お客様各位

カタログ等資料中の旧社名の扱いについて

2010年4月1日を以ってNECエレクトロニクス株式会社及び株式会社ルネサステクノロジが合併し、両社の全ての事業が当社に承継されております。従いまして、本資料中には旧社名での表記が残っておりますが、当社の資料として有効ですので、ご理解の程宜しくお願い申し上げます。

ルネサスエレクトロニクス ホームページ（http://www.renesas.com）

2010年4月1日
ルネサスエレクトロニクス株式会社

【発行】ルネサスエレクトロニクス株式会社（http://www.renesas.com）

【問い合わせ先】http://japan.renesas.com/inquiry

ご注意書き

1. 本資料に記載されている内容は本資料発行時点のものであり、予告なく変更することがあります。当社製品のご購入およびご使用にあたりましては、事前に当社営業窓口で最新の情報をご確認いただきますとともに、当社ホームページなどを通じて公開される情報に常にご注意ください。
2. 本資料に記載された当社製品および技術情報の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、当社は、一切その責任を負いません。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。
3. 当社製品を改造、改変、複製等しないでください。
4. 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。
5. 輸出に際しては、「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。本資料に記載されている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の目的で使用しないでください。また、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に使用することができません。
6. 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。
7. 当社は、当社製品の品質水準を「標準水準」、「高品質水準」および「特定水準」に分類しております。また、各品質水準は、以下に示す用途に製品が使われることを意図しておりますので、当社製品の品質水準をご確認ください。お客様は、当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途に当社製品を使用することができません。また、お客様は、当社の文書による事前の承諾を得ることなく、意図されていない用途に当社製品を使用することができません。当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途または意図されてい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生じた損害等に関し、当社は、一切その責任を負いません。なお、当社製品のデータ・シート、データ・ブック等の資料で特に品質水準の表示がない場合は、標準水準製品であることを表します。

   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

   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、生命維持を目的として設計されていない医療機器（厚生労働省定義の管理医療機器に相当）

   特定水準： 航空機器、航空宇宙機器、海底中継機器、原子力制御システム、生命維持のための医療機器（生命維持装置、人体に埋め込み使用するもの、治療行為（患部切り出し等）を行うもの、その他直接人命に影響を与えるもの）（厚生労働省定義の高度管理医療機器に相当）またはシステム等
8. 本資料に記載された当社製品のご使用につき、特に、最大定格、動作電源電圧範囲、放熱特性、実装条件その他諸条件につきましては、当社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。
9. 当社は、当社製品の品質および信頼性の向上に努めておりますが、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、社会的損害など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、機器またはシステムとしての出荷保証をお願いいたします。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、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
10. 当社製品の環境適合性等、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。
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固くお断りいたします。
12. 本資料に関する詳細についてのお問い合わせその他お気付きの点等がございましたら当社営業窓口までご照会ください。

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。

RENESAS

ユーザーズ・マニュアル

V850E1

32 ビット・マイクロプロセッサ・コア

アーキテクチャ編

資料番号 U14559JJ3V1UM00（第3版）
発行年月 February 2004 NS CP(K)

［メ　モ］

目次要約

第 1 章　概　　説 … 13

第 2 章　レジスタ・セット … 16

第 3 章　データ・タイプ … 37

第 4 章　アドレス空間 … 41

第 5 章　命　　令 … 47

第 6 章　割り込みと例外 … 162

第 7 章　リセット … 171

第 8 章　パイプライン … 173

第 9 章　ディバグ・モードへの移行 … 194

付録 A　注意事項 … 203

付録 B　命令一覧 … 204

付録 C　命令オペコード・マップ … 219

付録 D　V850 CPU とのアーキテクチャ上の相違点 … 224

付録 E　V850 CPU に対して V850E1 CPU で追加した命令 … 226

付録 F　総合索引 … 228

付録 G　改版履歴 … 234

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。

本製品のうち，外国為替及び外国貿易法の規定により規制貨物等（または役務）に該当するものについては，日本国外に輸出する際に，同法に基づき日本国政府の輸出許可が必要です。

- **本資料に記載されている内容は2004年2月現在のもので，今後，予告なく変更することがあります。量産設計の際には最新の個別データ・シート等をご参照ください。**
- **文書による当社の事前の承諾なしに本資料の転載複製を禁じます。当社は，本資料の誤りに関し，一切その責を負いません。**
- **当社は，本資料に記載された当社製品の使用に関連し発生した第三者の特許権，著作権その他の知的財産権の侵害等に関し，一切その責を負いません。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。**
- **本資料に記載された回路，ソフトウエアおよびこれらに関する情報は，半導体製品の動作例，応用例を説明するものです。お客様の機器の設計において，回路，ソフトウエアおよびこれらに関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，一切その責を負いません。**
- **当社は，当社製品の品質，信頼性の向上に努めておりますが，当社製品の不具合が完全に発生しないことを保証するものではありません。当社製品の不具合により生じた生命，身体および財産に対する損害の危険を最小限度にするために，冗長設計，延焼対策設計，誤動作防止設計等安全設計を行ってください。**
- **当社は，当社製品の品質水準を「標準水準」，「特別水準」およびお客様に品質保証プログラムを指定していただく「特定水準」に分類しております。また，各品質水準は，以下に示す用途に製品が使われることを意図しておりますので，当社製品の品質水準をご確認ください。**

  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

  特別水準：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，生命維持を目的として設計されていない医療機器

  特定水準：航空機器，航空宇宙機器，海底中継機器，原子力制御システム，生命維持のための医療機器，生命維持のための装置またはシステム等

当社製品のデータ・シート，データ・ブック等の資料で特に品質水準の表示がない場合は，標準水準製品であることを表します。意図されていない用途で当社製品の使用をお客様が希望する場合には，事前に当社販売窓口までお問い合わせください。

（注）

（1）本事項において使用されている「当社」とは，NECエレクトロニクス株式会社およびNEC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う。

（2）本事項において使用されている「当社製品」とは，（1）において定義された当社の開発，製造製品をいう。

M8E　02.11

はじめに

対象者　このマニュアルは，V850E1 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 V850E1 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。

- レジスタ・セット
- データ・タイプ
- 命令形式と命令セット
- 割り込みと例外
- パイプライン

読み方　このマニュアルの読者には，電気，論理回路，マイクロコンピュータの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 5 章　命　　令をお読みください。

本文欄外の★印は，本版で改訂された主な箇所を示しています。

製品タイプ　このマニュアルは，製品名をタイプに分けて説明しています。
次の表より製品に対応するタイプを確認されたあと，お読みください。

製品タイプ	製品名
タイプA	NU85E CPUコア
タイプB	NU85ET CPUコア
タイプC	NB85E, NB85ET CPUコア
タイプD	V850E/IA1, V850E/IA2, V850E/MA1, V850E/SV2
タイプE	V850E/IA3, V850E/IA4, V850E/MA3
タイプF	V850E/MA2, V850E/ME2

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　：xxxB（**端子，信号名称のあとに** B）

注　：**本文中につけた注の説明**

注意　：**気をつ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 章　概　　説 … 13

1. 1　特　　徴 … 14
1. 2　内部構成 … 15

第 2 章　レジスタ・セット … 16

2. 1　プログラム・レジスタ … 17
2. 2　システム・レジスタ … 19
2. 2. 1　**割り込み時状態退避レジスタ**（EIPC, EIPSW） … 20
2. 2. 2　NMI **時状態退避レジスタ**（FEPC, FEPSW） … 21
2. 2. 3　**割り込み要因レジスタ**（ECR） … 21
2. 2. 4　**プログラム・ステータス・ワード**（PSW） … 22
2. 2. 5　CALLT **実行時状態退避レジスタ**（CTPC, CTPSW） … 24
2. 2. 6　**例外／ディバグ・トラップ時状態退避レジスタ**（DBPC, DBPSW） … 25
2. 2. 7　CALLT **ベース・ポインタ**（CTBP） … 26
2. 2. 8　**ディバグ・インタフェース・レジスタ**（DIR） … 27
2. 2. 9　**ブレークポイント制御レジスタ** 0, 1（BPC0, BPC1） … 30
2. 2. 10　**プログラム** ID **レジスタ**（ASID） … 32
2. 2. 11　**ブレークポイント・アドレス設定レジスタ** 0, 1（BPAV0, BPAV1） … 33
2. 2. 12　**ブレークポイント・アドレス・マスク・レジスタ** 0, 1（BPAM0, BPAM1） … 34
2. 2. 13　**ブレークポイント・データ設定レジスタ** 0, 1（BPDV0, BPDV1） … 35
2. 2. 14　**ブレークポイント・データ・マスク・レジスタ** 0, 1（BPDM0, BPDM1） … 36

第 3 章　データ・タイプ … 37

3. 1　データ形式 … 37
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 … 43
4. 2. 1　**命令アドレス** … 43

4. 2. 2 **オペランド・アドレス** … 45

第 5 章　命　　令 … 47

5. 1　命令フォーマット … 47
5. 2　命令の概要 … 51
5. 3　命令セット … 55

ADD … 57
ADDI … 58
AND … 59
ANDI … 60
Bcond … 61
BSH … 63
BSW … 64
CALLT … 65
CLR1 … 66
CMOV … 67
CMP … 68
CTRET … 69
DBRET … 70
DBTRAP … 71
DI … 72
DISPOSE … 73
DIV … 75
DIVH … 76
DIVHU … 78
DIVU … 79
EI … 80
HALT … 81
HSW … 82
JARL … 83
JMP … 84
JR … 85
LD.B … 86
LD.BU … 87
LD.H … 88
LD.HU … 90
LD.W … 92
LDSR … 94
MOV … 95
MOVEA … 96
MOVHI … 97
MUL … 98

MULH … 100
MULHI … 101
MULU … 102
NOP … 104
NOT … 105
NOT1 … 106
OR … 107
ORI … 108
PREPARE … 109
RETI … 111
SAR … 113
SASF … 114
SATADD … 115
SATSUB … 117
SATSUBI … 118
SATSUBR … 119
SET1 … 120
SETF … 121
SHL … 123
SHR … 124
SLD.B … 125
SLD.BU … 127
SLD.H … 129
SLD.HU … 131
SLD.W … 133
SST.B … 135
SST.H … 136
SST.W … 138
ST.B … 140
ST.H … 141
ST.W … 143
STSR … 145
SUB … 146
SUBR … 147
SWITCH … 148
SXB … 149
SXH … 150
TRAP … 151
TST … 152
TST1 … 153
XOR … 154
XORI … 155
ZXB … 156

ZXH … 157
5.4 命令実行クロック数 … 158

第6章 割り込みと例外 … 162

6.1 割り込み処理 … 163
6.1.1 **マスカブル割り込み** … 163
6.1.2 **ノンマスカブル割り込み** … 165
6.2 例外処理 … 166
6.2.1 **ソフトウエア例外** … 166
6.2.2 **例外トラップ** … 167
6.2.3 **ディバグ・トラップ** … 168
6.3 割り込み，例外処理からの復帰 … 169
6.3.1 **割り込み，ソフトウエア例外からの復帰** … 169
6.3.2 **例外トラップ，ディバグ・トラップからの復帰** … 170

第7章 リセット … 171

7.1 リセット後のレジスタの状態 … 171
7.2 起　　動 … 172

第8章 パイプライン … 173

8.1 特　　徴 … 174
8.1.1 **ノンブロッキング・ロード／ストア** … 175
8.1.2 **2クロック分岐** … 176
8.1.3 **効率的なパイプライン処理** … 177
8.2 各命令実行時のパイプラインの流れ … 178
8.2.1 **ロード命令** … 179
8.2.2 **ストア命令** … 180
8.2.3 **乗算命令** … 180
8.2.4 **算術演算命令** … 181
8.2.5 **飽和演算命令** … 182
8.2.6 **論理演算命令** … 182
8.2.7 **分岐命令** … 182
8.2.8 **ビット操作命令** … 184
8.2.9 **特殊命令** … 184
8.2.10 **ディバグ機能用命令** … 188
8.3 パイプラインの乱れ … 189
8.3.1 **アライン・ハザード** … 189
8.3.2 **ロード命令実行結果の参照** … 189
8.3.3 **乗算命令実行結果の参照** … 190
8.3.4 EIPC, FEPC **を対象とする** LDSR **命令実行結果の参照** … 191

8. 3. 5　プログラム作成時の注意点 … 191
8. 4　パイプラインに関する補足事項 … 192
8. 4. 1　ハーバード・アーキテクチャ … 192
8. 4. 2　ショート・パス … 192

★ 第 9 章　ディバグ・モードへの移行 … 194

9. 1　ディバグ・モードへの移行方法 … 194
9. 2　注意事項 … 201

★ 付録 A　注意事項 … 203

A. 1　sld 命令と割り込み競合に関する制限事項 … 203
A. 1. 1　内　　容 … 203
A. 1. 2　回 避 策 … 203

付録 B　命令一覧 … 204

付録 C　命令オペコード・マップ … 219

付録 D　V850 CPU とのアーキテクチャ上の相違点 … 224

付録 E　V850 CPU に対して V850E1 CPU で追加した命令 … 226

付録 F　総合索引 … 228

F. 1　50 音で始まる語句の索引 … 228
F. 2　数字，アルファベットで始まる語句の索引 … 231

★ 付録 G　改版履歴 … 234

G. 1　本版で改訂された主な箇所 … 234
G. 2　前版までの改版履歴 … 235

第1章　概　　説

リアルタイム制御システムには，HDD（Hard disk drive），PPC（Plain paper copier），プリンタ，ファクシミリをはじめとするOA機器，エンジン制御，ABS（Antilock braking system）制御などの各種自動車電装機器，NC（Numerical control）工作機，各種コントローラなどのFA機器などがあります。従来，これらの分野では8ビットまたは16ビットのマイクロコンピュータが用いられてきましたが，機器の制御の複雑化にしたがって，マイクロコンピュータに要求される機能も高度化するとともに，命令セットも複雑になり，ハードウエア規模が増大化してきました。同時に，機器の性能向上に対応すべくマイクロコンピュータの動作周波数の高速化もあわせて求められています。

これらの要求に答えるために開発されたV850シリーズは，よりシンプルなハードウエア構成により最大限の性能を実現できる手段としてRISCアーキテクチャを取り入れることにより，従来のCISC型シングルチップ・マイクロコンピュータ78K/IIIシリーズ，78K/IVシリーズの約15倍以上の性能を低コストで実現する次世代のシングルチップ・マイクロコンピュータです。

V850シリーズでは，従来のRISC型CPUの基本命令に加えて，各種応用機器，特にディジタル・サーボ制御の応用に最適な命令として，ハードウエア乗算器による乗算命令，飽和演算命令，ビット操作命令などを用意しました。また，コンパイラにおける高コード効率を最優先に意識した命令フォーマットを採用して，オブジェクト・コード・サイズのコンパクト化を図っています。

「V850E1 CPU」は，システム・オン・チップ時代のシステムLSIの核となるCPUコアとして新たに開発したASIC用32ビットRISC型CPUコアです。V850シリーズ搭載のCPUコア「V850 CPU」に対し，外部バス・インタフェースの性能を強化し，C言語のswitch文処理，テーブル・ルックアップの分岐，スタック・フレームの生成／削除，データ変換など，主に高級言語に対応した命令などを追加することにより，制御系だけでなく，データ処理系にも対応したCPUコアです。

なお，命令コードは，V850 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，V850 CPU搭載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 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：83
- 32 ビット汎用レジスタ：32 本
- ロング／ショート形式を持つロード／ストア命令
- 3 オペランド命令
- 1 クロック・ピッチの 5 段パイプライン構造
- レジスタ／フラグ・ハザードのインタロックをハードウエアにより対処
- メモリ空間：プログラム空間 … 64M バイト・リニア
  データ空間 … 4G 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

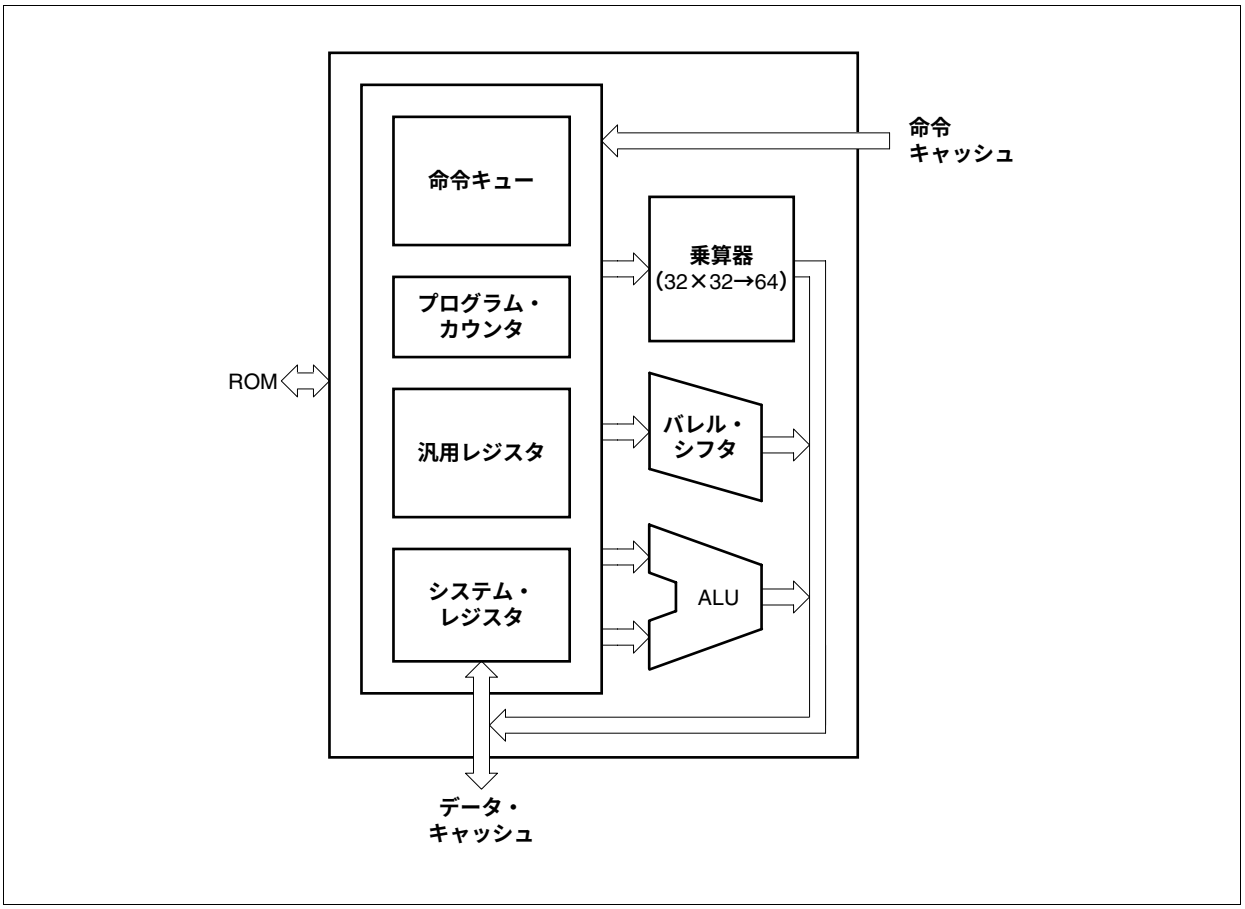
1. 2　内部構成

V850E1 CPU は，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 5 段パイプライン制御により 1 クロックで実行します。

乗算器（32 ビット ×32 ビット乗算），バレル・シフタ（32 ビット／1 クロック）などの専用ハードウエアを内蔵し，複雑な命令処理の高速化を図っています。

内部ブロック図を図 1－1 に示します。

図1－1　V850E1 CPUの内部ブロック図



第2章　レジスタ・セット

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

★ **注　ディバグ機能用に予約されたレジスタです。タイプ** A, B **の製品のみ使用できます。その他のタイプの製品では使用できません。**

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・レジスター覧

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 r2 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，**CA850（C コンパイラ・パッケージ）ユーザーズ・マニュアル　アセンブリ言語編**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 を使用する際には次のような注意が必要です。

(a) r0, 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を使用する演算やオフセット0のアドレシングで使用されます。

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, r3-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。

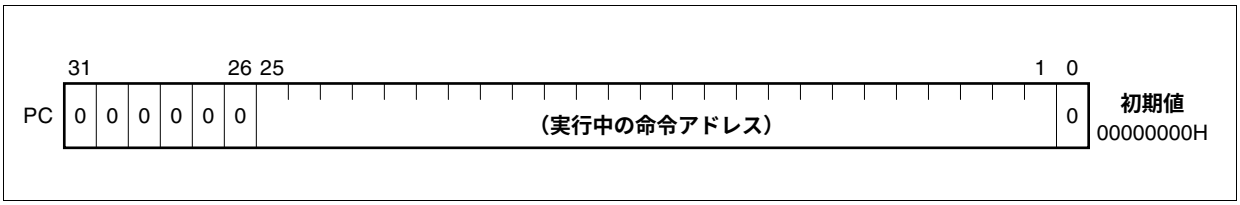
(c) r2

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。下位の 26 ビットが有効で，ビット 31-26 は将来の機能拡張のために予約されています（0 に固定）。ビット 25 からビット 26 へのキャリーがあっても無視します。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。

図2－2　プログラム・カウンタ（PC）



2. 2　システム・レジスタ

システム・レジスタは，CPU の状態制御，割り込み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，システム・レジスタのロード／ストア命令（LDSR, 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・レジスタ番号

	レジスタ番号	レジスタ名	オペランド指定の可否	
			LDSR 命令	STSR 命令
	0	割り込み時状態退避レジスタ（EIPC）	○	○
	1	割り込み時状態退避レジスタ（EIPSW）	○	○
	2	NMI 時状態退避レジスタ（FEPC）	○	○
	3	NMI 時状態退避レジスタ（FEPSW）	○	○
	4	割り込み要因レジスタ（ECR）	×	○
	5	プログラム・ステータス・ワード（PSW）	○	○
	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	CALLT 実行時状態退避レジスタ（CTPC）	○	○
	17	CALLT 実行時状態退避レジスタ（CTPSW）	○	○
★	18	例外／ディバグ・トラップ時状態退避レジスタ（DBPC）	○	○注1
★	19	例外／ディバグ・トラップ時状態退避レジスタ（DBPSW）	○	○注1
	20	CALLT ベース・ポインタ（CTBP）	○	○
★	21	ディバグ・インタフェース・レジスタ（DIR）	○注1	○
	22	ブレークポイント制御レジスタ 0, 1（BPC0, BPC1）注2	○注1	○注1
	23	プログラム ID レジスタ（ASID）	○	○
	24	ブレークポイント・アドレス設定レジスタ 0, 1（BPAV0, BPAV1）注2	○注1	○注1
	25	ブレークポイント・アドレス・マスク・レジスタ 0, 1（BPAM0, BPAM1）注2	○注1	○注1
	26	ブレークポイント・データ設定レジスタ 0, 1（BPDV0, BPDV1）注2	○注1	○注1
	27	ブレークポイント・データ・マスク・レジスタ 0, 1（BPDM0, BPDM1）注2	○注1	○注1
	28-31	（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 1. タイプ A, B の製品のディバグ・モード時だけアクセス可能です。その他のタイプの製品ではアクセス禁止です。アクセスした場合の動作は保証しません。

2. 実際にアクセスされるレジスタは，DIR.CS ビットによって指定されます。

注意　LDSR 命令により EIPC か FEPC，または CTPC のビット 0 をセット（1）したあと，割り込み処理を行い，RETI 命令で復帰する際に，ビット 0 の値は無視されます（PC のビット 0 が 0 固定のため）。EIPC, FEPC, CTPC に値を設定する場合は，偶数値（ビット 0 = 0）を設定してください。

備考　○：アクセス可能
×：アクセス禁止

2. 2. 1　割り込み時状態退避レジスタ（EIPC, EIPSW）

割り込み時状態退避レジスタには，EIPC と EIPSW があります。

ソフトウエア例外やマスカブル割り込みが発生した場合，プログラム・カウンタ（PC）の内容が EIPC に，プログラム・ステータス・ワード（PSW）の内容が EIPSW に退避されます（ノンマスカブル割り込み（NMI）発生時には，NMI 時状態退避レジスタ（FEPC, FEPSW）に退避されます）。

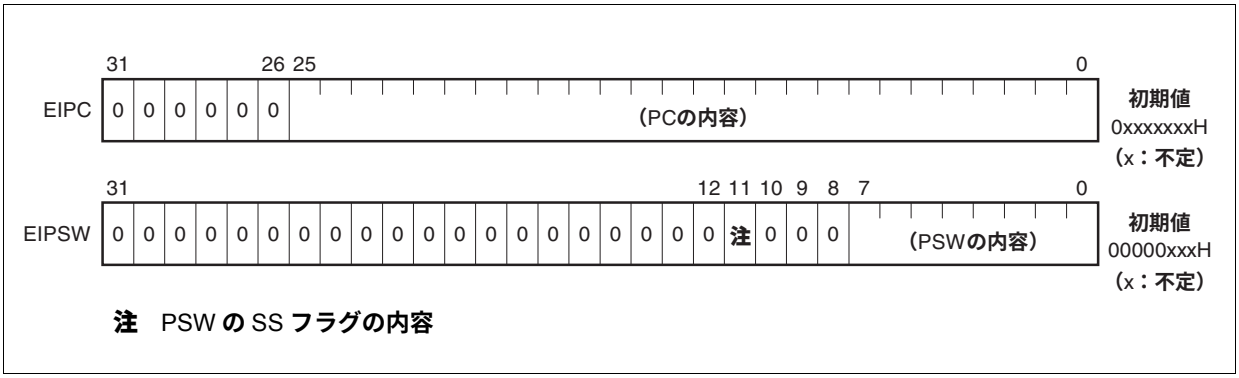
EIPC には，一部の命令を除き，ソフトウエア例外やマスカブル割り込みが発生したときに実行していた命令の次の命令のアドレスが退避されます（**表 6－1　割り込み，例外コード一覧**参照）。

EIPSW には，現在の PSW の内容が退避されます。

割り込み時状態退避レジスタは 1 組しかないため，多重割り込み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。

なお，EIPC のビット 31-26 と EIPSW のビット 31-12, 10-8 は，将来の機能拡張のために予約されています（0 に固定）。

図2－3　割り込み時状態退避レジスタ（EIPC, EIPSW）



2. 2. 2　NMI 時状態退避レジスタ（FEPC, FEPSW）

NMI 時状態退避レジスタには，FEPC と FEPSW があります。

ノンマスカブル割り込み（NMI）が発生した場合，プログラム・カウンタ（PC）の内容が FEPC に，プログラム・ステータス・ワード（PSW）の内容が FEPSW に退避されます。

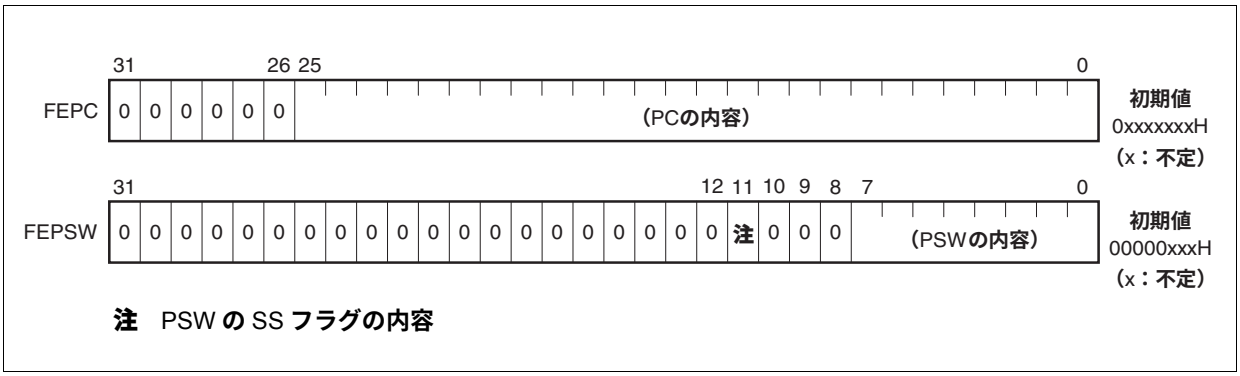
FEPC には，一部の命令を除き，NMI が発生したときに実行していた命令の次の命令のアドレスが退避されます（**表 6－1　割り込み，例外コード一覧**参照）。

FEPSW には，現在の PSW の内容が退避されます。

NMI 時状態退避レジスタは 1 組しかないため，多重割り込み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。

なお，FEPC のビット 31-26 と FEPSW のビット 31-12, 10-8 は，将来の機能拡張のために予約されています（0 に固定）。

図2－4　NMI時状態退避レジスタ（FEPC, FEPSW）

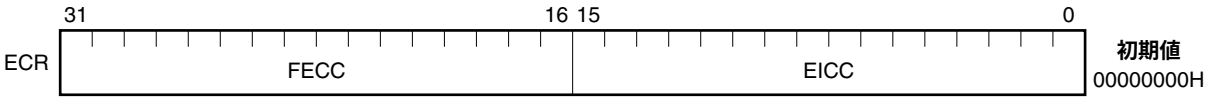


注　PSW の SS フラグの内容

2. 2. 3　割り込み要因レジスタ（ECR）

割り込み要因レジスタ（ECR）は，例外や割り込みが発生した場合に，その要因を保持するレジスタです。ECR が保持する値は，割り込み要因ごとにコード化された例外コードです（**表 6－1　割り込み，例外コード一覧**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。

図2－5　割り込み要因レジスタ（ECR）



ビット位置	ビット名	意　味
31-16	FECC	ノンマスカブル割り込み（NMI）の例外コード
15-0	EICC	例外，マスカブル割り込みの例外コード

2. 2. 4　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（PSW）は，プログラムの状態（命令実行の結果）や CPU の状態を示すフラグの集合です。

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。ただし，ID フラグをセット（1）する場合，LDSR 命令実行中から割り込み要求の受け付けを禁止します。

なお，ビット 31-12, 10-8 は，将来の機能拡張のために予約されています（0 に固定）。

図2－6　プログラム・ステータス・ワード（PSW）（1/2）

	31																			12	11	10	9	8	7	6	5	4	3	2	1	0	
PSW	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	SS	0	0	0	NP	EP	ID	SAT	CY	OV	S	Z	初期値 00000020H

ビット位置	フラグ名	意　味
11	SS 注	このフラグをセット（1）すると，シングルステップ実行で動作します（各命令の実行ごとにディバグ・トラップを発生）。 このフラグは，割り込み処理ルーチンに分岐する際にクリア（0）されます。 また，DIR レジスタの SE ビット = 0 の場合は，このフラグはセットされません（0 に固定）。
7	NP	ノンマスカブル割り込み（NMI）処理中であることを示します。NMI 要求が受け付けられるとセット（1）され，多重割り込みを禁止します。 0：NMI 処理中でない。 1：NMI 処理中である。
6	EP	例外処理中であることを示します。例外の発生でセット（1）されます。なお，このビットがセットされても割り込み要求は受け付けます。 0：例外処理中でない。 1：例外処理中である。
5	ID	マスカブル割り込み要求を受け付ける状態かどうかを示します。 0：割り込み許可（EI） 1：割り込み禁止（DI）

★ **注　タイプ** A, B **の製品のみ使用できます。その他のタイプの製品では使用できません。**

図2－6　プログラム・ステータス・ワード（PSW）（2/2）

ビット位置	フラグ名	意　味
4	SAT 注	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。 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，またはボローは発生していない。 1：キャリー，またはボローが発生した。
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：オーバフローは発生していない。 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。 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。 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	0	0	演算結果そのもの
負（最大値を越えない）	保持		1	

2. 2. 5　CALLT 実行時状態退避レジスタ（CTPC, CTPSW）

CALLT 実行時状態退避レジスタには，CTPC と CTPSW があります。

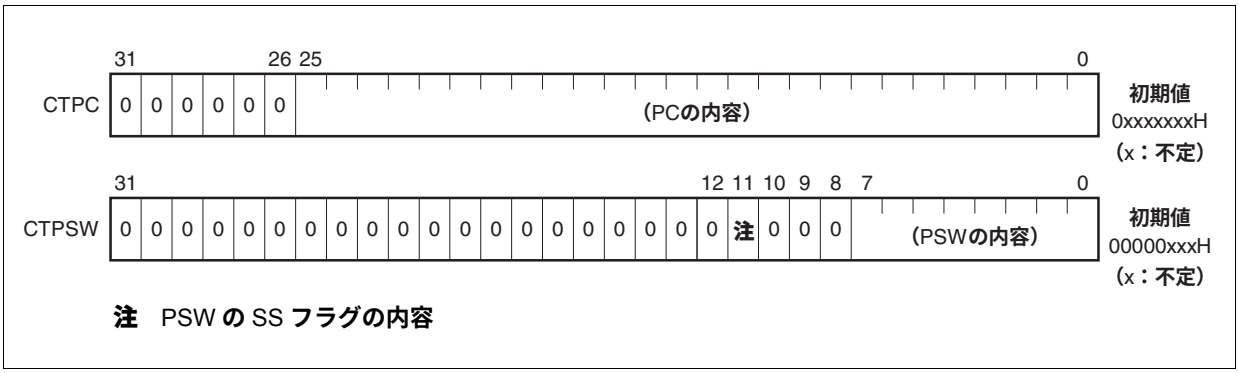
CALLT 命令が実行されると，プログラム・カウンタ（PC）の内容が CTPC に，プログラム・ステータス・ワード（PSW）の内容が CTPSW に退避されます。

CTPC に退避される内容は，CALLT 命令の次の命令のアドレスです。

CTPSW には，現在の PSW の内容が退避されます。

なお，CTPC のビット 31-26 と CTPSW のビット 31-12, 10-8 は，将来の機能拡張のために予約されています（0 に固定）。

図2－7　CALLT実行時状態退避レジスタ（CTPC, CTPSW）



2. 2. 6　例外／ディバグ・トラップ時状態退避レジスタ（DBPC, DBPSW）

例外／ディバグ・トラップ時状態退避レジスタには，DBPC と DBPSW があります。

★　例外トラップ，ディバグ・トラップ[注]，ディバグ・ブレークが発生，またはシングルステップ動作の実行時に，プログラム・カウンタ（PC）の内容が DBPC に，プログラム・ステータス・ワード（PSW）の内容が DBPSW に退避されます。

DBPC に退避される内容は，次のとおりです。

★　**表2－3　DBPCに退避される内容**

退避要因		DBPCに退避される内容
例外トラップ発生		例外トラップの発生要因となった命令の次の命令のアドレス
ディバグ・トラップ発生		ディバグ・トラップの発生要因となった命令の次の命令のアドレス
ディバグ・ブレーク発生	実行系トラップ	ブレークの発生要因となった命令のアドレス
	ミス・アライン・アクセス例外	
	アラインメント・エラー例外	
	アクセス系トラップ	ブレークの発生要因となった命令の次の命令のアドレス
シングルステップ動作実行		次に実行される命令（ディバグ・モニタ・ルーチンからの復帰時に実行される命令）のアドレス

備考　退避要因の詳細については，**第9章　ディバグ・モードへの移行**を参照してください。

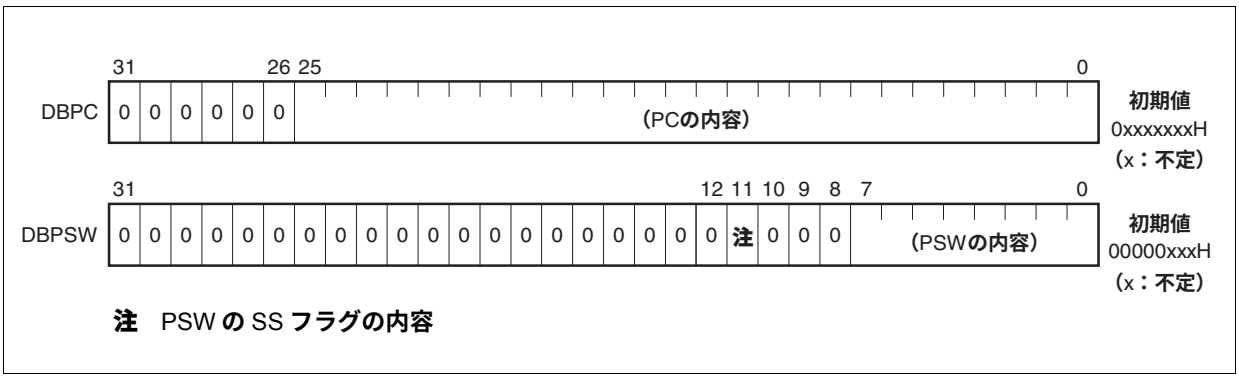
DBPSW には，現在の PSW の内容が退避されます。

★　このレジスタへのリードは，ディバグ・モード時（DIR.DM ビット = 1）だけ可能です（ライトは常に可能）。ユーザ・モード時（DM ビット= 0）に読み出した場合のリード値は不定となります。

なお，DBPC のビット 31-26 と DBPSW のビット 31-12, 10-8 は，将来の機能拡張のために予約されています（0 に固定）。

★　**注**　ディバグ・トラップは，タイプ C の製品ではサポートしていません。

図2－8　例外／ディバグ・トラップ時状態退避レジスタ（DBPC, DBPSW）



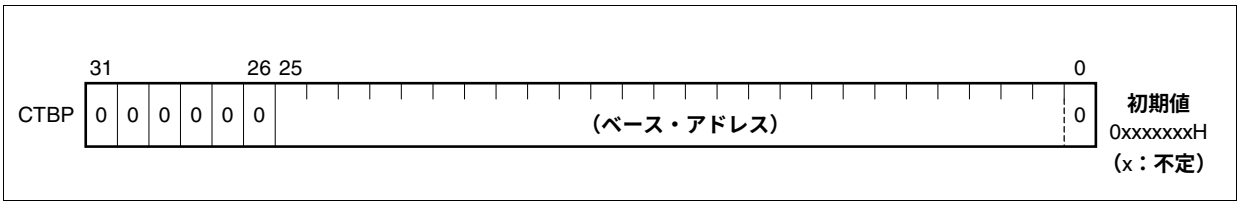
注　PSW の SS フラグの内容

2. 2. 7　CALLT ベース・ポインタ（CTBP）

CALLT ベース・ポインタ（CTBP）は，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（ビット 0 は 0 に固定）。

なお，ビット 31-26 は，将来の機能拡張のために予約されています（0 に固定）。

図2－9　CALLTベース・ポインタ（CTBP）



2. 2. 8　ディバグ・インタフェース・レジスタ（DIR）

ディバグ・インタフェース・レジスタ（DIR）は，ディバグ機能の制御や状態を示します。

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。

このレジスタへのライトは，ディバグ・モード時（DM ビット = 1）だけ可能です（ビット 3, 1 を除く）。リードは常に可能です。

また，ユーザ・モード時（DM ビット = 0）は，ビット 14-8, 6-4, 2, 1 は不定となります。

なお，ビット 31-15, 7 は，将来の機能拡張のために予約されています（0 に固定）。

★ **注意　ディバグ・インタフェース・レジスタ（DIR）は，タイプ A, B の製品のみ使用できます。その他のタイプの製品では使用できません。**

★ **図2－10　ディバグ・インタフェース・レジスタ（DIR）（1/3）**

	31															15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
DIR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	SQ	RE	CS	CE	MA	AE	SE	0	IN	T1	T0	CM	MT	AT	DM	初期値 00000040H

ビット位置	ビット名	意　味
★ 14	SQ注1, 2	シーケンシャル・ブレーク・モードを設定します。（チャネル0→1の順にブレークが発生した場合にブレーク）を設定します。 0：通常ブレーク・モード 1：シーケンシャル・ブレーク・モード
★ 13	RE注1, 2	レインジ・ブレーク・モードを設定します。（チャネル0, 1に同時にブレークが発生した場合だけブレーク）を設定します。 0：通常ブレーク・モード 1：レインジ・ブレーク・モード
12	CS注2	ブレーク・レジスタ・バンクを設定します。 0：バンク0レジスタ（チャネル0制御レジスタ）を選択 1：バンク1レジスタ（チャネル1制御レジスタ）を選択
11	CE	COMBO割り込みの許可／禁止を設定します。 0：COMBO割り込み禁止 1：COMBO割り込み許可
10	MA	ミス・アライン・アクセス例外の検出の許可／禁止を設定します。 0：ミス・アライン・アクセス例外の検出を禁止 1：ミス・アライン・アクセス例外の検出を許可
9	AE	アラインメント・エラー例外の検出の許可／禁止を設定します。 0：アラインメント・エラー例外の検出を禁止 1：アラインメント・エラー例外の検出を許可

★ **注**1. SQ, REビットは，常にどちらか一方をセット（1），または両ビットともクリア（0）してください。両ビットともセット（1）した場合の動作は保証しません。

★ 2. INビットがセット（1）されていると，SQ, RE, CSビットへのライトはできません。また，INビットをセット（1）すると各ビットは，自動的にクリア（0）されます。

図2－10　ディバグ・インタフェース・レジスタ（DIR）（2/3）

ビット位置	ビット名	意　味
8	SE	PSWのSSフラグへの書き込みの許可／禁止を設定します。 0：SSフラグへの書き込みを禁止（SSフラグは0に固定） 1：SSフラグへの書き込みを許可
6	IN[注1]	ディバグ機能のリセットにより，セット（1）されます。 リセット後はセット（1）されるため，必ずクリア（0）してください（このビットがセット（1）されていると，SQ, RE, CSビットへのライトができません。また，T1, T0ビットが動作しません）。
5	T1[注1, 2]	チャネル1のブレークが発生するとセット（1）されます。 0を設定するとクリア（0）されます。[注4]
4	T0[注1, 2]	チャネル0のブレークが発生するとセット（1）されます。 0を設定するとクリア（0）されます。[注4]
3	CM[注3]	COMBO割り込みルーチン，またはディバグ・モニタ・ルーチン2に移行するとセット（1）されます。 このビットへのライトはできません。
2	MT[注1]	ミス・アライン・アクセス例外が検出されるとセット（1）されます。 0を設定するとクリア（0）されます。[注4]
1	AT[注1]	アラインメント・エラー例外が検出されるとセット（1）されます。 0を設定するとクリア（0）されます。[注4]
0	DM[注3]	ディバグ・モードに移行するとセット（1）され，ユーザ・モードに移行するとクリア（0）されます。 このビットへのライトはできません。

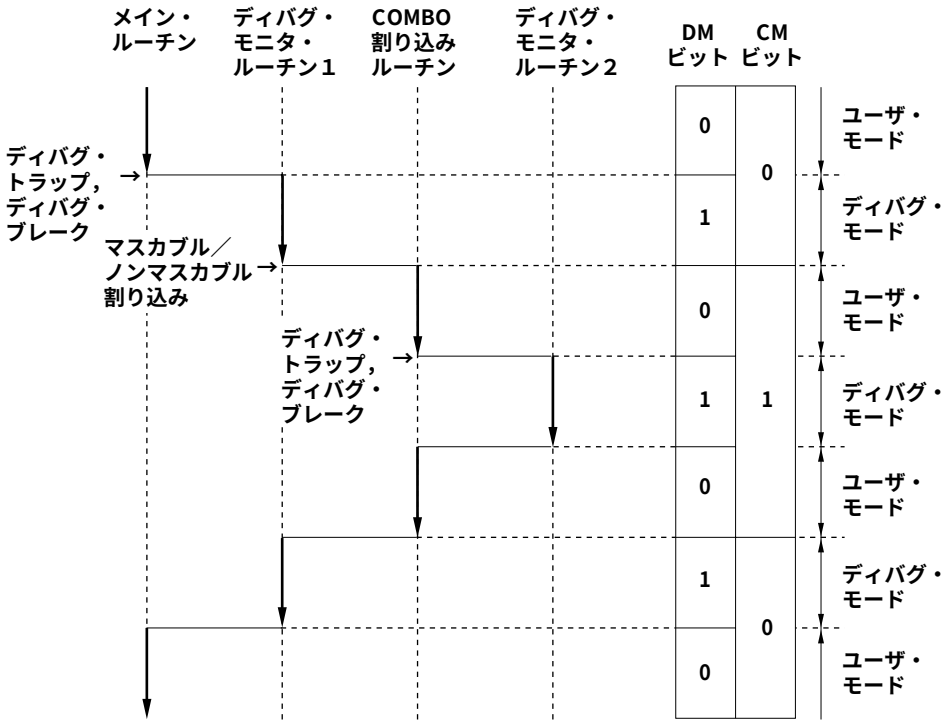
備考　注の説明は次ページに記載しています。

図2－10　ディバグ・インタフェース・レジスタ（DIR）（3/3）

注1. IN, T1, T0, MT, ATビットは，セット（1）されると自動的にクリア（0）されません（LDSR命令によってのみクリア（0）できます）。

2. INビットがセット（1）されていると，T1, T0ビットは動作しません（ブレークが発生してもセット（1）されません）。また，INビットをセット（1）すると自動的にクリア（0）されます。

3. DM, CMビットは，次のように変化します。



4. T1, T0, MT, ATビットは，ユーザ・プログラムによる任意のセット（1）はできません。

2. 2. 9　ブレークポイント制御レジスタ 0, 1（BPC0, BPC1）

ブレークポイント制御レジスタ 0, 1（BPC0, BPC1）は，ディバグ機能の制御や状態を示します。

DIR.CS ビットの設定により，どちらか一方のレジスタが有効となります。

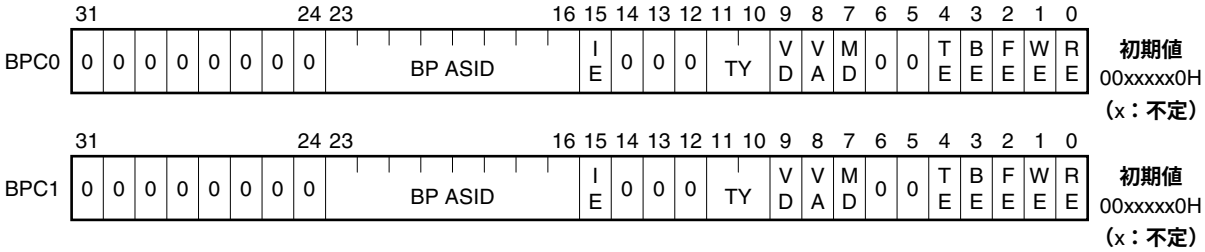
LDSR 命令を使用してこのレジスタの各ビットの内容を変更した場合は，LDSR 命令実行終了直後から変更内容が有効となります（FE ビットをセット（1）した場合は，有効となるタイミングが遅れますが，DBRET 命令を実行したあとには，必ず設定が反映されます）。

このレジスタへの設定は，ディバグ・モード時（DIR.DM ビット = 1）だけ可能です。ユーザ・モード時（DM ビット = 0）は，ビット 0 = 0，ビット 23-15, 11-7, 4-1 は不定となります。

なお，ビット 31-24, 14-12, 6, 5 は，将来の機能拡張のために予約されています（0 に固定）。

★　**注意　ブレークポイント制御レジスタ 0, 1（BPC0, BPC1）は，タイプ A, B の製品のみ使用できます。その他のタイプの製品では使用できません。**

図2－11　ブレークポイント制御レジスタ0, 1（BPC0, BPC1）



ビット位置	ビット名	意　味
23-16	BP ASID	ブレークを発生させるプログラム ID を設定します（IE ビット = 1 の場合のみ有効）。
15	IE	BP ASID ビットと ASID レジスタに設定されているプログラム ID の比較を設定します。 0：比較しない。 1：比較する。
11, 10	TY	ブレークを検出するアクセスの種類を設定します。 0,0：すべてのデータ・タイプによるアクセス 0,1：バイト・アクセス（ビット操作を含む） 1,0：ハーフワード・アクセス 1,1：ワード・アクセス なお，実行系トラップの場合は，このレジスタの設定内容は無視されます。
9	VD	データ・コンパレータの一致条件を設定します。 0：一致でブレーク 1：不一致でブレーク
8	VA	アドレス・コンパレータの一致条件を設定します。 0：一致でブレーク 1：不一致でブレーク
★ 7	MD	データ・コンパレータの動作を設定します。 0：データが条件に一致したときブレーク 1：VD ビットや BPDVx, BPDMx レジスタの設定に関係なくデータの一致判定（データ・コンパレータ）は無視
4	TE 注1	トリガ出力の許可／禁止を設定します。 0：トリガ出力禁止 1：トリガ出力許可（チャネル 0, 1 のブレーク発生時に，対応トリガを出力）
3	BE 注1	チャネル 0, 1 のブレークを CPU へ通知する／しないを設定します。 0：通知しない 1：通知する（ブレークする）
★ 2	FE	命令実行アドレス一致によるブレーク／トリガの許可／禁止を設定します。 0：ブレーク／トリガ禁止 1：ブレーク／トリガ許可注2
★ 1	WE	データ・ライト時のブレーク／トリガの許可／禁止を設定します。 0：ブレーク／トリガ禁止 1：ブレーク／トリガ許可注3
★ 0	RE	データ・リード時のブレーク／トリガの許可／禁止を設定します。 0：ブレーク／トリガ禁止 1：ブレーク／トリガ許可注3

★ **注**1. TE, BEビットはタイプBの製品のみ設定可能です。その他の製品では，“0”に固定されます（ただし，BEビットは“0”に固定されますが，CPUへのブレークの通知は行います）。

★ 2. FEビットをセット（1）した場合は，必ずWE, REビットをクリア（0）してください。

★ 3. WEビットまたはREビットをセット（1）した場合は，必ずFEビットをクリア（0）してください。

2. 2. 10　プログラム ID レジスタ（ASID）

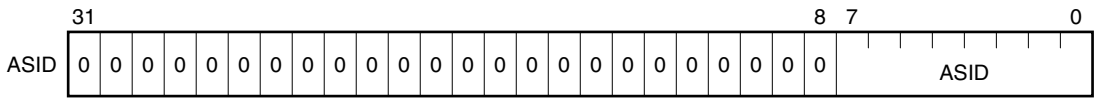
現在実行中のプログラム ID を設定します。

★　プログラム ID は，同じアドレス領域の RAM に異なるプログラムをダウンロードして実行する際に，特定のプログラムの実行時のみディバグ・モードへの移行を行う必要がある場合などに使用します。BPCn.IE ビットをセット（1）している場合は，BPCn.BP ASID ビットと ASID レジスタに設定したプログラム ID が一致しないと，ブレーク条件が一致してもディバグ・モードには移行しません（n = 0, 1）。

なお，ビット 31-8 は，将来の機能拡張のために予約されています（0 に固定）。

★　**注意　プログラム ID レジスタ（ASID）は，タイプ A, B の製品のみ使用できます。その他のタイプの製品では使用できません。**

図2－12　プログラムIDレジスタ（ASID）



ビット位置	フラグ名	意　味
7-0	ASID	現在実行中のプログラム ID

2. 2. 11　ブレークポイント・アドレス設定レジスタ 0, 1（BPAV0, BPAV1）

アドレス・コンパレータで使用するブレークポイント・アドレスを設定します。

DIR.CS ビットの設定により，どちらか一方のレジスタが有効となります。

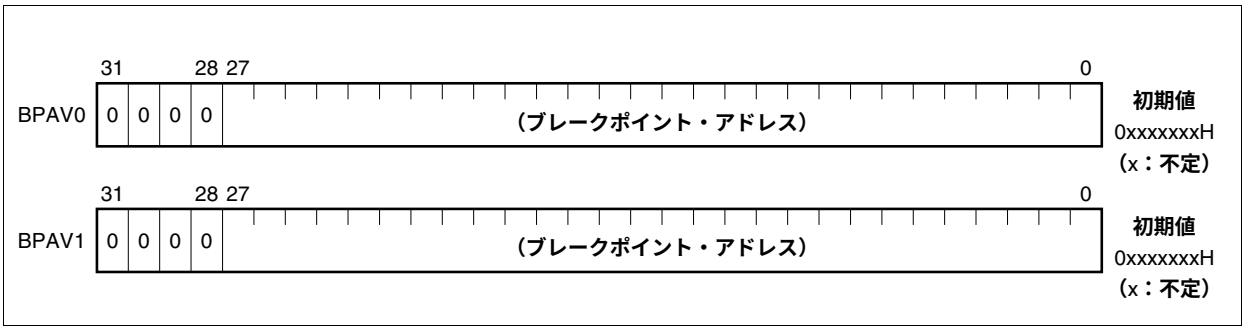
★　このレジスタへのリード／ライトは，ディバグ・モード時（DIR.DM ビット = 1）だけ可能です。ユーザ・モード時（DM ビット = 0）に読み出した場合のリード値は不定となります。

なお，使用しない場合は，必ず各ビットをセット（1）してください。

また，ビット 31-28 は，将来の機能拡張のために予約されています（0 に固定）。

★　**注意　ブレークポイント・アドレス設定レジスタ 0, 1（BPAV0, BPAV1）は，タイプ A, B の製品のみ使用できます。その他のタイプの製品では使用できません。**

図2－13　ブレークポイント・アドレス設定レジスタ0, 1（BPAV0, BPAV1）



2. 2. 12　ブレークポイント・アドレス・マスク・レジスタ 0, 1（BPAM0, BPAM1）

アドレス比較のビット・マスクを設定します（1 でマスク）。

DIR.CS ビットの設定により，どちらか一方のレジスタが有効となります。

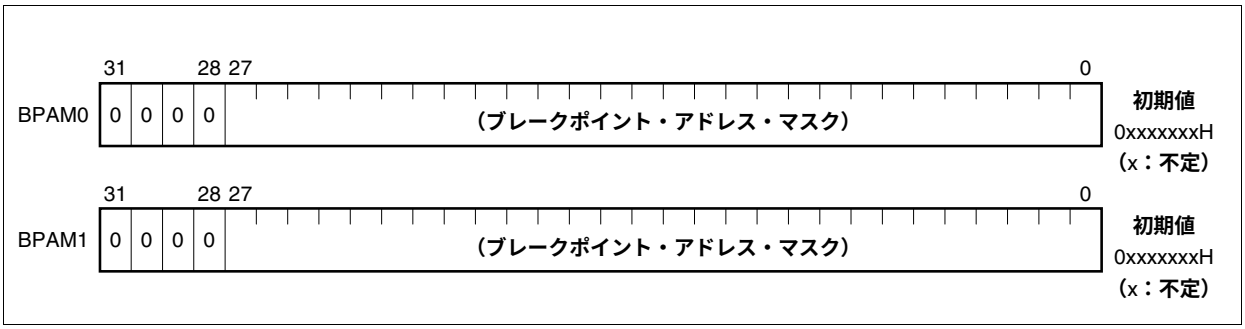
★ このレジスタへのリード／ライトは，ディバグ・モード時（DIR.DM ビット = 1）だけ可能です。ユーザ・モード時（DM ビット = 0）に読み出した場合のリード値は不定となります。

なお，使用しない場合は，必ず各ビットをセット（1）してください。

また，ビット 31-28 は，将来の機能拡張のために予約されています（0 に固定）。

★ **注意　ブレークポイント・アドレス・マスク・レジスタ 0, 1（BPAM0, BPAM1）は，タイプ A, B の製品のみ使用できます。その他のタイプの製品では使用できません。**

図2－14　ブレークポイント・アドレス・マスク・レジスタ0, 1（BPAM0, BPAM1）



2. 2. 13　ブレークポイント・データ設定レジスタ 0, 1（BPDV0, BPDV1）

データ・コンパレータで使用するブレークポイント・データを設定します。

DIR.CS ビットの設定により，どちらか一方のレジスタが有効となります。

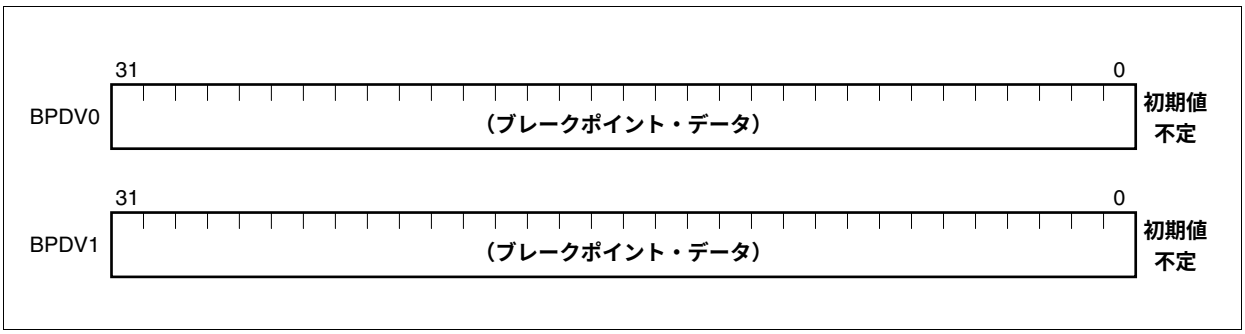
★　このレジスタへのリード／ライトは，ディバグ・モード時（DIR.DM ビット = 1）だけ可能です。ユーザ・モード時（DM ビット = 0）に読み出した場合のリード値は不定となります。

なお，使用しない場合は，必ず各ビットをセット（1）してください。

★　**注意　ブレークポイント・データ設定レジスタ 0, 1（BPDV0, BPDV1）は，タイプ A, B の製品のみ使用できます。その他のタイプの製品では使用できません。**

備考　16 ビット命令の命令コードを設定する場合は，LSB 詰めで設定してください。32 ビット命令の命令コードを設定する場合は，リトル・エンディアン形式で設定してください。

図2－15　ブレークポイント・データ設定レジスタ0, 1（BPDV0, BPDV1）



2. 2. 14　ブレークポイント・データ・マスク・レジスタ 0, 1（BPDM0, BPDM1）

データ比較のビット・マスクを設定します（1 でマスク）。

DIR.CS ビットの設定により，どちらか一方のレジスタが有効となります。

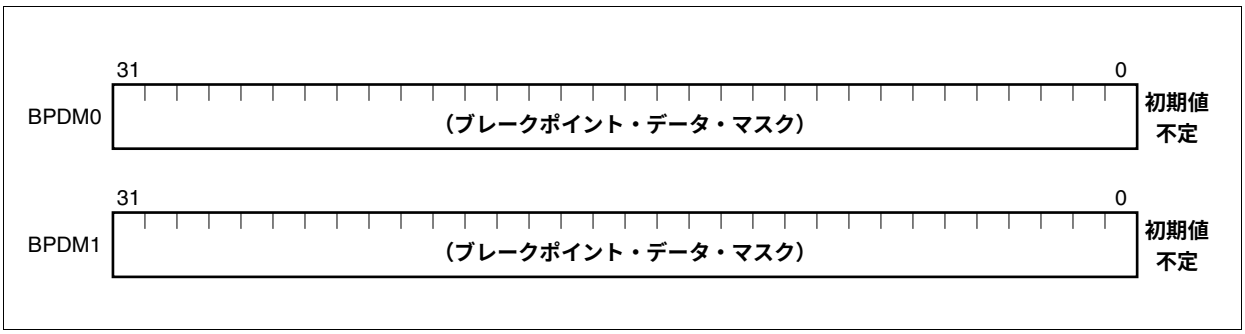
★　このレジスタへのリード／ライトは，ディバグ・モード時（DIR.DM ビット = 1）だけ可能です。ユーザ・モード時（DM ビット = 0）に読み出した場合のリード値は不定となります。

なお，使用しない場合は，必ず各ビットをセット（1）してください。

★　また，ブレークを検出するデータ・アクセスの種類をバイト・アクセスに設定している場合（BPCn.TY ビット = 0, 1）はビット 31-8 を，ハーフワード・アクセスに設定している場合（TY ビット = 1, 0）はビット 31-16 をセット（1）してください（n = 0, 1）。

★　**注意　ブレークポイント・データ・マスク・レジスタ 0, 1（BPDM0, BPDM1）は，タイプ A, B の製品のみ使用できます。その他のタイプの製品では使用できません。。**

図2－16　ブレークポイント・データ・マスク・レジスタ0, 1（BPDM0, BPDM1）



第3章　データ・タイプ

3.1　データ形式

サポートされているデータ・タイプは次のとおりです（**3.2　データ表現**参照）。

- **整数**（32, 16, 8 **ビット**）
- **符号なし整数**（32, 16, 8 **ビット**）
- **ビット**

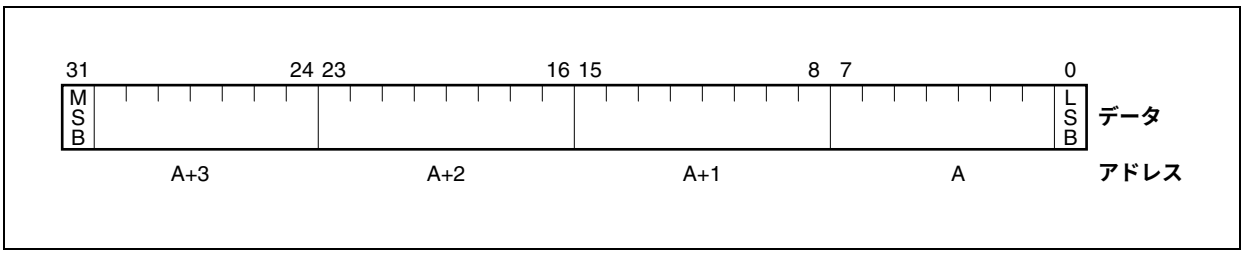
また，データ長として，ワード（32 ビット），ハーフワード（16 ビット），バイト（8 ビット）があります。これらのデータは，バイト 0 が常に最下位（最右端）バイトである構成になっています（リトル・エンディアン）。

固定長のデータがメモリにある場合のデータ形式を次に示します。

(1) ワード

ワードは任意のワード境界[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（Least significant bit）はビット 0，MSB（Most significant bit）はビット 31 に対応します。ワードはそのアドレス「A」（ミス・アライン・アクセス禁止の状態では下位 2 ビットは 0 [注]）で指定され，4 つのバイト「A」，「A+1」，「A+2」，「A+3」を占めます。

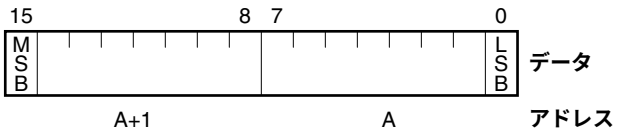
注　ミス・アライン・アクセス許可の状態では，ハーフワード・アクセス，ワード・アクセスにかかわらず，すべてのバイト境界にアクセスできます。**3.3　データ・アラインメント**を参照してください。



(2) ハーフワード

ハーフワードは任意のハーフワード境界[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」（下位1ビットは0[注]）で指定され，2つのバイト「A」，「A+1」を占めます。

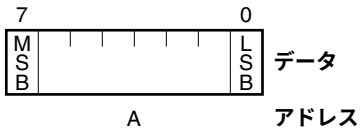
注　ミス・アライン・アクセス許可の状態では，ハーフワード・アクセス，ワード・アクセスにかかわらず，すべてのバイト境界にアクセスできます。**3.3　データ・アラインメント**を参照してください。



(3) バイト

バイトは，任意のバイト境界[注]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はビット0，MSB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。

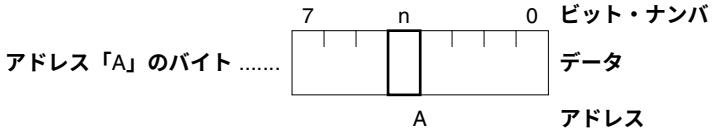
注　ミス・アライン・アクセス許可の状態では，ハーフワード・アクセス，ワード・アクセスにかかわらず，すべてのバイト境界にアクセスできます。**3.3　データ・アラインメント**を参照してください。



(4) ビット

ビットは，任意のバイト境界[注]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。

注　ミス・アライン・アクセス許可の状態では，ハーフワード・アクセス，ワード・アクセスにかかわらず，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. 2　データ表現

3. 2. 1　整　　数

整数は 2 の補数による 2 進数表現で表し，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したが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 進数表現で表し，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。

- SET1
- CLR1
- NOT1
- TST1

★ 3. 3　データ・アラインメント

ミス・アライン・アクセスの許可／禁止の設定に応じて，データのアライン（境界整列）を行う必要があります。

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。

備考　V850E1 CPUでは，IFIMAEN端子への入力レベルにより，ミス・アライン・アクセスの許可／禁止の設定を行います。

(1) ミス・アライン・アクセス許可の設定がされている場合

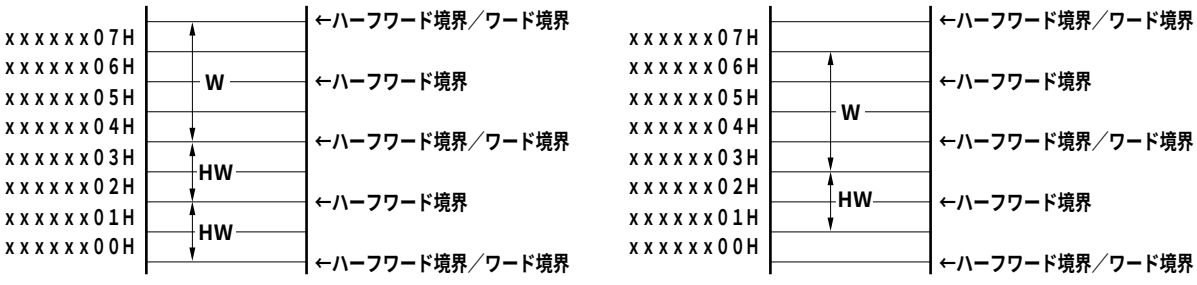
データ形式（バイト／ハーフワード／ワード）にかかわらず，すべてのアドレスにデータの配置が可能です。

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回発生し，バス効率が低下します。

(2) ミス・アライン・アクセス禁止の設定がされている場合

アドレスの下位ビット（ハーフワード・データの場合は，最下位ビット，ワード・データの場合は，下位2ビット）が0にマスクしてアクセスされるため，正しくアラインされていないとデータの消失や切り捨てが発生します。したがって，処理を行うデータがハーフワード形式の場合はハーフワード境界から，ワード形式の場合はワード境界から配置してください。

図3－1　ミスアライン・アクセス禁止時のデータ配置例



備考　W　：ワード・データ
HW：ハーフワード・データ

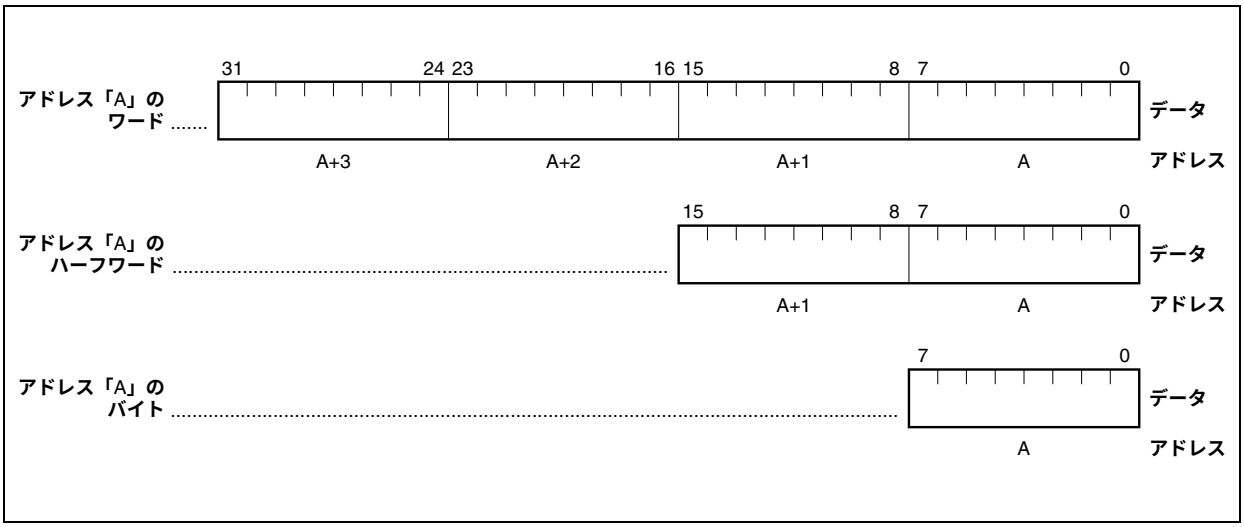
第4章　アドレス空間

V850E1 CPU は，4G 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）。

2 バイト構成のデータ形式をハーフワード，4 バイト構成のデータ形式をワードと呼びます。

このユーザーズ・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。



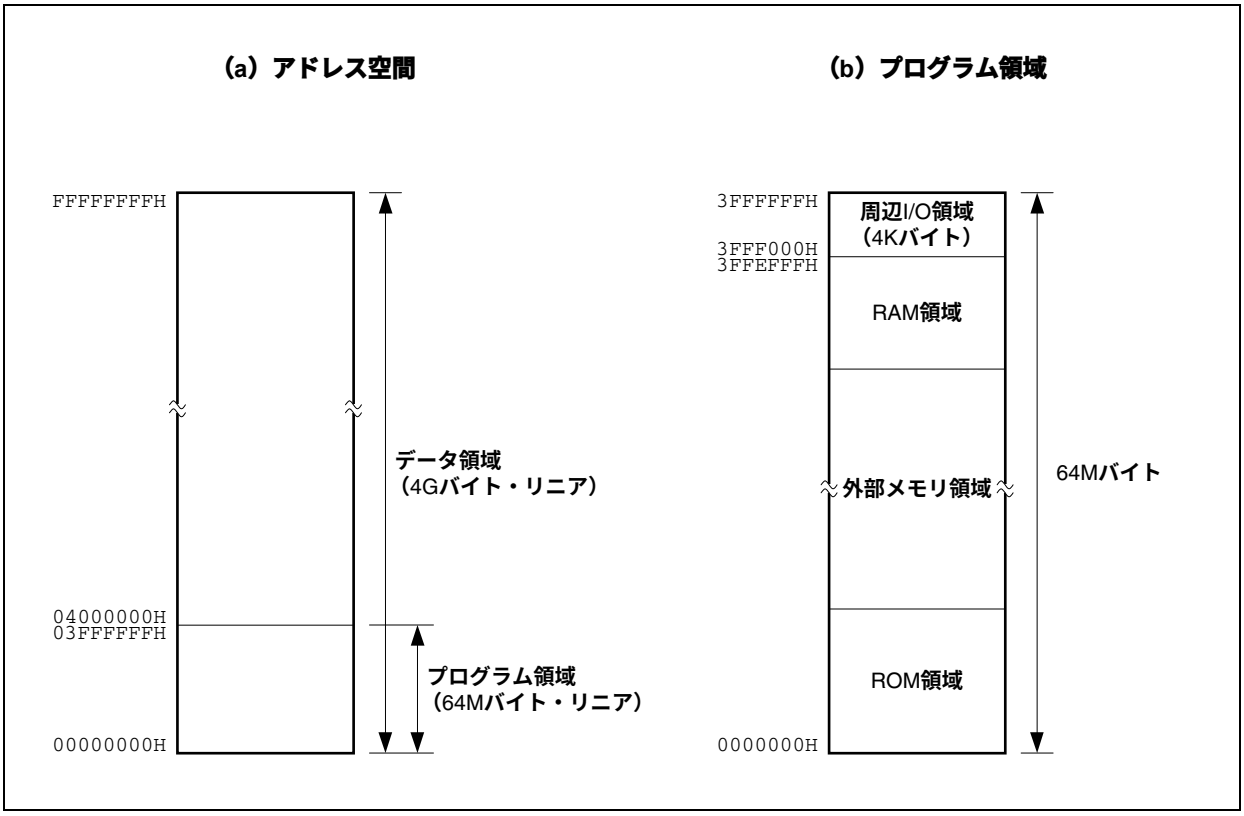
4.1　メモリ・マップ

V850E1 CPU は，32 ビット・アーキテクチャであり，オペランド・アドレシング（データ・アクセス）では，最大 4G バイトのリニア・アドレス空間（データ領域）をサポートします。

一方，命令アドレスのアドレシングにおいては，最大 64M バイトのリニア・アドレス空間（プログラム領域）をサポートします。

メモリ・マップを図 4－1 に示します。

図4－1　メモリ・マップ



4. 2　アドレシング・モード

アドレス生成には，分岐を伴う命令が使用する命令アドレス，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（+2）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 9 または 22 ビット・データ（ディスプレースメント：disp ×）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，それぞれビット 8 とビット 21 が符号ビット（S）となります。

JARL disp22, reg2 命令，JR disp22 命令，Bcond disp9 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（1/2）

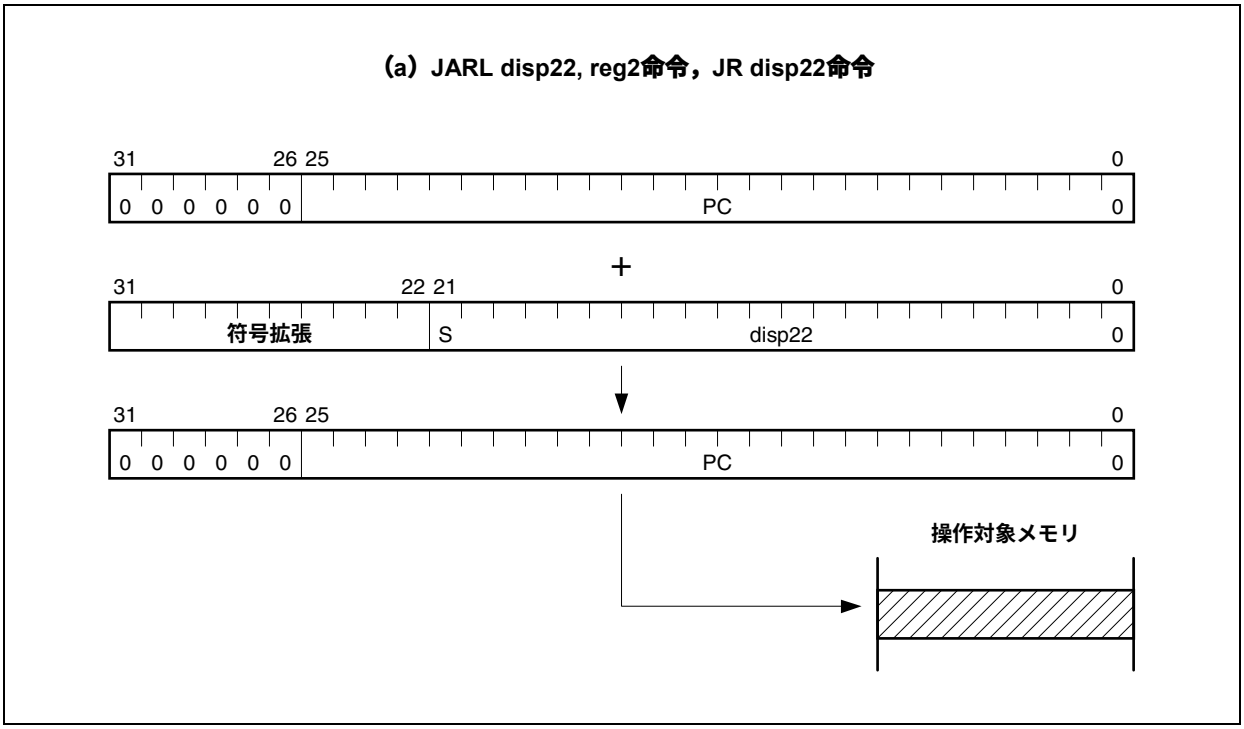
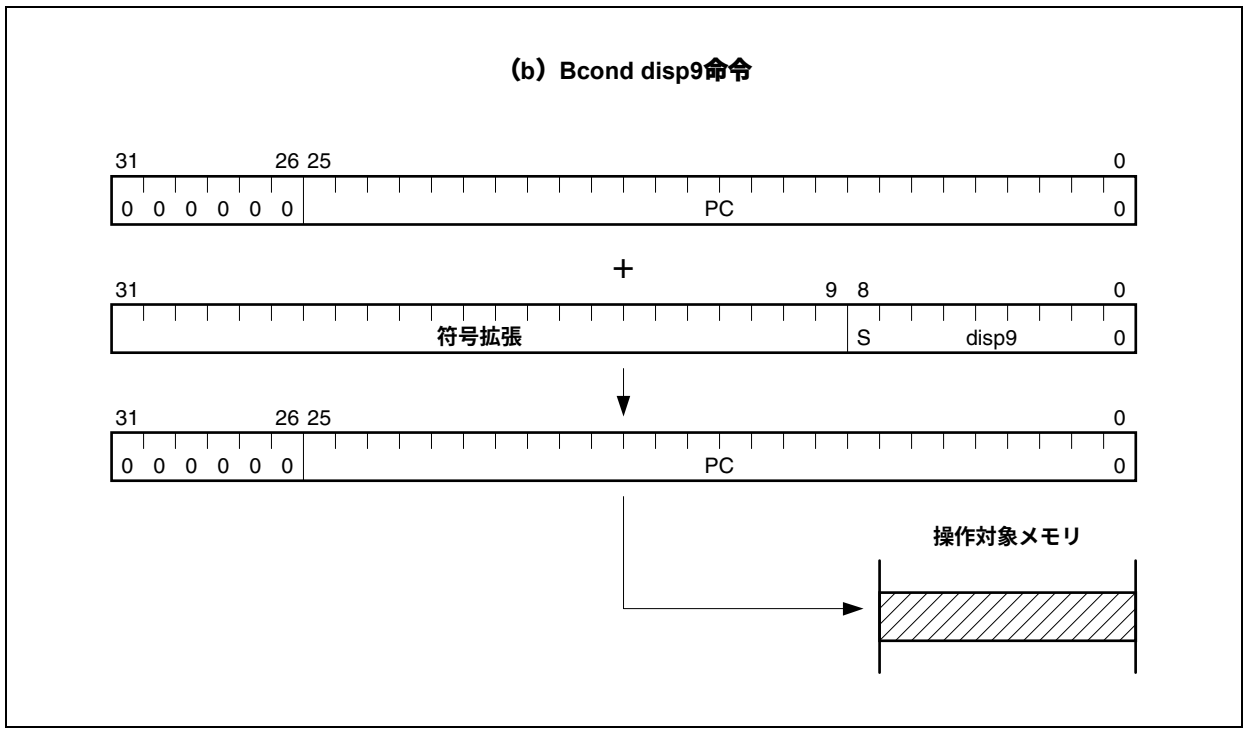


図4－2　レラティブ・アドレシング（2/2）

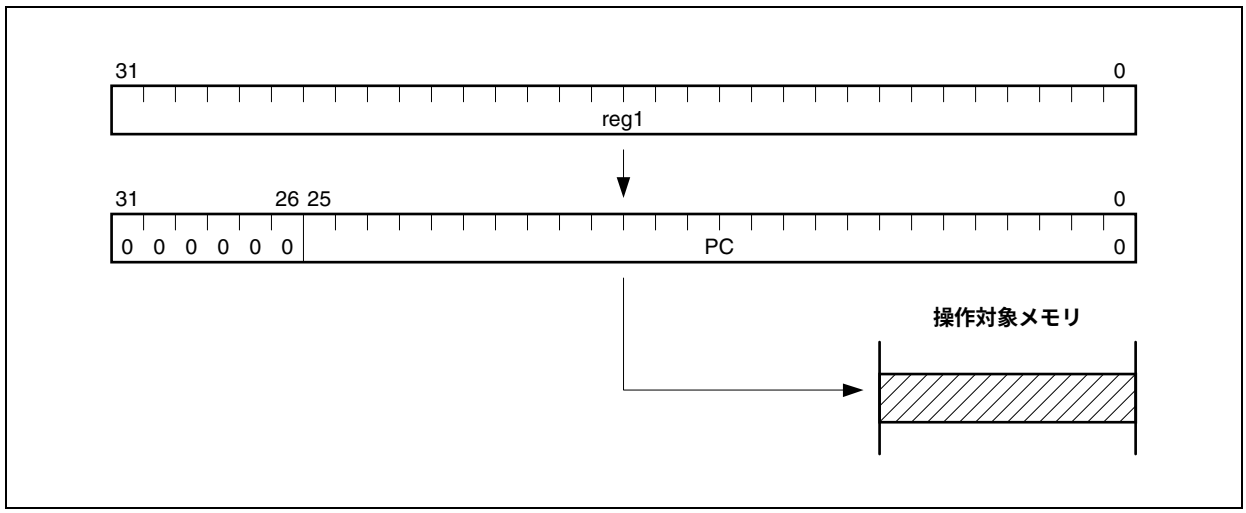


(2) レジスタ・アドレシング（レジスタ間接）

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

JMP [reg1] 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（JMP [reg1] 命令）



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 5 ビット・データ，16 ビット・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：トラップ・ベクタ（00H-1FH）を指定する 5 ビット・イミーディエトであり，TRAP 命令で使用されるオペランドです。

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

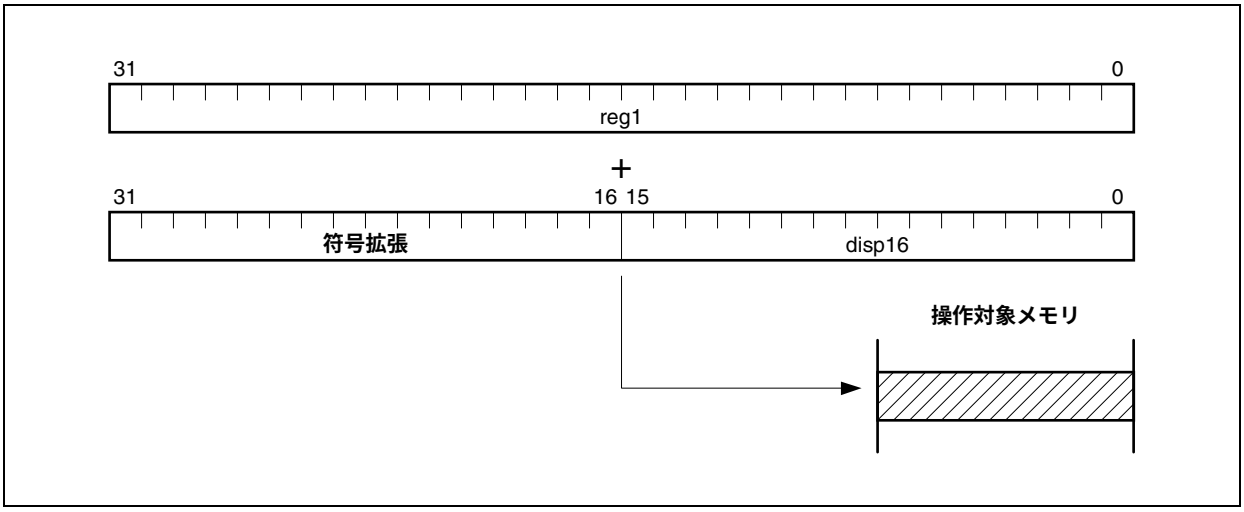
ベースト・アドレシングには，次に示す 2 種類があります。

(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 16 ビット・ディスプレースメント（disp16）の和がオペランド・アドレスとなって，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。

オペランドに，disp16 [reg1] を含む命令が，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（タイプ1）

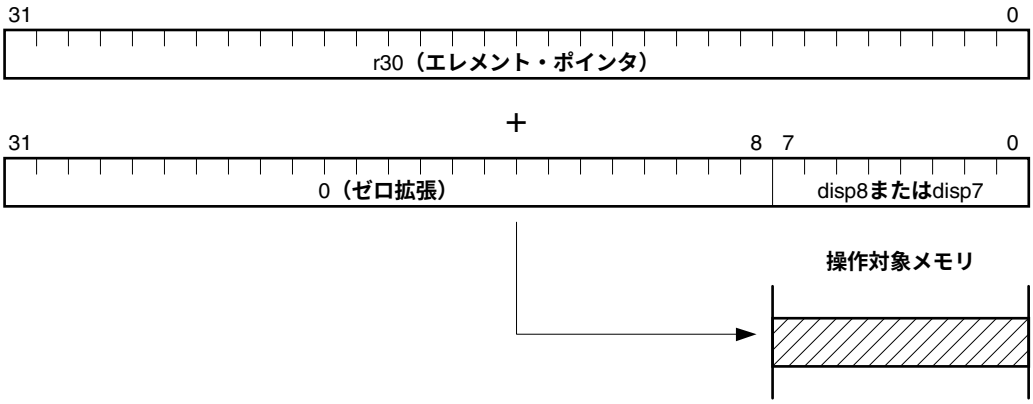


(b) タイプ2

エレメント・ポインタ（r30）**の内容と，**7 **または** 8 **ビット・ディスプレースメント・データ**（disp7, disp8）**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。**

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4－5　ベースト・アドレシング（タイプ2）



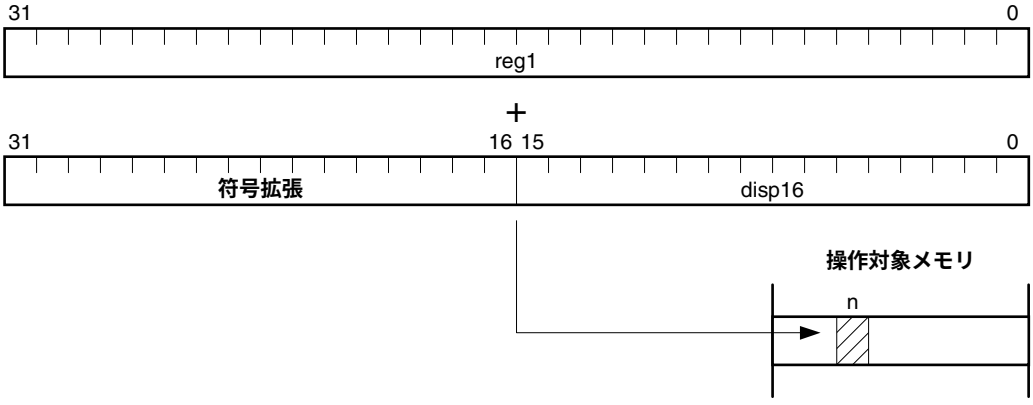
備考　**バイト・アクセス：**disp7
ハーフワード・アクセス，ワード・アクセス：disp8

(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** 16 **ビット・ディスプレースメント**（disp16）**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。**

ビット操作命令が，このアドレシングの対象となります。

図4－6　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）

第5章　命　　令

5.1　命令フォーマット

命令には，16 ビット・フォーマット，32 ビット・フォーマットの 2 種類があります。16 ビット・フォーマット命令には 2 項演算，制御，条件分岐などがあり，32 ビット・フォーマット命令にはロード，ストア，16 ビット・イミーディエトを扱う命令，ジャンプなどがあります。

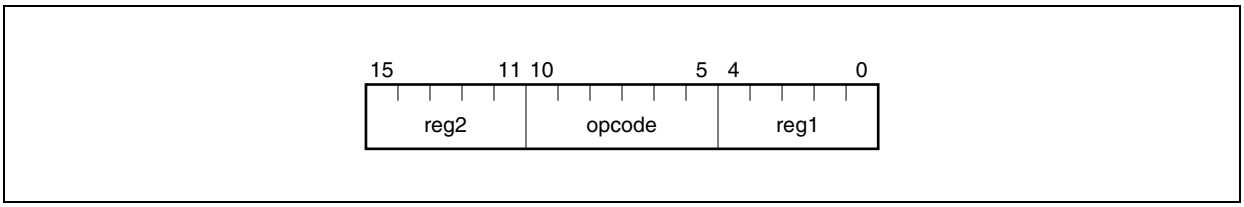
実際に命令がメモリに格納されるときは次のように配置されます。

- 各命令形式の下位部分（ビット 0 を含む）　→下位アドレス側
- 各命令形式の上位部分（ビット 15 またはビット 31 を含む）　→上位アドレス側

注意　一部の命令で未使用フィールド（RFU）がありますが，それらは将来の拡張用なので，0 に固定してください。

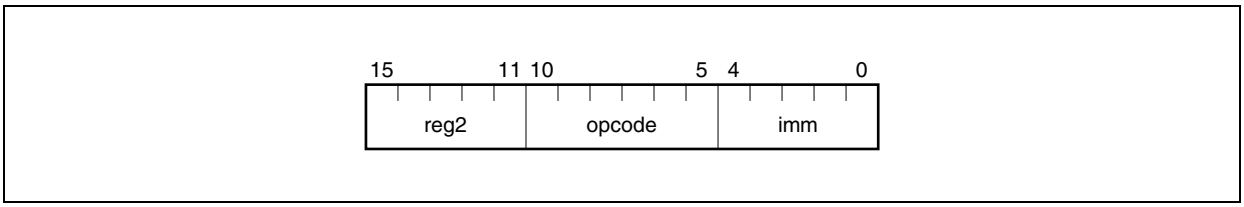
（1）reg-reg 命令形式（Format I）

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。



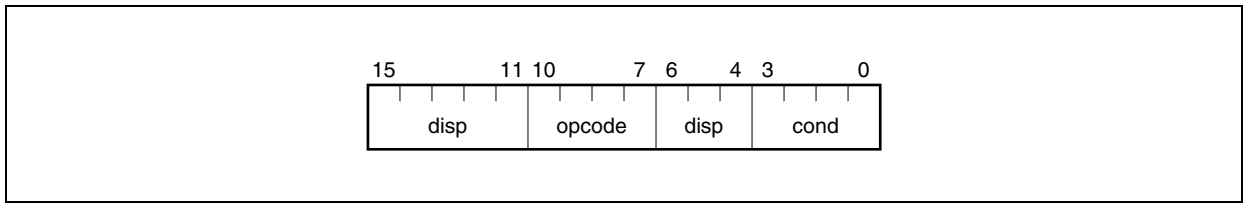
（2）imm-reg 命令形式（Format II）

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。



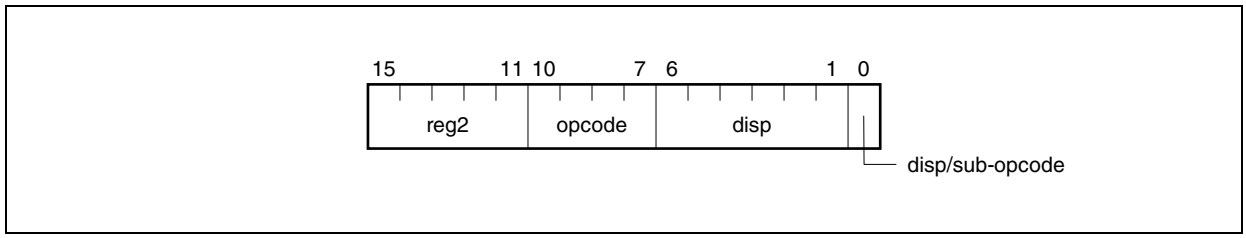
(3) 条件分岐命令形式（Format III）

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。

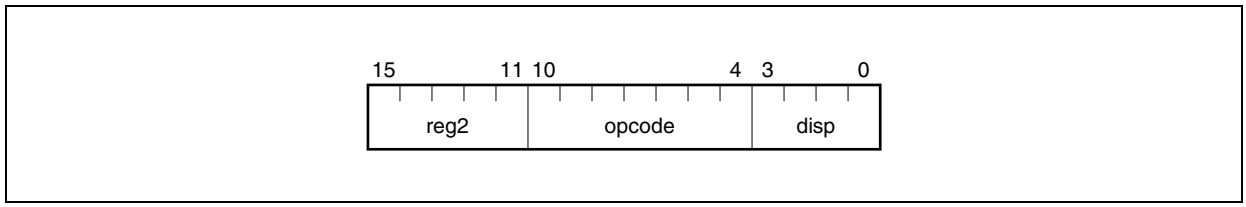


(4) ロード／ストア命令 16 ビット形式（Format IV）

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。

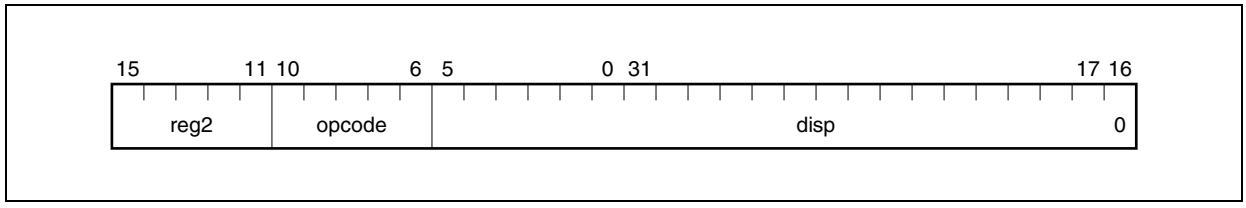


または，7 ビットのオペコード・フィールドと 1 つの汎用レジスタ指定フィールド，4 ビットのディスプレースメント・フィールドを持つ 16 ビット長命令形式。



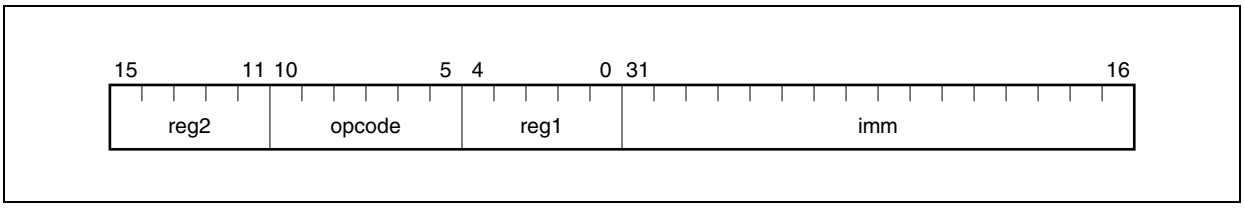
(5) ジャンプ命令形式（Format V）

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。



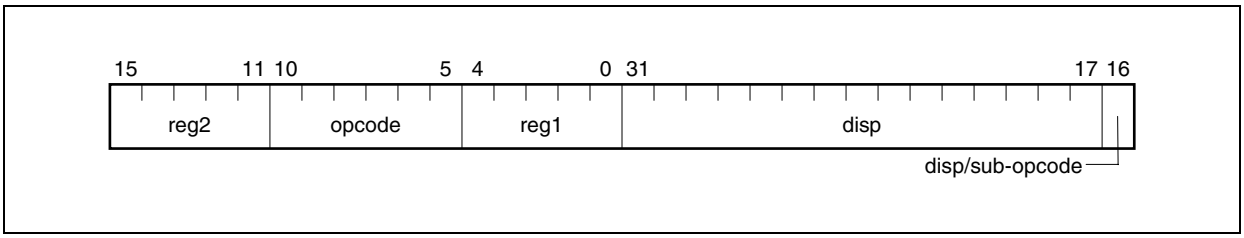
(6) 3 オペランド命令形式（Format VI）

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。



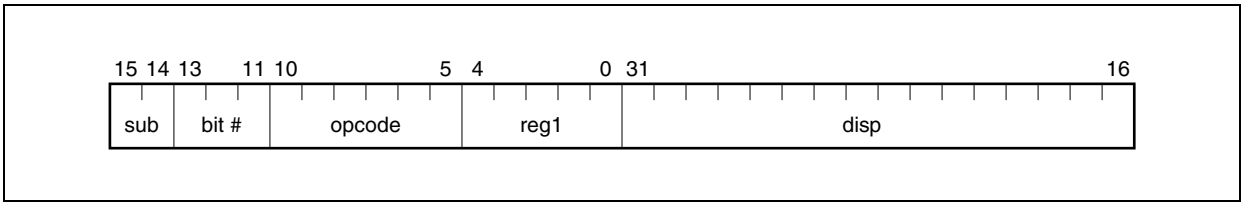
(7) ロード／ストア命令 32 ビット形式（Format VII）

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。



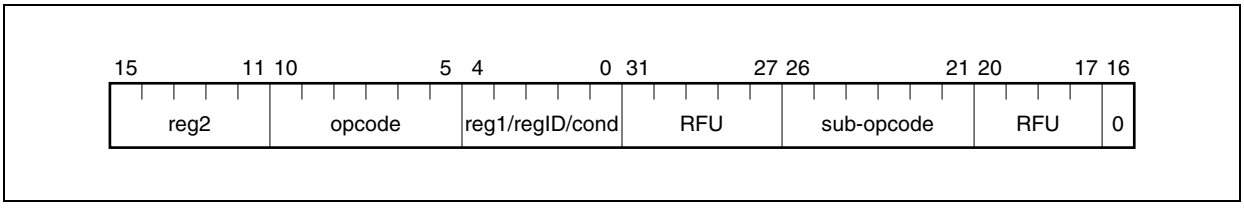
(8) ビット操作命令形式（Format VIII）

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。



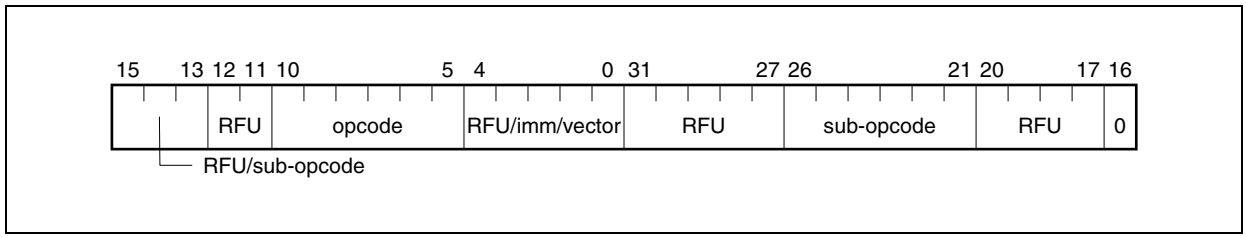
(9) 拡張命令形式 1（Format IX）

6 ビットのオペコード・フィールドと 6 ビットのサブオペコード・フィールド，2 つの汎用レジスタ指定フィールド（1 つはレジスタ番号フィールド（regID）または条件コード・フィールド（cond）の場合あり）を持つ 32 ビット長命令形式。



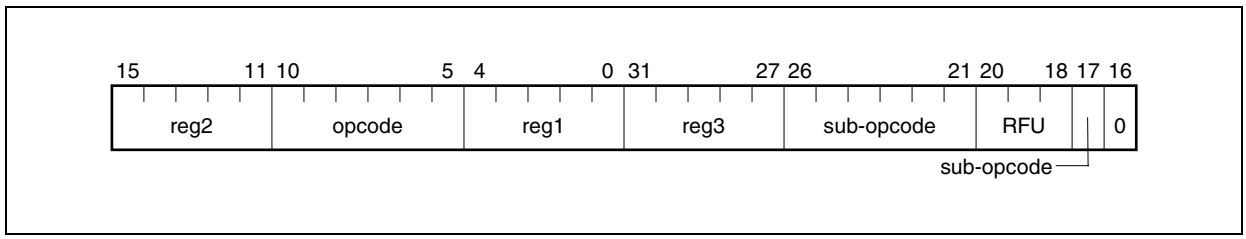
(10) 拡張命令形式 2 (Format X)

6 ビットのオペコード・フィールド，6 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。



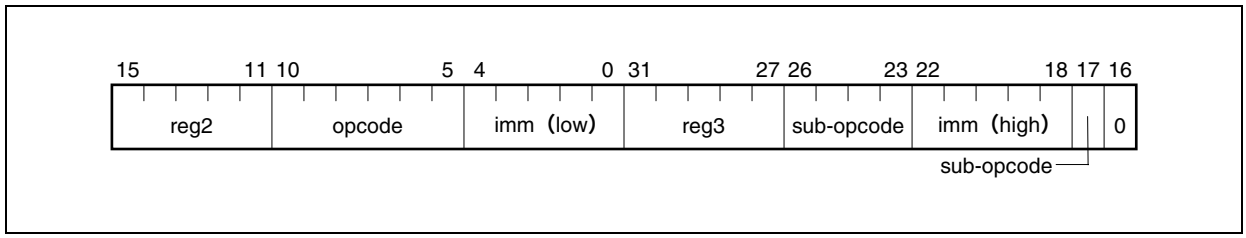
(11) 拡張命令形式 3 (Format XI)

6 ビットのオペコード・フィールドと 6 ビット＋1 ビットのサブオペコード・フィールド，3 つの汎用レジスタ指定フィールドを持つ 32 ビット長命令形式。



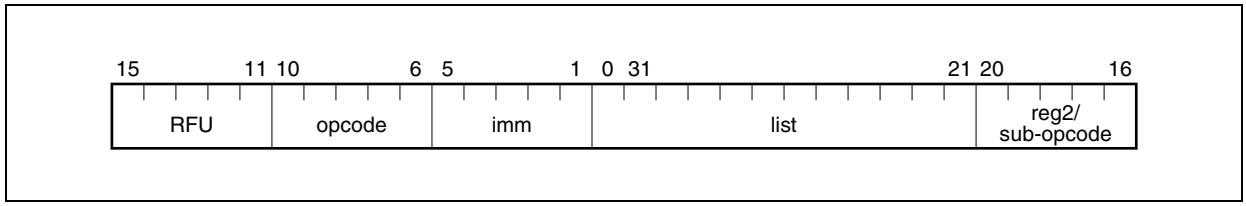
(12) 拡張命令形式 4 (Format XII)

6 ビットのオペコード・フィールド，4 ビット＋1 ビットのサブオペコード・フィールド，10 ビットのイミーディエト・フィールド，2 つの汎用レジスタ指定フィールドを持つ 32 ビット長命令形式。



(13) スタック操作命令形式 1 (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，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。



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，1-2 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) 算術演算命令

加減算，除算，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
- CMOV　：Conditional move
- CMP　：Compare
- DIV　：Divide word
- DIVH　：Divide half-word
- DIVHU　：Divide half-word unsigned
- DIVU　：Divide word unsigned
- MOV　：Move
- MOVEA　：Move effective address
- MOVHI　：Move high half-word
- SASF　：Shift and set flag condition
- SETF　：Set flag condition
- SUB　：Subtract
- SUBR　：Subtract reverse

(5) 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

(6) 論理演算命令

論理演算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。

- AND　：AND
- ANDI　：AND immediate
- BSH　：Byte swap half-word
- BSW　：Byte swap word
- HSW　：Half-word swap word
- NOT　：NOT
- OR　：OR
- ORI　：OR immediate

- SAR　：Shift arithmetic right
- SHL　：Shift logical left
- SHR　：Shift logical right
- SXB　：Sign extend byte
- SXH　：Sign extend half-word
- TST　：Test
- XOR　：Exclusive OR
- XORI　：Exclusive OR immediate
- ZXB　：Zero extend byte
- ZXH　：Zero extend half-word

(7) 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

(8) 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

(9) 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
- CTRET　：Return from CALLT
- DI　：Disable interrupt
- DISPOSE　：Function dispose
- EI　：Enable interrupt
- HALT　：Halt
- LDSR　：Load system register
- NOP　：No operation
- PREPARE　：Function prepare

- RETI　：Return from trap or interrupt
- STSR　：Store system register
- SWITCH　：Jump with table look up
- TRAP　：Trap

(10) ディバグ機能用命令

ディバグ機能用に予約された命令です。次の命令（ニモニック）があります。

- DBRET　：Return from debug trap
- DBTRAP　：Debug trap

★ **注意　ディバグ機能用命令は，タイプ C の製品ではサポートしていません。**

5. 3　命令セット

この節では，各命令のニモニックごとに，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　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	トラップ・ベクタ（00H-1FH）を指定する 5 ビット・データ
cccc	条件コードを示す 4 ビット・データ
sp	スタック・ポインタ（r3）
ep	エレメント・ポインタ（r30）
list 12	レジスタ・リスト

表5－2　オペレーションの凡例（1/2）

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。
load-memory-bit (a, b)	アドレス「a」のビット「b」を読み出す。
store-memory-bit (a, b, c)	アドレス「a」のビット「b」に「c」を書き込む。

表5－2 オペレーションの凡例（2/2）

略 号	意 味
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
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
cccc	条件コードを示す 4 ビット・データ
CCCC	Bcond 命令の条件コードを示す 4 ビット・データ
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

＜算術演算命令＞

Add register/immediate

ADD

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＞

ADDI	Add immediate 加算

［命令形式］　ADDI　imm16, reg1, reg2

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

［フォーマット］　Format VI

［オペコード］

15	0	31	16
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。

＜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＞

ANDI	AND immediate 論理積

［命令形式］　ANDI　imm16, reg1, reg2

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

［フォーマット］　Format VI

［オペコード］

15	0	31	16
rrrrr110110RRRRR		iiiiiiiiiiiiiiii	

［フ ラ グ］

CY　–

OV　0

S　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。

［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5－4　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
	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
	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。

＜論理演算命令＞

BSH

Byte swap half-word

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。

＜論理演算命令＞

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］　命令実行中に割り込みが発生すると，リード／ライト・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。割り込みから復帰したあとに再実行されます。

＜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)

Z フラグ← Not (Load-memory-bit (adr, bit#3))

Store-memory-bit (adr, bit#3, 0)

（2）adr ← GR [reg1]

Z フラグ ← Not (Load-memory-bit (adr, reg2))

Store-memory-bit (adr, reg2, 0)

［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。

（2）まず，汎用レジスタ reg1 の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。

［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。

＜算術演算命令＞

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］

(1)

15	0	31	16
rrrrr111111RRRRR		wwwww011001cccc0	

(2)

15	0	31	16
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。表 5−5　条件コード一覧で示されているコードのうちの 1 つを条件コード「cccc」として指定してください。

（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。表 5−5　条件コード一覧で示されているコードのうちの 1 つを条件コード「cccc」として指定してください。

［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＞

Return from CALLT

CTRET

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。

＜ディバグ機能用命令＞

Return from debug trap

DBRET

ディバグ・トラップからの復帰

［命令形式］　DBRET

［オペレーション］　PC ← DBPC
PSW ← DBPSW

［フォーマット］　Format X

［オペコード］

15	0	31	16
0000011111100000		0000000101000110	

［フ ラ グ］
CY　DBPSW から読み出した値が設定される
OV　DBPSW から読み出した値が設定される
S　DBPSW から読み出した値が設定される
Z　DBPSW から読み出した値が設定される
SAT　DBPSW 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，ディバグ・モードから復帰します。

［注　意］　(1) DBRET 命令はディバグを目的とした命令のため，基本的にディバグ・ツールが使用しています。このためディバグ・ツールが使用しているときに，アプリケーションが使用すると誤動作する場合があります。

★　(2) DBRET 命令は，タイプ C の製品ではサポートしていません。

＜ディバグ機能用命令＞

DBTRAP

Debug trap

ディバグ・トラップ

［命令形式］　DBTRAP

［オペレーション］　DBPC ← PC + 2 (復帰 PC)
DBPSW ← PSW
PSW.NP ← 1
PSW.EP ← 1
PSW.ID ← 1
PC ← 00000060H

［フォーマット］　Format I

［オペコード］

15	0
1111100001000000	

［フ ラ グ］
CY　–
OV　–
S　–
Z　–
SAT　–

［説　明］　復帰 PC（DBTRAP 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 DBPC と DBPSW に退避し，PSW の NP, EP, ID フラグをセット（1）します。
続いて，PC に例外トラップのハンドラ・アドレス（00000060H）をセットし，制御を移します。NP, EP, ID フラグ以外の PSW の各フラグは影響を受けません。
なお，DBPC に退避される値は，DBTRAP 命令の次の命令のアドレスです。

［注　意］　（1）DBTRAP 命令はディバグを目的とした命令のため，基本的にディバグ・ツールが使用しています。このためディバグ・ツールが使用しているときに，アプリケーションが使用すると誤動作する場合があります。

★　（2）DBTRAP 命令は，タイプ C の製品ではサポートしていません。

＜特殊命令＞

DI	Disable interrupt **マスカブル割り込みの禁止**

［命令形式］　DI

［オペレーション］　PSW.ID ← 1（**マスカブル割り込みの禁止**）

［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 フラグをセット（1）し，この命令実行中からマスカブル割り込みの受け付けを禁止します。

［補　足］　この命令の実行中は，割り込みのサンプリングをしません。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ですが，割り込みのサンプリングをこの命令実行中に行わないため，実際はこの命令実行中から割り込みを禁止します。ただし，ノンマスカブル割り込み（NMI）は，この命令の実行後も受け付けは禁止されません。

＜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）sp ← sp + zero-extend (imm5 logically shift left by 2)

GR [reg in list12] ← Load-memory (sp, Word)

sp ← sp + 4

repeat 2 steps above until all regs in list12 is loaded

（2）sp ← sp + zero-extend (imm5 logically shift left by 2)

GR [reg in list12] ← Load-memory (sp, Word)

sp ← sp + 4

repeat 2 states above until all regs in list12 is loaded

PC ← GR [reg1]

［フォーマット］　Format XIII

［オペコード］

(1)

15	0	31	16
0000011001iiiiiL		LLLLLLLLLLL00000	

(2)

15	0	31	16
0000011001iiiiiL		LLLLLLLLLLLRRRRR	

ただし，RRRRR **は，**00000 **以外です。**

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ものを，sp に加算します。そして，list12 に示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。なお，アドレスのビット 0 は，0 にマスクされます。
（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ものを，sp に加算します。そして，list12 に示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。なお，アドレスのビット 0 は，0 にマスクされます。

［補　足］　list12 の汎用レジスタは，下方にロードされます（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，ミス・アライン・アクセスがイネーブルであっても，0 にマスクされます。
また，sp の更新前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てから，割り込み処理完了後に再実行します（sp は割り込み実行開始前の元の値を保持します）。

［注　意］　命令実行中に割り込みが発生すると，スタック操作を行うため，リード／ライト・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。割り込みから復帰したあとに再実行されます。

＜算術演算命令＞

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。0 で割ったときは，オーバフローを生じ，商は不定となります。汎用レジスタ reg1 は影響を受けません。

［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（商は不定）に生じます。

この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てから，割り込み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。

また，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。

＜算術演算命令＞

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	

(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。**0 **で割ったときは，オーバフローを生じ，商は不定となります。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。**0 **で割ったときは，オーバフローを生じ，商は不定となります。汎用レジスタ** reg1 **は影響を受けません。**

［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（商が不定）に生じます。
この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てから，割り込み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。
なお，reg2 には r0 を指定しないでください。
また，除算の際，汎用レジスタ reg1 の上位 16 ビットを無視します。

(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（商が不定）に生じます。
この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てから，割り込み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。
また，除算の際，汎用レジスタ reg1 の上位 16 ビットを無視します。
なお，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。

＜算術演算命令＞

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。0 で割ったときは，オーバフローを生じ，商は不定となります。汎用レジスタ reg1 は影響を受けません。

［補　足］　オーバフローはゼロによる除算のとき（商は不定）に生じます。

この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てから，割り込み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。

なお，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。

＜算術演算命令＞

Divide word unsigned

DIVU

（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。0 で割ったときは，オーバフローを生じ，商は不定となります。汎用レジスタ reg1 は影響を受けません。

［補　足］　オーバフローはゼロによる除算のとき（商は不定）に生じます。

この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てから，割り込み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。

また，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。

＜特殊命令＞

EI	Enable interrupt **マスカブル割り込みの許可**

［命令形式］　EI

［オペレーション］　PSW.ID ← 0（**マスカブル割り込みの許可**）

［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 フラグをクリア（0）し，次の命令よりマスカブル割り込みの受け付けを許可します。

［補　足］　この命令の実行中は，割り込みのサンプリングをしません。

＜特殊命令＞

HALT	Halt 停止

［命令形式］　HALT

［オペレーション］　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 の動作クロックを停止させ，HALT モードに移行します。

［補　足］　HALT モードは次の 3 つの要因によって解除されます。

- リセット入力
- ノンマスカブル割り込み要求（NMI 入力）
- マスクされていないマスカブル割り込み要求（PSW の ID = 0 のとき）

なお，HALT モード中に割り込みを受け付けた場合，EIPC または FEPC には，この命令の次の命令アドレスが格納されます。

＜論理演算命令＞

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］　JARL　disp22, reg2

［オペレーション］　GR [reg2] ← PC + 4

PC ← PC + sign-extend (disp22)

［フォーマット］　Format V

［オペコード］

15	0	31	16
rrrrr11110dddddd		ddddddddddddddd0	

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

［フ ラ グ］　CY　–

OV　–

S　–

Z　–

SAT　–

［説　明］　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。

［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 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。

＜分岐命令＞

Jump register

JMP

無条件分岐（レジスタ間接）

［命令形式］　JMP　[reg1]

［オペレーション］　PC ← GR [reg1]

［フォーマット］　Format I

［オペコード］

15	0
00000000011RRRRR	

［フ ラ グ］　CY　–
OV　–
S　–
Z　–
SAT　–

［説　明］　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。

［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。なお，コールに相当する JARL 命令では，復帰 PC を汎用レジスタ reg2 に格納してください。

＜分岐命令＞

JR	Jump relative 無条件分岐（PC 相対）

［命令形式］　JR　disp22

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

［フォーマット］　Format V

［オペコード］

15	0 31	16
0000011110dddddd	ddddddddddddddd0	

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

［フ ラ グ］　CY　–
OV　–
S　–
Z　–
SAT　–

［説　明］　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。

［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＞

LD.B	Load byte ロード

［命令形式］　LD.B　disp16 [reg1] , reg2

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

［フォーマット］　Format VII

［オペコード］

15	0 31	16
rrrrr111000RRRRR	ddddddddddddddddd	

［フ ラ グ］　CY　–
OV　–
S　–
Z　–
SAT　–

［説　明］　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。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。

［補　足］　この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てから，割り込み処理完了後に再実行します。

★　［タイプ D, E, F の製品の場合］
アクセス対象の資源（内蔵 ROM，内蔵 RAM，内蔵周辺 I/O，外部メモリ）により，バス・サイクルが入れ替わる可能性があります（同じ資源に対するアクセスであれば，バス・サイクルが入れ替わることはありません）。

★　［タイプ A, B, C の製品の場合］
各バス（VFB, VDB, VSB, NPB, 命令キャッシュ用バス，データ・キャッシュ用バス）に接続される異なる資源に対するアクセスについては，バス・サイクルの順序が入れ替わる可能性があります（同一バスに対するアクセスでは，入れ替わることはありません）。

＜ロード命令＞

LD.BU

Load byte unsigned

ロード

［命令形式］　LD.BU　disp16 [reg1] , reg2

［オペレーション］　adr ← GR [reg1] + sign-extend (disp16)

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

［フォーマット］　Format VII

［オペコード］

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。

［フ ラ グ］

CY　–

OV　–

S　–

Z　–

SAT　–

［説　明］　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。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。

［補　足］　この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てから，割り込み処理完了後に再実行します。

★ ［タイプ D, E, F の製品の場合］

アクセス対象の資源（内蔵 ROM，内蔵 RAM，内蔵周辺 I/O，外部メモリ）により，バス・サイクルが入れ替わる可能性があります（同じ資源に対するアクセスであれば，バス・サイクルが入れ替わることはありません）。

★ ［タイプ A, B, C の製品の場合］

各バス（VFB, VDB, VSB, NPB, 命令キャッシュ用バス，データ・キャッシュ用バス）に接続される異なる資源に対するアクセスについては，バス・サイクルの順序が入れ替わる可能性があります（同一バスに対するアクセスでは，入れ替わることはありません）。

＜ロード命令＞

LD.H	Load half-word ロード

［命令形式］　LD.H　disp16 [reg1] , reg2

［オペレーション］　adr ← GR [reg1] + sign-extend (disp16)

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

［フォーマット］　Format VII

［オペコード］

15	0 31	16
rrrrr111001RRRRR	ddddddddddddddd0	

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

［フ ラ グ］　CY　–

OV　–

S　–

Z　–

SAT　–

［説　明］　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。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。

［注　意］　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の加算結果は，アクセスするデータ形態（ハーフワード，ワード）と，ミス・アライン・モード設定により次の 2 種類があります。

- 下位ビットを 0 にマスクしてアドレス生成（ミス・アライン・アクセス禁止の場合）
- 下位ビットをマスクしないでアドレス生成（ミス・アライン・アクセス許可の場合）

★　（タイプ D, E, F の製品は，ミス・アライン・アクセス許可の場合になります。）

ミス・アライン・アクセスについては，3. 3　データ・アラインメントを参照してください。

［補　足］　この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てから，割り込み処理完了後に再実行します。

★ ［タイプ D, E, F の製品の場合］

アクセス対象の資源（内蔵 ROM，内蔵 RAM，内蔵周辺 I/O，外部メモリ）により，バス・サイクルが入れ替わる可能性があります（同じ資源に対するアクセスであれば，バス・サイクルが入れ替わることはありません）。

★ ［タイプ A, B, C の製品の場合］

各バス（VFB, VDB, VSB, NPB, 命令キャッシュ用バス，データ・キャッシュ用バス）に接続される異なる資源に対するアクセスについては，バス・サイクルの順序が入れ替わる可能性があります（同一バスに対するアクセスでは，入れ替わることはありません）。

＜ロード命令＞

LD.HU	Load half-word unsigned ロード

［命令形式］　LD.HU　disp16 [reg1] , reg2

［オペレーション］　adr ← GR [reg1] + sign-extend (disp16)

GR [reg2] ← zero-extend (Load-memory (adr, Half-word))

［フォーマット］　Format VII

［オペコード］

15	0	31	16
rrrrr111111RRRRR		ddddddddddddddd1	

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

［フ ラ グ］　CY　–

OV　–

S　–

Z　–

SAT　–

［説　明］　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。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 reg2 に格納します。

［注　意］　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の加算結果は，アクセスするデータ形態（ハーフワード，ワード）と，ミス・アライン・モード設定により次の 2 種類があります。

- 下位ビットを 0 にマスクしてアドレス生成（ミス・アライン・アクセス禁止の場合）
- 下位ビットをマスクしないでアドレス生成（ミス・アライン・アクセス許可の場合）

★　（タイプ D, E, F の製品は，ミス・アライン・アクセス許可の場合になります。）

ミス・アライン・アクセスについては，3. 3　データ・アラインメントを参照してください。

［補　足］　この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てから，割り込み処理完了後に再実行します。

★ ［タイプ D, E, F の製品の場合］

アクセス対象の資源（内蔵 ROM，内蔵 RAM，内蔵周辺 I/O，外部メモリ）により，バス・サイクルが入れ替わる可能性があります（同じ資源に対するアクセスであれば，バス・サイクルが入れ替わることはありません）。

★ ［タイプ A, B, C の製品の場合］

各バス（VFB, VDB, VSB, NPB, 命令キャッシュ用バス，データ・キャッシュ用バス）に接続される異なる資源に対するアクセスについては，バス・サイクルの順序が入れ替わる可能性があります（同一バスに対するアクセスでは，入れ替わることはありません）。

＜ロード命令＞

LD.W	Load word ロード

［命令形式］　LD.W　disp16 [reg1] , reg2

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

［フォーマット］　Format VII

［オペコード］

15	0 31	16
rrrrr111001RRRRR	ddddddddddddddd1	

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

［フ ラ グ］　CY　–
OV　–
S　–
Z　–
SAT　–

［説　明］　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。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 に格納します。

［注　意］　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の加算結果は，アクセスするデータ形態（ハーフワード，ワード）と，ミス・アライン・モード設定により次の 2 種類があります。

- 下位ビットを 0 にマスクしてアドレス生成（ミス・アライン・アクセス禁止の場合）
- 下位ビットをマスクしないでアドレス生成（ミス・アライン・アクセス許可の場合）
  （タイプ D, E, F の製品は，ミス・アライン・アクセス許可の場合になります。）

★

ミス・アライン・アクセスについては，3. 3　データ・アラインメントを参照してください。

［補　足］　この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てから，割り込み処理完了後に再実行します。

★ ［タイプ D, E, F の製品の場合］

アクセス対象の資源（内蔵 ROM，内蔵 RAM，内蔵周辺 I/O，外部メモリ）により，バス・サイクルが入れ替わる可能性があります（同じ資源に対するアクセスであれば，バス・サイクルが入れ替わることはありません）。

★ ［タイプ A, B, C の製品の場合］

各バス（VFB, VDB, VSB, NPB, 命令キャッシュ用バス，データ・キャッシュ用バス）に接続される異なる資源に対するアクセスについては，バス・サイクルの順序が入れ替わる可能性があります（同一バスに対するアクセスでは，入れ替わることはありません）。

＜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　–（補足参照）

OV　–（補足参照）

S　–（補足参照）

Z　–（補足参照）

SAT　–（補足参照）

［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。

［補　足］　システム・レジスタ番号（regID）が 5（PSW）の場合は，PSW の各フラグには汎用レジスタ reg2 の対応するビットの値が設定されます。PSW への書き込み時のみ，割り込みのサンプリングをしません。この命令によって PSW の ID フラグをセット（1）する場合，ID フラグが有効になるのは次の命令からですが,割り込みのサンプリングを PSW への書き込み時には行わないため，実際はこの命令実行中から割り込みを禁止します。

［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。

＜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	

(2)

15	0
rrrrr010000iiiii	

(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。
なお，reg2 には r0 を指定しないでください。

（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。

＜算術演算命令＞

MOVEA

Move effective address

実効アドレスの転送

［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	

［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。
なお，reg2 には r0 を指定しないでください。

［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。

＜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	

［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。
なお，reg2 には r0 を指定しないでください。

［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。

＜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	

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。

★ **［注　意］**　**「MUL reg1, reg2, reg3」命令において，次の条件をすべて満たすレジスタの組み合わせは行わないでください。この条件に当てはまる命令を実行した場合の動作は保証しません。**

- reg1 = reg3
- reg1 ≠ reg2
- reg1 ≠ r0
- reg3 ≠ r0

＜乗算命令＞

MULH

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

（符号付き）ハーフワード・データの乗算

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

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

［フォーマット］　（1）Format I
（2）Format II

［オペコード］

(1)

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

(2)

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

［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 にワード・データとして格納します。汎用レジスタ reg1 は影響を受けません。
なお，reg2 には r0 を指定しないでください。
（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。
なお，reg2 には r0 を指定しないでください。

［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。

＜乗算命令＞

MULHI

Multiply half-word by immediate (16-bit)

（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	

［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。
なお，reg2 には r0 を指定しないでください。

［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。

＜乗算命令＞

MULU

Multiply word by register/immediate (9-bit)

（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。**

★ ［注　意］ **「MULU reg1, reg2, reg3」命令において，次の条件をすべて満たすレジスタの組み合わせは行わないでください。この条件に当てはまる命令を実行した場合の動作は保証しません。**

- reg1 = reg3
- reg1 ≠ reg2
- reg1 ≠ r0
- reg3 ≠ r0

＜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］　何もせず最低 1 クロック費やします。

［フォーマット］　Format I

［オペコード］

15															0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

［フ ラ グ］　CY　–
OV　–
S　–
Z　–
SAT　–

［説　明］　何もせず最低 1 クロック費やします。

［補　足］　PC は ＋2 されます。また，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)

Z フラグ ← Not (Load-memory-bit (adr, bit#3))

Store-memory-bit (adr, bit#3, Z フラグ)

（2）adr ← GR [reg1]

Z フラグ ← Not (Load-memory-bit (adr, reg2))

Store-memory-bit (adr, reg2, Z フラグ)

［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。

（2）まず，汎用レジスタ reg1 の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。

［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。

＜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）Store-memory (sp – 4, GR [reg in list12], Word)　sp ← sp – 4

repeat 1 step above until all regs in list12 is stored

sp ← sp – zero-extend (imm5)

（2）Store-memory (sp – 4, GR [reg in list12], Word)　sp ← sp – 4

repeat 1 step above until all regs in list12 is stored

sp ← sp – zero-extend (imm5)

ep ← sp/imm

［フォーマット］　Format XIII

［オペコード］

(1)

15 – 0	31 – 16
0000011110iiiiiL	LLLLLLLLLL00001

(2)

15 – 0	31 – 16	オプション（47-32 または，63-32）
0000011110iiiiiL	LLLLLLLLLLff011	imm16 / imm32

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

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

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

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

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

［フ ラ グ］

CY　–
OV　–
S　–
Z　–
SAT　–

［説　明］

（1）list12 に表示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 に表示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，ミス・アライン・アクセスが可能でも，0 にマスクされます。
また，sp の更新前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てから，割り込み処理完了後に再実行します（sp と ep は割り込み実行開始前の元の値を保持します）。

［注　意］

命令実行中に割り込みが発生すると，スタック操作を行うため，リード／ライト・サイクルとレジスタ値の書き換えが終了したあとに中止する場合があります。

＜特殊命令＞

Return from trap or interrupt

RETI

ソフトウエア例外または割り込みルーチンからの復帰

［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，ソフトウエア例外または割り込みルーチン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) PSW の EP フラグが 1 の場合，PSW の NP フラグの状態にかかわらず，EIPC, EIPSW から復帰 PC，PSW を取り出します。
PSW の EP フラグが 0 かつ PSW の NP フラグが 1 の場合，FEPC，FEPSW から復帰 PC, PSW を取り出します。
PSW の EP フラグが 0 かつ PSW の NP フラグが 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。

［注　意］　ノンマスカブル割り込み処理またはソフトウエア例外処理からの 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 PSW の NP フラグ，EP フラグを次の状態にしておく必要があります。

- RETI 命令によるノンマスカブル割り込み処理からの復帰時：
  NP = 1 かつ EP = 0
- RETI 命令によるソフトウエア例外処理からの復帰時：
  EP = 1

プログラムによる設定には LDSR 命令を使用します。
割り込みコントローラの動作絡みで，この命令の後半の ID ステージでは割り込みを受け付けません。

＜論理演算命令＞

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

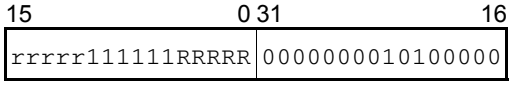
［命令形式］　（1）SAR　reg1, reg2
（2）SAR　imm5, reg2

［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

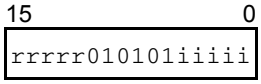
［フォーマット］　（1）Format IX
（2）Format II

［オペコード］

(1)



(2)



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

［説　明］　（1）汎用レジスタ reg2 のワード・データを汎用レジスタ reg1 の下位 5 ビットで示されるシフト数分，0 から+31 までを算術右シフトし（シフト以前の MSB の値をシフトを実行したあとの MSB にコピーする），汎用レジスタ reg2 に書き込みます。シフト数が 0 のときは，汎用レジスタ reg2 は命令実行前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。

＜算術演算命令＞

Shift and set flag condition

SASF

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 に 1 がセットされます。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 に 0 がセットされます。
表 5－5　条件コード一覧で示されているコードのうちの 1 つを条件コード「cccc」として指定してください。

［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

［オペレーション］　（1）GR [reg2] ← saturated (GR [reg2] + GR [reg1])

（2）GR [reg2] ← saturated (GR [reg2] + sigh-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	0	1	1	0	R	R	R	R	R

(2)

15															0
r	r	r	r	r	0	1	0	0	0	1	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　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。

なお，reg2 には r0 を指定しないでください。

（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。

なお，reg2 には r0 を指定しないでください。

［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］　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。

＜飽和演算命令＞

Saturated subtract

SATSUB

飽和減算

［命令形式］　SATSUB　reg1, reg2

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

［フォーマット］　Format I

［オペコード］

15	0
rrrrr000101RRRRR	

［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 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。
なお，reg2 には r0 を指定しないでください。

［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］　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。

＜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	

［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。

なお，reg2 には r0 を指定しないでください。

［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］　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。

＜飽和演算命令＞

Saturated subtract reverse

SATSUBR

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	

［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。
なお，reg2 には r0 を指定しないでください。

［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］　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。

＜ビット操作命令＞

Set bit

SET1

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)

Z フラグ ← Not (Load-memory-bit (adr, bit#3))

Store-memory-bit (adr, bit#3, 1)

（2）adr ← GR [reg1]

Z フラグ ← Not (Load-memory-bit (adr, reg2))

Store-memory-bit (adr, reg2, 1)

［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。

（2）まず，汎用レジスタ reg1 の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。

［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。

＜算術演算命令＞

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。条件コード「cccc」には，**表 5－5　条件コード一覧**に示す条件コードを指定します。

［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 R_{an} 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。

表5－5　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

＜論理演算命令＞

Shift logical left by register/immediate (5-bit)

SHL

論理左シフト

［命令形式］　（1）SHL　reg1, reg2

（2）SHL　imm5, reg2

［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)

［フォーマット］　（1）Format IX

（2）Format II

［オペコード］

(1)

15	0	31	16
rrrrr111111RRRRR		0000000011000000	

(2)

15	0
rrrrr010110iiiii	

［フ ラ グ］　CY　最後にシフト・アウトしたビットが 1 のとき 1，そうでないとき 0，ただしシフト数が 0 のときは 0

OV　0

S　演算結果が負のとき 1，そうでないとき 0

Z　演算結果が 0 のとき 1，そうでないとき 0

SAT　－

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

（2）汎用レジスタ reg2 のワード・データを，ワード長までゼロ拡張した 5 ビット・イミ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。

＜論理演算命令＞

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

［オペレーション］　（1）GR [reg2] ← GR [reg2] logically shift right by GR [reg1]

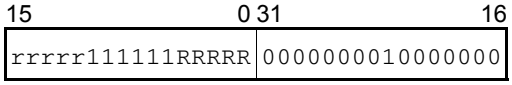
（2）GR [reg2] ← GR [reg2] logically shift right by zero-extend（imm5）

［フォーマット］　（1）Format IX

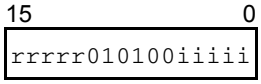
（2）Format II

［オペコード］

(1)



(2)



［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。

＜ロード命令＞

SLD.B	Short format load byte 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。

［補　足］　この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てから，割り込み処理完了後に再実行します。

★　［タイプ D, E, F の製品の場合］
アクセス対象の資源（内蔵 ROM，内蔵 RAM，内蔵周辺 I/O，外部メモリ）により，バス・サイクルが入れ替わる可能性があります（同じ資源に対するアクセスであれば，バス・サイクルが入れ替わることはありません）。

★　［タイプ A, B, C の製品の場合］
各バス（VFB, VDB, VSB, NPB, 命令キャッシュ用バス，データ・キャッシュ用バス）に接続される異なる資源に対するアクセスについては，バス・サイクルの順序が入れ替わる可能性があります（同一バスに対するアクセスでは，入れ替わることはありません）。

［注　意］　(1) 命令実行中に割り込みが発生すると，リード／ライト・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。この場合，割り込みから復帰したあとに再度この命令を実行します。したがって，リード・サイクルによって状態が変わる I/O や FIFO タイプの資源などに対するアクセスには，割り込みが発生しないことが明確である場合を除き，LD 命令を使用してください（LD 命令やストア命令は，命令実行中に割り込みが発生してもバス・サイクルは再実行されません）。

★　(2) sld 命令と割り込み競合に関する制限事項については，**付録** A　**注意事項**を参照してください。

＜ロード命令＞

SLD.BU	Short format load byte unsigned 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

ただし，rrrrr は 00000 以外です。

［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 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。

［補　足］　この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てから，割り込み処理完了後に再実行します。

★　［タイプ D, E, F の製品の場合］
アクセス対象の資源（内蔵 ROM，内蔵 RAM，内蔵周辺 I/O，外部メモリ）により，バス・サイクルが入れ替わる可能性があります（同じ資源に対するアクセスであれば，バス・サイクルが入れ替わることはありません）。

★　［タイプ A, B, C の製品の場合］
各バス（VFB, VDB, VSB, NPB, 命令キャッシュ用バス，データ・キャッシュ用バス）に接続される異なる資源に対するアクセスについては，バス・サイクルの順序が入れ替わる可能性があります（同一バスに対するアクセスでは，入れ替わることはありません）。

［注　意］　(1) 命令実行中に割り込みが発生すると，リード／ライト・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。この場合，割り込みから復帰したあとに再度この命令を実行します。したがって，リード・サイクルによって状態が変わる I/O や FIFO タイプの資源などに対するアクセスには，割り込みが発生しないことが明確である場合を除き，LD 命令を使用してください（LD 命令やストア命令は，命令実行中に割り込みが発生してもバス・サイクルは再実行されません）。

★　(2) sld 命令と割り込み競合に関する制限事項については，**付録** A　**注意事項**を参照してください。

＜ロード命令＞

SLD.H	Short format load half-word 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。

［補　足］　この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てから，割り込み処理完了後に再実行します。

★ ［タイプ D, E, F の製品の場合］
アクセス対象の資源（内蔵 ROM，内蔵 RAM，内蔵周辺 I/O，外部メモリ）により，バス・サイクルが入れ替わる可能性があります（同じ資源に対するアクセスであれば，バス・サイクルが入れ替わることはありません）。

★ ［タイプ A, B, C の製品の場合］
各バス（VFB, VDB, VSB, NPB, 命令キャッシュ用バス，データ・キャッシュ用バス）に接続される異なる資源に対するアクセスについては，バス・サイクルの順序が入れ替わる可能性があります（同一バスに対するアクセスでは，入れ替わることはありません）。

［注　意］　(1) 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の加算結果は，アクセスするデータ形態（ハーフワード，ワード）と，ミス・アライン・モード設定により次の 2 種類があります。

- 下位ビットを 0 にマスクしてアドレス生成（ミス・アライン・アクセス禁止の場合）
- 下位ビットをマスクしないでアドレス生成（ミス・アライン・アクセス許可の場合）

★　（タイプ D, E, F の製品は，ミス・アライン・アクセス許可の場合になります。）

ミス・アライン・アクセスについては，3.3　**データ・アラインメント**を参照してください。また，命令実行中に割り込みが発生すると，リード／ライト・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。この場合，割り込みから復帰したあとに再度この命令を実行します。したがって，リード・サイクルによって状態が変わる I/O や FIFO タイプの資源などに対するアクセスには，割り込みが発生しないことが明確である場合を除き，LD 命令を使用してください（LD 命令やストア命令は，命令実行中に割り込みが発生してもバス・サイクルは再実行されません）。

★　(2) sld 命令と割り込み競合に関する制限事項については，**付録** A　**注意事項**を参照してください。

＜ロード命令＞

Short format load half-word unsigned

SLD.HU

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	

ただし，dddd は disp5 の上位 4 ビット，rrrrr は 00000 以外です。

［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 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。

［補　足］　この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てから，割り込み処理完了後に再実行します。

★ ［タイプ D, E, F の製品の場合］
アクセス対象の資源（内蔵 ROM，内蔵 RAM，内蔵周辺 I/O，外部メモリ）により，バス・サイクルが入れ替わる可能性があります（同じ資源に対するアクセスであれば，バス・サイクルが入れ替わることはありません）。

★ ［タイプ A, B, C の製品の場合］
各バス（VFB, VDB, VSB, NPB, 命令キャッシュ用バス，データ・キャッシュ用バス）に接続される異なる資源に対するアクセスについては，バス・サイクルの順序が入れ替わる可能性があります（同一バスに対するアクセスでは，入れ替わることはありません）。

［注　意］　(1) 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の加算結果は，アクセスするデータ形態（ハーフワード，ワード）と，ミス・アライン・
★ モード設定により次の 2 種類があります。

- 下位ビットを 0 にマスクしてアドレス生成（ミス・アライン・アクセス禁止の場合）
- 下位ビットをマスクしないでアドレス生成（ミス・アライン・アクセス許可の場合）
  （タイプ D, E, F の製品は，ミス・アライン・アクセス許可の場合になります。）

ミス・アライン・アクセスについては，3.3　**データ・アラインメント**を参照してください。また，命令実行中に割り込みが発生すると，リード／ライト・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。この場合，割り込みから復帰したあとに再度この命令を実行します。したがって，リード・サイクルによって状態が変わる I/O や FIFO タイプの資源などに対するアクセスには，割り込みが発生しないことが明確である場合を除き，LD 命令を使用してください（LD 命令やストア命令は，命令実行中に割り込みが発生してもバス・サイクルは再実行されません）。

★ (2) sld 命令と割り込み競合に関する制限事項については，**付録** A　**注意事項**を参照してください。

＜ロード命令＞

SLD.W	Short format load word 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
r	r	r	r	r	1	0	1	0	d	d	d	d	d	d	0

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。生成したアドレスからワード・データを読み出し，reg2 に格納します。

［補　足］　この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てから，割り込み処理完了後に再実行します。

★　［タイプ D, E, F の製品の場合］
アクセス対象の資源（内蔵 ROM，内蔵 RAM，内蔵周辺 I/O，外部メモリ）により，バス・サイクルが入れ替わる可能性があります（同じ資源に対するアクセスであれば，バス・サイクルが入れ替わることはありません）。

★　［タイプ A, B, C の製品の場合］
各バス（VFB, VDB, VSB, NPB, 命令キャッシュ用バス，データ・キャッシュ用バス）に接続される異なる資源に対するアクセスについては，バス・サイクルの順序が入れ替わる可能性があります（同一バスに対するアクセスでは，入れ替わることはありません）。

［注　意］　(1) 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の加算結果は，アクセスするデータ形態（ハーフワード，ワード）と，ミス・アライン・モード設定により次の 2 種類があります。

- 下位ビットを 0 にマスクしてアドレス生成（ミス・アライン・アクセス禁止の場合）
- 下位ビットをマスクしないでアドレス生成（ミス・アライン・アクセス許可の場合）

★　（タイプ D, E, F の製品は，ミス・アライン・アクセス許可の場合になります。）

ミス・アライン・アクセスについては，3.3　**データ・アラインメント**を参照してください。また，命令実行中に割り込みが発生すると，リード／ライト・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。この場合，割り込みから復帰したあとに再度この命令を実行します。したがって，リード・サイクルによって状態が変わる I/O や FIFO タイプの資源などに対するアクセスには，割り込みが発生しないことが明確である場合を除き，LD 命令を使用してください（LD 命令やストア命令は，命令実行中に割り込みが発生してもバス・サイクルは再実行されません）。

★　(2) sld 命令と割り込み競合に関する制限事項については，**付録** A　**注意事項**を参照してください。

＜ストア命令＞

SST.B	Short format store byte 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。

［補　足］　この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てから，割り込み処理完了後に再実行します。

★ ［タイプ D, E, F の製品の場合］
アクセス対象の資源（内蔵 ROM，内蔵 RAM，内蔵周辺 I/O，外部メモリ）により，バス・サイクルが入れ替わる可能性があります（同じ資源に対するアクセスであれば，バス・サイクルが入れ替わることはありません）。

★ ［タイプ A, B, C の製品の場合］
各バス（VFB, VDB, VSB, NPB, 命令キャッシュ用バス，データ・キャッシュ用バス）に接続される異なる資源に対するアクセスについては，バス・サイクルの順序が入れ替わる可能性があります（同一バスに対するアクセスでは，入れ替わることはありません）。

＜ストア命令＞

Short format store half-word

SST.H

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・データを生成したアドレスに格納します。

［注　意］　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の加算結果は，アクセスするデータ形態（ハーフワード，ワード）と，ミス・アライン・モード設定により次の 2 種類があります。

- 下位ビットを 0 にマスクしてアドレス生成（ミス・アライン・アクセス禁止の場合）
- 下位ビットをマスクしないでアドレス生成（ミス・アライン・アクセス許可の場合）

★　（タイプ D, E, F の製品は，ミス・アライン・アクセス許可の場合になります。）

ミス・アライン・アクセスについては，3. 3　データ・アラインメントを参照してください。

［補　足］　この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てから，割り込み処理完了後に再実行します。

★ ［タイプ D, E, F の製品の場合］

アクセス対象の資源（内蔵 ROM，内蔵 RAM，内蔵周辺 I/O，外部メモリ）により，バス・サイクルが入れ替わる可能性があります（同じ資源に対するアクセスであれば，バス・サイクルが入れ替わることはありません）。

★ ［タイプ A, B, C の製品の場合］

各バス（VFB, VDB, VSB, NPB, 命令キャッシュ用バス，データ・キャッシュ用バス）に接続される異なる資源に対するアクセスについては，バス・サイクルの順序が入れ替わる可能性があります（同一バスに対するアクセスでは，入れ替わることはありません）。

＜ストア命令＞

Short format store word

SST.W

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・データを生成したアドレスに格納します。

［注　意］　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の加算結果は，アクセスするデータ形態（ハーフワード，ワード）と，ミス・アライン・モード設定により次の 2 種類があります。

- 下位ビットを 0 にマスクしてアドレス生成（ミス・アライン・アクセス禁止の場合）
- 下位ビットをマスクしないでアドレス生成（ミス・アライン・アクセス許可の場合）

★　（タイプ D, E, F の製品は，ミス・アライン・アクセス許可の場合になります。）

ミス・アライン・アクセスについては，3. 3　データ・アラインメントを参照してください。

［補　足］　この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てから，割り込み処理完了後に再実行します。

★ ［タイプ D, E, F の製品の場合］

アクセス対象の資源（内蔵 ROM，内蔵 RAM，内蔵周辺 I/O，外部メモリ）により，バス・サイクルが入れ替わる可能性があります（同じ資源に対するアクセスであれば，バス・サイクルが入れ替わることはありません）。

★ ［タイプ A, B, C の製品の場合］

各バス（VFB, VDB, VSB, NPB, 命令キャッシュ用バス，データ・キャッシュ用バス）に接続される異なる資源に対するアクセスについては，バス・サイクルの順序が入れ替わる可能性があります（同一バスに対するアクセスでは，入れ替わることはありません）。

＜ストア命令＞

ST.B	Store byte ストア

［命令形式］　ST.B　reg2, disp16 [reg1]

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

［フォーマット］　Format VII

［オペコード］

15	0	31	16
rrrrr111010RRRRR		ddddddddddddddddd	

［フ ラ グ］
CY　－
OV　－
S　－
Z　－
SAT　－

［説　明］　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。

［補　足］　この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てから，割り込み処理完了後に再実行します。

★　［タイプ D, E, F の製品の場合］
アクセス対象の資源（内蔵 ROM，内蔵 RAM，内蔵周辺 I/O，外部メモリ）により，バス・サイクルが入れ替わる可能性があります（同じ資源に対するアクセスであれば，バス・サイクルが入れ替わることはありません）。

★　［タイプ A, B, C の製品の場合］
各バス（VFB, VDB, VSB, NPB, 命令キャッシュ用バス，データ・キャッシュ用バス）に接続される異なる資源に対するアクセスについては，バス・サイクルの順序が入れ替わる可能性があります（同一バスに対するアクセスでは，入れ替わることはありません）。

＜ストア命令＞

Store half-word

ST.H

ストア

［命令形式］　ST.H　reg2, disp16 [reg1]

［オペレーション］　adr ← GR [reg1] + sign-extend (disp16)

Store-memory (adr, GR [reg2], Half-word)

［フォーマット］　Format VII

［オペコード］

15	0	31	16
rrrrr111011RRRRR		ddddddddddddddd0	

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

［フ ラ グ］　CY　－

OV　－

S　－

Z　－

SAT　－

［説　明］　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。

［注　意］　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の加算結果は，アクセスするデータ形態（ハーフワード，ワード）と，ミス・アライン・モード設定により次の 2 種類があります。

- 下位ビットを 0 にマスクしてアドレス生成（ミス・アライン・アクセス禁止の場合）
- 下位ビットをマスクしないでアドレス生成（ミス・アライン・アクセス許可の場合）

★　（タイプ D, E, F の製品は，ミス・アライン・アクセス許可の場合になります。）

ミス・アライン・アクセスについては，3. 3　データ・アラインメントを参照してください。

［補　足］　この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てから，割り込み処理完了後に再実行します。

★ ［タイプ D, E, F の製品の場合］

アクセス対象の資源（内蔵 ROM，内蔵 RAM，内蔵周辺 I/O，外部メモリ）により，バス・サイクルが入れ替わる可能性があります（同じ資源に対するアクセスであれば，バス・サイクルが入れ替わることはありません）。

★ ［タイプ A, B, C の製品の場合］

各バス（VFB, VDB, VSB, NPB, 命令キャッシュ用バス，データ・キャッシュ用バス）に接続される異なる資源に対するアクセスについては，バス・サイクルの順序が入れ替わる可能性があります（同一バスに対するアクセスでは，入れ替わることはありません）。

＜ストア命令＞

ST.W	Store word ストア

［命令形式］　ST.W　reg2, disp16 [reg1]

［オペレーション］　adr ← GR [reg1] + sign-extend (disp16)

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

［フォーマット］　Format VII

［オペコード］

15	0	31	16
rrrrr111011RRRRR		ddddddddddddddd1	

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

［フ ラ グ］　CY　－

OV　－

S　－

Z　－

SAT　－

［説　明］　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。

［注　意］　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の加算結果は，アクセスするデータ形態（ハーフワード，ワード）と，ミス・アライン・モード設定により次の 2 種類があります。

- 下位ビットを 0 にマスクしてアドレス生成（ミス・アライン・アクセス禁止の場合）
- 下位ビットをマスクしないでアドレス生成（ミス・アライン・アクセス許可の場合）

★　（タイプ D, E, F の製品は，ミス・アライン・アクセス許可の場合になります。）

ミス・アライン・アクセスについては，3. 3　データ・アラインメントを参照してください。

［補　足］　この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てから，割り込み処理完了後に再実行します。

★ ［タイプ D, E, F の製品の場合］

アクセス対象の資源（内蔵 ROM，内蔵 RAM，内蔵周辺 I/O，外部メモリ）により，バス・サイクルが入れ替わる可能性があります（同じ資源に対するアクセスであれば，バス・サイクルが入れ替わることはありません）。

★ ［タイプ A, B, C の製品の場合］

各バス（VFB, VDB, VSB, NPB, 命令キャッシュ用バス，データ・キャッシュ用バス）に接続される異なる資源に対するアクセスについては，バス・サイクルの順序が入れ替わる可能性があります（同一バスに対するアクセスでは，入れ替わることはありません）。

＜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	

［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。
(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。
(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。
(4) (3) で生成されたターゲット・アドレスへ分岐。

＜論理演算命令＞

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。

＜論理演算命令＞

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＞

Trap

TRAP

ソフトウエア例外

［命令形式］　TRAP　vector

★ ［オペレーション］　EIPC ← PC + 4 (復帰 PC)

EIPSW ← PSW

ECR.EICC ← 例外コード（40H-4FH, 50H-5FH）

PSW.EP ← 1

PSW.ID ← 1

PC ← 00000040H（vector が 00H-0FH（例外コード：40H-4FH）のとき）

00000050H（vector が 10H-1FH（例外コード：50H-5FH）のとき）

［フォーマット］　Format X

［オペコード］

15	0	31	16
00000111111iiiii		0000000100000000	

［フ ラ グ］　CY　–

OV　–

S　–

Z　–

SAT　–

［説　明］　復帰 PC，PSW を EIPC，EIPSW に退避し，例外コードの設定（ECR の EICC），PSW のフラグの設定（EP, ID フラグをセット（1））を行ったあと，「vector」で指定されるトラップ・ベクタ（00H-1FH）に対応するハンドラ・アドレスにジャンプし，例外処理を開始します。

EP, ID フラグ以外の PSW の各フラグは影響を受けません。

なお，復帰 PC とは，TRAP 命令の次の命令のアドレスです。

＜論理演算命令＞

TST	Test テスト

［命令形式］　TST　reg1, reg2

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

［フォーマット］　Format I

［オペコード］

15	0
rrrrr001011RRRRR	

［フ ラ グ］　CY　–

OV　0

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

Z　演算結果が 0 のとき 1，そうでないとき 0

SAT　–

［説　明］　汎用レジスタ reg2 のワード・データと汎用レジスタ reg1 のワード・データの論理積をとります。結果は格納されず，フラグだけが影響を受けます。汎用レジスタ reg1, reg2 は影響を受けません。

＜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)

Z フラグ ← Not (Load-memory-bit (adr, bit#3))

（2）adr ← GR [reg1]

Z フラグ ← Not (Load-memory-bit (adr, 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 PSW の 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。

（2）まず，汎用レジスタ reg1 の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 PSW の 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。

＜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。

＜論理演算命令＞

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。**

＜論理演算命令＞

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。

5. 4　命令実行クロック数

次に内蔵 ROM，内蔵 RAM を使用した場合における命令実行クロック数一覧を示します。なお，命令実行クロック数は，命令の組み合わせにより異なる場合があります。詳細については，**第 8 章 パイプライン**を参照してください。

また，表 5－6 に命令実行クロック数を示します。

表5－6　命令実行クロック数一覧（1/3）

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		
				i	r	l
ロード命令	LD.B	disp16 [reg1] , reg2	4	1	1	注 1
	LD.H	disp16 [reg1] , reg2	4	1	1	注 1
	LD.W	disp16 [reg1] , reg2	4	1	1	注 1
	LD.BU	disp16 [reg1] , reg2	4	1	1	注 1
	LD.HU	disp16 [reg1] , reg2	4	1	1	注 1
	SLD.B	disp7 [ep] , reg2	2	1	1	注 2
	SLD.BU	disp4 [ep] , reg2	2	1	1	注 2
	SLD.H	disp8 [ep] , reg2	2	1	1	注 2
	SLD.HU	disp5 [ep] , reg2	2	1	1	注 2
	SLD.W	disp8 [ep] , reg2	2	1	1	注 2
ストア命令	ST.B	reg2, disp16 [reg1]	4	1	1	1
	ST.H	reg2, disp16 [reg1]	4	1	1	1
	ST.W	reg2, disp16 [reg1]	4	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	2 注3	2
	MUL	imm9, reg2, reg3	4	1	2 注3	2
	MULH	reg1, reg2	2	1	1	2
	MULH	imm5, reg2	2	1	1	2
	MULHI	imm16, reg1, reg2	4	1	1	2
	MULU	reg1, reg2, reg3	4	1	2 注3	2
	MULU	imm9, reg2, reg3	4	1	2 注3	2
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
	CMOV	cccc, reg1, reg2, reg3	4	1	1	1
	CMOV	cccc, imm5, reg2, reg3	4	1	1	1
	CMP	reg1, reg2	2	1	1	1
	CMP	imm5, reg2	2	1	1	1
	DIV	reg1, reg2, reg3	4	35	35	35

表5－6　命令実行クロック数一覧（2/3）

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		
				i	r	l
算術演算命令	DIVH	reg1, reg2	2	35	35	35
	DIVH	reg1, reg2, reg3	4	35	35	35
	DIVHU	reg1, reg2, reg3	4	34	34	34
	DIVU	reg1, reg2, reg3	4	34	34	34
	MOV	reg1, reg2	2	1	1	1
	MOV	imm5, reg2	2	1	1	1
	MOV	imm32, reg1	6	2	2	2
	MOVEA	imm16, reg1, reg2	4	1	1	1
	MOVHI	imm16, reg1, reg2	4	1	1	1
	SASF	cccc, reg2	4	1	1	1
	SETF	cccc, reg2	4	1	1	1
	SUB	reg1, reg2	2	1	1	1
	SUBR	reg1, reg2	2	1	1	1
飽和演算命令	SATADD	reg1, reg2	2	1	1	1
	SATADD	imm5, reg2	2	1	1	1
	SATSUB	reg1, reg2	2	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
	BSH	reg2, reg3	4	1	1	1
	BSW	reg2, reg3	4	1	1	1
	HSW	reg2, reg3	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
	SAR	reg1, reg2	4	1	1	1
	SAR	imm5, reg2	2	1	1	1
	SHL	reg1, reg2	4	1	1	1
	SHL	imm5, reg2	2	1	1	1
	SHR	reg1, reg2	4	1	1	1
	SHR	imm5, reg2	2	1	1	1
	SXB	reg1	2	1	1	1
	SXH	reg1	2	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
	ZXB	reg1	2	1	1	1
	ZXH	reg1	2	1	1	1

表5－6　命令実行クロック数一覧（3/3）

	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		
					i	r	l
	分岐命令	Bcond	disp9（条件成立時）	2	2 注4	2 注4	2 注4
			disp9（条件不成立時）	2	1	1	1
★		JARL	disp22, reg2	4	2 注5	2 注5	2 注5
★		JMP	[reg1]	2	3 注5	3 注5	3 注5
★		JR	disp22	4	2 注5	2 注5	2 注5
	ビット操作命令	CLR1	bit#3, disp16 [reg1]	4	3 注6	3 注6	3 注6
		CLR1	reg2, [reg1]	4	3 注6	3 注6	3 注6
		NOT1	bit#3, disp16 [reg1]	4	3 注6	3 注6	3 注6
		NOT1	reg2, [reg1]	4	3 注6	3 注6	3 注6
		SET1	bit#3, disp16 [reg1]	4	3 注6	3 注6	3 注6
		SET1	reg2, [reg1]	4	3 注6	3 注6	3 注6
		TST1	bit#3, disp16 [reg1]	4	3 注6	3 注6	3 注6
		TST1	reg2, [reg1]	4	3 注6	3 注6	3 注6
★	特殊命令	CALLT	imm6	2	4 注5	4 注5	4 注5
★		CTRET	－	4	3 注5	3 注5	3 注5
		DI	－	4	1	1	1
		DISPOSE	imm5, list12	4	n+1 注7	n+1 注7	n+1 注7
		DISPOSE	imm5, list12, [reg1]	4	n+3 注7	n+3 注7	n+3 注7
		EI	－	4	1	1	1
		HALT	－	4	1	1	1
		LDSR	reg2, regID	4	1	1	1
		NOP	－	2	1	1	1
		PREPARE	list12, imm5	4	n+1 注7	n+1 注7	n+1 注7
		PREPARE	list12, imm5, sp	4	n+2 注7	n+2 注7	n+2 注7
		PREPARE	list12, imm5, imm16	6	n+2 注7	n+2 注7	n+2 注7
		PREPARE	list12, imm5, imm32	8	n+3 注7	n+3 注7	n+3 注7
★		RETI	－	4	3 注5	3 注5	3 注5
		STSR	regID, reg2	4	1	1	1
		SWITCH	reg1	2	5	5	5
★		TRAP	vector	4	3 注5	3 注5	3 注5
★	ディバグ機能用	DBRET	－	4	3 注5	3 注5	3 注5
★	命令注8	DBTRAP	－	2	3 注5	3 注5	3 注5
	未定義命令コード			4	3	3	3

注 1. 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は 2**）。**

2. 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は 1**）。**

3. reg2 = reg3**（結果の下位** 32 **ビットがレジスタに書き込まれない），または** reg3 = r0**（結果の上位** 32 **ビットはレジスタに書き込まれない）の場合は** 1 **クロック短縮される。**

★ **4. ［タイプ** D, E, F **の製品の場合］**

直前に PSW **の内容を書き換える命令がある場合は** 4**。**

［タイプ A, B, C **の製品の場合］**

直前に PSW **の内容を書き換える命令がある場合は** 3**。**

★ **5. タイプ** D **の製品の場合は＋**1 **クロック。**

タイプ E **の製品の場合は＋**2 **クロック。**

6. ウエイト・ステートがない場合（3**＋リード・アクセス・ウエイト・ステート数）。**

7. n **は，**list12 **のロード・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** list12 **のレジスタ合計数。**n = 0 **の場合，**n = 1 **と同じ動作）。**

★ **8. ディバグ機能用命令は，タイプ** C **の製品ではサポートしていません。**

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	**トラップ・ベクタ（**00H-1FH**）を指定する** 5 **ビット・データ**
cccc	**条件コードを示す** 4 **ビット・データ**
sp	**スタック・ポインタ（**r3**）**
ep	**エレメント・ポインタ（**r30**）**
list ×	×**個のレジスタ・リスト**

2. 実行クロックの凡例

略　　号	意　　味
I	**命令実行直後に他の命令を実行する場合（**issue**）**
r	**命令実行直後に同一命令を繰り返す場合（**repeat**）**
l	**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（**latency**）**

第6章　割り込みと例外

割り込みは，プログラムの実行とは別に独立に発生する事象で，マスカブル割り込みとノンマスカブル割り込み（NMI）があります。例外は，プログラムの実行に依存して発生する事象で，ソフトウエア例外，例外トラップ，ディバグ・トラップがあります。

割り込み，または例外が発生した場合には，各要因（割り込み／例外要因）ごとに固定的にアドレスが決まっているハンドラへと制御が移されます。割り込み／例外要因は，割り込み要因レジスタ（ECR）に格納される例外コードで知ることができます。各ハンドラでは ECR レジスタを解析し，適切な割り込み，または例外処理を行います。復帰 PC，復帰 PSW は，各種の状態退避レジスタ（EIPC，EIPSW または FEPC，FEPSW）に書き込まれます。

割り込み処理，ソフトウエア例外からの復帰は，RETI 命令により行われます。また，例外トラップ，ディバグ・トラップからの復帰は，DBRET 命令により行われます。復帰処理では，状態退避レジスタから復帰 PC，復帰 PSW を取り出し，復帰 PC へ制御を移します。

表6－1　割り込み，例外コード一覧

割り込み／例外要因		発生要因	分　類	例外コード	ハンドラ・アドレス	復帰 PC
名　称						
ノンマスカブル割り込み（NMI）[注1]		NMI0 入力	割り込み	0010H	00000010H	next PC [注2]
		NMI1 入力	割り込み	0020H	00000020H	next PC [注2,3]
		NMI2 入力[注4]	割り込み	0030H	00000030H	next PC [注2,3]
マスカブル割り込み		注5	割り込み	注5	注6	next PC [注2]
ソフトウエア例外	TRAP0n（n = 0-FH）	TRAP 命令	例外	004nH	00000040H	next PC
	TRAP1n（n = 0-FH）	TRAP 命令	例外	005nH	00000050H	next PC
例外トラップ（ILGOP）		不正命令コード	例外	0060H	00000060H	next PC [注7]
ディバグ・トラップ[注8]		DBTRAP 命令[注8]	例外	0060H	00000060H	next PC

注 1. 製品により，搭載しているノンマスカブル割り込みの発生要因は異なります。

2. 次の命令の実行中に割り込みを受け付けた場合を除きます（命令実行中に割り込みを受け付けると実行を中止し，割り込み処理完了後に再実行されます。この場合，中断された命令のアドレスが復帰 PC となります）。

- ロード命令（SLD.B, SLD.BU, SLD.H, SLD.HU, SLD.W），除算命令（DIV, DIVH, DIVU, DIVHU）
- PREPARE，DISPOSE 命令（スタック・ポインタの更新前に割り込みが発生した場合のみ）

3. RETI 命令による復帰はできません。割り込み処理後にシステム・リセットを行ってください。

4. PSW の NP フラグがセット（1）されていても受け付けられます。

5. 割り込みの種類ごとに異なります。

6. 上位 16 ビットは 0000H，下位 16 ビットは例外コードと同一の値です。

7. 不正命令の実行アドレスは，「復帰 PC – 4」で求められます。

★ **8.** タイプ C の製品ではサポートしていません。

備考 復帰 PC：割り込み，例外処理起動時に，EIPC または FEPC に退避される PC 値
next PC：割り込み，例外処理後に処理を開始する PC 値

6. 1　割り込み処理

6. 1. 1　マスカブル割り込み

割り込みコントローラ（INTC）の割り込み制御レジスタにより割り込み受け付けをマスクできる割り込みです。

INTC では，受け付けた割り込みの最高位の割り込みに基づき，CPU に対して割り込み要求を発生します。

割り込み要求入力（INT 入力）によりマスカブル割り込みが発生した場合，CPU は次の処理を行い，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> ECR の下位ハーフワード（EICC）に例外コードを書き込みます。

<4> PSW の ID フラグをセット（1）し，EP フラグをクリア（0）します。

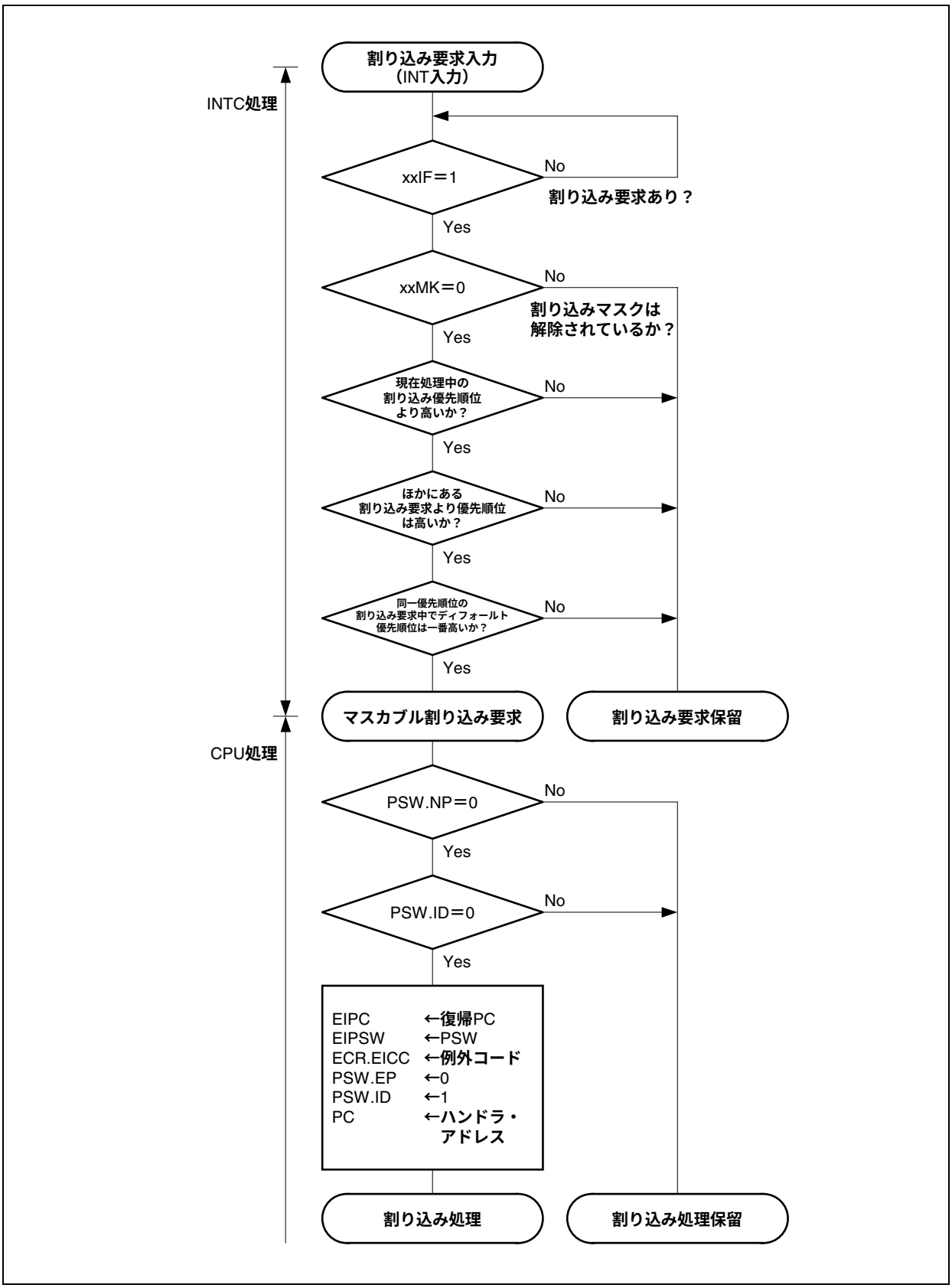
<5> PC に各割り込みに対するハンドラ・アドレスをセットし，制御を移します。

状態退避レジスタには EIPC，EIPSW を使用します。なお，割り込みコントローラにおいてマスクされている INT 入力や割り込み処理中（PSW の NP フラグが 1，または PSW の ID フラグが 1 のとき）に発生した INT 入力は，INTC 内部で保留されます。この場合，マスクを解除するか，LDSR 命令を使用して PSW の NP フラグと PSW の ID フラグを 0 にすると，保留していた INT 入力により，新たなマスカブル割り込み処理が開始されます。

なお，EIPC，EIPSW は 1 組しかないため，多重割り込みを許可する場合には，プログラムによってこのレジスタを退避する必要があります。

マスカブル割り込みの処理形態を次に示します。

図6－1　マスカブル割り込みの処理形態



6. 1. 2　ノンマスカブル割り込み

ノンマスカブル割り込みは，命令などによる割り込み受け付け禁止ができない常時受け付けが可能な割り込みです。ノンマスカブル割り込みは，NMI 入力により発生します。

ノンマスカブル割り込みが発生した場合は，CPU は次の処理を行いハンドラ・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> ECR の上位ハーフワード（FECC）に例外コード（0010H）を書き込みます。
<4> PSW の NP, ID フラグをセット（1）し，EP フラグをクリア（0）します。
<5> PC にノンマスカブル割り込みに対するハンドラ・アドレスをセットし，制御を移します。

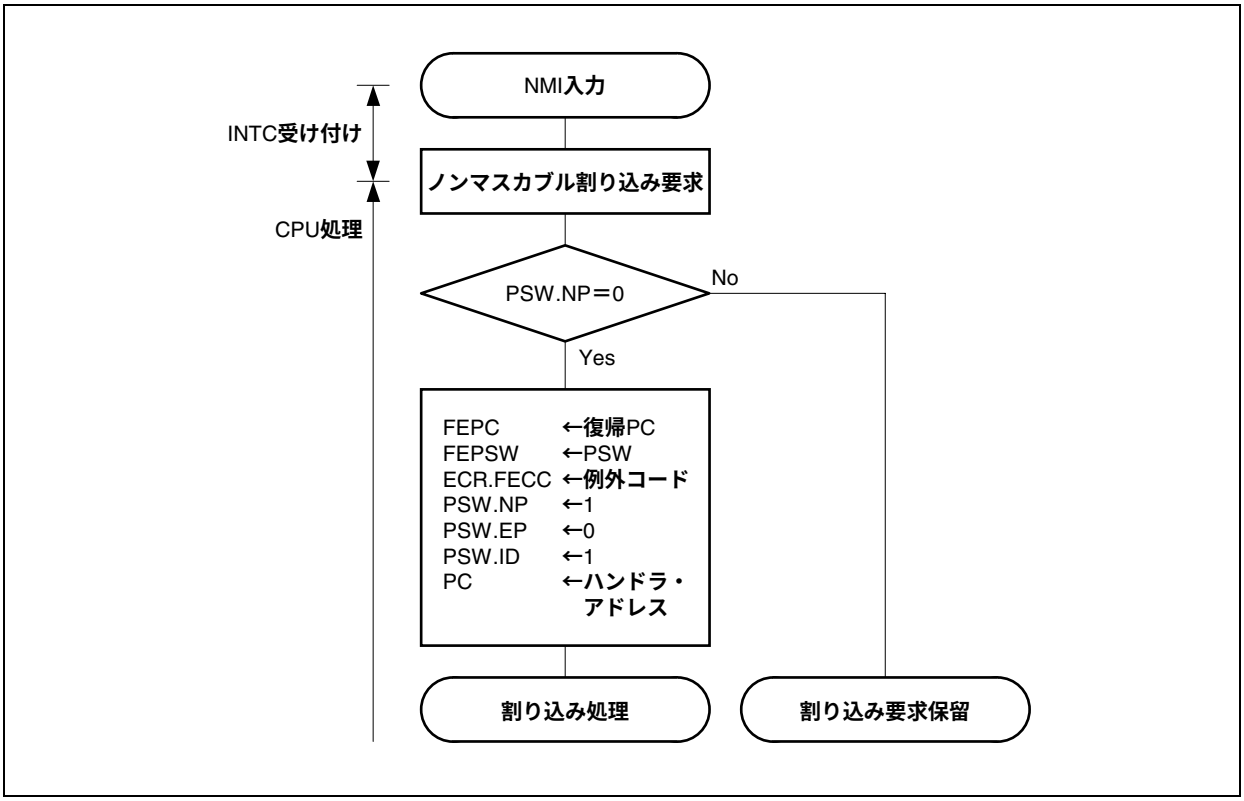
状態退避レジスタには，FEPC, FEPSW を使用します。

また，ノンマスカブル割り込み処理中（PSW の NP フラグが 1）に発生したノンマスカブル割り込み要求は，割り込みコントローラ内部で保留されます。この場合，RETI 命令と LDSR 命令を使用して，PSW の NP フラグを 0 にすると，保留されていたノンマスカブル割り込み要求により新たなノンマスカブル割り込み処理が開始されます。

★　タイプ A, B, C の製品の場合，NMI0, NMI1 の割り込み処理中に NMI2 が発生した場合だけ，NP フラグの値によらず，NMI2 処理が実行されます。

ノンマスカブル割り込みの処理形態を次に示します。

図6－2　ノンマスカブル割り込みの処理形態



6. 2　例外処理

6. 2. 1　ソフトウエア例外

ソフトウエア例外は，TRAP 命令の実行により発生する常時受け付けが可能な例外です。

ソフトウエア例外が発生した場合，CPU は次の処理を行いハンドラ・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。

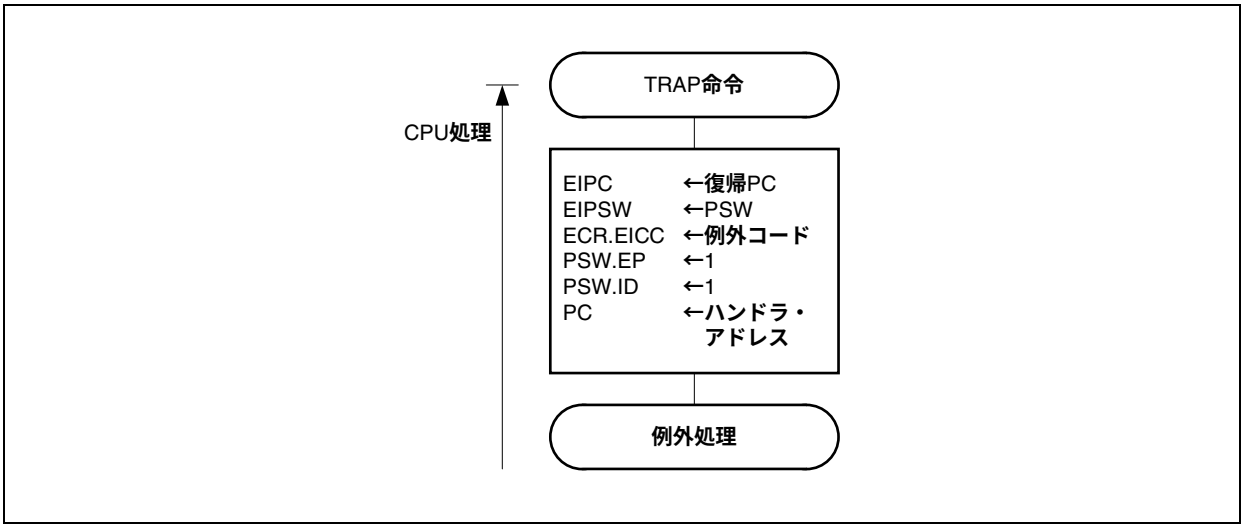
<3> ECR（割り込み要因）の下位 16 ビット（EICC）に例外コードを書き込みます。

<4> PSW の EP, ID フラグをセット（1）します。

<5> PC にソフトウエア例外に対するハンドラ・アドレス（00000040H または 00000050H）をセットし，制御を移します。

ソフトウエア例外の処理形態を次に示します。

図6－3　ソフトウエア例外の処理形態

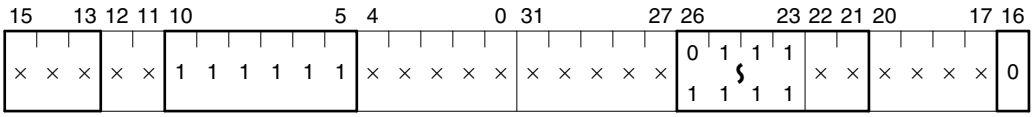


6. 2. 2　例外トラップ

例外トラップは，命令の不正実行が発生した場合に要求される例外です。不正命令コード・トラップ（ILGOP：Illegal opcode trap）が例外トラップにあたります。

不正命令は，命令コードのオペコード（ビット 10-5）が 111111B で，サブオペコード（ビット 26-23）が 0111B-1111B，サブオペコード（ビット 16）が 0B であるものです。この不正命令に当てはまる命令を実行したときに，例外トラップが発生します。

図6－4　不正命令コード



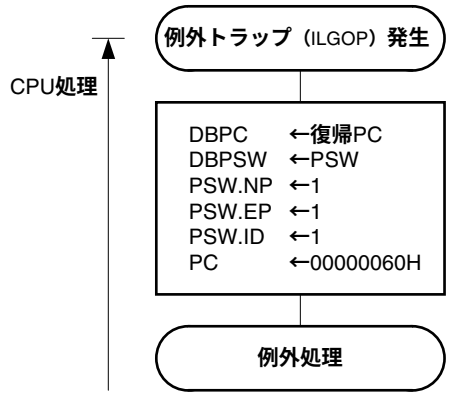
備考　×：任意，□：オペコード／サブオペコード部

例外トラップが発生した場合，CPU は次の処理を行いハンドラ・ルーチン（ディバグ・モニタ・ルーチン）へ制御を移します。

<1> 復帰 PC を DBPC に退避します。
<2> 現在の PSW を DBPSW へ退避します。
<3> PSW の NP, EP, ID フラグをセット（1）します。
★ <4> DIR レジスタの DM ビットをセット（1）します。
<5> PC に例外トラップに対するハンドラ・アドレス（00000060H）をセットし，ディバグ・モニタ・ルーチンに制御を移します。

例外トラップの処理形態を次に示します。

図6－5　例外トラップの処理形態



注意　命令，または，不正命令として定義されていない命令を実行したときの動作は保証しません。

備考　不正命令の実行アドレスは，「復帰 PC – 4」で求められます。

6. 2. 3　ディバグ・トラップ

★　ディバグ・トラップは，DBTRAP 命令の実行，またはディバグ機能のトラップにより発生する常時受け付けが可能な例外です。

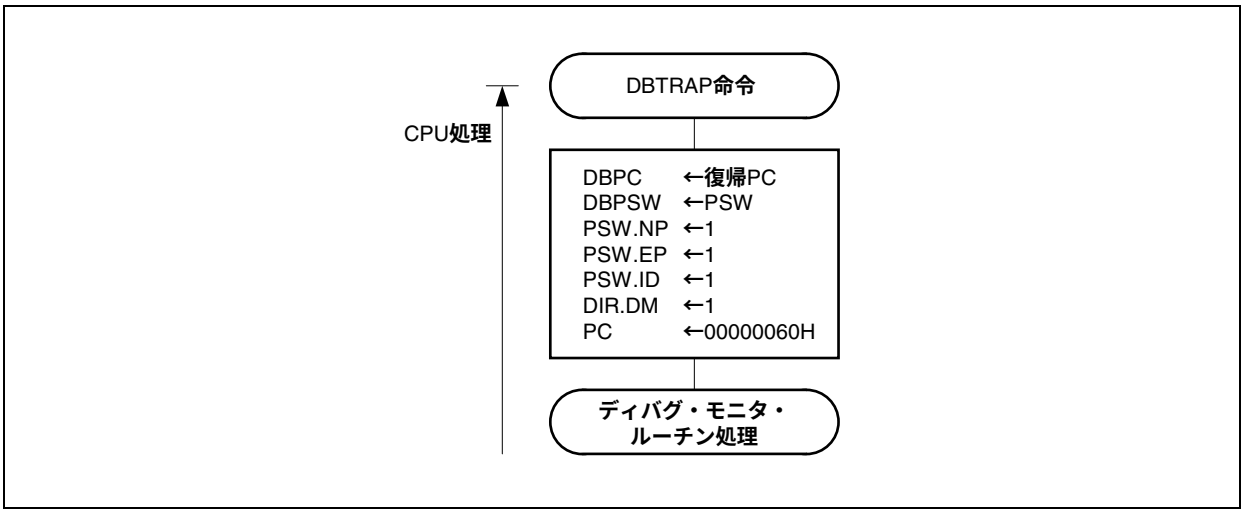
ディバグ・トラップが発生した場合，CPU は次の処理を行います。

<1> 復帰 PC を DBPC に退避します。
<2> 現在の PSW を DBPSW へ退避します。
<3> PSW の NP, EP, ID フラグをセット（1）します。
<4> DIR の DM フラグをセット（1）します。
<5> PC にディバグ・トラップに対するハンドラ・アドレス（00000060H）をセットし，ディバグ・モニタ・ルーチンに制御を移します。

★　注意　ディバグ・トラップは，タイプ C の製品ではサポートしていません。

ディバグ・トラップの処理形態を次に示します。

図6－6　ディバグ・トラップの処理形態



6. 3　割り込み，例外処理からの復帰

6. 3. 1　割り込み，ソフトウエア例外からの復帰

割り込み，ソフトウエア例外からの復帰は，すべて RETI 命令により行われます。

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 の EP フラグが 0 かつ PSW の NP フラグが 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。

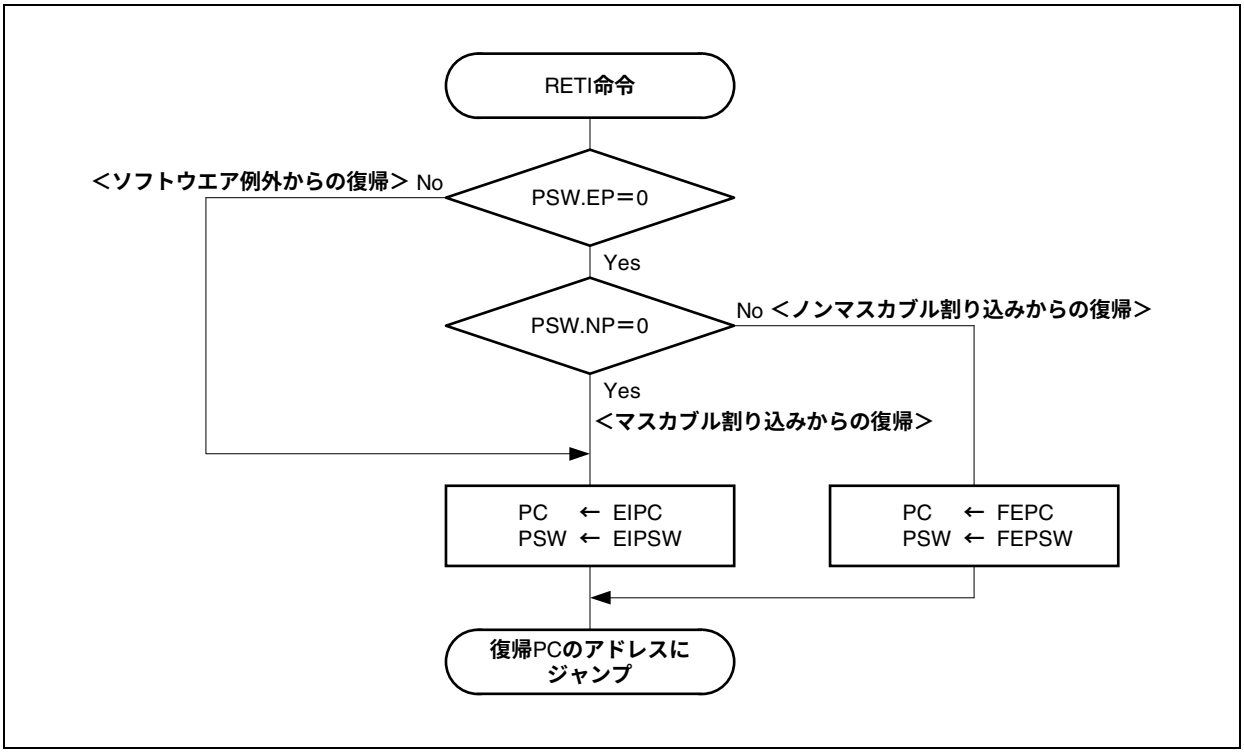
各割り込み処理からの復帰時は，PC, PSW を正常にリストアするために，RETI 命令の直前で，LDSR 命令を使用し，PSW の NP フラグ，EP フラグの各フラグを次の状態にしておく必要があります。

- ノンマスカブル割り込み処理からの復帰時[注]：PSW の NP フラグ= 1，EP フラグ = 0
- マスカブル割り込み処理からの復帰時　　：PSW の NP フラグ = 0，EP フラグ = 0
- 例外処理からの復帰時　　：PSW の EP フラグ = 1

★　注　タイプ A, B, C の製品の場合，NMI1, NMI2 は，RETI 命令による復帰はできません。割り込み処理後にシステム・リセットを行ってください。また，NMI2 は PSW の NP フラグがセット（1）されていても受け付けられます。

割り込み，例外処理からの復帰の処理形態を次に示します。

図6－7　割り込み，ソフトウエア例外からの復帰の処理形態



6. 3. 2　例外トラップ，ディバグ・トラップからの復帰

例外トラップ，ディバグ・トラップからの復帰は，DBRET 命令により行われます。

DBRET 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。

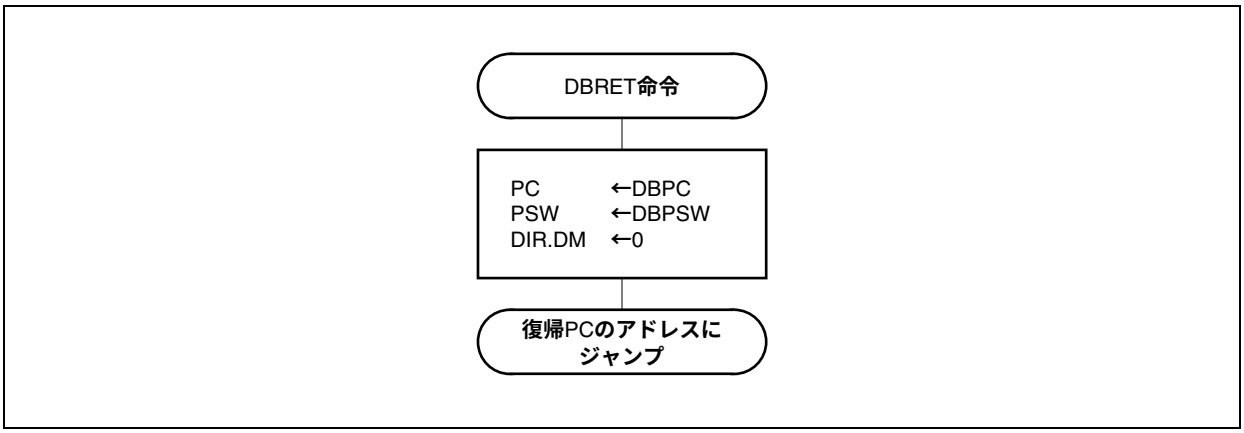
<1> DBPC, DBPSW から復帰 PC, PSW を取り出します。

<2> 取り出した復帰 PC, PSW のアドレスに制御を移します。

<3> 例外トラップ，ディバグ・トラップから復帰すると，DIR の DM フラグをクリア（0）します。

例外トラップ，ディバグ・トラップからの復帰の処理形態を次に示します。

図6－8　例外トラップ，ディバグ・トラップからの復帰の処理形態



第7章　リセット

7.1　リセット後のレジスタの状態

リセット端子にロウ・レベルが入力されると，システム・リセットがかかり，プログラム・レジスタとシステム・レジスタは，表 7－1 に示す状態になります。リセット端子への入力がハイ・レベルになるとリセット状態が解除され，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イニシャライズしてください。

表7－1　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	割り込み時状態退避レジスタ（EIPC）	0xxxxxxxH
	割り込み時状態退避レジスタ（EIPSW）	00000xxxH
	NMI 時状態退避レジスタ（FEPC）	0xxxxxxxH
	NMI 時状態退避レジスタ（FEPSW）	00000xxxH
	割り込み要因レジスタ（ECR）	00000000H
	プログラム・ステータス・ワード（PSW）	00000020H
	CALLT 実行時状態退避レジスタ（CTPC）	0xxxxxxxH
	CALLT 実行時状態退避レジスタ（CTPSW）	00000xxxH
	例外／ディバグ・トラップ時状態退避レジスタ（DBPC）	0xxxxxxxH
	例外／ディバグ・トラップ時状態退避レジスタ（DBPSW）	00000xxxH
	CALLT ベース・ポインタ（CTBP）	0xxxxxxxH
	ディバグ・インタフェース・レジスタ（DIR）	00000040H
	ブレークポイント制御レジスタ 0（BPC0）	00xxxxx0H
	ブレークポイント制御レジスタ 1（BPC1）	00xxxxx0H
	プログラム ID レジスタ（ASID）	000000xxH
	ブレークポイント・アドレス設定レジスタ 0（BPAV0）	0xxxxxxxH
	ブレークポイント・アドレス設定レジスタ 1（BPAV1）	0xxxxxxxH
	ブレークポイント・アドレス・マスク・レジスタ 0（BPAM0）	0xxxxxxxH
	ブレークポイント・アドレス・マスク・レジスタ 1（BPAM1）	0xxxxxxxH
	ブレークポイント・データ設定レジスタ 0（BPDV0）	不定
	ブレークポイント・データ設定レジスタ 1（BPDV1）	不定
	ブレークポイント・データ・マスク・レジスタ 0（BPDM0）	不定
	ブレークポイント・データ・マスク・レジスタ 1（BPDM1）	不定

備考　x：不定

7.2　起　　動

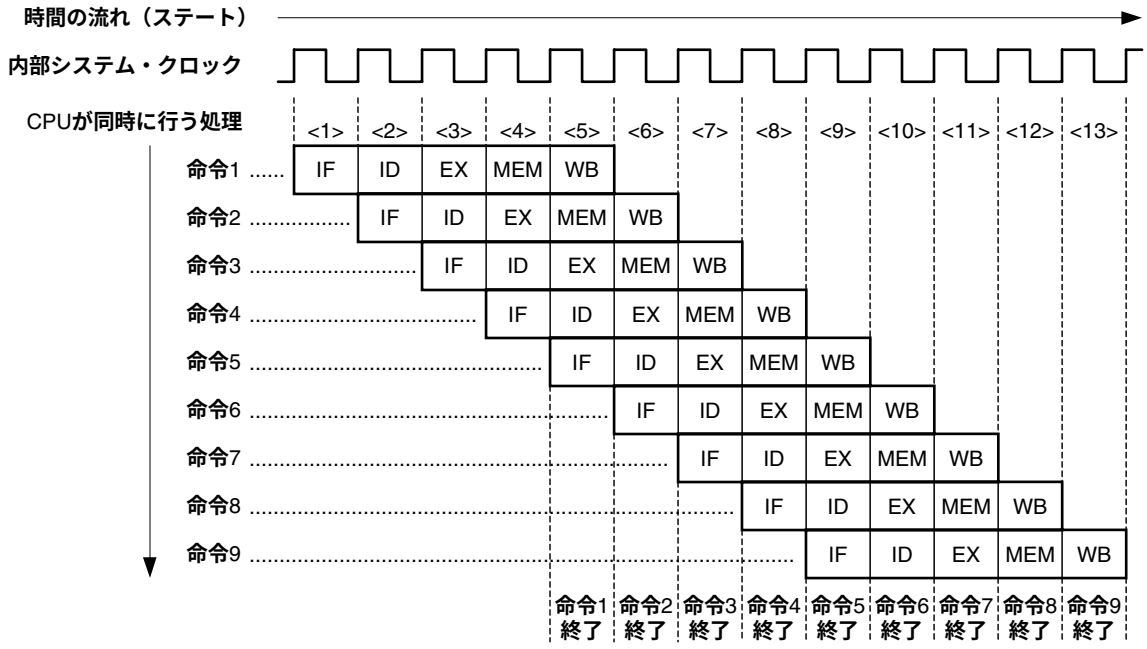
CPU は，リセットにより 00000000H 番地からプログラムの実行を開始します。

なお，リセット直後は，割り込み要求は受け付けられません。プログラムで割り込み処理を使用する場合は，PSW の ID フラグをクリア（0）してください。

第8章　パイプライン

V850E1 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，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 つ続けて実行したときの CPU の処理を図 8－1 に示します。

図8－1　標準的な命令を9つ続けて実行する例



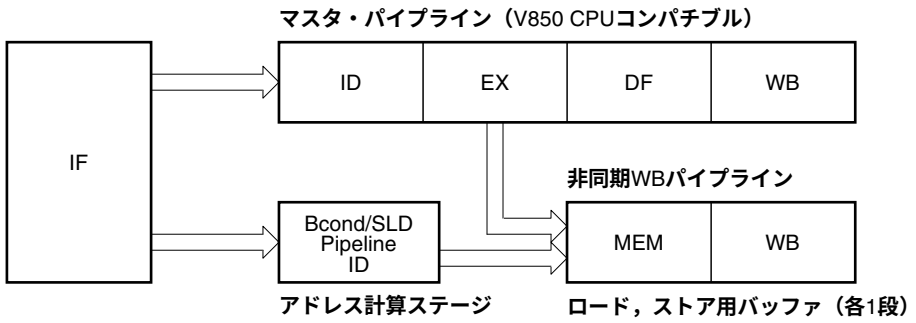
IF（インストラクション・フェッチ）	：命令のフェッチを行い，フェッチ・ポインタをインクリメントします。
ID（インストラクション・デコード）	：命令をデコードし，イミーディエト・データの作成とレジスタの読み出しを行います。
EX（ALU，乗算器，バレル・シフタの実行）	：デコードした命令を実行します。
MEM（メモリ・アクセス）	：対象となるアドレスのメモリにアクセスします。
WB（ライトバック）	：実行した結果をレジスタに書き込みます。

<1> - <13>は，CPU のステートを示します。各ステートでは，命令「n」のライトバック（WB），命令「n＋1」のメモリ・アクセス（MEM），命令「n＋2」の実行（EX），命令「n＋3」のデコード（ID），命令「n＋4」のフェッチ（IF）が同時に行われます。標準的な命令では，IF ステージから WB ステージまで 5 クロックかかります。しかし，同時に 5 命令を処理できるため，標準的な命令では平均 1 クロックごとに実行可能です。

8.1 特　　徴

V850E1 CPU は，パイプラインの最適化を行うことにより，CPI（Cycle per instruction）を従来の V850 CPU よりも向上させています。V850E1 CPU のパイプライン構成を図 8－2 に示します。

図8－2　パイプライン構成



備考　DF（データ・フェッチ）：実行データを WB ステージに転送します。

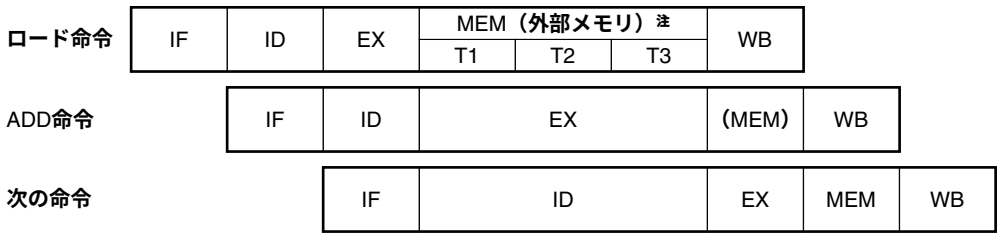
8. 1. 1　ノンブロッキング・ロード／ストア

外部メモリ・アクセス時にパイプラインが停止することなく，効率的な処理が可能です。

例として，外部メモリに対するロード命令実行後に ADD 命令を実行する場合の V850 CPU と V850E1 CPU のパイプライン動作の比較を図 8－3 に示します。

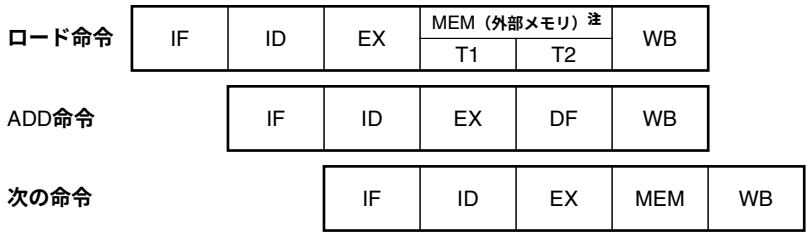
図8－3　ノンブロッキング・ロード／ストア

（a）従来（V850 CPU）：MEMステージ完了までパイプラインが停止



注　外部に対する基本バス・サイクルは 3 クロックです。

（b）V850E1 CPU：非同期WBパイプラインによる効率的なパイプライン処理



注　MEMC の外部に対する基本バス・サイクルは 2 クロックです。

（1）V850 CPU の場合

ADD 命令の EX ステージは，本来 1 クロックで実行されます。しかし，直前のロード命令の MEM ステージ実行中，ADD 命令の EX ステージに待ち時間が発生します。これは，パイプライン上の 5 つの命令が，同一内部クロック間に同じステージを実行できないためです。この影響により，ADD 命令の次の命令の ID ステージにも待ち時間が発生します。

（2）V850E1 CPU の場合

マスタ・パイプラインのほかに，MEM ステージを必要とする命令用に非同期 WB パイプラインを持っています。このため，ロード命令の MEM ステージは，非同期 WB パイプラインで処理されます。図 8－3 の例では ADD 命令はマスタ・パイプラインで処理されるためパイプラインの待ち時間が生じることなく，効率的に命令を実行できます。

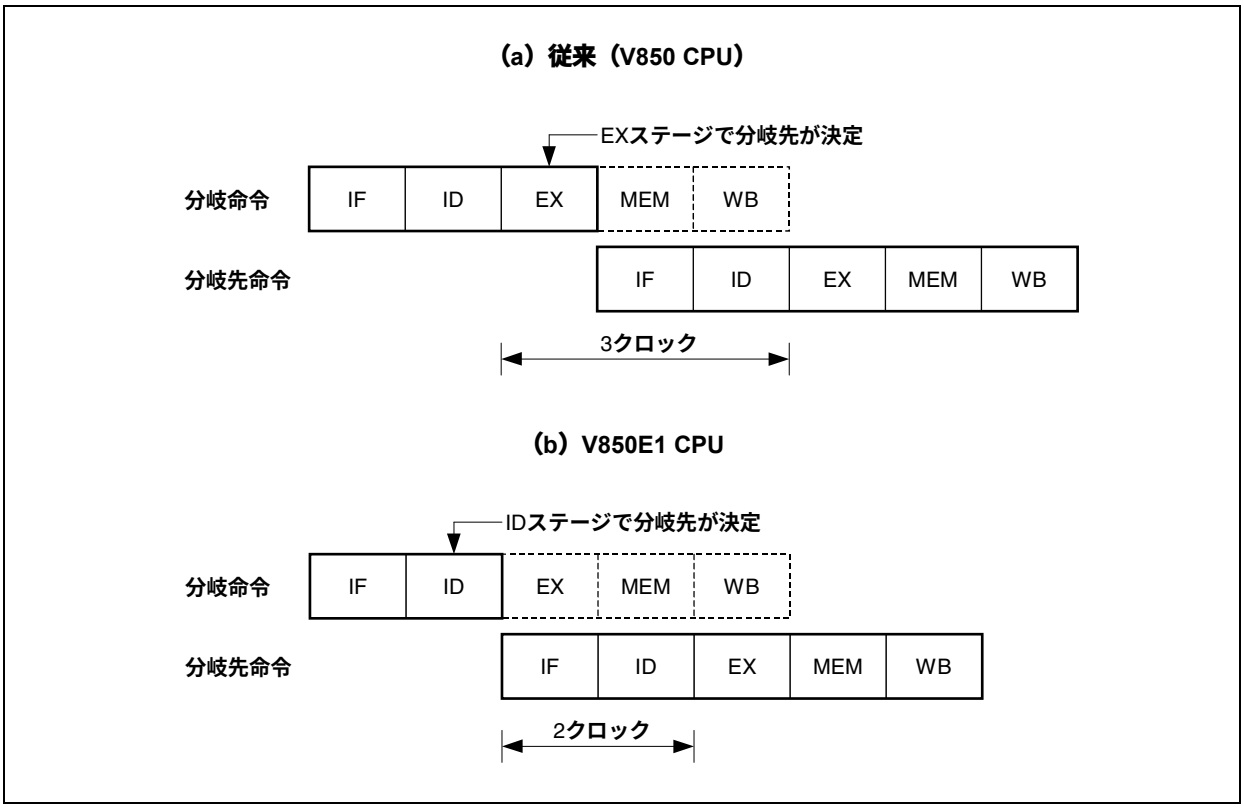
8. 1. 2　2 クロック分岐

分岐命令実行時は，ID ステージで分岐先が決定します。

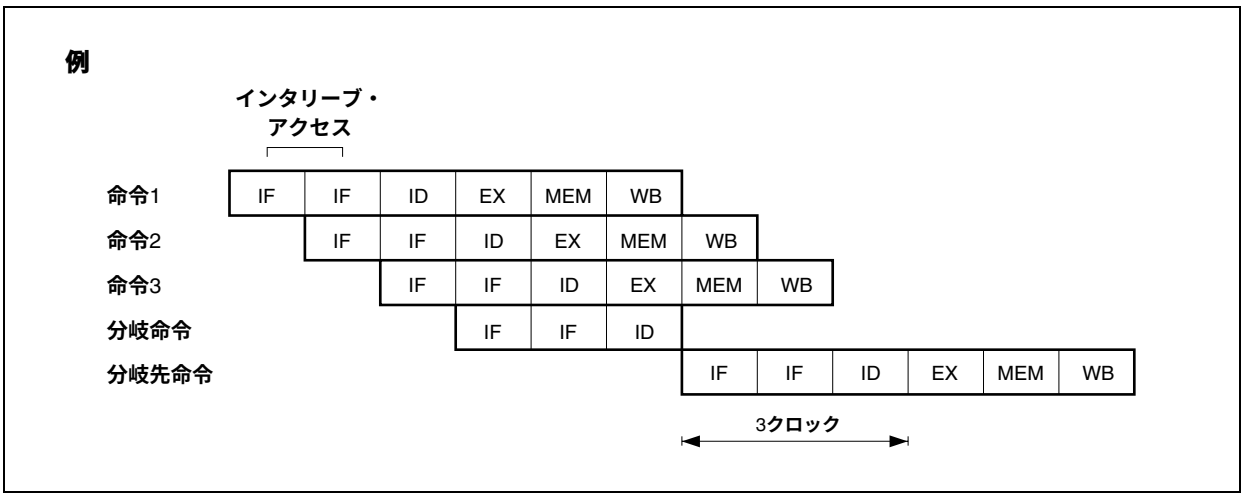
従来の V850 CPU の場合，分岐命令実行時の分岐先は，EX ステージ実行後に決定されていましたが，V850E1 CPU では，分岐／SLD 命令用に追加したアドレス計算ステージにより，ID ステージで分岐先が決定します。このため，従来の V850 CPU より，1 クロック早く分岐先の命令をフェッチすることができます。

V850 CPU と V850E1 CPU の分岐命令でのパイプライン動作の比較を図 8－4 に示します。

図8－4　分岐命令でのパイプライン動作



★ **備考**　タイプ D, E の製品は，内蔵フラッシュ・メモリまたは内蔵マスク ROM に対しインタリーブ・アクセスを行います。このため割り込み発生直後の命令フェッチや，分岐命令実行後の命令フェッチに 2 クロック，タイプ E の製品は 3 クロックかかります。したがって，分岐先命令の ID ステージ実行までに 3 クロック，タイプ E の製品は 4 クロックかかります。

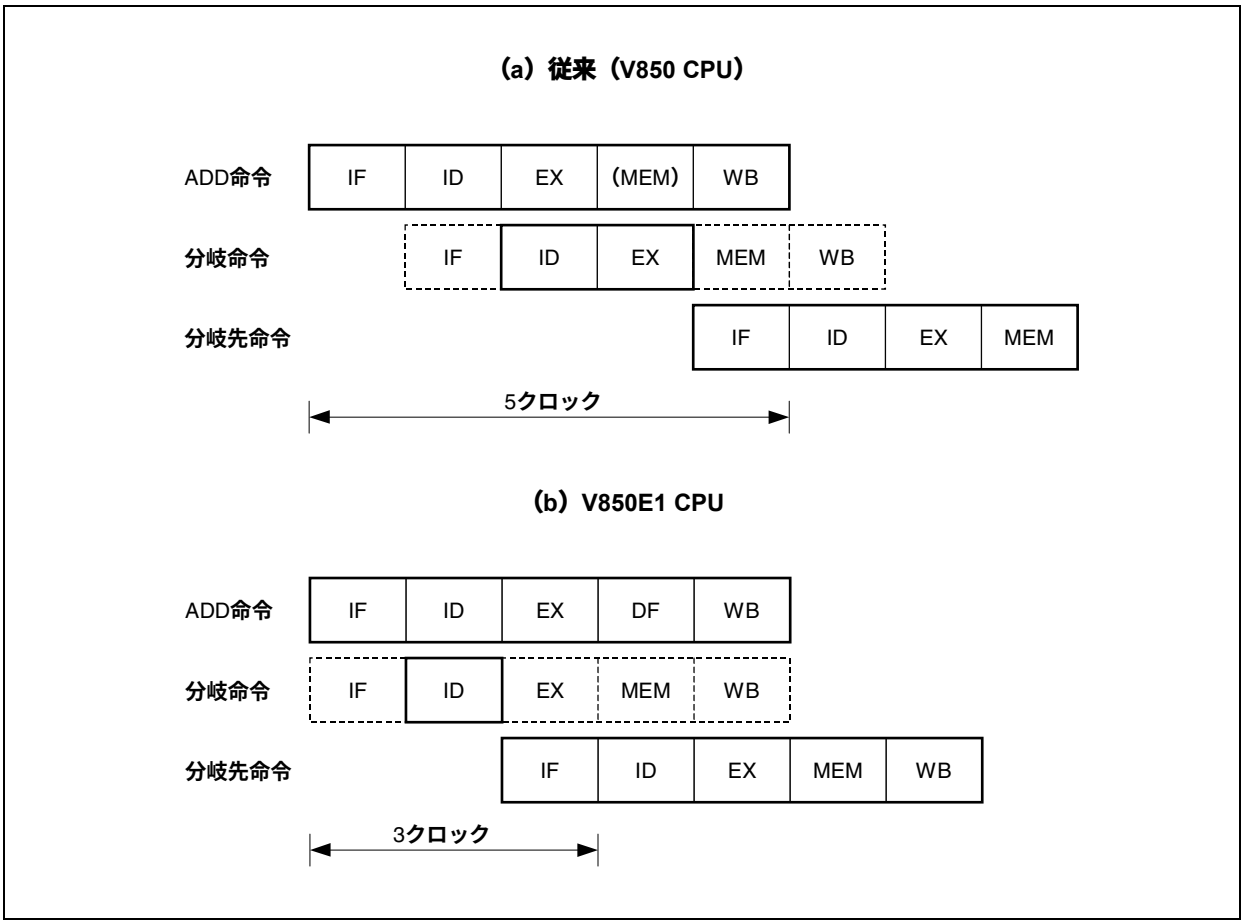


8. 1. 3　効率的なパイプライン処理

V850E1 CPU は，マスタ・パイプライン上の ID ステージのほかに，分岐／SLD 命令用の ID ステージを持っているため，効率的なパイプライン処理が行えます。

例として，ADD 命令の IF ステージで，次の分岐命令をフェッチした場合のパイプライン動作例を図 8－5 に示します（専用バスに直結された ROM に対する命令フェッチは，32 ビット単位で行われます。図 8－5 での ADD 命令，分岐命令はどちらも 16 ビット・フォーマットの命令です）。

図8－5　分岐命令の並列実行



（1）V850 CPU の場合

ADD 命令の IF ステージで次の分岐命令の命令コードまでフェッチしますが，ADD 命令の ID ステージと分岐命令の ID ステージを同一クロック中に実行できません。そのため，分岐命令のフェッチから分岐先命令のフェッチまで，5 クロックかかります。

（2）V850E1 CPU の場合

マスタ・パイプライン上の ID ステージのほかに，分岐／SLD 命令用の ID ステージを持っているため，同一クロック中に並行して ADD 命令の ID ステージと分岐命令の ID ステージを実行できます。このため，分岐命令のフェッチ開始から分岐先の命令フェッチ完了まで 3 クロックで実行できます。

注意　SLD 命令と Bcond 命令については，ほかの 16 ビット・フォーマットの命令と同時実行される場合があるため注意が必要です。たとえば，SLD 命令と NOP 命令が同時に実行された場合，NOP 命令によるディレイ・タイムの発生が行われない可能性があります。

8. 2　各命令実行時のパイプラインの流れ

この節では各命令実行時のパイプラインの流れについて説明します。

パイプライン処理のため，メモリや I/O のライト・サイクルが発生する時点で，CPU は後続の命令をすでに実行しています。そのため，I/O 操作や割り込み要求マスクの反映は，次の命令発行（ID ステージ）に対して遅れて実行されます。

★ **（1）タイプ A, B, C の製品の場合**

NPB（NEC ペリフェラル・バス）に専用の割り込みコントローラ（INTC）が接続されている場合は，INTC へのアクセスを CPU が検出して割り込み要求のマスク処理を行うため，直後の命令からマスカブル割り込みの受け付けを禁止します。

★ **（2）タイプ D, E, F の製品の場合**

内蔵 INTC へのアクセスを CPU が検出（ID ステージ）して割り込み要求のマスク処理を行うため，割り込みマスク操作を行う場合，直後の命令からマスカブル割り込みの受け付けを禁止します。

8. 2. 1　ロード命令

注意　ノンブロッキング制御のため，MEM ステージの間にバス・サイクルが完了している保証はありません。ただし，周辺 I/O 領域へのアクセスはブロッキング制御となるため，MEM ステージでバス・サイクルの完了を待ち合わせます。

★ **タイプ A, B, C の製品の場合，プログラマブル周辺 I/O 領域へのアクセスは，ノンブロッキング制御です。**

(1) LD 命令

[対象の命令]　LD.B, LD.BU, LD.H, LD.HU, LD.W

[パイプライン]

	<1>	<2>	<3>	<4>	<5>	<6>
LD命令	IF	ID	EX	MEM	WB	
次命令		IF	ID	EX	MEM	WB

[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。**

(2) SLD 命令

[対象の命令]　SLD.B, SLD.BU, SLD.H, SLD.HU, SLD.W

[パイプライン]

	<1>	<2>	<3>	<4>	<5>	<6>
SLD命令	IF	ID	MEM	WB		
次命令		IF	ID	EX	MEM	WB

[説　　明]　**パイプラインは** IF, ID, MEM, WB **の** 4 **ステージです。**SLD **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。**

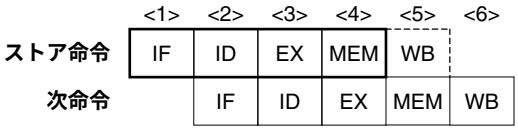
8. 2. 2　ストア命令

注意　ノンブロッキング制御のため，MEM ステージの間にバス・サイクルが完了している保証はありません。ただし，周辺 I/O 領域へのアクセスはブロッキング制御となるため，MEM ステージでバス・サイクルの完了を待ち合わせます。

★ **タイプ A, B, C の製品の場合，プログラマブル周辺 I/O 領域へのアクセスは，ノンブロッキング制御です。**

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

[パイプライン]

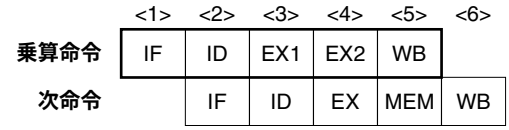


[説　　明]　**パイプラインは** IF, ID, EX, MEM, WB **の** 5 **ステージですが，レジスタへのデータの書き込みがないので** WB **ステージでは何も行いません。**

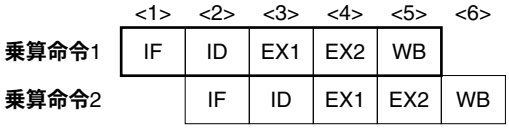
8. 2. 3　乗算命令

[対象の命令]　MUL, MULH, MULHI, MULU

[パイプライン]　**(a) 次命令が乗算命令以外の場合**



(b) 次命令が乗算命令の場合



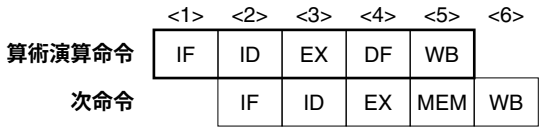
[説　　明]　**パイプラインは** IF, ID, EX1, EX2, WB **の** 5 **ステージです。**EX **ステージは乗算器実行のため** 2 **クロックかかりますが，**EX1 **と** EX2**（通常の** EX **ステージとは異なります）は独立して動作できます。したがって，乗算命令を繰り返しても命令実行クロック数は** 1 **クロックとなります。ただし，乗算命令の直後に実行結果を使用する命令を配置すると，データの待ち合わせ時間が発生します。**

8. 2. 4　算術演算命令

(1) 除算／ワード転送命令以外

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

[パイプライン]



[説　　明]　パイプラインは IF, ID, EX, DF, WB の 5 ステージです。

(2) ワード転送命令

[対象の命令]　MOV　imm32

[パイプライン]

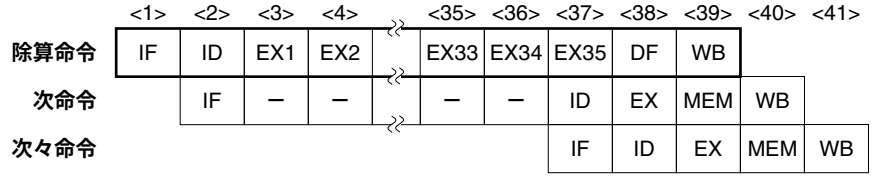
ー：待ちあわせのために挿入されるアイドル

[説　　明]　パイプラインは IF, ID, EX1, EX2（通常の EX ステージ）, DF, WB の 6 ステージです。

(3) 除算命令

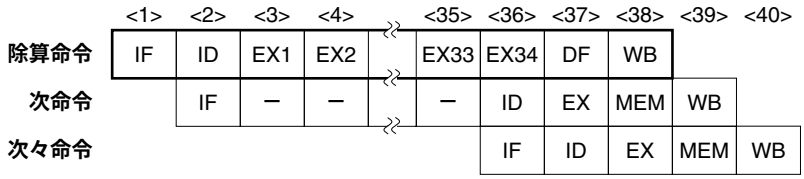
[対象の命令]　DIV, DIVH, DIVHU, DIVU

[パイプライン]　(a) DIV, DIVH 命令の場合



ー：待ちあわせのために挿入されるアイドル

(b) DIVHU, DIVU 命令の場合



ー：待ちあわせのために挿入されるアイドル

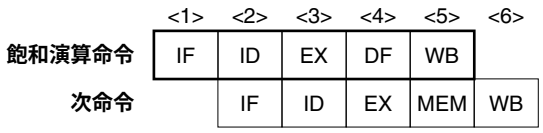
[説　　明]　パイプラインは，DIV, DIVH 命令の場合，IF, ID, EX1-EX35（通常の EX ステージ）, DF, WB の 39 ステージ，DIVHU, DIVU 命令の場合，IF, ID, EX1-EX34（通常の EX ステージ）, DF, WB の 38 ステージです。

[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。

8. 2. 5　飽和演算命令

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

［パイプライン］

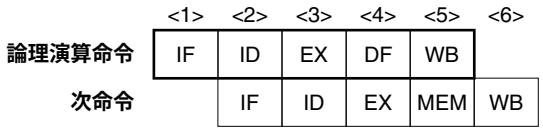


［説　　明］　パイプラインは IF, ID, EX, DF, WB の 5 ステージです。

8. 2. 6　論理演算命令

［対象の命令］　AND, ANDI, BSH, BSW, HSW, NOT, OR, ORI, SAR, SHL, SHR, SXB, SXH, TST, XOR, XORI, ZXB, ZXH

［パイプライン］



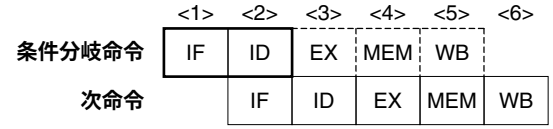
［説　　明］　パイプラインは IF, ID, EX, DF, WB の 5 ステージです。

8. 2. 7　分岐命令

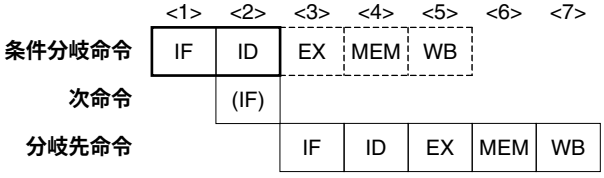
(1) 条件分岐命令（BR 命令を除く）

［対象の命令］　Bcond 命令（BC, BE, BGE, BGT, BH, BL, BLE, BLT, BN, BNC, BNE, BNH, BNL, BNV, BNZ, BP, BSA, BV, BZ）

［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 ステージでは何も行いません。

（a）条件が成立しない場合

分岐命令の命令実行クロック数は 1 となります。

（b）条件が成立した場合

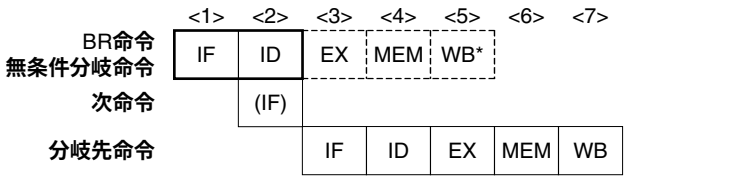
分岐命令の命令実行クロック数は 2 となります。分岐命令の次命令の IF は無効となります。

ただし，直前に PSW の内容を書き換える命令があると，フラグ・ハザード発生のために命令実行クロック数は 3 となります。

(2) BR 命令，無条件分岐命令（JMP 命令を除く）

［対象の命令］　BR, JARL, JR

［パイプライン］

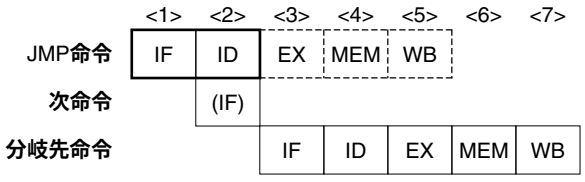


(IF)　：無効となる命令フェッチ
WB*：JR, BR命令の場合は，何も行われませんが，
JARL命令の場合は復帰PCの書き込みが行われます。

［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

［パイプライン］



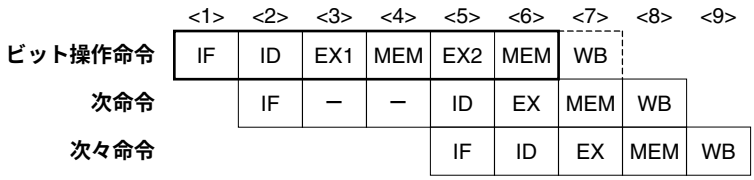
(IF)：無効となる命令フェッチ

［説　　明］　パイプラインは，IF, ID, EX, MEM, WB の 5 ステージですが，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。

8. 2. 8　ビット操作命令

(1) CLR1, NOT1, SET1 命令

[パイプライン]

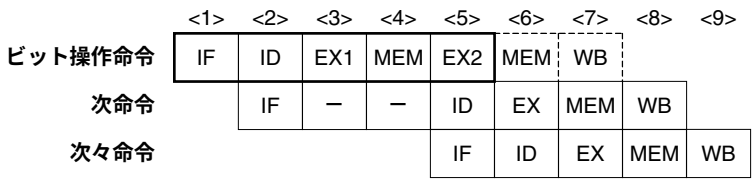


ー：待ちあわせのために挿入されるアイドル

[説　　明]　パイプラインは 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]



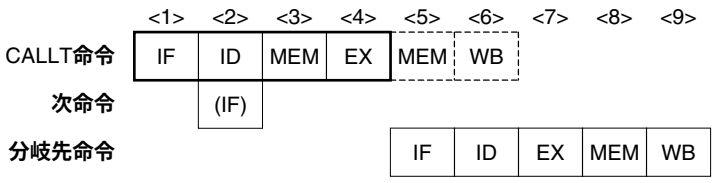
ー：待ちあわせのために挿入されるアイドル

[説　　明]　パイプラインは 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。

8. 2. 9　特殊命令

(1) CALLT 命令

[パイプライン]

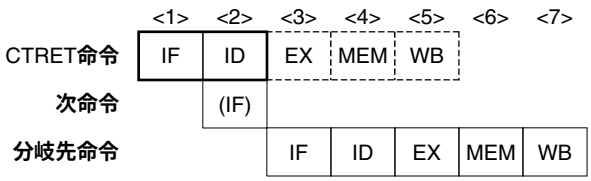


(IF)：無効となる命令フェッチ

[説　　明]　パイプラインは，IF, ID, MEM, EX, MEM, WB の 6 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。

(2) CTRET 命令

[パイプライン]

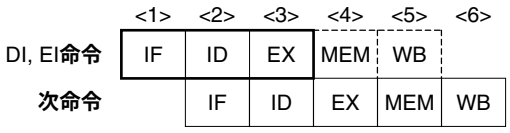


(IF)：無効となる命令フェッチ

[説　　明]　パイプラインは，IF, ID, EX, MEM, WB の 5 ステージですが，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。

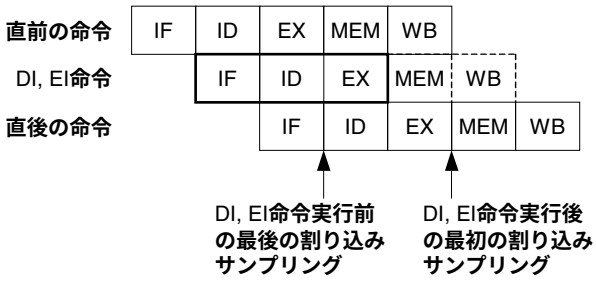
(3) DI, EI 命令

[パイプライン]



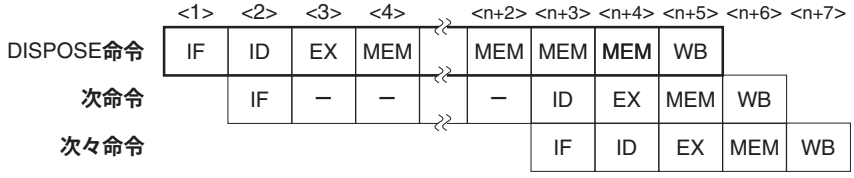
[説　　明]　パイプラインは IF, ID, EX, MEM, WB の 5 ステージですが，メモリへのアクセス，レジスタへのデータ書き込みがないので，MEM ステージ，WB ステージでは何も行いません。

[備　　考]　DI, EI 命令は，いずれも割り込み要求非サンプル命令です。これらの命令実行時における割り込みサンプリング・タイミングは，次のようになります。



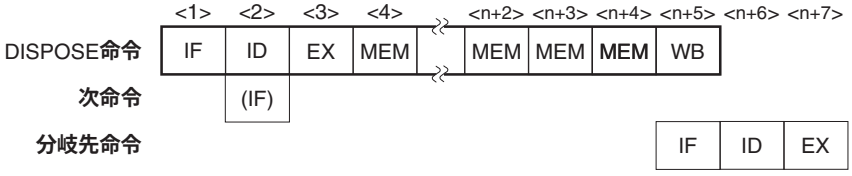
(4) DISPOSE 命令

[パイプライン]　(a) 分岐しない場合



－：待ちあわせのために挿入されるアイドル
n：レジスタ・リスト（list12）で指定されるレジスタの数

（b）分岐する場合

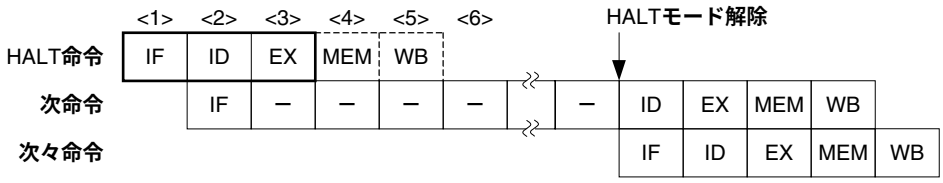


(IF)：無効となる命令フェッチ
－　：待ちあわせのために挿入されるアイドル
n　：レジスタ・リスト（list12）で指定されるレジスタの数

［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。

(5) HALT 命令

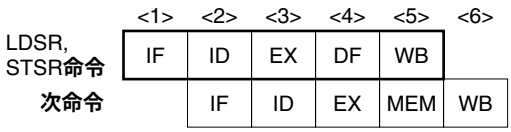
［パイプライン］



［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。

(6) LDSR, STSR 命令

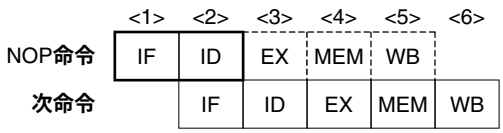
［パイプライン］



［説　　明］　パイプラインは IF, ID, EX, DF, WB の 5 ステージです。システム・レジスタの EIPC, FEPC を設定する 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。

(7) NOP 命令

［パイプライン］



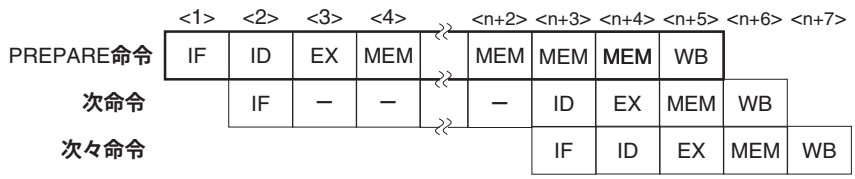
［説　　明］　パイプラインは IF, ID, EX, MEM, WB の 5 ステージですが，演算，メモリへのアクセス，レジスタへのデータ書き込みがないので，EX ステージ，MEM ステージ，WB ステージでは何も行いません。

［注　　意］　SLD 命令と Bcond 命令については，ほかの 16 ビット・フォーマットの命令と同時実

行される場合があるため注意が必要です。たとえば，SLD 命令と NOP 命令が同時に実行された場合，NOP 命令によるディレイ・タイムの発生が行われない可能性があります。

(8) PREPARE 命令

［パイプライン］

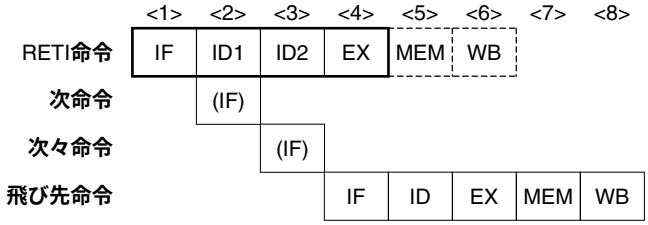


－：待ちあわせのために挿入されるアイドル
n：レジスタ・リスト（list12）で指定されるレジスタの数

［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。

(9) RETI 命令

［パイプライン］

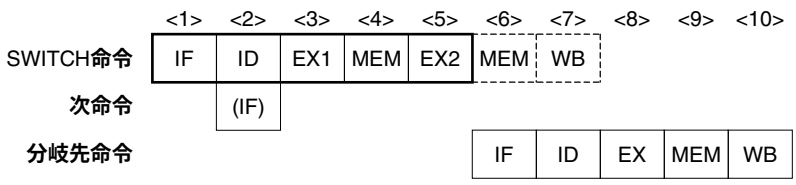


(IF)：無効となる命令フェッチ
ID1：レジスタ選択
ID2：EIPC/FEPC読み込み

［説　　明］　パイプラインは IF, ID1, ID2, EX, MEM, WB の 6 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。ID ステージには 2 クロックかかります。また，次命令の IF と次々命令の IF は無効となります。

(10) SWITCH 命令

［パイプライン］

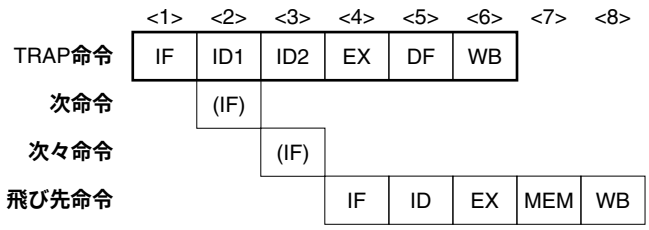


(IF)：無効となる命令フェッチ

［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) TRAP 命令

[パイプライン]



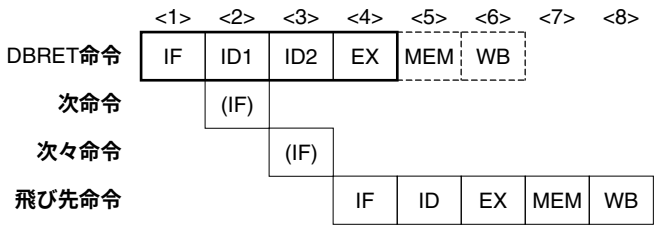
(IF)：無効となる命令フェッチ
ID1：例外コード（004nH, 005nH）検出（n = 0-FH）
ID2：アドレス生成

[説　　明]　パイプラインは IF, ID1, ID2, EX, DF, WB の 6 ステージです。ID ステージには 2 クロックかかります。また，次命令の IF と次々命令の IF は無効となります。

8. 2. 10　ディバグ機能用命令

(1) DBRET 命令

[パイプライン]

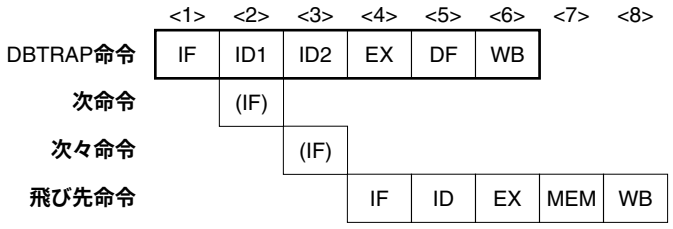


(IF)：無効となる命令フェッチ
ID1：レジスタ選択
ID2：DBPC読み込み

[説　　明]　パイプラインは IF, ID1, ID2, EX, MEM, WB の 6 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。ID ステージには 2 クロックかかります。また，次命令の IF と次々命令の IF は無効となります。

(2) DBTRAP 命令

[パイプライン]



(IF)：無効となる命令フェッチ
ID1：例外コード（0060H）検出
ID2：アドレス生成

[説　　明]　パイプラインは IF, ID1, ID2, EX, DF, WB の 6 ステージです。ID ステージには 2 クロックかかります。また，次命令の IF と次々命令の IF は無効となります。

8. 3　パイプラインの乱れ

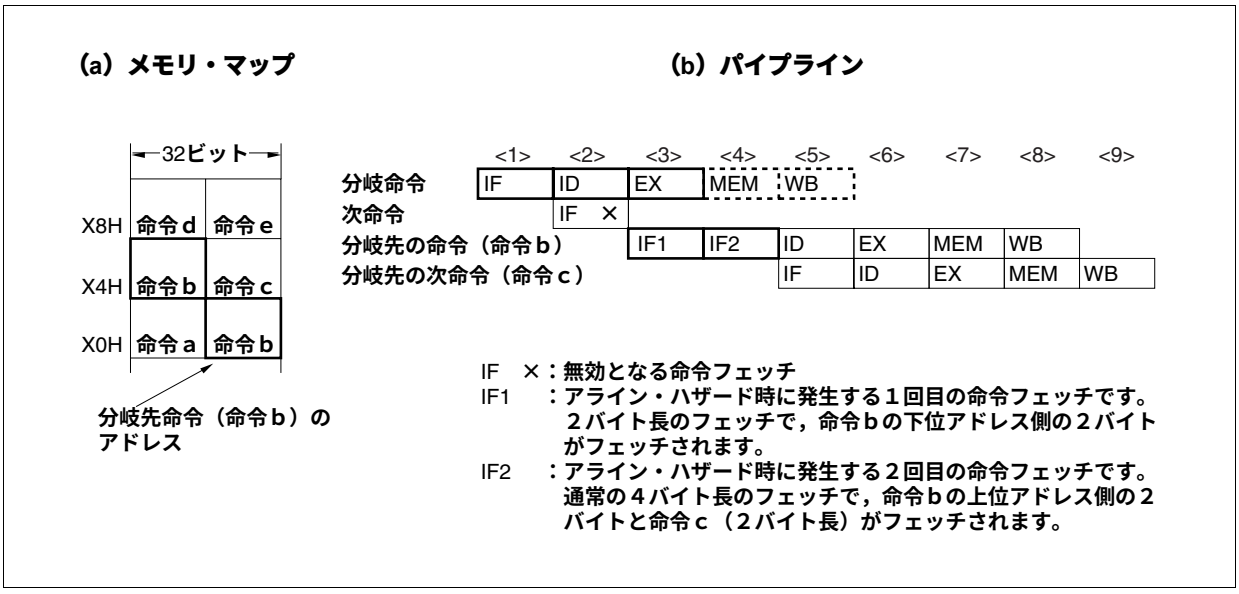
パイプラインはIF（インストラクション・フェッチ）からWB（ライトバック）までの5ステージで構成され，基本的にはそれぞれのステージは1クロックで処理されますが，場合によってはパイプラインが乱れて，実行クロック数が増加する場合があります。この節ではパイプラインを乱す主な要因を示します。

8. 3. 1　アライン・ハザード

分岐先命令のアドレスがワード・アラインでなく（A1 = 1, A0 = 0），かつ4バイト長命令の場合，命令をワード単位にそろえるためにIFを2回続ける必要があります。これをアライン・ハザードと呼びます。

たとえば，命令aから命令eまでがアドレスX0Hから配置されており，命令bは4バイト長命令で，その他の命令は2バイト長命令であるとします。この場合，命令bはX2Hに配置され（A1 = A0 = 0），ワード・アライン（A1 = 0, A0 = 0）となっていません。したがって，この命令bが分岐先命令となる場合，アライン・ハザードが発生します。アライン・ハザードが発生した場合の分岐命令の実行クロック数は4となります。

図8－6 アライン・ハザードの例



アライン・ハザードは，次のような対処によって回避が可能で，命令実行速度の向上が図れます。

- 分岐先命令に2バイト長命令を使用する
- 分岐先命令に，ワード境界（A1 = 0, A0 = 0）に配置した4バイト長命令を使用する

8. 3. 2　ロード命令実行結果の参照

ロード命令（LD, SLD）では，MEMステージで読み出されたデータの格納がWBステージで行われます。したがって，ロード命令の直後の命令で同一のレジスタの内容を使用する場合，ロード命令がレジスタの使用を終えるまで，直後の命令はレジスタの使用を遅らせる必要があります。これをハザードと呼びます。

V850E1 CPUは，このハザードを自動的に対処するインタロック機能を持っており，次命令のIDステージを遅らせます。

またV850E1 CPUは，MEMステージで読み出したデータを次命令のIDステージで使用できるように，シ

ョート・パスを持っています。このショート・パスによって，ロード命令によって MEM ステージでデータを読み出すことと，このデータを次命令の ID ステージで使用することを，同一タイミングで行うことができます。

以上のことより，結果を直後の命令で使用する場合，ロード命令の実行クロック数は 2 になります。

図8－7　ロード命令実行結果の参照例

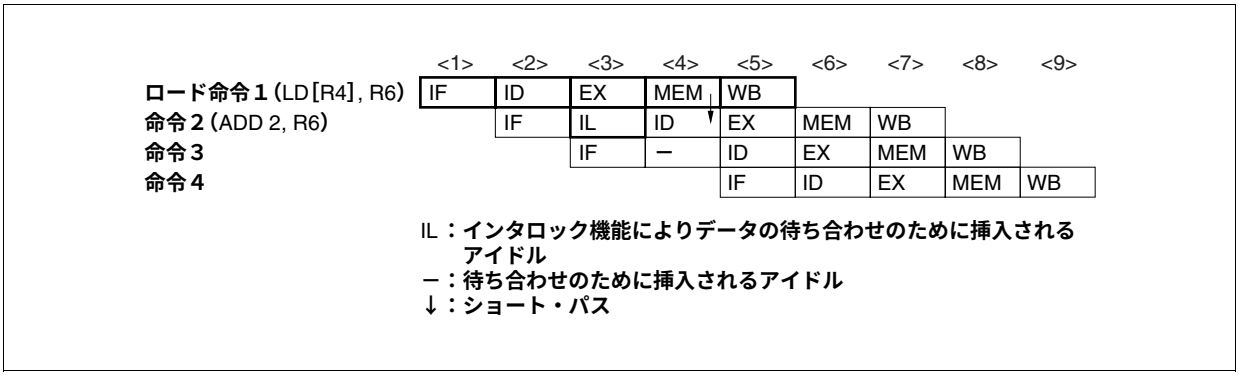


図 8－7 のように，ロード命令の直後にその結果を使用する命令を配置すると，インタロック機能によるデータの待ち合わせ時間が発生し，実行速度が低下します。ロード命令の結果を使用する命令は，ロード命令の 2 命令以後に配置することにより，実行速度の低下が防げます。

8. 3. 3　乗算命令実行結果の参照

乗算命令（MULH, MULHI）では，演算結果のレジスタへの格納が WB ステージで行われます。したがって，乗算命令の直後の命令で同一レジスタの内容を使用する場合，乗算命令がレジスタの使用を終えるまで，直後の命令はレジスタの使用を遅らせる必要があります（ハザードの発生）。

V850E1 CPU ではインタロック機能により直後の命令の ID ステージを遅らせます。また，ショート・パスにより，乗算命令の EX2 ステージと，この演算結果を直後の命令の ID ステージで使用することが，同一タイミングで行えます。

図8－8　乗算命令実行結果の参照例

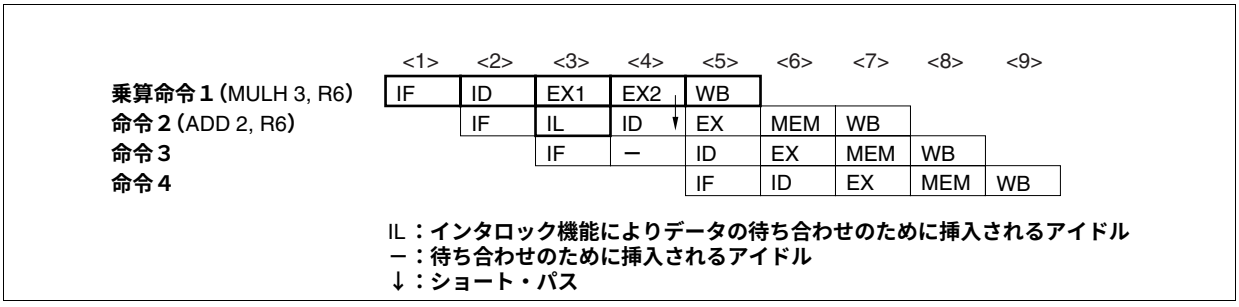


図 8－8 のように，乗算命令の直後にその結果を使用する命令を配置すると，インタロック機能によるデータの待ち合わせ時間が発生し，実行速度が低下します。乗算命令の結果を使用する命令は，乗算命令の 2 命令以後に配置することにより，実行速度の低下を防げます。

8. 3. 4　EIPC, FEPC を対象とする LDSR 命令実行結果の参照

LDSR 命令によって，システム・レジスタの EIPC, FEPC のデータ設定を行い，直後に STSR 命令で同一システム・レジスタの参照を行う場合，LDSR 命令のシステム・レジスタ設定が終わるまで，直後の STSR 命令はシステム・レジスタの使用が遅れます（ハザードの発生）。

V850E1 CPU ではインタロック機能により，直後の STSR 命令の ID ステージを遅らせます。

以上のことより，EIPC, FEPC の LDSR 命令実行結果を直後の STSR 命令で使用する場合，LDSR 命令の実行クロック数は 3 になります。

図8－9　EIPC, FEPCを対象とするLDSR命令実行結果の参照例

IL：インタロック機能によりデータの待ち合わせのために挿入されるアイドル
ー：待ち合わせのために挿入されるアイドル

注　LDSR, STSR 命令で使用しているシステム・レジスタ番号の 0 は EIPC を表します。

図 8－9 のように，EIPC，または FEPC をオペランドとする LDSR 命令の直後に，STSR 命令によってその結果を使用すると，インタロック機能によるデータの待ち合わせ時間が発生し，実行速度が低下します。LDSR 命令の結果を参照する STSR 命令は，LDSR 命令の 3 命令以後に配置することにより，実行速度の低下を防げます。

8. 3. 5　プログラム作成時の注意点

プログラムを作成する場合，次のことに注意するとパイプラインが乱れず，命令実行速度が向上します。

- ロード命令（LD, SLD）の結果を使用する命令は，ロード命令の 2 命令以後に配置する。
- 乗算命令（MULH, MULHI）の結果を使用する命令は，乗算命令の 2 命令以後に配置する。
- LDSR 命令による EIPC，または FEPC への設定結果を STSR 命令により読み出す場合には，LDSR 命令の 3 命令以後に STSR 命令を配置する。
- 分岐先の最初の命令は，2 バイト長命令か，またはワード境界に配置された 4 バイト長命令を使用する。

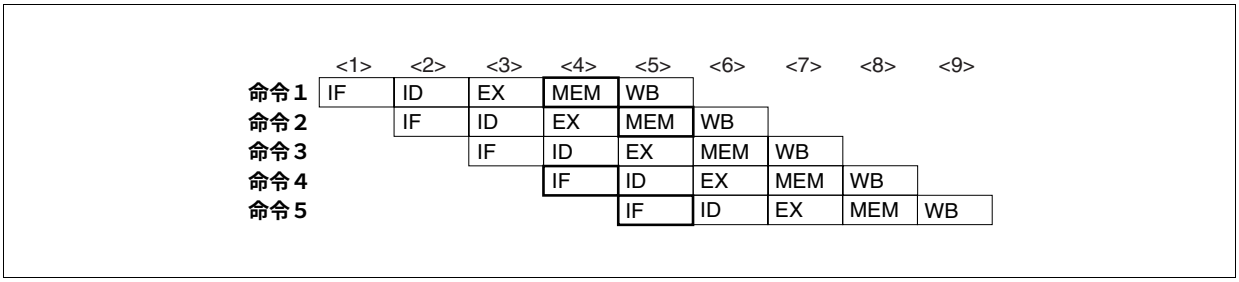
8.4　パイプラインに関する補足事項

8.4.1　ハーバード・アーキテクチャ

V850E1 CPU ではハーバード・アーキテクチャを採用しており，内蔵 ROM からの命令フェッチ用のパスと，内蔵 RAM へのメモリ・アクセス用のパスが独立して動作します。これにより，IF ステージと MEM ステージのバス・アービトレーションの競合が発生せず，パイプラインがスムーズに流れます。

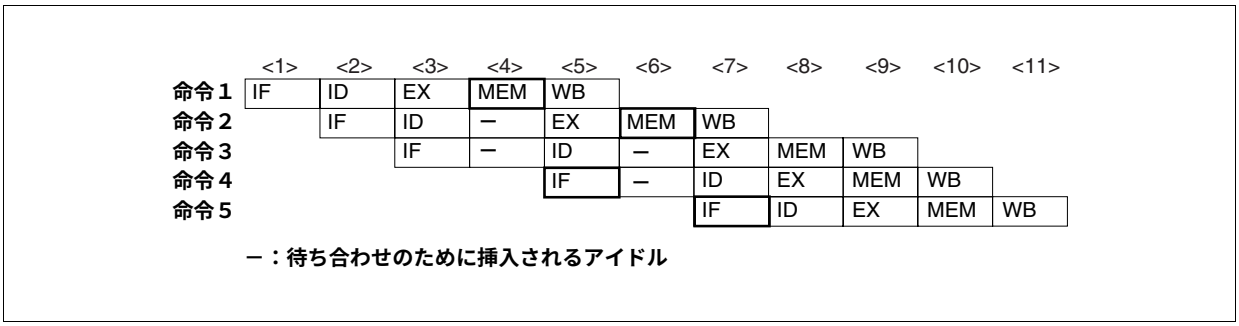
(1) V850E1 CPU（ハーバード・アーキテクチャ）の場合

命令 1 の MEM と命令 4 の IF，および命令 2 の MEM と命令 5 の IF が同時に実行でき，パイプラインが乱れません。



(2) 非ハーバード・アーキテクチャの場合

命令 1 の MEM と命令 4 の IF，および命令 2 の MEM と命令 5 の IF が競合するためバスの待ち合わせが発生し，パイプラインが乱れ実行速度が低下します。



ー：待ち合わせのために挿入されるアイドル

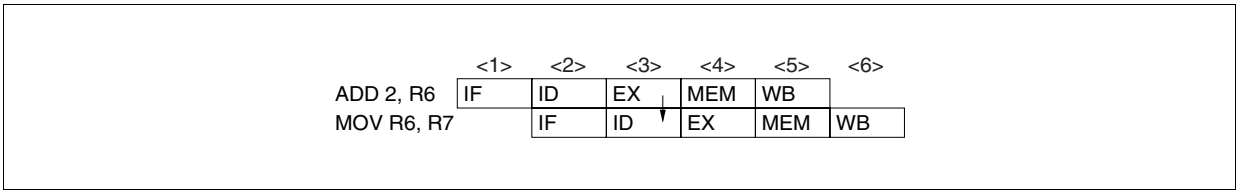
8.4.2　ショート・パス

V850E1 CPU はショート・パスを内蔵しているため，前命令のライトバック（WB）が終了する前に，後続の命令がその結果を使用できます。

例 1．算術演算命令，論理演算の実行結果を直後の命令で使用する場合

• V850E1 CPU（ショート・パス内蔵）の場合

前命令の WB **を待たず実行結果が出た時点（**EX **ステージ）で，直後の命令の** ID **を処理できます。**



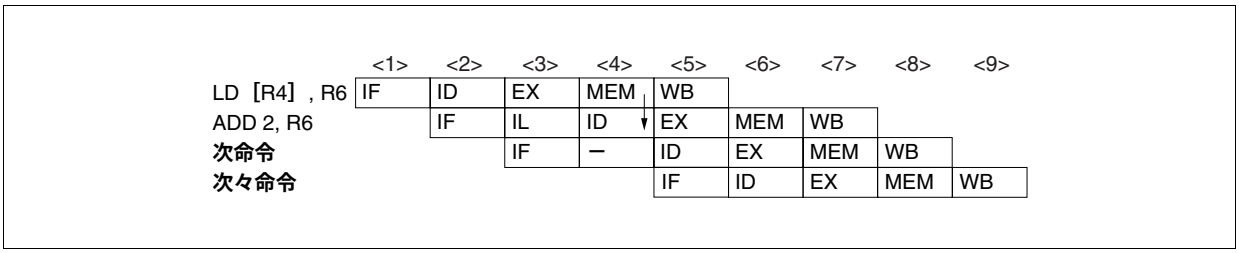
• ショート・パスがない場合

前命令の WB **まで，直後の** ID **が遅れます。**

例 2．ロード命令によりメモリから読み出したデータを，直後の命令で使用する場合

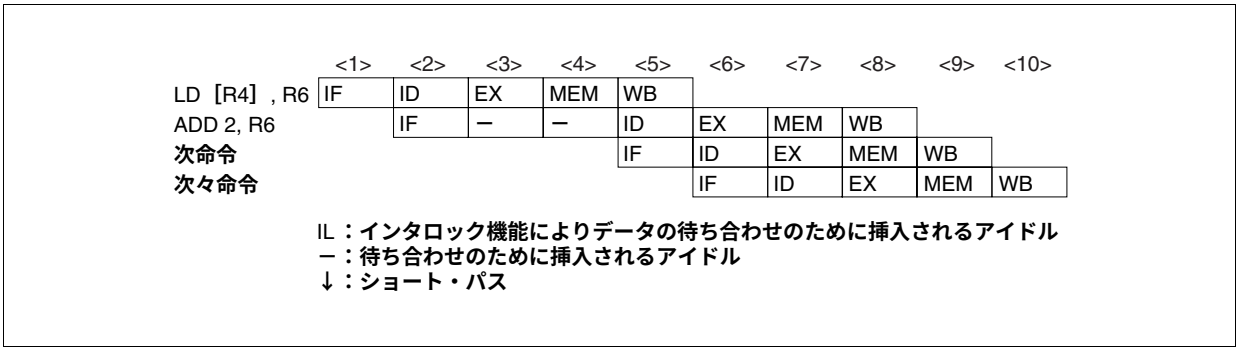
• V850E1 CPU（ショート・パス内蔵）の場合

前命令の WB **を待たず実行結果が出た時点（**MEM **ステージ）で，直後の命令の** ID **を処理できます。**



• ショート・パスがない場合

前命令の WB **まで，直後の** ID **が遅れます。**



★

第9章　ディバグ・モードへの移行

V850E1 CPUは，ディバグ・トラップ，例外トラップ，ディバグ・ブレークが発生するとプログラム・カウンタ（PC）にハンドラ・アドレス（00000060H）をセットし，ディバグ・モードへ移行します。
また，シングルステップ動作の設定を行うと各命令の実行ごとにディバグ・モードに移行できます。

注意　ディバグ・モードに移行すると，データ・キャッシュがホールドされ，データやタグの更新は行われません。ディバグ・モード中にキャッシュ許可領域の外部メモリにアクセスすると，データ・キャッシュが有効でも外部メモリだけにアクセスするため，コヒーレンシ性が崩れてしまいます。したがって，ディバグ・モニタ・ルーチン中でキャッシュ可能領域のデータを操作する場合は，ユーザ・モードに戻る前に，データ・キャッシュをクリア（ライトスルーの場合），またはフラッシュ，クリア（ライトバックの場合）してください。

9. 1　ディバグ・モードへの移行方法

(1) ディバグ・トラップ

DBTRAP命令の実行によりディバグ・トラップが発生し，ディバグ・モードに移行します（6. 2. 3　**ディバグ・トラップ**参照）。

(2) 例外トラップ

命令の不正実行により例外トラップが発生し，ディバグ・モードに移行します（6. 2. 2　**例外トラップ**参照）。

(3) ディバグ・ブレーク

ディバグ・ブレークには，次の3種類があります。

- ブレークポイント設定によるブレーク（2チャネル）
- ミス・アライン・アクセス例外発生によるブレーク
- アラインメント・エラー例外発生によるブレーク

ディバグ・ブレークの設定は，次に示すシステム・レジスタにより行います。

- ディバグ・インタフェース・レジスタ（DIR）
- ブレークポイント制御レジスタ0, 1（BPC0, BPC1）
- ブレークポイント・アドレス設定レジスタ0, 1（BPAV0, BPAV1）
- ブレークポイント・アドレス・マスク・レジスタ0, 1（BPAM0, BPAM1）
- ブレークポイント・データ設定レジスタ0, 1（BPDV0, BPDV1）
- ブレークポイント・データ・マスク・レジスタ0, 1（BPDM0, BPDM1）

備考　ASIDレジスタを除く各レジスタは，ディバグ・モードでのみリード／ライト可能です（DIRレジスタはユーザ・モードでもリード可能）。したがって，各レジスタの初期設定や任意のタイミングでのリード／ライトは，ディバグ・トラップ（DBTRAP命令の実行）によってディバグ・モードに移行してから行ってください。

(a) ブレーク・ポイント設定によるブレーク（2チャネル）

次に示すブレーク条件の成立によるブレーク・ポイント設定（2チャネル）に基づいて，ディバグ・モードに移行します。各条件の設定は，BPCnレジスタで行います（n = 0, 1）。

注意　BPCnレジスタのIEビットをセット（1）している場合は，BP ASIDビットとASIDレジスタに設定したプログラムIDが一致しないと，ブレーク条件が成立してもディバグ・モードには移行しません。

表9－1　ブレーク条件

種類	ブレーク条件		ブレーク・タイミング	BPxxnレジスタの設定[注2]				BPCnレジスタのMD, FE, RE, WEビットの設定		
	アドレス[注1]	データ		BP AVn	BP AMn	BP DVn	BP DMn	MD	FE	RE, WE
実行系トラップ	任意の実行アドレス	特定の命令コード	実行直前	<1>	<1>	○	<0>	0	1	0[注5]
		特定の命令コード範囲		<1>	<1>	○	○			
	特定の実行アドレス	任意の命令コード		○	<0>	<1>	<1>	任意		
		特定の命令コード		○	<0>	○	<0>	0		
		特定の命令コード範囲		○	<0>	○	○			
	特定の実行アドレス範囲	任意の命令コード		○	○	<1>	<1>	任意		
		特定の命令コード		○	○	○	<0>	0		
		特定の命令コード範囲		○	○	○	○			
アクセス系トラップ	任意のアクセス・アドレス	特定のデータ	実行後[注3]	<1>	<1>	○	<0>	0	0	0/1[注6]
		特定のデータ範囲	実行直後	<1>	<1>	○	○			
	特定のアクセス・アドレス	任意のデータ	実行後[注3]	○	<0>	<1>	<1>	任意[注4]		
		特定のデータ		○	<0>	○	<0>	0		
		特定のデータ範囲		○	<0>	○	○			
	特定のアクセス・アドレス範囲	任意のデータ	実行直後	○	○	<1>	<1>	任意[注4]		
		特定のデータ	実行後[注3]	○	○	○	<0>	0		
		特定のデータ範囲		○	○	○	○			

注1. 実行アドレスとは，命令フェッチ時のアドレスを，アクセス・アドレスとは命令の実行に伴いアクセスが発生するアドレスを意味します。

2. 次のように設定してください。

○　：ブレーク条件を設定してください。

<0>：すべてのビットをクリア（0）してください。

<1>：条件設定の必要はありませんが，初期値が不定のため，すべてのビットをセット（1）してください（BPAVn, BPAMnレジスタのビット31-28は0固定のため，セット（1）できません）。

ただし，実行系トラップの場合，または，アクセス系トラップで64 Mバイトのデータ領域を対象とする場合，BPAVn, BPAMnレジスタのビット27, 26は無視されますが，初期値が不定のため，セット（1）してください。

3. データ・ライト時：実行直後

データ・リード時：数命令実行後（スリップ）

4. MDビットをセット（1）すると，データ・コンパレータによる一致判定が無視されるため，ブレークのレイテンシが1クロック速くなります（MD = 0の場合はMEMステージで，MD = 1の場合はEXステージでブレーク）。

5. 必ず0を設定してください（1を設定した場合の動作は保証しません）。

6. アクセスの種類（リードのみ，ライトのみ，リード／ライト両方）に応じて設定してください。

注意1. **実行系トラップとアクセス系トラップでは，ブレーク条件が一致するタイミングが異なります（実行系トラップは，IDステージ，アクセス系トラップは，MEMステージ）。したがって，シーケンシャル・ブレーク・モードに設定しても，アクセス系トラップのあとに実行系トラップが発生する場合は，正常に動作しないことがあります。**

2. **レインジ・ブレーク・モードでは，実行系トラップとアクセス系トラップのどちらか一方をチャネル0, 1に設定してください。**

備考1. n = 0, 1

2. **複数のブレーク条件が設定されている場合，条件のうちの1つでも成立すればディバグ・モードに移行します。**
3. **チャネル**0, 1**を連動させて，次の**2**種類の動作を行うことも可能です（同時に動作させることは不可）。**

（ｉ）シーケンシャル実行によるブレーク（シーケンシャル・ブレーク・モード）

ディバグ・インタフェース・レジスタ（DIR**）の**SQ**ビットをセット（**1**）することにより，設定されます。チャネル**0→1**の順にブレーク条件が一致した場合だけ，ディバグ・モードに移行します。**

（ⅱ）同時実行によるブレーク（レインジ・ブレーク・モード）

ディバグ・インタフェース・レジスタ（DIR**）の**RE**ビットをセット（**1**）することにより，設定されます。チャネル**0, 1**のブレーク条件が同時に一致した場合だけ，ディバグ・モードに移行します。**

(b**) ミス・アライン・アクセス例外発生によるブレーク**

ディバグ・インタフェース・レジスタ（DIR**）の**MA**ビットをセット（**1**）することにより，設定されます。**

ロード命令，ストア命令実行時にミス・アライン・アクセスが発生するとディバグ・モードに移行します（CPU**に対するミス・アライン・アクセス許可／禁止の設定には依存しません）。**

(c**) アラインメント・エラー例外発生によるブレーク**

ディバグ・インタフェース・レジスタ（DIR**）の**AE**ビットをセット（**1**）することにより，設定されます。**

アラインメント・エラーが発生するとディバグ・モードに移行します。

アラインメント・エラーが発生するのは，次の場合です。

- PREPARE, DISPOSE **命令実行時にスタック・ポインタ（**SP**）がワード境界以外に強制的にアラインされたとき**

備考　**CPU**に対するミス・アライン・アクセス許可／禁止の設定は，ハードウエア設定（端子入力）により行います（**V850E1**コアでは，**IFIMAEN**端子への入力レベルにより，設定を行います）。**

アクセス系トラップの場合を除くディバグ・ブレークでは，ブレークの発生要因となった命令のアドレスがDBPC**に退避されます（命令実行完了前にディバグ・モードに移行するため）。したがって，ディバグ・モードからユーザ・モードへの移行後に，ブレークの発生要因となった命令が実行されますが，再度のディバグ・ブレークは発生しません（無視されます）。**

(4) シングルステップ動作

PSWのSSフラグをセット（1）すると，シングルステップ動作が設定され，各命令の実行ごとにディバグ・モードに移行します。シングルステップ動作は，次の手順で設定／解除されます。

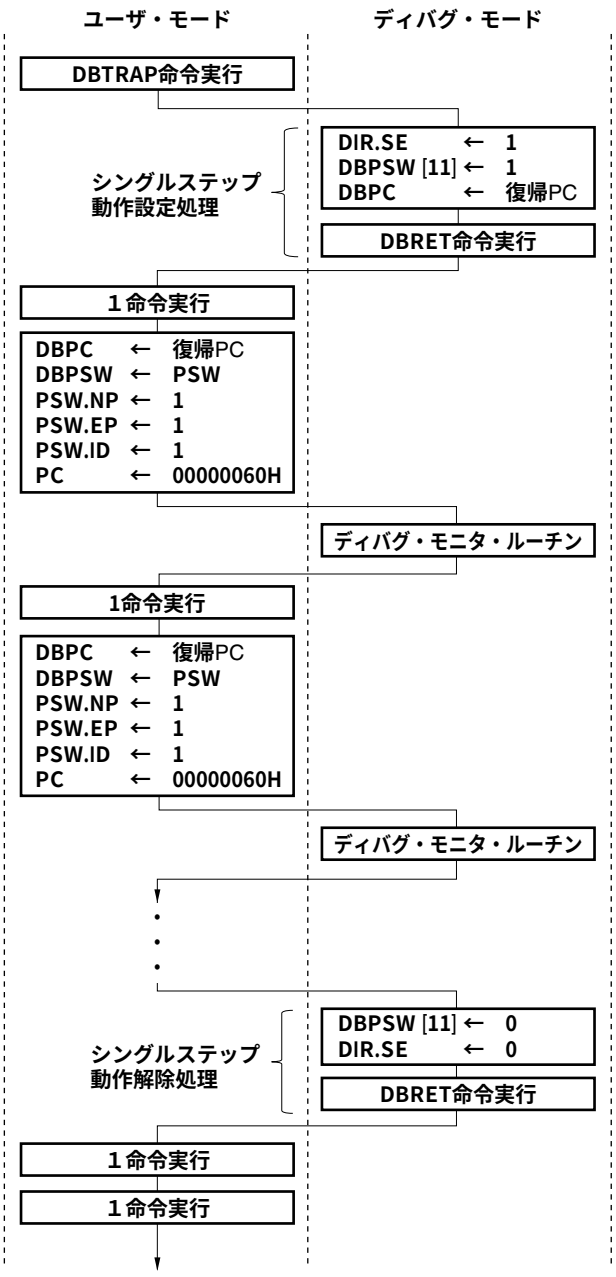
(a) シングルステップ動作の設定手順

① ディバグ・トラップ（DBTRAP命令の実行）により，ディバグ・モードに移行する。
② PSWのSSフラグを制御するために，DIRレジスタのSEビットをセット（1）する。
③ ユーザ・モードへの移行時にPSWのSSフラグをセット（1）するためにDBPSWレジスタのビット11をセット（1）する。
④ DBPCレジスタに復帰PC値を転送する。
⑤ DBRET命令によりユーザ・モードへ移行する（移行時にPSWのSSフラグがセット（1）され，シングルステップ動作が設定される）。

(b) シングルステップ動作の解除手順

① ディバグ・モードでの動作時に，DBPSWレジスタのビット11をクリア（0）する（この操作により，ユーザ・モードへの移行時にPSWのSSフラグがクリア（0）される）。
② DIRレジスタのSEビットをクリア（0）する（ただし，この操作を省略すると，ユーザ・モードでPSWのSSフラグのセット（1）が可能）。
③ DBRET命令によりユーザ・モードへ移行する（移行時にPSWのSSフラグがクリア（0）され，シングルステップ動作が解除される）。

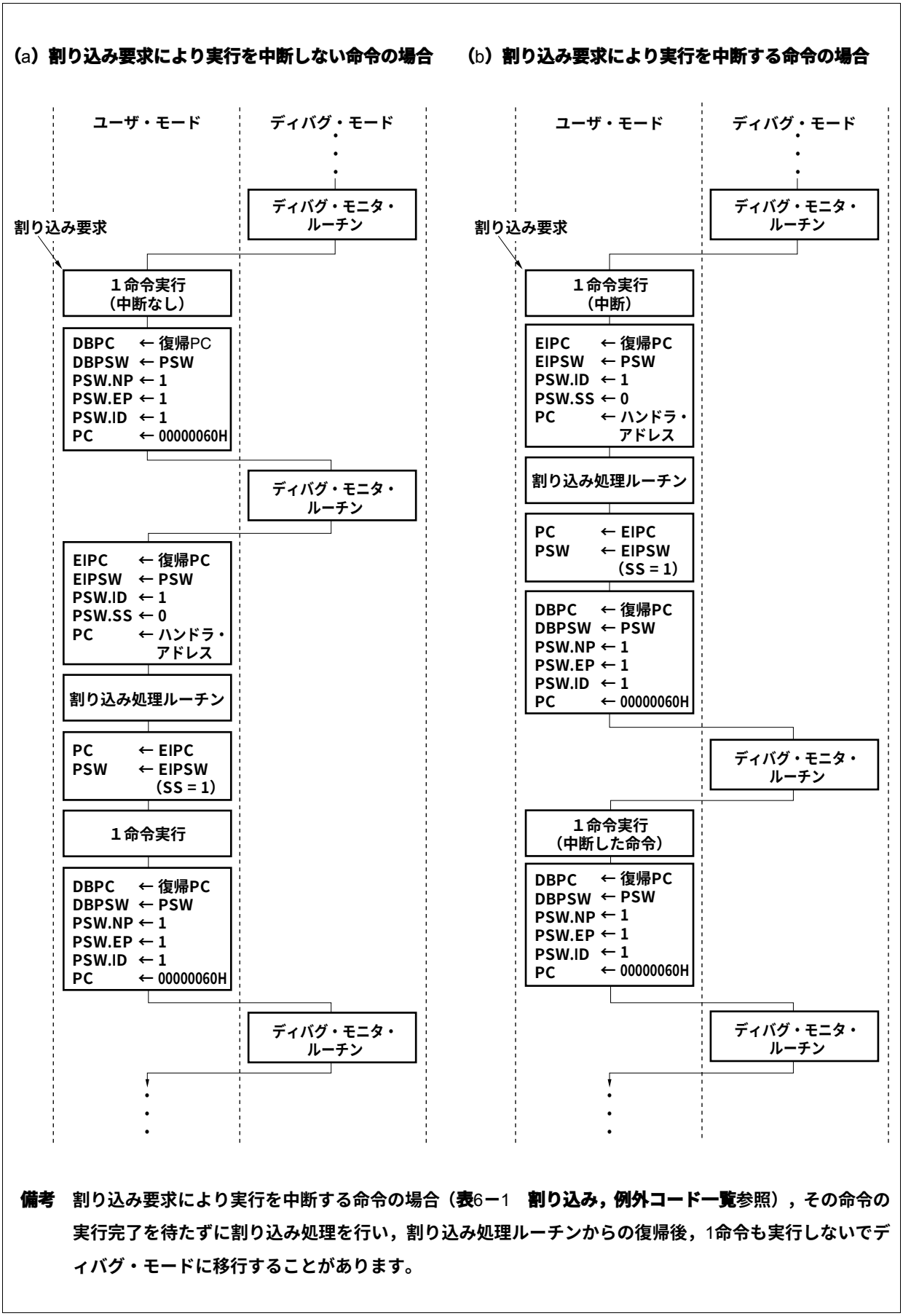
図9－1　シングルステップ動作の実行フロー



備考　シングルステップ動作時，ユーザ・モードで割り込み要求が発生すると，PSWのSSフラグは自動的にクリア（0）されるため，割り込み処理ルーチンでのシングルステップ動作は行われません（SSフラグは，割り込み処理ルーチンからの復帰処理（EIPSW→PSW）により，再びセット（1）されます）。

また，割り込み発生時に実行されている命令により，処理フローが異なることがあります（**図**9－2参照）。

図9－2　シングルステップ動作時に割り込み要求が発生した場合の処理フロー



備考　割り込み要求により実行を中断する命令の場合（**表6－1　割り込み，例外コード一覧**参照），その命令の実行完了を待たずに割り込み処理を行い，割り込み処理ルーチンからの復帰後，1命令も実行しないでディバグ・モードに移行することがあります。

9.2　注意事項

ミス・アライン・アクセス時とビット操作命令によるアクセス時は，アクセスするアドレスにより，BPDVnレジスタへの設定値が異なります（n = 0, 1）。

ミス・アライン・アクセスでは，メモリ・アクセス・サイクルが複数回に分割されて発生しますが，ライト・アクセスの場合，分割されたうちの最初のサイクルのアドレス，データ，およびアクセスの種類（ハーフワード／バイト）だけがブレーク条件の比較対象になります。また，ビット操作命令によるアクセスの場合も，アクセスするアドレスによって，BPDVnレジスタへの設定値が異なります。

各アクセス・サイズに対するアクセス・アドレスごとのブレーク条件の設定例を次に示します。

表9－2　ブレーク条件設定例

アクセス・サイズ（かっこ内はデータの例）	アクセス・アドレス[注1]	バス・サイクル	BPCnレジスタのTYビット		BPAVnレジスタ[注1]	BPDVnレジスタ[注2]	
			ライト時	リード時		ライト時	リード時
ワード（44332211H）	0H	W	1, 1（W）	1, 1（W）	0H	44332211H	44332211H
	1H	B→HW→B	0, 1（B）		1H	xxxx11xxH	
	2H	HW→HW	1, 0（HW）		2H	2211xxxxH	
	3H	B→HW→B	0, 1（B）		3H	11xxxxxxH	
ハーフワード（2211H）	0H	HW	1, 0（HW）	1, 0（HW）	0H	xxxx2211H	xxxx2211H
	1H	B→B	0, 1（B）		1H	xxxx11xxH	
	2H	HW	1, 0（HW）		2H	2211xxxxH	
						xxxx2211H[注3]	
	3H	B→B	0, 1（B）		3H	11xxxxxxH	
バイト（11H）	0H	B	0, 1（B）		0H	xxxxxx11H	xxxxxx11H
	1H				1H	xxxx11xxH	
						xxxxxx11H[注4]	
	2H				2H	xx11xxxxH	
						xxxxxx11H[注4]	
	3H				3H	11xxxxxxH	
						xxxxxx11H[注4]	
ビット（11H）	0H	B	0, 1（B）		0H	xxxxxx11H	
	1H				1H	xxxx11xxH	
	2H				2H	xx11xxxxH	
	3H				3H	11xxxxxxH	

注1.　下位2ビットの値を示します。

2.　「x」は，BPDMnレジスタによりマスクされていることを示します。

3.　ハーフワード・アライン・アクセス時のみ有効です。

4.　バイト・アライン・アクセス時のみ有効です。

備考1.　W　　：ワード・データ転送サイクル

HW　：ハーフワード・データ転送サイクル

B　　：バイト・データ転送サイクル

2.　n = 0, 1

たとえば，ワード・データ「44332211H」の「03FFEFF1H」番地へのライト・アクセスの場合，最初のメモリ・アクセスは03FFEFF1H番地へのバイト・データ「11H」のライトとなります。したがって，このアクセスをチャネル0のブレーク条件として指定する場合の設定例は次のようになります。

- BPAV0レジスタ　　：03FFEFF1H
- BPAM0レジスタ　　：00000000H
- BPDV0レジスタ　　：xxxx11xxH（x：任意）
- BPDM0レジスタ　　：FFFF00FFH
- BPC0レジスタのTYビット：0, 1（バイト・アクセス）

★

付録A　注意事項

A. 1　sld命令と割り込み競合に関する制限事項

A. 1. 1　内　　容

次の命令<1>の事項が完了する前に，後続のsld命令の直前の命令<2>のデコード動作と割り込み要求が競合した場合，先の命令<1>の実行結果がレジスタに格納されないことがあります。

命令<1>

- ld命令　　：ld.b, ld.h, ld.w, ld.bu, ld.hu
- sld命令　　：sld.b, sld.h, sld.w, sld.bu, sld.hu
- 乗算命令　　：mul, mulh, mulhi, mulu

命令<2>

mov　reg1, reg2	not　reg1, reg2	satsubr　reg1, reg2	satsub　reg1, reg2
satadd　reg1, reg2	satadd　imm5, reg2	or　reg1, reg2	xor　reg1, reg2
and　reg1, reg2	tst　reg1, reg2	subr　reg1, reg2	sub　reg1, reg2
add　reg1, reg2	add　imm5, reg2	cmp　reg1, reg2	cmp　imm5, reg2
mulh　reg1, reg2	shr　imm5, reg2	sar　imm5, reg2	shl　imm5, reg2

<例>

< i >　ld.w　[r11], r10
　　・
　　・
　　・
< ii >　mov　r10, r28
<iii>　sld.w　0x28, r10

< i >のld命令の実行が完了する前に，<iii>のsld命令の直前のmov命令< ii >のデコード動作と割り込み要求が競合した場合，< i >のld命令の実行結果がレジスタに格納されないことがあります。

A. 1. 2　回 避 策

命令< ii >の直後にsld命令を実行する場合は，次のいずれかの方法を用いて，上記動作を回避してください。

- sld命令の直前にnop命令を入れる。
- sld命令のディスティネーション・レジスタと同じレジスタを，sld命令の直前で実行する上記< ii >の命令で使用しない。

付録B　命令一覧

アルファベット順の命令機能一覧を表 B－1 に，フォーマット順の命令一覧を表 B－2 に示します。

表B－1　命令機能一覧（アルファベット順）（1/12）

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。 reg2 のワード・データに reg1 のワード・データを加算し，その結果を reg2 に格納します。
ADD	imm5, reg2	II	0/1	0/1	0/1	0/1	－	加算。 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 reg2 に格納します。
ADDI	imm16, reg1, reg2	VI	0/1	0/1	0/1	0/1	－	加算。 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 reg2 に格納します。
AND	reg1, reg2	I	－	0	0/1	0/1	－	論理積。 reg2 のワード・データと reg1 のワード・データの論理積をとり，その結果を reg2 に格納します。
ANDI	imm16, reg1, reg2	VI	－	0	0	0/1	－	論理積。 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・イミーディエトの論理積をとり，その結果を reg2 に格納します。
Bcond	disp9	III	－	－	－	－	－	条件分岐（if Carry）。 命令が指定する PSW のフラグをテストし，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。
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。 エンディアン変換を行います。
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。 エンディアン変換を行います。
CALLT	imm6	II	－	－	－	－	－	テーブル参照によるサブルーチン・コール。 CTBP の内容に基づき，PC の値を変更し，制御を移します。

表B－1　命令機能一覧（アルファベット順）（2/12）

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	
CLR1	bit#3, disp16 [reg1]	VIII	－	－	－	0/1	－	ビット・クリア。 まず，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。生成したアドレスのバイト・データの 3 ビットのビット・ナンバで示されるビットをクリアします。
CLR1	reg2 [reg1]	IX	－	－	－	0/1	－	ビット・クリア。 まず，reg1 の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 reg2 の下位 3 ビットで示されるビットをクリアします。
CMOV	cccc, reg1, reg2, reg3	XI	－	－	－	－	－	条件付き転送。 条件コード「cccc」で指定された条件が，満たされた場合は reg1 のデータを，満たされなかった場合は reg2 のデータを，reg3 に転送します。
CMOV	cccc, imm5, reg2, reg3	XII	－	－	－	－	－	条件付き転送。 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 reg2 のデータを，reg3 に転送します。
CMP	reg1, reg2	I	0/1	0/1	0/1	0/1	－	比較。 reg2 のワード・データと 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 reg2 のワード・データから reg1 の内容を減算することで行います。
CMP	imm5, reg2	II	0/1	0/1	0/1	0/1	－	比較。 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 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。
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。 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。
DBRET 注	（なし）	X	0/1	0/1	0/1	0/1	0/1	ディバグ・トラップからの復帰。 システム・レジスタから復帰 PC と PSW を取り出し，ディバグ・モニタ・ルーチンから復帰します。
DBTRAP 注	（なし）	I	－	－	－	－	－	ディバグ・トラップ。 復帰 PC,PSW をシステム・レジスタに退避し，PC にハンドラ・アドレス（00000060H）をセットして制御を移します。

★　注　タイプ C の製品ではサポートしていません。

表B－1　命令機能一覧（アルファベット順）（3/12）

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	
DI	（なし）	X	－	－	－	－	－	マスカブル割り込みの禁止。 PSW 中の ID フラグをセット（1）し，この命令実行中からマスカブル割り込みの受け付けを禁止します。
DISPOSE	imm5, list12	XIII	－	－	－	－	－	スタック・フレームの削除。 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ものを，sp に加算します。そして，list12 に示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。
DISPOSE	imm5, list12, [reg1]	XIII	－	－	－	－	－	スタック・フレームの削除。 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ものを，sp に加算します。そして，list12 に示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，reg1 で指定されたアドレスに制御を移します。
DIV	reg1, reg2, reg3	XI	－	0/1	0/1	0/1	－	符号付きワード・データの除算。 reg2 のワード・データを reg1 のワード・データで除算し，その商を reg2 に，余りを reg3 に格納します。
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。 reg2 のワード・データを reg1 の下位ハーフワード・データで除算し，その商を reg2 に格納します。
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。 reg2 のワード・データを reg1 の下位ハーフワード・データで除算し，その商を reg2 に，余りを reg3 に格納します。
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。 reg2 のワード・データを reg1 の下位ハーフワード・データで除算し，その商を reg2 に，余りを reg3 に格納します。
DIVU	reg1, reg2, reg3	XI	－	0/1	0/1	0/1	－	符号なしワード・データの除算。 reg2 のワード・データを reg1 のワード・データで除算し，その商を reg2 に，余りを reg3 に格納します。
EI	（なし）	X	－	－	－	－	－	マスカブル割り込みの許可。 PSW 中の ID フラグをクリア（0）し，次の命令からマスカブル割り込みの受け付けを許可します。
HALT	（なし）	X	－	－	－	－	－	CPU 停止。 CPU の動作クロックを停止させ，HALT 状態に入ります。
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。 エンディアン交換を行います。

表B－1　命令機能一覧（アルファベット順）（4/12）

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	
JARL	disp22, reg2	V	－	－	－	－	－	分岐とレジスタ・リンク。 現在の PC に 4 を加算した値を reg2 に退避し，現在の PC に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，その PC に制御を移します。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。
JMP	[reg1]	I	－	－	－	－	－	レジスタ間接無条件分岐。 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。
JR	disp22	V	－	－	－	－	－	無条件分岐。 現在の PC に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，その PC に制御を移します。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。
LD.B	disp16 [reg1] , reg2	VII	－	－	－	－	－	バイト・ロード。 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。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。
LD.BU	disp16 [reg1] , reg2	VII	－	－	－	－	－	符号なしバイト・ロード。 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。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。
LD.H	disp16 [reg1], reg2	VII	－	－	－	－	－	ハーフワード・ロード。 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。この，32 ビット・アドレスのビット 0 を 0 にマスク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，reg2 に格納します。
LD.HU	disp16 [reg1] , reg2	VII	－	－	－	－	－	符号なしハーフワード・ロード。 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。この，32 ビット・アドレスのビット 0 を 0 にマスク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，reg2 に格納します。

表B－1　命令機能一覧（アルファベット順）（5/12）

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	
LD.W	disp16 [reg1] , reg2	VII	－	－	－	－	－	ワード・ロード。 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。この，32 ビット・アドレスのビット 0 とビット 1 を 0 にマスクしたアドレスからワード・データを読み出し，reg2 に格納します。
LDSR	reg2, regID	IX	－	－	－	－	－	システム・レジスタへのロード。 reg2 のワード・データを regID で指定されるシステム・レジスタに設定します。regID が PSW の場合は，PSW のフラグには reg2 の対応するビットの値が設定されます。
MOV	reg1, reg2	I	－	－	－	－	－	データの転送。 reg1 のワード・データを reg2 にコピーし，転送します。
MOV	imm5, reg2	II	－	－	－	－	－	データの転送。 ワード長まで符号拡張した 5 ビット・イミーディエトを reg2 にコピーし，転送します。
MOV	imm32, reg1	VI	－	－	－	－	－	データの転送。 32 ビット・イミーディエトを reg1 にコピーし，転送します。
MOVEA	imm16, reg1, reg2	VI	－	－	－	－	－	実行アドレスの転送。 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 reg2 に格納します。
MOVHI	imm16, reg1, reg2	VI	－	－	－	－	－	上位ハーフワードの転送。 reg1 のワード・データに上位 16 ビット（16 ビット・イミーディエト）と下位 16 ビット（0）を合わせたワード･データを加算し,その結果を reg2 に格納します。
MUL	reg1, reg2, reg3	XI	－	－	－	－	－	符号付きワード・データの乗算。 reg2 のワード・データに reg1 のワード・データを乗算し，その結果を reg2 と reg3 に格納します。
MUL	imm9, reg2, reg3	XII	－	－	－	－	－	符号付きワード・データの乗算。 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，その結果を reg2 と reg3 に格納します。
MULH	reg1, reg2	I	－	－	－	－	－	符号付きハーフワード・データの乗算。 reg2 の下位ハーフワード・データに reg1 の下位ハーフワード・データを乗算し，その結果を reg2 にワード・データとして格納します。

表B－1　命令機能一覧（アルファベット順）（6/12）

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	
MULH	imm5, reg2	II	－	－	－	－	－	符号付きハーフワード・データの乗算。 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 reg2 にワード・データとして格納します。
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。 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 reg2 に格納します。
MULU	reg1, reg2, reg3	XI	－	－	－	－	－	符号なしワード・データの乗算。 reg2 のワード・データに reg1 のワード・データを乗算し，その結果を reg2 と reg3 に格納します。
MULU	imm9, reg2, reg3	XII	－	－	－	－	－	符号なしワード・データの乗算。 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，その結果を reg2 と reg3 に格納します。
NOP	（なし）	I	－	－	－	－	－	ノー・オペレーション。
NOT	reg1, reg2	I	－	0	0/1	0/1	－	論理否定。 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 reg2 に格納します。
NOT1	bit#3, disp16 [reg1]	VIII	－	－	－	0/1	－	ビット・ノット。 まず，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。生成したアドレスのバイト・データの 3 ビットのビット・ナンバで示されるビットを反転します。
NOT1	reg2, [reg1]	IX	－	－	－	0/1	－	ビット・ノット。 まず，reg1 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 reg2 の下位 3 ビットで示されるビットを反転します。
OR	reg1, reg2	I	－	0	0/1	0/1	－	論理和。 reg2 のワード・データと reg1 のワード・データの論理和をとり，その結果を reg2 に格納します。
ORI	imm16, reg1, reg2	VI	－	0	0/1	0/1	－	論理和。 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・イミーディエトの論理和をとり，その結果を reg2 に格納します。

表B－1　命令機能一覧（アルファベット順）（7/12）

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。 list12 に表示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。
PREPARE	list12, imm5, sp/imm	XIII	－	－	－	－	－	スタック・フレームの生成。 list12 に表示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 オペランドで指定されるデータを ep にロードします。
RETI	（なし）	X	0/1	0/1	0/1	0/1	0/1	割り込みまたは例外処理ルーチンから復帰。 システム・レジスタから復帰 PC と PSW を取り出し，割り込みまたは例外処理ルーチンから復帰する命令です。
SAR	reg1,reg2	IX	0/1	0	0/1	0/1	－	算術右シフト。 reg2 のワード・データを reg1 の下位 5 ビットで示されるシフト数分，算術右シフト（シフト以前の MSB の値を順に MSB にコピーする）し，reg2 に書き込みます。
SAR	imm5, reg2	II	0/1	0	0/1	0/1	－	算術右シフト。 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，算術右シフト（シフト以前の MSB の値を順に MSB にコピーする）し，reg2 に書き込みます。
SASF	cccc, reg2	IX	－	－	－	－	－	シフトとフラグ条件の設定。 条件コード「cccc」で指定された条件が満たされた場合は，reg2 のデータを 1 ビット論理左シフトし，LSB に 1 がセットされます。満たされなかった場合は，reg2 のデータを 1 ビット論理左シフトし，LSB に 0 がセットされます。
SATADD	reg1, reg2	I	0/1	0/1	0/1	0/1	0/1	飽和加算。 reg2 のワード・データに reg1 のワード・データを加算し，その結果を reg2 に格納します。ただし，その結果が正の最大値を越えたときは正の最大値を，負の最大値を越えたときは負の最大値を reg2 に格納し，SAT フラグをセット（1）します。

表B－1　命令機能一覧（アルファベット順）（8/12）

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	
SATADD	imm5,reg2	II	0/1	0/1	0/1	0/1	0/1	飽和加算。 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 reg2 に格納します。ただし，その結果が正の最大値を越えたときは正の最大値を，負の最大値を越えたときは負の最大値を reg2 に格納し，SAT フラグをセット（1）します。
SATSUB	reg1, reg2	I	0/1	0/1	0/1	0/1	0/1	飽和減算。 reg2 のワード・データから reg1 のワード・データを減算し，その結果を reg2 に格納します。ただし，その結果が正の最大値を越えたときは正の最大値を，負の最大値を越えたときは負の最大値を reg2 に格納し，SAT フラグをセット（1）します。
SATSUBI	imm16, reg1, reg2	VI	0/1	0/1	0/1	0/1	0/1	飽和減算。 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 reg2 に格納します。ただし，その結果が正の最大値を越えたときは正の最大値を，負の最大値を越えたときは負の最大値を reg2 に格納し，SAT フラグをセット（1）します。
SATSUBR	reg1, reg2	I	0/1	0/1	0/1	0/1	0/1	飽和逆減算。 reg1 のワード・データから reg2 のワード・データを減算し，その結果を reg2 に格納します。ただし，その結果が正の最大値を越えたときは正の最大値を，負の最大値を越えたときは負の最大値を reg2 に格納し，SAT フラグをセット（1）します。
SET1	bit#3, disp16 [reg1]	VIII	－	－	－	0/1	－	ビット・セット。 まず，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。生成したアドレスのバイト・データの 3 ビットのビット・ナンバで示されるビットをセット（1）します。
SET1	reg2, [reg1]	IX	－	－	－	0/1	－	ビット・セット。 まず，reg1 の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 reg2 の下位 3 ビットで示されるビットをセット（1）します。
SETF	cccc, reg2	IX	－	－	－	－	－	フラグ条件の設定。 条件コード cccc の示す条件が，対象となる PSW のフラグと一致した場合には，reg2 に 1 を，そうでない場合には 0 を格納します。

表B－1　命令機能一覧（アルファベット順）（9/12）

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	
SHL	reg1, reg2	IX	0/1	0	0/1	0/1	－	論理左シフト。 reg2 のワード・データを reg1 の下位 5 ビットで示されるシフト数分，論理左シフト（LSB 側に 0 を送り込む）し，reg2 に書き込みます。
SHL	imm5, reg2	II	0/1	0	0/1	0/1	－	論理左シフト。 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，論理左シフト（LSB 側に 0 を送り込む）し，reg2 に書き込みます。
SHR	reg1, reg2	IX	0/1	0	0/1	0/1	－	論理右シフト。 reg2 のワード・データを reg1 の下位 5 ビットで示されるシフト数分，論理右シフト（MSB 側に 0 を送り込む）し，reg2 に書き込みます。
SHR	imm5, reg2	II	0/1	0	0/1	0/1	－	論理右シフト。 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，論理右シフト（MSB 側に 0 を送り込む）し，reg2 に書き込みます。
SLD.B	disp7 [ep], reg2	IV	－	－	－	－	－	バイト・ロード。 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。
SLD.BU	disp4 [ep], reg2	IV	－	－	－	－	－	符号なしバイト・ロード。 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。
SLD.H	disp8 [ep], reg2	IV	－	－	－	－	－	ハーフワード・ロード。 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。この，32 ビット・アドレスのビット 0 を 0 にマスク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，reg2 に格納します。

表B－1　命令機能一覧（アルファベット順）（10/12）

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	
SLD.HU	disp5 [ep], reg2	IV	－	－	－	－	－	符号なしハーフワード・ロード。 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。この，32ビット・アドレスのビット0を0にマスク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，reg2に格納します。
SLD.W	disp8 [ep], reg2	IV	－	－	－	－	－	ワード・ロード。 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。この，32ビット・アドレスのビット0とビット1を0にマスクしたアドレスからワード・データを読み出し，reg2に格納します。
SST.B	reg2, disp7 [ep]	IV	－	－	－	－	－	バイト・ストア。 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。
SST.H	reg2,disp8 [ep]	IV	－	－	－	－	－	ハーフワード・ストア。 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・アドレスのビット0を0にマスクしたアドレスに格納します。
SST.W	reg2, disp8 [ep]	IV	－	－	－	－	－	ワード・ストア。 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・アドレスのビット0とビット1を0にマスクしたアドレスに格納します。
ST.B	reg2, disp16 [reg1]	VII	－	－	－	－	－	バイト・ストア。 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。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。
ST.H	reg2, disp16 [reg1]	VII	－	－	－	－	－	ハーフワード・ストア。 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。reg2の下位ハーフワードのデータを生成した32ビット・アドレスのビット0を0にマスクしたアドレスに格納します。

表B－1　命令機能一覧（アルファベット順）（11/12）

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・ストア。 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。reg2 のワード・データを生成した 32 ビット・アドレスのビット 0 とビット 1 を 0 にマスクしたアドレスに格納します。
STSR	regID, reg2	IX	－	－	－	－	－	システム・レジスタの内容のストア。 regID で指定されるシステム・レジスタの内容を reg2 に設定します。
SUB	reg1, reg2	I	0/1	0/1	0/1	0/1	－	減算。 reg2 のワード・データから reg1 のワード・データを減算し，その結果を reg2 に格納します。
SUBR	reg1, reg2	I	0/1	0/1	0/1	0/1	－	逆減算。 reg1 のワード・データから reg2 のワード・データを減算し，その結果を reg2 に格納します。
SWITCH	reg1	I	－	－	－	－	－	テーブル参照分岐。 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 reg1 のデータを加算したテーブル・エントリ・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。続いて，ロードしたデータを 1 ビット論理左シフトし，ワード長まで符号拡張したあと，テーブルの先頭アドレス（SWITCH 命令の次のアドレス）を加算したターゲット・アドレスに分岐します。
SXB	reg1	I	－	－	－	－	－	バイト・データの符号拡張。 reg1 の最下位バイトをワード長に符号拡張します。
SXH	reg1	I	－	－	－	－	－	ハーフワード・データの符号拡張。 reg1 の下位ハーフワードをワード長に符号拡張します。
TRAP	vector	X	－	－	－	－	－	ソフトウエア・トラップ。 復帰 PC,PSW をシステム・レジスタに退避し，例外コードの設定，PSW のフラグの設定を行ったあと，vector で示されるトラップ・ベクタに対応するトラップ・ハンドラのアドレスに分岐し，例外処理を開始します。
TST	reg1, reg2	I	－	0	0/1	0/1	－	テスト。 reg2 のワード・データと 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，フラグのみが影響を受けます。

表B－1　命令機能一覧（アルファベット順）（12/12）

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	
TST1	bit#3, disp16 [reg1]	VIII	－	－	－	0/1	－	ビット・テスト。 まず，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。生成したアドレスのバイト・データの 3 ビットのビット・ナンバで示されるビットが 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。
TST1	reg2, [reg1]	IX	－	－	－	0/1	－	ビット・テスト。 まず，reg1 の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 reg2 の下位 3 ビットで示されるビットが 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。
XOR	reg1, reg2	I	－	0	0/1	0/1	－	排他的論理和。 reg2 のワード・データと reg1 のワード・データの排他的論理和をとり，その結果を reg2 に格納します。
XORI	imm16, reg1, reg2	VI	－	0	0/1	0/1	－	排他的論理和。 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 reg2 に格納します。
ZXB	reg1	I	－	－	－	－	－	バイト・データのゼロ拡張。 reg1 の最下位バイトをワード長にゼロ拡張します。
ZXH	reg1	I	－	－	－	－	－	ハーフワード・データのゼロ拡張。 reg1 の下位ハーフワードをワード長にゼロ拡張します。

表B－2　命令一覧（フォーマット順）（1/3）

フォーマット	オペコード		ニモニック	オペランド
	15　0	31　16		
I	0000000000000000	－	NOP	－
	rrrrr000000RRRRR	－	MOV	reg1, reg2
	rrrrr000001RRRRR	－	NOT	reg1, reg2
	rrrrr000010RRRRR	－	DIVH	reg1, reg2
	0000000010RRRRR	－	SWITCH	reg1
	0000000011RRRRR	－	JMP	[reg1]
	rrrrr000100RRRRR	－	SATSUBR	reg1, reg2
	rrrrr000101RRRRR	－	SATSUB	reg1, reg2
	rrrrr000110RRRRR	－	SATADD	reg1, reg2
	rrrrr000111RRRRR	－	MULH	reg1, reg2
	0000000100RRRRR	－	ZXB	reg1
	0000000101RRRRR	－	SXB	reg1
	0000000110RRRRR	－	ZXH	reg1
	0000000111RRRRR	－	SXH	reg1
	rrrrr001000RRRRR	－	OR	reg1, reg2
	rrrrr001001RRRRR	－	XOR	reg1, reg2
	rrrrr001010RRRRR	－	AND	reg1, reg2
	rrrrr001011RRRRR	－	TST	reg1, reg2
	rrrrr001100RRRRR	－	SUBR	reg1, reg2
	rrrrr001101RRRRR	－	SUB	reg1, reg2
	rrrrr001110RRRRR	－	ADD	reg1, reg2
	rrrrr001111RRRRR	－	CMP	reg1, reg2
	1111100001000000	－	DBTRAP 注	－
II	rrrrr010000iiiii	－	MOV	imm5, reg2
	rrrrr010001iiiii	－	SATADD	imm5, reg2
	rrrrr010010iiiii	－	ADD	imm5, reg2
	rrrrr010011iiiii	－	CMP	imm5, reg2
	0000001000iiiiii	－	CALLT	imm6
	rrrrr010100iiiii	－	SHR	imm5, reg2
	rrrrr010101iiiii	－	SAR	imm5, reg2
	rrrrr010110iiiii	－	SHL	imm5, reg2
	rrrrr010111iiiii	－	MULH	imm5, reg2
III	ddddd1011dddCCCC	－	Bcond	disp9

★　**注　タイプ C の製品ではサポートしていません。**

表B－2　命令一覧（フォーマット順）（2/3）

フォーマット	オペコード		ニモニック	オペランド
	15　0	31　16		
IV	rrrrr0000110dddd	－	SLD.BU	disp4 [ep], reg2
	rrrrr0000111dddd	－	SLD.HU	disp5 [ep], reg2
	rrrrr0110ddddddd	－	SLD.B	disp7 [ep], reg2
	rrrrr0111ddddddd	－	SST.B	reg2, disp7 [ep]
	rrrrr1000ddddddd	－	SLD.H	disp8 [ep], reg2
	rrrrr1001ddddddd	－	SST.H	reg2, disp8 [ep]
	rrrrr1010ddddddd0	－	SLD.W	disp8 [ep], reg2
	rrrrr1010ddddddd1	－	SST.W	reg2, disp8 [ep]
V	rrrrr11110ddddddd	ddddddddddddddd0	JARL	disp22, reg2
	0000011110ddddddd	ddddddddddddddd0	JR	disp22
VI	rrrrr110000RRRRR	iiiiiiiiiiiiiiii	ADDI	imm16, reg1, reg2
	rrrrr110001RRRRR	iiiiiiiiiiiiiiii	MOVEA	imm16, reg1, reg2
	rrrrr110010RRRRR	iiiiiiiiiiiiiiii	MOVHI	imm16, reg1, reg2
	rrrrr110011RRRRR	iiiiiiiiiiiiiiii	SATSUBI	imm16, reg1, reg2
	0000011000 1RRRRR	**注**	MOV	imm32, reg1
	rrrrr110100RRRRR	iiiiiiiiiiiiiiii	ORI	imm16, reg1, reg2
	rrrrr110101RRRRR	iiiiiiiiiiiiiiii	XORI	imm16, reg1, reg2
	rrrrr110110RRRRR	iiiiiiiiiiiiiiii	ANDI	imm16, reg1, reg2
	rrrrr110111RRRRR	iiiiiiiiiiiiiiii	MULHI	imm16, reg1, reg2
VII	rrrrr111000RRRRR	ddddddddddddddd	LD.B	disp16 [reg1], reg2
	rrrrr111001RRRRR	ddddddddddddddd0	LD.H	disp16 [reg1], reg2
	rrrrr111001RRRRR	ddddddddddddddd1	LD.W	disp16 [reg1], reg2
	rrrrr111010RRRRR	ddddddddddddddd	ST.B	reg2, disp16 [reg1]
	rrrrr111011RRRRR	ddddddddddddddd0	ST.H	reg2, disp16 [reg1]
	rrrrr111011RRRRR	ddddddddddddddd1	ST.W	reg2, disp16 [reg1]
	rrrrr11110bRRRRR	ddddddddddddddd1	LD.BU	disp16 [reg1], reg2
	rrrrr111111RRRRR	ddddddddddddddd1	LD.HU	disp16 [reg1], reg2
VIII	00bbb111110RRRRR	ddddddddddddddd	SET1	bit#3, disp16 [reg1]
	01bbb111110RRRRR	ddddddddddddddd	NOT1	bit#3, disp16 [reg1]
	10bbb111110RRRRR	ddddddddddddddd	CLR1	bit#3, disp16 [reg1]
	11bbb111110RRRRR	ddddddddddddddd	TST1	bit#3, disp16 [reg1]

注　32 ビット・イミーディエト・データです。上位 32 ビット（ビット 16-47）は次のようになります。

31　16	47　32
iiiiiiiiiiiiiiii	IIIIIIIIIIIIIIII

表B－2　命令一覧（フォーマット順）（3/3）

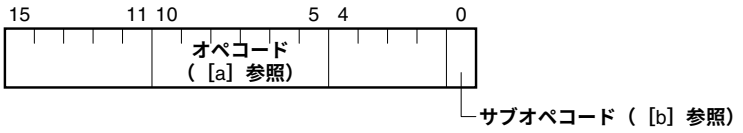
フォーマット	オペコード		ニモニック	オペランド
	15　0	31　16		
IX	rrrrr1111110cccc	0000000000000000	SETF	cccc, reg2
	rrrrr111111RRRRR	0000000000100000	LDSR	reg2, regID
	rrrrr111111RRRRR	0000000001000000	STSR	regID, reg2
	rrrrr111111RRRRR	0000000010000000	SHR	reg1, reg2
	rrrrr111111RRRRR	0000000010100000	SAR	reg1, reg2
	rrrrr111111RRRRR	0000000011000000	SHL	reg1, reg2
	rrrrr111111RRRRR	0000000011100000	SET1	reg2, [reg1]
	rrrrr111111RRRRR	0000000011100010	NOT1	reg2, [reg1]
	rrrrr111111RRRRR	0000000011100100	CLR1	reg2, [reg1]
	rrrrr111111RRRRR	0000000011100110	TST1	reg2, [reg1]
	rrrrr1111110cccc	0000001000000000	SASF	cccc, reg2
X	0000011111iiiii	0000000100000000	TRAP	vector
	0000011111100000	0000000100100000	HALT	－
	0000011111100000	0000000101000000	RETI	－
	0000011111100000	0000000101000100	CTRET	－
	0000011111100000	0000000101000110	DBRET 注	－
	0000011111100000	0000000101100000	DI	－
	1000011111100000	0000000101100000	EI	－
XI	rrrrr111111RRRRR	wwwww01000100000	MUL	reg1, reg2, reg3
	rrrrr111111RRRRR	wwwww01000100010	MULU	reg1, reg2, reg3
	rrrrr111111RRRRR	wwwww01010000000	DIVH	reg1, reg2, reg3
	rrrrr111111RRRRR	wwwww01010000010	DIVHU	reg1, reg2, reg3
	rrrrr111111RRRRR	wwwww01011000000	DIV	reg1, reg2, reg3
	rrrrr111111RRRRR	wwwww01011000010	DIVU	reg1, reg2, reg3
	rrrrr111111RRRRR	wwwww011001cccc0	CMOV	cccc, reg1, reg2, reg3
XII	rrrrr111111iiiii	wwwww01001IIII00	MUL	imm9, reg2, reg3
	rrrrr111111iiiii	wwwww01001IIII10	MULU	imm9, reg2, reg3
	rrrrr111111iiiii	wwwww011000cccc0	CMOV	cccc, imm5, reg2, reg3
	rrrrr11111100000	wwwww01101000000	BSW	reg2, reg3
	rrrrr11111100000	wwwww01101000010	BSH	reg2, reg3
	rrrrr11111100000	wwwww01101000100	HSW	reg2, reg3
XIII	0000011001iiiiiL	LLLLLLLLLLLRRRRR	DISPOSE	imm5, list12, [reg1]
	0000011001iiiiiL	LLLLLLLLLLL00000	DISPOSE	imm5, list12
	0000011110iiiiiL	LLLLLLLLLLL00001	PREPARE	list12, imm5
	0000011110iiiiiL	LLLLLLLLLLLff011	PREPARE	list12, imm5, sp/imm

★ **注　タイプ C の製品ではサポートしていません。**

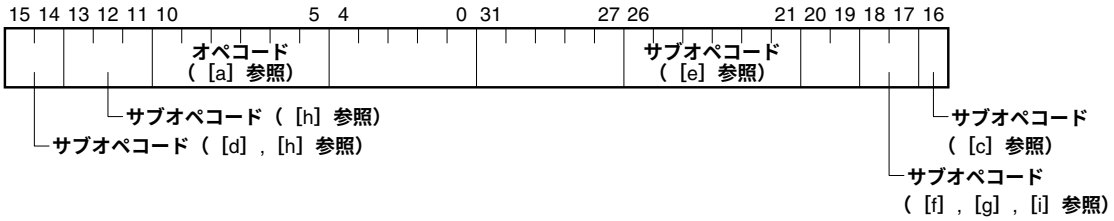
付録C　命令オペコード・マップ

この章では，次に示す命令コードに対応したオペコード・マップを示します。

(1) 16 ビット・フォーマット命令



(2) 32 ビット・フォーマット命令



備考　オペランドの凡例

略　　号	意　　味
R	reg1：汎用レジスタ（ソース・レジスタとして使用）
r	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。）
w	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
cccc	条件コードを示す 4 ビット・データ

[a] オペコード部

ビット10	ビット9	ビット8	ビット7	ビット6, 5 0,0	0,1	1,0	1,1	フォーマット
0	0	0	0	MOV R, r NOP 注1	NOT	DIVH SWITCH 注2 DBTRAP 未定義注3	JMP 注4 SLD.BU 注5 SLD.HU 注6	I, IV
0	0	0	1	SATSUBR ZXB 注4	SATSUB SXB 注4	SATADD R, r ZXH 注4	MULH SXH 注4	I
0	0	1	0	OR	XOR	AND	TST	
0	0	1	1	SUBR	SUB	ADD R, r	CMP R, r	
0	1	0	0	MOV imm5, r CALLT 注4	SATADD imm5, r	ADD imm5, r	CMP imm5, r	II
0	1	0	1	SHR imm5, r	SAR imm5, r	SHL imm5, r	MULH imm5, r 未定義注4	
0	1	1	0	SLD.B				IV
0	1	1	1	SST.B				
1	0	0	0	SLD.H				
1	0	0	1	SST.H				
1	0	1	0	SLD.W 注7 SST.W 注7				
1	0	1	1	Bcond				III
1	1	0	0	ADDI	MOVEA MOV imm32, R 注4	MOVHI DISPOSE 注4	SATSUBI	VI, XIII
1	1	0	1	ORI	XORI	ANDI	MULHI 未定義注4	VI
1	1	1	0	LD.B	LD.H 注8 LD.W 注8	ST.B	ST.H 注8 ST.W 注8	VII
1	1	1	1	JR JARL LD.BU 注10 PREPARE 注11		ビット操作1 注9	LD.HU 注10 未定義注11 拡張1 注12	V, VII, VIII, XIII

注 1. R（reg1）が r0，かつ r（reg2）が r0（reg1, reg2 を持たない命令）の場合。

2. R（reg1）が r0 でなく，かつ r（reg2）が r0（reg1 を持ち，reg2 を持たない命令）の場合。

3. R（reg1）が r0，かつ r（reg2）が r0 でない（reg1 を持たず，reg2 を持つ命令）場合。

4. r（reg2）が r0（reg2 を持たない命令）の場合。

5. ビット 4 が 0，かつ r（reg2）が r0 でない（reg2 を持つ命令）場合。

6. ビット 4 が 1，かつ r（reg2）が r0 でない（reg2 を持つ命令）場合。

7. [b] を参照してください。

8. [c] を参照してください。

9. [d] を参照してください。

10. ビット 16 が 1，かつ r（reg2）が r0 でない（reg2 を持つ命令）場合。

11. ビット 16 が 1，かつ r（reg2）が r0（reg2 を持たない命令）の場合。

12. [e] を参照してください。

★ **備考**　DBTRAP 命令は，タイプ C の製品ではサポートしていません。

[b] ショート・フォーマット・ロード／ストア命令（ディスプレースメント／サブオペコード部）

ビット10	ビット9	ビット8	ビット7	ビット0	
				0	1
0	1	1	0	SLD.B	
0	1	1	1	SST.B	
1	0	0	0	SLD.H	
1	0	0	1	SST.H	
1	0	1	0	SLD.W	SST.W

[c] ロード／ストア命令（ディスプレースメント／サブオペコード部）

ビット6	ビット5	ビット16	
		0	1
0	0	LD.B	
0	1	LD.H	LD.W
1	0	ST.B	
1	1	ST.H	ST.W

[d] ビット操作命令1（サブオペコード部）

ビット15	ビット14	
	0	1
0	SET1　bit#3, disp16 [R]	NOT1　bit#3, disp16 [R]
1	CLR1　bit#3, disp16 [R]	TST1　bit#3, disp16 [R]

[e] 拡張1（サブオペコード部）

ビット 26	ビット 25	ビット 24	ビット 23	ビット 22, 21 0,0	0,1	1,0	1,1	フォーマット
0	0	0	0	SETF	LDSR	STSR	未定義	IX
0	0	0	1	SHR	SAR	SHL	ビット操作 2 注1	
0	0	1	0	TRAP	HALT	RETI 注2 CTRET 注2 DBRET 注2 未定義	EI 注3 DI 注3 未定義	X
0	0	1	1	未定義		未定義		−
0	1	0	0	SASF	MUL　R, r, w MULU　R, r, w 注4	MUL　imm9, r, w MULU　imm9, r, w 注4		IX, XI, XII
0	1	0	1	DIVH DIVHU 注4		DIV DIVU 注4		XI
0	1	1	0	CMOV cccc, imm5, r, w	CMOV cccc, R, r, w	BSW 注5 BSH 注5 HSW 注5	未定義	XI, XII
0	1	1	1	不正命令				−
1	x	x	x					

注 1. [f] を参照してください。

2. [g] を参照してください。

3. [h] を参照してください。

4. ビット 17 が 1 の場合。

5. [i] を参照してください。

★ **備考**　DBRET 命令は，タイプ C の製品ではサポートしていません。

[f] ビット操作命令2（サブオペコード部）

ビット 18	ビット 17 0	1
0	SET1　r, [R]	NOT1　r, [R]
1	CLR1　r, [R]	TST1　r, [R]

[g] 復帰命令（サブオペコード部）

ビット 18	ビット 17 0	1
0	RETI	未定義
1	CTRET	DBRET

[h] PSW操作命令（サブオペコード部）

ビット 15	ビット 14	ビット 13, 12, 11							
		0,0,0	0,0,1	0,1,0	0,1,1	1,0,0	1,0,1	1,1,0	1,1,1
0	0	DI	未定義						
0	1	未定義							
1	0	EI	未定義						
1	1	未定義							

[i] エンディアン変換命令（サブオペコード部）

ビット 18	ビット 17	
	0	1
0	BSW	BSH
1	HSW	未定義

付録D　V850 CPUとのアーキテクチャ上の相違点

（1/2）

項　　目		V850E1 CPU	V850 CPU
命令 （オペランドを含む）	BSH　reg2, reg3	あり	なし
	BSW　reg2, reg3		
	CALLT　imm6		
	CLR1　reg2, [reg1]		
	CMOV　cccc, imm5, reg2, reg3		
	CMOV　cccc, reg1, reg2, reg3		
	CTRET		
	DBRET 注		
	DBTRAP 注		
	DISPOSE　imm5, list12		
	DISPOSE　imm5, list12 [reg1]		
	DIV　reg1, reg2, reg3		
	DIVH　reg1, reg2, reg3		
	DIVHU　reg1, reg2, reg3		
	DIVU　reg1, reg2, reg3		
	HSW　reg2, reg3		
	LD.BU　disp16 [reg1], reg2		
	LD.HU　disp16 [reg1], reg2		
	MOV　imm32, reg1		
	MUL　imm9, reg2, reg3		
	MUL　reg1, reg2, reg3		
	MULU　reg1, reg2, reg3		
	MULU　imm9, reg2, reg3		
	NOT1　reg2, [reg1]		
	PREPARE　list12, imm5		
	PREPARE　list12, imm5, sp/imm		
	SASF　cccc, reg2		
	SET1　reg2, [reg1]		
	SLD.BU　disp4 [ep], reg2		
	SLD.HU　disp5 [ep], reg2		
	SWITCH　reg1		
	SXB　reg1		
	SXH　reg1		
	TST1　reg2, [reg1]		
	ZXB　reg1		
	ZXH　reg1		

★　**注　タイプ C の製品ではサポートしていません。**

（2/2）

項　目		V850E1 CPU	V850 CPU
命令フォーマット	Format IV	一部，形式が異なります。	
	Format XI	あり	なし
	Format XII		
	Format XIII		
命令実行クロック数		一部の命令で数値が異なります。	
プログラム空間		64M バイト・リニア	16M バイト・リニア
プログラム・カウンタ（PC）の有効ビット		下位 26 ビット	下位 24 ビット
システム・レジスタ	CALLT 実行時状態退避レジスタ（CTPC, CTPSW）	あり	なし
	例外／ディバグ・トラップ時状態退避レジスタ（DBPC, DBPSW）		
	CALLT ベース・ポインタ（CTBP）		
	ディバグ・インタフェース・レジスタ（DIR）注1		
	ブレークポイント制御レジスタ 0, 1（BPC0, BPC1）注1		
	プログラム ID レジスタ（ASID）注1		
	ブレークポイント・アドレス設定レジスタ 0, 1（BPAV0, BPAV1）注1		
	ブレークポイント・アドレス・マスク・レジスタ 0,1（BPAM0,BPAM1）注1		
	ブレークポイント・データ設定レジスタ 0, 1（BPDV0, BPDV1）注1		
	ブレークポイント・データ・マスク・レジスタ 0, 1（BPDM0, BPDM1）注1		
	例外トラップ時の状態退避レジスタ	DBPC, DBPSW	EIPC, EIPSW
不正命令コード		命令コードのコード領域が異なります。	
ミス・アライン・アクセス許可／禁止の設定		製品により設定可能	設定不可（ミス・アライン・アクセス禁止）
★ ノンマスカブル割り込み（NMI）	入力	3（タイプ A, B, C の製品）	1
★		1（タイプ D, E, F の製品）	
	例外コード	0010H, 0020H, 0030H	0010H
	ハンドラ・アドレス	00000010H, 00000020H, 00000030H	00000010H
ディバグ・トラップ注2		あり	なし
パイプライン		次の命令で，パイプラインの流れが異なります。 • 算術演算命令 • 分岐命令 • ビット操作命令 • 特殊命令（TRAP, RETI）	

★ 注 1.　タイプ A, B の製品のみ使用できます。

★ 2.　タイプ C の製品ではサポートしていません。

付録E　V850 CPUに対してV850E1 CPUで追加した命令

V850E1 CPU の命令コードは，V850 CPU の命令コードに対し，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ます。V850E1 CPU では，V850 CPU で実行しても意味を持たない命令（主に r0 レジスタへの書き込みを行う命令）を追加命令として拡張しています。

次の表に V850E1 CPU で追加した命令コードに対応した V850 CPU の命令を示します。V850 CPU を搭載した製品から V850E1 CPU を搭載した製品に置き換えるときの参考にしてください。

表E－1　V850E1 CPUで追加した命令と命令コードが同一のV850 CPUの命令（1/2）

V850E1 CPU で追加した命令	V850E1 CPU の命令コードと同一の V850 CPU の命令
CALLT　imm6	MOV　imm5, r0 または SATADD　imm5, r0
DISPOSE　imm5, list12	MOVHI　imm16, reg1, r0 または SATSUBI　imm16, reg1, r0
DISPOSE　imm5, list12 [reg1]	MOVHI　imm16, reg1, r0 または SATSUBI　imm16, reg1, r0
MOV　imm32, reg1	MOVEA　imm16, reg1, r0
SWITCH　reg1	DIVH　reg1, r0
SXB　reg1	SATSUB　reg1, r0
SXH　reg1	MULH　reg1, r0
ZXB　reg1	SATSUBR　reg1, r0
ZXH　reg1	SATADD　reg1, r0
(RFU)	MULH　imm5, r0
(RFU)	MULHI　imm16, reg1, r0
BSH　reg2, reg3	不正命令
BSW　reg2, reg3	
CMOV　cccc, imm5, reg2, reg3	
CMOV　cccc, reg1, reg2, reg3	
CTRET	
DIV　reg1, reg2, reg3	
DIVH　reg1, reg2, reg3	
DIVHU　reg1, reg2, reg3	
DIVU　reg1, reg2, reg3	
HSW　reg2, reg3	
MUL　imm9, reg2, reg3	
MUL　reg1, reg2, reg3	
MULU　reg1, reg2, reg3	
MULU　imm9, reg2, reg3	
SASF　cccc, reg2	
CLR1　reg2, [reg1]	未定義
DBRET 注	
DBTRAP 注	

★　**注　タイプ C の製品ではサポートしていません。**

表E－1　V850E1 CPUで追加した命令と命令コードが同一のV850 CPUの命令（2/2）

V850E1 CPU で追加した命令	V850E1 CPU の命令コードと同一の V850 CPU の命令
LD.BU　disp16 [reg1], reg2	未定義
LD.HU　disp16 [reg1], reg2	
NOT1　reg2, [reg1]	
PREPARE　list12, imm5	
PREPARE　list12, imm5, sp/imm	
SET1　reg2, [reg1]	
SLD.BU　disp4 [ep], reg2	
SLD.HU　disp5 [ep], reg2	
TST1　reg2, [reg1]	

付録F　総合索引

F. 1　50音で始まる語句の索引

【あ】

アドレシング・モード … 43
アドレス空間 … 41
アライン・ハザード … 189

【い】

イミーディエト・アドレシング … 45

【お】

オペランド・アドレス … 45

【か】

拡張命令形式1 … 49
拡張命令形式2 … 50
拡張命令形式3 … 50
拡張命令形式4 … 50
各命令実行時のパイプラインの流れ … 178

【き】

起　　動 … 172

【こ】

効率的なパイプライン処理 … 177

【さ】

算術演算命令 … 52
算術演算命令（パイプライン） … 181

【し】

システム・レジスタ … 19
ジャンプ命令形式 … 48
条件分岐命令形式 … 48
乗算命令 … 51
乗算命令（パイプライン） … 180
ショート・パス … 192
除算命令（パイプライン） … 181

【す】

スタック操作命令形式1 … 50
ストア命令 … 51
ストア命令（パイプライン） … 180

【せ】

整　　数 … 39

【そ】

ソフトウエア例外 … 166

【て】

ディバグ・インタフェース・レジスタ … 27
ディバグ機能用命令 … 54
ディバグ機能用命令（パイプライン） … 188
ディバグ・トラップ … 168
ディバグ・モードへの移行 … 194
ディバグ・モードへの移行方法 … 194
データ・アラインメント … 40
データ形式 … 37
データ・タイプ … 37
データ表現 … 39

【と】

特殊命令 … 53
特殊命令（パイプライン） … 184

【な】

内部構成 … 15

【の】

ノンブロッキング・ロード／ストア … 175
ノンマスカブル割り込み … 165

【は】

ハーバード・アーキテクチャ … 192
ハーフワード … 38
バイト … 38
パイプライン … 173
パイプラインに関する補足事項 … 192
パイプラインの乱れ … 189
汎用レジスタ … 17

【ひ】

ビット … 38, 39
ビット・アドレシング … 46
ビット操作命令 … 53
ビット操作命令（パイプライン） … 184
ビット操作命令形式 … 49

【ふ】

符号なし整数 … 39
ブレークポイント・アドレス設定レジスタ0, 1 … 33
ブレークポイント・アドレス・マスク・レジスタ0, 1 … 34
ブレークポイント制御レジスタ0, 1 … 30
ブレークポイント・データ設定レジスタ0, 1 … 35
ブレークポイント・データ・マスク・レジスタ0, 1 … 36
プログラムIDレジスタ … 32
プログラム・カウンタ … 18
プログラム作成時の注意点 … 191
プログラム・ステータス・ワード … 22
プログラム・レジスタ … 17
分岐命令 … 53
分岐命令（パイプライン） … 182

【へ】

ベースト・アドレシング … 45

【ほ】

飽和演算命令 … 52
飽和演算命令（パイプライン） … 182

【ま】

マスカブル割り込み … 163

【む】

無条件分岐命令 … 183

【め】

命令アドレス … 43
命令オペコード・マップ … 218
命令セット … 55
命令フォーマット … 47
メモリ・マップ … 42

【り】

リセット … 171
リセット後のレジスタの状態 … 171

【れ】

例外処理 … 166
例外／ディバグ・トラップ時状態退避レジスタ … 25
例外トラップ … 167
例外トラップ，ディバグ・トラップからの復帰 …170

レジスタ・アドレシング … 45
レジスタ・アドレシング（レジスタ間接） … 44
レジスタ・セット … 16
レラティブ・アドレシング（PC相対） … 43

【ろ】

ロード／ストア命令16ビット形式 … 48
ロード／ストア命令32ビット形式 … 49
ロード命令 … 51
ロード命令（パイプライン） … 179
論理演算命令 … 52
論理演算命令（パイプライン） … 182

【わ】

ワード … 37
ワード転送命令（パイプライン） … 181
割り込み時状態退避レジスタ … 20
割り込み処理 … 163
割り込み要因レジスタ … 21
割り込み，例外処理からの復帰 … 169

F. 2　数字，アルファベットで始まる語句の索引

【数字】

16ビット・フォーマット命令 … 218
2クロック分岐 … 176
32ビット・フォーマット命令 … 218
3オペランド命令形式 … 49

【A】

ADD … 57
ADDI … 58
AND … 59
ANDI … 60
ASID … 32

【B】

Bcond … 61
BPAM0 … 34
BPAM1 … 34
BPAV0 … 33
BPAV1 … 33
BPC0 … 30
BPC1 … 30
BPDM0 … 36
BPDM1 … 36
BPDV0 … 35
BPDV1 … 35
BR命令（パイプライン） … 183
BSH … 63
BSW … 64

【C】

CALLT … 65
CALLT実行時状態退避レジスタ … 24
CALLTベース・ポインタ … 26
CALLT命令（パイプライン） … 184
CLR1 … 66
CLR1命令（パイプライン） … 184
CMOV … 67
CMP … 68
CTBP … 26
CTPC … 24
CTPSW … 24
CTRET … 69
CTRET命令（パイプライン） … 185

【D】

DBPC … 25
DBPSW … 25
DBRET … 70
DBRET命令（パイプライン） … 188
DBTRAP … 71
DBTRAP命令（パイプライン） … 188
DI … 72
DI命令（パイプライン） … 185
DIR … 27
DISPOSE … 73
DISPOSE命令（パイプライン） … 185
DIV … 75
DIVH … 76
DIVHU … 78
DIVU … 79

【E】

ECR … 21
EI … 80
EIPC … 20
EIPSW … 20
EI命令（パイプライン） … 185

【F】

FEPC … 21
FEPSW … 21
Format I … 47
Format II … 47
Format III … 48

Format IV … 48
Format V … 48
Format VI … 49
Format VII … 49
Format VIII … 49
Format IX … 49
Format X … 50
Format XI … 50
Format XII … 50
Format XIII … 50

【H】

HALT … 81
HALT**命令（パイプライン）** … 186
HSW … 82

【I】

imm-reg**命令形式** … 47

【J】

JARL … 83
JMP … 84
JMP**命令（パイプライン）** … 183
JR … 85

【L】

LD.B … 86
LD.BU … 87
LD.H … 88
LD.HU … 90
LD.W … 92
LDSR … 94
LDSR**命令（パイプライン）** … 186
LD**命令** … 51
LD**命令（パイプライン）** … 179

【M】

MOV … 95
MOVEA … 96
MOVHI … 97
MUL … 98
MULH … 100
MULHI … 101
MULU … 102

【N】

NMI**時状態退避レジスタ** … 21
NOP … 104
NOP**命令（パイプライン）** … 186
NOT … 105
NOT1 … 106
NOT1**命令（パイプライン）** … 184

【O】

OR … 107
ORI … 108

【P】

PC … 18
PREPARE … 109
PREPARE**命令（パイプライン）** … 187
PSW … 22

【R】

r0-r31 … 17
reg-reg**命令形式** … 47
RETI … 111
RETI**命令（パイプライン）** … 187

【S】

SAR … 113
SASF … 114
SATADD … 115
SATSUB … 117
SATSUBI … 118
SATSUBR … 119

SET1 … 120
SET1**命令（パイプライン）** … 184
SETF … 121
SHL … 123
SHR … 124
SLD.B … 125
SLD.BU … 127
SLD.H … 129
SLD.HU … 131
SLD.W … 133
SLD**命令** … 51
SLD**命令（パイプライン）** … 179
SST.B … 135
SST.H … 136
SST.W … 138
SST**命令** … 51
ST.B … 140
ST.H … 141
ST.W … 143
STSR … 145
STSR**命令（パイプライン）** … 186
ST**命令** … 51
SUB … 146
SUBR … 147
SWITCH … 148
SWITCH**命令（パイプライン）** … 187
SXB … 149
SXH … 150

【T】

TRAP … 151
TRAP**命令（パイプライン）** … 188
TST … 152
TST1 … 153
TST1**命令（パイプライン）** … 184

【X】

XOR … 154
XORI … 155

【Z】

ZXB … 156
ZXH … 157

★

付録G　改版履歴

G. 1　本版で改訂された主な箇所

箇　所	内　　容
全般	・対象製品から製品名を削除，製品タイプを追加
p.17	2. 1（1）**汎用レジスタ**（r0-r31）　説明変更
p.19	**表**2－2　**システム・レジスタ番号**　変更
p.25	2. 2. 6　**例外／ディバグ・トラップ時状態退避レジスタ**（DBPC, DBPSW）　説明変更および追加
p.25	**表**2－3　DBPC**に退避される内容**　追加
p.27	**図**2－10　**ディバグ・インタフェース・レジスタ**（DIR）　変更
p.31	**図**2－11　**ブレークポイント制御レジスタ**0, 1（BPC0, BPC1）　変更
p.32	2. 2. 10　**プログラム**ID**レジスタ**（ASID）　説明追加
p.33	2. 2. 11　**ブレークポイント・アドレス設定レジスタ**0, 1（BPAV0, BPAV1）　説明追加
p.34	2. 2. 12　**ブレークポイント・アドレス・マスク・レジスタ**0, 1（BPAM0, BPAM1）　説明追加
p.35	2. 2. 13　**ブレークポイント・データ設定レジスタ**0, 1（BPDV0, BPDV1）　説明追加
p.36	2. 2. 14　**ブレークポイント・データ・マスク・レジスタ**0, 1（BPDM0, BPDM1）　説明追加
p.40	3. 3　**データ・アラインメント**　変更
p.98, 99	5. 3　**命令セット**　MUL, MULU　説明変更および注意追加
p.151	5. 3　**命令セット**　TRAP　オペレーション修正
p.160, 161	**表**5－6　**命令実行クロック数一覧**　注変更および注追加
p.167	6. 2. 2　**例外トラップ**　<4>追加
p.168	6. 2. 3　**ディバグ・トラップ**　説明追加
p.194	**第**9**章　ディバグ・モードへの移行**　追加
p.233	**付録**F　**改版履歴**　追加
修正版（U14559JJ3V1UM00）で改訂された主な箇所	
p.98, 99	5. 3　**命令セット**　MUL, MULU　注意修正
p.126	5. 3　**命令セット**　SLD.B　注意（2）追加
p.128	5. 3　**命令セット**　SLD.BU　注意（2）追加
p.130	5. 3　**命令セット**　SLD.H　注意（2）追加
p.132	5. 3　**命令セット**　SLD.HU　注意（2）追加
p.134	5. 3　**命令セット**　SLD.W　注意（2）追加
p.203	**付録**A　**注意事項**　追加

G. 2　前版までの改版履歴

前版までの改版履歴を次に示します。なお，適用箇所は各版での章を示します。

(1/2)

版　数	前版までの改版内容	適用箇所
第2版	・対象製品に次の製品（開発中）を追加 NB85ET, NU85E, NU85ET, μPD703108, 703114, 70F3114, 703116 ・対象製品から次の製品を削除 μPD703117 ・次の製品が開発中→開発済み μPD703106, 703107, 70F3107	全般
	図2－1　レジスター覧　注変更	第2章　レジスタ・セット
	表2－2　システム・レジスタ番号　変更	
	図2－6　プログラム・ステータス・ワード（PSW）　注追加	
	2. 2. 6　例外／ディバグ・トラップ時状態退避レジスタ（DBPC, DBPSW）　注追加	
	2. 2. 8　ディバグ・インタフェース・レジスタ（DIR）　注意変更	
	2. 2. 9　ブレークポイント制御レジスタ0, 1（BPC0, BPC1）　注意変更	
	図2－11　ブレークポイント制御レジスタ0, 1（BPC0, BPC1）　変更	
	2. 2. 10　プログラムIDレジスタ（ASID）　注意変更	
	2. 2. 11　ブレークポイント・アドレス設定レジスタ0, 1（BPAV0, BPAV1）　注意変更	
	2. 2. 12　ブレークポイント・アドレス・マスク・レジスタ0, 1（BPAM0, BPAM1）　注意変更	
	2. 2. 13　ブレークポイント・データ設定レジスタ0, 1（BPDV0, BPDV1）　注意変更	
	2. 2. 14　ブレークポイント・データ・マスク・レジスタ0, 1（BPDM0, BPDM1）　注意変更	
	5. 2（10）ディバグ機能用命令　注意追加	第5章　命　　令
	5. 3　命令セット　DBRET　注意追加	
	5. 3　命令セット　DBTRAP　注意追加	
	表5－6　命令実行クロック数一覧（NB85E, NB85ET, NU85E, NU85ET）　注変更と追加	
	表5－7　命令実行クロック数一覧（V850E/MA1, V850E/MA2, V850E/IA1, V850E/IA2）　注変更	
	表6－1　割り込み，例外コード一覧　注追加	第6章　割り込みと例外
	6. 2. 3　ディバグ・トラップ　注意追加	
	8. 1. 2　2クロック分岐　備考，例追加	第8章　パイプライン
	8. 1. 3　効率的なパイプライン処理　注意追加	
	8. 2（2）V850E/MA1, V850E/MA2, V850E/IA1, V850E/IA2の場合　説明修正	
	8. 2. 1（2）SLD命令　説明修正	
	8. 2. 3　乗算命令　説明修正	
	8. 2. 4（3）除算命令　備考追加	
	8. 2. 8（2）TST1命令　説明修正	
	8. 2. 9（3）DI, EI命令　備考追加	
	8. 2. 9（7）NOP命令　注意追加	
	8. 3　パイプラインの乱れ　追加	
	8. 4　パイプラインに関する補足事項　追加	
	表A－1　命令機能一覧（アルファベット順）　注追加	付録A　命令一覧
	表A－2　命令一覧（フォーマット順）　注追加	

（2/2）

版　数	前版までの改版内容	適用箇所
第2版	付録B（2）32ビット・フォーマット命令　図修正	付録B　命令オペコード・マップ
	付録B［a］オペコード部　備考追加	
	付録B［e］拡張1（サブオペコード部）　備考追加	
	付録C　V850 CPUとのアーキテクチャ上の相違点　注追加	付録C　V850 CPUとのアーキテクチャ上の相違点
	表D－1　V850E1 CPUで追加した命令と命令コードが同一のV850 CPUの命令　注追加	付録D　V850 CPUに対してV850E1 CPUで追加した命令

［メ　モ］

【発　行】
NEC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
電話（代表）：044(435)5111

お問い合わせ先

【ホームページ】

NECエレクトロニクスの情報がインターネットでご覧になれます。
URL(アドレス)　http://www.necel.co.jp/

【営業関係，技術関係お問い合わせ先】

半導体ホットライン
（電話：午前 9:00～12:00，午後 1:00～5:00）
電　話　：044−435−9494
E-mail　：info@necel.com

【資料請求先】

NECエレクトロニクスのホームページよりダウンロードいただくか，NECエレクトロニクスの販売特約店へお申し付けください。