お客様各位

カタログ等資料中の旧社名の扱いについて

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

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

2010年4月1日
ルネサスエレクトロニクス株式会社

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

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

ご注意書き

1. 本資料に記載されている内容は本資料発行時点のものであり、予告なく変更することがあります。当社製品のご購入およびご使用にあたりましては、事前に当社営業窓口で最新の情報をご確認いただきますとともに、当社ホームページなどを通じて公開される情報に常にご注意ください。
2. 本資料に記載された当社製品および技術情報の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、当社は、一切その責任を負いません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
3. 当社製品を改造、改変、複製等しないでください。
4. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器の設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因しお客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
5. 輸出に際しては、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。本資料に記載されている当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途の目的で使用しないでください。また、当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器に使用することができません。
6. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質水準を「標準水準」、「高品質水準」および「特定水準」に分類しております。また、各品質水準は、以下に示す用途に製品が使われることを意図しておりますので、当社製品の品質水準をご確認ください。お客様は、当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途に当社製品を使用することができません。また、お客様は、当社の文書による事前の承諾を得ることなく、意図されていない用途に当社製品を使用することができません。当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途または意図されていない用途に当社製品を使用したことによりお客様または第三者に生じた損害等に関し、当社は、一切その責任を負いません。なお、当社製品のデータ・シート、データ・ブック等の資料で特に品質水準の表示がない場合は、標準水準製品であることを表します。

   標準水準： コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット

   高品質水準： 輸送機器（自動車、電車、船舶等）、交通用信号機器、防災・防犯装置、各種安全装置、生命維持を目的として設計されていない医療機器（厚生労働省定義の管理医療機器に相当）

   特定水準： 航空機器、航空宇宙機器、海底中継機器、原子力制御システム、生命維持のための医療機器（生命維持装置、人体に埋め込み使用するもの、治療行為（患部切り出し等）を行うもの、その他直接人命に影響を与えるもの）（厚生労働省定義の高度管理医療機器に相当）またはシステム等
8. 本資料に記載された当社製品のご使用につき、特に、最大定格、動作電源電圧範囲、放熱特性、実装条件その他諸条件につきましては、当社保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めておりますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害などを生じさせないようお客様の責任において冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、機器またはシステムとしての出荷保証をお願いいたします。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
10. 当社製品の環境適合性等、詳細につきましては製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを固くお断りいたします。
12. 本資料に関する詳細についてのお問い合わせその他お気付きの点等がございましたら当社営業窓口までご照会ください。

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

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

お客様各位

資料中の「三菱電機」、「三菱XX」等名称の株式会社ルネサス テクノロジへの変更について

2003年4月1日を以って株式会社日立製作所及び三菱電機株式会社のマイコン、ロジック、アナログ、ディスクリート半導体、及びDRAMを除くメモリ(フラッシュメモリ・SRAM等)を含む半導体事業は株式会社ルネサス テクノロジに承継されました。
従いまして、本資料中には「三菱電機」、「三菱電機株式会社」、「三菱半導体」、「三菱XX」といった表記が残っておりますが、これらの表記は全て「株式会社ルネサス　テクノロジ」に変更されておりますのでご理解の程お願い致します。尚、会社商標・ロゴ・コーポレートステートメント以外の内容については一切変更しておりませんので資料としての内容更新ではありません。

注:「高周波・光素子事業、パワーデバイス事業については三菱電機にて引き続き事業運営を行います。」

2003年4月1日
株式会社ルネサス テクノロジ
カスタマサポート部

User's Manual

TW32R V.3.10 GNU ツールリファレンス

M32R 用　クロスツールキット(GNU 版)

ルネサス エレクトロニクス
www.renesas.com

2000.03

Edited and Translated by 三菱電機株式会社
Edited and Translated by 三菱電機セミコンダクタシステム株式会社

はじめに

この GNU ツールリファレンスは、次の GNU ドキュメントを参考にして、三菱 32 ビット MCU M32R ファミリ用 GNU ツールの仕様および機能をまとめたものです。

- Using as
- The GNU Bainary Utilities
- The C Preprocessor
- Using and Porting GNU CC
- Using LD, the GNU linker

GNU ドキュメントには、本リファレンスに記載した機能以外にも豊富な機能があり、各機能に関する説明、アドバイス、例などの情報も豊富に記載されています。さらに詳しい情報や他の便利な機能はないかとお探しの場合には、これらのドキュメントを一度参照してみてください。

■対象読者

このリファレンスでは、次の方を対象としています。

- C 言語による組み込み用プログラム開発およびデバッグ経験のある方
- 組み込み用プログラム開発において、C コンパイラ、アセンブラ、リンカのご使用経験のある方

1 gcc

Using and Porting GNU CC

Richard M. Stallman

Last updated 16 March 1998

For EGCS Version 1.0

Published by the Free Software Foundation 59 Temple Place - Suite 330 Boston, MA 02111-1307, USA Last printed April, 1998. Printed copies are available for $50 each. ISBN 1-882114-37-X

1.1 gcc オプション

1.1.1 ファイル拡張子および全般的な出力制御オプション

入力ファイルの拡張子は、どのようにコンパイルされるかを決定します。

file.c　　プリプロセス処理されなければならない C ソースコード

file.i　　プリプロセス処理されるべきでない C ソースコード

file.ii　　プリプロセス処理されるべきでない C++ソースコード

file.m　　Objective-C ソースコード。

file.h　　C ヘッダファイル (コンパイルやリンクされない)

file.cc
file.cxx
file.cpp
file.C　　プリプロセス処理されなければならない C++ソースコード

file.s　アセンブラコード

file.S　プリプロセス処理されなければならないアセンブラコード

other　その他のファイル名は、オブジェクトファイルとしてリンカへ直接渡されます。

-x *language*

'-x'オプションにより、入力言語を明確に指定することができます。後続の入力ファイルが、*language*であることを明確に指定します。ファイル拡張子によるデフォルトの選択よりも優先されます。この指定は、次の '-x'オプションが現れるまで、後続の入力ファイルに対して適用されます。*language* には、次の値を指定することができます。

```
c  objective-c  c++
c-header  cpp-output  c++-cpp-output
assembler  assembler-with-cpp
```

-x none　言語指定を無効にします。

-c　コンパイルまたはアセンブルのみ行い、オブジェクトファイルを生成します。リンクはされません。デフォルトでは、オブジェクトファイル名は、.c .i .s 等の拡張子が.o に変換されたものになります。

-S　コンパイルのみ行い、アセンブラコードファイルを生成します。アセンブル処理は、実施しません。デフォルトでは、アセンブラコードファイル名は、.c .i 等の拡張子が .s に変換されたものになります。また、コンパイルの必要ない入力ファイルの場合、無視されます。

-E　プリプロセス処理のみ行い、プリプロセス処理後のソースコードを標準出力に出力します。コンパイル処理は、実施しません。プリプロセス処理の必要ない入力ファイルの場合、無視されます。

-o *file*　出力ファイル名を FILE にします。

-v　コンパイラの各ステージで実行されるコマンドを標準エラー出力に表示します。また、バージョン番号を表示します。

-pipe　コンパイラの各ステージ間において、テンポラリファイルを使用しないで pipe を使用します。アセンブラが、pipe からリードできないシステムでは動作しません。GNU アセンブラは、問題なく動作します。

@*cmdfile*　*cmdfile* で指定されたファイルの内容は、空白文字 (改行、タブ含む) で区切られた引数としてコマンドラインに展開し、コマンドを実行するものです。この機能は、Windows 環境でのみ有効です。

例えば、cmdfile に次のように記述し、

cmdfile の内容:

```
-o a.out
-O
main.c
func.c
```

次のように実行した場合、

実行:

```
gcc @cmdfile
```

以下のように実行したのと同じ効果が得られます。

```
gcc -o a.out -O main.c func.c
```

1.1.2 C 言語の方言を制御するオプション

`-ansi` すべての ANSI C プログラムをサポートします。

このオプションは、以下の機能を有効／無効にします。

- ANSI C と互換性のない GNU C 特有の機能を無効にします。
  - asm, inline, typeofキーワード
  - unix, vaxなどの事前定義されたシステム識別マクロ
- ANSI trigraph の機能を有効にします。
- C++スタイルのコメント//を無効にします。

条件キーワード__asm__、__extension__、__inline__、__typeof__などは、-ansi の指定に関わらず動作します。-ansi オプションは、非 ANSI プログラムを受け付けないようにはしません。このためには、-ansi オプションと共に -pedantic オプションを指定する必要があります。-ansi オプションが指定された場合、__STRICT_ANSI__マクロが定義されます。'-ansi'オプションが指定された場合、abort、exit、_exitなどの組み込み関数は、組み込まれません。

`-fno-asm` asm、inline、typeofを識別子として使用できるように、キーワードとして認識しません。その代わりに、__asm__、__inline__、__typeof__は、使用することができます。'-ansi'は、暗黙に'-fno-asm'を適用します。

C++では、asm、inlineは標準のキーワードであるため、typeofキーワードのみに影響します。'-fno-gnu-keywords'を代わりに指定することもできます。これは、他の C++仕様の拡張キーワード (headofなど）を無効にします。

`-fno-builtin`

__で始まらない組み込み関数を認識しません。現在、影響する関数には、abort、abs、alloca、cos、exit、fabs、ffs、labs、memcmp、memcpy、sin、sqrt、strcmp、strcpy、strlenが含まれています。

-ansiオプションは、ANSI 標準でない allocaと ffsが組み込まれるのを無効にします。

`-trigraphs`

ANSI C の trigraph をサポートします。'-ansi'オプションは、暗黙に-trigraphs を適用します。

`-traditional`

K&R C などの伝統的な C コンパイラの仕様をサポートしようとします。サポートする項目は、次のようなものです。

- 関数内の extern宣言を、グローバルに影響するようにします。これは、暗黙の関数の宣言も含みます。
- 新しいキーワード、typeof、inline、signed、const、volatileを、認識しません。__typeof__、__inline__などは、使用することができます。
- ポインタ型と整数型の比較を常に許可します。
- Integral promotions において、unsigned shortと unsigned charを unsigned intに変換します。
- 範囲外の浮動小数点数の即値は、エラーになりません。
- ANSI のプリプロセス処理において不正とみなされる '0xe-0xd' のような数値を、式として扱います。
- 文字列定数は、コンスタントでなくても構いません。書き込み可能な空間に配置され、コンスタントを探す識別子は、別々に割り付けられます。（ '-fwritable-strings'の効果と同じ）

- register宣言されていない全てのオートマティック変数は、longjmpによって維持されます。通常 GNU C は、ANSI C に従っており、通常 volatile宣言されていないオートマティック変数は、維持されません。
- エスケープシーケンス '\x'と '\a'は、各々文字 'x'と 'a'と解釈します。
- C++プログラムでは、'-traditional'オプションを指定すると、thisへの代入を許可します。('-fthis-is-variable' もこの効果があります)。

通常の GNU C の組み込み関数と同じ名前をプログラムで使用するために、'-traditional'と同様-fno-builtinを使用することができます。

'-traditional'オプションは、'-traditional-cpp' オプションも有効にします。

-traditional-cpp
K&R C などの伝統的な C プリプロセッサの仕様をサポートしようとします。

- コメントは、スペースにするのではなく全く削除します。これは、伝統的なトークンの連結を許可します。
- プリプロセス処理ディレクティブの '#'は、必ず行の最初の文字でなければなりません。
- マクロ引数は、マクロ定義内では、文字列定数として認識されます(出現した場合、その値は、ダブルクォーテーションが削除された文字列となります)。プリプロセッサは、常に文字列定数は改行で終わるとみなします。
- '-traditional'が使用された時、事前定義マクロ__STDC__は、定義されません。しかし、__GNUC__は、定義されたままです。'-traditional'が使用されているかどうかに依存する処理を、ヘッダファイルに記述したいならば、これらの事前マクロをテストすることにより、GNU C、traditional GNU C、ANSI C、old C の区別することができます。事前定義マクロ__STDC__VERSION__も定義されません。
- もし改行が '\'でエスケープされていないならば、文字列定数は改行で終わるとみなします。('-traditional' が指定されない場合、文字列定数はタイプされた通り改行を含めることができます)。

-fcond-mismatch
Conditional Operator(? expr:expr) の式において、2 番目および 3 番目の引数に型の異なる式を記述することを許可します。それらの式の値は、空になります。

-funsigned-char
plain な charを unsigned charのように unsigned とみなします。

-fsigned-char
plain な charを signed charのように signed とみなします。これは、'-funsigned-char'の逆の '-fno-unsigned-char'の指定と等価です。一方、'-fno-signed-char'は、'-funsigned-char'の指定と等価です。

-fsigned-bitfields
-funsigned-bitfields
-fno-signed-bitfields
-fno-unsigned-bitfields
これらのオプションは、ビットフィールドの宣言に signed や unsignedの指定がない場合に、ビットフィールドが singed か unsigned かを制御します。デフォルトでは、intのような基本整数型が、符号付き型であることに一致するため、signed です。

しかし、'-traditional'が指定された場合、ビットフィールドはすべて unsigned になります。

`-fwritable-strings`
文字列定数を書き込み可能データのセグメントに格納し、独特のものとして扱いません。このオプションは、文字列定数に書き込みができると仮定しています、古いプログラムとの互換性のためのものです。‘`-traditional`’オプションもこれと同様の効果があります。

文字列定数に書き込みを行うことは、非常に悪い考えです。constants は、constant であるべきです。

`-fallow-single-precision`
単精度の数学演算を倍精度に変換しません。これは、‘`-traditional`’ オプションが指定されている場合にも有効です。K&R C では、オペランドサイズに関係なく全ての浮動小数点演算を倍精度に変換してしまいます。アーキテクチャによっては、単精度の方が倍精度よりも高速かもしれません。‘`-traditional`’を使用しなければならないが、オペランドが単精度であり、単精度で演算したい場合にこのオプションを指定することができます。このオプションは、ANSI または GNU C 互換(デフォルト) でコンパイルする時には有効となりません。

1.1.3 ワーニング制御オプション

次から説明する-W*flag* 形式オプションの多くは、機能の有効/無効の指定が可能です。例えば、‘`-Wimplicit`’の機能を無効にするオプション指定は、‘`-fno-implicit`’のように、‘`no-`’を付けて指定します。以下の説明では、有効/無効指定のうちどちらか一方の形式で説明してあります。

`-fsyntax-only`
syntax error に対するチェックのみ行い、それ以上は何もしません。

`-pedantic`
厳密な ANSI standard C と ISO C++に要求される全てのワーニングを表示します。また、禁止された拡張を使用するプログラムを排除します。

`-pedantic-errors`
ワーニングでなく、エラーにすることを除き、‘`-pedantic`’と同じです。

`-w` 全てのワーニングメッセージを抑止します。

`-Wno-import`
‘`#import`’の使用に関するワーニングメッセージを抑止します。

`-Wchar-subscripts`
配列の添字が `char`型である場合、ワーニングを出力します。これはプログラマが、あるマシン上でこの型が signed であることを忘れているので、一般にエラーを引き起こす原因となります。

`-Wcomment`
‘`/*`’コメントの中で、コメント開始文字列 ‘`/*`’が出現した場合、ワーニングを出力します。あるいは、‘`//`’コメント内で、バックスラッシュと改行が出現した場合も出力します。

`-Wformat` `printf`、`scanf`などの関数呼び出しにおいて、引数がフォーマット指定文字列に適した型であることをチェックします。

`-Wimplicit-int`
宣言が型を指定していない時、ワーニングを表示します。

`-Wimplicit-function-declaration`
`-Werror-implicit-function-declaration`
: 関数が宣言される前に使用された時、ワーニング (またはエラー) を表示します。

`-Wimplicit`
: '`-Wimplicit-int`'と '`-Wimplicit-function-declarations`'を指定したのと同じです。

`-Wmain`
: '`main`'の型が疑わしい場合にワーニングを表示します。

`-Wmultichar`
: ('F00F) のようなマルチバイト文字定数を使用している場合にワーニングを表示します。

`-Wparentheses`
: 限定されたコンテキストにおいて、かっこが省略されている場合にワーニングを出力します。例えば、明らかに値が予期される代入、間違い易い優先度を含む演算がネストしている場合です。

`-Wreturn-type`
: 関数がデフォルトの int 型の戻り値を持つよう定義されている場合、ワーニングを出力します。また、戻り値の型が void でない関数において、戻り値が記述されていない場合もワーニングを出力します。

`-Wswitch`
: enum 型のインデックスをもつ `switch`文において、enum 型の 1 つ以上の `case`ラベルが不足している場合にワーニングを出力します(`default`ラベルが存在する場合、このワーニングは出力されません)。このオプションを指定した場合、`case`ラベルが enum の範囲外である場合にもワーニングを出力します。

`-Wtrigraphs`
: trigraphs が見つかった場合、それらは有効であると仮定しワーニングを出力します。

`-Wunused`
: 次の場合にワーニングを出力します。
  - 変数が宣言を除き使用されていない
  - 関数が static 宣言されているが、定義されていない
  - ラベルが宣言されているが使用されていない
  - 文が、明らかに使用されない結果を計算する

  使用されない関数のパラメータに対してワーニングを表示するには、'`-W`'と '`-Wunused`'の両方のオプションを指定します。

  式に対するワーニングを抑止する簡単な方法は、void にキャストすることです。使用されない変数やパラメータに対しては、'`unused`' attibute を使用することです。

`-Wuninitialized`
: オートマティック変数が初期化されないで使用されている場合にワーニングを表示します。

  これらのワーニングは、最適化した場合のみ出力可能です。なぜならば、この処理は、最適化の時のみ処理されるデータフロー解析情報が必要になるからです。もし、'`-O`'オプションを指定していないならば、単にこれらのワーニングは出力されません。

  これらのワーニングは、register allocation の候補である変数に対してのみ発生します。従って、以下のような場合、たとえそれらがレジスタに保持されることがあっても、ワーニングは表示されません。
  - `volatile`宣言されている変数
  - アドレスが使用されている変数

- 1,2,4,8 バイトを越えるサイズの変数
- 構造体変数
- 共用体変数
- 配列

決して使用されない値を計算するためだけに使用される変数についても、このワーニングは出力されないことがあります。これは、ワーニングが出力される以前に、データフロー解析により計算処理が削除されることがあるからです。

これらのワーニングは、選択可能になっています。コードはエラーがあることが明かであるにも関わらず、正しいかもしれない全ての理由を発見することは、GNU CC にとっては十分できないからです。以下に、例を示します。

```
{
  int x;
  switch (y)
    {
    case 1: x = 1;
      break;
    case 2: x = 4;
      break;
    case 3: x = 5;
    }
  foo (x);
}
```

yが常に 1,2,3 の値ならば、xは常に初期化されます。しかし、このことは、GNU CC には分かりません。次は同様の別の例です。

```
{
  int save_y;
  if (change_y) save_y = y, y = new_y;
  ...
  if (change_y) y = save_y;
}
```

これは、バグではありません。save_yは、セットされている時のみ使用されるからです。

決して return しない関数に対するワーニング出力は、noreturnとして宣言することにより回避することができます。

`-Wunknown-pragmas`
GCC が理解できない #pragma である場合、ワーニングを表示します。

`-Wall` ここまでに説明した全ての '-W'オプションを、暗黙に指定します。

次から説明する '-W...'オプションは、'-Wall'によって暗黙に指定されません。

`-W` 次の場合に、特別なワーニングメッセージを出力します。

- volatile宣言されていない変数が、longjmpをコールすることにより変わるかもしれない場合。これらのワーニングは、最適化を指定した場合のみ可能です。
- 関数が戻り値を返したり返さなかったりする場合。例えば、次のようなプログラムの場合にワーニングが出力されます。

```
foo (a)
{
  if (a > 0)
    return a;
}
```

- 式の文やコンマ式 (comma expression) の左辺が side effects を含まない場合。使用されない式を、void にキャストすることにより、ワーニングを抑止することができます。例えば、‘x[i,j]’のような式の場合にワーニングが出力されます。しかし、‘x[(void)i,j]’のように記述すればワーニングは発生しません。
- unsigned の値が、ゼロ (0) と ‘<’または ‘<=’により比較される場合。
- ‘x<=y<=z’のような比較が現れた場合。これは、‘(x<=y ? 1:0) <= z’と同じであり、一般の数学表記の解釈とは異なります。
- staticのような Storage-class specifiers が、宣言の最初に記述されていない場合。C 標準では、この仕様は既に使用されていません。
- ‘-Wall’あるいは ‘-Wunused’が指定されているならば、未使用の引数に関するワーニングが出力されます。
- signed 値が、unsigned に変換される時、signed と unsigned 値との比較が不正な結果になる可能性がある場合。
- 集合型が、ブラケットで囲まれた初期化式を持っている場合。例えば、以下の場合、x.hに対する初期化式のかっこが不足しているためワーニングが出力されます。

```
struct s { int f, g; };
struct t { struct s h; int i; };
struct t x = { 1, 2, 3 };
```

-Wtraditional
: K&R C のなどの伝統的な C と ANSI C で、異なる振る舞いをするものについてワーニングを出力します。
  - マクロ本体中の文字列定数内に起こるマクロ引数の場合伝統的な C では引数は、置換されますが、ANSI C では constant の一部になります。
  - 1 つのブロック内で external 宣言された関数が、そのブロックの終わる前に使用された場合
  - switch文が long型のオペランドを持つ場合

-Wundef
: 未定義の識別子が、‘#if’ directive で評価された場合にワーニングを出力します。

-Wshadow
: ローカル変数が、別のローカル変数を隠してしまう場合ワーニングを出力します。

-Wid-clash-*len*
: 異なる 2 つの識別子が、最初の *len* 文字で一致する場合、ワーニングを出力します。

-Wlarger-than-*len*
: *len* バイトより大きなオブジェクトが定義された場合にワーニングを出力します。

-Wpointer-arith
: 関数型や voidなどで、サイズに依存する (“size of” など) ものについてワーニングを出力します。GNU C では、void *と関数へのポインタの計算において利便のためにこれらの型をサイズ 1 としています。

-Wbad-function-cast
: 関数呼び出しが、一致しない型へキャストされている場合にワーニングを出力します。例えば、int malloc()が、anything *にキャストされている場合にワーニングを出力します。

`-Wcast-qual`
ポインタが、ターゲットの型から型修飾子を削除したような型へキャストされている場合にワーニングを出力します。例えば、`const char *`が普通の `char *`へキャストされている場合にワーニングを出力します。

`-Wcast-align`
ポインタが、ターゲットの要求されたアライメントを増加するような型へキャストされている場合にワーニングを出力します。例えば、int 型が、2 または 4 バイト境界でアクセスされることかできるマシンにおいて、`char *`が `int *`へキャストされるような場合にワーニングを出力します。

`-Wwrite-strings`
文字列定数のアドレスを、`const`でない `char *`へコピーする場合にワーニングになるように、文字列定数を `const char[`*length*`]`の型にします。

`-Wconversion`
プロトタイプ宣言において、プロトタイプ宣言がない場合に同じ引数に起こる型変換と異なる型変換を引き起こす場合にワーニングを出力します。これは、固定小数点へや逆の型変換、デフォルトプロモーションの型変換を除く固定小数点引数の幅や符号を変更する型変換を含みます。

また、負の整数定数表現が暗黙に符号なし型に型変換される場合にも、ワーニングを出力します。例えば、`x`が符号なしの場合に `x = -1` の代入などに対してワーニングを出力します。しかし、`(unsigned)-1` のように明確にキャストしたものについてはワーニングを出力しません。

`-Wsign-compare`
signed 値が、unsigned に変換される時に、signed と unsigned の比較が不正な結果となる場合にワーニングを出力します。

`-Waggregate-return`
構造体や共用体を返す関数が、定義あるいは呼び出しされている場合にワーニングを出力します。

`-Wstrict-prototypes`
関数が、引数の型を指定することなく宣言あるいは定義されている場合にワーニングを出力します。

`-Wmissing-prototypes`
グローバル関数が、それ以前にプロトタイプ宣言なしに定義されている場合にワーニングを出力します。このワーニングは、その定義自身がプロトタイプとして提供されている場合にもワーニングを出力します。この目的は、ヘッダファイルに宣言し忘れたグローバル関数を発見するためのものです。

`-Wmissing-declarations`
グローバル関数が、それ以前に宣言されていない場合にワーニングを出力します。

`-Wredundant-decls`
同じスコープ内で一度以上宣言されているものについて、ワーニングを出力します。複数の宣言が正しく、何も変更がない場合にも出力します。

`-Wnested-externs`
関数内に `extern`宣言されている場合にワーニングを出力します。

`-Winline` 関数が inline として定義されているか、'`-finline-functions`' オプションが指定されている場合、関数をインライン化できないならばワーニングを出力します。

`-Wold-style-cast`
プログラム内で、old-style(C-style) のキャストが使用されている場合にワーニングを出力します。

`-Wlong-long`
long long 型が使用されている場合にワーニングを表示します。このオプションは-pedantic が指定されている時のみ有効になります。

`-Werror` 全てのワーニングをエラーにします。

1.1.4 プログラムデバッギングのためのオプション

`-g` stabs デバッグ情報を生成します。

`-gdwarf` DWARF バージョン 1 フォーマットのデバッグ情報を生成します。

`-gdwarf+` GNU デバッガ (GDB) によってのみ使用される情報を含む、DWARF バージョン 1 フォーマットのデバッグ情報を生成します。

`-gdwarf-2`
DWARF バージョン 2 フォーマットのデバッグ情報を生成します。

`-Q` 関数毎に、コンパイル処理の各パスの統計情報を表示します。

`-save-temps`
テンポラリに作成するコンパイラの中間ファイルを残します。カレントディレクトリに、ソースファイル名をベースにした名前で残します。'`foo.c`'を '`-c -save-temps`'オプションでコンパイルすると、'`foo.o`'と同様に'`foo.i`'と'`foo.s`'を残します。

`-print-file-name=`*library*
リンクする時に使用されるライブラリファイルで、*library* で指定したライブラリファイルの絶対パスを表示します。他には何もしません。このオプションが指定された場合、コンパイルやリンクは行いません。ファイル名を表示するだけです。

`-print-prog-name=`*program*
'`-print-file-name`'と同じですが、'`cpp`'のようなプログラムを探します。

`-print-libgcc-file-name`
'`-print-file-name=libgcc.a`'と指定したのと同じです。これは、'`-nostdlib`'や'`-nodefaultlibs`'を指定しているが'`libgcc.a`'とリンクしたい場合に有効です。

`-print-search-dirs`
コンパイラ構築時のインストレーションディレクトリと、gcc が検索するプログラムとライブラリのディレクトリを表示します。他には何もしません。

1.1.5 最適化オプション

`-O`
`-O1` 最適化を実施します。

'`-O`'を指定していない場合、コンパイラは、`register` 宣言されている変数のみレジスタに割り付けます。

'`-O`'を指定した場合、コンパイラは、コードサイズの削減と実行速度を向上するコードを生成しようとします。また、'`-fthread-jumps`'と'`-fdefer-pop`'を有効にします。ディレイスロットを持つマシンでは、'`-fdelayed-branch`'を有効にします。

`-O2` ‘`-O`’よりも、さらに最適化を実施します。サポートされている最適化のうち、サイズと速度のトレードオフに影響しないほぼ全ての最適化を実施します。‘`-O2`’を指定した場合、コンパイラは、loop unrolloing や function inlining を実施しません。‘`-O`’と比較すると、このオプションは、コンパイル処理時間の増加と生成されたコードのパフォーマンスを向上します。

‘`-O2`’は、loop unrolling と function inlining 以外の選択可能な最適化を全て有効にします。また、‘`-fforce-mem`’オプションも有効にします。

`-O3` ‘`-O2`’よりもさらに、最適化を実施します。‘`-O3`’は、‘`-O2`’で実施される全ての最適化と、‘`inline-funtions`’オプションも有効にします。

`-O0` 最適化を実施しません。

`-Os` コードサイズを削減することを目的とした最適化を実施します。‘`-Os`’は、‘`-O2`’で実施される全ての最適化のうち、一般にコードサイズを増加しない最適化をすべて実施します。また、さらにコードサイズを削減するための最適化を実施します。

他の ‘`-O`’オプションなどと複数指定した場合、最後に指定されたオプションが有効になります。

次から説明する-f*flag* 形式オプションの多くは、機能の有効/無効の指定が可能です。例えば、‘`-ffoo`’の機能を無効にするオプション指定は、‘`-fno-foo`’のように、‘`no-`’を付けて指定します。以下の説明では、有効/無効指定のうちどちらか一方の形式で説明してあります。

`-ffloat-store`
浮動小数点変数をレジスタに保持しません。また、これを変更するオプションを禁止します。

`-fno-defer-pop`
関数呼び出しから戻った直後に、常に関数への引数をポップするようにします。

`-fforce-mem`
メモリオペランドを、演算の前にレジスタにコピーするようにします。‘`-O2`’オプションは、このオプションを有効にします。

`-fforce-addr`
メモリアドレス定数を、演算の前にレジスタにコピーするようにします。これは、‘`-fforce-mem`’のみ指定するよりも、より良いコードを生成することがあります。

`-fomit-frame-pointer`
フレームポインタを必要としない関数において、フレームポインタをレジスタに保持しません。ただし、このオプションを有効にした場合、デバッグができなくなる場合があります。

`-fno-inline`
`inline`キーワードを無視します。通常、このオプションは、コンパイラが関数を inline 化することを防ぐために使用されます。最適化指定していないならば、関数は inline 化されません。

`-finline-functions`
全ての単純な関数をそれらの呼び出し元へ inline 化します。

`-fkeep-inline-functions`
関数が inline 化され、`static`で定義されている場合にも、関数の呼び出し形式も別に出力します。`extern inline`された関数には、効果はありません。

`-fkeep-static-consts`
最適化オプションが有効でない場合に、参照されていない `static const`定義された変数を出力します。

`-fno-function-cse`
関数のアドレスをレジスタ内に置きません。

`-ffast-math`
このオプションは、生成されるコードのスピードを上げる最適化のために、ANSI や IEEE および (あるいは) 仕様に違反することを許可します。例えば、`sqrt`関数への引数が非負数であること、および、浮動小数点数が NaN(非数) でないと仮定します。

`-fstrength-reduce`
ループ演算の強さの軽減と、繰り返し変数の省略に関する最適化を実施します。

`-fthread-jumps`
ある分岐が、その分岐によって包含される比較の場所へ分岐するかどうかを調べる最適化を実施します。最初の分岐は、その条件が真か偽かわかっているかどうかに依存して、2 番目の分岐先かその分岐の直後にされます。

`-fcse-follow-jumps`
共通部分式の削除 (common subexpression elimination) において、到達することのない分岐先への分岐があるかどうかを細かく調べます。例えば、else 節付きの if 文で、条件が偽になる場合に分岐にします。

`-fcse-skip-blocks`
これは、'`-cse-follow-jumps`'と同様ですが、条件付きでブロックを越えるような分岐にします。else 節のない `if`文で、'`-fcse-skip-blocks`'は、`if`文の本体を分岐します。

`-frerun-cse-after-loop`
ループ最適化 (loop optimization) が実施された後に、共通部分式の削除 (common subexpression elimination) を再実施します。

`-frerun-loop-opt`
ループ最適化 (loop optimizer) を 2 回実施します。

`-fgcse` グローバルな共通部分式削除 (global common subexpression elimination) パスを実施します。

`-fexpensive-optimizations`
比較的実施するのにコストがかかり、効果の少ない最適化を実施します。

`-fschedule-insns`
ターゲットマシンでサポートされているならば、利用準備のできてないデータに対する要求による実行ストールを削減するために、命令を並び替えるようとします。

`-fschedule-insns2`
'`-fschedule-insns`'と同様ですが、レジスタ割り付け後、命令スケジューリングパスを実施します。

`-ffunction-sections`
ターゲットが、任意のセクションをサポートしているならば、出力ファイル中に各関数をそれぞれのセクションに置くようにします。関数の名前が、出力ファイル中のセクション名になります。

`-funroll-loops`
ループ展開 (loop unrolling) の最適化を実施します。この最適化は、ループの繰り返し回数が、コンパイル時や実行時に決定することができる場合のみ実施されます。

`-funroll-all-loops`
ループ展開 (loop unrolling) の最適化を実施します。この最適化は、全てのループに対して実施されます。また、通常プログラムの実行速度を遅くします。

`-fmove-all-movables`
ループ内の全ての不変計算処理 (invariant computations) を、ループの外へ移動します。

`-freduce-all-givs`
全てのループ内の一般誘導変数に対して、演算の強さが軽減されるようにします。

`-fno-peephole`
マシン依存のピープホール最適化を実施しません。

`-fregmove`
あるマシンでは、命令により 2 オペランドのみサポートします。そのようなマシンでは、GNU CC は、特別なコピーをしなければなりません。'`-fregmove`'オプションは、レジスタ割り付けの前に、コピーするべきマシンのデフォルトを無効にします。

1.1.6 プリプロセッサ制御オプション

`-include` *file*
正規の入力ファイルを処理する前に、*file* を入力ファイルとして処理します。

`-imacros` *file*
正規の入力ファイルの前に、*file* を入力として処理します。*file* から生成される結果の出力は、捨てられます。*file* から生成される出力は捨てられるので、'`-imacros` *file*'の効果は、メイン入力内で使用するために *file* 内で定義されたマクロを利用可能にすることです。

`-idirafter` *dir*
2 番目のインクルードパス (second include path) に、*dir* で指定されたディレクトリを追加します。

`-iprefix` *prefix*
次の'`-iwithprefix`'オプションに対する前置指定子として、*prefix* を指定します。

`-iwithprefix` *dir*
2 番目のインクルードパスに *dir* ディレクトリを追加します。ディレクトリ名は、*prefix* と *dir* を結合することにより生成されます。

`-iwithprefixbefore` *dir*
メインインクルードパスに *dir* ディレクトリを追加します。ディレクトリ名は、*prefix* と *dir* を結合することにより生成されます。

`-isystem` *dir*
2 番目のインクルードパスの最初に、*dir* ディレクトリを追加します。また、システムディレクトリとしてマークされます。

`-nostdinc`
ヘッダファイルのために標準のシステムディレクトリを検索しません。

`-undef` 標準でないマクロを事前定義しません (アーキテクチャフラグも含みます)。

`-E` C プリプロセッサのみ実行します。指定された全ての C ソースファイルをプリプロセス処理し、結果を標準出力または指定された出力ファイルへ出力します。

`-C` コメントを捨てないように、プリプロセッサに指示します。'`-E`'オプションと使用します。

`-P` '`#line`'ディレクティブを生成しないように、プリプロセッサに指示します。'`-E`'オプションと使用します。

`-M` プリプロセッサに、`make`が使用するための各々のオブジェクトファイルの依存関係を記述したルールを出力するように指示します。

`-MM` '`-M`'と同様ですが、'`#include "`*file*`"`'でインクルードされたユーザーヘッダファイルのみ出力します。'`#include <`*file*`>`'で入力されたシステムヘッダファイルは、省略されます。

`-MD` '`-M`'と同様ですが、依存情報は、入力ファイル名の".c"を".d"に変えたファイルに出力されます。

`-MMD` ユーザーヘッダファイルのみの記述が出力されることを除き、'`-MD`'と同じです。

`-MG` 不足ヘッダファイルを、生成されるファイルであり、ソースファイルと同じディレクトリに存在すると仮定します。'`-MG`'を指定する場合、'`-M`'または'`-MM`'のどちらかも指定しなければなりません。'`-MG`'は、'`-MD`'と'`-MMD`'ではサポートされていません。

`-H` 通常の動作とともに、使用されるヘッダファイル名を表示します。

`-A`*question*`(`*answer*`)`
question に対する解 *answer* を Assert します。'`#if #`*question*`(`*answer*`)`'のようなプリプロセス条件でテストされます。'`-A-`'は、通常ターゲットマシンを指定する標準 Assertion を無効にします。

`-D`*macro* マクロ *macro* を文字'`1`'として定義します。

`-D`*macro*`=`*defn*
マクロ *macro* を *defn* として定義します。コマンド行での全ての'`-D`'指定は、どんな'`-U`'オプションよりも前に処理されます。

`-U`*macro* マクロ *macro* を未定義にします。'`-U`'オプションは、全ての'`-D`'オプションの後に評価されます。しかし、'`-include`'と'`-imacros`'オプションの前に処理されます。

`-dM` プリプロセッサに、プリプロセス処理の最後で有効なマクロ定義のリストを表示するように指示します。'`-E`'オプションと使用します。

`-dD` プリプロセッサに、全てのマクロ定義を出力に渡すよう指示します。

`-dN` マクロ引数と内容が省略されることを除いて、'`-dD`'と同じです。'`#define` *name*'だけが、出力に含まれます。

`-trigraphs`
ANSI C の trigraphs をサポートします。'`-ansi`'オプションもこの効果があります。

`-Wp,`*option*
プリプロセッサにオプションとして *option* を渡します。*option* が、コンマ (,) を含んでいる場合、コンマで複数のオプションに分けられます。

1.1.7 アセンブラ制御オプション

`-Wa,`*option*
アセンブラへ、オプションとして *option* を渡します。*option* が、コンマ (,) を含んでいる場合、コンマで複数のオプションに分けられます。

1.1.8 ディレクトリ検索制御オプション

`-I`*dir* ヘッダファイルを検索すべきディレクトリのリストの先頭に、*dir* ディレクトリを追加します。

`-I-` '`-I-`'オプションの前に'`-I`'オプションで指定されたディレクトリは、'`#include "`*file*`"`'の場合のみ検索され、'`#include <`*file*`>`' は検索されません。

1.1.9 M32R ファミリ専用オプション

`-mmodel=small`
全てのオブジェクトは、低アドレス側の 16MB のメモリ内に配置されるものと仮定します (それらのアドレスが、`ld24`命令でロードできると仮定します)。また、全てのサブルーチンが、`bl`命令で到達可能であると仮定します。これはデフォルトです。
特定のオブジェクトのアドレッシングは、`model` attribute で設定することができます。

`-mmodel=medium`
オブジェクトが、32 ビットアドレス空間に配置されるかもしれないと仮定します (コンパイラは、これらのアドレスをロードするために、`seth/add3`命令を生成するようになります)。また、全てのサブルーチンは、`bl`命令で到達可能であると仮定します。

`-mmodel=large`
オブジェクトが、32 ビットアドレス空間に配置されるかもしれないと仮定します (コンパイラは、これらのアドレスをロードするために、`seth/add3`命令を生成するようになります)。また、全てのサブルーチンは、`bl`命令で届かないかもしれないと仮定します (コンパイラは、より遅い `seth/add3/jl`命令列を生成します)。

`-msdata=none`
small data 領域 ('`.sdata`'、'`.sbss`') の使用を無効にします。変数は、`section` attribute が指定されていなければ、'`.data`'、'`.bss`'、'`.rodata`'のいずれかに置かれます。

`-msdata=sdata`
small data 領域に、small global および static data を配置します。しかし、それらを参照するための特別なコードは生成されません。

`-msdata=use`
small data 領域に、small global および static data を配置します。また、それらを参照するための特別なコードを生成します。'`_SDA_BASE_`'シンボルからの相対参照になります。

`-G` *num* *num* バイト以下の global と static なオブジェクトを、通常の data や bss セクションに配置する代わりに、small data や bss セクションに配置するようにします。デフォルトでは、この *num* の値は 8 です。
このオプションの効果を得るためには、'`-msdata`'オプションには、'`sdata`'または'`use`'のいずれかが指定されなければなりません。
全てのモジュールは、同じ'`-G` *num*'値でコンパイルされるべきです。

`-m32rx` M32Rx の命令セットをサポートします。

`-m32r` M32R の命令セットをサポートします。

1.1.10 コード生成制御オプション

`-fshort-enums`
: enum型に対して、宣言された値を表現できる範囲のバイト数だけ割り当てるようにします。具体的には、値を保持することができる最も小さい整数型に等しくなります。

`-fshort-double`
: doubleに対して、floatと同じサイズを使用します。

`-fno-ident`
: '#indent'ディレクティブを無視します。

`-fverbose-asm`
: 生成されたアセンブリコード中に、特別なコメント情報を出力します。

`-fvolatile`
: volatile であるべきポインタを介しての全てのメモリ参照を考慮します。

`-fvolatile-global`
: volatile であるべき extern および global data オブジェクトへの全てのメモリ参照を考慮します。

`-ffixed-`*reg*
: *reg* で指定されたレジスタを、固定されたレジスタとして扱います。生成されるコードは、決してそのレジスタを参照しません (スタックポインタやフレームポインタとして使用される場合を除いて)。

`-fcall-used-`*reg*
: *reg* で指定されたレジスタを、関数呼び出しにより破壊されてもよい、割り付け可能なレジスタとして扱います。この方法でコンパイルされた関数は、*reg* レジスタを退避/復帰しません。

`-fcall-saved-`*reg*
: *reg* で指定されたレジスタを、関数呼び出しによりセーブされてもよい、割り付け可能なレジスタとして扱います。この方法でコンパイルされた関数は、*reg* レジスタ使用する場合、そのレジスタを退避/ 復帰します。

`-funaligned-pointers`
: 全てのポインタが、アライメント調整されていないアドレスであると仮定します。

1.1.11 オフセット情報オプション

`-offset-info output-file`

このオプションは、アセンブラから C の構造体へのアクセスを簡単にするものです。コンパイラは、構造体の各メンバに対して、構造体内でのメンバのバイトオフセットを定義した `.equ` シンボル定義ファイルを生成します。シンボルの名前は、構造体タグ名とメンバ名を、アンダースコアで結合した名前になります。

使用例は、次の通りです。

```
gcc -fsyntax-only -offset-info m.s -x c m.h
```

`m.h`は、構造体を含むヘッダファイル名です。また、`m.s` は、出力ファイルです。

以下は、-offset-infoによって生成される出力の例です。

入力ファイル例:

```
struct W {
    double d;
    int i;
    };

struct X {
    int a;
    int b;

    struct Y {
        int a;
        int b;
        };

    struct Y y;
    struct Y yy[10];
    struct Y* p;
    };
```

出力ファイル例:

```
.equ W_d,0
.equ W_i,8
.equ Y_a,0
.equ Y_b,4
.equ X_a,0
.equ X_b,4
.equ X_y,8
.equ X_yy,16
.equ X_p,96
```

-offset-infoオプションは、次の注意事項があります。

- ビットフィールドメンバは、出力されません。
- ホストのワードサイズを超えるオフセットを持つメンバは、出力されません。
- 定数でないオフセットを持つメンバは出力されません。

1.1.12 環境変数

TMPDIR　TMPDIRは、テンポラリファイルを使用するためのディレクトリを指定します。

GCC_EXEC_PREFIX
GCC_EXEC_PREFIXは、コンパイラによって実行されるサブプログラム名の接頭辞を指定します。この接頭辞とサブプログラム名が連結される時、slash は付加されません。もし必要ならば、接頭辞の最後が slash で終わるように指定することが

できます。もし、GNU CC が指定された接頭辞を使用してサブプログラムを見つけることができなかった場合、標準の場所を探します。

GCC_EXEC_PREFIXのデフォルト値は、‘*prefix*/lib/gcc-lib/’ です。ここでいう *prefix* は、‘configure’ script 実行時に指定した prefixの値です。

‘-B’で指定された接頭辞は、この環境変数で指定した接頭辞よりも先に取り扱われます。この環境変数で指定した接頭辞は、‘crt0.o’のようなリンク時に使用されるファイルを検索するためにも使用されます。

また、この環境変数で指定した接頭辞は、別の方法において、ヘッダファイル用ディレクトリを検索するためにも使用されます。標準のディレクトリ名は、通常‘/usr/local/lib/gcc-lib’(より正確には、GCC_INCLUDE_DIRの値）で始まります。GNU CC は、この接頭辞で始まるディレクトリ名から立替のディレクトリ名を生成し、そのディレクトリを検索します。例えば、‘-Bfoo/’と指定した場合、GNU CC は、通常は‘/usr/local/lib/bar’を検索しますが、‘foo/bar’を検索するようになります。これらの立替ディレクトリは、最初に検索され、次に標準のディレクトリが検索されます。

COMPILER_PATH
: COMPILER_PATHは、PATHのようにコロンで区切られたディレクトリのリストです。GCC_EXEC_PREFIXを使用してサブプログラムを見つけることができない場合、GNU CC は、ここで指定されたディレクトリを使用してサブプログラムを検索します。

LIBRARY_PATH
: LIBRARY_PATHは、PATHのようにコロンで区切られたディレクトリのリストです。ネイティブコンパイラとして構築されている時には、GCC_EXE_PREFIXを使用してリンカ用の特別なファイルを見つけることができない場合、GNU CC は、ここで指定されたディレクトリを使用してリンカ用の特別なファイルを検索します。GNU CC を使用してリンクする場合には、‘-l’オプションで指定した通常のライブラリを検索する時もこれらのディレクトリが使用されます。(しかし、‘-L’が指定された場合、-L で指定したディレクトリが最初に検索されます)。

C_INCLUDE_PATH
CPLUS_INCLUDE_PATH
OBJC_INCLUDE_PATH
: これらの環境変数は、各々の言語に関連したものです。各々の環境変数値は、PATH のようにコロンで区切られたディレクトリのリストである。GNU CC は、ヘッダファイルを捜す時、‘-I’で指定されたディレクトリを検索した後、言語に対応するここで指定した環境変数で指定したディレクトリリストを使用して検索します。その後、標準のヘッダファイルディレクトリが検索されます。

DEPENDENCIES_OUTPUT
: make する際必要となるヘッダファイルの依存関係の出力方法を指定します。この出力は、ほとんど‘-M’オプションの出力ですが、別ファイルに出力され、通常のコンパイルの結果に付加したものです。

  DEPENDENCIES_OUTPUTの値は、単に１つのファイル名です。make rule がそのファイルに記述される場合、ソースファイル名からターゲット名を推測します。あるいは、その値は、‘*file target*’の形式を持つ事もできます。この場合ターゲット名として *target* を使用して *file* にルールを書き出します。

1.2 C 言語の方言 (M32R ファミリ特有)

1.2.1 プリプロセッサシンボル

コンパイラは、デフォルトで次のプリプロセッサシンボルを定義します。前後に 2 つのアンダースコアが付いています。

- `__M32R__`

1.3 メモリモデル

オブジェクトの配置アドレス、関数の配置アドレスにより、次のメモリモデルに分類しています。対応するメモリモデルのアプリケーションを開発するには、対応するメモリモデルのコードを生成するように、コンパイラにコンパイルオプション `-mmodel=<`*model-name*`>`によって指示します。

モデル	説明
small	全てのオブジェクトは、低アドレス側の 16MB のメモリ内に配置されると仮定します (それらのアドレスが、`ld24`命令でロードできると仮定します)。また、全てのサブルーチンが、`bl`命令で到達可能であると仮定します。(`-mmodel=<`*model-name*`>` 指定がない場合のデフォルト)
medium	オブジェクトが、32 ビットアドレス空間に配置されると仮定します (コンパイラは、これらのアドレスをロードするために、`seth/add3`命令を生成するようになります)。また、全てのサブルーチンは、`bl`命令で到達可能であると仮定します。
large	オブジェクトが、32 ビットアドレス空間に配置されると仮定します (コンパイラは、これらのアドレスをロードするために、`seth/add3`命令を生成するようになります)。また、全てのサブルーチンは、`bl`命令で届かないと仮定します (コンパイラは、より遅い `seth/add3/jl`命令列を生成します)。

1.4 データ型

データ型とそのサイズは、次の通りです。

データ型	サイズ (バイト)
`char`	1
`short`	2
`int`	4
`long`	4
`long long`	8
`float`	4
`double`	8
ポインタ	4

1.5 構造体、共用体、ビットフィールドのデータ表現

構造体および共用体メンバのメモリ内の割り付けは、次の規則が適用されます。

- メンバが short 型の場合、2 バイト境界にアライメントされます。int, long, float, double, long long 型は、4 バイト境界にアライメントされます。char 型は、アラインメントされません。
- 構造体や共用体メンバの場合、その構造体や共用体を構成している最も大きいメンバのアライメントになります。
- ビットフィールドは、ビッグエンディアンでパッキングされます。また、それらの型の境界をまたがないようにアライメントされます。
  次のような記述の場合、

```
struct {
     int x:8;
     int y:31;
} s = { 0x2, 0x4 };
```

  以下のような割り付けになります。

```
.byte 5
.zero 3
.byte 0x0
.byte 0x0
.byte 0x0
.byte 0x8
```

- 共用体は、そのメンバの最も大きいアライメントサイズにアライメントされます。

1.6 レジスタの使用規則

レジスタの使用規則は、次の通りです。

`r0 - r3` 関数へ引数を渡すために使用されます。
追加の引数についてはスタックを使用して渡されます。r0及び r1は、関数の戻り値としても使用されます。
これらのレジスタの値は、関数間で保持されません(関数呼び出し後のレジスタの値が、関数呼び出し前の値と一致することは保証されない)。

`r4 - r7` 式評価のためのテンポラリレジスタとして使用されます。これらのレジスタの値は、関数間で保持されません。r4は、関数プロローグで使用するテンポラリレジスタ用に予約されています。
r6 は、Position Independent Code(PIC) の呼び出し手続きにおいて、テンポラリとしても予約されているので、関数の呼び出し手続きや入り口で使用されないこともあります。
r7は、ネスト関数(GNU C の拡張機能) の static chain pointer としても使用されるので、関数呼び出し手続きや入り口で使用されないこともあります。

`r8 - r11` 式評価のためのテンポラリレジスタとして使用されます。これらのレジスタの値は、関数間で保持されます。

`r12` 式評価のためのテンポラリレジスタとして使用されます。このレジスタの値は、関数間で保持されます。また、"global pointer"用としての使用目的にも予約されています。

`r13(fp)` フレームポインタとして使用されます。

`r14(lr)` リンクレジスタとして使用されます。このレジスタは、関数の戻り番地を保持します。また、戻り番地がサーブされている場合、式評価に使用されることもあります。

`r15(sp)` スタックポインタとして使用されます。

`accumulator`
アキュムレータは、関数間で保持されません。

`psw(condition bit)`
psw中のコンディションビットは、関数間で保持されません。

1.7 スタックフレームの構成

スタックフレームは、関数が呼び出されるたびに (関数の入り口で) スタック上に割り付けられる領域であり、基本構成は下図のようになります。
スタックは上位アドレスから下位アドレス方向に向かって積まれ、スタックポインタ (SP) がスタックのトップを指し示します。スタックポインタは、常に 4 バイト境界に調整されていなければなりません。スタックフレームは、次の領域から構成されます。

- レジスタ退避領域
  関数内で使用する汎用レジスタを退避 (現在の値を保存) する領域です。フレームポインタおよびリンクレジスタもこの領域に退避されます。また、可変個数の引数を持つ関数では、レジスタで渡される引数の退避にも使用されます。
- ローカル変数領域
  関数内で宣言されたローカル変数の領域です。
- alloca 確保領域
  関数内で alloca 関数によって確保される領域です。
- スタック上引数領域
  他の関数へ引数を渡す場合に使用される領域です。

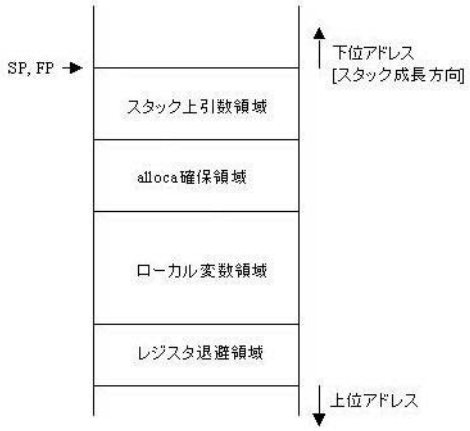

図 1. スタックフレームの構成

1.8 引数の設定規則

引数の設定は、次の規則に従います。

- 引数は r0から r3までのレジスタ引数として渡されます。r0から r3までのレジスタを使いきってしまった場合、スタックを使って渡されます。
- 1 つの引数のサイズが 8 バイト以下で、引数設定のためにレジスタを使用することが可能ならばレジスタ引数として渡されます。
  但し、構造体や共用体などのように複数のメンバから構成されており、4 バイトを越えるものについては、レジスタで渡されるとともにスタックでも渡されます。
- 1 つの引数のサイズが 8 バイトを越えるものは、常にポインタ参照の形で渡されます。つまり、引数がスタック上にコピーされ、そのスタック上のコピーへのポインタがレジスタに設定されて渡されます。
- データの型がレジスタ引数のサイズを越える場合、レジスタとスタックで渡されます。例えば、r3レジスタを使って渡されるデータ型が 'long long'型である場合、上位側 4 バイトが r3で渡され、下位側 4 バイトがスタックで渡されます。
- スタックで渡される各々の引数は、4 バイトのアライメントに調整されます。

1.9 戻り値の設定規則

整数型、浮動小数点型、8 バイト以下の複合型は、r0レジスタを使って返されます。r0レジスタ (上位側) で足りない場合、r1 (下位側) が使用されます。
8 バイトを越える戻り値を返す場合、まず関数呼び出し側で、戻り値を設定してもらう領域へのポインタを 'invisible'な第 1 引数として r0レジスタに渡します。そして、呼び出された側でこのポインタを使って戻り値を設定するとともに、ポインタを r0レジスタに返します。

1.10 asm 関数

1.10.1 asm 関数の概要

asm 関数を使用することにより、C 言語の関数中にアセンブリ言語の命令を埋め込むことができます。記述するには、asm関数 ('asm'キーワード または '__asm__'キーワード) を用いて記述します。[1]

また、'asm'関数では、C 言語の式をアセンブラ命令のオペランドとして指定することもできます。このため、C 言語のデータがどのレジスタに格納されているか、メモリ中のどこに配置されているかを考慮する必要はありません。

1.10.2 asm 関数の記述形式

asm関数の記述形式は、次の通りです。

[1] コンパイル時に '-ansi'オプションを指定している場合、'asm'キーワードを使用することはできません。この場合には、'__asm__'キーワードを使用します。

```
asm( "template"
     : "output_constraint" (output_expression), ...
     : "input_constraint" (input_expression), ...
     : "regname", ... );
```

':'は、asm関数の各パラメータの区切り文字です。各パラメータの意味を以下に説明します。

template　*template* にはアセンブラ命令を記述します[2]。
template は必ずダブルクォート (") で囲んで記述します。

アセンブラ命令のオペランドに C 言語の式を指定するには、C 言語の式を *output_expression* および *input_expression* に記述し、それらを%*n*(*n* は、0 から始まる数字[3]) の形式を使って *template* 中に指定します。簡単な例を以下に示します。

```
asm( "sub %0, %1" : "+r" (x) : "r" (y + 1) );
```

この例では、%0 が *x* に対応し、%1 が *y*+1 に対応します。

output_constraint
output_expression

output_expression には C 言語の式を記述します。
式[4] は、アセンブラ命令において、'代入のみ'または '参照と代入'に使用される式 (以後、出力オペランドといいます) を記述します。*output_expression* の記述は必ずカッコ '()'で囲んで記述します。

output_constraint には、*output_expression* で指定した式がアセンブラ命令のオペランドとしてどのように表現されるかを指定する束縛条件と条件修飾子を記述します。

束縛条件には、'オペランドが必ずレジスタでなければならない'、'オペランドが必ず符号付き 8 ビット整数でなければならない'などの条件を 'r'や 'I'などの文字を使って指定します。束縛条件を指定する文字の一覧を以下に示します。[5]。

r	汎用レジスタでなければならない。
I	符号付き 8 ビット整数でなければならない。
J	符号付き 16 ビット整数でなければならない。
K	符号なし 16 ビット整数でなければならない。
M	符号なし 24 ビット整数でなければならない。

[2] これをアセンブラ命令のテンプレートコードと呼びます。
[3] *output_expression* に指定された式から順に、%0, %1, %2,... と割り当てられます。
[4] ここでの式とは、単に変数などを意味します。*x* + 1 のような式に対して代入することはできません。
[5] ここにあげた条件文字全てが、出力オペランドまたは入力オペランドの束縛条件として指定可能な訳ではありません。適切な条件文字を指定しなければなりません。

`I`	符号なし 5 ビット整数でなければならない (シフト命令時のシフトカウント)。
`J`	24 ビットで扱うことのできるシンボルアドレスでなければならない。
`0,1,...,9`	指定した番号の C 言語式と同じでなければならない。

条件修飾子には、‘式が代入のみされる’、‘式が参照と代入される’などの指定を、‘`=`’や ‘`+`’などの文字を使って記述します。条件修飾子を指定する文字の一覧を以下に示します。

`=`	式は、代入のみされる。参照はされない。
`+`	式は、参照および代入される。

output_constraint の記述は、束縛条件および修飾子の文字をダブルクォート (") で囲んで、`"=r"`や`"+r"`のように記述します。

出力オペランドが複数ある場合には、*output_constraint* と *output_expression* の組をカンマ (‘`,`’) で区切って記述します。
出力オペランドがない場合、*output_constraint* と *output_expression* は省略できます。

input_constraint
input_expression

input_expression には C 言語の式を記述します。
式は、アセンブラ命令の実行において参照される式 (以後、入力オペランドといいます) を記述します。*input_expression* は必ずカッコ ‘`()`’で囲んで記述します。

input_constraint には、*input_expression* で指定した式が、アセンブラ命令のオペランドとしてどのように表現されるかを指定する束縛条件と条件修飾子を指定します。使用できる束縛条件の文字一覧、および、条件修飾子の文字一覧は、*output_constraint* の項を参照してください。

input_constraint の記述は、束縛条件をダブルクォート (") で囲んで、`"r"`や`"J"`のように記述します。

入力オペランドが複数ある場合には、*input_constraint* と *input_expression* の組をカンマ (‘`,`’) で区切って記述します。
入力オペランドがない場合、*input_constraint* と *input_expression* は省略できます。

regname

regname にはアセンブラ命令実行の副作用により、破壊される可能性のあるレジスタを記述します。
regname は必ずダブルクォート (") で囲んで、`"r0"`や`"r14"`のように記述します。
例えば、"r14"レジスタが破壊される命令の場合には、次のように記述します。

```
asm("bl foo" : : : "r14");
```

条件ビットが破壊される可能性がある場合には、レジスタ名として"cbit"を指定します。

```
asm("addv %0, %1" : "+r" (x) : "r" (y) : "cbit");
```

複数のレジスタ名を指定する場合には、カンマ (,) で区切って"r0","r1"のように記述します。
破壊される可能性があるレジスタがない場合、*regname* は省略できます。

1.10.3 asm 関数の記述例

asm関数を記述する簡単な例として、次のような関数を用いて説明します。

```
int foo(int x, int y)
{
      asm("add %0, %1" : "+r" (x) : "r" (y));
      return x;
}
```

この関数は、引数 x および引数 y の足し算を asm関数で add命令を用いて計算し、その結果を関数の戻り値として返すというものです。これは最終的な asm関数の記述例ですが、この記述に至るまでを順を追って説明します。

まず初めに、最も簡単な足し算の例として、レジスタ名 (例えば、r0および r1) を直接指定する add命令の記述を示します。

```
asm("add r0, r1");
```

この add命令では、r0 = r0 + r1の計算を行います。今回の例では、r0を x、r1を y とし、x = x + y の計算を行うと考えます。この場合、x は入力オペランドおよび出力オペランドに相当し、y は入力オペランドに相当することになります。したがって、asm関数の記述としては、次のような記述イメージになります。

```
asm("add %0, %2"
    : output_constraint (x)
    : input_constraint (x), input_constraint (y) );
```

次に束縛条件を指定します。add命令では、オペランドには汎用レジスタしか指定できません。したがって、‘%0’および‘%2’に相当する出力オペランド x と入力オペランド y は、汎用レジスタであることを示す"r"を指定します。また、‘%0’である出力オペランド x は、結果を x に代入する必要があるので、代入を示す"="の条件修飾子も指定します。この結果、次のような記述イメージになります。

```
asm("add %0, %2"
    : "=r" (x)
    : input_constraint (x), "r" (y) );
```

入力オペランド x の束縛条件については、出力オペランドと同じレジスタでなければならないため、出力オペランド x と同じであることを示す"0"(式の 0 番目) を指定します。
出力オペランド x がレジスタ"r"であるため、入力オペランド x も"r"で良いのではないかと考えられますが、"r"と指定しただけでは同じレジスタが割り付けられるとは限らないため、必ず"0"と指定する必要があります。この結果、asm関数の記述は次のようになり、目的とする計算ができます。

```
asm("add %0, %2"
    : "=r" (x)
    : "0"  (x), "r" (y) );
```

さて、この記述では出力オペランド x の条件修飾子に"=" を指定しているため、入力オペランド x の指定を省略することができませんでした。しかし、x は入力および出力オペランドであることから、出力オペランド x の条件修飾子を"+"にすることにより、同じ式 x の入力オペランドの式を省略することができます[6] 。

```
int foo(int x, int y)
{
      asm("add %0, %1" : "+r" (x) : "r"  (y) );
      return x;
}
```

このように記述することで $x = x + y$ が計算され、計算結果 x を戻り値として返すことができます。

[6] 入力オペランド x を省略すると式の数が減るので、"add %0, %2"から"add %0, %1"に変わることに注意してください。

1.10.4 asm 関数使用時の注意事項

asm 関数を記述する場合、次の点に注意してください。

正しい命令コードの記述
asm 関数に記述された命令コードはそのままアセンブラに渡され、コンパイラでは記述された命令が正しいかどうかはチェックしません。したがって、asm 関数内の命令コードの記述は、アセンブリ言語の仕様に従って正しく記述してください。

生成されたコードの確認
asm 関数の記述の仕方やコンパイラの最適化の影響により、期待通りのコードが生成されない場合があります。したがって、コンパイル後に期待通りの正しいコードが生成されているかどうかを必ず確認してください。

1.11 Funding Free Software

If you want to have more free software a few years from now, it makes sense for you to help encourage people to contribute funds for its development. The most effective approach known is to encourage commercial redistributors to donate.

Users of free software systems can boost the pace of development by encouraging for-a-fee distributors to donate part of their selling price to free software developers—the Free Software Foundation, and others.

The way to convince distributors to do this is to demand it and expect it from them. So when you compare distributors, judge them partly by how much they give to free software development. Show distributors they must compete to be the one who gives the most.

To make this approach work, you must insist on numbers that you can compare, such as, “We will donate ten dollars to the Frobnitz project for each disk sold.” Don’t be satisfied with a vague promise, such as “A portion of the profits are donated,” since it doesn’t give a basis for comparison.

Even a precise fraction “of the profits from this disk” is not very meaningful, since creative accounting and unrelated business decisions can greatly alter what fraction of the sales price counts as profit. If the price you pay is $50, ten percent of the profit is probably less than a dollar; it might be a few cents, or nothing at all.

Some redistributors do development work themselves. This is useful too; but to keep everyone honest, you need to inquire how much they do, and what kind. Some kinds of development make much more long-term difference than others. For example, maintaining a separate version of a program contributes very little; maintaining the standard version of a program for the whole community contributes much. Easy new ports contribute little, since someone else would surely do them; difficult ports such as adding a new CPU to the GNU C compiler contribute more; major new features or packages contribute the most.

By establishing the idea that supporting further development is “the proper thing to do” when distributing free software for a fee, we can assure a steady flow of resources into making more free software.

1.12 GNU GENERAL PUBLIC LICENSE

Version 2, June 1991

1.12.1 Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

1.12.2 TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program " means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

   Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

   You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

   a. You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

   b. You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

   c. If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

   These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

   Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

   a. Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

   b. Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

   c. Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

   The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

   If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit

royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.
9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

   Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.
10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.
12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR

CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

1.12.3 How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

> *one line to give the program's name and a brief idea of what it does.*
> Copyright (C) 19*yy* *name of author*
>
> This program is free software; you can redistribute it and/or modify
> it under the terms of the GNU General Public License as published by
> the Free Software Foundation; either version 2 of the License, or
> (at your option) any later version.
>
> This program is distributed in the hope that it will be useful,
> but WITHOUT ANY WARRANTY; without even the implied warranty of
> MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
> GNU General Public License for more details.
>
> You should have received a copy of the GNU General Public License
> along with this program; if not, write to the Free Software
> Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

> Gnomovision version 69, Copyright (C) 19*yy* *name of author*
> Gnomovision comes with ABSOLUTELY NO WARRANTY; for details
> type 'show w'.
> This is free software, and you are welcome to redistribute it
> under certain conditions; type 'show c' for details.

The hypothetical commands '`show w`' and '`show c`' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than '`show w`' and '`show c`'; they could even be mouse-clicks or menu items—whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

> Yoyodyne, Inc., hereby disclaims all copyright interest in the program
> 'Gnomovision' (which makes passes at compilers) written by James Hacker.
>
> *signature of Ty Coon*, 1 April 1989

```
Ty Coon, President of Vice
```

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

2 as

Using as

The GNU Assembler

for the M32R family

Dean Elsner, Jay Fenlason & friends

Edited by Cygnus Support

The Free Software Foundation Inc. thanks The Nice Computer Company of Australia for loaning Dean Elsner to write the first (Vax) version of `as` for Project GNU. The proprietors, management and staff of TNCCA thank FSF for distracting the boss while they got some work

2.1 as オプション

```
as [ -a[cdhlmns][=file] ] [ -D ]  [ --defsym sym=val ]
   [ -f ] [ --gstabs ] [ --help ] [ -I dir ] [ -J ] [ -K ] [ -L ]
   [ --keep-locals ] [ -o objfile ] [ -R ] [ --statistics ]
   [ --strip-local-absolute ] [ -v ]
   [ -version ] [ --version ] [ -W ] [ -w ] [ -x ] [ -Z ]
   [ -m32rx | --[no-]warn-explicit-parallel-conflicts | --W[n]p ]
   [ -- | files ... ]
```

`@`*cmdfile* *cmdfile* で指定されたファイルの内容を、空白文字 (改行、タブ含む) で区切られた引数としてコマンドラインに展開してコマンドを実行します。この機能は、Windows 環境でのみ有効です。

`-a[cdhlmns]`
次のような各種様式のリスティングファイルの生成機能を有効にします。

`-ac` 偽判定の記述 (false conditionals) を省略します。

`-ad` デバッグ用の疑似命令を省略します。

`-ah` high-level ソースを含めます。

`-al` アセンブリ言語ソースを含めます。

`-am` マクロ展開を含めます。

`-an` 書式処理を省略します。

`-as` シンボルを含めます。

`=file` リスティングファイル名を '`file`'にします。

これらのオプションは、組み合わせて指定することができます。例えば、書式処理を省略したリスティングファイルを生成するには '`-aln`'を指定します。'`=file`' 指定は、一番最後に指定しなければなりません。'`-a`'のみの指定は、'`-ahls`'と同じ動作になります。

`-D` 無視されます。このオプションは、他のアセンブラとの互換性のためだけに受け付けます。

`--defsym` *sym*`=`*value*
入力ファイルをアセンブルする前に、シンボル *sym* を *value* に定義します。*value* は、1 つの整数定数でなければなりません。C 言語と同様に、'`0x`'で始まるならば 16 進数、'`0`'で始まるならば 8 進数と解釈します。

`-f` 高速処理します。これは、ソースファイルがコンパイラから生成されたものであると仮定し、空白文字やコメントの処理を省略します。

`--gstabs` 各アセンブラ行に対する `stabs`デバッグ情報を生成します。デバッガがこの情報を扱うことができれば、アセンブラコード のデバッグの助けになるでしょう。

`--help` コマンド行オプションの要約を表示します。表示のみ行い終了します。

`-I` *dir* `.include`疑似命令の検索リストにディレクトリ *dir* を追加します。

`-J` 符号付き数値がオーバーフローした場合にもワーニングを出力しません。

`-K` 距離差分テーブル (difference table) が、long displacement のために変更される場合にワーニングを出力します。

`-L`
`--keep-locals`
ローカルシンボルをシンボルテーブルに保持します。a.out システムではこれらのローカルシンボルは、'`L`'で始まりますが、異なるシステムでは違う文字 (local label prefixes) になります。

`-o` *objfile* 生成されるオブジェクトファイル名を *objfile* にします。

`-R` data セクションを text セクションに含めます。

`--statistics`
アセンブル処理に要した最大メモリ使用量 (バイト) および総時間 (秒) を表示します。

`--strip-local-absolute`
ローカルな絶対シンボル (local absolute symbols) を、出力するシンボルテーブルから削除します。

`-v`
`-version` `as`のバージョン情報を表示します。

`--version`
`as`のバージョン情報を表示して終了します。

`-W` ワーニングメッセージを抑止します。

`-w` 無視されます。

`-x` 無視されます。

`-Z` エラーが発生してもオブジェクトファイルを生成します。

`--` | *files* ...
アセンブル対象のプログラムを標準入力から入力 (‘`--`’指定時) 、または、*files* で指定したソースファイルから入力 (*files* ...) します。

`-warn-unmatched-high`
`-Wuh` ‘`high`’(または ‘`shigh`’) と ‘`low`’のリロケーション情報の矛盾に対してワーニングメッセージを出力します。

`-no-warn-unmatched-high`
`-Wnuh` ‘`high`’(または ‘`shigh`’) と ‘`low`’のリロケーション情報の矛盾に対してワーニングメッセージを出力しません。

`-m32rx` M32Rx コア CPU がターゲットであることを指定します。デフォルトは、M32R コア用ですが、このオプションは、M32Rx コア用であることを指定します。このオプションを指定することにより M32Rx コア CPU の命令セットを使用することができます。

`-warn-explicit-parallel-conflicts`
`-Wp` 疑わしい並列実行 (オペランド干渉の発生するような並列実行) に遭遇した場合に、ワーニングメッセージを出力します。

`-no-warn-explicit-parallel-conflicts`
`-Wnp` 疑わしい並列実行 (オペランド干渉の発生するような並列実行) に遭遇した場合にも、ワーニングメッセージを出力しません。

2.2 コメント

行のコメント記述には、セミコロン (`;`) を使用することができます。
また、C 言語風に`/*`から`*/`に囲まれた範囲は、コメントとして扱われます。C 言語風コメントのネストはできません。

2.3 シンボル

シンボルは、1 つ以上の文字から構成されます。シンボルには、数字、アンダースコア (`_`)、ドル (`$`)、ドット (`.`) を使用することができます。但し、数字から始めることはできません。大文字と小文字は区別されます。

2.4 文

一つの文は、改行コード (`\n`) で終わります。ラベルを記述する場合、シンボルのすぐ後ろにコロン (`:`) を付けなければなりません。ドット (`.`) から始まるシンボルは、疑似命令として扱われます。

2.5 定数

2.5.1 文字列定数

文字列定数は、ダブルクォート (") で囲んで記述します。

2.5.2 文字定数

文字定数は、シングルクォート (') で囲んで記述します。

2.5.3 整数定数

バイナリ整数は '0b' または '0B' を先頭につけて、'01' からなる 1 つ以上の数字で記述します。
8 進数整数は '0' を先頭につけて、'01234567' からなる 1 つ以上の数字で記述します。
10 進数整数は '0' 以外の数字から始まり、'0123456789' からなる 1 つ以上の数字で記述します。
16 進数整数は '0x' または '0X' を先頭につけて、'0123456789abcdefABCDEF' からなる 1 つ以上の 16 進数の数字で記述します。

2.6 レジスタ

次のレジスタ名を使用できます。

汎用レジスタ名	r0, r1, r2, r3, r4, r5, r6, r7, r8, r9, r10, r11, r12, r13(fp), r14(lr), r15(sp)
コントロールレジスタ	cr0から cr15
プロセッサ状態語レジスタ	pswまたは cr0
条件ビットレジスタ	cbrまたは cr1
割り込み用スタックポインタ	spiまたは cr2
ユーザ用スタックポインタ	spuまたは cr3
バックアップ PC	bpcまたは cr6
アキュムレータ	a0, a1

2.7 オペランド

次のアドレッシングモードを指定することができます。

Rn	レジスタ直接
@Rn	レジスタ間接

@(disp,Rn)	レジスタ相対間接
Rn+	レジスタ間接 + レジスタ更新
+Rn	レジスタ間接 + レジスタ更新
-Rn	レジスタ間接 + レジスタ更新
#imm	イミディエート
pcdisp	PC 相対

2.8 命令

2.8.1 オペランド サイズおよびディスプレースメント サイズ指定のある命令

オペランドサイズおよびディスプレースメントサイズを指定することのできる命令は、次のように記述します。

bc.s label	8 ビットディスプレースメント
bc.l label	24 ビットディスプレースメント
bl.s label	8 ビットディスプレースメント
bl.l label	24 ビットディスプレースメント
bnc.s label	8 ビットディスプレースメント
bnc.l label	24 ビットディスプレースメント
bra.s label	8 ビットディスプレースメント
bra.l label	24 ビットディスプレースメント
ldi8 reg, #const	8 ビット定数
ldi16 reg, #const	16 ビット定数

2.8.2 スタック操作命令

スタック操作命令は、次のように記述することもできます。

push reg	reg のプッシュ。st reg, @-sp と同じ。
pop reg	reg のポップ。ld reg, @sp+と同じ。

2.9 並列実行命令の記述

ターゲット CPU として M32Rx コアを指定した場合 (オプションに-m32rx を指定した場合)、2 つの命令を並列実行することを指定することができます。2 つの命令を並列実行することを指定するには、次のように ‘||’文字を命令間に指定します。

```
mv r1, r2 || add r3, r4
```

‘mv r1,r2’命令と ‘add r3,r4’命令が並列に実行されます。

2.10 逐次実行命令の記述

ターゲット CPU として M32Rx コアを指定した場合 (オプションに-m32rx を指定した場合)、2 つの命令を逐次実行することを指定することができます。2 つの命令を逐次実行することを指定するには、次のように '->'文字を命令間に指定します。

```
mv r1, r2 -> add r3, r4
```

'->'により逐次実行することを指定した場合には、左側命令 'mv r1, r2' が実行された後、右側命令 'add r3, r4'が実行されます。

2.11 アセンブラマクロ

M32R の命令セットでは、1 命令で扱える即値は 24 ビットまでです。24 ビットを越える即値を操作するには、上位 16 ビットと下位 16 ビットの 2 命令に分けて取り扱います。アセンブラには、この操作を記述するためのマクロとして、'high'、'shigh'、'low'が用意されています。

これらのマクロがオペランドに指定できる命令は次の通りです。

マクロ名	命令
high	seth
shigh	seth
low	add3, or3, ld, ldb, ldh, ldub, lduh, st, stb, sth

マクロの記述規則を以下に示します。

マクロ名	記述形式と説明
high	seth Rdst,#high(*imm*) 即値 *imm* の上位 16 ビットのみが seth命令の即値オペランドとして扱われます。seth命令の実行によって、Rdst には *imm* の下位 16 ビットを 0 にした値が転送されます。
shigh	seth Rdst,#shigh(*imm*) 即値 *imm* の下位 16 ビットの符号拡張を補正する値を加えた、*imm* の上位 16 ビットを seth命令の即値オペランドとします。 つまり、*imm* の下位 16 ビットの最上位ビットが 1 の時には 1 が加算された値になり、最上位ビットが 0 の時にはそのままの値になります。

low

mnemonic Rdst,Rsrc,#low(*imm*)
(*mnemonic*:add3|or3)

即値 imm の下位 16 ビットのみが *mnemonic* 命令の即値オペランドとして扱われます。したがって、この命令より前に seth Rdst,#high(*imm*) のように high を使用して記述することで、Rdst に 32 ビットの即値が設定できます。

mnemonic Rdst,@(low(*imm*),Rsrc)
(*mnemonic*:ld|ldb|ldh|ldub|lduh|st|stb|sth)

imm の下位 16 ビットが *mnemonic* 命令のディスプレースメントとして扱われます。このディスプレースメントは符号拡張されます。したがって、この命令より前に、seth Rdst,#shigh(*imm*)のように shigh を使用して記述することで *imm* のメモリ内容に正しくアクセスすることができます。

以下に記述例を示します。

例 1: 32bit 即値をレジスタに格納する場合

```
seth r0, #high(imm_32)
or3  r0, r0, #low(imm_32)
```

例 2: 32bit で示されるアドレスのメモリ内容をロードする場合

```
seth r0, #shigh(imm_32)
ld   r0, @(low(imm_32),r0)
```

2.12 疑似命令

アセンブラ疑似命令の名前は、ピリオド (.) から始まり、残りが文字の形式になっています。

2.12.1 .align

```
.align alignment [, value]
```

ロケーションカウンタを指定された境界にします。*alignement* はアライメント境界値を示す値で、ロケーションカウンタの低ビット側の何ビット分を 0 にするかを指定します。したがって、'.align 3'を指定した場合、ロケーションカウンタは 8 の整数倍に調整されます。

value は、アライメント調整された領域に埋め込む値をバイトで指定します。*value* は省略できます。*value* を省略した場合、コードセクションでは NOP命令が埋め込まれ、データセクションでは 0が埋め込まれます。

2.12.2 .ascii

```
.ascii "string" [, ...]
```

文字列定数を定義します。複数の文字列をカンマ (,) で区切って指定することができます。
カンマ (,) で区切られた文字列定数は、連続したアドレスに配置されます。
各文字列定数の最後にゼロ (0) は付加されません。

2.12.3 .asciz

```
.asciz "string" [, ...]
```

文字列定数を定義します。複数の文字列をカンマ (,) で区切って指定することができます。
カンマ (,) で区切られた文字列定数は、連続したアドレス配置されます。
各文字列定数の最後にゼロ (0) が自動的に付加されます。

2.12.4 .balign

```
.balign  alignment [, value]
```

ロケーションカウンタを *alignment* で指定された境界にします。'.balign 4'を指定した場合、ロケーションカウンタは 4 の整数倍に調整されます。

value は、アライメント調整された領域に埋め込む値をバイトで指定します。*value* は省略できます。*value* を省略した場合、コードセクションでは NOP命令が埋め込まれ、データセクションでは 0が埋め込まれます。

2.12.5 .byte

```
.byte expressions [, ...]
```

expressions で指定した値を持つ、8 ビットの領域を定義します。複数定義するには、カンマ (,) で区切って指定します。

2.12.6 .comm

```
.comm symbol, length
```

symbol を.bss セクション内の common 領域として宣言します。少なくとも *length* バイトの領域が定義されます。
length は、絶対式でなければなりません。

2.12.7 .data

```
.data
```

この疑似命令以降に続く文をデータセクションの最後に置くことを指示します。

2.12.8 .equ

```
.equ symbol,expressions
```

symbol の値を *expressions* に宣言します。
‘.set’疑似命令と同意です。

2.12.9 .extern

```
.extern
```

.externは他のアセンブラとの互換性のために受け入れられますが、無視されます。アセンブラ asは、全ての未定義シンボルを外部参照シンボルとして扱います。

2.12.10 .global

```
.global symbol
```

`.global`疑似命令は、リンカ `ld`に *symbol* が参照できるようにします。プログラムの一部で`.global`を使って *symbol* を定義すると、リンク時に他のプログラムからその *symbol* を参照できるようになります。

2.12.11 .hword

```
.hword expressions [, ...]
```

expressions で指定した値を持つ、16 ビットの領域を定義します。複数定義するには、カンマ (`,`) で区切って指定します。

2.12.12 .section

```
.section name [, "flags" [, @type]]
```

この疑似命令以降に続く文を *name* で指定されたセクションに配置することを指示します。

flags には、次の文字を指定することができます。

a	セクションは配置可能である。
w	セクションは書き込み可能である。
x	セクションは実行可能である。

type には、次の文字を指定することができます。

`@`progbits	セクションは有効なデータを含んでいる。
`@`nobits	セクションは有効なデータを含んでいない (領域確保のみ)。

2.12.13 .set

```
.set symbol, expressions
```

symbol の値を *expressions* に宣言します。
‘`.equ`’疑似命令と同意です。

2.12.14 .space

```
.space size, fill
```

fill で指定された値を持つ *size* バイトの領域を定義します。

2.12.15 .text

```
.text
```

この疑似命令以降に続く文をテキストセクションの最後に置くことを指示します。

2.12.16 .word

```
.word expressions [, ...]
```

expressions で指定した値を持つ、32 ビットの領域を定義します。複数定義するには、カンマ (,) で区切って指定します。

3 ld

Using ld

The GNU linker

ld version 2

April 1998

Steve Chamberlain Ian Lance Taylor

Cygnus Solutions

Cygnus Solutions
ian@cygnus.com, doc@cygnus.com

Using LD, the GNU linker

Edited by Jeffrey Osier (jeffrey@cygnus.com)

3.1 ld オプション

`@`*cmdfile* *cmdfile* で指定されたファイルの内容を、空白文字 (改行、タブ含む) で区切られた引数としてコマンドラインに展開してコマンドを実行します。この機能は、Windows 環境でのみ有効です。

`-a`*keyword*
このオプションは、HP/UX との互換のためにサポートされています。*keyword* は、'`archive`', '`shared`', または '`default`' の内の 1 つでなければなりません。'`-aarchive`'は、機能的に '`-Bstatic`'と等価で、他の 2 つのキーワードは、'`-Bdynamic`'と等価です。このオプションは、何回指定されても構いません。

`-d`
`-dc`
`-dp` これらの 3 つのオプションは、等価な機能です。他のリンカとの互換性のために複数の形式がサポートされています。これらは、リロケータブルファイルの出力指定された時 ('`-r`') でさえ、領域を common シンボルへ割り当てます。スクリプトコマンドの `FORCE_COMMON_ALLOCATION`も同様の効果があります。

`-e` *entry*
`--entry=`*entry*
プログラムの実行開始シンボルとして、*entry* を使用します。

`--force-exe-suffix`
出力ファイルの拡張子を .exe にします。

`-g` 無視します。他のツールとの互換性のために提供されています。

`-i` インクリメンタルリンクを実施します ('`-r`'オプションと同じです)。

`-l`*archive*
`--library=`*archive*
archive で指定されたアーカイブファイルを、リンクするファイルのリストに追加します。このオプションは、複数回指定することができます。`ld`は、アーカイブのパスリストを検査し、指定された全ての *archive* に対する `lib`*archive*`.a`という名前のファイルを探します。

リンカは、コマンド行で指定された場所で一度だけアーカイブを探します。もし、コマンド行のアーカイブ指定の前に現れたオブジェクト内に、未定義のシンボルがあり、そのシンボルがアーカイブ内で定義されていた場合、リンカは、そのアーカイブから適切なファイルをインクルードします。しかし、コマンド行のアーカイブ指定より後ろに現れたオブジェクト内の未定義シンボルに対して、再度アーカイブを検索することはありません。アーカイブを強制的に複数回検索する方法については、`-(`オプションを参照してください。

コマンド行に複数回、同じアーカイブを指定しても構いません。

このアーカイブの検索方法は、Unix linkers では標準的です。しかし、他のシステム上のリンカの振る舞いと異なる場合があるので、注意してください (AIX のリンカなど)。

`-L`*searchdir*
`--library-path=`*searchdir*
`ld`が、アーカイブライブラリを検索するためと `ld` control scripts を探すために参照するパスのリストに *searchdir* で指定されたパスを追加します。このオプションは、複数回指定することができます。ディレクトリは、コマンド行で指定された順番で検索されます。コマンド行で指定されたディレクトリは、デフォルトのディレクトリより先に検索されます。全ての`-L`オプションは、オプションが指定された順番に関係なく、全ての`-l`オプションの対象になります。

パスは、linker script 内で SEARCH_DIRコマンドを使って指定することもできます。この方法で指定されたディレクトリは、コマンド行に linker script が指定されたポイントで検索されます。

`-M`
`--print-map`
: 標準出力へリンクマップを表示します。リンクマップは、次のようなリンクに関する情報を提供します。
  - オブジェクトファイルとシンボルが、メモリ内にマップされた場所
  - どのように common シンボルが割り付けられたか
  - リンク時にインクルードされた全てのアーカイブメンバと、アーカイブメンバのインクルードを引き起こしたシンボル

`-n`
`--nmagic`
: text セグメントを読み出し専用に設定します。また、可能ならば、NMAGICとして出力をマークします。

`-N`
`--omagic`
: text と data セクションを、読み出し/書き込み可能に設定します。また、data セグメントのページ調整を実施しません。出力フォーマットが、Unix スタイルの magic 番号をサポートしている場合、OMAGICとして出力をマークします。

`-o` *output*
`--output=`*output*
: ldによって生成されるプログラムの名前を *output* にします。このオプションが指定されない場合、デフォルトで‘a.out’という名前になります。script command の OUTPUTでも、出力ファイル名を指定することができます。

`-r`
`--relocateable`
: リロケータブルな出力を生成します (例えば、ldへ再入力することができる出力ファイルなど)。これは、*partial linking* と呼ばれることがあります。副作用として、標準の Unix magic 番号をサポートしている環境においては、出力ファイルの magic 番号を OMAGICに設定します。

  このオプションが指定されていない場合、アブソリュートファイルが生成されます。C++プログラムをリンクする時、このオプションはコンストラクタへの参照を解決しません。この目的には、‘-Ur’ を使用します。

  このオプションは、‘-i’と同様の動作をします。

`-R` *filename*
`--just-symbols=`*filename*
: シンボル名とそれらのアドレスを *filename* で指定されたファイルから読み込みます。ただし、それらを再配置や出力に含めたりはしません。他のプログラム内で定義されたアブソリュートなメモリ位置をシンボル名を使用して参照できるようにします。このオプションは、複数回指定することができます。

  他の ELF リンカとの互換のために、-Rオプションにファイル名でなくディレクトリ名が指定されたならば、-rpathオプションとして扱われます。

`-s`
`--strip-all`
: 出力ファイルから、全てのシンボル情報を取り除きます。

`-S`
`--strip-debug`
: 出力ファイルから、デバッガ用シンボルの情報 (全てのシンボルではない) を取り除きます。

`-t`
`--trace` `ld`が処理中の入力ファイル名を表示します。

`-T` *scriptfile*
`--script=`*scriptfile*
linker script として *scriptfile* を使用します。この script は、`ld`のデフォルトの linker script を置き換えます (デフォルトの linker script に追加するのではなく)。そのため、*scriptfile* には、出力ファイルのために必要な全ての記述がされていなければなりません。`SECTIONS`や`MEMORY`コマンドのような、linker script 内に一度だけ指定するようなコマンドを使用したい場合、このオプションを使う必要があります。*scriptfile* がカレントディレクトリに存在しない場合、`ld`は、それ以前に '`-L`'オプションで指定されたディレクトリ内を探します。

`-u` *symbol*
`--undefined=`*symbol*
未定義シンボルとして、出力ファイル内に含まれるべきシンボルを *symbol* で指定します。この指定は、例えば、標準ライブラリから追加するモジュールをリンクするためのトリガとして使用することができます。'`-u`'は、追加する未定義シンボルを繰り返して指定することができます。このオプションは、リンカコマンドの`EXTERN`と等価です。

`-v`
`--version`
`-V` `ld`のバージョン番号を表示します。`-V`オプションは、サポートしているエミュレーションもリスト表示します。

`-x`
`--discard-all`
全てのローカルシンボルを削除します。

`-X`
`--discard-locals`
全てのテンポラリ・ローカルシンボルを削除します。ほとんどのターゲットでは、これは '`L`'で始まる名前の全てのローカルシンボルになります。

`-y` *symbol*
`--trace-symbol=`*symbol*
リンクされたファイルの内、*symbol* で指定したシンボルが現れるファイル名を表示します。このオプションは、複数回指定することができます。多くのシステムでは、アンダースコアを付加する必要があります。

このオプションは、未定義シンボルがどこで定義されているのか知りたい場合に役に立ちます。

`-Y` *path* デフォルトのライブラリ検索パスに *path* で指定したディレクトリパスを追加します。このオプションは、Solaris との互換性のためにあります。

`-z` *keyword*
Solaris 互換のためのものであり、無視されます。

`-(` *archives* `-)`
`--start-group` *archives* `--end-group`
archives には、アーカイブファイルのリストを指定します。それらには、ファイル名でも '`-l`'オプションでも指定することができます。

指定したライブラリは、新たな未定義参照がなくなるまで、繰り返し検索されます。

通常リンカは、コマンド行で指定された場所で一度だけアーカイブを探します。もし、コマンド行のアーカイブ指定の前に現れたオブジェクト内に、未定義のシ

ンボルがあり、そのシンボルがアーカイブ内で定義されていた場合、リンカは、そのアーカイブから適切なファイルをインクルードします。しかし、コマンド行のアーカイブ指定より後ろに現れたオブジェクト内の未定義シンボルに対して、再度アーカイブを検索することはありません。アーカイブをグルーピングすることにより、全ての参照が解決するまで繰り返し検索させることができます。

`-assert` *keyword*
SunOS 互換のためのものであり、無視されます。

`--cref` クロスリファレンステーブルを出力します。リンクマップファイルが生成されているならば、クロスリファレンステーブルは、マップファイルに表示されます。そうでなければ、標準出力へ表示されます。

`--check-sections`
セクションのオーバーラップをチェックします (デフォルト)。セクションのオーバーラップがある場合、エラーを表示します。

`--no-check-sections`
セクションのオーバーラップをチェックしません。オーバーレイ機能を使用する場合に用います。

`--defsym` *symbol*=*expression*
expression によって与えられた absolute アドレスを含むグローバルシンボルを、出力ファイルに生成します。このオプションは、コマンド行に複数回指定することができます。

expression のために限定された算術記述がサポートされています。16 進数の定数、存在するシンボルの名前、定数やシンボルの加算/減算のための+および-を使用することができます。さらに凝った式を記述したい場合には、linker script からリンカコマンド言語を使用することもできます。

symbol と '=' と *expression* 間にスペースを入れてはいけません。

`--help` コマンド行オプションの要約を、標準出力に表示して終了します。

`-Map` *mapfile*
リンクマップを *mapfile* ファイルに出力します。'`-M`'オプションの解説も参照してください。

`--no-keep-memory`
通常、`ld`は、メモリの使用に関しては、メモリ内に入力ファイルのシンボルテーブルをキャッシングすることにより、リンク速度の最適化を行います。このオプションは、`ld`にメモリの使用に関する最適化の変わりに、必要ならばシンボルテーブルを再読み込みすることを指示します。サイズの大きな実行モジュールをリンクする際、メモリ不足になるような場合に、このオプションを指定する必要があるかもしれません。

`--no-whole-archive`
`--whole-archive`の効果を無効にします。

`--noinhibit-exec`
エラーが発生した時にも、出力ファイルを生成します。

`--retain-symbols-file` *filename*
filename のファイル内にリストされたシンボルのみ残し、他の全てのシンボルを取り除きます。*filename* には、1 行に 1 つのシンボルのみ記述します。

未定義のシンボルや、リロケーションのために必要なシンボルは、取り除かれません。コマンド行に一度だけ指定することができます。また、'`-s`'と'`-S`'オプションを無効にします。

`--sort-common`
: common シンボルを出力セクションに配置する際、サイズによりソートします。最初に全ての 1 バイトシンボルが配置され、次に 2 バイト、3 バイトのような順になります。これは、アライメントの強制のために、シンボル間のギャップを防ぐために使用します。

`--stats`
: リンカの実行時間/使用メモリ量などの統計情報を、計算して表示します。

`--verbose`
: `ld`のバージョン番号とサポートしているエミュレーションのリストを表示します。オープンすることができた入力ファイル、オープンできなかった入力ファイルも表示します。デフォルトの組み込み script を使用している場合、その linker script も表示します。

`--warn-once`
: 各々の未定義シンボルに対するワーニングを、モジュール毎ではなく、一度だけ表示します。

`--warn-section-align`
: 出力セクションのアドレスが、アライメントのために変更された場合にワーニング表示します。

`--whole-archive`
: コマンド行で、`--whole-archive`オプションより後ろに指定された、全てのアーカイブファイルの全てのオブジェクトファイルを、出力ファイルに含めます。

`--wrap` *symbol*
: *symbol* の代わりに wrapper 関数を使用します。*symbol* への未定義参照は、`__wrap_`*symbol*への参照として解決されます。また、`__real_`*symbol*への未定義参照が、*symbol* への参照として解決されます。

  これは、システム関数の代わりに、wrapper を提供するために使用することができます。wrapper 関数は、`__wrap_`*symbol*と見なされるべきです。もしシステム関数するを呼び出したいならば、`__real_`*symbol*を呼び出すべきです。

  以下に簡単な例を示します。

```
void *
__wrap_malloc (int c)
{
  printf ("malloc called with %ld\n", c);
  return __real_malloc (c);
}
```

  このファイルと他のコードを、`--wrap malloc`を使用してリンクすると、`malloc`の呼び出しは、`__wrap_malloc`を呼び出し、`__wrap_malloc`内での`__read_malloc`呼び出しは、本物の`malloc`関数を呼び出すようになります。

4 The GNU Binary Utilities

The GNU Binary Utilities

Version 2.9-gnupro-99r1p1

May 1993

Roland H.Pesch Jeffrey M. Osier Cygnus Support

Cygnus Support

TEXinfo 2.227

4.1 ar

`ar [-]`*p*`[`*mod* `[`*relpos*`]] `*archive* `[`*member*`...]`

`ar`を Unix Style で使用する場合、`ar`実行時に少なくとも二つの引数を指定しなければなりません。一つは、*operation* を指定する keyletter であり (*modifiers* を指定する他の keyletters を付加することも選択できます)、もう一つは実行される archive 名です。

ほとんどの操作は、処理される特定のファイルを指定するための *member* 引数を受け付けることもできます。

GNU `ar`は、一番最初のコマンド行引数内で、operation code *p* と modifier flags *mod* を順不同で指定することを許可します。

必要ならば、一番最初のコマンド行引数を dash`-`付きで始めることができます。

p keyletter には、どんな操作を実行するかを指定します。次のどれかを指定することができますが、必ず一つだけ指定しなければなりません。

`@`*cmdfile* *cmdfile* で指定されたファイルの内容を、空白文字 (改行、タブ含む) で区切られた引数としてコマンドラインに展開してコマンドを実行します。この機能は、Windows 環境でのみ有効です。

`d` archive から指定されたモジュールを削除します。*member*... として削除されるべきモジュールの名前を指定します。もし削除されるべきファイルが指定されていない場合、その archive は変更されません。

'`v`' modifier を指定した場合、`ar`は、モジュールが削除された時にそのモジュールをリスト表示します。

`m` 一つの archive 内で members を移動するために使用します。

シンボルが一つ以上の member 内に定義されているならば、archive 内の members の順番は、ライブラリ内で使用されているプログラムがどのようにリンクされるかという違いを生みます。

`m`と共に modifiers を何も指定していなければ、*member* 引数に指定した members が archive の最後に移動されます。位置を指定する代わりに、'`a`', '`b`', '`i`'modifiers を指定することもできます。

`p` archive 内の指定された members を標準出力へ出力します。'`v`'modifier が指定されている場合、標準出力へその内容をコピーする前に member 名を表示します。

member 引数を指定していない場合、archive 内の全てのファイルを出力します。

`q` 高速追加指定。*member*... に指定されたファイルを *archive* の最後に追加します。置換のためのチェックは行いません。

'`a`', '`b`', '`i`' modifiers は、この操作に影響しません。新しい members は、常に archive の最後に配置されます。

'`v`' modifier は、`ar`が各々のファイルを追加した時に各々のファイル名をリスト表示することを指定します。

この操作のポイントは速度であるため、archive のシンボル・テーブル・インデックスは存在していても更新されません。シンボル・テーブル・インデックスを明確に更新するには、'`ar s`'または`ranlib`を使用してください。

`r` *member*...で指定したファイルを *archive* に挿入します (置換機能付き)。この操作が 'q'と異なる点は、追加される member 名と一致する member が archive 内に存在する場合、以前の member が削除されることです。

member...に指定されたファイルの一つが存在しない場合、`ar` はエラーメッセージを表示し、その名前に一致する archive 内の members には何も影響なく終了します。

デフォルトでは、新しい members はファイルの最後に追加されますが、既に存在する member に対する相対位置を指定する 'a', 'b', 'i' modifiers の一つを指定することもできます。

この操作に 'v'を指定することにより、挿入される各々のファイルに対する一行の情報を出力します。ファイルが追加された (削除されるべき古い member は存在しない) ことを意味する 'a'文字、または、置換されたことを意味する 'r'文字が出力されます。

`t` *archive* の内容を並べたテーブルを表示します。あるいは、archive 内に存在する *member*...に指定されたファイルのテーブルを表示します。通常、member 名のみ表示します。モード (permissions), タイムスタンプ, オーナー, グループ, サイズ情報を知りたい場合、'v' modifier を指定することにより表示することができます。

member が指定されていない場合、archive 内の全てのファイルを表示します。

archive 同じ名前 ('fie'とする) のファイルが archive('b.a'とする) 内に一つ以上存在する場合、'ar t b.a file'は、最初のもののみリスト表示します。すべてのものを表示するためには、'ar t b.a'のように指定しなければなりません。

`x` archive から、名前が *member* である member を抽出します。'v' modifier をこの操作時に指定することにより、抽出した時に各々の名前を表示することができます。

member を指定していない場合、archive 内の全てのファイルを抽出します。

各操作 (operation) の振る舞いを指定するために、modifiers(*mod*) を、*p* keyletter のすぐ後に指定することができます。

`a` archive 内に存在している member の後ろに、新しいファイルを追加します。'a' modifier を指定する場合、*archive* を指定する前に、*relpos* 引数として archive 内に存在する member 名を指定しなければなりません。

`b` archive 内に存在する member の前に、新しいファイルを追加します。'b' modifier を指定する場合、*archive* を指定する前に、*relpos* 引数として archive 内に存在する member 名を指定しなければなりません ('i'と同様)。

`c` archive を生成します。更新要求した場合、指定された *archive* が、存在しなければ常に生成されます。しかし、この modifier を指定することにより、前もって archive が生成されることを指定していなければ、ワーニングが表示されます。

`f` archive 内の名前を切り詰めます。通常、GNU `ar`は、どんな長さのファイル名も許しています。これは、いくつかのシステムのネイティブ `ar` との互換性で問題になります。これが問題になる場合、'f' modifier を指定することにより、archive 内にファイルを格納する時にファイル名を切り詰めることができます。

`i` archive 内に存在する member の前に新しいファイルを挿入します。'i' modifier を指定する場合、*archive* を指定する前に、*relpos* 引数として archive 内に存在する member 名を指定しなければなりません ('b'と同様)。

`l` この modifier は受け付けられますが、使用されません。

`o` members を抽出する際、member の元の日付を保持します。この modifier を指定していない場合、archive から抽出されたファイルのタイムスタンプは、抽出した時刻になります。

`s` archive に何も変更されていない場合でも、archive 内に object-file index を書き込みます。また、object-file index が存在している場合にも更新します。他の操作時、あるいは、単独でこの modifier を使用することができます。archive に対して 'ar s'を実行することは、'ranlib'を実行するのと等価です。

`S` archive のシンボルテーブルを作成しません。この modifier 指定は、大きなライブラリファイルをいくつかの段階に分けて構築する際に、作成速度を向上することができます。この modifier を指定して作成されたライブラリは、リンカで使用することはできません。したがって、シンボルテーブルを作成するために、'ar' の最後の段階でこの modifier の指定を外すか、あるいは、archive に対して 'ranlib' を実行しなければなりません。

`u` 通常、'ar r'...は、リストされた全てのファイルを archive 内に挿入します。もしリストされたファイルの内、同じ名前の既に存在している member よりも新しいものだけを挿入したい場合に、この modifier を使用します。'u' modifier は、'r'(replace) operation の時のみ指定することができます。特に、'qu'では、使用することはできません。高速追加指定の 'q' では、タイムスタンプをチェックすることになり時間をとられてしまうからです。

`v` 操作時の冗長な情報を表示することを要求します。多くの操作では、'v' modifier を指定した場合に表示される、処理対象のファイル名のような追加情報を表示します。

`V` arのバージョン番号を表示します。

4.2 nm

```
nm [ -a | --debug-syms ]  [ -g | --extern-only ]
   [ -B ]  [ -C | --demangle ] [ -D | --dynamic ]
   [ -s | --print-armap ]  [ -A | -o | --print-file-name ]
   [ -n | -v | --numeric-sort ]  [ -p | --no-sort ]
   [ -r | --reverse-sort ]  [ --size-sort ] [ -u | --undefined-only ]
   [ -t radix | --radix=radix ] [ -P | --portability ]
   [ --target=bfdname ] [ -f format | --format=format ]
   [ --defined-only ] [-l | --line-numbers ]
   [ --no-demangle ] [ -V | --version ]  [ --help ]  [ objfile... ]
```

GNU nmは、*objfile*...に指定したオブジェクトファイルのシンボル情報をリスト表示します。オブジェクトファイルが引数に指定されていない場合、nmは、'a.out'が指定されたものとして処理します。

各々のシンボルに対して、nmは次の情報を表示します。

- シンボル値。後述のオプションにより指定された基数で、シンボル値を表示します。デフォルトでは、16 進数で表示します。
- シンボルタイプ。少なくとも次のタイプが使用されます。また、オブジェクトファイルフォーマットに依存して他のタイプも同様に使用されます。シンボルタイプが小文字である場合、そのシンボルはローカルなシンボルです。シンボルタイプが大文字である場合、そのシンボルはグローバル (external) なシンボルです。

`A` シンボルの値は、絶対値です。また、更にリンクしても値は変更されません。

`B` シンボルは、初期化されない data セクション (BSS として知られている) にあります。

`C` シンボルは、common です。common シンボルは、初期化されないデータです。リンク時に複数の common シンボルが同じ名前で現れるかもしれません。もしそのシンボルがどこにも定義されていなければ、その common シンボルは未定義の参照であるとして取り扱われます。

`D` シンボルは初期化されるデータセクションにあります。

`G` シンボルは、小さなオブジェクトの初期化された data セクションにあります。いくつかのオブジェクトフォーマットでは、大きなグローバル配列に対立するものとしてグローバルな整数型変数のような小さなデータオブジェクトに対する有効なアクセスを許可します。

`I` シンボルは、他のシンボルへの間接参照です。これは、まれに使用される a.out オブジェクトフォーマットの GNU の拡張機能です。

`N` シンボルは、デバッグ用シンボルです。

`R` シンボルは、read only data セクションにあります。

`S` シンボルは、小さなオブジェクトの初期化された data セクションにあります。

`T` シンボルは、text(code) セクションにあります。

`U` シンボルは定義されていません。

`W` シンボルは、weak です。weak に定義されたシンボルが、普通に定義されたシンボルとリンクされる場合、普通に定義されたシンボルは、エラーなしに使用されます。weak な未定義シンボルがリンクされ、そのシンボルが定義されていない場合、weak シンボルの値は、エラーなしにゼロ (0) になります。

`-` シンボルは、a.out オブジェクトファイル内の stabs シンボルです。この場合、表示されている値は、stabs other field、stabs desc field、stab type になります。stabs シンボルは、デバッグ情報を保持するために使用されます。

`?` シンボルタイプがわかりません。または、オブジェクトファイルフォーマット専用のものです。

- シンボルの名前。

long 形式と short 形式のオプションは、機能的に等価です。どちらかを選択することができます。

`@`*cmdfile* *cmdfile* で指定されたファイルの内容を、空白文字 (改行、タブ含む) で区切られた引数としてコマンドラインに展開してコマンドを実行します。この機能は、Windows 環境でのみ有効です。

`-A`
`-o`
`--print-file-name`
各々のシンボルの前に、そのシンボルが見つかった入力ファイルの名前 (または、archive element 名) を表示します。入力ファイルを 1 度だけ表示して同一とするのではなく、全てのシンボルの前に表示します。

`-a`
`--debug-syms`
全てのシンボルを表示します。通常は表示されないデバッグ用のシンボルも表示します。

`-B` ‘`--format=bsd`’と同じです。MIPS 用`nm`との互換用です。

`-C`
`--demangle`
アセンブリ言語のラベル (low-level シンボル) を、C++言語の関数名 (user-level の名前) に解読 (*demangle*) します。システムによって付加された最初のアンダースコアを取り除くほかに、C++言語の関数名を読めるようにします。

`--no-demangle`
アセンブリ言語ラベル (low-level シンボル) の名前を、C++言語の関数名に解読 (demangle) しません。デフォルトでは、この動作になります。

`-D`
`--dynamic`
normal シンボルではなく dynamic シンボルを表示します。これは、共有ライブラリのような dynamic オブジェクトに対してのみ有効です。

`-f` *format*
`--format=`*format*
出力フォーマットに *format* を使用します。これには、`bsd`、`sysv`、`posix`を指定することができます。デフォルトは`bsd`になります。*format* の一番最初の文字だけが認識されます。大文字、小文字のどちらでも構いません。

`-g`
`--extern-only`
external シンボルのみ表示します。

`-l`
`--line-numbers`
各々のシンボルに対して、ファイル名と行番号のデバッグ情報を表示します。定義されているシンボルに対しては、シンボルのアドレスの行番号を捜します。未定義のシンボルに対しては、シンボルを参照している relocation entry の行番号を捜します。行番号情報を見つけることができた場合、他のシンボル情報の後にそれを表示します。

`-n`
`-v`
`--numeric-sort`
シンボルをアドレスによる数値順にソートします。

`-p`
`--no-sort`
シンボルをソートしません。出現した順に表示します。

`-P`
`--portability`
デフォルトのフォーマットではなく、POSIX.2 standard output format を使用します。‘`-f posix`’と同じです。

`-s`
`--print-armap`
archive members のシンボル表示の際、その index も含めます。モジュールがどの名前の定義を含んでいるかを示す mapping(`ar`または`ranlib` により archive 内に格納されている) 情報です。

`-r`
`--reverse-sort`
数値順またはアルファベット順にソートされたものを逆順に表示します。

`--size-sort`
: シンボルを size によりソートします。size は、そのシンボルの値と次に高アドレスなシンボルの値との差分で計算されます。値ではなく、シンボルの size が表示されます。

`-t` *radix*
`--radix=`*radix*
: シンボル値を表示する基数として *radix* を使用します。10 進数の場合、'`d`'、8 進数の場合、'`o`'、16 進数の場合、'`x`'を指定します。

`--target=`*bfdname*
: システムデフォルトのフォーマット以外のオブジェクトコードフォーマットを指定します。

`-u`
`--undefined-only`
: 各々のオブジェクトファイルで external な未定義シンボルのみ表示します。

`--defined-only`
: 各々のオブジェクトファイルで定義されているシンボルのみ表示します。

`-V`
`--version`
: nmのバージョン番号を表示して終了します。

`--help`
: nmのオプション要約を表示して終了します。

4.3 objcopy

```
objcopy [ -F bfdname | --target=bfdname ]
        [ -I bfdname | --input-target=bfdname ]
        [ -O bfdname | --output-target=bfdname ]
        [ -S | --strip-all ]  [ -g | --strip-debug ]
        [ -K symbolname | --keep-symbol=symbolname ]
        [ -N symbolname | --strip-symbol=symbolname ]
[ -L symbolname | --localize-symbol=symbolname ]
[ -W symbolname | --weaken-symbol=symbolname ]
        [ -x | --discard-all ]  [ -X | --discard-locals ]
        [ -b byte | --byte=byte ]
        [ -i interleave | --interleave=interleave ]
        [ -R sectionname | --remove-section=sectionname ]
[ -p | --preserve-dates ] [ --debugging ]
        [ --gap-fill=val ] [ --pad-to=address ]
        [ --set-start=val ] [ --adjust-start=incr ]
        [ --adjust-vma=incr ]
        [ --adjust-section-vma=section{=,+,-}val ]
        [ --adjust-warnings ] [ --no-adjust-warnings ]
        [ --set-section-flags=section=flags ]
        [ --add-section=sectionname=filename ]
        [ --change-leading-char ] [ --remove-leading-char ]
        [ --weaken ]
        [ -v | --verbose ] [ -V | --version ]  [ --help ]
        infile [outfile]
```

GNU objcopy utility は、1 つのオブジェクトファイルの内容を別のファイルへコピーします。objcopyは、オブジェクトファイルの読み込みおよび書き出しのために GNU BFD Library を使用します。入力オブジェクトファイルと異なるフォーマットで、出力オブジェクトファイルに書き出すことができます。objcopyの正確な振る舞いは、コマンド行オプションによって制御することができます。

objcopyは、変換を行うためにテンポラリファイルを生成し、変換後削除します。objcopyは、すべての変換作業を行う上で、BFD を使用します。BFD で説明されている全てのフォーマットにアクセスすることができ、明確に指定することなく自動でほとんどのフォーマットを認識することができます。

objcopyは、出力ターゲットに 'srec'を指定することによって S-records を生成することができます ('-O srec'を指定します)。

objcopyは、出力ターゲットに 'binary'を指定することによってバイナリファイルを生成することができます ('-O binary'を指定する)。objcopyがバイナリファイルを生成する時は、入力オブジェクトファイルの内容のメモリダンプが生成されます。全てのシンボルおよびリロケーション情報は削除されます。メモリダンプは、出力ファイル中にコピーされる最下位のセクションのアドレスから開始されます。

S-record やバイナリファイルを生成する場合、デバッグ情報を含んでいるセクションを削除するために '-S'を指定するのが有用かもしれません。いくつかの場合においては、バイナリファイルによって必要ない情報を含んでいるセクションを削除するために、'-R'オプションが有用かもしれません。

@*cmdfile*
: *cmdfile* で指定されたファイルの内容を、空白文字 (改行、タブ含む) で区切られた引数としてコマンドラインに展開してコマンドを実行します。この機能は、Windows 環境でのみ有効です。

infile
outfile
: 各々入力ファイル、出力ファイルを指定します。*outfile* が指定されない場合、objcopyは、テンポラリファイルを作成し、その結果を *infile* の名前にします。従って、*infile* は破壊されることになります。

-I *bfdname*
--input-target=*bfdname*
: 入力ファイルのオブジェクトフォーマットを推測するのではなく、*bfdname* であるとみなします。

-O *bfdname*
--output-target=*bfdname*
: *bfdname* で指定されたオブジェクトフォーマットを使用して、出力ファイルに書き出します。

-F *bfdname*
--target=*bfdname*
: 入力ファイルと出力ファイルの両方のオブジェクトフォーマットが、*bfdname* であるとして処理します。例えば、入力ファイルから出力ファイルへ変換なしでデータを転送したい場合に使用します。

-R *sectionname*
--remove-section=*sectionname*
: *sectionname* で指定した名前のセクションを出力ファイルから削除します。このオプションは、1 回以上指定することができます。このオプションを使用するこ

とによって、利用価値のない出力ファイルが生成されるかもしれないので注意が必要です。

`-S`
`--strip-all`
入力ファイルからリロケーション情報とシンボル情報をコピーしません。

`-g`
`--strip-debug`
入力ファイルからデバッグ用シンボル情報をコピーしません。

`--strip-unneeded`
リロケーション処理に必要のなり全てのシンボルを削除します。

`-K` *symbolname*
`--keep-symbol=`*symbolname*
入力ファイルから、*symbolname* で指定してシンボルのみコピーします。このオプションは、1 回以上指定することができます。

`-N` *symbolname*
`--strip-symbol=`*symbolname*
入力ファイルから、*symbolname* で指定したシンボルをコピーしません。このオプションは、1 回以上指定することができます。`-K`以外の strip options と同時に指定することができます。

`-L` *symbolname*
`--localize-symbol=`*symbolname*
symbolname で指定されたシンボルを、外部に見えないようにローカルなシンボルにします。このオプションは、1 回以上指定することができます。

`-W` *symbolname*
`--weaken-symbol=`*symbolname*
symbolname で指定されたシンボルを、weak なシンボルにします。このオプションは、1 回以上指定することができます。

`-x`
`--discard-all`
入力ファイルからグローバルでないシンボルをコピーしません。

`-X`
`--discard-locals`
コンパイラが生成したローカルシンボルをコピーしません。これらのシンボルは、通常、'L'または '.'から始まります。

`-b` *byte*
`--byte=`*byte*
入力ファイルの全ての *byte*(th) のデータを出力します (ヘッダのデータは影響しません)。*byte* には、0 から *interleave*-1 の範囲の値を指定することができます。*interleave* は、'`-i`' または '`--interleave`'オプションによって指定されたもので、デフォルトは 4 です。このオプションは、ROM をプログラムするためのファイルを生成するために有用です。通常は、出力ターゲットに `srec` を指定して使用します。

`-i` *interleave*
`--interleave=`*interleave*
interleave バイトごとデータを出力します。どのバイトをコピーするかは-*b* または '`--byte`'オプションで選択します。デフォルトは 4 です。'`-b`'または '`--byte`' オプションが指定されていない場合、`objcopy`はこのオプションを無視します。

`-p`
`--preserve-dates`
入力ファイルと同じアクセス日と変更日の情報を出力ファイルに設定します。

`--debugging`
可能ならばデバッグ情報を変換します。特定のデバッグ情報のみがサポートされているのみであり、また、変換処理は時間を消費するため、デフォルトでは行いません。

`--gap-fill` *val*
セクション間のギャップを *val* で埋めます。これは低アドレス側のセクションサイズを増加することにより行われます。拡張された領域には、*val* が埋め込まれます。

`--pad-to` *address*
address で指定した仮想的なアドレスまで、ファイルの出力を詰めます。これは、最後のセクションのサイズを増加することにより行われます。拡張された領域には、'`--gap-fill`'で指定した値が埋め込まれます (デフォルトでは、0 が埋め込まれる)。

`--set-start` *val*
新しいファイルの開始アドレスを *val* に設定します。全てのオブジェクトファイルフォーマットが、開始アドレスの設定をサポートしているわけではありません。

`--adjust-start` *incr*
incr を加算することにより、開始アドレスを調整します。全てのオブジェクトファイルフォーマットが、開始アドレスの設定をサポートしているわけではありません。

`--adjust-vma` *incr*
全てのセクションのアドレスを、開始アドレスと同様に、*incr* を加算することによって調整します。

いくつかのオブジェクトフォーマットでは、勝手にセクションのアドレスを変更することは許可されていません。この処理は、セクションの再配置 (relocate) は行わないので注意が必要です。プログラムが、あるアドレスにロードされるべきセクションを期待している時に、セクションのアドレスを変更するためにこのオプションが使用されると、それらのセクションは異なるアドレスにロードされることになり、プログラムが正しく動作しないかもしれません。

`--adjust-section-vma` *section*`{=,+,-}`*val*
section 名のセクションのアドレスを設定または調整します。'`=`'が指定された場合、セクションのアドレスは、*val* に設定されます。そうでない場合、*val* で指定した値がセクションのアドレスに加算または減算されます。'`--adjust-vma`' に記述されている `Note:` も参照してください。もし *section* が入力ファイル中に存在しない場合、warning を出力します。warning の出力を抑止したい場合、'`--no-adjust-warnings`' を指定します。

`--adjust-warnings`
'`--adjust-section-vma`'が指定された場合に、指定されたセクションが存在しなければ、warning を出力します。これはデフォルトです。

`--no-adjust-warnings`
'`--adjust-section-vma`'が指定された場合に、指定されたセクションが存在しなくても、warning を出力しません。

`--set-section-flags` *section*`=`*flags*
指定された名前のセクションに対してフラグを設定します。*flags* 引数には、コンマ`,`で区切られたフラグ名を指定します。指定可能なフラグ名は、'`alloc`'、'`load`'、

'readonly'、'code'、'data'、'rom'です。全てのフラグ名が、全てのオブジェクトファイルフォーマットに対して意味があるとは限りません。

`--add-section` *sectionname*=*filename*
: ファイルをコピーすると共に *sectionname* で指定された新しいセクションを追加します。新しいセクションの内容は、*filename* で指定したファイルから取り込まれます。セクションサイズは、指定されたファイルのサイズになります。このオプションは、セクション名の変更が可能なファイルフォーマットでのみ動作します。

`--change-leading-char`
: いくつかのオブジェクトフォーマットでは、シンボル名の最初に特別な文字を使用しています。最も一般的な文字は、アンダースコア_であり、コンパイラが全てのシンボルの先頭に付加します。このオプションは、objcopyがオブジェクトファイルフォーマットを変換する時、全てのシンボルの先頭文字を変更することを指定します。もしオブジェクトファイルフォーマットが、同じ先頭文字を使用しているならば、このオプションは影響しません。そうでない場合、objcopyは、文字の追加、文字の削除、文字の変更を適切に行います。

`--remove-leading-char`
: グローバルシンボルの最初の文字が、オブジェクトファイルフォーマットにより使用されている先頭文字がついている特別なシンボルならば、その文字を削除します。最も一般的な先頭文字のシンボルは、アンダースコア_です。このオプションは、全てのグローバルシンボルから先頭のアンダースコアを削除します。このオプションは、異なるシンボル名のコンベンションを持つ異なるファイルフォーマットのオブジェクトと一緒にリンクしたい場合に有用です。このオプションの動作は、`--change-leading-char`と異なります。なぜならば`--code-leading-char`は、出力ファイルのオブジェクトファイルフォーマットに構わず適切な時に常にシンボル名を変更するからです。

`--weaken` すべてのグローバルシンボルが、weak になるように変更します。これは、リンカオプションの`-R`を使用して、他のオブジェクトとリンクされるオブジェクトを構築する時に有用です。このオプションは、weak シンボルをサポートしているオブジェクトファイルフォーマットを使用する時のみ有効です。

`-V`
`--version`
: objcopyのバージョン番号を表示します。

`-v`
`--verbose`
: 変更される全てのオブジェクトファイル名のリストなど、処理過程の冗長な情報を出力します。archives の場合、'`objcopy -V`'オプションにより archive の全てのメンバを表示します。

`--help` オプションの要約を表示します。

4.4 objdump

```
objdump [ -a | --archive-headers ]
        [ -b bfdname | --target=bfdname ] [ --debugging ]
[ -C | --demangle ] [ -d | --disassemble ]
[ -D | --disassemble-all ] [ --disassemble-zeroes ]
        [ -EB | -EL | --endian={big | little } ]
        [ -f | --file-headers ]
```

```
            [ -h | --section-headers | --headers ]  [ -i | --info ]
            [ -j section | --section=section ]
            [ -l | --line-numbers ] [ -S | --source ]
            [ -m machine | --architecture=machine ]
            [ -r | --reloc ] [ -R | --dynamic-reloc ]
            [ -s | --full-contents ]  [ --stabs ]
            [ -t | --syms ] [ -T | --dynamic-syms ] [ -x | --all-headers ]
            [ -w | --wide ] [ --start-address=address ]
            [ --stop-address=address ]
            [ --prefix-addresses] [ --[no-]show-raw-insn ]
    [ --adjust-vma=offset ]
            [ --version ]  [ --help ]
            objfile...
```

objdumpは、指定されたオブジェクトファイルの情報を表示します。オプション指定により何の情報を表示するか制御することができます。この情報は、単にコンパイルして実行したいだけのプログラマとは反対に、コンパイラ系ツールで仕事を続けるプログラマにとっては有益な情報です。

objfile...には、対象のオブジェクトファイルを指定します。archive を指定した場合、archive 内の各々のオブジェクトファイルに対する情報を表示します。

二者択一で示している long and short forms オプションは、同じ意味を持っています。'-l'オプションを除くオプションのうち、少なくとも 1 つのオプションが指定されなければなりません。

@*cmdfile*
: *cmdfile* で指定されたファイルの内容を、空白文字 (改行、タブ含む) で区切られた引数としてコマンドラインに展開してコマンドを実行します。この機能は、Windows 環境でのみ有効です。

-a
--archive-header
: *objfile* files のどれかが archives である場合、archive header information を表示します ('ls -l'の出力に似た書式で表示されます)。'objdump -a'では、'ar tv'で表示される情報の他に、archive 内メンバのオブジェクトファイルフォーマット情報を表示します。

--adjust-vma=*offset*
: 情報を表示する際、最初に全てのセクションアドレスに *offset* を加算します。このオプションは、セクションアドレスが、シンボルテーブルに一致しない場合に有用です。例えば、a.out のようなセクションアドレスを表現することができないフォーマットを使用している際、特定のアドレスにセクションを配置した場合に起こります。

-b *bfdname*
--target=*bfdname*
: 指定したオブジェクトファイルの object-code format が *bfdname* であることを指定します。*objdump* は、多くのフォーマットを自動的に識別することができるので、このオプションを指定する必要はないかもしれません。

  以下に例を示します。

```
objdump -b oasys -m vax -h fu.o
```

この例では、ファイル‘fu.o’のセクションヘッダ情報の概要を表示し、Oasys コンパイラによって生成された VAX Object として扱われることを指定しています。‘-i’オプションにより、指定可能なフォーマットのリストを表示することができます。

-C
--demangle
low-level なシンボル名を、ユーザーレベルの名前に解読 (*demangle*) します。システムによって付加された最初のアンダースコアを取り除くことに加えて、C++ の関数名を読めるようにします。

--debugging
デバッグ情報を表示します。ファイル内のデバッグ情報を解析して、C 言語ライクな文法で表示します。デバッグ情報の限定されたタイプについてのみ実装されています。DWARF デバッグ情報は、サポートしていません。

-d
--disassemble
objfile 内の命令コードに対するアセンブラニーモニックを表示します。このオプションは、命令が含まれていることが期待されるセクションのみディスアセンブルします。

-D
--disassemble-all
‘-d’と同様ですが、命令が含まれていることが期待されるセクションのみでなく、全てのセクションについてディスアセンブルします。

--prefix-addresses
ディスアセンブルの際、各行に完全なアドレスを表示します。これは、ディスアセンブルフォーマットの古い形式です。

--disassemble-zeroes
通常、ディスアセンブル出力は、連続する 0 のブロックを省略します。このオプションは、他のデータと同様に、これらのブロックをディスアセンブルすることを指示します。

-EB
-EL
--endian={big|little}
オブジェクトファイルのエンディアンを指定します。この指定は、ディスアセンブル処理にのみ影響します。 S-records のようなエンディアン情報がないファイルフォーマットをディスアセンブルするのに有効です。

-f
--file-header
objfile で指定された各々のファイルの全てのファイルヘッダ情報を表示します。

-h
--section-header
--header 指定オブジェクトファイルのセクションヘッダ情報を表示します。
ファイルのセグメントは、標準と異なるアドレスに再配置されていることがあります。例えば、ldに対するオプションに‘-Ttext’ 、‘-Tdata’や‘-Tbss’オプションを指定している場合、セグメントは指定のアドレスに再配置されています。しかし、いくつかのオブジェクトファイルフォーマット (例えば、a.out フォーマット) では、ファイルセグメントの開始アドレスは保持されません。従って、ldはセクションを正しく再配置しますが、‘objdump -h’を使用して、ファイルのセクションヘッダリストを正しいアドレスで表示することはできません。この場合暗黙にターゲットに対する一般アドレスを表示します。

`--help` objdumpのオプションの要約を表示して終了します。

`-i`
`--info` '`-b`'と '`-m`'オプションに指定できる全てのアーキテクチャとオブジェクトフォーマットを表示します。

`-j` *name*
`--section=`*name*
name で指定された情報のみ表示します。

`-l`
`--line-numbers`
表示されるオブジェクトコードや配置情報に一致したファイル名と行番号を一緒に表示します。'`-d`'、'`-D`'または '`-r`'を指定した時のみ有効です。

`-m` *machine*
`--architecture=`*machine*
オブジェクトファイルをディスアセンブルする時に使用するアーキテクチャを指定します。これは、S-records のようなアーキテクチャの情報がないオブジェクトファイルをディスアセンブルする時に有効です。'`-i`'オプションを指定することにより、指定可能なアーキテクチャのリストを表示させることができます。

`-r`
`--reloc` ファイルの relocation entries を表示します。'`-d`'または '`-D`' オプションを指定している場合、relocations は、ディスアセンブルに混在して表示されます。

`-R`
`--dynamic-reloc`
ファイルの dynamic relocation entries を表示します。これは、共有ライブラリのような dynamic objects に対してのみ意味を持ちます。

`-s`
`--full-contents`
要求されたセクションの全ての内容を表示します。

`-S`
`--source` 可能ならばディスアセンブルと共にソースコードを混在して表示します。'`-d`'の機能を含みます。

`--show-raw-insn`
ディスアセンブル時に、命令をシンボル形式と同様に 16 進数で表示します。これは、`--prefix-addresses`を指定した場合を除き、デフォルトになっています。

`--no-show-raw-insn`
ディスアセンブル時に、命令を 16 進数で表示しません。これは、`--prefix-addresses`を指定した場合のデフォルトです。

`--stabs` 要求されたセクションの全ての内容を表示します。この表示には、ELF ファイルの.stab、.stab.index、.stab.excl セクションの内容も含まれます。これは、`.stab` debugging symbol-table entries が ELF セクションに導入されているシステム (例えば、Solaris 2.0) においてのみ有効です。他のほとんどのファイルフォーマットでは、debugging symbol-table entries は linkage symbols と関連付けられており、'`--syms`' 指定により表示することができます。

`--start-address=`*address*
データの表示を指定アドレスから開始します。`-d`、`-r`、`-s` オプションの出力に対して影響します。

`--stop-address=`*address*
データの表示を指定アドレスで止めます。`-d`、`-r`、`-s` オプションの出力に対して影響します。

`-t`
`--syms` ファイルの symbol table entries を表示します。これは、'`nm`' プログラムによって得られる情報とほぼ同じです。

`-T`
`--dynamic-syms`
ファイルの dynamic symbol table entries を表示します。このオプションは、共有ライブラリのような dynamic objects に対してのみ意味を持ちます。'`-D`'('`--dynamic`') オプションが指定された時、'`nm`'プログラムによって得られる情報とほぼ同じです。

`--version`
`objdump`のバージョン番号を表示して終了します。

`-x`
`--all-header`
symbol table と relocation entries を含むヘッダ情報を表示します。'`-x`'の指定は、'`-a -f -h -r -t`'の全てを指定した場合と等価です。

`-w`
`--wide` 80 カラム以上に対応した出力デバイスに対するフォーマット指定です。

4.5 ranlib

`ranlib [-vV]` *archive*

`ranlib`は、archive 内容の index を生成し、その情報を archive 内に格納します。index は、relocatable object file である archive の member で定義されている各々のシンボルリストです。

'`nm -s`'または '`nm --print-armap`'を指定することにより、この index を表示することができます。

インデックスの付いた archive は、ライブラリのリンクを高速にします。また、archive 内の位置を考慮することなく、お互いにライブラリ内のルーチンを呼び出すことを許します。

GNU `ranlib`プログラムは、GNU `ar`の別形態です。`ranlib`の実行は、'`ar -s`'の実行と完全に等価です。

`@`*cmdfile* *cmdfile* で指定されたファイルの内容を、空白文字 (改行、タブ含む) で区切られた引数としてコマンドラインに展開してコマンドを実行します。この機能は、Windows 環境でのみ有効です。

`-v`
`-V` `ranlib`のバージョン番号を表示します。

4.6 size

`size [ -A | -B | --format=`*compatibility* `]`
`[ --help ] [ -d | -o | -x | --radix=`*number* `]`
`[ --target=`*bfdname* `] [ -V | --version ]`
objfile`...`

GNU `size` utility は、引数に指定されたオブジェクトおよびアーカイブファイル中のオブジェクトの個々のセクションサイズおよび合計サイズの情報を出力します。デフォルトでは、オブジェクトおよびアーカイブ中の各々のモジュールに対する情報は、1 行で出力されます。

objfile...には、対象のオブジェクトファイルを指定します。

コマンド行オプションは、次の意味を持っています。

`@`*cmdfile*　*cmdfile* で指定されたファイルの内容を、空白文字 (改行、タブ含む) で区切られた引数としてコマンドラインに展開してコマンドを実行します。この機能は、Windows 環境でのみ有効です。

`-A`
`-B`
`--format=`*compatibility*
　これらのオプションのうち 1 つを指定することができます。GNU `size`の出力形式を、System V `size`仕様にする場合、‘`-A`’または‘`--format=sysv`’を指定します。Berkeley `size`仕様にする場合、‘`-B`’または‘`--format=berkeley`’を指定します。デフォルトでは、Berkeley `size`の仕様に似た 1 行形式で出力されます。

　以下に、`size`が出力する Berkelay 仕様の例を示します。

```
size --format=Berkeley ranlib size
text    data    bss     dec     hex     filename
294880  81920   11592   388392  5ed28   ranlib
294880  81920   11888   388688  5ee50   size
```

　同じデータを使用して、System V 仕様で出力した例を以下に示します。

```
size --format=SysV ranlib size
ranlib  :
section         size         addr
.text         294880         8192
.data          81920       303104
.bss           11592       385024
Total         388392


size  :
section         size         addr
.text         294880         8192
.data          81920       303104
.bss           11888       385024
Total         388688
```

`--help`　引数とオプションの概要を表示します。

`-d`
`-o`
`-x`
`--radix=`*number*
　これらのオプションのうち 1 つを指定することができます。各々のセクションサイズを、10 進数 (‘`-d`’または‘`--radix=10`’) 、8 進数 (‘`-o`’ または‘`--radix=8`’)、16 進数 (‘`-x`’または‘`--radix=16`’) で表示するかを指定します。‘`--radix=`*number*’では、3 つの値 (8、10、16) のみ指定可能です。合計サイズは常に 2 つの基数で表

示されます。'-d'または'-x' を指定した場合には 10 進数と 16 進数、'-o'を指定した場合には 8 進数と 16 進数で表示されます。

--target=*bfdname*
: *objfile* のオブジェクトコードフォーマットが *bfdname* であることを指定します。sizeは、多くのフォーマットを自動的に識別することができるので、このオプションを指定する必要はないかもしれません。

-V
--version
: sizeのバージョン番号を表示します。

4.7 strings

```
strings [-afov] [-min-len] [-n min-len] [-t radix] [-]
        [--all] [--print-file-name] [--bytes=min-len]
        [--radix=radix] [--target=bfdname]
        [--help] [--version] file...
```

file で指定された各々に対して、GNU stringsは、少なくとも 4 文字の長さ (あるいは、オプションで指定された長さ) の非印字文字に続く連続した印字文字を表示します。デフォルトでは、オブジェクトファイル中の initialized sections および loaded sections のみ表示します。他のタイプのファイルの場合、ファイル内すべての文字列を表示します。

stringsは、テキストでないファイルの内容を確認するのに有用です。

@*cmdfile*
: *cmdfile* で指定されたファイルの内容を、空白文字 (改行、タブ含む) で区切られた引数としてコマンドラインに展開してコマンドを実行します。この機能は、Windows 環境でのみ有効です。

-a
--all
-
: オブジェクトファイル中の initialized sections および loaded sections のみでなく、ファイル内すべてをスキャンします。

-f
--print-file-name
: 各々の文字列の前にファイル名を表示します。

--help
: プログラムの使用方法の要約を標準出力に表示して終了します。

-*min-len*

-n *min-len*
--bytes=*min-len*
: 少なくとも *min-len* 文字の長さをもつ連続した文字を表示します。デフォルト 4 文字を変更するのに使用します。

-o
: '-t o'と同様。stringsの他のバージョンでは、'-t d'の動作の代わりに同様の動作をする'-o'の機能があります。両方を互換にすることができないため、どちらかの一方を選択してください。

-t *radix*
--radix=*radix*
: 各々の文字列の前に、ファイル内でのオフセットを表示します。オフセットの基数指定に、'o'(8 進数)、'x'(16 進数)、'd'(10 進数) を指定します。

`--target=`*bfdname*
: システムデフォルトのフォーマット以外のオブジェクトコードフォーマットを指定します。

`-v`
`--version`
: 標準出力にバージョン番号を表示して終了します。

4.8 strip

```
strip [ -F bfdname | --target=bfdname ]
      [ -I bfdname | --input-target=bfdname ]
      [ -O bfdname | --output-target=bfdname ]
      [ -s | --strip-all ] [ -S | -g | --strip-debug ]
      [ -K symbolname | --keep-symbol=symbolname ]
      [ -N symbolname | --strip-symbol=symbolname ]
      [ -x | --discard-all ] [ -X | --discard-locals ]
      [ -R sectionname | --remove-section=sectionname ]
      [ -o file ] [ -p | --preserve-dates ]
      [ -v | --verbose ]  [ -V | --version ]  [ --help ]
      objfile...
```

GNU `strip`は、*objfile* に指定したオブジェクトファイルから、全てのシンボルを削除します。オブジェクトファイルのリストは、archives を含んでいるかもしれません。少なくともオブジェクトファイルを一つ指定しなければなりません。

`strip`は、引数で指定された名前のファイル自身を更新します。別の名前のコピーを更新するのではありません。

`@`*cmdfile*
: *cmdfile* で指定されたファイルの内容を、空白文字 (改行、タブ含む) で区切られた引数としてコマンドラインに展開してコマンドを実行します。この機能は、Windows 環境でのみ有効です。

`-F` *bfdname*
`--target=`*bfdname*
: オリジナルの *objfile* を、*bfdname* のオブジェクトコードフォーマットのファイルとして扱い、同じフォーマットで更新します。

`--help`
: `strip`のオプションの要約を表示して終了します。

`-I` *bfdname*
`--input-target=`*bfdname*
: オリジナルの *objfile* を、*bfdname* のオブジェクトコードフォーマットのファイルとして扱います。

`-O` *bfdname*
`--output-target=`*bfdname*
: *objfile* を *bfdname* で指定された出力フォーマットのファイルに置換します。

`-R` *sectionname*
`--remove-section=`*sectionname*
: 出力ファイルから *sectionname* のセクションを削除します。このオプションは、一回以上指定することができます。このオプションは、利用することができない出力ファイルを生成することもできるため注意が必要です。

`-s`
`--strip-all`
: 全てのシンボルを削除します。

`-g`
`-S`
`--strip-debug`
: デバッグ用シンボルのみ削除します。

`--strip-unneeded`
: relocation 処理に不必要な全てのシンボルを削除します。

`-K` *symbolname*
`--keep-symbol=`*symbolname*
: ソースファイルから、*symbolname* のシンボルのみ残すようにします。このオプションは、一回以上指定することができます。

`-N` *symbolname*
`--strip-symbol=`*symbolname*
: ソースファイルから、*symbolname* のシンボルのみ削除します。このオプションは、一回以上指定することができ、`-K`以外の strip オプションと組み合わせて使用することができます。

`-o` *file*
: 既存のファイルを置換するのでなく、*file* で指定されたファイルにストリップの結果を出力します。この指定を行った場合、一つの *objfile* しか指定することができません。

`-p`
`--preserve-dates`
: ファイルのアクセス日と更新日を維持します。

`-x`
`--discard-all`
: グローバルでないシンボルを削除します。

`-X`
`--discard-locals`
: コンパイラが生成したローカルシンボルを削除します。(これらのシンボルは、通常 'L'または '.'ではじまります。

`-V`
`--version`
: `strip`のバージョン番号を表示します。

`-v`
`--verbose`
: 冗長な情報を出力します。情報としては、変更された全てのオブジェクトファイル名が表示されます。archives の場合、'`strip -v`'はすべての archive の members を表示します。

Short Contents

Table of Contents

TW32R V.3.10 GNU　ツールリファレンス
M32R 用　クロスツールキット (GNU 版)

ルネサス エレクトロニクス株式会社
神奈川県川崎市中原区下沼部1753 〒211-8668