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

■本ドキュメントに記載されているURLは、以下のとおり読み替えをお願いいたします。
http://www.necel.com/
http://www2.renesas.com/
↓
開発環境トップページ　http://japan.renesas.com/tools
ダウンロードポータル　http://japan.renesas.com/tool_download

■技術問合せについては、以下のページをご覧ください。
http://japan.renesas.com/tech_inquiry

■ツールユーザ登録については、以下のページをご覧ください。
http://japan.renesas.com/myrenesas

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

CA850 Ver.3.20

Cコンパイラ・パッケージ

アセンブリ言語編

対象デバイス
V850シリーズ

資料番号　U18514JJ1V0UM00（第1版）
発行年月　May 2007　CP(K)

〔メ　モ〕

目次要約

Windowsは，米国Microsoft Corporationの米国およびその他の国における登録商標または商標です。

- 本資料に記載されている内容は2007年5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

対象デバイス　V850シリーズ Cコンパイラ・パッケージは，NECエレクトロニクス製RISCマイクロプロセッサV850シリーズ用のオブジェクト・コードを生成するためのコンパイラ・パッケージです。
このマニュアルは，CA850 Cコンパイラ・パッケージを対象としています。

対 象 者　このマニュアルは，V850シリーズ Cコンパイラ・パッケージを使用して，アプリケーション・システムを開発するユーザを対象としています。

目　　的　このマニュアルでは，CA850 Cコンパイラ・パッケージに含まれるアセンブラ（as850）がサポートするアセンブリ言語仕様について説明します。

構　　成　このマニュアルは，次の内容で構成されています。

- 概要
- アセンブリ言語仕様
- 命令セット
- アセンブリ言語命令
- 疑似命令

読み方の注意
- このマニュアルでは，Cコンパイラ・パッケージの各プログラム名を次のように記述します。
  Cコンパイラ・パッケージ→CA850
  アセンブラ→as850
  Cコンパイラ→ca850
- このマニュアルでは，V850シリーズでも“ V850E ”に特有の主な部分については，タイトル名や“ 【V850E】 ”などで示しており，“ V850E2 ”に特有の主な部分については，タイトル名や“ 【V850E2】 ”などで示しています。

関連資料　**このマニュアルを使用する場合は，次の資料もあわせてご覧ください。**

関連資料は暫定版の場合がありますが，この資料では「暫定」の表示をしておりません。

あらかじめご了承ください。

○開発ツールに関する資料（ユーザーズ・マニュアル）

資　料　名		資料番号	
		和文	英文
CA850 Ver.3.20 **Cコンパイラ・パッケージ**	**操作編**	U18512J	U18512E
	C言語編	U18513J	U18513E
	アセンブリ言語編	**このマニュアル**	U18514E
	リンク・ディレクティブ編	U18515J	U18515E
PM+ Ver.6.30 **プロジェクト・マネージャ**		U18416J	U18416E
ID850 Ver.3.00 **統合ディバッガ**	**操作編**	U17358J	U17358E
ID850NW Ver.3.10 **統合ディバッガ**	**操作編**	U17369J	U17369E
ID850QB Ver.3.20 **統合ディバッガ**	**操作編**	U17964J	U17964E
SM+ **システム・シミュレータ**	**操作編**	U17246J	U17246E
	ユーザ・オープン・インタフェース編	U18212J	U18212E
SM850 Ver.2.50 **システム・シミュレータ**	**操作編**	U16218J	U16218E
SM850 Ver.2.00**以上　システム・シミュレータ**	**外部部品ユーザ・オープン・インタフェース仕様編**	U14873J	U14873E
RX850 Ver.3.20**以上　リアルタイム**OS	**基礎編**	U13430J	U13430E
	インストレーション編	U17419J	U17419E
	テクニカル編	U13431J	U13431E
	タスク・デバッガ編	U17420J	U17420E
RX850 Pro Ver.3.21 **リアルタイム**OS	**基礎編**	U18165J	U18165E
	内部構造編	U18164J	U18164E
	タスク・デバッガ編	U17422J	U17422E
RX850V4 Ver.4.22 **リアルタイム**OS	**機能編**	U16643J	U16643E
	内部構造編	U16644J	U16644E
	タスク・デバッガ編	U16811J	U16811E
RX-NET **ネットワーク・ライブラリ**（TCP/IP）Ver.1.30		U15083J	－
RX-NET **ネットワーク・ライブラリ**（PPP）Ver.1.30		U15303J	－
RX-NET **ネットワーク・ライブラリ**（DNS）Ver.1.30		U15304J	－
RX-NET **ネットワーク・ライブラリ**（DHCP）Ver.1.30		U15382J	－
RX-NET **ネットワーク・ライブラリ**（SMTP）		U15505J	－
RX-NET **ネットワーク・ライブラリ**（POP）		U15539J	－
RX-NET Ver.1.10 **ネットワーク・ライブラリ**（telnet）		U16085J	－
RX-NET Ver.1.00 **ネットワーク・ライブラリ**（FTP）		U15946J	－
RX-NET Ver.1.00 **ネットワーク・ライブラリ**（WebServer）		U16294J	－
AZ850 Ver.3.30 **システム・パフォーマンス・アナライザ**		U17423J	U17423E
AZ850V4 Ver.4.10 **システム・パフォーマンス・アナライザ**		U17093J	U17093E
TW850 Ver.2.00 **性能解析チューニング・ツール**		U17241J	U17241E

〔メ　モ〕

目次

図の目次

図番号　タイトル,ページ

表の目次

表番号　タイトル,ページ

第 1 章　アセンブリ言語仕様

この章では，CA850 アセンブラ（as850）がサポートするアセンブリ言語仕様について説明します。

1.1　アセンブリ言語文の構成

アセンブリ言語文は，“ ラベル ”, “ ニモニック ”, “ オペランド ”，および “ コメント ” から構成されます。

```
［ラベル］：［ニモニック］　［オペランド］，　［オペランド］ --　［コメント］
```

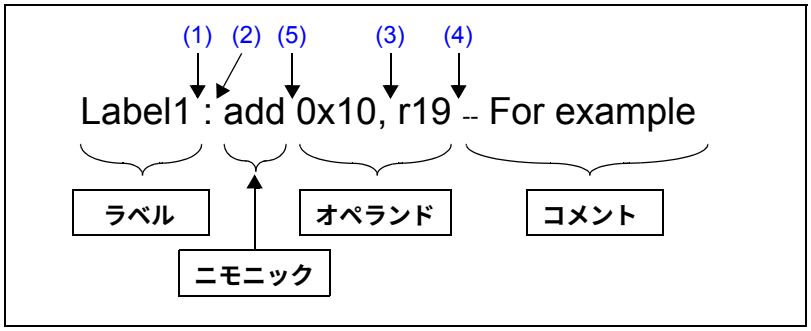
次の (1) ～ (4) には，1 つ以上の空白があってもなくても問題ありません。

(1)　ラベルとコロンの間
(2)　コロンとニモニックの間
(3)　2 つ目以降のオペランドの前
(4)　コメントの始まり “ -- ” の前

次の (5) には，1 つ以上の空白が必要です。

(5)　ニモニックとオペランドの間

図 1 － 1　アセンブリ言語文の構成



アセンブリ言語文は，基本的に 1 行に 1 文を記述します。文の最後は改行（リターン）します。なお，“ ; （セミコロン）” を使用すると，1 行に複数のアセンブリ言語文を記述することができます。

1.1.1 ラベル

ラベルとは，プログラムの任意の行に記述できる，いわゆる“名札”です。ラベルは，条件分岐するときや，サブルーチンへ分岐するとき，その分岐先の名前として使用できます。

たとえば，分岐命令の 1 つである“jr 命令”を使うときは，次のように記述します。

```
        jr      Label1
```

この命令の実行によって Label1 のある箇所へ分岐します。つまり，Label1 という名前にラベルを記述するときは，次のように記述します。

```
Label1 :
```

ラベルは，複数行に渡って違うラベルを定義することができます。

```
Label1 :
Label2 :
```

しかし，1 行に複数のラベルを指定することはできません。

```
Label1: Label2:     -- 1 行に複数のラベルを指定することはできません。
```

なお，ラベル名とコロンの間は，1 つ以上の空白があってもなくても問題ありません。

ラベルは，使用する前に“定義／宣言”をする必要があります。定義方法，宣言方法については「1.2.2 ラベル」を参照してください。

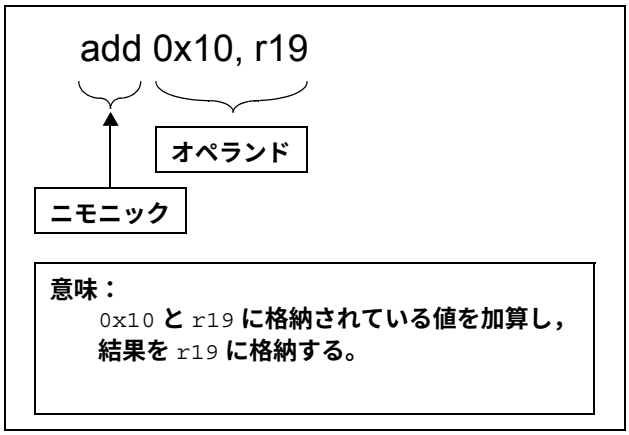
1.1.2 ニモニックとオペランド

ニモニックとは，各命令（V850 マシン・コード）に対し，それぞれ割り当てられた英文字列です。マシン・コードは，そのままの状態では人間には理解しづらいので，各マシン・コードに対してつけられた名前が“ニモニック”です。つまり，命令そのものを意味します。ニモニックはその操作内容が容易に推定できるように，英語を基本とした“言葉に近い表記”になっています。

たとえば，“add”というニモニックは“加算”を意味し，“mul”であれば“乗算”を意味します。オペランドとは，各命令操作に操作される対象を指します。ニモニックが“add”（加算）だった場合，オペランドはその演算の対象となるものです。ニモニックの次（右）にオペランドを記述します。

なお，ニモニックとオペランドの間は，1 つ以上の空白が必要です。

図 1 － 2 ニモニックとオペランド



アセンブリ言語命令は“ニモニック”と“オペランド”の組み合わせで成り立っています。オペランドの数はニモニックによって異なります。

V850 に搭載されているアセンブリ言語命令の一覧とその仕様については，「第 3 章 アセンブリ言語命令」を参照してください。

1.1.3 コメント

アセンブリ言語プログラム内にコメントを記入することができます。as850 では，次の記述以降をその行の終わりまでをコメントとして扱います。

```
--
```

```
#
```

ただし，“ # ” は文の先頭[注]にあった場合のみ，その行の終わりまでをコメントとして扱います。

```
# comment
       add    0x10, r19 -- comment-1
       sub    r18, r19  -- comment-2
```

なお，コメント内では “ EUC ” または “ シフト JIS コード ”の日本語を記述することができます。

注　行頭の空白は，文に含まれません。“ # ” の前に空白が含まれる場合も，その行の終わりまでをコメントとして扱います。

1.1.4 文字セット

as850 がサポートするソース・プログラムで，使用できる文字は次のとおりです。

表 1 － 1 文字セットとその用途

文字	用途
英小文字（a-z）	ニモニック，識別子，および定数の構成
英大文字（A-Z）	識別子，および定数の構成
_（アンダースコア）	識別子の構成
.（ピリオド）	識別子，および定数の構成
数字	識別子，および定数の構成
:（コロン）	ラベルの終わり
,（カンマ）	オペランドの区切り
-（ハイフン）	負符号，減算演算子，およびコメントの開始
#	ラベルの絶対アドレス参照，およびコメントの開始
;（セミコロン）	文の終わり
'（クォーテーション）	文字定数の開始と終了
"（ダブルクォーテーション）	文字列定数の開始と終了
$	ラベルの gp オフセット参照
[ ]	ベース・レジスタの指定
+	加算演算子
*	乗算演算子
/	除算演算子
%	ラベルのセクション内オフセット参照（命令展開なし），および剰余演算子
<<	左シフト演算子
>>	右シフト演算子
!	ラベルの絶対アドレス参照（命令展開なし），および否定演算子
&	論理積演算子
|	論理和演算子
^	排他的論理和演算子
()	演算順序の指定

1.1.5 アセンブリ言語文の例

アセンブリ言語プログラムの簡単な例は，次のようになります。

```
# sample program
       .extern __tp_TEXT, 4
       .extern __gp_DATA, 4
       .extern _main
       .section "RESET", text     -- Reset Handler address
       jr     __boot              -- Jump to __boot
       .text                      -- Text section
       .align 4                   -- Code alignment
       .globl __boot              -- Alignment
__boot:
       mov    #__tp_TEXT, tp      -- Set tp
       mov    #__gp_DATA, gp      -- Set gp

       .extern __ssbss, 4
       .extern __esbss, 4

# start of bss initialize
       mov    #__ssbss, r13
       mov    #__esbss, r13
       cmp    r12, r13
       jnl    sbss_init_end
sbss_init_loop:
       st.w   r0, 0[r13]
       add    4, r13
       cmp    r12, r13
       jl     sbss_init_loop
sbss_init_end:
# end of bss initialize

       jarl   _main, lp           -- Call main function
       .data
       .align 4
data_area:
       .word 0x00                 -- data1
       .hword 0x01                -- data2
       .byte 0xff ; .byte 0xfe    -- data3, data4
```

1.2 アセンブリ言語プログラムの構成

1.2.1 シンボル

シンボルとは，値（整数値）を持った名前のことをいい，ユーザが定義します。シンボルを定義するときには，“ .set 疑似命令 ” を使用します。

```
.set sym1, 0x10          -- sym1 は 0x10 という値を持つシンボル
mov  sym1, r10           -- sym1 の値 (0x10) をレジスタに格納
```

as850 では，ファイルの先頭から最初の .set 疑似命令までの間に出現したシンボル参照を “ その時点において未定義なシンボル参照 ” とし，定義済みのシンボル参照とは区別して扱われます（「1.2.6 式」の「(1) 絶対値式」も参照してください）。

(1) シンボルとして使用できる文字

シンボルとして使用できるのは「1.1.4 文字セット」で示したうち，次の文字になります。

- 英小文字
- 英大文字
- _（アンダースコア）
- .（ピリオド）
- 数字

ただし，シンボルの先頭に数字は使用できません。数字で始まるシンボルを指定した場合，次のメッセージが出力され，アセンブルが中止されます。

```
E3249: illegal syntax
```

また，「1.2.4 予約語」で示す予約語をラベルとして使用することもできません。

注意 “ _（アンダースコア）” で始まるシンボルは，コンパイラにより出力されたラベル名と一致する可能性があり，予期せぬ動作を招く可能性があるので注意してください。なお “ .（ピリオド）” で始まるシンボルが，今後予約語となる可能性がありますので，なるべく使用しないでください。

(2) シンボルの最大文字数と最大個数

シンボルの最大文字数は 1037 文字です。1038 文字以上のシンボルが指定された場合，次のメッセージが出力され，アセンブルが中止されます。

```
E3260: token too long
```

定義可能なシンボルの最大個数は，利用可能なメモリ領域の大きさに依存します。

1.2.2 ラベル

ラベルとは，プログラムの任意の行に記述できる名前のことをいい，ユーザが定義します。ラベルの定義方法，宣言には次のものがあります。

(1) ラベルの定義方法

ラベルの定義方法には 2 つあります。

(a) 文の先頭で名前の後に “ : ” を続けることにより，ローカルなラベルとして定義

```
label1:
```

ローカルなラベル定義方法としては，この方法が一般的で，以後 “ 通常のラベル定義 ” と呼びます。

(b) .lcomm 疑似命令により，ローカルなラベルとして定義

```
        .lcomm label1, 0x100, 4
```

上記の意味は「4 バイトでアラインされたアドレスから，サイズ “ 0x100 バイト ” 分を確保し，その領域の先頭ラベルを “ label1 ” とする」という意味です。

(2) ラベルの宣言

ラベルの宣言には 4 つあります。

(a) .comm 疑似命令により，未定義外部ラベルとして宣言

```
        .comm label1, 4, 4
```

上記の意味は「サイズ “ 4 バイト ” で，整列条件 4 バイトの未定義外部ラベル “ label1 ” を宣言する」という意味です。

(b) .extern 疑似命令により，外部ラベルとして宣言（指定したファイル内に定義を持たないラベル）

```
        .extern label1
```

(c) .globl 疑似命令により，外部ラベルとして宣言（指定したファイル内に定義を持つラベル）

```
        .globl label1
```

(d) ファイル内に定義を持たせないことにより，外部ラベルとして宣言

```
        .mov label1, r10
```

上記で，label1 の定義が同一ファイル内にない場合，label1 は外部ラベルとみなされます。

(3) ラベルとして使用できる文字

ラベルとして使用できるのは「1.1.4 文字セット」で示したうち，次の文字になります。

- 英小文字
- 英大文字
- _（アンダースコア）
- .（ピリオド）
- 数字

ただし，ラベルの先頭に数字は使用できません。数字で始まるラベルを指定した場合，次のメッセージが出力され，アセンブルが中止されます。

```
E3249: illegal syntax
```

また，「1.2.4 予約語」で示す予約語をラベルとして使用することもできません。

注意 “_（アンダースコア）”で始まるラベルは，コンパイラにより出力されたラベル名と一致する可能性があり，予期せぬ動作を招く可能性があるので注意してください。なお“.（ピリオド）”で始まるラベルが，今後予約語となる可能性がありますので，なるべく使用しないでください。

(4) ラベルの最大文字数と最大個数

ラベルの最大文字数は 1037 文字です。1038 文字以上のラベルが指定された場合，次のメッセージが出力され，アセンブルが中止されます。

```
E3260: token too long
```

定義可能なラベルの最大個数は，利用可能なメモリ領域の大きさに依存します。

(5) sbss ／ bss 属性 セクションにおける通常のラベル定義

sbss ／ bss 属性セクションにおいて，通常のラベル定義を行った場合，次のメッセージが出力され，アセンブルが中止されます。

```
E3246: illegal section
```

このエラーが出力された場合は，.lcomm 疑似命令を使用して定義してください。

1.2.3 マクロ

マクロとは，一連の決まった手順パターンを登録し，それを利用して記述するものです。マクロはユーザが定義します。マクロの定義方法は次のように，マクロ本体を “ .macro ” と “ .endm ” で囲む形になります。

```
        .macro PUSH REG      -- 次の 2 つの文がマクロ本体
        add    -4, sp
        st.w   REG, 0x0[sp]
        .endm
```

上記を定義したあと，次のように記述した場合，「r19 をスタックに格納する」というコードに置き換えられます。

```
        PUSH   r19
```

したがって，次のようなコードに展開されます。

```
        add    -4, sp
        st.w   r19, 0x0[sp]
```

1.2.4 予約語

as850 には予約語が存在します。予約語をシンボル，ラベル，セクション名に使用することはできません。予約語を指定した場合は，次のメッセージが出力され，アセンブルが中止されます。

```
E3245: identifier is reserved word
```

予約語は次のとおりです。

- 命令（add，sub，mov など）
- 疑似命令（.section，.lcomm，.globl など）
- hi,lo,hi1(hi(),lo(),hi1() として使用しているため「2.2.7 hi() ／ lo() ／ hi1() について」を参照してください）
- レジスタ名

1.2.5 定数

as850 では “ 数値定数 ”, “ 文字定数 ”, および “ 文字列定数 ” を定数として扱うことができます。

(1) 数値定数

数値定数には “ 整定数 ” と “ 浮動小数点数 ” があります。

(a) 整定数

整定数，つまり，整数の定数は，32 ビット幅をもつ定数です。負の数は 2 の補数で表現されます。32 ビットで表現することのできる値を越える整数値が指定された場合，as850 はその整数値の下位 32 ビットの値を用いて処理を続行します（メッセージ等は出力しません）。

(i) 2 進定数

2進定数は,“ 0b ”,または“ 0B ”と,その後ろに続く 1 個以上の“ 0 ”,または“ 1 ”の数字の並びで構成されます。

例

```
0b00010110111101010111111010010111
```

(ii) 8 進定数

8 進定数は，“ 0 ” と，その後ろに続く 1 個以上の “ 0 ” から “ 7 ” までの数字の並びで構成されます。

例

```
02675277227
```

(iii) 10 進定数

10 進定数は，“ 0 ” 以外の数字で始まる 1 個以上の数字の並びで構成されます。

例

```
385187479
```

(iv) 16 進定数

16 進定数は，“ 0x ”，または “ 0X ” と，その後ろに続く 1 個以上の “ 0 ” から “ 9 ” までの数字，“ a ” から “ f ” までの英小文字，または “ A ” から “ F ” までの英大文字の並びで構成されます。

例

```
0x16f57e97
```

(b) 浮動小数点数

浮動小数点数は 32 ビット幅を持ちます。浮動小数点数は，次に示す要素で構成されます。

(i) 仮数部の符号（“ + ” は省略可）
(ii) 仮数値
(iii) 指数部を示す “ e ”，または “ E ”
(iv) 指数部の符号（“ + ” は省略可）
(v) 指数値

指数値，および仮数値は 10 進定数で指定します。ただし，指数表現を用いない場合，(iii)，(iv)，および (v) は用いられません。

例

```
123.4
-100.
10e-2
-100.2E+5
```

なお，仮数値の先頭に“ 0f ”または“ 0F ”を置くことにより，浮動小数点数であることを示すこともできます。たとえば，as850 では“ 10 ”は整定数であるとみなされますが，“ 0f10 ”は浮動小数点数であるとみなされます。また，“ 060 ”のように“ 0 ”で始まり，小数点を持たない数字列を指定することはできません（“ 0 ”のみは指定できます）。

(c) 文字定数

文字定数は，1 つの文字をシングル・クォート“ ' ”で囲むことにより構成され，囲まれた文字の値を示します注。表 1 － 2 に示したエスケープ・シーケンスを“ ' ”と“ ' ”の間に指定した場合，as850 では，1 つの文字を表すものとして扱われます。

例

```
'a'
'\0'
'\012'
'\x0a'
```

注 文字定数を指定した場合，as850 では，その文字定数の値を持つ整定数を指定したものとみなして扱われます。

表 1 － 2 エスケープ・シーケンスの値と意味

エスケープ・シーケンス	値	意味
\0	0x00	null 文字
\a	0x07	アラート
\b	0x08	バックスペース
\f	0x0c	フォーム・フィード
\n	0x0a	ニューライン
\r	0x0d	キャリッジ・リターン
\t	0x09	水平タブ
\v	0x0b	垂直タブ
\\	0x5c	バックスラッシュ
\'	0x27	シングル・クォート
\"	0x22	ダブル・クォート
\?	0x3f	疑問符
\ddd	0 ～ 0377	3 桁までの 8 進数（0 ≤ d ≤ 7）注
\xhh	0 ～ 0xff	2 桁までの 16 進数 （0 ≤ h ≤ 9，a ≤ h ≤ f，または A ≤ h ≤ F）

注 “ \377 ” を越える指定の場合，そのエスケープ・シーケンスの値は，下位 1 バイト分のみの値となります。0377 より大きい値にはなりません。たとえば “ \777 ” の値は 0377 です。

(2) 文字列定数

文字列定数は，文字列のダブル・クォート “ " ” で囲むことにより構成され，囲まれた文字列を示します。表 1 － 2 に示したエスケープ・シーケンスを “ " ” と “ " ” の間に指定した場合，as850 では，1 つの文字を表すものとして扱われます。したがって，“ \ddd ” 形式のエスケープ・シーケンスで，“ 0 ” から “ 7 ” までの数字以外のものを続けた場合，その直前までが “ \ddd ” 形式のエスケープ・シーケンスとして扱われます。

例

"abc"	‘ a ’, ‘ b ’, ‘ c ’
"ABC\n"	‘ A ’, ‘ B ’, ‘ C ’, ‘ \n ’
"\033abc\t\0"	‘ \033 ’, ‘ a ’, ‘ b ’, ‘ c ’, ‘ \t ’, ‘ \0 ’
"\12345"	‘ \123 ’, ‘ 4 ’, ‘ 5 ’
"\12845"	‘ \12 ’, ‘ 8 ’, ‘ 4 ’, ‘ 5 ’

1.2.6 式

式は“定数”,“シンボル”,“ラベルの参照”,“演算子”，および“かっこ”を要素とし，これらの要素で構成される値を示します。as850 では，絶対値式と相対値式に分けて扱います。

(1) 絶対値式

定数値を示す式を“絶対値式”と呼びます。絶対値式は，命令においてオペランドを指定する場合，または疑似命令において値／サイズ／整列条件／フィリング値／ビット幅を指定する場合に用いることができます。通常，絶対値式は，定数，またはシンボル（「2.2.3 シンボル」参照）によって構成されます。as850 では，次に示した形式が絶対値式として扱われます。ただし，ビット幅の指定を持たない .byte ／ .hword ／ .shword【V850E】／ .word 疑似命令，および .frame 疑似命令以外の疑似命令に対しては，“定数式”形式以外の絶対値式は指定できません（ビット幅の指定を持たない .byte ／ .hword ／ .shword【V850E】／ .word 疑似命令に対しては，値の指定において次に示したすべての形式の絶対値式が指定でき，.frame 疑似命令に対してはサイズの指定において“定数式”形式以外に“シンボル”形式の絶対値式が指定できます）。

(a) 定数式

例

```
.set sym1, 0x100     -- シンボル sym1 を定義

mov    sym1, r10     -- 定義済みの sym1 は定数式として扱う
```

定義済みのシンボル参照を指定した場合，as850 では，そのシンボルに対して定義した値の定数が指定されたものとして扱われます。したがって，定数式は，定義済みのシンボル参照を，その構成要素として持つことができます。

(b) シンボル

シンボルに関する式には，次のものがあります（“±”は“＋”か“－”のどちらかになります）。

- シンボル
- シンボル±定数式
- シンボル－シンボル
- シンボル－シンボル±定数式

ここで言う“シンボル”とは，その時点において未定義なシンボル参照を指します。定義済みのシンボル参照を指定した場合は，as850 はそのシンボルに対して定義した値の“定数”が指定されたものとして扱います。

例

```
add    SYM1 + 0x100, r11    -- この時点で SYM1 は未定義シンボル

.set   SYM1, 0x10           -- SYM1 を定義
```

(c) ラベル参照

ラベル参照に関する式には，次のものがあります（“ ± ”は“ ＋ ”か“ － ”のどちらかになります）。

- ラベル参照－ラベル参照
- ラベル参照－ラベル参照±定数式

ラベル参照に関する式の例は，次のようになります。

例

```
        mov $label1-$label2, r11
```

この例のような“ 2 つのラベル参照 ”は，次のように参照にする必要があります。

- 指定したファイル内で，同じセクションに定義をもつ
- 同じ参照方法（$label は $label 同士，#label は #label 同士など）
- 指定したファイル内に定義を持たないラベル参照が指定された場合は，次のメッセージが出力され，アセンブルが中止されます。

```
E3209: illegal expression (labels must be defined)
```

同じセクションに定義を持たない 2 つのラベル参照が指定された場合は，次のメッセージが出力され，アセンブルが中止されます。

```
E3209: illegal expression (labels in different sections)
```

同じ参照方法によらない 2 つのラベル参照が指定された場合は，次のメッセージが出力され，アセンブルが中止されます。

```
E3209: illegal expression (labels have different reference types)
```

ただし，現在のアセンブラの構成上，“ ラベル参照に関する式 ”の中の“ －ラベル参照 ”側のラベル参照に，指定されたファイル内に定義を持たないラベルの絶対アドレス参照が指定された場合，他方のラベル参照の参照方法と同じ参照方法が用いられたものとみなして扱われます。なお，分岐命令に対して，この形式の絶対値式は指定できません。指定した場合は，次のメッセージが出力され，アセンブルが中止されます。

```
E3221: illegal operand (label-label)
```

(2) 相対値式

特定のアドレスからのオフセット値[注 1] を示す式を“ 相対値式 ”と呼びます。相対値式は，命令においてオペランドを指定する場合，またはビット幅の指定を持たない .byte ／ .hword ／ .word 疑似命令において値を指定する場合に用いることができます。通常，相対値式は，ラベル参照（「2.2.4 ラベル参照」参照）によって構成されます。as850 では，次に示した形式[注 2] が相対値式として扱われます。

(a) ラベル参照

ラベル参照に関する式には，次のものがあります（“ ± ”は“ ＋ ”か“ － ”のどちらかになります）。

- ラベル参照
- ラベル参照±定数式
- ラベル参照－シンボル
- ラベル参照－シンボル±定数式

ラベル参照に関する式の例は次のようになります。

例

```
        add     #labe11 + 0x10, r10
        add     #label2 - SIZE, r10
        .set SIZE, 0x10
```

注 1 このアドレスは CA850 に含まれるリンカ（ld850）におけるリンク時に定められます。このため，このオフセットの値もリンク時に定められます。

注 2 （絶対値式に対しても同じことが言えますが）as850 では，“ - シンボル + ラベルの参照 ”の形式の式を“ ラベルの参照 - シンボル ”の形式の式とみなすことはできますが，“ ラベルの参照 - （+ シンボル）”の形式の式を“ ラベルの参照 - シンボル ”の形式の式とみなすことはできません。このため，括弧“ ( ) ”は定数式においてのみ用いるようにしてください。

1.2.7 演算子

演算子は，式における演算の指示に用いることができます。

(1) 演算子の種類

演算子は，“ 算術演算子 ”, “ シフト演算子 ”, “ ビット論理演算子 ”，および “ 比較演算子 ” の 4 つに分類できます。なお，“ - ” は，単項演算子としても二項演算子としても用いることができます。

表 1 － 3 演算子

種類	演算子
算術演算子	+ - * / %
シフト演算子	<< >>
ビット論理演算子	! | & ^
比較演算子	== < <= != > >= && ||

次の説明では，演算子の左側オペランドを第一オペランド，右側オペランドを第二オペランドと呼んでいます。また，単項演算子の場合は，単にオペランドと呼んでいます。

(a) 算術演算子

(i) +

第一オペランドと第二オペランドの和を求めます。

(ii) -

第一オペランドと第二オペランドの差を求めます。なお，単項演算子の場合はオペランドの 2 の補数を求めます。

(iii) *

第一オペランドと第二オペランドの積を求めます。

(iv) /

第一オペランドと第二オペランドの商を求めます。

(v) %

第一オペランドを第二オペランドで割った余りを求めます。

(b) シフト演算子

(i) <<

第一オペランドを第二オペランドで指定したビット数だけ左シフトします。なお，右側（LSB 注 1）には，シフトしたビット数分だけ 0 を入れます。

例

`0x12345678 << 4`	`0x23456780`

(ii) >>

第一オペランドを，第二オペランドで指定したビット数だけ右シフトします。なお，左側（MSB[注2]**）には，第一オペランドが正（**MSB **が** 0**）の場合は，シフトしたビット数分だけ** 0 **を，第一オペランドが負（**MSB **が** 1**）の場合は，シフトしたビット数分だけ** 1 **を入れます。**

例

0x12345678 >> 4	0x01234567
0x87654321 >> 4	0xF8765432

注 1 Least Significant Bit**（最も下位の桁に対応するビット）の略です。**

注 2 Most Significant Bit**（最も上位の桁に対応するビット）の略です。**

(c) **ビット論理演算子**

(i) !

オペランドのビットごとの論理否定を求めます。

例

!0x12345678	0xEDCBA987

(ii) |

第一オペランドと第二オペランドの論理和を求めます。

例

0x1234 | 0x5678	0x567C

(iii) &

第一オペランドと第二オペランドの論理積を求めます。

例

0x1234 & 0x5678	0x1230

(iv) ^

第一オペランドと第二オペランドの排他的論理和を求めます。

例

0x1234 ^ 0x5678	0x444C

(d) **比較演算子**

(i) ==

第一オペランドと第二オペランドを比較し，等しい場合は 1 **を返し，等しくない場合は** 0 **を返します。**

例

1 == 1	1
1 == 0	0

(ii) <

第一オペランドと第二オペランドを比較し，第一オペランドが第二オペランドより小さい場合は 1 を返し，第一オペランドが第二オペランドより大きい，または等しい場合は 0 を返します。

例

`1 < 10`	`1`
`10 < 1`	`0`

(iii) <=

第一オペランドと第二オペランドを比較し，第一オペランドが第二オペランドより小さいか等しい場合は 1 を返し，第一オペランドが第二オペランドより大きい場合は 0 を返します。

例

`1 <= 1`	`1`
`1 <= 2`	`1`
`1 <= 0`	`0`

(iv) !=

第一オペランドと第二オペランドを比較し，等しくない場合は 1 を返し，等しい場合は 0 を返します。

例

`1 != 0`	`1`
`1 != 1`	`0`

(v) >

第一オペランドと第二オペランドを比較し，第一オペランドが第二オペランドより大きい場合は 1 を返し，第一オペランドが第二オペランドより小さい，または等しい場合は 0 を返します。

例

`1 > 0`	`1`
`1 > 2`	`0`

(vi) >=

第一オペランドと第二オペランドを比較し，第一オペランドが第二オペランドより大きい，または等しい場合は 1 を返し，第一オペランドが第二オペランドより大きい場合は 0 を返します。

例

`1 >= 1`	`1`
`1 >= 0`	`1`
`1 >= 2`	`0`

(vii) &&

第一オペランドの論理値と第二オペランドの論理値の論理積を求めます。

例

`1 != 3 && 1 <= 3`	`1`
`1 == 1 && 1 != 1`	`0`
`1 != 1 && 3 <= 1`	`0`

(viii) ||

第一オペランドの論理値と第二オペランドの論理値の論理和を求めます。

例

`1 != 3 || 1 <= 3`	`1`
`1 == 1 || 1 != 1`	`1`
`1 != 1 || 3 <= 1`	`0`

(2) 演算子の優先順位

次表に，演算子の優先順位を示します。なお，優先順位の等しい演算子が隣り合っている場合，一方の側がかっこで囲まれている場合には，そのかっこで囲まれた方が先に実行されますが，どちちもかっこで囲まれていない場合，またはどちらもかっこで囲まれている場合には，左側にあるものが先に実行されます[注]。

注　ただし，かっこは定数式においてのみ用いるようにしてください（「1.2.6 式」参照）。

表 1 － 4 演算子の優先順位

優先順位	演 算 子
高い ↑ ↓ 低い	- ! （単項演算子） * / << >> % & | ^ + - == < <= != > >= && ||

(3) 演算規則

as850 の演算規則を，次に示します[注]。

注 ただし，その時点において未定義なシンボルの参照，またはラベルの参照を含む式に関しては「1.2.6 式」に示した規則を優先します。

(a) 単項演算

単項演算子のオペランドに，絶対値式以外のものは指定できません。また，単項演算子！のオペランドに，浮動小数点数値を扱う式は指定できません。

(b) 二項演算

二項演算子のオペランドに指定できる整数値式の組み合わせを，表 1 － 5 に示します。なお，表内では，演算子とオペランドの組み合わせによって構成される式について，次に示す記号を使用しています。

abs	絶対値式
rel	“ 指定したファイル内に定義を持つラベルを参照する ” 相対値式
ext	“ 指定したファイル内に定義を持たないラベルを参照する ” 相対値式
×	as850 では，その演算子とオペランドの組み合わせが許されない

ただし，浮動小数点数値に関しては浮動小数点数値同士の演算でなければならず，浮動小数点数値と相対値式を同一式内に混在させることはできません。

表 1 － 5 二項演算における演算規則

オペランド		演算子											
		+			-			*，/			その他		
第二オペランド		abs	rel	ext	abs	rel	ext	abs	rel	ext	abs	rel	ext
第一オペランド	abs	abs	rel	ext	abs	×	×	abs	×	×	abs	×	×
	rel	rel	×	×	rel	abs [注]	×	×	×	×	×	×	×
	ext	ext	×	×	ext	×	×	×	×	×	×	×	×

注 この部分の詳細に関しては，「1.2.6 式」を参照してください。

1.3 識別子

識別子とは，シンボル，ラベル，マクロ名などに使用する名前です。識別子として使用できるのは「1.1.4 文字セット」で示した文字のうち，次の文字になります。

- 英小文字
- 英大文字
- _（アンダースコア）
- .（ピリオド）
- 数字

ただし，名前の先頭に数字は使用できません。また，“_（アンダースコア）”で始まる識別子は，コンパイラにより出力されたラベル名と一致する可能性があり，予期せぬ動作を招く可能性があるので注意してください。なお，“.（ピリオド）”については，予約語となる可能性があるため，使用を避けてください。

第 2 章　命令セット

この章では，CA850 アセンブラ（as850）がサポートする命令セットについて説明します。

2.1　記号の説明

次表に，この章以降で用いる記号の意味を示します。

表 2 － 1 記号の意味

記号	意味
CMD	命令
CMDi	命令（andi，ori，または xori）
reg，reg1，reg2	レジスタ
r0	ゼロ・レジスタ
r1	アセンブラ予約レジスタ
gp	グローバル・ポインタ（r4）
ep	エレメント・ポインタ（r30）
[reg]	ベース・レジスタ
disp	ディスプレースメント（アドレスからの偏位） 特に記述のない場合 32 ビット幅を持ちます。
imm	イミーディエト（即値） 特に記述のない場合 32 ビット幅を持ちます。
bit#3	ビット・ナンバ指定用 3 ビット・データ
#label	ラベルの絶対アドレス参照
label	ラベルのセクション内オフセット参照，または PC オフセット参照 ただし，tp シンボルの生成において対象となっているセグメントに割り当てられたセクションに対しては，セクション内オフセットの代わりに tp シンボルからのオフセット参照
$label	ラベルの gp オフセット参照
!label	ラベルの絶対アドレス参照（命令展開なし）
%label	ラベルのセクション内オフセット参照（命令展開なし）
hi(*value*)	*value* の上位 16 ビット
lo(*value*)	*value* の下位 16 ビット
hi1(*value*)	*value* の上位 16 ビット + *value* のビット番号 15 のビット値[注]
addr	アドレス

表 2 － 1 記号の意味

記号	意味
PC	プログラム・カウンタ
PSW	プログラム・ステータス・ワード
regID	システム・レジスタ番号（0 ～ 31）
vector	トラップ・ベクタ（0 ～ 31）
BITIO	周辺 I/O レジスタ（1 ビット操作専用）

注 LSB（Least Significant Bit）はビット番号 0 です。

2.2 オペランド

この節では，as850 におけるオペランドの記述形式について説明します。as850 では，命令，および疑似命令に対するオペランドとして，レジスタ，定数，シンボル，ラベル参照，および定数，シンボル，ラベル参照，演算子（「1.2.7 演算子」参照），かっこで構成した式（「1.2.6 式」参照）を指定できます。

2.2.1 レジスタ

as850 において指定できるレジスタを次に示します[注]。

注 PSW，およびシステム・レジスタは，ldsr ／ stsr 命令において，番号で指定します。なお，as850 では，PC をオペランドに指定する方法はありません。

```
r0，zero，r1，r2，hp，r3，sp，r4，gp，r5，tp，r6，r7，r8，r9，
r10，r11，r12，r13，r14，r15，r16，r17，r18，r19，
r20，r21，r22，r23，r24，r25，r26，r27，r28，r29，
r30，ep，r31，lp
```

r0 と zero（ゼロ・レジスタ），r2 と hp（ハンドラ・スタック・ポインタ），r3 と sp（スタック・ポインタ），r4 と gp（グローバル・ポインタ），r5 と tp（テキスト・ポインタ），r30 と ep（エレメント・ポインタ），r31 と lp（リンク・ポインタ）は同じレジスタを示します。

(1) r0

r0 は，常に 0 の値を持つレジスタです。したがって，デスティネーション・レジスタとして指定した場合にも，結果の代入は行われません。なお，r0 をデスティネーション・レジスタとして指定した場合，次のメッセージが出力され[注]，アセンブルが続行されます。

注 このメッセージの出力は，as850 の起動時に警告メッセージ抑止オプション（-w）を指定することにより抑止できます。

```
        mov    0x10, r0
               ↓
W3013: register r0 used as destination register
```

(a) ターゲット・デバイスに V850Ex を使用する場合，次の命令で r0 をデスティネーション・レジスタとして指定すると，警告メッセージではなくエラー・メッセージが出力されます。

dispose，divh 命令の形式 (1)，および (2)，ld.bu，ld.hu，
mov 命令の形式 (2)，movea，movhi，mulh，
mulhi，satadd，satsub，satsubi，satsubr，
sld.bu，sld.hu

```
        divh   r10, r0
               ↓
E3240: illegal operand (can not use r0 as destination in V850E mode)
```

(b) 次の命令で，ターゲット・デバイスに V850Ex を使用する場合，次の命令で r0 をソース・レジスタとして指定すると，警告メッセージではなくエラー・メッセージが出力されます。

divh 命令の形式 (1)， switch

```
        divh   r0, r10
               ↓
E3239: illegal operand (can not use r0 as source in V850E mode)
```

(2) r1

アセンブラ予約レジスタ（r1）は，as850 において，命令展開を行う際のテンポラリ・レジスタとして用いられるレジスタです。なお，r1 をソース・レジスタ，またはデスティネーション・レジスタとして指定した場合，次のメッセージが出力され[注]，アセンブルが続行されます。

注 このメッセージの出力は，as850 の起動時に警告メッセージ抑止オプション（-w）を指定することにより抑止できます。

```
        mov    0x10, r1
               ↓
W3013: register r1 used as destination register
```

```
        mov    r1, r10
               ↓
W3013: register r1 used as source register
```

2.2.2 定数

as850 では，命令，および疑似命令のオペランド指定で使用可能な絶対値式，または相対値式の構成要素として，整定数，および文字定数を用いることができます。また，ld ／ st 命令，およびビット操作命令のオペランド指定には，各デバイス・ファイルで定義されている「周辺 I/O レジスタ名」も指定できます。これにより，ポート・アドレスに対する入出力を行うことができます。また，.float 疑似命令のオペランド指定には浮動小数点定数を，.str 疑似命令のオペランド指定には文字列定数を用いることができます。

2.2.3 シンボル

as850 では，命令，および疑似命令のオペランド指定で使用可能な絶対値式，または相対値式の構成要素として，シンボルを用いることができます。

2.2.4 ラベル参照

as850 では，次に示した命令／疑似命令のオペランド指定で，使用可能な相対値式の構成要素として，ラベル参照を用いることができます。

- メモリ参照命令（ロード／ストア命令，およびビット操作命令）
- 演算命令（算術演算命令，飽和演算命令，および論理演算命令）
- 分岐命令
- 領域確保疑似命令（ただし，.word ／ .hword ／ .byte 疑似命令のみ）

as850 では，ラベル参照は参照方法の違い，およびそのラベル参照を用いている命令／疑似命令の違いにより次に示すように異なる意味を持ちます。

表 2 － 2 ラベル参照

参照方法	用いている命令	意味
#label	メモリ参照命令，演算命令，jmp 命令	ラベル label 定義の存在する位置の絶対アドレス（アドレス 0 からのオフセット[注 1]）。 32 ビットのアドレスを持ち，必ず 2 命令に展開。
	領域確保疑似命令（.word ／ .hword ／ .byte）	ラベル label 定義の存在する位置の絶対アドレス（アドレス 0 からのオフセット[注 1]）。 ただし，32 ビットのアドレスを，確保した領域の大きさに準じてマスクした値。
label	メモリ参照命令，演算命令	ラベル label 定義の存在する位置のセクション内オフセット（ラベル label 定義の存在するセクションの先頭アドレスからのオフセット[注 2]）。 32 ビットのオフセットを持ち，必ず 2 命令に展開。 ただし，tp シンボルの生成において対象となっているセグメントに割り当てられたセクションに対しては，tp シンボルからのオフセット参照。
	jmp 命令を除く分岐命令	ラベル label 定義の存在する位置の PC オフセット（ラベル label 参照を用いている命令の先頭アドレスからのオフセット[注 2]）。
	領域確保疑似命令（.word ／ .hword ／ .byte）	ラベル label 定義の存在する位置のセクション内オフセット（ラベル label 定義の存在するセクションの先頭アドレスからのオフセット[注 2]）。 ただし，32 ビットのオフセットを，確保した領域の大きさに準じてマスクした値。
$label	メモリ参照命令，演算命令	ラベル label 定義の存在する位置 gp オフセット（グローバル・ポインタの指すアドレスからのオフセット[注 3]）。

表 2 － 2 ラベル参照

参照方法	用いている命令	意味
!label	メモリ参照命令，演算命令	ラベル label 定義の存在する位置の絶対アドレス（アドレス 0 からのオフセット[注 1]）。 16 ビットのアドレスを持ち，16 ビット・ディスプレースメント，またはイミーディエトをもつ命令に指定した場合，命令展開は行われません。 その他の命令に指定した場合，適切な 1 命令に展開されます。 ラベル label の定義したアドレスが，16 ビットで表現できる範囲でない場合，リンク時にエラーになります。
	領域確保疑似命令 (.word ／ .hword ／ .byte)	ラベル label 定義の存在する位置の絶対アドレス（アドレス 0 からのオフセット[注 1]）。 ただし，32 ビットのアドレスを，確保した領域の大きさに準じてマスクした値。
%label	メモリ参照命令，演算命令	ラベル label 定義の存在する位置のセクション内オフセット（ラベル label 定義の存在するセクションの先頭アドレスからのオフセット[注 2]）。 16 ビットのオフセットを持ち，16 ビット・ディスプレースメント，またはイミーディエトをもつ命令に指定した場合，命令展開は行われません。 その他の命令に指定した場合，適切な 1 命令に展開されます。 ラベル label の定義したアドレスが，16 ビットで表現できる範囲でない場合，リンク時にエラーになります。 または，ラベル label 定義の存在する位置の ep オフセット（エレメント・ポインタの示すアドレスからのオフセット）。
	領域確保疑似命令 (.word ／ .hword ／ .byte)	ラベル label 定義の存在する位置のセクション内オフセット（ラベル label 定義の存在するセクションの先頭アドレスからのオフセット[注 2]）。 ただし，32 ビットのオフセットを，確保した領域の大きさに準じてマスクした値。

注 1 リンク後のオブジェクト・ファイルにおけるアドレス 0 からのオフセットです。

注 2 リンク後のオブジェクト・ファイルにおいて，ラベル label の定義の存在するセクションが割り当てられたセクション（出力セクション）の先頭アドレスからのオフセットです。

注 3 上記出力セクションが割り当てられたセグメントに対する，テキスト・ポインタ・シンボルの値＋グローバル・ポインタ・シンボルの値の指すアドレスからのオフセットです。

次に，メモリ参照命令，演算命令，分岐命令，および領域確保疑似命令におけるラベル参照の意味を示します。

表 2 － 3 メモリ参照命令

参照方法	意味
#label[reg]	ラベル label の絶対アドレスがディスプレースメントとして扱われます。 32 ビットの値を持ち，必ず 2 命令に展開されます。#label[r0] とすることにより絶対アドレスによる参照を指定できます。 [reg] の部分が省略でき，省略した場合，as850 では，[r0] が指定されたものとみなされます。
label[reg]	ラベル label のセクション内オフセットがディスプレースメントして扱われます。32 ビットの値を持ち，必ず 2 命令に展開されます。reg に対象とするセクションの先頭アドレスを指すレジスタを指定し，label[reg] とすることにより一般的なレジスタ相対の参照が指定できます。 ただし，tp シンボルの生成において対象となっているセグメントに割り当てられたセクションに対しては，tp シンボルからのオフセットをディスプレースメントとして扱います。
$label[reg]	ラベル label の gp オフセットがディスプレースメントとして扱われます。ラベル label の定義されたセクションにより，32，または 16 ビットの値を持ち，命令展開のパターンが変化します[注]。16 ビットの値を持つ命令展開が行われた場合，ラベル label の定義したアドレスより算出されたオフセットが 16 ビットで表現できる範囲でない場合，リンク時にエラーになります。$label[gp] とすることにより gp レジスタ相対の参照（gp オフセット参照と呼ぶ）が指定できます。[reg] の部分が省略でき，省略した場合，as850 では，[gp] が指定されたものとみなされます。
!label[reg]	ラベル label の絶対アドレスがディスプレースメントとして扱われます。16 ビットの値を持ち，命令展開は行われません。ラベル label に定義したアドレスが 16 ビットで表現できない場合，リンク時にエラーになります。!lable[r0] とすることにより絶対アドレスによる参照が指定できます。 [reg] の部分が省略でき，省略した場合は [r0] が指定されたものとみなされます。 ただし，#label[reg] 参照とは異なり，命令展開は行われません。
%label[reg]	ラベル label のセクション内オフセットがディスプレースメントとして扱われます。ラベル label が ep シンボルとなっているセクションに配置された場合には，ep シンボルからのオフセットをディスプレースメントとして扱われます。16 ビット，または命令によってはそれ以下の値を持ち，その範囲で表現できる値でない場合，リンク時にエラーになります。 [reg] の部分が省略でき，省略した場合，as850 では，[ep] が指定されたものとみなされます。

注 「2.2.6 gp オフセット参照」を参照してください。

表 2 － 4 演算命令

参照方法	意味
#label	ラベル label の絶対アドレスがイミーディエトとして扱われます。 32 ビットの値を持ち，必ず 2 命令に展開されます。
label	ラベル label のセクション内オフセットがイミーディエトとして扱われます。 32 ビットの値を持ち，必ず 2 命令に展開されます。 ただし，tp シンボルの生成において対象となっているセグメントに割り当てられたセクションに対しては，tp シンボルからのオフセットがイミーディエトとして扱われます。
$label	ラベル label の gp オフセットがイミーディエトとして扱われます。 ラベル label の定義されたセクションにより，32，または 16 ビットの値を持ち，命令のパターンが変化します注 1。16 ビットの値を持つ展開をされた場合，ラベル label の定義したアドレスより算出されたオフセットが 16 ビットで表現できる範囲でない場合，リンク時にエラーになります。
!label	ラベル label の絶対アドレスがイミーディエトとして扱われます。 16 ビットの値を持ち，イミーディエトとして 16 ビットの値を指定できるアーキテクチャの演算命令注 2 に指定した場合，命令展開は行われません。add，mov，および mulh 命令に指定した場合，適切な 1 命令に展開されます。それ以外の命令に指定することはできません。16 ビットで表現できる範囲でない場合，リンク時にエラーになります。
%label	ラベル label のセクション内オフセットがイミーディエトとして扱われます。 ラベル label が ep シンボルの対象となっているセクションに配置された場合には，ep シンボルからのオフセットをディスプレースメントとして扱われます。 16 ビットの値を持ち，イミーディエトとして 16 ビットの値を指定できるアーキテクチャの演算命令注 2 に指定した場合，命令展開は行われません。 ただし，label 参照とは異なり，命令展開は行われません。また，この参照方法は，イミーディエトとして 16 ビットの値を指定できるアーキテクチャの演算命令と，add，mov，および mulh 命令のみで指定できます。add，mov，および mulh 命令に指定した場合，適切な 1 命令に展開します。それ以外の命令に指定することはできません。 16 ビットで表現できる範囲でない場合，リンク時にエラーになります。

注 1 「2.2.6 gp オフセット参照」を参照してください。

注 2 イミーディエトとして 16 ビットの値を指定できる命令は，addi，andi，movea，mulhi，ori，satsubi，および xori です。

表 2 － 5 分岐命令

参照方法	意味
#label	jmp 命令において，ラベル label の絶対アドレスが飛び先アドレスとして扱われます。 32 ビットの値を持ち，必ず 3 命令に展開されます。
label	jmp 命令以外の分岐命令において，ラベル label の PC オフセットがディスプレースメントとして扱われます。 22 ビットの値を持ち，表現できない範囲である場合，リンク時にエラーになります。

表 2 － 6 領域確保疑似命令

参照方法	意味
#label !label	.word ／ .hword ／ .byte 疑似命令において，ラベル label の絶対アドレスを値として扱われます。 32 ビットの値を持ちますが，各疑似命令のビット幅に応じてマスクされます。
label %label	.word ／ .hword ／ .byte 疑似命令において，ラベル label の定義されたセクション内オフセットを値として扱われます。 32 ビットの値を持ちますが，各疑似命令のビット幅に応じてマスクされます。
$label	.word ／ .hword ／ .byte 疑似命令において，ラベル label の gp オフセットを値として扱われます。 32 ビットの値を持ちますが，各疑似命令のビット幅に応じてマスクされます。

2.2.5 ep オフセット参照

この項では，ep オフセット参照について説明します。CA850 では，明示的に内蔵 RAM に置かれるデータについては，基本的に，次のことが想定されています。

エレメント・ポインタ（ep）の指すアドレスからのオフセットによって参照する

なお，内蔵 RAM に置かれるデータは，次の 2 つに分けられます。

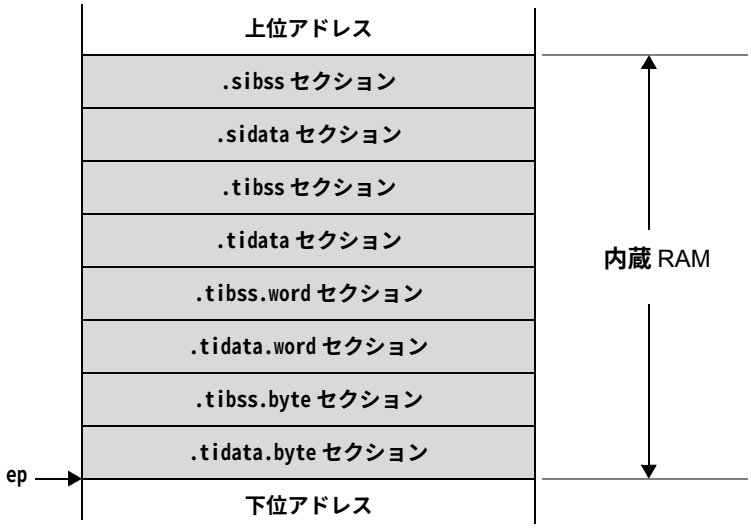
(1) .tidata ／ .tibss ／ .tidata.byte ／ .tibss.byte ／ .tidata.word ／ .tibss.word セクション

コード・サイズが小さいメモリ参照命令（sld ／ sst）で参照するデータ

(2) .sidata ／ .sibss セクション

コード・サイズが大きいメモリ参照命令（ld ／ st）で参照するデータ

図 2 － 1 内蔵 RAM のメモリ配置イメージ



(1) データの割り当て

内蔵 RAM に置くセクションへのデータの割り当ては，次の方法で行います。

(a) C 言語を用いてプログラムを作成する場合

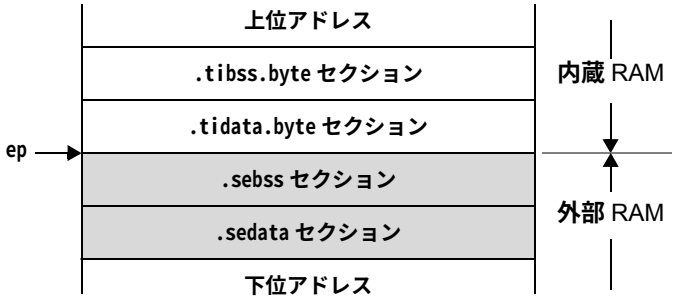
(i) “ #pragma section ”指令により，セクション種別“ tidata ”,“ tidata.byte ”,“ tidata.word ”，または“ sidata ” を指定してデータを割り当てます。

(ii) セクション・ファイルで，セクション種別 “ tidata ”, “ tidata.byte ”, “ tidata.word ”，または “ sidata ” を指定してデータを割り当てます。ca850 のオプションにより，そのセクション・ファイルをコンパイル時に入力します。

(b) アセンブリ言語を用いてプログラムを作成する場合

セクション定義疑似命令により，セクション種別 .tidata.byte，.tibss.byte，.tidata.word，.tibss.word，.sidata，または .sibss のセクションへデータを割り当てます。なお，上記と同様の方法により，.sedata，または .sebss セクションへデータを割り当てることで，外部 RAM に置かれる一定範囲のデータに対しても，ep オフセット参照を行うことができます。

図 2 － 2 外部 RAM（.sedata，／ .sebss セクション）のメモリ配置イメージ



(2) データの参照

(1) のデータの割り付けに従い，as850 では，次のように機械語命令列が生成されます。

(a) .tidata，.tibss，.tidata.byte，.tibss.byte，.tidata.word，.tibss.word，.sidata，.sibss，.sedata，または .sebss セクションに割り当てるデータの %label による参照に対しては，ep オフセット参照を行う機械語命令が生成

(b) 上記以外のセクションに割り当てるデータの %label による参照に対しては，セクション内オフセット参照を行う機械語命令列が生成

例

```
        .sidata
sidata:.hword 0xfff0
        .data
data:   .hword 0xfff0

        .text
        ld.h    %sidata, r20  -- (1)
        ld.h    %data, r20    -- (2)
```

as850 では，%label による参照に対して，(1) の場合，定義したデータが .sidata セクションに配置されているため ep オフセット参照とみなされ，(2) の場合，セクション内オフセット参照とみなされて，機械語の命令列が生成されます。なお，as850 では，データが配置されたセクションが正しいものとして処理が行われます。このため，データの配置に誤りがある場合でも，検出できません。

例

```
        .text
        ld.h    %label[ep], r20
```

label を .sidata セクションに配置し，ep オフセット参照を行う命令を記述したが，配置誤りにより .data セクションに配置された場合，ベース・レジスタである ep シンボル値 + label の .data セクション内オフセット値にあるデータがロードされます。

例

```
        .text
        ld.h    %label1[r10], r20    -- (i)
        .option ep_label
        ld.h    %label2[ep], r21     -- (ii)
        .option no_ep_label
        ld.h    %label3[r10], r22    -- (iii)
```

(i) 定義したデータが配置されたセクションによって，ep オフセット参照，またはセクション内オフセット参照となります（デフォルト）。

(ii) .option ep_label 疑似命令によって指定された範囲内であるので，定義したデータが配置されたセクションにかかわらず，ep オフセット参照となります。

(iii) .option no_ep_label 疑似命令によって指定された範囲内であるので，(i) の場合と同じ扱いとなります。

2.2.6 gp オフセット参照

この項では，gp オフセット参照について説明します。CA850 では，（.sedata，／ .sebss セクション以外の）外部 RAM に置かれるデータについては，基本的に次のことが想定されています。

グローバル・ポインタ（gp）の指すアドレスからのオフセットによって参照する

また，C 言語における“ #pragma section ”指令や，C コンパイラに入力するセクション・ファイル，アセンブリ言語におけるセクション定義疑似命令による，内蔵 ROM ／内蔵 RAM などの r0 相対のメモリ割り当てを行わない場合，すべてのデータが gp オフセット参照となります。

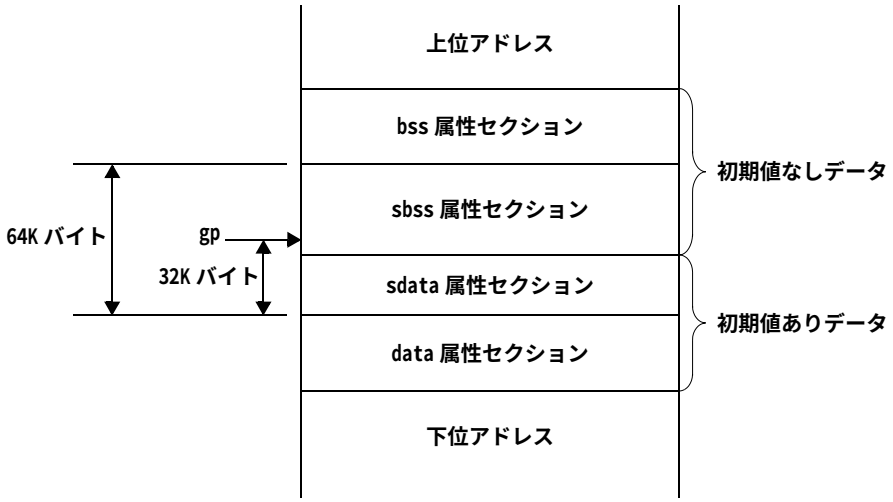
(1) データの割り当て

V850 マイクロコントローラの機械語命令におけるメモリ参照命令（ld ／ st）は，ディスプレースメントに 16 ビットのイミーディエトしかとれません。このため，CA850 ではデータを

(a) グローバル・ポインタ（gp）と 16 ビットのディスプレースメントを用いて参照することのできるメモリ範囲に割り当てるデータ

(b) グローバル・ポインタ（gp）と（複数の命令によって構成される）32 ビットのディスプレースメントを用いて参照することのできるメモリ範囲に割り当てるデータ の 2 つに分け，前者のデータが sdata 属性セクション，または sbss 属性セクションに，後者のデータが data 属性セクション，または bss 属性セクションに割り当てられます。なお，初期値ありデータは sdata ／ data 属性セクションに，初期値なしデータは sbss ／ bss 属性セクションに割り当てられます。CA850 では，デフォルトで，data ／ sdata ／ sbss ／ bss 属性セクションの順に下位アドレスから割り当てられます。また，グローバル・ポインタ（gp）は sdata 属性セクションの先頭アドレスに 32 K バイトを加算したアドレスを指すよう，スタート・アップ・モジュールなどにおいて設定することを想定しています。

図 2 － 3 gp オフセット参照セクションのメモリ配置イメージ



備考 sdata 属性セクションと sbss 属性セクションをあわせて 64 K バイトです。gp は sdata 属性セクションの先頭から 32 K バイトの位置です。

したがって，sdata ／ sbss 属性セクションのデータを参照する場合は，1 命令で行えますが，data ／ bss 属性セクションのデータを参照する場合は，複数の命令が必要となります。つまり，より多くのデータを sdata ／ sbss 属性セクションに割り当てたほうが，生成された機械語命令の実行効率，およびオブジェクト効率は向上します。しかし，16 ビットのディスプレースメントで参照できるメモリ範囲の大きさは限られています。

そこで，すべてのデータを sdata ／ sbss 属性セクションに割り当てられない場合，どのデータを sdata ／ sbss 属性セクションに割り当てるかを定めなければなりません。

CA850 では，“ できるだけ多くのデータを sdata ／ sbss 属性セクションに割り当てる ” という方針がとられています。デフォルトの処理では，すべてのデータが sdata ／ sbss 属性セクションに割り当てられますが，割り当てられるデータを選別する場合，次の方法により行います。

(i) -G*num* オプションを指定する場合

C コンパイラ（ca850），またはアセンブラ（as850）起動時に -G*num* オプションを指定することにより，sdata ／ sbss 属性セクションへ *num* バイト以下のデータが割り当てられます。

(ii) プログラムでデータを割り当てるセクションを指定する場合

参照頻度の高いデータを，明示的に sdata ／ sbss 属性セクションへ割り当てます。アセンブリ言語の場合，セクション定義疑似命令で，C 言語の場合，#pragma section 指令で割り当てられます。

(iii) セクション・ファイルで指定する場合

C 言語の場合，セクション・ファイルでセクション種別 “ sdata ” を指定してデータを割り当てます。ca850 のオプションにより，そのセクション・ファイルをコンパイル時に入力します。

(2) データの参照

前述のデータの割り付けに従い，as850 では，次のようになります。

(a) sdata ／ sbss 属性セクションに割り当てるデータの gp オフセット参照に対しては，16 ビットのディスプレースメントを用いた参照を行う機械語命令が生成されます。

(b) data ／ bss 属性セクションに割り当てるデータの gp オフセット参照に対しては，（複数の機械語命令によって構成される）32 ビットのディスプレースメントを用いた参照を行う機械語命令列が生成されます。

例

```
        .data
data:   .word 0xfff00010            -- (1)

        .text
        ld.w    $data[gp], r20      -- (2)
```

as850 では，(1) で定義したデータを gp オフセット参照している (2) の ld.w 命令に対して，次の命令列に等価な機械語の命令列が生成されます[注]。

```
        movhi   hi1($data), gp, r1
        ld.w    lo($data)[r1], r20
```

注 hi1( ) ／ lo( ) に関しては，「2.2.7 hi() ／ lo() ／ hi1() について」を参照してください。

なお，as850 では，1 ファイルずつ処理が行われます。このため，指定したファイル内に定義を持つデータに対しては，そのデータがどの属性セクションに割り当てられるデータであるかを判断することができますが，指定したファイル内の定義を持たないデータに対しては判定できません。そこで，as850 では，“ 前述の割り当ての方針（一定サイズ以下のデータを sdata ／ sbss 属性セクションに割り当てる）が守られている ” ことを想定し，起動時に -G*num* オプションが指定された場合，次のように機械語命令が生成されます[注]。

注 .option 疑似命令のオプションで，data，または sdata が指定されたデータに対しては，サイズに関係なく .data セクション，.sdata セクションへ割り当てられるものとして扱われます。

(c) 指定したファイル内に定義を持たない，*num* バイト以下のデータの gp オフセット参照に対しては，16 ビットのディスプレースメントを用いた参照を行う機械語命令が生成されます。

(d) 指定したファイル内に定義を持たない，*num* バイトより大きいデータの gp オフセット参照に対しては，（複数の機械語命令によって構成される）32 ビットのディスプレースメントを用いた参照を行う機械語命令列が生成生成されます。

しかし，この判定を行うには，指定したファイル内に定義を持たない gp オフセット参照されているデータのサイズが，判定できなければなりません。そこで，アセンブリ言語を用いてプログラムを作成する場合，指定したファイル内に定義を持たないデータ（実際には指定したファイル内に定義を持たず gp オフセット参照されているラベル）に対し，.extern 疑似命令を用いて，サイズを指定してください。

例

```
        .extern data, 4             -- (1)

        .text
        ld.w    $data[gp], r20      -- (2)
```

as850 では，起動時に -G2 を指定した場合，(1) で宣言したデータを gp オフセット参照している (2) の ld.w 命令に対しては，次の命令列に等価な機械語の命令列が生成されます[注]。

```
        movhi   hi1($data), gp, r1
        ld.w    lo($data)[r1], r20
```

注　hi1( ) ／ lo( ) に関しては，「2.2.7 hi() ／ lo() ／ hi1() について」を参照してください。

また，C 言語を用いてプログラム作成する場合，CA850 に含まれる C コンパイラ（ca850）が，指定したファイル内に定義を持たないデータ（実際には指定したファイル内に定義を持たず gp オフセット参照されているラベル）に対して，自動的に .extern 疑似命令を生成し，サイズを指定するコードを出力しています。

【まとめ】

as850 におけるデータの gp オフセット参照（具体的には，ラベルの gp オフセット参照を持つ相対値式をディスプレースメントとするメモリ参照命令）の扱いをまとめると，次のようになります。

(1) そのデータが，指定したファイル内に定義を持つ場合

(a) sdata ／ sbss 属性セクションに割り付けるデータである場合[注]

16 ビットのディスプレースメントを用いた参照を行う機械語命令が生成されます。

(b) sdata ／ sbss 属性セクションに割り付けるデータでない場合

32 ビットのディスプレースメントを用いた参照を行う機械語命令列が生成されます。

注 “ ラベル±定数式 ” の形式の相対値式で，定数式の値が 16 ビットの範囲を越える場合は，as850 では，32 ビットのディスプレースメントを用いた参照を行う機械語命令列が生成されます。

(2) そのデータが，指定したファイル内に定義を持たない場合

(a) 起動時に -G*num* オプションを指定した場合

データ（gp オフセット参照されているラベル）に対し，.comm ／ .extern ／ .globl ／ .lcomm ／ .size 疑似命令により，

[0 以外かつ *num* バイト以下を指定した場合]

sdata ／ sbss 属性セクションに割り当てるデータであるとみなされ，16 ビットのディスプレースメントを用いた参照を行う機械語命令が生成されます。

[上記以外の場合]

sdata ／ sbss 属性セクションに割り当てるデータでないとみなされ，32 ビットのディスプレースメントを用いた参照を行う機械語命令が生成されます。

(b) 起動時に -G*num* オプションを指定しなかった場合

sdata ／ sbss 属性セクションに割り当てるデータであるとみなされ，16 ビットのディスプレースメントを用いた参照を行う機械語命令が生成されます。

2.2.7 hi( ) ／lo( ) ／hi1( ) について

(1) 32 ビットの定数値をレジスタに入れる場合

V850 マイクロコントローラの V850 コアでは，1 命令で 32 ビットの定数値をレジスタに格納する機械語命令を持ちません。このため，as850 では，32 ビットの定数値をレジスタに格納する場合，命令展開を行い movhi 命令と movea 命令を用いて，32 ビットの定数値を上位 16 ビットと下位 16 ビットに分けてレジスタに格納する命令列が生成されます。

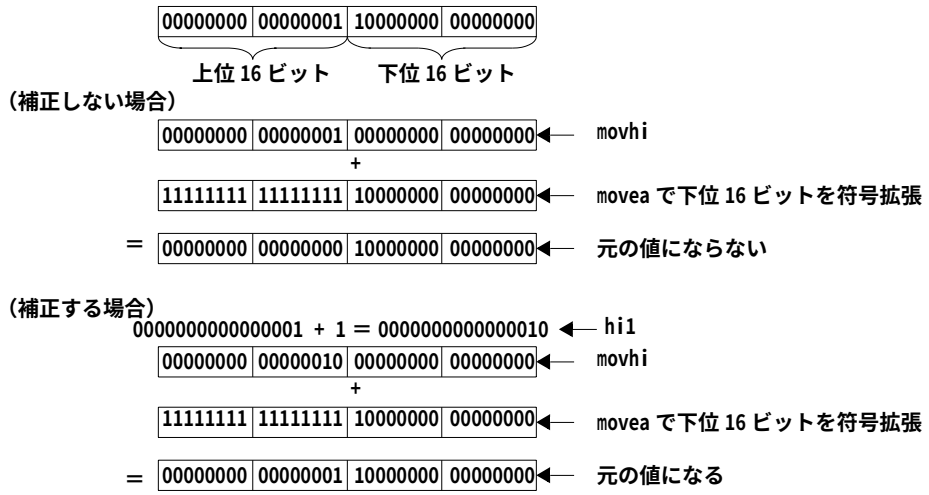
例

`mov 0x18000, r11`	`movhi hi1(0x18000), r0, r1` `movea lo(0x18000), r1, r11`

この際，下位 16 ビットの格納に用いられる機械語の movea 命令は，指定された 16 ビットの値を符号拡張し 32 ビットの値として扱われます。この符号拡張された部分を補正するため as850 では，movhi 命令を用いて上位 16 ビットをレジスタに格納する際，ただ単に上位 16 ビットをレジスタに格納するのではなく，

上位 16 ビット ＋ 下位 16 ビットの最上位ビット（ビット番号 15 のビット）

の値をレジスタに格納します。



(2) 32 ビットのディスプレースメントを用いてメモリを参照する場合

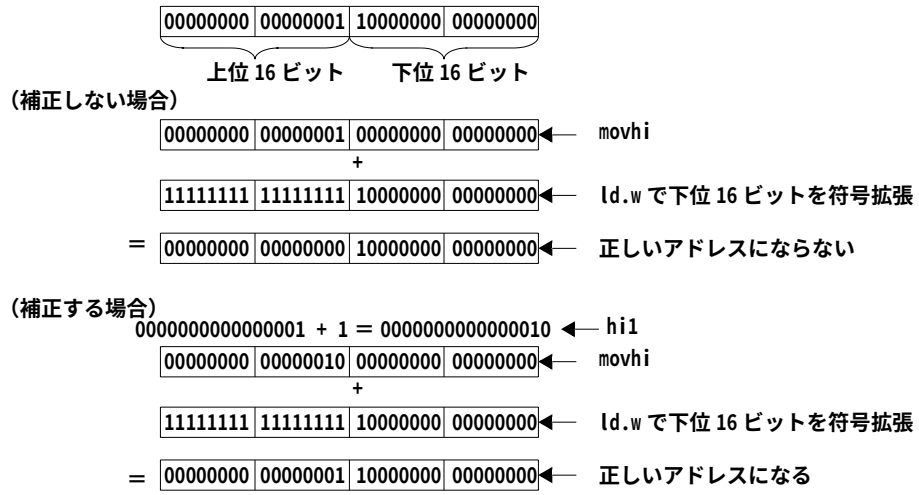
V850 マイクロコントローラの機械語のメモリ参照命令（ロード／ストア命令，およびビット操作命令）は，ディスプレースメントに 16 ビットのイミーディエトしかとれません。このため，as850 では，32 ビットのディスプレースメントを用いてメモリを参照する場合，命令展開が行われ，movhi 命令とメモリ参照命令が用いられて，32 ビットのディスプレースメントの上位 16 ビットと，下位 16 ビットから，32 ビットのディスプレースメントが構成されて参照を行う命令列が生成されます。

例

`ld.w  0x18000[r11], r12`	`movhi  hi1(0x18000), r11, r1` `ld.w  lo(0x18000)[r1], r12`

この際，下位 16 ビットをディスプレースメントとして用いる機械語のメモリ参照命令は，指定された 16 ビットのディスプレースメントを符号拡張し，32 ビットの値として扱います。この符号拡張された部分を補正するために，as850 では，movhi 命令を用いて上位 16 ビットのディスプレースメントを構成する際，ただ単に上位 16 ビットのディスプレースメントを構成するのではなく，次ののディスプレースメントを構成します。

上位 16 ビット ＋ 下位 16 ビットの最下位ビット（ビット番号 15 のビット）



(3) hi( ) ／ lo( ) ／ hi1( )

次表のように as850 では，hi( )，lo( )，および hi1( ) を用いることにより，32 ビットの値の上位 16 ビット，32 ビットの値の下位 16 ビット，および 32 ビットの値の上位 16 ビット + ビット番号 15 のビット値を指定できます注。

注　アセンブラ内部では解決できない場合，この情報はリロケーション情報に反映されリンカ（ld850）において解決されます。

表 2 － 7　hi( ) ／ lo( ) ／ hi1( ) の意味

hi( ) ／ lo( ) ／ hi1( )	意味
hi(*value*)	*value* の上位 16 ビット
lo(*value*)	*value* の下位 16 ビット
hi1(*value*)	*value* の上位 16 ビット + *value* のビット番号 15 のビット値

例

```
        .data
L1 :
              :
        .text
        movhi   hi($L1), r0, r10      -- L1 の gp オフセットの値の上位 16 ビットを r10 の上位
                                      -- 16 ビットに格納し，下位 16 ビットに 0 を格納する。

        movea   lo($L1), r0, r10      -- L1 の gp オフセットの値の下位 16 ビットを
                                      -- 符号拡張し，r10 に格納する。
              :

        movhi   hi1($L1), r0, r1      -- L1 の gp オフセットの値を r10 に格納する。
        movea   lo($L1), r1, r10
```

2.3 ランタイム・ライブラリ

V850 マイクロコントローラのアーキテクチャでは，浮動小数点演算の命令を持ちません。そこで CA850 では，ANSI 規格の言語仕様を満たすために，浮動小数点演算を，libc.a ファイルに含まれるランタイム・ライブラリから呼び出して行います。また，V850 マイクロコントローラのうち V850Ex 以外では，32 ビット・データの乗算，除算，剰余算についても命令を持たないため，浮動小数点演算と同様にランタイム・ライブラリから呼び出して行います。なお，ランタイム・ライブラリは，ca850 が C 言語ソース・プログラムをコンパイルする際に使用するルーチンですが，アセンブリ言語ソース・プログラムで使用することもできます。この場合，実行可能なオブジェクト・ファイル作成時に，ld850 で libc.a のリンク指定を行う必要があります。

2.4 マクロ・オペレータ

この節では，マクロ本体内において長さ 0 の区切り子として用いることができるチルダ記号 “ ~ ”，マクロ呼び出しにおいてシンボル値を引数として指定する場合に用いるダラー記号 “ $ ” について説明します。

2.4.1 チルダ記号

as850 では，マクロ本体内に現れたチルダ記号 “ ~ ”は長さ 0 の区切り子として扱われます。ただし，文字列定数やコメントの中に現れた場合，区切り子としては扱われず，通常のチルダ記号 “ ~ ” として扱われます。

例 1

```
.macro abc x
       abc~x: mov    r10, r20
              sub    def~x, r20
.endm

abc    NECEL
```

展開すると次のようになります。

```
abcNECEL:mov  r10, r20
       sub    defNECEL, r20
```

例 2

```
.macro abc x, xy
       a_~xy:mov     r10, r20
       a_~x~y:mov    r20, r10
.endm
abc    necel, NECEL
```

展開すると次のようになります。

```
a_NECEL:mov   r10, r20
a_necely:mov  r20, r10
```

例 3

```
.macro abc x, xy
       ~ab:mov       r10, r20
.endm
abc    necel, NECEL
```

展開すると次のようになります。

```
ab:mov        r10, r20
```

2.4.2 ダラー記号

as850 では，マクロ呼び出しにおける実引数としてダラー記号 “ $ ” を前に置いたシンボルを指定した場合，そのシンボルの値が実引数として指定されたものみなされます。ただし，ダラー記号 $ に続けてシンボル以外の識別名，または未定義シンボル名を指定した場合，次のメッセージが出力され，アセンブルが中止されます。

```
F3520: $ must be followed by defined symbol
```

例

```
.macro mac1 x
       mov    x, r10
.endm

.macro mac2
       .set   value, 10
       mac1   value
       mac1   $value
.endm

mac2
```

展開すると次のようになります。

```
       .set   value, 10
       mov    value, r10
       mov    10, r10
```

第 3 章　アセンブリ言語命令

この章では，CA850 アセンブラ（as850）がサポートするアセンブリ言語命令について説明します。

3.1　形式の説明

as850 がサポートするアセンブリ言語命令について，次の形式で説明します。なお，as850 が生成する機械語命令についての詳細は，V850 マイクロコントローラの各デバイスの“ユーザーズ・マニュアル アーキテクチャ編”を参照してください。

命令

〔概要〕

命令の意味を英語と日本語で示します。

〔形式〕

命令の形式を示します。

〔機能〕

命令の機能を示します。

〔説明〕

命令の動作の仕方を示します。

〔フラグ〕

命令実行によるフラグ（PSW）の動きを示します。ただし，clr1，not1，set1 命令では，実行前のフラグの値を示しています。なお，― は，フラグの値は変化しないことを示します。

〔注意事項〕

命令における注意事項を示します。

3.2 ロード／ストア命令

この節では，ロード／ストア命令について説明します。次に，この節において説明する命令を示します。

表 3 － 1 ロード／ストア命令

命令		意味
ld	ld.b	ロード（バイト）
	ld.bu	ロード（符号なしバイト）【V850E】
	ld.h	ロード（ハーフワード）
	ld.hu	ロード（符号なしハーフワード）【V850E】
	ld.w	ロード（ワード）
sld	sld.b	バイト・データのロード（ショート・フォーマット）
	sld.bu	符号なしバイト・データのロード（ショート・フォーマット）【V850E】
	sld.h	ハーフワード・データのロード（ショート・フォーマット）
	sld.hu	符号なしハードワード・データのロード（ショート・フォーマット）【V850E】
	sld.w	ワード・データのロード（ショート・フォーマット）
sst	sst.b	バイト・データのストア（ショート・フォーマット）
	sst.h	ハーフワード・データのストア（ショート・フォーマット）
	sst.w	ワード・データのストア（ショート・フォーマット）
st	st.b	バイト・データのストア
	st.h	ハーフワード・データのストア
	st.w	ワード・データのストア

ld

〔概要〕

データのロード（Load）

〔形式〕

(1) ld.b disp[reg1], reg2

(2) ld.h disp[reg1], reg2

(3) ld.w disp[reg1], reg2

(4) ld.bu disp[reg1], reg2【V850E】

(5) ld.hu disp[reg1], reg2【V850E】

ディスプレースメント（disp）に指定できるものを次に示します。

- 32 ビット幅までの値を持つ絶対値式
- 相対値式
- 上記のものに hi( )，lo( )，または hi1( ) を適用したもの

〔機能〕

ld.b，ld.h，ld.w，ld.bu，ld.hu 命令は，第一オペランドに指定したアドレスから 1 バイト分，1 ハーフワード分，および 1 ワード分のデータを取り込み，第二オペランドに指定したレジスタにロードします。

〔説明〕

- disp に次のものを指定した場合，as850 では，機械語命令の ld 命令[注]が 1 つ生成されます。なお，例中の“ld”は ld.b，ld.h，ld.w，ld.bu，ld.hu のいずれかになります。

(a) -32768 ～ +32767 の範囲の絶対値式

`ld disp16[reg1], reg2`	`ld disp16[reg1], reg2`

(b) sdata ／ sbss 属性セクションに定義を持つラベルの $label を持つ相対値式

`ld $label[reg1], reg2`	`ld $label[reg1], reg2`

(c) !label，または %label を持つ相対値式

`ld !label[reg1], reg2`	`ld !label[reg1], reg2`

`ld %label[reg1], reg2`	`ld %label[reg1], reg2`

(d) hi( ) ，lo( ) ，または hi1( ) を適用したもの

`ld disp16[reg1], reg2`	`ld disp16[reg1], reg2`

注 機械語命令の ld 命令は，ディスプレースメントに -32768 ～ +32767（0xffff8000 ～ 0x7fff）の範囲のイミーディエトをとります。

- disp に次のものを指定した場合，as850 では，命令展開が行われ，複数個の機械語命令が生成されます。

(a) -32768 ～ +32767 の範囲を越える絶対値式

`ld disp[reg1], reg2`	`movhi hi1(disp), reg1, r1` `ld lo(disp)[r1], reg2`

(b) #label，または label を持つ相対値式，および sdata ／ sbss 属性セクションに定義を持たないラベルの $label を持つ相対値式

`ld #label[reg1], reg2`	`movhi hi1(#label), reg1, r1` `ld lo(#label)[r1], reg2`

`ld label[reg1], reg2`	`movhi hi1(label), reg1, r1` `ld lo(label)[r1], reg2`

`ld $label[reg1], reg2`	`movhi hi1($label), reg1, r1` `ld lo($label)[r1], reg2`

- disp を省略した場合，as850 では，0 を指定したものとみなされます。
- disp に #label を持つ相対値式，または #label を持つ相対値式に hi( ) ，lo( ) ，または hi1( ) を適用したものを指定した場合，その後ろの [reg1] の部分が省略できます。ただし，省略した場合，as850 では，[r0] が指定されたものとみなされます。
- disp に $label を持つ相対値式，または $label を持つ相対値式に hi( ) ，lo( ) ，または hi1( ) を適用したものを指定した場合，その後ろの [reg1] の部分が省略できます。ただし，省略した場合 as850 では，[gp] が指定されたものとみなされます。
- disp にデバイス・ファイルで定義されている周辺 I/O レジスタ名を指定した場合，その後ろの [reg1] の部分が省略できます。ただし，省略した場合，as850 は [r0] が指定されたものとみなされます。

〔フラグ〕

CY	—
OV	—
S	—
Z	—
SAT	—

〔注意事項〕

- ld.b，および ld.h は，1 バイト分，および 1 ハーフワード分のデータを符号拡張し，1 ワード分のデータとしてレジスタにロードされます。
- ld.bu，および ld.hu は，1 バイト分，および 1 ハーフワード分のデータをゼロ拡張し，1 ワード分のデータとしてレジスタにロードされます。
- ld.h，ld.w，ld.hu の disp に 2 の倍数でない値を指定した場合，as850 では，disp に対して，2 でアライメントしたコードが生成され，次のいずれかのメッセージが出力されます。

```
W3010: illegal operand (range error in immediate).
```

```
W4659: relocated value (value) of relocation entry (symbol: symbol, file:
file, section: section, offset: offset, type: relocation type) for load/store
command become odd value.
```

- ld.bu，および ld.hu 命令に対し，第ニオペランドに r0 を指定した場合，次のメッセージが出力され，アセンブルが中止されます。

```
E3240: illegal operand (can not use r0 as destination in V850E mode)
```

sld

〔概要〕

データのロード（Short format Load）

〔形式〕

(1) sld.b disp7[ep], reg2

(2) sld.h disp8[ep], reg2

(3) sld.w disp8[ep], reg2

(4) sld.bu disp4[ep], reg2【V850E】

(5) sld.hu disp5[ep], reg2【V850E】

ディスプレースメント（disp4 ／ 5 ／ 7 ／ 8）に指定できるものを次に示します。

- sld.b では 7 ビット，sld.h，および sld.w では 8 ビット，sld.bu では 4 ビット，sld.hu では 5 ビット幅までの値を持つ絶対値式
- 相対値式

〔機能〕

sld.b，sld.bu，sld.h，sld.hu，sld.w 命令は，第一オペランドに指定したディスプレースメントとレジスタ ep の内容を加算して得たアドレスから，1 バイト分，1 ハーフワード分，および 1 ワード分のデータを取り込み，第二オペランドに指定したレジスタにロードします。

〔説明〕

- as850 では，機械語命令の sld 命令が 1 つ生成されます。ベース・レジスタの指定“ [ep] ”は省略できます。

〔フラグ〕

CY	—
OV	—
S	—
Z	—
SAT	—

〔注意事項〕

- sld.b，および sld.h は，1 バイト分，および 1 ハーフワード分のデータを符号拡張し，1 ワードのデータとしてレジスタに格納されます。
- sld.bu，および sld.hu は，1 バイト分，および 1 ハーフワード分のデータをゼロ拡張し，1 ワードのデータとしてレジスタに格納されます。
- sld.h ／ sld.hu の disp8 ／ disp5 に 2 の倍数でない値を指定した場合，および sld.w の disp8 に 4 の倍数でない値を指定した場合，as850 では，disp8 ／ disp5 に対して，それぞれ 2 の倍数，4 の倍数にアライメントしたコードが生成され，次のいずれかのメッセージが出力されます。

```
W3010: illegal operand (range error in immediate).
```

```
W4659: relocated value (value) of relocation entry (symbol: symbol, file:
file, section: section, offset: offset, type: relocation type) for load/store
command become odd value.
```

- sld.b の disp7 に 127 を越える値を指定した場合,sld.h，sld.w の disp8 に 255 を越える値を指定した場合，sld.bu の disp4 に 16 を越える値を指定した場合,および sld.hu の disp5 に 32 を越える値を指定した場合，次のメッセージが出力され，disp7，disp8，disp4，disp5 をそれぞれ 0x7f，0xff，0xf，0x1f でマスクしたコードが生成されます。

```
W3011: illegal operand (range error in immediate)
```

- sld.bu，および sld.hu の第ニオペランド（reg2）に r0 を指定した場合，次のメッセージが出力され，アセンブルが中止されます。

```
E3240: illegal operand (can not use r0 as destination in V850E mode)
```

sst

〔概要〕

データのストア（Short format Store）

〔形式〕

(1) sst.b reg2, disp7[ep]

(2) sst.h reg2, disp8[ep]

(3) sst.w reg2, disp8[ep]

ディスプレースメント（disp7 ／ 8）**に指定できるものを次に示します。**

- sst.b **では** 7 **ビット，**sst.h**，および** sst.w **では** 8 **ビット幅までの値を持つ絶対値式**
- **相対値式**

〔機能〕

sst.b**，**sst.h**，および** sst.w **命令は，第一オペランドに指定したレジスタの下位** 1 **バイト分，下位** 1 **ハーフワード分，および** 1 **ワード分のデータを，第二オペランドに指定したディスプレースメントとしてレジスタ** ep **の内容を加算して得たアドレスに格納します。**

〔説明〕

- as850 **では，機械語命令の** sst **命令が** 1 **つ生成されます。ベース・レジスタの指定“** [ep] **”は省略できます。**

〔フラグ〕

CY	—
OV	—
S	—
Z	—
SAT	—

〔注意事項〕

- sst.h の disp8 に 2 の倍数でない値を指定した場合，および sst.w の disp8 に 4 の倍数でない値を指定した場合，as850 では，disp8 に対して，それぞれ 2 の倍数，4 の倍数にアライメントしたコードが生成され，次のいずれかのメッセージが出力されます。

```
W3010: illegal operand (range error in immediate).
```

```
W4659: relocated value (value) of relocation entry (symbol: symbol, file:
file, section: section, offset: offset, type: relocation type) for load/store
command become odd value.
```

- sst.b の disp7 に 127 を越える値を指定した場合，および sst.h，sst.w の disp8 に 255 を越える値を指定した場合，次のメッセージが出力され，disp7，disp8 をそれぞれ 0x7f，0xff でマスクしたコードが生成されます。

```
W3011: illegal operand (range error in immediate)
```

st

〔概要〕

データのストア（Store）

〔形式〕

(1) st.b reg2, disp[reg1]

(2) st.h reg2, disp[reg1]

(3) st.w reg2, disp[reg1]

ディスプレースメント（disp）に指定できるものを，次に示します。

- 32 ビット幅までの値を持つ絶対値式
- 相対値式
- 上記のものに hi( ) ，lo( ) ，または hi1( ) を適用したもの

〔機能〕

st.b，st.h，および st.w 命令は，第一オペランドに指定したレジスタの下位 1 バイト分，下位 1 ハーフワード分，および 1 ワード分のデータを取り込み，第二オペランドに指定したアドレスにストアします。

〔説明〕

- disp に次のものを指定した場合，as850 では，機械語命令の st 命令[注]が 1 つ生成されます。なお，例中の“ st ”は st.b，st.h のいずれかになります。

(a) -32768 ～ +32767 の範囲の絶対値式

`st reg2, disp16[reg1]`	`st reg2, disp16[reg1]`

(b) sdata ／ sbss 属性セクションに定義を持つラベルの $label を持つ相対値式

`st reg2, $label[reg1]`	`st reg2, $label[reg1]`

(c) !label，または %label を持つ相対値式

`st reg2, !label[reg1]`	`st reg2, !label[reg1]`

`st reg2, %label[reg1]`	`st reg2, %label[reg1]`

(d) hi( ) ，lo( ) ，または hi1( ) を適用したもの

`st reg2, disp16[reg1]`	`st reg2, disp16[reg1]`

注 機械語命令の st 命令は，ディスプレースメントに -32768 ～ +32767（0xffff8000 ～ 0x7fff）の範囲のイミーディエトをとります。

- disp に次のものを指定した場合，as850 では，命令展開が行われ，複数個の機械語命令が生成されます。

(a) -32768 ～ +32767 の範囲を越える絶対値式

`st reg2, disp[reg1], reg2`	`movhi hi1(disp), reg1, r1` `st reg2, lo(disp)[r1], reg2`

(b) #label，または label を持つ相対値式，および sdata ／ sbss 属性セクションに定義を持たないラベルの $label を持つ相対値式

`st reg2, #label[reg1]`	`movhi hi1(#label), reg1, r1` `st reg2, lo(#label)[r1]`

`st reg2, label[reg1]`	`movhi hi1(label), reg1, r1` `st reg2, lo(label)[r1]`

`st reg2, $label[reg1]`	`movhi hi1($label), reg1, r1` `st reg2, lo($label)[r1]`

- disp を省略した場合，as850 では，0 を指定したものとみなされます。
- disp に #label を持つ相対値式，または #label を持つ相対値式に hi( ) ，lo( ) ，または hi1( ) を適用したものを指定した場合，その後ろの [reg1] の部分が省略できます。ただし，省略した場合，as850 では，[r0] が指定されたものとみなされます。
- disp に $label を持つ相対値式，または $label を持つ相対値式に hi( ) ，lo( ) ，または hi1( ) を適用したものを指定した場合，その後ろの [reg1] の部分が省略できます。ただし，省略した場合，as850 では，[gp] が指定されたものとみなされます。
- disp にデバイス・ファイルで定義されている周辺 I/O レジスタ名を指定した場合，その後ろの [reg1] の部分が省略できます。ただし，省略した場合，as850 では，[r0] が指定されたものとみなされます。

〔フラグ〕

CY	—
OV	—
S	—
Z	—
SAT	—

〔注意事項〕

- st.h，および st.w の disp に 2 の倍数でない値を指定した場合，as850 では，disp に対して 2 でアライメントしたコードが生成され，次のいずれかのメッセージが出力されます。

```
W3010: illegal operand (range error in immediate).
```

```
W4659: relocated value (value) of relocation entry (symbol: symbol, file: file, section: section, offset: offset, type: relocation type) for load/store command become odd value.
```

3.3 算術演算命令

この節では，算術演算命令について説明します。次に，この節において説明する命令を示します。

表 3 － 2 算術演算命令

命令	意味
add	加算
addi	加算（イミーディエト）
cmov	フラグ条件付きデータの転送 【V850E】
cmp	比較
div	符号付き除算（ワード）【V850E】
divh	符号付き除算（ハーフワード）
divhu	符号なし除算（ハーフワード）【V850E】
divu	符号なし除算（ワード）【V850E】
mov	データの転送
mov32	32 ビット・データの転送【V850E】
movea	実効アドレスの転送
movhi	上位ハーフワードの転送
mul	符号付き乗算（ワード）【V850E】
mulh	符号付き乗算（ハーフワード）
mulhi	符号付き乗算（ハーフワード・イミーディエト）
mulu	符号なし乗算 【V850E】
mac	符号付きワード・データの加算付き乗算【V850E2】
macu	符号なしワード・データの加算付き乗算【V850E2】
sasf	論理左シフト付きフラグ条件の設定 【V850E】
setf	フラグ条件の設定
sub	減算
subr	逆減算
adf	条件付き加算【V850E2】
sbf	条件付き減算【V850E2】

add

〔概要〕

加算（Add）

〔形式〕

(1) add reg1, reg2

(2) add imm, reg2

imm に指定できるものを次に示します。

- 32 ビット幅までの値を持つ絶対値式
- 相対値式

〔機能〕

- (1) の形式

  第一オペランドに指定したレジスタ値と，第二オペランドに指定したレジスタ値を加算し，結果を第二オペランドに指定したレジスタに格納します。

- (2) の形式

  第一オペランドに指定した絶対値式，または相対値式の値と第二オペランドに指定したレジスタ値を加算し，結果を第二オペランドに指定したレジスタに格納します。

〔説明〕

- (1) の形式の命令に対し，as850 では，機械語命令の add 命令が 1 つ生成されます。
- (2) の形式で imm に次のものを指定した場合，as850 では，機械語命令の add 命令[注]が 1 つ生成されます。

(a) - 16 ～ +15 の範囲の絶対値式

`add imm15, reg`	`add imm5, reg`

注 機械語命令の add 命令は，第一オペランドにレジスタ，または -16 ～ +15（0xfffffff0 ～ 0xf）の範囲のイミーディエトをとります。

- (2) の形式で imm に次のものを指定した場合，as850 では，命令展開が行われ，1 つ，または複数個の機械語命令が生成されます。

(a) - 16 ～ +15 の範囲を越える絶対値式

`add imm16, reg`	`addi imm16, reg, reg`

(b) -32768 ～ +32767 の範囲を越える絶対値式

imm の値の下位 16 ビットがすべて 0 の場合

add imm, reg	movhi hi(imm), r0, r1 add r1, reg

上記以外の場合

add imm, reg	movhi hi1(imm), r0, r1
	movea lo(imm), r1, r1
	add r1, reg

(c) - 32768 ～ +32767 の範囲を越える絶対値式【V850E】

imm の値の下位 16 ビットがすべて 0 の場合

add imm, reg	movhi hi(imm), r0, r1 add r1, reg

上記以外の場合

add imm, reg	mov imm, r1 add r1, reg

(d) !label，または %label を持つ相対値式，および sdata ／ sbss 属性セクションに定義を持つラベルの $label を持つ相対値式

add !label, reg	addi !label, reg, reg

add %label, reg	addi %label, reg, reg

add $label, reg	addi $label, reg, reg

(e) #label，または label を持つ相対値式，および sdata ／ sbss 属性セクションに定義を持たないラベルの $label を持つ相対値式

add #label, reg	movhi hi1(#label), r0, r1
	movea lo(#label), r1, r1
	add r1, reg

add label, reg	movhi hi1(label), r0, r1
	movea lo(label), r1, r1
	add r1, reg

add $label, reg	movhi hi1($label), r0, r1
	movea lo($label), r1, r1
	add r1, reg

(f) #label，または label を持つ相対値式，および sdata ／ sbss 属性セクションに定義を持たないラベルの $label を持つ相対値式【V850E】

`add #label, reg`	`mov #label, r1` `add r1, reg`

`add label, reg`	`mov label, r1` `add r1, reg`

`add $label, reg`	`mov $label, r1` `add r1, reg`

〔フラグ〕

CY	MSB（Most Significant Bit）からのキャリーを生じた場合 1，そうでない場合 0
OV	Integer-Overflow を生じた場合 1，そうでない場合 0
S	結果が負になった場合 1，そうでない場合 0
Z	結果が 0 になった場合 1，そうでない場合 0
SAT	—

addi

〔概要〕

加算（Add Immediate）

〔形式〕

(1) addi imm, reg1, reg2

imm に指定できるものを，次に示します。

- 32 ビット幅までの値を持つ絶対値式
- 相対値式
- 上記のものに hi( ) ，lo( ) ，または hi1( ) を適用したもの

〔機能〕

第一オペランドに指定した絶対値式，相対値式，あるいは，hi( ) ，lo( ) ，または hi1( ) を適用した値と第二オペランドに指定したレジスタ値を加算し，結果を第三オペランドに指定したレジスタに格納します。

〔説明〕

- imm に次のものを指定した場合，as850 では，機械語命令の addi 命令[注]が 1 つ生成されます。

(a) -32768 ～ +32767 の範囲の絶対値式

`addi imm16, reg1, reg2`	`addi imm16, reg1, reg2`

(b) sdata ／ sbss 属性セクションに定義を持つラベルの $label を持つ相対値式

`addi $label, reg1, reg2`	`addi $label, reg1, reg2`

(c) !label，または %label を持つ相対値式

`addi !label, reg1, reg2`	`addi !label, reg1, reg2`

`addi %label, reg1, reg2`	`addi %label, reg1, reg2`

(d) hi( ) ，lo( ) ，または hi1( ) を適用したもの

`addi imm16, reg1, reg2`	`addi imm16, reg1, reg2`

注　機械語命令の addi **命令は，第一オペランドに** -32768 **～** +32767（0xffff8000 **～** 0x7fff）**の範囲のイミーディエトをとります。**

- imm **に次のものを指定した場合，**as850 **では，命令展開が行われ，複数個の機械語命令が生成されます。**

(a)　-32768 **～** +32767 **の範囲を越える絶対値式**

imm の値の下位 16 **ビットがすべて** 0 **の場合**

`addi  imm, reg1, reg2`	`movhi  hi(imm), r0, reg2` `add  reg1, reg2`

imm の値の下位 16 **ビットがすべて** 0**，ただし** reg2 **が** r0 **の場合**

`addi  imm, reg1, r0`	`movhi  hi(imm), r0, r1` `add  reg1, r1`

上記以外の場合

`addi  imm, reg1, reg2`	`movhi  hi1(imm), r0, r1`
	`movea  lo(imm), r1, reg2`
	`add  reg1, reg2`

上記以外，ただし reg2 **が** r0 **の場合**

`addi  imm, reg1, r0`	`movhi  hi1(imm), r0, r1`
	`movea  lo(imm), r1, r1`
	`add  reg1, r1`

(b)　-32768 **～** +32767 **の範囲を越える絶対値式【**V850E**】**

imm の値の下位 16 **ビットがすべて** 0 **の場合**

`addi  imm, reg1, reg2`	`movhi  hi(imm), r0, reg2` `add  reg1, reg2`

imm の値の下位 16 **ビットがすべて** 0**，ただし** reg2 **が** r0 **の場合**

`addi  imm, reg1, r0`	`movhi  hi(imm), r0, r1` `add  reg1, r1`

上記以外の場合

`addi  imm, reg1, reg2`	`mov  imm, reg2` `add  reg1, reg2`

上記以外，ただし reg2 **が** r0 **の場合**

`addi  imm, reg1, r0`	`mov  imm, r1` `add  reg1, r1`

(c) #label，または label を持つ相対値式，および sdata ／ sbss 属性セクションに定義を持たないラベルの $label を持つ相対値式

reg2 が r0 の場合

addi #label, reg1, r0	movhi hi1(#label), r0, r1
	movea lo(#label), r1, r1
	add reg1, reg2

addi label, reg1, r0	movhi hi1(label), r0, r1
	movea lo(label), r1, r1
	add reg1, r1

addi $label, reg1 r0	movhi hi1($label), r0, r1
	movea lo($label), r1, r1
	add reg1, r1

上記以外の場合

addi #label, reg1, reg2	movhi hi1(#label), r0, r1
	movea lo(#label), r1, reg2
	add reg1, reg2

addi label, reg1, reg2	movhi hi1(label), r0, r1
	movea lo(label), r1, reg2
	add reg1, reg2

addi $label, reg1 reg2	movhi hi1($label), r0, r1
	movea lo($label), r1, reg2
	add reg1, reg2

(d) #label，または label を持つ相対値式，および sdata ／ sbss 属性セクションに定義を持たないラベルの $label を持つ相対値式【V850E】

reg2 が r0 の場合

addi #label, reg1, r0	mov #label, r1 addi reg1, r1

addi label, reg1, r0	mov label, r1 add reg1, r1

addi $label, reg1, r0	mov $label, r1 add reg1, r1

上記以外の場合

`addi  #label, reg1, reg2`	`mov  #label, reg2` `addi  reg1, reg2`

`addi  label, reg1, reg2`	`mov  label, reg2` `add  reg1, reg2`

`addi  $label, reg1, reg2`	`mov  $label, reg2` `add  reg1, reg2`

〔フラグ〕

CY	MSB（Most Significant Bit）**からのキャリーを生じた場合** 1**，そうでない場合** 0
OV	Integer-Overflow **を生じた場合** 1**，そうでない場合** 0
S	**結果が負になった場合** 1**，そうでない場合** 0
Z	**結果が** 0 **になった場合** 1**，そうでない場合** 0
SAT	—

cmov

【V850E】

〔概要〕

フラグ条件付きデータの転送（Conditional Move）

〔形式〕

(1) cmov imm4, reg1, reg2, reg3

(2) cmov imm4, imm, reg2, reg3

(3) cmov*cond* reg1, ret2, reg3

(4) cmov*cond* imm, reg2, reg3

imm4 に指定できるものを次に示します。

- 4 ビット幅までの値を持つ定数式[注]

注 機械語命令の cmov 命令は，第一オペランドに 0 ～ 15（0x0 ～ 0xf）の範囲のイミーディエトをとります。

imm に指定できるものを次に示します。

- 32 ビット幅までの値を持つ絶対値式

〔機能〕

- (1) の形式

  第一オペランドに指定した定数式の値の下位 4 ビットの値で示されるフラグ状態と，現在のフラグ状態を比較し，値が一致した場合は，第二オペランドに指定したレジスタ値を，一致しなかった場合は第三オペランドに指定したレジスタ値を，第四オペランドに指定したレジスタに格納します。

- (2) の形式

  第一オペランドに指定した定数式の値の下位 4 ビットの値で示されるフラグ状態と，現在のフラグ状態を比較し，値が一致した場合は，第二オペランドに指定した絶対値式の値を，一致しなかった場合は第三オペランドに指定したレジスタ値を，第四オペランドに指定したレジスタに格納します。

- (3) の形式

  cond 部分の文字列で示されるフラグ状態と現在のフラグの状態を比較し，値が一致した場合は，第一オペランドに指定したレジスタ値を，一致しなかった場合は第二オペランドに指定したレジスタ値を，第三オペランドに指定したレジスタに格納します。

- (4) の形式

  cond 部分の文字列で示されるフラグ状態と現在のフラグの状態を比較し，値が一致した場合は，第一オペランドに指定した絶対値式の値を，一致しなかった場合は第二オペランドに指定したレジスタ値を，第三オペランドに指定したレジスタに格納します。

表 3 － 3 cmov*cond* 命令一覧

命令	フラグ状態	フラグ状態の意味	命令展開
cmovgt	((S xor OV) or Z) = 0	Greater than（signed）	cmov 0xf
cmovge	(S xor OV) = 0	Greater than or equal（signed）	cmov 0xe
cmovlt	(S xor OV) = 1	Less than（signed）	cmov 0x6
cmovle	((S xor OV) or Z) = 1	Less than or equal（signed）	cmov 0x7
cmovh	(CY or Z) = 0	Higher（Greater than）	cmov 0xb
cmovnl	CY = 0	Not lower（Greater than or equal）	cmov 0x9
cmovl	CY = 1	Lower（Less than）	cmov 0x1
cmovnh	(CY or Z) = 1	Not higher（Less than or equal）	cmov 0x3
cmove	Z = 1	Equal	cmov 0x2
cmovne	Z = 0	Not equal	cmov 0xa
cmovv	OV = 1	Overflow	cmov 0x0
cmovnv	OV = 0	No overflow	cmov 0x8
cmovn	S = 1	Negative	cmov 0x4
cmovp	S = 0	Positive	cmov 0xc
cmovc	CY = 1	Carry	cmov 0x1
cmovnc	CY = 0	No carry	cmov 0x9
cmovz	Z = 1	Zero	cmov 0x2
cmovnz	Z = 0	Not zero	cmov 0xa
cmovt	always 1	Always 1	cmov 0x5
cmovsa	SAT = 1	Saturated	cmov 0xd

〔説明〕

- (1) **の形式の命令に対し，**as850 **では，機械語命令の** cmov **命令**[注]**が** 1 **つ生成されます。**

注　機械語命令の cmov **命令は，第二オペランドにレジスタ，または** -16 **～** +15（0xfffffff0 **～** 0xf）**の範囲のイミーディエトをとります。**

- (2) **の形式で** imm **に次のものを指定した場合，**as850 **では，機械語命令の** cmov **命令が** 1 **つ生成されます。**

(a)　-16 **～** +15 **の範囲の絶対値式**

`cmov  imm4, imm5, reg2, reg3`	`cmov  imm4, imm5, reg2, reg3`

- (2) の形式で imm に次のものを指定した場合，as850 では，命令展開が行われ，複数個の機械語命令が生成されます。

(a) -16 ～ +15 の範囲を越え，-32768 ～ +32767 の範囲の絶対値式

`cmov imm4, imm16, reg2, reg3`	`movea imm16, r0, r1` `cmov imm4, r1, reg2, reg3`

(b) - 32768 ～ +32767 の範囲を越える絶対値式

imm の値の下位 16 ビットがすべて 0 の場合

`cmov imm4, imm, reg2, reg3`	`movhi hi(imm), r0, r1` `cmov imm4, r1, reg2, reg3`

上記以外の場合

`cmov imm4, imm, reg2, reg3`	`mov imm, r1` `cmov imm4, r1, reg2, reg3`

(c) sdata ／ sbss 属性セクションに定義を持つラベルの $label を持つ相対値式

`cmov imm4, $label, reg2, reg3`	`movea $label, r0, r1` `cmov imm4, r1, reg2, reg3`

(d) #label，または label を持つ相対値式，および sdata ／ sbss 属性セクションに定義を持たないラベルの $label を持つ相対値式

`cmov imm4, #label, reg2, reg3`	`mov #label, r1` `cmov imm4, r1, reg2, reg3`

`cmov imm4, label, reg2, reg3`	`mov label, r1` `cmov imm4, r1, reg2, reg3`

`cmov imm4, $label, reg2, reg3`	`mov $label, r1` `cmov imm4, r1, reg2, reg3`

- (3) の形式の命令に対し，as850 では，対応する cmov 命令が生成され（「表 3 － 3 cmovcond 命令一覧」参照），(1) の形式に展開されます。
- (4) の形式で imm に次のものを指定した場合，as850 では，対応する cmov 命令が生成され（「表 3 － 3 cmovcond 命令一覧」参照），(2) の形式に展開されます。

(a) -16 ～ +15 の範囲の絶対値式

- (4) の形式で imm に次のものを指定した場合，as850 では，命令展開が行われ，複数個の機械語命令が生成されます。

(a) - 16 ～ +15 の範囲を越え，-32768 ～ +32767 の範囲の絶対値式

cmov*cond* imm16, reg2, reg3	movea imm16, r0, r1 cmov*cond* r1, reg2, reg3

(b) - 32768 ～ +32767 の範囲を越える絶対値式

imm の値の下位 16 ビットがすべて 0 の場合

cmov*cond* imm, reg2, reg3	movhi hi(imm), r0, r1 cmov*cond* r1, reg2, reg3

上記以外の場合

cmov*cond* imm, reg2, reg3	mov imm, r1 cmov*cond* r1, reg2, reg3

(c) sdata ／ sbss 属性セクションに定義を持つラベルの $label を持つ相対値式

cmov*cond* $label, reg2, reg3	movea $label, r0, r1 cmov*cond* r1, reg2, reg3

(d) #label，または label を持つ相対値式，および sdata ／ sbss 属性セクションに定義を持たないラベルの $label を持つ相対値式

cmov*cond* #label, reg2, reg3	mov #label, r1 cmov*cond* r1, reg2, reg3

cmov*cond* label, reg2, reg3	mov label, r1 cmov*cond* r1, reg2, reg3

cmov*cond* $label, reg2, reg3	mov $label, r1 cmov*cond* r1, reg2, reg3

〔フラグ〕

CY	—
OV	—
S	—
Z	—
SAT	—

〔注意事項〕

- cmov 命令の imm4 に 4 ビットの範囲を越える定数式を指定した場合，次のメッセージが出力されます。なお，4 ビット幅を越える場合は，0xf でマスクした値を用いてアセンブルが続行されます。

```
W3011: illegal operand (range error in immediate)
```

- cmov 命令の imm4 に定数式以外[注]を指定した場合，次のメッセージが出力され，アセンブルが中止されます。

```
E3249: illegal syntax
```

注　未定義シンボルやラベルの参照です。

cmp

〔概要〕

比較（Compare）

〔形式〕

(1) cmp reg1, reg2

(2) cmp imm, reg2

imm に指定できるものを次に示します。

- 32 ビット幅までの値を持つ絶対値式
- 相対値式

〔機能〕

- (1) の形式

  第一オペランドに指定したレジスタ値と第二オペランドに指定したレジスタ値を比較し，結果をフラグに示します。なお，比較は，第二オペランドに指定したレジスタ値から第一オペランドに指定したレジスタ値を減算することにより行われます。

- (2) の形式

  第一オペランドに指定した絶対値式，または相対値式の値と第二オペランドに指定したレジスタ値を比較し，結果をフラグに示します。なお，比較は，第二オペランドに指定したレジスタ値から第一オペランドに指定した値を減算することにより行われます。

〔説明〕

- (1) の形式の命令に対し，as850 では，機械語命令の cmp 命令が 1 つ生成されます。
- (2) の形式で imm に次のものを指定した場合，as850 では，機械語命令の cmp 命令[注]が 1 つ生成されます。

(a) -16 ～ +15 の範囲の絶対値式

`cmp imm5, reg`	`cmp imm5, reg`

注 機械語命令の cmp 命令は，第一オペランドにレジスタ，または -16 ～ +15（0xfffffff0 ～ 0xf）の範囲のイミーディエトをとります。

- (2) の形式で imm に次のものを指定した場合，as850 では，命令展開が行われ，複数個の機械語命令が生成されます。

(a) -16 ～ +15 の範囲を越える絶対値式

`cmp imm16, reg`	`movea imm16, r0, r1` `cmp r1, reg`

(b) - 32768 ～ +32767 の範囲を越える絶対値式

imm の値の下位 16 ビットがすべて 0 の場合

cmp imm, reg	movhi hi(imm), r0, r1 cmp r1, reg

上記以外の場合

cmp imm, reg	movhi hi1(imm), r0, r1
	movea lo(imm), r1, r1
	cmp r1, reg

(c) - 32768 ～ +32767 の範囲を越える絶対値式【V850E】

imm の値の下位 16 ビットがすべて 0 の場合

cmp imm, reg	movhi hi(imm), r0, r1 cmp r1, reg

上記以外の場合

cmp imm, reg	mov imm, r1 cmp r1, reg

(d) sdata ／ sbss 属性セクションに定義を持つラベルの $label を持つ相対値式

cmp $label, reg	movea $label, r0, r1 cmp r1, reg

(e) #label，または label を持つ相対値式，および sdata ／ sbss 属性セクションに定義を持たないラベルの $label を持つ相対値式

cmp #label, reg	movhi hi1(#label), r0, r1
	movea lo(#label), r1, r1
	cmp r1, reg

cmp label, reg	movhi hi1(label), r0, r1
	movea lo(label), r1, r1
	cmp r1, reg

cmp $label, reg	movhi hi1($label), r0, r1
	movea lo($label), r1, r1
	cmp r1, reg

(f) #label，または label を持つ相対値式，および sdata ／ sbss 属性セクションに定義を持たないラベルの $label を持つ相対値式【V850E】

`cmp #label, reg`	`mov #label, r1` `cmp r1, reg`

`cmp label, reg`	`mov label, r1` `cmp r1, reg`

`cmp $label, reg`	`mov $label, r1` `cmp r1, reg`

〔フラグ〕

CY	MSB（Most Significant Bit）へのボローが生じた場合 1，そうでない場合 0
OV	Integer-Overflow を生じた場合 1，そうでない場合 0
S	結果が負になった場合 1，そうでない場合 0
Z	結果が 0 になった場合 1，そうでない場合 0
SAT	—

div

【V850E】

〔概要〕

符号付き除算（ワード）（Divide Word）

〔形式〕

(1) div　　reg1, reg2, reg3

(2) div　　imm, reg2, reg3

imm に指定できるものを次に示します。

- 32 ビット幅までの値を持つ絶対値式
- 相対値式

〔機能〕

- (1) の形式

  第二オペランドに指定したレジスタ値を，第一オペランドに指定したレジスタ値で符号付きの値として除算し，商を第二オペランドに指定したレジスタに，剰余を第三オペランドに指定したレジスタに，それぞれ格納します。第二オペランドと第三オペランドが同じレジスタの場合，レジスタには剰余を格納します。

- (2) の形式

  第二オペランドに指定したレジスタ値を，第一オペランドに指定した絶対値式，または相対値式の値で符号付きの値として除算し，商を第二オペランドに指定したレジスタに，剰余を第三オペランドに指定したレジスタに格納します。第二オペランドと第三オペランドが同じレジスタの場合，レジスタには剰余を格納します。

〔説明〕

- (1) の形式の命令に対し，as850 では，機械語命令の div 命令[注]が 1 つ生成されます。
- (2) の形式で imm に次のものを指定した場合，as850 では，命令展開が行われ，複数個の機械語命令が生成されます[注]。

(a)　0

`div　　0, reg2, reg3`	`div　　r0, reg2, reg3`

(b)　0 以外の -16 ～ +15 の範囲の絶対値式

`div　　imm5, reg2, reg3`	`mov　　imm5, r1` `div　　r1, reg2, reg3`

(c) - 16 ～ +15 の範囲を越え，-32768 ～ +32767 の範囲の絶対値式

`div imm16, reg2, reg3`	`movea imm16, r0, r1` `div r1, reg2, reg3`

(d) - 32768 ～ +32767 の範囲を越える絶対値式

imm の値の下位 16 ビットがすべて 0 の場合

`div imm, reg2, reg3`	`movhi hi(imm), r0, r1` `div r1, reg2, reg3`

上記以外の場合

`div imm, reg2, reg3`	`mov imm, r1` `div r1, reg2, reg3`

(e) sdata ／ sbss 属性セクションに定義を持つラベルの $label を持つ相対値式

`div $label, reg2, reg3`	`movea $label, r0, r1` `div r1, reg2, reg3`

(f) #label，または label を持つ相対値式，および sdata ／ sbss 属性セクションに定義に持たないラベルの $label を持つ相対値式

`div #label, reg2, reg3`	`mov #label, r1` `div r1, reg2, reg3`

`div label, reg2, reg3`	`mov label, r1` `div r1, reg2, reg3`

`div $label, reg2, reg3`	`mov $label, r1` `div r1, reg2, reg3`

注 機械語命令の div 命令は，オペランドにイミーディエトをとりません。

〔フラグ〕

CY	—
OV	Integer-Overflow を生じた場合 1，そうでない場合 0
S	結果が負になった場合 1，そうでない場合 0
Z	結果が 0 になった場合 1，そうでない場合 0
SAT	—

divh

〔概要〕

符号付き除算（ハーフワード）（Divide Half-word）

〔形式〕

(1) divh reg1, reg2

(2) divh imm, reg2

(3) divh reg1, reg2, reg3【V850E】

(4) divh imm, reg2, reg3【V850E】

imm に指定できるものを次に示します。

- 16 ビット幅までの値を持つ絶対値式[注]
- 相対値式

注 as850 では，16 ビットを越えるかどうかのチェックは行われません。生成された機械語命令では，下位 16 ビットを用いて演算が行われます。

〔機能〕

- (1) の形式

  第二オペランドに指定したレジスタ値を，第一オペランドに指定したレジスタの下位ハーフワード・データの値で符号付きの値として除算し，商を第二オペランドに指定したレジスタに格納します。

- (2) の形式

  第二オペランドに指定したレジスタ値を，第一オペランドに指定した絶対値式，または相対値式の下位ハーフワード・データの値で符号付きの値として除算し，商を第二オペランドに指定したレジスタに格納します。

- (3) の形式

  第二オペランドに指定したレジスタ値を，第一オペランドに指定したレジスタの下位ハーフワード・データの値で符号付きの値として除算し，商を第二オペランドに指定したレジスタに，剰余を第三オペランドに指定したレジスタに格納します。第二オペランドと第三オペランドが同じレジスタの場合，レジスタには剰余を格納します。

- (4) の形式

  第二オペランドに指定したレジスタ値を，第一オペランドに指定した絶対値式，または相対値式の下位ハーフワード・データの値で符号付きの値として除算し，商を第二オペランドに指定したレジスタに，剰余を第三オペランドに指定したレジスタに格納します。第二オペランドと第三オペランドが同じレジスタの場合，レジスタには剰余を格納します。

〔説明〕

- (1)，および (3) の形式の命令に対し，as850 では，機械語命令の divh 命令が 1 つ生成されます。
- (2) の形式で imm に次のものを指定した場合，as850 では，命令展開が行われ，1 つ，または複数個の機械語命令が生成されます[注]。

(a) 0

`divh  0, reg`	`divh  r0, reg`

(b) - 16 ～ +15 の範囲の絶対値式（0 以外 ）

`divh  imm5, reg`	`mov  imm5, r1` `divh  r1, reg`

(c) - 16 ～ +15 の範囲の絶対値式（0 以外 ）【V850E】

`divh  imm5, reg`	`mov  imm5, r1` `divh  r1, reg`

(d) - 16 ～ +15 の範囲を越え，-32768 ～ +32767 の範囲の絶対値式

`divh  imm16, reg`	`movea  imm16, r0, r1` `divh  r1, reg`

(e) imm に -32768 ～ +32767 の範囲を越える絶対値式

imm の値の下位 16 ビットがすべて 0 の場合

`divh  imm, reg`	`movhi  hi(imm), r0, r1` `divh  r1, reg`

上記以外の場合

`divh  imm, reg`	`movhi  hi1(imm), r0, r1`
	`movea  lo(imm), r1, r1`
	`divh  r1, reg`

(f) imm に -32768 ～ +32767 の範囲を越える絶対値式【V850E】

imm の下位 16 ビットがすべて 0 の場合

`divh  imm, reg`	`movhi  hi(imm), r0, r1` `divh  r1, reg`

上記以外の場合

`divh imm, reg`	`mov imm, r1` `divh r1, reg`

(g) sdata ／ sbss **属性セクションに定義を持つラベルの** $label **を持つ相対値式**

`divh $label, reg`	`movea $label, r0, r1` `divh r1, reg`

(h) #label**，または** label **を持つ相対値式，および** sdata ／ sbss **属性セクションに定義を持たないラベルの** $label **を持つ相対値式**

`divh #label, reg`	`movhi hi1(#label), r0, r1`
	`movea lo(#label), r1, r1`
	`divh r1, reg`

`divh label, reg`	`movhi hi1(label), r0, r1`
	`movea lo(label), r1, r1`
	`divh r1, reg`

`divh $label, reg`	`movhi hi1($label), r0, r1`
	`movea lo($label), r1, r1`
	`divh r1, reg`

(i) #label**，または** label **を持つ相対値式，および** sdata ／ sbss **属性セクションに定義を持たないラベルの** $label **を持つ相対値式【**V850E**】**

`divh #label, reg`	`mov #label, r1` `divh r1, reg`

`divh label, reg`	`mov label, r1` `divh r1, reg`

`divh $label, reg`	`mov $label, r1` `divh r1, reg`

注 **機械語命令の** div **命令は，オペランドにイミーディエトをとりません。**

- (4) の形式で imm に次のものを指定した場合，as850 では，命令展開が行われ，複数個の機械語命令が生成されます。【V850E】

(a) 0

`divh 0, reg2, reg3`	`divh r0, reg2, reg3`

(b) 0 以外の -16 ～ +15 の範囲の絶対値式

`divh imm5, reg2, reg3`	`mov imm5, r1` `divh r1, reg2, reg3`

(c) - 16 ～ +15 の範囲を越え，-32768 ～ +32767 の範囲の絶対値式

`divh imm16, reg2, reg3`	`movea imm16, r0, r1` `divh r1, reg2, reg3`

(d) - 32768 ～ +32767 の範囲を越える絶対値式

imm の値の下位 16 ビットがすべて 0 の場合

`divh imm, reg2, reg3`	`movhi hi(imm), r0, r1` `divh r1, reg2, reg3`

上記以外の場合

`divh imm, reg2, reg3`	`mov imm, r1` `divh r1, reg2, reg3`

(e) sdata ／ sbss 属性セクションに定義を持つラベルの $label を持つ相対値式

`divh $label, reg2, reg3`	`movea $label, r0, r1` `divh r1, reg2, reg3`

(f) #label，または label を持つ相対値式，および sdata ／ sbss 属性セクションに定義に持たないラベルの $label を持つ相対値式

`divh #label, reg2, reg3`	`mov #label, r1` `divh r1, reg2, reg3`

`divh label, reg2, reg3`	`mov label, r1` `divh r1, reg2, reg3`

`divh  $label, reg2, reg3`	`mov  $label, r1` `divh  r1, reg2, reg3`

〔フラグ〕

CY	—
OV	`Integer-Overflow` を生じた場合 `1`，そうでない場合 `0`
S	結果が負になった場合 `1`，そうでない場合 `0`
Z	結果が `0` になった場合 `1`，そうでない場合 `0`
SAT	—

〔注意事項〕

- ターゲット・デバイスが V850Ex の場合，形式 (1) の第一オペランド（reg1）に r0 を指定すると，次のメッセージが出力され，アセンブルが中止されます。

```
E3239: illegal operand (can not use r0 as source in V850E mode)
```

なお，V850Ex 以外の場合，次のメッセージが出力され，アセンブルが続行されます。

```
W3013: register r0 used as source register
```

- ターゲット・デバイスが V850Ex の場合，形式 (1)，および (2) の第二オペランド（reg2）に r0 を指定すると，次のメッセージが出力され，アセンブルが中止されます。

```
E3240: illegal operand (can not use r0 as destination in V850E mode)
```

なお，V850Ex 以外の場合，次のメッセージが出力され，アセンブルが続行されます。

```
W3013: register r0 used as destination register
```

divhu

【V850E】

〔概要〕

符号なし除算（ハーフワード）（Divide Half-word Unsigned）

〔形式〕

(1) divhu reg1, reg2, reg3

(2) divhu imm, reg2, reg3

imm に指定できるものを次に示します。

- 16 ビット幅までの値を持つ絶対値式[注]
- 相対値式

注 as850 では，16 ビットを越えるかどうかのチェックは行われません。生成された機械語命令では，下位 16 ビットを用いて演算が行われます。

〔機能〕

- (1) の形式

第二オペランドに指定したレジスタ値を，第一オペランドに指定したレジスタの下位ハーフワード・データの値で符号なしの値として除算し，商を第二オペランドに指定したレジスタに，剰余を第三オペランドに指定したレジスタに格納します。第二オペランドと第三オペランドが同じレジスタの場合，レジスタには剰余を格納します。

- (2) の形式

第二オペランドに指定したレジスタ値を，第一オペランドに指定した絶対値式，または相対値式の下位ハーフワード・データの値で符号なしの値として除算し，商を第二オペランドに指定したレジスタに，剰余を第三オペランドに指定したレジスタに格納します。第二オペランドと第三オペランドが同じレジスタの場合，レジスタには剰余を格納します。

〔説明〕

- (1) の形式の命令に対し，as850 では，機械語命令の divhu 命令が 1 つ生成されます。
- (2) の形式で imm に次のものを指定した場合，as850 では，命令展開が行われ，複数個の機械語命令が生成されます[注]。

(a) 0

`divhu 0, reg2, reg3`	`divhu r0, reg2, reg3`

(b) 0 以外の -16 ～ +15 の範囲の絶対値式

`divhu imm5, reg2, reg3`	`mov imm5, r1` `divhu r1, reg2, reg3`

(c) - 16 ～ +15 の範囲を越え，-32768 ～ +32767 の範囲の絶対値式

`divhu imm16, reg2, reg3`	`movea imm16, r0, r1` `divhu r1, reg2, reg3`

(d) - 32768 ～ +32767 の範囲を越える絶対値式

imm の値の下位 16 ビットがすべて 0 の場合

`divhu imm, reg2, reg3`	`movhi hi(imm), r0, r1` `divhu r1, reg2, reg3`

上記以外の場合

`divhu imm, reg2, reg3`	`mov imm, r1` `divhu r1, reg2, reg3`

(e) sdata ／ sbss 属性セクションに定義を持つラベルの $label を持つ相対値式

`divhu $label, reg2, reg3`	`movea $label, r0, r1` `divhu r1, reg2, reg3`

(f) #label，または label を持つ相対値式，および sdata ／ sbss 属性セクションに定義に持たないラベルの $label を持つ相対値式

`divhu #label, reg2, reg3`	`mov #label, r1` `divhu r1, reg2, reg3`

`divhu label, reg2, reg3`	`mov label, r1` `divhu r1, reg2, reg3`

`divhu $label, reg2, reg3`	`mov $label, r1` `divhu r1, reg2, reg3`

注 機械語命令の divhu 命令は，オペランドにイミーディエトをとりません。

〔フラグ〕

CY	—
OV	Integer-Overflow を生じた場合 1，そうでない場合 0
S	演算結果のワード・データの MSB が 1 の場合 1，そうでない場合 0
Z	結果が 0 になった場合 1，そうでない場合 0
SAT	—

divu

【V850E】

〔概要〕

符号なし除算（ワード）（Divide Word Unsigned）

〔形式〕

(1) divu reg1, reg2, reg3

(2) divu imm, reg2, reg3

imm に指定できるものを次に示します。

- 32 ビット幅までの値を持つ絶対値式
- 相対値式

〔機能〕

- (1) の形式

  第二オペランドに指定したレジスタ値を，第一オペランドに指定したレジスタ値で符号なしの値として除算し，商を第二オペランドに指定したレジスタに，剰余を第三オペランドに指定したレジスタに格納します。第二オペランドと第三オペランドが同じレジスタの場合，レジスタには剰余を格納します。

- (2) の形式

  第二オペランドに指定したレジスタ値を，第一オペランドに指定した絶対値式，または相対値式の値で符号なしの値として除算し，商を第二オペランドに指定したレジスタに，剰余を第三オペランドに指定したレジスタに格納します。第二オペランドと第三オペランドが同じレジスタの場合，レジスタには剰余を格納します。

〔説明〕

- (1) の形式の命令に対し，as850 では，機械語命令の divu 命令が 1 つ生成されます。
- (2) の形式で imm に次のものを指定した場合，as850 では，命令展開が行われ，複数個の機械語命令が生成されます[注]。

(a) 0

`divu 0, reg2, reg3`	`divu r0, reg2, reg3`

(b) 0 以外の -16 ～ +15 の範囲の絶対値式

`divu imm5, reg2, reg3`	`mov imm5, r1` `divu r1, reg2, reg3`

(c) - 16 ～ +15 の範囲を越え，-32768 ～ +32767 の範囲の絶対値式

`divu imm16, reg2, reg3`	`movea imm16, r0, r1` `divu r1, reg2, reg3`

(d) - 32768 ～ +32767 の範囲を越える絶対値式

imm の値の下位 16 ビットがすべて 0 の場合

`divu imm, reg2, reg3`	`movhi hi(imm), r0, r1` `divu r1, reg2, reg3`

上記以外の場合

`divu imm, reg2, reg3`	`mov imm, r1` `divu r1, reg2, reg3`

(e) sdata ／ sbss 属性セクションに定義を持つラベルの $label を持つ相対値式

`divu $label, reg2, reg3`	`movea $label, r0, r1` `divu r1, reg2, reg3`

(f) #label，または label を持つ相対値式，および sdata ／ sbss 属性セクションに定義に持たないラベルの $label を持つ相対値式

`divu #label, reg2, reg3`	`mov #label, r1` `divu r1, reg2, reg3`

`divu label, reg2, reg3`	`mov label, r1` `divu r1, reg2, reg3`

`divu $label, reg2, reg3`	`mov $label, r1` `divu r1, reg2, reg3`

注 機械語命令の divu 命令は，オペランドにイミーディエトをとりません。

〔フラグ〕

CY	—
OV	Integer-Overflow を生じた場合 1，そうでない場合 0
S	演算結果のワード・データの MSB が 1 の場合 1，そうでない場合 0
Z	結果が 0 になった場合 1，そうでない場合 0
SAT	—

mov

〔概要〕

データの転送（Move）

〔形式〕

(1) mov reg1, reg2

(2) mov imm, reg2

imm に指定できるものを次に示します。

- 32 ビット幅までの値を持つ絶対値式
- 相対値式

〔機能〕

- (1) の形式

  第一オペランドに指定したレジスタ値を，第二オペランドに指定したレジスタに格納します。

- (2) の形式

  第一オペランドに指定した絶対値式，または相対値式の値を，第二オペランドに指定したレジスタに格納します。

〔説明〕

- (1) の形式の命令に対し，as850 では，機械語命令の mov 命令が 1 つ生成されます。
- (2) の形式で imm に次のものを指定した場合，as850 では，機械語命令の mov 命令[注]が 1 つ生成されます。

(a) -16 ～ +15 の範囲の絶対値式

`mov imm5, reg`	`mov imm5, reg`

注 機械語命令の mov 命令は，V850 の場合 16 ビット長形式のみであり，V850Ex の場合 48 ビット長形式がサポートされています。したがって，機械語命令の mov 命令は，V850 の場合，第一オペランドにレジスタ，または -16 ～ +15（0xfffffff0 ～ 0xf）の範囲のイミーディエトをとります。V850Ex の場合，これらに加えて，-2147483648 ～ -2147483647（0x80000000 ～ 0x7fffffff）の範囲のイミーディエトをとります。

- (2) の形式で imm に次のものを指定した場合，as850 では，命令展開が行われ，1 つ，または複数個の機械語命令が生成されます。

(a) - 16 ～ +15 の範囲を越え -32768 ～ +32767 の範囲の絶対値式

`mov imm16, reg`	`movea imm16, r0, reg`

(b) - 32768 ～ +32767 の範囲を越える絶対値式

imm の値の下位 16 ビットがすべて 0 の場合

mov imm, reg	movhi hi(imm), r0, reg

上記以外の場合

mov imm, reg	movhi hi1(imm), r0, r1 movea lo(imm), r1, reg

(c) - 32768 ～ +32767 の範囲を越える絶対値式【V850E】

imm の値の下位 16 ビットがすべて 0 の場合

mov imm, reg	movhi hi(imm), r0, reg

上記以外の場合[注]

mov imm, reg	mov imm, reg

注 16 bit 長の mov 命令を，48 bit 長の mov 命令に置き換えます。

(d) !label，または %label を持つ相対値式，および sdata ／ sbss 属性セクションに定義を持つラベルの $label を持つ相対値式

mov !label, reg	movea !label, r0, reg

mov %label, reg	movea %label, r0, reg

mov $label, reg	movea $label, r0, reg

(e) #label，または label を持つ相対値式，および sdata ／ sbss 属性セクションに定義を持たないラベルの $label を持つ相対値式

mov #label, reg	movhi hi1(#label), r0, r1 movea lo(#label), r1, reg

mov label, reg	movhi hi1(label), r0, r1 movea lo(label), r1, reg

mov $label, reg	movhi hi1($label), r0, r1 movea lo($label), r1, reg

(f) #label，または label を持つ相対値式，および sdata ／ sbss 属性セクションに定義を持たないラベルの $label を持つ相対値式[注]【V850E】

`mov #label, reg`	`mov #label, reg`
`mov label, reg`	`mov label, reg`
`mov $label, reg`	`mov $label, reg`

注 16 bit 長の mov 命令を，48 bit 長の mov 命令に置き換えます。

〔フラグ〕

CY	—
OV	—
S	—
Z	—
SAT	—

〔注意事項〕

- 形式 (1) の第一オペランドに r0 を指定し，かつ，第ニオペランドに r0 を指定すると，アセンブルした結果は nop の命令コードになります。
- ターゲット・デバイスが V850Ex の場合，形式 (2) の命令に対し，第一オペランドに -16 ～ + 15 の範囲の絶対値式を指定し，かつ第二オペランドに r0 を指定した場合，次のメッセージが出力され，アセンブルが中止されます。

```
E3240: illegal operand (can not use r0 as destination in V850E mode)
```

- ターゲット・デバイスが V850Ex の場合，形式 (2) の命令に対し，第一オペランドに -32768 ～ + 32767 の範囲を越える絶対値式，#label，または label を持つ相対値式，および sdata ／ sbss 属性セクションに定義を持たないラベルの $label を持つ相対値式を指定し，疑似命令 .option nomacro の指定により命令展開を抑止している場合，次のメッセージが出力され，アセンブルが中止されます。

```
E3249: illegal syntax
```

このような場合は，mov32 命令を使用してください。

mov32

【V850E】

〔概要〕

32 ビット・データの転送（32bit Move）

〔形式〕

(1) mov32 imm, reg2

imm に指定できるものを次に示します。

- 32 ビット幅までの値を持つ絶対値式
- 相対値式

〔機能〕

第一オペランドに指定した絶対値式，または相対値式の値を，第二オペランドに指定したレジスタに格納します。

〔説明〕

- as850 では，機械語命令の 48 ビット長 mov 命令が 1 つ生成されます。

〔フラグ〕

CY	—
OV	—
S	—
Z	—
SAT	—

movea

〔概要〕

実効アドレスの転送（Move Effective Address）

〔形式〕

(1) movea imm, reg1, reg2

imm に指定できるものを次に示します。

- 32 ビット幅までの値を持つ絶対値式
- 相対値式
- 上記のものに hi( ) ，lo( ) ，または hi1( ) を適用したもの

〔機能〕

第一オペランドに指定した絶対値式，相対値式，あるいは，hi( ) ，lo( ) ，または hi1( ) を適用した値を，第二オペランドに指定したレジスタ値と加算し，結果を第三オペランドに指定したレジスタに格納します。

〔説明〕

- imm に次のものを指定した場合，as850 では，機械語命令の movea 命令[注]が 1 つ生成されます。

(a) -32768 ～ +32767 の範囲の絶対値式

`movea imm16, reg1, reg2`	`movea imm16, reg1, reg2`

(b) sdata ／ sbss 属性セクションに定義を持つラベルの $label を持つ相対値式

`movea $label, reg1, reg2`	`movea $label, reg1, reg2`

(c) !label，または %label を持つ相対値式

`movea !label, reg1, reg2`	`movea !label, reg1, reg2`

`movea %label, reg1, reg2`	`movea %label, reg1, reg2`

(d) hi( ) ，lo( ) ，または hi1( ) を適用したもの

movea imm16, reg1, reg2	movea imm16, reg1, reg2

注 機械語命令の movea 命令は，第一オペランドに -32768 ～ +32767（0xffff8000 ～ 0x7fff）の範囲のイミーディエトをとります。

- imm に次のものを指定した場合，as850 では，命令展開が行われ，1 つ，または複数個の機械語命令が生成されます。

(a) -32768 ～ +32767 の範囲を越える絶対値式

imm の値の下位 16 ビットがすべて 0 の場合

movea imm, reg1, reg2	movhi hi(imm), reg1, reg2

上記以外の場合

movea imm, reg1, reg2	movhi hi1(imm), reg1, r1 movea lo(imm), r1, reg2

(b) #label，または label を持つ相対値式，および sdata ／ sbss 属性セクションに定義を持たないラベルの $label を持つ相対値式

movea #label, reg1, reg2	movhi hi1(#label), reg1, r1 movea lo(#label), r1, reg2

movea label, reg1, reg2	movhi hi1(label), reg1, r1 movea lo(label), r1, reg2

movea $label, reg1, reg2	movhi hi1($label), reg1, r1 movea lo($label), r1, reg2

〔フラグ〕

CY	—
OV	—
S	—
Z	—
SAT	—

〔注意事項〕

- **ターゲット・デバイスが** V850Ex **の場合，第三オペランドに** r0 **を指定すると，次のメッセージが出力され，アセンブルが中止されます。**

```
E3240: illegal operand (can not use r0 as destination in V850E mode)
```

なお，V850Ex **以外の場合，次のメッセージが出力され，アセンブルが続行されます。**

```
W3013: register r0 used as destination register
```

movhi

〔概要〕

上位ハーフワードの転送（Move High half-word）

〔形式〕

(1) movhi imm16, reg1, reg2

imm16 に指定できるものを次に示します。

- 16 ビット幅までの値を持つ絶対値式
- 相対値式
- 上記のものに hi( )，lo( )，または hi1( ) を適用したもの

〔機能〕

第一オペランドに指定した値を上位 16 ビットの値，0 を下位 16 ビットの値とするワード・データと，第二オペランドに指定したレジスタ値を加算し，結果を第三オペランドに指定したレジスタに格納します。

〔説明〕

- as850 では，機械語命令の movhi 命令が 1 つ生成されます。

〔フラグ〕

CY	—
OV	—
S	—
Z	—
SAT	—

〔注意事項〕

- **imm16 に** 0 ～ 65535 **の範囲を越える絶対値式を指定した場合，次のメッセージが出力され，アセンブルが中止されます。**

```
E3231: illegal operand (range error in immediate)
```

- **ターゲット・デバイスが** V850Ex **の場合，第三オペランドに** r0 **を指定すると，次のメッセージが出力され，アセンブルが中止されます。**

```
E3240: illegal operand (can not use r0 as destination in V850E mode)
```

なお，V850Ex **以外の場合，次のメッセージが出力され，アセンブルが続行されます。**

```
W3013: register r0 used as destination register
```

mul

【V850E】

〔概要〕

符号付き乗算（ワード）（Multiply Word）

〔形式〕

(1) mul reg1, reg2, reg3

(2) mul imm, reg2, reg3

imm に指定できるものを次に示します。

- 32 ビット幅までの値を持つ絶対値式
- 相対値式

〔機能〕

- (1) の形式

  第一オペランドに指定したレジスタ値と，第二オペランドに指定したレジスタ値を，符号付きの値として乗算し，結果の下位 32 ビットを第二オペランドに指定したレジスタに，上位 32 ビットを第三オペランドに指定したレジスタに格納します。第二オペランドと第三オペランドが同じレジスタの場合，レジスタには乗算結果の上位 32 ビットを格納します。

- (2) の形式

  第一オペランドに指定した絶対値式，または相対値式の値と，第二オペランドに指定したレジスタの値を，符号付きの値として乗算し，結果の下位 32 ビットを第二オペランドに指定したレジスタに，上位 32 ビットを第三オペランドに指定したレジスタに格納します。第二オペランドと第三オペランドが同じレジスタの場合，レジスタには乗算結果の上位 32 ビットを格納します。

〔説明〕

- (1) の形式の命令に対し，as850 では，機械語命令の mul 命令が 1 つ生成されます。
- (2) の形式で imm に次のものを指定した場合，as850 では，命令展開が行われ，1 つ，または複数個の機械語命令が生成されます。

(a) 0

`mul 0, reg2, reg3`	`mul r0, reg2, reg3`

(b) 0 以外の -256 ～ +255 の範囲の絶対値式

`mul imm9, reg2, reg3`	`mul imm9, reg2, reg3`

(c) - 256 ～ +255 の範囲を越え，-32768 ～ +32767 の範囲の絶対値式

`mul imm16, reg2, reg3`	`movea imm16, r0, r1` `mul r1, reg2, reg3`

(d) - 32768 ～ +32767 の範囲を越える絶対値式

imm の値の下位 16 ビットがすべて 0 の場合

`mul imm, reg2, reg3`	`movhi hi(imm), r0, r1` `mul r1, reg2, reg3`

上記以外の場合

`mul imm, reg2, reg3`	`mov imm, r1` `mul r1, reg2, reg3`

(e) sdata ／ sbss 属性セクションに定義を持つラベルの $label を持つ相対値式

`mul $label, reg2, reg3`	`movea $label, r0, r1` `mul r1, reg2, reg3`

(f) #label，または label を持つ相対値式，および sdata ／ sbss 属性セクションに定義を持たないラベルの $label を持つ相対値式

`mul #label, reg2, reg3`	`mov #label, r1` `mul r1, reg2, reg3`

`mul label, reg2, reg3`	`mov label, r1` `mul r1, reg2, reg3`

`mul $label, reg2, reg3`	`mov $label, r1` `mul r1, reg2, reg3`

〔フラグ〕

CY	—
OV	—
S	—
Z	—
SAT	—

〔注意事項〕

- (1) の形式の命令に対して “ reg1 と reg3 は同一のレジスタである ”, “ reg2 は reg1, reg3 と異なるレジスタである ”, “ reg1，reg3 が r0，または r1 ではない ” の条件をすべて満たす場合，as850 は命令展開を行い，複数個の機械語命令を生成します。

```
        mov     reg1, r1
        mul     r1, reg2, reg3
```

- (1) の形式の命令に対し，“ reg1 と reg3 は同一のレジスタである ”, “ reg2 は reg1, reg3 と異なるレジスタである ”, “ reg1, reg3 が r1 である ” の条件をすべて満たす場合，as850 は次のメッセージを出力し，アセンブルを中止します。

```
W3013: register r1 used as source register
W3013: register r1 used as destination register
E3259: can not use r1 as destination in mul/mulu
```

- (2) の形式の命令に対し，“ reg2 と reg3 が同一のレジスタである ”, “ reg3 が r1 である ” の条件をすべて満たす場合，as850 は次のメッセージを出力し，アセンブルを中止します。

```
W3013: register r1 used as destination register
E3259: can not use r1 as destination in mul/mulu
```

警告メッセージ抑止オプション -wr1- を指定した場合，次のメッセージを出力し，アセンブルを中止します。

```
E3259: can not use r1 as destination in mul/mulu
```

mulh

〔概要〕

符号付き乗算（ハーフワード）（Multiply Half-word）

〔形式〕

(1) mulh reg1, reg2

(2) mulh imm, reg2

imm に指定できるものを次に示します。

- 16 ビット幅までの値を持つ絶対値式[注]
- 相対値式

注 as850 では，16 ビットを越えるかどうかチェックは行われません。生成された機械語命令 mulh 命令では，下位 16 ビットを用いて演算が行われます。

〔機能〕

- (1) の形式

  第一オペランドに指定したレジスタの下位ハーフワード・データの値と，第二オペランドに指定したレジスタの下位ハーフワード・データの値を，符号付きの値として乗算し，結果を第二オペランドに指定したレジスタに格納します。

- (2) の形式

  第一オペランドに指定した絶対値式，または相対値式の下位ハーフワード・データの値と，第二オペランドに指定したレジスタの下位ハーフワード・データの値を，符号付きの値として乗算し，結果を第二オペランドに指定したレジスタに格納します。

〔説明〕

- (1) の形式の命令に対し，as850 では，機械語命令の mulh 命令が 1 つ生成されます。
- (2) の形式で imm に次のものを指定した場合，as850 では，機械語命令の add 命令[注]が 1 つ生成されます。

(a) - 16 ～ +15 の範囲の絶対値式

`mulh imm15, reg`	`mulh imm5, reg`

注 機械語命令の add 命令は，第一オペランドにレジスタ，または -16 ～ +15（0xfffffff0 ～ 0xf）の範囲のイミーディエトをとります。

- (2) の形式で imm に次のものを指定した場合，as850 では，命令展開が行われ，1 つ，または複数個の機械語命令が生成されます。

(a) - 16 ～ +15 の範囲を越える絶対値式

`mulh  imm16, reg`	`mulhi  imm16, reg, reg`

(b) -32768 ～ +32767 の範囲を越える絶対値式

imm の値の下位 16 ビットがすべて 0 の場合

`mulh  imm, reg`	`movhi  hi(imm), r0, r1` `mulh  r1, reg`

上記以外の場合

`mulh  imm, reg`	`movhi  hi1(imm), r0, r1`
	`movea  lo(imm), r1, r1`
	`mulh  r1, reg`

(c) -32768 ～ +32767 の範囲を越える絶対値式【V850E】

imm の値の下位 16 ビットがすべて 0 の場合

`mulh  imm, reg`	`movhi  hi(imm), r0, r1` `mulh  r1, reg`

上記以外の場合

`mulh  imm, reg`	`mov  imm, r1` `mulh  r1, reg`

(d) !label，または %label を持つ相対値式，および sdata ／ sbss 属性セクションに定義を持つラベルの $label を持つ相対値式

`mulh  !label, reg`	`mulhi  !label, reg, reg`

`mulh  %label, reg`	`mulhi  %label, reg, reg`

`mulh  $label, reg`	`mulhi  $label, reg, reg`

(e) #label，または label を持つ相対値式，および sdata ／ sbss 属性セクションに定義を持たないラベルの $label を持つ相対値式

`mulh #label, reg`	`movhi hi1(#label), r0, r1`
	`movea lo(#label), r1, r1`
	`mulh r1, reg`

`mulh label, reg`	`movhi hi1(label), r0, r1`
	`movea lo(label), r1, r1`
	`mulh r1, reg`

`mulh $label, reg`	`movhi hi1($label), r0, r1`
	`movea lo($label), r1, r1`
	`mulh r1, reg`

(f) #label，または label を持つ相対値式，および sdata ／ sbss 属性セクションに定義を持たないラベルの $label を持つ相対値式【V850E】

`mulh #label, reg`	`mov #label, r1` `mulh r1, reg`

`mulh label, reg`	`mov label, r1` `mulh r1, reg`

`mulh $label, reg`	`mov $label, r1` `mulh r1, reg`

〔フラグ〕

CY	—
OV	—
S	—
Z	—
SAT	—

〔注意事項〕

- **ターゲット・デバイスが** V850Ex **の場合，第二オペランドに** r0 **を指定すると，次のメッセージが出力され，アセンブルが中止されます。**

```
E3240: illegal operand (can not use r0 as destination in V850E mode)
```

なお，V850Ex **以外の場合，次のメッセージが出力され，アセンブルが続行されます。**

```
W3013: register r0 used as destination register
```

mulhi

〔概要〕

符号付き乗算（ハーフワード・イミーディエト）（Multiply Half-word Immediate）

〔形式〕

(1) mulhi imm, reg1, reg2

imm に指定できるものを次に示します。

- 16 ビット幅までの値を持つ絶対値式[注]
- 相対値式
- 上記のものに hi( )，lo( )，または hi1( ) を適用したもの

注 as850 では，16 ビットを越えるかどうかチェックは行われません。生成された機械語命令 mulhi 命令では，下位 16 ビットを用いて演算が行われます。

〔機能〕

第一オペランドに指定した絶対値式，相対値式，あるいは，hi( )，lo( )，または hi1( ) を適用した値と，第二オペランドに指定したレジスタ値を符号付きの値として乗算し，結果を第三オペランドに指定したレジスタに格納します。

〔説明〕

- imm に次のものを指定した場合，as850 では，機械語命令の mulhi 命令[注]が 1 つ生成されます。

(a) -32768 ～ +32767 の範囲の絶対値式

`mulhi imm16, reg1, reg2`	`mulhi imm16, reg1, reg2`

(b) sdata ／ sbss 属性セクションに定義を持つラベルの $label を持つ相対値式

`mulhi $label, reg1, reg2`	`mulhi $label, reg1, reg2`

(c) !label，または %label を持つ相対値式

`mulhi !label, reg1, reg2`	`mulhi !label, reg1, reg2`

`mulhi %label, reg1, reg2`	`mulhi %label, reg1, reg2`

(d) hi( ) ，lo( ) ，または hi1( ) を適用したもの

`mulhi imm16, reg1, reg2`	`mulhi imm16, reg1, reg2`

注 機械語命令の mulhi 命令は，第一オペランドに -32768 ～ +32767（0xffff8000 ～ 0x7fff）の範囲のイミーディエトをとります。

- imm に次のものを指定した場合，as850 では，命令展開が行われ，複数個の機械語命令が生成されます。

(a) -32768 ～ +32767 の範囲を越える絶対値式

imm の値の下位 16 ビットがすべて 0 の場合

`mulhi imm, reg1, reg2`	`movhi hi(imm), r0, reg2` `mulh reg1, reg2`

imm の値の下位 16 ビットがすべて 0，ただし reg2 が r0 の場合

`mulhi imm, reg1, r0`	`movhi hi(imm), r0, r1` `mulh reg1, r1`

上記以外の場合

`mulhi imm, reg1, reg2`	`movhi hi1(imm), r0, r1`
	`movea lo(imm), r1, reg2`
	`mulh reg1, reg2`

上記以外，ただし reg2 が r0 の場合

`mulhi imm, reg1, reg2`	`movhi hi1(imm), r0, r1`
	`movea lo(imm), r1, r1`
	`mulh reg1,r1`

(b) -32768 ～ +32767 の範囲を越える絶対値式【V850E】

imm の値の下位 16 ビットがすべて 0 の場合

`mulhi imm, reg1, reg2`	`movhi hi(imm), r0, reg2` `mulh reg1, reg2`

上記以外の場合

`mulhi imm, reg1, reg2`	`mov imm, reg2` `mulh reg1, reg2`

(c) #label，または label を持つ相対値式，および sdata ／ sbss 属性セクションに定義を持たないラベルの $label を持つ相対値式

reg2 が r0 の場合

`mulhi #label, reg1, r0`	`movhi hi1(#label), r0, r1`
	`movea lo(#label), r1, r1`
	`mulh reg1, r1`

`mulhi label, reg1, r0`	`movhi hi1(label), r0, r1`
	`movea lo(label), r1, r1`
	`mulh reg1, r1`

`mulhi $label, reg1 r0`	`movhi hi1($label), r0, r1`
	`movea lo($label), r1, r1`
	`mulh reg1, r1`

上記以外の場合

`mulhi #label, reg1, reg2`	`movhi hi1(#label), r0, r1`
	`movea lo(#label), r1, reg2`
	`mulh reg1, reg2`

`mulhi label, reg1, reg2`	`movhi hi1(label), r0, r1`
	`movea lo(label), r1, reg2`
	`mulh reg1, reg2`

`mulhi $label, reg1 reg2`	`movhi hi1($label), r0, r1`
	`movea lo($label), r1, reg2`
	`mulh reg1, reg2`

(d) #label，または label を持つ相対値式，および sdata ／ sbss 属性セクションに定義を持たないラベルの $label を持つ相対値式【V850E】

`mulhi #label, reg1, reg2`	`mov #label, reg2` `mulhi reg1, reg2`
`mulhi label, reg1, reg2`	`mov label, reg2` `mulh reg1, reg2`
`mulhi $label, reg1, reg2`	`mov $label, reg2` `mulh reg1, reg2`

〔フラグ〕

CY	—
OV	—
S	—
Z	—
SAT	—

〔注意事項〕

- ターゲット・デバイスが V850Ex の場合，第二オペランドに r0 を指定すると，次のメッセージが出力され，アセンブルが中止されます。

```
E3240: illegal operand (can not use r0 as destination in V850E mode)
```

なお，V850Ex 以外の場合，次のメッセージが出力され，アセンブルが続行されます。

```
W3013: register r0 used as destination register
```

mulu

【V850E】

〔概要〕

符号なし乗算（ワード）（Multiply Word Unsigned）

〔形式〕

(1) mulu reg1, reg2, reg3

(2) mulu imm, reg2, reg3

imm に指定できるものを次に示します。

- 32 ビット幅までの値を持つ絶対値式
- 相対値式

〔機能〕

- (1) の形式

  第一オペランドに指定したレジスタ値と，第二オペランドに指定したレジスタ値を，符号なしの値として乗算し，結果の下位 32 ビットを第二オペランドに指定したレジスタに，上位 32 ビットを第三オペランドに指定したレジスタに格納します。第二オペランドと第三オペランドが同じレジスタの場合，レジスタには結果の上位 32 ビットを格納します。

- (2) の形式

  第一オペランドに指定した絶対値式，または相対値式の値と，第二オペランドに指定したレジスタの値を，符号なしの値として乗算し，結果の下位 32 ビットを第二オペランドに指定したレジスタに，上位 32 ビットを第三オペランドに指定したレジスタに格納します。第二オペランドと第三オペランドが同じレジスタの場合，レジスタには結果の上位 32 ビットを格納します。

〔説明〕

- (1) の形式の命令に対し，as850 では，機械語命令の mulu 命令が 1 つ生成されます。
- (2) の形式で imm に次のものを指定した場合，as850 では，命令展開が行われ，1 つ，または複数個の機械語命令が生成されます。

(a) 0

`mulu 0, reg2, reg3`	`mulu r0, reg2, reg3`

(b) 0 以外の 0 ～ 511 の範囲

`mulu imm9, reg2, reg3`	`mulu imm9, reg2, reg3`

(c) 0 ～ 511 の範囲を越え，-32768 ～ +32767 の範囲の絶対値式

`mulu imm16, reg2, reg3`	`movea imm16, r0, r1` `mulu r1, reg2, reg3`

(d) - 32768 ～ +32767 の範囲を越える絶対値式

imm の値の下位 16 ビットがすべて 0 の場合

`mulu imm, reg2, reg3`	`movhi hi(imm), r0, r1` `mulu r1, reg2, reg3`

上記以外の場合

`mulu imm, reg2, reg3`	`mov imm, r1` `mulu r1, reg2, reg3`

(e) sdata ／ sbss 属性セクションに定義を持つラベルの $label を持つ相対値式

`mulu $label, reg2, reg3`	`movea $label, r0, r1` `mulu r1, reg2, reg3`

(f) #label，または label を持つ相対値式，および sdata ／ sbss 属性セクションに定義を持たないラベルの $label を持つ相対値式

`mulu #label, reg2, reg3`	`mov #label, r1` `mulu r1, reg2, reg3`

`mulu label, reg2, reg3`	`mov label, r1` `mulu r1, reg2, reg3`

`mulu $label, reg2, reg3`	`mov $label, r1` `mulu r1, reg2, reg3`

〔フラグ〕

CY	—
OV	—
S	—
Z	—
SAT	—

〔注意事項〕

- (1) の形式の命令に対して “ reg1 と reg3 は同一のレジスタである ”, “ reg2 は reg1, reg3 と異なるレジスタである ”, “ reg1，reg3 が r0，または r1 ではない ” の条件をすべて満たす場合，as850 は命令展開を行い，複数個の機械語命令を生成します。

```
mov    reg1, r1
mulu   r1, reg2, reg3
```

- (1) の形式の命令に対し，“ reg1 と reg3 は同一のレジスタである ”, “ reg2 は reg1, reg3 と異なるレジスタである ”, “ reg1, reg3 が r1 である ” の条件をすべて満たす場合，as850 は次のメッセージを出力し，アセンブルを中止します。

```
W3013: register r1 used as source register
W3013: register r1 used as destination register
E3259: can not use r1 as destination in mul/mulu
```

- (2) の形式の命令に対し，“ reg2 と reg3 が同一のレジスタである ”, “ reg3 が r1 である ” の条件をすべて満たす場合，as850 は次のメッセージを出力し，アセンブルを中止します。

```
W3013: register r1 used as destination register
E3259: can not use r1 as destination in mul/mulu
```

警告メッセージ抑止オプション -wr1- を指定した場合，次のメッセージを出力し，アセンブルを中止します。

```
E3259: can not use r1 as destination in mul/mulu
```

mac

【V850E2】

〔概要〕

符号付きワード・データの加算付き乗算（Multiply Word and Add）

〔形式〕

(1) mac reg1, reg2, reg3, reg4

〔機能〕

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

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

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

〔説明〕

- as850 では，機械語命令の mac 命令が 1 つ生成されます。

〔フラグ〕

CY	—
OV	—
S	—
Z	—
SAT	—

〔注意事項〕

reg3，または reg4 に指定できる汎用レジスタは，偶数番号の付いたレジスタ（r0，r2，r4，…，r30）だけです。奇数番号の付いたレジスタ（r1，r3，…，r31）を指定した場合は，次のメッセージが出力され，偶数番号の付いたレジスタ（r0，r2，r4，…，r30）を指定したとして，アセンブルが続行されます。

```
W3026: illegal register number, aligned odd register(rXX) to be even register(rYY).
```

macu

【V850E2】

〔概要〕

符号なしワード・データの加算付き乗算（Multiply Word Unsigned and Add）

〔形式〕

(1) macu reg1, reg2, reg3, reg4

〔機能〕

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

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

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

〔説明〕

- as850 では，機械語命令の macu 命令が 1 つ生成されます。

〔フラグ〕

CY	—
OV	—
S	—
Z	—
SAT	—

〔注意事項〕

reg3，または reg4 に指定できる汎用レジスタは，偶数番号の付いたレジスタ（r0，r2，r4，…，r30）だけです。奇数番号の付いたレジスタ（r1，r3，…，r31）を指定した場合は，次のメッセージが出力され，偶数番号の付いたレジスタ（r0，r2，r4，…，r30）を指定したとして，アセンブルが続行されます。

```
W3026: illegal register number, aligned odd register(rXX) to be even register(rYY).
```

sasf

【V850E】

〔概要〕

論理左シフト付きフラグ条件の設定（Shift And Set Flag Condition）

〔形式〕

(1) sasf　　imm4, reg

(2) sasf*cond*　　reg

imm4 **に指定できるものを次に示します。**

- 4 **ビット幅までの値を持つ絶対値式**

〔機能〕

- sasf

  第一オペランドに指定した絶対値式の値の下位 4 **ビットの値で示されるフラグ状態（「表** 3 － 4 sasfcond **命令一覧」参照）と，現在のフラグ状態を比較します。値が一致した場合は，第二オペランドで指定したレジスタの内容を左** 1 **ビット論理シフトした値と** 1 **とを論理和して，第二オペランドに指定したレジスタに格納し，一致しなかった場合は，第二オペランドで指定したレジスタの内容を左** 1 **ビット論理シフトして第二オペランドで指定したレジスタに格納します。**

- sasf*cond*

  cond **部分の文字列で示されるフラグ状態と現在のフラグの状態を比較します。値が一致した場合は，第二オペランドで指定したレジスタの内容を左** 1 **ビット論理シフトした値と** 1 **とを論理和して第二オペランドで指定したレジスタに格納し，一致しなかった場合は，第二オペランドで指定したレジスタの内容を左** 1 **ビット論理シフトして第二オペランドで指定したレジスタに格納します。**

〔説明〕

- sasf **命令に対し，**as850 **では，機械語命令の** sasf **命令が** 1 **つ生成されます。**
- sasf*cond* **命令に対し，**as850 **では，対応する** sasf **命令が生成され（「表** 3 － 4 sasfcond **命令一覧」参照），**(1) **の形式に展開されます。**

表 3 － 4 sasf*cond* 命令一覧

命令	フラグ状態	フラグ状態の意味	命令展開
sasfgt	((S xor OV) or Z) = 0	Greater than（signed）	sasf 0xf
sasfge	(S xor OV) = 0	Greater than or equal（signed）	sasf 0xe
sasflt	(S xor OV) = 1	Less than（signed）	sasf 0x6
sasfle	((S xor OV) or Z) = 1	Less than or equal（signed）	sasf 0x7
sasfh	(CY or Z) = 0	Higher（Greater than）	sasf 0xb
sasfnl	CY = 0	Not lower（Greater than or equal）	sasf 0x9
sasfl	CY = 1	Lower（Less than）	sasf 0x1
sasfnh	(CY or Z) = 1	Not higher（Less than or equal）	sasf 0x3
sasfe	Z = 1	Equal	sasf 0x2
sasfne	Z = 0	Not equal	sasf 0xa
sasfv	OV = 1	Overflow	sasf 0x0
sasfnv	OV = 0	No overflow	sasf 0x8
sasfn	S = 1	Negative	sasf 0x4
sasfp	S = 0	Positive	sasf 0xc
sasfc	CY = 1	Carry	sasf 0x1
sasfnc	CY = 0	No carry	sasf 0x9
sasfz	Z = 1	Zero	sasf 0x2
sasfnz	Z = 0	Not zero	sasf 0xa
sasft	always 1	Always 1	sasf 0x5
sasfsa	SAT = 1	Saturated	sasf 0xd

〔フラグ〕

CY	—
OV	—
S	—
Z	—
SAT	—

〔注意事項〕

- sasf 命令の imm4 に 4 ビットの範囲を越える絶対値式を指定した場合，次のメッセージが出力され，指定された値の下位 4 ビットが用いられてアセンブルが続行されます。

```
W3011: illegal operand (range error in immediate).
```

setf

〔概要〕

フラグ条件の設定（Set Flag Condition）

〔形式〕

(1) setf imm4, reg

(2) setf*cond* reg

imm4 に指定できるものを次に示します。

- 4 ビット幅までの値を持つ絶対値式

〔機能〕

- setf

  第一オペランドに指定した絶対値式の値の下位 4 ビットの値で示されるフラグ状態と，現在のフラグ状態を比較し，値が一致した場合は 1 を，一致しなかった場合は 0 を第二オペランドに指定したレジスタに格納します。

- setf*cond*

  cond 部分の文字列で示されるフラグ状態と現在のフラグの状態を比較し，一致する場合は 1，一致しなかった場合は 0 を第二オペランドに指定したレジスタに格納します。

〔説明〕

- setf 命令に対し，as850 では，機械語命令の setf 命令が 1 つ生成されます。
- setf*cond* 命令に対し，as850 では，対応する sesf 命令が生成され（「表 3 － 5 setfcond 命令一覧」参照），(1) の形式に展開されます。

表 3 － 5 setf*cond* 命令一覧

命令	フラグ状態	フラグ状態の意味	命令展開
setfgt	((S xor OV) or Z) = 0	Greater than（signed）	setf 0xf
setfge	(S xor OV) = 0	Greater than or equal（signed）	setf 0xe
setflt	(S xor OV) = 1	Less than（signed）	setf 0x6
setfle	((S xor OV) or Z) = 1	Less than or equal（signed）	setf 0x7
setfh	(CY or Z) = 0	Higher（Greater than）	setf 0xb
setfnl	CY = 0	Not lower（Greater than or equal）	setf 0x9
setfl	CY = 1	Lower（Less than）	setf 0x1
setfnh	(CY or Z) = 1	Not higher（Less than or equal）	setf 0x3
setfe	Z = 1	Equal	setf 0x2
setfne	Z = 0	Not equal	setf 0xa
setfv	OV = 1	Overflow	setf 0x0
setfnv	OV = 0	No overflow	setf 0x8
setfn	S = 1	Negative	setf 0x4
setfp	S = 0	Positive	setf 0xc
setfc	CY = 1	Carry	setf 0x1
setfnc	CY = 0	No carry	setf 0x9
setfz	Z = 1	Zero	setf 0x2
setfnz	Z = 0	Not zero	setf 0xa
setft	always 1	Always 1	setf 0x5
setfsa	SAT = 1	Saturated	setf 0xd

〔フラグ〕

CY	—
OV	—
S	—
Z	—
SAT	—

〔注意事項〕

- setf 命令の imm4 に 4 ビットの範囲を越える絶対値式を指定した場合，次のメッセージが出力され，指定された値の下位 4 ビットが用いられてアセンブルが続行されます。

```
W3011: illegal operand (range error in immediate).
```

sub

〔概要〕

減算（Subtract）

〔形式〕

(1) sub reg1, reg2

(2) sub imm, reg2

imm に指定できるものを次に示します。

- 32 ビット幅までの値を持つ絶対値式
- 相対値式

〔機能〕

- (1) の形式

  第二オペランドに指定したレジスタ値から，第一オペランドに指定したレジスタ値を減算し，結果を第二オペランドに指定したレジスタに格納します。

- (2) の形式

  第二オペランドに指定したレジスタ値から，第一オペランドに指定した絶対値式，または相対値式の値を減算し，結果を第二オペランドに指定したレジスタに格納します。

〔説明〕

- (1) の形式の命令に対し，as850 では，機械語命令の sub 命令が 1 つ生成されます。
- (2) の形式で imm に次のものを指定した場合，as850 では，命令展開が行われ，1 つ，または複数個の機械語命令が生成されます[注]。

(a) 0

`sub 0, reg`	`sub r0, reg`

(b) - 16 ～ +15 の範囲の絶対値式（0 以外 ）

`sub imm5, reg`	`mov imm5, r1` `sub r1, reg`

(c) - 16 ～ +15 の範囲の絶対値式（0 以外 ）【V850E】

`sub imm5, reg`	`mov imm5, r1` `sub r1, reg`

(d) - 16 ～ +15 の範囲を越え，-32768 ～ +32767 の範囲の絶対値式

sub imm16, reg	movea imm16, r0, r1 sub r1, reg

(e) imm に -32768 ～ +32767 の範囲を越える絶対値式

imm の値の下位 16 ビットがすべて 0 の場合

sub imm, reg	movhi hi(imm), r0, r1 sub r1, reg

上記以外の場合

sub imm, reg	movhi hi1(imm), r0, r1
	movea lo(imm), r1, r1
	sub r1, reg

(f) imm に -32768 ～ +32767 の範囲を越える絶対値式【V850E】

imm の下位 16 ビットがすべて 0 の場合

sub imm, reg	movhi hi(imm), r0, r1 sub r1, reg

上記以外の場合

sub imm, reg	mov imm, r1 sub r1, reg

(g) sdata ／ sbss 属性セクションに定義を持つラベルの $label を持つ相対値式

sub $label, reg	movea $label, r0, r1 sub r1, reg

(h) #label，または label を持つ相対値式，および sdata ／ sbss 属性セクションに定義を持たないラベルの $label を持つ相対値式

`sub #label, reg`	`movhi hi1(#label), r0, r1`
	`movea lo(#label), r1, r1`
	`sub r1, reg`

`sub label, reg`	`movhi hi1(label), r0, r1`
	`movea lo(label), r1, r1`
	`sub r1, reg`

`sub $label, reg`	`movhi hi1($label), r0, r1`
	`movea lo($label), r1, r1`
	`sub r1, reg`

(i) #label，または label を持つ相対値式，および sdata ／ sbss 属性セクションに定義を持たないラベルの $label を持つ相対値式【V850E】

`sub #label, reg`	`mov #label, r1` `sub r1, reg`
`sub label, reg`	`mov label, r1` `sub r1, reg`
`sub $label, reg`	`mov $label, r1` `sub r1, reg`

注 機械語命令の sub 命令は，オペランドにイミーディエトをとりません。

〔フラグ〕

CY	MSB（Most Significant Bit）へのボローが生じた場合 1，そうでない場合 0
OV	Integer-Overflow を生じた場合 1，そうでない場合 0
S	結果が負になった場合 1，そうでない場合 0
Z	結果が 0 になった場合 1，そうでない場合 0
SAT	—

subr

〔概要〕

逆減算（Subtract Reverse）

〔形式〕

(1) subr reg1, reg2

(2) subr imm, reg2

imm に指定できるものを次に示します。

- 32 ビット幅までの値を持つ絶対値式
- 相対値式

〔機能〕

- (1) の形式

  第一オペランドに指定したレジスタ値から，第二オペランドに指定したレジスタ値を減算し，結果を第二オペランドに指定したレジスタに格納します。

- (2) の形式

  第一オペランドに指定した絶対値式，または相対値式の値から，第二オペランドに指定したレジスタ値を減算し，結果を第二オペランドに指定したレジスタに格納します。

〔説明〕

- (1) の形式の命令に対し，as850 では，機械語命令の subr 命令が 1 つ生成されます。
- (2) の形式で imm に次のものを指定した場合，as850 では，命令展開が行われ，1 つ，または複数個の機械語命令が生成されます[注]。

(a) 0

`subr 0, reg`	`subr r0, reg`

(b) - 16 ～ +15 の範囲の絶対値式（0 以外 ）

`subr imm5, reg`	`mov imm5, r1` `subr r1, reg`

(c) - 16 ～ +15 の範囲の絶対値式（0 以外 ）【V850E】

`subr imm5, reg`	`mov imm5, r1` `subr r1, reg`

(d) - 16 ～ +15 の範囲を越え，-32768 ～ +32767 の範囲の絶対値式

`subr  imm16, reg`	`movea  imm16, r0, r1` `subr  r1, reg`

(e) imm に -32768 ～ +32767 の範囲を越える絶対値式

imm の値の下位 16 ビットがすべて 0 の場合

`subr  imm, reg`	`movhi  hi(imm), r0, r1` `subr  r1, reg`

上記以外の場合

`subr  imm, reg`	`movhi  hi1(imm), r0, r1`
	`movea  lo(imm), r1, r1`
	`subr  r1, reg`

(f) imm に -32768 ～ +32767 の範囲を越える絶対値式【V850E】

imm の下位 16 ビットがすべて 0 の場合

`subr  imm, reg`	`movhi  hi(imm), r0, r1` `subr  r1, reg`

上記以外の場合

`subr  imm, reg`	`mov  imm, r1` `subr  r1, reg`

(g) sdata ／ sbss 属性セクションに定義を持つラベルの $label を持つ相対値式

`subr  $label, reg`	`movea  $label, r0, r1` `subr  r1, reg`

(h) #label，または label を持つ相対値式，および sdata ／ sbss 属性セクションに定義を持たないラベルの $label を持つ相対値式

`subr #label, reg`	`movhi hi1(#label), r0, r1`
	`movea lo(#label), r1, r1`
	`subr r1, reg`

`subr label, reg`	`movhi hi1(label), r0, r1`
	`movea lo(label), r1, r1`
	`subr r1, reg`

`subr $label, reg`	`movhi hi1($label), r0, r1`
	`movea lo($label), r1, r1`
	`subr r1, reg`

(i) #label，または label を持つ相対値式，および sdata ／ sbss 属性セクションに定義を持たないラベルの $label を持つ相対値式【V850E】

`subr #label, reg`	`mov #label, r1` `subr r1, reg`
`subr label, reg`	`mov label, r1` `subr r1, reg`
`subr $label, reg`	`mov $label, r1` `subr r1, reg`

注 機械語命令の subr 命令は，オペランドにイミーディエトをとりません。

〔フラグ〕

CY	MSB（Most Significant Bit）へのボローが生じた場合 1，そうでない場合 0
OV	Integer-Overflow を生じた場合 1，そうでない場合 0
S	結果が負になった場合 1，そうでない場合 0
Z	結果が 0 になった場合 1，そうでない場合 0
SAT	—

adf

【V850E2】

〔概要〕

条件付き加算（Add on Condition Flag）

〔形式〕

(1) adf imm4, reg1, reg2, reg3

(2) adf*cond* reg1, reg2, reg3

imm4 に指定できるものを次に示します。

- 4 ビット幅までの値を持つ絶対値式（0xd は指定できません）

〔機能〕

- adf

  第三オペランドに指定したレジスタのワード・データに，第二オペランドに指定したレジスタのワード・データを加算します。

  第一オペランドに指定した絶対値式の下位 4 ビットの値で示されるフラグ状態（「表 3 － 6 adfcond 命令一覧」参照）と加算結果のフラグ状態を比較します。値が一致した場合には、加算した結果に 1 を加算し，その結果を第四オペランドに指定したレジスタに格納します。値が一致しなかった場合には、加算した結果に 0 を加算し，その結果を第四オペランドに指定したレジスタに格納します。

- adf*cond*

  第二オペランドに指定したレジスタのワード・データに，第一オペランドに指定したレジスタのワード・データを加算します。

  cond 部分の文字列で示されるフラグ状態と加算結果のフラグの状態を比較します。値が一致した場合には、加算した結果に 1 を加算し，その結果を第三オペランドに指定したレジスタに格納します。値が一致しなかった場合には、加算した結果に 0 を加算し，その結果を第三オペランドに指定したレジスタに格納します。

〔説明〕

- adf 命令に対し，as850 では，機械語命令の adf 命令が 1 つ生成されます。
- adf*cond* 命令に対し，as850 では，対応する adf 命令が生成され（「表 3 － 6 adfcond 命令一覧」参照），(1) の形式に展開されます。

表 3 － 6 adf*cond* 命令一覧

命令	フラグ状態	フラグ状態の意味	命令展開
adfgt	((S xor OV) or Z) = 0	Greater than（signed）	adf 0xf
adfge	(S xor OV) = 0	Greater than or equal（signed）	adf 0xe
adflt	(S xor OV) = 1	Less than（signed）	adf 0x6
adfle	((S xor OV) or Z) = 1	Less than or equal（signed）	adf 0x7
adfh	(CY or Z) = 0	Higher（Greater than）	adf 0xb
adfnl	CY = 0	Not lower（Greater than or equal）	adf 0x9
adfl	CY = 1	Lower（Less than）	adf 0x1
adfnh	(CY or Z) = 1	Not higher（Less than or equal）	adf 0x3
adfe	Z = 1	Equal	adf 0x2
adfne	Z = 0	Not equal	adf 0xa
adfv	OV = 1	Overflow	adf 0x0
adfnv	OV = 0	No overflow	adf 0x8
adfn	S = 1	Negative	adf 0x4
adfp	S = 0	Positive	adf 0xc
adfc	CY = 1	Carry	adf 0x1
adfnc	CY = 0	No carry	adf 0x9
adfz	Z = 1	Zero	adf 0x2
adfnz	Z = 0	Not zero	adf 0xa
adft	always 1	Always 1	adf 0x5

〔フラグ〕

CY	MSB からのキャリーがあれば 1，そうでない場合 0
OV	オーバーフローが起こった場合 1，そうでない場合 0
S	演算結果が負の場合 1，そうでない場合 0
Z	演算結果が 0 の場合 1，そうでない場合 0
SAT	—

〔注意事項〕

- adf 命令の imm4 に 4 ビットの範囲を越える絶対値式を指定した場合，次のメッセージが出力され，指定された値の下位 4 ビットが用いられてアセンブルが続行されます。

```
W3011: illegal operand (range error in immediate).
```

- adf 命令の imm4 に 0xd を指定した場合，次のメッセージが出力され，アセンブルが中止されます。

```
E3261: illegal condition code.
```

sbf

【V850E2】

〔概要〕

条件付き減算（Subtract on Condition Flag）

〔形式〕

(1) sbf　　imm4, reg1, reg2, reg3

(2) sbf*cond*　　reg1, reg2, reg3

imm4 **に指定できるものを次に示します。**

- 4 **ビット幅までの値を持つ絶対値式**（0xd **は指定できません**）

〔機能〕

- sbf

  第三オペランドに指定したレジスタのワード・データから，第二オペランドに指定したレジスタのワード・データを減算します。

  第一オペランドに指定した絶対値式の下位 4 **ビットの値で示されるフラグ状態（「表** 3 － 7 sbfcond **命令一覧」参照）と減算結果のフラグ状態を比較します。値が一致した場合には、減算した結果から** 1 **を減算し，その結果を第四オペランドに指定したレジスタに格納します。値が一致しなかった場合には、減算した結果から** 0 **を減算し，その結果を第四オペランドに指定したレジスタに格納します。**

- sbf*cond*

  第二オペランドに指定したレジスタのワード・データから，第一オペランドに指定したレジスタのワード・データを減算します。

  cond **部分の文字列で示されるフラグ状態と減算結果のフラグの状態を比較します。値が一致した場合には、減算から結果に** 1 **を減算し，その結果を第三オペランドに指定したレジスタに格納します。値が一致しなかった場合には、減算した結果から** 0 **を減算し，その結果を第三オペランドに指定したレジスタに格納します。**

〔説明〕

- sbf **命令に対し，**as850 **では，機械語命令の** sbf **命令が** 1 **つ生成されます。**
- sbf*cond* **命令に対し，**as850 **では，対応する** sbf **命令が生成され（「表** 3 － 7 sbfcond **命令一覧」参照），**(1) **の形式に展開されます。**

表 3 － 7 sbf*cond* 命令一覧

命令	フラグ状態	フラグ状態の意味	命令展開
sbfgt	((S xor OV) or Z) = 0	Greater than（signed）	sbf 0xf
sbfge	(S xor OV) = 0	Greater than or equal（signed）	sbf 0xe
sbflt	(S xor OV) = 1	Less than（signed）	sbf 0x6
sbfle	((S xor OV) or Z) = 1	Less than or equal（signed）	sbf 0x7
sbfh	(CY or Z) = 0	Higher（Greater than）	sbf 0xb
sbfnl	CY = 0	Not lower（Greater than or equal）	sbf 0x9
sbfl	CY = 1	Lower（Less than）	sbf 0x1
sbfnh	(CY or Z) = 1	Not higher（Less than or equal）	sbf 0x3
sbfe	Z = 1	Equal	sbf 0x2
sbfne	Z = 0	Not equal	sbf 0xa
sbfv	OV = 1	Overflow	sbf 0x0
sbfnv	OV = 0	No overflow	sbf 0x8
sbfn	S = 1	Negative	sbf 0x4
sbfp	S = 0	Positive	sbf 0xc
sbfc	CY = 1	Carry	sbf 0x1
sbfnc	CY = 0	No carry	sbf 0x9
sbfz	Z = 1	Zero	sbf 0x2
sbfnz	Z = 0	Not zero	sbf 0xa
sbft	always 1	Always 1	sbf 0x5

〔フラグ〕

CY	MSB からのボローがあれば 1，そうでない場合 0
OV	オーバーフローが起こった場合 1，そうでない場合 0
S	演算結果が負の場合 1，そうでない場合 0
Z	演算結果が 0 の場合 1，そうでない場合 0
SAT	—

〔注意事項〕

- sbf 命令の imm4 に 4 ビットの範囲を越える絶対値式を指定した場合，次のメッセージが出力され，指定された値の下位 4 ビットが用いられてアセンブルが続行されます。

```
W3011: illegal operand (range error in immediate).
```

- sbf 命令の imm4 に 0xd を指定した場合，次のメッセージが出力され，アセンブルが中止されます。

```
E3261: illegal condition code.
```

3.4 飽和演算命令

この節では，飽和演算命令について説明します。

次に，この節において説明する命令を示します。

表 3 － 8 飽和演算命令

命令	意味
satadd	飽和加算
satsub	飽和減算
satsubi	飽和減算（イミーディエト）
satsubr	飽和逆減算

satadd

〔概要〕

飽和加算（Saturated Add）

〔形式〕

(1) satadd reg1, reg2

(2) satadd imm, reg2

(3) satadd reg1, reg2, reg3【V850E2】

imm に指定できるものを次に示します。

- 32 ビット幅までの値を持つ絶対値式
- 相対値式

〔機能〕

- (1) の形式

  第一オペランドに指定したレジスタ値と，第二オペランドに指定したレジスタ値を加算し，結果を第二オペランドに指定したレジスタに格納します。ただし，結果が正の最大値 0x7fffffff を越えた場合は 0x7fffffff を，負の最大値 0x80000000 を越えた場合は 0x80000000 を第二オペランドに指定したレジスタに格納し，SAT フラグに 1 を設定します。

- (2) の形式

  第一オペランドに指定した絶対値式，または相対値式の値と，第二オペランドに指定したレジスタの値を加算し，結果を第二オペランドに指定したレジスタに格納します。ただし，結果が正の最大値 0x7fffffff を越えた場合は 0x7fffffff を，負の最大値 0x80000000 を越えた場合は 0x80000000 を第二オペランドに指定したレジスタに格納し，SAT フラグに 1 を設定します。

- (3) の形式

  第一オペランドに指定したレジスタ値と，第二オペランドに指定したレジスタ値を加算し，結果を第三オペランドに指定したレジスタに格納します。ただし，結果が正の最大値 0x7fffffff を越えた場合は 0x7fffffff を，負の最大値 0x80000000 を越えた場合は 0x80000000 を第三オペランドに指定したレジスタに格納し，SAT フラグに 1 を設定します。

〔説明〕

- (1)，および (3) の形式の命令に対し，as850 では，機械語命令の satadd 命令が 1 つ生成されます。
- (2) の形式で imm に次のものを指定した場合，as850 では，機械語命令の satadd 命令[注]が 1 つ生成されます。

(a) -16 ～ +15 の範囲の絶対値式

satadd imm5, reg	satadd imm5, reg

注　機械語命令の satadd **命令は，第一オペランドにレジスタ，または** -16 ～ +15（0xfffffff0 ～ 0xf）**の範囲のイミーディエトをとります。**

- (2) **の形式で** imm **に次のものを指定した場合，**as850 **では，命令展開が行われ，複数個の機械語命令が生成されます。**

(a)　-16 ～ +15 **の範囲を越える絶対値式**

`satadd imm16, reg`	`movea  imm16, r0, r1` `satadd r1, reg`

(b)　- 32768 ～ +32767 **の範囲を越える絶対値式**

imm **の値の下位** 16 **ビットがすべて** 0 **の場合**

`satadd imm, reg`	`movhi  hi(imm), r0, r1` `satadd r1, reg`

上記以外の場合

`satadd imm, reg`	`movhi  hi1(imm), r0, r1`
	`movea  lo(imm), r1, r1`
	`satadd r1, reg`

(c)　- 32768 ～ +32767 **の範囲を越える絶対値式【**V850E**】**

imm **の値の下位** 16 **ビットがすべて** 0 **の場合**

`satadd imm, reg`	`movhi  hi(imm), r0, r1` `satadd r1, reg`

上記以外の場合

`satadd imm, reg`	`mov    imm, r1` `satadd r1, reg`

(d)　sdata ／ sbss **属性セクションに定義を持つラベルの** $label **を持つ相対値式**

`satadd $label, reg`	`movea  $label, r0, r1` `satadd r1, reg`

(e)　#label**，または** label **を持つ相対値式，および** sdata ／ sbss **属性セクションに定義を持たないラベルの** $label **を持つ相対値式**

`satadd #label, reg`	`movhi  hi1(#label), r0, r1`
	`movea  lo(#label), r1, r1`
	`satadd r1, reg`

satadd label, reg	movhi hi1(label), r0, r1
	movea lo(label), r1, r1
	satadd r1, reg

satadd $label, reg	movhi hi1($label), r0, r1
	movea lo($label), r1, r1
	satadd r1, reg

(f) #label，または label を持つ相対値式，および sdata ／ sbss 属性セクションに定義を持たないラベルの $label を持つ相対値式【V850E】

satadd #label, reg	mov #label, r1 satadd r1, reg

satadd label, reg	mov label, r1 satadd r1, reg

satadd $label, reg	mov $label, r1 satadd r1, reg

〔フラグ〕

CY	MSB（Most Significant Bit）からのキャリーを生じた場合 1，そうでない場合 0
OV	Integer-Overflow を生じた場合 1，そうでない場合 0
S	結果が負になった場合 1，そうでない場合 0
Z	結果が 0 になった場合 1，そうでない場合 0
SAT	OV = 1 になった場合 1，そうでない場合 —

〔注意事項〕

- (1) ，および (2) の形式の命令に対し，ターゲット・デバイスが V850Ex の場合，第二オペランドに r0 を指定すると，次のメッセージが出力され，アセンブルが中止されます。

```
E3240: illegal operand (can not use r0 as destination in V850E mode)
```

なお，V850Ex 以外の場合，次のメッセージが出力され，アセンブルが続行されます。

```
W3013: register r0 used as destination register
```

satsub

〔概要〕

飽和減算（Saturated Subtract）

〔形式〕

(1) satsub reg1, reg2

(2) satsub imm, reg2

(3) satsub reg1, reg2, reg3【V850E2】

imm に指定できるものを次に示します。

- 32 ビット幅までの値を持つ絶対値式
- 相対値式

〔機能〕

- (1) の形式

  第二オペランドに指定したレジスタ値から，第一オペランドに指定したレジスタ値を減算し，結果を第二オペランドに指定したレジスタに格納します。ただし，結果が正の最大値 0x7fffffff を越えた場合は 0x7fffffff を，負の最大値 0x80000000 を越えた場合は 0x80000000 を第二オペランドに指定したレジスタに格納し，SAT フラグに 1 を設定します。

- (2) の形式

  第二オペランドに指定したレジスタ値から，第一オペランドに指定した絶対値式，または相対値式の値を減算し，結果を第二オペランドに指定したレジスタに格納します。ただし，結果が正の最大値 0x7fffffff を越えた場合は 0x7fffffff を，負の最大値 0x80000000 を越えた場合は 0x80000000 を第二オペランドに指定したレジスタに格納し，SAT フラグに 1 を設定します。

- (3) の形式

  第二オペランドに指定したレジスタ値から，第一オペランドに指定したレジスタ値を減算し，結果を第三オペランドに指定したレジスタに格納します。ただし，結果が正の最大値 0x7fffffff を越えた場合は 0x7fffffff を，負の最大値 0x80000000 を越えた場合は 0x80000000 を第三オペランドに指定したレジスタに格納し，SAT フラグに 1 を設定します。

〔説明〕

- (1)，および (3) の形式の命令に対し，as850 では，機械語命令の satsub 命令が 1 つ生成されます。
- (2) の形式で imm に次のもの，as850 では，命令展開が行われ，1 つ，または複数個の機械語命令が生成されます[注]。

(a) 0

`satsub 0, reg`	`satsub r0, reg`

(b) - 32768 ～ +32767 の範囲の絶対値式

`satsub imm16, reg`	`satsubi imm16, reg, reg`

(c) - 32768 ～ +32767 の範囲を越える絶対値式

imm の値の下位 16 ビットがすべて 0 の場合

`satsub imm, reg`	`movhi hi(imm), r0, r1` `satsub r1, reg`

上記以外の場合

`satsub imm, reg`	`movhi hi1(imm), r0, r1`
	`movea lo(imm), r1, r1`
	`satsub r1, reg`

(d) - 32768 ～ +32767 の範囲を越える絶対値式【V850E】

imm の値の下位 16 ビットがすべて 0 の場合

`satsub imm, reg`	`movhi hi(imm), r0, r1` `satsub r1, reg`

上記以外の場合

`satsub imm, reg`	`mov imm, r1` `satsub r1, reg`

(e) data ／ sbss 属性セクションに定義を持つラベルの $label を持つ相対値式

`satsub $label, reg`	`satsubi $label, reg, reg`

(f) #label，または label を持つ相対値式，および sdata ／ sbss 属性セクションに定義を持たないラベルの $label を持つ相対値式

`satsub #label, reg`	`movhi hi1(#label), r0, r1`
	`movea lo(#label), r1, r1`
	`satsub r1, reg`

`satsub label, reg`	`movhi hi1(label), r0, r1`
	`movea lo(label), r1, r1`
	`satsub r1, reg`

`satsub $label, reg`	`movhi hi1($label), r0, r1`
	`movea lo($label), r1, r1`
	`satsub r1, reg`

(g) #label，または label を持つ相対値式，および sdata ／ sbss 属性セクションに定義を持たないラベルの $label を持つ相対値式【V850E】

`satsub #label, reg`	`mov #label, r1` `satsub r1, reg`
`satsub label, reg`	`mov label, r1` `satsub r1, reg`
`satsub $label, reg`	`mov $label, r1` `satsub r1, reg`

注 機械語命令の satsub 命令は，オペランドにイミーディエトをとりません。

〔フラグ〕

CY	MSB（Most Significant Bit）へのボローが生じた場合 1，そうでない場合 0
OV	Integer-Overflow を生じた場合 1，そうでない場合 0
S	結果が負になった場合 1，そうでない場合 0
Z	結果が 0 になった場合 1，そうでない場合 0
SAT	OV = 1 になった場合 1，そうでない場合 —

〔注意事項〕

- (1)，および (2) の形式の命令に対し，ターゲット・デバイスが V850Ex の場合，第二オペランドに r0 を指定すると，次のメッセージが出力され，アセンブルが中止されます。

```
E3240: illegal operand (can not use r0 as destination in V850E mode)
```

なお，V850Ex 以外の場合，次のメッセージが出力され，アセンブルが続行されます。

```
W3013: register r0 used as destination register
```

satsubi

〔概要〕

飽和減算（Saturated Subtract Immediate）

〔形式〕

(1) satsubi imm, reg1, reg2

imm **に指定できるものを次に示します。**

- 32 **ビット幅までの値を持つ絶対値式**
- **相対値式**
- **上記のものに** hi( ) **，** lo( ) **，または** hi1( ) **を適用したもの**

〔機能〕

第一オペランドに指定した絶対値式，相対値式，あるいは，hi( ) **，**lo( ) **，または** hi1( ) **を適用したものの値を，第二オペランドに指定したレジスタ値から減算し，結果を第三オペランドに指定したレジスタに格納します。ただし，結果が正の最大値** 0x7fffffff **を越えた場合は** 0x7fffffff **を，負の最大値** 0x80000000 **を越えた場合は** 0x80000000 **を第三オペランドに指定したレジスタに格納し，**SAT **フラグに** 1 **を設定します。**

〔説明〕

- imm **に次のものを指定した場合，**as850 **では，機械語命令の** satsubi **命令**[注]**が** 1 **つ生成されます。**

(a) -32768 **～** +32767 **の範囲の絶対値式**

`satsubi imm16, reg1, reg2`	`satsubi imm16, reg1, reg2`

(b) sdata **／** sbss **属性セクションに定義を持つラベルの** $label **を持つ相対値式**

`satsubi $label, reg1, reg2`	`satsubi $label, reg1, reg2`

(c) !label**，または** %label **を持つ相対値式**

`satsubi !label, reg1, reg2`	`satsubi !label, reg1, reg2`

`satsubi %label, reg1, reg2`	`satsubi %label, reg1, reg2`

(d) hi( ) ，lo( ) ，または hi1( ) を適用したもの

satsubi imm16, reg1, reg2	satsubi imm16, reg1, reg2

注 機械語命令の satsubi 命令は，第一オペランドに -32768 ～ +32767（0xffff8000 ～ 0x7fff）の範囲のイミーディエトをとります。

- imm に次のものを指定した場合，ass850 では命令展開が行われ，1 つ，または複数個の機械語命令が生成されます。

(a) -32768 ～ +32767 の範囲を越える絶対値式

imm の値の下位 16 ビットがすべて 0 の場合

satsubi imm, reg1, reg2	movhi hi(imm), r0, reg2 satsubr reg1, reg2

imm の値の下位 16 ビットがすべて 0，ただし reg2 が r0 の場合

satsubi imm, reg1, r0	movhi hi(imm), r0, r1 satsubr reg1, r1

上記以外の場合

satsubi imm, reg1, reg2	movhi hi1(imm), r0, r1
	movea lo(imm), r1, reg2
	satsubr reg1, reg2

上記以外，ただし reg2 が r0 の場合

satsubi imm, reg1, reg2	movhi hi1(imm), r0, r1
	movea lo(imm), r1, r1
	satsubr reg1, r1

(b) -32768 ～ +32767 の範囲を越える絶対値式【V850E】

imm の値の下位 16 ビットがすべて 0 の場合

satsubi imm, reg1, reg2 →	movhi hi(imm), r0, reg2 satsubr reg1, reg2

imm の値の下位 16 ビットがすべて 0，ただし reg2 が r0 の場合

satsubi imm, reg1, r0	movhi hi(imm), r0, r1 satsubr reg1, r1

上記以外の場合

satsubi imm, reg1, reg2 →	mov imm, reg2 satsubr reg1, reg2

上記以外，ただし reg2 が r0 の場合

satsubi imm, reg1, reg2	mov imm, r1 satsubr reg1, r1

(c) #label，または label を持つ相対値式，および sdata ／ sbss 属性セクションに定義を持たないラベルの $label を持つ相対値式

reg2 が r0 の場合

satsubi #label, reg1, r0	movhi hi1(#label), r0, r1
	movea lo(#label), r1, r1
	satsubr reg1, r1

satsubi label, reg1, r0	movhi hi1(label), r0, r1
	movea lo(label), r1, r1
	satsubr reg1, r1

satsubi $label, reg1, r0	movhi hi1($label), r0, r1
	movea lo($label), r1, r1
	satsubr reg1, r1

上記以外の場合

satsubi #label, reg1, r	movhi hi1(#label), r0, r1
	movea lo(#label), r1, reg2
	satsubr reg1, reg2

satsubi label, reg1, reg2	movhi hi1(label), r0, r1
	movea lo(label), r1, reg2
	satsubr reg1, reg2

satsubi $label, reg1, reg2	movhi hi1($label), r0, r1
	movea lo($label), r1, reg2
	satsubr reg1, reg2

(d) #label，または label を持つ相対値式，および sdata ／ sbss 属性セクションに定義を持たないラベルの $label を持つ相対値式【V850E】

reg2 が r0 の場合

satsubi #label, reg1, r0 →	movhi #label, r1 satsubr reg1, r1

satsubi label, reg1, r0	mov label, r1 satsubr reg1, r1

satsubi $label, reg1, r0	mov $label, r1 satsubr reg1, r1

上記以外の場合

`satsubi #label, reg1, reg2` →	`movhi  #label, reg2` `satsubr reg1, reg2`

`satsubi label, reg1, reg2`	`mov    label, reg2` `satsubr reg1, reg2`

`satsubi $label, reg1, reg2`	`mov    $label, reg2` `satsubr reg1, reg2`

〔フラグ〕

CY	MSB（Most Significant Bit）**へのボローが生じた場合** 1**，そうでない場合** 0
OV	Integer-Overflow **を生じた場合** 1**，そうでない場合** 0
S	**結果が負になった場合** 1**，そうでない場合** 0
Z	**結果が** 0 **になった場合** 1**，そうでない場合** 0
SAT	OV = 1 **になった場合** 1**，そうでない場合** —

〔注意事項〕

- **ターゲット・デバイスが** V850Ex **の場合，第二オペランドに** r0 **を指定すると，次のメッセージが出力され，アセンブルが中止されます。**

```
E3240: illegal operand (can not use r0 as destination in V850E mode)
```

なお，V850Ex **以外の場合，次のメッセージが出力され，アセンブルが続行されます。**

```
W3013: register r0 used as destination register
```

satsubr

〔概要〕

飽和逆減算（Saturated Subtract Reverse）

〔形式〕

(1) satsubr reg1, reg2

(2) satsubr imm, reg2

imm に指定できるものを次に示します。

- 32 ビット幅までの値を持つ絶対値式
- 相対値式

〔機能〕

- (1) の形式

  第一オペランドに指定したレジスタ値から，第二オペランドに指定したレジスタ値を減算し，結果を第二オペランドに指定したレジスタに格納します。ただし，結果が正の最大値 0x7fffffff を越えた場合は 0x7fffffff を，負の最大値 0x80000000 を越えた場合は 0x80000000 を第二オペランドに指定したレジスタに格納し，SAT フラグに 1 を設定します。

- (2) の形式

  第一オペランドに指定した絶対値式，または相対値式の値から，第二オペランドに指定したレジスタの値を減算し，結果を第二オペランドに指定したレジスタに格納します。ただし，結果が正の最大値 0x7fffffff を越えた場合は 0x7fffffff を，負の最大値 0x80000000 を越えた場合は 0x80000000 を第二オペランドに指定したレジスタに格納し，SAT フラグに 1 を設定します。

〔説明〕

- (1) の形式の命令に対し，as850 では，機械語命令の satsubr 命令が 1 つ生成されます。
- (2) の形式で imm に次のものを指定した場合，as850 では，命令展開が行われ，1 つ，または複数個の機械語命令が生成されます[注]。

(a) 0

`satsubr 0, reg`	`satsubr r0, reg`

(b) - 16 ～ +15 の範囲の絶対値式（0 以外 ）

`satsubr imm5, reg`	`mov imm5, r1` `satsubr r1, reg`

(c) - 16 ～ +15 の範囲の絶対値式（0 以外 ）【V850E】

satsubr imm5, reg	mov imm5, r1 satsubr r1, reg

(d) - 16 ～ +15 の範囲を越え，-32768 ～ +32767 の範囲の絶対値式

satsubr imm16, reg	movea imm16, r0, r1 satsubr r1, reg

(e) imm に -32768 ～ +32767 の範囲を越える絶対値式

imm の値の下位 16 ビットがすべて 0 の場合

satsubr imm, reg	movhi hi(imm), r0, r1 satsubr r1, reg

上記以外の場合

satsubr imm, reg	movhi hi1(imm), r0, r1
	movea lo(imm), r1, r1
	satsubr r1, reg

(f) imm に -32768 ～ +32767 の範囲を越える絶対値式【V850E】

imm の値の下位 16 ビットがすべて 0 の場合

satsubr imm, reg	movhi hi(imm), r0, r1 satsubr r1, reg

上記以外の場合

satsubr imm, reg	mov imm, r1 satsubr r1, reg

(g) sdata ／ sbss 属性セクションに定義を持つラベルの $label を持つ相対値式

satsubr $label, reg	movea $label, r0, r1 satsubr r1, reg

(h) #label，または label を持つ相対値式，および sdata ／ sbss 属性セクションに定義を持たないラベルの $label を持つ相対値式

satsubr #label, reg	movhi hi1(#label), r0, r1
	movea lo(#label), r1, r1
	satsubr r1, reg

satsubr label, reg	movhi hi1(label), r0, r1
	movea lo(label), r1, r1
	satsubr r1, reg

satsubr $label, reg	movhi hi1($label), r0, r1
	movea lo($label), r1, r1
	satsubr r1, reg

(i) #label，または label を持つ相対値式，および sdata ／ sbss 属性セクションに定義を持たないラベルの $label を持つ相対値式【V850E】

satsubr #label, reg	mov #label, r1 satsubr r1, reg
satsubr label, reg	mov label, r1 satsubr r1, reg
satsubr $label, reg	mov $label, r1 satsubr r1, reg

注 機械語命令の satsubr 命令は，オペランドにイミーディエトをとりません。

〔フラグ〕

CY	MSB（Most Significant Bit）へのボローが生じた場合 1，そうでない場合 0
OV	Integer-Overflow を生じた場合 1，そうでない場合 0
S	結果が負になった場合 1，そうでない場合 0
Z	結果が 0 になった場合 1，そうでない場合 0
SAT	OV = 1 になった場合 1，そうでない場合 —

〔注意事項〕

- **ターゲット・デバイスが** V850Ex **の場合，第二オペランドに** r0 **を指定すると，次のメッセージが出力され，アセンブルが中止されます。**

```
E3240: illegal operand (can not use r0 as destination in V850E mode)
```

なお，V850Ex **以外の場合，次のメッセージが出力され，アセンブルが続行されます。**

```
W3013: register r0 used as destination register
```

3.5 論理演算命令

この節では，論理演算命令について説明します。

次に，この節において説明する命令を示します。

表 3 － 9 論理演算命令

命令	意味
and	論理積
andi	論理積（イミーディエト）
bsh	ハーフワード・データのバイト・スワップ【V850E】
bsw	ワード・データのバイト・スワップ【V850E】
hsh	ハーフワード・データのハーフワード・スワップ【V850E2】
hsw	ワード・データのハーフワード・スワップ【V850E】
not	論理否定（1 の補数をとる）
or	論理和
ori	論理和（イミーディエト）
sar	算術右シフト
shl	論理左シフト
shr	論理右シフト
sxb	バイト・データの符号拡張【V850E】
sxh	ハーフワード・データの符号拡張【V850E】
tst	テスト
xor	排他的論理和
xori	排他的論理和（イミーディエト）
zxb	バイト・データのゼロ拡張【V850E】
zxh	ハーフワード・データのゼロ拡張【V850E】
sch0l	MSB 側からのビット（0）検索【V850E2】
sch0r	LSB 側からのビット（0）検索【V850E2】
sch1l	MSB 側からのビット（1）検索【V850E2】
sch1r	LSB 側からのビット（1）検索【V850E2】

and

〔概要〕

論理積（And）

〔形式〕

(1) and reg1, reg2

(2) and imm, reg2

imm に指定できるものを次に示します。

- 32 ビット幅までの値を持つ絶対値式
- 相対値式

〔機能〕

- (1) の形式

  第一オペランドに指定したレジスタ値と，第二オペランドに指定したレジスタ値の論理積をとり，結果を第二オペランドに指定したレジスタに格納します。

- (2) の形式

  第一オペランドに指定した絶対値式，または相対値式の値と，第二オペランドに指定したレジスタ値の論理積をとり，結果を第二オペランドに指定したレジスタに格納します。

〔説明〕

- (1) の形式の命令に対し，as850 では，機械語命令の and 命令が 1 つ生成されます。
- (2) の形式で imm に次のものを指定した場合，as850 では，命令展開が行われ，1 つ，または複数個の機械語命令が生成されます[注]。

(a) 0

`and 0, reg`	`and r0, reg`

(b) 1 ～ 65535 の範囲の絶対値式

`and imm16, reg`	`andi imm16, reg, reg`

(c) -16 ～ -1 の範囲の絶対値式

`and imm5, reg`	`mov imm5, r1` `and r1, reg`

(d) -32768 ～ -17 の範囲の絶対値式

and imm16, reg	movea imm16, r0, r1 and r1, reg

(e) 上記の範囲を越える絶対値式

imm の値の下位 16 ビットがすべて 0 の場合

and imm, reg	movhi hi(imm), r0, r1 and r1, reg

上記以外の場合

and imm, reg	movhi hi1(imm), r0, r1
	movea lo(imm), r1, r1
	and r1, reg

(f) 上記の範囲を越える絶対値式【V850E】

imm の値の下位 16 ビットがすべて 0 の場合

and imm, reg	movhi hi(imm), r0, r1 and r1, reg

上記以外の場合

and imm, reg	mov imm, r0, r1 and r1, reg

(g) sdata ／ sbss 属性セクションに定義を持つラベルの $label を持つ相対値式

and $label, reg	movea $label, r0, r1 and r1, reg

(h) #label，または label を持つ相対値式，および sdata ／ sbss 属性セクションに定義を持たないラベルの $label を持つ相対値式

and #label, reg	movhi hi1(#label), r0, r1
	movea lo(#label), r1, r1
	and r1, reg

and label, reg	movhi hi1(label), r0, r1
	movea lo(label), r1, r1
	and r1, reg

and $label, reg	movhi hi1($label), r0, r1
	movea lo($label), r1, r1
	and r1, reg

(i) #label，または label を持つ相対値式，および sdata ／ sbss 属性セクションに定義を持たないラベルの $label を持つ相対値式【V850E】

`and #label, reg`	`mov #label, r1` `and r1, reg`

`and label, reg`	`mov label, r1` `and r1, reg`

`and $label, reg`	`mov $label, r1` `and r1, reg`

注 機械語命令の and 命令は，オペランドにイミーディエトをとりません。

〔フラグ〕

CY	—
OV	0
S	演算結果のワード・データの MSB が 1 の場合 1，そうでない場合 0
Z	結果が 0 になった場合 1，そうでない場合 0
SAT	—

andi

〔概要〕

論理積（And Immediate）

〔形式〕

(1) andi imm, reg1, reg2

imm に指定できるものを次に示します。

- 32 ビット幅までの値を持つ絶対値式
- 相対値式
- 上記のものに hi( ) ，lo( ) ，または hi1( ) を適用したもの

〔機能〕

第一オペランドに指定した絶対値式，相対値式，あるいは，hi( ) ，lo( ) ，または hi1( ) を適用した値と，第二オペランドに指定したレジスタ値の論理積をとり，結果を第三オペランドに指定したレジスタに格納します。

〔説明〕

- imm に次のものを指定した場合，as850 では，機械語命令の andi 命令[注]が 1 つ生成されます。

(a) 0 ～ 65535 の範囲の絶対値式

`andi imm16, reg1, reg2`	`andi imm16, reg1, reg2`

(b) !label，または %label を持つ相対値式

`andi !label, reg1, reg2`	`andi !label, reg1, reg2`

`andi %label, reg1, reg2`	`andi %label, reg1, reg2`

(c) hi( ) ，lo( ) ，または hi1( ) を適用したもの

`andi imm16, reg1, reg2`	`andi imm16, reg1, reg2`

注 機械語命令の andi 命令は，第一オペランドに 0 ～ 65535（0 ～ 0xffff）のイミーディエトをとります。

- imm に次のものを指定した場合，as850 では，命令展開が行われ，1 つ，または複数個の機械語命令が生成されます。

(a) -16 ～ -1 の範囲の絶対値式

`andi imm5, reg1, reg2`	`mov imm5, reg2` `and reg1, reg2`

(b) -32768 ～ -17 の範囲の絶対値式

reg2 が r0 の場合

`andi imm16, reg1, r0`	`movea imm16, r0, r1` `and reg1, r1`

上記以外の場合

`andi imm16, reg1, reg2`	`movea imm16, r0, reg2` `and reg1, reg2`

(c) 上記の範囲を越える絶対値式

imm の値の下位 16 ビットがすべて 0 の場合

`andi imm, reg1, reg2`	`movhi hi(imm), r0, reg2` `and reg1, reg2`

imm の値の下位 16 ビットがすべて 0，ただし reg2 が r0 の場合

`andi imm, reg1, r0`	`movhi hi(imm), r0, r1` `and reg1, r1`

上記以外の場合

`andi imm, reg1, reg2`	`movhi hi1(imm), r0, r1`
	`movea lo(imm), r1, reg2`
	`and reg1, reg2`

上記以外，ただし reg2 が r0 の場合

`andi imm, reg1, r0`	`movhi hi1(imm), r0, r1`
	`movea lo(imm), r1, r1`
	`and reg1, r1`

(d) 上記の範囲を越える絶対値式【V850E】

imm の値の下位 16 ビットがすべて 0 の場合

`andi imm, reg1, reg2`	`movhi hi(imm), r0, reg2` `and reg1, reg2`

imm の値の下位 16 ビットがすべて 0，ただし reg2 が r0 の場合

`andi imm, reg1, r0`	`movhi hi(imm), r0, r1` `and reg1, r1`

上記以外の場合

`andi imm, reg1, reg2`	`mov imm, reg2` `and reg1, reg2`

上記以外，ただし reg2 **が** r0 **の場合**

`andi imm, reg1, reg2`	`mov imm, r1` `and reg1, r1`

(e) sdata ／ sbss **属性セクションに定義を持つラベルの** $label **を持つ相対値式**

reg2 **が** r0 **の場合**

`andi $label, reg1, r0`	`movea $label, r0, r1` `and reg1, r1`

上記以外の場合

`andi $label, reg1, reg2`	`movea $label, r0, reg2` `and reg1, reg2`

(f) #label**，または** label **を持つ相対値式，および** sdata ／ sbss **属性セクションに定義を持たないラベルの** $label **を持つ相対値式**

reg2 **が** r0 **の場合**

`andi #label, reg1, r0`	`movhi hi1(#label), r0, r1`
	`movea lo(#label), r1, r1`
	`and reg1, r1`

`andi label, reg1, r0`	`movhi hi1(label), r0, r1`
	`movea lo(label), r1, r1`
	`and reg1, r1`

`andi $label, reg1, r0`	`movhi hi1($label), r0, r1`
	`movea lo($label), r1, r1`
	`and reg1, r1`

上記以外の場合

`andi #label, reg1, reg2`	`movhi hi1(#label), r0, r1`
	`movea lo(#label), r1, reg2`
	`and reg1, reg2`

`andi label, reg1, reg2`	`movhi hi1(label), r0, r1`
	`movea lo(label), r1, reg2`
	`and reg1, reg2`

`andi $label, reg1, reg2`	`movhi hi1($label), r0, r1`
	`movea lo($label), r1, reg2`
	`and reg1, reg2`

(g) #label，または label を持つ相対値式，および sdata ／ sbss 属性セクションに定義を持たないラベルの $label を持つ相対値式【V850E】

reg2 が r0 の場合

`andi #label, reg1, r0`	`mov #label, r1` `and reg1, r1`
`andi label, reg1, r0`	`mov label, r1` `and reg1, r1`
`andi $label, reg1, r0`	`mov $label, r1` `and reg1, r1`

上記以外の場合

`andi #label, reg1, reg2`	`mov #label, reg2` `and reg1, reg2`
`andi label, reg1, reg2`	`mov label, reg2` `and reg1, reg2`
`andi $label, reg1, reg2`	`mov $label, reg2` `and reg1, reg2`

〔フラグ〕

CY	—
OV	0
S	演算結果のワード・データの MSB が 1 の場合 1，そうでない場合 0
Z	結果が 0 になった場合 1，そうでない場合 0
SAT	—

bsh

【V850E】

〔概要〕

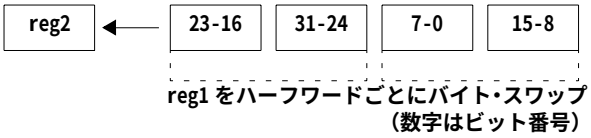
ハーフワード・データのバイト・スワップ（Byte Swap Half-word）

〔形式〕

(1) bsh reg1, reg2

〔機能〕

第一オペランドに指定したレジスタ値を，ハーフワード単位でバイト・スワップして，第二オペランドに指定したレジスタに格納します。



〔説明〕

- as850 では，機械語命令の bsh 命令が 1 つ生成されます。

〔フラグ〕

CY	レジスタの下位ハーフワード中の 1 つ以上のバイトが 0 の場合 1，そうでない場合 0
OV	0
S	演算結果のワード・データの MSB が 1 の場合 1，そうでない場合 0
Z	演算結果の下位ハーフ・ワード・データが 0 になった場合 1，そうでない場合 0
SAT	—

bsw

【V850E】

〔概要〕

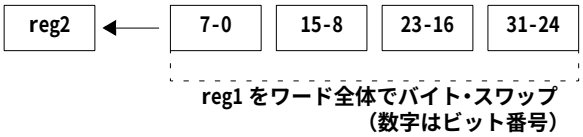
ワード・データのバイト・スワップ（Byte Swap Word）

〔形式〕

(1) bsw reg1, reg2

〔機能〕

第一オペランドに指定したレジスタ値を，バイト・スワップして，第二オペランドに指定したレジスタに格納します。



reg1 をワード全体でバイト・スワップ
（数字はビット番号）

〔説明〕

- as850 では，機械語命令の bsw 命令が 1 つ生成されます。

〔フラグ〕

CY	レジスタのワード中の 1 つ以上のバイトが 0 の場合 1，そうでない場合 0
OV	0
S	演算結果のワード・データの MSB が 1 の場合 1，そうでない場合 0
Z	演算結果のワード・データが 0 になった場合 1，そうでない場合 0
SAT	—

hsh

【V850E2】

〔概要〕

ハーフワード・データのハーフワード・スワップ（Half-word Swap Half-word）

〔形式〕

(1) hsh reg2, reg3

〔機能〕

第一オペランドに指定したレジスタ値を第二オペランドに指定したレジスタに格納し，フラグの判定結果を PSW レジスタに格納します。

〔説明〕

- as850 では，機械語命令の hsh 命令が 1 つ生成されます。

〔フラグ〕

CY	演算結果の下位ハーフワードが 0 の場合 1，そうでない場合 0
OV	0
S	演算結果のワード・データの MSB が 1 の場合 1，そうでない場合 0
Z	演算結果の下位ハーフワードが 0 の場合 1，そうでない場合 0
SAT	—

hsw

【V850E】

〔概要〕

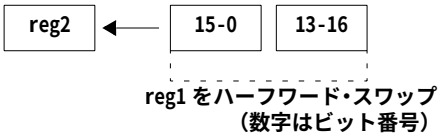
ワード・データのハーフワード・スワップ（Half-word Swap Word）

〔形式〕

(1) hsw reg1, reg2

〔機能〕

第一オペランドに指定したレジスタ値を，ハーフワード・スワップして，第二オペランドに指定したレジスタに格納します。



reg1 をハーフワード・スワップ
（数字はビット番号）

〔説明〕

- as850 では，機械語命令の hsw 命令が 1 つ生成されます。

〔フラグ〕

CY	レジスタのワード中の 1 つ以上のバイトが 0 の場合 1，そうでない場合 0
OV	0
S	演算結果のワード・データの MSB が 1 の場合 1，そうでない場合 0
Z	演算結果のワード・データが 0 になった場合 1，そうでない場合 0
SAT	—

not

〔概要〕

論理否定（1 の補数をとる）（Not）

〔形式〕

(1) not reg1, reg2

(2) not imm, reg2

imm に指定できるものを次に示します。

- 32 ビット幅までの値を持つ絶対値式
- 相対値式

〔機能〕

- (1) の形式

  第一オペランドに指定したレジスタ値の論理否定（1 の補数）をとり，結果を第二オペランドに指定したレジスタに格納します。

- (2) の形式

  第一オペランドに指定した絶対値式，または相対値式の値の論理否定（1 の補数）をとり，結果を第二オペランドに指定したレジスタに格納します。

〔説明〕

- (1) の形式の命令に対し，as850 では，機械語命令の not 命令が 1 つ生成されます。
- (2) の形式で imm に次のものを指定した場合，as850 では，命令展開が行われ，1 つ，または複数個の機械語命令が生成されます[注]。

(a) 0

`not 0, reg`	`not r0, reg`

(b) - 16 〜 +15 の範囲の絶対値式（0 以外 ）

`not imm5, reg`	`mov imm5, r1` `not r1, reg`

(c) - 16 〜 +15 の範囲の絶対値式（0 以外 ）【V850E】

`not imm5, reg`	`mov imm5, r1` `not r1, reg`

(d) - 16 ～ +15 の範囲を越え，-32768 ～ +32767 の範囲の絶対値式

not imm16, reg	movea imm16, r0, r1 not r1, reg

(e) imm に -32768 ～ +32767 の範囲を越える絶対値式

imm の値の下位 16 ビットがすべて 0 の場合

not imm, reg	movhi hi(imm), r0, r1 not r1, reg

上記以外の場合

not imm, reg	movhi hi1(imm), r0, r1
	movea lo(imm), r1, r1
	not r1, reg

(f) imm に -32768 ～ +32767 の範囲を越える絶対値式【V850E】

imm の値の下位 16 ビットがすべて 0 の場合

not imm, reg	movhi hi(imm), r0, r1 not r1, reg

上記以外の場合

not imm, reg	mov imm, r1 not r1, reg

(g) sdata ／ sbss 属性セクションに定義を持つラベルの $label を持つ相対値式

not $label, reg	movea $label, r0, r1 not r1, reg

(h) #label，または label を持つ相対値式，および sdata ／ sbss 属性セクションに定義を持たないラベルの $label を持つ相対値式

`not  #label, reg`	`movhi  hi1(#label), r0, r1`
	`movea  lo(#label), r1, r1`
	`not  r1, reg`

`not  label, reg`	`movhi  hi1(label), r0, r1`
	`movea  lo(label), r1, r1`
	`not  r1, reg`

`not  $label, reg`	`movhi  hi1($label), r0, r1`
	`movea  lo($label), r1, r1`
	`not  r1, reg`

(i) #label，または label を持つ相対値式，および sdata ／ sbss 属性セクションに定義を持たないラベルの $label を持つ相対値式【V850E】

`not  #label, reg`	`mov  #label, r1` `not  r1, reg`

`not  label, reg`	`mov  label, r1` `not  r1, reg`

`not  $label, reg`	`mov  $label, r1` `not  r1, reg`

注　機械語命令の not 命令は，オペランドにイミーディエトをとりません。

〔フラグ〕

CY	—
OV	0
S	演算結果のワード・データの MSB が 1 の場合 1，そうでない場合 0
Z	結果が 0 になった場合 1，そうでない場合 0
SAT	—

or

〔概要〕

論理和（Or）

〔形式〕

(1) or reg1, reg2

(2) or imm, reg2

imm に指定できるものを次に示します。

- 32 ビット幅までの値を持つ絶対値式
- 相対値式

〔機能〕

- (1) の形式

  第一オペランドに指定したレジスタ値と，第二オペランドに指定したレジスタ値の論理和をとり，結果を第二オペランドに指定したレジスタに格納します。

- (2) の形式

  第一オペランドに指定した絶対値式，または相対値式の値と，第二オペランドに指定したレジスタ値の論理和をとり，結果を第二オペランドに指定したレジスタに格納します。

〔説明〕

- (1) の形式の命令に対し，as850 では，機械語命令の or 命令が 1 つ生成されます。
- (2) の形式で imm に次のものを指定した場合，as850 では，命令展開が行われ，1 つ，または複数個の機械語命令が生成されます[注]。

(a) 0

`or 0, reg`	`or r0, reg`

(b) 1 ～ 65535 の範囲の絶対値式

`or imm16, reg`	`ori imm16, reg, reg`

(c) -16 ～ -1 の範囲の絶対値式

`or imm5, reg`	`mov imm5, r1` `or r1, reg`

(d) -32768 ～ -17 の範囲の絶対値式

`or imm16, reg`	`movea imm16, r0, r1` `or r1, reg`

(e) 上記の範囲を越える絶対値式

imm の値の下位 16 ビットがすべて 0 の場合

`or imm, reg`	`movhi hi(imm), r0, r1` `or r1, reg`

上記以外の場合

`or imm, reg`	`movhi hi1(imm), r0, r1`
	`movea lo(imm), r1, r1`
	`or r1, reg`

(f) 上記の範囲を越える絶対値式【V850E】

imm の値の下位 16 ビットがすべて 0 の場合

`or imm, reg`	`movhi hi(imm), r0, r1` `or r1, reg`

上記以外の場合

`or imm, reg`	`mov imm, r0, r1` `or r1, reg`

(g) sdata ／ sbss 属性セクションに定義を持つラベルの $label を持つ相対値式

`or $label, reg`	`movea $label, r0, r1` `or r1, reg`

(h) #label，または label を持つ相対値式，および sdata ／ sbss 属性セクションに定義を持たないラベルの $label を持つ相対値式

`or #label, reg`	`movhi hi1(#label), r0, r1`
	`movea lo(#label), r1, r1`
	`or r1, reg`

`or label, reg`	`movhi hi1(label), r0, r1`
	`movea lo(label), r1, r1`
	`or r1, reg`

`or $label, reg`	`movhi hi1($label), r0, r1`
	`movea lo($label), r1, r1`
	`or r1, reg`

(i) #label，または label を持つ相対値式，および sdata ／ sbss 属性セクションに定義を持たないラベルの $label を持つ相対値式【V850E】

`or #label, reg`	`mov #label, r1` `or r1, reg`

`or label, reg`	`mov label, r1` `or r1, reg`

`or $label, reg`	`mov $label, r1` `or r1, reg`

注 機械語命令の or 命令は，オペランドにイミーディエトをとりません。

〔フラグ〕

CY	—
OV	0
S	演算結果のワード・データの MSB が 1 の場合 1，そうでない場合 0
Z	結果が 0 になった場合 1，そうでない場合 0
SAT	—

ori

〔概要〕

論理和（Or Immediate）

〔形式〕

(1) ori imm, reg1, reg2

imm に指定できるものを次に示します。

- 32 ビット幅までの値を持つ絶対値式
- 相対値式
- 上記のものに hi( ) ，lo( ) ，または hi1( ) を適用したもの

〔機能〕

第一オペランドに指定した絶対値式，相対値式，あるいは，hi( ) ，lo( ) ，または hi1( ) を適用した値と，第二オペランドに指定したレジスタ値の論理和をとり，結果を第三オペランドに指定したレジスタに格納します。

〔説明〕

- imm に次のものを指定した場合，as850 では，機械語命令の ori 命令[注]が 1 つ生成されます。

(a) 0 ～ 65535 の範囲の絶対値式

`ori imm16, reg1, reg2`	`ori imm16, reg1, reg2`

(b) !label，または %label を持つ相対値式

`ori !label, reg1, reg2`	`ori !label, reg1, reg2`

`ori %label, reg1, reg2`	`ori %label, reg1, reg2`

(c) hi( ) ，lo( ) ，または hi1( ) を適用したもの

`ori imm16, reg1, reg2`	`ori imm16, reg1, reg2`

注 機械語命令の ori 命令は，第一オペランドに 0 ～ 65535（0 ～ 0xffff）のイミーディエトをとります。

- imm に次のものを指定した場合，as850 では，命令展開が行われ，1 つ，または複数個の機械語命令が生成されます。

(a) -16 ～ -1 の範囲の絶対値式

`ori imm5, reg1, reg2`	`mov imm5, reg2` `or reg1, reg2`

(b) -32768 ～ -17 の範囲の絶対値式

reg2 が r0 の場合

`ori imm16, reg1, r0`	`movea imm16, r0, r1` `or reg1, r1`

上記以外の場合

`ori imm16, reg1, reg2`	`movea imm16, r0, reg2` `or reg1, reg2`

(c) 上記の範囲を越える絶対値式

imm の値の下位 16 ビットがすべて 0 の場合

`ori imm, reg1, reg2`	`movhi hi(imm), r0, reg2` `or reg1, reg2`

imm の値の下位 16 ビットがすべて 0，ただし reg2 が r0 の場合

`ori imm, reg1, r0`	`movhi hi(imm), r0, r1` `or reg1, r1`

上記以外の場合

`ori imm, reg1, reg2`	`movhi hi1(imm), r0, r1`
	`movea lo(imm), r1, reg2`
	`or reg1, reg2`

上記以外，ただし reg2 が r0 の場合

`ori imm, reg1, r0`	`movhi hi1(imm), r0, r1`
	`movea lo(imm), r1, r1`
	`or reg1, r1`

(d) 上記の範囲を越える絶対値式【V850E】

imm の値の下位 16 ビットがすべて 0 の場合

`ori imm, reg1, reg2`	`movhi hi(imm), r0, reg2` `or reg1, reg2`

imm の値の下位 16 ビットがすべて 0，ただし reg2 が r0 の場合

`ori imm, reg1, r0`	`movhi hi(imm), r0, r1` `or reg1, r1`

上記以外の場合

`ori imm, reg1, reg2`	`mov imm, reg2` `or reg1, reg2`

上記以外，ただし reg2 **が** r0 **の場合**

`ori imm, reg1, r0`	`mov imm, r1` `or reg1, r1`

(e) sdata ／ sbss **属性セクションに定義を持つラベルの** $label **を持つ相対値式**

reg2 **が** r0 **の場合**

`ori $label, reg1, r0`	`movea $label, r0, r1` `or reg1, r1`

上記以外の場合

`ori $label, reg1, reg2`	`movea $label, r0, reg2` `or reg1, reg2`

(f) #label**，または** label **を持つ相対値式，および** sdata ／ sbss **属性セクションに定義を持たないラベルの** $label **を持つ相対値式**

reg2 **が** r0 **の場合**

`ori #label, reg1, r0`	`movhi hi1(#label), r0, r1`
	`movea lo(#label), r1, r1`
	`or reg1, r1`

`ori label, reg1, r0`	`movhi hi1(label), r0, r1`
	`movea lo(label), r1, r1`
	`or reg1, r1`

`ori $label, reg1, r0`	`movhi hi1($label), r0, r1`
	`movea lo($label), r1, r1`
	`or reg1, r1`

上記以外の場合

`ori #label, reg1, reg2`	`movhi hi1(#label), r0, r1`
	`movea lo(#label), r1, reg2`
	`or reg1, reg2`

`ori label, reg1, reg2`	`movhi hi1(label), r0, r1`
	`movea lo(label), r1, reg2`
	`or reg1, reg2`

`ori $label, reg1, reg2`	`movhi hi1($label), r0, r1`
	`movea lo($label), r1, reg2`
	`or reg1, reg2`

(g) #label，または label を持つ相対値式，および sdata ／ sbss 属性セクションに定義を持たないラベルの $label を持つ相対値式【V850E】

reg2 が r0 の場合

`ori #label, reg1, r0`	`mov #label, r1` `or reg1, r1`
`ori label, reg1, r0`	`mov label, r1` `or reg1, r1`
`ori $label, reg1, r0`	`mov $label, r1` `or reg1, r1`

上記以外の場合

`ori #label, reg1, reg2`	`mov #label, reg2` `or reg1, reg2`
`ori label, reg1, reg2`	`mov label, reg2` `or reg1, reg2`
`ori $label, reg1, reg2`	`mov $label, reg2` `or reg1, reg2`

〔フラグ〕

CY	—
OV	0
S	演算結果のワード・データの MSB が 1 の場合 1，そうでない場合 0
Z	結果が 0 になった場合 1，そうでない場合 0
SAT	—

sar

〔概要〕

算術右シフト（Shift Arithmetic Right）

〔形式〕

(1) sar reg1, reg2

(2) sar imm5, reg2

(3) sar reg1, reg2, reg3【V850E2】

imm5 に指定できるものを次に示します。

- 5 ビット幅までの値を持つ絶対値式

〔機能〕

- (1) の形式

  第一オペランドに指定したレジスタの値の下位 5 ビットで示されるビット数分，第二オペランドに指定したレジスタ値を右に算術シフトし，結果を第二オペランドに指定したレジスタに格納します。

- (2) の形式

  第一オペランドに指定した絶対値式の値で示されるビット数分，第二オペランドに指定したレジスタ値を右に算術シフトし，結果を第二オペランドに指定したレジスタに格納します。

- (3) の形式

  第一オペランドに指定したレジスタ値の下位 5 ビットで示されるビット数分，第二オペランドに指定したレジスタ値を右に算術シフトし，結果を第三オペランドに指定したレジスタに格納します。

〔説明〕

- as850 では，機械語命令の sar 命令が 1 つ生成されます。

〔フラグ〕

CY	最後にシフト・アウトしたビットの値が 1 の場合 1，そうでない場合 0 （指定したビット数が 0 の場合 0）
OV	0
S	結果が負になった場合 1，そうでない場合 0
Z	結果が 0 になった場合 1，そうでない場合 0
SAT	—

〔注意事項〕

- (2) の形式で imm5 に 0 ～ 31 の範囲を越える絶対値式を指定した場合，次のメッセージが出力され，指定した値の下位 5 ビット[注]が用いられてアセンブルが続行されます。

```
W3011: illegal operand (range error in immediate).
```

注　機械語命令の sar 命令は，第一オペランドに 0 ～ 31（0x0 ～ 0x1f）のイミーディエトをとります。

shl

〔概要〕

論理左シフト（Shift Logical Left）

〔形式〕

(1) shl reg1, reg2

(2) shl imm5, reg2

(3) shl reg1, reg2, reg3【V850E2】

imm5 に指定できるものを次に示します。

- 5 ビット幅までの値を持つ絶対値式

〔機能〕

- (1) の形式

  第一オペランドに指定したレジスタ値の下位 5 ビットで示されるビット数分，第二オペランドに指定したレジスタ値を左に論理シフトし，結果を第二オペランドに指定したレジスタに格納します。

- (2) の形式

  第一オペランドに指定した絶対値式の値で示されるビット数分，第二オペランドに指定したレジスタ値を左に論理シフトし，結果を第二オペランドに指定したレジスタに格納します。

- (3) の形式

  第一オペランドに指定したレジスタ値の下位 5 ビットで示されるビット数分，第二オペランドに指定したレジスタ値を左に論理シフトし，結果を第三オペランドに指定したレジスタに格納します。

〔説明〕

- as850 では，機械語命令の shl 命令が 1 つ生成されます。

〔フラグ〕

CY	最後にシフト・アウトしたビットの値が 1 の場合 1，そうでない場合 0 （指定したビット数が 0 の場合 0）
OV	0
S	結果が負になった場合 1，そうでない場合 0
Z	結果が 0 になった場合 1，そうでない場合 0
SAT	—

〔注意事項〕

- (2) の形式で imm5 に 0 ～ 31 の範囲を越える絶対値式を指定した場合，次のメッセージが出力され，指定した値の下位 5 ビット[注]が用いられてアセンブルが続行されます。

```
W3011: illegal operand (range error in immediate).
```

注　機械語命令の shl 命令は，第一オペランドに 0 ～ 31（0x0 ～ 0x1f）のイミーディエトをとります。

shr

〔概要〕

論理右シフト（Shift Logical Right）

〔形式〕

(1) shr reg1, reg2

(2) shr imm5, reg2

(3) shr reg1, reg2, reg3【V850E2】

imm5 に指定できるものを次に示します。

- 5 ビット幅までの値を持つ絶対値式

〔機能〕

- (1) の形式

  第一オペランドに指定したレジスタ値の下位 5 ビットで示されるビット数分，第二オペランドに指定したレジスタ値を右に論理シフトし，結果を第二オペランドに指定したレジスタに格納します。

- (2) の形式

  第一オペランドに指定した絶対値式の値で示されるビット数分，第二オペランドに指定したレジスタ値を右に論理シフトし，結果を第二オペランドに指定したレジスタに格納します。

- (3) の形式

  第一オペランドに指定したレジスタ値の下位 5 ビットで示されるビット数分，第二オペランドに指定したレジスタ値を右に論理シフトし，結果を第三オペランドに指定したレジスタに格納します。

〔説明〕

- as850 では，機械語命令の shr 命令が 1 つ生成されます。

〔フラグ〕

CY	最後にシフト・アウトしたビットの値が 1 の場合 1，そうでない場合 0 （指定したビット数が 0 の場合 0）
OV	0
S	結果が負になった場合 1，そうでない場合 0
Z	結果が 0 になった場合 1，そうでない場合 0
SAT	—

〔注意事項〕

- (2) の形式で imm5 に 0 ～ 31 の範囲を越える絶対値式を指定した場合，次のメッセージが出力され，指定した値の下位 5 ビット[注]が用いられてアセンブルが続行されます。

```
W3011: illegal operand (range error in immediate).
```

注　機械語命令の shr 命令は，第一オペランドに 0 ～ 31（0x0 ～ 0x1f）のイミーディエトをとります。

sxb

【V850E】

〔概要〕

バイト・データの符号拡張（Sign Extend Byte）

〔形式〕

(1) sxb reg

〔機能〕

第一オペランドに指定したレジスタの下位 1 バイトのデータを，ワード長に符号拡張します。

〔説明〕

- as850 では，機械語命令の sxb 命令が 1 つ生成されます。

〔フラグ〕

CY	—
OV	—
S	—
Z	—
SAT	—

sxh

【V850E】

〔概要〕

ハーフワード・データの符号拡張（Sign Extend Half-word）

〔形式〕

(1) sxh reg

〔機能〕

第一オペランドに指定したレジスタの下位 2 バイトのデータを，ワード長に符号拡張します。

〔説明〕

- as850 では，機械語命令の sxh 命令が 1 つ生成されます。

〔フラグ〕

CY	—
OV	—
S	—
Z	—
SAT	—

tst

〔概要〕

テスト（Test）

〔形式〕

(1) tst reg1, reg2

(2) tst imm, reg2

imm に指定できるものを次に示します。

- 32 ビット幅までの値を持つ絶対値式
- 相対値式

〔機能〕

- (1) の形式

  第二オペランドに指定したレジスタ値と，第一オペランドに指定したレジスタ値の論理積をとり，結果は格納せず，フラグのみを設定します。

- (2) の形式

  第二オペランドに指定したレジスタ値と，第一オペランドに指定した絶対値式，または相対値式の値の論理積をとり，結果は格納せず，フラグのみを設定します。

〔説明〕

- (1) の形式の命令に対し，as850 では，機械語命令の tst 命令が 1 つ生成されます。
- (2) の形式で imm に次のものを指定した場合，as850 では，命令展開が行われ，1 つ，または複数個の機械語命令が生成されます。

(a) 0

`tst 0, reg`	`tst r0, reg`

(b) - 16 ～ +15 の範囲の絶対値式（0 以外 ）

`tst imm5, reg`	`mov imm5, r1` `tst r1, reg`

(c) - 16 ～ +15 の範囲の絶対値式（0 以外 ）【V850E】

`tst imm5, reg`	`mov imm5, r1` `tst r1, reg`

(d) - 16 ～ +15 の範囲を越え，-32768 ～ +32767 の範囲の絶対値式

`tst imm16, reg`	`movea imm16, r0, r1` `tst r1, reg`

(e) imm に -32768 ～ +32767 の範囲を越える絶対値式

imm の値の下位 16 ビットがすべて 0 の場合

`tst imm, reg`	`movhi hi(imm), r0, r1` `tst r1, reg`

上記以外の場合

`tst imm, reg`	`movhi hi1(imm), r0, r1`
	`movea lo(imm), r1, r1`
	`tst r1, reg`

(f) imm に -32768 ～ +32767 の範囲を越える絶対値式【V850E】

imm の値の下位 16 ビットがすべて 0 の場合

`tst imm, reg`	`movhi hi(imm), r0, r1` `tst r1, reg`

上記以外の場合

`tst imm, reg`	`mov imm, r1` `tst r1, reg`

(g) sdata ／ sbss 属性セクションに定義を持つラベルの $label を持つ相対値式

`tst $label, reg`	`movea $label, r0, r1` `tst r1, reg`

(h) #label，または label を持つ相対値式，および sdata ／ sbss 属性セクションに定義を持たないラベルの $label を持つ相対値式

`tst #label, reg`	`movhi hi1(#label), r0, r1`
	`movea lo(#label), r1, r1`
	`tst r1, reg`

`tst label, reg`	`movhi hi1(label), r0, r1`
	`movea lo(label), r1, r1`
	`tst r1, reg`

`tst $label, reg`	`movhi hi1($label), r0, r1`
	`movea lo($label), r1, r1`
	`tst r1, reg`

(i) #label，または label を持つ相対値式，および sdata ／ sbss 属性セクションに定義を持たないラベルの $label を持つ相対値式【V850E】

`tst  #label, reg`	`mov  #label, r1` `tst  r1, reg`

`tst  label, reg`	`mov  label, r1` `tst  r1, reg`

`tst  $label, reg`	`mov  $label, r1` `tst  r1, reg`

〔フラグ〕

CY	—
OV	0
S	演算結果のワード・データの MSB が 1 の場合 1，そうでない場合 0
Z	結果が 0 になった場合 1，そうでない場合 0
SAT	—

xor

〔概要〕

排他的論理和（Exclusive Or）

〔形式〕

(1) xor reg1, reg2

(2) xor imm, reg2

imm に指定できるものを次に示します。

- 32 ビット幅までの値を持つ絶対値式
- 相対値式

〔機能〕

- (1) の形式

  第一オペランドに指定したレジスタ値と，第二オペランドに指定したレジスタ値の排他的論理和をとり，結果を第二オペランドに指定したレジスタに格納します。

- (2) の形式

  第一オペランドに指定した絶対値式，または相対値式の値と，第二オペランドに指定したレジスタ値の排他的論理和をとり，結果を第二オペランドに指定したレジスタに格納します。

〔説明〕

- (1) の形式の命令に対し，as850 では，機械語命令の xor 命令が 1 つ生成されます。
- (2) の形式で imm に次のものを指定した場合，as850 では，命令展開が行われ，1 つ，または複数個の機械語命令が生成されます[注]。

(a) 0

`xor 0, reg`	`xor r0, reg`

(b) 1 ～ 65535 の範囲の絶対値式

`xor imm16, reg`	`xori imm16, reg, reg`

(c) -16 ～ -1 の範囲の絶対値式

`xor imm5, reg`	`mov imm5, r1` `xor r1, reg`

(d) -32768 ～ -17 の範囲の絶対値式

xor imm16, reg	movea imm16, r0, r1 xor r1, reg

(e) 上記の範囲を越える絶対値式

imm の値の下位 16 ビットがすべて 0 の場合

xor imm, reg	movhi hi(imm), r0, r1 xor r1, reg

上記以外の場合

xor imm, reg	movhi hi1(imm), r0, r1
	movea lo(imm), r1, r1
	xor r1, reg

(f) 上記の範囲を越える絶対値式【V850E】

imm の値の下位 16 ビットがすべて 0 の場合

xor imm, reg	movhi hi(imm), r0, r1 xor r1, reg

上記以外の場合

xor imm, reg	mov imm, r0, r1 xor r1, reg

(g) sdata ／ sbss 属性セクションに定義を持つラベルの $label を持つ相対値式

xor $label, reg	movea $label, r0, r1 xor r1, reg

(h) #label，または label を持つ相対値式，および sdata ／ sbss 属性セクションに定義を持たないラベルの $label を持つ相対値式

xor #label, reg	movhi hi1(#label), r0, r1
	movea lo(#label), r1, r1
	xor r1, reg

xor label, reg	movhi hi1(label), r0, r1
	movea lo(label), r1, r1
	xor r1, reg

xor $label, reg	movhi hi1($label), r0, r1
	movea lo($label), r1, r1
	xor r1, reg

(i) #label，または label を持つ相対値式，および sdata ／ sbss 属性セクションに定義を持たないラベルの $label を持つ相対値式【V850E】

`xor #label, reg`	`mov #label, r1` `xor r1, reg`

`xor label, reg`	`mov label, r1` `xor r1, reg`

`xor $label, reg`	`mov $label, r1` `xor r1, reg`

注 機械語命令の xor 命令は，オペランドにイミーディエトをとりません。

〔フラグ〕

CY	—
OV	0
S	演算結果のワード・データの MSB が 1 の場合 1，そうでない場合 0
Z	結果が 0 になった場合 1，そうでない場合 0
SAT	—

xori

〔概要〕

排他的論理和（Exclusive Or Immediate）

〔形式〕

(1) xori imm, reg1, reg2

imm に指定できるものを次に示します。

- 32 ビット幅までの値を持つ絶対値式
- 相対値式
- 上記のものに hi( ) ，lo( ) ，または hi1( ) を適用したもの

〔機能〕

第一オペランドに指定した絶対値式，相対値式，あるいは，hi( ) ，lo( ) ，または hi1( ) を適用した値と，第二オペランドに指定したレジスタ値の排他的論理和をとり，結果を第三オペランドに指定したレジスタに格納します。

〔説明〕

- imm に次のものを指定した場合，as850 では，機械語命令の xori 命令[注]が 1 つ生成されます。

(a) 0 ～ 65535 の範囲の絶対値式

`xori imm16, reg1, reg2`	`xori imm16, reg1, reg2`

(b) !label，または %label を持つ相対値式

`xori !label, reg1, reg2`	`xori !label, reg1, reg2`

`xori %label, reg1, reg2`	`xori %label, reg1, reg2`

(c) hi( ) ，lo( ) ，または hi1( ) を適用したもの

`xori imm16, reg1, reg2`	`xori imm16, reg1, reg2`

注 機械語命令の xori 命令は，第一オペランドに 0 ～ 65535（0 ～ 0xffff）のイミーディエトをとります。

- imm に次のものを指定した場合，as850 では，命令展開が行われ，1 つ，または複数個の機械語命令が生成されます。

(a) -16 ～ -1 の範囲の絶対値式

`xori imm5, reg1, reg2`	`mov imm5, reg2` `xor reg1, reg2`

(b) -32768 ～ -17 の範囲の絶対値式

reg2 が r0 の場合

`xori imm16, reg1, r0`	`movea imm16, r0, r1` `xor reg1, r1`

上記以外の場合

`xori imm16, reg1, reg2`	`movea imm16, r0, reg2` `xor reg1, reg2`

(c) 上記の範囲を越える絶対値式

imm の値の下位 16 ビットがすべて 0 の場合

`xori imm, reg1, reg2`	`movhi hi(imm), r0, reg2` `xor reg1, reg2`

imm の値の下位 16 ビットがすべて 0，ただし reg2 が r0 の場合

`xori imm, reg1, r0`	`movhi hi(imm), r0, r1` `xor reg1, r1`

上記以外の場合

`xori imm, reg1, reg2`	`movhi hi1(imm), r0, r1`
	`movea lo(imm), r1, reg2`
	`xor reg1, reg2`

上記以外，ただし reg2 が r0 の場合

`xori imm, reg1, r0`	`movhi hi1(imm), r0, r1`
	`movea lo(imm), r1, r1`
	`xor reg1, r1`

(d) 上記の範囲を越える絶対値式【V850E】

imm の値の下位 16 ビットがすべて 0 の場合

`xori imm, reg1, reg2`	`movhi hi(imm), r0, reg2` `xor reg1, reg2`

imm の値の下位 16 ビットがすべて 0，ただし reg2 が r0 の場合

`xori imm, reg1, r0`	`movhi hi(imm), r0, r1` `xor reg1, r1`

上記以外の場合

`xori imm, reg1, reg2`	`mov imm, reg2` `xor reg1, reg2`

上記以外，ただし reg2 **が** r0 **の場合**

`xori imm, reg1, r0`	`mov imm, r1` `xor reg1, r1`

(e) sdata ／ sbss **属性セクションに定義を持つラベルの** $label **を持つ相対値式**

reg2 **が** r0 **の場合**

`xori $label, reg1, r0`	`movea $label, r0, r1` `xor reg1, r1`

上記以外の場合

`xori $label, reg1, reg2`	`movea $label, r0, reg2` `xor reg1, reg2`

(f) #label**，または** label **を持つ相対値式，および** sdata ／ sbss **属性セクションに定義を持たないラベルの** $label **を持つ相対値式**

reg2 **が** r0 **の場合**

`xori #label, reg1, r0`	`movhi hi1(#label), r0, r1`
	`movea lo(#label), r1, r1`
	`xor reg1, r1`

`xori label, reg1, r0`	`movhi hi1(label), r0, r1`
	`movea lo(label), r1, r1`
	`xor reg1, r1`

`xori $label, reg1, r0`	`movhi hi1($label), r0, r1`
	`movea lo($label), r1, r1`
	`xor reg1, r1`

上記以外の場合

`xori #label, reg1, reg2`	`movhi hi1(#label), r0, r1`
	`movea lo(#label), r1, reg2`
	`xor reg1, reg2`

`xori label, reg1, reg2`	`movhi hi1(label), r0, r1`
	`movea lo(label), r1, reg2`
	`xor reg1, reg2`

`xori $label, reg1, reg2`	`movhi hi1($label), r0, r1`
	`movea lo($label), r1, reg2`
	`xor reg1, reg2`

(g) #label，または label を持つ相対値式，および sdata ／ sbss 属性セクションに定義を持たないラベルの $label を持つ相対値式【V850E】

reg2 が r0 の場合

`xori  #label, reg1, r0`	`mov  #label, r1` `xor  reg1, r1`

`xori  label, reg1, r0`	`mov  label, r1` `xor  reg1, r1`

`xori  $label, reg1, r0`	`mov  $label, r1` `xor  reg1, r1`

上記以外の場合

`xori  #label, reg1, reg2`	`mov  #label, reg2` `xor  reg1, reg2`

`xori  label, reg1, reg2`	`mov  label, reg2` `xor  reg1, reg2`

`xori  $label, reg1, reg2`	`mov  $label, reg2` `xor  reg1, reg2`

〔フラグ〕

CY	—
OV	0
S	演算結果のワード・データの MSB が 1 の場合 1，そうでない場合 0
Z	結果が 0 になった場合 1，そうでない場合 0
SAT	—

zxb

【V850E】

〔概要〕

バイト・データのゼロ拡張（Zero Extend Byte）

〔形式〕

(1) zxb reg

〔機能〕

第一オペランドに指定したレジスタの下位 1 バイトのデータを，ワード長にゼロ拡張します。

〔説明〕

- as850 では，機械語命令の zxb 命令が 1 つ生成されます。

〔フラグ〕

CY	—
OV	—
S	—
Z	—
SAT	—

zxh

【V850E】

〔概要〕

ハーフワード・データのゼロ拡張（Zero Extend Half-word）

〔形式〕

(1) zxh reg

〔機能〕

第一オペランドに指定したレジスタの下位ハーフワードのデータを，ワード長にゼロ拡張します。

〔説明〕

- as850 では，機械語命令の zxh 命令が 1 つ生成されます。

〔フラグ〕

CY	—
OV	—
S	—
Z	—
SAT	—

sch0l

【V850E2】

〔概要〕

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

〔形式〕

(1) sch0l reg2, reg3

〔機能〕

第一オペランドに指定したレジスタのワード・データを左側（MSB 側）から検索し，最初にビット（0）が見つかった位置を第二オペランドに指定したレジスタに 16 進数で格納します。（たとえば，第一オペランドに指定したレジスタのビット 31 が 0 の場合は，第二オペランドに指定したレジスタに 01H を格納します）。

ビット（0）が見つからなかった場合は，第二オペランドに指定したレジスタに 0 を書き込み，同時に Z フラグをセット（1）します。最後にビット（0）が見つかった場合は，CY フラグをセット（1）します。

〔説明〕

- as850 では，機械語命令の sch0l 命令が 1 つ生成されます。

〔フラグ〕

CY	最後にビット（0）が見つかった場合 1，そうでない場合 0
OV	0
S	0
Z	ビット（0）が見つからなかった場合 1，そうでない場合 0
SAT	—

sch0r

【V850E2】

〔概要〕

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

〔形式〕

(1) sch0r reg2, reg3

〔機能〕

第一オペランドに指定したレジスタのワード・データを右側（LSB 側）から検索し，最初にビット（0）が見つかった位置を第二オペランドに指定したレジスタに 16 進数で格納します（たとえば，第一オペランドに指定したレジスタのビット 0 が 0 の場合は，第二オペランドに指定したレジスタ に 01H を格納します）。

ビット（0）が見つからなかった場合は，第二オペランドに指定したレジスタに 0 を書き込み，同時に Z フラグをセット（1）します。最後にビット（0）が見つかった場合は，CY フラグをセット（1）します

〔説明〕

- as850 では，機械語命令の sch0r 命令が 1 つ生成されます。

〔フラグ〕

CY	最後にビット（0）が見つかった場合 1，そうでない場合 0
OV	0
S	0
Z	ビット（0）が見つからなかった場合 1，そうでない場合 0
SAT	—

sch1l

【V850E2】

〔概要〕

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

〔形式〕

(1) sch1l reg2, reg3

〔機能〕

第一オペランドに指定したレジスタのワード・データを左側（MSB 側）から検索し，最初にビット（1）が見つかった位置を第二オペランドに指定したレジスタに 16 進数で書き込みます（たとえば，第一オペランドに指定したレジスタのビット 31 が 1 の場合は，第二オペランドに指定したレジスタに 01H を格納します）。

ビット（1）が見つからなかった場合は，第二オペランドに指定したレジスタに 0 を書き込み，同時に Z フラグをセット（1）します。最後にビット（1）が見つかった場合は，CY フラグをセット（1）します。

〔説明〕

- as850 では，機械語命令の sch1l 命令が 1 つ生成されます。

〔フラグ〕

CY	最後にビット（1）が見つかった場合 1，そうでない場合 0
OV	0
S	0
Z	ビット（1）が見つからなかった場合 1，そうでない場合 0
SAT	—

sch1r

【V850E2】

〔概要〕

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

〔形式〕

(1) sch1r reg2, reg3

〔機能〕

第一オペランドに指定したレジスタのワード・データを右側（LSB 側）から検索し，最初にビット（1）が見つかった位置を第二オペランドに指定したレジスタに 16 進数で格納します（たとえば，第一オペランドに指定したレジスタのビット 0 が 1 の場合は，第二オペランドに指定したレジスタに 01H を格納します）。

ビット（1）が見つからなかった場合は，第二オペランドに指定したレジスタに 0 を書き込み，同時に Z フラグをセット（1）します。最後にビット（1）が見つかった場合は，CY フラグをセット（1）します。

〔説明〕

- as850 では，機械語命令の sch1r 命令が 1 つ生成されます。

〔フラグ〕

CY	最後にビット（1）が見つかった場合 1，そうでない場合 0
OV	0
S	0
Z	ビット（1）が見つからなかった場合 1，そうでない場合 0
SAT	—

3.6 分岐命令

この節では，分岐命令について説明します。
次に，この節において説明する命令を示します。

表 3 － 10 分岐命令

命令	意味
jarl	ジャンプ・アンド・レジスタ・リンク
jarl22	ジャンプ・アンド・レジスタ・リンク【V850E2】
jarl32	ジャンプ・アンド・レジスタ・リンク【V850E2】
jcond	条件分岐
jmp	無条件分岐
jmp32	無条件分岐【V850E2】
jr	無条件分岐（PC 相対）
jr22	無条件分岐（PC 相対）【V850E2】
jr32	無条件分岐（PC 相対）【V850E2】

jarl

〔概要〕

ジャンプ・アンド・レジスタ・リンク（Jump and Register Link）

〔形式〕

(1) jarl disp22, reg2

(2) jarl disp32, reg1【V850E2】

ディスプレースメント（disp22）に指定できるものを次に示します。

- 22 ビット幅までの値を持つ絶対値式
- ラベルの PC オフセット参照を持つ相対値式

〔機能〕

- (1) の形式

  第一オペランドに指定した絶対値式，または相対値式の値と，現在のプログラム・カウンタ（PC）値を加算したアドレスに制御を移します。なお，戻りアドレスは，第二オペランドに指定したレジスタに格納されます。

- (2) の形式

  第一オペランドに指定した絶対値式，または相対値式の値と，現在のプログラム・カウンタ（PC）値を加算したアドレスに制御を移します。なお，戻りアドレスは，第二オペランドに指定したレジスタに格納されます。

〔説明〕

- (1) の形式の命令に対し，disp22 に次のものを指定した場合，as850 では，機械語命令の jarl 命令[注]が 1 つ生成されます。

(a) -2097152 ～ +2097151 の範囲の絶対値

(b) 本命令と同じファイル内の同じセクションに定義を持つラベルの PC オフセット参照を持ち，-2097152 ～ +2097151 の範囲の相対値式

(c) 本命令と同じファイル内に定義を持たないか，同じセクションに定義を持たないラベルの PC オフセット参照を持つ相対値式

注 機械語命令の jarl は，ディスプレースメントに -2097152 ～ +2097151（0xfe00000 ～ 0x1fffff）の範囲のイミーディエトをとります。

- (2) の形式の命令に対し，as850 では，機械語命令の jarl 命令（6 バイト長命令）が 1 つ生成されます。

〔フラグ〕

CY	—
OV	—
S	—
Z	—
SAT	—

〔注意事項〕

- disp22 に，-2097152 ～ +2097151 の範囲を越える絶対値式，または本命令と同じファイル内の同じセクションに定義を持つラベルの PC オフセット参照を持ち -2097152 ～ +2097151 の範囲を越える相対値式を指定した場合，次のメッセージが出力され，アセンブルが中止されます。

```
E3230: illegal operand (range error in displacement)
```

- disp22 に，奇数値を持つ絶対値式，または本命令と同じファイル内の同じセクションに定義を持つラベルの PC オフセット参照を持ち奇数値を持つ相対値式を指定した場合，次のメッセージが出力され，アセンブルが中止されます。

```
E3226: illegal operand (must be even displacement)
```

- アセンブラオプション -Xfar_jump を指定しない場合に，disp32 に，-2097152 ～ +2097151 の範囲を越える絶対値式，または本命令と同じファイル内の同じセクションに定義を持つラベルの PC オフセット参照を持ち -2097152 ～ +2097151 の範囲を越える相対値式を指定した場合，次のメッセージが出力され，アセンブルが中止されます。

```
E3230: illegal operand (range error in displacement)
```

jarl22

【V850E2】

〔概要〕

ジャンプ・アンド・レジスタ・リンク（Jump and Register Link）

〔形式〕

(1) jarl22　　disp22, reg1

ディスプレースメント（disp22）に指定できるものを次に示します。

- 22 ビット幅までの値を持つ絶対値式
- ラベルの PC オフセット参照を持つ相対値式

〔機能〕

第一オペランドに指定した絶対値式，または相対値式の値と，現在のプログラム・カウンタ（PC）値を加算したアドレスに制御を移します。なお，戻りアドレスは，第二オペランドに指定したレジスタに格納されます。

〔説明〕

- disp22 に次のものを指定した場合，as850 では，機械語命令の jarl 命令[注]が 1 つ生成されます。

(a) -2097152 ～ +2097151 の範囲の絶対値

(b) 本命令と同じファイル内の同じセクションに定義を持つラベルの PC オフセット参照を持ち，-2097152 ～ +2097151 の範囲の相対値式

(c) 本命令と同じファイル内に定義を持たないか，同じセクションに定義を持たないラベルの PC オフセット参照を持つ相対値式

注　機械語命令の jarl は，ディスプレースメントに -2097152 ～ +2097151（0xfe00000 ～ 0x1fffff）の範囲のイミーディエトをとります。

〔フラグ〕

CY	—
OV	—
S	—
Z	—
SAT	—

jarl32

【V850E2】

〔概要〕

ジャンプ・アンド・レジスタ・リンク（Jump and Register Link）

〔形式〕

(1) jarl32 disp32, reg1

〔機能〕

第一オペランドに指定した絶対値式，または相対値式の値と，現在のプログラム・カウンタ（PC）値を加算したアドレスに制御を移します。なお，戻りアドレスは，第二オペランドに指定したレジスタに格納されます。

〔説明〕

- as850 では，機械語命令の jarl 命令（6 バイト長命令）が 1 つ生成されます。

〔フラグ〕

CY	—
OV	—
S	—
Z	—
SAT	—

jcond

〔概要〕

条件分岐（Jump on Condition）

〔形式〕

(1) j*cond* disp22

ディスプレースメント（disp22）**に指定できるものを次に示します。**

- **22 ビット幅までの値を持つ絶対値式**
- **ラベルの** PC **オフセット参照を持つ相対値式**

〔機能〕

cond **部分の文字列で示されるフラグ状態（「表 3 － 11 jcond 命令一覧」参照）と，現在のフラグ状態を比較し，一致した場合は，オペランドに指定した絶対値式，または相対値式の値と現在のプログラム・カウンタ**（PC）**の値を加算したアドレスに制御を移します注。**

注 jbr **以外の** j*cond* **命令に対しては，**b*cond* **というニモニックを，**jbr **命令に対しては** br **というニモニックを用いることもできます（機能に違いはありません）。**

表 3 － 11 j*cond* 命令一覧

命令	フラグ状態	フラグ状態の意味
jgt	((S xor OV) or Z) = 0	Greater than（signed）
jge	(S xor OV) = 0	Greater than or equal（signed）
jlt	(S xor OV) = 1	Less than（signed）
jle	((S xor OV) or Z) = 1	Less than or equal（signed）
jh	(CY or Z) = 0	Higher（Greater than）
jnl	CY = 0	Not lower（Greater than or equal）
jl	CY = 1	Lower（Less than）
jnh	(CY or Z) = 1	Not higher（Less than or equal）
je	Z = 1	Equal
jne	Z = 0	Not equal
jv	OV = 1	Overflow
jnv	OV = 0	No overflow
jn	S = 1	Negative
jp	S = 0	Positive
jc	CY = 1	Carry
jnc	CY = 0	No carry
jz	Z = 1	Zero
jnz	Z = 0	Not zero
jbr	－	Always （無条件）
jsa	SAT = 1	Saturated

〔説明〕

- disp22 に次のものを指定した場合，as850 では，機械語命令の b*cond* 命令[注]が 1 つ生成されます。

(a) -256 ～ +255 の範囲の絶対値式

(b) 本命令と同じファイル内の同じセクションに定義を持つラベルの PC オフセット参照を持ち，-256 ～ +255 の範囲の相対値式

j*cond* disp9	b*cond* disp9

注 機械語命令の b*cond* は，ディスプレースメントに -256 ～ +255（0xffffff00 ～ 0xff）の範囲のイミーディエトをとります。

- disp22 に次のものを指定した場合，as850 では，命令展開が行われ，複数の機械語命令が生成されます。

(a) -256 ～ +255 の範囲を越え -2097150 ～ +2097153 の範囲[注]の絶対値式

(b) 本命令と同じファイル内の同じセクションに定義を持つラベルの PC オフセット参照を持ち，-256 ～ +255 の範囲を越え -2097150 ～ +2097153 の範囲の相対値式

(c) 本命令と同じファイル内に定義を持たないか同じセクションに定義を持たないラベルの PC オフセット参照を持つ相対値式

注 ただし，-2097150 ～ +2097153 の範囲は，jbr，および jsa 命令以外の場合の範囲で，jbr 命令の場合は -2097152 ～ +2097151，jsa 命令の場合は -2097148 ～ +2097155 となります。

jbr disp22	jr disp22

jsa disp22	bsa Label1 br Label2 Label1： jr disp22 - 4 Label2：

j*cond* disp22	bn*cond* Label[注] jr disp22 - 2 Label：

注 bn*cond* は，たとえば，bz に対する bnz. bgt に対する ble というように，逆の条件で分岐を行う命令を示しています。

〔フラグ〕

CY	—
OV	—
S	—
Z	—
SAT	—

〔注意事項〕

- disp22 に，-2097150 ～ +2097153 の範囲を越える絶対値式，または本命令と同じファイル内の同じセクションに定義を持つラベルの PC オフセット参照を持ち -2097150 ～ +2097153 の範囲を越える相対値式を指定した場合，次のメッセージが出力され，アセンブルが中止されます。

```
E3230: illegal operand (range error in displacement)
```

- disp22 に，奇数値を持つ絶対値式，または本命令と同じファイル内の同じセクションに定義を持つラベルの PC オフセット参照を持ち奇数値を持つ相対値式を指定した場合，次のメッセージが出力され，アセンブルが中止されます。

```
E3226: illegal operand (must be even displacement)
```

jmp

〔概要〕

無条件分岐（Jump）

〔形式〕

(1) jmp [reg]

(2) jmp disp32[reg]【V850E2】

(3) jmp addr

addr に指定できるものを次に示します。

- ラベルの絶対アドレス参照を持つ相対値式

〔機能〕

- (1) の形式

  オペランドに指定したレジスタ値が示すアドレスに制御を移します。

- (2) の形式

  オペランドに指定したディスプレースメントとレジスタの内容を加算して得たアドレスに制御を移します。

- (3) の形式

  オペランドに指定した相対値式の値が示すアドレスに制御を移します。

〔説明〕

- (1) の形式の命令に対し，as850 では，機械語命令の jmp 命令が 1 つ生成されます。
- (2) の形式の命令に対し，as850 では，機械語命令の jmp 命令（6 バイト長命令）が 1 つ生成されます。
- (3) の形式の命令に対し，as850 では，命令展開が行われ，複数個の機械語命令が生成されます。

`jmp #label`	`movhi hi1(#label), r0, r1`
	`movea lo(#label), r1, r1`
	`jmp [r1]`

【V850E】

`jmp #label`	`mov #label, r1` `jmp [r1]`

〔フラグ〕

CY	—
OV	—
S	—
Z	—
SAT	—

〔注意事項〕

- (3) の形式において，addr にラベルの絶対アドレス参照を持つ相対値式以外のものを指定した場合，次のメッセージが出力され，アセンブルが中止されます。

```
E3224: illegal operand (label reference for jmp must be #label)
```

jmp32

【V850E2】

〔概要〕

無条件分岐（Jump）

〔形式〕

(1) jmp32　　disp32[reg]

〔機能〕

オペランドに指定したディスプレースメントとレジスタの内容を加算して得たアドレスに制御を移します。

〔説明〕

- as850 **では，機械語命令の** jmp **命令（**6 **バイト長命令）が** 1 **つ生成されます。**

〔フラグ〕

CY	—
OV	—
S	—
Z	—
SAT	—

jr

〔概要〕

無条件分岐（PC 相対）（Jump Relative）

〔形式〕

(1) jr disp22

(2) jr disp32【V850E2】

ディスプレースメント（disp22）に指定できるものを次に示します。

- 22 ビット幅までの値を持つ絶対値式
- ラベルの PC オフセット参照を持つ相対値式

〔機能〕

- (1) の形式

  オペランドに指定した絶対値式，または相対値式の値と，現在のプログラム・カウンタ（PC）の値を加算したアドレスに制御を移します。
- (2) の形式

  オペランドに指定した絶対値式，または相対値式の値と，現在のプログラム・カウンタ（PC）の値を加算したアドレスに制御を移します。

〔説明〕

- (1) の形式の命令に対し，disp22 に次のものを指定した場合，as850 で機械語命令の jr 命令[注]が 1 つ生成されます。

(a) -2097152 ～ +2097151 の範囲の絶対値式

(b) 本命令と同じファイル内の同じセクションに定義を持つラベルの PC オフセット参照を持ち，-2097152 ～ +2097151 の範囲の相対値式

(c) 本命令と同じファイル内に定義を持たないか同じセクションに定義を持たないラベルの PC オフセット参照を持つ相対値式

注 機械語命令の jr は，ディスプレースメントに -2097152 ～ +2097151（0xfe00000 ～ 0x1fffff）の範囲のイミーディエトをとります。

- (2) の形式の命令に対し，as850 では，機械語命令の jr 命令（6 バイト長命令）が 1 つ生成されます。

〔フラグ〕

CY	—
OV	—
S	—
Z	—
SAT	—

〔注意事項〕

- disp22 に，-2097152 ～ +2097151 の範囲を越える絶対値式，または本命令と同じファイル内の同じセクションに定義を持つラベルの PC オフセット参照を持ち -2097152 ～ +2097151 の範囲を越える相対値式を指定した場合，次のメッセージが出力され，アセンブルが中止されます。

```
E3230: illegal operand (range error in displacement)
```

- disp22 に，奇数値を持つ絶対値式，または本命令と同じファイル内の同じセクションに定義を持つラベルの PC オフセット参照を持ち奇数値を持つ相対値式を指定した場合，次のメッセージが出力され，アセンブルが中止されます。

```
E3226: illegal operand (must be even displacement)
```

- アセンブラオプション -Xfar_jump を指定しない場合に，disp32 に，-2097152 ～ +2097151 の範囲を越える絶対値式，または本命令と同じファイル内の同じセクションに定義を持つラベルの PC オフセット参照を持ち -2097152 ～ +2097151 の範囲を越える相対値式を指定した場合，次のメッセージが出力され，アセンブルが中止されます。

```
E3230: illegal operand (range error in displacement)
```

jr22

【V850E2】

〔概要〕

無条件分岐（PC 相対）（Jump Relative）

〔形式〕

(1) jr22　　disp22

ディスプレースメント（disp22）**に指定できるものを次に示します。**

- 22 **ビット幅までの値を持つ絶対値式**
- **ラベルの** PC **オフセット参照を持つ相対値式**

〔機能〕

オペランドに指定した絶対値式，または相対値式の値と，現在のプログラム・カウンタ（PC）**の値を加算したアドレスに制御を移します。**

〔説明〕

- disp22 **に次のものを指定した場合，**as850 **では，機械語命令の** jr **命令**[注]**が** 1 **つ生成されます。**

(a) -2097152 ～ +2097151 **の範囲の絶対値式**

(b) **本命令と同じファイル内の同じセクションに定義を持つラベルの** PC **オフセット参照を持ち，**-2097152 ～ +2097151 **の範囲の相対値式**

(c) **本命令と同じファイル内に定義を持たないか同じセクションに定義を持たないラベルの** PC **オフセット参照を持つ相対値式**

注　**機械語命令の** jr **は，ディスプレースメントに** -2097152 ～ +2097151（0xfe00000 ～ 0x1fffff）**の範囲のイミーディエトをとります。**

〔フラグ〕

CY	—
OV	—
S	—
Z	—
SAT	—

jr32

【V850E2】

〔概要〕

無条件分岐（PC 相対）（Jump Relative）

〔形式〕

(1) jr　　disp32

〔機能〕

第一オペランドに指定した絶対値式，または相対値式の値と，現在のプログラム・カウンタ（PC）の値を加算したアドレスに制御を移します。

〔説明〕

- as850 では，機械語命令の jr 命令（6 バイト長命令）が 1 つ生成されます。

〔フラグ〕

CY	—
OV	—
S	—
Z	—
SAT	—

3.7 ビット操作命令

この節では，ビット操作命令について説明します。
次に，この節において説明する命令を示します。

表 3 － 12 ビット操作命令

命令	意味
clr1	ビット・クリア
not1	ビット・ノット
set1	ビット・セット
tst1	ビット・テスト

clr1

〔概要〕

ビット・クリア（Clear Bit）

〔形式〕

(1) clr1 bit# 3, disp[reg1]

(2) clr1 reg2, [reg1]【V850E】

(3) clr1 BITIO

ディスプレースメント（disp）に指定できるものを次に示します。

- 32 ビット幅までの値を持つ絶対値式
- 相対値式
- 上記のものに hi( )，lo( )，または hi1( ) を適用したもの

注 (2) の形式では disp は指定できません。

〔機能〕

- (1) の形式

  第二オペランドで指定したアドレスが示すデータの，第一オペランドに指定したビットをクリアします。指定したビット以外は影響を受けません。

- (2) の形式

  第二オペランドのレジスタ値で指定したアドレスが示すデータの，第一オペランドで指定したレジスタ値の下位 3 ビットが示すビットをクリアします。指定したビット以外は影響を受けません。

- (3) の形式

  第一オペランドで指定したアドレスが示すデータにおいて，周辺 I/O レジスタのビット名（デバイス・ファイルで定義されている予約語のみ）で指定したビットをクリアします。

〔説明〕

- disp に次のものを指定した場合，as850 では，機械語命令の clr1 命令[注]が 1 つ生成されます。

(a) -32768 ～ +32767 の範囲の絶対値式

clr1 disp16[reg1], reg2	clr1 disp16[reg1], reg2

(b) sdata ／ sbss 属性セクションに定義を持つラベルの $label を持つ相対値式

clr1 $label[reg1], reg2	clr1 $label[reg1], reg2

(c) !label，または %label を持つ相対値式

clr1 !label[reg1], reg2	clr1 !label[reg1], reg2

clr1 %label[reg1], reg2	clr1 %label[reg1], reg2

(d) hi( )，lo( )，または hi1( ) を適用したもの

clr1 disp16[reg1], reg2	clr1 disp16[reg1], reg2

注 機械語命令の clr1 命令は，ディスプレースメントに -32768 ～ +32767（0xffff8000 ～ 0x7fff）の範囲のイミーディエトをとります。

- disp に次のものを指定した場合，as850 では，命令展開が行われ，複数個の機械語命令が生成されます。

(a) -32768 ～ +32767 の範囲を越える絶対値式

clr1 disp[reg1], reg2	movhi hi1(disp), reg1, r1 clr1 lo(disp)[r1], reg2

(b) #label，または label を持つ相対値式，および sdata ／ sbss 属性セクションに定義を持たないラベルの $label を持つ相対値式

clr1 #label[reg1], reg2	movhi hi1(#label), reg1, r1 clr1 lo(#label)[r1], reg2

clr1 label[reg1], reg2	movhi hi1(label), reg1, r1 clr1 lo(label)[r1], reg2

clr1 $label[reg1], reg2	movhi hi1($label), reg1, r1 clr1 lo($label)[r1], reg2

- disp を省略した場合，as850 では，0 が指定されたものとみなされます。
- disp に #label を持つ相対値式，または #label を持つ相対値式に hi( )，lo( )，または hi1( ) を適用したものを指定した場合，その後ろの [reg1] の部分を省略できます。ただし，省略した場合，as850 では，[r0] が指定されたものとみなされます。
- disp に $label を持つ相対値式，または $label を持つ相対値式に hi( )，lo( )，または hi1( ) を適用したものを指定した場合，その後ろの [reg1] の部分が省略できます。ただし，省略した場合，as850 では，[gp] が指定されたものとみなされます。
- disp にデバイス・ファイルで定義されている周辺 I/O レジスタ名を指定した場合，その後ろの [reg1] の部分が省略できます。ただし，省略した場合，as850 では，[r0] が指定されたものとみなされます。

〔フラグ〕

CY	—
OV	—
S	—
Z	指定したビットが 0 の場合 1，1 の場合 0 注
SAT	—

注　Z フラグの値は，この命令実行前の該当ビットの値を示しています。この命令実行後を示すものではありません。

not1

〔概要〕

ビット・ノット（Not Bit）

〔形式〕

(1) not1 bit# 3, disp[reg1]

(2) not1 reg2, [reg1]【V850E】

(3) not1 BITIO

ディスプレースメント（disp）に指定できるものを次に示します。

- 32 ビット幅までの値を持つ絶対値式
- 相対値式
- 上記のものに hi( )，lo( )，または hi1( ) を適用したもの

注 (2) の形式では disp は指定できません。

〔機能〕

- (1) の形式

  第二オペランドで指定したアドレスが示すデータの，第一オペランドに指定したビットを反転（0 → 1，1 → 0）します。指定したビット以外は影響を受けません。

- (2) の形式

  第二オペランドのレジスタ値で指定したアドレスが示すデータの，第一オペランドで指定したレジスタ値の下位 3 ビットが示すビットを反転（0 → 1，1 → 0）します。指定したビット以外は影響を受けません。

- (3) の形式

  第一オペランドで指定したアドレスが示すデータにおいて，周辺 I/O レジスタのビット名（デバイス・ファイルで定義されている予約語のみ）で指定したビットを反転（0 → 1，1 → 0）します。

〔説明〕

- disp に次のものを指定した場合，as850 では，機械語命令の not1 命令[注]が 1 つ生成されます。

(a) -32768 ～ +32767 の範囲の絶対値式

`not1  disp16[reg1], reg2`	`not1  disp16[reg1], reg2`

(b) sdata ／ sbss 属性セクションに定義を持つラベルの $label を持つ相対値式

`not1  $label[reg1], reg2`	`not1  $label[reg1], reg2`

(c) !label，または %label を持つ相対値式

`not1  !label[reg1], reg2`	`not1  !label[reg1], reg2`

`not1  %label[reg1], reg2`	`not1  %label[reg1], reg2`

(d) hi( )，lo( )，または hi1( ) を適用したもの

`not1  disp16[reg1], reg2`	`not1  disp16[reg1], reg2`

注 機械語命令の not1 命令は，ディスプレースメントに -32768 ～ +32767（0xffff8000 ～ 0x7fff）の範囲のイミーディエトをとります。

- disp に次のものを指定した場合，as850 では，命令展開が行われ，複数個の機械語命令が生成されます。

(a) -32768 ～ +32767 の範囲を越える絶対値式

`not1  disp[reg1], reg2`	`movhi  hi1(disp), reg1, r1` `not1  lo(disp)[r1], reg2`

(b) #label，または label を持つ相対値式，および sdata ／ sbss 属性セクションに定義を持たないラベルの $label を持つ相対値式

`not1  #label[reg1], reg2`	`movhi  hi1(#label), reg1, r1` `not1  lo(#label)[r1], reg2`

`not1  label[reg1], reg2`	`movhi  hi1(label), reg1, r1` `not1  lo(label)[r1], reg2`

`not1  $label[reg1], reg2`	`movhi  hi1($label), reg1, r1` `not1  lo($label)[r1], reg2`

- disp を省略した場合，as850 では，0 が指定されたものとみなされます。
- disp に #label を持つ相対値式，または #label を持つ相対値式に hi( )，lo( )，または hi1( ) を適用したものを指定した場合，その後ろの [reg1] の部分が省略できます。ただし，省略した場合，as850 では，[r0] が指定されたものとみなされます。
- disp に $label を持つ相対値式，または $label を持つ相対値式に hi( )，lo( )，または hi1( ) を適用したものを指定した場合，その後ろの [reg1] の部分が省略できます。ただし，省略した場合，as850 では，[gp] が指定されたものとみなされます。
- disp にデバイス・ファイルで定義されている周辺 I/O レジスタ名を指定した場合，その後ろの [reg1] の部分が省略できます。ただし，省略した場合，as850 では，[r0] が指定されたものとみなされます。

〔フラグ〕

CY	—
OV	—
S	—
Z	指定したビットが 0 の場合 1，1 の場合 0 注
SAT	—

注　Z フラグの値は，この命令実行前の該当ビットの値を示しています。この命令実行後を示すものではありません。

set1

〔概要〕

ビット・セット（Set Bit）

〔形式〕

(1) set1 bit # 3, disp[reg1]

(2) set1 reg2, [reg1]【V850E】

(3) set1 BITIO

ディスプレースメント（disp）に指定できるものを次に示します。

- 32 ビット幅までの値を持つ絶対値式
- 相対値式
- 上記のものに hi( )，lo( )，または hi1( ) を適用したもの

注 (2) の形式では disp は指定できません。

〔機能〕

- (1) の形式

  第二オペランドで指定したアドレスが示すデータの，第一オペランドに指定したビットをセットします。指定したビット以外は影響を受けません。

- (2) の形式

  第二オペランドのレジスタ値で指定したアドレスが示すデータの，第一オペランドで指定したレジスタ値の下位 3 ビットが示すビットをセットします。指定したビット以外は影響を受けません。

- (3) の形式

  第一オペランドで指定したアドレスが示すデータにおいて，周辺 I/O レジスタのビット名（デバイス・ファイルで定義されている予約語のみ）で指定したビットをセットします。

〔説明〕

- disp に次のものを指定した場合，as850 では，機械語命令の set1 命令[注]が 1 つ生成されます。

(a) -32768 ～ +32767 の範囲の絶対値式

`set1 disp16[reg1], reg2`	`set1 disp16[reg1], reg2`

(b) sdata ／ sbss 属性セクションに定義を持つラベルの $label を持つ相対値式

`set1 $label[reg1], reg2`	`set1 $label[reg1], reg2`

(c) !label，または %label を持つ相対値式

`set1 !label[reg1], reg2`	`set1 !label[reg1], reg2`

`set1 %label[reg1], reg2`	`set1 %label[reg1], reg2`

(d) hi( )，lo( )，または hi1( ) を適用したもの

`set1 disp16[reg1], reg2`	`set1 disp16[reg1], reg2`

注 機械語命令の set1 命令は，ディスプレースメントに -32768 ～ +32767（0xffff8000 ～ 0x7fff）の範囲のイミーディエトをとります。

- disp に次のものを指定した場合，as850 では，命令展開が行われ，複数個の機械語命令が生成されます。

(a) -32768 ～ +32767 の範囲を越える絶対値式

`set1 disp[reg1], reg2`	`movhi hi1(disp), reg1, r1` `set1 lo(disp)[r1], reg2`

(b) #label，または label を持つ相対値式，および sdata ／ sbss 属性セクションに定義を持たないラベルの $label を持つ相対値式

`set1 #label[reg1], reg2`	`movhi hi1(#label), reg1, r1` `set1 lo(#label)[r1], reg2`

`set1 label[reg1], reg2`	`movhi hi1(label), reg1, r1` `set1 lo(label)[r1], reg2`

`set1 $label[reg1], reg2`	`movhi hi1($label), reg1, r1` `set1 lo($label)[r1], reg2`

- disp を省略した場合，as850 では，0 が指定されたものとみなされます。
- disp に #label を持つ相対値式，または #label を持つ相対値式に hi( )，lo( )，または hi1( ) を適用したものを指定した場合，その後ろの [reg1] の部分が省略できます。ただし，省略した場合，as850 では，[r0] が指定されたものとみなされます。
- disp に $label を持つ相対値式，または $label を持つ相対値式に hi( )，lo( )，または hi1( ) を適用したものを指定した場合，その後ろの [reg1] の部分が省略できます。ただし，省略した場合，as850 では，[gp] が指定されたものとみなされます。
- disp にデバイス・ファイルで定義されている周辺 I/O レジスタ名を指定した場合，その後ろの [reg1] の部分が省略できます。ただし，省略した場合，as850 では，[r0] が指定されたものとみなされます。

〔フラグ〕

CY	—
OV	—
S	—
Z	指定したビットが 0 の場合 1，1 の場合 0 注
SAT	—

注 Z フラグの値は，この命令実行前の該当ビットの値を示しています。この命令実行後を示すものではありません。

tst1

〔概要〕

ビット・テスト（Test Bit）

〔形式〕

(1) tst1 bit# 3, disp[reg1]

(2) tst1 reg2, [reg1]【V850E】

(3) tst1 BITIO

ディスプレースメント（disp）に指定できるものを次に示します。

- 32 ビット幅までの値を持つ絶対値式
- 相対値式
- 上記のものに hi( ) ，lo( ) ，または hi1( ) を適用したもの

注 (2) の形式では disp は指定できません。

〔機能〕

- (1) の形式

  第二オペランドで指定したアドレスが示すデータの，第一オペランドに指定したビットの値に従い，フラグのみを設定します。第二オペランドの値，および指定したビットは変更されません。

- (2) の形式

  第二オペランドで指定したアドレスが示すデータの,第一オペランドで指定したレジスタ値の下位 3 ビットが示すビットの値に従い，フラグのみを設定します。第二オペランドの値，および指定したビットは変更されません。

- (3) の形式

  第一オペランドで指定したアドレスが示すデータにおいて，周辺 I/O レジスタのビット名（デバイス・ファイルで定義されている予約語のみ）で指定したビットの値に従い，フラグのみを設定します。周辺 I/O レジスタのビットの値は変更されません。

〔説明〕

- disp に次のものを指定した場合，as850 では，機械語命令の tst1 命令[注]が 1 つ生成されます。

(a) -32768 ～ +32767 の範囲の絶対値式

`tst1 disp16[reg1], reg2`	`tst1 disp16[reg1], reg2`

(b) sdata ／ sbss 属性セクションに定義を持つラベルの $label を持つ相対値式

`tst1 $label[reg1], reg2`	`tst1 $label[reg1], reg2`

(c) !label，または %label を持つ相対値式

`tst1 !label[reg1], reg2`	`tst1 !label[reg1], reg2`

`tst1 %label[reg1], reg2`	`tst1 %label[reg1], reg2`

(d) hi( )，lo( )，または hi1( ) を適用したもの

`tst1 disp16[reg1], reg2`	`tst1 disp16[reg1], reg2`

注 機械語命令の tst1 命令は，ディスプレースメントに -32768 ～ +32767（0xffff8000 ～ 0x7fff）の範囲のイミーディエトをとります。

- disp に次のものを指定した場合，as850 では，命令展開が行われ，複数個の機械語命令が生成されます。

(a) -32768 ～ +32767 の範囲を越える絶対値式

`tst1 disp[reg1], reg2`	`movhi hi1(disp), reg1, r1` `tst1 lo(disp)[r1], reg2`

(b) #label，または label を持つ相対値式，および sdata ／ sbss 属性セクションに定義を持たないラベルの $label を持つ相対値式

`tst1 #label[reg1], reg2`	`movhi hi1(#label), reg1, r1` `tst1 lo(#label)[r1], reg2`

`tst1 label[reg1], reg2`	`movhi hi1(label), reg1, r1` `tst1 lo(label)[r1], reg2`

`tst1 $label[reg1], reg2`	`movhi hi1($label), reg1, r1` `tst1 lo($label)[r1], reg2`

- disp を省略した場合，as850 では，0 が指定されたものとみなされます。
- disp に #label を持つ相対値式，または #label を持つ相対値式に hi( )，lo( )，または hi1( ) を適用したものを指定した場合，その後ろの [reg1] の部分が省略できます。ただし，省略した場合，as850 では，[r0] が指定されたものとみなされます。
- disp に $label を持つ相対値式，または $label を持つ相対値式に hi( )，lo( )，または hi1( ) を適用したものを指定した場合，その後ろの [reg1] の部分が省略できます。ただし，省略した場合，as850 では，[gp] が指定されたものとみなされます。
- disp にデバイス・ファイルで定義されている周辺 I/O レジスタ名を指定した場合，その後ろの [reg1] の部分が省略できます。ただし，省略した場合，as850 では，[r0] が指定されたものとみなされます。

〔フラグ〕

CY	—
OV	—
S	—
Z	指定したビットが 0 の場合 1，1 の場合 0
SAT	—

3.8 スタック操作命令

この節では，スタック操作命令について説明します。

次に，この節において説明する命令を示します。

表 3 － 13 スタック操作命令

命令	意味
pop	スタック領域からのポップ（単一レジスタ）
popm	スタック領域からのポップ（複数レジスタ）
push	スタック領域へのプッシュ（単一レジスタ）
pushm	スタック領域へのプッシュ（複数レジスタ）

pop

〔概要〕

スタック領域からのポップ（Pop）

〔形式〕

(1) pop　　reg

〔機能〕

オペランドに指定したレジスタ値を，スタック領域からポップします。

〔説明〕

- pop 命令に対し，as850 では，命令展開が行われ，複数個の機械語命令が生成されます。

`pop　reg`	`ld.w　[sp], reg` `add　4, sp`

〔フラグ〕

命令展開が行われ，add 命令により設定されます。

CY	MSB（Most Significant Bit）からのキャリーを生じた場合 1，そうでない場合 0
OV	Integer-Overflow を生じた場合 1，そうでない場合 0
S	結果が負になった場合 1，そうでない場合 0
Z	結果が 0 になった場合 1，そうでない場合 0
SAT	—

popm

〔概要〕

スタック領域からのポップ（複数レジスタ）（Pop Multiple）

〔形式〕

(1) popm reg1, reg2, ...regN

〔機能〕

オペランドに指定したレジスタ値を，指定した順にスタック領域からポップします。なお，オペランドに指定可能なレジスタ数は，最大 32 **個です。**

〔説明〕

- popm **命令に対し，**as850 **では，命令展開が行われ，複数個の機械語命令が生成されます。**

レジスタが 3 **個以下の場合**

`popm reg1,...,regN`	`ld.w 4 * 0[sp], reg1`
	`:`
	`ld.w 4 * (N - 1)[sp], regN`
	`add 4 * N, sp`

レジスタが 4 **個以上の場合**

`popm reg1, reg2,...,regN`	`ld.w 4 * 0[sp], reg1`
	`ld.w 4 * 1[sp], reg2`
	`:`
	`ld.w 4 * (N - 1)[sp], regN`
	`addi 4 * N, sp, sp`

〔フラグ〕

命令展開が行われ，add ／ addi **命令により設定されます。**

CY	MSB（Most Significant Bit）**からのキャリーを生じた場合** 1**，そうでない場合** 0
OV	Integer-Overflow **を生じた場合** 1**，そうでない場合** 0
S	**結果が負になった場合** 1**，そうでない場合** 0
Z	**結果が** 0 **になった場合** 1**，そうでない場合** 0
SAT	—

push

〔概要〕

スタック領域へのプッシュ（Push）

〔形式〕

(1) push reg

〔機能〕

オペランドに指定したレジスタ値を，スタック領域にプッシュします。

〔説明〕

- push 命令に対し，as850 では，命令展開が行われ，複数個の機械語命令が生成されます。

`push reg`	`add -4. sp` `st.w reg, [sp]`

〔フラグ〕

命令展開が行われ，add 命令により設定されます。

CY	MSB（Most Significant Bit）からのキャリーを生じた場合 1，そうでない場合 0
OV	Integer-Overflow を生じた場合 1，そうでない場合 0
S	結果が負になった場合 1，そうでない場合 0
Z	結果が 0 になった場合 1，そうでない場合 0
SAT	—

pushm

〔概要〕

スタック領域へのプッシュ（複数レジスタ）（Push Multiple）

〔形式〕

(1) pushm reg1, reg2, ...regN

〔機能〕

オペランドに指定したレジスタ値を，スタック領域へプッシュします。なお，オペランドに指定可能なレジスタ数は，最大 32 個です。

〔説明〕

- pushm 命令に対し，as850 では，命令展開が行われ，複数個の機械語命令が生成されます。

レジスタが 4 個以下の場合

`pushm reg1, reg2,...,regN`	`add -4 * N, sp`
	`st.w regN, 4 * (N - 1)[sp]`
	`:`
	`st.w reg2, 4 * 1[sp]`
	`st.w reg1, 4 * 0[sp]`

レジスタが 5 個以上の場合

`pushm reg1, reg2,...,regN`	`addi -4 * N, sp, sp`
	`st.w regN, 4 * (N - 1)[sp]`
	`:`
	`st.w reg2, 4 * 1[sp]`
	`st.w reg1, 4 * 0[sp]`

〔フラグ〕

命令展開が行われ，add ／ addi 命令により設定されます。

CY	MSB（Most Significant Bit）からのキャリーを生じた場合 1，そうでない場合 0
OV	Integer-Overflow を生じた場合 1，そうでない場合 0
S	結果が負になった場合 1，そうでない場合 0
Z	結果が 0 になった場合 1，そうでない場合 0
SAT	—

3.9 特殊命令

この節では，特殊命令について説明します。

次に，この節において説明する命令を示します。

表 3 － 14 特殊命令

命令	意味
callt	テーブル参照コール【V850E】
ctret	callt からの復帰【V850E】
dbret	デバッグ・トラップからの復帰【V850E】
dbtrap	デバッグ・トラップ【V850E】
di	マスカブル割り込みの禁止
dispose	スタック・フレームの削除（関数の後処理）【V850E】
ei	マスカブル割り込みの許可
halt	プロセッサの停止
ldsr	システム・レジスタへのロード
nop	ノー・オペレーション
prepare	スタック・フレームの生成（関数の前処理）【V850E】
reti	トラップ，または割り込みルーチンからの復帰
stsr	システム・レジスタの内容のストア
switch	テーブル参照分岐【V850E】
trap	ソフトウエア・トラップ

callt

【V850E】

〔概要〕

テーブル参照コール（Call With Table Look Up）

〔形式〕

(1) callt imm6

imm6 に指定できるものを次に示します。

- 6 ビット幅までの値を持つ絶対値式

〔機能〕

- 次の順に処理を行います注。

(1) 復帰 PC と PSW の値を CTPC と CTPSW に退避します。

(2) オペランドで指定された値を 1 ビット左シフトして CTBP（CALLT Base Pointer）からのオフセット値とし，CTBP 値と加算してテーブル・エントリ・アドレスを生成します。

(3) 生成したテーブル・エントリ・アドレスから符号なしハーフワード・データをロードします。

(4) ロードした値と CTBP 値を加算してアドレスを生成します。

(5) 生成したアドレスへ分岐します。

注 システム・レジスタについては，各デバイスの“ユーザーズ・マニュアル アーキテクチャ編”を参照してください。

〔フラグ〕

CY	—
OV	—
S	—
Z	—
SAT	—

ctret

【V850E】

〔概要〕

callt からの復帰（Return From Callt）

〔形式〕

(1) ctret

〔機能〕

- callt による分岐から復帰します。次の順に処理を行います注。

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

(2) 取り出した値を PC と PSW に設定し，制御を移します。

注 システム・レジスタについては，各デバイスの“ユーザーズ・マニュアル アーキテクチャ編”を参照してください。

〔フラグ〕

CY	取り出した値
OV	取り出した値
S	取り出した値
Z	取り出した値
SAT	取り出した値

dbret

【V850E】

〔概要〕

デバッグ・トラップからの復帰（Return From Debug Trap）

〔形式〕

(1) dbret

〔機能〕

- **デバッグ・トラップから復帰します注。**

注　機能の詳細に関しては，各デバイスの“ユーザーズ・マニュアル アーキテクチャ編”を参照してください。

〔フラグ〕

CY	**取り出した値**
OV	**取り出した値**
S	**取り出した値**
Z	**取り出した値**
SAT	**取り出した値**

dbtrap

【V850E】

〔概要〕

デバッグ・トラップ（Debug Trap）

〔形式〕

(1) dbtrap

〔機能〕

- **デバッグ・トラップを起こします注。**

注　機能の詳細に関しては，各デバイスの“ユーザーズ・マニュアル アーキテクチャ編”を参照してください。

〔フラグ〕

CY	—
OV	—
S	—
Z	—
SAT	—

di

〔概要〕

マスカブル割り込みの禁止（Disable Interrupt）

〔形式〕

(1) di

〔機能〕

- PSW 中の ID ビットに 1 を設定し，本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。

〔フラグ〕

CY	—
OV	—
S	—
Z	—
SAT	—
ID	1

dispose

【V850E】

〔概要〕

スタック・フレームの削除（関数の後処理）（Function Dispose）

〔形式〕

(1) dispose imm, list

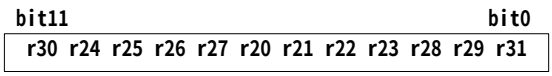
(2) dispose imm, list, [reg]

imm に指定できるものを次に示します。

- 32 ビット幅までの値を持つ絶対値式

list は，dispose 命令でポップ可能な 12 本のレジスタを指定するものです。list に指定できるものを，次に示します。

- レジスタ
  ポップの対象となるレジスタ（r20 ～ r31）をカンマで区切って指定します。
- 12 ビット幅までの値を持つ定数式
  12 ビットと 12 本のレジスタとの対応は次のとおりです。



次の 2 つの指定は等価です。

`dispose 0x10, r26, r29, r31`	`dispose 0x10, 0x103`

〔機能〕

dispose 命令は，関数の後処理をする命令です。

- (1) の形式

(1) 第一オペランドで指定した絶対値式の値をスタック・ポインタ（sp）に加算[注]し，sp をレジスタ退避領域に設定します。

(2) 第二オペランドで指定したレジスタを 1 つポップし，sp に 4 を加算します。

(3) 第二オペランドで指定したレジスタをすべてポップし終わるまで (2) を繰り返します。

注 機械語命令で実際に sp に加算される値は，imm を左へ 2 ビット・シフトした値となります。したがって，アセンブラは，指定した imm をあらかじめ右へ 2 ビット・シフトしてコードに反映します。

- (2) の形式

(1) 第一オペランドで指定した絶対値式の値をスタック・ポインタ（sp）に加算[注]し，sp をレジスタ退避領域に設定します。
(2) 第二オペランドで指定したレジスタを 1 つポップし，sp に 4 を加算します。
(3) 第二オペランドで指定したレジスタをすべてポップし終わるまで (2) を繰り返します。
(4) 第三オペランドで指定したレジスタ値をプログラム・カウンタ（PC）に設定します。

注 未定義シンボルやラベルの参照です。

〔説明〕

- imm に次のものを指定した場合，as850 では，機械語命令の dispose 命令を 1 つ生成します。

(1) 0 ～ 127 の範囲の絶対値式

`dispose imm, list`	`dispose imm, list`

`dispose imm, list, [reg]`	`dispose imm, list, [reg]`

- list に定数式以外を指定した場合，次のメッセージが出力され，アセンブルが中止されます。

```
E3249: illegal syntax
```

- imm に次のものを指定した場合，as850 では，命令展開が行われ，複数個の機械語命令が生成されます。

(a) 0 ～ 127 の範囲を越え，0 ～ 32767 の範囲の絶対値式

`dispose imm, list`	`movea imm, sp, sp` `dispose 0, list`

`dispose imm, list, [reg]`	`movea imm, sp, sp` `dispose 0, list, [reg]`

(b) 0 ～ 32767 の範囲を越える絶対値式

`dispose imm, list`	`mov imm, r1`
	`add r1, sp`
	`dispose 0, list`

`dispose imm, list, [reg]`	`mov imm, r1`
	`add r1, sp`
	`dispose 0, list, [reg]`

〔フラグ〕

CY	—
OV	—
S	—
Z	—
SAT	—

注意 命令展開により add 命令が生じた場合，フラグ値は変化する可能性があります。

〔注意事項〕

- 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。そのため sp の値は 4 バイト・アライメントした値を設定してください。
- 形式 (2) の [reg] に r0 を指定すると，次のメッセージが出力され，アセンブルが中止されます。

```
E3240: illegal operand (can not use r0 as destination in V850E mode)
```

ei

〔概要〕

マスカブル割り込みの許可（Enable Interrupt）

〔形式〕

(1) ei

〔機能〕

- PSW 中の ID ビットに 0 を設定し，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。

〔フラグ〕

CY	—
OV	—
S	—
Z	—
SAT	—
ID	0

halt

〔概要〕

停止（Halt）

〔形式〕

(1) halt

〔機能〕

- プロセッサを停止し，HALT 状態に遷移します。なお，HALT 状態からの処理再開は，マスカブル割り込み，NMI，リセットにより行われます。

〔フラグ〕

CY	—
OV	—
S	—
Z	—
SAT	—

ldsr

〔概要〕

システム・レジスタへのロード（Load System Register）

〔形式〕

(1) ldsr reg, regID

regID **に指定できるものを次に示します。**

- **5 ビット幅までの値を持つ絶対値式**

〔機能〕

- **第一オペランドに指定したレジスタ値を，第二オペランドに指定したシステム・レジスタ番号で示されるシステム・レジスタ注に格納します。**

注　システム・レジスタに関しては，各デバイスの“ユーザーズ・マニュアル アーキテクチャ編”，および次表を参照してください。

表 3 － 15 システム・レジスタ番号一覧（ldsr）

番号	システム・レジスタ	
0	**割り込み時状態待避レジスタ**	EIPC
1	**割り込み時状態待避レジスタ**	EIPSW
2	NMI **時状態待避レジスタ**	FEPC
3	NMI **時状態待避レジスタ**	FEPSW
4	**割り込み要因レジスタ注**	ECR
5	**プログラム・ステータス・ワード**	PSW
6-31	**予約**	－

注　割り込み要因レジスタは，オペランド指定できません。アクセス禁止です。

表 3 － 16 システム・レジスタ番号一覧【V850E/MA1】（ldsr）

番号	システム・レジスタ	
0	割り込み時状態待避レジスタ	EIPC
1	割り込み時状態待避レジスタ	EIPSW
2	NMI 時状態待避レジスタ	FEPC
3	NMI 時状態待避レジスタ	FEPSW
4	割り込み要因レジスタ注	ECR
5	プログラム・ステータス・ワード	PSW
6-15	予約	－
16	テーブル参照コール時状態退避レジスタ	CTPC
17	テーブル参照コール時状態退避レジスタ	CTPSW
18	不正命令例外時状態退避レジスタ	DBPC
19	不正命令例外時状態退避レジスタ	DBPSW
20	CALLT 用ベース・ポインタ	CTBP
21-31	予約	－

注　割り込み要因レジスタは，オペランド指定できません。アクセス禁止です。

表 3 － 17 システム・レジスタ番号一覧【V850E/ME2】（ldsr）

番号	システム・レジスタ	
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注 1	ECR
5	プログラム・ステータス・ワード	PSW
6-15	予約	－
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／デバッグ・トラップ時状態退避レジスタ注 2	DBPC
19	例外／デバッグ・トラップ時状態退避レジスタ注 2	DBPSW
20	CALLT ベース・ポインタ	CTBP
21	デバッグ・インタフェース・レジスタ注 2	DIR
22	ブレークポイント制御レジスタ 0，1 注 2，3	BPC0，BPC1
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 注 2，3	BPAV0，BPAV1
25	ブレークポイント・アドレス・マスク・レジスタ 0，1 注 2，3	BPAM0，BPAM1
26	ブレークポイント・データ設定レジスタ 0，1 注 2，3	BPDV0，BPDV1
27	ブレークポイント・データ・マスク・レジスタ 0，1 注 2，3	BPDM0，BPDM1
28-31	予約	－

注 1　割り込み要因レジスタは，オペランド指定できません。アクセス禁止です。

注 2　デバッグ・モード時だけアクセス可能です。

注 3　実際にアクセスされるレジスタは，DIR レジスタの CS ビットによって指定されます。

〔フラグ〕

CY	—
OV	—
S	—
Z	—
SAT	—

システム・レジスタにプログラム・ステータス・ワード（PSW）を指定した場合，各フラグには reg に対応したビットを設定します。

〔注意事項〕

- 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，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。
- regID に 0 ～ 31 の範囲を越える絶対値式を指定した場合，次のメッセージが出力され，指定した値を下位 5 ビット[注]を用いてアセンブルが続行されます。

注　機械語命令の ldsr 命令は，第二オペランドに 0 ～ 31（0x0 ～ 0x1f）の範囲のイミーディエトをとります。

```
W3011: illegal operand (range error in immediate)
```

- regID に予約レジスタ番号やアクセス禁止のレジスタ番号（ECR など）を指定した場合，またはデバッグ・モード時だけアクセス可能のレジスタ番号を指定した場合，次のメッセージが出力され，そのままアセンブルが続行されます。

```
W3018: illegal regID for ldsr
```

nop

〔概要〕

ノー・オペレーション（No Operation）

〔形式〕

(1) nop

〔機能〕

- **何も行いません。命令シーケンス内に領域を確保したり，命令実行に遅延を挿入したりするために用いることができます。**

〔フラグ〕

CY	—
OV	—
S	—
Z	—
SAT	—

prepare

【V850E】

〔概要〕

スタック・フレームの生成（関数の前処理）（Function Prepare）

〔形式〕

(1) prepare list, imm1

(2) prepare list, imm1, imm2

(3) prepare list, imm1, sp

imm1 ／ imm2 に指定できるものを次に示します。

- 32 ビット幅までの値を持つ絶対値式

list は，prepare 命令でプッシュ可能な 12 本のレジスタを指定するものです。list に指定できるものを次に示します。

- レジスタ

  プッシュの対象となるレジスタ（r20 ～ r31）をカンマで区切って指定します。

- 12 ビット幅までの値を持つ定数式

  12 ビットと 12 本のレジスタとの対応は次のとおりです。

bit11											bit0
r30	r24	r25	r26	r27	r20	r21	r22	r23	r28	r29	r31

次の 2 つの指定は等価です。

`prepare r26, r29, r31, 0x10`	`prepare 0x103, 0x10`

〔機能〕

prepare 命令は，関数の前処理をする命令です。

- (1) の形式

(1) 第一オペランドで指定したレジスタを 1 つプッシュし，スタック・ポインタ（sp）から 4 を減算します。

(2) 第一オペランドで指定したレジスタをすべてプッシュし終わるまで (1) を繰り返します。

(3) 第二オペランドで指定した絶対値式の値を sp から減算[注]し，sp をレジスタ退避領域に設定します。

- (2) の形式

(1) 第一オペランドで指定したレジスタを 1 つプッシュし，sp から 4 を減算します。

(2) 第一オペランドで指定したレジスタをすべてプッシュし終わるまで (1) を繰り返します。

(3) 第二オペランドで指定した絶対値式の値を sp から減算[注]し，sp をレジスタ退避領域に設定します。

(4) 第三オペランドで指定した絶対値式の値を ep に設定します。

- (3) の形式

(1) 第一オペランドで指定したレジスタを 1 つプッシュし，sp から 4 を減算します。

(2) 第一オペランドで指定したレジスタをすべてプッシュし終わるまで (1) を繰り返します。

(3) 第二オペランドで指定した絶対値式の値を sp から減算[注]し，sp をレジスタ退避領域に設定します。

(4) 第三オペランドで指定した sp の値を ep に設定します。

注 機械語命令で実際に sp に加算される値は，imm1 を左へ 2 ビット・シフトした値となります。このためアセンブラは，指定した imm1 をあらかじめ右へ 2 ビット・シフトしてコードに反映します。

〔説明〕

- imm1 に次のものを指定した場合，as850 では，機械語命令の prepare 命令が 1 つ生成されます。

(a) 0 ～ 127 の範囲の絶対値式

`prepare list, imm1`	`prepare list, imm1`
`prepare list, imm1, imm2`	`prepare list, imm1, imm2`
`prepare list, imm1, sp`	`prepare list, imm1, sp`

- list に定数式以外[注]を指定した場合，次のメッセージが出力され，アセンブルが中止されます。

注 未定義シンボルやラベルの参照です。

```
E3249: illegal syntax
```

- imm1 に次のものを指定した場合，as850 では，命令展開が行われ，複数個の機械語命令が生成されます。

(a) 0 ～ 127 の範囲を越え，0 ～ 32767 の範囲の絶対値式

`prepare list, imm1`	`prepare list, 0` `movea -imm1, sp, sp`
`prepare list, imm1, imm2`	`prepare list, 0, imm2` `movea -imm1, sp, sp`
`prepare list, imm1, sp`	`prepare list, 0, sp` `movea -imm1, sp, sp`

(b) 0 ～ 32767 の範囲を越える絶対値式

`prepare list, imm1`	`prepare list, 0`
	`mov imm1, r1`
	`sub r1, sp`

`prepare list, imm1, imm2`	`prepare list, 0, imm2`
	`mov imm1, r1`
	`sub r1, sp`

`prepare list, imm1, sp`	`prepare list, 0, sp`
	`mov imm1, r1`
	`sub r1, sp`

〔フラグ〕

CY	—
OV	—
S	—
Z	—
SAT	—

注 命令展開により sub 命令が生じた場合，フラグ値は変化します。

〔注意事項〕

- 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。そのため sp の値は 4 バイト・アライメントした値を設定してください。

reti

〔概要〕

トラップ，または割り込みルーチンからの復帰（Return from Trap or Interrupt）

〔形式〕

(1) reti

〔機能〕

- トラップ，または割り込みルーチンから復帰します注。

注　機能の詳細に関しては，各デバイスの“ユーザーズ・マニュアル アーキテクチャ編”を参照してください。

〔フラグ〕

CY	取り出した値
OV	取り出した値
S	取り出した値
Z	取り出した値
SAT	取り出した値

stsr

〔概要〕

システム・レジスタの内容のストア（Store System Register）

〔形式〕

(1) stsr regID, reg

regID **に指定できるものを次に示します。**

- **5 ビット幅までの値を持つ絶対値式**

〔機能〕

- **第一オペランドに指定したシステム・レジスタ番号で示されるシステム・レジスタ注の値を，第二オペランドに指定したレジスタに格納します。**

注 **システム・レジスタに関しては，各デバイスの“ユーザーズ・マニュアル アーキテクチャ編”，および次表を参照してください。**

表 3 － 18 システム・レジスタ番号一覧（stsr）

番号	システム・レジスタ	
0	割り込み時状態待避レジスタ	EIPC
1	割り込み時状態待避レジスタ	EIPSW
2	NMI 時状態待避レジスタ	FEPC
3	NMI 時状態待避レジスタ	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
6-31	予約	－

表 3 － 19 システム・レジスタ番号一覧【V850E/MA1】（stsr）

番号	システム・レジスタ	
0	割り込み時状態待避レジスタ	EIPC
1	割り込み時状態待避レジスタ	EIPSW
2	NMI 時状態待避レジスタ	FEPC
3	NMI 時状態待避レジスタ	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	予約	－
16	テーブル参照コール時状態退避レジスタ	CTPC
17	テーブル参照コール時状態退避レジスタ	CTPSW
18	不正命令例外時状態退避レジスタ	DBPC
19	不正命令例外時状態退避レジスタ	DBPSW
20	CALLT 用ベース・ポインタ	CTBP
21-31	予約	－

表 3 － 20 システム・レジスタ番号一覧【V850E/ME2】（stsr）

番号	システム・レジスタ	
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	予約	－
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／デバッグ・トラップ時状態退避レジスタ注 1	DBPC
19	例外／デバッグ・トラップ時状態退避レジスタ注 1	DBPSW
20	CALLT ベース・ポインタ	CTBP
21	デバッグ・インタフェース・レジスタ注 1	DIR
22	ブレークポイント制御レジスタ 0，1 注 1，2	BPC0，BPC1
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 注 1，2	BPAV0，BPAV1
25	ブレークポイント・アドレス・マスク・レジスタ 0，1 注 1，2	BPAM0，BPAM1
26	ブレークポイント・データ設定レジスタ 0，1 注 1，2	BPDV0，BPDV1
27	ブレークポイント・データ・マスク・レジスタ 0，1 注 1，2	BPDM0，BPDM1
28-31	予約	－

注 1　デバッグ・モード時だけアクセス可能です。

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

〔フラグ〕

CY	—
OV	—
S	—
Z	—
SAT	—

〔注意事項〕

- regID に 0 ～ 31 の範囲を越える絶対値式を指定した場合，次のメッセージが出力され，指定した値の下位 5 ビット[注]を用いてアセンブルが続行されます。

注 機械語命令の stsr 命令は，第一オペランドに 0 ～ 31（0x0 ～ 0x1f）の範囲のイミーディエトをとります。

```
W3011: illegal operand (range error in immediate)
```

- regID に予約レジスタ番号を指定した場合，またはデバッグ・モード時だけアクセス可能のレジスタ番号を指定した場合，次のメッセージが出力され，そのままアセンブルが続行されます。

```
W3018: illegal regID for stsr
```

switch

【V850E】

〔概要〕

テーブル参照分岐（Jump With Table Look Up）

〔形式〕

(1) switch reg

〔機能〕

- **次の順に処理を行います。**

(1) **オペランドで指定した値を** 1 **ビット論理左シフトした値とテーブルの先頭アドレス**（switch **命令の次のアドレス）とを加算し，テーブル・エントリ・アドレスを生成します。**
(2) **生成したテーブル・エントリ・アドレスから符号付きハーフワード・データをロードします。**
(3) **ロードした値を** 1 **ビット論理左シフトし，ワード長に符号拡張したあと，テーブルの先頭アドレスを加算し，アドレスを生成します。**
(4) **生成したアドレスへ分岐します。**

〔フラグ〕

CY	—
OV	—
S	—
Z	—
SAT	—

〔注意事項〕

- reg **に** r0 **を指定した場合，次のメッセージが出力され，アセンブルが中止されます。**

```
E3240: illegal operand (can not use r0 as source in V850E mode)
```

trap

〔概要〕

ソフトウエア・トラップ（Trap）

〔形式〕

(1) trap vector

vector **に指定できるものを次に示します。**

- 5 **ビット幅までの値を持つ絶対値式**

〔機能〕

- **ソフトウエア・トラップを起こします**[注]**。**

注 **機能の詳細に関しては，各デバイスの“ユーザーズ・マニュアル アーキテクチャ編”を参照してください。**

〔フラグ〕

CY	—
OV	—
S	—
Z	—
SAT	—

〔注意事項〕

- vector **に** 0 ～ 31 **の範囲を越える絶対値式を指定した場合，次のメッセージが出力され，指定した値の下位** 5 **ビット**[注]**を用いてアセンブルが続行されます。**

注 **機械語命令の** trap **命令は，オペランドに** 0 ～ 31（0x0 ～ 0x1f）**の範囲のイミーディエトをとります。**

```
W3011: illegal operand (range error in immediate)
```

第 4 章　疑似命令

この章では，CA850 アセンブラ（as850）がサポートする疑似命令について説明します。

4.1　形式の説明

as850 がサポートするアセンブリ言語疑似命令について，次の形式で説明します。なお，疑似命令とは，アセンブラが機械語命令を生成するために必要な前処理を行うものです。アセンブラに対し，セクション定義やファイル入力などを指示します。また，条件による出力コードの加工やマクロ置換などの指示もできます。

疑似命令

〔形式〕

疑似命令の形式を示します。

〔機能〕

疑似命令の機能を示します。

〔説明〕

疑似命令の機能の補足的な説明を示します。

〔注意事項〕

疑似命令における注意事項を示します。

〔例〕

疑似命令の使用例を示します。

4.2 セクション定義疑似命令

as850 では，セクション定義疑似命令を用いることにより，ソース・プログラム（アセンブリ言語）に対して生成するコードを指定したセクション[注]に割り当てることができます。

次に，この節において説明するセクション定義疑似命令を示します。

注 CA850 では，機械語命令やデータをセクションと呼ばれる単位に割り当てて扱います。

表 4 － 1 セクション定義疑似命令

疑似命令	意味
.bss	.bss セクションへの割り当て
.const	.const セクションへの割り当て
.data	.data セクションへの割り当て
.previous	現在のセクション定義疑似命令を指定しているセクション定義疑似命令の前のセクション定義疑似命令の（再）指定
.sbss	.sbss セクションへの割り当て
.sconst	.sconst セクションへの割り当て
.sdata	.sdata セクションへの割り当て
.sebss	.sebss セクションへの割り当て
.section	指定した種類のセクションへの割り当て
.sedata	.sedata セクションへの割り当て
.sibss	.sibss セクションへの割り当て
.sidata	.sidata セクションへの割り当て
.text	.text セクションへの割り当て
.tibss	.tibss セクションへの割り当て
.tibss.byte	.tibss.byte セクションへの割り当て
.tibss.word	.tibss.word セクションへの割り当て
.tidata	.tidata セクションへの割り当て
.tidata.byte	.tidata.byte セクションへの割り当て
.tidata.word	.tidata.word セクションへの割り当て
.vdbstrtab	.vdbstrtab セクションへの割り当て
.vdebug	.vdebug セクションへの割り当て
.vline	.vline セクションへの割り当て

なお，アセンブリ言語ソース・プログラム中にセクション定義疑似命令が 1 つもない場合，そのプログラムで生成されるセクションは，.text セクションとなります。

.bss

〔形式〕

.bss

〔機能〕

次のセクション定義疑似命令まで，または次のセクション定義疑似命令が現れなかった場合は，そのアセンブリ言語ソース・ファイルの終わりまでの間に記述したアセンブリ言語ソース・プログラムに対して生成するコードを，.bss セクション[注]に割り当てます。

注　セクション名.bss，セクション・タイプNOBITS，およびセクション属性AWを持つ予約セクションです。

〔説明〕

.bss セクションは，初期値を持たず，gp と 2 命令によって構成される，32 ビットのディスプレースメントを用いて参照されるメモリ範囲に配置されるセクションです。

〔例〕

次のセクション定義疑似命令まで .bss セクションとなる

```
        .bss
        .lcomm __stack, 0x100, 4
```

.const

〔形式〕

.const

〔機能〕

次のセクション定義疑似命令まで，または次のセクション定義疑似命令が現れなかった場合は，そのアセンブリ言語ソース・ファイルの終わりまでの間に記述したアセンブリ言語ソース・プログラムに対して生成するコードを，.const セクション[注]に割り当てます。

注 セクション名 .const，セクション・タイプ PROGBITS，およびセクション属性 A を持つ予約セクションです。

〔説明〕

.const セクションは，定数データ用（読み出し専用）のセクションで，r0 と 2 命令によって構成される，32 ビットのディスプレースメントを用いて参照されるメモリ範囲に配置されるセクションです。

〔例〕

次のセクション定義疑似命令まで .const セクションとなる

```
        .const
        .align 4
        .globl _p, 4
_p:
        .word 10
```

.data

〔形式〕

.data

〔機能〕

次のセクション定義疑似命令まで，または次のセクション定義疑似命令が現れなかった場合は，そのアセンブリ言語ソース・ファイルの終わりまでの間に記述したアセンブリ言語ソース・プログラムに対して生成するコードを，.data セクション[注]に割り当てます。

注　セクション名 .data，セクション・タイプ PROGBITS，およびセクション属性 AW を持つ予約セクションです。

〔説明〕

.data セクションは，初期値を持ち，gp と 2 命令によって構成される，32 ビットのディスプレースメントを用いて参照されるメモリ範囲に配置されるセクションです。

〔例〕

次のセクション定義疑似命令まで .data セクションとなる

```
        .data
        .align 4
        .globl _p, 4
_p:
        .word 10
```

.previous

〔形式〕

.previous

〔機能〕

現在のセクション定義疑似命令を指定しているセクション定義疑似命令の前のセクション定義疑似命令を（再）指定します。

たとえば，.data 疑似命令，.text 疑似命令，.previous 疑似命令の順に指定した場合，.previous 疑似命令の指定は .data 疑似命令の指定と等価になります。

〔例〕

.previous は .data と等価となる

```
        .data
        .align 4
        .globl _p, 4
_p:
        .word 10
        .text
lab:
        jbr    LL
        .previous
```

.sbss

〔形式〕

.sbss

〔機能〕

次のセクション定義疑似命令まで，または次のセクション定義疑似命令が現れなかった場合は，そのアセンブリ言語ソース・ファイルの終わりまでの間に記述したアセンブリ言語ソース・プログラムに対して生成するコードを，.sbss セクション[注]に割り当てます。

注　セクション名 .sbss，セクション・タイプ NOBITS，およびセクション属性 AWG を持つ予約セクションです。

〔説明〕

.sbss セクションは，初期値を持たず，gp と 16 ビットのディスプレースメントを用いて 1 命令で参照されるメモリ範囲（.sdata セクションとあわせて最大 64 K バイト）に配置されるセクションです。

〔例〕

次のセクション定義疑似命令まで .sbss セクションとなる

```
        .sbss
        .globl _1, 4
        .lcomm _1, 4, 4
```

.sconst

〔形式〕

.sconst

〔機能〕

次のセクション定義疑似命令まで，または次のセクション定義疑似命令が現れなかった場合は，そのアセンブリ言語ソース・ファイルの終わりまでの間に記述したアセンブリ言語ソース・プログラムに対して生成するコードを，.sconst セクション[注]に割り当てます。

注 セクション名 .sconst，セクション・タイプ PROGBITS，およびセクション属性 A を持つ予約セクションです。

〔説明〕

.sconst セクションは，定数データ用（読み出し専用）のセクションで，r0 と 16 ビットのディスプレースメントを用いて 1 命令で参照されるメモリ範囲（r0 からプラス方向に最大 32 K バイト）に配置されるセクションです。

〔例〕

次のセクション定義疑似命令まで .sconst セクションとなる

```
        .sconst
        .align 4
        .globl _p, 4
_p:
        .word 10
```

.sdata

〔形式〕

.sdata

〔機能〕

次のセクション定義疑似命令まで，または次のセクション定義疑似命令が現れなかった場合は，そのアセンブリ言語ソース・ファイルの終わりまでの間に記述したアセンブリ言語ソース・プログラムに対して生成するコードを，.sdata セクション[注]に割り当てます。

注　セクション名 .sdata，セクション・タイプ PROGBITS，およびセクション属性 AWG を持つ予約セクションです。

〔説明〕

.sdata セクションは，初期値を持ち，gp と 16 ビットのディスプレースメントを用いて 1 命令で参照されるメモリ範囲（.sbss セクションとあわせて最大 64 K バイト）に配置されるセクションです。

〔例〕

次のセクション定義疑似命令まで .sdata セクションとなる

```
        .sdata
        .align 4
        .globl _p, 4
_p:
        .word 10
```

.sebss

〔形式〕

.sebss

〔機能〕

次のセクション定義疑似命令まで，または次のセクション定義疑似命令が現れなかった場合は，そのアセンブリ言語ソース・ファイルの終わりまでの間に記述したアセンブリ言語ソース・プログラムに対して生成するコードを，.sebss セクション[注]に割り当てます。

注　セクション名 .sebss，セクション・タイプ NOBITS，およびセクション属性 AW を持つ予約セクションです。

〔説明〕

.sebss セクションは，初期値を持たず，ep と 16 ビットのディスプレースメントを用いて 1 命令で参照されるメモリ範囲（ep からマイナス方向に最大 32 K バイト）のうち，.sedata セクションのサイズ分だけ上位のアドレスに配置されるセクションです。

〔例〕

次のセクション定義疑似命令まで .sebss セクションとなる

```
        .sebss
        .globl _1, 4
        .lcomm _1, 4, 4
```

.section

〔形式〕

.section " セクション名 " [, セクションの種類]

〔機能〕

次のセクション定義疑似命令まで，または次のセクション定義疑似命令が現れなかった場合は，そのアセンブリ言語ソース・ファイルの終わりまでの間に記述したアセンブリ言語ソース・プログラムに対して生成するコードを，第一オペランドに指定したセクション名で，第二オペランドに指定した種類のセクションに割り付けます。

セクションの種類には，次に示す 7 種類があります[注]。

注　セクションの種類の指定には大文字を用いることもできます（たとえば，text のかわりに TEXT を用いることもできます）。

表 4 － 2　セクションの種類

種類	意味
bss	bss 属性セクション セクション・タイプ NOBITS とセクション属性 AW を持つセクション
const	const 属性セクション セクション・タイプ PROGBITS とセクション属性 A を持つセクション
data	data 属性セクション セクション・タイプ PROGBITS とセクション属性 AW を持つセクション
sbss	sbss 属性セクション セクション・タイプ NOBITS とセクション属性 AWG を持つセクション
sdata	sdata 属性セクション セクション・タイプ PROGBITS とセクション属性 AWG を持つセクション
text	text 属性セクション セクション・タイプ PROGBITS とセクション属性 AX を持つセクション
comment	comment 属性セクション セクション・タイプ PROGBITS を持ちセクション属性を持たないセクション

〔例〕

sec という名前で data 属性セクションを定義する

```
        .section "sec", data
        .align 4
        .globl _p, 4
_p:
        .word 10
```

〔注意事項〕

- CA850 **において，セクション名** .text, .data, .bss, .sdata, .sbss, .sconst, .const, .sidata, .sibss, .sedata, .sebss, .tidata, .tibss, .tidata.byte, .tibss.byte, .tidata.word, .tibss.word, .pro_epi_runtime, .version **は予約されており，これらの予約セクションとセクションの種類の対応は次のようになっています。**

表 4 － 3　予約セクションとセクションの種類の対応

予約セクション名	セクションの種類
.pro_epi_runtime .text	text
.data .sedata .sidata .tidata .tidata.byte .tidata.word	data
.bss .sebss .sibss .tibss .tibss.byte .tibss.word	bss
.sdata	sdata
.sbss	sbss
.const .sconst	const
.version	comment

したがって，これらのセクション名を第一オペランドに指定した場合，第二オペランドは省略するか，それぞれの予約セクションに対して定められているセクションの種類を指定しなければなりません。なお，定められているセクションの種類と異なる種類を指定した場合，次のメッセージが出力され，アセンブルが中止されます。

```
F3504: illegal section kind
```

- **上記の予約セクション名以外の名前を第一オペランドに指定し，第二オペランドを省略した場合，セクションの種類として** text **が指定されたものとみなされます。**
- **ある名前を持つセクションに対して複数の異なるセクションの種類を指定した場合，次のメッセージが出力され，アセンブルが中止されます。**

```
F3504: illegal section kind
```

- **第一オペランドにデバイス・ファイルで定義されている割り込み要求名を指定した場合，リンカが，そのセクションを該当するハンドラ・アドレスに自動割り付けします。したがって，割り込み要求名を指定したセクションに対しては，リンカで配置アドレスを指定することはできません。また，割り込みハンドラでないセクションに対し，割り込み要求名を指定しないようにしてください。**

[割り込み要求名使用例]

リセット時に __start へジャンプするセクションを定義する

```
        .section "RESET", text
        jr      __start
```

.sedata

〔形式〕

.sedata

〔機能〕

次のセクション定義疑似命令まで，または次のセクション定義疑似命令が現れなかった場合は，そのアセンブリ言語ソース・ファイルの終わりまでの間に記述したアセンブリ言語ソース・プログラムに対して生成するコードを，.sedata セクション[注]に割り当てます。

注　セクション名 .sedata，セクション・タイプ PROGBITS，およびセクション属性 AW を持つ予約セクションです。

〔説明〕

.sedata セクションは，初期値を持ち，ep と 16 ビットのディスプレースメントを用いて 1 命令で参照されるメモリ範囲（ep からマイナス方向に最大 32 K バイト）のうち，.sebss セクションのサイズ分だけ下位のアドレスに配置されるセクションです。

〔例〕

次のセクション定義疑似命令まで .sedata セクションとなる

```
        .sedata
        .align 4
        .globl _p, 4
_p:
        .word 10
```

.sibss

〔形式〕

.sibss

〔機能〕

次のセクション定義疑似命令まで，または次のセクション定義疑似命令が現れなかった場合は，そのアセンブリ言語ソース・ファイルの終わりまでの間に記述したアセンブリ言語ソース・プログラムに対して生成するコードを，.sibss セクション[注]に割り当てます。

注　セクション名 .sibss，セクション・タイプ NOBITS，およびセクション属性 AW を持つ予約セクションです。

〔説明〕

.sibss セクションは，初期値を持たず，ep と 16 ビットのディスプレースメントを用いて 1 命令で参照されるメモリ範囲（epからプラス方向に最大32 Kバイト）のうち，使用された.tidata.byte，.tibss.byte，.tidata.word，.tibss.word，.tidata，.tibss，.sidata セクションのサイズ分だけ上位のアドレスに配置されるセクションです（「図 2 － 1 内蔵 RAM のメモリ配置イメージ」参照）。

〔例〕

次のセクション定義疑似命令まで .sibss セクションとなる

```
        .sibss
        .globl _1, 4
        .lcomm _1, 4, 4
```

.sidata

〔形式〕

.sidata

〔機能〕

次のセクション定義疑似命令まで，または次のセクション定義疑似命令が現れなかった場合は，そのアセンブリ言語ソース・ファイルの終わりまでの間に記述したアセンブリ言語ソース・プログラムに対して生成するコードを，.sidata セクション[注]に割り当てます。

注　セクション名 .sidata，セクション・タイプ PROGBITS，およびセクション属性 AW を持つ予約セクションです。

〔説明〕

.sidata セクションは，ep と 16 ビットのディスプレースメントを用いて 1 命令で参照されるメモリ範囲（ep からプラス方向に最大 32 K バイト）のうち，使用された .tidata.byte，.tibss.byte，.tidata.word，.tibss.word，.tidata，.tibss セクションのサイズ分だけ上位のアドレスに配置されるセクションです（「図 2 － 1 内蔵 RAM のメモリ配置イメージ」参照）。

〔例〕

次のセクション定義疑似命令まで .sidata セクションとなる

```
        .sidata
        .align 4
        .globl _p, 4
_p:
        .word 10
```

.text

〔形式〕

.text

〔機能〕

次のセクション定義疑似命令まで，または次のセクション定義疑似命令が現れなかった場合は，そのアセンブリ言語ソース・ファイルの終わりまでの間に記述したアセンブリ言語ソース・プログラムに対して生成するコードを，.text セクション[注1] に割り当てます[注2]。

注 1　セクション名 .text，セクション・タイプ PROGBITS，およびセクション属性 AX を持つ予約セクションです。

注 2　as850 では，1 つのアセンブリ言語ソース・ファイル中のアセンブリ言語ソース・プログラムの前には .text が 2 回指定されているものとみなされます（たとえば，セクション定義疑似命令を指定する前に“ .word 1 ”を指定した場合，それは .text セクションに割り当てられます）。ただし，.text セクションを明示的に指定せずかつデフォルトで指定された .text セクションに対しラベルの定義,命令,ロケーション・カウンタ制御疑似命令，領域確保疑似命令のいずれも指定しなかった場合，as850 では，.text セクションが生成されません。

〔例〕

次のセクション定義疑似命令まで .text セクションとなる

```
       .text
       .align 4
       .globl __start
__start:
       mov    #__tp_TEXT, tp
```

.tibss

〔形式〕

.tibss

〔機能〕

次のセクション定義疑似命令まで，または次のセクション定義疑似命令が現れなかった場合は，そのアセンブリ言語ソース・ファイルの終わりまでの間に記述したアセンブリ言語ソース・プログラムに対して生成するコードを，.tibss セクション[注]に割り当てます。

注　セクション名 .tibss，セクション・タイプ NOBITS，およびセクション属性 AW を持つ予約セクションです。

〔説明〕

.tidbss セクションは，V850 マイクロコントローラの内蔵 RAM に置かれ，sld ／ sst 命令を用いて ep 相対でアクセスされることを前提とした初期値なしデータです。as850，および ld850 は，.tidata.byte，.tibss.byte，.tidata.word，.tibss.word，.tidata のいずれのセクションも使用されていない場合，ep の指すアドレスに .tibss を配置します。.tidata.byte，.tibss.byte，.tidata.word，.tibss.word，.tidata のいずれかのセクションが使用されている場合，“ ep の示すアドレス + 使用された .tidata.byte ／ .tibss.byte ／ .tidata.word ／ .tibss.word ／ .tidata のセクションのサイズ ”のアドレスに，.tibss を配置します（「図２－１ 内蔵 RAM のメモリ配置イメージ」参照）。

sld ／ sst 命令では，データのサイズによってアクセスする領域の範囲が異なるため，より有効に sld ／ sst 命令を用いるためには，バイト・データは .tidata.byte ／ .tibss.byte セクションに，ハーフワード以上のデータは .tidata.word ／ .tibss.word セクションに置くことを推奨します。ただし，内蔵 RAM に置くデータが少なく，アクセス領域を細かく考慮する必要がない場合，初期値なしデータに関しては本疑似命令でデータを .tibss セクションに置いて，データをサイズで分ける手間を省くこともできます。

〔例〕

次のセクション定義疑似命令まで .tibss セクションとなる

```
        .tibss
        .globl _1, 4
        .lcomm _1, 4, 4
```

.tibss.byte

〔形式〕

.tibss.byte

〔機能〕

次のセクション定義疑似命令まで，または次のセクション定義疑似命令が現れなかった場合は，そのアセンブリ言語ソース・ファイルの終わりまでの間に記述したアセンブリ言語ソース・プログラムに対して生成するコードを，.tibss.byte セクション[注]に割り当てます。

注　セクション名 .tibss.byte，セクション・タイプ NOBITS，およびセクション属性 AW を持つ予約セクションです。

〔説明〕

.tibss.byte セクションは，V850 マイクロコントローラの内蔵 RAM に置かれ，sld ／ sst 命令を用いて ep 相対でアクセスされることを前提としています。sld ／ sst 命令では，

- アクセスするデータがバイト・データの場合，128 バイト以内
- ハーフワード以上のデータの場合，256 バイト以内

の領域範囲をアクセスできます。as850，および ld850 は，sld ／ sst 命令がアクセスできる領域を有効に使用するため，データのサイズによって，セクションを，.tidata.byte ／ .tibss.byte と .tidata.word ／ .tibss.word に分け，.tibss.byte を，“ ep の示すアドレスに + 使用された .tidata.byte セクションのサイズ ” のアドレスへ配置するようにしています（「図 2 － 1 内蔵 RAM のメモリ配置イメージ」参照）。したがって，内蔵 RAM に置く初期値なしバイト・データは，本疑似命令により，.tibss.byte セクションに置くことを推奨します[注]。

注　バイト・データを .tibss.word セクションに置いてアクセスすること自体はできます。

〔例〕

次のセクション定義疑似命令まで .tibss.byte セクションとなる

```
        .tibss.byte
        .globl _1, 4
        .lcomm _1, 4, 4
```

.tibss.word

〔形式〕

.tibss.word

〔機能〕

次のセクション定義疑似命令まで，または次のセクション定義疑似命令が現れなかった場合は，そのアセンブリ言語ソース・ファイルの終わりまでの間に記述したアセンブリ言語ソース・プログラムに対して生成するコードを，.tibss.word セクション[注]に割り当てます。

注　セクション名 .tibss.word，セクション・タイプ NOBITS，およびセクション属性 AW を持つ予約セクションです。

〔説明〕

.tibss.word セクションは，V850 マイクロコントローラの内蔵 RAM に置かれ，sld ／ sst 命令を用いて ep 相対でアクセスされることを前提としています。sld ／ sst 命令では，

- アクセスするデータがバイト・データの場合，128 バイト以内
- ハーフワード以上のデータの場合，256 バイト以内

の領域範囲をアクセスできます。as850，および ld850 は，sld ／ sst 命令がアクセスできる領域を有効に使用するため，データのサイズによって，セクションを，.tidata.byte ／ .tibss.byte と .tidata.word ／ .tibss.word に分け，.tibss.word を，“ ep の示すアドレス + 使用された .tidata.byte ／ .tibss.byte ／ .tidata.word セクションのサイズ ”のアドレスへ配置するようにしています（「図 2 － 1 内蔵 RAM のメモリ配置イメージ」参照）。したがって，内蔵 RAM に置くハーフワード以上の初期値なしデータは，本疑似命令により .tibss.word セクションに置くようにしてください。

〔例〕

次のセクション定義疑似命令まで .tibss.word セクションとなる

```
        .tibss.word
        .globl _1, 4
        .lcomm _1, 100000, 4
```

.tidata

〔形式〕

.tidata

〔機能〕

次のセクション定義疑似命令まで，または次のセクション定義疑似命令が現れなかった場合は，そのアセンブリ言語ソース・ファイルの終わりまでの間に記述したアセンブリ言語ソース・プログラムに対して生成するコードを，.tidata セクション[注]に割り当てます。

注　セクション名 .tidata，セクション・タイプ PROGBITS，およびセクション属性 AW を持つ予約セクションです。

〔説明〕

.tidata セクションは，V850 マイクロコントローラの内蔵 RAM に置かれ，sld ／ sst 命令を用いて ep 相対でアクセスされることを前提としています。as850，および ld850 は，.tidata.byte，.tibss.byte，.tidata.word，.tibss.word のいずれのセクションも使用されていない場合，ep の示すアドレスに .tidata を配置します。.tidata.byte，.tibss.byte，.tidata.word，.tibss.word のいずれかのセクションが使用されている場合，“ ep の示すアドレス + 使用された .tidata.byte ／ .tibss.byte ／ .tidata.word ／ .tibss.word セクションのサイズ ” のアドレスに，.tidata を配置します（「図 2 － 1 内蔵 RAM のメモリ配置イメージ」参照）。

sld ／ sst 命令では，データのサイズによってアクセスする領域の範囲が異なるため，より有効に sld ／ sst 命令を用いるためには，バイト・データは .tidata.byte ／ .tibss.byte セクションに，ハーフワード以上のデータは .tidata.word ／ .tibss.word セクションに置くことを推奨します。ただし，内蔵 RAM に置くデータが少なく，アクセス領域を細かく考慮する必要がない場合，本疑似命令でデータを .tidata セクションに置いて，データをサイズで分ける手間を省くこともできます。

〔例〕

次のセクション定義疑似命令まで .tidata セクションとなる

```
        .tidata
        .align 4
        .globl _p, 4
_p:
        .word 10
```

.tidata.byte

〔形式〕

.tidata.byte

〔機能〕

次のセクション定義疑似命令まで，または次のセクション定義疑似命令が現れなかった場合は，そのアセンブリ言語ソース・ファイルの終わりまでの間に記述したアセンブリ言語ソース・プログラムに対して生成するコードを，.tidata.byte セクション[注]に割り当てます。

注　セクション名 .tidata.byte，セクション・タイプ PROGBITS，およびセクション属性 AW を持つ予約セクションです。

〔説明〕

.tidata.byte セクションは，V850 マイクロコントローラの内蔵 RAM に置かれ，sld ／ sst 命令を用いて ep 相対でアクセスされることを前提としています。sld ／ sst 命令では，

- アクセスするデータがバイト・データの場合，128 バイト以内
- ハーフワード以上のデータの場合，256 バイト以内

の領域範囲をアクセスできます。as850，および ld850 は，sld ／ sst 命令がアクセスできる領域を有効に使用するため，データのサイズによって，セクションを，.tidata.byte ／ .tibss.byte と .tidata.word ／ .tibss.word に分け，.tidata.byte を ep が示すアドレスへ配置するようにしています（「図 2 － 1 内蔵 RAM のメモリ配置イメージ」参照）。したがって，内蔵 RAM に置く初期値ありバイト・データは，本疑似命令により .tidata.byte セクションに置くことを推奨します[注]。

注　初期値ありバイト・データを .tidata.word セクションに置いてアクセスすること自体はできます。

〔例〕

次のセクション定義疑似命令まで .tidata.byte セクションとなる

```
        .tidata.byte
        .globl _p, 1
_p:
        .byte 1
```

.tidata.word

〔形式〕

.tidata.word

〔機能〕

次のセクション定義疑似命令まで，または次のセクション定義疑似命令が現れなかった場合は，そのアセンブリ言語ソース・ファイルの終わりまでの間に記述したアセンブリ言語ソース・プログラムに対して生成するコードを，.tidata.word セクション[注]に割り当てます。

注　セクション名 .tidata.word，セクション・タイプ PROGBITS，およびセクション属性 AW を持つ予約セクションです。

〔説明〕

.tidata.word セクションは，V850 マイクロコントローラの内蔵 RAM に置かれ，sld ／ sst 命令を用いて ep 相対でアクセスされることを前提としています。sld ／ sst 命令では，

- アクセスするデータがバイト・データの場合，128 バイト以内
- ハーフワード以上のデータの場合，256 バイト以内

の領域範囲をアクセスできます。as850，および ld850 は，sld ／ sst 命令がアクセスできる領域を有効に使用するため，データのサイズによって，セクションを，.tidata.byte ／ .tibss.byte と .tidata.word ／ .tibss.word に分け，.tidata.word を，“ ep の示すアドレス + .tidata.byte ／ .tibss.byte セクションのサイズ ” のアドレスへ配置するようにしています（「図 2 － 1 内蔵 RAM のメモリ配置イメージ」参照）。したがって，内蔵 RAM に置くハーフワード以上の初期値ありデータは，本疑似命令により .tidata.word セクションに置くようにしてください。

〔例〕

次のセクション定義疑似命令まで .tidata.word セクションとなる

```
        .tidata.word
        .align 4
        .globl _p, 4
_p:
        .word 100000
```

.vdbstrtab

〔形式〕

.vdbstrtab

〔機能〕

次のセクション定義疑似命令まで，または次のセクション定義疑似命令が現れなかった場合は，そのアセンブリ言語ソース・ファイルの終わりまでの間に記述したアセンブリ言語ソース・プログラムに対して生成するコードを，.vdbstrtab **セクション[注]に割り当てます。**

注　　セクション名 .vdbstrtab **とセクション・タイプ** STRTAB **を持つ予約セクションです。**

.vdebug

〔形式〕

.vdebug

〔機能〕

次のセクション定義疑似命令まで，または次のセクション定義疑似命令が現れなかった場合は，そのアセンブリ言語ソース・ファイルの終わりまでの間に記述したアセンブリ言語ソース・プログラムに対して生成するコードを，.vdebug セクション[注]に割り当てます。

注　セクション名 .vdebug とセクション・タイプ PROGBITS を持つ予約セクションです。

.vline

〔形式〕

.vline

〔機能〕

次のセクション定義疑似命令まで，または次のセクション定義疑似命令が現れなかった場合は，そのアセンブリ言語ソース・ファイルの終わりまでの間に記述したアセンブリ言語ソース・プログラムに対して生成するコードを，.vline **セクション**[注]**に割り当てます。**

注　　セクション名 .vline **とセクション・タイプ** PROGBITS **を持つ予約セクションです。**

4.3 シンボル制御疑似命令

as850 では，シンボル制御疑似命令を用いることにより，シンボル・テーブル・エントリの生成，シンボルの定義，およびラベルの示すデータ・サイズの指定ができます。

次に，この節において説明するシンボル制御疑似命令を示します。

表 4 － 4　シンボル制御疑似命令

疑似命令	意味
.ext_ent_size	フラッシュ・テーブル・エントリのサイズ指定
.ext_func	フラッシュ・テーブル・エントリの生成
.file	シンボル・テーブル・エントリの生成（FILE タイプ）
.frame	シンボル・テーブル・エントリの生成（FUNC タイプ）
.set	シンボルの定義
.size	ラベルの示すデータ・サイズの指定

なお，シンボル制御疑似命令において指定するサイズ（バイト数）は，2^{31} 未満にしてください。2^{31} 以上の値を指定した場合，次のメッセージが出力され，アセンブルが中止されます。

```
E3247: illegal size value
```

.ext_ent_size

〔形式〕

.ext_ent_size サイズ

〔機能〕

オブジェクト・ファイルの生成時，フラッシュ・テーブル・エントリ・サイズにオペランドに指定した値を設定します。フラッシュ／外部 ROM 再リンク機能を使用する際に指定します。

〔説明〕

書き換え／取り換え不可能領域（以降，ブート領域）から，可能領域（以降，フラッシュ領域）へ分岐する場合，本疑似命令を指定することにより，フラッシュ領域側の指定したアドレスに分岐テーブルが作成され，テーブルを介しての二段分岐が行われます。このテーブルのエントリ・サイズはデフォルトで 4 バイトであり，jr 命令が生成され，分岐命令から 22 ビットの範囲に分岐可能です。このテーブルの分岐命令から 22 ビットの範囲を越えるアドレスに分岐する必要がある場合，本疑似命令でエントリ・サイズに V850 コアの場合には 10 を V850Ex コアの場合には 8 を指定することにより，32 ビットアドレス空間全体へ分岐可能です。

〔注意事項〕

- 本疑似命令は，ブート領域側の該当する分岐命令のあるソース・ファイル，およびフラッシュ領域側の該当するラベル定義のあるソース・ファイルに記述する必要があります。
- 本疑似命令にて指定するサイズは，ブート領域／フラッシュ領域を含め，全体で唯一の値となります。
  異なるサイズを指定した場合，次のメッセージを出力し，アセンブルを続行します。

```
W3021: .ext_ent_size already specified, ignored.
```

なお，複数のリロケータブル・オブジェクト・ファイル間で異なるサイズが指定された場合，リンク時にエラーとなります。

- 上記不整合を防止するため，該当するラベル名すべてを 1 ファイルにまとめて記述し，.include 疑似命令を用いて，ブート／フラッシュ両側のソース・ファイルにインクルードすることを推奨します。
- サイズは，4（デフォルト），8【V850E】，10【V850】のいずれかを指定してください。
  なお，共通オブジェクト（-cn オプション指定）作成時には，V850 と V850Ex の両方で動作する必要があるため，8【V850E】の指定はできません。

.ext_func

〔形式〕

.ext_func **ラベル名** , ID **値**

〔機能〕

オブジェクト・ファイルの生成時，オペランドに指定したラベル名と，ID **値を持つフラッシュ・テーブル・エントリを生成します。フラッシュ／外** ROM **再リンク機能を使用する際に指定します。**

〔説明〕

書き換え／取り換え不可能領域（以降，ブート領域）から，可能領域（以降，フラッシュ領域）へ分岐する場合，本疑似命令を指定することにより，フラッシュ領域側の指定したアドレスに分岐テーブルが作成され，テーブルを介しての二段分岐が行われます。

〔注意事項〕

本疑似命令は，ブート領域側の該当する分岐命令のあるソース・ファイル，およびフラッシュ領域側の該当するラベル定義のあるソース・ファイルに記述する必要があります。

- **同じラベル名で異なる** ID **値を指定した場合，次のメッセージが出力され，アセンブルが中止されます。**

```
E3253: symbol "identifier" already defined as another id
```

- **同じ** ID **値で異なるラベル名を指定した場合，次のメッセージが出力され，アセンブルが中止されます。**

```
E3252: id already defined as symbol "identifier"
```

- **上記不整合を防止するため，該当するラベル名すべてを** 1 **ファイルにまとめて記述し，**.include **疑似命令を用いて，ブート／フラッシュ両側のソース・ファイルにインクルードすることを推奨します。**
- ID **値は正数を指定してください。また，確保される分岐テーブルのサイズは** ID **値の最大に依存します。**ID **値は詰めて使用することを推奨します。**

.file

〔形式〕

.file " ファイル名 "

〔機能〕

オブジェクト・ファイルの生成時，オペランドに指定したファイル名と，タイプ FILE を持つシンボル・テーブル・エントリ注を生成します。なお，入力したソース・ファイル中に本疑似命令が存在しない場合，「.file “ 入力ファイル名 ”」を指定したものとみなし，入力ファイル名とタイプ FILE を持つシンボル・テーブル・エントリを生成します。

注　バインディング・クラスは LOCAL とします。

.frame

〔形式〕

.frame ラベル名 , サイズ

〔機能〕

オブジェクト・ファイル生成時，第一オペランドに指定したラベルに対するシンボル・テーブル・エントリの生成において，第二オペランドに指定したサイズとタイプ FUNC を持つシンボル・テーブル・エントリを生成します注。

注　C 言語レベルでのデバッグのために使用する疑似命令です。アセンブラ・デバッグ機能向けに記述する場合，サイズには 0 を指定してください。

.set

〔形式〕

.set シンボル名 , 値

〔機能〕

第一オペランドに指定したシンボル名と，第二オペランドに指定した値（整数値）を持つシンボルを定義します。1 つのアセンブリ言語ソース・ファイルにおいて，あるシンボルに対して複数回 .set 疑似命令を指定した場合，そのシンボルの参照は，その参照が現れた位置に従って，次の値を持ちます。

- ファイルの先頭からそのシンボルに対する最初の .set 疑似命令までの間に現れた場合
  そのシンボルに対する最後の .set 疑似命令で指定した値
- ある .set 疑似命令から次の .set 疑似命令までの間，または次の .set 疑似命令が現れなかった場合には，そのアセンブリ言語ソース・ファイルの終わりまでの間に現れた場合
  その .set 疑似命令で指定した値

〔注意事項〕

- 値の指定にラベル参照や，その時点において未定義なシンボル参照を用いることはできません。
  値の指定にラベル参照や，その時点において未定義なシンボル参照を用いた場合，次のメッセージが出力され，アセンブルが終了されます。

```
E3203: illegal expression (string)
```

- ラベル名，.macro 疑似命令で定義済みのマクロ名，およびマクロの仮パラメータと同名のシンボルを指定した場合，次のメッセージが出力され，アセンブルが終了されます。

```
E3212: symbol already define as string
```

〔例〕

シンボル sym1 の値を 0x10 として定義する

```
        .set sym1, 0x10
```

.size

〔形式〕

.size ラベル名 , サイズ

〔機能〕

第二オペランドに指定したサイズを，第一オペランドに指定したラベルの示すデータのサイズとして指定します注。

注　すでにサイズが設定されていた場合，その値は上書きされます。

〔注意事項〕

CA850 に含まれるリンカにおける -A オプションを用いる場合，sdata 属性セクションに割り当てるデータ（実際には gp オフセット参照されているラベル）の定義に対しては，本疑似命令，または .globl 疑似命令を用いてサイズを設定してください注。

注　上記の方法で設定しなかった場合，リンカにおける -A オプションにおいて正しい情報が得られません。

〔例〕

label1 のサイズは 15 とする

```
        .size label1, 15
```

4.4 ロケーション・カウンタ制御疑似命令

as850 では，ロケーション・カウンタ制御疑似命令を用いることにより，ロケーション・カウンタ[注]の値を整列したり進めたりできます。

次に，この節において説明するロケーション・カウンタ制御疑似命令を示します。

注 セクションごとに存在し，そのファイル内の対応するセクションに対する最初のセクション定義疑似命令が現れたときに 0 に初期化されます。

表 4 － 5 ロケーション・カウンタ制御疑似命令

疑似命令	意味
.align	ロケーション・カウンタの値の整列
.org	ロケーション・カウンタの値を進める

なお，sbss ／ bss 属性セクションにおいてロケーション・カウンタ制御疑似命令を指定した場合，次のメッセージが出力され，アセンブルが中止されます。

```
E3246: illegal section
```

.align

〔形式〕

.align 整列条件 [, フィリング値]

〔機能〕

前に指定されたセクション定義疑似命令によって指定される現在のセクションに対するロケーション・カウンタ値を，第一オペランドで指定した整列条件で整列します。なお，ロケーション・カウンタ値を整列したことによりホールが生じた場合，生じたホールを第二オペランドで指定したフィリング値，またはデフォルト値の 0 で埋めます。

たとえば，現在のロケーション・カウンタ値が 3 で .align 4 を指定した場合，ロケーション・カウンタ値を整列条件 4（ワード境界）で整列して 4 とし，生じた 1 バイト分のホールをデフォルト値の 0 で埋めます。

〔注意事項〕

- 整列条件は 2 以上 2^{31} 未満の偶数にしてください。それ以外のものを指定した場合，次のメッセージが出力され，アセンブルが中止されます。

```
E3200: illegal alignment value
```

- フィリング値を指定する場合，1 バイト分の値を指定します。1 バイト分以上の値を指定した場合，下位 1 バイト分が用いられます。
- sdata 属性セクションにおいて 4 より大きな整列条件を指定して本疑似命令を用いた場合，sdata ／ sbss 属性セクションに割り当てるデータのサイズを指定するための目安を表示する（ld850 の -A オプション）際に，正しい情報が得られなくなります。
- 本疑似命令は，そのセクションに対する指定したファイル内でのロケーション・カウンタ値を整列するだけであり，絶対アドレス[注 1] を整列するものでも，セクション内オフセット[注 2] を整列するものでもありません。

注 1　リンク後のオブジェクト・ファイルにおけるアドレス 0 からのオフセットです。

注 2　リンク後のオブジェクト・ファイルにおいてそのセクションが割り当てられたセクション（出力セクション）の先頭アドレスからのオフセットです。

〔例〕

16 バイト整列する

```
        .align 16
```

.org

〔形式〕

.org 値

〔機能〕

前に指定されたセクション定義疑似命令によって指定される現在のセクションに対するロケーション・カウンタ値を，オペランドで指定した値（2^{31} 未満）まで進めます。なお，ロケーション・カウンタ値を進めることによりホールが生じた場合，生じたホールを 0 で埋めます。

〔注意事項〕

- 現在のロケーション・カウンタ値より小さな値を指定した場合，次のメッセージが出力され，アセンブルが中止されます。

```
E3244: illegal origin value value
```

- sdata 属性セクションにおいて本疑似命令を用いた場合，sdata ／ sbss 属性セクションに割り当てるデータのサイズを指定するための目安を表示する（ld850 の -A オプション）際に，正しい情報が得られなくなります。
- 本疑似命令は，そのセクションに対する指定したファイル内でのロケーション・カウンタ値を進めるだけであり，絶対アドレス[注 1] を指定するものでも，セクション内オフセット[注 2] を指定するものでもありません。

注 1　リンク後のオブジェクト・ファイルにおけるアドレス 0 からのオフセットです。

注 2　リンク後のオブジェクト・ファイルにおいてそのセクションが割り当てられたセクション（出力セクション）の先頭アドレスからのオフセットです。

〔例〕

ロケーション・カウンタ値を 16 バイト進める

```
        .org 16
```

4.5 領域確保

as850 では，領域確保疑似命令を用いることにより，領域の確保，および確保した領域の初期化ができます。次に，この節において説明する領域確保疑似命令を示します。

表 4 － 6　領域確保疑似命令

疑似命令	意味
.byte	1 バイトごとの領域の確保
.float	浮動小数点数値の設定
.hword	1 ハーフワードごとの領域の確保
.lcomm	領域を確保したラベルの定義
.shword	1 ハーフワードごとの領域の確保【V850E】
.space	サイズ分の領域の確保
.str	文字列分の領域の確保
.word	1 ワードごとの領域の確保

なお，sbss ／ bss 属性セクションにおいて .lcomm 疑似命令以外の領域確保疑似命令を指定した場合，次のメッセージが出力され，アセンブルが中止されます。

```
E3246: illegal section
```

また，領域確保疑似命令において指定するサイズ（バイト数），および整列条件の値は，2^{31} 未満にしてください。2^{31} 以上の値を指定した場合，次のメッセージが出力され，アセンブルが中止されます。

```
E3247: illegal size value
       または
E3200: illegal alignment value
```

.byte

〔形式〕

.byte 値 [, 値]...

.byte ビット幅 : 値 [, ビット幅 : 値]...

〔機能〕

- .byte 疑似命令の最初の形式は，各オペランドに対して 1 バイト分の領域を確保し，確保した領域に指定した値の下位 1 バイトの値を格納します。
- .byte 疑似命令の 2 番目の形式は，指定したビット幅の領域を確保し，確保した領域に指定した値を格納します。

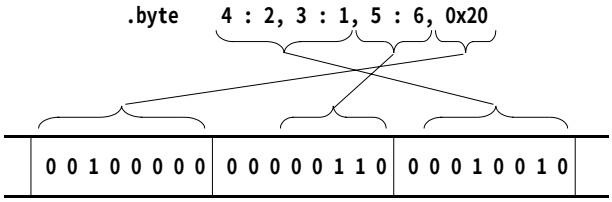
(1) ビット幅は 0 から 8 の範囲で指定します。

(2) バイト幅を越える場合，バイト幅でマスクします。

(3) 最初に指定したビット幅を持つ値は，バイト領域の最下位のビットから割り付けます。その前のビット幅を持つ値を割り付けた領域に続けて領域を割り付けると，その領域がバイト境界を越えてしまう場合，そのビット幅を持つ値は，バイト境界から割り付けます（図 4 － 1 参照）。

(4) ホールが生じた場合，生じたホールを 0 で埋めます。

図 4 － 1 ビット幅を指定した割り付けの例



- 1 つの .byte 疑似命令に，上記の 2 種類の指定を混在させることもできます（図 4 － 1 参照）。

〔例〕

1 バイト分確保し，1 を格納する

```
        .tidata.byte
        .align  4
        .globl  _p, 4
_p:
        .byte  1
```

.float

〔形式〕

.float 値 [, 値]...

〔機能〕

各オペランドに対して1ワード分の領域を確保し,確保した領域に指定した浮動小数点数値を格納します注。

注　整定数を指定した場合,1 ワード分の領域を確保し,確保した領域に指定した整定数の値を格納します。

〔例〕

1 ワード分確保し，1.2345 を格納する

```
        .sidata
        .align 4
        .globl _p, 4
_p:
        .float 1.2345
```

.hword

〔形式〕

.hword 値 [, 値]...

.hword ビット幅 : 値 [, ビット幅 : 値]...

〔機能〕

- .hword 疑似命令の最初の形式は，各オペランドに対して 1 ハーフワード（2 バイト）分の領域を確保し，確保した領域に指定した値の下位 1 ハーフワードの値を格納します。
- .hword 疑似命令の 2 番目の形式は，指定したビット幅の領域を確保し，確保した領域に指定した値を格納します。

(1) ビット幅は 0 から 16 の範囲で指定します。

(2) 値がハーフワード幅を越える場合，ハーフワード幅でマスクします。

(3) 最初に宣言したビット幅を持つ値は，ハーフワード領域の最下位のビットから割り付けます。その前のビット幅を持つ値を割り付けた領域に続けて領域を割り付けるとその領域がハーフワード境界を越えてしまう場合，そのビット幅を持つ値は，ハーフワード境界から割り付けます。

(4) ホールが生じた場合，生じたホールを 0 で埋めます。

- 1 つの .hword 疑似命令に，上記の 2 種類の指定を混在させることもできます。

〔例〕

1 ハーフワード分確保し，100 を格納する

```
        .tidata
        .align 4
        .globl _p, 4
_p:
        .hword 100
```

.lcomm

〔形式〕

.lcomm ラベル名 , サイズ , 整列条件

〔機能〕

前に指定されたセクション定義疑似命令によって指定される現在のセクションに対するロケーション・カウンタ値を第三オペランドで指定した整列条件で整列し，第二オペランドで指定したサイズの領域を確保し，その先頭のアドレスに対して第一オペランドで指定したラベル名を持つローカルなラベル[注]を定義します。

注　ローカル・シンボル（LOCAL のバインディング・クラスを持つシンボル）です。

〔注意事項〕

- 前に指定されたセクション定義疑似命令によって指定されている現在のセクションは，sbss ／ bss 属性セクション（「表 4 － 2　セクションの種類」参照）でなければなりません。これら以外のセクションにおいて本疑似命令を指定した場合，次のメッセージが出力され，アセンブルが中止されます。

```
E3246: illegal section
```

- sbss 属性セクションにおいて 4 より大きな整列条件を指定して本疑似命令を用いた場合，sdata ／ sbss 属性セクションに割り当てるデータのサイズを指定するための目安を表示する（ld850 の -A オプション）際に，正しい情報が得られなくなります。

〔例〕

__stack ラベルのサイズは 0x100 とし，4 バイト整列する

```
        .bss
        .lcomm __stack, 0x100, 4
```

.shword

【V850E】

〔形式〕

.shword 値 [, 値]...

.shword ビット幅 : 値 [, ビット幅 : 値]...

〔機能〕

- .shword 疑似命令の最初の形式は，各オペランドに対して 1 ハーフワード分の領域を確保し，確保した領域に指定した値を右に 1 ビット・シフトして格納します。
- .shword 疑似命令の 2 番目の形式は，指定したビット幅の領域を確保し，確保した領域に指定した値を右に 1 ビット・シフトして格納します。

(1) ビット幅は 0 から 16 の範囲で指定します。

(2) 値がハーフワード幅を越える場合，ハーフワード幅でマスクします。

(3) 最初に宣言したビット幅を持つ値は，ハーフワード領域の最下位のビットから割り付けます。その前のビット幅を持つ値を割り付けた領域に続けて領域を割り付けるとその領域がハーフワード境界を越えてしまう場合，そのビット幅を持つ値は，ハーフワード境界から割り付けます。

(4) ホールが生じた場合，生じたホールを 0 で埋めます。

- 1 つの .shword 疑似命令に，上記の 2 種類の指定を混在させることもできます。
- この疑似命令は，switch 命令用のテーブル作成に適しています。

〔例〕

1 ハーフワード分確保し，10 を右 1 ビット・シフトして格納する

```
        .sdata
        .align 4
        .globl _t, 4
_t:
        .shword 10
```

.space

〔形式〕

.space サイズ [, フィリング値]

〔機能〕

第一オペランドで指定したサイズ分の領域を確保し，確保した領域を第二オペランドで指定したフィリング値（デフォルト値は 0）で埋めます。

- フィリング値を指定する場合，1 バイト分の値を指定します。
  1 バイト分以上の値を指定した場合，下位 1 バイト分を用います。

〔例〕

4 バイトを 0 で埋める

```
        .sidata
        .globl _p, 4
_p:
        .space 4
```

.str

〔形式〕

.str " 文字列定数 " [, " 文字列定数 "]...

〔機能〕

各オペランドに対して，指定された文字列分の領域を確保し，確保した領域に指定した文字列を格納します[注]。

注 C 言語の場合と異なり，文字列の最後にデフォルトで "\0" を入れることはありません。

〔例〕

文字列 "hello" 分の領域を確保し，格納する

```
        .str "hello"
```

.word

〔形式〕

.word 値 [, 値]...

.word ビット幅 : 値 [, ビット幅 : 値]...

〔機能〕

- .word 疑似命令の最初の形式は，各オペランドに対して 1 ワード分の領域を確保し，確保した領域に指定した値を格納します。
- .word 疑似命令の 2 番目の形式は，指定したビット幅の領域を確保し，確保した領域に指定した値を格納します。

(1) ビット幅は 0 から 32 の範囲で指定します。

(2) 幅がワード幅を越える場合，ワード幅でマスクします。

(3) 最初に宣言したビット幅を持つ値は，ワード領域の最下位のビットから割り付けます。その前のビット幅を持つ値を割り付けた領域に続けて領域を割り付けるとその領域がワード境界を越えてしまう場合，そのビット幅を持つ値は，ワード境界から割り付けます。

(4) ホールが生じた場合，生じたホールを 0 で埋めます。

- 1 つの .word 疑似命令に，上記の 2 種類の指定を混在させることもできます。

〔例〕

1 ワード分確保し，0xa で埋める

```
        .sidata
        .align 4
        .globl _p, 4
_p:
        .word 0xa
```

4.6 プログラム・リンケージ疑似命令

as850 では，プログラム・リンケージ疑似命令を用いることにより，指定したサイズ，および整列条件を持つ未定義外部ラベル[注 1] や外部ラベル[注 2] を宣言することができます。

次に，この節において説明するプログラム・リンケージ疑似命令を示します。

注 1 未定義外部シンボル（GLOBAL のバインディング・クラスと GPCOMMON，または COMMON のセクション・ヘッダ・テーブル・インデクスを持つシンボル）です。

注 2 外部シンボル（GLOBAL のバインディング・クラスを持つシンボル）です。

表 4 － 7 プログラム・リンケージ疑似命令

疑似命令	意味
.comm	未定義外部ラベルの宣言
.extern	外部ラベルの宣言
.globl	外部ラベルの宣言

なお，プログラム・リンケージ疑似命令において指定するサイズ（バイト数），および整列条件は，2^{31} 未満にしてください。2^{31} 以上の値を指定した場合，次のメッセージが出力され，アセンブルが中止されます。

```
E3247: illegal size value
       または
E3200: illegal alignment value
```

.comm

〔形式〕

.comm ラベル名 , サイズ , 整列条件

〔機能〕

第一オペランドで指定したラベル名，第二オペランドで指定したサイズ，および第三オペランドで指定した整列条件を持つ未定義外部ラベルを宣言します。

CA850 に含まれるリンカ（ld850）は，未定義外部シンボル（GLOBAL のバインディング・クラスと GPCOMMON，または COMMON のセクション・ヘッダ・テーブル・インデクスを持つシンボル）に対する定義が存在しなかった場合，GPCOMMON のセクション・ヘッダ・テーブル・インデクスを持つ未定義外部シンボルに対しては .sbss セクションに，COMMON のセクション・ヘッダ・テーブル・インデクスを持つ未定義外部シンボルに対しては .bss セクションに，指定された整列条件で整列され指定されたサイズを持つ領域を割り付けます（異なるサイズを持つ未定義外部シンボルが複数存在した場合，ld850 は大きい方のサイズを用います）。定義が存在した場合，その定義の方を優先します。

- 起動時に -G*num* オプションを指定した場合

(1) 指定したサイズが 1 以上 *num* バイト以下の場合

オブジェクト・ファイル生成時のそのラベルに対するシンボル・テーブル・エントリの生成において，GPCOMMON のセクション・ヘッダ・テーブル・インデクスを持つシンボル・テーブル・エントリを生成します。

(2) 指定したサイズが 0 であるか *num* バイトより大きい場合

オブジェクト・ファイル生成時のそのラベルに対するシンボル・テーブル・エントリの生成において，COMMON セクション・ヘッダ・テーブル・インデクスを持つシンボル・テーブル・エントリを生成します。

- 起動時に -G*num* オプションを指定しなかった場合

オブジェクト・ファイル生成時のそのラベルに対するシンボル・テーブル・エントリの生成において，GPCOMMON のセクション・ヘッダ・テーブル・インデクスを持つシンボル・テーブル・エントリを生成します。

〔注意事項〕

- **第一オペランドで指定したラベル名と同名のラベルが，本疑似命令と同じファイル内において通常のラベル定義により定義されている場合**

(a) **そのラベルが本疑似命令により** GPCOMMON **のシンボル・テーブル・エントリ・インデクスを持つよう宣言され** data **属性セクションにおいて通常のラベル定義により定義されている場合，または本疑似命令により** COMMON **のシンボル・テーブル・エントリ・インデクスを持つように宣言され** sdata **属性セクションにおいて通常のラベル定義により定義されている場合**

```
        .comm lab1, 4, 4     -- -G なしでアセンブルすると GPCOMMON
              :
        .data
lab1:                        -- .data セクションで通常のラベル定義
```

次のメッセージが出力され，アセンブルが中止されます。

```
E3213: label identifier redefined
```

(b) **上記以外の場合**

```
        .comm lab1, 4, 4     -- -G なしでアセンブルすると GPCOMMON
              :
        .sdata
lab1：                       -- .sdata セクションで通常のラベル定義
```

通常のラベル定義により定義されたラベルが外部ラベルとみなされ，本疑似命令の指定が無視されます。オブジェクト・ファイル生成時のそのラベルに対するシンボル・テーブル・エントリの生成において， GLOBAL **のバインディング・クラスを持つシンボル・テーブル・エントリが生成されます。**

- **第一オペランドで指定したラベル名と同名のラベルが，本疑似命令と同じファイル内において** .lcomm **疑似命令により定義されている場合**

(a) .lcomm **疑似命令で指定されたサイズ，または整列条件と，本疑似命令において指定されたサイズ，または整列条件が異なる場合**

```
        .comm  lab1, 4, 4
              :
        .sbss
        .lcomm lab1, 4, 2    -- 整列条件が異なる
```

次のメッセージが出力され，アセンブルが中止されます。

```
E3213: label identifier redefined
```

(b) そのラベルが本疑似命令により GPCOMMON のセクション・ヘッダ・テーブル・インデクスを持つよう宣言され bss 属性セクションにおいて .lcomm 疑似命令により定義されている場合，または本疑似命令により COMMON のセクション・ヘッダ・テーブル・インデクスを持つよう宣言され sbss 属性セクションにおいて .lcomm 疑似命令により定義されている場合

```
        .comm lab1, 4, 4     -- -G なしでアセンブルすると GPCOMMON
                :
        .bss
        .lcomm lab1, 4, 4    -- .bss セクションで定義
```

次のメッセージが出力され，アセンブルが中止されます。

```
E3213: label identifier redefined
```

(c) 上記以外の場合

```
        .comm lab1, 4, 4     -- -G なしでアセンブルすると GPCOMMON
                :
        .sbss
        .lcomm lab1, 4, 4    -- .sbss セクションで定義
```

.lcomm により定義されたラベルが外部ラベルとみなされ，本疑似命令の指定が無視されます。オブジェクト・ファイル生成時のそのラベルに対するシンボル・テーブル・エントリの生成において，GLOBAL のバインディング・クラスを持つシンボル・テーブル・エントリが生成されます。

- 第一オペランドで指定したラベル名と同名のラベルが，本疑似命令と同じファイル内において本疑似命令により（再）定義されている場合

(a) サイズ，または境界条件において異なっている場合

```
        .comm lab1, 4, 4
              :
        .comm lab1, 2, 4     -- サイズが異なる
```

次のメッセージが出力され，アセンブルが中止されます。

```
E3213: label identifier redefined
```

(b) サイズ，および境界条件が同じ場合

.comm 疑似命令が 1 回だけ指定されたものと見なされます。

〔例〕

サイズ 4 の未定義外部ラベルを整列条件 4 で宣言する

```
        .sbss
        .comm _p, 4, 4
```

.extern

〔形式〕

.extern ラベル名 [, サイズ]

〔機能〕

第一オペランドで指定したラベル名と同名のラベルを，外部ラベル[注]として宣言します。なお，第二オペランドを指定した場合，指定した値をそのラベルの示すデータのサイズとして指定します。本疑似命令と .globl 疑似命令は外部ラベルを宣言するという機能において変わりませんが，指定したファイル内に定義を持たないラベルを外部ラベルとして宣言する場合，本疑似命令を用い，指定したファイル内に定義を持つラベルを外部ラベルとして宣言する場合，.globl 疑似命令を用いるようにしてください。

注　外部シンボル（GLOBAL のバインディング・クラスを持つシンボル）です。

〔注意事項〕

- as850 では，ラベルは指定したファイル内に定義を持たないことにより，デフォルトで外部ラベルとして宣言されます。このため，第一オペランドで指定したラベル名と同名のラベルが，指定したファイル内に定義を持たない場合，本疑似命令は，実質上，そのラベルの示すデータのサイズを指定するだけの意味になります。
- as850 では，指定したファイル内に定義を持たないデータの gp オフセット参照に対し，「16 ビットのディスプレースメントを用いた参照を行う機械語命令を生成するか」，「（複数の機械語命令によって構成される）32 ビットのディスプレースメントを用いた参照を行う機械語命令列を生成するか」ということを，そのデータのサイズに基づいて判定するため，指定したファイル内に定義を持たず gp オフセット参照されているラベルに対しては，本疑似命令を用いてそのサイズを指定してください。

〔例〕

外部ラベル _main を宣言する（_main はファイル内に定義を持たない）

```
        .extern _main
```

.globl

〔形式〕

.globl ラベル名 [, サイズ]

〔機能〕

第一オペランドで指定したラベル名と同名のラベルを外部ラベル[注]として宣言します。なお，第二オペランドを指定した場合，指定した値をそのラベルの示すデータのサイズとして指定します。本疑似命令と .extern 疑似命令は外部ラベルを宣言するという機能において変わりませんが，指定したファイル内に定義を持つラベルを外部ラベルとして宣言する場合，本疑似命令を用い，指定したファイル内に定義を持たないラベルを外部ラベルとして宣言する場合，.extern 疑似命令を用いるようにしてください。

注　外部シンボル（GLOBAL のバインディング・クラスを持つシンボル）です。

〔注意事項〕

- この宣言により，第一オペランドで指定したラベル名と同名のラベルを定義した場合，そのラベルは他のアセンブリ言語ソース・ファイルから参照できるようになります。
- sdata ／ sbss 属性セクションに割り当てるデータのサイズを指定するための目安を表示する（ld850 の -A オプション）場合，sdata 属性セクションに割り当てるデータ（実際には gp オフセット参照されているラベル）の定義に対しては，本疑似命令，または .size 疑似命令を用いてそのサイズを設定してください[注]。

注　上記の方法で設定しなかった場合，正しい情報が得られません。

〔例〕

外部ラベル _func を宣言する（_func はファイル内に定義を持つ）

```
        .globl  _func
```

4.7 アセンブラ制御疑似命令

as850 では，アセンブラ制御疑似命令を用いることにより，アセンブラが行う処理を制御できます。

次に，この節において説明するアセンブラ制御疑似命令を示します。

表 4 － 8　アセンブラ制御疑似命令

疑似命令	意味
.option	オプションによるアセンブラ制御

.option

〔形式〕

.option オプション

〔機能〕

オペランドに指定したオプションに従ってアセンブラを制御します。指定することのできるオプションを，次に示します[注]。

注　オプションの指定には大文字を用いることもできます（たとえば，nomacro のかわりに NOMACRO を用いることもできます）。

asm

本疑似命令以降の構文エラーに対し，c オプションの指定を解除します。

az_info_j

本疑似命令の直後の命令のアドレスが，AZ850 用のアドレス情報セクション（セクション名：az_info_j）に出力されます。このオプションは，関数呼び出しを行っている命令のアドレスの情報収集を指定するオプションです。

az_info_r

本疑似命令の直後の命令のアドレスが，AZ850 用のアドレス情報セクション（セクション名：az_info_r）に出力されます。このオプションは，関数からの復帰を行っている命令のアドレスの情報収集を指定するオプションです。

az_info_ri

本疑似命令の直後の命令のアドレスが，AZ850 用のアドレス情報セクション（セクション名：az_info_ri）に出力されます。このオプションは，割り込み関数からの復帰を行っている命令のアドレスの情報収集を指定するオプションです。

c *linenum* ["*filename*"]

本疑似命令以降の構文エラーに対し，エラー・メッセージの行数（*linenum*），ファイル名（*filename*）を，指定されたものに置き換えて出力します。アセンブリ言語ソース・ファイル内の二度目以降の本疑似命令の "*filename*" は省略可能です。省略した場合，直前に本疑似命令に指定したファイル名が指定されたものとして処理をします。この場合，直前の本疑似命令との間の asm オプションの有無は問いません。

アセンブリ言語ソース・ファイル内の一度目の "*filename*" を省略した場合，次のメッセージを出力し，アセンブルを中止します。

```
E3249: illegal syntax
```

callt

コンパイラ予約の疑似命令です。

注意 コンパイラが出力したアセンブリ言語ソース・ファイル中に存在する場合は，削除しないでください。削除した場合，プロローグ／エピローグ・ランタイムのリンクを確認する機能が動作しません。

cpu *devicename*

devicename で指定されたターゲット・デバイスのデバイス・ファイルを読み込みます。デバイス・ファイルを読み込むためのデバイス名指定は，as850 起動時に，-cpu オプションによっても指定できます。デバイス名指定は，オブジェクト・ファイルを作成する際に，必ず行います。-cpu オプションによる指定も本疑似命令による指定もない場合，次のエラー・メッセージが出力され，処理が中止されます。

```
F3522: unknown cpu type
```

-cpu オプション，疑似命令の両方で指定した場合，警告メッセージが出力され，オプション指定を優先します。ただし，オプション指定，または疑似命令で，複数の異なるデバイス名を指定した場合，次のエラー・メッセージが出力され，処理が中止されます。

```
F3523: duplicated cpu type
```

例

使用するデバイスとして V850ES/SA2 を指定する

```
        .option cpu 3201
```

なお，使用するデバイス・ファイルは，デバイス・ファイルの標準ディレクトリ，またはデバイス・ファイルのあるディレクトリを，as850 の “ -F オプション ” で指定するようにしてください。

data *extern_symbol*

シンボル名 *extern_symbol* の外部データが，ca850 や as850 の -G オプションで指定されたサイズ値に関わらず，data 属性，または bss 属性セクションに割り当てられているとみなされ，そのデータを参照する命令に対して命令展開が行われます。この形式は，# pragma section やセクション・ファイルで “ data ” を指定した変数を，アセンブリ言語ソース・ファイルで外部参照する場合に利用します。

例

_d はオプションに関係なく .data セクションとなり，参照時に命令展開される

```
        .option data _d
        .text
        mov     $_d, r11
```

ep_label

それ以降の命令に対し，%label によるラベル参照がすべて ep オフセット参照として扱われます。

macro

それ以降の命令に対し，nomacro オプションの指定が解除されます。

mask_reg

as850 が生成するリロケータブルなオブジェクト・ファイル中に，マスク・レジスタ機能を使用していることを示す情報が埋めこまれます。このオプションは，たとえば，マスク・レジスタ機能をサポートしていない旧バージョンの C コンパイラ（CA850 Ver.1.00）が出力したアセンブリ言語ソース・ファイルを，マスク・レジスタ機能指定として利用する場合に有効です。このオプションを指定することにより，マスク・レジスタ機能を使用しているとみなされ，マスク・レジスタ指定でコンパイルして作成したオブジェクトのリンク時にエラーとなりません。

注意 マスク・レジスタ機能では，マスク・レジスタとして r20，および r21 を C コンパイラが使用します。これらのレジスタに設定したマスク値を，アセンブリ言語ソース・プログラムにより変更しないようにしてください。

new_fcall

as850 が生成するリロケータブルなオブジェクト・ファイル中に，新呼び出し仕様であることを示す情報が埋めこまれます。このオプションは，たとえば，呼び出し仕様の異なる旧版の C コンパイラ（CA850 Ver.1.xx）が出力したアセンブリ言語ソース・ファイルを，現版の C コンパイラにより作成されたオブジェクトとともに利用する場合に有効です。このオプションを指定することにより，新呼び出し仕様を満たしているとみなされ，C コンパイラのデフォルトの新呼び出し仕様で作成されたオブジェクトとのリンク時に，エラーとなりません。

no_ep_label

それ以降の命令に対し，ep_label オプションの指定が解除されます。

nomacro

それ以降の setf*cond*，sasf*cond* 【V850E】，cmov*cond*【V850E】，jcond，および jmp 命令を除く命令に対し，命令展開が行われません。

nooptimize

それ以降の命令に対し，命令並べ換え最適化が行われません。

novolatile

それ以降の命令に対し，nooptimize ／ volatile オプションの指定が解除されます。

nowarning

それ以降の命令に対し，警告メッセージが出力されません。

optimize

novolatile オプションと同様です。

reg_mode *tnum pnum*

as850 が生成するリロケータブルなオブジェクト・ファイル中に，レジスタ・モード情報セクションが埋め込まれます。レジスタ・モード情報セクションとは，コンパイラが使用する作業用レジスタとレジスタ変数用レジスタの本数情報を保持するものです。この命令により，作業用レジスタとレジスタ変数用レジスタの本数が，*tnum*，*pnum* 本として設定されます。なお，22 レジスタ・モードを使用する場合，*tnum*，*pnum* は 5 本ずつとなり，26 レジスタ・モードを使用する場合，*tnum*，*pnum* は 7 本ずつとなります。

例

レジスタ・モード 22 **を用いる**

```
        .option reg_mode 5 5
```

sdata *extern_symbol*

シンボル名 *extern_symbo*l **の外部データが，**ca850 **や** as850 **の** -G **オプションで指定されたサイズ値に関わらず，**sdata **属性，または** sbss **属性セクションに割り当てられているとみなされ，そのデータを参照する命令に対して命令展開が行われません。この形式は，**# pragma section **やセクション・ファイルで “** sdata **” が指定された変数を，アセンブリ言語ソース・ファイルで外部参照する場合に利用します。**

例

_d **はオプションに関係なく** .sdata **セクションとなり，参照時に命令展開されない**

```
        .option sdata _d
        .text
        mov    $_d, r11
```

volatile

nooptimize **オプションと同様です。**

warning

それ以降の命令に対し，警告メッセージが出力されます。

4.8 ファイル入力制御疑似命令

as850 では，ファイル入力制御疑似命令を用いることにより，アセンブリ言語ソース・ファイル，またはバイナリ・ファイルを，指定した位置に取り込むことができます。

次に，この節において説明するファイル入力制御疑似命令を示します。

表４－９ ファイル入力制御疑似命令

疑似命令	意味
.binclude	バイナリ・ファイルの入力
.include	アセンブリ言語ソース・ファイルの入力

.binclude

〔形式〕

.binclude " ファイル名 "

〔機能〕

オペランドに指定したバイナリ・ファイルの内容を，本疑似命令の置かれている位置に置かれたソース・プログラムのアセンブル結果であるとみなして扱います。指定したファイルは，本疑似命令を含むソース・ファイルの置かれているディレクトリで検索されます。また，ファイル名には，ソース・ファイルの置かれているディレクトリからの相対パスによる記述も可能です。アセンブラのオプション（-I）でディレクトリを指定した場合，指定したディレクトリが先に検索されます。ソース・ファイルの置かれているディレクトリにファイルが存在しない場合，C 言語ソース・ファイルの置かれているディレクトリ（.file 疑似命令より導出），カレント・ディレクトリを検索します。

〔注意事項〕

- 本疑似命令は，バイナリ・ファイルの内容全体を扱います。リロケータブル・ファイルを指定した場合には，ELF フォーマットで構成されたファイル全体を扱います。.text セクション等の内容のみを扱うということではありません。
- 指定するファイル名は，" " で囲んでください。
- 存在しないファイルを指定した場合，次のメッセージが出力され，アセンブルが中止されます。

```
F3503: can not open file file
```

〔例〕

aa.bin ファイルをインクルードする

```
        .binclude "aa.bin"
```

.include

〔形式〕

.include " ファイル名 "

〔機能〕

オペランドに指定したファイルの内容を，本疑似命令の置かれている位置に置かれているものとして扱います。指定したファイルは，本疑似命令を含むソース・ファイルの置かれているディレクトリで検索されます。また，ファイル名には，ソース・ファイルの置かれているディレクトリからの相対パスによる記述もできます。アセンブラのオプション（-I）でディレクトリを指定した場合，指定したディレクトリが先に検索されます。ソース・ファイルの置かれているディレクトリにファイルが存在しない場合，C 言語ソース・ファイルの置かれているディレクトリ（.file 疑似命令より導出），カレント・ディレクトリを検索します。

〔注意事項〕

- 指定するファイル名は，" で囲んでください。
- 存在しないファイルを指定した場合，次のメッセージが出力され，アセンブルが中止されます。

```
F3503: can not open file file
```

- .include 文が 9 回以上ネストして用いられた場合,次のメッセージが出力され,アセンブルが中止されます。

```
F3517: include nest over
```

〔例〕

aa.s ファイルをインクルードする

```
        .include "aa.s"
```

4.9 繰り返しアセンブル疑似命令

as850 では，繰り返しアセンブル疑似命令とそれに対応する .endm 疑似命令とで囲まれた文の並び（ブロック）を，繰り返しアセンブル疑似命令の置かれている位置で，繰り返してアセンブルができます。

次に，この節において説明する繰り返しアセンブル疑似命令を示します。

表 4 － 10 繰り返しアセンブル疑似命令

疑似命令	意味
.irepeat	パラメータの指定による繰り返し
.repeat	回数の指定による繰り返し

.irepeat

〔形式〕

.irepeat 仮パラメータ 実パラメータ [, 実パラメータ]...

〔機能〕

本疑似命令と，本疑似命令に対応する .endm 疑似命令で囲まれる文の並び（ブロック）を，そのブロック内に現れた第一オペランドで指定した仮パラメータを第二オペランド以降に指定した実パラメータに置き換え，繰り返しアセンブルします。第二オペランド以降に指定した実パラメータがすべて置き換えに用いられた場合，繰り返しを中止します。

〔注意事項〕

- .irepeat と .endm は対応させてください。対応する .endm が存在しない場合，次のメッセージが出力され，アセンブルが中止されます。

```
F3513: unexpected EOF in .repeat/.irepeat
```

- 33 個以上の実パラメータを指定した場合，次のメッセージが出力され，アセンブルが中止されます。

```
F3514: paramater table overflow
```

- 仮パラメータと実パラメータに同じパラメータ名を指定した場合，次のメッセージが出力され，アセンブルが中止されます。

```
F3238: illegal operand (.irepeat parameter)
```

- 仮パラメータと実パラメータにラベルや他の疑似命令で定義済みのパラメータ名を指定した場合，次のメッセージが出力され，アセンブルが中止されます。

```
F3238: illegal operand (.irepeat parameter)
```

〔例〕

```
.irepeat x a, b, c, d
       .word x
.endm
```

展開すると次のようになります。

```
       .word a
       .word b
       .word c
       .word d
```

.repeat

〔形式〕

.repeat 絶対値式

〔機能〕

本疑似命令と本疑似命令に対応する .endm 疑似命令で囲まれている文の並び（ブロック）を，第一オペランドで指定した絶対値式で与えられた回数だけ繰り返してアセンブルします。

〔注意事項〕

- .repeat と .endm は対応させてください。対応する .endm が存在しない場合，次のメッセージが出力され，アセンブルが中止されます。

```
F3513: unexpected EOF in .repeat/.irepeat
```

- 値は，32 ビットの符号付き整数として評価されます。
- 文の並び（ブロック）がない場合，何も行いません。
- 式の評価結果が負になった場合，次のメッセージが出力され，アセンブルが続行されます。

```
E3225: illegal operand (must be evaluated positive or zero)
```

〔例〕

```
.repeat 2
        nop
.endm
```

展開すると次のようになります。

```
        nop
        nop
```

4.10 条件アセンブル疑似命令

as850 では，条件アセンブル疑似命令を用いることにより，条件式の評価結果に従って，アセンブルを行う範囲が制御できます。

次に，この節において説明する条件アセンブル疑似命令を示します。

表 4 － 11 条件アセンブル疑似命令

疑似命令	意味
.else	絶対値式／シンボルによる制御
.elseif	絶対値式による制御（真のときアセンブル）
.elseifn	絶対値式による制御（偽のときアセンブル）
.endif	制御範囲の終わり
.if	絶対値式による制御（真のときアセンブル）
.ifdef	シンボルによる制御（定義されているときアセンブル）
.ifn	絶対値式による制御（偽のときアセンブル）
.ifndef	シンボルによる制御（定義されていないときアセンブル）

条件アセンブル疑似命令が 17 回以上ネストして用いられた場合，次のメッセージが出力され，アセンブルが中止されます。

```
F3512: .if, .ifn, etc. too deeply nested
```

.else

〔形式〕

.else

〔機能〕

.if 疑似命令，.elseif 疑似命令，または .ifdef 疑似命令において絶対値式が偽（= 0）に評価された場合，あるいは本疑似命令に対応する .ifn 疑似命令，.elseifn 疑似命令，または .ifndef 疑似命令において絶対値式が真（≠ 0）に評価された場合，本疑似命令と本疑似命令に対応する .endif 疑似命令とで囲まれる文の並び（ブロック）をアセンブルします。

〔注意事項〕

- 本疑似命令に対応する .if 疑似命令，.ifn 疑似命令，.elseif 疑似命令，.elseifn 疑似命令，.ifdef 疑似命令，.ifndef 疑似命令が存在しない場合は，次のメッセージが出力され，アセンブルが中止されます。

```
F3510: .else unexpected
```

〔例〕

```
.if 0
        .word 10
.else
        .str "a"
.endif

.if 10 > 20
        .word 20
.else
        .str "b"
.endif

.set expr, 0
.if expr
        .word expr
.else
        .str "c"
.endif
```

展開すると次のようになります。

```
        .str "a"
        .str "b"
        .str "c"
```

.elseif

〔形式〕

.elseif 絶対値式

〔機能〕

- オペランドで指定した絶対値式が真（≠ 0）に評価された場合

(1) 本疑似命令と本疑似命令に対応する .else 疑似命令，.elseif 疑似命令，または .elseifn 疑似命令が存在する場合は，本疑似命令とその疑似命令とで囲まれるブロックをアセンブルします。

(2) それらの疑似命令が存在しない場合は，本疑似命令とその疑似命令に対応する .endif 疑似命令とで囲まれるブロックをアセンブルします。

- 偽（= 0）に評価された場合

本疑似命令に対応する .else 疑似命令，.elseif 疑似命令，.elseifn 疑似命令，または .endif 疑似命令までスキップします。

〔注意事項〕

- 対応する疑似命令が存在しない場合，次のメッセージが出力され，アセンブルが中止されます。

```
F3511: .endif unmatched
```

〔例〕

```
.if 0
       .word 10
.elseif 10
       .str "a"
.endif

.if 10 > 20
       .word 20
.elseif 10 == 20
       .str "b"
.endif

.set expr, 0
.if expr
       .word expr
.elseifn expr - 10
       .str "c"
.endif
```

展開すると次のようになります。

```
       .str "a"
```

.elseifn

〔形式〕

.elseifn 絶対値式

〔機能〕

- オペランドで指定した絶対値式が真（≠ 0）に評価された場合

  本疑似命令に対応する .else 疑似命令，.elseif 疑似命令，.elseifn 疑似命令，または .endif 疑似命令までスキップします。

- 偽（= 0）に評価された場合

(1) 本疑似命令と本疑似命令に対応する .else 疑似命令，.elseif 疑似命令，または .elseifn 疑似命令が存在する場合は，本疑似命令とその疑似命令とで囲まれるブロックをアセンブルします。

(2) それらの疑似命令が存在しない場合は，本疑似命令とその疑似命令に対応する .endif 疑似命令とで囲まれるブロックをアセンブルします。

〔注意事項〕

- 対応する疑似命令が存在しない場合，次のメッセージが出力され，アセンブルが中止されます。

```
F3511: .endif unmatched
```

〔例〕

```
.if 0
       .word 10
.elseifn 10
       .str "a"
.endif

.if 10 > 20
       .word 20
.elseifn 10 >= 20
       .str "b"
.endif

.set expr, 0
.if expr
       .word expr
.elseif expr - 10
       .str "c"
.endif
```

展開すると次のようになります。

```
       .str "b"
       .str "c"
```

.endif

〔形式〕

.endif

〔機能〕

条件アセンブル疑似命令による制御の範囲の終わりを示します。

〔注意事項〕

- 本疑似命令に対応する .if 疑似命令，.ifn 疑似命令，.elseif 疑似命令，.elseifn 疑似命令，.ifdef 疑似命令，.ifndef 疑似命令が存在しない場合，次のメッセージが出力され，アセンブルが中止されます。

```
F3510: .endif unexpected
```

.if

〔形式〕

.if 絶対値式

〔機能〕

- オペランドで指定した絶対値式が真（≠ 0）に評価された場合

(1) 本疑似命令と本疑似命令に対応する .else 疑似命令，.elseif 疑似命令，または .elseifn 疑似命令が存在する場合は，本疑似命令とその疑似命令とで囲まれるブロックをアセンブルします。

(2) それらの疑似命令が存在しない場合は，本疑似命令と本疑似命令に対応する .endif 疑似命令とで囲まれるブロックをアセンブルします。

- 偽（= 0）に評価された場合

本疑似命令に対応する .else 疑似命令，.elseif 疑似命令，.elseifn 疑似命令，または .endif 疑似命令までスキップします。

〔注意事項〕

- オペランドに未定義なシンボルを指定した場合，次のメッセージが出力され，アセンブルが中止されます。

```
E3202: illegal expression
```

- 対応する疑似命令が存在しない場合，次のメッセージが出力され，アセンブルが中止されます。

```
F3511: .endif unmatched
```

〔例〕

```
.if 10
       .word 10
.endif

.if 10 < 20
       .word 20
.endif

.set expr, 30
.if expr
       .word expr
.endif
```

展開すると次のようになります。

```
       .word 10
       .word 20
       .word 30
```

.ifdef

〔形式〕

.ifdef **名前**

〔機能〕

- **オペランドで指定した名前が定義されている場合**

(1) **本疑似命令と本疑似命令に対応する** .else **疑似命令，**.elseif **疑似命令，または** .elseifn **疑似命令が存在する場合は，本疑似命令とその疑似命令とで囲まれるブロックをアセンブルします。**

(2) **それらの疑似命令が存在しない場合は，本疑似命令とその疑似命令に対応する** .endif **疑似命令とで囲まれるブロックをアセンブルします。**

- **指定した名前が定義されていない場合**

本疑似命令に対応する .else **疑似命令，**.elseif **疑似命令，**.elseifn **疑似命令，または** .endif **疑似命令までスキップします。**

〔注意事項〕

- **名前には，シンボル，ラベル，マクロ名を指定できますが，予約語は指定できません。予約語を指定した場合，次のメッセージが出力され，アセンブルが中止されます。**

```
E3220: illegal operand ( identifier is reserved word)
```

- **対応する疑似命令が存在しない場合，次のメッセージが出力され，アセンブルが中止されます。**

```
F3511: .endif unmatched
```

〔例〕

```
define_symbol:
       .ifdef define_symbol
              .word 10
       .endif

       .ifdef undef_symbol
              .word 20
       .else
              .ifde define_symbol
                     .str "x"
              .endif
       .endif

       .set expr, 20
       .ifdef expr
              .word expr
       .endif
```

展開すると次のようになります。

```
       .word 10
       .str "x"
       .word 20
```

.ifn

〔形式〕

.ifn 絶対値式

〔機能〕

- オペランドで指定した絶対値式が真（≠ 0）に評価された場合

  本疑似命令に対応する .else 疑似命令，.elseif 疑似命令，.elseifn 疑似命令，または .endif 疑似命令までスキップします。

- 偽（= 0）に評価された場合

(1) 本疑似命令と本疑似命令に対応する .else 疑似命令，.elseif 疑似命令，または .elseifn 疑似命令が存在する場合は，本疑似命令とその疑似命令とで囲まれるブロックをアセンブルします。

(2) それらの疑似命令が存在しない場合は，本疑似命令と本疑似命令に対応する .endif 疑似命令とで囲まれるブロックをアセンブルします。

〔注意事項〕

- 対応する疑似命令が存在しない場合，次のメッセージが出力され，アセンブルが中止されます。

```
F3511: .endif unmatched
```

〔例〕

```
.ifn 0
       .word 10
.endif

.ifn 10 > 20
       .word 20
.endif

.set expr, 0
.ifn expr
       .word expr
.endif
```

展開すると次のようになります。

```
       .word 10
       .word 20
       .word 0
```

.ifndef

〔形式〕

.ifndef 名前

〔機能〕

- オペランドで指定した名前が定義されている場合

  本疑似命令に対応する .else 疑似命令，.elseif 疑似命令，.elseifn 疑似命令，または .endif 疑似命令までスキップします。

- 指定した名前が定義されていない場合

(1) 本疑似命令と本疑似命令に対応する .else 疑似命令，.elseif 疑似命令，または .elseifn 疑似命令が存在する場合は，本疑似命令とその疑似命令とで囲まれるブロックをアセンブルします。

(2) それらの疑似命令が存在しない場合は，本疑似命令と本疑似命令に対応する .endif 疑似命令とで囲まれるブロックをアセンブルします。

〔注意事項〕

- 名前には，シンボル，ラベル，マクロ名を指定できますが，予約語は指定できません。予約語を指定した場合，次のメッセージが出力され，アセンブルが中止されます。

```
E3220: illegal operand ( identifier is reserved word)
```

- 対応する疑似命令が存在しない場合，次のメッセージが出力され，アセンブルが中止されます。

```
F3511: .endif unmatched
```

〔例〕

```
define_symbol:
       .ifndef define_symbol
              .word 10
       .else
              .str "a"
       .endif

       .ifndef undef_symbol
              .word 20
       .else
              .ifndef define_symbol
                     .str "x"
              .endif
       .endif

       .set expr, 20
       .ifndef expr
              .word expr
       .endif
```

展開すると次のようになります。

```
       .str "a"
       .word 20
```

4.11 スキップ疑似命令

as850 では，スキップ疑似命令を用いることにより，繰り返しアセンブル疑似命令において残りの繰り返しをスキップすることができます。

次に，この節において説明するスキップ疑似命令を示します。

表 4 － 12 スキップ疑似命令

疑似命令	意味
.exitm	1 つ外側にスキップ
.exitma	一番外側にスキップ

.exitm

〔形式〕

.exitm

〔機能〕

本疑似命令は，本疑似命令を囲んでいる最も内側の繰り返しアセンブル疑似命令の繰り返しアセンブルをスキップします。

〔注意事項〕

- 本疑似命令が繰り返しアセンブル疑似命令に囲まれていない場合，次のメッセージが出力され，アセンブルが中止されます。

```
F3515: .exitm not in .repeat/.irepeat
```

〔例〕

```
.repeat 2
       .set expr, 1
       .word 10
       .repeat 10
              .if expr < 5
                     .byte expr
                     .set expr, expr + 1
              .else
                     .ifdef undefine_symbol
                            .byte expr
                            .set expr, expr + 1
                     .else
                            .exitm
                     .endif
              .endif
       .endm
       .hword 20
       .hword 30
.endm
.word expr
```

展開すると次のようになります。

```
       .word 10
       .byte 1
       .byte 2
       .byte 3
       .byte 4
       .hword 20
       .hword 30
       .word 10
       .byte 1
       .byte 2
       .byte 3
       .byte 4
       .hword 20
       .hword 30
       .word 5
```

.exitma

〔形式〕

.exitma

〔機能〕

本疑似命令は，本疑似命令を囲んでいる最も外側の繰り返しアセンブル疑似命令の繰り返しをスキップします。

〔注意事項〕

- 本疑似命令が繰り返しアセンブル疑似命令に囲まれていない場合，次のメッセージが出力され，アセンブルが中止されます。

```
F3515: .exitma not in .repeat/.irepeat
```

〔例〕

```
.repeat 2
       .set expr, 1
       .word 10
       .repeat 10
              .if expr < 5
                     .byte expr
                     .set expr, expr + 1
              .else
                     .ifdef undefine_symbol
                            .byte expr
                            .set expr, expr + 1
                     .else
                            .exitma
                     .endif
              .endif
       .endm
       .hword 20
       .hword 30
.endm
.word expr
```

展開すると次のようになります。

```
       .word 10
       .byte 1
       .byte 2
       .byte 3
       .byte 4
       .word 5
```

4.12 マクロ疑似命令

as850 では，マクロ疑似命令を用いることにより，任意の文の並びを指定したマクロ名に対応するマクロ本体として定義しできます。

また，ソース・プログラム内でこのマクロ名を参照することにより，その位置にこのマクロ名に対応する文の並びを記述したものとして扱うことができます。

次に，この節において説明するマクロ疑似命令を示します。

表 4 － 13 マクロ疑似命令

疑似命令	意味
.endm	繰り返しの区間の終わり，またはマクロ定義の終わり
.local	ローカル・シンボルの定義
.macro	マクロ定義の始まり

.endm

〔形式〕

.endm

〔機能〕

繰り返しの区間は終わり，またはマクロ本体の終わりを示します。

〔注意事項〕

- **本疑似命令に対応する** .repeat **疑似命令，**.irepeat **疑似命令，**.macro **疑似命令が存在しない場合，次のメッセージが出力され，アセンブルが中止されます。**

```
F3510: .endm unexpected
```

.local

〔形式〕

.local **ローカル・シンボル** [, **ローカル・シンボル**]...

〔機能〕

指定した文字列を特有の識別子として置き換えられるローカル・シンボルとして宣言します。

〔注意事項〕

- **本疑似命令の仮パラメータに** 33 **個以上のローカル・シンボルを指定した場合，次のメッセージが出力され，アセンブルが中止されます。**

```
F3514: paramater table overflow
```

- **アセンブラによって生成されるローカル・シンボル名は，**.??0000 **から** .??FFFF **までの範囲で生成されます。**

〔例〕

```
.macro m1 x
       .local a, b
       a: .word a
       b: .word x
.endm

m1 10
m1 20
```

展開すると次のようになります。

```
.??0000: .word .??0000
.??0001: .word 10
.??0002: .word .??0002
.??0003: .word 20
```

.macro

〔形式〕

.macro マクロ名 [仮パラメータ ,]...

〔機能〕

本疑似命令と .endm 疑似命令とで囲まれた文の並びを，第一オペランドで指定したマクロ名に対するマクロ本体として定義します。このマクロ名が参照された場合，その位置にこのマクロ名に対応するマクロ本体が記述されたものとして扱います。

〔注意事項〕

- 本疑似命令に対応する .endm 疑似命令が存在しない場合，次のメッセージが出力され，アセンブルが中止されます。

```
F3513: unexpected EOF in .macro
```

- あるマクロ名が再定義された場合，それ以降のそのマクロ呼び出しに対しては，再定義されたマクロ本体がそのマクロ名に対応するマクロ本体となります。
- 33 個以上の仮パラメータを指定した場合，次のメッセージが出力され，アセンブルが中止されます。

```
F3514: paramater table overflow
```

- マクロ本体内で参照されていない余分な仮パラメータは無視されます。
  この場合，as850 では，メッセージが出力されないので注意が必要です。
- マクロ呼び出しにおいて実パラメータが足りない場合，次のメッセージが出力され，アセンブルが中止されます。

```
F3519: argument mismatch
```

- マクロ本体内で，未定義なマクロの呼び出しが行われた場合，次のメッセージが出力され，アセンブルが中止されます。

```
E3249: illegal syntax
```

- マクロ本体内で，現在定義中のマクロの呼び出しが行われた場合，次のメッセージが出力され，アセンブルが中止されます。

```
F3518: unreasonable macro_call nesting
```

- 仮パラメータにラベルや疑似命令で定義済みのパラメータを指定した場合，次のメッセージが出力され，アセンブルが中止されます。

```
E3212: symbol already defined as string
```

- マクロ呼び出しにおいて実パラメータに指定可能なものは，ラベル名，シンボル名，数値，レジスタ，および命令ニモニックのみです。
  ラベル式（LABEL-1），参照方法指定ラベル（#LABEL），またはベース・レジスタ指定（[gp]）などを指定した場合，指定された実パラメータに依存したメッセージが出力され，アセンブルが中止されます。

〔例〕

```
.macro PUSH REG
       add    -4, sp
       st.w   REG, 0x0[sp]
.endm

.macro POP REG
       ld.w   0x0[sp],REG
       add    0x4, sp
.endm
       PUSH   r10
       mov    10, r10
       add    r10, r20
       POP    r10
```

展開すると次のようになります。

```
       add    -4, sp
       st.w   r10, 0x0[sp]
       mov    10, r10
       add    r10, r20
       ld.w   0x0[sp], r10
       add    0x4, sp
```

付録 A　命令一覧

次に，CA850 アセンブラ（as850）がサポートする命令一覧，および疑似命令一覧を，アルファベット順に示します。

表 A － 1 命令一覧

命令ニモニック	意味
add	加算
addi	加算（イミーディエト）
adf	条件付き加算【V850E2】
and	論理積
andi	論理積（イミーディエト）
bsh	バイト・スワップ・ハーフワード【V850E】
bsw	バイト・スワップ・ワード【V850E】
callt	テーブル参照コール【V850E】
clr1	ビット・クリア
cmov	フラグ条件付きデータの転送【V850E】
cmp	比較
ctret	callt からの復帰【V850E】
dbret	デバッグ・トラップからの復帰【V850E】
dbtrap	デバッグ・トラップ【V850E】
di	マスカブル割り込みの禁止
dispose	スタック・フレームの削除（関数の後処理）【V850E】
div	符号付き除算（ワード）【V850E】
divh	符号付き除算（ハーフワード）
divhu	符号なし除算（ハーフワード）【V850E】
divu	符号なし除算（ワード）【V850E】
ei	マスカブル割り込みの許可
halt	プロセッサの停止
hsh	ハーフワード・データのハーフワード・スワップ【V850E2】
hsw	ハーフワード・スワップ・ワード【V850E】
jarl	ジャンプ・アンド・レジスタ・リンク
jarl22	ジャンプ・アンド・レジスタ・リンク【V850E2】
jarl32	ジャンプ・アンド・レジスタ・リンク【V850E2】

表 A － 1 命令一覧

命令ニモニック	意味
jcond	条件分岐
jmp	無条件分岐
jmp32	無条件分岐【V850E2】
jr	無条件分岐（PC 相対）
jr22	無条件分岐（PC 相対）【V850E2】
jr32	無条件分岐（PC 相対）【V850E2】
ld.b	ロード（バイト）
ld.bu	ロード（符号なしバイト）【V850E】
ld.h	ロード（ハーフワード）
ld.hu	ロード（符号なしハーフワード）【V850E】
ld.w	ロード（ワード）
ldsr	システム・レジスタへのロード
mac	符号付きワード・データの加算付き乗算【V850E2】
macu	符号なしワード・データの加算付き乗算【V850E2】
mov	データの転送
mov32	32 ビット・データの転送【V850E】
movea	実効アドレスの転送
movhi	上位ハーフワードの転送
mul	符号付き乗算（ワード）【V850E】
mulh	符号付き乗算（ハーフワード）
mulhi	符号付き乗算（イミーディエト）
mulu	符号なし乗算（ワード）【V850E】
nop	ノー・オペレーション
not	論理否定（1 の補数をとる）
not1	ビット・ノット
or	論理和
ori	論理和（イミーディエト）
pop	スタック領域からのポップ（単一レジスタ）
popm	スタック領域からのポップ（複数レジスタ）
prepare	スタック・フレームの生成（関数の前処理）【V850E】
push	スタック領域へのプッシュ（単一レジスタ）
pushm	スタック領域へのプッシュ（複数レジスタ）

表 A － 1 命令一覧

命令ニモニック	意味
reti	トラップ，または割り込みルーチンからの復帰
sar	算術右シフト
sasf	論理左シフト付きフラグ条件の設定【V850E】
satadd	飽和加算
satsub	飽和減算
satsubi	飽和減算（イミーディエト）
satsubr	飽和逆減算
sch0l	MSB 側からのビット（0）検索【V850E2】
sch0r	LSB 側からのビット（0）検索【V850E2】
sch1l	MSB 側からのビット（1）検索【V850E2】
sch1r	LSB 側からのビット（1）検索【V850E2】
sbf	条件付き減算【V850E2】
set1	ビット・セット
setf	フラグ条件の設定
shl	論理左シフト
shr	論理右シフト
sld.b	バイト・データのロード（ショート・フォーマット）
sld.bu	符号なしバイト・データのロード（ショート・フォーマット）【V850E】
sld.h	ハーフワード・データのロード（ショート・フォーマット）
sld.hu	符号なしハードワード・データのロード（ショート・フォーマット）【V850E】
sld.w	ワード・データのロード（ショート・フォーマット）
sst.b	バイト・データのストア（ショート・フォーマット）
sst.h	ハーフワード・データのストア（ショート・フォーマット）
sst.w	ワード・データのストア（ショート・フォーマット）
st.b	バイト・データのストア
st.h	ハーフワード・データのストア
st.w	ワード・データのストア
stsr	システム・レジスタの内容のストア
sub	減算
subr	逆減算
switch	テーブル参照ジャンプ【V850E】
sxb	符号拡張バイト【V850E】

表 A － 1 命令一覧

命令ニモニック	意味
sxh	符号拡張ハーフワード【V850E】
trap	ソフトウエア・トラップ
tst	テスト
tst1	ビット・テスト
xor	排他的論理和
xori	排他的論理和（イミーディエト）
zxb	ゼロ拡張バイト【V850E】
zxh	ゼロ拡張ハーフワード【V850E】

表 A － 2 疑似命令一覧

疑似命令	意味
.align	ロケーション・カウンタの値の整列
.binclude	バイナリ・ファイルの入力
.bss	.bss セクションへの割り付け
.byte	1 バイトごとの領域の確保
.comm	未定義外部ラベルの宣言
.const	.const セクションへの割り付け
.data	.data セクションへの割り付け
.else	絶対値式／シンボルによる制御
.elseif	絶対値式による制御（真のときアセンブル）
.elseifn	絶対値式による制御（偽のときアセンブル）
.endif	制御範囲の終わり
.endm	繰り返しの区間の終わり，またはマクロ定義の終わり
.exitm	1 つ外側にスキップ
.exitma	一番外側にスキップ
.extern	外部ラベルの宣言
.ext_ent_size	フラッシュ・テーブル・エントリのサイズ指定
.ext_func	フラッシュ・テーブル・エントリの生成
.file	シンボル・テーブル・エントリの生成（FILE タイプ）
.float	浮動小数点数値の設定
.frame	シンボル・テーブル・エントリの生成（FUNC タイプ）
.globl	外部ラベルの宣言
.hword	1 ハーフワードごとの領域の確保
.if	絶対値式による制御（真のときアセンブル）
.ifdef	シンボルによる制御（定義されているときアセンブル）
.ifn	絶対値式による制御（偽のときアセンブル）
.ifndef	シンボルによる制御（定義されていないときアセンブル）
.include	アセンブリ言語ソース・ファイルの入力
.irepeat	パラメータの指定による繰り返し
.lcomm	領域を確保したラベルの定義
.local	ローカル・シンボルの定義
.macro	マクロ定義の始まり
.option	オプションによるアセンブラ制御

表 A － 2 疑似命令一覧

疑似命令	意味
.org	ロケーション・カウンタの値を進める
.previous	現在のセクション定義疑似命令を指定しているセクション定義疑似命令の前のセクション定義疑似命令の（再）指定
.repeat	回数の指定による繰り返し
.sbss	.sbss セクションへの割り付け
.sconst	.sconst セクションへの割り付け
.sdata	.sdata セクションへの割り付け
.sebss	.sebss セクションへの割り付け
.section	指定した種類のセクションへの割り付け
.sedata	.sedata セクションへの割り付け
.set	シンボルの定義
.shword	1 ハーフワードごとの領域の確保（switch 命令用）【V850E】
.sibss	.sibss セクションへの割り付け
.sidata	.sidata セクションへの割り付け
.size	ラベルの示すデータ・サイズの指定
.space	サイズ分の領域の確保
.str	文字列分の領域の確保
.text	.text セクションへの割り付け
.tibss	.tibss セクションへの割り付け
.tibss.byte	.tibss.byte セクションへの割り付け
.tibss.word	.tibss.word セクションへの割り付け
.tidata	.tidata セクションへの割り付け
.tidata.byte	.tidata.byte セクションへの割り付け
.tidata.word	.tidata.word セクションへの割り付け
.vdbstrtab	.vdbstrtab セクションへの割り付け
.vdebug	.vdebug セクションへの割り付け
.vline	.vline セクションへの割り付け
.word	1 ワードごとの領域の確保

付録 B　索引

【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。