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

RENESAS

RASM77 V.5.10

ユーザーズマニュアル

77xx シリーズ用リロケータブルアセンブラ

ルネサス エレクトロニクス
www.renesas.com

Rev.1.00　2003.08

- Microsoft 、MS- DOS 、Windows および Windows NT は、米国 Microsoft Corporation の米国およびその他の国における登録商標です。
- HP- UX は、米国 Hewlett- Packard Company のオペレーティングシステムの名称です。
- Sun 、Java およびすべての Java 関連の商標およびロゴは、米国およびその他の国における米国 Sun Microsystems,Inc.の商標または登録商標です。
- UNIX は、X/Open Company Limited が独占的にライセンスしている米国ならびに他の国における登録商標です。
- Linux は、Linus Torvalds 氏の米国およびその他の国における登録商標あるいは商標です。
- Turbolinux の名称およびロゴは、Turbolinux,Inc.の登録商標です。
- IBM および AT は、米国 International Business Machines Corporation の登録商標です。
- HP 9000 は、米国 Hewlett- Packard Company の商品名称です。
- SPARC および SPARCstation は、米国 SPARC International,Inc.の登録商標です。
- Intel,Pentium は、米国 Intel Corporation の登録商標です。
- Adobe および Acrobat は、Adobe Systems Incorporated （アドビシステムズ社）の登録商標です。
- Netscape および Netscape Navigator は、米国およびその他の諸国の Netscape Communications Corporation 社の登録商標です。
- その他すべてのブランド名および製品名は個々の所有者の登録商標もしくは商標です。

安全設計に関するお願い

- 弊社は品質、信頼性の向上に努めておりますが、半導体製品は故障が発生したり、誤動作する場合があります。弊社の半導体製品の故障又は誤動作によって結果として、人身事故火災事故、社会的損害などを生じさせないような安全性を考慮した冗長設計、延焼対策設計、誤動作防止設計などの安全設計に十分ご留意ください。

本資料ご利用に際しての留意事項

- 本資料は、お客様が用途に応じた適切なルネサス テクノロジ製品をご購入いただくための参考資料であり、本資料中に記載の技術情報について株式会社ルネサス テクノロジおよび株式会社ルネサス ソリューションズが所有する知的財産権その他の権利の実施、使用を許諾するものではありません。
- 本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他応用回路例の使用に起因する損害、第三者所有の権利に対する侵害に関し、株式会社ルネサス テクノロジおよび株式会社ルネサス ソリューションズは責任を負いません。
- 本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他全ての情報は本資料発行時点のものであり、株式会社ルネサス テクノロジおよび株式会社ルネサス ソリューションズは、予告なしに、本資料に記載した製品又は仕様を変更することがあります。ルネサス テクノロジ半導体製品のご購入に当たりましては、事前に株式会社ルネサス テクノロジ、株式会社ルネサス ソリューションズ、株式会社ルネサス販売又は特約店へ最新の情報をご確認頂きますとともに、ルネサス テクノロジホームページ（http://www.renesas.com）などを通じて公開される情報に常にご注意ください。
- 本資料に記載した情報は、正確を期すため、慎重に制作したものですが万一本資料の記述誤りに起因する損害がお客様に生じた場合には、株式会社ルネサス テクノロジおよび株式会社ルネサス ソリューションズはその責任を負いません。
- 本資料に記載の製品データ、図、表に示す技術的な内容、プログラム及びアルゴリズムを流用する場合は、技術内容、プログラム、アルゴリズム単位で評価するだけでなく、システム全体で十分に評価し、お客様の責任において適用可否を判断してください。株式会社ルネサス テクノロジおよび株式会社ルネサス ソリューションズは、適用可否に対する責任は負いません。
- 本資料に記載された製品は、人命にかかわるような状況の下で使用される機器あるいはシステムに用いられることを目的として設計、製造されたものではありません。本資料に記載の製品を運輸、移動体用、医療用、航空宇宙用、原子力制御用、海底中継用機器あるいはシステムなど、特殊用途へのご利用をご検討の際には、株式会社ルネサス テクノロジ、株式会社ルネサス ソリューションズ、株式会社ルネサス販売又は特約店へご照会ください。
- 本資料の転載、複製については、文書による株式会社ルネサス テクノロジおよび株式会社ルネサス ソリューションズの事前の承諾が必要です。
- 本資料に関し詳細についてのお問い合わせ、その他お気付きの点がございましたら株式会社ルネサス テクノロジ、株式会社ルネサス ソリューションズ、株式会社ルネサス販売又は特約店までご照会ください。

製品の内容及び本書についてのお問い合わせ先

インストーラが生成する以下のテキストファイルに必要事項を記入の上、ツール技術サポート窓口 support_tool@renesas.com まで送信ください。

¥SUPPORT¥製品名¥SUPPORT.TXT

株式会社ルネサス ソリューションズ マイコンツール部
ツール技術サポート窓口　support_tool@renesas.com
ユーザ登録窓口　regist_tool@renesas.com
ホームページ　http://www.renesas.com/jp/tools

はじめに

RASM77 は 7700 ファミリ 専用のリロケータブルマクロアセンブラです。RASM77 は 7700 ファミリ 構造化記述言語、及びアセンブリ言語ソースプログラムから、7700 ファミリ の機械語データファイル、デバッグ情報ファイル等を生成します。本書は、RASM77 に含まれている以下の 5 つのソフトウエアの機能と操作方法について記述しています。

1. リロケータブルマクロアセンブラ RASM77
2. 構造化記述プリプロセッサ PRE77
3. リンケージエディタ LINK77
4. ライブラリアン LIB77
5. クロスリファレンサ CRF77

本書の構成

本書は、以下に示す 5 部から構成されています。マニュアルは、各ソフトウエアごとに 5 部に分れており、各部は可能な限り同じ順序で説明を行っています。例えば環境変数について調べようと思った場合には、操作説明の章の最後の節を見ることにより、各ソフトウエアの情報がわかります。

- 第 1 部 ：RASM77 操作マニュアル
  リロケータブルマクロアセンブラ RASM77 の操作方法とソースプログラム記述方法について説明します。
- 第 2 部 ：PRE77 操作マニュアル
  プリプロセッサ PRE77 の操作方法と構造化記述言語について説明します。
- 第 3 部 ：LINK77 操作マニュアル
  リンカ LINK77 の操作方法とセクションの機能について説明します。
- 第 4 部 ：LIB77 操作マニュアル
  ライブラリアン LIB77 の操作方法について説明します。
- 第 5 部 ：CRF77 操作マニュアル
  クロスリファレンサ CRF77 の操作方法について説明します。

第 1 部

7700 ファミリ 用
リロケータブルマクロアセンブラ

RASM77 操作マニュアル

目 次

図 目 次

表 目 次

第 1 章

RASM77 操作マニュアルの構成

RASM77 操作マニュアルは、以下の章から構成されています。

- 第 2 章 概要
  RASM77 の基本的な機能と RASM77 が生成するファイルについて説明します。
- 第 3 章 ソースプログラムの記述方法
  RASM77 で処理するアセンブリ言語ソースプログラムの構成について説明します。
- 第 4 章 命令の記述方法
  RASM77 で使用可能な 7700 ファミリ の命令記述方法について説明します。
- 第 5 章 疑似命令の記述方法
  RASM77 で用意している疑似命令の内容と記述方法について説明します。
- 第 6 章 マクロ命令
  RASM77 で使用可能なマクロ命令について説明します。
- 第 7 章 操作方法
  RASM77 のコマンド入力方法について説明します。
- 付録 A エラーメッセージ一覧表
  RASM77 が表示するエラーメッセージについて、その内容と対策を一覧表で示します。
- 付録 B 疑似命令一覧表
  RASM77 で使用可能な全疑似命令について、疑似命令ごとの内容を示します。
- 付録 C マクロ命令一覧
  RASM77 で使用可能な全マクロ命令について、マクロ命令ごとの内容を示します。
- 付録 D 命令一覧表
  RASM77 で使用可能な 7700 ファミリ の全命令を示します。
- 付録 E アドレッシングモード別命令一覧表
  RASM77 で使用可能な 7700 ファミリ の命令をアドレッシングモード別に示します。

第 2 章

概要

RASM77 は､アセンブリ言語ソースプログラム (以下ソースファイルと呼びます) を LINK77[1] 及び LIB77[2] で処理可能なリロケータブルファイルに変換します。以下、この作業をアセンブルと呼びます。リロケータブルファイルは LINK77 により機械語データに変換されます。

2.1 機能

大規摸なソフトウエアの開発では、複数エンジニア間でのデータ交換や既存ソフトウエアの再利用等、プログラムの共有を容易に行う機能が必要になります。RASM77 では、以下の機能によりこれらの作業を効率良く行うことができます。

1. 疑似命令.SECTION により、セクション名をユーザが自由に指定できます。
2. セクション数に制限がないので、多くの領域に分割された ROM 、RAM を持つユーザシステムに対しても、リンク時にアドレス指定を行うことが可能です。
3. 疑似命令.VER によってバージョンを宣言することにより、リンク時にリロケータブルファイル間のバージョン確認を行うことが可能です。
4. 疑似命令.LIB 及び.OBJ により、リンク対象ファイルをリロケータブルファイルから指定できます (リンク時のファイル名指定が不要になります) 。

この他に、アセンブル機能として以下のような特長があります。

1. ダイレクトページレジスタ (DPR) 、データバンクレジスタ (DT) の値に対応した効率の良いアドレッシングモードを自動的に選択します。
2. エラー内容を格納したタグファイル[3]を生成できます (アセンブルエラーの修正を効率的に行うことができます) 。
3. 1 つのファイル中に RAM 領域、ROM 領域を混在できるため、アブソリュートアセンブラとしての使用もできます (リンクは必要です) 。

[1] 7700 ファミリ用リンケージエディタのプログラム名。
[2] 7700 ファミリ用ライブラリアンのプログラム名。
[3] タグの名称は、エラーやワーニングの場所を示す荷札 (タグ) に由来しています。

2.2 生成ファイル

RASM77 では、以下の 3 種類のファイルを生成します。

1. リロケータブルファイル
   - 機械語データと、その再配置情報を格納したファイルです。
   - シンボリックデバッグのためのシンボル情報を含んでいます (ローカルシンボル情報は、コマンドオプション “–S”を付加してアセンブルしたときのみ含まれます) 。
   - LINK77 で処理することにより、インテル HEX 形式の機械語データを生成します。
   - アセンブルエラーが発生した場合、リロケータブルファイルは出力しません。
   - ファイル属性は、.R77 です。
   - このファイルはバイナリ形式ファイルですのでプリンタ及び画面への出力は、行なわないでください。
2. プリントファイル (以下 PRN ファイルと呼びます)
   - 処理対象のソースファイルと、その配置アドレス及び生成したデータを示すファイルです。
   - このファイルはプリンタへ出力して、デバッグ等のためにお使いください。
   - PRN ファイルは、コマンドパラメータ “–L”を指定した時に出力します。
   - ファイル属性は、.PRN です。
   - ファイル構成の詳細については 2.3 節で説明します。
3. タグファイル (以下 TAG ファイルと呼びます)
   - アセンブル中に発生したアセンブルエラーメッセージ及びワーニングメッセージを格納したファイルです。
   - ファイル属性は、.TAG です。
   - タグジャンプ機能を持つエディタを使用することで効率よくエラーの修正ができます。
   - TAG ファイルは、エディタによるエラー修正時に参照用としてお使いください。
   - TAG ファイルは、コマンドパラメータ “–E”を指定した時に出力します。
   - ファイル構成の詳細については 2.4 節で説明します。

2.3 PRN ファイルの構成

図 2.1から図 2.5に、PRN ファイルの出力例を示します。PRN ファイルは、以下の情報を示しています。

1. ソースファイルの内容とこれに対応するアドレス、生成データを示す情報 (図 2.1から図 2.3までの部分) 。

- 外部ラベル[4]を参照している行は、ソースの横に 'E' を表示します。
- パブリックラベル[5]を参照している行は、ソースの横に 'P' を表示します。
- ローカルラベル[6]を参照している行は、ソースの横に 'L' を表示します。
- マクロ展開をしている行は、ソースの横に '+' を表示します (図 2.6の部分) 。
- .DEFINE で定義されたシンボルを参照している行は、ソースの横に '–' を表示します。
- .INCLUDE のネスティングレベルをソースの横にレベルに対応する数字で表示します (図 2.6の部分) 。
- 構造化記述ソース行は、コメント行としてプリントファイルに出力されます (図 2.7の部分) 。

2. アセンブル結果概要 (図 2.4の部分)
   エラー数、ワーニング数、全行数、コメント行数、セクションごとのメモリ容量を示します。

3. シンボルリスト (図 2.5の前半部分)
   シンボル一覧の行当りの数は ".COL" 疑似命令に対応し、可変です。".COL" で指定した桁数が 99 桁以下の場合の出力シンボル数は 4 、100 桁以上 119 桁以下の場合のシンボル数は 5 、120 桁以上の場合のシンボル数は 6 となります。
   プログラム中のシンボルとその値を、以下の 3 種類にわけて表示します。

   - –D OPTION
     コマンド行からコマンドパラメータ "–D"により定義され、プログラム中で参照されているシンボルを示します。
   - EQUATE
     疑似命令.EQU により定義され、プログラム中で参照されているシンボルを示します。
   - UNUSED
     上記の方法により定義されており、プログラム中で参照されていないシンボルを示します。

4. ラベルリスト (図 2.5の後半部分)
   プログラム中のラベルとその値を、以下の 2 種類にわけて表示します。

   - USED
     定義されており、プログラム中で参照されているラベルを示します。
   - UNUSED
     定義されているが、プログラム中で参照されていないラベルを示します。

[4] 他のファイル中で定義しているラベルを指します。なお、外部ラベルとパブリックラベルを総称してグローバルラベルと呼びます。
[5] このファイル中で定義しているラベルで、他のファイルから参照可能なラベルを指します。
[6] このファイル中で定義しているラベルで、ファイル中でのみ参照可能なラベルを指します。

5. 疑似命令.COL によるカラム数指定が 132 文字の場合、リストのヘッダ部にアセンブルを実行した時間を次の形式で表示します。

```
Sun Mar 31 15:06:42 1991
```

```
 1                              ;
 2                              ;       Start_up Routine
 3                              ;
 4
 5                              ;
 6                              ;       DATA セクション
 7                              ;         初期化された静的変数領域
 8                              ;
 9                                      .SECTION    DATA
10                                      .PUB        DATATOP
11  000000                      DATATOP:
12 (000000)    8H BYTE         NULLDT: .BLKB       8
13 (000008)   1AH BYTE         TITLE:  .BLKB       26
14
15                              ;
16                              ;       BSS セクション
17                              ;         初期化されていない静的変数領域
18                              ;
19                                      .SECTION    BSS
20                                      .PUB        BSSTOP
21  000000                      BSSTOP:
22 (000000)    1H BYTE         WORK1:  .BLKB       1
23 (000001)    1H BYTE         WORK2:  .BLKB       1
24 (000002)    2H BYTE         WORK3:  .BLKW       1
25
26                              ;
27                              ;       HEAP セクション
28                              ;         malloc 等のメモリ操作関数で使用する領域
29                              ;
30                                      .SECTION    HEAP
31                                      .PUB        HEAPTOP
32  000000                      HEAPTOP:
33 (000000) 1000H BYTE         HEAP_A: .BLKB       1000H
34
35                              ;
36                              ;       STACK セクション
37                              ;         スタック領域
38                              ;
39                                      .SECTION    STACK
40                                      .PUB        STKTOP
41 (000000) 1000H BYTE                  .BLKB       1000H
42  001000                      STKTOP:
```

図 2.1: PRN ファイル例 (ソースファイル前半部分)

```
43                                      .PAGE
44                                ;
45                                ;     PROGRAM セクション
46                                ;       プログラム領域
47                                ;
48                                      .SECTION    PROGRAM
49                                      .EXT        MAIN
50                                      .DPEXT      DPPAGE1:WK1,WK2
51                                      .DTEXT      DTPAGE1:TBL1,TBL2
52                                      .PUB        _INIT
53                                      .DATA       16
54                                      .INDEX      16
55                              E       .DP         OFFSET DPPAGE1
                                ↑外部ラベルの参照を示す
56                              E       .DT         BANK DTPAGE1
57  000000                        _INIT:
58  000000  C2FF                       CLP         #0FFH
59  000002  A90010              P      LDA         A,#STKTOP
                                ↑パブリックラベルの参照を示す
60  000005  1B                         TAS
61  000006  A90000                     LDA         A,#SIZEOF DATA
62  000009  A20000              P      LDX         #OFFSET CONSTOP
63  00000C  A00000              P      LDY         #OFFSET DATATOP
```

図 2.2: PRN ファイル例 (ソースファイル中間部分)

```
64  00000F  540000        P           MVN         BANK CONSTOP, BANK DATATOP
65  000012  AD0000        E           LDA         A,DT:TBL1
66  000015 *42AD0000      E           LDA         B,TBL2
           ↑最適化を示す
67  000019  A20000        E           LDX         #WK1
68  00001C  202200        L           JSR         _SUB
                          ↑ローカルラベルの参照を示す
69  00001F  4C0000        E           JMP         MAIN
70                            ;
71  000022                    _SUB:
72  000022  9500                      STA  A,0,X
73  000024  42950A                    STA  B,10,X
74  000027  60                        RTS
75
76                            ;
77                            ;       CONST セクション
78                            ;         初期化データ領域
79                            ;
80                                    .SECTION    CONST
81                                    .PUB        CONSTOP
82  000000
83  000000  000000000000              .BYTE       0,0,0,0,0,0,0,0
    000006  0000
84  000008  4D3337373030              .BYTE       'M37700 C Compiler Ver 1.0',0
    00000E  204320436F6D
    000014  70696C657220
    00001A  56657220312E
    000020  3000
85                            ;
86                                    .END
```

図 2.3: PRN ファイル例 (ソースファイル後半部分)

```
ERROR   COUNT    00000   (0000H)
WARNING COUNT    00000   (0000H)
TOTAL   LINE     00086   (0056H) LINES
COMMENT LINE     00033   (0021H) LINES
DATA          00000034 (000022H) BYTES
↑セクション名を示す
BSS           00000004 (000004H) BYTES
HEAP          00004096 (001000H) BYTES
STACK         00004096 (001000H) BYTES
PROGRAM       00000040 (000028H) BYTES
CONST         00000034 (000022H) BYTES
```

図 2.4: PRN ファイル例 (アセンブル情報)

```
 *** SYMBOLS ( TYPE = -d OPTION ) ***


*** SYMBOLS ( TYPE = EQUATE    ) ***


*** SYMBOLS ( TYPE = UNUSED    ) ***


*** LABELS  ( TYPE = USED      ) ***

_SUB        000022' CONSTOP     000000p DATATOP     000000p DPPAGE1     000000e
                  ↑ローカルラベルを示す

DTPAGE1     000000e MAIN        000000e STKTOP      001000p TBL1        000000e
                                                                ↑パブリックラベルを示す

TBL2        000000e WK1         000000e
                  ↑外部ラベルを示す
*** LABELS  ( TYPE = UNUSED    ) ***

_INIT       000000p BSSTOP      000000p HEAP_A      000000' HEAPTOP     000000p
NULLDT      000000' TITLE       000008' WK2         000000e WORK1       000000'
WORK2       000001' WORK3       000002'
```

図 2.5: PRN ファイル例 (シンボル及びラベルリスト)

```
31  000014                    INITIAL:
32                                    .section        prog
33                                    .include        initial.a77
34                          1  .DATA 16
35                          1  .INDEX 16
36                          1  .DP 0
37                          1  .DT 0
38  000000  78              1  SEI
39  000001  C238            1  CLP m,x,D
40                          1
41                                    .include        clr_ram.a77;initial.a77
                                                                  ; での呼び出し
42  000003  A27F02          2  LDX #027FH
43  000006  9A              2  TXS
44  000007  A90000          2  LDA A,#0
45  00000A  5B              2  TAD
46  00000B  89C200          2  LDT #0
47  00000E  A90000          2  LDA A,#0
48  000011                  2 RAM_CLEAR:
49  000011 *9500            2  STA A,0,X
50  000013  CA              2  DEX
51  000014  CA              2  DEX
52  000015  E07E00          2  CPX #07EH
53  000018  D0F7           L2  BNE RAM_CLEAR
54                          2
55                          1          .DATA   8
56  00001A  F8              1          SEM
57  00001B  58              1          CLI
                           ↑
                          .INCLUDE のネスティングを示す
58  00001C                    MAIN:
59  00001C  A00A00                     LDY     #10
60  00001F                    LOOP:
61  00001F                             ADD     A,WORK,Y        ; マクロ呼び出し
62  00001F  18                +        CLC
63                            +        .IF     "Y"
64  000020  790000         L  +        ADC     A,WORK,Y
65                            +        .ELSE
67                            +        .ENDIF
68                            +        .ENDM
                             ↑
                            マクロ展開部を示す
69  000023  990A00         L           STA     A,DATA,Y
70  000026  88                         DEY
71  000027  C00000                     CPY     #0
72  00002A  D0F3           L           BNE     LOOP
73                                     .END
```

図 2.6: PRN ファイル例 (マクロ及びインクルード展開部)

```
 1                              ; *** 7700 Family PREPROCESSOR V.1.01.01 ***
 2                                  .language    PRE77_Rev01
 3                                  .source      sample.p77
 4                                      .SECTION    RAM
 5                                      .func       _sample_0
 6                                      .ORG        0000H
 7 (000000)      1H BYTE     WORK1:  .BLKB       1
 8 (000001)      1H BYTE     WORK2:  .BLKB       1
 9                                      .endfunc    _sample_0
10                                      .SECTION    INIT
11                                      .func       _sample_1
12                              ;FLAG_0  .EQU       0,WORK1 ←構造化記述ソース行
13                              FLAG_0    .define   WORK1 ←構造化記述展開行
14                              ;FLAG_1  .EQU       1,WORK1
15                              FLAG_1    .define   WORK1
16                                      .endfunc    _sample_1
17                                      .section    structprog
18                                      .func       _sample_2
19                              ;    for [ DP:FLAG_0 ] == 0
20                                  .cline    9   ←ソースレベルデバッグ情報
21  000000                      ..F1: ←構造化記述ラベル
22  000000  2400010011   -L      BBS    #00001H,DP:WORK1,..F2
                        ↑文字列定義シンボルを参照していることを示す
23                              ;        if [ DP:FLAG_1 ] == 1
24                                  .cline    10
25  000005  340002000A   -L      BBC    #00002H,DP:WORK1,..I3
26                              ;          [ WORK1 ] = 0
27                                  .cline    11
28  00000A  A90000               LDA    A,#0
29  00000D *8500          L      STA    A,WORK1
30                              ;          [ WORK2 ] = 0
31                                  .cline    12
32  00000F  A90000               LDA    A,#0
33  000012 *8501          L      STA    A,WORK2
34                              ;        endif
35                                  .cline    13
36  000014                      ..I3:
37  000014  80EA          L      BRA    ..F1
38                              ;    next
39                                  .cline    14
40  000016                      ..F2:
41                                      .endfunc    _sample_2
42                                      .end
```

図 2.7: PRN ファイル例 (構造化記述プログラム部)

2.4 TAG ファイルの構成

図 2.8に TAG ファイルの出力例を示します。TAG ファイルは、以下の情報を示しています。

1. ソース情報
   エラー又はワーニングが発生した行のリスト行番号、アドレス、オブジェクトコード、ソースファイル内容を示します。

2. TAG 情報
   エラー又はワーニングが発生した場所のファイル名、そのファイル中の行番号、通し行番号、エラー番号、エラーメッセージを示します。

TAG ファイルはプリンタに出力してエディタでのエラー修正時に参照用としてお使いください。

```
  115 00F025  EAEA                    BCC     LOOP2
TEST.ASM 115 ( TOTAL LINE 115 ) Error 18: Relative jump is out of range
  127 00F031  EAEAEA                  LDA     A,#data
TEST.ASM 127 ( TOTAL LINE 127 ) Error 20: Reference to undefined label or symbol "data"
  551 00F42B  EAEA                    BRA     TEST2
TEST.ASM 551 ( TOTAL LINE 551 ) Error 20: Reference to undefined label or symbol "TEST2"
  593 00F4FC  EAEAEAEAEAEA            LDA     A,(work,x ; data set
TEST.ASM 593 ( TOTAL LINE 593 ) Error 23: "()" format error ";"
```

図 2.8: TAG ファイル例

第 3 章

ソースプログラムの記述方法

3.1　ソースプログラムの構成

アセンブリ言語ソースプログラムは、行を単位として構成しています。行記述の規則を以下に示します。

1. 各行は 1 行ごとに完結しており、1 命令を 2 行以上にわたって記述することはできません。
2. 1 行の文字数は、改行コードを含めて最大 256 文字です。アセンブラは、256 文字を越えた部分を無視します。
3. 1 行は以下の欄から構成しています。
   - シンボル/ラベル欄
     行を他の場所から参照するためのラベル、又は疑似命令.EQU により値を設定するシンボルを記述します。
   - オペコード/疑似命令欄
     7700 ファミリ 命令ニーモニック (以下オペコードと呼びます) 又は疑似命令を記述します。
   - オペランド欄
     オペコード又は疑似命令の処理対象を記述します。
   - コメント欄
     これ以降の欄はアセンブラで処理しませんのでユーザが自由に使用できます。

行は内容別に以下の 5 種類に分類できます。

1. 命令行
   7700 ファミリ の命令を記述する行です。対応する機械語データを生成します。
2. 構造化記述命令行
   PRE77 で処理する構造化記述言語を記述する行です。
3. 疑似命令行
   アセンブル時に必要な指示を行います。

4. マクロ命令行
   マクロ定義を記述する行です。アセンブラで処理します。

5. コメント行
   この行はアセンブラで処理しませんのでユーザが自由に使用できます。

3.2 行の構成

本節では、行の構成を示します。以下に表記上の規則を示します。

1. △、▲はスペース又はタブコードを示しています。
   △の部分は必須、▲の部分は省略可能です。

2. ラベルを記述する場合 ':'(コロン) は必須です。ただし、従来との互換性のためにコマンドオプション ("–U") により ':' を省略することも可能にしています。':' を省略した場合には疑似命令との間にスペース又はタブコードが必要になります。

3.2.1 命令行

命令行の構成を以下に示します。

▲ | ラベル: | ▲ | オペコード | △ | オペランド | ▲ | ; コメント | <RET>

▲† | オペコード | △ | オペランド | ▲ | ; コメント | <RET>

注意事項

† RASM77 は命令を予約語として識別しますので、ラベルがない場合は行の先頭からオペコードの記述が可能です。

3.2.2 構造化記述命令行

この行は、RASM77 では処理を行いません。この行の詳細については、第 2 部 第 3 章を参照して下さい。

3.2.3 疑似命令行

疑似命令行の構成を以下に示します。

▲ | ラベル: | ▲ | 疑似命令 | △ | オペランド | ▲ | ; コメント | <RET>

▲ | シンボル | △ | .EQU | △ | オペランド | ▲ | ; コメント | <RET>

▲‡ | 疑似命令 | △ | オペランド | ▲ | ; コメント | <RET>

注意事項

‡ RASM77 は疑似命令を予約語として識別しますので、ラベルがない場合は行の先頭から疑似命令の記述が可能です。
また、疑似命令によってはラベル記述のできないものもありますので詳細については、第 5 章、及び付録 B を参照してください。

3.2.4 マクロ命令行

マクロ命令行の構成を以下に示します。この行の詳細については、第 6 章、及び付録 E を参照してください。

▲ マクロ名: ▲ マクロ命令 △ オペランド ▲ ; コメント <RET>

注意事項

オペランドにデータが 2 つ以上ある場合、データ間を ‘ , ’(カンマ) で区切ってください。カンマの両側には、スペース又はタブコードが記述できます。

3.2.5 コメント行

コメント行は、行の最初の文字を ‘;’(セミコロン) で始めてください。コメント行の構成を以下に示します。

▲ ; コメント <RET>

3.3 欄の記述方法

3.3.1 シンボル/ラベル欄

RASM77 は、シンボルとラベルを区別[1] して管理しますが、名前の記述形式は同一です。以下に記述形式を示します。

1. 名前には、英数字、特殊文字 ‘_’ (アンダーライン) 及び ‘?’(クエッションマーク) が使用できます。ただし、 1 文字目に使用できる文字は、英字と特殊文字だけです。

2. 名前に、予約語は使用できません。RASM77 では、レジスタ名、フラグ名、疑似命令及びオペランド記述用演算子 (DP,DT,LG を含む) を予約語として処理します。

3. 大文字/小文字を区別します。したがって BIG と Big は異なった名前として判断します。

4. ラベル及びシンボルの文字数は ‘:’ を含めて最大 255 文字です。

5. 特殊文字 ‘ ..’(ピリオド 2 個) で始まるラベルは、マクロ命令及び PRE77[2] で作成されるラベルですので、使用しないでください。また、ピリオド 1 個及びピリオド 3 個以上で始まるラベルも使用できません。

[1] 疑似命令.EQU 又は “–D”パラメータにより定義したものをシンボル、これ以外で定義したものをラベルとして扱います。
[2] 構造化記述を処理するプリプロセッサ

ラベル欄を記述する場合、名前の直後に ':'(コロン) をつけることが必要です。ただし、従来との互換性のためにコマンドオプション ("–U") により ':' を省略することも可能にしています。':' を省略した場合には疑似命令との間にスペース又はタブコードが必要になります。シンボルとの区別を容易にするため、またエディタでのラベル検索を効率的に行うため ':' を付加した記述を推奨します。

3.3.2 オペコード/疑似命令欄

オペコード/疑似命令欄には、7700 ファミリの命令ニーモニック又は疑似命令を記述します。以下に記述形式を示します。

1. 命令ニーモニック及び疑似命令は、大文字/小文字を区別しません。したがって NOP 、nop いずれも有効です。

この欄の詳細については、第 4 章、第 5 章で説明します。

3.3.3 オペランド欄

オペランド欄には、命令又は疑似命令の実行対象の情報を記述します。以下に記述形式を示します。

1. オペランドにデータが 2 つ以上ある場合、データ間を ','(カンマ) で区切ってください。
2. カンマの両側には、スペース又はタブコードが記述可能です。

この欄の詳細については、3.4 節で説明します。

3.3.4 コメント欄

ユーザーの任意の情報を記述することができます。以下に記述形式を示します。

1. コメントの先頭に ';'(セミコロン) を付けて記述してください。
2. コメント欄にはどのような文字も記述できます。

3.4 オペランド欄の記述方法

3.4.1 データ形式

オペランド欄には、以下の 4 種類のデータ形式が記述可能です。

1. 数値定数
   - 数値定数の前に、'+' 又は '–' の演算子をつけて正又は負の値の表現ができます。'+' も '–' もない場合は、正の値として処理します。
   - 数値定数には、2 進、8 進、10 進、16 進数が使用できます。
   - 2 進数 ⇒ 数値の最後に 'B' 又は 'b' を付けて記述してください。
     例)　　`.BYTE  100110B`

- 8 進数 ⇒ 数値の最後に 'O' 又は 'o' を付けて記述してください。
  例)　　.BYTE　70o
- 10 進数 ⇒ 23 、256 のように整数のみで記述してください。
  例)　　.BYTE　100
- 16 進数 ⇒ 数値の最後に 'H' 又は 'h' を付けて記述してください。先頭が英文字 (A ～ F) で始まる場合は先頭に 0 を付加してください。
  例 1)　　.BYTE　64H
  例 2)　　.BYTE　0ABH

2. 文字列定数

- 文字列には、ASCII コードで定義されている文字が記述できます。
- 文字列定数は、'(シングルクォート) 、又は"(ダブルクォート) で囲んで記述してください。
  例)　　.BYTE　'A'　⇒　41_{16} を設定します。
- 文字列定数中に '¥' を用いた場合、'¥' 直後の文字を文字列データとして処理します。文字列に '¥' を用いる場合には、'¥¥' と記述してください。

3. ラベル又はシンボル

- ラベルは 24 ビット、シンボルは 32 ビット整数値をもちます。

4. 式

- 数式は、演算子、数値定数、文字列定数、ラベル又はシンボルの組み合わせで構成します。
- 式は左から右に計算します (演算子の優先順位はありません) 。
  例 1) 2*3　⇒　結果は 6 になります。
  例 2) 2+6/2　⇒　結果は 4 になります。
- 式は 32 ビット符号付き整数として処理されます。

3.4.2 演算子

表 3.1に、RASM77 で使用可能な演算子の一覧表を示します。

表 3.1: 演算子一覧表

演算子 [1]	内容
＋	加算
－	減算
＊	乗算
／	除算
%	除算の余り
＜＜	左ビットシフト
＞＞	右ビットシフト
&	ビットごとの AND
|	ビットごとの OR
^	ビットごとの排他的な OR
＋	正の数を表す単項演算子
－	負の数を表す単項演算子
～	ビットごとの反転を表す単項演算子
@[2]	直後のシンボル数値を文字列に展開する演算子
SIZEOF [3,4]	セクションの大きさを求める単項演算子
BANK [4]	ラベルの上位 8 ビット又はシンボルの 16 ～ 23 ビットを切り出す単項演算子
OFFSET [4]	ラベル又はシンボルの下位 16 ビットを切り出す単項演算子

注意事項

1. 演算は左から右へ行います (演算子の優先順位はありません) 。
   例 1) `2+6/2` ⇒ 結果は 4 になります。
   例 2) `1<<3+1` ⇒ 結果は 9 になります。

2. @演算子にて連結するシンボルは絶対値である必要があります。前方参照になるようなシンボルを指定した場合にはエラーを発生します。

3. SIZEOF の値は、参照セクションがリロケータブルかアブソリュートかに関係なくリンク時に決まります。したがってオペランド中に、SIZEOF 演算子が記述できる場所は、外部ラベルと同じ制限が適用されます (疑似命令.ORG 、.BLKB 等のオペランドには記述できません) 。

4. SIZEOF 、BANK 又は OFFSET と、ラベル、シンボル又はセクション名の間にはスペースを入れてください。
   例)
   ```
   .DT     BANK DATA1
   ```

第 4 章

命令の記述方法

4.1 アドレッシングモード

アドレッシングモードは、命令が処理対象データを指定する基本的な方式 (モード) のことです。7700 ファミリ では 28 通りのアドレッシングモードがあり、アドレッシングモードごとにオペランド形式が決まっています。

以下にオペランド記述に関係するアドレッシングモードの基本的な内容を示します。

1. アキュムレータ

   アキュレータを処理対象とする方法です。7700 ファミリ の CPU は、2 個のアキュムレータを持っているため、使用するアキュムレータ名をオペランドの最初に指定してください。なおアキュムレータ B を指定した場合、機械語データのバイト数は 1 バイト増加します。

   例)
   ```
   LDA     A,#IMMDATA
   ```

2. イミディエイト

   処理対象データをオペランドに直接記述する方法です。オペランドに記述する値は、'#' で示してください。

   例)
   ```
   LDM     #IMMDATA,MEMORY
   ```

3. ダイレクト

   ダイレクトページレジスタ (DPR) の持つ 16 ビットの値をベースアドレスとし、機械語データでベースアドレスに対するオフセット値 8 ビットを指定する方法です。バンクアドレスは 0 に固定です。アセンブラでの記述方法は、4.3 節で詳細に説明します。

4. アブソリュート

   データバンクレジスタ (DT) の持つ 8 ビットの値をページアドレスとし、機械語データで下位 16 ビットのアドレスを指定する方法です。アセンブラでの記述方法は、4.3 節で詳細に説明します。

5. アブソリュートロング

   オペランドで 24 ビットのアドレス値を指定する方法です (7700 ファミリの全メモリ空間が参照できます) 。オペランドに書いたラベルが、ダイレクト又はアブソリュートアドレッシングで処理できない場合にこのアドレッシングモードを使用します。アセンブラでの記述方法は、4.3 節で詳細に説明します。

6. インデクスト

   対象アドレスを、レジスタ X 又は Y で修飾する方法です。‘,’(カンマ) の後にレジスタ名を記述してください。

   例)　　ADC　　A,DATA1,X

7. ダイレクトインダイレクト

   処理対象データをメモリ間接で示す方法です。オペランドには対象アドレスを格納したメモリアドレスを記述し、メモリに 2 バイトの対象アドレスを格納します。上位 8 ビットには、データバンクレジスタ値を使用します。オペランドのメモリアドレスは、ダイレクトアドレッシングと同じ方法で示します。オペランドに記述する値は、() で示してください。

   例)　　LDA　　A,(INDATA)

   メモリに 3 バイトのアドレスを格納する場合、オペコードに L をつけてください。

   例)　　LDAL　　A,(INDATA)

8. アブソリュートインダイレクト

   処理対象データをメモリ間接で示す方法です。オペランドには対象アドレスを格納したメモリアドレスを記述し、メモリに 2 バイトの対象アドレスを格納します。上位 8 ビットには、プログラムバンクレジスタ値を使用します。オペランドのメモリアドレスは、アドレスの下位 16 ビットを示し、上位 8 ビットには、プログラムバンクレジスタ値を使用します。オペランドに記述する値は、() で示してください。

   例)　　JMP　　(PROCESS1)

   メモリに 3 バイトのアドレスを格納する場合、オペコードに L をつけてください。

   例)　　JMPL　　(PROCESS2)

9. レラティブ

   分岐先のアドレスを現在のプログラムカウンタ値との相対バイト数で表現する方法です。ただし、ソースプログラムには相対値そのものを記述することはできません。オペランドにラベル又は対象アドレスを記述すると、アセンブラが相対値を計算します。

   例)　　BRA　　LABEL

各アドレッシングモードごとの記述形式は、付録 E を参照してください。

4.2 データ長指定

7700 ファミリ CPU は、CPU 内部のフラグ状態でデータ長及びインデックスレジスタ長を制御することができます。表 4.1に、フラグとアセンブラの対応を示します。

表 4.1: CPU 内部のフラグとアセンブラとの対応

- データ長選択フラグ (m)

フラグ状態	内容	初期状態
$m = 0$	16 ビット演算	CPU リセット後の初期状態です。 RASM77 起動後の既定値です。
$m = 1$	8 ビット演算	

- インデックスレジスタ長選択フラグ (x)

フラグ状態	内容	初期状態
$x = 0$	16 ビット長	CPU リセット後の初期状態です。 RASM77 起動後の既定値です。
$x = 1$	8 ビット長	

各命令の機械語コードはフラグの状態には関係なく同一であるため、イミディエイトアドレッシングの場合を除きフラグ状態はアセンブラに影響を与えません。イミディエイトアドレッシングの場合、データ長に対応してオペランドに必要な即値データのバイト数が異なります。このため、アセンブラはフラグの状態に対応したコード生成を行う必要があります。RASM77 ではフラグ状態を、以下の 2 通りの方法で指定できます。

1. 命令のオペコード部で直接指定する。

```
例 1)    LDA.B   A,#50H ; 8 ビット即値 (50H) を指定
例 2)    LDA.W   A,#50H ; 16 ビット即値 (0050H) を指定
```

2. 疑似命令.INDEX 、.DATA により既定値を宣言する。

```
例)      .INDEX  16     ; インデックス長を 16 ビットに宣言する
         LDX     #200H  ; インデックス長の既定値 (16 ビット) で処理する
```

RASM77 では、データ長及び、インデックス長の既定値によってアセンブル制御が可能です。

コマンドパラメータ “–F” を指定したとき “M_FLAG” 及び、“X_FLAG” というシンボルを予約語として取り扱います。コマンドパラメータ “–F” を指定しないとき “M_FLAG” 及び、“X_FLAG” は任意のシンボル又は、ラベルとして使用することができます。

これらの予約語は条件付きアセンブルを行う擬似命令 “.IF” と共に使用することで、アセンブラが現在認識している m,x の内容を確認することができます。

表 4.2に “M_FLAG” 及び、“X_FLAG” の機能について示します。

表 4.2: M_FLAG,X_FLAG の機能表

シンボル	機能
M_FLAG	“.DATA”の設定値によって値が変化します。 データ長が 16 ビットに設定されているとき “M_FLAG”の値は 1 になります。 データ長が 8 ビットに設定されているとき “M_FLAG”の値は 0 になります。
X_FLAG	“.INDEX”の設定値によって値が変化します。 インデックス長が 16 ビットに設定されているとき “X_FLAG”の値は 1 になります。 インデックス長が 8 ビットに設定されているとき “X_FLAG”の値は 0 になります。

図 4.1に “M_FLAG”及び、“X_FLAG”を利用した条件アセンブルのプログラム例を示します。

```
BIT8:    .MACRO
         .DATA   8
         .INDEX  8
         .IF     M_FLAG
         sep     m,x
         .ELSE
         clp     m,x
         .ENDIF
```

図 4.1: データ長及び、インデックス長による条件アセンブル例

注意事項

1. 疑似命令自身は CPU 内部のフラグを操作する命令を生成しません。したがって疑似命令の宣言値と CPU の内部状態が一致するようユーザプログラムで管理してください。
2. データ長を命令のオペコード部で直接指定する場合のオペランドは、必ずイミディエイト値でなければなりません。

4.3 ダイレクトページレジスタ、データバンクレジスタ指定

7700 ファミリ CPU は、アドレス空間に応じてより少ないコードでメモリアクセスを行なえるよう、ダイレクトページレジスタ (DPR) 、データバンクレジスタ (DT) という名の 2 つのレジスタを内蔵しています。これらのレジスタを使用することにより、7700 ファミリ のメモリ空間 (16M バイト) を効率よくアクセスすることが可能です。以下に各レジスタの機能を説明します。

1. ダイレクトページレジスタ (DPR)

   16 ビット構成のレジスタです。機械語データのレベルでは、 ダイレクトアドレッシングは DPR に対する 8 ビットのオフセット値で対象アドレスを指定します (バンクは 0 に固定です) 。

   ソースプログラムでオペランドにラベルのみを記述すると、アセンブラが、現在の DPR 値から 00_{16} 〜 $0FF_{16}$ のオフセット値内にラベルが存在するかどうかを判断します。ダイレクトアドレッシングが使用可能な場合、アセンブラが オフセット値を計算し機械語データを生成します。

2. データバンクレジスタ (DT)

   8 ビット構成のレジスタです。機械語データのレベルでは、 アブソリュートアドレッシングは DT の値をバンクアドレスとする下位 16 ビットのアドレスを指定します。

   ソースプログラムでオペランドにラベルのみを記述すると、アセンブラが、現在の DT 値とラベルの上位 8 ビットが同じかどうかを判断します。アブソリュートアドレッシングが使用可能な場合アセンブラが、下位 16 ビットを計算し機械語データを生成します。

表 4.3に、DPR 、DT とアセンブラの対応を示します。

表 4.3: DPR 、DT の状態とアセンブラの対応

レジスタ名	初期状態
DPR	CPU リセット後は 0000_{16} になります。 RASM77 起動後の既定値は 0000_{16} です。
DT	CPU リセット後は 00_{16} になります。 RASM77 起動後の既定値は 00_{16} です。

4.4 アドレッシングモードの選択

RASM77 では、命令のオペランドにシンボル、ラベル、絶対数値を記述したとき、アドレッシングモードは以下の 3 種類のモードに大別できます。

1. ダイレクトアドレッシングモード
2. アブソリュートアドレッシングモード
3. アブソリュートロングアドレッシングモード

RASM77 は、任意のアドレッシングモードを選択することができます。アドレッシングモードの選択方法は操作の対象となるシンボル及び絶対数値と、ラベルによって異なります。また、ラベル操作に関してはダイレクトページレジスタ (DPR) とデータバンクレジスタ (DT) の値が直接アドレッシングモードの選択に関わってきます。

ここでは前の項で説明しましたDPR レジスタとDT レジスタの値の設定方法を説明した後に、シンボル及び絶対数値、ラベル操作時のアドレッシングモードに関して説明します。

4.4.1 ダイレクトページレジスタとデータバンクレジスタ値の設定方法

RASM77 では、使用するダイレクトページとバンクを変更する場合、ダイレクトページレジスタ値とデータバンクレジスタ値を、.DP と.DT 疑似命令でそれぞれ宣言した上で以下の例に示す方法で変更する必要があります。

例)

```
        .DP             100H    ←ダイレクトページレジスタ値を100H(100Hから1FFH)
        .DT             1       ←データバンクレジスタ値を1(バンク1)
        LDA             A, #100H
        TAD
        LDT             #1
```

.DP と.DT 疑似命令のオペランドには直接指定するダイレクトページの開始アドレス値とバンク値を数値で記述する外に、ダイレクトページ名ラベル、バンク名ラベルで指定することができます。

以下の例に示すように、samp2.a77 の PRO セクションで samp1.a77 で記述したDLAB と work のラベルを参照する場合、ダイレクトページ名ラベル DPR100 とバンク名ラベル BANK2 で DPR と DT レジスタ値を宣言することができます。

例)

[samp1.a77 のソースファイル]

```
        .SECTION   DATA1
        .ORG       100H
DPR100:
DLAB: .BLKW      2

        .SECTION   DATA2
        .ORG       20000H
BANK2:
work: .BLKB      2
```

[samp2.a77 のソースファイル]

```
   .SECTION   PRO
   .DPEXT     DPR100:DLAB
   .DTEXT     BANK2:work

   .DP        OFFSET  DPR100 ←ダイレクトページレジスタ値を 100(100H から 1FFH)
   .DT        BANK    BANK2 ←データバンクレジスタ値を 2(バンク 2)
   LDA        A, #OFFSET DPR100
   TAD
   LDT        #BANK BANK2
```

.DP と.DT のオペランドに記述したダイレクトページ及びバンクの有効範囲はリンク時に判定されます。

4.4.2 シンボル・絶対数値操作時のアドレッシングモード

データのパブリック指定疑似命令に.PUB 、外部参照指定疑似命令に.EXT を使用し、命令のオペランド中のシンボル・絶対数値に以下のアドレッシングモード指定子を付加して記述した場合、DPR と DT レジスタの設定値に関係なく強制的にアドレッシングモードが選択できます (.DP 又は.DT 疑似命令のオペランドに OFF を記述した場合は例外) 。

- DP: ..... ダイレクトアドレッシング
- DT: ..... アブソリュートアドレッシング
- LG: ..... アブソリュートロングアドレッシング

例)

```
        .EXT            SYM1            ←他のファイルでパブリック指定 (.PUB)

        .SECTION        PRO
        AND             A, DP:SYM1      ←ダイレクトアドレッシング
        AND             A, DT:SYM1      ←アブソリュートアドレッシング
        AND             A, LG:SYM1      ←アブソリュートロングアドレッシング
```

また、対象となる命令がダイレクトモード、アブソリュートモード、又はアブソリュートロングモードを選択できる場合、シンボルがアブソリュートな値（絶対数値）を持つとき、その命令が記述された場所の DPR と DT レジスタの値と比較し、最もメモリ効率が良くなるアドレッシングモードを選択します。

例)

```
        .SECTION        DATA
        .ORG            100H
LAB1:   .BLKW           1

        .SECTION        PRO
        .DT             0
        .DP             100H
        LDA.W           A, #100H
        TAD
        LDT             #0

        AND             A, LAB1         ←ダイレクトアドレッシング
```

4.4.3 ラベル操作時のアドレッシングモード

データのパブリック指定疑似命令に.PUB 、外部参照指定疑似命令に.EXT を使用し、命令のオペランド中のラベルに以下のアドレッシングモード指定子を付加して記述した場合、DPR と DT レジスタの設定値に関係なく強制的にアドレッシングモードが選択できます (.DP 又は.DT 疑似命令のオペランドに OFF を記述した場合は例外) 。

- DP: ..... ダイレクトアドレッシング
- DT: ..... アブソリュートアドレッシング
- LG: ..... アブソリュートロングアドレッシング

例)

```
        .EXT            LAB1            ←他のファイルでパブリック指定 (.PUB)

        .SECTION        PRO
        AND             A, DP:LAB1      ←ダイレクトアドレッシング
        AND             A, DT:LAB1      ←アブソリュートアドレッシング
        AND             A, LG:LAB1      ←アブソリュートロングアドレッシング
```

また、以下の外部参照指定疑似命令を使用して、データの参照時のアドレッシングモードを指定する方法があります。このとき、オペランドにはアドレッシングモード指定子を省略することができます。

- .DPEXT ..... ダイレクトアドレッシング
- .DTEXT ..... アブソリュートアドレッシング
- .EXT ..... アブソリュートロングアドレッシング

例)

```
        .DPEXT          LAB1            ←他のファイルでパブリック指定(.PUB)
        .DTEXT          LAB2            ←他のファイルでパブリック指定(.PUB)
        .EXT            LAB3            ←他のファイルでパブリック指定(.PUB)
        .SECTION        PRO
        AND             A, LAB1         ←ダイレクトアドレッシング
        AND             A, LAB2         ←アブソリュートアドレッシング
        AND             A, LAB3         ←アブソリュートロングアドレッシング
```

対象となる命令がダイレクトモード、アブソリュートモード、又はアブソリュートロングモードを選択できる場合、データが.DPEXT 又は.DTEXT でダイレクトページ名ラベル又はバンク名ラベルを付加して記述されたリロケータブルな値を持つとき、.DP 又は.DT に記述されたダイレクトページ名ラベル又はバンク名ラベルとの判定を行い、適合するアドレッシングモードを選択します。

例)

```
  .DPEXT     directpage_name:DPLAB ← directpage_name:ダイレクトページ名ラベル
  .DTEXT     databank_name:DTLAB   ← databank_name:バンク名ラベル

  .SECTION   PRO
  .DP        OFFSET  directpage_name
  .DT        BANK    databank_name

  AND        A, DPLAB          ←ダイレクトアドレッシング
  AND        A, DTLAB          ←アブソリュートアドレッシング
```

4.4.4 アドレッシングモードの選択禁止

RASM77 で選択されるアドレッシングモードを意図的に禁止することができます。ダイレクトアドレッシングモードを使用しないときは、.DP のオペランドに OFF を記述してください。また、アブソリュートアドレッシングモードを使用しないときは、.DT のオペランドに OFF を記述してください。

ダイレクトアドレッシングモードを禁止した場合、アブソリュート又はアブソリュートロングのアドレッシングモードを選択します。また、アブソリュートアドレッシングモードを禁止した場合、ダイレクト又はアブソリュートロングのアドレッシングモードを選択します。

.DP 又は.DT のオペランドに OFF を記述した場合、表 4.4に示す規則に従ってアセンブルします。

表 4.4: アドレッシングモード選択規則

記述形式	RASM77 の処理
DP:ラベル	エラー
DP:シンボル	ダイレクトページアドレッシングモード選択
DP:絶対値	ダイレクトページアドレッシングモード選択
DT:ラベル	エラー
DT:シンボル	アブソリュートアドレッシングモード選択
DT:絶対値	アブソリュートアドレッシングモード選択

例)

```
        .DPEXT          DPLAB
        .SECTION        DATA
LAB:    .BLKB           1

        .SECTION        PRO
        .DP             OFF             ←ダイレクトページアドレッシングを禁止

        LDA             A, DPLAB        ←アブソリュートロングアドレッシング
        STA             A, DP:ADDR1     ← Error 22: Value is out of range
```

注意事項

1. リロケータブルなローカルラベルをダイレクト又はアブソリュートアドレッシングで使用する場合、オペランドに “DP:”又は “DT:”を記述してください。ローカルラベルのみを記述した場合、アブソリュートロングアドレッシングを使用します。
2. 疑似命令.DPEXT で指定したラベルをオペランドに記述し、そのラベルに対して演算を行った場合、演算結果に対してもダイレクトアドレッシングでのコード生成を行います。

   例)
   ```
   .DPEXT  WORKA     ; ダイレクトページの外部ラベルを指定する
   STA     A,WORKA+1 ; 演算結果もダイレクトページとして扱う
   ```

   この場合、命令 STA はダイレクトアドレッシングによりアセンブルします。ただし、演算結果がダイレクトアドレッシングの範囲内かどうかは、リンク時に確認されます。.DTEXT で指定されたラベルに対しても同様に処理されます。
3. コマンドオプション “–Q”を指定することにより、.DPEXT , .DTEXT で指定されたラベルに対して “LG:” 等の記述による他のアドレッシングモードを指定した命令行に対してワーニングを発生します。

第 5 章

疑似命令の記述方法

5.1 疑似命令の機能

疑似命令は、命令が目的とする機械語データを生成するようにアセンブラに対する指示[1]を行います。RASM77 は、52 個の疑似命令を用意していますが、これらは機能的に以下の 5 つのグループに分類できます。

1. アセンブル制御
   - 疑似命令自身はデータを生成しませんが、命令に対応する機械語データの生成を制御します。
   - アドレスの更新には影響しません。
   - このグループは以下の 14 個の疑似命令を含みます。

.INDEX .DATA	データ長宣言
.DT .DP	データバンク/ダイレクトページレジスタ値宣言
.IF .ELSE .ENDIF	条件付きアセンブル
.INCLUDE	ファイル読み込み
.EQU	同義定義
.END	プログラム終了指定
.ASSERT	メッセージ出力
.ERROR	アセンブルエラー宣言
.DEFINE	文字列定義
.MCU	7700 シリーズ名定義

[1] 疑似命令の名称は、アセンブラに対する既定値を指示するものを “宣言”、出力ファイルに影響するものを “指定” と呼んでいます。

2. アドレス制御

   - アドレスの更新を行います。
   - データ設定疑似命令は定数データの生成を行います。
   - このグループは以下の 10 個の疑似命令を含みます。

.ORG	アドレス宣言
.BLKB .BLKW .BLKA .BLKD	RAM 領域確保
.BYTE .WORD .ADDR .DWORD	データ設定
.EVEN	アドレス補正

3. リンク制御

   - リンク処理に関係する制御を行います。
   - このグループは以下の 8 個の疑似命令を含みます。

.SECTION	セクション名指定
.DPEXT .DTEXT .EXT .PUB	グローバルラベル名指定
.OBJ .LIB	リンクファイル名指定
.VER	バージョン指定

4. リスト制御

   - PRN ファイル出力に関係する制御を行います。
   - このグループは以下の 7 個の疑似命令を含みます。

.PAGE	改ページ及びタイトル指定
.COL .LINE	リスト形式 (カラム数/行数) 指定
.LIST .NLIST	リスト出力/抑止指定
.LISTM .NLISTM	マクロ展開行リスト出力/抑止指定

5. ソースレベルデバッグサポート

   - ソースラインデバッグに必要な情報をオブジェクトファイルへ出力します。
   - このグループは以下の 6 個の疑似命令を含みます。

.CLINE	行頭情報出力
.FUNC .ENDFUNC	ファンクション開始/終了指定
.LANGUAGE	使用言語名出力
.POINTER	ポインタ長設定
.SOURCE	ソースファイル名設定

6. 予約

   - 将来の拡張のための予約疑似命令です。これらの疑似命令はアセンブル処理には影響を与えません。
   - このグループは以下の 9 個の疑似命令を含みます。

.PROGNAME	プログラム名宣言
.IO .ENDIO .RAM .ENDRAM	領域名宣言
.PROCMAIN .PROCSUB .PROCINT .ENDPROC	モジュール名宣言

以下に各グループの機能を説明します。

5.2 アセンブル制御

5.2.1 データ長宣言

.INDEX

インデックスレジスタ長選択フラグ (x) の既定値を宣言します。詳細内容は、4.2 節を参照してください。

.DATA

データ長選択フラグ (m) の既定値を宣言します。詳細内容は、4.2 節を参照してください。

5.2.2 ダイレクトページレジスタ、データバンクレジスタ値宣言

.DP

ダイレクトページレジスタの既定値を宣言します。アセンブラはこの宣言の DPR 値にしたがって、この行以降のアドレッシングモードの最適化を行います。詳細内容は、4.3 節を参照してください。

.DT

データバンクレジスタの既定値を宣言します。アセンブラはこの宣言の DT 値にしたがって、この行以降のアドレッシングモードの最適化を行います。詳細内容は、4.3 節を参照してください。

5.2.3 条件付きアセンブル

.IF .ELSE .ENDIF

シンボル値の内容により、アセンブルを行う場所を選択します。複数仕様に対応するプログラムを 1 つのソースプログラムで管理する場合、テストルーチンのアセンブルを制御する場合等に使用できます。

5.2.4 ファイル読み込み

.INCLUDE

この命令を記述した場所に、他のファイル内容を読み込みます。大きなソースプログラムを分割して編集する場合に使用できます。

5.2.5 同義定義

.EQU

シンボルに絶対値を定義します。同一のプログラム中に再定義が可能です。なお、再定義されているシンボルを前方参照した場合の参照値は、最終の定義値となります。

5.2.6 アセンブル終了指定

.END

アセンブルソースの終了を宣言します。アセンブラは、この行以降の内容を処理しません。

5.2.7 メッセージ出力

.ASSERT

画面上に指定の文字列を表示します。

5.2.8 アセンブルエラー出力

.ERROR

画面上に指定の文字列を表示しアセンブルを中止します。アセンブラは、この行以降の内容を処理しません。

5.2.9 文字列定義

.DEFINE

シンボルに文字列を設定します。

5.2.10 7700 シリーズ名定義

.MCU

命令生成を行う MCU タイプを指定します。

5.3 アドレス制御

5.3.1 アドレス宣言

.ORG

次の行のアドレスを宣言します。この疑似命令を記述したセクションは絶対属性となり、リンク時にアドレス指定を行うことはできません。割り込みベクタ等、アドレスが固定している領域に使用できます。

5.3.2 領域確保

.BLKB .BLKW .BLKA .BLKD

オペランドで指定した容量のメモリ領域を、RAM 領域に確保します。

5.3.3 データ設定

.BYTE .WORD .ADDR .DWORD

オペランドで指定したデータを、ROM 領域に生成します。

5.3.4 アドレス補正

.EVEN

奇数アドレスを偶数に補正します。現在のアドレスが偶数の場合は、何も行いません。文字列データのようなバイト単位のデータ領域で、各データの先頭を偶数アドレスに設定したい場合等に使用できます。

5.4 リンク制御

5.4.1 セクション名指定

.SECTION

この行以下のプログラムのセクション名を指定します。RASM77 では、プログラムを記述する前に必ずこの疑似命令でセクション名を指定してください。

5.4.2 グローバルラベル名指定

.DPEXT .DTEXT .EXT

外部参照するラベル及びシンボル名を指定します。なお、ここで指定したラベル名は必ず他のファイルでパブリック指定されていなければなりません。

.PUB

このファイル中で定義しているラベル又はシンボルを他のファイルで参照可能であることを指定します。

5.4.3 リンクファイル名指定

.OBJ .LIB

リンク対象のリロケータブルファイル名及びライブラリファイル名を指定します。ここで宣言したファイルは、リンカが自動的に参照するためリンク時のコマンドを簡略化できます。

5.4.4 バージョン指定

.VER

リロケータブルファイルのバージョン名を指定します。リンク時にバージョン確認を行う “–V” コマンドパラメータを指定すると、リロケータブルファイル間のバージョン名の一致を確認できます。

5.5 リスト制御

.PAGE

リストの改ページ及びタイトルを指定します。

.COL .LINE

リストのカラム数、行数を指定します。これらの疑似命令は、ソースファイル中に 1 回だけ記述できます。

.LIST .NLIST

PRN ファイルへのリスト出力制御を行います。プログラムの一部をデバッグする場合等で、リストの一部のみが必要な時に使用してください。

.LISTM .NLISTM

PRN ファイルへのマクロ展開行のリスト出力制御を行います。

5.6 ソースレベルデバッグサポート

.CLINE

ソースデバッグに必要な行番号を設定します。

.FUNC .ENDFUNC

ファンクション (サブルーチン) の開始、終了を指定します。

.LANGUAGE

使用言語の情報を設定します。

.POINTER

C コンパイラで使用されたポインタ変数のバイト長を設定します。

.SOURCE

ソースデバッグに必要なソースファイル名を設定します。

注意事項

1. 以上の擬似命令は、PRE77 及び、C コンパイラが出力するものです。これらの擬似命令をソースプログラムに記述した場合、エラーを出力することがあります。

5.7 予約疑似命令

予約疑似命令は、アセンブラが将来の拡張のために予約している疑似命令です。これらの疑似命令は、ソースファイルに記述してもエラーにはなりません。また、アセンブル結果にも影響しません。一般的には、市販の文字列検索プログラムと組み合わせてソースファイルの内容を確認する場合等に使用できます。

例) ソースファイル中に以下のように疑似命令.PROCMAIN を記述しておけば、文字列検索プログラムで ".PROCMAIN"を検索することにより、カレントディレクトリの全ソースファイルのメインプログラム名リストを作成できます。

```
.PROCMAIN START_KEY_SCAN    ; キースキャンプログラム部エントリ
```

第 6 章

マクロ命令

6.1　マクロ命令の機能

7700 ファミリ のアセンブリ言語を使用したプログラムをマクロとして定義しておくことにより、ユーザーはその際につけた名前 (マクロ名) をソースプログラムのオペランド欄にオペコードと同様に記述することができます。したがって、いろいろなマクロ定義を用意しておけば、プログラミングの際に 7700 ファミリ を拡張した CPU として利用することができます。このようにマクロ機能は、ユーザーが自分自身のプログラミング環境を整えることができる機能を提供します。

6.2　マクロ命令の種類

マクロ命令は、RASM77 に組み込まれているマクロとユーザーが定義するマクロの 2 種類に分類できます。

1. システムマクロ命令
   - .REPEATI ～ .ENDM
     オペランドに記述された引数の個数回、繰り返し処理を行います。
   - .REPEATC ～ .ENDM
     オペランドで引数として与えた文字数回、繰り返し処理を行います。
   - .REPEAT ～ .ENDM
     オペランドで指定した回数、繰り返し処理を行います。
2. ユーザーマクロ命令
   - .MACRO ～ .ENDM
     マクロ命令を定義します。
   - .EXITM
     マクロ展開を強制的に打ち切ります。
   - .LOCAL
     マクロ定義の中で使用されているラベルをマクロ内ローカルラベルにします。

注意事項

1. システムマクロ命令は単独で使用することもできますが、ユーザーマクロ定義中でも自由に使用することができます。
2. ユーザーマクロを使用する場合は、前もってマクロ定義をしておかなければなりません。このためマクロ定義は、通常プログラムの最初に定義するか、先頭でマクロ定義のファイルを.INCLUDE 疑似命令で取り込んでおきます。ユーザーマクロ定義のネスティングは 20 レベルまでです。
3. マクロ定義のファイルを、別のファイル (マクロライブラリ) としておけば、プログラムの初めにインクルードするだけでマクロが使用できるようになり、各プログラムごとにマクロ定義を記述する必要がなくなります。
4. マクロ展開された部分は、プリントファイル中ソースの横に '+' で示されます。
5. .LOCAL 宣言されたラベルは、その順に..n(n は 10 進数で 0 ～ 65535) というラベルを割り当ててアセンブルします。RASM77 は、今後の拡張のために .. で始まるラベルは予約しているラベルです。したがってユーザーは、.. で始まるラベルは使用できませんので注意してください。
6. マクロ名は大文字／小文字を区別しています。したがって MAC と Mac は異なったマクロ名として判断します。

6.3 マクロ演算子

表 6.1に、マクロ命令で使用可能な演算子の一覧表を示します。

表 6.1: マクロ演算子一覧表

演算子	内容
¥[1]	マクロの引数として使用できない特殊文字 (' ', ',', '¥') の前において '¥' の後の文字を引数として認識させます。 [書式] ¥文字
;; [2]	マクロ定義において展開を行わないコメントを定義します。 [書式] ;; コメント
" [3]	マクロの呼び出し時に、引数中にスペース、タブ、',(カンマ)' 、 及び予約語等が含まれる場合に用います。 [書式] " 文字列 "
$ [4]	マクロの引数を文字列に連結する場合に用います。 [書式] (1) 文字列$引数 (2) 引数$文字列

注意事項

1. エスケープキャラクタの働きをして次の文字の特別な意味を打ち消します。

   例)
   [マクロ定義]

```
DATA:  .MACRO VAL
       .BYTE  VAL
       .ENDM
```

   [呼び出し例]

```
DATA   "¥"HELLO !¥""
```

   [マクロ展開]

```
       .BYTE  "HELLO !"
       .ENDM
```

2. マクロの展開結果をプリントファイルに出力する場合、';' (セミコロン１個) で開始するコメントについてはマクロ展開される度に出力されますが、';;'(セミコロン２個) で始まるコメントは出力されません。

   例)
   [マクロ定義]

```
LOOP:  .MACRO
       .LOCAL LOOP1
       LDA    A,#20          ;  COMMENT
LOOP1: DEC    A              ;;  COMMENT
       BNE    LOOP1          ;;  COMMENT
       .ENDM
```

   [呼び出し例]

```
       LOOP
```

   [マクロ展開]

```
       LDA    A,#20          ; COMMENT
  ..0: DEC    A
       BNE    ..0
       .ENDM
```

3. オペランドで引数として与えた中にスペース、タブ、,(カンマ) 、予約語等が含まれる場合は全体を‘ ” ’(ダブルクオート) で囲んで表します。

例)

[マクロ定義]

```
SUB:    .REPEATI        INST,"NOP","LDA A,#1","RTS"
        INST
        .ENDM
```

[マクロ展開]

```
SUB:
        NOP
        LDA A,#1
        RTS
        .ENDM
```

4. 以下のような記述を可能とします。

例)

[マクロ定義]

```
W_LOAD:  .MACRO  MEM
         LDA     A,MEM$_L
         LDA     B,MEM$_H
         .ENDM
```

[呼び出し例]

```
         W_LOAD  DATA
```

[マクロ展開]

```
         LDA     A,DATA_L
         LDA     B,DATA_H
         .ENDM
```

第 7 章

操作方法

7.1　起動方法

RASM77 を実行するするためには、以下の情報 (入力パラメータ) を入力する必要があります。

1. ソースファイル名 (必須項目)
2. コマンドパラメータ

RASM77 では、これらの情報をコマンド行から指定します。7.2 節で入力パラメータについて説明し、7.3 節で例を示しながらコマンド行入力方法を説明します。

7.2　入力パラメータ

7.2.1　ソースファイル名

1. アセンブル対象のソースファイル名を指定します。ソースファイル名は必ず入力してください。ソースファイル名は 1 個のみ指定できます。
2. ファイル属性 (.A77) を省略した場合、既定値として属性.A77 を選びます。
3. ファイル名をフルネームで指定することにより、.A77 以外の属性 (例 .ASM) のファイルもアセンブル可能です。
4. ファイル名はディレクトリパス指定が可能です。ファイル名のみを指定した場合は、カレントドライブのカレントディレクトリ中のファイルを処理します。
   以下に C ドライブの WORK ディレクトリ中の TEST.A77 ファイルをアセンブルする例を示します。
   例)　`A>RASM77 C:¥WORK¥TEST<RET>`

7.2.2　コマンドパラメータ

1. コマンドパラメータは大文字/小文字どちらでも有効です。
2. 各パラメータは同時に複数指定することができます。この場合は、各パラメータをスペースで区切って入力してください。

表 7.1に、コマンドパラメータの内容を示します。

表 7.1: コマンドパラメータ一覧表

コマンドパラメータ	内容
−.	画面へのすべてのメッセージ出力を抑止します。バッチファイル等で RASM77 を実行する場合に、画面に何も表示したくない時にお使いください。
−A	条件アセンブル記述内の条件が偽となる部分にエラーとなる記述があってもエラーを出力しません。
−B	ビット長の整合性をチェックします。[1] この指定をした場合、疑似命令 “.BYTE” , “.WORD” , “.BLKB” , “.BLKW” で宣言したローカルラベルの参照時に、“.DATA” , “.INDEX” で宣言しているビット長とが一致していない場合にワーニング 6 を出力します。
−C	ソースデバッグ情報をオブジェクトファイルに出力します。デバッグ時にソースデバッグを行う場合、アセンブル時にこのオプションを指定してください。
−D	シンボルに数値を設定します。コマンドで設定されたシンボルは、疑似命令.EQU で定義されたシンボルと同等に扱われます。指定の書式は、次のようになります (複数のシンボルを同時に定義する場合、‘:’ で区切ってください) 。 –D シンボル=数値 [:シンボル=数値 .... :シンボル=数値] 例) `A>RASM77 SRCFILE -DS1=10:S2=20<RET>`
−E	TAG ファイルの生成及びエディタの起動を行います。エディタのプログラム名指定の書式は、次のようになります。[2] –E[エディタ名] 例) `A>RASM77 SRCFILE -EMI<RET>` [ ] の部分は省略可能です。省略した場合、TAG ファイルの生成のみを行います。エディタ名が指定された場合はアセンブル終了後、TAG ファイルを引き数としてエディタを起動します。 エラーが発生しなかった場合、エディタの起動は行いません。

コマンドパラメータ	内容
−F	RASM77 が現在仮定している m 、x フラグの状態を保持しているシンボル ‘M_FLAG’ 、‘X_FLAG’ を利用可能にします。 このコマンドパラメータを使用して、‘M_FLAG’ 、‘X_FLAG’ を条件アセンブル疑似命令 ‘.IF’ のオペランドで ‘8’ または ‘16’ との条件判断を行なうことで、RASM77 が仮定している上記フラグの状態 (8 ビット長か 16 ビット長) を判別しマクロ内部等での処理を分けることが可能となります。
−L	PRN ファイルを生成します。この指定がない場合 PRN ファイルは生成しません (ただし、−M オプションを指定した場合はこの指定がなくても PRN ファイルを生成します) 。
−LC	.IF 命令による条件アセンブル部の条件偽の部分、@演算子及び.DEFINE 疑似命令での変換文字列を PRN ファイルへ出力します。この指定がない場合、PRN ファイルへはこれらの部分は出力しません。
−LD または−LD0	‘.DEFINE’ 疑似命令、‘@’ 演算子を用いた行について、置換された結果行のみをプリントファイルに出力します。
−LD1	‘.DEFINE’ 疑似命令、‘@’ 演算子を用いた行について、置換前の行と置換された結果の行をプリントファイルに出力します。
−M	マクロ展開部を PRN ファイルへ出力し、PRN ファイルを生成します。この指定がない場合、PRN ファイルへはマクロ展開部は出力しません。
−N	PRN ファイル最後のシンボルリスト出力を抑止します。

コマンドパラメータ	内容
−Q	疑似命令 “.EQU” による再設定時にワーニング 7 を出力します。この指定がない場合には、同一シンボルに対して異なる値を設定した場合にもワーニングとなりません。 アブソリュートロングアドレッシングモードを選択するために、“LG:” を記述した場合にワーニング 8 を出力します。この指定がない場合には、“LG:”を記述してもワーニングとなりません。
−O	生成ファイルの出力先パスを指定します。パスにはディレクトリ又はドライブ名が指定できます。この指定がない場合、ソースファイルと同じパスに出力します。 指定の書式は、次のようになります。 −O パス名 例) A>RASM77 SRCFILE -OB:¥WORK<RET>
−P	プリントファイルのマクロ展開部分の行番号を更新しません。
−S	ローカルシンボル情報をオブジェクトファイルに出力します。デバッグ時にローカルシンボルを含むデバッグを行う場合アセンブル時にこのオプションを指定してください。[3]
−T	ユーザマクロ内でエラーが発生した場合、マクロ定義行ではなくマクロ呼出し行の行番号情報をタグファイルに出力します。
−U	ラベル直後の ‘:’(コロン) の省略を許します。
−V	RASM77 のバージョン番号を画面に出力して終了します。
−X	アセンブル終了後、クロスリファレンサ CRF77 を起動します。[4] 例) A>RASM77 SRCFILE -X<RET>

注意事項

1. “SEM” , “CLM” 等の命令によってビット長を変化することはできません。また、外部参照には適用しません。

   RASM77 では ‘–B’ コマンドオプションを用いた場合には 8 ビットサイズとして確保されたメモリ (RAM の場合には ‘.BLKB’ 、ROM の場合には ‘.BYTE’ 疑似命令にて確保されたメモリ) に対して、RASM77 が認識しているビット長 (‘.DATA’ 、‘.INDEX’ 疑似命令で設定されるビット長) が 16 ビットであった場合にワーニング 6 を出力します。

   また、逆に 16 ビットサイズ以上として確保されたメモリについて、RASM77 が認識しているビット長が 8 ビットであった場合にワーニング 6 を出力します。
   ただし、‘.B’ 、‘.W’ の記述を用いて直接命令のオペコード部分でビット長を指定した場合にはワーニング 6 は出力しません。

   以下にそのビット長の整合性を評価する命令を記します。

   ADC, AND, CMP, EOR, LDA, ORA, SBC, DIV, MPY, ASL, DEC, INC, LSR
   ROL, ROR, CLB, SEB, LDM, STA, CPX, CPY, LDX, LDY, STX, STY

2. エディタの起動は、MS–DOS の COMMAND.COM を介して間接的に行いますので、COMMAND.COM が MS–DOS のコマンドパス中に存在していることを確認してください。また、COMMAND.COM が存在するドライブ以外で作業する場合、CONFIG.SYS ファイル中に下記の指定を記述してください。コマンドパスの詳細及び CONFIG.SYS の設定については、MS–DOS の説明書を参照してください。以下に A ドライブのルートディレクトリに COMMAND.COM ファイルが存在する場合の設定の例を示します。

   例) `SHELL = A:¥COMMAND.COM A:¥ /P`

   カレントディレクトリ又はコマンドパス中にエディタが存在しない場合は、MS–DOS によりエラーを表示します。

3. このオプションを指定しない場合 RASM77 はローカルシンボル情報を出力しません。デバッグ時にローカルシンボル情報が必要な場合に使用してください。

4. カレントディレクトリ又はコマンドパス中に CRF77 が存在しない場合は、システムエラーになります。

7.3 入力方法

RASM77 は、OS のプロンプト状態でコマンド行を入力することにより起動します。図 7.1 に、起動コマンドの入力例を示します。

```
A>RASM77 TESTNAME -L -E <RET>
         ↓        ↓
アセンブル対象の   コマンドパラメータ
ソースファイル名
```

図 7.1: RASM77 の起動コマンド行入力例

コマンド行入力に誤りを検出すると、図 7.2のようにヘルプ画面を表示しアセンブルを中止します。
コマンド行入力が正常に行われるとアセンブルを開始します。アセンブルが終了するとエラー数、ワーニング数、全行数、コメント行数、セクション単位のメモリ容量を画面に出力します。アセンブルが正常に終了したときの画面表示例を図 7.3に示します。

```
A>RASM77<RET>
7700 Family RELOCATABLE ASSEMBLER V.5.00.00
Copyright 1998, MITSUBISHI ELECTRIC CORPORATION
AND MITSUBISHI ELECTRIC SEMICONDUCTOR SYSTEMS CORPORATION
All Rights Reserved.

Usage: rasm77 <filename> [options]
       -. : all messages suppressed
       -a : don't care error at conditional assembly FALSE block
       -b : bit length check
       -c : source line information output to .r77 file
       -d : define symbol ( syntax -dSYMBOL1=DATA1:SYMBOL2=DATA2 )
       -e : make tag file
       -f : x, m flag information symbol enable
       -l : make list file
       -lc: conditional assembly statements output to .prn file
       -ld: .DEFINE statements output control
       -m : macro expansion statements output to .prn file
       -n : don't make symbol list to .prn file
       -o : select drive and directory for output ( syntax -o/tmp )
       -p : print file number control
       -q : .EQU symbol multi define warning message output
       -s : local symbol data output to .r77 file
       -t : tag-file form change
       -u : don't care ':' at end of label
       -v : rasm77 version display
       -x : execute crf77
```

図 7.2: コマンド行エラー時のヘルプ画面

```
A>RASM77 TEST<RET>
7700 Family RELOCATABLE ASSEMBLER V.5.00.00
Copyright 1998, MITSUBISHI ELECTRIC CORPORATION
AND MITSUBISHI ELECTRIC SEMICONDUCTOR SYSTEMS CORPORATION
All Rights Reserved.
now processing pass 1  ( TEST.A77 )
----*----
now processing pass 2  ( TEST.A77 )
----*----

ERROR   COUNT    00000   (0000H)
WARNING COUNT    00000   (0000H)
TOTAL   LINE     00994   (03E2H) LINES
COMMENT LINE     00247   (00F7H) LINES
WORK          00000010 (00000AH) BYTES
DATA          00000054 (000036H) BYTES
PROG          00001938 (000792H) BYTES

A>
```

図 7.3: 正常終了時の画面表示

7.4 エラー

7.4.1 エラーの種類

RASM77 実行時に発生するエラーは、以下の原因によるものがあります。

1. OS に関するエラー
   ディスクやメモリ容量の不足等、RASM77 を実行する環境に関わるエラーです。付録 A エラーメッセージ一覧表を参照の上、OS のコマンドにより対応してください。
2. RASM77 のコマンド行入力に関するエラー
   RASM77 起動時のコマンド行入力に関わるエラーです。本章の内容を確認の上コマンドを再入力してください。
3. アセンブル対象のソースファイルの内容に関するエラー
   ラベルの 2 重定義、未定義シンボルの参照等のソースファイルの内容に関わるエラーです。該当箇所のソースファイル内容を修正して再度アセンブルを行なってください。
   アセンブルエラーを検出した場合リロケータブルファイルは生成しません。

RASM77 はエラー及びワーニングを検出すると、図 7.4の形式でエラー内容 (ファイル名、ファイル中の行番号、通し行番号、エラー番号及びエラーメッセージ) を画面と PRN ファイルに出力します。エラー情報については、.NLIST 擬似命令中でも PRN ファイルに出力します。付録 A のエラー番号順のエラー一覧表を参照の上対応してください。

```
A>RASM77 TEST<RET>
7700 Family RELOCATABLE ASSEMBLER V.5.00.00
Copyright 1998, MITSUBISHI ELECTRIC CORPORATION
AND MITSUBISHI ELECTRIC SEMICONDUCTOR SYSTEMS CORPORATION
All Rights Reserved.
now processing pass 1  ( TEST.A77 )
----*----
now processing pass 2  ( TEST.A77 )
-
  115 00F025  EAEA  BCC     LOOP2
TEST.ASM 115 ( TOTAL LINE 115 ) Error 18: Relative jump is out of range
  127 00F031  EAEAEA        LDA     A,#data
TEST.ASM 127 ( TOTAL LINE 127 ) Error 20: Reference to undefined label or symbol "data"
---*
  551 00F42B  EAEA  BRA     TEST2
TEST.ASM 551 ( TOTAL LINE 551 ) Error 20: Reference to undefined label or symbol "TEST2"
  593 00F4FC  EAEAEAEAEAEA   LDA     A,(work,x ; data set
TEST.ASM 593 ( TOTAL LINE 593 ) Error 23: "()" format error ";"
----

ERROR   COUNT    00004   (0004H)
WARNING COUNT    00000   (0000H)
TOTAL   LINE     00994   (03E2H) LINES
COMMENT LINE     00247   (00F7H) LINES
WORK          00000010 (00000AH) BYTES
DATA          00000053 (000035H) BYTES
PROG          00001938 (000792H) BYTES

A>
```

図 7.4: エラー表示例

7.4.2 OS への戻り値

バッチファイル等に実行コマンドを記述する場合、実行結果に応じて処理の内容を変えたい場合があります。RASM77 では、実行結果を表 7.2の 5 つのエラーレベルに分けて OS に返すようにしています。エラーレベルの利用方法については OS の説明書を参照してください。

表 7.2: エラーレベル一覧表

エラーレベル	実行結果の内容
0	正常終了
1	アセンブル対象のソースファイルの内容に関するエラー
2	RASM77 のコマンド入力に関するエラー
3	OS に関するエラー
4	^C(コントロールC) 入力による強制終了

7.5 環境変数

RASM77 は、以下の環境変数を使用しています。

1. TMP77
   アセンブルする時に作成するテンポラリファイルの作成ディレクトリ名を指定します。この環境変数設定されていない場合は、テンポラリファイルはカレントディレクトリに作成されます。

   例) `SET TMP77=A: ¥TMP`

2. INC77
   アセンブルする時にインクルードするファイルのディレクトリを指定します。.INCLUDE 疑似命令で指定したファイルが見つからない場合は、この環境変数に指定したディレクトリからファイルを読み込みます。ただし、".INCLUDE"オペランドに絶対パスが指定されている場合はこの環境変数は使用しません。

   例) `SET INC77=A: ¥INCLUDE`

付録 A

エラーメッセージ一覧表

A.1 システムエラー一覧表

アセンブル中にシステムエラーを検出すると、エラーメッセージを画面に表示しアセンブルを中止します。表 A.1にシステムエラー一覧表を示します。

表 A.1: システムエラー一覧表

エラーメッセージ	エラー内容と対策
Usage:rasm77 <filename> [-.] [-dSYMBOL1=DATA1] [-e] [-l] [-oPATH] [-x]	コマンド入力が誤っています。 ⇒ ヘルプ画面を参照して、コマンドを再入力してください。
Can't open xxx	該当ファイルが見つかりません。 ⇒ ソースファイル名を確認して再入力してください。
Can't create xxx	該当ファイルが生成できません。 ⇒ –O パラメータの指定を確認して再入力してください。
Out of disk space	ファイルを出力するためのディスク容量が不足しています。 ⇒ ディスク上に空き領域を作ってください。
Out of heap space	アセンブラが動作するために必要なメモリが不足しています。 ⇒ シンボル又はラベルの数を減らしてください。
Can't find crf77.exe	CRF77 が見つかりません。 ⇒ CRF77 をカレントディレクトリ又は OS のコマンドパス内のディレクトリにコピーしてください。
Can't find command.com for execute xxx	–E オプションで指定されたエディタを起動するために必要なファイル COMMAND.COM が見つかりません。 ⇒MS–DOS のコマンドパス指定を確認してください。

A.2 アセンブルエラー一覧表

アセンブルエラーを検出すると、エラーメッセージを画面と PRN ファイルに出力します。
表 A.2にアセンブルエラーとその内容を示します。

表 A.2: アセンブルエラー一覧表

エラー番号	エラーメッセージ	エラー内容と対策

1	Already had same statement	ソースファイル中に 1 回しか使用できない疑似命令を、2 回以上使用しています。
		例) `.LINE 60`
		`:`
		`.LINE 80`
		⇒ 宣言を 1 回にしてください。

2	Reference to forward label or symbol	疑似命令が前方のラベル又はシンボルを参照しています。 例) `.ORG TOP` `TOP:` ⇒ ラベル又はシンボルの定義を参照以前に行ってください。
3	Division by 0	数式中に 0 による割り算を含んでいます。 ⇒ 数式の記述を確認してください。
4	Illegal operand	オペランドに使用できない文字を含んでいます。 例) `LDA A,#$10` ⇒ オペランドの記述を確認してください。
5	Improper operand type	ニーモニックとオペランドの組み合わせが誤っています。 例) `ADC.B A,work` ⇒ 命令記述形式を確認してください。
6	Invalid label definition	ラベル定義ができない場所でラベル定義を行っています。 例 1) `LABEL1: .LINE 60` ⇒ ラベルを削除してください。 例 2) `LABEL2: .EQU 100` ⇒ ラベルをシンボルに変更してください。

エラー番号	エラーメッセージ	エラー内容と対策
7	Invalid symbol definition	シンボル定義ができない場所でシンボル定義を行っています。 例) `SYMBOL .LINE 60` ⇒ シンボルを削除してください。
8	Out of maximum program size	アドレスが 0FFFFFFH を超えています。 例) `.ORG 0FFFFF0H` `.WORD 1,2,3,4,5,6,7,8,9` ⇒ アドレスが範囲内になるようにプログラムを変更してください。
9	Label or symbol is multiple defined	同一のラベルが 2 回以上定義されています。 例) `MAIN: NOP` `MAIN: NOP` ⇒ ラベル名を確認してください。
10	Nesting error	.INCLUDE のネスティングレベルがオーバーになっています。 ⇒ ネスティングレベルがオーバーにならないようにプログラムを変更してください。
11	No .END statement	ソースファイル内に.END 文がありません。 ⇒ プログラムの最後に.END 文を記述してください。
12	No symbol definition	シンボルが記述されていません。 例) `.EQU 60` ⇒ シンボルを記述してください。
13	No ';' at the top of comment	コメント欄の先頭に ';'(セミコロン) がありません。 例) `LDA A,#CNT counter set` ⇒ コメント欄の先頭に ';' を付加してください。

エラー番号	エラーメッセージ	エラー内容と対策

14	Not in conditional block	対応する.IF 文がないのに、.ELSE 又は.ENDIF が記述されています (対応する.IF 文がエラーになっている場合もこのエラーが発生します)。
		例)
		`.IF DATA1`
		`:`
		`.ENDIF`
		`:`
		`.ELSE`
		`:`
		`.ENDIF`
		⇒.IF 文を確認してください。

15	Operand is expected	命令に必要なオペランドが不足しています。 例) `.BYTE` ⇒ オペランドの記述を確認してください。
16	Questionable syntax	ニーモニックのスペルに誤りがあります。 例) `ADD A,#DATA` ⇒ ニーモニックのスペルを確認してください。

17	Reference to multi defined label or symbol	重複定義されているラベル又はシンボルを参照しています。
		例)
		`MAIN: NOP`
		`MAIN: NOP`
		`BRA MAIN`
		⇒ ラベル名又はシンボル名を確認してください。

18	Relative jump is out of range	相対ジャンプ命令のジャンプ先アドレスが範囲内にありません。 ⇒ プログラムの再配置を行うか、命令を変更してください。
19	Label or symbol is reserved word	ラベル又はシンボルにレジスタと同一名称が使われています。 例) `A .EQU 1FFH` ⇒ ラベル名又はシンボル名を変更してください。
20	Reference to undefined label or symbol	未定義のラベル又はシンボルを参照しています。 ⇒ ラベル又はシンボルを確認してください。

エラー番号	エラーメッセージ	エラー内容と対策
21	Value error	データ記述形式に誤りがあります。 例) `ADC A,#'A` ⇒ データ記述形式を確認してください。
22	Value is out of range	データの範囲が許容値を越えています。 例) `ADC.B A,#100H` ⇒ オペランド記述形式を確認してください。
23	"()" format error	左カッコ'(' と右カッコ')' の数があっていません。 例) `ADC A,(WORK` ⇒ オペランド記述形式を確認してください。

24	Relocatable error	リロケータブルセクションの中で疑似命令.ORG を記述しています。
		例)
		`.SECTION PROG`
		`LDA A,WORK`
		`:`
		`.ORG 1000H`
		`LDA A,WORK`
		⇒ セクションを分けてください。

		シンボル定義疑似命令 .EQU のオペランドにリロケータブルな値を記述しています。
		例)
		`.EXT WORK0`
		`.EXT WORK1`
		`SYMBOL .EQU WORK0-WORK1`
		⇒ 疑似命令 .EQU のオペランドにはローカルな値を記述して下さい。

25	No .SECTION statement	疑似命令.SECTION を記述していません。 ⇒ プログラム記述の前に、疑似命令.SECTION を記述してください。
26	Reference to undefined section	未定義のセクション名を参照しています。 例) `LDA A,#SIZEOF UNDEF_SECT` ⇒ 該当セクションを確認してください。

エラー番号	エラーメッセージ	エラー内容と対策

27	Page error	外部参照指定疑似命令.DPEXT 又は.DTEXT で指定しているダイレクトページ名又はデータバンク名と、現在の DPR 値又は DT 値の既定値が異なっています。
		例)
		`.DPEXT PG1:LABEL`
		`.DP BANK PG2`
		`LDA A,LABEL`
		⇒ 外部参照宣言に対応する DPR、DT 値を宣言するか、疑似命令.DPEXT 又は.DTEXT のオペランドからダイレクトページ名又はデータバンク名を削除してください。(この場合、疑似命令.DP、.DT によって宣言した現在の既定値の DPR、DT 値に対して処理を行います)。

28	Section type mismatch	命令又はデータ設定疑似命令 (.BYTE 等) が、領域確保命令 (.BLKB 等) と混在しています。
		例)
		`LDA A,#WORK`
		`.BLKB 1`
		⇒ セクションを分けてください。

29	Function is multiple defined	.FUNC で指定したラベル名が 2 回以上定義されています。 例) `.FUNC FUNC_1` `.FUNC FUNC_1` ⇒ ラベル名を確認してください。

30	Macro nesting error	マクロ命令のネスティングがオーバーしています。
		例)
		`MAC: .MACRO DATA`
		`LDA A,DATA`
		`MAC2`
		⇒ マクロ命令のネスティングレベルを減らしてください。

31	No .ENDM statement	ソースファイル中に.ENDM 文がありません。 ⇒ マクロ定義の最後に.ENDM 文を記述してください。

エラー番号	エラーメッセージ	エラー内容と対策
32	Illegal mnemonic	記述されたニーモニックが MCU のタイプにあっていません。 ⇒.MCU 疑似命令で正しい MCU タイプを指定してください。
33	Illegal processor type	.MCU のオペランドに記述された MCU タイプが間違っています。 ⇒.MCU オペランドに正しい MCU タイプを指定してください。

A.3 ワーニング一覧表

ワーニングを検出すると、ワーニングメッセージを画面と PRN ファイルに出力します。
表 A.3にワーニングとその内容を示します。

表 A.3: ワーニング一覧表

ワーニング番号	ワーニングメッセージ	ワーニング内容と対策

1	Phase warning	1) 疑似命令.ORG で指定したアドレスが、それ以前のアドレスよりも前になっています。
		例)
		`.ORG 0E000H`
		`MAIN: LDA A,WORK`
		`:`
		`.ORG 0C000H`

		2) 命令がこの行より前方のラベル又はシンボルを参照しています。
		例)
		`LDA A,TBL,X`
		`:`
		`TBL: .BYTE 0,1,2,3,4`
		⇒ ラベル又はシンボルを参照する行より前で定義するか、オペランドに DP:又は DT:指定を記述してください。

2	.END statement in include file	疑似命令.END をインクルードファイル中に記述しています。 ⇒ .END をソースファイル中に記述してください。
3	Line number is out of range	.CLINE 命令のオペランドの値が 9999 を越えています。
4	Too many actual macro parameters	マクロ呼び出しでの仮引数より実引数の数が多くなっています。
5	Too few actual macro parameters	マクロ呼び出しでの仮引数が実引数の数より少なくなっています。
6	Bit length is different from label or symbol	定義と参照で異なったビット長でラベル又はシンボルを使用しています。
7	.EQU symbol is multiple defined	疑似命令.EQU で同一シンボルを 2 回以上参照しています。
8	Addressing mode warning	ラベルに指定された以外のアドレッシングモードを使用しています。
9	Value warning	オペランドに意味のない値を記述しています。

付録 B

疑似命令一覧

B.1 疑似命令一覧の見方

RASM77 で使用可能な疑似命令をアルファベット順に掲載します。表記上の規則を以下に示します。

1. 説明中の [ ] 内の部分は省略可能部分を示しています。
2. △、▲はスペース又はタブコードを示しています。△の部分は必須、▲の部分は省略可能を示します。
3. 以下の説明ではラベルと疑似命令間は▲で示しています。ラベルを記述する場合 ':'(コロン) は必須です。コマンドパラメータ "–U" により、':' の省略を許した場合には疑似命令との間にスペース又はタブコードが必要になります。

B.2 疑似命令一覧

.ADDR　アドレス (3 バイト) データ設定

書式

▲ [ラベル:] ▲.ADDR △数式

内容

- 3 バイトの定数データを設定します。
- データは下位バイトより設定します (PRN ファイルの OBJ 欄には上位バイトから設定します) 。
- 2 個以上のデータを設定する場合は、各データを ‘,’ で区切って記述してください。記述できる個数は 8 個までです。
- オペランドにグローバルラベルを記述できます。

記述例

```
TABLE:  .ADDR    sub1    ; sub1 の値を下位バイトより設定する
```

.ASSERT　メッセージ出力

書式

▲.ASSERT △’文字列’

内容

- 条件アセンブル命令 (.IF) と組み合わせて使用します。
- CRT に文字列を出力します。

記述例

```
.IF      JAPAN
.ASSERT  ’国内向け仕様でアセンブルします’
   :
.ELSE
.ASSERT  ’海外向け仕様でアセンブルします’
   :
.ENDIF
```

.BLKA　RAM 領域 3 バイト確保

書式

▲ [ラベル:] ▲.BLKA △数式

内容

- 数式で指定した大きさの領域を確保します (単位:3 バイト) 。
- 数式中に使用するラベル又はシンボルは、この行より前に定義してください。
- オペランドに、リロケータブルな値を持つラベルは記述できません。

記述例

```
label:    .BLKA    10        ; 30 バイトの領域を確保する
```

.BLKB　RAM 領域 1 バイト確保

書式

▲ [ラベル:] ▲.BLKB △数式

内容

- 数式で指定した大きさの領域を確保します (単位:1 バイト) 。
- 数式中に使用するラベル又はシンボルは、この行より前に定義してください。
- オペランドに、リロケータブルな値を持つラベルは記述できません。

記述例

```
label:   .BLKB    10        ; 10 バイトの領域を確保する
```

.BLKD　RAM 領域ダブルワード確保

書式

▲ [ラベル:] ▲.BLKD △数式

内容

- 数式で指定した大きさの領域を確保します (単位:4 バイト) 。
- 数式中に使用するラベル又はシンボルは、この行より前に定義してください。
- オペランドに、リロケータブルな値を持つラベルは記述できません。

記述例

```
label:  .BLKD   10      ; 40 バイトの領域を確保する
```

.BLKW　RAM 領域ワード確保

書式

▲ [ラベル:] ▲.BLKW △数式

内容

- 数式で指定した大きさの領域を確保します (単位:2 バイト) 。
- 数式中に使用するラベル又はシンボルは、この行より前に定義してください。
- オペランドに、リロケータブルな値を持つラベルは記述できません。

記述例

```
label:  .BLKW   10      ; 20 バイトの領域を確保する
```

.BYTE　バイトデータ設定

書式

▲ [ラベル:] ▲.BYTE △数式

内容

- 1 バイトの定数データを設定します。
- 2 個以上のデータを設定する場合は、各データを ',' で区切って指定してください。指定できる個数は 255 個までです。
- 文字列データを設定する場合は、文字列を ' ' でくくって指定してください。ただし、文字列に漢字は使用できません。
- 文字列データとして ¥を設定する場合は ¥¥と指定してください。
- オペランドに、グローバルラベルを記述できます。

記述例

```
label1: .BYTE   10      ; 0AH を設定する
label2: .BYTE   'A','B' ; 41H 、42H を設定する
label3: .BYTE   'ABCD'  ; 41H 、42H 、43H 、44H を設定する
```

.COL　カラム数指定 (既定値は 132)

書式

▲.COL △数式

内容

- リストの 1 行の文字数 (80 ～ 132) を指定します。
- 79 以下を指定した場合は 80 に、133 以上を指定した場合は 132 文字になります。
- 本疑似命令は、プログラム中に 1 回だけ記述できます。

記述例

```
        .COL    100     ; カラム数を 100 に設定する
```

.DATA　データ長宣言 (既定値は 16)

書式

▲.DATA △数式

内容

- CPU 内部のデータ長 (8 又は 16) を宣言します。数式の値が 8 であれば 8 ビット、16 であれば 16 ビットを示します。
- 本疑似命令は、M フラグと関連するアドレッシングモードのデータ長に影響します。
- SEM 、CLM 命令等でデータ長を変更する場合には、本疑似命令で宣言を行なってください。
- データ長は、オペコード部に ".B" 又は ".W" を付加して指定することも可能です。オペコード部での指定と本疑似命令による既定値が異なる場合、オペコード部の指定にしたがってアセンブルします。
- この命令はアセンブラに対してデータ長を宣言するだけで、CPU 内部のプロセッサステータスレジスタのデータ長選択フラグ (m) は操作しませんので注意してください。

記述例

```
SEM              ; M フラグの設定を行う
.DATA   8        ; データ長の指示を行う
ADC     A,#DATA  ; 8 ビット加算を行う
```

.DEFINE 文字列定義

書式

シンボル△.DEFINE △文字列

内容

- シンボルに文字列を設定します。
- スペース、タブを含む文字列を記述する場合は、‘ ’(シングルクォート) 又は‘ ” ’(ダブルクォート) で囲んで記述してください。
- オペランドに構造化記述中で文字列定義された文字列は記述できません。
- プリントファイルには置換後のデータを出力します。ただし、コマンドオプションに “–LC”を指定した場合は置換前のデータも出力します。
- シンボルの置換処理はマクロ展開処理よりも先に行われます。

注意事項

- .DEFINE で定義されたシンボルは、定義したファイル内でしか使用することができません (.EXT , .DPEXT , .DTEXT 及び .PUB のオペランドに記述することはできません) 。

記述例

```
FLAG1  .DEFINE "#01H,DATA1" ; DATA1 のビットパターン 01H を FLAG1 へ
                            ; 設定する
CLB    FLAG1                ; DATA1 の最下位ビットをクリアする
```

.DP　ダイレクトページレジスタ値宣言 (既定値は 0000_{16})

書式

▲.DP △数値又は OFF

内容

- ダイレクトページレジスタ値 (0000_{16} ～ $FFFF_{16}$) を宣言します。
- プログラム中でダイレクトページレジスタ (DPR) を変更する場合には、本疑似命令で宣言を行ってください。
- オペランドに “OFF” を記述した場合、アセンブラはダイレクトアドレッシングを使用しません (アブソリュート又はアブソリュートロングから選択します) 。

注意事項

- アセンブラに対して、ダイレクトページレジスタ値を宣言するのみで、実際のダイレクトページレジスタ値は変化しません。

記述例

```
.DP     1000H ; 001000 ～ 0010FFH をダイレクトページ領域に宣言する
.DP     OFF   ; ダイレクトアドレッシングを使用しない
```

.DPEXT　外部参照指定 (ダイレクトページ)

書式

▲.DPEXT △ [ダイレクトページ名ラベル:] ラベル [, ラベル, . . , ラベル]

内容

- オペランドに指定したラベルをダイレクトページアドレッシングで外部参照することを宣言します。
- ダイレクトページ名ラベルを指定するとダイレクトページの値としてこのラベルの値を使用します。演算子 “OFFSET” は付けないでください。
- ダイレクトページ名を省略した場合、ラベルは疑似命令.DP で指定した現在の DPR 内に含まれるものとして処理します。
- 本疑似命令は、ラベルを参照する行より前に記述してください。

記述例

```
.DPEXT   DRPG1:WORK1,WORK2,WORK3
```

.DT　データバンクレジスタ値宣言 (既定値は 00_{16})

書式

▲.DT △数値又は OFF

内容

- データバンクレジスタ値 (00_{16} 〜 FF_{16}) を宣言します。
- ソースプログラム中のラベルを用いてバンク値を指定する場合、演算子 “BANK” を使用してください。
- プログラム中でデータバンクレジスタを変更する場合には、本疑似命令で宣言を行ってください。
- オペランドに “OFF” を記述した場合、アセンブラはアブソリュートアドレッシングを使用しません (ダイレクト又はアブソリュートロングから選択します) 。

注意事項

- アセンブラに対して、データバンクレジスタ値を宣言するのみで、実際のデータバンクレジスタ値は変化しません。

記述例

```
.DT     01H         ; 010000 〜 01FFFFH をデータバンク領域に宣言する
.DT     BANK LABEL  ; LABEL の上位 8 ビットをデータバンクレジスタ値の
                    ; 値とする
.DT     OFF         ; アブソリュートアドレッシングを使用しない
```

.DTEXT 外部参照指定 (データバンク)

書式

▲.DTEXT △ [バンク名ラベル:] ラベル [, ラベル, . . , ラベル]

内容

- オペランドに指定したラベルをアブソリュートアドレッシングで外部参照することを宣言します。
- バンク名を指定するとデータバンクの値としてこのラベルの値を使用します。演算子 “BANK” は付けないでください。
- バンク名を省略した場合、ラベルは疑似命令.DT で指定した現在の DT 値内に含まれるものとして処理します。
- 本疑似命令は、ラベルを参照する行より前に記述してください。

記述例

```
        .DTEXT   DTPG1:WORK1,WORK2,WORK3
```

.DWORD ダブルワードデータ設定

書式

▲ [ラベル:] ▲.DWORD △数式

内容

- 1 ダブルワードの定数データを設定します。
- 2 個以上のデータを設定する場合は、各データを ‘,’ で区切って指定してください。指定できる個数は 8 個までです。
- データは、下位バイトより設定します (PRN ファイルの OBJ 欄には上位バイトから設定します) 。
- オペランドにグローバルラベルを記述できます。

記述例

```
label:  .DWORD  0E1000H ; 00H,10H, 0EH,00H を設定する
        .DWORD  symbol  ; symbol のもつ値を下位バイトより設定する
```

.END　プログラム終了指定

書式

▲.END

内容

- ソースプログラムの終了を指定します。
- 本疑似命令以降の行はアセンブルしません。

記述例

```
        .END                    ; プログラムの終了を宣言する
```

.EQU　再設定可能な同義定義

書式

シンボル△.EQU △数式

内容

- シンボルに数値 (ダブルワードの値) を設定します。
- 数式中に使用するラベル又はシンボルは、この行より前に定義してください。
- オペランドに、リロケータブルな値を持つラベルは記述できません。
- コマンドオプション “–Q”を指定することにより、再設定時にワーニングを出力します。

注意事項

- シンボルに、結果がダブルワード以上の値になるような数式を記述した場合は、下位 4 バイトの値が有効となり、エラーを出力しません。

記述例

```
SYMBOL      .EQU      1           ; SYMBOL へ 1 を設定する
             :
SYMBOL      .EQU      3           ; SYMBOL へ 3 を設定する
             :
SYMBOL      .EQU      SYMBOL+7    ; SYMBOL へ 10 を設定する
```

.ERROR　アセンブルエラー宣言

書式

▲.ERROR △’文字列’

内容

- 条件アセンブル命令 (.IF) と組み合わせて使用します。
- 条件として有り得ないものが指定された場合に、オペランドに記述した文字列を CRT へ出力しアセンブル作業を終了します。

記述例

```
.IF      MODE
   :
.ELSE
.ERROR   'Undefined assemble mode'
.ENDIF
```

.EVEN　アドレス補正

書式

▲.EVEN

内容

- アドレスを偶数番地に補正します。
- ROM 属性をもつセクション (命令又はデータ設定命令の部分) 内で使用した場合は EA_{16} を出力し、RAM 属性をもつセクション (領域確保命令の部分) 内で使用した場合はアドレスの更新のみを行います。なお、補正するアドレスが偶数の場合は何も行いません。

注意事項

- 本疑似命令は ‘.EVEN’ を記述したアドレスがセクションの開始アドレスから相対的に奇数であるときに NOP 命令に対応するコードを生成して、次に記述する命令を偶数アドレスに配置するものです。この疑似命令によるアドレス補正は RASM77 で処理するもので、LINK77 では補正に関する処理は行いません。
  したがって、この疑似命令で偶数補正したセクションをリンクする場合、セクションの開始アドレスは必ず偶数アドレスに指定してください。

記述例

```
.BYTE   01H ; 01H のデータを設定する
.EVEN       ; ロケーションの補正を行う（EAH コードを出力する）
```

.EXT 外部参照指定

書式

▲.EXT △ラベル又はシンボル [, ラベル又はシンボル, . . , ラベル又はシンボル]

内容

- オペランドに指定したラベルを、外部参照指定します。
- ".EXT"によって参照される外部ラベルに対して、":DT" , ":DP" の指定によりダイレクト、アブソリュートのアドレッシングのコードを生成します。このとき、コマンドオプション "–Q"を指定すると " :DT " , " :DP " の指定箇所でワーニングを出力します。
- ダイレクト、アブソリュートのアドレッシングモードの範囲をこえる場合は、リンク時にエラーになります。
- 本疑似命令は、ラベル又はシンボルを参照する行より前に記述してください。

注意事項

- メモリ参照以外のオペランド (JMP 命令の飛び先やイミディエートアドレッシングモード等) に本疑似命令で外部参照したラベルが記述されている場合は、ラベルの下位値のみが有効になりエラーとはなりません。

記述例

```
.EXT    WORK1,WORK2,WORK3

.EXT    E_LABEL       ; 外部参照ラベルの宣言
        :
LDA     A,E_LABEL     ; アブソリュートロングアドレッシング
        :
LDA     A,DP:E_LABEL  ; ダイレクトアドレッシング
        :
LDA     A,DT:E_LABEL  ; アブソリュートアドレッシング
```

.IF (.ELSE) .ENDIF　条件付きアセンブル

書式

▲.IF △数式
　　<文 1>
▲.ELSE
　　<文 2>
▲.ENDIF

内容

- .IF に続く数式が真 (0 でない又は文字列データがある) なら文 1 をアセンブルし、偽 (0 又は文字列データがない) なら文 2 をアセンブルします。
- .IF に続く数式が真なら文 1 をアセンブルし、偽なら文 2 をアセンブルします。
- ネスティングは 20 レベルまで可能です。
- 文 1 、文 2 には複数行が記述できます。
- 数式中に使用するラベル又はシンボルは、この行より前に定義してください。
- オペランドに、リロケータブルな値を持つラベルは記述できません。
- オペランドに条件式判定の比較演算子が記述可能です。記述可能な条件判定式を表に示します。

表 B.1: 使用可能な条件判定式

<	より小さい
>	より大きい
==	等しい
!=	等しくない
< =	小さいか等しい
> =	大きいか等しい

記述例

(1)

```
.IF     FLAG    ; FLAG の値が真なら.ELSE までをアセンブルする
 :
 :
.ELSE           ; 偽なら.ENDIF までをアセンブルする
 :
 :
.ENDIF
```

(2)

```
ADD:    .MACRO  OP1,OP2,OP3
        .IF     "OP3"       ;引数 OP3 が存在した場合.ELSE までを
                            ; アセンブルする。
        ADC     OP1,OP2,OP3
        .ELSE
        ADC     OP1,OP2
        .ENDIF
```

.INCLUDE　ファイル読み込み

書式

▲.INCLUDE △ファイル名

内容

- 本疑似命令を記述した場所に、オペランドで指定したファイルの内容を読み込みます。
- ファイル名はフルネームで指定してください。
- ネスティングは 9 レベルまで可能です。
- ネスティングレベルをプリントファイルに出力します。

記述例

```
.INCLUDE    TEST.INC    ; TEST.INC の内容を読み込む
```

.INDEX　インデックスレジスタ長宣言 (既定値は 16)

書式

▲.INDEX △数式

内容

- CPU 内部のインデックスレジスタ長 (8 又は 16) を指示します。
- 数式の値が 8 であれば 8 ビット、16 であれば 16 ビットを示します。
- “CLP X” 、“SEP X” 命令等でインデックスレジスタ長選択フラグを変更する場合には、本疑似命令で宣言を行なってください。
- インデックスレジスタ長は、オペコード部に “.B” 又は “.W” を付加して指定することも可能です。オペコード部での指定と本疑似命令による既定値が異なる場合、オペコード部の指定にしたがってアセンブルします。
- 本疑似命令はアセンブラに対してインデックスレジスタ長を宣言するだけで、CPU 内部のプロセッサステータスレジスタのインデックスレジスタ長選択フラグ (x) は操作しませんので注意してください。

記述例

```
SEP     X       ; X フラグの設定を行う
.INDEX  8       ; インデックスレジスタ長を指定する
LDX     #DATA   ; 8 ビットロードを行う
```

.LIB　ライブラリファイル名指定

書式

▲.LIB △ファイル名 [, ファイル名,..., ファイル名]

内容

- リンク対象のライブラリファイル名を指定します。
- 指定可能なファイルは、.LIB 属性のファイルだけです。これ以外のファイルを指定した場合、リンク時にエラーになります。
- 本疑似命令は入れ子にできません。
- ファイル名にディレクトリパス、ファイル属性 (.LIB) は記述できません。

記述例

```
        .LIB     LIB1,LIB2,LIB3
```

.LINE　1 ページ当りの行数指定 (既定値は 54)

書式

▲.LINE △数式

内容

- リストの 1 ページ当たりの行数 (5 〜 255) を指定します。
- 本疑似命令は、プログラム中に 1 回だけ記述できます。

記述例

```
        .LINE   60      ; 行数を 60 に設定する
```

.LIST　リスト出力開始

書式

▲.LIST

内容

- PRN ファイルへのリスト出力を行います。
- 疑似命令.NLIST により PRN ファイルへの出力を中止後、リスト出力を再開する場合に使用します。

記述例

```
.NLIST          ; リスト出力の抑止を行う
 :              ; ".LIST"までの間を PRN ファイルへ出力しない
 :
.LIST           ; リスト出力の開始を行う
 :              ; この疑似命令以降を PRN ファイルへ出力する
 :
```

.LISTM　マクロ展開部のリスト出力開始

書式

▲.LISTM

内容

- PRN ファイルへ、マクロ展開部分を出力します。

注意事項

- 疑似命令.NLIST によりリスト出力全体が抑止されている場合は.LISTM 命令は無効となります。
- コマンドラインパラメータで “–M” を指定しない場合は.LISTM 命令は無効となります。

記述例

```
.NLISTM         ; マクロ展開部のリスト出力の抑止を行います。
   :            ;".LISTM"までのマクロ展開部分を出力しません。
   :            ;
.LISTM          ; マクロ展開部のリスト出力の開始を行います。
   :            ; この疑似命令以降のマクロ展開部分は出力します。
```

.MCU　　命令生成 MCU タイプ設定

書式

▲.MCU △ MCU タイプ

内容

- 命令生成を行なう MCU タイプを設定します。
- オペランドに記述できるタイプは以下のものです (アルファベットは大文字/小文字のどちらでもかまいません) 。

1. M37700

   7700 ファミリ 命令の生成を行ないます。

2. M37750

   M37750 シリーズ命令の生成を行ないます。

3. M37751

   M37751 シリーズ命令の生成を行ないます。

- 本疑似命令にて設定された MCU タイプは、再度 '.MCU' で設定される範囲まで有効となります。
- 本疑似命令を省略した場合には 7700 ファミリ の命令生成を行います (M37750 、M37751 命令記述の際にはエラーとなります) 。

記述例

```
.MCU      M37750     ; M37750 の命令生成を指定します。
```

.NLIST　リスト出力抑止

書式

▲.NLIST

内容

- PRN ファイルへの出力を抑止します。
- この状態は疑似命令.LIST により解除できます。

記述例

```
.NLIST          ; リスト出力の抑止を行う
 :              ; ".LIST"までの間を PRN ファイルへ出力しない
 :
.LIST           ; リスト出力の開始を行う
 :              ; この疑似命令以降を PRN ファイルへ出力する
 :
```

.NLISTM　マクロ展開部のリスト出力抑止

書式

▲.NLISTM

内容

- PRN ファイルへマクロ展開部を出力しません。
- この状態は疑似命令.LISTM により解除できます。

注意事項

- コマンドラインパラメータの “–M” を指定しない場合は本疑似命令は無効となります。

記述例

```
.NLISTM         ; マクロ展開部のリスト出力の抑止を行います。
   :            ;".LISTM"までのマクロ展開部分を出力しません。
   :            ;
.LISTM          ; マクロ展開部のリスト出力の開始を行います。
   :            ; この疑似命令以降のマクロ展開部分は出力します。
```

.OBJ　リロケータブルファイル名指定

書式

▲.OBJ △ファイル名 [, ファイル名,..., ファイル名]

内容

- リンク対象のリロケータブルファイル名を指定します。
- 指定可能なファイルは、.R77 属性のファイルだけです。これ以外のファイルを指定した場合、リンク時にエラーになります。
- 本疑似命令は入れ子にできません。
- ファイル名にディレクトリパス、ファイル属性 (.R77) は記述できません。

記述例

```
        .OBJ    OBJ1,OBJ2,OBJ3
```

.ORG　アドレス宣言 (既定値は 000000H)

書式

▲.ORG △数式

内容

- この行以降の開始アドレスを宣言します。
- 指定がない場合、開始アドレスは 000000_{16} として処理します。
- 本疑似命令を記述したセクションは、絶対属性になります。本疑似命令を含まないセクションは、相対属性になります。
- 数式中に使用するラベル又はシンボルは、この行より前に定義してください。
- オペランドに、リロケータブルな値を持つラベルは記述できません。

記述例

```
        .ORG    0C000H  ; ロケーションを 0C000H に設定する
```

.PAGE　リスト改ページ及びタイトル指定

書式

▲.PAGE △ [’タイトル’]

内容

- この命令の直前でリストの改ページを行い、オペランドで指定したタイトルをリストのヘッダ部に出力します。タイトルは、‘’’(シングルクォート) 又は‘”’(ダブルクォート) で囲んで記述してください。
- タイトルの文字数は、カラム指定が 80 のとき最大 20 文字、105 から 132 のとき最大 45 文字まで、81 から 104 まではカラム数から 60 を引いた文字数まで、各々使用できます。なおタイトル省略時には改ページのみを行います。

記述例

```
.PAGE   ’PROG1’ ; PROG1 を PRN ファイルのヘッダ部に出力する
```

.PUB　パブリック指定

書式

▲.PUB △ラベル又はシンボル [, ラベル又はシンボル, . . , ラベル又はシンボル]

内容

- オペランドに指定したシンボル又はラベルを、他のソースファイルから参照可能にします。
- 本疑似命令は、ラベル又はシンボルを定義する行より前に記述してください。

記述例

```
        .PUB    WORK1,WORK2,WORK3
WORK1:  .BLKW   1
WORK2:  .BLKW   1
WORK3:  .BLKW   1
```

.SECTION　セクション名指定

書式

▲.SECTION △セクション名

内容

- RASM77 では、ソースプログラムをセクションという単位に分けて処理します。本疑似命令は次の行以降のプログラムに対し、オペランドに記述したセクション名を指定します。
- セクション名には任意の名前を付けることが可能です。同じ名前をもつセクションがファイル内に複数個存在してもかまいません。
- 本疑似命令は、プログラム記述を始める前に必ず記述してください (ない場合はエラーになります) 。

記述例

```
        .SECTION    DATA    ; DATA セクションを開始する。
datatop:
nulldt: .BLKB    8
         :
        .SECTION     STACK  ; STACK セクションを開始する。
        .BLKB        1000H
stacktop:
         :
        .SECTION     PROG   ; PROG セクションを開始する。
_init:
        .DATA        16
        .INDEX       16
        CLP          #0FFH
        LDA          A,#stacktop
        TAS
        LDA          A,#SIZEOF DATA
        LDX          #OFFSET constop
        LDY          #OFFSET datatop
        MVN          BANK constop,BANK datatop
         :
        .SECTION      CONT  ; CONT セクションを開始する。
constop:
        .BYTE         0,0,0,0,0,0,0,0
         :
```

.VER　プログラムバージョン指定

書式

▲.VER △’文字列’

内容

- リロケータブルファイルのバージョン指定を行います。
- LINK77 は “–V” パラメータが指定されると、各リロケータブルファイル中のバージョン指定の一致を確認します。これによりリロケータブルファイル間のバージョン整合性の確認が行えます。“–V” パラメータの詳細については、LINK77 操作マニュアルを参照してください。
- バージョンの一致は、文字列の比較により確認します。大文字/小文字も区別しますので、注意して記述してください。
- 本疑似命令は、プログラム中に 1 回だけ記述できます。

記述例

```
.VER    'V.1.0' ; バージョン"V.1.0"を宣言します
```

.WORD　ワードデータ設定

書式

▲ [ラベル:] ▲.WORD △数式

内容

- 数式のもつ値を設定します (単位:2 バイト)
- 2 つ以上のデータを設定する場合は、各データを ‘,’ で区切って指定してください。指定できる個数は 16 個までです。
- データは下位バイトより設定します (PRN ファイルの OBJ 欄には上位バイトから設定します) 。
- オペランドにグローバルラベルを記述できます。

記述例

```
label:  .WORD   0E000H          ; 00H,E0H を設定する
        .WORD   OFFSET symbol   ; symbol のもつ値を下位バイトより設定する
```

B.3 デバッグ用疑似命令一覧

以下の疑似命令は、ソースラインデバッグのための疑似命令です。これらの疑似命令は 7700 ファミリ用 C 言語、及び構造化命令で記述されたプログラムのソースラインデバッグ情報などをデバッガに渡します。これらの疑似命令は、プリプロセッサが生成します。

.CLINE 行頭情報出力

書式

▲.CLINE △数値

内容

- ソースデバッグ時に必要な行番号を設定します。
- 本疑似命令は、C コンパイラでコンパイル又は、PRE77 で処理されたアセンブリファイルに出力されます。

注意事項

- オペランドに指定できる数値は 1 から 9999 までの値です。
  10000 以上の数値を指定した場合はワーニングを出力し行頭情報はオブジェクトファイルに出力しません。
- オペランドに記述された数値は、再定義のチェックは行っていません。
- マクロ命令中に記述された本疑似命令は処理しません。

記述例

```
        .CLINE  10
        JSR     _SUB
        .CLINE  11
           :
```

.ENDFUNC ファンクション終了指定

書式

▲.ENDFUNC △ラベル

内容

- ファンクション (サブルーチン) の終了を指定します。
- この命令により、ソースラインデバッグが可能となります。
- 本疑似命令は、ネスティングできません。

記述例

```
        .FUNC     SUB
SUB:    LDA       A,#0
           :
        RTS
        .ENDFUNC  SUB      ; ファンクションの終了を指定します。
```

.FUNC　ファンクション開始指定

書式

▲.FUNC △ラベル

内容

- ファンクション (サブルーチン) の開始を指定します。
- この命令により、ソースラインデバッグが可能となります。
- 本疑似命令は、ネスティングできません。

記述例

```
        .FUNC     SUB       ; ファンクションの開始を指定します。
SUB:    LDA       A,#0
          :
        RTS
        .ENDFUNC  SUB
```

.LANGUAGE　使用言語名出力

書式

▲.LANGUAGE △言語名

内容

- ソースデバッグ時に必要な言語情報を設定します。

注意事項

- コマンドラインパラメータ “–C” を指定しない場合、言語情報はオブジェクトファイルへは出力しません。
- 本疑似命令はソースファイルに１回のみ使用可能です。

記述例

```
        .LANGUAGE    C
```

.POINTER ポインタ長設定

書式

▲.POINTER △数値

内容

- C コンパイラで使用されたポインタ変数のバイト長を設定します。
- オペランドに指定できる数値は 1 から 255 までの値です。

注意事項

- 本疑似命令は、ソースファイル中に 1 回のみ使用可能です。

記述例

```
        .POINTER    2
```

.SOURCE ソースファイル名設定

書式

▲.SOURCE △ソースファイル名

内容

- ソースデバッグ時に必要なファイル名を設定します。

注意事項

- ファイル名、パス名の大文字、小文字は区別されます。
- 本疑似命令は.CLINE 命令よりも前に記述してください。

記述例

```
        .SOURCE  PROG1.C
        .CLINE   1
           :
        .SOURCE  INCLUDE/STDIO.H
        .CLINE   1
```

B.4 予約疑似命令一覧

以下の疑似命令は、将来の拡張のために予約しています。これらの疑似命令を記述してもアセンブルには影響を与えません。

注意事項

1. “.IO ～.ENDIO”，“.PROCINT ～.ENDPROC”，“.PROCMAIN ～.ENDPROC”，“.PROCSUB ～.ENDPROC”，“.RAM ～.ENDRAM” の組合せの片方しか記述していない場合や、組み合わせの記述順が正しくない場合はアセンブルエラーを発生しますので注意してください。

.ENDIO　I/O 領域終了宣言 (予約)

書式

▲.ENDIO

内容

- I/O 領域の終了宣言を行います。
- .IO より.ENDIO までに記述したラベル及びシンボルを I/O 領域として解釈します。

記述例

```
        .IO
port0   .EQU    00H
port1   .EQU    01H
        .ENDIO
```

.ENDPROC　プログラムモジュール終了宣言 (予約)

書式

▲.ENDPROC

内容

- メインプログラム、サブプログラム、割り込み処理プログラムの各モジュールの終了を宣言します。
- .PROCINT 、.PROCMAIN 、.PROCSUB より.ENDPROC で囲んだ範囲を一つのプログラムモジュールとして解釈します。

記述例

```
        .PROCMAIN
MAIN:
         :
        JMP      MAIN
        .ENDPROC
```

.ENDRAM　RAM 領域終了宣言 (予約)

書式

▲.ENDRAM

内容

- RAM 領域の終了宣言を行います。
- .RAM より.ENDRAM までに記述したラベル及びシンボルを RAM 領域として解釈します。

記述例

```
        .RAM
work0:  .BLKB    1
work1:  .BLKB    1
        .ENDRAM
```

.IO　I/O 領域開始宣言 (予約)

書式

▲.IO

内容

- I/O 領域の開始宣言を行います。
- .IO より.ENDIO までに記述したラベル及びシンボルを I/O 領域として解釈します。

記述例

```
        .IO
port0   .EQU    00H
port1   .EQU    01H
        .ENDIO
```

.PROCINT 割り込み処理プログラム開始宣言 (予約)

書式

▲.PROCINT △ [ラベル]

内容

- 割り込み処理プログラムの開始宣言を行います。
- .PROCINT より.ENDPROC で囲んだ範囲を割り込み処理プログラムとして解釈します。
- RASM77 は、オペランドに記述したラベルをこの行のラベルとして処理します。

記述例

```
.PROCINT INT
 :
 :
.ENDPROC
```

.PROCMAIN メインプログラム開始宣言 (予約)

書式

▲.PROCMAIN △ [ラベル]

内容

- メインプログラムの開始宣言を行います。
- .PROCMAIN より.ENDPROC で囲んだ範囲をメインプログラムとして解釈します。
- RASM77 は、オペランドに記述したラベルをこの行のラベルとして処理します。

記述例

```
.PROCMAIN MAIN
 :
.ENDPROC
```

.PROCSUB　サブプログラム開始宣言 (予約)

書式

▲.PROCSUB △ [ラベル]

内容

- サブプログラムの開始宣言を行います。
- .PROCSUB より.ENDPROC で囲んだ範囲をサブプログラムとして解釈します。
- RASM77 は、オペランドに記述したラベルをこの行のラベルとして処理します。

記述例

```
.PROCSUB SUB
 :
 :
.ENDPROC
```

.PROGNAME　プログラム名宣言 (予約)

書式

▲.PROGNAME △プログラム名

内容

- プログラム名の宣言を行います。
- オペランドの内容をプログラムタイトルとして解釈します。

記述例

```
.PROGNAME ’プリンタ制御プログラム’
```

.RAM　　RAM 領域開始宣言 (予約)

書式

▲.RAM

内容

- RAM 領域の開始宣言を行います。
- .RAM より.ENDRAM までに記述したラベル及びシンボルを RAM 領域として解釈します。

記述例

```
        .RAM
work0:  .BLKB    1
work1:  .BLKB    1
        .ENDRAM
```

付録 C

マクロ命令一覧

C.1　マクロ命令一覧の見方

RASM77 で使用可能なマクロ命令をアルファベット順に掲載します。表記上の規則を以下に示します。

1. 説明中の [ ] 内の部分は省略可能部分を示しています。
2. △、▲はスペース又はタブコードを示しています。△の部分は必須、▲の部分は省略可能を示します。
3. 以下の説明ではラベルとマクロ命令間は▲で示しています。ラベルを記述する場合 ':'(コロン) は必須ではありませんが、':' を省略した場合にはマクロ命令との間にスペース、又はタブコードが必要になります。

C.2　マクロ命令一覧

.ENDM　　ENDM マクロ命令

書式

▲.ENDM

内容

- この命令は、すべてのマクロ定義部の終了を指定します。

記述例

```
[マクロ定義]
    ADD:    .MACRO   VAL
            CLC
            ADC      A,VAL
            .ENDM
[呼び出し例]
            ADD      #10
[マクロ展開]
            CLC
            ADC      A,#10
            .ENDM
```

.EXITM　EXITM マクロ命令

書式

▲.EXITM

内容

- この命令は、マクロ展開を中止し一番近い.ENDM に制御を渡します。

記述例

```
[マクロ定義]
    DATA1:  .MACRO   VAL
            .IF      LABEL
            .BYTE    VAL
            .EXITM
            .ENDIF
            .WORD    VAL
            .ENDM
[呼び出し例]
    LABEL   .EQU     1
            DATA1    10
[マクロ展開]
            .IF      LABEL
            .BYTE    10
            .EXITM
            .ENDIF
            .ENDM
```

.LOCAL　LOCAL マクロ命令

書式

▲.LOCAL △ラベル,[ラベル,..., ラベル]

内容

- この命令は、マクロ定義のなかで定義されているラベルをマクロ内ローカルラベルとします。
- .LOCAL 宣言されたラベルを、その順に..n(n は 10 進数で 0 ～ 65535) というラベルを割り当ててアセンブルします。したがってユーザは、.. で始まるラベルは使用できませんので注意してください。
- .LOCAL 宣言は、ラベルを使用する前に行う必要があります。

記述例

```
[マクロ定義]
    LOOP:   .MACRO
            .LOCAL   LOOP1
            LDA      A,#20
    LOOP1:  DEC      A
            BNE      LOOP1
            .ENDM
[呼び出し例]
            LOOP
[マクロ展開]
            LDA      A,#20
    ..0:    DEC      A
            BNE      ..0
            .ENDM
```

.MACRO ～.ENDM　　MACRO マクロ命令

書式

▲マクロ名:▲.MACRO △ [引数 1, 引数 2, . . , 引数 n]

内容

- .MACRO と記述された行からマクロ定義が始まり、.ENDM が記述された行でそのマクロ定義が終了します。
- .MACRO 命令の行に付けた文字列がこのマクロ定義に付けられた名前となります。
- マクロ定義には引数を持つマクロと持たないマクロがあり、引数を持つマクロを使用する場合には、マクロの呼び出し文で必要な引数を渡す必要があります。
- マクロ呼び出しが行われると引数は展開される際、マクロ定義で記述された仮の引数の順序にしたがって引数を渡します。
- .MACRO ～.ENDM の間には、アセンブリ言語命令、ユーザーマクロ命令、システムマクロ命令、及び .INCLUDE 以外の疑似命令が記述できます。ただし、マクロ定義のネスティングは 20 レベルまでです。
- マクロの呼び出しはマクロ定義の後であれば、プログラムの任意の場所で行えます。マクロ定義がマクロ呼び出しの後にある場合はエラーとなります。マクロ呼び出しのネスティングは 20 レベルまでです。
- マクロ呼び出しでオペランドに指定した引数は、マクロ定義の仮の引数と左から順に対応する引数と置き換えられます。なお、引数の数と定義した時の仮の引数の数は一致していなくてもかまいませんが、この場合ワーニングが出力されます。また、引数の数が仮の引数より多いときは余分な引数は無視され、少ない場合は、不足する仮の引数は空文字 (長さ 0 の文字列) とみなされます。
- マクロ定義に用いた仮の引数に関係なく、引数は任意の個数指定できますが、そのすべてが一行以内に収まらなければなりません。引数と引数の間は‘,’で区切るため、カンマやスペースを引数として渡す場合は‘”’でくくらなければなりません。ただし、‘()’内のカンマは引数の区切りとして扱いません。
- マクロ展開された部分は、リストファイル中に‘+’で示されます。
- マクロ定義は同一のマクロ名で複数回行う事ができます。このとき、マクロ呼び出しを行なった場合、その呼び出し直前に定義されたものが呼び出し対象となります。

記述例

例 1）オペランドを持たないマクロ定義

[マクロ定義]

```
ADD1:   .MACRO
        LDA     A,ABC
        LDX     #DEF
        CLC
        ADC     A,TABLE,X
        STA     A,GHI
        .ENDM
```

[呼び出し例]

```
        ADD1
```

[マクロ展開]

```
        LDA     A,ABC
        LDX     #DEF
        CLC
        ADC     A,TABLE,X
        STA     A,GHI
        .ENDM
```

例 2）オペランドを持つマクロ定義

[マクロ定義]

```
ADD2:   .MACRO   V1,IMM,V2
                   ↑
                 仮の引数
        LDA     A,V1
        LDX     #IMM
        CLC
        ADC     A,TABLE,X
        STA     A,V2
        .ENDM
```

[呼び出し例]

```
        ADD2    WORK1,10,WORK2
```

[マクロ展開]

```
        LDA     A,WORK1
        LDX     #10
        CLC
        ADC     A,TABLE,X
        STA     A,WORK2
        .ENDM
```

記述例

例 3）マクロのネスティング

[マクロ定義]

```
ADD:    .MACRO  OP1,OP2
        CLC
        ADC     OP1,OP2
        .ENDM
```

[呼び出し例]

```
ADD2:   .MACRO  OP1,OP2
        ADD     OP1,OP2     ; マクロ内でのマクロ呼び出し
        ADC     OP1+1,OP2+2
        .ENDM
```

例 4）マクロの再帰定義

[マクロ定義]

```
MAC:    .MACRO              ; 一度目の定義
DATA:   .BLKB   1
MAC:    .MACRO  VALUE       ; 二度目の定義
        LDM     #VALUE,DATA
        .ENDM
        .ENDM
```

[呼び出し例]

```
        MAC                 ; 領域の確保と新たなマクロ定義
         :
        MAC     10H         ; 新たに定義されたマクロの呼び出し
```

.REPEAT ～.ENDM　　REPEAT マクロ命令

書式

▲ [ラベル:] ▲.REPEAT △回数

内容

- .REPEAT から.ENDM までにある 7700 ファミリ の命令をオペランドで指定した回数だけ繰り返しアセンブルします。
- 繰り返し可能な回数は 254 以下です。
- .REPEAT 命令の行に付けたラベルは生成された行の最初のラベルとなります。
- .REPEAT ～.ENDM の間には、アセンブリ言語命令、ユーザーマクロ命令、システムマクロ命令、及び ".INCLUDE"以外の疑似命令が記述できます。
- オペランドには、数値定数、記号定数 (ラベル) が記述できますが、リロケータブルな値を持つラベルは記述できません。
- ネスティングは 20 レベルまで可能です。

記述例

[ソース記述例]

```
TIME5:  .REPEAT  5
        NOP
        .ENDM
```

[マクロ展開後]

```
TIME5:  .REPEAT  5
        NOP
        .ENDM
TIME5:
        NOP
        NOP
        NOP
        NOP
        NOP
        .ENDM
```

.REPEATC～.ENDM　　REPEATC マクロ命令

書式

▲ [ラベル:] ▲.REPEATC △仮引数, 実引数

内容

- オペランドで実引数で与えた文字数回.ENDM までを繰り返しアセンブルします。
- 繰り返すたびに実引数から 1 文字ずつ取り出しこれを仮引数に渡します。
- .REPEATC 命令の行に付けたラベルは生成された行の最初のラベルとなります。
- .REPEATC ～.ENDM の間には、アセンブリ言語命令、ユーザーマクロ命令、システムマクロ命令、及び “.INCLUDE”以外の疑似命令が記述できます。
- 文字列にスペース、タブ、‘ , ’ (カンマ) 等の特殊文字を含む場合は、全体を‘ ” ’ でくくって文字列を指定します。この時文字列は‘ ” ’ を取り除いた残りが用いられます。
- ネスティングは 20 レベルまで可能です。

記述例

例 1）

[ソース記述例]

```
DATA:   .REPEATC VAL,ABCDE
                 ↑    ↑
              仮引数 実引数
        .BYTE   'VAL'
        .ENDM
```

[マクロ展開後]

```
DATA:   .REPEATC VAL,ABCDE
        .BYTE   'VAL'
        .ENDM
DATA:
        .BYTE   'A'
        .BYTE   'B'
        .BYTE   'C'
        .BYTE   'D'
        .BYTE   'E'
        .ENDM
```

例 2）

[ソース記述例]

```
DATA:   .REPEATC VAL,"ABC,;"
                 ↑    ↑
              仮引数 実引数
        .BYTE   'VAL'
        .ENDM
```

[マクロ展開後]

```
DATA:   .REPEATC VAL,"ABC,;"
        .BYTE   'VAL'
        .ENDM
DATA:
        .BYTE   'A'
        .BYTE   'B'
        .BYTE   'C'
        .BYTE   ','
        .BYTE   ';'
        .ENDM
```

.REPEATI～.ENDM　　REPEATI マクロ命令

書式

▲ [ラベル:] ▲.REPEATI △仮引数, 実引数 [, 実引数, . . , 実引数]

内容

- オペランドに記述された実引数の個数回.ENDM までを繰り返しアセンブルします。
- 繰り返すたびにオペランドに記述された実引数から 1 つずつ取り出しこれを仮引数に渡します。
- .REPEATI 命令の行に付けたラベルは生成された行の最初のラベルとなります。
- .REPEATI ～.ENDM の間には、アセンブリ言語命令、ユーザーマクロ命令、システムマクロ命令、及び ".INCLUDE"以外の疑似命令が記述できます。
- 実引数には数値定数、文字定数、記号定数 (ラベル) 、文字列定数が記述できます。他のマクロ命令は記述できません。
- 実引数の中にスペース、タブ、','(カンマ) が含まれる場合は全体を ' " ' でくくって文字列を指定します。この時、文字列は '" ' を取り除いた残りが用いられます。
- ネスティングは 20 レベルまで可能です。

記述例

例 1）

[ソース記述例]

```
SUB:    .REPEATI INST,"NOP","LDA A,#1","JSR SUB1","RTS"
                  ↑              ↑
              仮引数          実引数
        INST
        .ENDM
```

[マクロ展開後]

```
SUB:    .REPEATI INST,"NOP","LDA A,#1","JSR SUB1","RTS"
        INST
        .ENDM
SUB:
        NOP
        LDA A,#1
        JSR SUB1
        RTS
        .ENDM
```

例 2）

[ソース記述例]

```
DATA:   .REPEATI VAL,0,1,2,"'HELLO !!'"
                  ↑        ↑
              仮引数   実引数
        .BYTE   VAL
        .ENDM
```

[マクロ展開後]

```
DATA:   .REPEATI VAL,0,1,2,"'HELLO !!'"
        .BYTE   VAL
        .ENDM
DATA:
        .BYTE   0
        .BYTE   1
        .BYTE   2
        .BYTE   'HELLO !!'
        .ENDM
```

付録 D

命令一覧表

D.1　記号表

命令一覧表で使用している記号の意味を、表D.1に示します。

表 D.1: 命令一覧用記号表

Acc	アキュムレータ A もしくはアキュムレータ B
X	インデックスレジスタ X
Y	インデックスレジスタ Y
S	スタックポインタ S
d8	8 ビットのデータ
d16	16 ビットのデータ
[DP:]zz	ダイレクトアドレッシングモードでのダイレクトページレジスタからの 8 ビット相対アドレス値
[DT:]hhll	アブソリュートアドレッシングモードでの下位 16 ビットのアドレス値
[LG:]hhmmll	アブソリュートロングアドレッシングモードでの 24 ビットのアドレス値
imm	8 ビット及び 16 ビットの即値データ
rr	−128 ～ +127 の範囲の相対アドレス値
rrll	−65535 ～ +65534 の範囲の相対アドレス値
DPR	ダイレクトページレジスタ
PBR	プログラムバンクレジスタ
PG	プログラムバンクレジスタ
DBR	データバンクレジスタ
DT	データバンクレジスタ
PS	プロセッサステータスレジスタ
PSR	プロセッサステータスレジスタ
C	キャリーフラグ
Z	ゼロフラグ
I	割り込み禁止フラグ
D	10 進モードフラグ
X	インデックスレジスタ長選択フラグ
M	データ長選択フラグ
V	オーバーフローフラグ
N	ネガティブフラグ

注意事項

1. [DP:] 、[DT:] 、[LG:] について
   - ダイレクトアドレッシング、アブソリュートアドレッシング又はアブソリュートロングアドレッシングを明示的に指定したい場合に使用してください。
   - zz 、又は hhll の部分にシンボルを記述した場合のみ、その値が現在のダイレクトページ又はデータバンクの先頭からのオフセット値そのものになります (“–D” パラメータで定義したシンボルを含みます) 。
   - zz 、又は hhll の部分にラベルを記述した場合、ラベルのもつ値と現在指定されているレジスタ値との差がオフセット値になります。
2. イミディエイトデータ (imm) の記述について
   - 通常、データ長は疑似命令.DATA 、.INDEX により宣言した既定値に従いますが、命令コードの後に “.B” 、“.W” を付加して記述することにより命令単位で指定することができます。

     例 1)　　`ADC.B  A,#data`
     $\Longrightarrow$ 8 ビットデータとしてアセンブルします。
     例 2)　　`ADC.W  A,#data`
     $\Longrightarrow$ 16 ビットデータとしてアセンブルします。
   - CPU 内部のデータ長選択フラグ (m) 、インデックスレジスタ長選択フラグ (x) の状態は RASM77 では操作しませんので、ユーザプログラムで設定してください。
3. CLP 、SEP 命令のオペランドに記述した ‘X’ は、インデックスレジスタ長選択フラグとして解釈します。

D.2　命令一覧表

RASM77 で使用可能な全命令を表 D.2に示します。各命令の横に、記述可能なデータ長指定、アドレッシングモード名、記述形式を示します。

表 D.2: 命令一覧表

命令名	データ長	アドレッシングモード名	記述形式	
ADC	.B/.W	イミディエイト	ADC.B	Acc,#imm
ADCL		ダイレクト	ADC	Acc,[DP:]zz
		ダイレクト X	ADC	Acc,[DP:]zz,X
		ダイレクト・インダイレクト	ADC	Acc,([DP:]zz)
		ダイレクト・インダイレクト X	ADC	Acc,([DP:]zz,X)
		ダイレクト・インダイレクト Y	ADC	Acc,([DP:]zz),Y
		ダイレクト・インダイレクト・ロング	ADCL	Acc,([DP:]zz)
		ダイレクト・インダイレクト・ロング Y	ADCL	Acc,([DP:]zz),Y
		アブソリュート	ADC	Acc,[DT:]hhll
		アブソリュート X	ADC	Acc,[DT:]hhll,X
		アブソリュート Y	ADC	Acc,[DT:]hhll,Y
		アブソリュート・ロング	ADC	Acc,[LG:]hhmmll
		アブソリュート・ロング X	ADC	Acc,[LG:]hhmmll,X
		スタックポインタ・リラティブ	ADC	Acc,d8,S
		スタックポインタ・リラティブ・インダイレクト Y	ADC	Acc,(d8,S),Y
AND	.B/.W	イミディエイト	AND.B	Acc,#imm
ANDL		ダイレクト	AND	Acc,[DP:]zz
		ダイレクト X	AND	Acc,[DP:]zz,X
		ダイレクト・インダイレクト	AND	Acc,([DP:]zz)
		ダイレクト・インダイレクト X	AND	Acc,([DP:]zz,X)
		ダイレクト・インダイレクト Y	AND	Acc,([DP:]zz),Y
		ダイレクト・インダイレクト・ロング	ANDL	Acc,([DP:]zz)
		ダイレクト・インダイレクト・ロング Y	ANDL	Acc,([DP:]zz),Y
		アブソリュート	AND	Acc,[DT:]hhll
		アブソリュート X	AND	Acc,[DT:]hhll,X
		アブソリュート Y	AND	Acc,[DT:]hhll,Y
		アブソリュート・ロング	AND	Acc,[LG:]hhmmll
		アブソリュート・ロング X	AND	Acc,[LG:]hhmmll,X
		スタックポインタ・リラティブ	AND	Acc,d8,S
		スタックポインタ・リラティブ・インダイレクト Y	AND	Acc,(d8,S),Y

命令名	データ長	アドレッシングモード名	記述形式	
ASL		アキュムレータ	ASL	Acc
		ダイレクト	ASL	[DP:]zz
		ダイレクト X	ASL	[DP:]zz,X
		アブソリュート	ASL	[DT:]hhll
		アブソリュート X	ASL	[DT:]hhll,X
ASR		アキュムレータ	ASR	Acc
		ダイレクト	ASR	[DP:]zz
		ダイレクト X	ASR	[DP:]zz,X
		アブソリュート	ASR	[DT:]hhll
		アブソリュート X	ASR	[DT:]hhll,X
BBC	.B/.W	ダイレクト・ビット・リラティブ	BBC.B	#imm,[DP:]zz,rr
	.B/.W	アブソリュート・ビット・リラティブ	BBC.B	#imm,[DT:]hhll,rr
BBS	.B/.W	ダイレクト・ビット・リラティブ	BBS.B	#imm,[DP:]zz,rr
	.B/.W	アブソリュート・ビット・リラティブ	BBS.B	#imm,[DT:]hhll,rr
BCC		リラティブ	BCC	rr
BCS		リラティブ	BCS	rr
BEQ		リラティブ	BEQ	rr
BMI		リラティブ	BMI	rr
BNE		リラティブ	BNE	rr
BPL		リラティブ	BPL	rr
BRA		リラティブ	BRA	rr
BRAL		リラティブ	BRAL	rrll
BRK		インプライド	BRK	#d8
BVC		リラティブ	BCC	rr
BVS		リラティブ	BCS	rr
CLB	.B/.W	ダイレクト・ビット	CLB.B	#imm,[DP:]zz
	.B/.W	アブソリュート・ビット	CLB.B	#imm,[DT:]hhll
CLC		インプライド	CLC	
CLI		インプライド	CLI	
CLM		インプライド	CLM	

命令名	データ長	アドレッシングモード名	記述形式
CLP[1]		イミディエイト	CLP #d8
CLV		インプライド	CLV
CMP	.B/.W	イミディエイト	CMP.B Acc,#imm
CMPL		ダイレクト	CMP Acc,[DP:]zz
		ダイレクト X	CMP Acc,[DP:]zz,X
		ダイレクト・インダイレクト	CMP Acc,([DP:]zz)
		ダイレクト・インダイレクト X	CMP Acc,([DP:]zz,X)
		ダイレクト・インダイレクト Y	CMP Acc,([DP:]zz),Y
		ダイレクト・インダイレクト・ロング	CMPL Acc,([DP:]zz)
		ダイレクト・インダイレクト・ロング Y	CMPL Acc,([DP:]zz),Y
		アブソリュート	CMP Acc,[DT:]hhll
		アブソリュート X	CMP Acc,[DT:]hhll,X
		アブソリュート Y	CMP Acc,[DT:]hhll,Y
		アブソリュート・ロング	CMP Acc,[LG:]hhmmll
		アブソリュート・ロング X	CMP Acc,[LG:]hhmmll,X
		スタックポインタ・リラティブ	CMP Acc,d8,S
		スタックポインタ・リラティブ・インダイレクト Y	CMP Acc,(d8,S),Y
CPX	.B/.W	イミディエイト	CPX.B #imm
		ダイレクト	CPX [DP:]zz
		アブソリュート	CPX [DT:]hhll
CPY	.B/.W	イミディエイト	CPY.B #imm
		ダイレクト	CPY [DP:]zz
		アブソリュート	CPY [DT:]hhll
DEC		アキュムレータ	DEC Acc
		ダイレクト	DEC [DP:]zz
		ダイレクト X	DEC [DP:]zz,X
		アブソリュート	DEC [DT:]hhll
		アブソリュート X	DEC [DT:]hhll,X
DEX		インプライド	DEX
DEY		インプライド	DEY

注意事項

1. CLP 命令はオペランド欄にフラグ名を記述できます。

   例)　　`CLP　　C,Z,I,D,X,M,V,N`

命令名	データ長	アドレッシングモード名	記述形式	
DIV	.B/.W	イミディエイト	DIV.B	#imm
DIVL		ダイレクト	DIV	[DP:]zz
		ダイレクト X	DIV	[DP:]zz,X
		ダイレクト・インダイレクト	DIV	([DP:]zz)
		ダイレクト・インダイレクト X	DIV	([DP:]zz,X)
		ダイレクト・インダイレクト Y	DIV	([DP:]zz),Y
		ダイレクト・インダイレクト・ロング	DIVL	([DP:]zz)
		ダイレクト・インダイレクト・ロング Y	DIVL	([DP:]zz),Y
		アブソリュート	DIV	[DT:]hhll
		アブソリュート X	DIV	[DT:]hhll,X
		アブソリュート Y	DIV	[DT:]hhll,Y
		アブソリュート・ロング	DIV	[LG:]hhmmll
		アブソリュート・ロング X	DIV	[LG:]hhmmll,X
		スタックポインタ・リラティブ	DIV	d8,S
		スタックポインタ・リラティブ・インダイレクト Y	DIV	(d8,S),Y
DIVS	.B/.W	イミディエイト	DIVS.B	#imm
DIVSL		ダイレクト	DIVS	[DP:]zz
		ダイレクト X	DIVS	[DP:]zz,X
		ダイレクト・インダイレクト	DIVS	([DP:]zz)
		ダイレクト・インダイレクト X	DIVS	([DP:]zz,X)
		ダイレクト・インダイレクト Y	DIVS	([DP:]zz),Y
		ダイレクト・インダイレクト・ロング	DIVSL	([DP:]zz)
		ダイレクト・インダイレクト・ロング Y	DIVSL	([DP:]zz),Y
		アブソリュート	DIVS	[DT:]hhll
		アブソリュート X	DIVS	[DT:]hhll,X
		アブソリュート Y	DIVS	[DT:]hhll,Y
		アブソリュート・ロング	DIVS	[LG:]hhmmll
		アブソリュート・ロング X	DIVS	[LG:]hhmmll,X
		スタックポインタ・リラティブ	DIVS	d8,S
		スタックポインタ・リラティブ・インダイレクト Y	DIVS	(d8,S),Y

命令名	データ長	アドレッシングモード名	記述形式	
EOR	.B/.W	イミディエイト	EOR.B	Acc,#imm
EORL		ダイレクト	EOR	Acc,[DP:]zz
		ダイレクト X	EOR	Acc,[DP:]zz,X
		ダイレクト・インダイレクト	EOR	Acc,([DP:]zz)
		ダイレクト・インダイレクト X	EOR	Acc,([DP:]zz,X)
		ダイレクト・インダイレクト Y	EOR	Acc,([DP:]zz),Y
		ダイレクト・インダイレクト・ロング	EORL	Acc,([DP:]zz)
		ダイレクト・インダイレクト・ロング Y	EORL	Acc,([DP:]zz),Y
		アブソリュート	EOR	Acc,[DT:]hhll
		アブソリュート X	EOR	Acc,[DT:]hhll,X
		アブソリュート Y	EOR	Acc,[DT:]hhll,Y
		アブソリュート・ロング	EOR	Acc,[LG:]hhmmll
		アブソリュート・ロング X	EOR	Acc,[LG:]hhmmll,X
		スタックポインタ・リラティブ	EOR	Acc,d8,S
		スタックポインタ・リラティブ・インダイレクト Y	EOR	Acc,(d8,S),Y
EXTS		アキュムレータ	EXTS	Acc
EXTZ		アキュムレータ	EXTZ	Acc
INC		アキュムレータ	INC	Acc
		ダイレクト	INC	[DP:]zz
		ダイレクト X	INC	[DP:]zz,X
		アブソリュート	INC	[DT:]hhll
		アブソリュート X	INC	[DT:]hhll,X
INX		インプライド	INX	
INY		インプライド	INY	

命令名	データ長	アドレッシングモード名	記述形式	
JMP JMPL		アブソリュート	JMP	hhll
		アブソリュート・ロング	JMPL	[LG:]hhmmll
		アブソリュート・インダイレクト	JMP	(hhll)
		アブソリュート・インダイレクト・ロング	JMPL	(hhll)
		アブソリュート・インダイレクト・インデクスト X	JMP	(hhll,X)
JSR JSRL		アブソリュート	JSR	hhll
		アブソリュート・ロング	JSRL	[LG:]hhmmll
		アブソリュート・インダイレクト X	JSR	(hhll,X)
LDA LDAL	.B/.W	イミディエイト	LDA.B	Acc,#imm
		ダイレクト	LDA	Acc,[DP:]zz
		ダイレクト X	LDA	Acc,[DP:]zz,X
		ダイレクト・インダイレクト	LDA	Acc,([DP:]zz)
		ダイレクト・インダイレクト X	LDA	Acc,([DP:]zz,X)
		ダイレクト・インダイレクト Y	LDA	Acc,([DP:]zz),Y
		ダイレクト・インダイレクト・ロング	LDAL	Acc,([DP:]zz)
		ダイレクト・インダイレクト・ロング Y	LDAL	Acc,([DP:]zz),Y
		アブソリュート	LDA	Acc,[DT:]hhll
		アブソリュート X	LDA	Acc,[DT:]hhll,X
		アブソリュート Y	LDA	Acc,[DT:]hhll,Y
		アブソリュート・ロング	LDA	Acc,[LG:]hhmmll
		アブソリュート・ロング X	LDA	Acc,[LG:]hhmmll,X
		スタックポインタ・リラティブ	LDA	Acc,d8,S
		スタックポインタ・リラティブ・インダイレクト Y	LDA	Acc,(d8,S),Y

命令名	データ長	アドレッシングモード名	記述形式	
LDT		イミディエイト	LDT	#d8
LDM	.B/.W	ダイレクト	LDM.B	#imm,[DP:]zz
	.B/.W	ダイレクト X	LDM.B	#imm,[DP:]zz,X
	.B/.W	アブソリュート	LDM.B	#imm,[DT:]hhll
	.B/.W	アブソリュート X	LDM.B	#imm,[DT:]hhll,X
LDX	.B/.W	イミディエイト	LDX.B	#imm
		ダイレクト	LDX	[DP:]zz
		ダイレクト Y	LDX	[DP:]zz,Y
		アブソリュート	LDX	[DT:]hhll
		アブソリュート Y	LDX	[DT:]hhll,Y
LDY	.B/.W	イミディエイト	LDY.B	#imm
		ダイレクト	LDY	[DP:]zz
		ダイレクト X	LDY	[DP:]zz,X
		アブソリュート	LDY	[DT:]hhll
		アブソリュート X	LDY	[DT:]hhll,X
LSR		アキュムレータ	LSR	Acc
		ダイレクト	LSR	[DP:]zz
		ダイレクト X	LSR	[DP:]zz,X
		アブソリュート	LSR	[DT:]hhll
		アブソリュート X	LSR	[DT:]hhll,X

命令名	データ長	アドレッシングモード名	記述形式	
MPY	.B/.W	イミディエイト	MPY.B	#imm
MPYL		ダイレクト	MPY	[DP:]zz
		ダイレクト X	MPY	[DP:]zz,X
		ダイレクト・インダイレクト	MPY	([DP:]zz)
		ダイレクト・インダイレクト X	MPY	([DP:]zz,X)
		ダイレクト・インダイレクト Y	MPY	([DP:]zz),Y
		ダイレクト・インダイレクト・ロング	MPYL	([DP:]zz)
		ダイレクト・インダイレクト・ロング Y	MPYL	([DP:]zz),Y
		アブソリュート	MPY	[DT:]hhll
		アブソリュート X	MPY	[DT:]hhll,X
		アブソリュート Y	MPY	[DT:]hhll,Y
		アブソリュート・ロング	MPY	[LG:]hhmmll
		アブソリュート・ロング X	MPY	[LG:]hhmmll,X
		スタックポインタ・リラティブ	MPY	d8,S
		スタックポインタ・リラティブ・インダイレクト Y	MPY	(d8,S),Y
MPYS	.B/.W	イミディエイト	MPYS.B	#imm
MPYSL		ダイレクト	MPYS	[DP:]zz
		ダイレクト X	MPYS	[DP:]zz,X
		ダイレクト・インダイレクト	MPYS	([DP:]zz)
		ダイレクト・インダイレクト X	MPYS	([DP:]zz,X)
		ダイレクト・インダイレクト Y	MPYS	([DP:]zz),Y
		ダイレクト・インダイレクト・ロング	MPYSL	([DP:]zz)
		ダイレクト・インダイレクト・ロング Y	MPYSL	([DP:]zz),Y
		アブソリュート	MPYS	[DT:]hhll
		アブソリュート X	MPYS	[DT:]hhll,X
		アブソリュート Y	MPYS	[DT:]hhll,Y
		アブソリュート・ロング	MPYS	[LG:]hhmmll
		アブソリュート・ロング X	MPYS	[LG:]hhmmll,X
		スタックポインタ・リラティブ	MPYS	d8,S
		スタックポインタ・リラティブ・インダイレクト Y	MPYS	(d8,S),Y

命令名	データ長	アドレッシングモード名	記述形式
MVN		ブロック転送	MVN d8,d8
MVP		ブロック転送	MVP d8,d8
NOP		インプライド	NOP
ORA	.B/.W	イミディエイト	ORA.B Acc,#imm
ORAL		ダイレクト	ORA Acc,[DP:]zz
		ダイレクト X	ORA Acc,[DP:]zz,X
		ダイレクト・インダイレクト	ORA Acc,([DP:]zz)
		ダイレクト・インダイレクト X	ORA Acc,([DP:]zz,X)
		ダイレクト・インダイレクト Y	ORA Acc,([DP:]zz),Y
		ダイレクト・インダイレクト・ロング	ORAL Acc,([DP:]zz)
		ダイレクト・インダイレクト・ロング Y	ORAL Acc,([DP:]zz),Y
		アブソリュート	ORA Acc,[DT:]hhll
		アブソリュート X	ORA Acc,[DT:]hhll,X
		アブソリュート Y	ORA Acc,[DT:]hhll,Y
		アブソリュート・ロング	ORA Acc,[LG:]hhmmll
		アブソリュート・ロング X	ORA Acc,[LG:]hhmmll,X
		スタックポインタ・リラティブ	ORA Acc,d8,S
		スタックポインタ・リラティブ・インダイレクト Y	ORA Acc,(d8,S),Y
PEA		スタック	PEA #d16
PEI		スタック	PEI #d8
PER		スタック	PER #d16
PHA		スタック	PHA
PHB		スタック	PHB
PHD		スタック	PHD
PHG		スタック	PHG
PHP		スタック	PHP
PHT		スタック	PHT
PHX		スタック	PHX
PHY		スタック	PHY

命令名	データ長	アドレッシングモード名	記述形式	
PLA		スタック	PLA	
PLB		スタック	PLB	
PLD		スタック	PLD	
PLP		スタック	PLP	
PLT		スタック	PLT	
PLX		スタック	PLX	
PLY		スタック	PLY	
PSH[1]		スタック	PSH	#d8
PUL[2]		スタック	PUL	#d8
RLA	.B/.W	イミディエイト	RLA	#imm
RMPA		イミディエイト	RMPA	#imm
ROL		アキュムレータ	ROL	Acc
		ダイレクト	ROL	[DP:]zz
		ダイレクト X	ROL	[DP:]zz,X
		アブソリュート	ROL	[DT:]hhll
		アブソリュート X	ROL	[DT:]hhll,X
ROR		アキュムレータ	ROR	Acc
		ダイレクト	ROR	[DP:]zz
		ダイレクト X	ROR	[DP:]zz,X
		アブソリュート	ROR	[DT:]hhll
		アブソリュート X	ROR	[DT:]hhll,X
RTI		インプライド	RTI	
RTL		インプライド	RTL	
RTS		インプライド	RTS	

注意事項

1.,2. PSH 、PUL 命令はオペランド欄にレジスタ名が使用できます。

例 1)　　`PSH　　A,B,X,Y,DPR,PG,DT,PS`
例 2)　　`PUL　　A,B,X,Y,DPR,PG,DT,PS`

命令名	データ長	アドレッシングモード名	記述形式	
SBC	.B/.W	イミディエイト	SBC.B	Acc,#imm
SBCL		ダイレクト	SBC	Acc,[DP:]zz
		ダイレクト X	SBC	Acc,[DP:]zz,X
		ダイレクト・インダイレクト	SBC	Acc,([DP:]zz)
		ダイレクト・インダイレクト X	SBC	Acc,([DP:]zz,X)
		ダイレクト・インダイレクト Y	SBC	Acc,([DP:]zz),Y
		ダイレクト・インダイレクト・ロング	SBCL	Acc,([DP:]zz)
		ダイレクト・インダイレクト・ロング Y	SBCL	Acc,([DP:]zz),Y
		アブソリュート	SBC	Acc,[DT:]hhll
		アブソリュート X	SBC	Acc,[DT:]hhll,X
		アブソリュート Y	SBC	Acc,[DT:]hhll,Y
		アブソリュート・ロング	SBC	Acc,[LG:]hhmmll
		アブソリュート・ロング X	SBC	Acc,[LG:]hhmmll,X
		スタックポインタ・リラティブ	SBC	Acc,d8,S
		スタックポインタ・リラティブ・インダイレクト Y	SBC	Acc,(d8,S),Y
SEB	.B/.W	ダイレクト・ビット	SEB.B	#imm,[DP:]zz
	.B/.W	アブソリュート・ビット	SEB.B	#imm,[DT:]hhll
SEC		インプライド	SEC	
SEI		インプライド	SEI	
SEM		インプライド	SEM	
SEP[1]		イミディエイト	SEP	#d8

注意事項

1. SEP 命令はオペランド欄にレジスタ名が使用できます。

   例)　　`SEP　　C,Z,I,D,X,M,V,N`

命令名	データ長	アドレッシングモード名	記述形式	
STA STAL		ダイレクト	STA	Acc,[DP:]zz
		ダイレクト X	STA	Acc,[DP:]zz,X
		ダイレクト・インダイレクト	STA	Acc,([DP:]zz)
		ダイレクト・インダイレクト X	STA	Acc,([DP:]zz,X)
		ダイレクト・インダイレクト Y	STA	Acc,([DP:]zz),Y
		ダイレクト・インダイレクト・ロング	STAL	Acc,([DP:]zz)
		ダイレクト・インダイレクト・ロング Y	STAL	Acc,([DP:]zz),Y
		アブソリュート	STA	Acc,[DT:]hhll
		アブソリュート X	STA	Acc,[DT:]hhll,X
		アブソリュート Y	STA	Acc,[DT:]hhll,Y
		アブソリュート・ロング	STA	Acc,hhmmll
		アブソリュート・ロング X	STA	Acc,hhmmll,X
		スタックポインタ・リラティブ	STA	Acc,d8,S
		スタックポインタ・リラティブ・インダイレクト Y	STA	Acc,(d8,S),Y
STP		インプライド	STP	
STX		ダイレクト	STX	[DP:]zz
		ダイレクト Y	STX	[DP:]zz,Y
		アブソリュート	STX	[DT:]hhll
STY		ダイレクト	STY	[DP:]zz
		ダイレクト X	STY	[DP:]zz,X
		アブソリュート	STY	[DT:]hhll
TAD		インプライド	TAD	
TAS		インプライド	TAS	
TAX		インプライド	TAX	
TAY		インプライド	TAY	
TBD		インプライド	TBD	
TBS		インプライド	TBS	
TBX		インプライド	TBX	
TBY		インプライド	TBY	

命令名	データ長	アドレッシングモード名	記述形式
TDA		インプライド	TDA
TDB		インプライド	TDB
TSA		インプライド	TSA
TSB		インプライド	TSB
TSX		インプライド	TSX
TXA		インプライド	TXA
TXB		インプライド	TXB
TXS		インプライド	TXS
TXY		インプライド	TXY
TYA		インプライド	TYA
TYB		インプライド	TYB
TYX		インプライド	TYX
WIT		インプライド	WIT
XAB		インプライド	XAB

付録 E

アドレッシングモード別命令一覧表

E.1　アドレッシングモード別命令一覧表

アドレッシングモードごとの命令記述形式と該当命令を示します。説明で使用している記号は、付録 B.1 の記号表に従います。

1. インプライド

該当命令	記述形式
BRK, CLC, CLI, CLM, CLV DEX, DEY, INX, INY, NOP SEC, SEI, SEM, STP, TAX RTI, RTL, RTS, TAY, TAD TAS, TBB, TBD, TBS, TBX TBY, TDA, TDB, TSA, TSB TSX, TXA, TXB, TXS, TXY TYA, TYB, TYX, WIT, XAB	CLC

2. イミディエイト

該当命令	記述形式	
ADC, AND, CMP, EOR, LDA ORA, SBC	ADC	Acc,#imm
DIV, DIVS, MPY, MPYS, RMPA	DIV	#imm
CPX, CPY, LDX, LDY	CPX	#imm
CLP, SEP	CLP CLP	#d8 C,I,M
LDT	LDT	#d8

3. アキュムレータ

該当命令	記述形式
ASL, ASR, DEC, EXTS EXTZ, INC, LSR, ROL, ROR	ASL Acc
RLA	RLA #d8

4. ダイレクト・ビット

該当命令	記述形式
CLB, SEB	CLB #imm,zz CLB #imm,dp:zz

5. アブソリュート・ビット

該当命令	記述形式
CLB, SEB	CLB #imm,hhll CLB #imm,dt:hhll

6. スタック

該当命令	記述形式
PHA, PHB, PHD, PHG, PHP PHT, PHX, PHY, PLA, PLB PLD, PLP, PLT, PLX, PLY	PHA
PEI	PEI #d8
PSH, PUL	PSH #d8 PSH A,X,Y
PEA, PER	PEA #d16

7. リラティブ

該当命令	記述形式
BCC, BCS, BEQ, BMI, BNE BPL, BVC, BVS	BCC rr
BRA	BRA rr
BRAL	BRAL rrll

8. ブロック転送

該当命令	記述形式
MVN, MVP	MVN d8,d8

9. ダイレクト

該当命令	記述形式	
ADC, AND, CMP, EOR, LDA ORA, SBC, STA	ADC ADC	Acc,zz Acc,dp:zz
DIV, DIVS, MPY, MPYS	DIV DIV	zz dp:zz
ASL, CPX, CPY, DEC, INC LDX, LDY, LSR, ROL, ROR STX, STY	ASL ASL	zz dp:zz
LDM	LDM LDM	#imm,zz #imm,dp:zz

10. ダイレクト・インデックス X

該当命令	記述形式	
ADC, AND, CMP, EOR, LDA ORA, SBC, STA	ADC ADC	Acc,zz,X Acc,dp:zz,x
DIV, DIVS, MPY, MPYS	DIV DIV	zz,X dp:zz,X
ASL, DEC, INC, LDY, LSR ROL, ROR, STY	ASL ASL	zz,X dp:zz,X
LDM	LDM LDM	#imm,zz,X #imm,dp:zz,X

11. ダイレクト・インデックス Y

該当命令	記述形式	
LDX, STX	LDX LDX	zz,Y dp:zz,Y

12. ダイレクト・インダイレクト

該当命令	記述形式	
ADC, AND, CMP, EOR, LDA ORA, SBC, STA	ADC ADC	Acc,(zz) Acc,(dp:zz)
DIV, DIVS, MPY, MPYS	DIV DIV	(zz) (dp:zz)

13. ダイレクト・インダイレクト・インデックス X

該当命令	記述形式	
ADC, AND, CMP, EOR, LDA	ADC	Acc,(zz,X)
ORA, SBC, STA	ADC	Acc,(dp:zz,X)
DIV, DIVS, MPY, MPYS	DIV	(zz,X)
	DIV	(dp:zz,X)

14. ダイレクト・インダイレクト・インデックス Y

該当命令	記述形式	
ADC, AND, CMP, EOR, LDA	ADC	Acc,(zz),Y
ORA, SBC, STA	ADC	Acc,(dp:zz),Y
DIV, DIVS, MPY, MPYS	DIV	(zz),Y
	DIV	(dp:zz),Y

15. ダイレクト・インダイレクト・ロング

該当命令	記述形式	
ADCL, ANDL, CMPL, EORL	ADCL	Acc,(zz)
LDAL, ORAL, SBCL, STAL	ADCL	Acc,(dp:zz)
DIVL, DIVSL, MPYL, MPYSL	DIVL	(zz)
	DIVL	(dp:zz)

16. ダイレクト・インダイレクト・ロング・インデックス Y

該当命令	記述形式	
ADCL, ANDL, CMPL, EORL	ADCL	Acc,(zz),Y
LDAL, ORAL, SBCL, STAL	ADCL	Acc,(dp:zz),Y
DIVL, DIVSL, MPYL, MPYSL	DIVL	(zz),Y
	DIVL	(dp:zz),Y

17. アブソリュート

該当命令	記述形式	
ADC, AND, CMP, EOR, LDA ORA, SBC, STA	ADC ADC	Acc,hhll Acc,dt:hhll
DIV, DIVS, MPY, MPYS	DIV DIV	hhll dt:hhll
ASL, ASR, CPX, CPY, DEC INC, LDX, LDY, LSR, ROL ROR, STX, STY	ASL ASL	hhll dt:hhll
JMP, JSR	JSR	hhll
LDM	LDM LDM	#imm,hhll #imm,dt:hhll

18. アブソリュート・インデックス X

該当命令	記述形式	
ADC, AND, CMP, EOR, LDA ORA, SBC, STA	ADC ADC	Acc,hhll,X Acc,dt:hhll,X
DIV, DIVS, MPY, MPYS	DIV DIV	hhll,X dt:hhll,X
ASL, ASR, DEC, INC, LDY LSR, ROL, ROR	ASL ASL	hhll,X dt:hhll,X
LDM	LDM LDM	#imm,hhll,X #imm,dt:hhll,X

19. アブソリュート・インデックス Y

該当命令	記述形式	
ADC, AND, CMP, EOR, LDA ORA, SBC, STA	ADC ADC	Acc,hhll,Y Acc,dt:hhll,Y
DIV, DIVS, MPY, MPYS	DIV DIV	hhll,Y dt:hhll,Y
LDX	LDX LDX	hhll,Y dt:hhll,Y

20. アブソリュートロング

該当命令	記述形式	
ADC, AND, CMP, EOR, LDA ORA, SBC, STA	ADC ADC	Acc,hhmmll Acc,lg:hhmmll
DIV, DIVS, MPY, MPYS	DIV DIV	hhmmll lg:hhmmll
JMPL, JSRL	JMPL JMPL	hhmmll lg:hhmmll

21. アブソリュートロング・インデックス X

該当命令	記述形式	
ADC, AND, CMP, EOR, LDA ORA, SBC, STA	ADC ADC	Acc,hhmmll,X Acc,lg:hhmmll,x
DIV, DIVS, MPY, MPYS	DIV DIV	hhmmll,X lg:hhmmll,x

22. スタックポインタ・リラティブ

該当命令	記述形式	
ADC, AND, CMP, EOR, LDA ORA, SBC, STA	ADC	Acc,d8,S
DIV, DIVS, MPY, MPYS	DIV	d8,S

23. スタックポインタ・リラティブ・インダイレクト・インデックス Y

該当命令	記述形式	
ADC, AND, CMP, EOR, LDA ORA, SBC, STA	ADC	Acc,(d8,S),Y
DIV, DIVS, MPY, MPYS	DIV	(d8,S),Y

24. アブソリュート・インダイレクト

該当命令	記述形式	
JMP	JMP	(hhll)

25. アブソリュート・インダイレクト・インデックス X

該当命令	記述形式
JMP, JSR	JMP (hhll,X)

26. アブソリュート・インダイレクト・ロング

該当命令	記述形式
JMPL	JMPL (hhll)

27. ダイレクト・ビット・リラティブ

該当命令	記述形式
BBC, BBS	BBC #imm,zz,rr BBC #imm,dp:zz,rr

28. アブソリュート・ビット・リラティブ

該当命令	記述形式
BBC, BBS	BBC #imm,hhll,rr BBC #imm,dt:hhll,rr

E.2 アドレッシングモード対応表

命令語のオペランドに書かれたデータに対して、アセンブラがどのアドレッシングモードを選択するかを一覧表にして示します。また、記述できないデータの組合せについても、アセンブラの出力するエラーを表示してあります。プログラム記述の際に参考にして下さい。

以下に、表中に書かれている記号の意味について説明します。

○	——	アドレスの範囲チェックを行う
◎	——	無条件に指定されたアドレッシングモードを選択する
◇	——	ページのチェックを行う
●	——	命令のもつ最大のアドレッシングモードを選択する
PE	——	エラー 27:Page error を出力する
VO	——	エラー 22:Value is out of range を出力する
＊1	——	表の横軸は、疑似命令、“.DT” , “.DP” で指定するアドレッシングモードを示す
＊2	——	表の縦軸は、命令語のオペランドで指定するアドレッシングモードを示す
ページなし	——	ダイレクトページ名ラベルを使用しないことを示す
ページあり	——	ダイレクトページ名ラベルを使用することを示す
バンクなし	——	バンク名ラベルを使用しないことを示す
バンクあり	——	バンク名ラベルを使用することを示す

表 E.1: アドレッシングモード対応表

＊2 ＼ ＊1			OFF	IMM	SYM	LOC		EXT				
						REL	ABS	DPEXT ページなし	DPEXT ページあり	DTEXT バンクなし	DTEXT バンクあり	EXT
DP: 指定あり	IMM		○	○	○	○	○	○	○	○	○	○
	SYM		○	○	○	○	○	○	○	○	○	○
	LOC	REL	VO	◎	◎	◎	◎	◎	◎	◎	◎	◎
		ABS	VO	○	○	◎	○	◎	◎	◎	◎	◎
	EXT	ページなし DPEXT	VO	◎	◎	◎	◎	◎	◎	◎	◎	◎
		ページあり DPEXT	VO	PE	PE	PE	PE	PE	◇	PE	PE	PE
		バンクなし DTEXT	VO	VO	VO	VO	VO	VO	VO	VO	VO	VO
		バンクあり DTEXT	VO	VO	VO	VO	VO	VO	VO	VO	VO	VO
		EXT	VO	VO	VO	VO	VO	VO	VO	VO	VO	VO

DT: 指定 あり	IMM		○	○	○	○	○	○	○	○	○	○
	SYM		○	○	○	○	○	○	○	○	○	○
	LOC	REL	VO	◎	◎	◎	◎	◎	◎	◎	◎	◎
		ABS	VO	○	○	◎	○	◎	◎	◎	◎	◎
	EXT	ページなし DPEXT	VO	VO	VO	VO	VO	VO	VO	VO	VO	VO
		ページあり DPEXT	VO	VO	VO	VO	VO	VO	VO	VO	VO	VO
		バンクなし DTEXT	VO	◎	◎	◎	◎	◎	◎	◎	◎	◎
		バンクあり DTEXT	VO	PE	PE	PE	PE	PE	PE	PE	◇	PE
		EXT	VO	VO	VO	VO	VO	VO	VO	VO	VO	VO
DT: DP: 指定 なし	IMM		●	○	○	●	○	●	●	●	●	●
	SYM		●	○	○	●	○	●	●	●	●	●
	LOC	REL	●	●	●	●	●	●	●	●	●	●
		ABS	●	○	○	●	○	●	●	●	●	●
	EXT	ページなし DPEXT	●	◎	◎	◎	◎	◎	◎	◎	◎	◎
		ページあり DPEXT	●	PE	PE	PE	PE	PE	◇	PE	PE	PE
		バンクなし DTEXT	●	◎	◎	◎	◎	◎	◎	◎	◎	◎
		バンクあり DTEXT	●	PE	PE	PE	PE	PE	PE	PE	◇	PE
		EXT	●	●	●	●	●	●	●	●	●	●

E.3　アドレッシングモードの選択

RASM77 において “.DP OFF”および “.DT OFF”使用時のアドレッシングモード選択には以下の規則があります。

記述形式	RASM77 の処理
DP:ラベル	エラー
DP:シンボル	ダイレクトページアドレッシングモード選択
DP:絶対値	ダイレクトページアドレッシングモード選択
DT:ラベル	エラー
DT:シンボル	アブソリュートページアドレッシングモード選択
DT:絶対値	アブソリュートページアドレッシングモード選択

第 2 部

7700 ファミリ 用
構造化記述命令処理プリプロセッサ

PRE77 操作マニュアル

目 次

図 目 次

表 目 次

第 1 章

PRE77 操作マニュアルの構成

PRE77 操作マニュアルは、以下の章から構成されています。

- 第 2 章 概要
  PRE77 の基本的な機能と生成ファイルについて説明します。
- 第 3 章 ソースプログラムの記述方法
  PRE77 の構造化記述言語を含むソースプログラムの記述方法について説明します。
- 第 4 章 構造化記述言語
  PRE77 に使用できる構造化記述言語の種類と記述方法について説明します。
- 第 5 章 疑似命令
  PRE77 で処理を行う疑似命令について説明します。
- 第 6 章 操作方法
  PRE77 のコマンド入力方法について説明します。
- 付録 A エラーメッセージ一覧表
  PRE77 が表示するエラーメッセージについて、その内容と対策を一覧表で示します。
- 付録 B 構造化命令一覧
  PRE77 で使用可能な全構造化記述言語を一覧表で示します。
- 付録 C 疑似命令一覧
  PRE77 で処理を行う疑似命令を一覧表で示します。

第 2 章

概要

構造化記述を用いたプログラムはアセンブリ言語によるプログラムに比べ、if 文や for 文等の構造化ステートメントを用いたプログラム記述が行えます。また、7700 ファミリ用デバッガコントロールソフトウエアでのソースレベルデバッグ機能により、プログラム開発がアセンブリ言語のみで開発する場合に比べ開発効率のよい環境が得られます。

2.1 機能

PRE77 は 7700 ファミリ用構造化記述言語プログラムを RASM77 用アセンブリ言語プログラムに変換します。 PRE77 は RASM77, LINK77[1], LIB77[2]及びデバッガコントロールソフトウェアと共に利用できます。また、これらの機能を十分に生かすため以下の機能を備えています。

1. RASM77 用疑似命令 “.INCLUDE” , “.DATA” , “.INDEX” , “.EQU” を処理します。
2. “.INCLUDE” の対象ファイルを処理します。
3. RASM77 でアセンブル可能なアセンブリ言語ソースプログラムファイルを生成します。
4. デバッガコントロールソフトウェアでソースラインデバッグが可能な “.CLINE” , “.FUNC” , “.ENDFUNC” を出力します。
5. マクロ定義内の構造化記述言語を処理します。ただし、マクロ展開は RASM77 でのみ処理し、PRE77 では処理しません。

この他に、以下のような特徴があります。

1. エラー内容を格納したタグファイル[3]を生成できます (プリプロセスエラーの修正を効率的に行うことができます) 。

[1] 7700 ファミリ用リンケージエディタのプログラム名。
[2] 7700 ファミリ用ライブラリアンのプログラム名。
[3] タグの名称は、エラーやワーニングの場所を示す荷札 (タグ) に由来しています。

2.2 生成ファイル

PRE77 では以下の 2 種類のファイルを生成します。

1. RASM77 用のアセンブリ言語ソースプログラムファイル (以下アセンブラファイルと呼びます)
   - RASM77 のソースレベルデバッグ用疑似命令 “.CLINE”，“.FUNC”，“.ENDFUNC” を含んでいます。
   - RASM77 及び LINK77 で処理することにより、インテル HEX 形式の機械語データを生成します。
   - ファイル属性は、.A77 です。
   - 図 2.1、図 2.2 に PRE77 のアセンブリファイルの出力例を示します。

```
; *** 7700 Family PREPROCESSOR V.5.00.00 ***
    .language    PRE77_Rev01
;    sample list
;
;FLAG_0   .EQU   0,000H
FLAG_0    .define 000H
;FLAG_1   .EQU   1,000H
FLAG_1    .define 000H
;FLAG_2   .EQU   2,000H
FLAG_2    .define  000H
;
         .EXT   WORK1
         .EXT   WORK2
;
         .SECTION   PROGRAM
;
;for [ DP:FLAG_0 ] == 1
..F1:
    BBC    #00001H,DP:FLAG_0,..F2
;    if [ DP:FLAG_1 ] == 1
    BBC    #00002H,DP:FLAG_1,..I3
;        [ DP:WORK1 ] = 0
    LDM    #0,DP:WORK1
;       if [ DP:FLAG_2 ] == 1
    BBC    #00004H,DP:FLAG_2,..I5
;           [ DP:WORK2 ] = 0
    LDM    #0,DP:WORK2
;       endif
..I5:
;    endif
..I3:
    BRA    ..F1
;next
..F2:
         .END
```

図 2.1: アセンブリファイル例

```
; *** 7700 Family PREPROCESSOR V.5.00.00 ***
    .language    PRE77_Rev01
    .source      sample.p77
;    sample list
;
;FLAG_0   .EQU   0,000H
FLAG_0    .define 000H
;FLAG_1   .EQU   1,000H
FLAG_1    .define 000H
;FLAG_2   .EQU   2,000H
FLAG_2    .define 000H
;
         .EXT   WORK1
         .EXT   WORK2
;
         .SECTION   PROGRAM
         .func      _sample_0
;
;for [ DP:FLAG_0 ] == 1
    .cline    12
..F1:
    BBC    #00001H,DP:FLAG_0,..F2
;    if [ DP:FLAG_1 ] == 1
    .cline    13
    BBC    #00002H,DP:FLAG_1,..I3
;        [ DP:WORK1 ] = 0
    .cline    14
    LDM    #0,DP:WORK1
;       if [ DP:FLAG_2 ] == 1
    .cline    15
    BBC    #00004H,DP:FLAG_2,..I5
;           [ DP:WORK2 ] = 0
    .cline    16
    LDM    #0,DP:WORK2
;       endif
    .cline    17
..I5:
;    endif
    .cline    18
..I3:
    BRA    ..F1
;next
    .cline    19
..F2:
         .endfunc    _sample_0
         .END
```

図 2.2: アセンブリファイル例 (ソースレベルデバッグ情報出力)

2. タグファイル

- PRE77 処理中に発生したプリプロセスエラーメッセージ及びワーニングメッセージを格納したファイルです。
- ファイル属性は、.PTG です。
- タグジャンプ機能を持つエディタを利用することにより効率よくエラーの修正が行えます。
- タグファイルは、エディタによるエラー修正時に参照用としてお使いください。
- タグファイルは、コマンドパラメータ “–E”を指定した時に出力します。
- 図 2.3にタグファイルの出力例を示します。

```
SAMPLE.P77 120 ( TOTAL LINE 120 ) Error  9: else not associated with if
SAMPLE.P77 135 ( TOTAL LINE 135 ) Error 13: break not inside for, do or switch
SAMPLE.P77 140 ( TOTAL LINE 140 ) Error  6: Not in conditional block
```

図 2.3: タグファイル例

第 3 章

ソースプログラムの記述方法

3.1　ソースプログラムの構成

PRE77 ソースプログラムは、行を単位として構成しています。行記述の規則を以下に示します。

1. 各行は 1 行ごとに完結しており、1 命令を 2 行以上に渡って記述することはできません。
2. 1 行の文字数は、改行コードを含めて最大 256 文字です。PRE77 は、256 文字を越えた部分を無視します。
3. 1 行は以下の欄から構成しています。
   - シンボル/ラベル欄
     行を他の場所から参照するためのラベル、又は疑似命令.EQU により値を設定するシンボルを記述します。
   - オペコード/疑似命令欄
     7700 ファミリ命令ニーモニック (以下オペコードと呼びます) 又は疑似命令を記述します。
   - オペランド欄
     オペコード又は疑似命令の処理対象を記述します。
   - コメント欄
     これ以降の桁は PRE77 で処理しませんのでユーザが自由に使用できます。

行は内容別に以下の 5 種類に分類できます。

1. 命令行
   7700 ファミリの命令を記述する行です。PRE77 では処理を行いません。
2. 構造化記述言語命令行 7700 ファミリの構造化記述言語を記述する行です。
   この行は対応するアセンブラ命令行を生成します。
3. 疑似命令行
   プリプロセス時に必要な指示を行います。
   PRE77 の処理対象となる疑似命令のみ処理を行います。

4. マクロ命令行
   マクロ定義を記述する行です。アセンブラで処理します。

5. コメント行
   この行は PRE77 で処理しませんのでユーザが自由に使用できます。

3.2 行の構成

本節では、行の構成を示します。以下に表記上の規則を示します。

1. △、▲はスペース又はタブコードを示しています。
   △の部分は必須、▲の部分は省略可能です。

2. ラベルを記述する場合 ':'(コロン) は必須です。

3.2.1 命令行

命令行の構成を以下に示します。

▲ [ラベル:] ▲ [オペコード] △ [オペランド] ▲ [; コメント] <RET>

▲† [オペコード] △ [オペランド] ▲ [; コメント] <RET>

注意事項

† PRE77 は命令を予約語として識別しますので、ラベルがない場合は行の先頭からオペコードの記述が可能です。

3.2.2 構造化記述言語命令行

構造化記述言語命令 (以下構造化命令と呼びます) 行の構成を以下に示します。

▲ [ラベル:] ▲ [構造化命令] △ [条件式] ▲ [; コメント] <RET>

▲† [構造化命令] △ [条件式] ▲ [; コメント] <RET>

1. ラベル欄
   行を他の場所から参照するためのラベルを記述してください。
   PRE77 では構造化命令を処理し、7700 ファミリ用アセンブリ言語生成時に、この欄の内容をそのまま出力します。

2. 構造化命令欄
   7700 ファミリの構造化命令を記述します。構造化命令は、英大文字/小文字を区別しません。したがって IF 、if いずれも有効です。

3. 条件式欄
   7700 ファミリの構造化命令の処理対象を記述してください。

4. コメント欄
   この欄は、PRE77 で処理しませんのでユーザが自由に使用できます。

注意事項

† PRE77 は構造化命令を予約語として識別しますので、ラベルを記述しない場合は行の先頭から記述することができます。

3.2.3 疑似命令行

疑似命令行の構成を以下に示します。

▲ [ラベル:] ▲ [疑似命令] △ [オペランド] ▲ [; コメント] <RET>

▲ [シンボル] △ [.EQU] △ [オペランド] ▲ [; コメント] <RET>

▲‡ [疑似命令] △ [オペランド] ▲ [; コメント] <RET>

注意事項

‡ PRE77 は RASM77 の疑似命令を予約語として識別しますので、ラベルがない場合は行の先頭から疑似命令の記述が可能です。

3.2.4 マクロ命令行

マクロ命令行の構成を以下に示します。
ただし、この行は PRE77 では処理は行いません。従ってマクロ引数を用いた構造化記述は正しく処理することができません (この行の詳細については、RASM77 ユーザズマニュアル第 1 部 第 6 章、及び第 1 部 付録 C を参照してください) 。

▲ [マクロ名:] ▲ [マクロ命令] △ [オペランド] ▲ [; コメント] <RET>

3.2.5 コメント行

コメント行は、行の最初の文字を ';'(セミコロン) で始めてください。コメント行の構成を以下に示します。

▲ [; コメント] <RET>

3.3 欄の記述方法

3.3.1 シンボル/ラベル欄

PRE77 は、シンボルとラベルを区別して管理しますが、名前の記述形式は同一です。以下に記述形式を示します。

1. 名前には、英数字、特殊文字 '_' (アンダーライン) 及び '?'(クエッションマーク) が使用できます。ただし、 1 文字目に使用できる文字は、英字と特殊文字だけです。

2. 名前に、予約語は使用できません。PRE77 では、レジスタ名、フラグ名、疑似命令、オペランド記述用演算子 (DP,DT,LG を含む) 及び構造化命令を予約語として処理します。

3. 大文字/小文字を区別します。したがって BIG と Big は異なった名前として判断します。

4. ラベルの文字数は ':' を含めて最大 255 文字です。

5. 特殊文字 '..'(ピリオド 2 個) で始まる以下のラベルは、PRE77 で作成されるラベルですので、使用できません。また、以下のラベル以外についても '..' で始まるラベルは PRE77, RASM77 の今後の拡張のために使用する場合がありますので使用できませんので注意してください。

   - `..D0 ～ ..D65535`
   - `..F0 ～ ..F65535`
   - `..I0 ～ ..I65535`
   - `..S0 ～ ..S65535`

6. ラベル欄を記述する場合、名前の直後に ':'(コロン) をつける必要があります。ただし、シンボル名の直後に ':'(コロン) をつけるとワーニングを発生します。

3.3.2 オペコード/疑似命令欄

オペコード/疑似命令欄には、7700 ファミリの命令ニーモニック又は疑似命令を記述します。以下に記述形式を示します。

1. 命令ニーモニック及び疑似命令は、大文字/小文字を区別しません。したがって NOP 、nop いずれも有効です。

3.3.3 オペランド欄

オペランド欄には、命令又は疑似命令の実行対象の情報を記述します。以下に記述形式を示します。

1. オペランドにデータが 2 つ以上ある場合、データ間を ','(カンマ) で区切ってください。

2. カンマの両側には、スペース又はタブコードが記述可能です。

3.3.4 コメント欄

ユーザーの任意の情報を記述することができます。以下に記述形式を示します。

1. コメントの先頭に ';'(セミコロン) を付けて記述してください。

2. コメント欄にはどのような文字も記述できます。

3.4 オペランド欄の記述方法

3.4.1 データ形式

オペランド欄には、以下の 4 種類のデータ形式が記述可能です。

1. 数値定数

   - 数値定数の前に、'+' 又は '−' の演算子をつけて正又は負の値の表現ができます。'+' も '−' もない場合は、正の値として処理します。
   - 数値定数には、2 進、8 進、10 進、16 進数が使用できます。
   - 2 進数 ⇒ 数値の最後に 'B' 又は 'b' を付けて記述してください。
     例) `DATA    .EQU    100110B`
   - 8 進数 ⇒ 数値の最後に 'O' 又は 'o' を付けて記述してください。
     例) `DATA    .EQU    70o`
   - 10 進数 ⇒ 23 、256 のように整数のみで記述してください。
     例) `DATA    .EQU    100`
   - 16 進数 ⇒ 数値の最後に 'H' 又は 'h' を付けて記述してください。先頭が英文字 (A ～ F) で始まる場合は先頭に 0 を付加してください。
     例 1) `DATA    .EQU    64H`
     例 2) `DATA    .EQU    0ABH`

2. 文字列定数

   - 文字列には、ASCII コードで定義されている文字が記述できます。
   - 文字列定数は、' ' '(シングルクォート) 、又は " " '(ダブルクォート) で囲んで記述してください。
     例) `CHAR    .EQU    'A'` ⇒ 41_{16} を設定します。

3. ラベル又はシンボル

   - ラベルは通常アドレスを示しますが PRE77 ではラベルについての処理は行いませんので、値を持ちません。シンボルは 32 ビット整数値を持ちます。

4. 式

   - 数式は、演算子、数値定数、文字列定数、シンボルの組み合わせで構成します。
   - 式は左から右に計算します (演算子の優先順位はありません) 。
     例 1) `2*3` ⇒ 結果は 6 になります。
     例 2) `2+6/2` ⇒ 結果は 4 になります。
   - 値は、32 ビットの符号付き整数値として演算します。
   - 表 3.1に、PRE77 で使用可能な演算子の一覧表を示します。

表 3.1: 演算子一覧表

演算子 [1]	内容
＋	加算
－	減算
＊	乗算
／	除算
%	除算の余り
＜＜	左シフト
＞＞	右シフト
&	ビットごとの AND
|	ビットごとの OR
^	ビットごとの排他的な OR
＋	正の数を表す単項演算子
－	負の数を表す単項演算子
～	ビットごとの反転を表す単項演算子
BANK [3]	ラベルの上位 8 ビット又はシンボルの 16 ～ 23 ビットを切り出す単項演算子
OFFSET [3]	ラベル又はシンボルの下位 16 ビットを切り出す単項演算子

注意事項

1. 演算は左から右へ行います (演算子の優先順位はありません) 。
   例 1) `2+6/2` ⇒ 結果は 4 になります。
   例 2) `1<<3+1` ⇒ 結果は 9 になります。

2. 一行に、複数の単項演算子を記述することはできません。

第 4 章

構造化記述命令

4.1 構造化命令の機能

構造化命令を用いるとプログラムの流れの表現を、アセンブリ言語のブランチ命令、ジャンプ命令等の分岐命令でなく、if 文や for 文等の構造化ステートメントで記述することができます。また、構造化記述ステートメントで記述した場合には、分岐先を探す必要もなくなります。

4.2 文の種類

構造化記述言語には、以下の 7 種類の文があります。“l”(エル) 及び “ll”(エルエル) を付加した場合は、それぞれロング分岐となります。各文の詳細は付録 B を参照してください。

1. 代入文

   右辺を左辺に代入する。

2. if ～ (l(l))else ～ endif 文

3. lif ～ (l(l))else ～ endif 文

4. llif ～ (l(l))else ～ endif 文

   if 文は制御の流れを 2 方向に変える命令であり、分岐する方向は条件式によって決定されます。

5. for ～ next 文

6. lfor ～ next 文

7. llfor ～ next 文

   for 文は繰り返しを制御する命令であり、指定した条件式が真である間、文を繰り返し実行します。

8. do ～ while 文

9. ldo ～ while 文

10. lldo ～ while 文

    do 文は条件式が満たされている (真である) 間、文を繰り返し実行します。

11. switch ～ case ～ ends 文

12. lswitch ～ case ～ ends 文

13. llswitch ～ case ～ ends 文

    switch 文は条件式の値によって制御をいずれかの文に渡すものです。

14. break 文

15. lbreak 文

16. llbreak 文

    break 文は該当する for 文、do 文、又は switch 文の実行を中止してその次に実行する文に制御を渡します。

17. continue 文

18. lcontinue 文

19. llcontinue 文

    continue 文はそれを含む最小の繰り返しの for 文、do 文中の最後の文の後に仮想的に空文を置き、その空文に制御を渡します。

20. goto 文

21. lgoto 文

22. llgoto 文

    goto 文はプログラム内で明示した任意のアドレスに無条件に分岐します。

4.3 記述上の注意事項

構造化記述言語でプログラミングを行う場合の記述上の注意事項について説明します。

1. 代入文、又は各制御文 (if 文等) の条件式欄に 7700 ファミリの各アドレッシングモードによって参照されるメモリを記述する場合は " [ ] "、又は " { } " で囲んで記述します。

```
例 1)   [ WORK ] = 10
例 2)   if [ WORK ]
             :
        else
             :
        endif
```

2. 代入文、又は各制御命令 (if 文等) の条件式欄にビットシンボルで参照できる任意のビットを記述する場合は " [ ] "、又は " { } " で囲んで記述します。ただし、アキュムレータの任意のビットを参照するために以下の予約語が用意されています。この場合は、" [ ] "、又は " { } " で囲む必要はありません。また、大文字/小文字を区別しませんので、BIT_A0,bit_a0 いずれでも有効です。 表 4.1にアキュムレータビット参照の予約語一覧表を示します。

表 4.1: アキュムレータビット参照予約語一覧表

BIT_A0	アキュムレータ A のビット 0	BIT_B0	アキュムレータ B のビット 0
BIT_A1	アキュムレータ A のビット 1	BIT_B1	アキュムレータ B のビット 1
BIT_A2	アキュムレータ A のビット 2	BIT_B2	アキュムレータ B のビット 2
BIT_A3	アキュムレータ A のビット 3	BIT_B3	アキュムレータ B のビット 3
BIT_A4	アキュムレータ A のビット 4	BIT_B4	アキュムレータ B のビット 4
BIT_A5	アキュムレータ A のビット 5	BIT_B5	アキュムレータ B のビット 5
BIT_A6	アキュムレータ A のビット 6	BIT_B6	アキュムレータ B のビット 6
BIT_A7	アキュムレータ A のビット 7	BIT_B7	アキュムレータ B のビット 7
BIT_A8	アキュムレータ A のビット 8	BIT_B8	アキュムレータ B のビット 8
BIT_A9	アキュムレータ A のビット 9	BIT_B9	アキュムレータ B のビット 9
BIT_A10	アキュムレータ A のビット 10	BIT_B10	アキュムレータ B のビット 10
BIT_A11	アキュムレータ A のビット 11	BIT_B11	アキュムレータ B のビット 11
BIT_A12	アキュムレータ A のビット 12	BIT_B12	アキュムレータ B のビット 12
BIT_A13	アキュムレータ A のビット 13	BIT_B13	アキュムレータ B のビット 13
BIT_A14	アキュムレータ A のビット 14	BIT_B14	アキュムレータ B のビット 14
BIT_A15	アキュムレータ A のビット 15	BIT_B15	アキュムレータ B のビット 15

ビットシンボルの定義は、".EQU" 疑似命令で行います。この定義行は、RASM77 用のアセンブラファイルではコメント行として出力されます。ただし、RASM77 用の記述行はそのまま処理行として、RASM77 へ渡されます。 PRE77 は以下の例で示す書式で記述した ".EQU" 疑似命令のみを処理します。

例 1）メモリビットの場合

```
BITSYM   .EQU   1,1000h  ; ビットシンボルの定義
   if [ DP:BITSYM ]
         :
   else
         :
   endif
```

例 2）アキュムレータビットの場合

```
if BIT_A0
    :
else
    :
endif
if BIT_B3
    :
else
    :
endif
```

3. 代入文、又は各制御文 (if 文等) の条件式に 7700 ファミリの各種レジスタを記述する場合には次のように記述します。また、これらは大文字/小文字を区別しませんので、A, a いずれでも有効です。

A	アキュムレータ A	B	アキュムレータ B
X	インデックスレジスタ X	Y	インデックスレジスタ Y
S	スタックポインタ	DPR	ダイレクトページレジスタ
DT	データバンクレジスタ	PS	プロセッサステータスレジスタ

4. 代入文、又は制御文 (if 文等) の条件式欄に、7700 ファミリ のステータスレジスタ内のフラグを記述する場合には次の様に記述します。また、大文字/小文字を区別しませんので、C, c いずれでも有効です。

C	キャリーフラグ	Z	ゼロフラグ
I	割り込み禁止フラグ	D	10 進演算モードフラグ
XF	インデックスレジスタ長選択フラグ	M	データ長選択フラグ
V	オーバーフローフラグ	N	ネガティブフラグ
IPL	プロセッサ割り込み優先レベル		

例 1) C = 1

例 2) if C == 0
:
else
:
endif

5. PRE77 では以下のような記述 (前方参照) を行った場合はラベルとして処理します (LDA 等によるコードの展開が行われます) 。したがって “BITSYM” が後でビットシンボルとして定義されると展開コードが合わなくなります (このような場合 PRE77 ではエラーとなります) 。ビットシンボル (BITSYM) を記述する場合は、まず定義してから参照するようにプログラムを書き換えてください。

例) if [BITSYM]
:
else
:
endif
:
BITSYM .EQU 1,10h

6. PRE77 では、以下のような記述を行った場合、疑似命令 “.DEFINE” に対する処理は行わず、そのまま RASM77 に処理を渡します。したがって、構造化記述中に文字列定義された文字列を記述した場合も通常の文字列として扱います。この場合、RASM77 では RASM77 は “OTHER”でエラーを出力します。

例) OTHER .DEFINE else
:
if BIT_A0
:
OTHER
:
endif

構造化記述命令についての詳細は付録 B を参照してください。

7. PRE77 は構造化記述行を RASM77 用のアセンブリ言語に変換するために以下のラベルを生成しますので、これらのラベルをユーザーが使用することはできません。
   また、ラベルの最大数 (65535) を越える場合はエラーとなります。表 4.2に、PRE77 が生成するラベルの一覧表を示します。

表 4.2: 生成ラベル一覧表

`..D0 ～ ..D65535`	do ～ while 用ラベル
`..F0 ～ ..F65535`	for ～ next 用ラベル
`..I0 ～ ..I65535`	if ～ else ～ endif 用ラベル
`..S0 ～ ..S65535`	switch ～ case ～ ends 用ラベル

8. 分岐命令を生成する構造化記述命令では、メモリ効率をあげるために各命令の記述によって生成される分岐命令が変わります。表 4.3に生成される分岐命令の一覧表を示します。

表 4.3: 生成分岐命令一覧表

制御文	生成分岐命令	分岐命令バイト数
if, for, do, switch, break, continue, goto	BRA	2
lif, lfor, ldo, lswitch, lbreak, lcontinue, lgoto	BRAL	3
llif, llfor, lldo, llswitch, llbreak, llcontinue, llgoto	JMPL	4

9. PRE77 は、マクロ命令の展開を行わないため、以下の疑似命令及び構造化命令のオペランドには、マクロ引数を記述できません。

   - .INDEX のオペランド
   - .DATA のオペランド
   - .EQU のオペランド
   - 構造化命令の条件式
   - CASE の定数

10. 条件式をアセンブリ言語に変換する際に、レジスタ A を使用するコードを生成する場合があります。次のような記述をすると、レジスタ A を使用するコードを生成しますので、注意して下さい。

    - メモリ変数又は、メモリビット変数のみを条件式として記述した場合。

      例）
      ```
      if [MEM]
       jsr sub1
      endif
      ```

    - アブソリュートロングアドレッシングモードで、メモリビット変数代入文を記述した場合。

11. X,Y レジスタを演算対象とする式を記述した場合に、レジスタ A を使用するアセンブリ言語命令を生成します。

12. PRE77 ではマクロ命令そのものについては処理を行なっておりません。したがってマクロ引数を用いた構造化記述は正しく処理することができません。

13. 構造化記述言語で乗除算命令を記述した場合の生成コードは、7700 ファミリのデータ長選択フラグに設定されているデータ長で処理を行います。よって、演算結果の上位 8 または 16 ビットの値については考慮していません。例えば、乗算結果の上位 8 または 16 ビットの値を利用する場合は、以下に示すように乗算命令を記述した行に続けて、B レジスタの内容をメモリに格納する命令を記述してください。

```
          .data   16
mem1:     .blkw   2
mem2:     .blkw   1
mem3:     .blkw   1
          [mem1] = [mem2] * [mem3]
          [mem1+2] = B
```

4.4 構造化演算子

表 4.4に、構造化記述命令で使用可能な演算子の一覧表を示します。

表 4.4: 構造化演算子一覧表

分類	演算子	内容
単項演算子	＋	正の数を表す
	－	負の数を表す
	～	1 の補数をとる
	＋＋	インクリメント
	－－	デクリメント
2項演算子	＋	加算
	－	減算
	＊	乗算
	／	除算
	%	除算の余り
	&	ビットごとの AND
	|	ビットごとの OR
	^	ビットごとの排他的な OR
	&＆	論理 AND
	||	論理 OR
	＜＜	左シフト
	＞＞	右シフト
比較演算子	＜	より小さい
	＞	より大きい
	＝＝	等しい
	!=	等しくない
	＜＝	小さいか等しい
	＞＝	大きいか等しい

[**注意事項**]

1. 7700 ファミリ命令を生成するすべての演算は符号無しの数値として処理され、ビット長はその処理を行う時点での ".DATA" もしくは ".INDEX" にて指定されたビット長で処理します。 ただし、演算 (+, – のみ) 結果に対して大小比較等を行う場合は、演算結果についてのオーバーフローは考慮しません。従って以下のような大小比較を行なう場合には、オーバーフローが発生しないようにご注意ください (以下の例では work の内容が FE_{16} の場合演算結果は 00_{16} となり条件は成立しません) 。

   例)

```
        .DATA     16
if [work] + 2 > 10
            :
else
            :
endif
```

2. 2 項演算子 "&&" 又は "||"を使用して複数の条件分岐文を一行に記述することができます。

   例 ）

```
if [ WORK ]           if [ WORK ] && [ WORK2 ]
  if [ WORK2 ]                    :
       :                          :
  endif                           :
endif                 endif
```

4.5 マクロ中の構造化命令

構造化命令は、マクロ定義内でも記述が可能です。このときマクロ定義内の構造化処理中の生成ラベルは ".LOCAL" 命令によりマクロローカルラベルとして定義します。

また、PRE77 はユーザーマクロ定義中の構造化命令のコード生成をしますが、マクロ展開はされません。

4.6 RASM77 の命令行、疑似命令行

PRE77 では RASM77 の命令行、疑似命令行は、すべてそのままの形で RASM77 用アセンブリソースファイルへ出力します。したがってこれらの行でのエラーチェックは行いません。

第 5 章

疑似命令

5.1 疑似命令の機能

疑似命令は、命令が目的とする機械語データを生成するようにプリプロセッサ及びアセンブラに対する指示[1]を行います。
構造化記述言語を使用するソースプログラムに記述可能な疑似命令は RASM77 と同一です。ただし、PRE77 で処理を行うのは以下の疑似命令のみです。また、PRE77 が処理する疑似命令の機能は RASM77 と必ずしも同一ではありません。

1. プリプロセス制御

```
.CLINE    .DATA     .END     .ENDFUNC    .EQU
.FUNC     .INCLUDE  .INDEX   .SECTION    .SOURCE
```

5.2 プリプロセス制御

PRE77 では構造化命令の処理を行うために処理されるもので、RASM77 と同一の疑似命令ですが、必ずしも RASM77 と同一の機能ではありません。

.CLINE

ソースレベルデバッグに必要な行番号を設定します。

.DATA

データ長選択フラグ (m) の既定値を宣言します。

.END

プリプロセスの終了を宣言します。PRE77 はこの行以降の内容を処理しません。

[1] 疑似命令の名称は、アセンブラに対する既定値を指示するものを “宣言”、出力ファイルに影響するものを “指定” と呼んでいます。

.EQU

シンボルに数値 (ダブルワードの値) を設定します。

.FUNC .ENDFUNC

ソースレベルデバッグに必要なファンクションの開始、終了を設定します。この疑似命令は PRE77 が生成します。

.INCLUDE

この命令を記述した場所に他のファイルを読み込みます。大きなソースプログラムを分割して編集する場合に使用できます。PRE77 で処理可能なネスティングレベルは 9 レベルまでです。

.INDEX

インデックスレジスタ長選択フラグ (x) の既定値を宣言します。

.SECTION

この行以降のプログラムのセクション名を指定します。プログラムを記述する前に必ずこの疑似命令でセクション名を指定してください。

.SOURCE

ソースレベルデバッグに必要なソースファイル名を指定します。この疑似命令は PRE77 が生成します。

第 6 章

操作方法

6.1 起動方法

PRE77 を実行するには以下の情報 (入力パラメータ) を入力する必要があります。

1. 構造化記述ソースプログラムファイル名 (必須項目)
2. コマンドパラメータ

6.2 入力パラメータ

6.2.1 ソースファイル名

1. PRE77 で処理を行う構造化記述プログラムソースファイル名を指定します。ソースファイル名は必ず入力してください。ソースファイル名は 1 個のみ指定できます。
2. ファイル属性 (.P77) を省略した場合、規定値として属性 '.P77' を選びます。
3. ファイル名をフルネームで指定することにより、 '.P77' 以外の属性も指定可能です。ただし、PRE77 は生成するファイルの属性を '.A77' とするため、ファイル属性に '.A77' を指定した場合はエラーとなります。
4. ファイル名はディレクトリパス指定が可能です。ファイル名のみを指定した場合は、カレントドライブのカレントディレクトリ中のファイルを処理します。以下に C ドライブの WORK ディレクトリ中の TEST.P77 ファイルをアセンブルする例を示します。

例) `A¥>PRE77 C:¥WORK¥TEST<RET>`

6.2.2 コマンドパラメータ

1. コマンドパラメータは大文字/小文字どちらでも有効です。
2. 各パラメータは同時に複数指定することができます。この場合は、各パラメータをスペースで区切って入力してください。

表 6.1に、コマンドパラメータの内容の一覧表を示します。

表 6.1: コマンドパラメータ一覧

パラメータ	内容
–.	画面へのすべてのメッセージ出力を抑止します。
–C	構造化記述命令のソースレベルデバッグを可能とするための処理を行います。
–E	エラーが発生した場合のタグファイルの生成を行います。 また、パソコン版の場合はこのオプション直後に指定されたプログラムの起動をタグファイルを引数として行います。 UNIX 版ではこのオプション直後に何等かの指定があった場合にはエラーとなります。
–L	構造化命令をアセンブリ言語に変換する際に、分岐命令のオペランドにラベルを生成します。 このコマンドパラメータを指定しない場合は、分岐命令のオペランドは相対アドレスになります。
–O	生成ファイルの出力先パスを指定します。 指定はフルパスの指定が可能です。
–RASM77	プリプロセッサ終了後、RASM77 を起動します。 このコマンドパラメータ以降の文字列は RASM77 起動時に RASM77 のコマンドパラメータとします。 即ち “SAMPLE.P77” を処理後 RASM77 の起動を行い RASM77 の出力するプリントファイルを得る場合は以下のように指定します。 `A>PRE77 SAMPLE -RASM77 -L` また、自動的に PRE77 は RASM77 に –C パラメータを指定した形式で起動します。
–S	構造化命令をアセンブリ言語に変換したことを示すコメントを出力します。
–V	PRE77 のバージョン番号を画面に出力して終了します。

注意事項

1. PRE77 は、“–O” オプションが指定されていない場合、構造化記述ソースのあるディレクトリにアセンブリファイルを生成します。

6.3 入力方法

PRE77 は、MS–DOS のプロンプト状態でコマンド行を入力することにより起動します。図 6.1 に起動コマンドの入力例を示します。コマンド行入力に誤りを検出すると、図 6.2 のようにヘルプ画面を表示しプリプリロセスを中止します。

```
A>PRE77 TESTNAME -C<RET>
          ↓     ↓
プリプロセス対象の    コマンドパラメータ
ソースファイル名
```

図 6.1: PRE77 の起動コマンド行入力例

```
7700 Family PREPROCESSOR V.5.00.00
Copyright 1998, MITSUBISHI ELECTRIC CORPORATION
AND MITSUBISHI ELECTRIC SEMICONDUCTOR SYSTEMS CORPORATION
All Rights Reserved.

Usage: pre77 <filename> [-.][-c][-e][-o][-rasm77 <rasm77 options>]

       -.      : all messages suppressed
       -c      : source line information output to .a77 file
       -e      : make tag file
       -o      : select drive and directory for output ( syntax -o/WORK )
       -l      : make structured label
       -s      : structured block message output to .a77 file
       -v      : pre77 version display
       -rasm77 : rasm77 option select
```

図 6.2: パソコン版のヘルプメッセージ

6.4 エラー

6.4.1 エラーの種類

PRE77 実行時に発生するエラーは、以下の原因によるものがあります。

1. OS に関するエラー
   ディスクやメモリ容量の不足等、PRE77 を実行する OS に関わるエラーです。

2. PRE77 のコマンド入力に関するエラー
   PRE77 起動時のコマンド入力に関わるエラーです。

3. 処理対照の構造化記述ファイルの内容に関するエラー
   PRE77 に処理を指定したファイル (.INCLUDE 疑似命令で指定されたファイルも含む) の内容に関するエラーです。

PRE77 はエラーを検出すると図 6.3の様に表示されます。

```
7700 Family PREPROCESSOR V.5.00.00

Copyright 1998, MITSUBISHI ELECTRIC CORPORATION
AND MITSUBISHI ELECTRIC SEMICONDUCTOR SYSTEMS CORPORATION
All Rights Reserved.

now processing ( SAMPLE.P77 ) ← 処理ファイル名を示します。
----*----*-- ← 処理の進行状況表示 (100 行ごとに '–' 、500 行ごとに '*' を表示します) 。
if
SAMPLE.P77 1240 ( TOTAL LINE 1240 ) Error  8: Questionable syntax
```

図 6.3: エラー表示例

6.4.2 エラー情報

PRE77 実行時に発生したエラー及び、ワーニングについての情報を PRE77 が生成するアセンブリソースファイルに出力します。

6.5 OS への戻り値

PRE77 では実行結果を表 6.2のエラーレベルに分けて OS に返します。

表 6.2: エラーレベル

エラーレベル	実行結果の内容
0	正常終了
1	プリプロセス時の対象のファイル内容に関するエラー
2	PRE77 のコマンド入力に関するエラー
3	OS に関するエラー
4	コントロール C 入力による強制終了

6.6 環境変数

PRE77 は以下の環境変数を使用しています。

1. **TMP77**
   プリプロセスする際のテンポラリファイルを作成するディレクトリを指定します。
   この環境変数が設定されていない場合には、テンポラリファイルはカレントディレクトリに作成します。

2. **INC77**
   ".INCLUDE"疑似命令を処理する際のディレクトリを指定します。
   疑似命令で指定されたディレクトリにファイルがない場合には、この環境変数で設定したディレクトリから指定されたファイルを捜します。
   ただし、".INCLUDE"オペランドに絶対パスが指定されていた場合はこの環境変数は利用しません。

3. **BIN77**
   プリプロセッサから、 RASM77 を起動する際に、 RASM77 を検索するディレクトリを指定します。カレントディレクトリに RASM77 がない場合に、この環境変数で指定されたディレクトリを検索します。

付録 A

エラーメッセージ一覧表

A.1 システムエラー一覧

プリプロセス中にシステムエラーを検出すると、エラーメッセージを画面に表示し、PRE77を中止します。表 A.1 に以降にシステムエラー一覧表を示します。

表 A.1: システムエラーメッセージ一覧

エラーメッセージ	内容
Usage:pre77.....	コマンド入力が誤っています ⇒ ヘルプ画面を参照して、コマンドを再入力してください。
Can't open xxx	該当ファイルが見つかりません ⇒ ソースファイル名を確認して再入力してください。
Can't create xxx	該当ファイルが作成できません ⇒ –O パラメータの指定を確認して再入力してください。
Out of disk space	ファイルを作成するための容量がディスクにありません ⇒ ディスク上に空き領域を作ってください。
Can't fined rasm77	RASM77.EXE が見つかりません ⇒ RASM77.EXE をカレントディレクトリ又は環境変数で指定したディレクトリにコピーしてください。
Can't fined command.com for execute xxx	–E オプションで指定されたエディタを起動するために必要な COMMAND.COM ファイルが見つかりません ⇒MS–DOS のコマンドパス指定を確認してください。
Out of heap space	PRE77 が処理するためのメモリ容量が足りません ⇒ シンボル又はラベルの数を減らしてください。

A.2 プリプロセスエラーメッセージ一覧表

プリプロセスエラーを検出すると、エラーメッセージを画面に表示します。表 A.2 にプリプロセスエラー一覧表を示します。

表 A.2: プリプロセスエラーメッセージ一覧

エラー番号	エラーメッセージ	内　　容
1	Division by 0	0 による割り算を行っています。 ⇒ 数式の記述を確認してください。
2	Nesting error	処理のネスティングレベルを越えています。 ⇒ ネスティングレベルがオーバにならないようにプログラムを変更してください。
3	No ';' at the top of comment	コメント欄に ';' がありません。 ⇒ コメント欄の先頭に ';' を記述してください。
4	Operand is expected	必要なオペランドが不足しています。 ⇒ オペランドの記述を確認してください。
5	Questionable syntax	ニーモニックの記述にミスがあります。 ⇒ ニーモニックの記述を確認してください。
6	Label or symbol is reserved word	ラベル、シンボルに予約語が使用されています。 ⇒ ラベル名又はシンボル名を変更してください。
7	Value is out of range	値が範囲を越えています。 ⇒ オペランドの記述を確認してください。

エラー番号	エラーメッセージ	内　　容
8	ELSE not associated with IF	ELSE に対する IF がありません。 ⇒ プログラムを確認してください。
9	ENDIF not associated with IF	ENDIF に対する IF がありません。 ⇒ プログラムを確認してください。
10	NEXT not associated with FOR	NEXT に対する FOR がありません。 ⇒ プログラムを確認してください。
11	WHILE not associated with DO	WHILE に対する DO がありません。 ⇒ プログラムを確認してください。
12	ENDS not associated with SWITCH	ENDS に対する SWITCH がありません。 ⇒ プログラムを確認してください。
13	BREAK not inside FOR,DO or SWITCH	BREAK の使用箇所が不適当です。 ⇒ プログラムを確認してください。
14	CONTINUE not inside FOR or DO	CONTINUE の使用箇所が不適当です。 ⇒ プログラムを確認してください。
15	CASE not inside SWITCH	CASE が SWITCH の範囲外です。 ⇒ プログラムを確認してください。
16	DEFAULT not inside SWITCH	DEFAULT が SWITCH の範囲外です。 ⇒ プログラムを確認してください。
17	Duplicate CASE value	CASE の値として同一値が重複しています。 ⇒ プログラムを確認してください。
18	More than one DEFAULT	SWITCH 中に DEFAULT が複数あります ⇒ プログラムを確認してください。
19	Bit length is different type	.INDEX .DATA にて指定されたビット長では処理できません。 ⇒ ビット長を変更してください。

エラー番号	エラーメッセージ	内　　容
20	Label or symbol is multiple defined	.EQU 疑似命令で同一のシンボルが 2 回以上定義されています。 ⇒ プログラムを確認してください。
21	No .END statement	ソースファイル内に.END 疑似命令がありません。 ⇒ .END をソースファイル中に記述してください。
22	Illegal assign	不適当な代入文を記述しています。 ⇒ プログラムを確認してください。
23	No endif statement	ソースファイル内に if 文に対応した endif 文がありません。 ⇒ プログラムを確認してください。
24	No next statement	ソースファイル内に for 文に対応した next 文がありません。 ⇒ プログラムを確認してください。
25	No while statement	ソースファイル内に do 文に対応した while 文がありません。 ⇒ プログラムを確認してください。
26	No ends statement	ソースファイル内に switch 文に対応した ends 文がありません。 ⇒ プログラムを確認してください。
27	Label overflow	PRE77 が生成するラベル数の制限 (65000) を越えました。 ⇒ ソースファイルを複数に分割して PRE77 で処理しなおしてください。
28	.ENDM not associated with macro	マクロ命令に対する '.ENDM' 記述がありません。 ⇒ '.ENDM' を記述してください。
29	No .SECTION statement	'.SECTION' 記述命令以前に命令コードを生成する記述があります。 ⇒ 命令コードを生成する記述より前に '.SECTION' を記述してください。

A.3 ワーニング一覧表

ワーニングを検出すると、ワーニングメッセージを画面に出力します。表 A.3 にワーニング一覧表を示します。

表 A.3: ワーニングメッセージ一覧

ワーニング番号	ワーニングメッセージ	内容
1	Statement has not effect	命令行として意味を持ちません
2	Not CASE values for SWITCH statement	SWITCH 文中に CASE が一つもありません
3	Statement not preceded by CASE or DEFAULT	SWITCH 文において CASE 、DEFAULT よりも先に命令行があります
4	.EQU operand is not symbol	.EQU 命令オペランドがシンボルではありません
5	.END statement in include file	.END 疑似命令をインクルードファイル中に記述しています。
6	Different DATA or INDEX length	.DATA または.INDEX で指定したビット長と実際の値があっていません。

付録 B

構造化命令一覧

B.1 構造化命令一覧の見方

RASM77 で使用可能な構造化命令をアルファベット順に掲載します。表記上の規則を以下に示します。

1. △、▲はスペース又はタブコードを示しています。
   △の部分は必須、▲の部分は省略可能です。
2. [ ] の部分は省略できます。
3. ラベルを記述する場合 ':'(コロン) は必須です。

B.2 構造化命令一覧

BREAK　　BREAK 文

書式

▲ [ラベル:] ▲ BREAK

内容

- break 文は、該当する for 文、do 文、又は switch 文の実行を中止して、その次に実行する文に制御を渡します。
- for 文、do 文、switch 文の中でのみ使用できます。
- 相対アドレス分岐命令 BRA を生成します。

記述例

```
for [ flag1 ]
       if [ flag2 ]
            break
       endif
       jsr   output
next
```

CONTINUE　CONTINUE 文

書式

▲ [ラベル:] ▲ CONTINUE

内容

- continue 文は、それを含む最小の繰り返し文 for 文、do 文中の最後の文に仮想的に空文を置き、その空文に制御を渡します。
- for 文、do 文の中でのみ使用できます。
- 相対アドレス分岐命令 BRA を生成します。

記述例

```
for [ flag1 ]
    if [ flag2 ]
        continue
    endif
    jsr   output
next
```

DO ～ WHILE

DO 文

書式

▲ [ラベル:] ▲ DO
<文>
▲ [ラベル:] ▲ WHILE △条件式

内容

- do 文は、条件式が満たされている (真である) 間、文を繰り返し実行します。ただし、繰り返すかどうかの判定を文の実行後に行います。そのため、do 文はある条件で繰り返しを終了する場合、終了前にもう一度実行して繰り返しから抜けたい場合に使用すると便利です。
- 条件式に、ever を記述すると無限ループとなります。
- 条件式の詳細は、構文図を参照してください。

記述例

```
do
        jsr   output
while [ flag ]
```

FOR～NEXT　　FOR 文

書式

▲ [ラベル:] ▲ FOR △条件式
　　<文>
▲ [ラベル:] ▲ NEXT

内容

- for 文は、繰り返しを制御する命令で、指定した条件式が真である間、文を繰り返し実行します。
- 条件式に、ever を記述すると無限ループとなります。
- 条件式の詳細は、構文図を参照してください。

記述例

```
for [ flag ]
       jsr   output
next
```

GOTO　　GOTO 文

書式

▲ [ラベル:] ▲ GOTO

内容

- goto 文はプログラム内で明示した任意のアドレスに無条件に分岐します。
- プログラム中の任意の位置に記述できます。
- 分岐先のアドレスはラベルで指定します。
- 相対アドレス分岐命令 BRA を生成します。

記述例

```
        for [ flag1 ]
                if [ flag2 ]
                     goto    LAB1
                endif
                jsr   output
LAB1:
                jsr   inout
        next
```

IF ～ (ELSE) ～ ENDIF　　IF 文

書式

▲ [ラベル:] ▲ IF △条件式
　　<文>
▲ [[ラベル:] ▲ ELSE]
　　<文>
▲ [ラベル:] ▲ ENDIF

内容

- if 文は、制御の流れを 2 方向に変える命令で、分岐する方向は条件式によって決定されます。条件式の演算結果がゼロ (偽) か否 (真) かによって分岐が判定されます。真なら、その直後の命令、偽で else がある場合は else の直後の命令、なければ endif 以降の命令へ制御を渡します。
- else 部は省略可能です。
- if 文の入れ子の数には制限がありません。
- if 文が複雑な入れ子になる場合、if と else の対応関係は最も近い if と else が順次ペアになります。
- 条件式の詳細は、構文図を参照してください。

記述例

```
if [ flag ]
        [ work ] = 1
else
        [ work ] = 2
endif
```

LBREAK　　LBREAK 文

書式

▲ [ラベル:] ▲ LBREAK

内容

- lbreak 文は、該当する for 文、do 文、又は switch 文の実行を中止して、その次に実行する文に制御を渡します。
- for 文、do 文、switch 文の中でのみ使用できます。
- 相対アドレス分岐命令 BRAL を生成します。

記述例

```
for [ flag1 ]
       if [ flag2 ]
            lbreak
       endif
       jsr   output
next
```

LCONTINUE LCONTINUE 文

書式

▲ [ラベル:] ▲ LCONTINUE

内容

- lcontinue 文は、それを含む最小の繰り返し文 for 文、do 文中の最後の文に仮想的に空文を置き、その空文に制御を渡します。
- for 文、do 文の中でのみ使用できます。
- 相対アドレス分岐命令 BRAL を生成します。

記述例

```
for [ flag1 ]
    if [ flag2 ]
        lcontinue
    endif
    jsr   output
next
```

LDO ～ WHILE

LDO 文

書式

▲ [ラベル:] ▲ LDO
<文>
▲ [ラベル:] ▲ WHILE △条件式

内容

- ldo 文は、条件式が満たされている (真である) 間、文を繰り返し実行します。ただし、繰り返すかどうかの判定を文の実行後に行います。そのため、ldo 文はある条件で繰り返しを終了する場合、終了前にもう一度実行して繰り返しから抜けたい場合に使用すると便利です。
- 条件式に、ever を記述すると無限ループとなります。
- 条件式の詳細は、構文図を参照してください。

注意事項

- 生成される分岐命令は、BRAL として出力されます。

記述例

```
ldo
        jsr   output
while [ flag ]
```

LFOR ～ NEXT　　LFOR 文

書式

▲ [ラベル:] ▲ LFOR △条件式
　　<文>
▲ [ラベル:] ▲ NEXT

内容

- lfor 文は、繰り返しを制御する命令で、指定した条件式が真である間、文を繰り返し実行します。
- 条件式に、ever を記述すると無限ループとなります。
- 条件式の詳細は、構文図を参照してください。

注意事項

- 生成される分岐命令は、BRAL として出力されます。

記述例

```
lfor [ flag ]
       jsr   output
next
```

LGOTO　　LGOTO 文

書式

▲ [ラベル:] ▲ LGOTO

内容

- lgoto 文はプログラム内で明示した任意のアドレスに無条件に分岐します。
- プログラム中の任意の位置に記述できます。
- 分岐先のアドレスはラベルで指定します。
- 相対アドレス分岐命令 BRAL を生成します。

記述例

```
        for [ flag1 ]
                if [ flag2 ]
                     lgoto    LAB1
                endif
                jsr   output
LAB1:
                jsr   inout
        next
```

LIF ～ (LELSE) ～ ENDIF　　LIF 文

書式

▲ [ラベル:] ▲ LIF △条件式
　　<文>
▲ [[ラベル:] ▲ LELSE]
　　<文>
▲ [ラベル:] ▲ ENDIF

内容

- lif 文は、制御の流れを 2 方向に変える命令で、分岐する方向は条件式によって決定されます。条件式の演算結果がゼロ (偽) か否 (真) かによって分岐が判定されます。真なら、その直後の命令、偽で lelse がある場合は lelse の直後の命令、なければ endif 以降の命令へ制御を渡します。
- else 部は省略可能です。
- lif 文の入れ子の数には制限がありません。
- lif 文が複雑な入れ子になる場合、lif と lelse の対応関係は最も近い lif と lelse が順次ペアになります。
- 条件式の詳細は、構文図を参照してください。

注意事項

- 生成される分岐命令は、BRAL として出力されます。

記述例

```
lif [ flag ]
       [ work ] = 1
lelse
       [ work ] = 2
endif
```

LLBREAK　　LLBREAK 文

書式

▲ [ラベル:] ▲ LLBREAK

内容

- llbreak 文は、該当する for 文、do 文、又は switch 文の実行を中止して、その次に実行する文に制御を渡します。
- for 文、do 文、switch 文の中でのみ使用できます。
- 絶対アドレス分岐命令 JMPL を生成します。

記述例

```
for [ flag1 ]
       if [ flag2 ]
            llbreak
       endif
       jsr   output
next
```

LLCONTINUE LLCONTINUE 文

書式

▲ [ラベル:] ▲ LLCONTINUE

内容

- llcontinue 文は、それを含む最小の繰り返し文 for 文、do 文中の最後の文に仮想的に空文を置き、その空文に制御を渡します。
- for 文、do 文の中でのみ使用できます。
- 絶対アドレス分岐命令 JMPL を生成します。

記述例

```
for [ flag1 ]
    if [ flag2 ]
        llcontinue
    endif
    jsr   output
next
```

LLDO ～ WHILE

LLDO 文

書式

▲ [ラベル:] ▲ LLDO
<文>
▲ [ラベル:] ▲ WHILE △条件式

内容

- lldo 文は、条件式が満たされている (真である) 間、文を繰り返し実行します。ただし、繰り返すかどうかの判定を文の実行後に行います。そのため、lldo 文はある条件で繰り返しを終了する場合、終了前にもう一度実行して繰り返しから抜けたい場合に使用すると便利です。
- 条件式に、ever を記述すると無限ループとなります。
- 条件式の詳細は、構文図を参照してください。

注意事項

- 生成される分岐命令は、JMPL として出力されます。

記述例

```
lldo
       jsr   output
while [ flag ]
```

LLFOR～NEXT　　LLFOR 文

書式

▲ [ラベル:] ▲ LLFOR △条件式
　　<文>
▲ [ラベル:] ▲ NEXT

内容

- llfor 文は、繰り返しを制御する命令で、指定した条件式が真である間、文を繰り返し実行します。
- 条件式に、ever を記述すると無限ループとなります。
- 条件式の詳細は、構文図を参照してください。

注意事項

- 生成される分岐命令は、JMPL として出力されます。

記述例

```
llfor [ flag ]
       jsr   output
next
```

LLGOTO　　LLGOTO 文

書式

▲ [ラベル:] ▲ LLGOTO

内容

- llgoto 文はプログラム内で明示した任意のアドレスに無条件に分岐します。
- プログラム中の任意の位置に記述できます。
- 分岐先のアドレスはラベルで指定します。
- 相対アドレス分岐命令 JMPL を生成します。

記述例

```
        for [ flag1 ]
              if [ flag2 ]
                   llgoto    LAB1
              endif
              jsr   output
LAB1:
              jsr   inout
        next
```

LLIF ～ (LLELSE) ～ ENDIF　　LLIF 文

書式

▲ [ラベル:] ▲ LLIF △条件式
　　<文>
▲ [[ラベル:] ▲ LLELSE]
　　<文>
▲ [ラベル:] ▲ ENDIF

内容

- llif 文は、制御の流れを 2 方向に変える命令で、分岐する方向は条件式によって決定されます。条件式の演算結果がゼロ (偽) か否 (真) かによって分岐が判定されます。真なら、その直後の命令、偽で llelse がある場合は llelse の直後の命令、なければ endif 以降の命令へ制御を渡します。
- llelse 部は省略可能です。
- llif 文の入れ子の数には制限がありません。
- llif 文が複雑な入れ子になる場合、llif と llelse の対応関係は最も近い llif と llelse が順次ペアになります。
- 条件式の詳細は、構文図を参照してください。

注意事項

- 生成される分岐命令は、JMPL として出力されます。

記述例

```
llif [ flag ]
       [ work ] = 1
llelse
       [ work ] = 2
endif
```

SWITCH～CASE～ENDS　　SWITCH 文

書式

▲ [ラベル:] ▲ SWITCH △条件式
▲ [ラベル:] ▲ CASE △定数
　　<文>
▲ [ラベル:] ▲ CASE △定数
　　<文>
　　:
▲ [ラベル:] ▲ DEFAULT
　　<文>
▲ [ラベル:] ▲ ENDS

内容

- switch 文は、条件式の値によって制御をいずれかの文に渡すものです。式の値が文の前に付けられた case 接頭辞の定数と比較され、一致した文に制御が渡されます。式の値と case で指定した値に一致するものがない場合、default 接頭辞があれば、その文に制御が渡されます。default がなければどの文も実行されずに switch 文を抜けます。
- default 部は省略可能です。
- 一致した case 文に制御が渡された後は、それ以降に記述されている case 文も順次実行していきます。もし、次の case 文に移らずに switch 文を抜けたいときには、break,lbreak 又は llbreak 文を用いて switch 文から抜けます。
- 条件式、定数の詳細は、構文図を参照してください。

注意事項

- SWITCH の条件式及び CASE の定数に、マクロ引数を記述することはできません。

記述例

```
switch [ work ]
    case 1
        jsr   output1
        break
    case 2
        jsr   output2
        break
    case 3
        jsr   output3
        break
    default
        jsr   output4
ends
```

LSWITCH ～ CASE ～ ENDS　　LSWITCH 文

書式

▲ [ラベル:] ▲ LSWITCH △条件式
▲ [ラベル:] ▲ CASE △定数
　　<文>
▲ [ラベル:] ▲ CASE △定数
　　<文>
　　:
▲ [ラベル:] ▲ DEFAULT
　　<文>
▲ [ラベル:] ▲ ENDS

内容

- lswitch 文は、条件式の値によって制御をいずれかの文に渡すものです。式の値が文の前に付けられた case 接頭辞の定数と比較され、一致した文に制御が渡されます。式の値と case で指定した値に一致するものがない場合、default 接頭辞があれば、その文に制御が渡されます。default がなければどの文も実行されずに lswitch 文を抜けます。
- default 部は省略可能です。
- 一致した case 文に制御が渡された後は、それ以降に記述されている case 文も順次実行していきます。もし、次の case 文に移らずに lswitch 文を抜けたいときには、break,lbreak 又は llbreak 文を用いて lswitch 文から抜けます。
- 条件式、定数の詳細は、構文図を参照してください。

注意事項

- lswitch を記述した場合、break が生成する分岐命令は BRAL になります。
- SWITCH の条件式及び CASE の定数に、マクロ引数を記述することはできません。

記述例

```
lswitch [ work ]
    case 1
        jsr   output1
        break
    case 2
        jsr   output2
        break
    case 3
        jsr   output3
        break
    default
        jsr   output4
ends
```

LLSWITCH～CASE～ENDS　　LLSWITCH 文

書式

▲ [ラベル:] ▲ LLSWITCH △条件式
▲ [ラベル:] ▲ CASE △定数
　　<文>
▲ [ラベル:] ▲ CASE △定数
　　<文>
　　:
▲ [ラベル:] ▲ DEFAULT
　　<文>
▲ [ラベル:] ▲ ENDS

内容

- llswitch 文は、条件式の値によって制御をいずれかの文に渡すものです。式の値が文の前に付けられた case 接頭辞の定数と比較され、一致した文に制御が渡されます。式の値と case で指定した値に一致するものがない場合、default 接頭辞があれば、その文に制御が渡されます。default がなければどの文も実行されずに llswitch 文を抜けます。
- default 部は省略可能です。
- 一致した case 文に制御が渡された後は、それ以降に記述されている case 文も順次実行していきます。もし、次の case 文に移らずに llswitch 文を抜けたいときには、break,lbreak 又は llbreak 文を用いて llswitch 文から抜けます。
- 条件式、定数の詳細は、構文図を参照してください。

注意事項

- llswitch を記述した場合、break が生成する分岐命令は JMPL になります。
- SWITCH の条件式及び CASE の定数に、マクロ引数を記述することはできません。

記述例

```
llswitch [ work ]
    case 1
        jsr   output1
        break
    case 2
        jsr   output2
        break
    case 3
        jsr   output3
        break
    default
        jsr   output4
ends
```

＝ 代入文

書式

▲ [ラベル:] ▲左辺▲=▲右辺

内容

- 右辺を左辺に代入します。なお、メモリビット変数、レジスタビット変数、フラグ変数への代入は、0 又は 1 の値をもつ数値定数及び記号定数のみ使用できます。
- 左辺、右辺の詳細は、構文図を参照してください。

記述例

```
C = 0                   ; キャリーフラグを 0 にセットします。
BIT_A5 = 1              ; A レジスタのビット 5 を 1 にセットします。
[ bit0 ] = 0            ; bit0 で指定されるメモリビットを 0 に
                        ; セットします。
[ bit0 ] = 1            ; bit0 で指定されるメモリビットを 1 に
                        ; セットします。
X = Y                   ; Y レジスタの内容を X レジスタに転送します。
[ work ] = 10           ; WORK で指定されるメモリに 10 を転送します。
[ work ] = [ work1 ]    ; WORK で指定されるメモリに WORK1 で指定
                        ; されるメモリの内容を転送します。
```

B.3 構造化命令の構文図

PRE77 で使用可能な構造化命令の文法を構文図により示します。
以下、説明において用いているおもな用語の意味について示します。

- 変数

  メモリ変数、メモリビット変数、レジスタ変数、レジスタビット変数、フラグ変数を総称して変数と呼んでいます。

1. メモリ変数

   7700 ファミリ の各アドレッシングモードで参照される任意のメモリ、又はスタックのことです。記述する場合は [ ] 、又は{ }で囲んで記述します。

[ZZ]	ダイレクト	[ZZ,X]	ダイレクト X
[ZZ,Y]	ダイレクト Y	[(ZZ)]	ダイレクトインダイレクト
[(ZZ,X)]	ダイレクトインダイレクト X	[(ZZ),Y]	ダイレクトインダイレクト Y
[HHLL]	アブソリュート	[HHLL,X]	アブソリュート X
[HHLL,Y]	アブソリュート Y	[HHMMLL]	アブソリュートロング
[HHMMLL,X]	アブソリュートロング X	[nn,S]	スタック
[(nn,S),Y]	スタックポインタリラティブ インダイレクト Y		

2. メモリビット変数

   7700 ファミリ のビットアドレッシングモードで参照される任意のビットのことです。記述する場合は [ ] 、又は{ }で囲んで記述します。ただし、この変数は.EQU 疑似命令により以下のように、参照する以前に定義しておく必要があります。

   例）

```
BITSYM    .EQU 1,1000H        ; ビットシンボルの定義
          if [ BITSYM ]       ; ビットシンボルの参照
                 :
          else
                 :
          endif
```

3. レジスタ変数

   7700 ファミリ の各種レジスタのことです。PRE77 ではこれらのレジスタ名を予約語として処理しますのでそのまま記述してください。[ ] 、又は{ }で囲む必要は有りません。また、大文字/小文字を区別しませんので A 、a いずれでも有効です。

A	アキュムレータ A	B	アキュムレータ B
X	インデックスレジスタ X	Y	インデックスレジスタ Y
S	スタックポインタ	PC	プログラムカウンタ
DT	データバンクレジスタ	DPR	ダイレクトページレジスタ
PS	プロセッサステータスレジスタ		

4. レジスタビット変数

7700 ファミリ のアキュムレータビットアドレッシングモードで参照されるアキュムレータ内の任意のビットのことです。これは PRE77 では予約語として以下のような名前で用意されています。また、大文字/小文字を区別しませんので BIT_A0 、bit_a0 いずれでも有効です。
予約語一覧は、表 4.1 を参照してください。

5. フラグ変数

7700 ファミリ のステータスレジスタ内のフラグのことです。これは PRE77 では予約語として以下のような名前で用意されています。また、大文字/小文字を区別しませんので C 、c いずれでも有効です。

C	キャリーフラグ	Z	ゼロフラグ
I	割り込み禁止フラグ	D	10 進モードフラグ
XF	インデックスレジスタ長選択フラグ	M	データ長選択フラグ
V	オーバフローフラグ	N	ネガティブフラグ
IPL	プロセッサ割り込み優先レベル		

- WITH_C

キャリー付き演算指定のことです。これは PRE77 では予約語として用意されています。また、大文字/小文字を区別しませんので WITH_C 、with_c いずれでも有効です。

例)

```
[ work ] = [ work ] << 2 with_c    work の内容をキャリーを含めて
                                   左に 2 回シフトする。
```

- EVER

永久ループ指定のことです。これは PRE77 では予約語として用意されています。また、大文字/小文字を区別しませんので EVER 、ever いずれでも有効です。

例)

```
for ever        ; 永久ループの指定
   :
next
```

- 定数

数値定数、文字定数、記号定数、又はこれらを演算子で組み合わせたものを総称して定数と呼んでいます。

1. 数値定数

数値そのものを示します。

2. 文字定数

指定した文字をアスキーコードとして処理します。記述する際には‘ ’ ’ か‘ ” ’ で囲って指定してください。

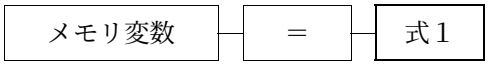
3. 記号定数

英数字及び特殊記号 (＊、__、?、.) を指定します。記述方法は構文図を参照してください。

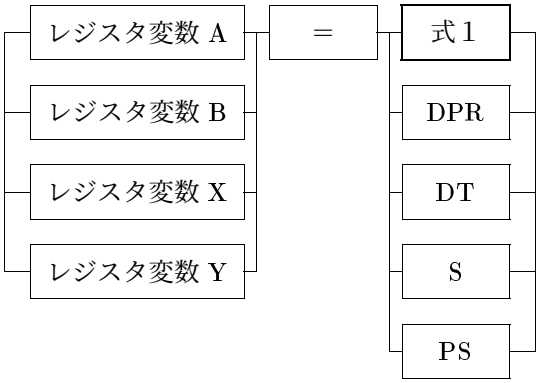
- 変数名変数名の記述規則を構文図で示します。

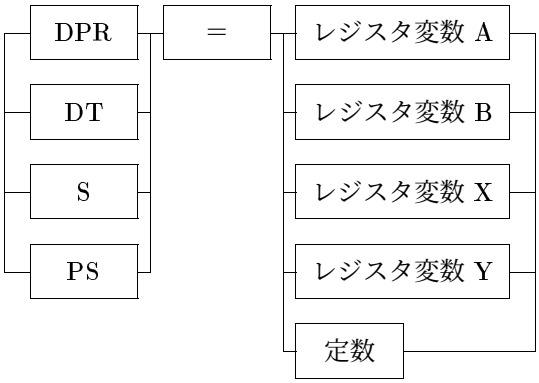
●代入文

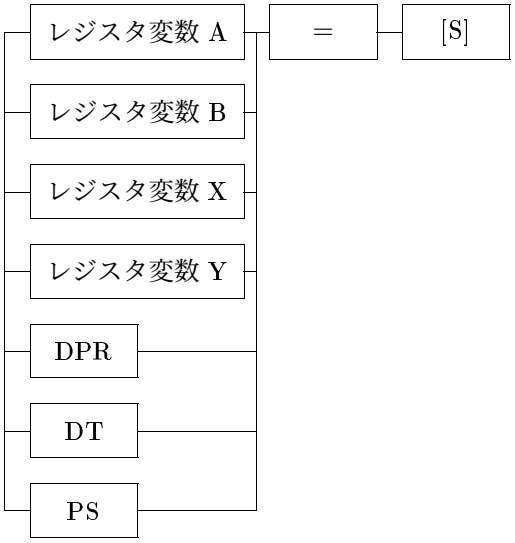
○メモリ変数代入文



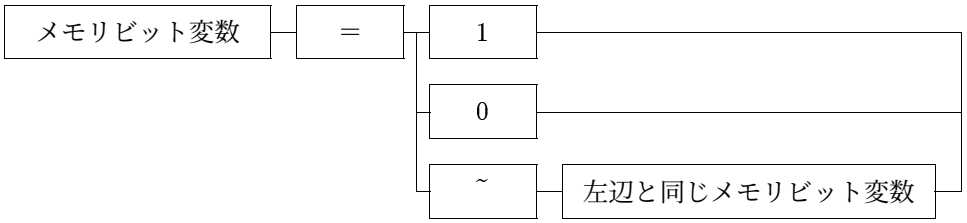
○レジスタ変数代入文

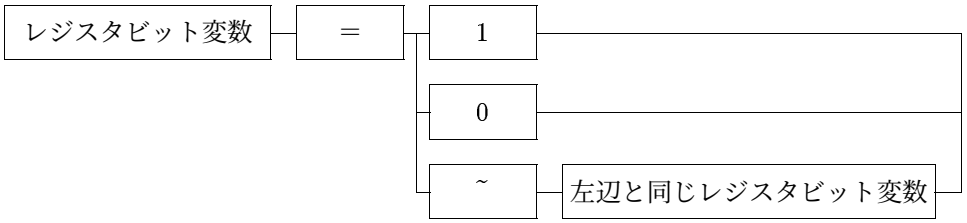




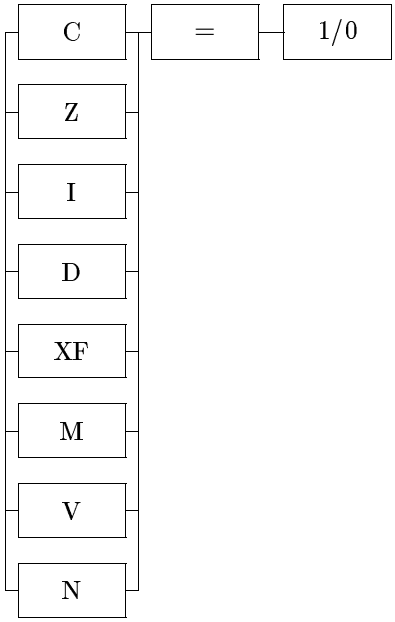


○メモリビット変数代入文

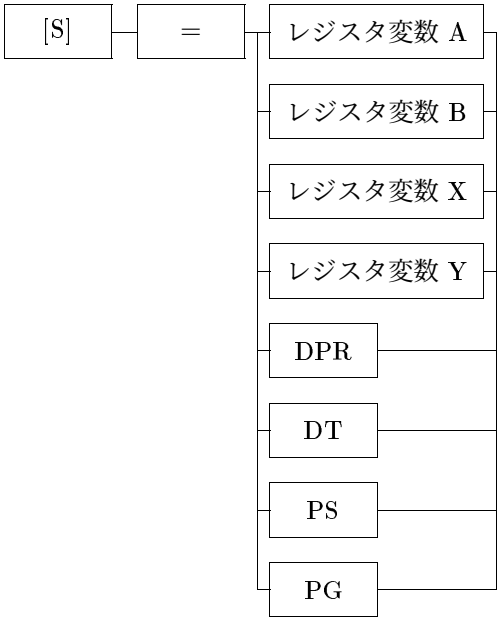




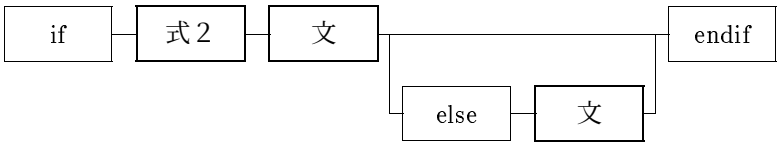
∘フラグ変数代入文



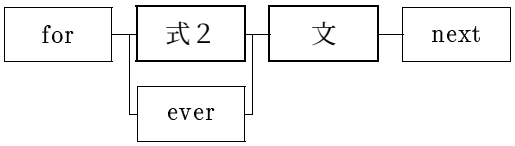
∘スタックフレーム変数代入文



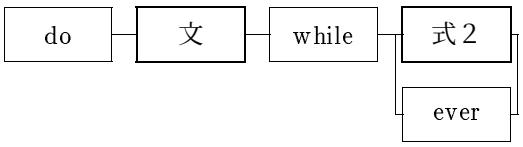
●if 文



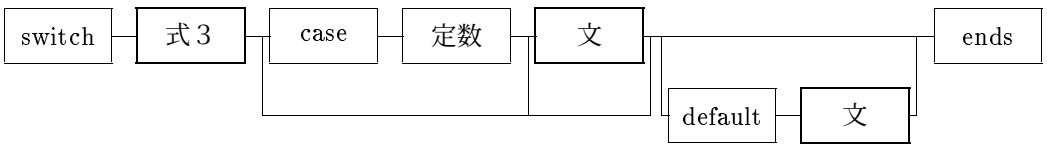
●for 文



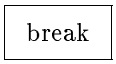
●do 文



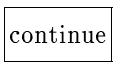
●switch 文



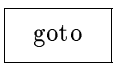
●break 文



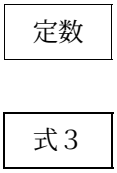
●continue 文



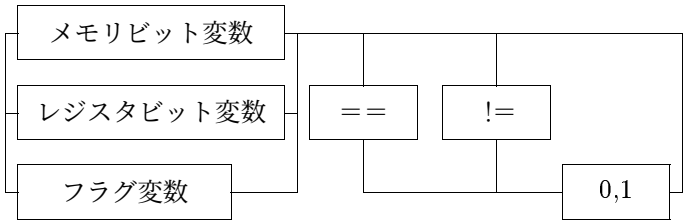
●goto 文

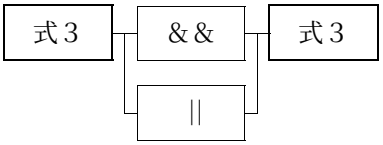


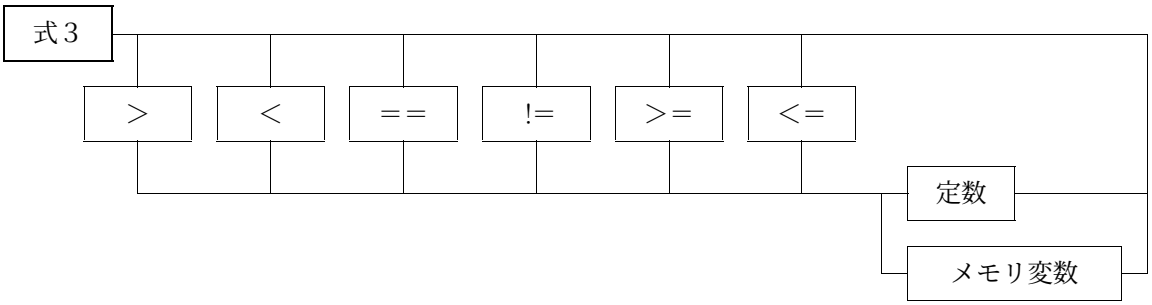
●式 1



●式 2

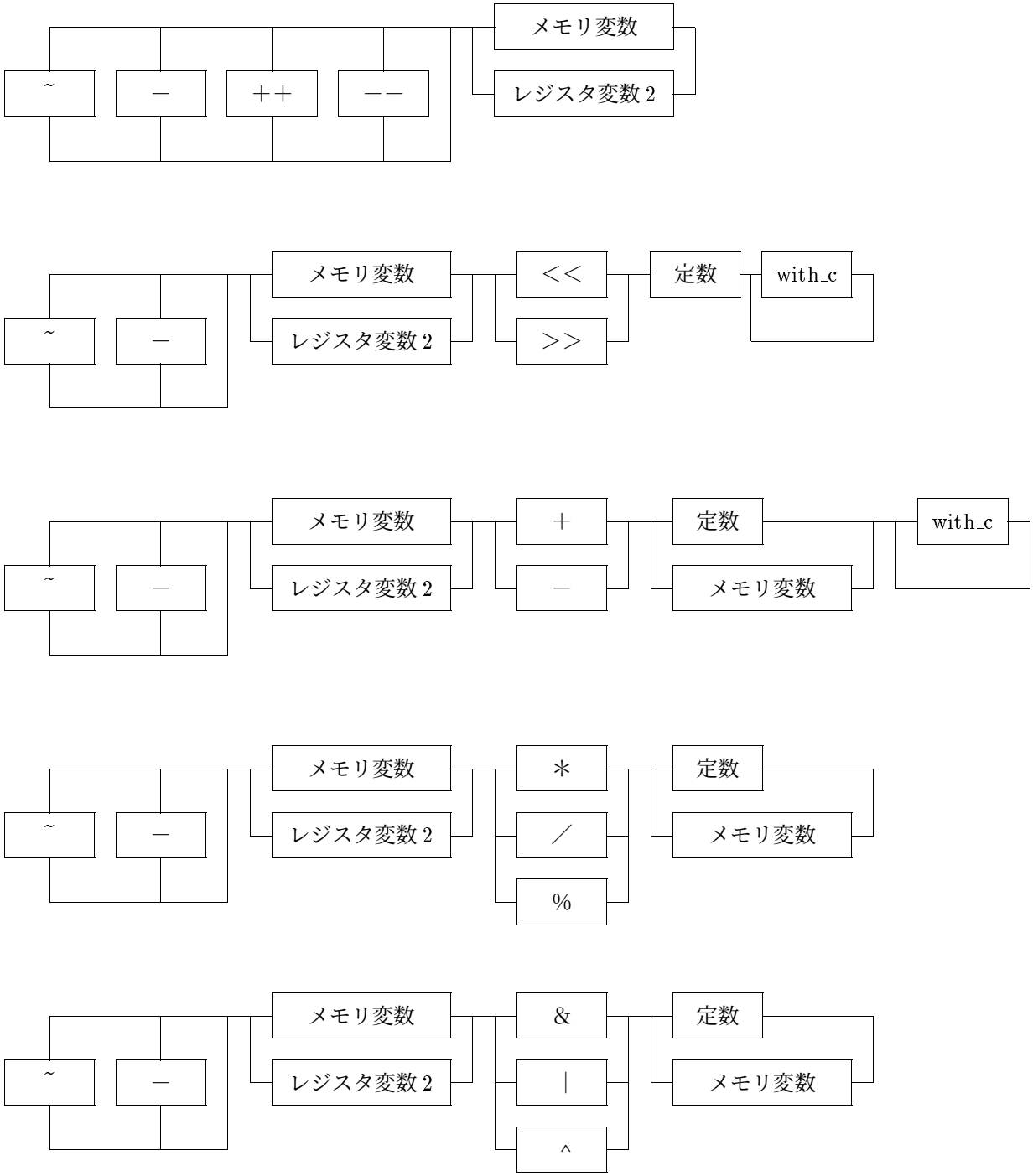






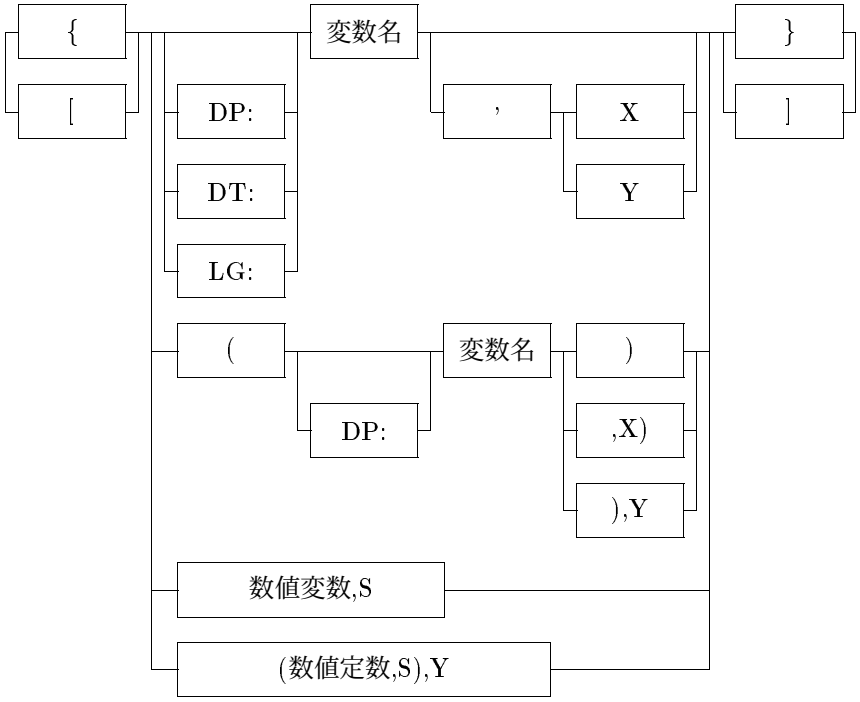
注意事項　式２のフラグ変数は、M および、XF は除きます。

●式 3

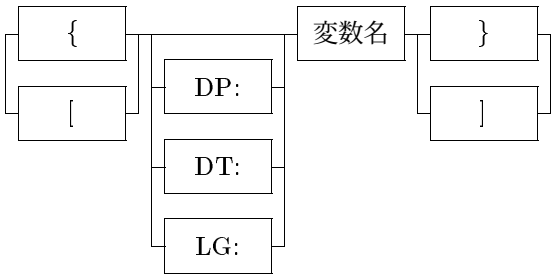

メモリ変数
レジスタ変数 2
~
－
＋＋
－－
メモリ変数
レジスタ変数 2
＜＜
＞＞
定数
with_c
~
－
メモリ変数
レジスタ変数 2
＋
－
定数
メモリ変数
with_c
~
－
メモリ変数
レジスタ変数 2
＊
／
%
定数
メモリ変数
~
－
メモリ変数
レジスタ変数 2
&
|
^
定数
メモリ変数

●変数

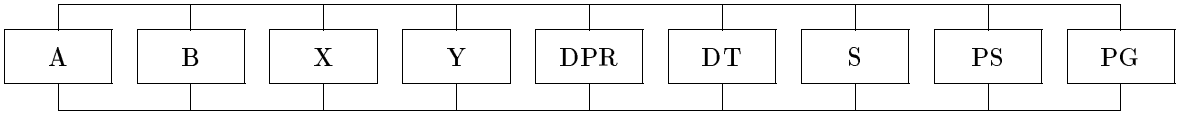
○メモリ変数



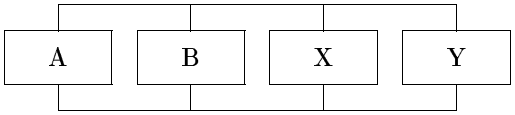
○メモリビット変数



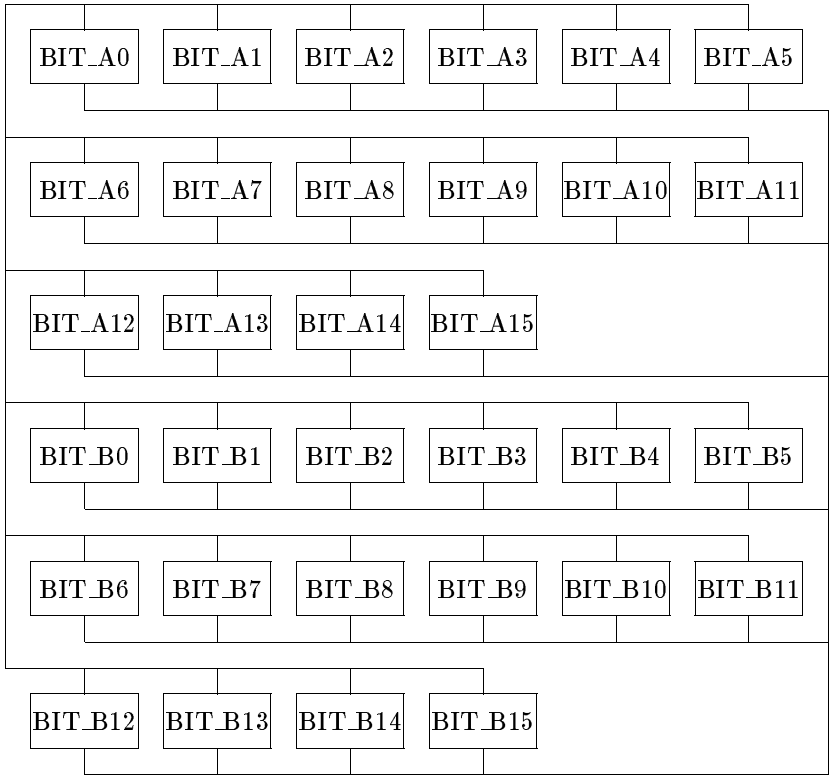
◦レジスタ変数



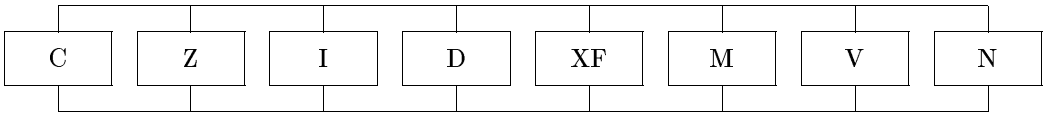
◦レジスタ変数 2



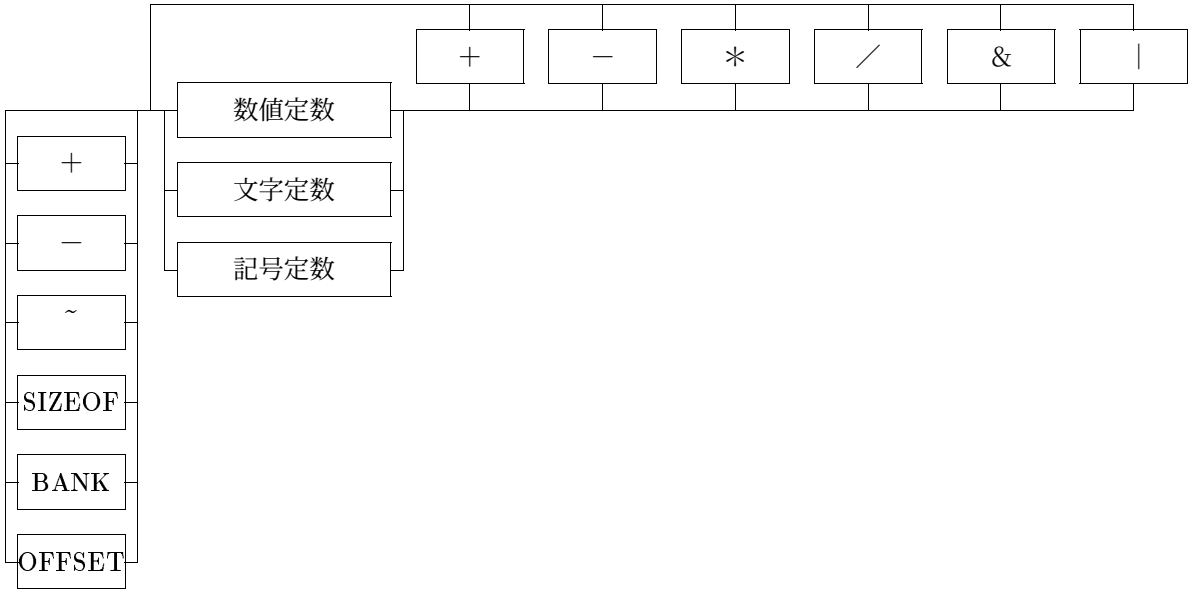
○レジスタビット変数



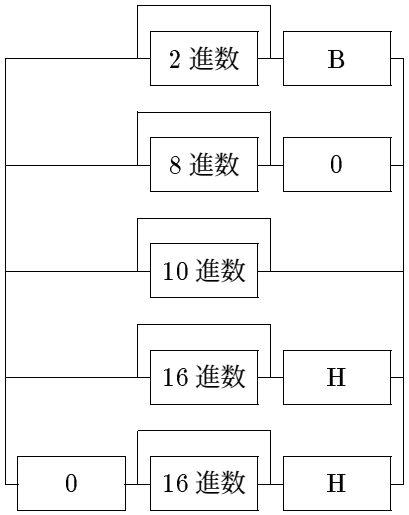
○フラグ変数



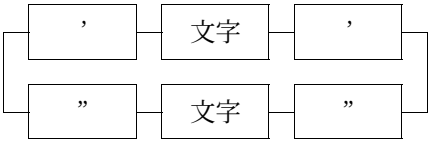
○定数


＋
－
＊
／
&
|
数値定数
文字定数
記号定数
＋
－
~
SIZEOF
BANK
OFFSET

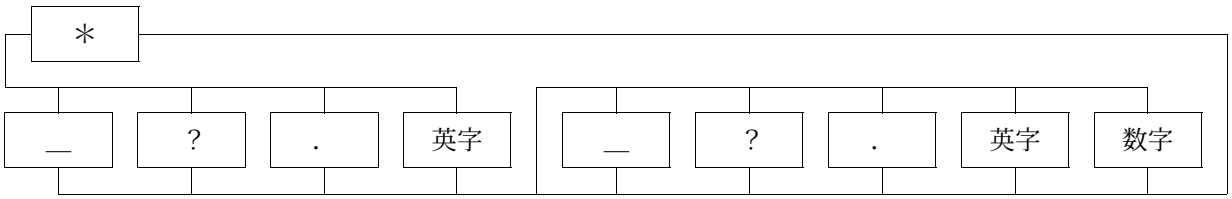
○数値定数


2 進数
B
8 進数
0
10 進数
16 進数
H
0
16 進数
H

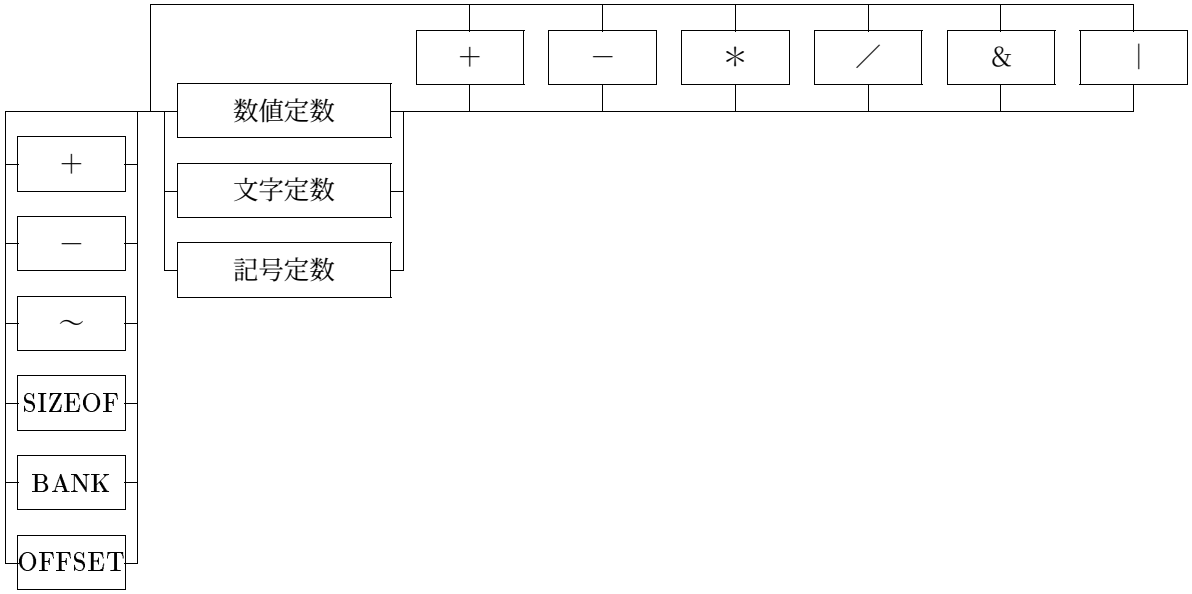
○文字定数



○記号定数



○変数名



付録 C

疑似命令一覧

C.1 疑似命令一覧の見方

PRE77 で処理を行う疑似命令をアルファベット順に掲載します。表記上の規則を以下に示します。

1. 説明中の [ ] 内の部分は省略可能部分を示しています。
2. △、▲はスペース又はタブコードを示しています。△の部分は必須、▲の部分は省略可能を示します。
3. 以下の説明ではラベルと疑似命令間は▲で示しています。ラベルを記述する場合 ':'(コロン) は必須です。

C.2 疑似命令一覧

.CLINE 行頭情報出力

書式

▲.CLINE △数値

内容

- 構造化命令ソースデバッグ時に必要な行番号を設定します。
- 本疑似命令は、PRE77 が自動で作成するものです。

注意事項

- 本疑似命令を元にソースデバッグが可能となりますので、PRE77 が作成する以外の記述時にはワーニングを出力しその行はコメント行に変換を行います。

.DATA データ長宣言 (既定値は 16)

書式

▲.DATA △数式

内容

- CPU 内部のデータ長 (8 又は 16) を宣言します。数式の値が 8 であれば 8 ビット、16 であれば 16 ビットを示します。
- 本疑似命令は、M フラグと関連するイミディエイトアドレッシングモードの即値データ長に影響します。
- SEM 、CLM 命令等でデータ長を変更する場合には、本疑似命令で宣言を行ってください。
- この命令は PRE77 及び RASM77 に対してデータ長を宣言するだけで、CPU 内部のプロセッサステータスレジスタのデータ長選択フラグ (m) は操作しませんので注意してください。

注意事項

- 本疑似命令のオペランドに、マクロ引数を記述することはできません。

記述例

```
SEM             ; M フラグの設定を行う
.DATA   8       ; データ長の指示を行う
A = A + VALUE   ; 8 ビット加算を行う
```

.END　　プログラム終了指定

書式

▲.END

内容

- ソースプログラムの終了を指定します。
- 本疑似命令以降の行は処理しません。

記述例

```
        .END                    ; プログラムの終了を宣言する
```

.ENDFUNC　　構造化記述プログラム終了指定

書式

▲.ENDFUNC △ラベル

内容

- 構造化記述プログラムの終了を指定します。
- この命令により、ソースラインデバッグが可能となります。
- 本疑似命令は、PRE77 が自動で作成します。

注意事項

- 本疑似命令を元にソースデバッグが可能となりますので、PRE77 が作成する以外の記述時にはワーニングを出力しその行はコメント行に変換を行います。

.EQU 同義定義

書式 1

シンボル△.EQU △数式

書式 2

ビットシンボル△.EQU △数式, 数式

内容

- シンボルに数値を設定します。
- 数式中に使用するシンボルは、この行より前に定義してください。
- 書式 1 はシンボルに 32 ビット符号付き整数値を割り当てます。書式 2 はシンボルに 0~15 のビット値とアドレスを割り当てます。

注意事項

- 数式中には既に定義されたシンボルのみが利用でき、ラベルを用いることはできません。
- 本疑似命令のオペランドに、マクロ引数を記述することはできません。

記述例

```
sym     .EQU     1000H    ; sym を 01000H に割り当てる
flag    .EQU     0,100H   ; flag を 100H のビット 0 に割り当てる
```

.FUNC 構造化記述プログラム開始指定

書式

▲.FUNC △ラベル

内容

- 構造化記述プログラムの開始を指定します。
- この命令により、ソースラインデバッグが可能となります。
- 本疑似命令は、PRE77 が自動で作成します。

注意事項

- 本疑似命令を元にソースデバッグが可能となりますので、PRE77 が作成する以外の記述時にはワーニングを出力しその行はコメント行に変換を行います。

.INCLUDE　ファイル読み込み

書式

▲.INCLUDE △ファイル名

内容

- 本疑似命令を記述した場所に、オペランドで指定した構造化記述プログラムファイルの内容を読み込みます。
- ファイル名はフルネームで指定してください。
- 本疑似命令は 9 レベルのネスティングが可能です。

記述例

```
.INCLUDE    TEST.P77    ; TEST.P77 の内容を読み込む
```

.INDEX　インデックスレジスタ長宣言 (既定値は 16)

書式

▲.INDEX △数式

内容

- CPU 内部のインデックスレジスタ長 (8 又は 16) を指示します。
- 数式の値が 8 であれば 8 ビット、16 であれば 16 ビットを示します。
- “CLP X” 、“SEP X” 命令等でインデックスレジスタ長選択フラグを変更する場合には、本疑似命令で宣言を行ってください。

注意事項

- 本疑似命令はアセンブラに対してインデックスレジスタ長を宣言するだけで、CPU 内部のプロセッサステータスレジスタのインデックスレジスタ長選択フラグ (x) は操作しませんので注意してください。
- 本疑似命令のオペランドに、マクロ引数を記述することはできません。

記述例

```
SEP     X       ; X フラグの設定を行う
.INDEX  8       ; インデックスレジスタ長を指定する
X = VALUE       ; 8 ビットロードを行う
```

.SECTION　セクション名指定

書式

▲.SECTION △セクション名

内容

- PRE77 では、本疑似命令の行以降のプログラムに対し、構造化記述プログラムの開始であることを認識します。
- セクション名には任意の名前を付けることが可能です。同じ名前をもつセクションがファイル内に複数個存在してもかまいません。

注意事項

- 本疑似命令は、プログラム記述を始める前に必ず記述してください (ない場合はエラーになります) 。
- 本擬似命令を RASM77 の条件アセンブル擬似命令 ".IF" の処理行に記述しないでください。PRE77 では条件アセンブルを処理しませんので、ソース行情報を出力するオプションを付加して生成したアセンブリソースを RASM77 で処理した場合エラーとなります。

記述例

```
        .SECTION    DATA    ; DATA セクションを開始する。
tabletop:
        .BYTE   "ABCDEFG"
         :
        .SECTION     PROG   ; PROG セクションを開始する。
_init:
        .DATA        16
        .INDEX       16
        X = 0
        for X <= 20
            [PORT0] = tabletop,X
            X = X+1
        next
```

.SOURCE　ソースファイル名設定

書式

▲.SOURCE △ソースファイル名

内容

- ソースデバッグ時に必要なファイル名を設定します。
- 本疑似命令は、PRE77 が自動で作成します。

注意事項

- 本疑似命令を元にソースデバッグが可能となりますので、PRE77 が作成する以外の記述時にはワーニングを出力しその行はコメント行に変換を行います。

第 3 部

7700 ファミリ 用
リンケージエディタ

LINK77 操作マニュアル

目 次

図 目 次

表 目 次

第 1 章

LINK77 操作マニュアルの構成

LINK77 操作マニュアルは、以下の章から構成されています。

- 第 2 章 概要
  LINK77 の基本的な機能と、LINK77 が生成するファイルについて紹介します。
- 第 3 章 セクションの機能
  LINK77 の基本的な操作単位であるセクションについて説明します。
- 第 4 章 操作方法
  LINK77 のコマンド入力方法について説明します。
- 付録 A エラーメッセージ一覧表
  LINK77 が表示するエラーメッセージについて、その内容と対策を一覧表で示します。
- 付録 B 7700 ファミリ 専用 HEX 形式
  1M バイト以上のメモリ領域を持つ機械語データを出力する時に使用される 7700 ファミリ 専用 HEX 形式の仕様を示します。

第 2 章

概要

LINK77 は、RASM77 で生成したリロケータブルファイルをライブラリファイルとリンクし、7700 ファミリ 用機械語データファイルを生成します。

2.1 機能

LINK77 は、デバッガコントロールソフトウェア及び LIB77[1] と共に利用できます。また、これらの各機能を十分生かすため以下の機能を備えています。

1. 複数のリロケータブルファイル中の同一セクション[2]名を持つ領域を、連結して配置します。
2. セクションの配置順序やセクション単位ごとの開始アドレスを指定できます。
3. LIB77 によって作成したライブラリファイルを利用できます。
4. 7700 ファミリ の全メモリ空間 (16M バイト) に対応する機械語データを生成できます。
5. デバッグ時に便利なマップファイルを生成します。
6. デバッガコントロールソフトウェアがシンボリックデバッグを行うために必要なシンボルファイルを生成します。

2.2 生成ファイル

LINK77 では、以下の 3 種類のファイルを生成します。

1. 機械語データファイル (以下 HEX ファイルと呼びます)
   - アドレス範囲が 1M バイト以内の場合、拡張インテル 16 進形式で出力します。アドレス範囲が 1M バイトを越えた場合、7700 ファミリ 専用 HEX 形式[3]で出力します。

[1] 7700 ファミリ 用ライブラリアンのプログラム名。
[2] セクションとは、プログラムを構成する ROM 領域や RAM 領域のような物理的に異なる性質をもった要素単位を指します。詳細については第 4 章を参照ください。
[3] 7700 ファミリ 専用 HEX 形式の仕様については付録 B を参照ください。

- 出力形式は、アドレス範囲に対応して自動的に変わります。
- ファイル属性は、.HEX です。

2. マップファイル (以下 MAP ファイルと呼びます)

- 各セクションが最終的に配置されたアドレス情報を示します。
- このファイルはプリンタへ出力して、デバッグ時や各セクションのメモリ容量の把握のためにお使いください。
- MAP ファイルは、コマンドパラメータ “–M” を指定した時に出力します。
- ファイル属性は、.MAP です。
- このファイルの構成については次節で説明します。

3. シンボルファイル (以下 SYM ファイルと呼びます)

- デバッガコントロールソフトウェアが、シンボリックデバッグを行う時に必要な各種情報を含んだファイルです。
- SYM ファイルは、コマンドパラメータ “–S” を指定した時に出力します。
- ファイル属性は、.SYM です。

以上の出力ファイル名 (拡張子を除く) は、コマンドオプション “–F” で指定可能です。

2.3 MAP ファイルの構成

図 2.1に、MAP ファイルの出力例を示します。MAP ファイルは、以下の情報を示しています。

1. どのリロケータブルファイルからどれだけのデータがリンクされたかについてのセクション単位での情報。ここには、次の情報を出力しています。

- ATR 部 : 相対属性か絶対属性[4]かを示します。REL は相対、ABS は絶対をそれぞれ示しています。
- TYPE 部 : RAM 領域か ROM 領域かを示します。
- START 部 : 開始アドレスを示します。
- LENGTH 部 : 領域の大きさをバイト数で示します。
- ALIGNMENT 部 : リンク時にワードアライメント[5]を実施したものについて “WORD” を表示します。
- ライブラリファイルをリンクした場合、ライブラリファイル名とリロケータブルファイル名の両方が示されます。リロケータブルファイル名は、() 内に表示します。

[4] アセンブリ言語ソース中に、疑似命令.ORG により開始アドレス指定を行なったものが絶対属性になります。
[5] ワードアライメントは、リンクコマンドで “–W” パラメータが指定された場合に行なわれます。

2. グローバルラベルリスト
   プログラム中のグローバルラベル[6] とその絶対番地を示します。この部分は、コマンドパラメータ “–MS” を指定した時のみ出力します。

3. グローバルシンボルリスト
   プログラム中のグローバルシンボル[7] とその絶対番地を示します。この部分は、コマンドパラメータ “–MS” を指定した時のみ出力します。

```
7700 Family LINKER V.2.02.10    MAP FILE                Mon Mar 23 14:58:09 1998

  SECTION           FILENAME          ATR.  TYPE  START    LENGTH   ALIGNMENT

  WORKRAM           MAIN.R77          ABS   RAM   000000   000080
                    SUB.R77           REL   RAM   000080   000100
                    UTIL.LIB          REL   RAM   000180   000008
                    (CALC.R77)
  PROM              MAIN.R77          REL   ROM   00C000   001800
                    SUB.R77           REL   ROM   00D800   001500
                    UTIL.LIB          REL   ROM   00ED00   000820
                    (CALC.R77)
  DROM              MAIN.R77          REL   ROM   00F520   000023   WORD
                    SUB.R77           REL   ROM   00F544   000030

  GLOBAL LABEL INFORMATION

  ADCNT           000030   COUNT           00009C   DATA0           0000A4
  DATA1           0000A6   MAIN            00C000   TIME            0000C6

  GLOBAL SYMBOL INFORMATION
```

図 2.1: MAP ファイル出力例

[6] 他のファイル中で定義しているラベルを指します。
[7] 他のファイルで疑似命令.EQU で定義しているラベルを指します。

第 3 章

セクションの機能

3.1　セクションの役割

アセンブリ言語で書かれたプログラムは、一般にRAM領域、プログラム領域及び固定データ領域から構成されます。RASM77でソースファイルをアセンブルするとリロケータブルファイルが生成されますが、リロケータブルファイルはこのような領域を1個以上含んでいます。セクションは、この各領域ごとに付けられた名前です。以下では具体的な例を示して、セクションの内容とその使い方を説明します。

最も簡潔な例として2つのリロケータブルファイルMAIN.R77とSUB.R77を想定します。各リロケータブルファイルは図3.1のように、それぞれRAM領域、プログラム領域、固定データ領域を持っています。

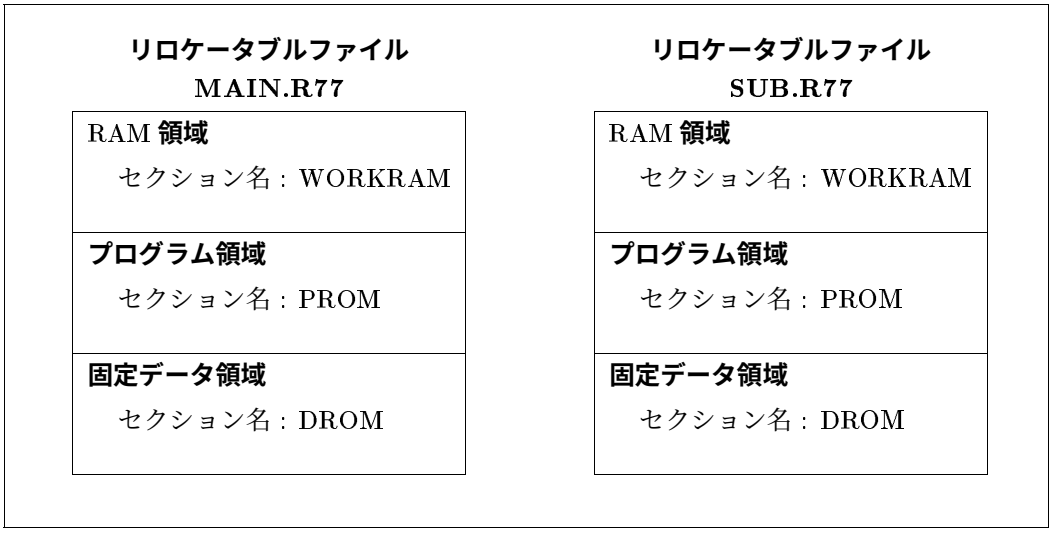


図 3.1: リロケータブルファイル構成図

これらのリロケータブルファイルを図3.2のようなメモリ空間に配置したい場合、あらかじ

めアセンブラ疑似命令.SECTION により結合したいセクションに同一の名前を指定しておきます。こうすることにより、各セクションはリンク時に連続した領域に配置されます。LINK77 では各セクションの開始アドレスをリンク時に指定できます。

図 3.2: システムメモリマップ

以上のように、LINK77 を使用すればリンク時のコマンドによってシステムの最終的なアドレスに対応した機械語データを作成することができます。

3.2 セクションの属性

セクションが持っている基本的な情報には、リンク時の配置アドレスに関するもの (アドレス属性) とその領域の物理的性質に関するもの (物理属性) があります。以下では、各属性の内容について説明します。

3.2.1 アドレス属性

アドレス属性は、ソースプログラムの該当セクション中の疑似命令.ORG の有無によって決定されます。アドレス属性の特徴を以下に示します。

1. 相対属性
   - セクション内に疑似命令.ORG がない場合、そのセクションは相対属性になります。
2. 絶対属性
   - セクション内に疑似命令.ORG が書かれている場合、そのセクションは絶対属性になります。
   - 絶対属性のセクションは、リンク時に開始アドレスの指定を行うことはできません。
3. 複数のリロケータブルファイルに存在する同一名セクションは、異なるアドレス属性を持つことができます。

3.2.2 物理属性

物理属性は、セクションが配置される領域の物理的性質を示しています。物理属性の特徴を以下に示します。

1. ROM 属性
   - ROM 属性のセクションは、リンクの結果 HEX ファイルを生成します。
   - アセンブリ言語ソース中にコードを発生するステートメントが記述されているセクションが ROM 属性になります。
   - アドレス属性との混同を避けるためこの属性を ROM タイプと呼びます。
2. RAM 属性
   - RAM 属性のセクションは、リンクの結果 HEX ファイルを生成しません。
   - アセンブリ言語ソース中に、領域確保疑似命令.BLKB 、.BLKW 等を記述しているセクションが RAM 属性になります。
   - アドレス属性との混同を避けるためこの属性を RAM タイプと呼びます。
3. 同一名のセクションは、すべて同一の物理属性でなければいけません。

3.3 セクションの基本機能

1. 同一名のセクションはリンク時に、連続して配置されます。同一名セクション間に他の名前のセクションが配置されることはありません。
2. 同一名セクション内での配置順序は、リンクコマンド中のリロケータブルファイル名の指定順序に従います。

第 4 章

操作方法

4.1 起動方法

LINK77 を実行するためには、以下の情報 (入力パラメータ) を入力する必要があります。

1. リロケータブルファイル名 (必須項目)
2. ライブラリファイル名
3. セクション制御情報
4. コマンドパラメータ

LINK77 はこれらの情報を入力する方法として、操作環境に応じた以下の 3 通りの方式を用意しています。

1. 対話式入力
2. コマンド行入力
3. コマンドファイル入力

これらの入力方式は、同一の入力パラメータを使用します。また、どの方式を用いても同一のコマンドが実行できます。4.2 節では、この入力パラメータについて説明し、4.3 節で、各入力方式について具体的な例を示しながら説明します。

4.2 入力パラメータ

4.2.1 リロケータブルファイル名

1. リロケータブルファイル名は必ず入力してください。
2. リロケータブルファイルは、属性.R77 を持ったものだけを扱います。コマンド入力では、ファイル属性を省略することができます。

3. ファイル名は、ディレクトリパスが指定可能です。ファイル名のみを記述した場合、カレントドライブのカレントディレクトリ中のファイルを処理します。

4. 最初に指定したリロケータブルファイル名が出力ファイル名になります。コマンドパラメータ “–F” で出力ファイル名を指定した場合は、“–F” パラメータを優先します。

5. 最初に指定したリロケータブルファイルのディレクトリに出力ファイルを出力します。コマンドパラメータ “–O” で出力ファイルを指定した場合は、“–O” パラメータを優先します。

4.2.2 ライブラリファイル名

1. ライブラリファイル名は省略できます。

2. ライブラリファイルは、属性.LIB を持ったものだけを扱います。コマンド入力の時は、ファイル属性を省略することができます。

3. ファイル名には、パス指定ができます。ファイル名のみを指定した場合、カレントディレクトリのファイルを処理します。

4. ライブラリファイルは、リロケータブルファイル中に解決されないグローバルラベル又はグローバルシンボルがあった場合のみ参照します。

4.2.3 セクション制御

1. セクション情報は省略できます。この場合、読み込んだリロケータブルファイル中で出会ったセクション順に配置されます。

2. セクション情報の指定は、相対属性のセクションに対して行ってください。絶対属性のセクションは、セクション指定の有無に関わらず疑似命令.ORG で指定した固定のアドレスから配置されます。

3. セクション配置の指定は、アドレスの下位に配置するセクションから順に、セクション名をスペースで区切って記述してください。

4. 各セクションの開始アドレスは、該当セクション名の後に ‘=’(イコール) に続けてアドレスを記述してください。アドレスは、16 進数で指定してください。この時、先頭の ‘0’(ゼロ) と最後の ‘H’ は不用です。

5. 相対属性のセクションの開始アドレスを省略した場合は、0 番地から配置します。

6. セクション制御がない場合、相対属性のセクションは前のセクションの直後に配置します。ただし、コマンドパラメータ “–W” によりワードアライメントが指定されている場合は、ワード境界に配置します。

7. セクション名は、大文字/小文字を区別します。

8. セクションのアドレスオーバーラップはエラーになります。
   ただし、コマンドパラメータ “–A”を指定すると、絶対アドレスをオーバーラップさせることができます。これにより、SFR 領域などの絶対アドレスラベルを外部参照指定せず、疑似命令 “.INCLUDE”により、複数のプログラムファイルに読み込むことが可能になります。

4.2.4 コマンドパラメータ

コマンドパラメータは、リンクの出力ファイル、バージョン確認の有無、ワードアライメント指定等の制御を行ないます。表 4.1に、コマンドパラメータの内容を示します。

表 4.1: コマンドパラメータ一覧表

コマンドパラメータ	内容
–A	同名の絶対属性セクションのオーバーラップを許可します。グローバルなメモリ領域を共有結合する場合に利用できます。
–BRAL	飛び先番地が xxFFFFH 番地となる BRAL 命令の命令コードを検出した際には、ワーニングメッセージを表示します。
–C	特定の分岐命令 [1] がバンク境界にあった場合にワーニングを出力します。ただし、命令の機械語と同じ値のデータが存在した場合もワーニングメッセージを出力します。
–F	出力ファイル名を指定します。指定の書式は次のようになります。 –FTEST : 出力ファイルを、TEST.HEX 、TEST.MAP 、TEST.SYM の名前で出力します。
–M	MAP ファイルの出力を行ないます (セクション情報のみ)
–MS	MAP ファイルを、グローバルラベル及びグローバルシンボルリスト付きで出力します。
–N	ソースファイル中の疑似命令.OBJ 、.LIB により指定された.R77 、.LIB ファイルの参照指定情報を無視します。
–O	出力ファイルのディレクトリを指定します。 指定の書式は次のようになります。 –OC:¥USR¥WORK C ドライブの ¥USR¥WORK を出力ディレクトリに指定します。
–S	SYM ファイルの出力を行ないます。
–V	リロケータブルファイル間のバージョン整合性の確認を行ないます。ただし、確認を行うためには、アセンブリ言語ソース中で疑似命令.VER によりバージョンの指定を行なっておく必要があります。
–W	セクションの配置をワードアライメントで行ないます。

注意事項

1. 7700 ファミリ では、特定の分岐命令が各バンクの最上位番地又はバンクを横切って配置される場合、プログラムで指定したアドレスには分岐せずに、次のバンク内のアドレスに分岐するという制限事項があります。表 4.2に制限のある分岐命令を示します。

表 4.2: 制限分岐命令

命令	アドレッシングモード	バイト数	命令の機械語
RTS	インプライド	1	60_{16}
JMP	アブソリュート	3	$4C_{16}$
	アブソリュート・インダイレクト	3	$6C_{16}$
	アブソリュート・インデクスト X・インダイレクト	3	$7C_{16}$
	アブソリュート・インダイレクトロング	3	DC_{16}
JSR	アブソリュート	3	20_{16}
	アブソリュート・インデクスト X・インダイレクト	3	FC_{16}

2. ワードアライメント

   ワードアライメントとは、セクションの開始をアドレスのワード境界に揃えることです。ワードアライメントを実施するとメモリ容量は増加しますが、セクションの開始命令を必ずワードで取り込むため実行速度が早くなる場合があります (ただし、バス幅が 8 ビットの場合は効果はありません) 。

   ワードアライメントが行なわれた結果、セクション間に 1 バイトのスペースが発生し、かつアドレス上位のセクションが ROM 領域であった場合、ROM 領域の先頭に NOP 命令 (EA_{16}) が書き込まれます。

   従って LINK77 の '–W' コマンドオプションを使用した場合にそのプログラムに奇数サイズのセクションがあった場合には、ワードアライメントを行なうことにより、1 バイドのスペースがとられます。

4.3 入力方法

4.3.1 対話式入力

対話式入力の特徴を以下に示します。

1. リロケータブルファイル、ライブラリファイル、セクション制御コマンド、コマンドパラメータの順に対話式に入力する方式です。
2. リロケータブルファイルやセクションの数が少ないときや、試行錯誤的にアドレス設定を試したい場合などに便利な方式です。
3. コマンド行入力やコマンドファイル入力でコマンドの数が不足している時は、自動的にこの方式に移行します。

対話式入力は、OS のプロンプト状態で LINK77<RET>と入力することにより起動します。LINK77 は、起動後次のような画面を出力します。

```
A>LINK77 <RET>
7700 Family LINKER V.2.02.10
Copyright 1998, MITSUBISHI ELECTRIC CORPORATION
AND MITSUBISHI ELECTRIC SEMICONDUCTOR SYSTEMS CORPORATION
All Rights Reserved.

Relocatable files (.R77) >>
```

図 4.1: LINK77 起動画面

図 4.1の最後の行が、リロケータブルファイルの入力待ちを示しています。>>に続いて、リンク対象のリロケータブルファイルを入力してください。対話式入力では、このように順次ライブラリファイル名、セクション制御、コマンドパラメータの入力待ちとなりますので、図 4.2 のように入力していってください。

```
A>LINK77 <RET>
MELPS 7700 LINKER V.2.02.10
Copyright 1998,  MITSUBISHI ELECTRIC CORPORATION
AND MITSUBISHI ELECTRIC SEMICONDUCTOR SYSTEMS CORPORATION
All Right Reserved.

Relocatable files (.R77) >> MAIN SUB<RET>
Libraries         (.LIB) >> UTIL1 UTIL2<RET>
Section information      >> WORKRAM=100 PROM=C000 DROM<RET>
Command parameter        >> -O¥WORK -M -S<RET>
```

図 4.2: 対話式入力時の画面

4.3.2 コマンド行入力

コマンド行入力の特徴を以下に示します。

1. OS のコマンド入力状態ですべてのコマンド入力を行う方式です。
2. コマンドの長さが、OS により制限されているためリロケータブルファイルやセクションの数が少ない場合に使用できます。
3. バッチファイルや、make ファイル中で実行コマンドを記述する場合に使用できます。
4. 入力パラメータの 4 種類の情報は、‘,’(カンマ) で区切って入力してください。図 4.2と同じコマンドをコマンド行で入力した例を図 4.3に示します。
5. ライブラリ名以降のパラメータが不用な場合も、カンマは必ず入力してください。図 4.3の例で、ライブラリが必要ない場合は図 4.4のようになります。
6. 特別な場合として、コマンドパラメータを省略する場合のみコマンドパラメータがないことを明確に示すために 2 つのカンマが必要になります。図 4.5でコマンドパラメータを省略した例を示します。
7. 入力パラメータの数が不足している時は、自動的に対話式入力状態に移行します。

```
A>LINK77 MAIN SUB, UTIL1 UTIL2, WORKRAM=100 PROM=C000 DROM, -O¥WORK -M
-S<RET>
```

図 4.3: コマンド行入力例 1

```
A>LINK77 MAIN SUB,, WORKRAM=100 PROM=C000 DROM, -O¥WORK -M -S<RET>
```

図 4.4: コマンド行入力例 2(ライブラリ名の省略)

```
A>LINK77 MAIN SUB,, WORKRAM=100 PROM=C000 DROM,,<RET>
```

図 4.5: コマンド行入力例 3(コマンドパラメータの省略)

4.3.3 コマンドファイル入力

コマンドファイル入力の特徴を以下に示します。

1. リンクコマンドをあらかじめエディタによりコマンドファイルとして作成し、起動時にこのファイル名を指定する方式です。
2. コマンドの文字数が多くコマンド行入力が使用できない場合に便利な方式です。
3. バッチファイルや make ファイルで実行コマンドを記述する場合に使用できます。
4. コマンドファイル名の指定は、LINK77 を起動する時に図 4.6のように、ファイル名の前に‘@’をつけて指定します。図 4.6の場合、ファイル名 CMD.DAT の内容がコマンドとして実行されます。
5. コマンドファイルに記述する内容は、コマンド行入力と同じです (ただし、LINK77 の起動を指示する LINK77 の部分は不用) 。改行コードは無視しますので長いコマンドは複数行に分けて記述できます。図 4.5をコマンドファイルで記述すると図 4.7のようになります。
6. コマンドパラメータが不足している場合、LINK77 は対話式入力状態に移ります。例えば、図 4.7の最後の行の 2 つ目のカンマがない場合 LINK77 は、コマンドパラメータの対話式入力画面になります。

```
A>LINK77 @CMD.DAT<RET>
```

図 4.6: コマンドファイルの指定

```
MAIN SUB
,
,WORKRAM=100 PROM=C000 DROM
,,
```

図 4.7: コマンドファイル記述例

4.4 エラー

4.4.1 エラーの種類

LINK77 実行時に発生するエラーは、以下の原因によるものがあります。

1. OS に関するエラー
   ディスクやメモリ容量の不足等、LINK77 を実行する OS 環境に関わるエラーです。付録 A エラーメッセージ一覧表を参照の上、OS のコマンドにより対応してください。

2. LINK77 のコマンド行入力に関するエラー
   LINK77 起動時のコマンド行入力に関わるエラーです。本章の内容を確認の上コマンドを再入力してください。

3. リンク対象のリロケータブルファイルの内容に関するエラー
   グローバルラベルの 2 重定義、外部ラベルの未定義等のリロケータブルファイルの内容に関わるエラーです。該当箇所のソースプログラム等を確認して、必要があればアセンブルから再実行してください。

4. LINK77 の機能に関するエラー
   RASM77 、LIB77 とのバージョン不整合等によるエラーです。不明な点については当社までご連絡ください。

LINK77 は、図 4.8の形式でエラー内容を画面に表示します。付録 A のエラー番号順のエラー一覧表を参照の上処理してください。

```
A>LINK77 MAIN SUB,,WORKRAM=100 PROM=C000 DROM,,
7700 Family LINKER V.2.02.10
Copyright 1998, MITSUBISHI ELECTRIC CORPORATION
AND MITSUBISHI ELECTRIC SEMICONDUCTOR SYSTEMS CORPORATION
All Rights Reserved.

now processing pass 1
processing "MAIN.R77"
ERROR NO.2: Out of heap space

A>
```

図 4.8: エラー表示例

4.4.2 OS への戻り値

OS のバッチファイル等に実行コマンドを記述する場合、実行結果に応じて処理の内容を変えたい場合があります。LINK77 では、実行結果を表 4.3の 5 つのエラーレベルに分けて OS に返すようにしています。エラーレベルの利用方法については、OS の説明書を参照ください。

表 4.3: エラーレベル

エラーレベル	実行結果の内容
0	正常終了
1	リンク対象のリロケータブルファイルの内容に関するエラー
2	LINK77 のコマンド入力に関するエラー
3	OS に関するエラー
4	LINK77 の機能に関するエラー

4.5 環境変数

LINK77 はライブラリファイルを参照する際に、環境変数 “LIB77” が利用できます。環境変数 “LIB77” にライブラリファイルのディレクトリ名を指定しておくと、LINK77 起動時にディレクトリ名の指定が不要になります。以下に A ドライブの USR ディレクトリをライブラリ参照のディレクトリとして環境変数を設定する例を示します。

例) `A>SET LIB77=A:¥USR<RET>`

付録 A

エラーメッセージ一覧表

A.1　リンクエラー一覧表

表 A.1: リンクエラー一覧表

エラー番号	エラーメッセージ	エラー内容と対策
0	xxx file not found	入力指定されたファイルが見つかりません。 入力指定には、疑似命令.OBJ 又は.LIB によるものも含みます。 ⇒ ファイル名を正しく入力してください。
1	Invalid command input	入力コマンドのパラメータ数が 5 個以上か、コマンドの文字数が合計 2048 文字以上です。 ⇒ 上記の範囲以内で再入力してください。
2	Out of heap space	リンカが動作するために必要なメモリが不足しています。 ⇒ パブリックシンボル数を減らしてください。
3	Invalid section information	セクション情報の指定が誤っています。 ⇒ “セクション名=アドレス” の形式で再入力してください。
4	Invalid parameter input ”xxx”	コマンドパラメータの指定が誤っています。 ⇒ 正しく入力してください。
5	Non relocatable file name	リロケータブルファイル名の入力がありません。 ⇒ ファイル名を入力してください。
6	Internal error	LINK77 の内部エラーが発生しました。 ⇒ LINK77 購入先に御連絡ください。

エラー番号	エラーメッセージ	エラー内容と対策
7	xxx relocatable format is mismatch	.R77 ファイルのリロケータブル形式バージョンが違います。 ⇒ このエラーは、ご使用中のアセンブラ又はライブラリアンとリンカの不整合が生じた時に発生します。リンク対象の.R77 及び.LIB ファイルを LINK77 と同じバージョンの RASM77 又は LIB77 で作成してください。
8	Program version is different	疑似命令.VER で宣言されているプログラムバージョンが一致していません。 ⇒ リンク対象のリロケータブルファイルの疑似命令.VER によるバージョン指定を一致させるか、“–V” パラメータによりリンク時のバージョン確認を解除してください。
9	Unresolved label ”xxx” in xxx	該当セクション内で、外部参照宣言されたラベル又はシンボルが定義されていません。 ⇒ 該当ラベル又はシンボルがパブリック宣言されているプログラムをリンクしてください。
10	”xxx” is multiple defined in xxx. others in xxx	該当ラベル又はシンボルが二重定義です。 ⇒ 該当ラベル又はシンボル名を変更してください。
11	Location overlap. SECTION=xxx ADDRESS=xxx in xxx	該当セクションのアドレス空間がオーバーラップしています。 ⇒ 該当セクションのアドレス配置を確認して、アドレスオーバラップが発生しないようにしてください。(絶対属性のセクションの場合は、ソースファイル中の疑似命令.ORG を変更してください。
12	SECTION xxx is an absolute	絶対属性のセクションに対して、セクション制御コマンドで先頭アドレスが指定されています。 ⇒ コマンド入力でのアドレス指定をやめるか、該当セクションを相対属性に変更してください。

エラー番号	エラーメッセージ	エラー内容と対策
14	Can't find SECTION xxx	該当セクションが存在しません。 ⇒ セクション情報を正しく指定してください。(セクション名は大文字/小文字を区別しますのでご注意ください) 。
15	Can't create xxx	該当ファイルが生成できません。 ⇒ "–O" パラメータの指定を確認して再入力してください。
16	File seek error xxx	該当ファイルがシークできません。 ⇒ このエラーは、OS にかかわるエラーです。一般にディスク装置のハードウェア的な異常による場合が多いと考えられます。
17	Expression value is out of range. SECTION=xxx ADDRESS=xxx OFFSET=xxx	該当箇所の演算結果が制限値を超えています。(エラー箇所の情報として、セクション名、絶対アドレス、セクションの先頭からのオフセットを表示します) ⇒ 制限値を越えないよう、プログラムを変更してください。ここでの制限値とは使用されているアドレッシングモードでは、アクセス対象に届かない場合が含まれます。
18	Out of disk space	ファイルを出力するためのディスク領域が不足しています。 ⇒ ディスク上に空き領域を作ってください。
19	Relative jump is out of range. SECTION=xxx ADDRESS=xxx OFFSET=xxx	該当箇所の相対ジャンプ先アクセス対象に届きません。 ⇒ ジャンプ先のラベルが、範囲内にくるようにプログラムを変更してください。(エラー箇所の情報として、セクション名、絶対アドレス、セクションの先頭からのオフセットを表示します)

エラー番号	エラーメッセージ	エラー内容と対策
20	Expression is out of DP range. SECTION=xxx ADDRESS=xxx OFFSET=xxx	ダイレクトアドレッシングで処理された計算式の結果が、疑似命令.DP で宣言されたダイレクトページレジスタの値～+0FF16 の範囲を越えました。 ⇒ 計算結果がダイレクトページレジスタ～+0FF16 の範囲内になるように変更してください。(エラー箇所の情報として、セクション名、絶対アドレス、セクションの先頭からのオフセットを表示します)
21	Expression is out of DT range. SECTION=xxx ADDRESS=xxx OFFSET=xxx	アブソリュートアドレッシングで処理された計算式の結果が、疑似命令.DT で宣言されたデータバンクレジスタのバンク領域を越えました。 ⇒ 計算結果がデータバンクレジスタのバンク内に入るように変更してください。(エラー箇所の情報として、セクション名、絶対アドレス、セクションの先頭からのオフセットを表示します)
22	Out of maximum program size	プログラム領域が最大領域 16M バイト (0FFFFFF16) を超えました。 ⇒ プログラム容量を縮小してください。
23	Section type mismatch in SECTION xxx	該当セクションで ROM タイプと RAM タイプが混在しています。 ⇒ 同一セクションは、ROM タイプ又は RAM タイプに統一してください。
24	Pointer length mismatch	C コンパイラで使用しているポインタ長が統一されていません。 ⇒ 疑似命令 ".POINTER"の設定値を統一してください。

A.2 ワーニング一覧表

表 A.2: ワーニング一覧表

ワーニング番号	ワーニングメッセージ	ワーニング内容と対策
0	XXX instruction exist at end of bank (address XXXXH)	XXX(RTS,JMP,JSR) 命令がバンク境界 (XXXXH 番地) にあります (このワーニングは、XXX 命令の機械語と同じデータがある場合にも表示されます) 。
1	BRAL specified address is xxFFFFh (description address xxxxh)	xxxxh 番地に飛び先番地が xxFFFFh 番地となる BRAL 命令が配置されている可能性があります (BRAL 命令の検出は機械語コードパターンにより検出します。したがって、該当するデータが実際に BRAL 命令記述によるものであるかどうかはソースプログラムで確認してください)。

付録 B

三菱専用 HEX 形式

B.1　三菱専用 HEX 形式

三菱専用 HEX 形式は、7700 ファミリの全メモリ空間を表現できるように、拡張インテル HEX 形式のアドレスレコードを一部変更しています。以下に、アドレスレコード形式を示します。

$$\underbrace{\texttt{:02}}_{1}\ \underbrace{\texttt{0000}}_{2}\ \underbrace{\texttt{FF}}_{3}\ \underbrace{\texttt{00xx}}_{4}\ \underbrace{\texttt{xx}}_{5}\ \texttt{<RET>}$$

各欄の内容は、以下の通りです。

1. 区切り記号 ':'(コロン) とデータ数 (02 に固定) を示します。
2. アドレス欄 (0000 に固定) を示します。
3. レコードタイプ (FF に固定) を示します。
4. 上位バイトは、00 に固定。
   下位バイトは、以下のデータレコードのアドレス 16 〜 23 ビットの内容 (xx の部分) を示します。
5. 1 バイトのチェックサムを示します。

参考

以下に、拡張インテル HEX 形式のアドレスレコード形式を示します。

$$\underbrace{\texttt{:02}}_{1}\ \underbrace{\texttt{0000}}_{2}\ \underbrace{\texttt{02}}_{3}\ \underbrace{\texttt{xxxx}}_{4}\ \underbrace{\texttt{xx}}_{5}\ \texttt{<RET>}$$

各欄の内容は、以下の通りです。

1. 区切り記号 ':'(コロン) とデータ数 (02 に固定) を示します。
2. アドレス欄 (0000 に固定) を示します。
3. レコードタイプ (02 に固定) を示します。

4. 2 バイトの拡張アドレスを示します。このアドレスは 4 ビットシフト後、データレコード部の 2 バイトのアドレスと加算して最終的な 20 ビットのアドレスを示します。LINK77 が出力する拡張インテル HEX 形式は、上位 4 ビットでページアドレスを示し、下位 12 ビットは常に 0 になります。
5. 1 バイトのチェックサムを示します。

B.2 HEXTOS2

B.2.1 概要

HEXTOS2 は、7700 ファミリ 用 HEX ファイルコンバータです。 LINK77 で作成したインテル HEX フォーマット、及び 7700 ファミリ 専用 HEX フォーマットの機械語ファイル (属性:.HEX) から、モトローラ S フォーマットの機械語ファイル (属性:S2) を作成します。起動方法を以下に示します。

B.2.2 機能

- HEX ファイル名は、コマンド入力行で指定します。属性は省略できます。ただし属性が.HEX 以外の場合や、ファイル名の数が 1 つ以上の場合はエラーになります。
- 出力ファイル名は、入力ファイル名の属性を.S2 に変更したファイル名になります。出力ファイル名を指定することはできません。
- 出力ファイルは、モトローラ S フォーマットの S2 データレコードと S9 エンドレコードで構成されています。
- 以下に HEXTOS2 の起動方法を示します。

[起動方法]

```
A>HEXTOS2 sample.hex <RET>
MOTOROLA S2-RECORD GENERATION UTILITY V.2.00.00
Copyright 1998, MITSUBISHI ELECTRIC CORPORATION
AND MITSUBISHI ELECTRIC SEMICONDACTOR SYSTEMS CORPORATION
All Rights Reserved


Complete!!


A>
```

第 4 部

7700 ファミリ 用
ライブラリアン

LIB77 操作マニュアル

目 次

図 目 次

表 目 次

第 1 章

LIB77 操作マニュアルの構成

LIB77 操作マニュアルは、以下の章から構成されています。

- 第 2 章 概要
  LIB77 の基本的な機能と、LIB77 が生成するファイルについて紹介します。
- 第 3 章 操作方法
  LIB77 のコマンド入力方法について説明します。
- 付録 A エラーメッセージ一覧表
  LIB77 が表示するエラーメッセージについて、その内容と対策を一覧表で示します。

第 2 章

概要

LIB77 は、RASM77 で生成したリロケータブルファイルをライブラリ形式で管理するソフトウエアです。よく利用するサブルーチンをライブラリ化することによりアセンブル時間の短縮や、プログラムの再利用を推進することができます。

2.1 機能

LIB77 は、LINK77[1] と共に利用できます。また、これらの各機能を十分生かすため以下の機能を備えています。

1. LINK77 で参照可能なライブラリファイルの作成及び修正を行います。
2. リロケータブルファイルを、ライブラリファイルに登録します。
3. ライブラリファイル中の不要なリロケータブルファイルを削除します。
4. ライブラリファイル中の古いリロケータブルファイルを、新しく作成したリロケータブルファイルに更新します。
5. ライブラリファイルに登録済のリロケータブルファイルを、登録前の状態に再現します。
6. ライブラリファイル中のリロケータブルファイルの情報を表示します。

2.2 特長

1. リンク処理の高速化。
   リロケータブルファイルをライブラリ化することにより、リンク時に必要な情報の取り出しが高速に行えるようになり、リンク処理を高速化できます。
2. ライブラリファイル中のリロケータブルファイルを更新する際、ファイルの更新日比較により、最新バージョンのリロケータブルファイルのみを更新します (コマンドパラメータ “–U” 指定時) 。

[1] 7700 ファミリ 用リンケージエディタのプログラム名。

2.3 生成ファイル

LIB77 では、以下の 4 種類のファイルを生成します。

1. ライブラリファイル
   - RASM77 で生成したリロケータブルファイルを編集し、ラベル及びシンボルのインデックス部を付けたファイルです。
   - ライブラリファイル中では、リロケータブルファイルを 1 つのモジュールとして管理します (以下ライブラリファイル中のリロケータブルファイルをモジュールと呼びます) 。
   - モジュール名は、ファイル属性も含むリロケータブルファイルの名前と同一です。
   - このファイルはリスト形式になっていないので、プリンタ及び画面への出力は行わないでください。
   - ファイル属性は、.LIB です。
2. リストファイル (以下 LST ファイルと呼びます)
   - ライブラリファイル内のリロケータブルファイル名、グローバルラベル及びシンボル等の一覧表を示します。
   - コマンドパラメータ “–L” を指定した時に生成します。
   - ファイル属性は、.LST です。
   - このファイルの構成については次節で説明します。
3. リロケータブルファイル
   - ライブラリファイル中のリロケータブルファイルを再生したファイルです。
   - コマンドパラメータ “–X” を指定した時に生成します。
   - 再生したリロケータブルファイルは、RASM77 が生成したライブラリ登録前のリロケータブルファイルと同じものです。
   - ファイル属性は、.R77 です。
4. バックアップファイル
   - ライブラリファイルを変更したとき、変更前のライブラリファイルをバックアップファイルとして残します。
   - このファイルは常に生成されます。
   - ファイル属性は、.BAK です。

2.4 LST ファイルの構成

図 2.1、図 2.2、図 2.3に、LST ファイルの出力例を示します。LST ファイルは、以下の情報を示しています。

1. モジュール名一覧表 (図 2.1)
   ライブラリファイル内に登録されている以下の情報を出力しています。

   - Module_name 部: ライブラリファイルに登録されているモジュール名を示します。出力順序は、ライブラリファイルに登録されている順番になります。
   - Offset 部: ライブラリファイルの先頭からモジュールの先頭位置までのバイト数を示します (16 進表記)
   - Module size 部: モジュールのメモリ容量を示します。(16 進表記)

```
LIB77 librarian V.5.00.00                    date 1998-Mar-23 16:17 page    1

Library file name:         SAMPLE.LIB
Relocatable format:        VER.A
Last update time:          1990-Aug-16 15:30
Number of modules:         2
Number of global symbol:   10

Module_name:
getvalue ..... Offset: 00000000H Module size: 00000100H
gettoken ..... Offset: 00000180H Module size: 00000400H
```

図 2.1: LST ファイルの出力例 1(モジュール名一覧表)

2. グローバルラベル及びシンボル一覧表 (アルファベット順)(図 2.2)
   ここでは、以下の 2 つのテーブルを出力しています。

   - パブリックラベルリスト (PUBLIC symbol tabel)

     Symbol_name 部: パブリックラベル又はパブリックシンボルを示します。疑似命令 .EQU で宣言したパブリックシンボルには “(e)” を付加して出力します。

     Module_name 部: パブリックラベル又はパブリックシンボルを含むモジュール名を示します。

   - 外部参照ラベルリスト (EXTERN symbol table)

     Symbol_name 部: 外部ラベル又は外部シンボルを示します。

     Module_name 部: 外部参照指定を行っているモジュール名を示します。

```
LIB77 librarian V.5.00.00                    date 1990-Aug-16 14:30  page  2

PUBLIC symbol table (symbol count = 0010)

  Symbol_name      Module_name           Symbol_name       Module_name

  _chgbin......... getvalue              _chgdigit........ getvalue
  _chghex......... getvalue              _chgoct.......... getvalue
  _gettoken....... gettoken              _getvalue........ getvalue
  _one(e)......... getvalue              _two(e).......... getvalue




LIB77 librarian V.5.00.00                    date 1990-Aug-16 14:30  page  3

EXTERN symbol table (symbol count = 0010)

  Symbol_name      Module_name           Symbol_name       Module_name

  _chgbi1......... getvalue              _chgdigi1........ getvalue
  _chghe1......... getvalue              _chgoc1.......... getvalue
  _gettoke1....... gettoken              _getvalu1........ getvalue
  _isbi1.......... getvalue              _ishe1........... getvalue
```

図 2.2: LST ファイルの出力例 2(グローバルラベル及びシンボル一覧表)

3. グローバルラベル及びシンボルのモジュール別一覧表 (図 2.3)
   ここでは、PUBLIC symbol table と EXTERN symbol table の 2 つのテーブルを出力します。どちらもモジュール名を示したあとに、そのモジュール内のグローバルラベル及びグローバルシンボルを列挙しています。

```
LIB77 librarian V.5.00.00                    date 1990-Aug-16 14:30  page   4

 PUBLIC symbol table

 Module_name : gettoken (symbol count = 0002)

 _gettoken  __gettoken

 Module_name : getvalue (symbol count = 0008)

 _chgbin _chgdigit _chghex _chgoct
 _gettoken _getvalue _one(e) _two(e)




LIB77 librarian V.5.00.00                    date 1990-Aug-16 14:30  page   5

 EXTERN symbol table

 Module_name : gettoken (symbol count = 0002)

 _gettoken1  __gettoken1

 Module_name : getvalue (symbol count = 0008)

 _chgbi1 _chgdigi1 _chghe1 _chgoc1
 _getvalu1 _isbi1 _ishe1 _isoc1
```

図 2.3: LST ファイルの出力例 3(グローバルラベル及びシンボルのモジュール別一覧表)

第 3 章

操作方法

3.1 起動方法

LIB77 を実行するためには、以下の情報を入力する必要があります。

1. ライブラリファイル名 (必須項目)
2. リロケータブルファイル名
3. コマンドパラメータ

LIB77 ではこれらの情報を入力する方法として、以下の 2 通りの方式を用意しています。

1. コマンド行入力
2. コマンドファイル入力

これらの入力方式は、同一の入力パラメータを使用し、どちらの方式を用いても同一のコマンドが実行できます。次節では、この入力パラメータについて説明し、最後に各入力方式について具体的な例を示しながら説明します。

3.2 入力パラメータ

3.2.1 ライブラリファイル名

1. ライブラリファイル名は、必ず入力してください。
2. コマンドパラメータ “−O” の後に、スペースを空けて編集対象のライブラリファイル名を入力してください。
3. ファイル名にはディレクトリパス名が指定できます。パスの指定がない場合、ディレクトリパスとして OS の環境変数 “LIB77” を参照します。更に環境変数の設定もない場合は、カレントディレクトリのファイルを処理します。
4. ファイル属性.LIB は省略可能です。

3.2.2 リロケータブルファイル名

1. リロケータブルファイル名は、スペースで区切り、複数個指定できます。
2. コマンドパラメータ “–F” の後に、スペースで区切り処理対象のリロケータブルファイル名を入力してください。
3. ファイル名にはディレクトリパス名が指定できます。パスの指定がない場合、カレントドライブのカレントディレクトリ中のファイルを処理します。
4. ファイル属性.R77 は省略可能です。

3.2.3 コマンドパラメータ

コマンドパラメータは、ライブラリファイルの操作内容の指定、出力ファイルの指定等の制御を行います。表 3.1に、コマンドパラメータ一覧表を示します。

表 3.1: コマンドパラメータ一覧表

番号 [1]	コマンドパラメータ [2]	内容
1	–O	編集対象のライブラリファイル名を指定します。
2	–F	ライブラリファイルへの追加や削除を行うリロケータブルファイル名を指定します。
3	–A	ライブラリファイルへ、リロケータブルファイルを追加します。
	–R	ライブラリファイル中のリロケータブルファイルを、更新します。
	–D	ライブラリファイル中から、指定したリロケータブルファイルを削除します。
	–L	LST ファイルを出力します。
	–X	ライブラリファイル中から、指定したリロケータブルファイルを登録前の状態に再生します。再生したリロケータブルファイルは、カレントディレクトリに生成します。
4	–V	処理中のファイル名を画面に表示します。
	–U	ライブラリファイル中のリロケータブルファイルを更新する際、最新のファイルのみを更新します。

注意事項

1. 番号は、以下の内容を示します。

   1. 必須項目。編集対象のライブラリファイル名は必ず指定してください。

   2. コマンドパラメータ “–A” 、“–R” 、“–D” 、“–X” のいずれかを指定し、リロケータブルファイルの追加、更新、削除、再生を行うとき、対象となるリロケータブルファイル名の指定を行ってください。

   3. 5 つの内 1 つだけを、必ず指定してください。

   4. 必要に応じて指定してください。

2. コマンドパラメータの大文字/小文字は区別しません。したがって、“–A” 、“–a” のいずれも有効です。

3. ライブラリに登録するリロケータブルファイルは、アセンブルオプション “-S”及び “-C” オプションをつけないでアセンブルしたものを登録するようにしてください。ローカルシンボル情報やデバッグ情報を含んだ状態では、ライブラリ作業に時間がかかりすぎたり、ライブラリファイルサイズが大きくなりすぎたりします。

3.2.4 コマンドパラメータの詳細説明

以下に、コマンドパラメータの詳細を説明します。

1. **–O**

   - 編集対象のライブラリファイル名を指定します。
   - ライブラリファイル名にディレクトリパス名を指定できます。
     以下に B ドライブの USR ディレクトリ内の TEST.LIB を編集対象のライブラリファイルとする例を示します。
     例) `A>LIB77 -LO B:¥USR¥TEST<RET>`
   - ディレクトリパス指定を省略した場合は、カレントディレクトリ内のライブラリファイルを参照します。カレントディレクトリにファイルがない場合に、ディレクトリパスとして OS の環境変数 “LIB77” を参照します。環境変数が以下のように設定されている場合、B ドライブの USR ディレクトリ中のファイルを処理します。
     例) `A>SET LIB77=B:¥USR`
   - ディレクトリパス指定を省略し、環境変数の設定もしていない場合は、カレントドライブのカレントディレクトリ中のファイルを処理します。
   - ファイル属性を省略した場合、既定値として属性.LIB を選択します。
   - ファイルをフルネームで記述することにより、“.LIB” 以外の属性のファイルも指定可能です。

2. **−F**

   - ライブラリファイルに対して追加、更新、削除、再生等を行うリロケータブルファイル名を指定します。
   - スペースで区切って複数個のファイル名を指定できます。
   - ファイル名にディレクトリパス名が指定できます。
     以下にリロケータブルファイルとして B ドライブの WORK ディレクトリ内の SUB1.R77 と C ドライブの SUB2.R77 ファイルを指定する例を示します。
     例)　`A>LIB77 -AO TEST.LIB -F B:¥WORK¥SUB1 C:SUB2<RET>`
   - ディレクトリパス指定を省略した場合は、カレントドライブのカレントディレクトリ中のファイルを処理します。
   - ファイル属性を省略した場合、既定値として属性.R77 を選択します。
   - ファイルをフルネームで記述することにより、".R77" 以外の属性のファイルも指定可能です。

3. **−A**

   - ライブラリファイルにリロケータブルファイルを追加します。
   - "–F" で指定したリロケータブルファイルを、指定した順番でライブラリファイルの最後に追加します。
   - 2 つのファイル内容が同じ場合には、ラベル又はシンボルの 2 重定義エラーが検出されます。ただし、既にライブラリファイル内にあるモジュールと同じ名前のリロケータブルファイルを指定した場合、登録時にエラーは発生しませんが、以降の "–F" で指定されたファイルの処理は、最初に登録されているモジュールに対してのみ行われます。

4. **−R**

   - ライブラリファイル中のモジュール (リロケータブルファイル) を更新します。
   - "–U" と共に使用すると、ライブラリファイル内のモジュールより更新日付けが新しいファイルだけを更新します。

5. **−D**

   - ライブラリファイルから、指定したモジュールを削除します。

6. **−L**

   - LST ファイルを出力します。
   - "–F" でファイル名を指定した場合、指定したリロケータブルファイルの情報を LST ファイルに出力します。
   - "–F" でファイル名を指定しなかった場合は、ライブラリファイル中のすべてのモジュールについての情報を LST ファイルに出力します。

- ファイル属性は、.LST です。

7. **–X**

   - ライブラリファイル中から、指定したリロケータブルファイルを登録前の状態に再生します。ただし、リロケータブルファイルの更新日時は LIB77 が本処理を行った時の日時になります。
   - 再生したリロケータブルファイルは、ライブラリファイルに登録する前のリロケータブルファイルと同じものです。
   - 再生したリロケータブルファイルは、カレントディレクトリに生成します。
   - “–F” でリロケータブルファイル名を指定しなかった場合、ライブラリファイル中のすべてのモジュールを再生します。
   - ライブラリファイルの内容は変化しません。
   - ファイル属性は.R77 です。

8. **–V**

   - LIB77 が現在どのファイルに対して処理を行っているかを画面に表示します。
   - このコマンドパラメータは “–A” 、“–C” 、“–D” 、“–X” 、“–L” のいずれかのコマンドと共に指定してください。

9. **–U**

   - ライブラリファイル中のモジュールを更新する際 (“–R” 指定時) 、最新のファイルのみを更新します。
   - ライブラリファイル中のモジュールを更新する際 “–F” で指定したリロケータブルファイルの更新日時と、ライブラリファイル内に登録されているモジュールの更新日時を比較し、“–F” で指定したリロケータブルファイルの方が新しい時、更新処理を行います。
   - リロケータブルファイルの更新日時は、OS のディレクトリ中の情報を使用します。カレンダ機能を持たないコンピュータ上では、このパラメータは使用しないでください。

3.3 入力方法

3.3.1 コマンド行入力

コマンド行入力の特徴を以下に示します。

1. OS のコマンド入力状態ですべてのコマンド入力を行う方式です。
2. コマンドの長さが、OS により入力文字数が制限されているため、指定するファイルやコマンドパラメータの数が少ない場合に使用できます。

3. バッチファイルや、make ファイルで実行コマンドを記述する場合に使用できます。

4. ライブラリファイル TEST.LIB からモジュール FILE1.R77 を削除する例を図 3.1に示します。

5. ライブラリファイル TEST.LIB にリロケータブルファイル FILE1.R77 、FILE2.R77 を追加する例を図 3.2に示します。

```
A>LIB77 -D -O TEST.LIB -F FILE1<RET>
```

図 3.1: コマンド行入力例 1(モジュールの削除)

```
A>LIB77 -AO TEST.LIB -F FILE2 FILE3<RET>
```

図 3.2: コマンド行入力例 2(リロケータブルファイルの追加)

3.3.2 コマンドファイル入力

コマンドファイル入力の特徴を以下に示します。

1. LIB77 の処理内容をあらかじめエディタによりコマンドファイルとして作成し、起動時にこのファイル名を指定する方式です。

2. 指定するファイル名やコマンドの文字が多くコマンド行入力が使用できない場合に便利な方式です。

3. バッチファイルや、make ファイルで実行コマンドを記述する場合に使用できます。

4. コマンドファイル名の指定は、LIB77 を起動する時に '@' 記号を用いて図 3.3のように指定します。図 3.3の場合、ファイル CMD.DAT の内容を実行します。

5. コマンドファイルに記述する内容は、コマンド行入力と同じです (ただし、LIB77 の起動を指示する LIB77 の部分は不用) 。改行コードはスペースとみなしますので、長いコマンドは複数行に分けて記述できます。図 3.2をコマンドファイルで記述すると図 3.4のように書くことができます。

```
A>LIB77 @CMD.DAT<RET>
```

図 3.3: コマンドファイルの指定

```
-AO
TEST.LIB
-F
FILE2
FILE3
```

図 3.4: コマンドファイル記述例

コマンドが正しく入力されると LIB77 は処理を開始します。LIB77 の各コマンド処理が終了すると、画面に終了メッセージを表示してプログラムを終了します。図 3.5に LIB77 が正常に処理を終了した場合の画面表示例を示します。

```
A>LIB77 -AVO TEST.LIB -F SUB_1 SUB_100<RET>
7700 Family LIBRARY MANAGER V.5.00.00
Copyright 1998, MITSUBISHI ELECTRIC CORPORATION
AND MITSUBISHI ELECTRIC SEMICONDUCTOR SYSTEMS CORPORATION
All Rights Reserved.
< test.lib > Create
APPEND FILE = sub_1,sub_100
MODULE COUNT          000002
GLOBAL SYMBOL COUNT   000019
A>
```

図 3.5: LIB77 正常終了時の画面表示例

3.4 エラー

3.4.1 エラーの種類

LIB77 実行時に発生するエラーは、以下の原因によるものがあります。

1. OS に関するエラー
   ディスクやメモリ容量の不足等、LIB77 を実行する環境に関わるエラーです。付録 A エラーメッセージ一覧表を参照の上、OS のコマンドにより対応してください。

2. LIB77 のコマンド入力に関するエラー
   LIB77 起動時のコマンド入力に関わるエラーです。コマンド入力に誤りがあった場合、図 3.6のようなヘルプ画面を画面に表示します。本章の内容を確認の上、コマンドを再入力してください。

3. 処理対象のリロケータブルファイルの内容に関するエラー
   パブリックラベルが二重定義されている等、リロケータブルファイルの内容に関わるエラーです。LST ファイルを参照して該当箇所を修正してください。

```
MELPS 7700 LIBRARY MANAGER V.5.00.00
Copyright 1998, MITSUBISHI ELECTRIC CORPORATION
AND MITSUBISHI ELECTRIC SEMICONDUCTOR SYSTEMS CORPORATION
All Rights Reserved.
Usage: A>LIB77 -[ardxluv]o <filename> [-f <filename> ...]
       -o : library file name
       -f : relocatable file names
       -a : append command
       -r : replace command
       -d : delete command
       -x : extract command
       -l : listout command
       -u : update check (option)
       -v : verbose (option)
```

図 3.6: コマンド行エラー時のヘルプ画面

LIB77 動作中にエラーが発生すると、画面にエラーメッセージを表示します。付録 A エラーメッセージ一覧表を参照の上、処理してください。

3.4.2 OS への戻り値

OS のバッチファイル等に実行コマンドを記述する場合、実行結果に応じて処理の内容を変えたい場合があります。LIB77 では、実行結果を表 3.2の 4 つのエラーレベルに分けて OS に返すようにしています。エラーレベルの利用方法については、OS の説明書を参照してください。

表 3.2: エラーレベル

エラーレベル	実行結果の内容
0	正常終了
1	処理対象のライブラリファイル又はリロケータブルファイルに関するエラー
2	LIB77 のコマンド入力に関するエラー
3	OS に関するエラー

3.5 環境変数

LIB77 は、以下の環境変数を使用しています。

1. TMP77
   ライブラリアン実行時に生成するテンポラリファイルの作成ディレクトリ名を指定します。この環境変数が設定されていない場合は、テンポラリファイルはカレントディレクトリに作成されます。以下に環境変数 TMP の設定例を示します。
   例) `A>SET TMP77=¥USR¥TMP<RET>`

2. LIB77
   編集対象のライブラリファイル名指定に、環境変数 “LIB77” が使用できます。カレントディレクトリにライブラリファイルがない場合に、環境変数 “LIB77” にライブラリファイルのディレクトリパス名を設定しておくと LIB77 起動時にディレクトリパス名の指定が不要になります。以下に、B ドライブの USR ディレクトリをライブラリファイルのディレクトリパスとして指定する環境変数の設定例を示します。
   例) `A>SET LIB77=B:¥USR<RET>`

付録 A

エラーメッセージ一覧表

A.1　システムエラー一覧表

LIB77 動作中にシステム上のエラーを検出すると、エラーメッセージを画面に表示して処理を中止します。表 A.1にシステムエラー一覧表を示します。

A.2　ライブラリアンエラー一覧表

LIB77 動作中に処理上のエラー検出すると、エラーメッセージを画面に表示して処理を中止します。表 A.2にライブラリアンエラー一覧表を示します。

表 A.1: システムエラー一覧表

エラーメッセージ	エラー内容と対策
Usage:A>LIB77 −[adluvxz]o <filename> [−f <filename>...]	コマンド入力が誤っています。 ⇒ ヘルプ画面を参照して、コマンドを再入力してください。
Can't open xxx	該当ファイルが見つかりません。 ⇒ コマンドパラメータ "–O" や "–F" で指定したファイルが、指定したディレクトリに存在するか確認してください。
Can't create xxx	該当ファイルが生成できません。 ⇒ コマンドパラメータ "–O" の指定を確認して再入力してください。
Out of disk space[1]	ファイルを出力するためのディスク領域が不足しています。 ⇒ ディスク上に空き領域を作ってください。
Input file read error xxx	該当ファイル読み込み中にエラーが発生しました。 ⇒ このエラーは、OS に関わるエラーです。一般にディスク装置のハードウェア的な異常による場合が多いと考えられます。
Internal error	LIB77 の内部エラーが発生しました。 ⇒ LIB77 購入先に御連絡ください。
File seek error xxx	該当ファイルがシークできません。 ⇒ このエラーは、OS に関わるエラーです。一般にディスク装置のハードウェア的な異常による場合が多いと考えられます。

注意事項

1. LIB77 を実行するときは、その実行過程で中間ファイルを作成するため、ライブラリファイルの約 2 〜 3 倍のディスク空き容量が必要です。

表 A.2: ライブラリアンエラー一覧表

エラーメッセージ	エラー内容と対策
xxx is a multiple defined in xxx. others in xxx	パブリックラベル又はパブリックシンボルを二重定義しています。 ⇒ LST ファイルでライブラリファイル内のパブリックラベル又はシンボルを確認してください。
xxx module is not in the library	ライブラリファイル中に該当モジュールが存在しません。 ⇒ LST ファイルでライブラリファイル中のモジュール名を確認してください。
Invalid module or library	ライブラリファイル中のモジュールと指定されたリロケータブルファイルのフォーマットが異なります。 ⇒ 編集対象のライブラリファイルとリロケータブルファイルを同じバージョンの RASM77 で作成してください。
xxx command file not found	該当コマンドファイルが存在しません。 ⇒ 指定したコマンドファイルを確認してください。
Out of heap space	ライブラリアンが動作するために必要なメモリが不足しています。 ⇒ グローバルラベル数を減らしてください。
Too many object module	ライブラリファイル内のモジュール数が多すぎます。 ⇒ ライブラリファイルを分割してください。 一つのライブラリファイル中のモジュール数は、最大 500 個です。
CPU number error	指定したライブラリファイル又はリロケータブルファイルが RASM77 で生成されたものではありません。 ⇒ 指定したライブラリファイル又はリロケータブルファイルを確認してください。

第 5 部

7700 ファミリ 用
クロスリファレンサ

CRF77 操作マニュアル

目 次

図 目 次

表 目 次

第 1 章

CRF77 操作マニュアルの構成

CRF77 操作マニュアルは、以下の章から構成されています。

- 第 2 章 概要
  CRF77 の基本的な機能と、CRF77 が生成するファイルについて紹介します。
- 第 3 章 操作方法
  CRF77 のコマンド入力方法について説明します。
- 付録 A エラー一覧表
  CRF77 が表示するエラーメッセージについて、その内容と対策を一覧表で示します。

第 2 章

概要

CRF77 は、ソースファイル内のラベル及びシンボルの相互参照リスト (以下クロスリファレンスリストと呼ぶ) を生成します。このリストを使用すれば、プログラム修正時にソースファイル各部間の依存関係を容易に把握することができます。

2.1 機能

CRF77 では、以下の機能によりソースファイルの把握を効率的に行うことができます。

1. ラベルの参照命令の種類を参照行番号に表示します。
2. 疑似命令.INCLUDE によるファイル読み込みの実行が可能です。
3. 疑似命令.PAGE によるヘッダの出力が可能です。

2.2 生成ファイル

CRF77 では、次のファイルを生成します。

1. クロスリファレンスファイル (以下 CRF ファイルと呼びます)
   - ラベル及びシンボル名の相互参照リストを示します。
   - 1 行当たりのカラム数は 80 、1 ページの行数は 57 行に固定です。
   - このファイルは、プリンタに出力してデバッグやエディットの際にお使いください。
   - ファイル属性は、.CRF です。

2.3 CRF ファイルの構成

図 2.1に、CRF ファイルの出力例を示します。CRF ファイルは、次の情報を示しています。

1. ラベル及びシンボル名と、その参照行及び定義行。
   定義行に‘#’(シャープ) 、参照行のうちサブルーチンコールによるものに‘&’(アンパサンド) を示します。

2. ラベル及びシンボル名は、32 文字まで表示します。なお、リストはもっとも長い名前に合わせてフォーマッティングを行います。

3. リストのヘッダには、疑似命令.PAGE で指定しているタイトルを表示します (ただし、表示は 30 文字まで行います) 。

4. CRF77 は、ソースプログラム中のラベル及びシンボルの値の判断は行いません。したがって、条件付きアセンブルの判断はできませんのでご注意ください。

```
7700 Family CROSS REFERENCE V.2.10.10                              P. 001

A0          3926   4285   8549   9079   9100
AA          3884   5545   5668
ABEND       9396&  9465&  9587#
ABEND10     9588   9593#
ABENDRT     9590#  9605
ACCHK       1201#  1408
ACCHK5      1213   1237#
ACCHKE      1235   1239   1241   1249#
ADDING      4994   5006#
ADDING0     5013   5014   5016#
ADDING1     5015   5019#
ADDRESS      300    318   1025
ADR_CHK     9154#
ADR_OUT     8302   8336   9145#
ADR_PNT     8157#
ADR_PNT2    8149#
```

図 2.1: CRF ファイル例

第 3 章

操作方法

3.1 起動方法

CRF77 を実行するためには、以下の情報 (入力パラメータ) を入力する必要があります。

1. ソースファイル名 (必須項目)
2. コマンドパラメータ

3.2 入力パラメータ

3.2.1 ソースファイル名

1. ソースファイル名は必ず入力してください。
2. ファイル属性 (.A77) を省略した場合、既定値として属性.A77 を選びます。
3. ファイル名をフルネームで指定することにより、.A77 以外の属性 (例.ASM) のファイルも処理可能です。
4. ファイル名は、ドライブ名が指定可能です。ファイル名のみを記述した場合、カレントドライブ中のファイルを処理します。ディレクトリパスの指定はできません。
5. ソースファイル名は 16 個まで指定できます。

3.2.2 コマンドパラメータ

コマンドパラメータは、ソースファイル中の疑似命令.INCLUDE の検出の有無や、出力ファイルのドライブ名を指定します。表 3.1に、コマンドパラメータの内容を示します。

表 3.1: コマンドパラメータ一覧表

コマンドパラメータ	内容
–O	クロスリファレンスファイルを出力するドライブ名及びディレクトリパスを指定します。指定の書式は、次のようになります。 例) A>CRF77 SRCFILE -OC:¥TMP <RET> クロスリファレンスファイルを C ドライブの TMP ディレクトリに出力します。
–I	疑似命令.INCLUDE を無視します。 RASM77 の '.INCLUDE' 疑似命令では、9 レベルまでのネスティングを許可していますが、CRF77 では、'.INCLUDE' 疑似命令のネスティングは許可されていません。 したがって、'.INCLUDE' 疑似命令がネストしているソースを CRF77 で処理する場合は、本コマンドオプション '–I' を使用してください。

3.3 入力方法

3.3.1 コマンド行入力

CRF77 は、OS のプロンプト状態でコマンド行を入力することにより起動します。図 3.1に、起動コマンドの入力例を示します。

```
A>CRF77 SRCFILE1 SRCFILE2 SRCFILE3<RET>
```

図 3.1: コマンド行入力例

コマンド行入力中に誤りを検出すると、図 3.2のようにヘルプ画面を表示し処理を中止します。

```
A>CRF77<RET>
7700 Family CROSS REFERENCE V.2.10.10
Copyright 1998, MITSUBISHI ELECTRIC CORPORATION
AND MITSUBISHI ELECTRIC SEMICONDUCTOR SYSTEMS CORPORATION
All Rights Reserved.

Usage: crf77 <filename> [-ifilename,..] [-opath]
       -i : not include specified files ( use -ifilename,.... )
       -o : select drive and directory for output ( use -otmp )
```

図 3.2: コマンド行エラー時のヘルプ画面

3.4 エラー

3.4.1 エラーの種類

CRF77 実行時に発生するエラーは、以下の原因によるものがあります。

1. OS に関するエラー
   ディスクやメモリ容量の不足等、CRF77 を実行する環境に関わるエラーです。付録 A エラーメッセージ一覧表を参照の上、OS のコマンドにより対応してください。

2. CRF77 のコマンド行入力に関するエラー
   CRF77 起動時のコマンド行入力に関わるエラーです。本章の内容を確認の上コマンドを再入力してください。

3. 処理対象のソースファイル内容に関するエラー
   疑似命令.INCLUDE によって指定されたソースファイルが存在しない場合に発生します。

CRF77 は、エラーを検出すると図 3.3の形式でエラー内容を画面に表示します。付録 A のエラー一覧表を参照の上処理してください。

```
A>CRF77 SRCFILE1<RET>
7700 Family CROSS REFERENCE V.2.10.10
Copyright 1998, MITSUBISHI ELECTRIC CORPORATION
AND MITSUBISHI ELECTRIC SEMICONDUCTOR SYSTEMS CORPORATION
All Rights Reserved.

now processing pass 1
now making cross reference ( SRCFILE1.A77 )
-----+
 Out of heap space

A>
```

図 3.3: エラー表示例

3.4.2 OS への戻り値

OS のバッチファイル等に実行コマンドを記述する場合、実行結果に応じて処理の内容を変えたい場合があります。CRF77 では、実行結果を表 3.2の 4 つのエラーレベルに分けて OS に返すようにしています。エラーレベルの利用方法については、OS の説明書を参照ください。

表 3.2: エラーレベル

エラーレベル	実行結果の内容
0	正常終了
1	疑似命令.INCLUDE で指定されたソースファイルが存在しない場合に発生するエラー
2	CRF77 のコマンド入力に関するエラー
3	OS に関するエラー

3.5 環境変数

CRF77 は、環境変数は使用していません。

付録 A

エラーメッセージ一覧表

A.1 システムエラー一覧表

CRF77 処理中にシステムエラーを検出すると、エラーメッセージを画面に表示し処理を中止します。表 A.1にシステムエラー一覧表を示します。

表 A.1: システムエラー一覧表

エラーメッセージ	エラー内容と対策
Usage: A>crf77 <filename> [-d] -I[filename,..]	コマンド入力が誤っています。 ⇒ ヘルプ画面を参照して、コマンドを再入力してください。
Can't open xxx	該当ファイルが見つかりません。 ⇒ ソースファイル名を確認して再入力してください。
Can't create xxx	該当ファイルが生成できません。 ⇒ ディスク上に空き領域を作ってください。
Out of disk space	ファイルを出力するためのディスク容量が不足しています。 ⇒ ディスク上に空き領域を作ってください。
Out of heap space	クロスリファレンサが動作するために必要なメモリが不足しています。 ⇒ シンボル又はラベルの数を減らしてください。

A.2 クロスリファレンスエラー一覧表

クロスリファレンス作成に関するエラーを検出すると、エラーメッセージを画面に出力し、処理はそのまま続行します。表 A.2にクロスリファレンスエラーとその内容を示します。

表 A.2: クロスリファレンスエラー一覧表

エラーメッセージ	エラー内容と対策
Can't open include file xxx	疑似命令.INCLUDE で指定されたソースファイルが存在しません。 ⇒ ディレクトリ内容を確認してください。

RASM77 V.5.10 ユーザーズマニュアル

Rev. 1.00
03.08.01
RJJ10J0196-0100Z

RASM77 V.5.10
ユーザーズマニュアル

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

RJJ10J0196-0100Z