User's Manual

32

V850E2S

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

V850E2S **マイクロプロセッサ・コア**

ルネサス エレクトロニクス
www.renesas.com

Rev.1.00　2014.05

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ものですが、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も、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

① 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デバイスの入力が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。

② 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。

③ 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。

④ 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。

⑤ 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。

⑥ 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，V850E2S 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V850E2S 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

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。

ハードウエアの機能について知りたいとき
　→各製品のユーザーズ・マニュアル　ハードウエ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 11

第1章 特　　徴 12
1. 1 **基本機能** 12
1. 2 **プロセッサ保護機能** 13

第2編　基本機能 14

第1章 概　　説 15
1. 1 **特　　徴** 16

第2章 レジスタ・セット 17
2. 1 **プログラム・レジスタ** 18
2. 2 **システム・レジスタ・バンク** 20
2. 2. 1 BSEL － **レジスタ・バンクの選択** 22
2. 3 CPU**機能グループ／基本バンク** 23
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** 24
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** 24
2. 3. 3 ECR － **例外要因** 25
2. 3. 4 PSW － **プログラム・ステータス・ワード** 25
2. 3. 5 SCCFG － SYSCALL**の動作設定** 29
2. 3. 6 SCBP － SYSCALL**ベース・ポインタ** 29
2. 3. 7 EIIC － EI**レベル例外要因** 29
2. 3. 8 FEIC － FE**レベル例外要因** 30
2. 3. 9 CTPC, CTPSW － CALLT**実行時の状態退避レジスタ** 30
2. 3. 10 CTBP － CALLT**ベース・ポインタ** 30
2. 3. 11 EIWR － EI**レベル例外用作業レジスタ** 31
2. 3. 12 FEWR － FE**レベル例外用作業レジスタ** 31
2. 3. 13 DBIC － DB**レベル例外要因** 31
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** 31
2. 3. 15 DBWR － DB**レベル例外用作業レジスタ** 31
2. 3. 16 DIR － **デバッグ・インタフェース・レジスタ** 31
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** 32
2. 4. 1 SW_CTL － **例外ハンドラ・アドレス切り替えの制御** 33
2. 4. 2 SW_CFG － **例外ハンドラ・アドレス切り替え設定** 33
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** 33
2. 4. 4 EH_CFG － **例外ハンドラ設定** 34
2. 4. 5 EH_BASE － **例外ハンドラ・ベース・アドレス** 34
2. 4. 6 EH_RESET － **リセット・アドレス** 35
2. 5 **ユーザ・グループ** 36

第3章 データ・タイプ 37
3. 1 **データ形式** 37
3. 1. 1 **バイト** 37
3. 1. 2 **ハーフワード** 37
3. 1. 3 **ワード** 38
3. 1. 4 **ビット** 38
3. 2 **データ表現** 39
3. 2. 1 **整　　数** 39
3. 2. 2 **符号なし整数** 39

3. 2. 3 **ビット** 39
3. 3 **データ・アラインメント** 40

第4章 アドレス空間 41
4. 1 **メモリ・マップ** 42
4. 2 **アドレシング・モード** 44
4. 2. 1 **命令アドレス** 44
4. 2. 2 **オペランド・アドレス** 46

第5章 命　　令 49
5. 1 **オペコードと命令フォーマット** 49
5. 1. 1 CPU**命令** 49
5. 1. 2 **コプロセッサ命令** 54
5. 1. 3 **予約命令** 54
5. 2 **命令の概要** 55
5. 3 **命令セット** 60
ADD 63
ADDI 64
ADF 65
AND 66
ANDI 67
Bcond 68
BSH 70
BSW 71
CALLT 72
CAXI 73
CLR1 74
CMOV 76
CMP 78
CTRET 79
DI 80
DISPOSE 81
DIV 83
DIVH 84
DIVHU 86
DIVQ 87
DIVQU 88
DIVU 89
EI 90
EIRET 91
FERET 92
FETRAP 93
HALT 94
HSH 95
HSW 96
JARL 97
JMP 99
JR 100
LD.B 101
LD.BU 102
LD.H 103
LD.HU 104
LD.W 105

LDSR 106
MAC 107
MACU 108
MOV 109
MOVEA 110
MOVHI 111
MUL 112
MULH 113
MULHI 114
MULU 115
NOP 116
NOT 117
NOT1 118
OR 120
ORI 121
PREPARE 122
RETI 124
RIE 126
SAR 127
SASF 129
SATADD 130
SATSUB 132
SATSUBI 133
SATSUBR 134
SBF 135
SCH0L 136
SCH0R 137
SCH1L 138
SCH1R 139
SET1 140
SETF 142
SHL 144
SHR 146
SLD.B 148
SLD.BU 149
SLD.H 150
SLD.HU 151
SLD.W 152
SST.B 153
SST.H 154
SST.W 155
ST.B 156
ST.H 157
ST.W 158
STSR 159
SUB 160
SUBR 161
SWITCH 162
SXB 163
SXH 164
SYNCE 165
SYNCM 166

SYNCP 167
SYSCALL 168
TRAP 170
TST 171
TST1 172
XOR 173
XORI 174
ZXB 175
ZXH 176

第6章 例　　外 177
6. 1 **例外の仕組み** 177
6. 1. 1 **例外要因一覧** 177
6. 1. 2 **例外の種別** 180
6. 1. 3 **例外処理フロー** 182
6. 1. 4 **例外受け付けの優先順位と保留条件** 183
6. 1. 5 **例外の受け付け条件** 183
6. 1. 6 **再開と回復** 183
6. 1. 7 **例外レベルとコンテキスト退避** 184
6. 1. 8 **復帰命令** 185
6. 2 **例外発生時の動作** 188
6. 2. 1 **受け付け条件のない**EI**レベル例外** 188
6. 2. 2 **受け付け条件のある**EI**レベル例外** 190
6. 2. 3 **受け付け条件のない**FE**レベル例外** 192
6. 2. 4 **受け付け条件のある**FE**レベル例外** 194
6. 2. 5 **特殊な動作** 196
6. 3 **例外の管理** 198
6. 4 **例外ハンドラ・アドレス切り替え機能** 199
6. 4. 1 **例外ハンドラ・アドレスの決定** 199
6. 4. 2 **例外ハンドラ・アドレスの切り替えの目的** 200
6. 4. 3 **例外ハンドラ・アドレス切り替え機能の設定方法** 200

第7章 コプロセッサ使用不可状態 201
7. 1 **コプロセッサ使用不可例外** 201
7. 2 **システム・レジスタ** 201

第8章 リセット 202
8. 1 **リセット後のレジスタの状態** 202
8. 2 **起　　動** 202

第3編　プロセッサ保護機能 203

第1章 概　　説 204
1. 1 **特　徴** 204

第2章 レジスタ・セット 205
2. 1 **システム・レジスタ・バンク** 205
2. 2 **システム・レジスタ** 207
2. 2. 1 PSW － **プログラム・ステータス・ワード** 210
2. 2. 2 MPM － **プロセッサ保護動作モードの設定** 210
2. 2. 3 MPC － **プロセッサ保護コマンドの指定** 212
2. 2. 4 TID － **タスク識別子** 213
2. 2. 5 **その他のシステム・レジスタ** 213

第3章 動作設定 214

3. 1 プロセッサ保護機能の利用開始 214
3. 2 実行レベル自動遷移機能の設定 214
3. 3 プロセッサ保護機能の利用停止 214

第4章 実行レベル 215
4. 1 プログラムの性質 215
4. 2 PSW上の保護ビット 215
4. 2. 1 Tステート（信頼状態） 216
4. 2. 2 NTステート（非信頼状態） 216
4. 3 実行レベルの定義 216
4. 4 実行レベルの遷移 216
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 217
4. 4. 2 例外の発生による遷移 217
4. 4. 3 復帰命令の実行による遷移 217
4. 5 プログラム・モデル 218
4. 6 タスク識別子 218

第5章 システム・レジスタ保護 219
5. 1 レジスタ・セット 220
5. 1. 1 VSECR － システム・レジスタ保護違反要因 221
5. 1. 2 VSTID － システム・レジスタ保護違反タスク識別子 221
5. 1. 3 VSADR － システム・レジスタ保護違反アドレス 221
5. 2 アクセス制御 222
5. 3 対象レジスタ 222
5. 4 違反の検出 223
5. 5 運用方法 223

第6章 メモリ保護 224
6. 1 レジスタ・セット 225
6. 1. 1 PAnL － 保護領域n下限アドレス（n = 0-3） 226
6. 1. 2 PAnU － 保護領域n上限アドレス（n = 0-3） 227
6. 1. 3 VMECR － メモリ保護違反要因 228
6. 1. 4 VMTID － メモリ保護違反タスク識別子 229
6. 1. 5 VMADR － メモリ保護違反アドレス 229
6. 2 アクセス制御 229
6. 3 保護領域の設定 230
6. 3. 1 有効ビット（Eビット） 231
6. 3. 2 実行許可ビット（Xビット） 231
6. 3. 3 リード許可ビット（Rビット） 231
6. 3. 4 ライト許可ビット（Wビット） 231
6. 3. 5 sp相対アクセス許可ビット（Sビット） 231
6. 3. 6 保護領域下限アドレス（AL31-AL0ビット） 231
6. 3. 7 保護領域上限アドレス（AU31-AU0ビット） 231
6. 4 保護領域設定時の注意事項 232
6. 4. 1 保護領域境界の交差 232
6. 4. 2 無効な保護領域の設定 232
6. 5 特殊なメモリ・アクセス命令 233
6. 5. 1 ミスアライン・アクセスを行うロード／ストア命令 234
6. 5. 2 一部のビット操作命令とCAXI命令 234
6. 5. 3 スタック・フレーム操作命令 234
6. 5. 4 SYSCALL命令 234
6. 6 保護違反と例外 235

第7章 プロセッサ保護例外 236
7. 1 違反の種類 236

7. 1. 1 システム・レジスタ保護違反 236
7. 1. 2 実行保護違反 236
7. 1. 3 データ保護違反 236
7. 2 例外の種類 237
7. 2. 1 MIP例外 237
7. 2. 2 MDP例外 237
7. 3 違反要因の特定 238
7. 3. 1 MIP例外 238
7. 3. 2 MDP例外 239

第8章 特殊機能 240
8. 1 メモリ保護設定の一括クリア 240

付録A 命令一覧 241
A. 1 基本命令 241

付録B 命令オペコード一覧 245
B. 1 基本命令オペコード一覧 245

付録C パイプライン 250
C. 1 特　　徴 251
C. 2 命令実行クロック数 253
C. 2. 1 基本命令の実行クロック数 253
C. 3 基本命令のパイプライン 258
C. 3. 1 ロード命令 258
C. 3. 2 ストア命令 258
C. 3. 3 乗算命令 259
C. 3. 4 加算付き乗算命令 260
C. 3. 5 算術演算命令 261
C. 3. 6 条件付き演算命令 261
C. 3. 7 飽和演算命令 261
C. 3. 8 論理演算命令 261
C. 3. 9 データ操作命令 261
C. 3. 10 ビット・サーチ命令 262
C. 3. 11 除算命令 262
C. 3. 12 高速除算命令 263
C. 3. 13 分岐命令 263
C. 3. 14 ビット操作命令 265
C. 3. 15 特殊命令 265

付録D V850E2S CPUと他のCPUの相違点 270
D. 1 V850E2, V850E2Mとの相違点 270

付録E 命令索引 272
E. 1 基本命令索引 272

第1編　概　　要

第1章　特　　徴

V850E2S 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。

V850E2S 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

V850E2S CPU は，5 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，V850E2S CPU は，V850 CPU, V850E1 CPU, V850E2 CPU, V850E2M CPU, V850ES 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エ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，ディスプレース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（16ビット×16ビット乗算），バレル・シフタ（32ビット／1クロック），除算器などの専用ハードウエア内蔵し，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，システム・レジスタ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。

第 2 編　基本機能

第1章　概　　説

V850E2S 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・ディスプレース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) 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。

(4) 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（ハードウエ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　レジスター覧

(a) プログラム・レジスタ

31 0

r0（ゼロ・レジスタ）
r1（アセンブラ予約レジスタ）
r2
r3（スタック・ポインタ（SP））
r4（グローバル・ポインタ（GP））
r5（テキスト・ポインタ（TP））
r6
r7
r8
r9
r10
r11
r12
r13
r14
r15
r16
r17
r18
r19
r20
r21
r22
r23
r24
r25
r26
r27
r28
r29
r30（エレメント・ポインタ（EP））
r31（リンク・ポインタ（LP））

PC（プログラム・カウンタ）

(b) システム・レジスタ

プロセッサ保護機能グループ・システム・レジスタ群

CPU機能グループ・システム・レジスタ群

31 0

EIPC － EIレベル例外受け付け時の状態退避レジスタ
EIPSW － EIレベル例外受け付け時の状態退避レジスタ
FEPC － FEレベル例外受け付け時の状態退避レジスタ
FEPSW － FEレベル例外受け付け時の状態退避レジスタ
ECR － 例外要因
PSW － プログラム・ステータス・ワード
SCCFG － SYSCALLの動作設定
SCBP － SYSCALLベース・ポインタ
EIIC － EIレベル例外要因
FEIC － FEレベル例外要因
DBIC注 － DBレベル例外要因
CTPC － CALLT実行時の状態退避レジスタ
CTPSW － CALLT実行時の状態退避レジスタ
DBPC注 － DBレベル例外受け付け時の状態退避
DBPSW注 － DBレベル例外受け付け時の状態退避
CTBP － CALLTベース・ポインタ
デバッグ機能レジスタ群注
EIWR － EIレベル例外用作業レジスタ
FEWR － FEレベル例外用作業レジスタ
DBWR注 － DBレベル例外用作業レジスタ
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です。

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エ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，ソフトウエ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　命令アドレッシング範囲は64Mバイトです。ビット31-26はビット25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

V850E2S 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。

V850E2S CPUには次のようなグループとバンクがあります。

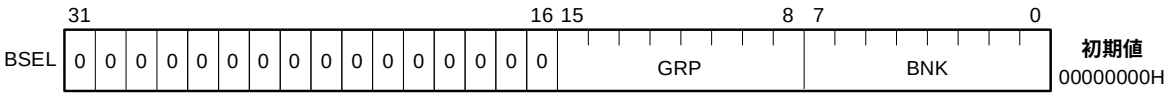
- CPU機能グループ
  - 基本バンク：従来のシステム・レジスタ群です。
  - 例外ハンドラ切り替え機能バンク0：例外ハンドラ切り替えを行うシステム・レジスタ群です。
  - 例外ハンドラ切り替え機能バンク1：例外ハンドラ切り替えを行うシステム・レジスタ群です。
- プロセッサ保護機能グループ
  - プロセッサ保護違反バンク：プロセッサ保護違反に関するシステム・レジスタ群です。
  - プロセッサ保護設定バンク：プロセッサ保護機能に関するシステム・レジスタ群です。
  - ソフトウエア・ページング・バンク：メモリ保護をページング方式で利用する場合に使用するシステム・レジスタ群です。
- 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。

図2−2 システム・レジスタ・バンク

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。



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 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）。

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	ソフトウエア・ページング・バンク
FFH	00H	ユーザ0バンク
上記以外		設定禁止

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・レジスター覧（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	×	○	○
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））	×	×	○
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注2	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注2	DBレベル例外受け付け時の状態退避レジスタ	○	○	○
19	DBPSW注2	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	○	○	×
21	DIR注2	デバッグ・インタフェース・レジスタ	○注3	○注4	○
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注2	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しません。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。

3. デバッグ・モードでのみ書き込み（更新）可能です。

4. ユーザ・モード時に読み出し値が不定となるビットがあります。

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. 3. 1　EIPC, EIPSW　ー　EIレベル例外受け付け時の状態退避レジスタ

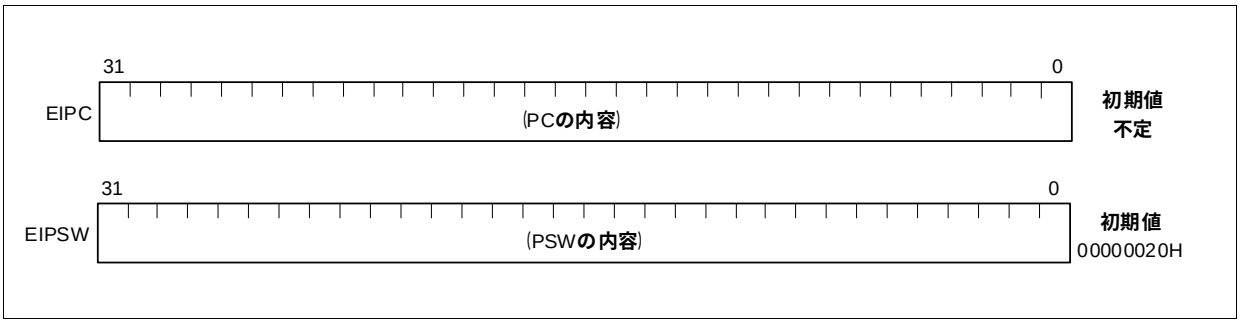
EIレベル例外時状態退避レジスタには，EIPCとEIPSWがあります。

EIレベル例外（EIレベル・ソフトウエア例外や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ー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　命令アドレッシング範囲は 64M バイトです。EIPC のビット 31-26 はビット 25 を符号拡張した値が自動的に設定されます。

2. 3. 2　FEPC, FEPSW　ー　FEレベル例外受け付け時の状態退避レジスタ

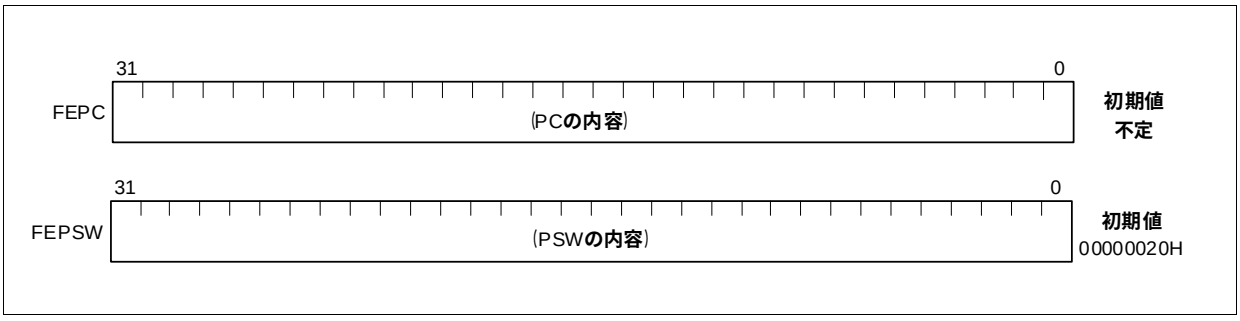
FE レベル例外時状態退避レジスタには，FEPC と FEPSW があります。

FE レベル例外（FE レベル・ソフトウエア例外や 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ー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。

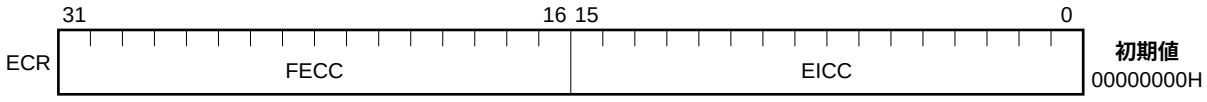


注意　命令アドレッシング範囲は 64 M バイトです。FEPC のビット 31-26 はビット 25 を符号拡張した値が自動的に設定されます。

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, V850ES 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31-19, 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[注]。

注　ビット 18-16 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）。

(1/3)

	31													18	17	16								7	6	5	4	3	2	1	0	
PSW	0	0	0	0	0	0	0	0	0	0	0	0	0	NPV	DMP	IMP	0	0	0	0	0	0	0	0	NP	EP	ID	SAT	CY	OV	S	Z	初期値 00000020H

ビット位置	ビット／フラグ名	意　味
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。
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。
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[注1]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

(2/3)

ビット位置	フラグ名	意 味
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注2	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	

（3/3）

ビット位置	フラグ名	意　味
2	OV 注	演算中にオーバフローが発生したかどうかを示します。 0：オーバフローが発生していない（初期値） 1：オーバフローが発生した
1	S 注	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。

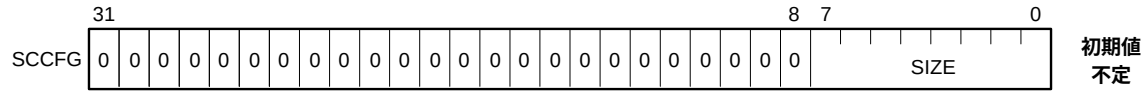
演算結果の状態	フラグの状態			飽和処理をした演算結果
	SAT	OV	S	
正の最大値を越えた	1	1	0	7FFFFFFFH
負の最大値を越えた	1	1	1	80000000H
正（最大値を越えない）	演算前の値を保持	0	0	演算結果そのもの
負（最大値を越えない）			1	

2.3.5 SCCFG ー 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。



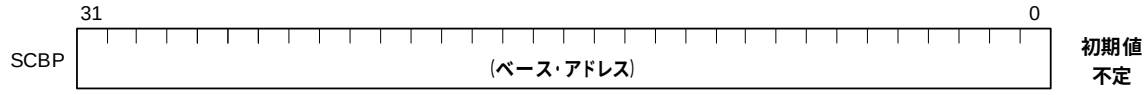
ビット位置	ビット名	意 味
7-0	SIZE	SYSCALL命令が参照するテーブルの最大エントリ数を指定します。SYSCALLが参照する最大エントリ数は，SIZEが0の場合は1エントリ，255の場合は256エントリです。SYSCALL命令で分岐する関数の数に合わせて，最大エントリ数を適切に設定することで，メモリ領域を有効に活用できます。 最大エントリ数を越えるベクタがSYSCALL命令で指定された場合には，先頭のエントリが選択されます。先頭のエントリには，エラー処理ルーチンを配置してください。

2.3.6 SCBP ー 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。

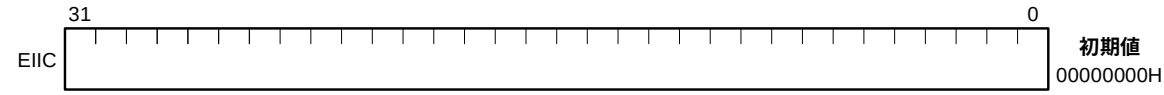
また，ビット1, 0は0に固定されています。



注意 命令アドレッシング範囲は 64M バイトです。SCBP のビット 31-26 はビット 25 を符号拡張した値が自動的に設定されます。

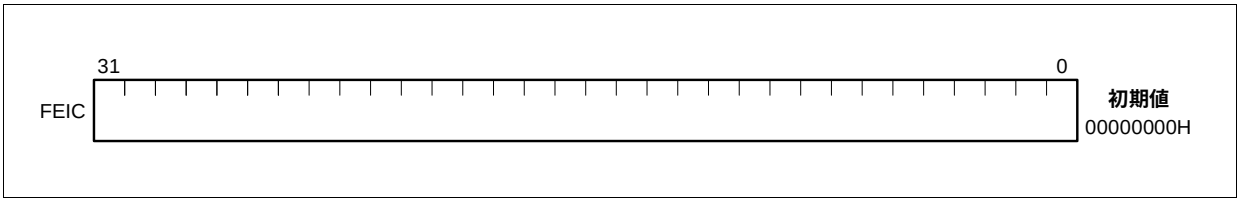
2.3.7 EIIC ー EIレベル例外要因

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ー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）。



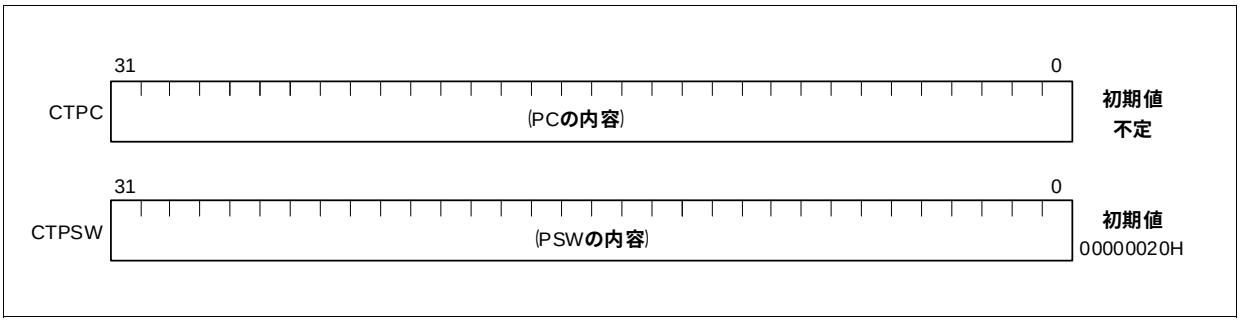
2. 3. 9　CTPC, CTPSW － CALLT実行時の状態退避レジスタ

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。



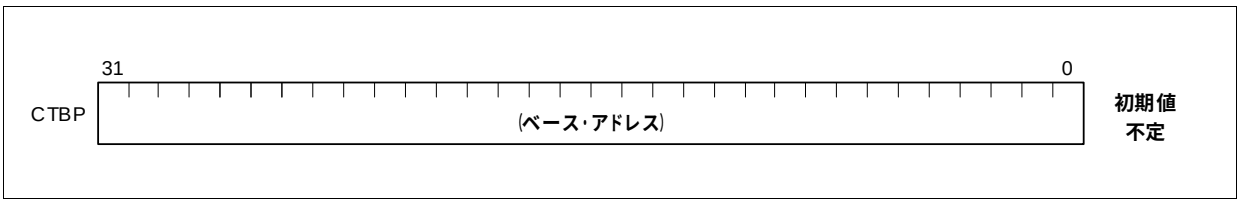
注意　命令アドレッシング範囲は 64M バイトです。CTPC のビット 31-26 はビット 25 を符号拡張した値が自動的に設定されます。

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。

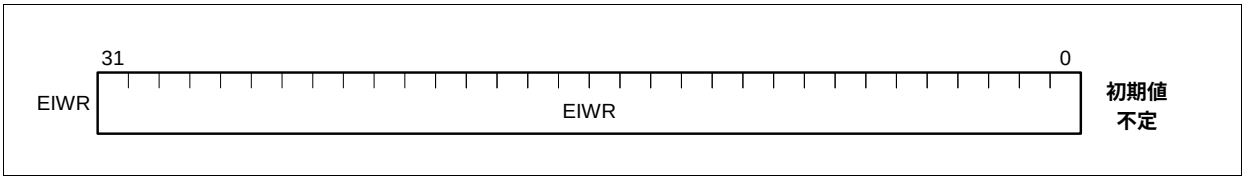


注意　命令アドレッシング範囲は 64 M バイトです。CTBP のビット 31-26 はビット 25 を符号拡張した値が自動的に設定されます。

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。

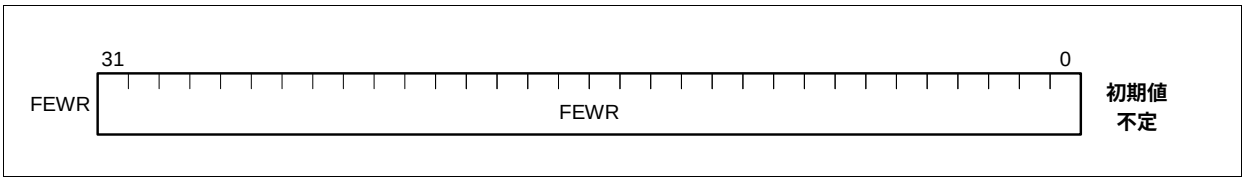
EIWR レジスタは，どのバンクが選択されているときでも常に参照可能です。



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，デバック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用します。

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用します。

2. 3. 15　DBWR － DBレベル例外用作業レジスタ

DBWRレジスタは，デバック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用します。

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用します。

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[注]	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						○	○	○

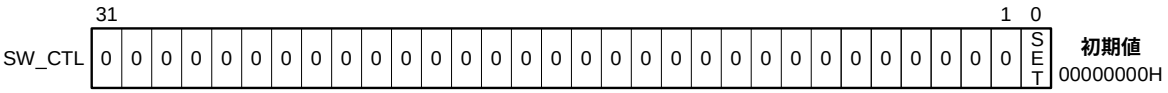
注 開発ツール向けのデバッグ機能のレジスタです。

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。

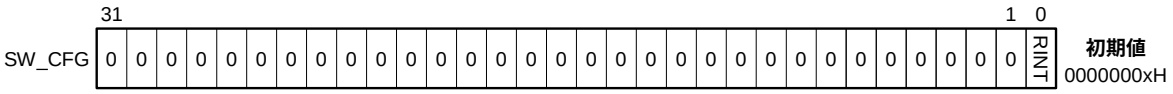


ビット位置	ビット名	意　味
0	SET	SETビットをセット（1）すると，SW_CFG, SW_BASEレジスタの値をEH_CFG, EH_BASEへ転送します。転送終了後，SETビットはクリア（0）されます。

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。

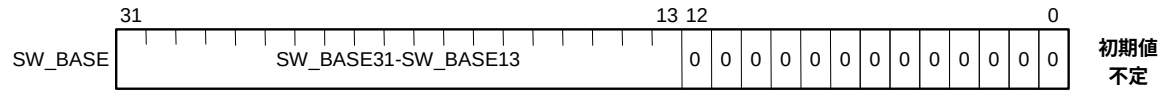


ビット位置	ビット名	意　味
0	RINT	SW_CTL.SETビットをセット（1）すると，SW_CFGレジスタの値がEH_CFGレジスタへ転送されます。

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。



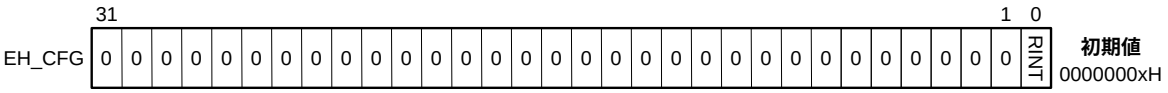
ビット位置	ビット名	意　味
31-13	SW_BASE31-SW_BASE13	SW_CTL.SETビットをセット（1）すると，このSW_BASEレジスタの内容がEH_BASEレジスタに転送されます。

注意　命令アドレッシング範囲は 64 M バイトです。SW_BASE のビット 31-26 はビット 25 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。

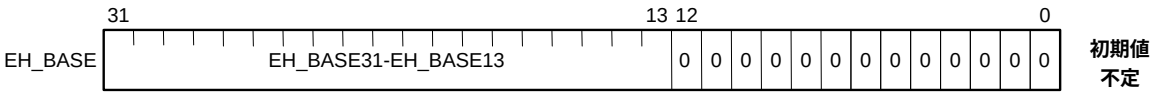


ビット位置	ビット名	意　味
0	RINT	RINTビットがセット（1）された場合，INT0-INT127の例外ハンドラ・アドレスを1つのハンドラ・アドレス（INT0）に縮小します。クリア（0）されている場合，INT0-INT127は独立した例外ハンドラ・アドレスを持ちます。EH_CFGレジスタは製品仕様によって，リセット時に初期値が設定されます。SW_CTL.SETビットをセット（1）することによって，SW_CFGの内容が転送されます。

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。



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。 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　命令アドレッシング範囲は 64M バイトです。EH_BASE のビット 31-26 はビット 25 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。

ビット位置	ビット名	意　味
31-13	EH_RESET31-EH_RESET13	EH_RESETレジスタは製品仕様によってリセット時に初期化される以外，値が変化しません。 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。

注意　命令アドレッシング範囲は 64 M バイトです。EH_RESET のビット 31-26 はビット 25 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・グループには次のバンクがあります。

・ユーザ0バンク（表2－4参照）

表2－4　システム・レジスター覧（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
6-15		（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注2	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しません。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。

×：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

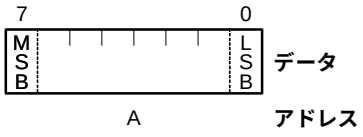
V850E2S CPUでは，データをリトル・エンディアン形式で取り扱います。つまり，ハーフワード，ワードではバイト0が常に最下位（最右端）バイトとなります。

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）

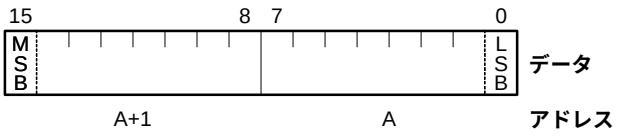
3.1.1　バイト

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

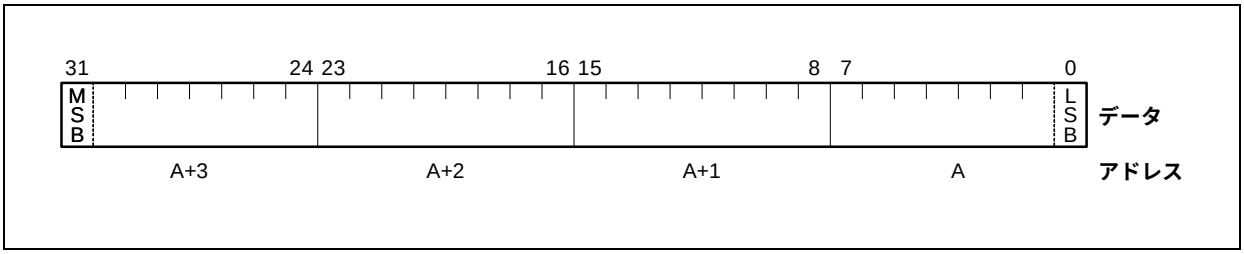
ハーフワードは任意のバイト境界[注]から始まる連続した2バイト（16ビット）のデータです。各ビットには，0から15までの番号が付けられており，LSBはビット0，MSBはビット15に対応します。ハーフワードはそのアドレス「A」で指定され，2つのアドレス「A」，「A＋1」のバイト・データを占めます。



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

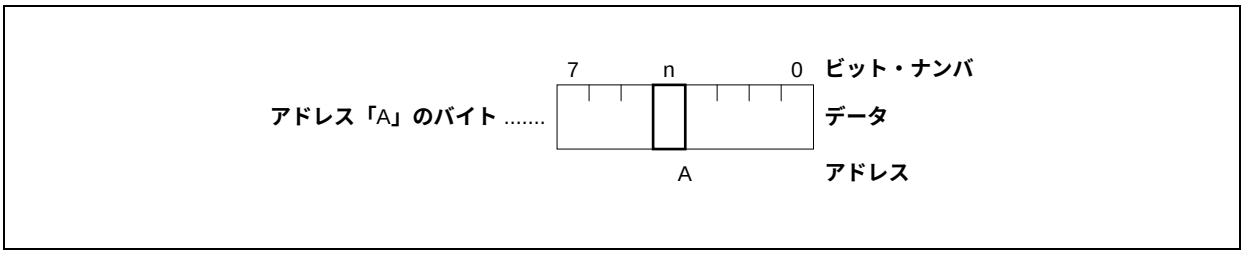
ワードは任意のバイト境界[注]から始まる連続した4バイト（32ビット）のデータです。各ビットには0から31までの番号が付けられており，LSBはビット0，MSBはビット31に対応します。ワードはそのアドレス「A」で指定され，4つのアドレス「A」，「A＋1」，「A＋2」，「A＋3」のバイト・データを占めます。



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

3. 3　データ・アラインメント

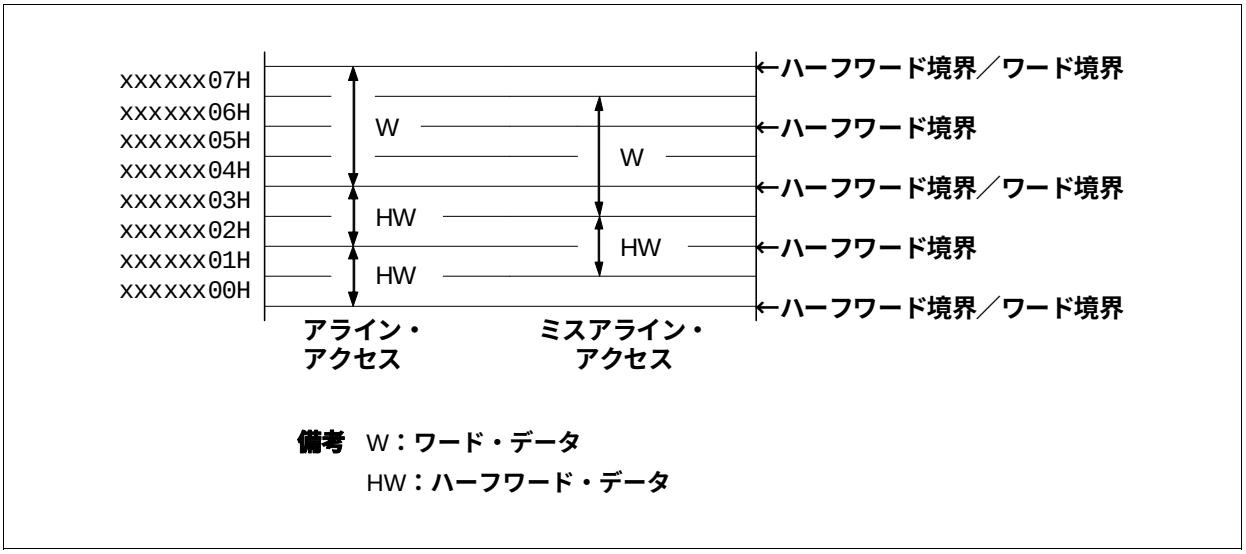
V850E2S CPU では，データのミスアライン配置を許可しています。

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。

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。

ただし，ハーフワード・データ，ワード・データの場合，データがアラインされていないと，バス・サイクルが最低 1 回余分に発生し，命令の実行時間が増加します。

図3－1　ミスアライン・アクセスのデータ配置例



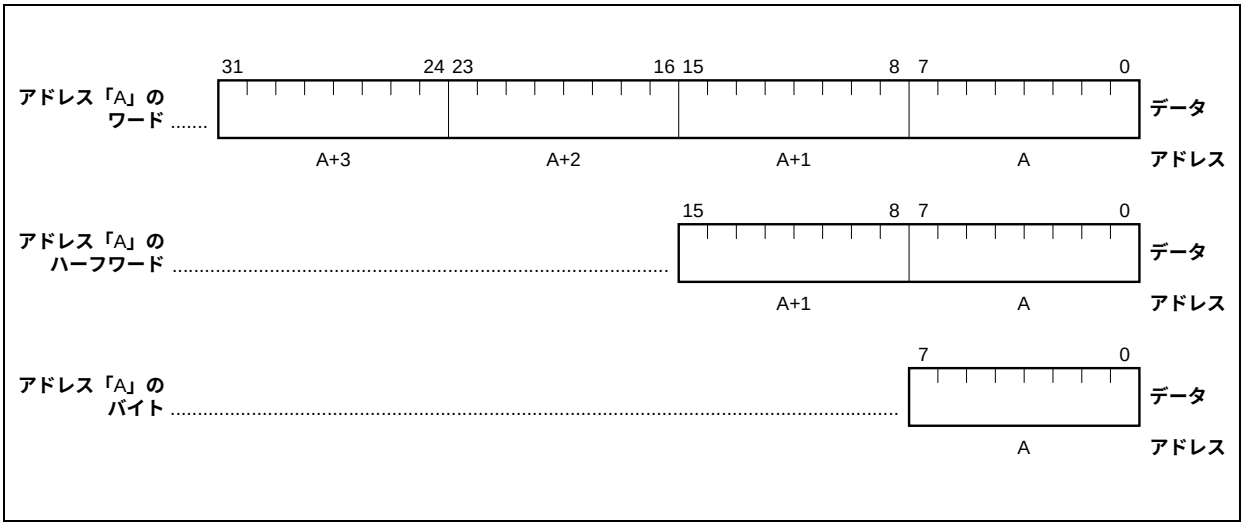
備考　W：ワード・データ
HW：ハーフワード・データ

第4章　アドレス空間

V850E2S 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。

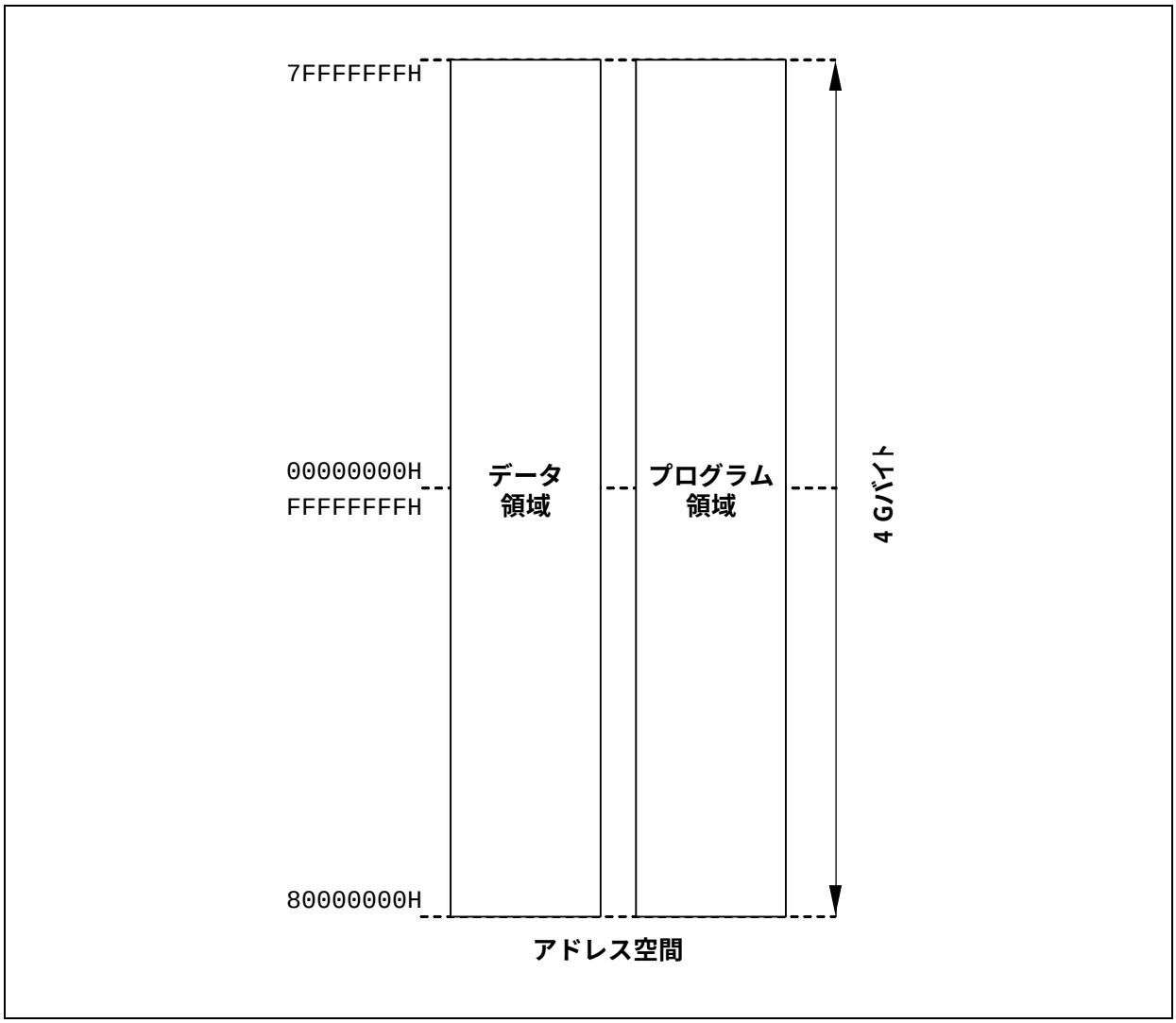


4. 1　メモリ・マップ

V850E2S 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。

メモリ・マップを図 4－1 に示します。

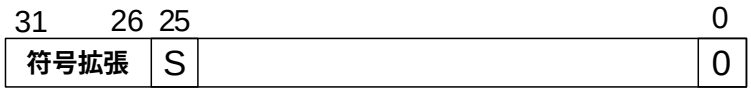
図4－1　メモリ・マップ（アドレス空間）



注意 V850E2S 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は64Mバイトです。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）
- VSADR（プロセッサ保護機能）

プログラム領域のアドレッシング可能範囲が64MバイトのV850E2S CPUにおいては，これらのレジスタの上位6ビットは25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-01FFFFFEHおよびFE000000H-FFFFFFFEHとなります（最下位ビットは常に0）。



この場合のメモリ・マップを次に示します。

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。データに関しても64Mバイトの範囲内に配置してください。

4. 2　アドレシング・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

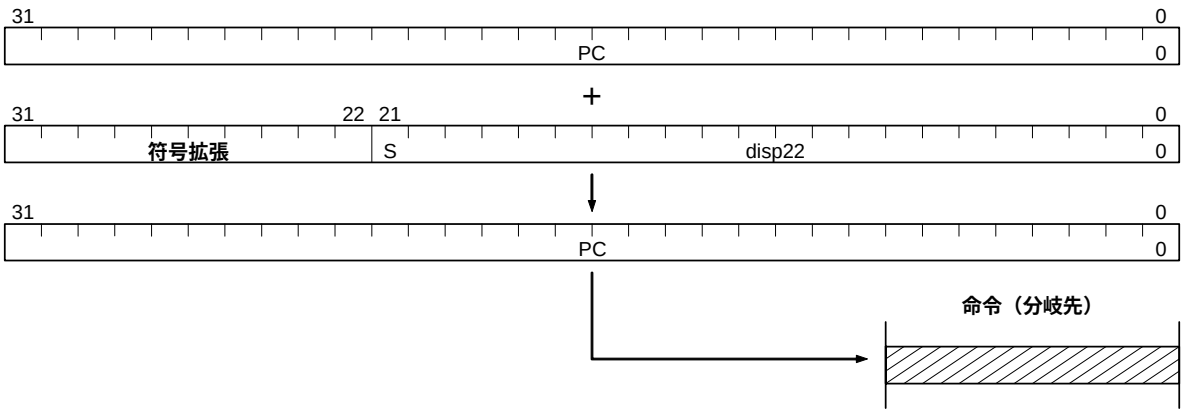
命令アドレスは，プログラム・カウンタ（PC）の内容によって決定され，実行した命令のバイト数に応じて自動的にインクリメントされます。また，分岐命令を実行する際には，次に示すアドレシングにより，分岐先アドレスをPCにセットします。

(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，このアドレシングの対象となります。

図4−2　レラティブ・アドレシング



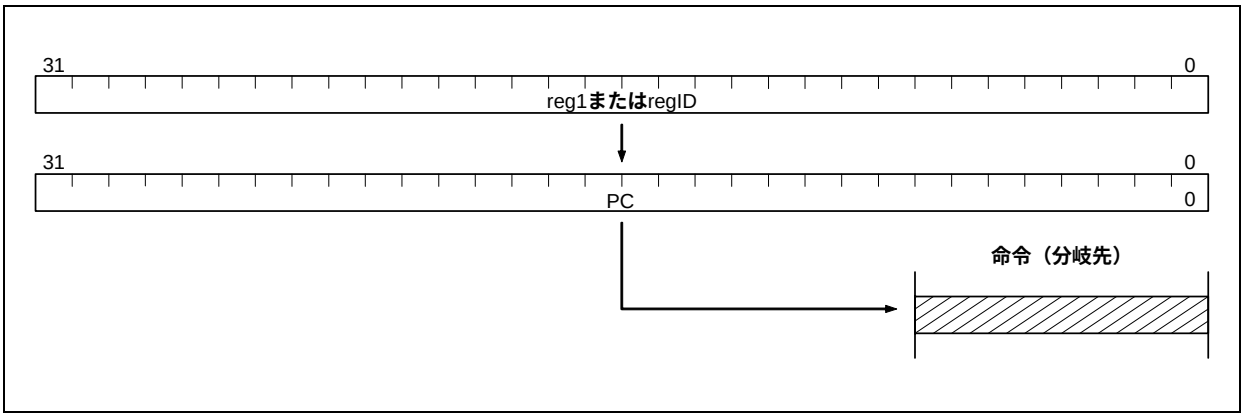
備考 22 ビット・ディスプレースメントの場合です。

(2) レジスタ・アドレシング（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，このアドレシングの対象となります。

図4－3　レジスタ・アドレシング

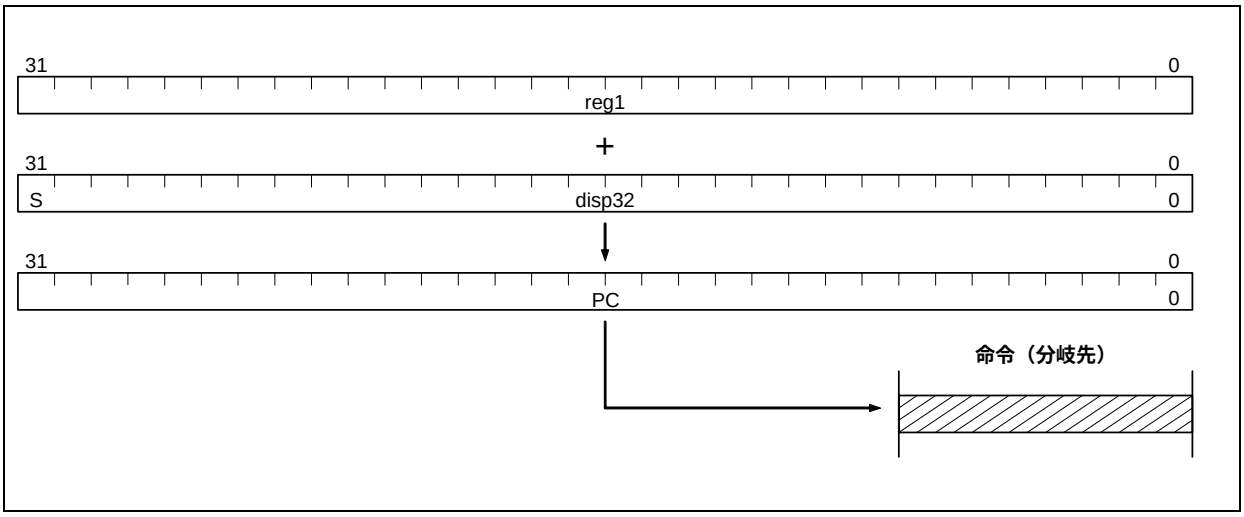


(3) ベースト・アドレシング

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，このアドレシングの対象となります。

図4－4　ベースト・アドレシング



(4) その他のアドレシング

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，このアドレシングの対象となります。

(2) イミーディエト・アドレシング

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）を指定するイミーディエトであり，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・イミーディエト・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・アドレシング

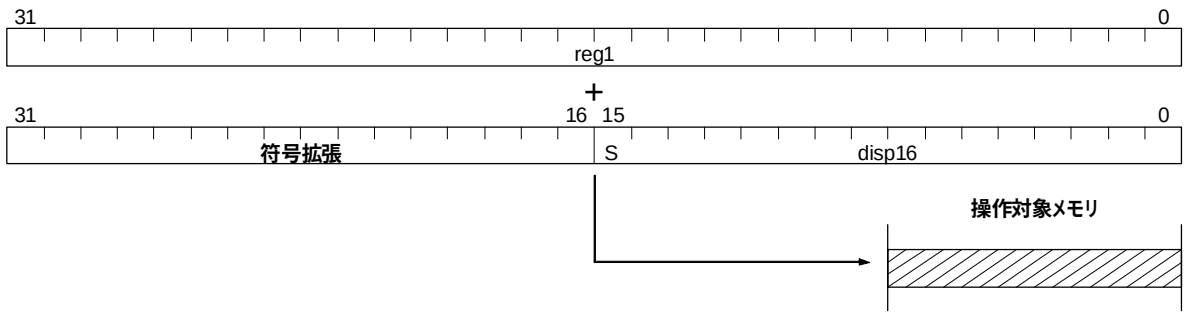
ベースト・アドレシングには，次に示す 2 種類があります。

(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シングです。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）。

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

図4－5　ベースト・アドレシング（タイプ1）



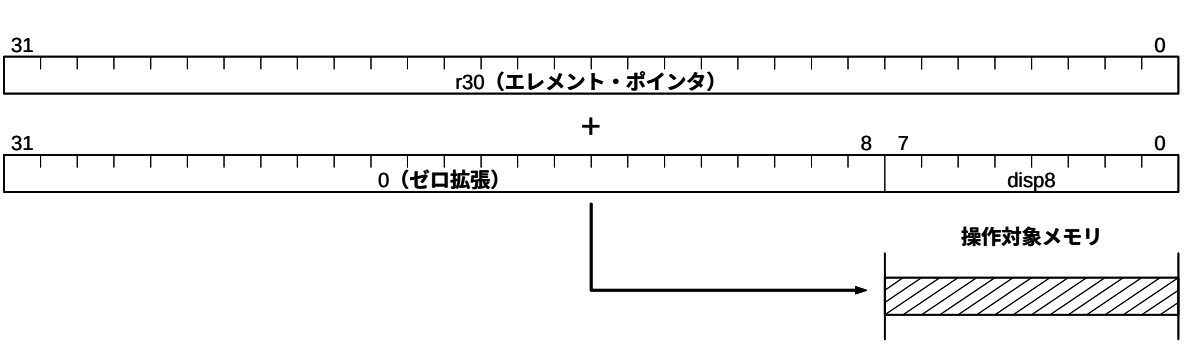
備考　16 ビット・ディスプレースメントの場合です。

(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・ディスプレースメント・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シングです。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，このアドレシングの対象となります。

図4－6　ベースト・アドレシング（タイプ2）



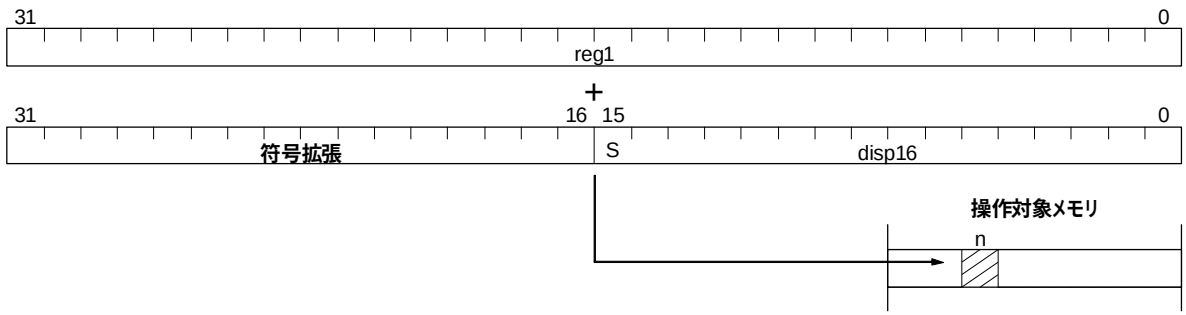
備考　8 ビット・ディスプレースメントの場合です

(4) ビット・アドレシング

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シングです。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）。

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

図4－7　ビット・アドレシング



備考　n：3 ビット・データ（bit #3）で指定されるビット位置（n = 0-7）
16 ビット・ディスプレースメントの場合です。

(5) その他のアドレシング

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

V850E2S CPUの命令には，基本命令として定義される「CPU命令」があります。

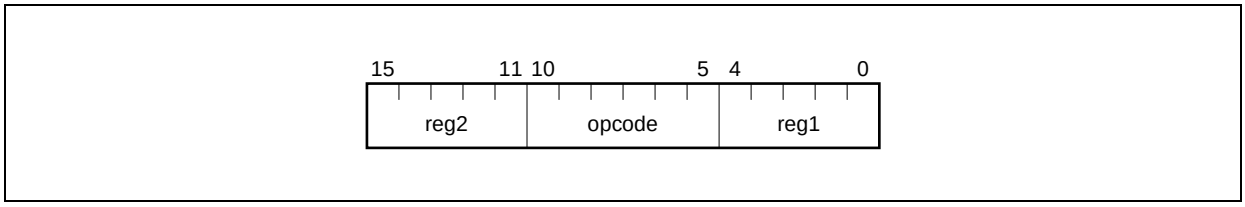
5. 1. 1　CPU命令

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。

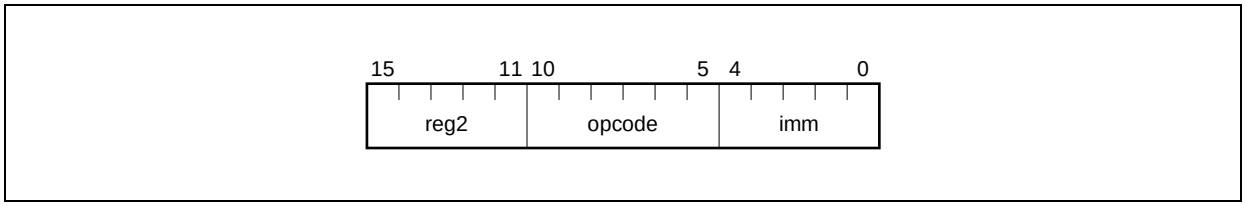
(1) reg-reg命令形式（Format I）

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。



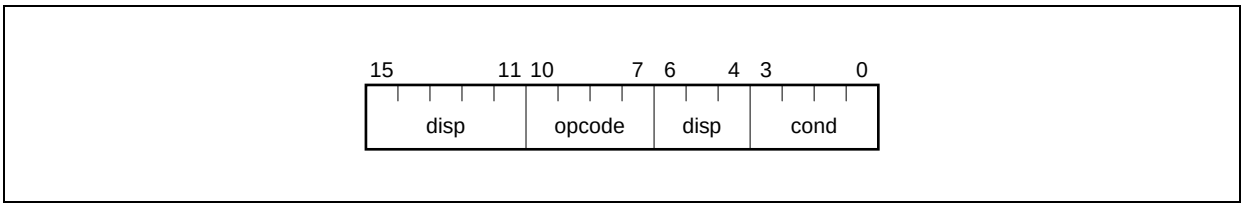
(2) imm-reg命令形式（Format II）

6 ビットのオペコード・フィールド，5 ビットのイミーディエト・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。



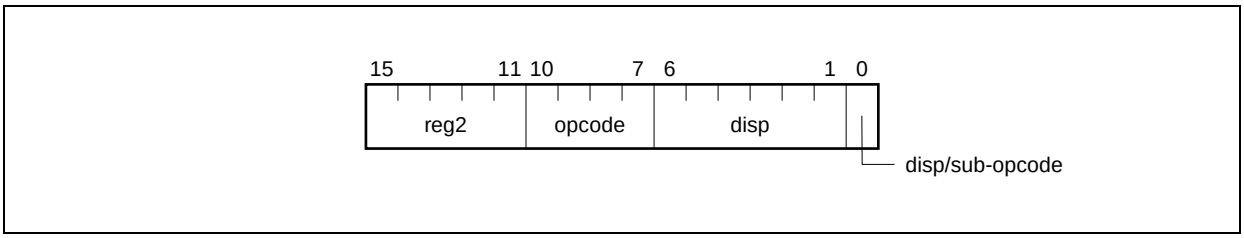
(3) 条件分岐命令形式（Format III）

4 ビットのオペコード・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ースメント・フィールドを持つ 16 ビット長命令形式。

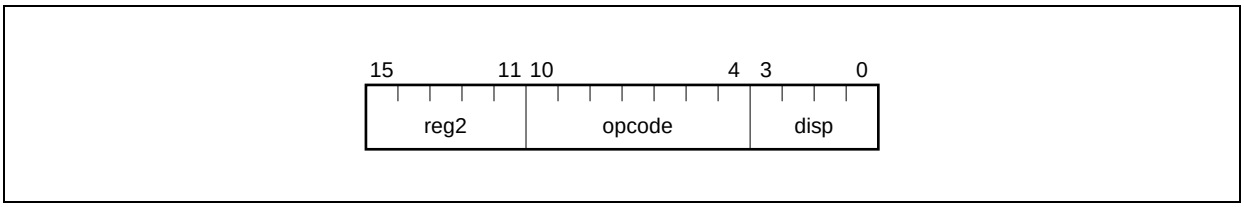


(4) ロード／ストア命令16ビット形式（Format IV）

4 ビットのオペコード・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ースメント・フィールド（または 6 ビット・ディスプレースメント・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 16 ビット長命令形式。

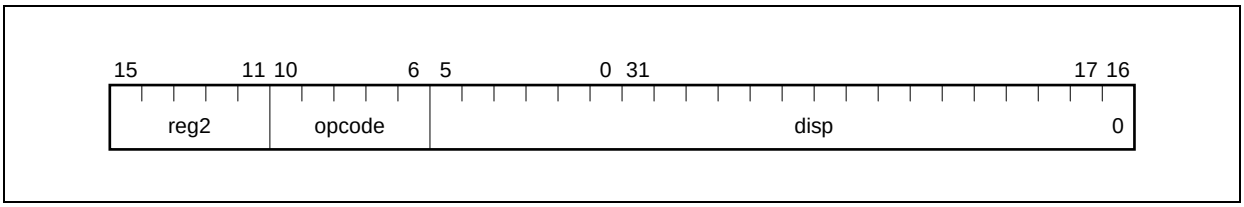


または，7 ビットのオペコード・フィールドと 1 つの汎用レジスタ指定フィールド，4 ビットのディスプレースメント・フィールドを持つ 16 ビット長命令形式。



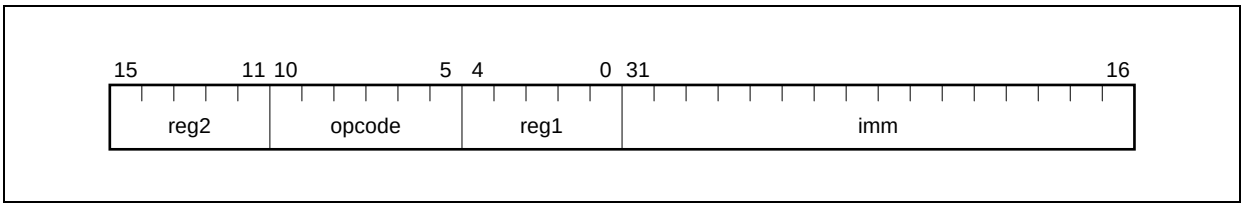
(5) ジャンプ命令形式（Format V）

5 ビットのオペコード・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ースメント・フィールドを持つ 32 ビット長命令形式。



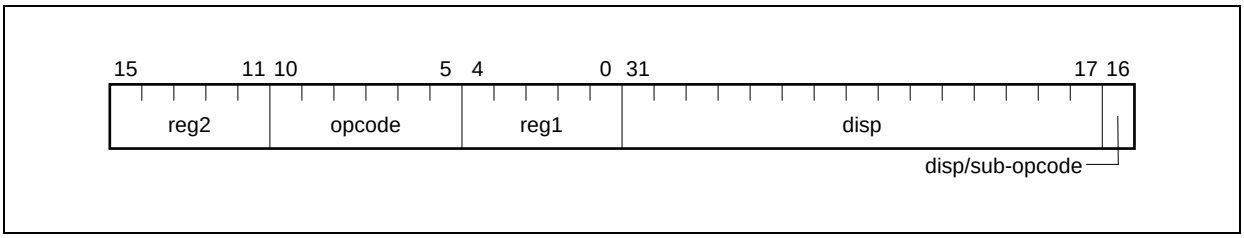
(6) 3オペランド命令形式（Format VI）

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，16 ビットのイミーディエト・フィールドを持つ 32 ビット長命令形式。



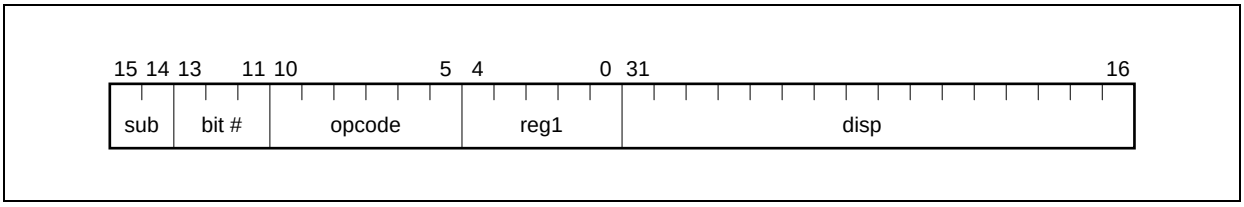
(7) ロード／ストア命令32ビット形式（Format VII）

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，16 ビットのディスプレースメント・フィールド（または 15 ビットのディスプレースメント・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）

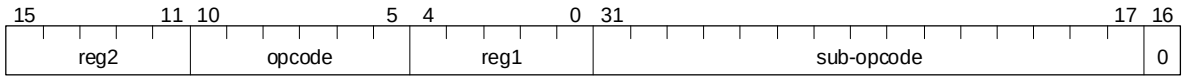
6 ビットのオペコード・フィールドと 2 ビットのサブオペコード・フィールド，3 ビットのビット指定フィールド，1 つの汎用レジスタ指定フィールド，16 ビットのディスプレースメント・フィールドを持つ 32 ビット長命令形式。



(9) 拡張命令形式1（Format IX）

6 ビットのオペコード・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オペコード・フィールドとして取り扱う 32 ビット長命令形式。

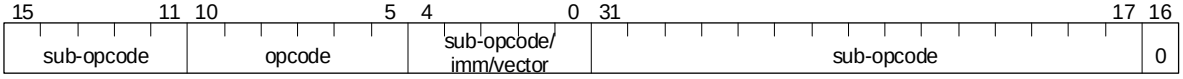
注意 拡張命令形式 1 では，汎用レジスタ指定フィールド，またはサブオペコード・フィールドの一部をシステム・レジスタ番号フィールド，条件コード・フィールド，イミーディエト・フィールド，ディスプレースメント・フィールドとして取り扱う場合があります。詳細は 5. 3 命令セットの各命令の説明を参照してください。



(10) 拡張命令形式2（Format X）

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

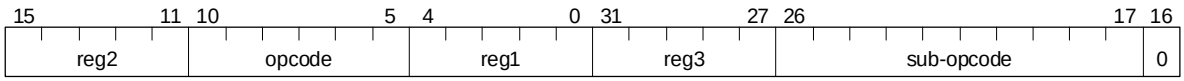
注意 拡張命令形式 2 では，汎用レジスタ指定フィールド，またはサブオペコード・フィールドの一部をシステム・レジスタ番号フィールド，条件コード・フィールド，イミーディエト・フィールド，ディスプレースメント・フィールドとして取り扱う場合があります。詳細は 5. 3 命令セットの各命令の説明を参照してください。



(11) 拡張命令形式3（Format XI）

6 ビットのオペコード・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オペコード・フィールドとして取り扱う 32 ビット長命令形式。

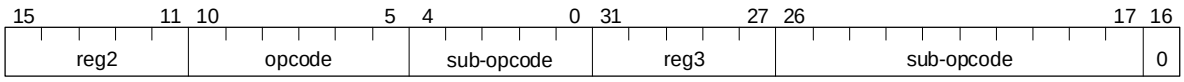
注意 拡張命令形式 3 では，汎用レジスタ指定フィールド，またはサブオペコード・フィールドの一部をシステム・レジスタ番号フィールド，条件コード・フィールド，イミーディエト・フィールド，ディスプレースメント・フィールドとして取り扱う場合があります。詳細は 5. 3 命令セットの各命令の説明を参照してください。



(12) 拡張命令形式4 (Format XII)

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ち，それ以外のビットをサブオペコード・フィールドとして取り扱う 32 ビット長命令形式。

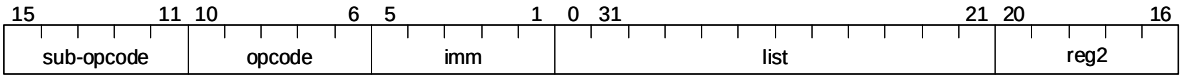
注意 拡張命令形式 4 では，汎用レジスタ指定フィールド，またはサブオペコード・フィールドの一部をシステム・レジスタ番号フィールド，条件コード・フィールド，イミーディエト・フィールド，ディスプレースメント・フィールドとして取り扱う場合があります。詳細は 5. 3 命令セットの各命令の説明を参照してください。



(13) スタック操作命令形式 (Format XIII)

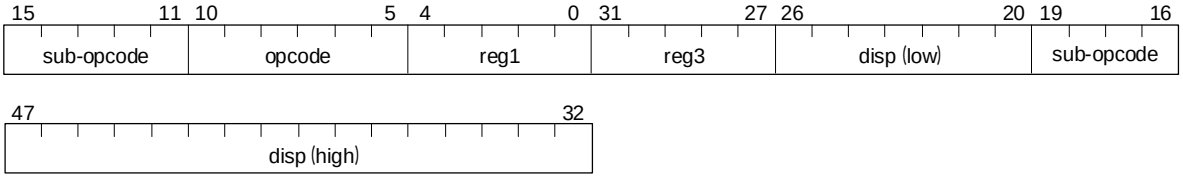
5 ビットのオペコード・フィールドと 5 ビットのイミーディエト・フィールド，12 ビットのレジスタ・リスト・フィールド，5 ビットのサブオペコード・フィールド，1 つの汎用レジスタ指定フィールド（または 5 ビットのサブオペコード・フィールド）を持つ 32 ビット長命令形式。

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



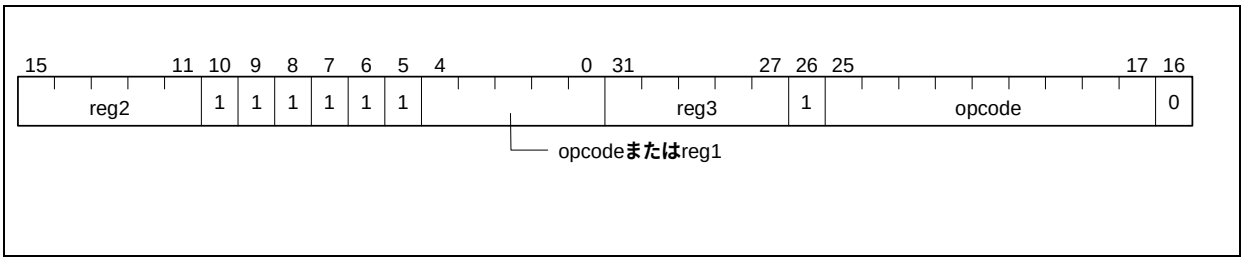
(14) ロード／ストア命令48ビット形式 (Format XIV)

6 ビットのオペコード・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ースメント・フィールドを持ち，それ以外のビットをサブオペコード・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。

V850E2Sでは，コプロセッサ命令は定義されていません。

(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。

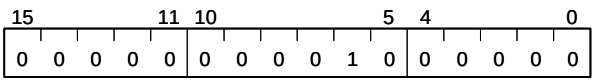
5. 1. 3　予約命令

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。

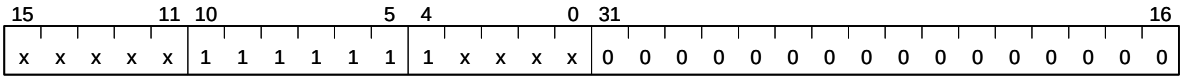
- 予約命令例外が発生する。
- いずれかの命令として動作する。

また，次のオペコードは V850E2S 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。

・RIE 命令（16 ビット長）



・RIE 命令（32 ビット長）



（x = 任意，0 または 1）

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

内蔵のハードウエア乗算器により，1 クロックでの乗算処理を行います。次の命令（ニモニック）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（ニモニック）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（ニモニック）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（ニモニック）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（ニモニック）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（ニモニック）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（ニモニック）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（ニモニック）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（ニモニック）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（ニモニック）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 ×	× ビット・イミーディエト・データ
disp ×	× 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 ×	ベクタを指定するデータ（×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の飽和処理を行う。 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	ディスプレースメントの1ビット分データ（ディスプレースメントの上位ビットを示す）
d	ディスプレースメントの1ビット分データ
I	イミーディエトの1ビット分データ（イミーディエトの上位ビットを示す）
i	イミーディエトの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

＜算術演算命令＞

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)

15															0
r	r	r	r	r	0	0	1	1	1	0	R	R	R	R	R

(2)

15															0
r	r	r	r	r	0	1	0	0	1	0	i	i	i	i	i

［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］ （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・イミーディエトを加算し，その結果を汎用レジスタ** reg2 **に格納します。**

＜算術演算命令＞

Add immediate

ADDI

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
rrrrr110000RRRRR	iiiiiiiiiiiiiiii

［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・イミーディエトを加算し，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。**

＜条件付き演算命令＞

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
rrrrr111111RRRRR	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	

＜論理演算命令＞

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。

＜論理演算命令＞

AND immediate

ANDI

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	iiiiiiiiiiiiiiii

［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・イミー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。

＜分岐命令＞

Branch on condition code with 9-bit displacement

Bcond

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
ddddd1011dddcccc	

ただし，dddddddd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。

＜データ操作命令＞

Byte swap half-word

BSH

ハーフワード・データのバイト・スワップ

［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
rrrrr11111100000		wwwww01101000010	

［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。**

＜データ操作命令＞

Byte swap word

BSW

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
rrrrr11111100000		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。**

＜特殊命令＞

Call with table look up

CALLT

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
0000001000iiiiii	

［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

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。
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ます。
3. メモリ保護（PSW.DMP = 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。

＜特殊命令＞

Compare and exchange for interlock

CAXI

比較と交換

［命令形式］ CAXI [reg1], reg2, reg3

［オペレーション］
```
adr    ← GR[reg1]注
token  ← Load-memory(adr, Word)
result ← GR[reg2] – token
If   result   == 0
then    Store-memory(adr, GR[reg3],Word)
        GR[reg3] ← token
else    Store-memory(adr, 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
rrrrr111111RRRRR	wwwww00011101110

［フ ラ グ］

CY	result の演算時に MSB へのボローがあれば 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	ー

［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。

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。

＜ビット操作命令＞

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］

（1）

15 0	31 16
10bbb111110RRRRR	ddddddddddddddddd

（2）

15 0	31 16
rrrrr111111RRRRR	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。

＜データ操作命令＞

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)	rrrrr111111RRRRR		wwwww011001cccc0	
(2)	rrrrr111111iiiii		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・イミーディエト・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 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。

＜算術演算命令＞

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）

15															0
r	r	r	r	r	0	0	1	1	1	1	R	R	R	R	R

（2）

15															0
r	r	r	r	r	0	1	0	0	1	1	i	i	i	i	i

［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］ （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・イミーディエトを比較し，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ー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。

＜特殊命令＞

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
0000011111100000	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。

＜特殊命令＞

DI	Disable interrupt EI **レベル・マスカブル例外の禁止**

［命令形式］ DI

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

［フォーマット］ Format X

［オペコード］

15	0	31	16
0000011111100000		0000000101100000	

［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・マスカ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＞

Function dispose

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) 注
        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) 注
        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

［オペコード］

	15 … 0	31 … 16
(1)	0000011001iiiiiL	LLLLLLLLLLL00000

	15 … 0	31 … 16
(2)	0000011001iiiiiL	LLLLLLLLLLLRRRRR

RRRRR ≠ 00000（reg1 には r0 を設定しないでください）

また，LLLLLLLLLLL 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：081FFFFFH

［フ ラ グ］

CY –
OV –
S –
Z –
SAT –

［説 明］

（1）5 ビット・イミーディエト・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・イミーディエト・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。

＜除算命令＞

Divide word

DIV

（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
rrrrr111111RRRRR	wwwww01011000000

［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ので，フラグは不定となります。

＜除算命令＞

Divide half-word

DIVH

（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］

（1）

15	0
rrrrr000010RRRRR	

RRRRR ≠ 00000（reg1 には r0 を設定しないでください）

rrrrr ≠ 00000（reg2 には r0 を設定しないでください）

（2）

15	0	31	16
rrrrr111111RRRRR		wwwww01010000000	

［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］

（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ので，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

＜除算命令＞

Divide half-word unsigned

DIVHU

（符号なし）ハーフワード・データの除算

［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
rrrrr111111RRRRR	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ので，フラグは不定となります。

＜高速除算命令＞

Divide word quickly

DIVQ

（符号付き）ワード・データの除算（可変ステップ）

［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となる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ので，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。

＜高速除算命令＞

Divide word unsigned quickly

DIVQU

（符号なし）ワード・データの除算（可変ステップ）

［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
rrrrr111111RRRRR	wwwww01011111110

［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となる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ので，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。

＜除算命令＞

DIVU

Divide word unsigned

（符号なし）ワード・データの除算

［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
rrrrr111111RRRRR	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ので，フラグは不定となります。

＜特殊命令＞

EI	Enable interrupt EI レベル・マスカブル例外の許可

［命令形式］ EI

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

［フォーマット］ Format X

［オペコード］

15	0	31	16
1000011111100000		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・マスカブル例外の受け付けを許可します。

＜特殊命令＞

Return from trap or interrupt

EIRET

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
0000011111100000	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。

＜特殊命令＞

Return from trap or interrupt

FERET

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
0000011111100000	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＞

FE-level Trap

FETRAP

FE レベル・ソフトウエア例外

［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
PC ← 00000030H

［フォーマット］ Format I

［オペコード］

15	0
0vvvv00001000000	

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.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。

＜特殊命令＞

	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
0000011111100000	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-INT127）
- 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）
- 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-INT127, 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-EIC127）
- EI レベル割り込みマスクレジスタ（IMR0-IMR15）

＜データ操作命令＞

Half-word swap half-word

HSH

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
rrrrr11111100000		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。**

＜データ操作命令＞

Half-word swap word

HSW

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
rrrrr11111100000		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。**

＜分岐命令＞

Jump and register link

JARL

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

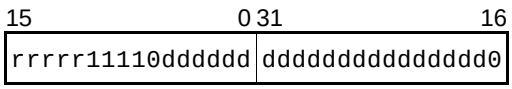
PC ← PC + disp32

［フォーマット］ （1）Format V

（2）Format VI

［オペコード］

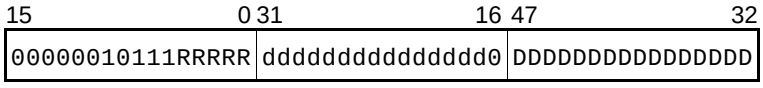
（1）



ただし，ddddddddddddddddddddd **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）**

（2）



ただし，DDDDDDDDDDDDDDDDddddddddddddddd **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。

＜分岐命令＞

Jump register

JMP

無条件分岐（レジスタ間接）

［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］

（1）

15	0
00000000011RRRRR	

（2）

15	0	31	16	47	32
00000110111RRRRR		ddddddddddddddd0		DDDDDDDDDDDDDDDD	

ただし，DDDDDDDDDDDDDDDDddddddddddddddd **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。**

＜分岐命令＞

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
0000011110ddddd		ddddddddddddddd0	

ただし，ddddddddddddddddddddd は disp22 の上位 21 ビットです。

（2）

15	0	31	16	47	32
0000001011100000		ddddddddddddddd0		DDDDDDDDDDDDDDDD	

ただし，DDDDDDDDDDDDDDDDddddddddddddddddd 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。

＜ロード命令＞

Load byte

LD.B

（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］

（1）

15	031	16
rrrrr111000RRRRR	ddddddddddddddddd	

（2）

15	031	1647	32
00000111100RRRRR	wwwwwddddddd0101	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，ワード長まで符号拡張し，汎用レジスタ** reg3 **に格納します。**

＜ロード命令＞

Load byte unsigned

LD.BU

（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]

（1）

15	0	31	16
rrrrr11110bRRRRR		ddddddddddddddd1	

ただし，ddddddddddddddd **は** disp16 **の上位** 15 **ビット，**b **は** disp16 **のビット** 0 **です。**

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

（2）

15	0	31	16	47	32
000001111101RRRRR		wwwwwddddddd0101		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，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，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。

＜ロード命令＞

Load half-word

LD.H

（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	0	31	16
rrrrr111001RRRRR		ddddddddddddddd0	

ただし，ddddddddddddddd **は** disp16 **の上位** 15 **ビットです。**

（2）

15	0	31	16	47	32
00000111100RRRRR		wwwwwddddd00111		DDDDDDDDDDDDDDDD	

ただし，RRRRR = reg1, wwwww = reg3 **です。**

ddddd **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。**

＜ロード命令＞

Load half-word unsigned

LD.HU

（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	0	31	16
rrrrr111111RRRRR		ddddddddddddddd1	

ただし，ddddddddddddddd **は** disp16 **の上位** 15 **ビットです。**

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

（2）

15	0	31	16	47	32
000001111101RRRRR		wwwwwddddddd00111		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,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。

＜ロード命令＞

Load word

LD.W

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］

（1）

15	0	31	16
rrrrr111001RRRRR		ddddddddddddddd1	

ただし，ddddddddddddddd **は** disp16 **の上位** 15 **ビットです。**

（2）

15	0	31	16	47	32
00000111100RRRRR		wwwwwddddd01001		DDDDDDDDDDDDDDDD	

ただし，RRRRR = reg1, wwwww = reg3 **です。**

ddddd **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ワード・データを読み出し，汎用レジスタ** reg3 **に格納します。**

＜特殊命令＞

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，ニモニック記述の都合上，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，ニモニッ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。

＜加算付き乗算命令＞

Multiply and add word

MAC

（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	wwww0011110mmmm0

［フ ラ グ］

CY –
OV –
S –
Z –
SAT –

［説 明］ **汇用レジスタ** reg2 **のワード・データに汇用レジスタ** reg1 **のワード・データを乗算した結果（**64 **ビット・データ）と，汇用レジスタ** reg3 **を下位** 32 **ビットとして，汇用レジスタ** 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 **ビットを汇用レジスタ** reg4+1 **に，下位** 32 **ビットを汇用レジスタ** reg4 **に格納します。**
汇用レジスタ reg1, reg2 **の内容を** 32 **ビットの符号付き整数として扱います。**
汇用レジスタ reg1, reg2, reg3, reg3+1 **は影響を受けません。**

注意 reg3，または reg4 に指定できる汇用レジスタは，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。

＜加算付き乗算命令＞

Multiply and add word unsigned

MACU

（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	wwww0011111mmmm0

［フ ラ グ］

CY –
OV –
S –
Z –
SAT –

［説 明］ **汇用レジスタ** reg2 **のワード・データに汇用レジスタ** reg1 **のワード・データを乗算した結果（**64 **ビット・データ）と，汇用レジスタ** reg3 **を下位** 32 **ビットとして，汇用レジスタ** 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 **ビットを汇用レジスタ** reg4+1 **に，下位** 32 **ビットを汇用レジスタ** reg4 **に格納します。**

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

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

注意 reg3，または reg4 に指定できる汇用レジスタは，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。

＜算術演算命令＞

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
rrrrr000000RRRRR	

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

（2）

15	0
rrrrr010000iiiii	

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

（3）

15	0	31	16	47	32
00000110001RRRRR		iiiiiiiiiiiiiiii		IIIIIIIIIIIIIIII	

i（**ビット** 31-16）**は** 32 **ビット・イミーディエト・データの下位** 16 **ビットです。**

I（**ビット** 47-32）**は** 32 **ビット・イミーディエト・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・イミー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・イミーディエトを，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。

＜算術演算命令＞

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		iiiiiiiiiiiiiiii	

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・イミーディエトを加算し，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。

＜算術演算命令＞

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^{16})

［フォーマット］ Format VI

［オペコード］

15	0	31	16
rrrrr110010RRRRR		iiiiiiiiiiiiiiii	

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，上位 16 ビットが 16 ビット・イミーディエト，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。

注意 reg2 には，r0 を指定しないでください。

＜乗算命令＞

Multiply word by register/immediate (9-bit)

MUL

（符号付き）ワード・データの乗算

［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）

15 0	31 16
rrrrr111111RRRRR	wwwww01000100000

（2）

15 0	31 16
rrrrr111111iiiii	wwwww01001IIII00

iiii は，9 ビット・イミーディエト・データの下位 5 ビットです。

IIII は，9 ビット・イミーディエト・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・イミーディエト・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。

＜乗算命令＞

Multiply half-word by register/immediate (5-bit)

MULH

（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］

15 0

（1）`rrrrr000111RRRRR`

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

15 0

（2）`rrrrr010111iiiii`

`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］ （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。

＜乗算命令＞

Multiply half-word by immediate (16-bit)

MULHI

（符号付き）ハーフワード・イミーディエトの乗算

[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
rrrrr110111RRRRR		iiiiiiiiiiiiiiii	

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 **の下位ハーフワード・データに，**16 **ビット・イミ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。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。

＜乗算命令＞

Multiply word unsigned by register/immediate (9-bit)

MULU

（符号なし）ワード・データの乗算

［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）

15 0	31 16
rrrrr111111RRRRR	wwwww01000100010

（2）

15 0	31 16
rrrrr111111iiiii	wwwww01001IIII10

iiiii **は，**9 **ビット・イミーディエト・データの下位** 5 **ビットです。**

IIII **は，**9 **ビット・イミーディエト・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・イミーディエト・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。**

＜特殊命令＞

NOP	No operation オペレーションなし

［命令形式］ NOP

［オペレーション］ 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
0000000000000000	

［フ ラ グ］

CY	–
OV	–
S	–
Z	–
SAT	–

［説 明］ 何もせず PC を+2 します。

［補 足］ オペコードは「MOV r0, r0」と同一になります。

＜論理演算命令＞

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
rrrrr000001RRRRR	

［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。

＜ビット操作命令＞

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］

（1）

15 0	31 16
01bbb111110RRRRR	ddddddddddddddddd

（2）

15 0	31 16
rrrrr111111RRRRR	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。

＜論理演算命令＞

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
rrrrr001000RRRRR	

［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。**

＜論理演算命令＞

OR immediate (16-bit)

ORI

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
rrrrr110100RRRRR	iiiiiiiiiiiiiiii

［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・イミー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。**

＜特殊命令＞

Function prepare

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 注

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 ) 注
}
sp ← adr － zero-extend (imm5 logically shift left by 2)
```

（2）adr ← sp

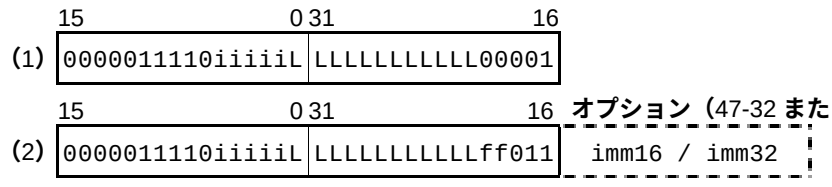
```
foreach (all regs in list12) {
        adr ← adr － 4
        Store-memory (adr, GR[reg in list12], Word) 注
}
sp ← adr － zero-extend (imm5 logically shift left by 2)

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］



32 **ビット・イミーディエト・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・イミーディエト・データ（ビット 63-32）を ep にロード

また，LLLLLLLLLLL 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：081FFFFFH

［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。

＜特殊命令＞

Return from trap or interrupt

RETI

EI レベル・ソフトウエア例外または割り込みからの復帰

［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
0000011111100000	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・ソフトウエ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・ソフトウエ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・ノンマスカブル割り込み例外（FENMI），FE レベル・マスカブル割り込み例外（FEINT），または EI レベル・ソフトウエア例外（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・ノンマスカブル割り込み例外（FENMI）からの復帰時：NP = 1 かつ EP = 0**
- **RETI 命令による FE レベル・マスカブル割り込み例外（FEINT）からの復帰時：NP = 1 かつ EP = 0**
- **RETI 命令による EI レベル・ソフトウエア例外（TRAP）からの復帰時：EP = 1**

＜特殊命令＞

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
PC ← 00000030H

［フォーマット］ （1）Format I

（2）Format X

［オペコード］

（1）

15	0
0000000001000000	

（2）

15	0	31	16
iiiii1111111IIII		0000000000000000	

ただし，iiiii は imm5 です。
IIII 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.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。

＜データ操作命令＞

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]

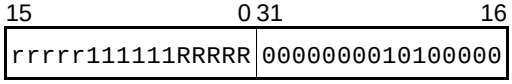
［フォーマット］ （1）Format IX

（2）Format II

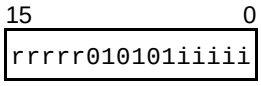
（3）Format XI

［オペコード］

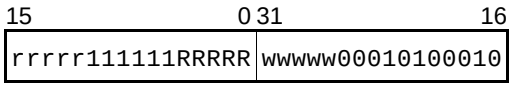
(1)



(2)



(3)



［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 ー

［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・イミーディエ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。

＜データ操作命令＞

SASF

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
rrrrr1111110cccc		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。**

＜飽和演算命令＞

Saturated add register/immediate (5-bit)

SATADD

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］

15 0

(1) `rrrrr000110RRRRR`

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

15 0

(2) `rrrrr010001iiiii`

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

15 0 31 16

(3) `rrrrr111111RRRRR` `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・イミーディエトを加算し，その結果を汎用レジスタ reg2 に格納します。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。

＜飽和演算命令＞

Saturated subtract

SATSUB

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］

（1）

15	0
rrrrr000101RRRRR	

rrrrr ≠ 00000（reg2 には r0 を設定しないでください）

（2）

15	0	31	16
rrrrr111111RRRRR		wwwww01110011010	

［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］

（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。

＜飽和演算命令＞

Saturated subtract immediate

SATSUBI

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
rrrrr110011RRRRR		iiiiiiiiiiiiiiii	

rrrrr ≠ 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・イミーディエトを減算し，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。

＜飽和演算命令＞

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
rrrrr000100RRRRR	

rrrrr ≠ 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。

＜条件付き演算命令＞

Subtract on condition flag

SBF

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
rrrrr111111RRRRR		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	

＜ビット・サーチ命令＞

Search zero from left

SCH0L

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
rrrrr11111100000		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 ー

［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。**

＜ビット・サーチ命令＞

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
rrrrr11111100000		wwwww01101100000	

［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。**

＜ビット・サーチ命令＞

Search one from left

SCH1L

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
rrrrr11111100000	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 ー

［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。**

＜ビット・サーチ命令＞

Search one from right

SCH1R

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
rrrrr11111100000	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。**

＜ビット操作命令＞

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］

（1）

15 0	31 16
00bbb111110RRRRR	ddddddddddddddddd

（2）

15 0	31 16
rrrrr111111RRRRR	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＞

SETF

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
rrrrr1111110cccc		0000000000000000	

［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（a_1, a_2, a_3, …）から成り立つとき，通常は if（a_1）then, if（a_2）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 a_n 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（a_n）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 Ra_n 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，下位からの桁上がりを数値として表現できます。

＜データ操作命令＞

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
rrrrr111111RRRRR		0000000011000000	

(2)

15	0
rrrrr010110iiiii	

(3)

15	0	31	16
rrrrr111111RRRRR		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	ー

［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・イミーディエ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。

(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。

＜データ操作命令＞

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
rrrrr111111RRRRR		0000000010000000	

(2)

15	0
rrrrr010100iiiii	

(3)

15	0	31	16
rrrrr111111RRRRR		wwwww00010000010	

［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・イミーディエ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。

＜ロード命令＞

SLD.B

Short format load byte

（符号付き）バイト・データのロード

［命令形式］ SLD.B disp7 [ep] , reg2

［オペレーション］ adr ← ep + zero-extend (disp7)

GR [reg2] ← sign-extend (Load-memory (adr, Byte))

［フォーマット］ Format IV

［オペコード］

15	0
rrrrr0110ddddddd	

［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。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，reg2 に格納します。

＜ロード命令＞

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
rrrrr0000110dddd	

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］ 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。

＜ロード命令＞

Short format load half-word

SLD.H

（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
rrrrr1000ddddddd	

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，ワード長まで符号拡張し，reg2 に格納します。

＜ロード命令＞

Short format load half-word unsigned

SLD.HU

（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
rrrrr0000111dddd	

rrrrr ≠ 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。

＜ロード命令＞

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
rrrrr1010dddddd0	

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・アドレスからワード・データを読み出し，**reg2 **に格納します。**

＜ストア命令＞

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
rrrrr0111ddddddd	

［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。**

＜ストア命令＞

Short format store half-word

SST.H

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
rrrrr1001ddddddd	

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。**

＜ストア命令＞

Short format store word

SST.W

ワード・データのストア

［命令形式］ SST.W reg2, disp8 [ep]

［オペレーション］ adr ← ep + zero-extend (disp8)

Store-memory (adr, GR [reg2] , Word)

［フォーマット］ Format IV

［オペコード］

15	0
rrrrr1010dddddd1	

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。**

＜ストア命令＞

Store byte

ST.B

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］

（1）

15	0	31	16
rrrrr111010RRRRR		ddddddddddddddd	

（2）

15	0	31	16	47	32
00000111100RRRRR		wwwwwddddddd1101		DDDDDDDDDDDDDDD	

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・ディスプレースメントを加算して** 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・ディスプレースメントを加算して** 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。**

＜ストア命令＞

Store half-word

ST.H

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）

15	0	31	16
rrrrr111011RRRRR		ddddddddddddddd0	

ただし，ddddddddddddddd**は**disp16**の上位**15**ビットです。**

（2）

15	0	31	16	47	32
00000111101RRRRR		wwwwwddddd01101		DDDDDDDDDDDDDDDD	

ただし，RRRRR = reg1, wwwww = reg3 **です。**

ddddd **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・ディスプレースメントを加算して** 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・ディスプレースメントを加算して** 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。**

＜ストア命令＞

Store word

ST.W

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	0	31	16
rrrrr111011RRRRR		ddddddddddddddd1	

ただし，ddddddddddddddd**は**disp16**の上位**15**ビットです。**

（2）

15	0	31	16	47	32
00000111100RRRRR		wwwwwddddd01111		DDDDDDDDDDDDDDDD	

ただし，RRRRR = reg1, wwwww = reg3 **です。**

ddddd **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・ディスプレースメントを加算して** 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・ディスプレースメントを加算して** 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。**

＜特殊命令＞

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
rrrrr111111RRRRR	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。

＜算術演算命令＞

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** 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。**

＜算術演算命令＞

SUBR

Subtract reverse

逆減算

［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
rrrrr001100RRRRR	

［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。**

＜特殊命令＞

Jump with table look up

SWITCH

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・エントリ・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・エントリ・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す。

3. メモリ保護（PSW.DMP = 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。

＜データ操作命令＞

Sign extend byte

SXB

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。**

＜データ操作命令＞

Sign extend half-word

SXH

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。**

＜特殊命令＞

Synchronize exceptions

SYNCE

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（synchronaization）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は，必ずこの命令の実行完了前に受け付けられます。

ただし，V850E2S CPU ではインプレサイス例外は発生しません。SYNCE 命令には対応しますが，実行の開始を待ち合わせる例外要因がないため，SYNCM 命令に置き換えて実行されます。

＜特殊命令＞

SYNCM

Synchronize memory

メモリ同期化命令

［命令形式］ SYNCM

［オペレーション］ メモリ・デバイス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
0000000000011110	

［フ ラ グ］

CY –
OV –
S –
Z –
SAT –

［説 明］ 先行するすべてのメモリ・アクセスが“同期化（synchronaization）”されるまで実行の開始を待ち合わせます。
“同期化”とは，システム内のいずれのマスタ・デバイスから参照しても，SYNCM 命令に先行するメモリ・アクセスの結果が参照できる状態を指します。
バッファリング等によってメモリ・アクセスを遅延していて“同期化”されていない場合，SYNCM 命令は完了せず，“同期化”されるまで待ち合わせます。SYNCM 命令の実行が完了するまで，後続の命令は実行されません。

＜特殊命令＞

SYNCP	Synchronize pipline 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
0000000000011111	

［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。

＜特殊命令＞

System call

SYSCALL

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
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.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）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。

＜特殊命令＞

TRAP	Trap **ソフトウエア例外**

［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**（40H-5FH）
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
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.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。**

＜論理演算命令＞

Test

TST

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。**

＜ビット操作命令＞

Test bit

TST1

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］

（1）

15	0	31	16
11bbb111110RRRRR		ddddddddddddddddd	

（2）

15	0	31	16
rrrrr111111RRRRR		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。**

＜論理演算命令＞

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	iiiiiiiiiiiiiiii

［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・イミー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。

＜データ操作命令＞

Zero extend byte

ZXB

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。**

＜データ操作命令＞

Zero extend half-word

ZXH

ハーフワード・データのゼロ拡張

［命令形式］ ZXH reg1

［オペレーション］ GR [reg1] ← zero-extend (GR [reg1] (15:0))

［フォーマット］ Format I

［オペコード］

15	0
00000000110RRRRR	

［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　**V850E2S 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

V850E2S 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（×：0または1）		例外要因コード[注1]	復帰PC[注1]	レジスタ更新値（s：保持）					復帰命令
								PSW				ハンドラ・オフセット[注2]	PSW				
								ID	NP				実行レベル[注3]	NP	EP	ID	
CPU初期化	RESET	リセット入力	1	ー	非同期	不可	不可	×	×	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[注4]	3	FE	割り込み	不可	不可	×	×	00000020H	currentPC	+0020H	注5	1	0	1	FERET
システム・エラー例外	SYSERR	SYSERR入力（4要因）	4	FE	注6	不可	不可	×	×	00000230H ： 00000233H	currentPC	+0030H	注5	1	1	1	FERET
FEレベル・マスカブル割り込み	FEINT	FEINT入力[注4]	7	FE	割り込み	可能	可能	×	0	00000010H	currentPC	+0010H	注5	1	0	1	FERET
EIレベル・マスカブル割り込み	INT	INTn入力[注4]（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. INTC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. 要因ごとに非同期になります。製品の実装に依存します。

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） PSW ID	受け付け条件（×：0または1） PSW NP	例外要因コード[注1]	復帰PC[注1]	レジスタ更新値（s：保持） ハンドラ・オフセット[注2]	レジスタ更新値（s：保持） PSW 実行レベル[注3]	レジスタ更新値（s：保持） PSW NP	レジスタ更新値（s：保持） PSW EP	レジスタ更新値（s：保持） PSW ID	復帰命令
実行保護例外	MIP	実行保護違反	11	FE	プレサイス	可能[注4]	可能[注4]	×	×	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[注5]	12	FE	プレサイス	不可能[注4]	不可能[注4]	×	×	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[注7]	FE	プレサイス	可能[注4]	可能[注4]	×	×	00000431H	currentPC	+0030H	0	1	1	1	FERET
コプロセッサ使用不可例外	UCPOP	コプロセッサ命令		FE	プレサイス	可能[注4]	可能[注4]	×	×	00000530H 00000537H	currentPC	+0030H	注6	1	1	1	FERET
予約命令例外	RIE	予約命令		FE	プレサイス	可能[注4]	可能[注4]	×	×	00000130H	currentPC	+0030H	注6	1	1	1	FERET
FEレベル・ソフトウエア例外	FETRAP	FETRAP命令（vector = 1H-FH）		FE	プレサイス	可能[注4]	可能[注4]	×	×	00000031H 0000003FH	nextPC	+0030H	注6	1	1	1	FERET
EIレベル・ソフトウエア例外	TRAP	TRAP0n命令（vector = 00-0FH）		EI	プレサイス	可能[注4]	可能[注4]	×	×	00000040H 0000004FH	nextPC	+0040H	注6	s	1	1	EIRET
EIレベル・ソフトウエア例外	TRAP	TRAP1n命令（vector = 10H-1FH）		EI	プレサイス	可能[注4]	可能[注4]	×	×	00000050H 0000005FH	nextPC	+0050H	注6	s	1	1	EIRET
システム・コール例外	SYSCALL	SYSCALL命令（vector = 00H-FFH）		EI	プレサイス	可能[注4]	可能[注4]	×	×	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

V850E2S CPUでは，例外を発生タイミングや性質によって，次の3つの種類に分類します。

- プレサイス例外
- 非同期例外
- 割り込み

(1) プレサイス例外

ソフトウエ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[注]。

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
- コプロセッサ使用不可例外
- 予約命令例外
- FEレベル・ソフトウエア例外
- EIレベル・ソフトウエア例外
- 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) 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）

(3) 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。

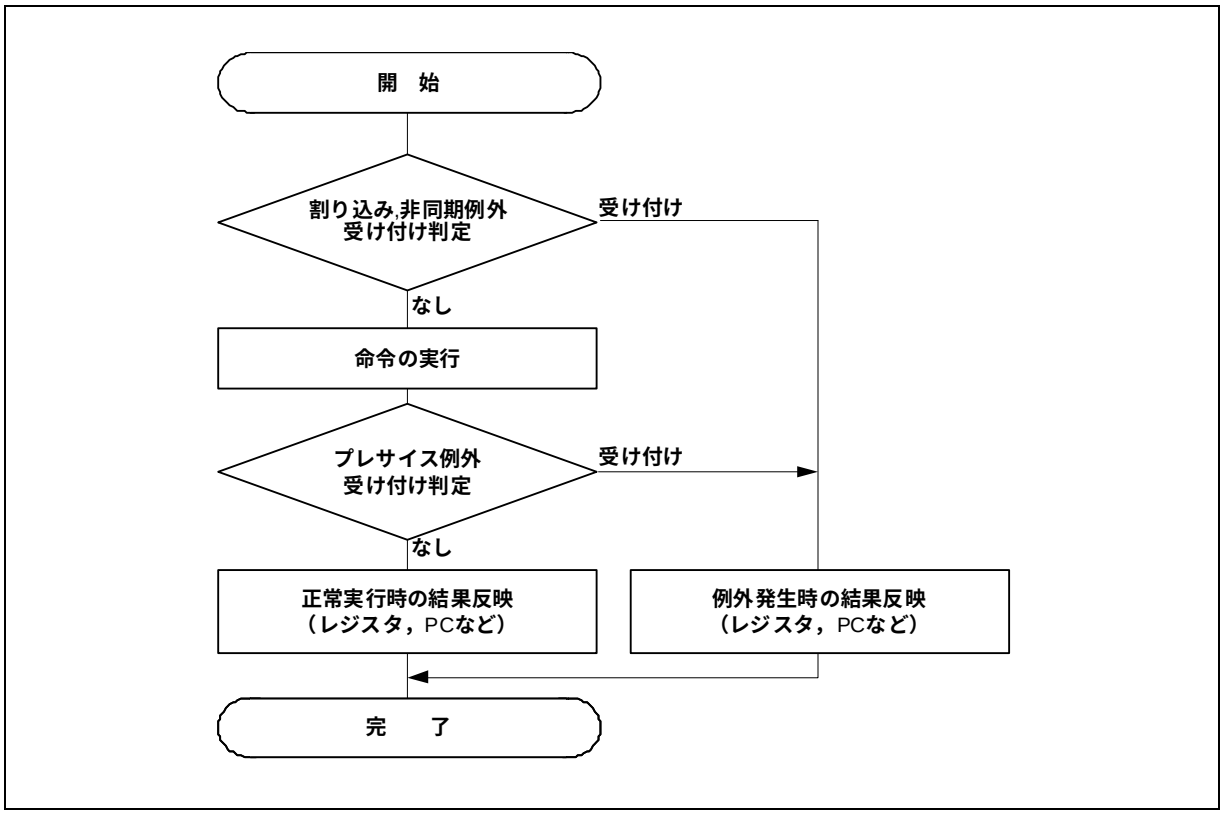
- FEレベル・ノンマスカブル割り込み
- FEレベル・マスカブル割り込み
- EIレベル・マスカブル割り込み

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-INT127, 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るため，復帰PCは現在の命令（Current PC）を格納します。

一方，プレサイス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，プレサイス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，ソフトウエア・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 ↑	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）	
	FEレベル・マスカブル割り込み（FEINT）	
	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）注	
	コプロセッサ使用不可例外（UCPOP）注	
	予約命令例外（RIE）注	
	FEレベル・ソフトウエア例外（FETRAP）注	
	EIレベル・ソフトウエア例外（TRAP）注	
↓ 低い	システム・コール例外（SYSCALL）注	

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

V850E2S 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[注]
EIレベル・マスカブル割り込み EIレベル・ソフトウエア例外 システム・コール例外	システム・エラー例外 FEレベル・マスカブル割り込み FEレベル・ノンマスカブル割り込み FEレベル・ソフトウエア例外 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 コプロセッサ使用不可例外	デバッグ例外[注]

注 DBレベル例外は開発ツ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[注]	DBPC[注], DBPSW[注], DBIC[注], DBWR[注]

注 DBレベル例外は開発ツール向けのデバッグ機能で使用します。

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。

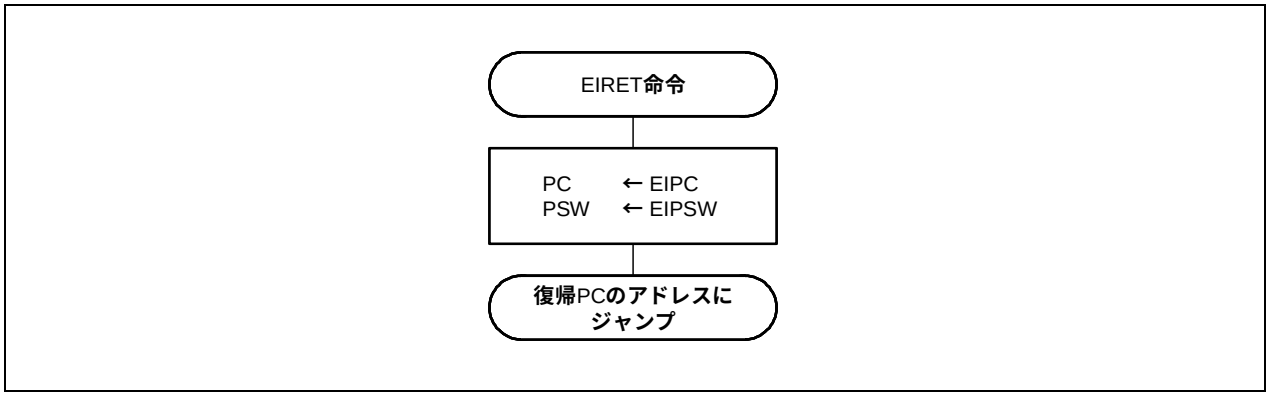
<1> EIPC, EIPSWレジスタから復帰PC, PSWを取り出します。

<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。

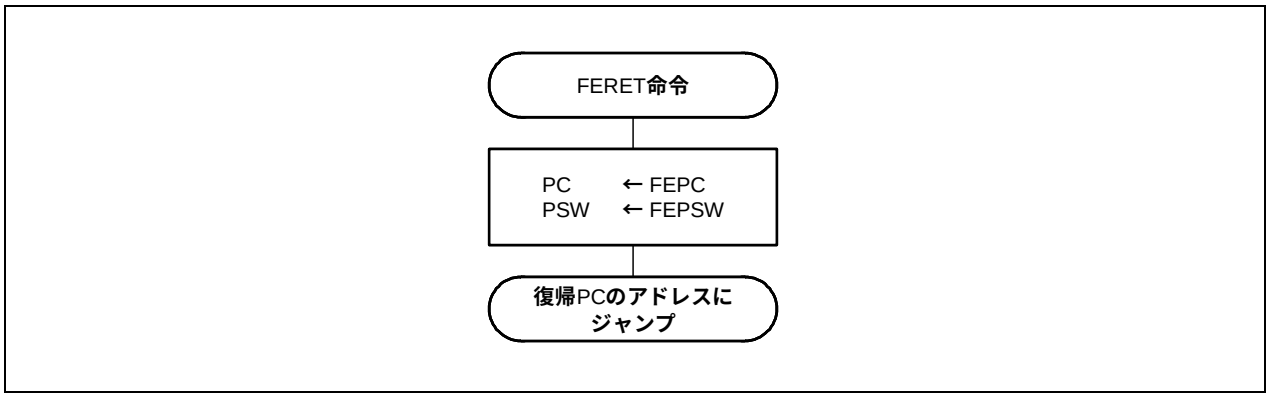
FERET命令の実行により，CPUは次の処理を行い復帰PCのアドレスへ制御を移します。

<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・ソフトウエア例外（TRAP）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・ソフトウエア例外（TRAP）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・ソフトウエア例外（TRAP）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。

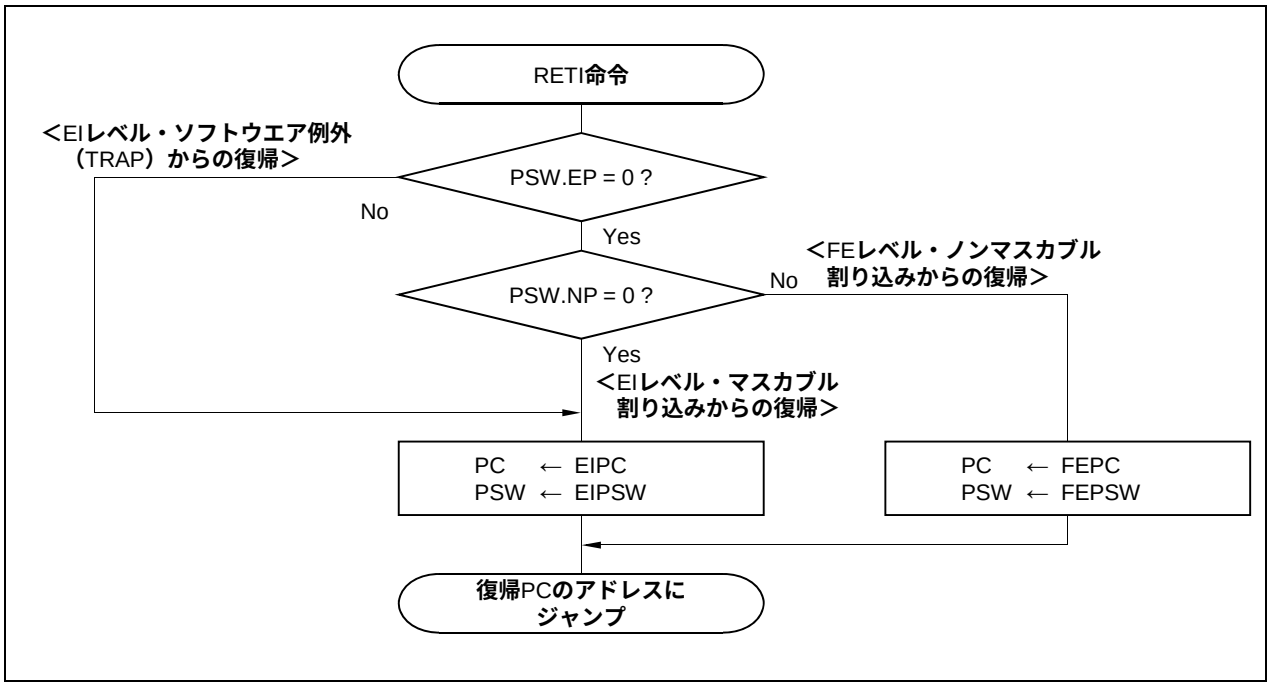
各例外処理からの復帰時は，PC, PSWを正常にリストアするために，RETI命令の直前で，LDSR命令を使用し，PSW.NPビット，PSW.EPビットの各フラグを次の状態にしておく必要があります。

- FEレベル・マスカブル割り込み処理からの復帰時[注] ：PSW.NPビット= 1，PSW.EPビット = 0
- EIレベル・マスカブル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・ソフトウエア例外（TRAP）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す。

RETI命令による復帰の処理形態を次に示します。

図6ー3 RETI命令


RETI命令
＜EIレベル・ソフトウエア例外
（TRAP）からの復帰＞
PSW.EP = 0 ?
No
Yes
＜FEレベル・ノンマスカブル
割り込みからの復帰＞
PSW.NP = 0 ?
No
Yes
＜EIレベル・マスカブル
割り込みからの復帰＞
PC ← EIPC
PSW ← EIPSW
PC ← FEPC
PSW ← FEPSW
復帰PCのアドレスに
ジャンプ

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注。
<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.NPV, DMP, IMPビットをクリア（0）します。それ以外の場合は，PSW.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。

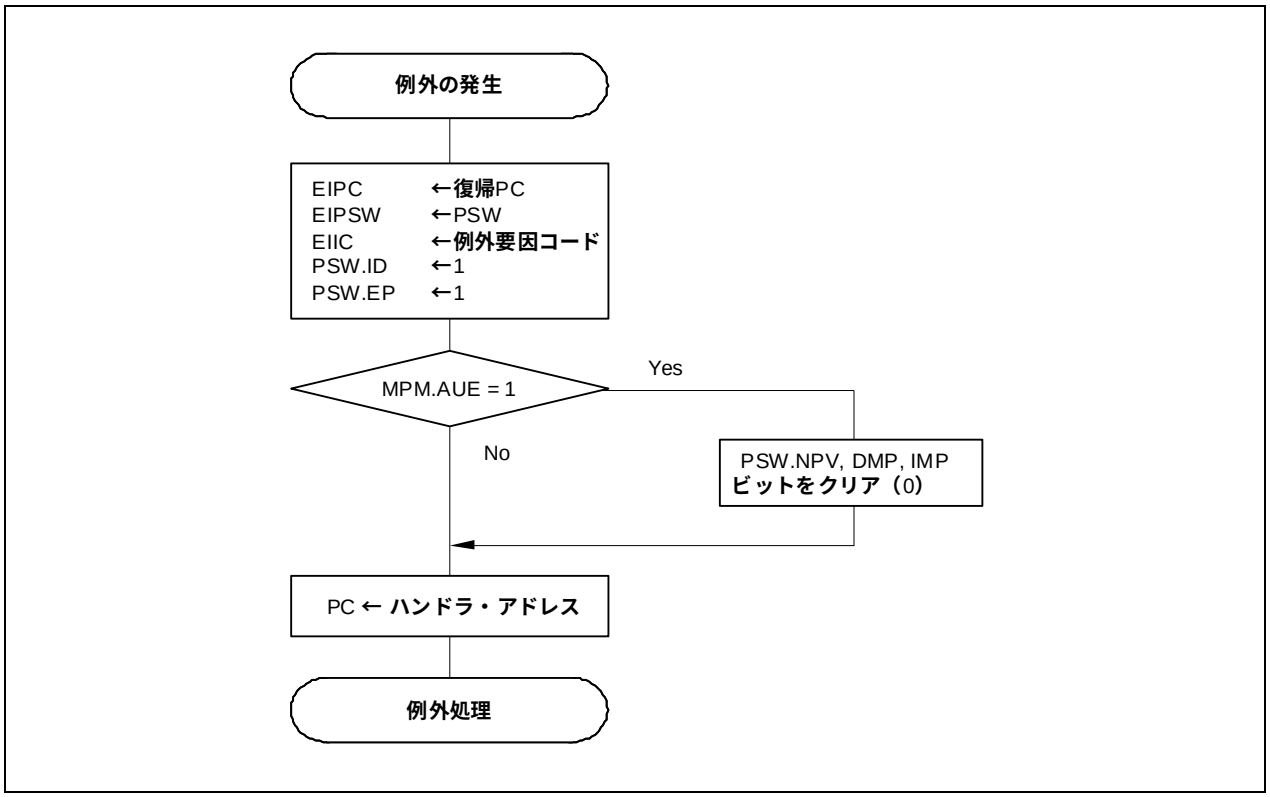
注 ECRレジスタの下位16ビット（EICC）にも例外要因コードが書き込まれますが，修正の不可能な既存プログラム以外はEIICレジスタを使用してください。

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[注]。
<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.NPV, DMP, IMPビットをクリア（0）します。それ以外の場合は，PSW.NPV, DMP, IMP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。

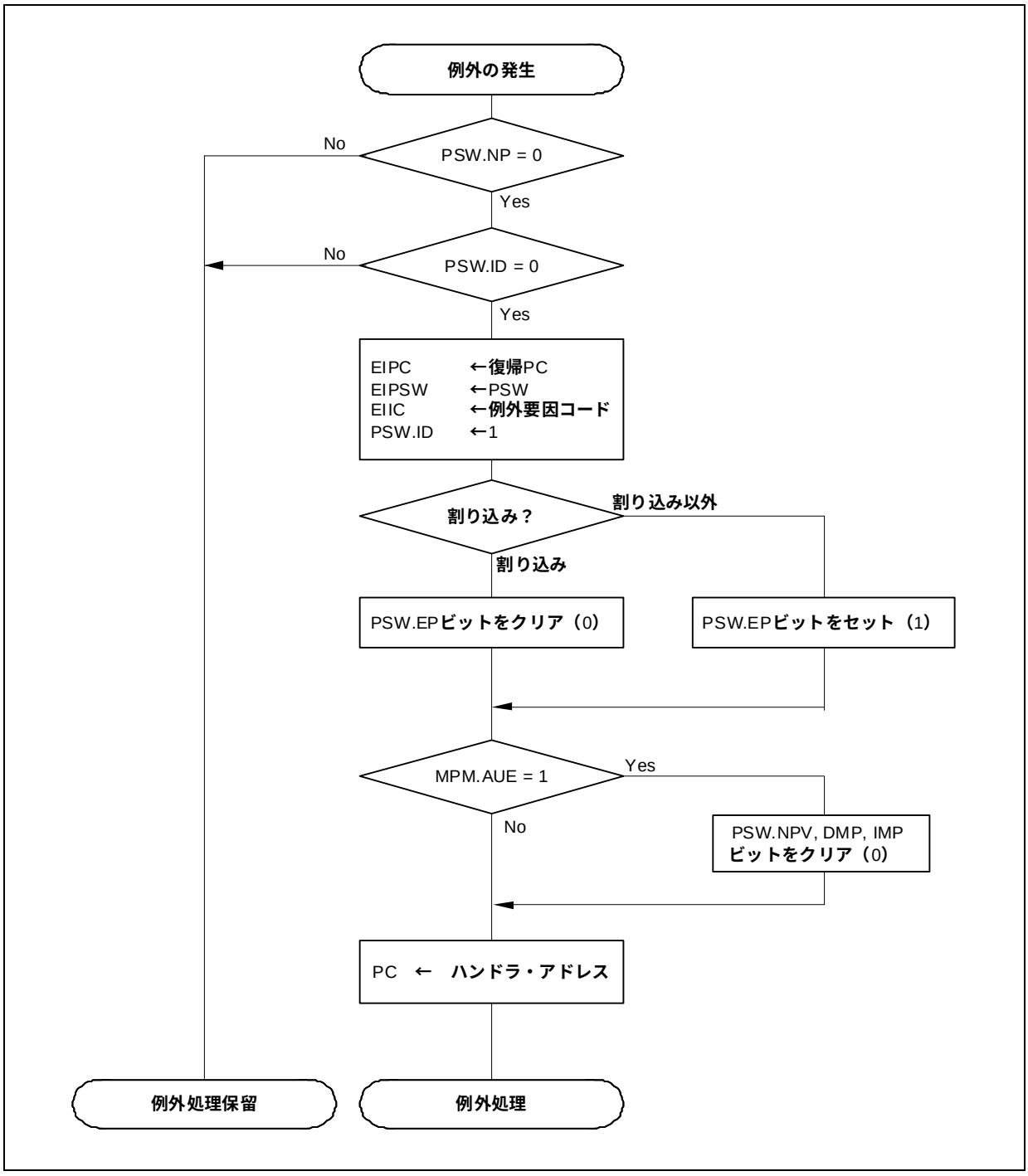
注 ECRレジスタの下位16ビット（EICC）にも例外要因コードが書き込まれますが，修正の不可能な既存プログラム以外はEIICレジスタを使用してください。

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FE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> 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[注1]。

<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.NPV, DMP, IMPビットをクリア（0）します。
それ以外の場合は，PSW.NPV, DMP, IMPは更新しません[注2]。

<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。

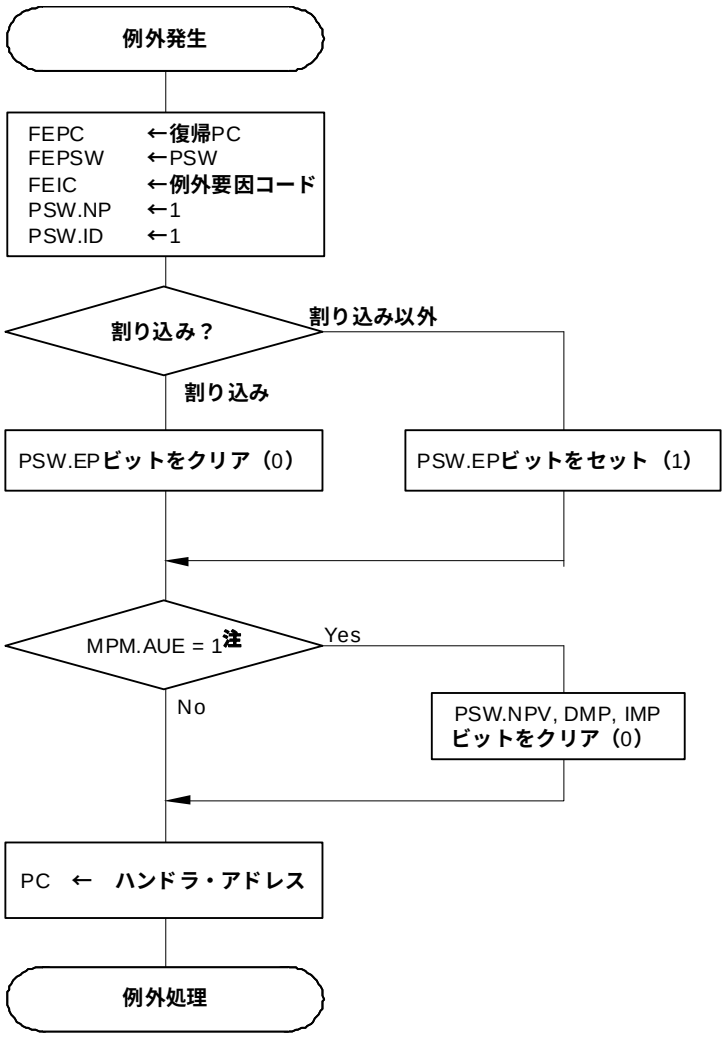
2. プロセッサ保護に関する例外（MDP例外，MIP例外）の場合は，PSW.NPV, DMP, IMPは常にクリア（0）します。

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FE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FE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



注 プロセッサ保護例外（MDP, MIP）の場合は，MPM.AUEの状態にかかわらず，常にPSW.NPV, DMP, IMPビットをクリア（0）します。

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[注1]。
<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.NPV, DMP, IMPビットをクリア(0)します。
それ以外の場合は，PSW.NPV, DMP, IMPは更新しません。
<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。

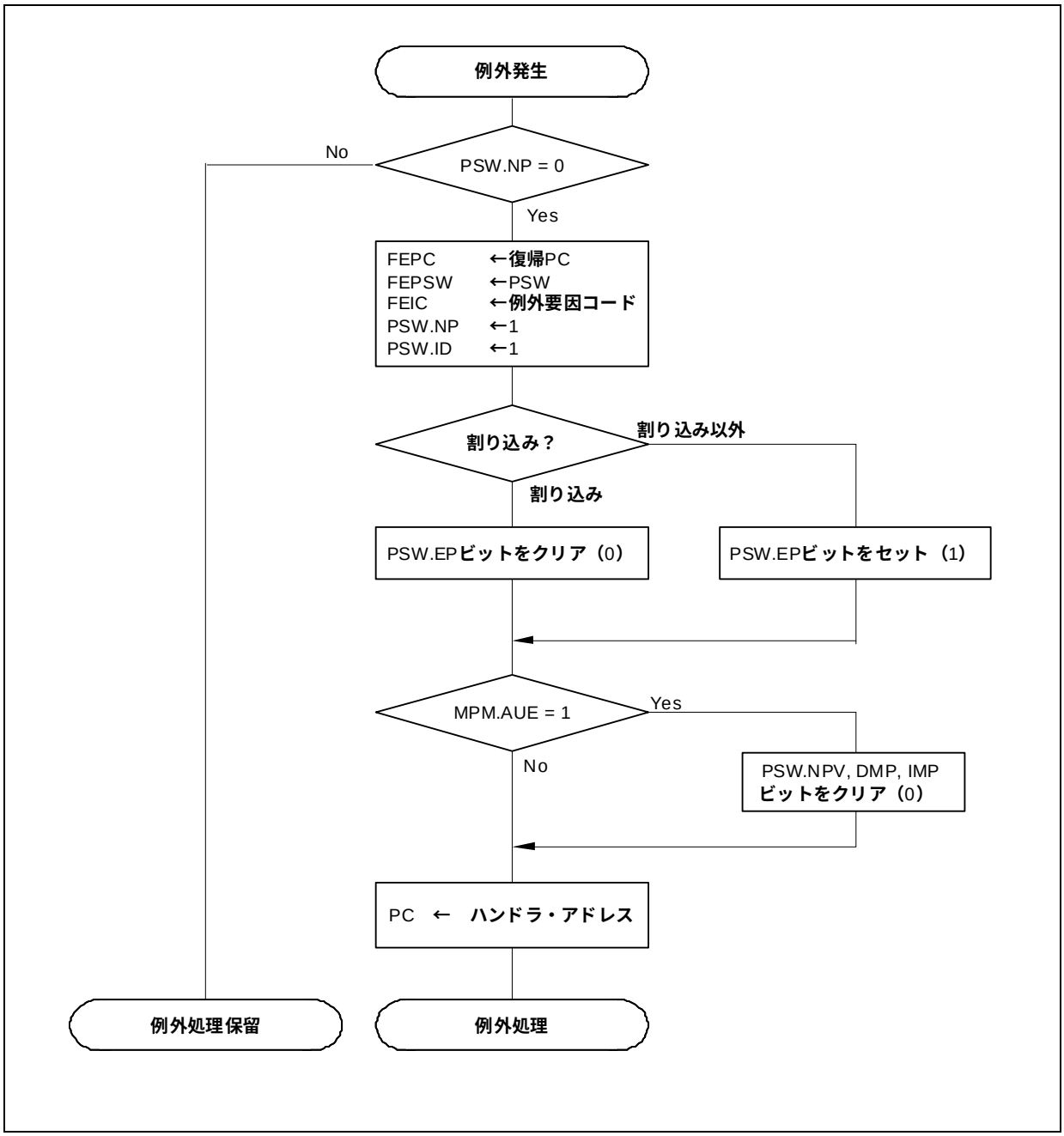
注1 ECRレジスタの下位16ビット（FECC）にも例外要因コードが書き込まれますが，修正の不可能な既存プログラム以外はFEICレジスタを使用してください。

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NPV, DMP, IMPビット

プロセッサ保護例外を受け付けた場合，PSW.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.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.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（RIE）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＋1～255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＋nH	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

V850E2S CPU は，例外同期命令（SYNCE 命令）を備えます。SYNCE 命令により同期化される例外はインプレサイス例外です。しかし，V850E2S CPU は，インプレサイス例外に対応していないため，SYNCE 命令は SYNCM 命令に置き換えて実行されます。

6. 4 例外ハンドラ・アドレス切り替え機能

V850E2S 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-INT127）
- 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-INT127）

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，実行中にソフトウエ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-INT127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128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-INT127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-INT127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-INT127 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，実行中にソフトウエ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，ソフトウエアによって設定します。

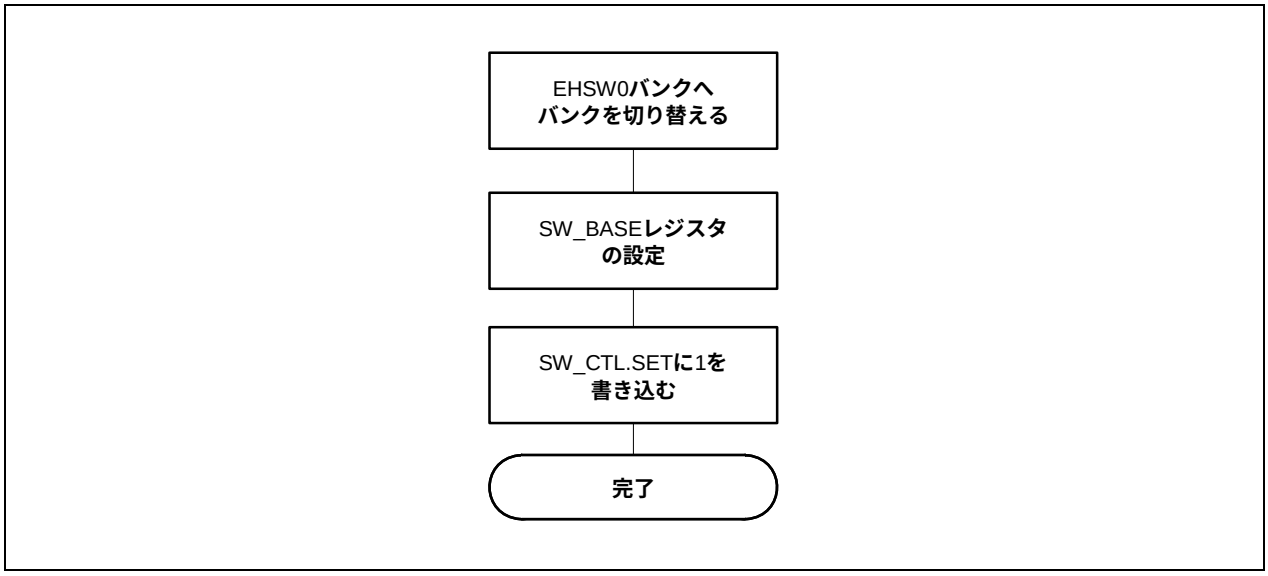
(1) ソフトウエアによる切り替え

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) ソフトウエアによる切り替え

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。システム的に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，ソフトウエ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

V850E2S 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。

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
未定義	UCPOP0-UCPOP7注	530H-537H注

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。
V850E2S では，コプロセッサ命令は定義されていません。

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 － SYSCAL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[注]－ 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レベル例外用作業レジスタ	
	FEWR － FEレベル例外用作業レジスタ	
	BSEL － レジスタ・バンクの選択	00000000H

注　DBICレジスタは開発ツール向けのデバッグ機能で使用します。

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

V850E2S 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

V850E2S CPUは次の2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／定数保護領域とデータ保護領域を共用で使用する保護領域を最大4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。

(2) 実行レベルによる管理

V850E2S 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。

V850E2S 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エ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エア・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[注], 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用します。

- 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）
- ソフトウエア・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用されます。

- 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）		ソフトウエア・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**	**PA0L**	**保護領域0下限アドレス**	PA0L
7	**機能拡張用に予約**		**PA0U**	**保護領域0上限アドレス**	PA0U
8			**PA1L**	**保護領域1下限アドレス**	PA1L
9			**PA1U**	**保護領域1上限アドレス**	PA1U
10			**PA2L**	**保護領域2下限アドレス**	PA2L
11			**PA2U**	**保護領域2上限アドレス**	PA2U
12			**PA3L**	**保護領域3下限アドレス**	PA3L
13			**PA3U**	**保護領域3上限アドレス**	PA3U
14			**機能拡張用に予約**		**機能拡張用に予約**
15					
16					
17					
18					
19					
20					
21					
22					
23					
24					
25					
26					
27					
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 注	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用します。**

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ー一覧を示します。

表2－2　プロセッサ保護設定レジスター一覧

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[注]
			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	PA0L	保護領域0下限アドレス	○	○	○
7	PA0U	保護領域0上限アドレス	○	○	○
8	PA1L	保護領域1下限アドレス	○	○	○
9	PA1U	保護領域1上限アドレス	○	○	○
10	PA2L	保護領域2下限アドレス	○	○	○
11	PA2U	保護領域2上限アドレス	○	○	○
12	PA3L	保護領域3下限アドレス	○	○	○
13	PA3U	保護領域3上限アドレス	○	○	○
14-27	（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））		×	×	○

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ー一覧を示します。

表2－3　プロセッサ保護違反レジスター覧

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[注]
			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	○	○	○
7-27	（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））		×	×	○

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) ソフトウエア・ページング・レジスタ

ソフトウエア・ページング・レジスタは，ソフトウエ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。ソフトウエア・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。

V850E2S 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エア・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エア・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エア・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，ソフトウエア・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，ソフトウエア・オーバヘッドを減少させます。

表2－4にソフトウエア・ページング・レジスター覧を示します。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 ソフトウエア・ページング・レジスター覧

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・レジスタ保護注
			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	PA0L	保護領域0下限アドレス	○	○	○
7	PA0U	保護領域0上限アドレス	○	○	○
8	PA1L	保護領域1下限アドレス	○	○	○
9	PA1U	保護領域1上限アドレス	○	○	○
10	PA2L	保護領域2下限アドレス	○	○	○
11	PA2U	保護領域2上限アドレス	○	○	○
12	PA3L	保護領域3下限アドレス	○	○	○
13	PA3U	保護領域3上限アドレス	○	○	○
14-27	（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））		×	×	○

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. 2. 1　PSW － プログラム・ステータス・ワード

CPU機能バンク上のPSWレジスタのビット16-18に，プロセッサ保護機能に関するビットが配置されています。

図2－1　PSW上のメモリ保護動作状態ビット

	31													18	17	16					11	10	9		7	6	5	4	3	2	1	0	
PSW	0	0	0	0	0	0	0	0	0	0	0	0	0	NPV	DMP	IMP	0	0	0	0	SS	SB	SE	0	NP	EP	ID	SAT	CY	OV	S	Z	初期値 00000020H

ビット位置	ビット名	意　味
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（初期値：0）。 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。
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（初期値：0）。 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。

2. 2. 2　MPM － プロセッサ保護動作モードの設定

プロセッサ保護モード・レジスタはプロセッサ保護機能の基本的な動作状態を決定します。ビット31-6，2には必ず0を設定してください。

(1/2)

	31																								5	4	3		1	0	
MPM	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DWE	DRE	DXE	0	AUE	MPE	初期値 00000000H

ビット位置	ビット名	意　味
5	DWE[注1]	領域指定されていない空間に対するライト・アクセスの許可／禁止を設定します。 DWEビットがクリア（0）されている場合，領域指定されていない空間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。 禁止された空間に対するライト・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例外がただちに受け付けられます。 0：領域指定されていない空間に対するライト・アクセスの禁止 1：領域指定されていない空間に対するライト・アクセスの許可
4	DRE[注1]	領域指定されていない空間に対するリード・アクセスの許可／禁止を設定します。 DREビットがクリア（0）されている場合，領域指定されていない空間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。 禁止された空間に対するリード・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例外がただちに受け付けられます。 0：領域指定されていない空間に対するリード・アクセスの禁止 1：領域指定されていない空間に対するリード・アクセスの許可
3	DXE[注1]	領域指定されていない空間に対する命令実行の許可／禁止を設定します。 DXEビットがクリア（0）されている場合，領域指定されていない空間に置かれたプログラムの実行は禁止されます。 禁止された空間に対する命令実行を行おうとした場合，命令保護違反を検出し，MIP例外がただちに受け付けられます。 0：領域指定されていない空間に対する命令実行の禁止 1：領域指定されていない空間に対する命令実行の許可
1	AUE	例外発生時に自動的に実行レベルを遷移し， PSW.NPV, DMP, IMPビットの更新を行う機能の設定をします（初期値：0）。 0：自動更新をしない 例外が発生しても，PSW.NPV, DMP, IMPビットの値を変更しません[注2,3] 1：自動更新をする 例外が発生すると，PSW.NPV, DMP, IMPビットの値を0に設定します

注1. DWE，DRE，DXEは，0000_0000H~01FF_FFFFH，FE00_0000H~FFFF_FFFFHの64MB空間のみ有効です。上記64MB空間以外は，命令実行／アクセス禁止です。

2. DBレベルへ遷移する例外，およびプロセッサ保護違反例外（MDP例外／MIP例外）を除きます。これらの例外受け付け時には，いかなる状態でもPSW.NPV, DMP, IMPビットは0に更新されます

3. 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。

備考 アドレッシング可能な64MBの空間内においては，いずれの設定に対しても，PAnL/Uレジスタの設定が優先します。
例えば，DRE=1で領域指定されていない空間に対するリード・アクセスを許可しても，PAnL/Uレジスタでリード禁止に設定した領域はリード・アクセス禁止です。

(2/2)

ビット位置	ビット名	意　味
0	MPE	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）。 0：プロセッサ保護機能無効 ・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。 ・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。 1：プロセッサ保護機能有効 ・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。 ・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。

備考 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。

アドレッシング不可能な64MB領域外部へデータ・アクセスした場合，プロセッサ保護機能が有効(MPE=1)であれば，すべてのデータ・アクセスはメモリ保護違反として検出され，MDP例外が発生します。一方，アドレッシング不可能な64MB領域外部の命令実行によるメモリ保護違反(MIP例外)は検出できません。

2. 2. 3 MPC － プロセッサ保護コマンドの指定

プロセッサ保護機能の特殊操作を行うビットが配置されたレジスタです。

ビット31-1には必ず0を設定してください。

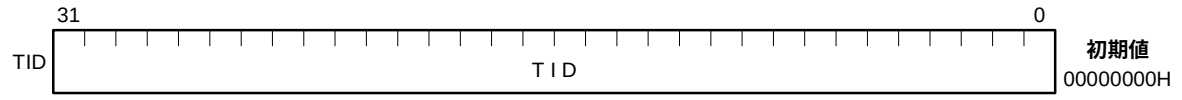
	31																												1	0	
MPC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	ALDS	初期値 00000000H

ビット位置	ビット名	意　味
0	ALDS	このビットをセット（1）するとただちに全メモリ保護領域の保護動作ビットがクリア（0）されます。対象となるは次のビットです。 PAnL.Eビット(n = 0-3) 対象となるすべてのビットのクリア（0）が完了したあと，ALDSビットもクリア（0）されます。

注意 ALDSビットをセット（1）したLDSR命令の直後のLDSR命令で，ALDSビットの機能によってクリア（0）されるビットをセット（1）した場合にも，命令の実行手順に従った結果が得られます（対象のビットはセット（1）されます）。

2.2.4　TID － タスク識別子

実行中タスクの識別子を設定するレジスタです。TIDレジスタは自動的には変更されません。タスクを切り替える際には，必ずプログラムによって適切な値をTIDレジスタに設定してください。



ビット位置	ビット名	意　味
31-0	TID	実行中タスクの識別子を設定します。

2.2.5　その他のシステム・レジスタ

その他のシステム・レジスタに関しての詳細は，**第5章　システム・レジスタ保護**，**第6章　メモリ保護**，**第8章　特殊機能**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.NPV, DMP, IMPビットが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.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.NPV, DMP, IMPビットが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.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，メモリ保護機能を利用できません）。

第4章　実行レベル

V850E2S CPUでは，PSW（プログラム・ステータス・ワード）上の次の3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。

- 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，プログラム領域という3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エアが区別するための情報として，PSW.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（NPV，DMP，IMPビット）を，PSW上に配置しています。PSW上のビット18，17，16をそれぞれ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.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。

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

V850E2S CPUでは，PSW.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	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	例外ハンドラ，OSカーネルなど
1	0	0	0	デバイス・ドライバなど
2	0	1	1	共有ライブラリなど
3	1	1	1	ユーザ・タスク

4. 4　実行レベルの遷移

V850E2S 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.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　V850E2S 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.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.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.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）の発生時は，AUE ビットの設定にかかわらず，PSW.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NPV, DMP, IMPビットに対応するビットの値が，PSW上の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.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，既存ソフトウエ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

V850E2S 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

V850E2S 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，違反情報がプロセッサ保護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。

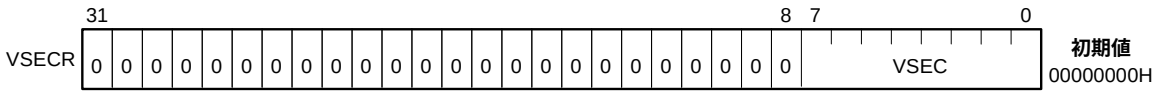
表5－1　システム・レジスタ・バンク

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）		ソフトウエア・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	PA0L	保護領域0下限アドレス	PA0L
7	機能拡張用に予約		PA0U	保護領域0上限アドレス	PA0U
8			PA1L	保護領域1下限アドレス	PA1L
9			PA1U	保護領域1上限アドレス	PA1U
10			PA2L	保護領域2下限アドレス	PA2L
11			PA2U	保護領域2上限アドレス	PA2U
12			PA3L	保護領域3下限アドレス	PA3L
13			PA3U	保護領域3上限アドレス	PA3U
14			機能拡張用に予約		機能拡張用に予約
15					
16					
17					
18					
19					
20					
21					
22					
23					
24					
25					
26					
27					

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。

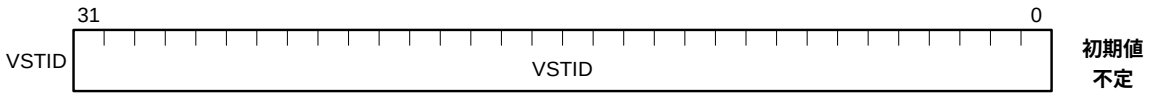
ビット31-8には必ず0を設定してください。



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。

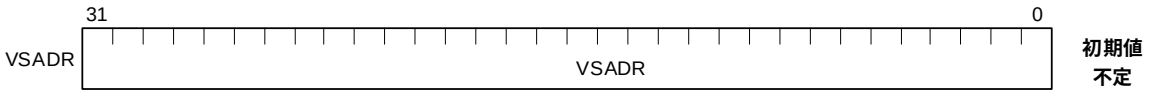


ビット位置	ビット名	意　味
31-0	VSTID	VSECRレジスタのVSECビットが0の状態で，システム・レジスタ保護違反を検出した場合に，その時点のTIDレジスタ（プロセッサ保護設定バンク）の値を格納します。 VSTIDレジスタは違反処理後，最初にシステム・レジスタ保護違反を起こしたタスクの識別子を保存します。

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。



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の状態で，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　命令アドレッシング範囲は 64M バイトです。VSADR レジスタのビット 31-26 はビット 25 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[注]。

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・レジスター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・レジスター覧（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・レジスター覧
- 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ー覧
- 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ー覧
- ソフトウエア・ページング・バンク
  - 第3編　表2－4　ソフトウエア・ページング・レジスター覧

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ステートで動作させてください。CPUがNTステート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

V850E2S 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。

V850E2S 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，64Mバイトのアドレス空間に対する保護設定が可能です。

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）		ソフトウエア・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	PA0L	保護領域0下限アドレス	PA0L
7	機能拡張用に予約		PA0U	保護領域0上限アドレス	PA0U
8			PA1L	保護領域1下限アドレス	PA1L
9			PA1U	保護領域1上限アドレス	PA1U
10			PA2L	保護領域2下限アドレス	PA2L
11			PA2U	保護領域2上限アドレス	PA2U
12			PA3L	保護領域3下限アドレス	PA3L
13			PA3U	保護領域3上限アドレス	PA3U
14			機能拡張用に予約		機能拡張用に予約
15					
16					
17					
18					
19					
20					
21					
22					
23					
24					
25					
26					
27					

6. 1. 1　PAnL － 保護領域n下限アドレス（n = 0-3）

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, 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
PAnL	AL31	AL30	AL29	AL28	AL27	AL26	AL25	AL24	AL23	AL22	AL21	AL20	AL19	AL18	AL17	AL16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	初期値
AL15	AL14	AL13	AL12	AL11	AL10	AL9	AL8	AL7	AL6	AL5	AL4	0	0	S	E	00000000H

ビット位置	ビット名	意　味
31-4	AL31- AL4	保護領域の下限アドレスを設定します。 また，PAnL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，下限アドレスのビット3-0（AL3-0）は暗黙的に0を使用します。 00000000H～01FFFFFFH、FE000000H～FFFFFFFFHの64MB空間内を指定してください。 AL31～AL26への書き込みは無視され、AL25をAL31まで拡張した値が使用されます。
1	S	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（保護領域nを使用しません） 1：有効（保護領域nを使用します）

備考　n = 0-3

6. 1. 2　PAnU － 保護領域n上限アドレス（n = 0-3）

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
PAnU	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	初期値
AU15	AU14	AU13	AU12	AU11	AU10	AU9	AU8	AU7	AU6	AU5	AU4	0	W	R	X	00000000H

ビット位置	ビット名	意　味
31-4	AU31- AU4	保護領域の上限アドレスを設定します。 また，PAnU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，上限アドレスのビット3-0(AU3-0)は暗黙的に1を使用します。 00000000H～01FFFFFFH、FE000000H～FFFFFFFFHの64MB空間内を指定してください。
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例外がただ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
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 = 0-3

6. 1. 3　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

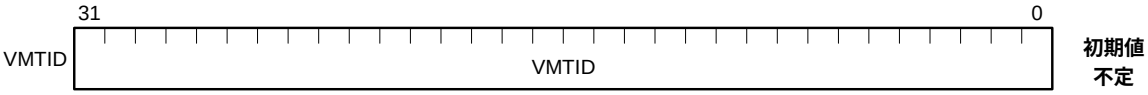
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	初期値
0	0	0	0	0	0	0	0	0	VMMS	VMRMW	VMS	VMW	VMR	VMX	0	00000000H

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，7. 3　違反要因の特定を参照してください。

6. 1. 4　VMTID － メモリ保護違反タスク識別子

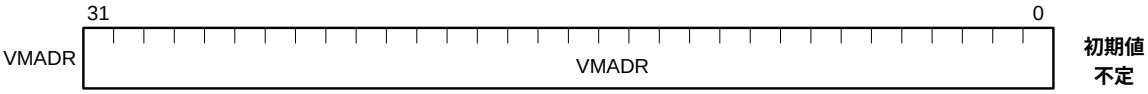
VMTIDレジスタはMIP例外，MDP例外発生時のタスクの識別子を格納します。



ビット位置	ビット名	意　味
31-0	VMTID	実行保護違反，データ保護違反のいずれか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）が格納されます。

6. 1. 5　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。



ビット位置	ビット名	意　味
31-0	VMADR	実行保護違反を検出した場合は違反を起こした命令のPC，データ保護違反を検出した場合は，そのデータ・アクセス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。

V850E2S CPUでは，これらの保護領域を設定するために次のようなレジスタを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。

- PAnL/PAnU（n = 0-3）

各保護領域の設定は，上限レジスタと下限レジスタの2つの組み合せで表現されます。V850E2S CPUは，最大で4個の保護領域を設定できるレジスタを定義しています。

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		PAnU					PAnL			
		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）	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		S	E
PA0U/L	命令保護領域設定	上限アドレス	0	0	(0)	(0)	下限アドレス	0	(0)	(0)
PA1U/L		上限アドレス	0	0	(0)	(0)	下限アドレス	0	(0)	(0)
PA2U/L		上限アドレス	0	0	(0)	(0)	下限アドレス	0	(0)	(0)
PA3U/L		上限アドレス	0	0	(0)	(0)	下限アドレス	0	(0)	(0)

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。

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
PA0L-PA3L	0000 0000H
PA0U-PA3U	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**がただちに受け付けられます。

6. 3. 6　保護領域下限アドレス（AL31-AL0ビット）

保護領域の下限アドレスを示します。

また，PAnL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，下限アドレスのビット3-0（AL3-AL0）は暗黙的に0を使用します（n = 0-3）。

6. 3. 7　保護領域上限アドレス（AU31-AU0ビット）

保護領域の上限アドレスを示します。

また，PAnU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，上限アドレスのビット3-0（AU3-AU0）は暗黙的に1を使用します（n = 0-3）。

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

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) 命令保護領域として使用する場合

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) データ保護領域として使用する場合

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　特殊なメモリ・アクセス命令

V850E2S 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. 5. 1　ミスアライン・アクセスを行うロード／ストア命令

V850E2S 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て，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. 5. 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. 5. 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. 5. 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. 6　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例外に関しての詳細は，第7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　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。

7. 1　違反の種類

V850E2S 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を発生します。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3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

7. 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，例外を発生しません。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。

7. 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例外を発生します。

7. 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例外を発生します。

7. 2　例外の種類

V850E2S CPUはプロセッサ保護機能で定められた2種類の例外を発生します。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。

7. 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，違反を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。

7. 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，違反を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。

7.3 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7－1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7-1 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
その他の例外	－	－	－	－

7.3.1 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7-2 MIP例外発生時のVMECR設定値

レジスタ	VMECR						
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

7. 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7-3　MDP例外発生時のVMECR設定値

レジスタ	VMECR						
ビット番号	6	5	4	3	2	1	0
ビット名	VMMS	VMRMW	VMS	VMW	VMR	VMX	-
リード命令（アライン）注1	0	0	0/1注5	0	1	0	0
ライト命令（アライン）注2	0	0	0/1注5	1	0	0	0
リード命令（ミスアライン）注3	1	0	0/1注5	0	1	0	0
ライト命令（ミスアライン）注2	1	0	0/1注5	1	0	0	0
CAXI/SET1/NOT1/CLR1命令注4	0	1	0/1注5	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

第8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。

8. 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。

- PAnL.Eビット（n = 0-3）

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	フォーマット	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・マスカブル例外の禁止
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・マスカブル例外の許可
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・ソフトウエア例外命令

表A－1 基本命令機能一覧（アルファベット順）（2/4）

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	
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・イミーディエトの乗算
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・ソフトウエ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・データのロード
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	－	－	－	－	－	ソフトウエア例外
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

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）

ニモニック	オペランド	フォーマット	オペコード 15 11	10 5	4 0	31 27	26 21	20 16	備 考
NOP		I	00000	000000	00000				
SYNCE		I	00000	000000	11101				
SYNCM		I	00000	000000	11110				
SYNCP		I	00000	000000	11111				
MOV	reg1, reg2	I	rrrrr	000000	RRRRR				rrrrr≠00000
NOT	reg1, reg2	I	rrrrr	000001	RRRRR				
RIE		I	00000	0 0 0 1 0	00000				
SWITCH	reg1	I	00000	000010	RRRRR				RRRRR≠00000
FETRAP	vector4	I	0iiii	000010	00000				iiii≠0000
DIVH	reg1, reg2	I	rrrrr	000010	RRRRR				rrrrr≠00000, RRRRR≠00000
JMP	[reg1]	I	000000	000011	RRRRR				
SLD.BU	disp4 [ep], reg2	IV	rrrrr	000011	0dddd				rrrrr≠00000
SLD.HU	disp5 [ep],reg2	IV	rrrrr	000011	1dddd				rrrrr≠00000
ZXB	reg1	I	00000	000100	RRRRR				
SXB	reg1	I	00000	000101	RRRRR				
ZXH	reg1	I	00000	000110	RRRRR				
SXH	reg1	I	00000	000111	RRRRR				
SATSUBR	reg1, reg2	I	rrrrr	000100	RRRRR				rrrrr≠00000
SATSUB	reg1, reg2	I	rrrrr	000101	RRRRR				rrrrr≠00000
SATADD	reg1, reg2	I	rrrrr	000110	RRRRR				rrrrr≠00000
MULH	reg1, reg2	I	rrrrr	000111	RRRRR				rrrrr≠00000
OR	reg1, reg2	I	rrrrr	001000	RRRRR				
XOR	reg1, reg2	I	rrrrr	001001	RRRRR				
AND	reg1, reg2	I	rrrrr	001010	RRRRR				
TST	reg1, reg2	I	rrrrr	001011	RRRRR				
SUBR	reg1, reg2	I	rrrrr	001100	RRRRR				
SUB	reg1, reg2	I	rrrrr	001101	RRRRR				
ADD	reg1, reg2	I	rrrrr	001110	RRRRR				
CMP	reg1, reg2	I	rrrrr	001111	RRRRR				
MOV	imm5, reg2	I	rrrrr	010000	iiiii				rrrrr≠00000
SATADD	imm5, reg2	I	rrrrr	010001	iiiii				rrrrr≠00000

表B－1 基本命令オペコード一覧（16, 32ビット命令）（2/4）

ニモニック	オペランド	フォーマット	オペコード 15 11	10 5	4 0	31 27	26 21	20 16	備 考
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 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 0	**表B－2参照** RRRRR≠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 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 L	L L L L L	L L L L L L	R R R R R	RRRRR≠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 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 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 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 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 1	
PREPARE	list12, imm5	XIII	0 0 0 0 0	1 1 1 1 0 i	i i i i L	L L L L L	L L L L L L	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	フォーマット	オペコード 15 11	10 5	4 0	31 27	26 21	20 16	備 考
ST.B	reg3, disp23[reg1]	XIV	00000	111100	RRRRRR	wwwww	dddddd	d1101	**表B－2参照**
ST.W	reg3, disp23[reg1]	XIV	00000	111100	RRRRRR	wwwww	dddddd	01111	**表B－2参照**
LD.BU	disp23[reg1], reg3	XIV	00000	111101	RRRRRR	wwwww	dddddd	d0101	**表B－2参照**
LD.HU	disp23[reg1], reg3	XIV	00000	111101	RRRRRR	wwwww	dddddd	00111	**表B－2参照**
ST.H	reg3, disp23[reg1]	XIV	00000	111101	RRRRRR	wwwww	dddddd	01101	**表B－2参照**
JR	disp22	V	00000	11110D	DDDDD	ddddd	dddddd	dddd0	
JARL	disp22, reg2	V	rrrrr	11110D	DDDDD	ddddd	dddddd	dddd0	rrrrr≠00000
LD.BU	disp16 [reg1], reg2	VII	rrrrr	11110b	RRRRR	ddddd	dddddd	dddd1	
SET1	bit3#, disp16 [reg1]	VIII	00bbb	111110	RRRRR	ddddd	dddddd	ddddd	
NOT1	bit#3, disp16 [reg1]	VIII	01bbb	111110	RRRRR	ddddd	dddddd	ddddd	
CLR1	bit3#, disp16 [reg1]	VIII	10bbb	111110	RRRRR	ddddd	dddddd	ddddd	
TST1	bit3#, disp16 [reg1]	VIII	11bbb	111110	RRRRR	ddddd	dddddd	ddddd	
LD.HU	disp16 [reg1], reg2	VII	rrrrr	111111	RRRRR	ddddd	dddddd	dddd1	rrrrr≠00000
SETF	cond, reg2	IX	rrrrr	111111	0CCCC	00000	000000	00000	
RIE		X	xxxxx	111111	1xxxx	00000	000000	00000	
LDSR	reg2, regID	IX	rrrrr	111111	RRRRR	00000	000001	00000	
STSR	sr1, reg2	IX	rrrrr	111111	RRRRR	00000	000010	00000	
SHR	reg1, reg2	IX	rrrrr	111111	RRRRR	00000	000100	00000	
SHR	reg1, reg2, reg3	IX	rrrrr	111111	RRRRR	wwwww	000100	00010	
SAR	reg1, reg2	IX	rrrrr	111111	RRRRR	00000	000101	00000	
SAR	reg1, reg2, reg3	XI	rrrrr	111111	RRRRR	wwwww	000101	00010	
SHL	reg1, reg2	IX	rrrrr	111111	RRRRR	00000	000110	00000	
SHL	reg1, reg2, reg3	XI	rrrrr	111111	RRRRR	wwwww	000110	00010	
SET1	reg2, [reg1]	IX	rrrrr	111111	RRRRR	00000	000111	00000	
NOT1	reg2, [reg1]	IX	rrrrr	111111	RRRRR	00000	000111	00010	
CLR1	reg2, [reg1]	IX	rrrrr	111111	RRRRR	00000	000111	00100	
TST1	reg2, [reg1]	IX	rrrrr	111111	RRRRR	00000	000111	00110	
CAXI	[reg1], reg2, reg3	XI	rrrrr	111111	RRRRR	wwwww	000111	01110	
TRAP	imm5	X	00000	111111	iiiii	00000	001000	00000	
HALT		X	00000	111111	00000	00000	001001	00000	
RETI		X	00000	111111	00000	00000	001010	00000	
CTRET		X	00000	111111	00000	00000	001010	00100	
EIRET		X	00000	111111	00000	00000	001010	01000	
FERET		X	00000	111111	00000	00000	001010	01010	
DI		X	00000	111111	00000	00000	001011	00000	
EI		X	10000	111111	00000	00000	001011	00000	
SYSCALL	vector8	X	11010	111111	vvvvv	00VVV	001011	00000	
SASF	cccc, reg2	IX	rrrrr	111111	0cccc	00000	010000	00000	
MUL	reg1, reg2, reg3	XI	rrrrr	111111	RRRRR	wwwww	010001	00000	
MULU	reg1, reg2, reg3	XI	rrrrr	111111	RRRRR	wwwww	010001	00010	
MUL	imm9, reg2, reg3	XII	rrrrr	111111	iiiii	wwwww	01001I	III00	
MULU	imm9, reg2, reg3	XII	rrrrr	111111	iiiii	wwwww	01001I	III10	

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	フォーマット	オペコード 15 11	10 5	4 0	31 27	26 21	20 16	備 考
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）

ニモニック	オペランド	フォーマット	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	ddddd	dddddd	dddd0	DDDDD	DDDDDD	DDDDD
JARL	disp32, reg1	VI	00000	010111	RRRRR	ddddd	dddddd	dddd0	DDDDD	DDDDDD	DDDDD
MOV	imm32, reg1	VI	00000	110001	RRRRR	iiiii	iiiii	iiiii	IIIII	IIIIII	IIIII
JMP	disp32 [reg1]	VI	00000	110111	RRRRR	ddddd	dddddd	dddd0	DDDDD	DDDDDD	DDDDD
PREPARE	list12, imm5, sp/imm	XIII	00000	11110i	iiiiiL	LLLLL	LLLLLL	ff[注]011	IIIII	IIIIII	IIIII
LD.B	disp23[reg1], reg3	XIV	00000	111100	RRRRRR	wwwww	dddddd	d0101	DDDDD	DDDDDD	DDDDD
LD.H	disp23[reg1], reg3	XIV	00000	111100	RRRRRR	wwwww	dddddd	00111	DDDDD	DDDDDD	DDDDD
LD.W	disp23[reg1], reg3	XIV	00000	111100	RRRRRR	wwwww	dddddd	01001	DDDDD	DDDDDD	DDDDD
ST.B	reg3, disp23[reg1]	XIV	00000	111100	RRRRRR	wwwww	dddddd	d1101	DDDDD	DDDDDD	DDDDD
ST.W	reg3, disp23[reg1]	XIV	00000	111100	RRRRRR	wwwww	dddddd	01111	DDDDD	DDDDDD	DDDDD
LD.BU	disp23[reg1], reg3	XIV	00000	111101	RRRRRR	wwwww	dddddd	d0101	DDDDD	DDDDDD	DDDDD
LD.HU	disp23[reg1], reg3	XIV	00000	111101	RRRRRR	wwwww	dddddd	00111	DDDDD	DDDDDD	DDDDD
ST.H	reg3, disp23[reg1]	XIV	00000	111101	RRRRRR	wwwww	dddddd	01101	DDDDD	DDDDDD	DDDDD

注 ff = 01, 10

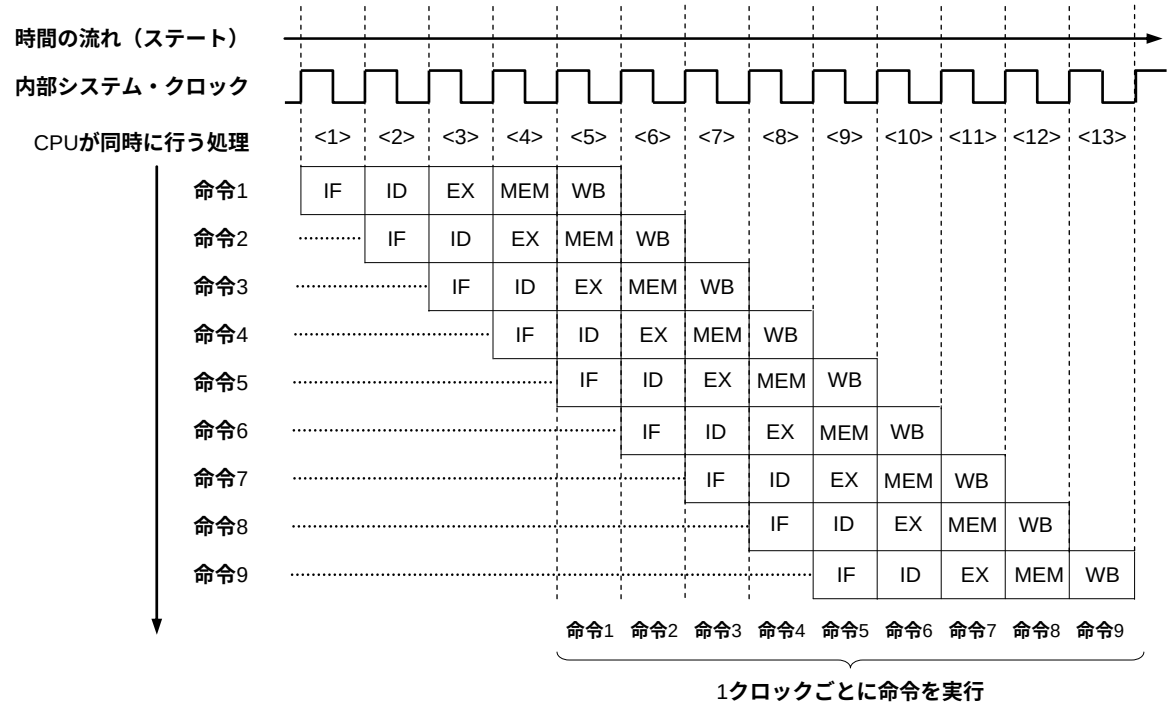
表B－3 基本命令オペコード一覧（64ビット命令）

ニモニック	オペランド	フォーマット	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	iiiiiL	LLLLL	LLLLLL	11011	
			47 32			63 48			
			IIIII	IIIIII	IIIII	IIIII	IIIIII	IIIII	

付録C　パイプライン

V850E2S CPUは，RISCアーキテクチャをベースとし，5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5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9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9個続けて実行する例



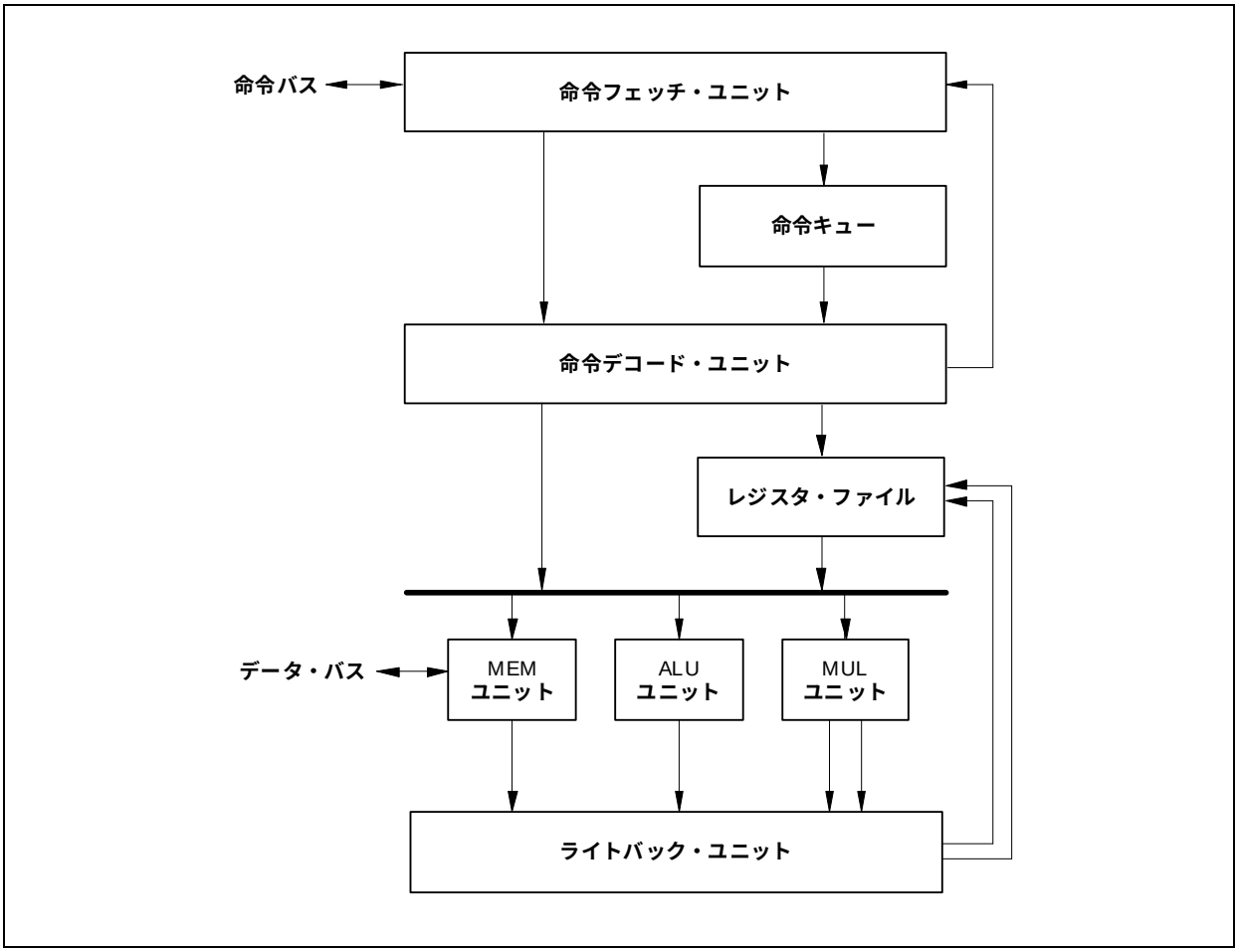
IF（インストラクション・フェッチ）	：命令のフェッチを行い，フェッチ・ポインタをインクリメントします。
ID（インストラクション・デコード）	：命令をデコードし，イミーディエト・データの作成とレジスタの読み出しを行います。
EX（ALU，乗算器，バレル・シフタの実行）	：デコードした命令を実行します。
MEM（メモリ・アクセス）	：対象となるメモリにアクセスします。
WB（ライトバック）	：実行した結果をレジスタに書き込みます。

<1> - <13>は，CPUのステートを示します。

C. 1　特　　徴

V850E2S 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。

図C－2　パイプライン構成



パイプラインは以下の機能で構成されています。

(a) 命令フェッチ・ユニット

32 ビットのフェッチ・バス（CPU フェッチ・バス）から 1 サイクルでフェッチします。

(b) 命令デコード・ユニット

命令フェッチ・ユニットから発行された命令をデコードします。

(c) ALUユニット

整数演算，論理演算を行う命令を実行します。

(d) 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。

(e) MULユニット

整数乗算を行う命令を実行します。

(f) 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	ニモニック	オペランド	バイト数	実行クロック数		
				issue	repeat	latency
ロード命令	LD.B	disp16 [reg1] , reg2	4	1	1	2注1
	LD.B	disp23 [reg1] , reg3	6	1	1	2注1
	LD.BU	disp16 [reg1] , reg2	4	1	1	2注1
	LD.BU	disp23 [reg1] , reg3	6	1	1	2注1
	LD.H	disp16 [reg1] , reg2	4	1	1	2注1
	LD.H	disp23 [reg1] , reg3	6	1	1	2注1
	LD.HU	disp16 [reg1] , reg2	4	1	1	2注1
	LD.HU	disp23 [reg1] , reg3	6	1	1	2注1
	LD.W	disp16 [reg1] , reg2	4	1	1	2注1
	LD.W	disp23 [reg1] , reg3	6	1	1	2注1
	SLD.B	disp7 [ep] , reg2	2	1	1	2注1
	SLD.BU	disp4 [ep] , reg2	2	1	1	2注1
	SLD.H	disp8 [ep] , reg2	2	1	1	2注1
	SLD.HU	disp5 [ep] , reg2	2	1	1	2注1
	SLD.W	disp8 [ep] , reg2	2	1	1	2注1
ストア命令	ST.B	reg2, disp16 [reg1]	4	1	1	1
	ST.B	reg3, disp23 [reg1]	6	1	1	1
	ST.H	reg2, disp16 [reg1]	4	1	1	1
	ST.H	reg3, disp23 [reg1]	6	1	1	1
	ST.W	reg2, disp16 [reg1]	4	1	1	1
	ST.W	reg3, disp23 [reg1]	6	1	1	1
	SST.B	reg2, disp7 [ep]	2	1	1	1
	SST.H	reg2, disp8 [ep]	2	1	1	1
	SST.W	reg2, disp8 [ep]	2	1	1	1
乗算命令	MUL	reg1, reg2, reg3	4	1	4	4
	MUL	imm9, reg2, reg3	4	1	4	4
	MULH	reg1, reg2	2	1	1	1
	MULH	imm5, reg2	2	1	1	1
	MULHI	imm16, reg1, reg2	4	1	1	1
	MULU	reg1, reg2, reg3	4	1	4	4
	MULU	imm9, reg2, reg3	4	1	4	4
加算付き乗算命令	MAC	reg1, reg2, reg3, reg4	4	2	5	5
	MACU	reg1, reg2, reg3, reg4	4	2	5	5

表C－1 命令実行クロック数一覧（2/4）

命令の種類	ニモニック	オペランド	バイト数	実行クロック数		
				issue	repeat	latency
算術演算命令	ADD	reg1, reg2	2	1	1	1
	ADD	imm5, reg2	2	1	1	1
	ADDI	imm16, reg1, reg2	4	1	1	1
	CMP	reg1, reg2	2	1	1	1
	CMP	imm5, reg2	2	1	1	1
	MOV	reg1, reg2	2	1	1	1
	MOV	imm5, reg2	2	1	1	1
	MOV	imm32, reg1	4	1	1	1
算術演算命令	MOVEA	imm16, reg1, reg2	4	1	1	1
	MOVHI	imm16, reg1, reg2	4	1	1	1
	SUB	reg1, reg2	2	1	1	1
	SUBR	reg1, reg2	2	1	1	1
条件付き演算命令	ADF	cccc, reg1, reg2, reg3	4	1	1	1
	SBF	cccc, reg1, reg2, reg3	4	1	1	1
飽和演算命令	SATADD	reg1, reg2	2	1	1	1
	SATADD	imm5, reg2	2	1	1	1
	SATADD	reg1, reg2, reg3	4	1	1	1
	SATSUB	reg1, reg2	2	1	1	1
	SATSUB	reg1, reg2, reg3	4	1	1	1
	SATSUBI	imm16, reg1, reg2	4	1	1	1
	SATSUBR	reg1, reg2	2	1	1	1
論理演算命令	AND	reg1, reg2	2	1	1	1
	ANDI	imm16, reg1, reg2	4	1	1	1
	NOT	reg1, reg2	2	1	1	1
	OR	reg1, reg2	2	1	1	1
	ORI	imm16, reg1, reg2	4	1	1	1
	TST	reg1, reg2	2	1	1	1
	XOR	reg1, reg2	2	1	1	1
	XORI	imm16, reg1, reg2	4	1	1	1
データ操作命令	BSH	reg2, reg3	4	1	1	1
	BSW	reg2, reg3	4	1	1	1
	CMOV	cccc, reg1, reg2, reg3	4	1	1	1
	CMOV	cccc, imm5, reg2, reg3	4	1	1	1
	HSH	reg2, reg3	4	1	1	1
	HSW	reg2, reg3	4	1	1	1
	SAR	reg1, reg2	4	1	1	1
	SAR	imm5, reg2	2	1	1	1
	SAR	reg1, reg2, reg3	4	1	1	1
	SASF	cccc, reg2	4	1	1	1
	SETF	cccc, reg2	4	1	1	1
	SHL	reg1, reg2	4	1	1	1
	SHL	imm5, reg2	2	1	1	1
	SHL	reg1, reg2, reg3	4	1	1	1

表C－1 命令実行クロック数一覧（3/4）

命令の種類	ニモニック	オペランド	バイト数	実行クロック数		
				issue	repeat	latency
データ操作命令	SHR	reg1, reg2	4	1	1	1
	SHR	imm5, reg2	2	1	1	1
	SHR	reg1, reg2, reg3	4	1	1	1
データ操作命令	SXB	reg1	2	1	1	1
	SXH	reg1	2	1	1	1
	ZXB	reg1	2	1	1	1
	ZXH	reg1	2	1	1	1
ビット・サーチ命令	SCH0L	reg2, reg3	4	1	1	1
	SCH0R	reg2, reg3	4	1	1	1
	SCH1L	reg2, reg3	4	1	1	1
	SCH1R	reg2, reg3	4	1	1	1
除算命令	DIV	reg1, reg2, reg3	4	36	36	36
	DIVH	reg1, reg2	2	36	36	36
	DIVH	reg1, reg2, reg3	4	36	36	36
	DIVHU	reg1, reg2, reg3	4	35	35	35
	DIVU	reg1, reg2, reg3	4	35	35	35
高速除算命令	DIVQ	reg1, reg2, reg3	4	N+7注4	N+7注4	N+7注4
	DIVQU	reg1, reg2, reg3	4	N+6注4	N+6注4	N+6注4
分岐命令	Bcond	disp9（条件成立時）	2	2注2	2注2	2注2
		disp9（条件不成立時）	2	1注2	1注2	1注2
	JARL	disp22, reg2	4	2	2	2
	JARL	disp32, reg1	4	3	3	3
	JMP	[reg1]	2	3	3	3
	JMP	disp32 [reg1]	4	3	3	3
	JR	disp22	4	2	2	2
	JR	disp32	4	3	3	3
ビット操作命令	CLR1	bit#3, disp16 [reg1]	4	3注1	3注1	3注1
	CLR1	reg2, [reg1]	4	3注1	3注1	3注1
	NOT1	bit#3, disp16 [reg1]	4	3注1	3注1	3注1
	NOT1	reg2, [reg1]	4	3注1	3注1	3注1
	SET1	bit#3, disp16 [reg1]	4	3注1	3注1	3注1
	SET1	reg2, [reg1]	4	3注1	3注1	3注1
	TST1	bit#3, disp16 [reg1]	4	3注1	3注1	3注1
	TST1	reg2, [reg1]	4	3注1	3注1	3注1
特殊命令	CALLT	imm6	2	5	5	5
	CAXI	[reg1], reg2, reg3	4	5	5	5
	CTRET	－	4	3	3	3
	DI	－	4	1	1	1
	DISPOSE	imm5, list12	4	n+1注3	n+1注3	n+1注3
	DISPOSE	imm5, list12, [reg1]	4	n+3注3	n+3注3	n+3注3
	EI	－	4	1	1	1
	EIRET	－	4	3	3	3
	FERET	－	4	3	3	3

表C－1 命令実行クロック数一覧（4/4）

命令の種類	ニモニック	オペランド	バイト	実行クロック数		
				issue	repeat	latency
特殊命令	FETRAP	vector	2	3	3	3
	HALT	－	4	不定	不定	不定
	LDSR	reg2, regID	4	1	1	1
	NOP	－	2	1	1	1
	PREPARE	list12, imm5	4	n+1注3	n+1注3	n+1注3
	PREPARE	list12, imm5, sp	4	n+2注3	n+2注3	n+2注3
	PREPARE	list12, imm5, imm16	4	n+2注3	n+2注3	n+2注3
	PREPARE	list12, imm5, imm16<<16	4	n+2注3	n+2注3	n+2注3
	PREPARE	list12, imm5, imm32	4	n+2注3	n+2注3	n+2注3
	RETI	－	4	3	3	3
	RIE	－	2	3	3	3
	RIE	－	4	3	3	3
	STSR	regID, reg2	4	1	1	1
	SWITCH	reg1	2	5	5	5
	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	5	5	5
	TRAP	vector5	4	3	3	3
予約未定義命令コード（RIE命令として動作）			4	3	3	3

注 1. ウエイト・ステートがある場合（+リード・アクセス・ウエイト・ステート数）。

2. 直前に PSW レジスタの内容を書き換える命令がある場合は+1。

3. n は，list ×で指定されるレジスタの合計数（ウエイト・ステート数による。ウエイト・ステートがない場合，n は list ×で指定されるレジスタの合計数と一致）。

4. 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 0 以下の場合、N=1 とします。

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 ×	× ビット・イミーディエト・データ
disp ×	× 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 ×	ベクタを指定するデータ(×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　条件コード一覧参照）
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

2. 実行クロックの凡例

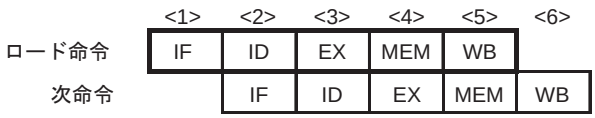
略　　号	意　　味
issue	命令実行直後に他の命令を実行する場合
repeat	命令実行直後に同一命令を繰り返す場合
latency	命令実行結果をその命令実行直後の命令で利用する場合

C. 3 基本命令のパイプライン

C. 3. 1 ロード命令

［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］

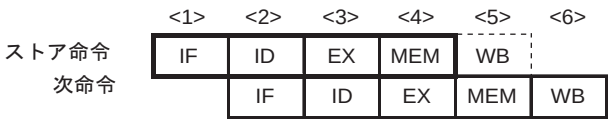


［説　　明］ パイプラインはIF, ID, EX, MEM, WBの5ステージです。LD命令の直後に，実行結果を使用する命令を配置すると，データの待ち合わせ時間が発生します。

C. 3. 2 ストア命令

［対象の命令］ ST.B, ST.H, ST.W, SST.B, SST.H, SST.W

［パイプライン］



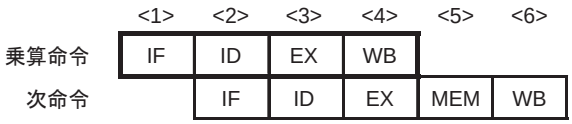
［説　　明］ パイプラインはIF, ID, EX, MEM, WBの5ステージですが，レジスタへのデータの書き込みがないのでWBステージでは何も行いません。

C. 3. 3 乗算命令

(1) ハーフワード・データ乗算命令

［対象の命令］ MULH, MULHI

［パイプライン］

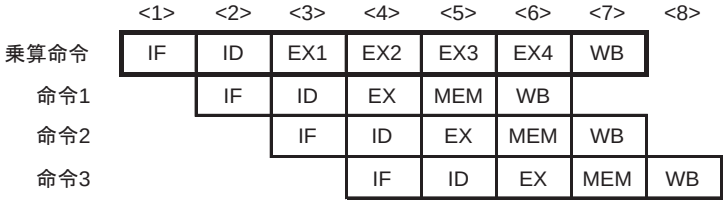


［説　　明］ パイプラインはIF, ID, EX, EBの4ステージです。

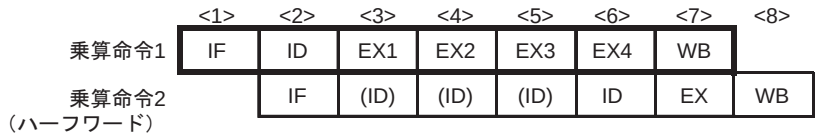
(2) ワード・データ乗算命令

［対象の命令］ MUL, MULU

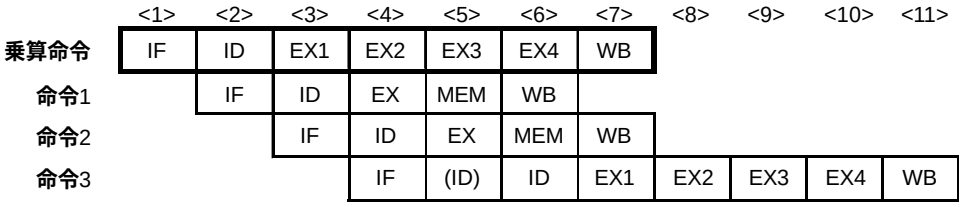
［パイプライン］ (a) 続く3命令が乗算命令以外の場合



(b) 次命令がハーフワード乗算命令の場合



(c) 続く3命令目が乗算命令の場合



［説　　明］ パイプラインはIF, ID, EX1, EX2, EX3, EX4, WBの7ステージです。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。

C. 3. 4 加算付き乗算命令

［対象の命令］ MAC, MACU

［パイプライン］ (a) 続く3命令が（加算付き）乗算命令以外の場合

	<1>	<2>	<3>	<4>	<5>	<6>	<7>	<8>
加算付き乗算命令	IF	ID1	ID2	EX1	EX2	EX3	EX4	WB
命令1		IF	ID	EX	MEM	WB		
命令2			IF	ID	EX	MEM	WB	
命令3				IF	ID	EX	MEM	WB

(b) 次命令がハーフワード乗算命令の場合

	<1>	<2>	<3>	<4>	<5>	<6>	<7>	<8>	<9>
加算付き乗算命令	IF	ID1	ID2	EX1	EX2	EX3	EX4	WB	
乗算命令（ハーフワード）		IF	(ID)	(ID)	(ID)	(ID)	ID	EX	WB

(c) 次命令が加算付き乗算命令の場合

	<1>	<2>	<3>	<4>	<5>	<6>	<7>	<8>	<9>	<10>	<11>	<12>	<13>
加算付き乗算命令1	IF	ID1	ID2	EX1	EX2	EX3	EX4	WB					
加算付き乗算命令2		IF	(ID)	(ID)	(ID)	(ID)	ID1	ID2	EX1	EX2	EX3	EX4	WB

(d) 続く3命令目がハーフワード乗算命令の場合

	<1>	<2>	<3>	<4>	<5>	<6>	<7>	<8>	<9>
加算付き乗算命令	IF	ID1	ID2	EX1	EX2	EX3	EX4	WB	
命令1		IF	ID	EX	MEM	WB			
命令2			IF	ID	EX	MEM	WB		
乗算命令 (ハーフワード)				IF	(ID)	(ID)	ID	EX	WB

(e) 続く3命令目が加算付き乗算命令の場合

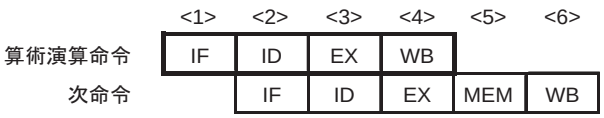
	<1>	<2>	<3>	<4>	<5>	<6>	<7>	<8>	<9>	<10>	<11>	<12>	<13>
加算付き乗算命令1	IF	ID1	ID2	EX1	EX2	EX3	EX4	WB					
命令1		IF	ID	EX	MEM	WB							
命令2			IF	ID	EX	MEM	WB						
加算付き乗算命令2				IF	(ID)	(ID)	ID1	ID2	EX1	EX2	EX3	EX4	WB

［説　　明］ パイプラインはIF, ID1, ID2, EX1, EX2, EX3, EX4, WBの8ステージです。加算付き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。

C. 3. 5 算術演算命令

［対象の命令］ ADD, ADDI, CMP, MOV, MOVEA, MOVHI, SUB, SUBR

［パイプライン］

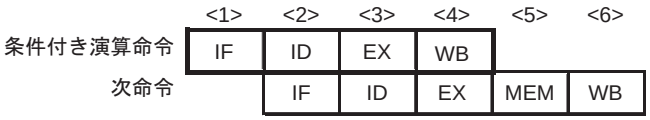


［説　　明］ パイプラインはIF, ID, EX, WBの4ステージです。

C. 3. 6 条件付き演算命令

［対象の命令］ ADF, SBF

［パイプライン］

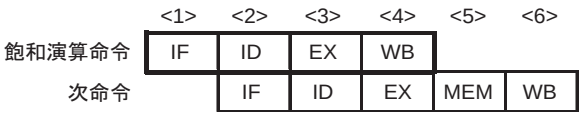


［説　　明］ パイプラインはIF, ID, EX, WBの4ステージです。

C. 3. 7 飽和演算命令

［対象の命令］ SATADD, SATSUB, SATSUBI, SATSUBR

［パイプライン］

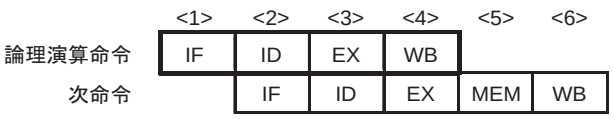


［説　　明］ パイプラインはIF, ID, EX, WBの4ステージです。

C. 3. 8 論理演算命令

［対象の命令］ AND, ANDI, NOT, OR, ORI, TST, XOR, XORI

［パイプライン］

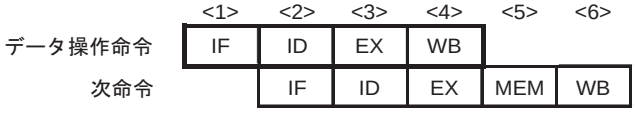


［説　　明］ パイプラインはIF, ID, EX, WBの4ステージです。

C. 3. 9 データ操作命令

［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］

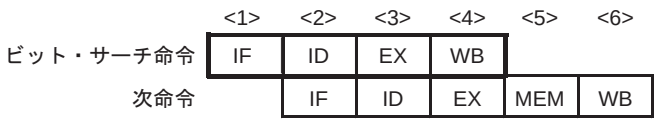


［説　　明］ パイプラインはIF, ID, EX, WBの4ステージです。

C. 3. 10 ビット・サーチ命令

［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, ID, EX, WBの4ステージです。

C. 3. 11 除算命令

［対象の命令］ DIV, DIVH, DIVHU, DIVU

［パイプライン］ (a) DIV, DIVHの場合

(b) DIVU, DIVHUの場合

［説　　明］ パイプラインは，DIV, DIVH命令の場合，IF, ID, EX1-EX36（通常のEXステージ）, WBの39ステージ，DIVHU, DIVU命令の場合，IF, ID, EX1-EX35（通常のEXステージ）, WBの38ステージです。

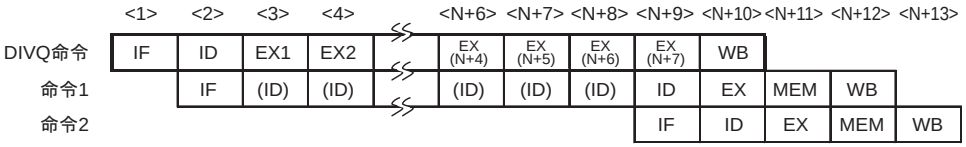
［備　　考］ 除算命令実行中に割り込みが発生すると実行を中止し，戻り番地をこの命令の先頭アドレスとして割り込みを処理します。割り込み処理完了後，この命令を再実行します。この場合，汎用レジスタreg1と汎用レジスタreg2は，この命令実行前の値を保持します。

C. 3. 12 高速除算命令

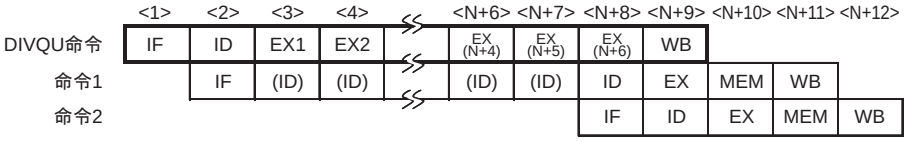
この命令では，演算に必要な最小ステップ数を自動的に決定します。

［対象の命令］ DIVQ, DIVQU

［パイプライン］ (a) DIVQの場合



(b) DIVQUの場合



［説　明］ パイプラインは，DIVQ命令の場合，IF, ID, EX1-EX (N+7), WBのN+10ステージ，DIVQU命令の場合，IF, ID, EX1-EX (N+6), WBのN+9ステージです。

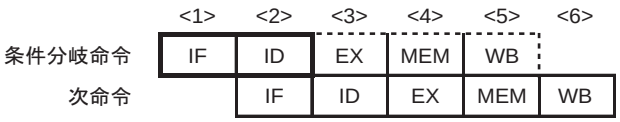
［備　考］ N = （被除数の有効ビット数）－（除数の有効ビット数）です。
ただし，Nがマイナスとなった場合，N = 1として取り扱います。
また，除算命令実行中に割り込みが発生すると実行を中止し，戻り番地をこの命令の先頭アドレスとして割り込みを処理します。割り込み処理完了後，この命令を再実行します。
この場合，汎用レジスタreg1と汎用レジスタreg2は，この命令実行前の値を保持します。

C. 3. 13 分岐命令

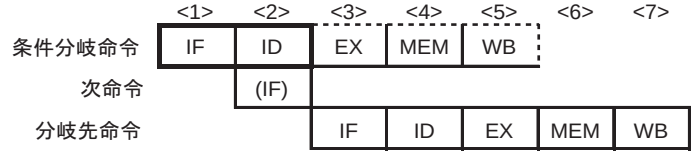
(1) 条件分岐命令（BR命令を除く）

［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



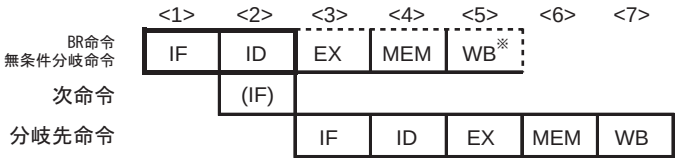
(IF): 無効となる命令フェッチ

［説　明］ パイプラインは IF, ID, EX, MEM, WB の 5 ステージですが，ID ステージで分岐先が決定するため，EX ステージ，MEM ステージ，WB ステージは何も行いません。

(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］



(IF): 無効となる命令フェッチ

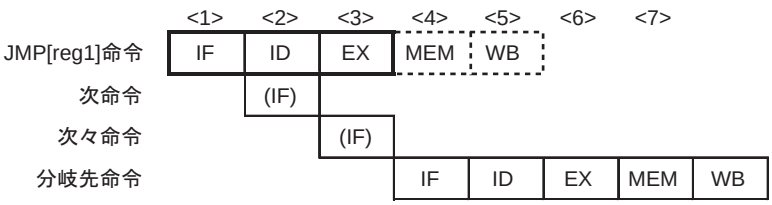
WB※: JR，BR 命令の場合は，何も行われませんが、JARL 命令の場合は復帰 PC の書き込みが行われます。

［説　　明］ パイプラインは IF, ID, EX, MEM, WB の 5 ステージですが，ID ステージで分岐先が決定するため，EX ステージ，MEM ステージ，WB ステージでは何も行いません。ただし，JARL 命令の場合には WB ステージにおいて，復帰 PC の書き込みが行われます。また，分岐命令の次命令の IF は無効となります。

(3) JMP命令

(a) JMP [reg1]命令

［パイプライン］

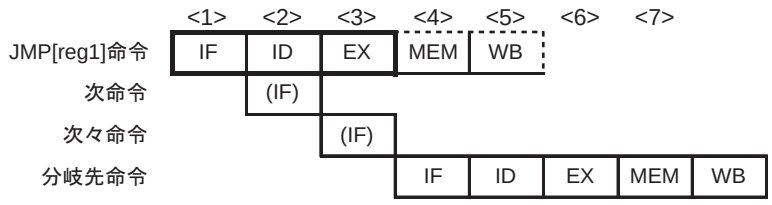


(IF): 無効となる命令フェッチ

［説　　明］ パイプラインは，IF，ID，EX，MEM，WB の 5 ステージですが，EX ステージで分岐先が決定するため，MEM ステージ，WB ステージでは何も行いません。

(b) JMP dip32 [reg1]命令

［パイプライン］



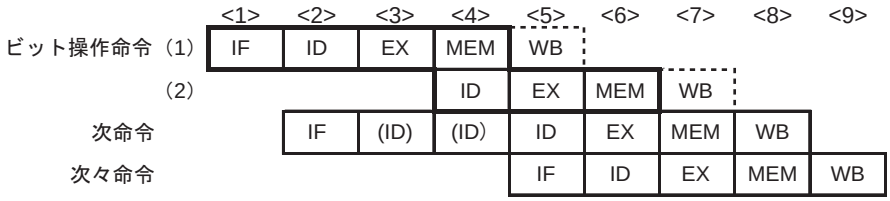
(IF): 無効となる命令フェッチ

［説　　明］ パイプラインは，IF，ID，EX，MEM，WB の 5 ステージですが，EX ステージで分岐先が決定するため，MEM ステージ，WB ステージでは何も行いません。

C. 3. 14 ビット操作命令

(1) CLR1, NOT1, SET1命令

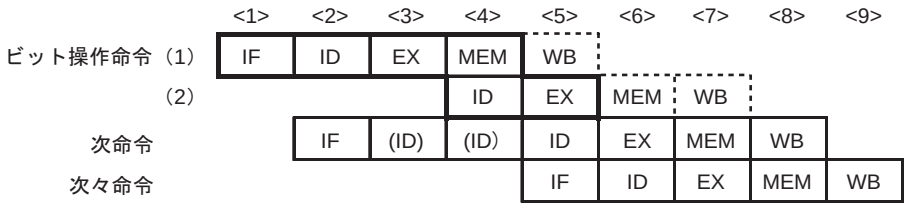
［パイプライン］



［説　　明］ パイプラインは IF, ID, EX1, MEM, EX2（通常のステージ），MEM, 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。この命令では，メモリ・アクセスがリード・モディファイ・ライトとなり，EX ステージに計 2 クロック，MEM ステージに計 2 サイクルかかります。

(2) TST1命令

［パイプライン］

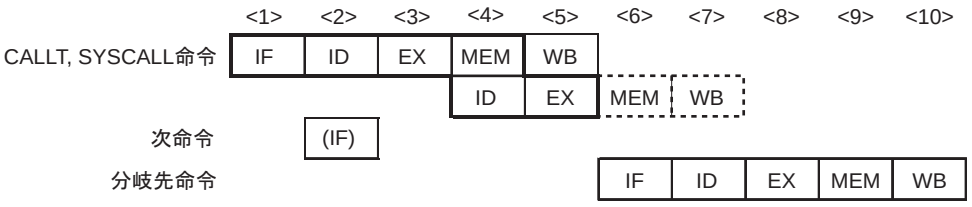


［説　　明］ パイプラインは IF, ID, EX1, MEM, EX2（通常のステージ），MEM, WB の 7 ステージですが，2 回目のメモリ・アクセス，レジスタへのデータの書き込みがないので，2 回目の MEM ステージ，WB ステージでは何も行いません。この命令では計 2 クロックかかります。

C. 3. 15 特殊命令

(1) CALLT, SYSCALL命令

［パイプライン］

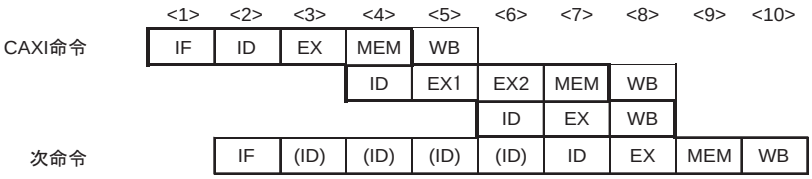


(IF): 無効となる命令フェッチ

［説　　明］ パイプラインは IF, ID, EX1, MEM, EX2, MEM, WB の 7 ステージです。ただし，2 番目の MEM ステージと WB ステージでは，メモリ・アクセスがなく，レジスタにデータ書き込みがないため，何も行いません。

(2) CAXI命令

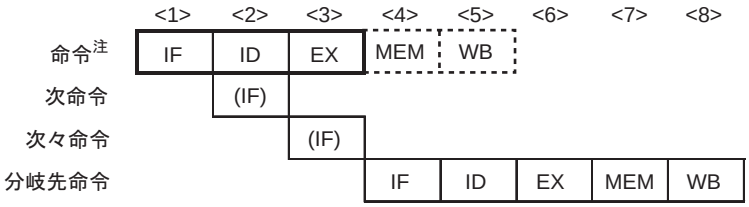
［パイプライン］



［説　　明］　パイプラインは IF, ID, EX1, MEM, EX2, EX3, MEM, WB の 8 ステージです。

(3) CTRET, EIRET, FERET, FETRAP, RETI, RIE, TRAP命令

［パイプライン］



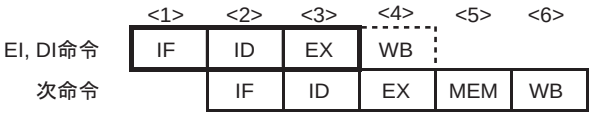
(IF): 無効となる命令フェッチ

注　CTRET, EIRET, FERET, FETRAP, RETI, RIE, TRAP命令

［説　　明］　パイプラインは，IF, ID, EX, MEM, WB の 5 ステージですが，EX ステージで分岐先が決定します。MEM ステージ，WB ステージでは何も行いません。

(4) DI, EI, LDSR命令

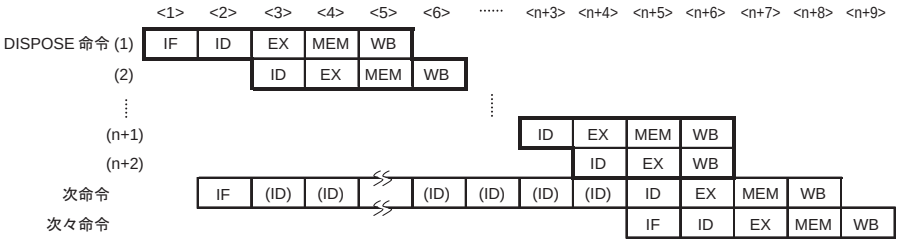
［パイプライン］



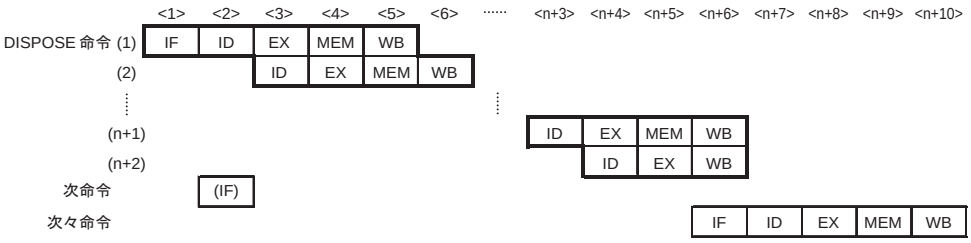
［説　　明］　パイプラインは，IF, ID, EX, WB の 4 ステージですが，WB のステージではメモリ・アクセスがなく，レジスタにデータ書き込みがないため，何も行いません。

(5) DISPOSE命令

［パイプライン］ (a) 分岐しない場合



（b）分岐する場合

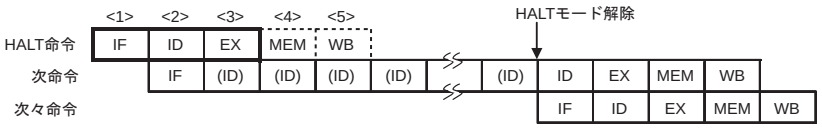


備考 n は，レジスタ・リスト（list12）で指定されるレジスタの数です。

［説　　明］ パイプラインは，IF, ID, EX, n+1 回の MEM, WB の「n+5」ステージです(n: レジスタ・リスト・ナンバ)。MEM ステージは，n+1 サイクル必要です。

(6) HALT命令

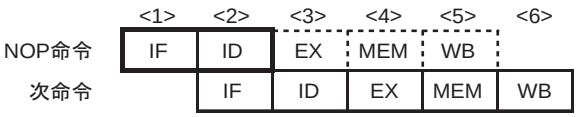
［パイプライン］



［説　　明］ パイプラインは IF, ID, EX, MEM, WB の 5 ステージですが，メモリへのアクセス，レジスタへのデータ書き込みがないので，MEM ステージ，WB ステージでは何も行いません。また，次命令では HALT モードが解除されるまで ID ステージが遅れます。

(7) NOP命令

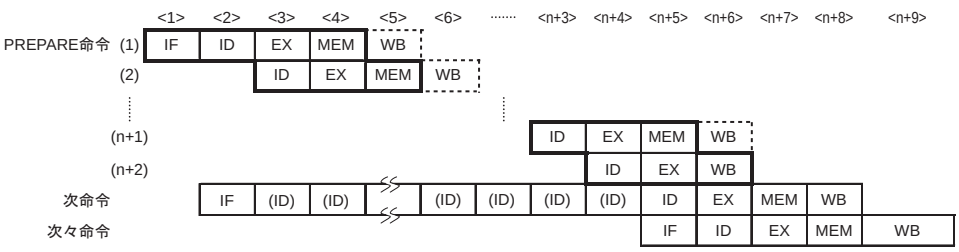
［パイプライン］



［説　　明］ パイプラインは IF, ID, EX, MEM, WB の 5 ステージですが，演算，メモリへのアクセス，レジスタへのデータ書き込みがないので，EX ステージ, MEM ステージ，WB ステージでは何も行いません。

(8) PREPARE命令

［パイプライン］

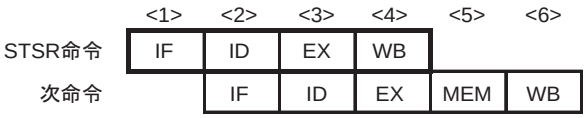


備考 n は，レジスタ・リスト（list12）で指定されるレジスタの数です。

［説　　明］ パイプラインは IF, ID, EX, n+1 回の MEM, WB の「n+5」ステージです（n: レジスタ・リスト・ナンバ）。MEM ステージは，n+1 サイクル必要です。ただし、レジスタへのデータの書き込みがないため WB ステージでは何も行いません。

(9) STSR命令

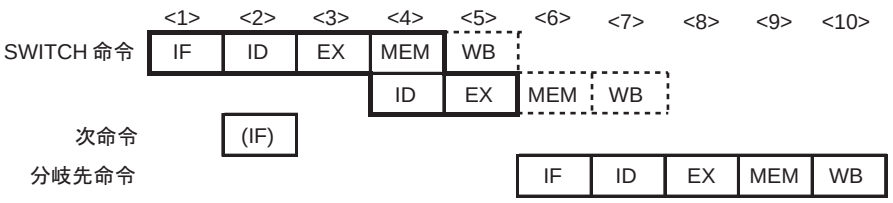
［パイプライン］



［説　　明］ パイプラインは IF, ID, EX, WB の 4 ステージです。LDSR 命令の直後に，同一レジスタを使用する STSR 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。

(10) SWITCH命令

［パイプライン］



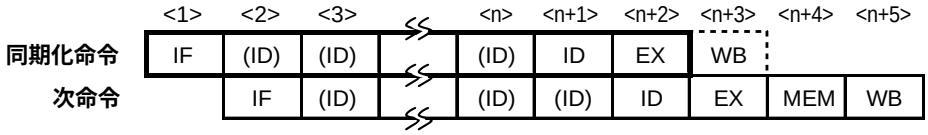
(IF): 無効となる命令フェッチ

［説　　明］ パイプラインは，IF, ID, EX1（通常の EX ステージ），MEM, EX2, MEM, WB の 7 ステージです。ただし，2 番目の MEM と WB のステージでは，メモリ・アクセスがなく，レジスタにデータ書き込みがないため，何も行いません。

(11) 同期化命令

［対象の命令］ SYNCM, SYNCP

［パイプライン］



備考 n は不定値です。

［説　　明］ 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。
レジスタへのデータの書き込みがないためWBステージでは何も行いません。

付録D V850E2S CPUと他のCPUの相違点

D. 1 V850E2, V850E2Mとの相違点

(1/2)

項目		V850E2S	V850E2M	V850E2
命令（オペランドを含む）	ADF cccc, reg1, reg2, reg3	あり		
	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	なし	あり	なし

(2/2)

項　目			V850E2S	V850E2M	V850E2
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			64 Mバイト	4 Gバイト[注2]	512 Mバイト
プログラム・カウンタ（PC）の有効ビット			32ビット[注1]	32ビット[注2]	下位29ビット
データ領域			64 Mバイト	4 G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[注3]
		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グ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				
	ソフトウエア・ページング・バンク				
	FPUステータス・バンク		なし	あり	なし
		FPEC			
	ユーザ0バンク		あり		なし
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[注4]
	FEレベル・マスカブル例外		FEINT		NMI0, NMI1[注4]
	EIレベル・マスカブル例外		INT		
	メモリ保護例外		あり（30H）		なし
	浮動小数点演算例外		なし	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		不正命令例外
			FEレベル例外（30H）		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			5段	7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. V850E2Sの命令アドレッシング範囲は64Mバイトです。EIPCのビット31-26はビット25を符号拡張した値が自動的に設定されます。

2. 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。

3. 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。

4. 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E 命令索引

E. 1 基本命令索引

[A]
ADD ... 63
ADDI ... 64
ADF ... 65
AND ... 66
ANDI ... 67

[B]
Bcond ... 68
BSH ... 70
BSW ... 71

[C]
CALLT ... 72
CAXI ... 73
CLR1 ... 74
CMOV ... 76
CMP ... 78
CTRET ... 79

[D]
DI ... 80
DISPOSE ... 81
DIV ... 83
DIVH ... 84
DIVHU ... 86
DIVQ ... 87
DIVQU ... 88
DIVU ... 89

[E]
EI ... 90
EIRET ... 91

[F]
FERET ... 92
FETRAP ... 93

[H]
HALT ... 94
HSH ... 95
HSW ... 96

[J]
JARL ... 97
JMP ... 99
JR ... 100

[L]
LD.B ... 101
LD.BU ... 102
LD.H ... 103
LD.HU ... 104
LD.W ... 105
LDSR ... 106

[M]
MAC ... 107
MACU ... 108
MOV ... 109
MOVEA ... 110
MOVHI ... 111
MUL ... 112
MULH ... 113
MULHI ... 114
MULU ... 115

[N]
NOP ... 116
NOT ... 117
NOT1 ... 118

[O]
OR ... 120
ORI ... 121

[P]
PREPARE ... 122

[R]
RETI ... 124
RIE ... 126

[S]
SAR ... 127
SASF ... 129
SATADD ... 130
SATSUB ... 132
SATSUBI ... 133
SATSUBR ... 134
SBF ... 135
SCH0L ... 136
SCH0R ... 137
SCH1L ... 138
SCH1R ... 139
SET1 ... 140
SETF ... 142
SHL ... 144
SHR ... 146
SLD.B ... 148
SLD.BU ... 149
SLD.H ... 150
SLD.HU ... 151
SLD.W ... 152
SST.B ... 153
SST.H ... 154
SST.W ... 155
ST.B ... 156
ST.H ... 157
ST.W ... 158
STSR ... 159
SUB ... 160
SUBR ... 161
SWITCH ... 162
SXB ... 163
SXH ... 164
SYNCE ... 165
SYNCM ... 166
SYNCP ... 167
SYSCALL ... 168

[T]
TRAP ... 170
TST ... 171
TST1 ... 172

[X]
XOR ... 173
XORI ... 174

[Z]
ZXB ... 175
ZXH ... 176

改訂記録	V850E2S　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	2011.12.19	ー	初版発行
1.00	2014.05.29	全体	EI レベル・ソフトウエア例外、予約命令例外、FE レベル・ソフトウエア例外、システム・コール例外　略称を変更
		第 2 編 第 2 章 レジスタ・セット	
		25-26	2. 3. 4 PSW　注とレジスタ図、表を変更
		32	表 2－3 システム・レジスタ・バンク　変更
		第 3 編 第 5 章　システム・レジスタ保護	
		219	導入文を変更
		第 3 編 第 6 章 メモリ保護	
		226	6. 1. 1 PAnL　変更
		第 3 編 第 7 章 プロセッサ保護例外	
		236	7. 1. 3 データ保護違反　説明削除
		付録 A 命令一覧	
		243	表 A－1 基本命令機能一覧（アルファベット順）（3/4）　変更
		付録 C パイプライン	
		259	C. 3. 3 (2) ワード・データ乗算命令 - (b), (c)　変更
		260	C. 3. 4 加算付き乗算命令 - (b) - (e)　変更
		268	C. 3. 15（8）PREPARE 命令　変更
		269	C. 3. 15（11）同期化命令　変更

V850E2S **ユーザーズマニュアル　アーキテクチャ編**

発行年月日　2011 **年** 12 **月** 19 **日** Rev.0.01
2014 **年** 5 **月** 29 **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 エレクトロニクス株式会社 〒100-0004 千代田区大手町2-6-2（日本ビル）

■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/

V850E2S

RENESAS

ルネサス エレクトロニクス株式会社

R01US0037JJ0100