (16, 32ビット命令) (4/4)

ニモニック	オペランド	フォー マツト	オペコード										備 考				
			15	11	10	5	4	0	31	27	26	21		20	16		
DIVH	reg1, reg2, reg3	XI	r r r r r	1 1 1 1 1 1	R R R R R	w w w w w	0 1 0 1 0 0	0 0 0 0 0									
DIVHU	reg1, reg2, reg3	XI	r r r r r	1 1 1 1 1 1	R R R R R	w w w w w	0 1 0 1 0 0	0 0 0 1 0									
DIV	reg1, reg2, reg3	XI	r r r r r	1 1 1 1 1 1	R R R R R	w w w w w	0 1 0 1 1 0	0 0 0 0 0									
DIVQ	reg1, reg2, reg3	XI	r r r r r	1 1 1 1 1 1	R R R R R	w w w w w	0 1 0 1 1 1	1 1 1 0 0									
DIVU	reg1, reg2, reg3	XI	r r r r r	1 1 1 1 1 1	R R R R R	w w w w w	0 1 0 1 1 0	0 0 0 1 0									
DIVQU	reg1, reg2, reg3	XI	r r r r r	1 1 1 1 1 1	R R R R R	w w w w w	0 1 0 1 1 1	1 1 1 1 0									
CMOV	cccc, imm5, reg2, reg3	XI	r r r r r	1 1 1 1 1 1	i i i i i	w w w w w	0 1 1 0 0 0	c c c c 0									
CMOV	cccc, reg1, reg2, reg3	XI	r r r r r	1 1 1 1 1 1	R R R R R	w w w w w	0 1 1 0 0 1	c c c c 0									
BSW	reg2, reg3	XII	r r r r r	1 1 1 1 1 1	0 0 0 0 0	w w w w w	0 1 1 0 1 0	0 0 0 0 0									
BSH	reg2, reg3	XII	r r r r r	1 1 1 1 1 1	0 0 0 0 0	w w w w w	0 1 1 0 1 0	0 0 0 1 0									
HSW	reg2, reg3	XII	r r r r r	1 1 1 1 1 1	0 0 0 0 0	w w w w w	0 1 1 0 1 0	0 0 1 0 0									
HSH	reg2, reg3	XII	r r r r r	1 1 1 1 1 1	0 0 0 0 0	w w w w w	0 1 1 0 1 0	0 0 1 1 0									
SCH0R	reg2, reg3	IX	r r r r r	1 1 1 1 1 1	0 0 0 0 0	w w w w w	0 1 1 0 1 1	0 0 0 0 0									
SCH1R	reg2, reg3	IX	r r r r r	1 1 1 1 1 1	0 0 0 0 0	w w w w w	0 1 1 0 1 1	0 0 0 1 0									
SCH0L	reg2, reg3	IX	r r r r r	1 1 1 1 1 1	0 0 0 0 0	w w w w w	0 1 1 0 1 1	0 0 1 0 0									
SCH1L	reg2, reg3	IX	r r r r r	1 1 1 1 1 1	0 0 0 0 0	w w w w w	0 1 1 0 1 1	0 0 1 1 0									
SBF	cccc, reg1, reg2, reg3	XI	r r r r r	1 1 1 1 1 1	R R R R R	w w w w w	0 1 1 1 0 0	c c c c 0	cccc 1101								
SATSUB	reg1, reg2, reg3	XI	r r r r r	1 1 1 1 1 1	R R R R R	w w w w w	0 1 1 1 0 0	1 1 0 1 0									
ADF	cccc, reg1, reg2, reg3	XI	r r r r r	1 1 1 1 1 1	R R R R R	w w w w w	0 1 1 1 0 1	c c c c 0	cccc 1101								
SATADD	reg1, reg2, reg3	XI	r r r r r	1 1 1 1 1 1	R R R R R	w w w w w	0 1 1 1 0 1	1 1 0 1 0									
MAC	reg1, reg2, reg3, reg4	XI	r r r r r	1 1 1 1 1 1	R R R R R	w w w w 0	0 1 1 1 1 0	m m m m 0									
MACU	reg1, reg2, reg3, reg4	XI	r r r r r	1 1 1 1 1 1	R R R R R	w w w w 0	0 1 1 1 1 1	m m m m 0									

表B-2 基本命令オペコード一覧(48ビット命令)

二モニク	オペランド	フォー マツ	オペコード																			
			15	11	10	5	4	0	31	27	26	21	20	16	47	43	42	37	36	32		
JR	disp32	VI	00000	010111	00000			dddd	dddddd	ddd0			DDDD	DDDDDD	DDDD							
JARL	disp32, reg1	VI	00000	010111	RRRRR			dddd	dddddd	ddd0			DDDD	DDDDDD	DDDD							
MOV	imm32, reg1	VI	00000	110001	RRRRR			iiii	iiiiii	iiii			IIII	IIIIII	IIII							
JMP	disp32 [reg1]	VI	00000	110111	RRRRR			dddd	dddddd	ddd0			DDDD	DDDDDD	DDDD							
PREPARE	list12, imm5, sp/imm	XIII	00000	11110i	iiiiL			LLLL	LLLLLL	ff*011			IIII	IIIIII	IIII							
LD.B	disp23[reg1], reg3	XIV	00000	111100	RRRRR			www	dddd	d0101			DDDD	DDDDDD	DDDD							
LD.H	disp23[reg1], reg3	XIV	00000	111100	RRRRR			www	dddd	00111			DDDD	DDDDDD	DDDD							
LD.W	disp23[reg1], reg3	XIV	00000	111100	RRRRR			www	dddd	01001			DDDD	DDDDDD	DDDD							
ST.B	reg3, disp23[reg1]	XIV	00000	111100	RRRRR			www	dddd	d1101			DDDD	DDDDDD	DDDD							
ST.W	reg3, disp23[reg1]	XIV	00000	111100	RRRRR			www	dddd	01111			DDDD	DDDDDD	DDDD							
LD.BU	disp23[reg1], reg3	XIV	00000	111101	RRRRR			www	dddd	d0101			DDDD	DDDDDD	DDDD							
LD.HU	disp23[reg1], reg3	XIV	00000	111101	RRRRR			www	dddd	00111			DDDD	DDDDDD	DDDD							
ST.H	reg3, disp23[reg1]	XIV	00000	111101	RRRRR			www	dddd	01101			DDDD	DDDDDD	DDDD							

注 ff = 01, 10

表B-3 基本命令オペコード一覧(64ビット命令)

二モニク	オペランド	フォー マツ	オペコード												備 考								
			15	11	10	5	4	0	31	27	26	21	20	16									
PREPARE	list12, imm5, sp/imm	XIII	00000	11110i	iiiiL			LLLL	LLLLLL	11011													
			47			32	63			48													
			IIIII	IIIIII	IIIII	IIIII	IIIIII	IIIII															

## B.2 浮動小数点演算命令オペコード一覧

浮動小数点演算命令コードに対応したオペコード・マップを次に示します。

表B-4 浮動小数点演算命令オペコード一覧 (1/2)

二モニック	オペランド	形式	オペコード												備考								
			15	11	10	5	4	0	31	27	26	21	20	16									
ABSF.D	reg2, reg3	倍精度	rrrr0	111111	00000	www0	100010	11000															
ABSF.S	reg2, reg3	単精度	rrrrr	111111	00000	www	100010	01000															
ADDF.D	reg1, reg2, reg3	倍精度	rrrr0	111111	RRRR0	www0	100011	10000															
ADDF.S	reg1, reg2, reg3	単精度	rrrrr	111111	RRRRR	www	100011	00000															
CEILF.DL	reg2, reg3	倍精度	rrrr0	111111	00010	www0	100010	10100															
CEILF.DUL	reg2, reg3	倍精度	rrrr0	111111	10010	www0	100010	10100															
CEILF.DUW	reg2, reg3	倍精度	rrrr0	111111	10010	www	100010	10000															
CEILF.DW	reg2, reg3	倍精度	rrrr0	111111	00010	www	100010	10000															
CEILF.SL	reg2, reg3	単精度	rrrrr	111111	00010	www0	100010	00100															
CEILF.SUL	reg2, reg3	単精度	rrrrr	111111	10010	www0	100010	00100															
CEILF.SUW	reg2, reg3	単精度	rrrrr	111111	10010	www	100010	00000															
CEILF.SW	reg2, reg3	単精度	rrrrr	111111	00010	www	100010	00000															
CMOVF.D	cc, reg1, reg2, reg3	倍精度	rrrr0	111111	RRRR0	www0	100000	1fff0	www	0000													
CMOVF.S	cc, reg1, reg2, reg3	単精度	rrrrr	111111	RRRRR	www	100000	0fff0	www	0000													
CMPF.D	cond, reg1, reg2, cc#3	倍精度	rrrr0	111111	RRRR0	0FFFF	100001	1fff0															
CMPF.S	cond, reg1, reg2, cc#3	単精度	rrrrr	111111	RRRRR	0FFFF	100001	0fff0															
CVTF.DL	reg2, reg3	倍精度	rrrr0	111111	00100	www0	100010	10100															
CVTF.DS	reg2, reg3	倍精度	rrrr0	111111	00011	www	100010	10010															
CVTF.DUL	reg2, reg3	倍精度	rrrr0	111111	10100	www0	100010	10100															
CVTF.DUW	reg2, reg3	倍精度	rrrr0	111111	10100	www	100010	10000															
CVTF.DW	reg2, reg3	倍精度	rrrr0	111111	00100	www	100010	10000															
CVTF.LD	reg2, reg3	倍精度	rrrr0	111111	00001	www0	100010	10010															
CVTF.LS	reg2, reg3	単精度	rrrr0	111111	00001	www	100010	00010															
CVTF.SD	reg2, reg3	倍精度	rrrrr	111111	00010	www0	100010	10010															
CVTF.SL	reg2, reg3	単精度	rrrrr	111111	00100	www0	100010	00100															
CVTF.SUL	reg2, reg3	単精度	rrrrr	111111	10100	www0	100010	00100															
CVTF.SUW	reg2, reg3	単精度	rrrrr	111111	10100	www	100010	00000															
CVTF.SW	reg2, reg3	単精度	rrrrr	111111	00100	www	100010	00000															
CVTF.ULD	reg2, reg3	倍精度	rrrr0	111111	10001	www0	100010	10010															
CVTF.ULS	reg2, reg3	単精度	rrrr0	111111	10001	www	100010	00010															
CVTF.UWD	reg2, reg3	倍精度	rrrrr	111111	10000	www0	100010	10010															
CVTF.UWS	reg2, reg3	単精度	rrrrr	111111	10000	www	100010	00010															
CVTF.WD	reg2, reg3	倍精度	rrrrr	111111	00000	www0	100010	10010															
CVTF.WS	reg2, reg3	単精度	rrrrr	111111	00000	www	100010	00010															
DIVF.D	reg1, reg2, reg3	倍精度	rrrr0	111111	RRRR0	www0	100011	11110															
DIVF.S	reg1, reg2, reg3	単精度	rrrrr	111111	RRRRR	www	100011	01110															
FLOORF.DL	reg2, reg3	倍精度	rrrr0	111111	00011	www0	100010	10100															
FLOORF.DUL	reg2, reg3	倍精度	rrrr0	111111	10011	www0	100010	10100															
FLOORF.DUW	reg2, reg3	倍精度	rrrr0	111111	10011	www	100010	10000															

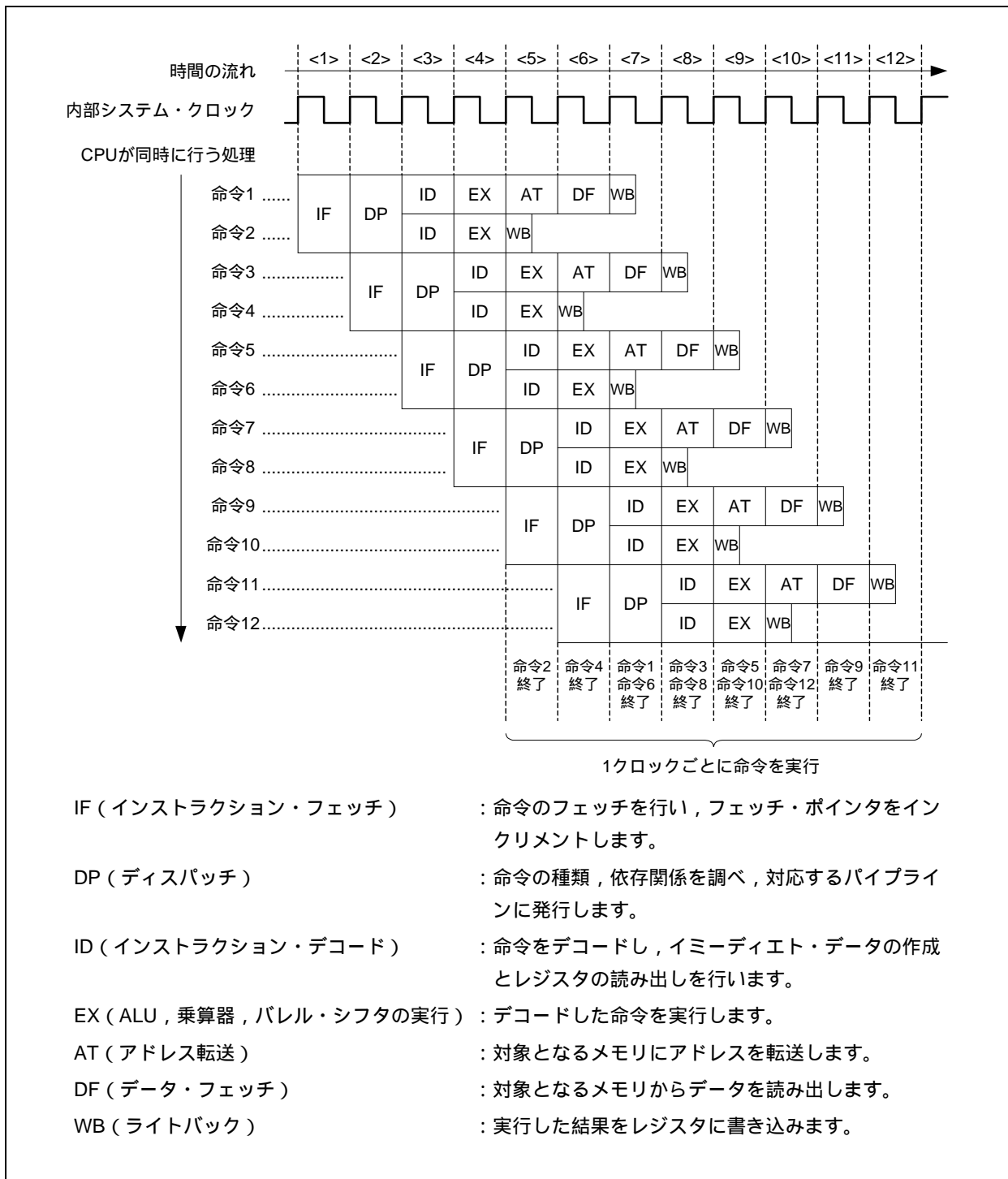
表B-4 浮動小数点演算命令オペコード一覧 (2/2)

二モニック	オペランド	形式	オペコード								備考		
			15	11	10	5	4	0	31	27		26	21
FLOORF.DW	reg2, reg3	倍精度	rrrr0	111111	00011	wwwww	100010	10000					
FLOORF.SL	reg2, reg3	単精度	rrrrr	111111	00011	www0	100010	00100					
FLOORF.SUL	reg2, reg3	単精度	rrrrr	111111	10011	www0	100010	00100					
FLOORF.SUW	reg2, reg3	単精度	rrrrr	111111	10011	wwwww	100010	00000					
FLOORF.SW	reg2, reg3	単精度	rrrrr	111111	00011	wwwww	100010	00000					
MADDF.S	reg1, reg2, reg3, reg4	単精度	rrrrr	111111	RRRR	wwwww	101W00	WWWW0					
MAXF.D	reg1, reg2, reg3	倍精度	rrrr0	111111	RRR0	www0	100011	11000					
MAXF.S	reg1, reg2, reg3	単精度	rrrrr	111111	RRRR	wwwww	100011	01000					
MINF.D	reg1, reg2, reg3	倍精度	rrrr0	111111	RRR0	www0	100011	11010					
MINF.S	reg1, reg2, reg3	単精度	rrrrr	111111	RRRR	wwwww	100011	01010					
MSUBF.S	reg1, reg2, reg3, reg4	単精度	rrrrr	111111	RRRR	wwwww	101W01	WWWW0					
MULF.D	reg1, reg2, reg3	倍精度	rrrr0	111111	RRR0	www0	100011	10100					
MULF.S	reg1, reg2, reg3	単精度	rrrrr	111111	RRRR	wwwww	100011	00100					
NEGF.D	reg2, reg3	倍精度	rrrr0	111111	00001	www0	100010	11000					
NEGF.S	reg2, reg3	単精度	rrrrr	111111	00001	wwwww	100010	01000					
NMADDF.S	reg1, reg2, reg3, reg4	単精度	rrrrr	111111	RRRR	wwwww	101W10	WWWW0					
NMSUBF.S	reg1, reg2, reg3, reg4	単精度	rrrrr	111111	RRRR	wwwww	101W11	WWWW0					
RECIPF.D	reg2, reg3	倍精度	rrrr0	111111	00001	www0	100010	11110					
RECIPF.S	reg2, reg3	単精度	rrrrr	111111	00001	wwwww	100010	01110					
RSQRTF.D	reg2, reg3	倍精度	rrrr0	111111	00010	www0	100010	11110					
RSQRTF.S	reg2, reg3	単精度	rrrrr	111111	00010	wwwww	100010	01110					
SQRTF.D	reg2, reg3	倍精度	rrrr0	111111	00000	www0	100010	11110					
SQRTF.S	reg2, reg3	単精度	rrrrr	111111	00000	wwwww	100010	01110					
SUBF.D	reg1, reg2, reg3	倍精度	rrrr0	111111	RRR0	www0	100011	10010					
SUBF.S	reg1, reg2, reg3	単精度	rrrrr	111111	RRRR	wwwww	100011	00010					
TRFSR	cc#3	単精度	00000	111111	00000	00000	100000	0fff0					
TRNCF.DL	reg2, reg3	倍精度	rrrr0	111111	00001	www0	100010	10100					
TRNCF.DUL	reg2, reg3	倍精度	rrrr0	111111	10001	www0	100010	10100					
TRNCF.DUW	reg2, reg3	倍精度	rrrr0	111111	10001	wwwww	100010	10000					
TRNCF.DW	reg2, reg3	倍精度	rrrr0	111111	00001	wwwww	100010	10000					
TRNCF.SL	reg2, reg3	単精度	rrrrr	111111	00001	www0	100010	00100					
TRNCF.SUL	reg2, reg3	単精度	rrrrr	111111	10001	www0	100010	00100					
TRNCF.SUW	reg2, reg3	単精度	rrrrr	111111	10001	wwwww	100010	00000					
TRNCF.SW	reg2, reg3	単精度	rrrrr	111111	00001	wwwww	100010	00000					

## 付録C パイプライン

V850E2M CPUは、RISCアーキテクチャをベースとし、7段パイプラインの制御によりほとんどの命令を1クロックで実行します。命令実行手順は、通常、インストラクション・フェッチ（IF）からライトバック（WB）までの7つのステージで構成されています。各ステージの実行時間は、命令の種類やアクセスの対象となるメモリの種類などによって異なります。パイプラインの動作例として、標準的な命令を12個続けて実行したときのCPUの処理を図C - 1に示します。

図C - 1 標準的な命令を12個続けて実行する例



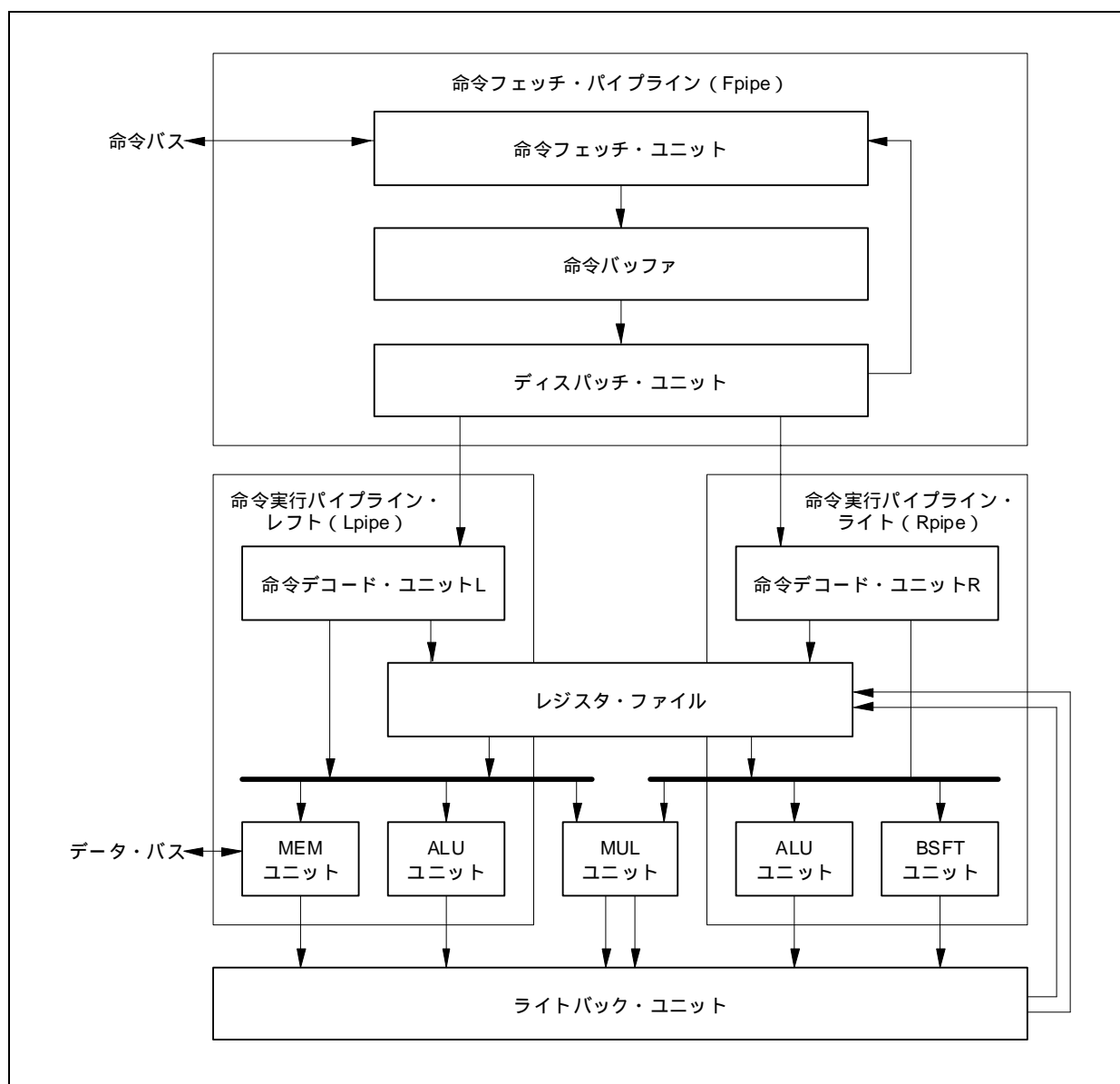
<1> - <12>は、CPUのステータスを示します。標準的な命令では、1クロックに2つの命令の実行 (EX) が並列に行えます。

## C.1 特 徴

V850E2M CPUが想定するCPUのパイプライン構成を図C - 2に示します。CPUは次に示す独立した3つのパイプラインを持ち、命令の依存関係を検出し、最大で2つの命令を同時に発行します。

- 命令フェッチ・パイプライン (Fpipe)
- 命令実行パイプライン・レフト (Lpipe)
- 命令実行パイプライン・ライト (Rpipe)

図C - 2 パイプライン構成





**(1) 命令フェッチ・パイプライン (Fpipe)**

次に示す 3 つのユニットで構成されています。

**(a) 命令フェッチ・ユニット**

128 ビットのフェッチ・バス (iLB) から最大 8 命令 (1 命令が 16 ビットの場合) を 1 サイクルでフェッチします。

**(b) ディスパッチ・ユニット**

128 ビット×3 段の命令キューを内蔵しており、このキューで命令の依存関係を検出し、最大で 2 つの命令を効率良く命令実行パイプラインに発行します。

**(c) 命令バッファ**

命令フェッチ・ユニットによってフェッチされた命令を格納します。

**(2) 命令実行パイプライン・レフト (Lpipe)**

次に示す 3 つのユニットで構成されています。

**(a) 命令デコード・ユニットL**

ディスパッチ・ユニットから発行された命令をデコードします。

**(b) ALUユニット**

整数演算，論理演算を行う命令を実行します。

**(c) MEMユニット**

ロード命令，ストア命令を含むメモリ・アクセスを行う命令を実行します。

**(3) 命令実行パイプライン・ライト (Rpipe)**

次に示す 3 つのユニットで構成されています。

**(a) 命令デコード・ユニットR**

ディスパッチ・ユニットから発行された命令をデコードします。

**(b) ALUユニット**

整数演算，論理演算を行う命令を実行します。

**(c) BSFTユニット**

データ操作を行う命令を実行します。

**(4) MULユニット**

整数乗算を行う命令を実行します。

**(5) ライトバック・ユニット**

レジスタ・ファイルにライトバックする制御をします。

## C.2 命令実行クロック数

### C.2.1 基本命令の実行クロック数

表C - 1に基本命令の実行クロック数一覧を示します。なお、実行クロック数は、命令の組み合わせにより異なる場合があります。詳細については、C.3 基本命令のパイプラインを参照してください。

表C - 1 基本命令の命令実行クロック数一覧 (1/4)

命令の種類	二モニック	オペランド	バイト数	実行クロック数			並列発行 <sup>注1</sup>
				issue	repeat	latency	
ロード命令	LD.B	disp16 [reg1], reg2	4	1	1	3 <sup>注2</sup>	(L)
	LD.B	disp23 [reg1], reg3	6	1	1	3 <sup>注2</sup>	x
	LD.BU	disp16 [reg1], reg2	4	1	1	3 <sup>注2</sup>	(L)
	LD.BU	disp23 [reg1], reg3	6	1	1	3 <sup>注2</sup>	x
	LD.H	disp16 [reg1], reg2	4	1	1	3 <sup>注2</sup>	(L)
	LD.H	disp23 [reg1], reg3	6	1	1	3 <sup>注2</sup>	x
	LD.HU	disp16 [reg1], reg2	4	1	1	3 <sup>注2</sup>	(L)
	LD.HU	disp23 [reg1], reg3	6	1	1	3 <sup>注2</sup>	x
	LD.W	disp16 [reg1], reg2	4	1	1	3 <sup>注2</sup>	(L)
	LD.W	disp23 [reg1], reg3	6	1	1	3 <sup>注2</sup>	x
	SLD.B	disp7 [ep], reg2	2	1	1	3 <sup>注2</sup>	(L)
	SLD.BU	disp4 [ep], reg2	2	1	1	3 <sup>注2</sup>	(L)
	SLD.H	disp8 [ep], reg2	2	1	1	3 <sup>注2</sup>	(L)
	SLD.HU	disp5 [ep], reg2	2	1	1	3 <sup>注2</sup>	(L)
	SLD.W	disp8 [ep], reg2	2	1	1	3 <sup>注2</sup>	(L)
ストア命令	ST.B	reg2, disp16 [reg1]	4	1	1	1	(L)
	ST.B	reg3, disp23 [reg1]	6	1	1	1	x
	ST.H	reg2, disp16 [reg1]	4	1	1	1	(L)
	ST.H	reg3, disp23 [reg1]	6	1	1	1	x
	ST.W	reg2, disp16 [reg1]	4	1	1	1	(L)
	ST.W	reg3, disp23 [reg1]	6	1	1	1	x
	SST.B	reg2, disp7 [ep]	2	1	1	1	(L)
	SST.H	reg2, disp8 [ep]	2	1	1	1	(L)
	SST.W	reg2, disp8 [ep]	2	1	1	1	(L)
乗算命令	MUL	reg1, reg2, reg3	4	1	1	3	(L)
	MUL	imm9, reg2, reg3	4	1	1	3	(L)
	MULH	reg1, reg2	2	1	1	3	(L)
	MULH	imm5, reg2	2	1	1	3	(L)
	MULHI	imm16, reg1, reg2	4	1	1	3	(L)
	MULU	reg1, reg2, reg3	4	1	1	3	(L)
	MULU	imm9, reg2, reg3	4	1	1	3	(L)
加算付き乗算命令	MAC	reg1, reg2, reg3, reg4	4	1	1	3	x
	MACU	reg1, reg2, reg3, reg4	4	1	1	3	x

表C - 1 命令実行クロック数一覧 (2/4)

命令の種類	二モニック	オペランド	バイト数	実行クロック数			並列発行 <sup>※1</sup>
				issue	repeat	latency	
算術演算命令	ADD	reg1, reg2	2	1	1	1	(R/L)
	ADD	imm5, reg2	2	1	1	1	(R/L)
	ADDI	imm16, reg1, reg2	4	1	1	1	(R/L)
	CMP	reg1, reg2	2	1	1	1	(R/L)
	CMP	imm5, reg2	2	1	1	1	(R/L)
	MOV	reg1, reg2	2	1	1	1	(R/L)
	MOV	imm5, reg2	2	1	1	1	(R/L)
	MOV	imm32, reg1	6	1	1	1	x
算術演算命令	MOVEA	imm16, reg1, reg2	4	1	1	1	(R/L)
	MOVHI	imm16, reg1, reg2	4	1	1	1	(R/L)
	SUB	reg1, reg2	2	1	1	1	(R/L)
	SUBR	reg1, reg2	2	1	1	1	(R/L)
条件付き演算命令	ADF	cccc, reg1, reg2, reg3	4	1	1	1	x
	SBF	cccc, reg1, reg2, reg3	4	1	1	1	x
飽和演算命令	SATADD	reg1, reg2	2	1	1	1	(R/L)
	SATADD	imm5, reg2	2	1	1	1	(R/L)
	SATADD	reg1, reg2, reg3	4	1	1	1	(R/L)
	SATSUB	reg1, reg2	2	1	1	1	(R/L)
	SATSUB	reg1, reg2, reg3	4	1	1	1	(R/L)
	SATSUBI	imm16, reg1, reg2	4	1	1	1	(R/L)
	SATSUBR	reg1, reg2	2	1	1	1	(R/L)
論理演算命令	AND	reg1, reg2	2	1	1	1	(R/L)
	ANDI	imm16, reg1, reg2	4	1	1	1	(R/L)
	NOT	reg1, reg2	2	1	1	1	(R/L)
	OR	reg1, reg2	2	1	1	1	(R/L)
	ORI	imm16, reg1, reg2	4	1	1	1	(R/L)
	TST	reg1, reg2	2	1	1	1	(R/L)
	XOR	reg1, reg2	2	1	1	1	(R/L)
	XORI	imm16, reg1, reg2	4	1	1	1	(R/L)
データ操作命令	BSH	reg2, reg3	4	1	1	1	(R)
	BSW	reg2, reg3	4	1	1	1	(R)
	CMOV	cccc, reg1, reg2, reg3	4	1	1	1	(R)
	CMOV	cccc, imm5, reg2, reg3	4	1	1	1	(R)
	HSH	reg2, reg3	4	1	1	1	(R)
	HSW	reg2, reg3	4	1	1	1	(R)
	SAR	reg1, reg2	4	1	1	1	(R)
	SAR	imm5, reg2	2	1	1	1	(R)
	SAR	reg1, reg2, reg3	4	1	1	1	(R)
	SASF	cccc, reg2	4	1	1	1	(R)
	SETF	cccc, reg2	4	1	1	1	(R)
	SHL	reg1, reg2	4	1	1	1	(R)
	SHL	imm5, reg2	2	1	1	1	(R)
	SHL	reg1, reg2, reg3	4	1	1	1	(R)

表C - 1 命令実行クロック数一覧 (3/4)

命令の種類	二モニック	オペランド	バイト数	実行クロック数			並列発行 <sup>注1</sup>
				issue	repeat	latency	
データ操作命令	SHR	reg1, reg2	4	1	1	1	(R)
	SHR	imm5, reg2	2	1	1	1	(R)
	SHR	reg1, reg2, reg3	4	1	1	1	(R)
データ操作命令	SXB	reg1	2	1	1	1	x
	SXH	reg1	2	1	1	1	x
	ZXB	reg1	2	1	1	1	x
	ZXH	reg1	2	1	1	1	x
ビット・サーチ命令	SCH0L	reg2, reg3	4	1	1	1	(R)
	SCH0R	reg2, reg3	4	1	1	1	(R)
	SCH1L	reg2, reg3	4	1	1	1	(R)
	SCH1R	reg2, reg3	4	1	1	1	(R)
除算命令	DIV	reg1, reg2, reg3	4	36	36	36	x
	DIVH	reg1, reg2	2	36	36	36	x
	DIVH	reg1, reg2, reg3	4	36	36	36	x
	DIVHU	reg1, reg2, reg3	4	35	35	35	x
	DIVU	reg1, reg2, reg3	4	35	35	35	x
高速除算命令	DIVQ	reg1, reg2, reg3	4	N+5 <sup>注3</sup>	N+5 <sup>注3</sup>	N+5 <sup>注3</sup>	x
	DIVQU	reg1, reg2, reg3	4	N+4 <sup>注3</sup>	N+4 <sup>注3</sup>	N+4 <sup>注3</sup>	x
分岐命令	Bcond	disp9 (条件成立時)	2	4 <sup>注4</sup>	4 <sup>注4</sup>	4 <sup>注4</sup>	(R/L)
		disp9 (条件不成立時)	2	1	1	1	(R/L)
	JARL	disp22, reg2	4	4	4	4	(R/L)
	JARL	disp32, reg1	6	4	4	4	x
	JMP	[reg1]	2	4	4	4	(R/L)
	JMP	disp32 [reg1]	6	5	5	5	x
	JR	disp22	4	4	4	4	(R/L)
	JR	disp32	6	4	4	4	x
ビット操作命令	CLR1	bit#3, disp16 [reg1]	4	4 <sup>注5</sup>	4 <sup>注5</sup>	4 <sup>注5</sup>	x
	CLR1	reg2, [reg1]	4	4 <sup>注5</sup>	4 <sup>注5</sup>	4 <sup>注5</sup>	x
	NOT1	bit#3, disp16 [reg1]	4	4 <sup>注5</sup>	4 <sup>注5</sup>	4 <sup>注5</sup>	x
	NOT1	reg2, [reg1]	4	4 <sup>注5</sup>	4 <sup>注5</sup>	4 <sup>注5</sup>	x
	SET1	bit#3, disp16 [reg1]	4	4 <sup>注5</sup>	4 <sup>注5</sup>	4 <sup>注5</sup>	x
	SET1	reg2, [reg1]	4	4 <sup>注5</sup>	4 <sup>注5</sup>	4 <sup>注5</sup>	x
	TST1	bit#3, disp16 [reg1]	4	4 <sup>注5</sup>	4 <sup>注5</sup>	4 <sup>注5</sup>	x
	TST1	reg2, [reg1]	4	4 <sup>注5</sup>	4 <sup>注5</sup>	4 <sup>注5</sup>	x
特殊命令	CALLT	imm6	2	10	10	10	x
	CAXI	[reg1], reg2, reg3	4	4 <sup>注5</sup>	4 <sup>注5</sup>	4 <sup>注5</sup>	x
	CTRET	-	4	7	7	7	x
	DI	-	4	2	2	2	x
	DISPOSE	imm5, list12	4	n+2 <sup>注6</sup>	n+2 <sup>注6</sup>	n+2 <sup>注6</sup>	x
	DISPOSE	imm5, list12, [reg1]	4	n+6 <sup>注6</sup>	n+6 <sup>注6</sup>	n+6 <sup>注6</sup>	x
	EI	-	4	2	2	2	x
	EIRET	-	4	7	7	7	x
	FERET	-	4	7	7	7	x

表C - 1 命令実行クロック数一覧 (4/4)

命令の種類	二モニック	オペランド	バイト	実行クロック数			並列発行 <sup>注1</sup>	
				issue	repeat	latency		
特殊命令	FETRAP	vector	2	7	7	7	×	
	HALT	-	4	1	1	1	×	
	LDSR	reg2, regID ( BSELレジスタ , MCCレジスタ )	4	4	4	4	×	
			reg2, regID ( MPUグループ ( MCCレジスタを除く ) )	4	1	1	1	( R )
			reg2, regID ( FPUグループ , ユーザ・グループ )	4	3	3	3	×
			reg2, regID ( その他の定義済 みバンク )	4	2	2	2	×
	NOP	-	2	1	1	1	×	
	PREPARE	list12, imm5	4	$n+2$ <sup>注6</sup>	$n+2$ <sup>注6</sup>	$n+2$ <sup>注6</sup>	×	
	PREPARE	list12, imm5, sp	4	$n+2$ <sup>注6</sup>	$n+2$ <sup>注6</sup>	$n+2$ <sup>注6</sup>	×	
	PREPARE	list12, imm5, imm16	6	$n+2$ <sup>注6</sup>	$n+2$ <sup>注6</sup>	$n+2$ <sup>注6</sup>	×	
	PREPARE	list12, imm5, imm16<<16	6	$n+2$ <sup>注6</sup>	$n+2$ <sup>注6</sup>	$n+2$ <sup>注6</sup>	×	
	PREPARE	list12, imm5, imm32	8	$n+2$ <sup>注6</sup>	$n+2$ <sup>注6</sup>	$n+2$ <sup>注6</sup>	×	
	RETI	-	4	7	7	7	×	
	RIE	-	4	7	7	7	×	
	STSR	regID, reg2	4	1	1	1	×	
	SWITCH	reg1	2	8	8	8	×	
	SYNCE	-	2	不定	不定	不定	×	
	SYNCM	-	2	不定	不定	不定	×	
	SYNCP	-	2	不定	不定	不定	×	
	SYSCALL	vector8	4	10	10	10	×	
TRAP	vector5	4	7	7	7	×		
予約未定義命令コード ( RIE命令として動作 )			4	7	7	7	×	

注1. 「 」は、ほかの命令との並列発行が可能であることを、「×」は、ほかの命令との並列発行が不可能であること(単独で発行)を示します。また、カッコ内は、並列発行を行う際に使用するパイプラインを示します(L : Lpipe, R : Rpipe, R/L : Rpipe または Lpipe)。使用するパイプラインが同一のものは並列発行は不可能です。詳細については、C.3 基本命令のパイプラインを参照してください。

2. ウェイト・ステートがない場合 (3+リード・アクセス・ウェイト・ステート数)。

3.  $N = (\text{被除数の有効ビット数}) - (\text{除数の有効ビット数})$  です。

ただし、N がマイナスとなった場合、 $N = 0$  として取り扱います。

4. 直前に PSW レジスタの内容を書き換える命令がある場合でも4クロックです。また、直前の命令が PSW レジスタを書き換える場合でも、並列発行が可能です。

5. ウェイト・ステートがない場合 (4+リード・アクセス・ウェイト・ステート数)。

6. n は、list x で指定されるレジスタの合計数 (ウェイト・ステート数による。ウェイト・ステートがない場合、n は list x で指定されるレジスタの合計数と一致)。

補足 :  $n = 0$  ,  $n = 1$  の場合は4クロックです (JMP を伴う DISPOSE 命令の場合は8クロック)。

## 備考 1. オペランドの凡例

略号	意味
reg1	汎用レジスタ (ソース・レジスタとして使用)
reg2	汎用レジスタ (主にデスティネーション・レジスタとして使用 (一部の命令で, ソース・レジスタとしても使用))
reg3	汎用レジスタ (主に除算結果の余り, 乗算結果の上位32ビットを格納)
bit#3	ビット・ナンバ指定用3ビット・データ
imm x	x ビット・イミディエト・データ
disp x	x ビット・ディスプレースメント・データ
regID	システム・レジスタ番号
vector x	ベクタを指定するデータ (xはビット・サイズをあらわします)
cond	条件名を示します (第2編 表5-4 条件コード一覽参照)
cccc	条件コードを示す4ビット・データ (第2編 表5-4 条件コード一覽参照)
sp	スタック・ポインタ (r3)
ep	エレメント・ポインタ (r30)
list12	レジスタ・リスト

## 2. 実行クロックの凡例

略号	意味
issue	命令実行直後に他の命令を実行する場合
repeat	命令実行直後に同一命令を繰り返す場合
latency	命令実行結果をその命令実行直後の命令で利用する場合

## C. 2.2 浮動小数点演算命令の命令実行クロック数

表C - 2に単精度浮動小数点演算命令の実行クロック数一覧,表C - 3に倍精度浮動小数点演算命令の実行クロック数一覧を示します。なお,命令実行クロック数は,命令の組み合わせにより異なる場合があります。

表C - 2 単精度浮動小数点演算命令実行クロック数一覧 (1/2)

ニモニック	オペランド	バイト	実行クロック数						並列発行 <sup>注</sup>
			インプレサイズ			プレサイズ			
			issue	repeat	latency	issue	repeat	latency	
ABSF.S	reg2, reg3	4	1	1	4	4	4	4	×
ADDF.S	reg1, reg2, reg3	4	1	1	4	4	4	4	×
CEILF.SL	reg2, reg3	4	1	1	4	4	4	4	×
CEILF.SUL	reg2, reg3	4	1	1	4	4	4	4	×
CEILF.SUW	reg2, reg3	4	1	1	4	4	4	4	×
CEILF.SW	reg2, reg3	4	1	1	4	4	4	4	×
CMOVF.S	cc, reg1, reg2, reg3	4	1	1	4	4	4	4	×
CMPF.S	cond, reg1, reg2, cc	4	1	1	4	4	4	4	×
CVTF.LS	reg2, reg3	4	1	1	4	4	4	4	×
CVTF.SL	reg2, reg3	4	1	1	4	4	4	4	×
CVTF.SUL	reg2, reg3	4	1	1	4	4	4	4	×
CVTF.SUW	reg2, reg3	4	1	1	4	4	4	4	×
CVTF.SW	reg2, reg3	4	1	1	4	4	4	4	×
CVTF.ULS	reg2, reg3	4	1	1	4	4	4	4	×
CVTF.UWS	reg2, reg3	4	1	1	4	4	4	4	×
CVTF.WS	reg2, reg3	4	1	1	4	4	4	4	×
DIVF.S	reg1, reg2, reg3	4	14	14	17	17	17	17	×
FLOORF.SL	reg2, reg3	4	1	1	4	4	4	4	×
FLOORF.SUL	reg2, reg3	4	1	1	4	4	4	4	×
FLOORF.SUW	reg2, reg3	4	1	1	4	4	4	4	×
FLOORF.SW	reg2, reg3	4	1	1	4	4	4	4	×
MADDF.S	reg1, reg2, reg3, reg4	4	2	2	5	5	5	5	×
MAXF.S	reg1, reg2, reg3	4	1	1	4	4	4	4	×
MINF.S	reg1, reg2, reg3	4	1	1	4	4	4	4	×
MSUBF.S	reg1, reg2, reg3, reg4	4	2	2	5	5	5	5	×
MULF.S	reg1, reg2, reg3	4	1	1	4	4	4	4	×
NEGF.S	reg2, reg3	4	1	1	4	4	4	4	×
NMADDF.S	reg1, reg2, reg3, reg4	4	2	2	5	5	5	5	×
NMSUBF.S	reg1, reg2, reg3, reg4	4	2	2	5	5	5	5	×
RECIPF.S	reg2, reg3	4	10	10	13	13	13	13	×
RSQRTF.S	reg2, reg3	4	13	13	16	16	16	16	×
SQRTF.S	reg2, reg3	4	14	14	17	17	17	17	×
SUBF.S	reg1, reg2, reg3	4	1	1	4	4	4	4	×

表C - 2 単精度浮動小数点演算命令実行クロック数一覧 (2/2)

二モニック	オペランド	バイト	実行クロック数						並列発行 <sup>注</sup>
			インプレサイズ			プレサイズ			
			issue	repeat	latency	issue	repeat	latency	
TRFSR	cc	4	1	1	1	1	1	1	(R)
TRNCF.SL	reg2, reg3	4	1	1	4	4	4	4	×
TRNCF.SUL	reg2, reg3	4	1	1	4	4	4	4	×
TRNCF.SUW	reg2, reg3	4	1	1	4	4	4	4	×
TRNCF.SW	reg2, reg3	4	1	1	4	4	4	4	×

表C - 3 倍精度浮動小数点演算命令実行クロック数一覧 (1/2)

二モニック	オペランド	バイト	実行クロック数						並列発行 <sup>注</sup>
			インプレサイズ			プレサイズ			
			issue	repeat	latency	issue	repeat	latency	
ABSF.D	reg2, reg3	4	1	1	4	4	4	4	×
ADDF.D	reg1, reg2, reg3	4	1	1	4	4	4	4	×
CEILF.DL	reg2, reg3	4	1	1	4	4	4	4	×
CEILF.DUL	reg2, reg3	4	1	1	4	4	4	4	×
CEILF.DUW	reg2, reg3	4	1	1	4	4	4	4	×
CEILF.DW	reg2, reg3	4	1	1	4	4	4	4	×
CMOVF.D	cc, reg1, reg2, reg3	4	1	1	4	4	4	4	×
CMPF.D	cond, reg1, reg2, cc	4	1	1	4	4	4	4	×
CVTF.DL	reg2, reg3	4	1	1	4	4	4	4	×
CVTF.DS	reg2, reg3	4	1	1	4	4	4	4	×
CVTF.DUL	reg2, reg3	4	1	1	4	4	4	4	×
CVTF.DUW	reg2, reg3	4	1	1	4	4	4	4	×
CVTF.DW	reg2, reg3	4	1	1	4	4	4	4	×
CVTF.LD	reg2, reg3	4	1	1	4	4	4	4	×
CVTF.SD	reg2, reg3	4	1	1	4	4	4	4	×
CVTF.ULD	reg2, reg3	4	1	1	4	4	4	4	×
CVTF.UWD	reg2, reg3	4	1	1	4	4	4	4	×
CVTF.WD	reg2, reg3	4	1	1	4	4	4	4	×
DIVF.D	reg1, reg2, reg3	4	29	29	32	32	32	32	×
FLOORF.DL	reg2, reg3	4	1	1	4	4	4	4	×
FLOORF.DUL	reg2, reg3	4	1	1	4	4	4	4	×
FLOORF.DUW	reg2, reg3	4	1	1	4	4	4	4	×
FLOORF.DW	reg2, reg3	4	1	1	4	4	4	4	×
MAXF.D	reg1, reg2, reg3	4	1	1	4	4	4	4	×
MINF.D	reg1, reg2, reg3	4	1	1	4	4	4	4	×
MULF.D	reg1, reg2, reg3	4	2	2	5	5	5	5	×
NEGF.D	reg2, reg3	4	1	1	4	4	4	4	×
RECIPF.D	reg2, reg3	4	22	22	25	25	25	25	×
RSQRTF.D	reg2, reg3	4	30	30	33	33	33	33	×
SQRTF.D	reg2, reg3	4	29	29	32	32	32	32	×
SUBF.D	reg1, reg2, reg3	4	1	1	4	4	4	4	×



表C - 3 倍精度浮動小数点演算命令実行クロック数一覧 (2/2)

二モニック	オペランド	バイト	実行クロック数						並列発行 <sup>注</sup>
			インプレサイズ			プレサイズ			
			issue	repeat	latency	issue	repeat	latency	
TRNCF.DL	reg2, reg3	4	1	1	4	4	4	4	×
TRNCF.DUL	reg2, reg3	4	1	1	4	4	4	4	×
TRNCF.DUW	reg2, reg3	4	1	1	4	4	4	4	×
TRNCF.DW	reg2, reg3	4	1	1	4	4	4	4	×

注 「 」は、ほかの命令との並列発行が可能であることを、「×」は、ほかの命令との並列発行が不可能であること(単独で発行)を、「 」は、その命令の直前の命令が2バイトまたは4バイト命令である場合に並列発行が可能であることを示します。また、かっこ内は、並列発行を行う際に使用するパイプラインを示します(L: Lpipe, R: Rpipe, R/L: Rpipe または Lpipe)。使用するパイプラインが同一のものは並列発行は不可能です。

備考1. 表C - 2, 表C - 3は変更される可能性があります。

2. 実行クロックの凡例

略号	意味
issue	命令実行直後に他の命令を実行する場合
repeat	命令実行直後に同一命令を繰り返す場合
latency	命令実行結果をその命令実行直後の命令で利用する場合

## C.3 基本命令のパイプライン

### C.3.1 ロード命令

ロード命令は、命令実行パイプライン・レフト（Lpipe）のMEMユニットで実行されます。

[対象の命令] LD.B, LD.H, LD.W, LD.BU, LD.HU, SLD.B, SLD.BU, SLD.H, SLD.HU, SLD.W

[パイプライン]

	<1>	<2>	<3>	<4>	<5>	<6>	<7>	<8>
ロード命令	IF	DP	ID	EX	AT	DF	WB	
次命令		IF	DP	ID	EX	AT	DF	WB

[説明] パイプラインはIF, DP, ID, EX, AT, DF, WBの7ステージです。この図では、Lpipeでロード命令が実行され、Lpipeに次命令が発行された場合の動作を示しています。命令実行パイプライン・ライト（Rpipe）はロード命令と依存関係がないかぎり独立に処理を実行します。ロード命令の直後に、実行結果を使用する命令を配置すると、データの待ち合わせ時間が発生します。

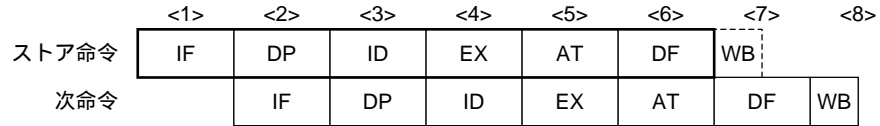
ロード命令は、ほかの命令との並列発行が可能です。

### C. 3.2 ストア命令

ストア命令は、命令実行パイプライン・レフト (Lpipe) のMEMユニットで実行されます。

[対象の命令] ST.B, ST.H, ST.W, SST.B, SST.H, SST.W

[パイプライン]



[説明] パイプラインはIF, DP, ID, EX, AT, DF, WBの7ステージですが、レジスタへのデータの書き込みがないのでWBステージでは何も行いません。

この図では、Lpipeでストア命令が実行され、Lpipeに次命令が発行された場合の動作を示しています。命令実行パイプライン・ライト (Rpipe) はストア命令と依存関係がないかぎり独立に処理を実行します。

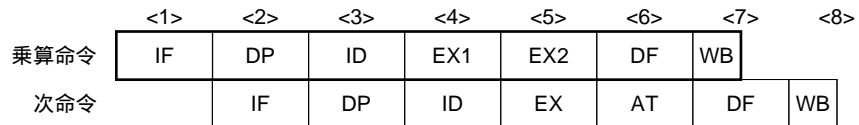
ストア命令は、ほかの命令との並列発行が可能です。

### C. 3.3 乗算命令

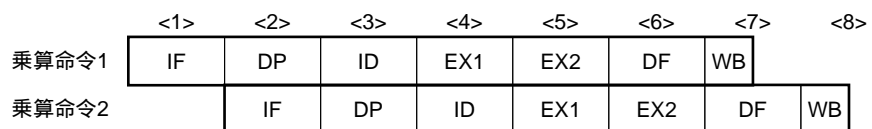
乗算命令は、命令実行パイプライン・レフト (Lpipe) のMULユニットで実行されます。

[対象の命令] MUL, MULH, MULHI, MULL

[パイプライン] (a) 次命令が乗算命令 (または、加算付き乗算命令) 以外の場合



(b) 次命令が乗算命令 (または、加算付き乗算命令) の場合



[説明] パイプラインはIF, DP, ID, EX, AT, DF, WBの7ステージです。EXステージは2クロックかかりますが、EX1とEX2は独立して動作できます。したがって、乗算命令 (または、加算付き乗算命令) を繰り返しても命令実行クロック数は1クロックとなります。

この図では、Lpipeで乗算命令が実行され、Lpipeに次命令が発行された場合の動作を示しています。命令実行パイプライン・ライト (Rpipe) は乗算命令と依存関係がないかぎり独立に処理を実行します。ただし、乗算命令の直後に実行結果を使用する命令を配置すると、データの待ち合わせ時間が発生します。

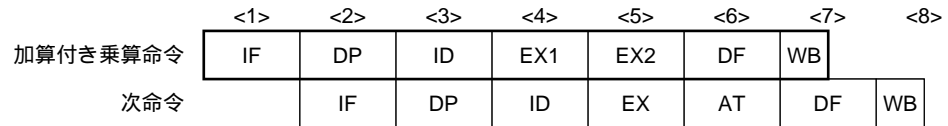
乗算命令は、ほかの命令との並列発行が可能です。

### C. 3.4 加算付き乗算命令

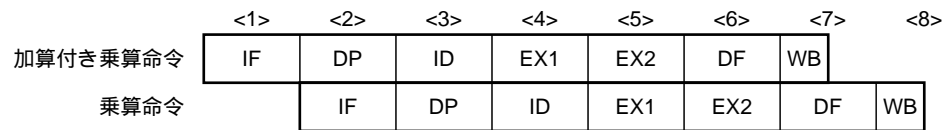
加算付き乗算命令は、命令実行パイプライン・レフト (Lpipe) のMULユニットで実行されます。

[対象の命令] MAC, MACU

[パイプライン] (a) 次命令が乗算命令 (または、加算付き乗算命令) 以外の場合



(b) 次命令が乗算命令 (または、加算付き乗算命令) の場合



[説明] パイプラインはIF, DP, ID, EX, AT, DF, WBの7ステージです。EXステージは2クロックかかりますが、EX1とEX2は独立して動作できます。したがって、乗算命令 (または、加算付き乗算命令) を繰り返しても命令実行クロック数は1クロックとなります。

この図では、Lpipeで乗算命令が実行され、Lpipeに次命令が発行された場合の動作を示しています。命令実行パイプライン・ライト (Rpipe) は乗算命令と依存関係がないかぎり独立に処理を実行します。ただし、乗算命令の直後に実行結果を使用する命令を配置すると、データの待ち合わせ時間が発生します。

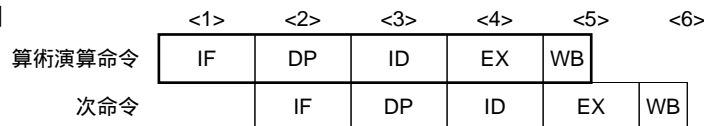
加算付き乗算命令は単独で発行されます。

### C. 3.5 算術演算命令

算術演算命令は、命令実行パイプライン・レフトまたはライト (LpipeまたはRpipe) のALUユニットで実行されます。

[対象の命令] ADD, ADDI, CMP, MOV, MOVEA, MOVHI, SUB, SUBR

[パイプライン]



[説明] パイプラインはIF, DP, ID, EX, WBの5ステージです。

この図では、Rpipeで算術演算命令が実行され、Rpipeに次命令が発行された場合の動作を示しています。Lpipeは算術演算命令と依存関係がないかぎり独立に処理を実行します。

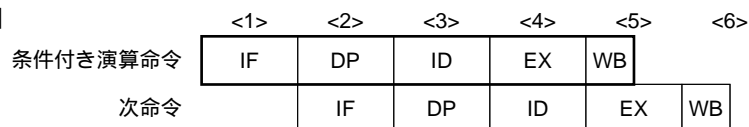
MOV imm32, reg1命令を除く算術演算命令は、ほかの命令との並列発行が可能です (MOV imm32, reg1命令は、単独で発行されます)。

### C. 3.6 条件付き演算命令

条件付き演算命令は、命令実行パイプライン・ライト ( Rpipe ) のALUユニットで実行されます。

[ 対象の命令 ]    ADF, SBF

[ パイプライン ]



[ 説 明 ]        パイプラインはIF, DP, ID, EX, WBの5ステージです。

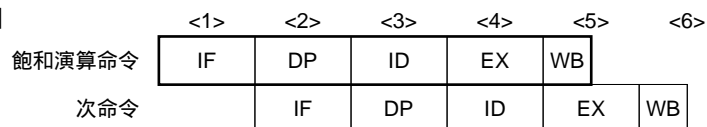
この図では、Rpipeで算術演算命令が実行され、Rpipeに次命令が発行された場合の動作を示しています。条件付き演算命令は単独で発行されます。

### C. 3.7 飽和演算命令

飽和演算命令は、命令実行パイプライン・レフトまたはライト ( LpipeまたはRpipe ) のALUユニットで実行されます。

[ 対象の命令 ]    SATADD, SATSUB, SATSUBI, SATSUBR

[ パイプライン ]



[ 説 明 ]        パイプラインはIF, DP, ID, EX, WBの5ステージです。

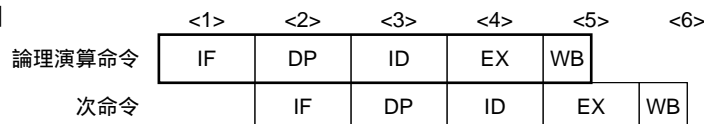
この図では、Rpipeで飽和演算命令が実行され、Rpipeに次命令が発行された場合の動作を示しています。Lpipeは飽和演算命令と依存関係がないかぎり独立に処理を実行します。飽和演算命令はほかの命令との並列発行が可能です。

### C. 3. 8 論理演算命令

論理演算命令は命令実行パイプライン・レフトまたはライト（LpipeまたはRpipe）のALUユニットで実行されます。

[対象の命令] AND, ANDI, NOT, OR, ORI, TST, XOR, XORI

[パイプライン]



[説明] パイプラインはIF, DP, ID, EX, WBの5ステージです。

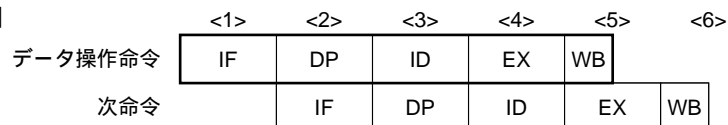
この図では、Rpipeで論理演算命令が実行され、Rpipeに次命令が発行された場合の動作を示しています。Lpipeは論理演算命令と依存関係がないかぎり独立に処理を実行します。論理演算命令は、ほかの命令との並列発行が可能です。

### C. 3. 9 データ操作命令

データ操作命令は、命令実行パイプライン・ライト（Rpipe）のBSFTユニットで実行されます。

[対象の命令] BSH, BSW, CMOV, HSH, HSW, SAR, SASF, SETF, SHL, SHR, SXB, SXH, ZXB, ZXH

[パイプライン]



[説明] パイプラインはIF, DP, ID, EX, WBの5ステージです。

この図では、Rpipeでデータ操作命令が実行され、Rpipeに次命令が発行された場合の動作を示しています。命令実行パイプライン・レフト（Lpipe）はデータ操作命令と依存関係がないかぎり独立に処理を実行します。

データ操作命令のうちSXB, SXH, ZXB, ZXHは単独で発行されます。

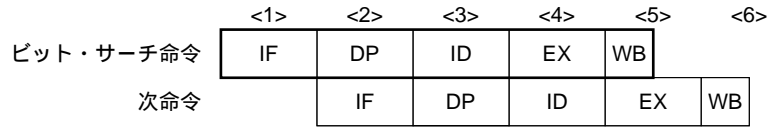
それ以外のデータ操作命令は、ほかの命令との並列発行が可能です。

### C. 3. 10 ビット・サーチ命令

ビット・サーチ命令は、命令実行パイプライン・ライト ( Rpipe ) のBSFTユニットで実行されます。

[ 対象の命令 ] SCH0L, SCH0R, SCH1L, SCH1R

[ パイプライン ]



[ 説明 ] パイプラインはIF, DP, ID, EX, WBの5ステージです。

この図では、Rpipeでデータ操作命令が実行され、Rpipeに次命令が発行された場合の動作を示しています。命令実行パイプライン・レフト ( Lpipe ) はデータ操作命令と依存関係がないかぎり独立に処理を実行します。

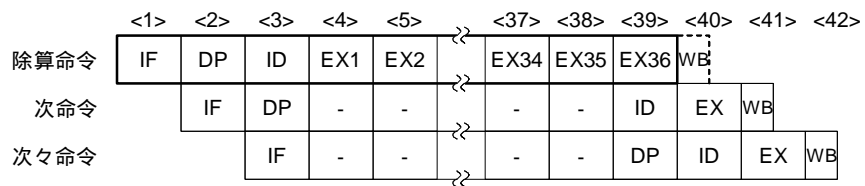
ビット・サーチ命令は、ほかの命令との並列発行が可能です。

### C. 3. 11 除算命令

除算命令は、命令実行パイプライン・ライト ( Rpipe ) のALUユニットで実行されます。

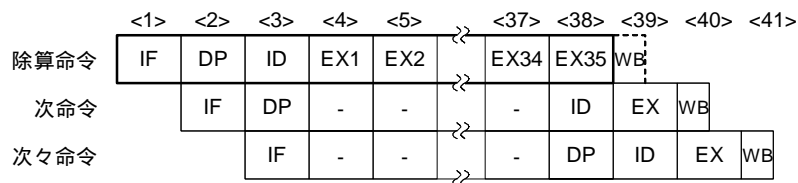
[ 対象の命令 ] DIV, DIVH, DIVHU, DIVU

[ パイプライン ] ( a ) DIV, DIVHの場合



- : 待ちあわせのために挿入されるアイドル

( b ) DIVU, DIVHUの場合



- : 待ちあわせのために挿入されるアイドル

[ 説明 ] パイプラインはDIV, DIVH命令の場合、IF, DP, ID, EX1-EX36, WBの40ステージ、DIVU, DIVHU命令の場合、IF, DP, ID, EX1-EX35, WBの39ステージです。

この図では、Rpipeで除算命令が実行され、Rpipeに次命令が発行された場合の動作を示しています。

ただし、除算命令がIDステージで命令をデコードしている期間とEXステージで命令を実行している期間は、ディスパッチ・ユニットは命令を発行しません。

除算命令は単独で発行されます。

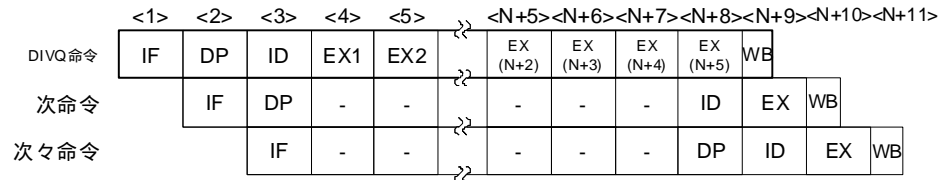
### C. 3. 12 高速除算命令

高速除算命令は、命令実行パイプライン・ライト ( Rpipe ) のALUユニットで実行されます。

この命令では、演算に必要な最小ステップ数を自動的に決定します。

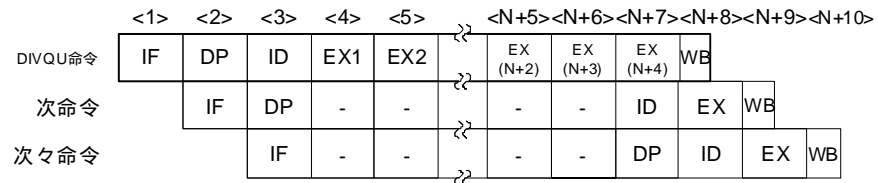
[ 対象の命令 ] DIVQ, DIVQU

[ パイプライン ] ( a ) DIVQの場合



- : 待ち合わせのために挿入されるアイドル

( b ) DIVQUの場合



- : 待ち合わせのために挿入されるアイドル

[ 説 明 ] パイプラインは、DIVQ命令の場合、IF, DP, ID, EX1-EX (N+5), WBのN+9ステージ、DIVQU命令の場合、IF, DP, ID, EX1-EX (N+4), WBのN+8ステージです。

この図では、Rpipeで高速除算命令が実行され、Rpipeに次命令が発行された場合の動作を示しています。

ただし、除算命令がIDステージで命令をデコードしている期間とEXステージで命令を実行している期間は、ディスパッチ・ユニットは命令を発行しません。

各命令は単独で発行されます。

**備考** N = ( 被除数の有効ビット数 ) - ( 除数の有効ビット数 ) です。

ただし、Nがマイナスとなった場合、N = 0として取り扱います。



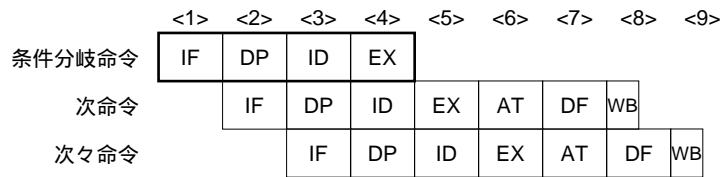
### C. 3. 13 分岐命令

分岐命令は、命令実行パイプライン・レフトまたはライト（LpipeまたはRpipe）のALUユニットで実行されます。

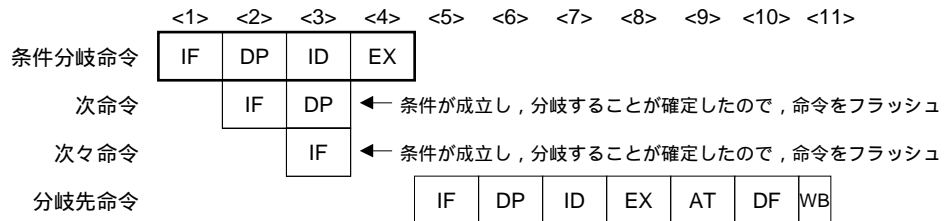
#### (1) 条件分岐命令（BR命令を除く）

[ 対象の命令 ] Bcond 命令

[ パイプライン ] (a) 条件が成立しない場合



(b) 条件が成立した場合

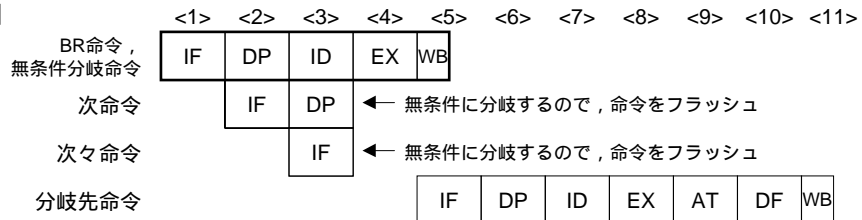


[ 説明 ] この図では、命令実行パイプライン・ライト（Rpipe）で Bcond 命令が実行された場合の動作を示しています。  
分岐命令は、ほかの命令との並列発行が可能です。

#### (2) BR命令、無条件分岐命令（JMP命令を除く）

[ 対象の命令 ] BR, JARL, JR 命令

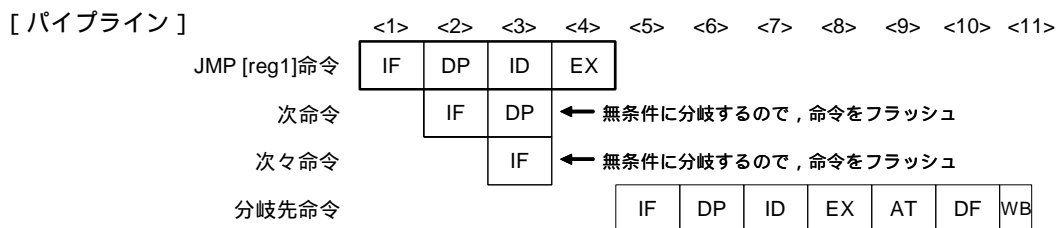
[ パイプライン ]



[ 説明 ] この図では、命令実行パイプライン・ライト（Rpipe）で Bcond 命令が実行され、命令実行パイプライン・レフト（Lpipe）ですべての命令が実行された場合の動作を示しています。  
BR 命令、無条件分岐命令は、ほかの命令との並列発行が可能です。

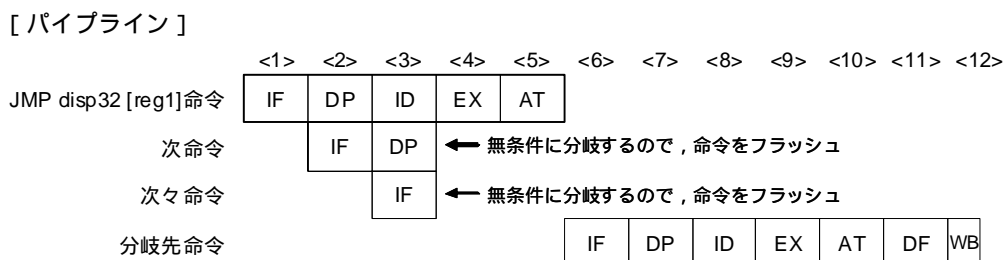
## (3) JMP命令

## (a) JMP [reg1]命令



- [説明] この図では、命令実行パイプライン・ライト (Rpipe) で JMP 命令が実行され、命令実行パイプライン・レフト (Lpipe) ですべての命令が実行された場合の動作を示しています。  
この命令は、ほかの命令との並列発行が可能です。

## (b) JMP disp32 [reg1]命令



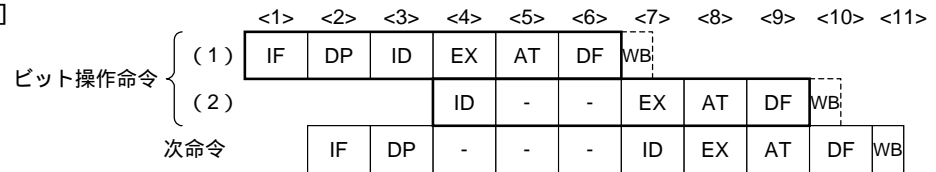
- [説明] この図では、命令実行パイプライン・ライト (Rpipe) で JMP 命令が実行され、命令実行パイプライン・レフト (Lpipe) ですべての命令が実行された場合の動作を示しています。  
この命令は、ほかの命令との並列発行が可能です。

### C. 3. 14 ビット操作命令

ビット操作命令は、命令実行パイプライン・レフト (Lpipe) のALUユニットで実行されます。

#### (1) CLR1, NOT1, SET1命令

[パイプライン]

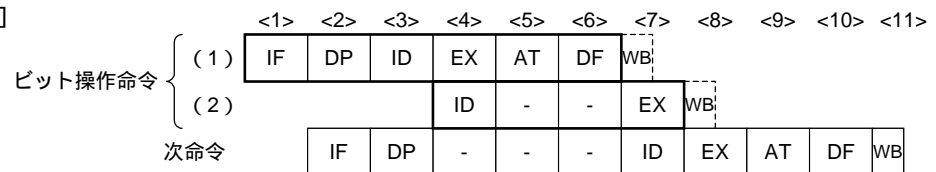


- : 待ち合わせのために挿入されるアイドル

[説明] ID ステージで 2 つの命令に分割され、最初にロード命令を、次にビット操作を含むストア命令を実行します。ただし、レジスタへのデータの書き込みがないので、WB ステージでは何も行いません。この図では、Lpipe でビット操作命令が実行され、Lpipe に次命令が発行された場合の動作を示しています。ID ステージで命令をデコードしている期間はディスパッチ・ユニットは Lpipe には命令を発行しません。ビット操作命令は単独で発行されます。

#### (2) TST1命令

[パイプライン]



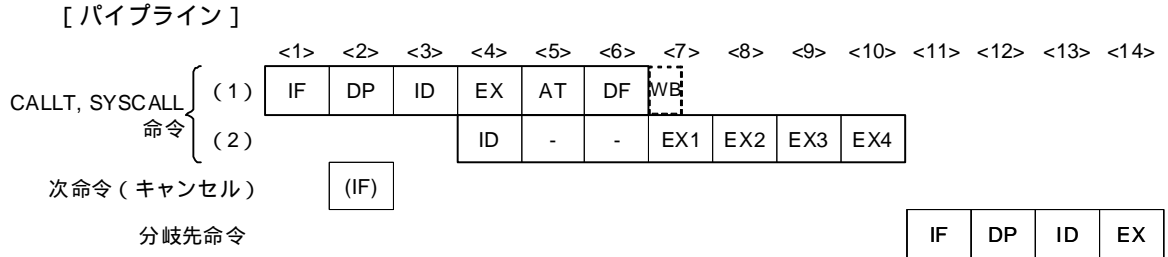
- : 待ち合わせのために挿入されるアイドル

[説明] ID ステージで 2 つの命令に分割され、最初にロード命令を、次にビット操作命令を実行します。ただし、レジスタへのデータの書き込みがないので、WB ステージでは何も行いません。この図では、Lpipe で TST1 命令が実行され、Lpipe に次命令が発行された場合の動作を示しています。ID ステージで命令をデコードしている期間はディスパッチ・ユニットは Lpipe には命令を発行しません。TST1 命令は単独で発行されます。

### C. 3. 15 特殊命令

#### (1) CALLT, SYSCALL命令

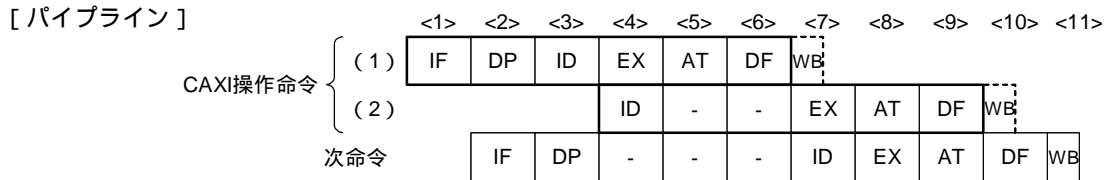
CALLT, SYSCALL命令は、命令実行パイプライン・レフト (Lpipe) のALUユニットで実行されます。



- : 待ち合わせのために挿入されるアイドル  
 (IF) : 無効となる命令フェッチ

[説明] ID ステージで 2 つの命令に分割され、最初にロード命令を、次に CTBP または、SCBP 相対の分岐命令を実行します。ただし、レジスタへのデータの書き込みがないので、WB ステージでは何も行いません。この図では、Lpipe で CALLT, SYSCALL 命令が実行され、分岐先から命令をフェッチして実行するまでの動作を示しています。この命令は単独で発行されます。

#### (2) CAXI命令



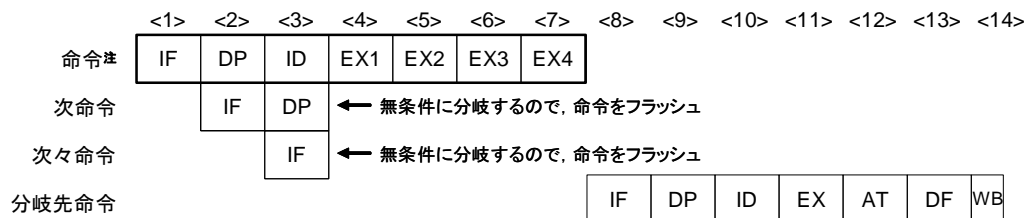
- : 待ち合わせのために挿入されるアイドル

[説明] ID ステージで 2 つの命令に分割され、最初にロード命令を、次にストア命令を実行します。この図では、Lpipe で CAXI 命令が実行され、Lpipe に次命令が発行された場合の動作を示しています。ID ステージで命令をデコードしている期間はディスパッチ・ユニットは Lpipe には命令を発行しません。この各命令は単独で発行されます。

## (3) CTRET, EIRET, FERET, FETRAP, RETI, RIE, TRAP命令

CTRET, EIRET, FERET, FETRAP, RETI, RIE, TRAP 命令は、命令実行パイプライン・ライト (Rpipe) で実行されます。

## [パイプライン]



注 CTRET, EIRET, FERET, FETRAP, RETI, RIE, TRAP命令

[説明] この図では、命令実行パイプライン・ライト (Rpipe) で各命令が実行され、Lpipe ですべての命令が発行された場合の動作を示しています。

CTRET, EIRET, FERET, FETRAP, RETI, RIE, TRAP 命令は単独で発行されます。

## (4) DI, EI, LDSR命令

DI, EI, LDSR命令は、命令実行パイプライン・ライト (Rpipe) のALUユニットで実行されます。

## [パイプライン]



[説明] パイプラインはIF, DP, ID, EX, WBの5ステージです。

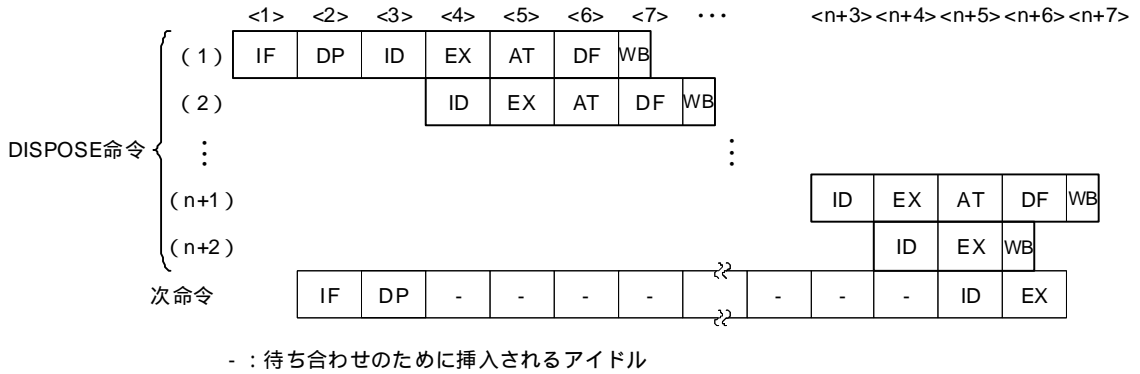
この図では、RpipeでDI, EI, LDSR命令が実行され、Rpipeですべての命令が実行された場合の動作を示しています。

DI, EI, LDSR命令は単独で発行されます。

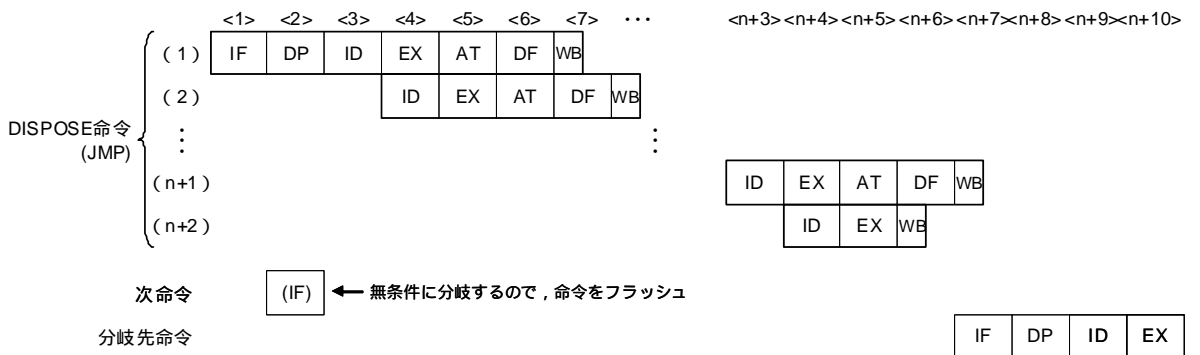
(5) DISPOSE命令

DISPOSE命令は、命令実行パイプライン・レフト (Lpipe) のALUユニットで実行されます。

[パイプライン] (a) 分岐しない場合



(b) 分岐する場合



**備考** nは、レジスタ・リスト (list12) で指定されるレジスタの数です。

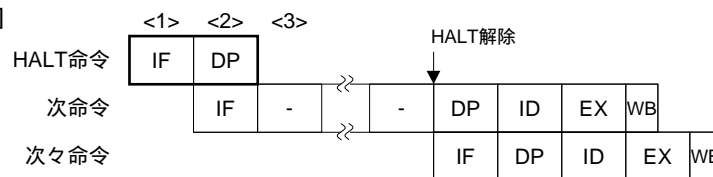
[説明] ID ステージで n+2 個の命令に分割され、最初に n 個のロード命令を、最後にスタック・ポインタ (SP) への書き込み命令を実行します。この図では、Lpipe で DISPOSE 命令が実行され、Lpipe に次命令が発行された場合の動作を示しています。命令実行パイプライン・ライト (Rpipe) は DISPOSE 命令と依存関係がないかぎり独立に処理を実行します。

この命令は単独で発行されます。

## (6) HALT命令

HALT命令は、命令実行パイプライン・ライト (Rpipe) で実行されます。

[パイプライン]



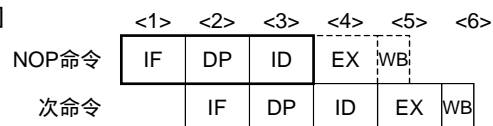
- : 待ち合わせのために挿入されるアイドル

[説明] DP ステージで HALT 命令が検出されると、HALT 命令が解除されるまで、ID ステージへの命令の発行を停止します。したがって、次命令では HALT 命令が解除されるまで ID ステージが遅れます。この図では、命令実行パイプライン・ライト (Rpipe) で HALT 命令が実行され、Rpipe に次命令が発行された場合の動作を示しています。この命令は単独で発行されます。

## (7) NOP命令

NOP命令は、命令実行パイプライン・ライト (Rpipe) のALUユニットで実行されます。

[パイプライン]

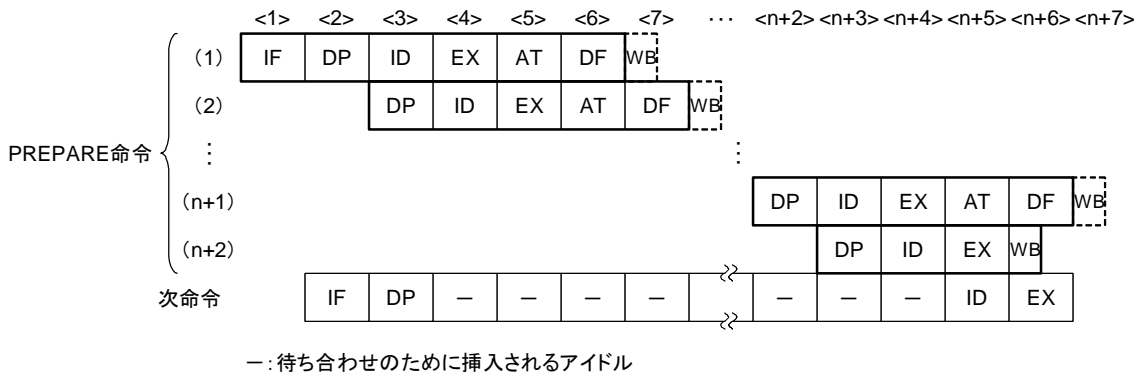


[説明] パイプラインは IF, DP, ID, EX, WB の 5 ステージですが、演算、レジスタへのデータの書き込みがないので、EX, WB ステージでは何も行いません。この命令は、単独で発行されます。

(8) PREPARE命令

PREPARE命令は、命令実行パイプライン・レフト (Lpipe) のALUユニットで実行されます。

[パイプライン]



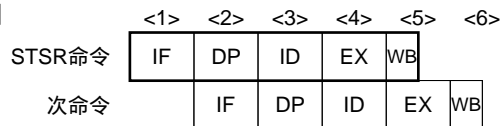
備考 n は、レジスタ・リスト (list12) で指定されるレジスタの数です。

[説明] ID ステージで n+2 個の命令に分割され、最初に n 個のストア命令を、最後にスタック・ポインタ (SP) への書き込み命令を実行します。ただし、ストア命令ではレジスタへのデータの書き込みがないので、WB ステージでは何も行いません。この図では、Lpipe で PREPARE 命令が実行され、Lpipe に次命令が発行された場合の動作を示しています。この命令は単独で発行されます。

(9) STSR命令

STSR命令は、命令実行パイプライン・ライト (Rpipe) のALUユニットで実行されます。

[パイプライン]



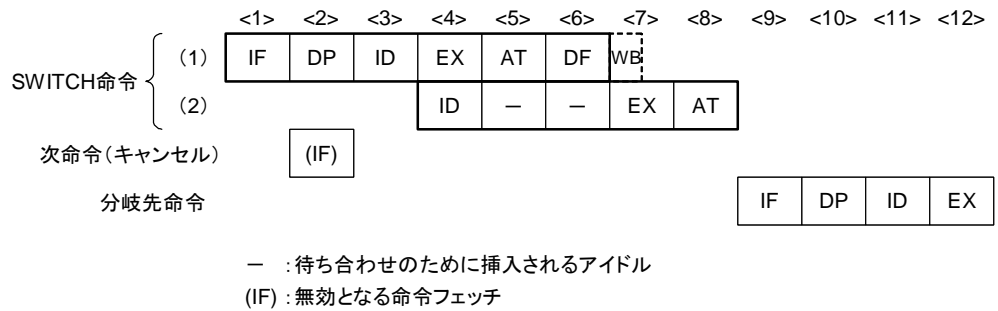
[説明] パイプラインは IF, DP, ID, EX, WB の 5 ステージです。この図では、Rpipe で STSR 命令が実行され、Rpipe に次命令が発行された場合の動作を示しています。命令実行パイプライン・レフト (Lpipe) は STSR 命令と依存関係がないかぎり独立に処理を実行します。この命令は単独で発行されます。



(10) SWITCH命令

SWITCH命令は、命令実行パイプライン・レフト (Lpipe) のALUユニットで実行されます。

[パイプライン]



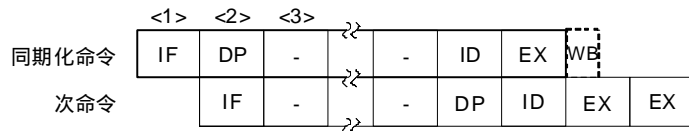
[説明] ID ステージで 2 つの命令に分割され、最初にロード命令を、次に PC 相対の分岐命令を実行します。ただし、レジスタへのデータの書き込みがないので、WB ステージでは何も行いません。この図では、Lpipe で SWITCH 命令が実行され、Lpipe に次命令が発行された場合の動作を示しています。命令実行パイプライン・ライト (Rpipe) は SWITCH 命令と依存関係がないかぎり独立に処理を実行します。この命令は単独で発行されます。

(11) 同期化命令

同期化命令は、命令実行パイプライン・ライト (Rpipe) のALUユニットで実行されます。

[対象の命令] SYNCE, SYNCM, SYNCP

[パイプライン]



[説明] この図では、Rpipeで同期化命令が実行され、命令実行パイプライン・レフト (Lpipe) ですべての命令が発行された場合の動作を示しています。この命令は単独で発行されます。CPUで保留されているすべての命令の処理が完了するまで同期化命令の発行を行いません。

## 付録D 周辺装置保護領域一覧

V850E2M CPUの周辺装置保護設定レジスタ（PPC0-PPC8）の各ビットと、周辺装置の対応を示します。

V850E2M CPUでは、PPC0レジスタ・セット内の括弧で囲んだ領域に対応するビットPPx2, PPx4-6, PPx12-15, PPx18, PPx19は0固定です。また、システム保護機能（周辺装置保護、タイミング監視）の設定レジスタが配置されているPPx16と予約領域のPPx17はOS周辺装置として1固定です。

### PPC0 (PPS0, PPP0, PPV0, PPT0)

ビット名	領域名	アドレス範囲	備考
PPx0	INTC	FFFF6000-FFFF645FH	INTC用
PPx1	Fcache	FFFF6480-FFFF6487H	FlashCache用
PPx2	(R.F.U)	R.F.U	予約
PPx3	SEG	FFFF64B0-FFFF64B3H	システム・エラー用
PPx4	(R.F.U)	R.F.U	予約
PPx5	(R.F.U)	FFFF6500-FFFF65FFH	予約
PPx6	(R.F.U)	FFFF6600-FFFF66FFH	予約
PPx7	EXTBRK	FFFF6700-FFFF67FFH	デバッグ機能用
PPx8	MIR	FFFF6800-FFFF687FH	PE間割り込み用
PPx9	MEV	FFFF6900-FFFF697FH	MEV用
PPx10	MEC	FFFF6980-FFFF699FH	MEC用
PPx11	PEG	FFFF69A0-FFFF69BFH	PEガード用
PPx12	(R.F.U)	FFFF6A00-FFFF6AFFH	予約
PPx13	(R.F.U)	FFFF6B00-FFFF6BFFH	予約
PPx14	(R.F.U)	FFFF6C00-FFFF6DFFH	予約
PPx15	(R.F.U)	FFFF6E00-FFFF6EFFH	予約
PPx16	SPF	FFFF5000-FFFF53FFH	TSU/PPU
PPx17	(R.F.U)	FFFF5400-FFFF57FFH	予約
PPx18	(R.F.U)	FFFF5800-FFFF5BFFH	予約
PPx19	(R.F.U)	FFFF5C00-FFFF5FFFH	予約
PPx20	EXT20	FFFF7000-FFFF70FFH	拡張領域
PPx21	EXT21	FFFF7100-FFFF71FFH	拡張領域
PPx22	EXT22	FFFF7200-FFFF72FFH	拡張領域
PPx23	EXT23	FFFF7300-FFFF73FFH	拡張領域
PPx24	EXT24	FFFF7400-FFFF74FFH	拡張領域
PPx25	EXT25	FFFF7500-FFFF76FFH	拡張領域
PPx26	EXT26	FFFF7700-FFFF78FFH	拡張領域
PPx27	EXT27	FFFF7900-FFFF7AFFH	拡張領域
PPx28	EXT28	FFFF7B00-FFFF7CFFH	拡張領域
PPx29	EXT29	FFFF7D00-FFFF7EFFH	拡張領域
PPx30	EXT30	FFFF7F00-FFFF7FFFH	拡張領域
PPx31	EXT31	FFFF7F80-FFFF7FFFH	拡張領域

## PPC1 (PPS1, PPP1, PPV1, PPT1)

ビット名	アドレス範囲	備考	ビット名	アドレス範囲	備考
PPx0	FF400000-FF40FFFFH	64 Kバイト	PPx16	FF500000-FF50FFFFH	64 Kバイト
PPx1	FF410000-FF41FFFFH	64 Kバイト	PPx17	FF510000-FF51FFFFH	64 Kバイト
PPx2	FF420000-FF42FFFFH	64 Kバイト	PPx18	FF520000-FF52FFFFH	64 Kバイト
PPx3	FF430000-FF43FFFFH	64 Kバイト	PPx19	FF530000-FF53FFFFH	64 Kバイト
PPx4	FF440000-FF44FFFFH	64 Kバイト	PPx20	FF540000-FF54FFFFH	64 Kバイト
PPx5	FF450000-FF45FFFFH	64 Kバイト	PPx21	FF550000-FF55FFFFH	64 Kバイト
PPx6	FF460000-FF46FFFFH	64 Kバイト	PPx22	FF560000-FF56FFFFH	64 Kバイト
PPx7	FF470000-FF47FFFFH	64 Kバイト	PPx23	FF570000-FF57FFFFH	64 Kバイト
PPx8	FF480000-FF48FFFFH	64 Kバイト	PPx24	FF580000-FF58FFFFH	64 Kバイト
PPx9	FF490000-FF49FFFFH	64 Kバイト	PPx25	FF590000-FF59FFFFH	64 Kバイト
PPx10	FF4A0000-FF4AFFFFH	64 Kバイト	PPx26	FF5A0000-FF5AFFFFH	64 Kバイト
PPx11	FF4B0000-FF4BFFFFH	64 Kバイト	PPx27	FF5B0000-FF5BFFFFH	64 Kバイト
PPx12	FF4C0000-FF4CFFFFH	64 Kバイト	PPx28	FF5C0000-FF5CFFFFH	64 Kバイト
PPx13	FF4D0000-FF4DFFFFH	64 Kバイト	PPx29	FF5D0000-FF5DFFFFH	64 Kバイト
PPx14	FF4E0000-FF4EFFFFH	64 Kバイト	PPx30	FF5E0000-FF5EFFFFH	64 Kバイト
PPx15	FF4F0000-FF4FFFFFH	64 Kバイト	PPx31	FF5F0000-FF5FFFFFH	64 Kバイト

## PPC2 (PPS2, PPP2, PPV2, PPT2)

ビット名	アドレス範囲	備考	ビット名	アドレス範囲	備考
PPx0	FF600000-FF60FFFFH	64 Kバイト	PPx16	FF700000-FF70FFFFH	64 Kバイト
PPx1	FF610000-FF61FFFFH	64 Kバイト	PPx17	FF710000-FF71FFFFH	64 Kバイト
PPx2	FF620000-FF62FFFFH	64 Kバイト	PPx18	FF720000-FF72FFFFH	64 Kバイト
PPx3	FF630000-FF63FFFFH	64 Kバイト	PPx19	FF730000-FF73FFFFH	64 Kバイト
PPx4	FF640000-FF64FFFFH	64 Kバイト	PPx20	FF740000-FF74FFFFH	64 Kバイト
PPx5	FF650000-FF65FFFFH	64 Kバイト	PPx21	FF750000-FF75FFFFH	64 Kバイト
PPx6	FF660000-FF66FFFFH	64 Kバイト	PPx22	FF760000-FF76FFFFH	64 Kバイト
PPx7	FF670000-FF67FFFFH	64 Kバイト	PPx23	FF770000-FF77FFFFH	64 Kバイト
PPx8	FF680000-FF68FFFFH	64 Kバイト	PPx24	FF780000-FF78FFFFH	64 Kバイト
PPx9	FF690000-FF69FFFFH	64 Kバイト	PPx25	FF790000-FF79FFFFH	64 Kバイト
PPx10	FF6A0000-FF6AFFFFH	64 Kバイト	PPx26	FF7A0000-FF7AFFFFH	64 Kバイト
PPx11	FF6B0000-FF6BFFFFH	64 Kバイト	PPx27	FF7B0000-FF7BFFFFH	64 Kバイト
PPx12	FF6C0000-FF6CFFFFH	64 Kバイト	PPx28	FF7C0000-FF7CFFFFH	64 Kバイト
PPx13	FF6D0000-FF6DFFFFH	64 Kバイト	PPx29	FF7D0000-FF7DFFFFH	64 Kバイト
PPx14	FF6E0000-FF6EFFFFH	64 Kバイト	PPx30	FF7E0000-FF7EFFFFH	64 Kバイト
PPx15	FF6F0000-FF6FFFFFH	64 Kバイト	PPx31	FF7F0000-FF7FFFFFH	64 Kバイト

## PPC3 (PPS3, PPP3, PPV3, PPT3)

ビット名	アドレス範囲	備考	ビット名	アドレス範囲	備考
PPx0	FF800000-FF800FFFH	4 Kバイト	PPx16	FF810000-FF810FFFH	4 Kバイト
PPx1	FF801000-FF801FFFH	4 Kバイト	PPx17	FF811000-FF811FFFH	4 Kバイト
PPx2	FF802000-FF802FFFH	4 Kバイト	PPx18	FF812000-FF812FFFH	4 Kバイト
PPx3	FF803000-FF803FFFH	4 Kバイト	PPx19	FF813000-FF813FFFH	4 Kバイト
PPx4	FF804000-FF804FFFH	4 Kバイト	PPx20	FF814000-FF814FFFH	4 Kバイト
PPx5	FF805000-FF805FFFH	4 Kバイト	PPx21	FF815000-FF815FFFH	4 Kバイト
PPx6	FF806000-FF806FFFH	4 Kバイト	PPx22	FF816000-FF816FFFH	4 Kバイト
PPx7	FF807000-FF807FFFH	4 Kバイト	PPx23	FF817000-FF817FFFH	4 Kバイト
PPx8	FF808000-FF808FFFH	4 Kバイト	PPx24	FF818000-FF818FFFH	4 Kバイト
PPx9	FF809000-FF809FFFH	4 Kバイト	PPx25	FF819000-FF819FFFH	4 Kバイト
PPx10	FF80A000-FF80AFFFH	4 Kバイト	PPx26	FF81A000-FF81AFFFH	4 Kバイト
PPx11	FF80B000-FF80BFFFH	4 Kバイト	PPx27	FF81B000-FF81BFFFH	4 Kバイト
PPx12	FF80C000-FF80CFFFH	4 Kバイト	PPx28	FF81C000-FF81CFFFH	4 Kバイト
PPx13	FF80D000-FF80DFFFH	4 Kバイト	PPx29	FF81D000-FF81DFFFH	4 Kバイト
PPx14	FF80E000-FF80EFFFH	4 Kバイト	PPx30	FF81E000-FF81EFFFH	4 Kバイト
PPx15	FF80F000-FF80FFFFH	4 Kバイト	PPx31	FF81F000-FF81FFFFH	4 Kバイト

## PPC4 (PPS4, PPP4, PPV4, PPT4)

ビット名	アドレス範囲	備考	ビット名	アドレス範囲	備考
PPx0	FF820000-FF820FFFH	4 Kバイト	PPx16	FF830000-FF830FFFH	4 Kバイト
PPx1	FF821000-FF821FFFH	4 Kバイト	PPx17	FF831000-FF831FFFH	4 Kバイト
PPx2	FF822000-FF822FFFH	4 Kバイト	PPx18	FF832000-FF832FFFH	4 Kバイト
PPx3	FF823000-FF823FFFH	4 Kバイト	PPx19	FF833000-FF833FFFH	4 Kバイト
PPx4	FF824000-FF824FFFH	4 Kバイト	PPx20	FF834000-FF834FFFH	4 Kバイト
PPx5	FF825000-FF825FFFH	4 Kバイト	PPx21	FF835000-FF835FFFH	4 Kバイト
PPx6	FF826000-FF826FFFH	4 Kバイト	PPx22	FF836000-FF836FFFH	4 Kバイト
PPx7	FF827000-FF827FFFH	4 Kバイト	PPx23	FF837000-FF837FFFH	4 Kバイト
PPx8	FF828000-FF828FFFH	4 Kバイト	PPx24	FF838000-FF838FFFH	4 Kバイト
PPx9	FF829000-FF829FFFH	4 Kバイト	PPx25	FF839000-FF839FFFH	4 Kバイト
PPx10	FF82A000-FF82AFFFH	4 Kバイト	PPx26	FF83A000-FF83AFFFH	4 Kバイト
PPx11	FF82B000-FF82BFFFH	4 Kバイト	PPx27	FF83B000-FF83BFFFH	4 Kバイト
PPx12	FF82C000-FF82CFFFH	4 Kバイト	PPx28	FF83C000-FF83CFFFH	4 Kバイト
PPx13	FF82D000-FF82DFFFH	4 Kバイト	PPx29	FF83D000-FF83DFFFH	4 Kバイト
PPx14	FF82E000-FF82EFFFH	4 Kバイト	PPx30	FF83E000-FF83EFFFH	4 Kバイト
PPx15	FF82F000-FF82FFFFH	4 Kバイト	PPx31	FF83F000-FF83FFFFH	4 Kバイト

## PPC5 (PPS5, PPP5, PPV5, PPT5)

ビット名	アドレス範囲	備考	ビット名	アドレス範囲	備考
PPx0	FFFF8000-FFFF80FFH	256バイト	PPx16	FFFF9000-FFFF90FFH	256バイト
PPx1	FFFF8100-FFFF81FFH	256バイト	PPx17	FFFF9100-FFFF91FFH	256バイト
PPx2	FFFF8200-FFFF82FFH	256バイト	PPx18	FFFF9200-FFFF92FFH	256バイト
PPx3	FFFF8300-FFFF83FFH	256バイト	PPx19	FFFF9300-FFFF93FFH	256バイト
PPx4	FFFF8400-FFFF84FFH	256バイト	PPx20	FFFF9400-FFFF94FFH	256バイト
PPx5	FFFF8500-FFFF85FFH	256バイト	PPx21	FFFF9500-FFFF95FFH	256バイト
PPx6	FFFF8600-FFFF86FFH	256バイト	PPx22	FFFF9600-FFFF96FFH	256バイト
PPx7	FFFF8700-FFFF87FFH	256バイト	PPx23	FFFF9700-FFFF97FFH	256バイト
PPx8	FFFF8800-FFFF88FFH	256バイト	PPx24	FFFF9800-FFFF98FFH	256バイト
PPx9	FFFF8900-FFFF89FFH	256バイト	PPx25	FFFF9900-FFFF99FFH	256バイト
PPx10	FFFF8A00-FFFF8AFFH	256バイト	PPx26	FFFF9A00-FFFF9AFFH	256バイト
PPx11	FFFF8B00-FFFF8BFFH	256バイト	PPx27	FFFF9B00-FFFF9BFFH	256バイト
PPx12	FFFF8C00-FFFF8CFFH	256バイト	PPx28	FFFF9C00-FFFF9CFFH	256バイト
PPx13	FFFF8D00-FFFF8DFFH	256バイト	PPx29	FFFF9D00-FFFF9DFFH	256バイト
PPx14	FFFF8E00-FFFF8EFFH	256バイト	PPx30	FFFF9E00-FFFF9EFFH	256バイト
PPx15	FFFF8F00-FFFF8FFFH	256バイト	PPx31	FFFF9F00-FFFF9FFFH	256バイト

## PPC6 (PPS6, PPP6, PPV6, PPT6)

ビット名	アドレス範囲	備考	ビット名	アドレス範囲	備考
PPx0	FFFFA000-FFFFA0FFH	256バイト	PPx16	FFFFB000-FFFFB0FFH	256バイト
PPx1	FFFFA100-FFFFA1FFH	256バイト	PPx17	FFFFB100-FFFFB1FFH	256バイト
PPx2	FFFFA200-FFFFA2FFH	256バイト	PPx18	FFFFB200-FFFFB2FFH	256バイト
PPx3	FFFFA300-FFFFA3FFH	256バイト	PPx19	FFFFB300-FFFFB3FFH	256バイト
PPx4	FFFFA400-FFFFA4FFH	256バイト	PPx20	FFFFB400-FFFFB4FFH	256バイト
PPx5	FFFFA500-FFFFA5FFH	256バイト	PPx21	FFFFB500-FFFFB5FFH	256バイト
PPx6	FFFFA600-FFFFA6FFH	256バイト	PPx22	FFFFB600-FFFFB6FFH	256バイト
PPx7	FFFFA700-FFFFA7FFH	256バイト	PPx23	FFFFB700-FFFFB7FFH	256バイト
PPx8	FFFFA800-FFFFA8FFH	256バイト	PPx24	FFFFB800-FFFFB8FFH	256バイト
PPx9	FFFFA900-FFFFA9FFH	256バイト	PPx25	FFFFB900-FFFFB9FFH	256バイト
PPx10	FFFFAA00-FFFFAAFFH	256バイト	PPx26	FFFFBA00-FFFFBAFFH	256バイト
PPx11	FFFFAB00-FFFFABFFH	256バイト	PPx27	FFFFBB00-FFFFBBFFH	256バイト
PPx12	FFFFAC00-FFFFACFFH	256バイト	PPx28	FFFFBC00-FFFFBCFFH	256バイト
PPx13	FFFFAD00-FFFFADFFH	256バイト	PPx29	FFFFBD00-FFFFBDFFH	256バイト
PPx14	FFFFAE00-FFFFAEFFH	256バイト	PPx30	FFFFBE00-FFFFBEFFH	256バイト
PPx15	FFFFAF00-FFFFAFFFH	256バイト	PPx31	FFFFBF00-FFFFBFFFH	256バイト

## PPC7 (PPS7, PPP7, PPV7, PPT7)

ビット名	アドレス範囲	備考	ビット名	アドレス範囲	備考
PPx0	FFFFC000-FFFFC0FFH	256バイト	PPx16	FFFFD000-FFFFD0FFH	256バイト
PPx1	FFFFC100-FFFFC1FFH	256バイト	PPx17	FFFFD100-FFFFD1FFH	256バイト
PPx2	FFFFC200-FFFFC2FFH	256バイト	PPx18	FFFFD200-FFFFD2FFH	256バイト
PPx3	FFFFC300-FFFFC3FFH	256バイト	PPx19	FFFFD300-FFFFD3FFH	256バイト
PPx4	FFFFC400-FFFFC4FFH	256バイト	PPx20	FFFFD400-FFFFD4FFH	256バイト
PPx5	FFFFC500-FFFFC5FFH	256バイト	PPx21	FFFFD500-FFFFD5FFH	256バイト
PPx6	FFFFC600-FFFFC6FFH	256バイト	PPx22	FFFFD600-FFFFD6FFH	256バイト
PPx7	FFFFC700-FFFFC7FFH	256バイト	PPx23	FFFFD700-FFFFD7FFH	256バイト
PPx8	FFFFC800-FFFFC8FFH	256バイト	PPx24	FFFFD800-FFFFD8FFH	256バイト
PPx9	FFFFC900-FFFFC9FFH	256バイト	PPx25	FFFFD900-FFFFD9FFH	256バイト
PPx10	FFFFCA00-FFFFCAFFH	256バイト	PPx26	FFFFDA00-FFFFDAFFH	256バイト
PPx11	FFFFCB00-FFFFCBFFH	256バイト	PPx27	FFFFDB00-FFFFDBFFH	256バイト
PPx12	FFFFCC00-FFFFCCFFH	256バイト	PPx28	FFFFDC00-FFFFDCFFH	256バイト
PPx13	FFFFCD00-FFFFCDFFH	256バイト	PPx29	FFFFDD00-FFFFDDFFH	256バイト
PPx14	FFFFCE00-FFFFCEFFH	256バイト	PPx30	FFFFDE00-FFFFDEFFH	256バイト
PPx15	FFFFCF00-FFFFCFFFH	256バイト	PPx31	FFFFDF00-FFFFDFFFH	256バイト

## PPC8 (PPS8, PPP8, PPV8, PPT8)

ビット名	アドレス範囲	備考	ビット名	アドレス範囲	備考
PPx0	FFFFE000-FFFFE0FFH	256バイト	PPx16	FFFFF000-FFFFF0FFH	256バイト
PPx1	FFFFE100-FFFFE1FFH	256バイト	PPx17	FFFFF100-FFFFF1FFH	256バイト
PPx2	FFFFE200-FFFFE2FFH	256バイト	PPx18	FFFFF200-FFFFF2FFH	256バイト
PPx3	FFFFE300-FFFFE3FFH	256バイト	PPx19	FFFFF300-FFFFF3FFH	256バイト
PPx4	FFFFE400-FFFFE4FFH	256バイト	PPx20	FFFFF400-FFFFF4FFH	256バイト
PPx5	FFFFE500-FFFFE5FFH	256バイト	PPx21	FFFFF500-FFFFF5FFH	256バイト
PPx6	FFFFE600-FFFFE6FFH	256バイト	PPx22	FFFFF600-FFFFF6FFH	256バイト
PPx7	FFFFE700-FFFFE7FFH	256バイト	PPx23	FFFFF700-FFFFF7FFH	256バイト
PPx8	FFFFE800-FFFFE8FFH	256バイト	PPx24	FFFFF800-FFFFF8FFH	256バイト
PPx9	FFFFE900-FFFFE9FFH	256バイト	PPx25	FFFFF900-FFFFF9FFH	256バイト
PPx10	FFFFEA00-FFFFEAFFH	256バイト	PPx26	FFFFFA00-FFFFFAFFH	256バイト
PPx11	FFFFEB00-FFFFEBFFH	256バイト	PPx27	FFFFFB00-FFFFFBFFH	256バイト
PPx12	FFFFEC00-FFFFECFFH	256バイト	PPx28	FFFFFC00-FFFFFCFFH	256バイト
PPx13	FFFFED00-FFFFEDFFH	256バイト	PPx29	FFFFFD00-FFFFFDFFH	256バイト
PPx14	FFFFEE00-FFFFEEFFH	256バイト	PPx30	FFFFFE00-FFFFFEFFH	256バイト
PPx15	FFFFEF00-FFFFEFFFH	256バイト	PPx31	FFFFF000-FFFFF0FFH	256バイト

## 付録E V850E2M CPUと他のCPUの相違点

### E.1 V850E1, V850E2との相違点

( 1/2 )

項 目		V850E2M	V850E2	V850E1			
命令 (オペランド を含む)	ADF cccc, reg1, reg2, reg3	あり		なし			
	HSH reg2, reg3						
	JARL disp32, reg1						
	JMP disp32, [reg1]						
	JR disp32						
	MAC reg1, reg2, reg3, reg4						
	MACU reg1, reg2, reg3, reg4						
	SAR reg1, reg2, reg3						
	SATADD reg1, reg2, reg3						
	SATSUB reg1, reg2, reg3						
	SBF cccc, reg1, reg2, reg3						
	SCH0L reg1, reg2						
	SCH0R reg1, reg2						
	SCH1L reg1, reg2						
	SCH1R reg1, reg2						
	SHL reg1, reg2, reg3						
	SHR reg1, reg2, reg3						
	CAXI [reg1], reg2, reg3				あり	なし	
	DIVQ reg1, reg2, reg3						
	DIVQU reg1, reg2, reg3						
	EIRET						
	FERET						
	FETRAP vector4						
RIE							
SYNCM							
SYNCP							
SYNCE							
SYSCALL vector8							
LD.B disp23 [reg1], reg3							
LD.BU disp23 [reg1], reg4							
LD.H disp23 [reg1], reg3							
LD.HU disp23 [reg1], reg3							
LD.W disp23 [reg1], reg3							
ST.B reg3, disp23 [reg1]							
ST.H reg3, disp23 [reg2]							
ST.W reg3, disp23 [reg3]							
浮動小数点演算命令	あり	なし					

(2/2)

項目	V850E2M	V850E2	V850E1
命令実行クロック数	一部の命令で数値が異なります		
プログラム領域	4 Gバイト <sup>注1</sup>	512 Mバイト	64 Mバイト
プログラム・カウンタ (PC) の有効ビット	32ビット <sup>注1</sup>	下位29ビット	下位26ビット
データ領域	4 Gバイト		256 M/64 Mバイト
システム・レジスタ・バンク	あり	なし	
基本バンク	あり	あり <sup>注2</sup>	
PSW	機能が異なります。		
ECR	あり (原則使用禁止)	あり	
EIWR	あり	なし	
FEWR			
EIIC			
FEIC			
BSEL			
SCCFG			
SCBP			
例外ハンドラ・アドレス切り替え機能バンク0			
例外ハンドラ・アドレス切り替え機能バンク1			
プロセッサ保護違反バンク			
プロセッサ保護設定バンク			
ソフトウェア・ページング・バンク			
FPUステータス・バンク			
FPEC			
ユーザ0バンク			
プロセッサ保護機能	あり	なし	
例外	FEレベル・ノンマスカブル例外	FENMI	NMI2 <sup>注3</sup>
	FEレベル・マスカブル例外	FEINT	NMI0, NMI1 <sup>注3</sup>
	EIレベル・マスカブル例外	INT	INT
	メモリ保護例外	あり (30H)	なし
	浮動小数点演算例外	あり (70H)	なし
	FEレベル例外からの復帰命令	FERET	RETI
	EIレベル例外からの復帰命令	EIRET	
	例外の確認 / 取り下げ	あり	なし
	定義されていないオペコードの実行	予約命令例外 FEレベル例外 (30H)	不正命令例外 DBレベル例外 (60H)
動作モード	ミスアライン・アクセスの許可設定	常に許可	許可 / 禁止を設定可能
パイプライン	7段		5段
	各命令で、パイプラインの流れが異なります。		
デバッグ機能	機能が異なります。		

注1. 製品仕様により、命令アドレッシング範囲が512 Mバイトに制限されたCPUでは、EIPCのビット31-29はビット28を符号拡張した値が自動的に設定されます。

- バンク構成をとっておらず、基本バンク相当のシステム・レジスタのみ存在します。
- 例外ハンドラ・アドレスや例外要因コードなど、一部仕様が異なります。



## 付録F 命令索引

### F.1 基本命令索引

[ A ]	[ H ]	[ P ]	
ADD..... 71	HALT ..... 103	PREPARE ..... 131	SWITCH ..... 171
ADDI..... 72	HSH..... 104		SXB ..... 172
ADF ..... 73	HSW ..... 105	[ R ]	SXH ..... 173
AND..... 74		RETI ..... 133	SYNCE ..... 174
ANDI..... 75	[ J ]	RIE ..... 135	SYNCM..... 175
	JARL..... 106		SYNCP ..... 176
[ B ]	JMP ..... 108	[ S ]	SYSCALL ..... 177
Bcond ..... 76	JR ..... 109	SAR ..... 136	
BSH..... 78		SASF ..... 138	[ T ]
BSW ..... 79	[ L ]	SATADD ..... 139	TRAP ..... 179
	LD.B ..... 110	SATSUB ..... 141	TST..... 180
[ C ]	LD.BU ..... 111	SATSUBI ..... 142	TST1..... 181
CALLT ..... 80	LD.H ..... 112	SATSUBR..... 143	
CAXI..... 81	LD.HU ..... 113	SBF ..... 144	[ X ]
CLR1 ..... 82	LD.W ..... 114	SCH0L..... 145	XOR..... 182
CMOV..... 84	LDSR..... 115	SCH0R ..... 146	XORI..... 183
CMP ..... 86		SCH1L..... 147	
CTRET ..... 87	[ M ]	SCH1R ..... 148	[ Z ]
	MAC ..... 116	SET1 ..... 149	ZXB ..... 184
[ D ]	MACU..... 117	SETF ..... 151	ZXH ..... 185
DI..... 88	MOV ..... 118	SHL ..... 153	
DISPOSE ..... 89	MOVEA ..... 119	SHR..... 155	
DIV ..... 91	MOVHI..... 120	SLD.B ..... 157	
DIVH..... 92	MUL..... 121	SLD.BU ..... 158	
DIVHU ..... 94	MULH ..... 122	SLD.H ..... 159	
DIVQ..... 95	MULHI ..... 123	SLD.HU ..... 160	
DIVQU ..... 97	MULU ..... 124	SLD.W ..... 161	
DIVU..... 98		SST.B ..... 162	
	[ N ]	SST.H..... 163	
[ E ]	NOP..... 125	SST.W ..... 164	
EI..... 99	NOT..... 126	ST.B ..... 165	
EIRET..... 100	NOT1..... 127	ST.H ..... 166	
		ST.W ..... 167	
[ F ]	[ O ]	STSR..... 168	
FERET..... 101	OR ..... 129	SUB ..... 169	
FETRAP ..... 102	ORI ..... 130	SUBR ..... 170	

## F.2 浮動小数点演算命令索引

### [ A ]

ABSF.D.....325  
 ABSF.S.....326  
 ADDF.D.....327  
 ADDF.S.....328

### [ C ]

CEILF.DL.....329  
 CEILF.DUL.....330  
 CEILF.DUW.....331  
 CEILF.DW.....332  
 CEILF.SL.....333  
 CEILF.SUL.....334  
 CEILF.SUW.....335  
 CEILF.SW.....336  
 CMOVF.D.....337  
 CMOVF.S.....338  
 CMPF.D.....339  
 CMPF.S.....342  
 CVTF.DL.....345  
 CVTF.DS.....346  
 CVTF.DUL.....347  
 CVTF.DUW.....348  
 CVTF.DW.....349  
 CVTF.LD.....350  
 CVTF.LS.....351  
 CVTF.SD.....352  
 CVTF.SL.....353  
 CVTF.SUL.....354  
 CVTF.SUW.....355  
 CVTF.SW.....356  
 CVTF.ULD.....357  
 CVTF.ULS.....358  
 CVTF.UWD.....359  
 CVTF.UWS.....360  
 CVTF.WD.....361  
 CVTF.WS.....362

### [ D ]

DIVF.D.....363  
 DIVF.S.....364

### [ F ]

FLOORF.DL.....365  
 FLOORF.DUL.....366  
 FLOORF.DUW.....367  
 FLOORF.DW.....368  
 FLOORF.SL.....369  
 FLOORF.SUL.....370  
 FLOORF.SUW.....371  
 FLOORF.SW.....372

### [ M ]

MADDF.S.....373  
 MAXF.D.....375  
 MAXF.S.....376  
 MINF.D.....377  
 MINF.S.....378  
 MSUBF.S.....379  
 MULF.D.....381  
 MULF.S.....382

### [ N ]

NEGF.D.....383  
 NEGF.S.....384  
 NMADDF.S.....385  
 NMSUBF.S.....387

### [ R ]

RECIPF.D.....389  
 RECIPF.S.....390  
 RSQRTF.D.....391  
 RSQRTF.S.....392

### [ S ]

SQRTF.D.....393  
 SQRTF.S.....394  
 SUBF.D.....395  
 SUBF.S.....396

### [ T ]

TRFSR.....397  
 TRNCF.DL.....398  
 TRNCF.DUL.....399  
 TRNCF.DUW.....400  
 TRNCF.DW.....401  
 TRNCF.SL.....402  
 TRNCF.SUL.....403  
 TRNCF.SUW.....404  
 TRNCF.SW.....405

改訂記録	V850E2M ユーザーズマニュアル アーキテクチャ編
------	-----------------------------

Rev.	発行日	改訂内容	
		ページ	ポイント
0.01	2009.12.08	—	初版発行
0.02	2010.08.27	p.89	第2編 5.3 命令セット DISPOSE [オペレーション] 変更
		p.181	第2編 5.3 命令セット TST1 [オペレーション] 変更
		p.255	第3編 表7-1 周辺装置保護機能レジスタ・セット 変更
		p.277	第3編 8.1.2 TSECR - タイミング監視例外要因 変更
		p.402	第4編 4.4 命令セット TRNCF.SL [オペコード] 変更
		p.403	第4編 4.4 命令セット TRNCF.SUL [オペコード] 変更
		p.424, 425	表B-1 基本命令オペコード一覧 (16, 32ビット命令) 変更
		p.428, 429	表B-4 浮動小数点演算命令オペコード一覧 変更
1.00	2012.10.17	p.434-437	表C-1 基本命令の命令実行クロック数一覧 変更
		p.187	第2編 表6-1 例外要因一覧(1/2) 注7 追加
		p.263	第3編 7.1.9 PPSn - 特殊周辺装置の指定 初期値の参照先 変更 第3編 7.1.10 PPPn - OS 周辺装置の指定 初期値の参照先 変更
		p.264	第3編 7.1.11 PPVn - 一般周辺装置保護の有効指定 初期値の参照先 変更
		p.265	第3編 7.1.12 PPTn - 一般周辺装置の保護種別の指定 初期値の参照先 変更

---

V850E2M ユーザーズマニュアル アーキテクチャ編

発行年月日 2012 年 10 月 17 日 Rev.1.00

発行 ルネサス エレクトロニクス株式会社  
〒211-8668 神奈川県川崎市中原区下沼部 1753

---



ルネサスエレクトロニクス株式会社

■営業お問合せ窓口

<http://www.renesas.com>

※営業お問合せ窓口の住所・電話番号は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス販売株式会社 〒100-0004 千代田区大手町2-6-2 (日本ビル)

(03)5201-5307

■技術的なお問合せおよび資料のご請求は下記へどうぞ。

総合お問合せ窓口：<http://japan.renesas.com/contact/>

V850E2M