お客様各位

カタログ等資料中の旧社名の扱いについて

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

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

2010年4月1日
ルネサスエレクトロニクス株式会社

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

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

ご注意書き

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

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

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

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

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

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

RENESAS

ユーザーズ・マニュアル

保守／廃止

WB77016

ワークベンチ

言語編

μPD77015
μPD77016
μPD77017
μPD77018
μPD77018A
μPD77019
μPD77110
μPD77111
μPD77112
μPD77113
μPD77114

資料番号　U10078JJ6V0UMJ1（第６版）
発行年月　October 2000 N　CP(K)

〔メ　モ〕

保守／廃止

目次要約

〔メ　モ〕

保守／廃止

- 本資料の内容は予告なく変更することがありますので，最新のものであることをご確認の上ご使用ください。
- 文書による当社の承諾なしに本資料の転載複製を禁じます。
- 本資料に記載された製品の使用もしくは本資料に記載の情報の使用に際して，当社は当社もしくは第三者の知的財産権その他の権利に対する保証または実施権の許諾を行うものではありません。上記使用に起因する第三者所有の権利にかかわる問題が発生した場合，当社はその責を負うものではありませんのでご了承ください。
- 本資料に記載された回路，ソフトウエア，及びこれらに付随する情報は，半導体製品の動作例，応用例を説明するためのものです。従って，これら回路・ソフトウエア・情報をお客様の機器に使用される場合には，お客様の責任において機器設計をしてください。これらの使用に起因するお客様もしくは第三者の損害に対して，当社は一切その責を負いません。

M7A 98.8

保守／廃止

本版で改訂された主な箇所

箇　所	内　容
全　般	対象デバイスにμPD77110, 77111, 77112, 77113, 77114を追加。
p.18	**図２－１　インストラクション・メモリ空間**にμPD77110, 77111, 77112, 77113, 77114を追加。
p.19	**図２－２　データ・メモリ空間**にμPD77110, 77111, 77112, 77113, 77114を追加。
p.66	**５.２.12　ハードウエア・ループ命令(ループ)(命令セット12)**に[記述上の注意](７)を追加。
p.68	**５.２.14　制御命令(条件命令以外)**(命令セット14)に[記述上の注意](２)を追加。
p.94	**表７－１　汎用コントロールの初期状態**に$ＰＡＧＥＬＥＮＧＴＨ, $PAGEWIDTH, $RADIXを追加
p.101	**７.５　リスト制御**に(５)PAGELENGTH, (６)PAGEWIDTHを追加。
p.104	**７.７　基数指定**を追加。

本文欄外の★印は，本版で改訂された主な箇所を示しています。

巻末にアンケート・コーナを設けております。このドキュメントに対するご意見をお気軽にお寄せください。

保守／廃止

はじめに

WB77016ワークベンチ（以降，本アセンブラと呼びます）は，ディジタル・シグナル・プロセッサμPD77016ファミリのソフトウエア開発に使用します。

このマニュアルは，μPD77016ファミリ・アセンブリ・プログラミング言語の文法的ことがらについて述べています。

WB77016ワークベンチの各ソフトウエアの操作については，**WB77016　ユーザーズ・マニュアル　操作編**を参照してください。

【対 象 者】

このマニュアルは，μPD77016ファミリの機能およびインストラクションについて理解されている方を対象として書かれています。

【構　　成】

このマニュアルの構成は次のとおりです。

第1章　概　　説

本アセンブラの概要について述べます。

第2章　アドレス空間とセグメント

μPD77016ファミリのアドレス空間，本アセンブラのアドレス空間などについて説明します。

第3章　ソース・モジュール記述形式

μPD77016ファミリ・アセンブリ言語のソース・モジュールの構成，ステートメントの記述方法について説明します。

第4章　言語要素

ソース・モジュールを記述するための基本単位となる言語要素について説明します。

第5章　アセンブリ命令

アセンブリ命令について，フィールドごとにマイクロコードのニモニックの一覧を示します。各命令の詳細説明は，***μ*PD77016ファミリ　ユーザーズ・マニュアル　命令編**をご覧ください。

第6章　疑似命令

アセンブラの疑似命令について説明します。

第7章　汎用コントロール

汎用コントロールについて説明します。

第8章　マ ク ロ

マクロ機能について説明します。

付録A　構 文 図

μPD77016ファミリ・アセンブリ言語の構文図を掲載しています。

付録B　予約語一覧

予約語一覧を掲載しています。

【読 み 方】

μPD77016ファミリは，μPD7701×ファミリ（μPD77015, 77016, 77017, 77018, 77018A, 77019）と，μPD77111ファミリ（μPD77110, 77111, 77112, 77113, 77114）の総称です。特に機能面に違いがない場合は，μPD77016ファミリを該当する製品に読み替えてご使用ください。機能面に違いがある場合は，製品名をあげて個別に説明しています。

アセンブラの疑似命令について知りたい方は，**第５章　アセンブリ命令**をお読みください。汎用コントロールについて知りたい方は，**第７章　汎用コントロール**をお読みください。

【凡　　例】

このマニュアル中で，アセンブラ・パッケージの操作やコントロールなどを説明するために共通に使用される記号の意味を次に示します。

↵	：キャリッジ・リターン（CR）を表します。
△	：１個以上の空白を表します（入力形式などで空白を明確にするために使用しています）。
[ ]	：[ ] 内の文字列が省略可能であることを表します。
{ }	：{ } 内の文字列のうち，１つを選択して入力する必要があることを表します。
“ ”	：“ ”で囲まれた文字そのものを表します。
…	：同一項目を繰り返し入力することを表します。
LF	：ライン・フィードを表します。
▲	：１個以上の空白とコメント欄を表します。
▽	：１個以上の空白，タブ，改行を表します。
数の表記	：２進数　…　0b××××
	８進数　…　0q××××
	10進数　…　××××，0t××××
	16進数　…　0x××××

【関連資料】

関連資料は暫定版の場合がありますが，この資料では「暫定」の表示をしておりません。あらかじめご了承ください。

μPD77016ファミリに関する資料

資料名 / 品名	パンフレット	データ・シート	ユーザーズ・マニュアル		アプリケーション・ノート	
			アーキテクチャ編	命令編	基本ソフトウエア編	ライブラリ編
μPD77016	U12395J	U10891J	U10503J	U13116J	U11958J	U12021J
μPD77015		U10902J				
μPD77017						
μPD77018						
μPD77018A		U11849J				
μPD77019						
μPD77019-013		U13053J				
μPD77110		U12801J	U14623J（作成中）			
μPD77111						
μPD77112						
μPD77113		U14373J				
μPD77114						

開発ツールに関する資料

資　料　名		資料番号
SM77016　ユーザーズ・マニュアル		U11602J
WB77016　ユーザーズ・マニュアル	言語編	このマニュアル
	操作編	U11506J
ID77016　ユーザーズ・マニュアル		U10118J
IE-77016-98, IE-77016-PC　ユーザーズ・マニュアル	ハードウエア編	U13044J
μPD77016　スタータ・キット　ユーザーズ・マニュアル		U13032J
IE77016-CM-LC　ユーザーズ・マニュアル		U14139J
RX77016　ユーザーズ・マニュアル	機能編	U14397J
	コンフィギュレーション・ツール編	U14404J
RX77016　アプリケーション・ノート	HOST API編	U14371J

注意　上記関連資料は，予告なしに内容を変更することがあります。設計などには，必ず最新の資料をご使用ください。

〔メ　モ〕

保守／廃止

目　　次

保守／廃止

図の目次

表の目次

保守／廃止

第1章　概　　説

本マニュアルは，μPD77016ファミリ・アセンブリ言語について説明しています。

WB77016ワークベンチのアセンブラ機能は，μPD77016ファミリ・アセンブリ言語で記述されたプログラムを機械語命令に変換するものです。

WB77016ワークベンチの操作については，**WB77016　ユーザーズ・マニュアル　操作編**を参照してください。

〔メ　モ〕

第２章　アドレス空間とセグメント

２.１　アドレス空間

μPD77016ファミリには，次の３つのメモリ空間があります。

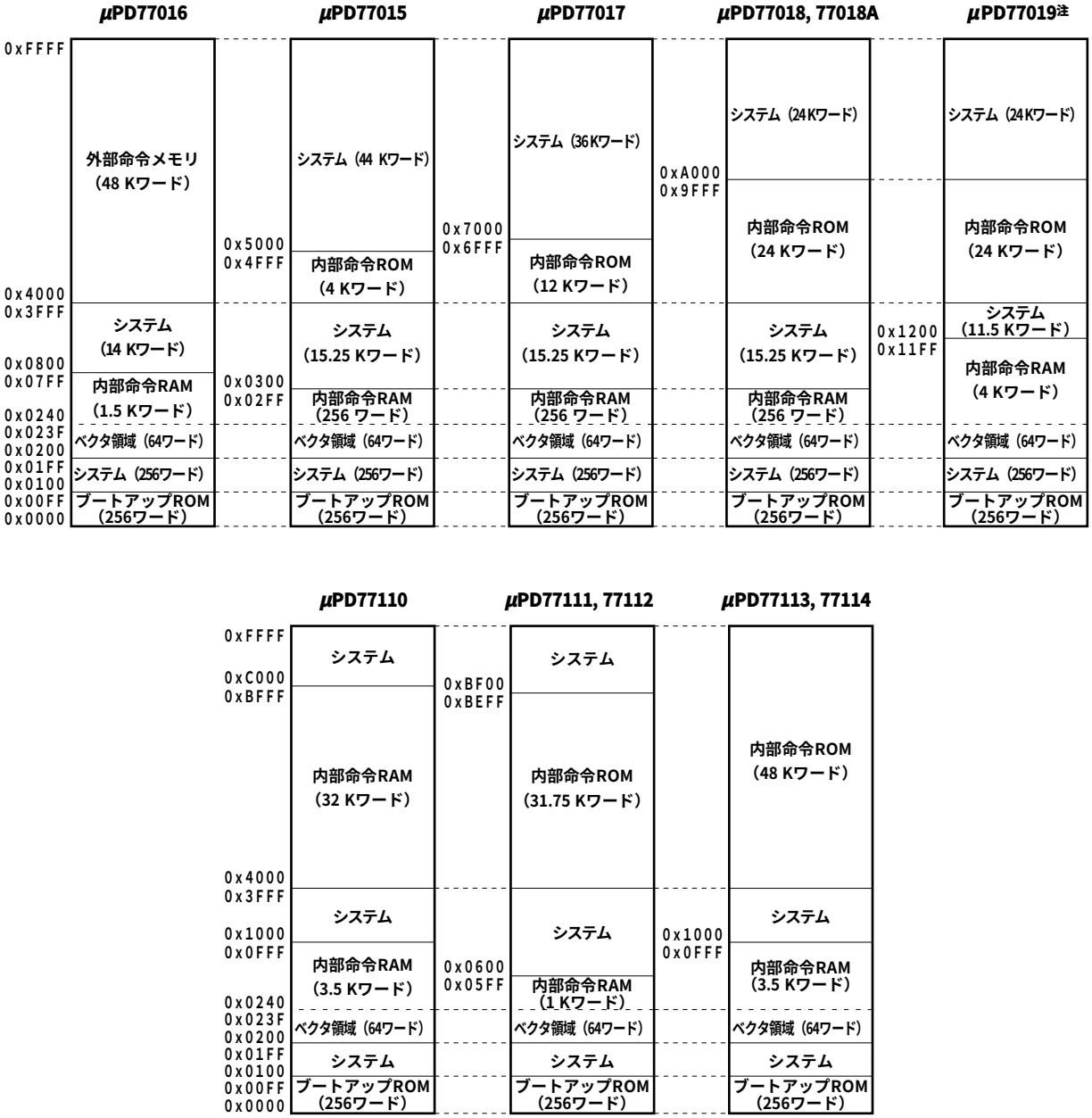
・インストラクション・メモリ空間
・Ｘメモリ空間
・Ｙメモリ空間

インストラクション・メモリ空間はμPD77016ファミリの命令（インストラクション）を格納するためのメモリ空間です。また，Ｘメモリ空間とＹメモリ空間は，データを格納するためのメモリ空間です。

それぞれのメモリ空間のメモリ・マップを，図２－１および図２－２に示します。

保守／廃止

★ **図2－1　インストラクション・メモリ空間**

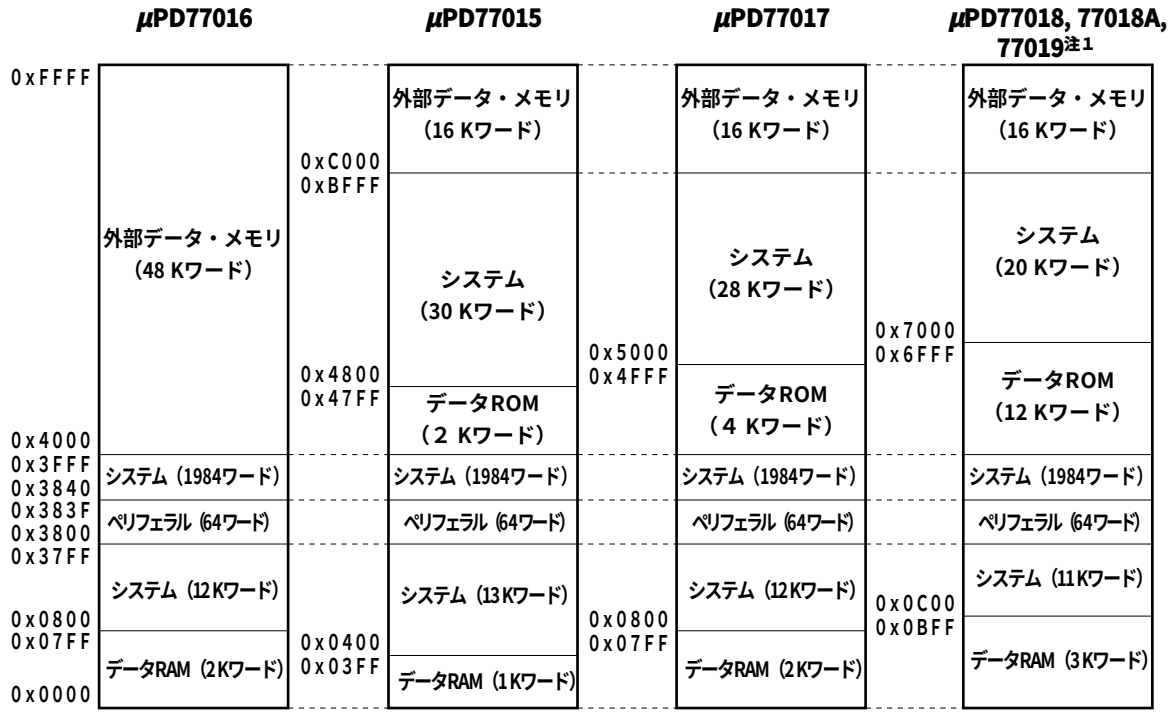


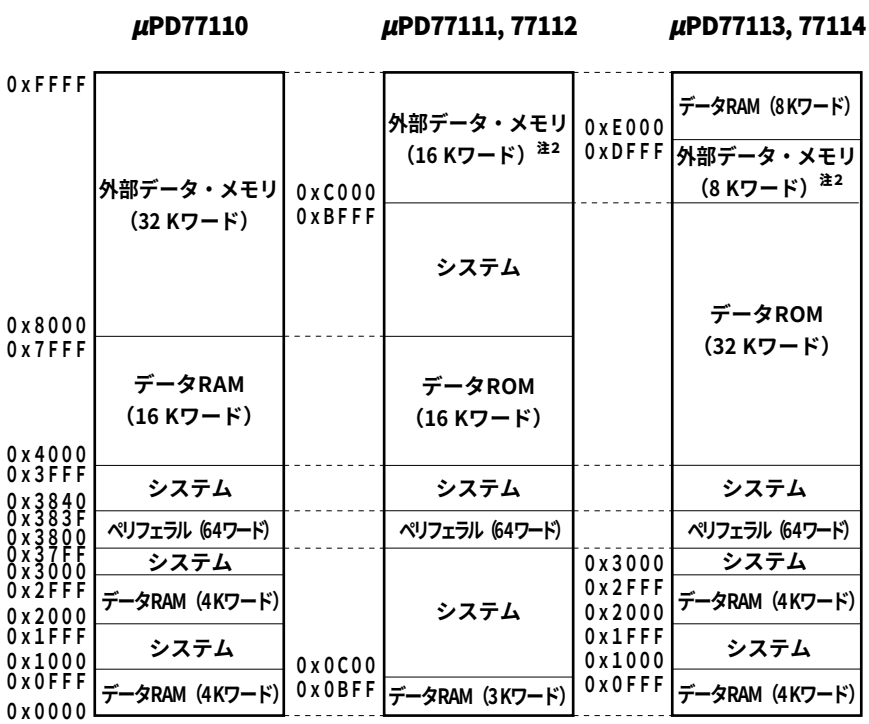
注意　システムとなっているアドレスには，プログラムやデータを置くことも，アクセスすることもできません。これらのアドレスをアクセスしたとき，μPD77016ファミリの正常な動作は保証されません。

注　μPD77019-013は，μPD77019の内部ROMを無効にした製品です。

保守／廃止

★ 図２－２　データ・メモリ空間





注意　システムとなっているアドレスには，プログラムやデータを置くことも，アクセスすることもできません。これらのアドレスをアクセスしたとき，μPD77016ファミリの正常な動作は保証されません。

注１．μPD77019-013は，μPD77019の内部ROMを無効にした製品です。

２．μPD77111, 77113では，システム領域になります。

保守／廃止

2.2　セグメント

アセンブリ言語には，セグメントという概念があります。セグメントとは，アセンブリ言語で記述されたプログラム中の命令，ROMに格納するデータ（初期値を格納することができます。），およびRAM上のデータ（初期値を格納することはできません。）を，明示的に区分するために使用します。アセンブラ・リンカはセグメントの種類や**2.2.2　セグメントの配置属性**で説明する配置属性をもとに，プログラムやデータをどのメモリ空間に配置するかを決定します。

2.2.1　セグメントの種類

セグメントには，次に示す３タイプ５種類のものがあります。

・インストラクション・メモリ用セグメント　：IMSEG
・データROM用セグメント　　Ｘメモリ用　：XROMSEG
　　Ｙメモリ用　：YROMSEG
・データRAM用セグメント　　Ｘメモリ用　：XRAMSEG
　　Ｙメモリ用　：YRAMSEG

インストラクション・メモリ用のセグメント（IMSEG）は，μPD77016ファミリのインストラクションを記述するためのセグメントです。インストラクション・メモリ用セグメントは，必ずインストラクション・メモリ空間に配置されます。

データROM用セグメントは，μPD77016ファミリの内部／外部データROMにデータ付き領域確保疑似命令（DW）などで定数値を設定するためのセグメントです。データROM用セグメントは，Ｘメモリ空間に配置されるXROMSEGとＹメモリ空間に配置されるYROMSEGの２種類があります。

データRAM用セグメントは，μPD77016ファミリの内部／外部データRAM上に領域確保疑似命令（DS）で領域を確保するためのセグメントです。データRAMセグメントは，Ｘメモリ空間に配置されるXRAMSEGと，Ｙメモリ空間に配置されるYRAMSEGの２種類があります。

保守／廃止

2.2.2　セグメントの配置属性

すべてのセグメントには，それぞれ配置属性があります。配置属性とは，アセンブラ・リンカがセグメントをメモリ空間に配置するときに参照する付加情報です。配置属性には次の２種類があります。

・アブソリュート／リロケータブル属性
・内部／外部メモリ属性

インストラクション・メモリ用セグメント，データROMセグメント，データRAM用セグメントは，これら両方の属性を持ちます。

（1）アブソリュート／リロケータブル属性

セグメントを配置するアドレスをソース・プログラム中で指定するのか，リンク時に決定するのかを指定します。

・アブソリュート属性

セグメントをメモリ領域に配置するとき，その先頭アドレスを，ソース・プログラム中に絶対番地で指定します。

・リロケータブル属性

セグメントを配置するアドレスをソース・プログラム中では指定しません。
リンク時にリンカが配置アドレスを決定します。

セグメントを配置する範囲については，**6.1.1　セグメント開始疑似命令**を参照してください。

（2）内／外部メモリ属性

インストラクション・メモリ，データROM，データRAM用セグメントを，μPD77016ファミリ内部のデータ・メモリに配置するのか，外部に拡張したデータ・メモリに配置するのかを指定します。

・内部メモリ属性

セグメントを，μPD77016ファミリ内部のデータ・メモリに配置します。

・外部メモリ属性

セグメントを，外部に拡張したデータ・メモリに配置します。

指定方法の詳細は，**6.1.1　セグメント開始疑似命令**を参照してください。

〔メ　モ〕

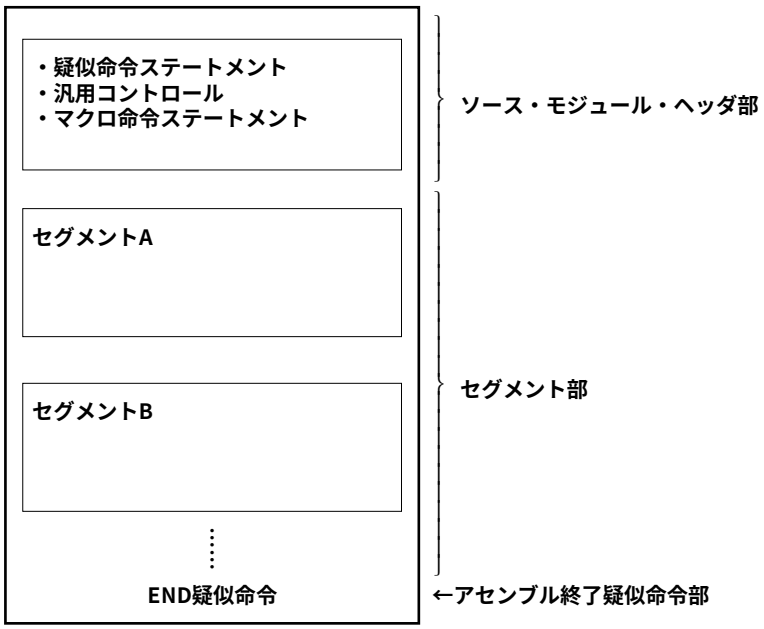
第３章　ソース・モジュール記述形式

３.１　ソース・モジュールの構成

μPD77016ファミリ・アセンブリ言語で記述した文を“ステートメント”といいます。ソース・モジュールは汎用コントロールおよびマクロ命令ステートメントを含むステートメント群で構成します。論理的にはセグメントに分けて記述します。セグメントの中または外に，アセンブラに対する疑似命令を記述し，適切なソース・モジュールを作成します。ソース・モジュール中にアセンブル終了疑似命令（END）が現れたら，そこでアセンブルは終了します。

図３－１にソース・モジュールの構成を示します。

図３－１　ソース・モジュールの構成



保守／廃止

３.１.１　ソース・モジュール・ヘッダ部

ソース・モジュール・ヘッダ部には次の命令が記述できます。

・疑似命令ステートメント
  - NAME疑似命令
  - シンボル定義疑似命令
  - PUBLIC/EXTRN疑似命令

・汎用コントロール

・マクロ命令ステートメント

３.１.２　セグメント部

セグメント開始疑似命令で定義されるセグメント・ブロックを記述します。すべてのセグメントにはソース・モジュール全体で固有のセグメントを割り当てる必要があります。セグメント部には次の命令を記述することができます。

・命令ステートメント

・疑似命令ステートメント（NAME, END疑似命令を除く）

・汎用コントロール

・マクロ命令ステートメント

各セグメントとそこに記述可能なステートメントを次に示します。

・インストラクション・メモリ用セグメント
  - アセンブリ命令ステートメント
  - 疑似命令ステートメント
    （セグメント開始疑似命令（IMSEG），
    シンボル定義疑似命令，
    ロケーション・カウンタ制御疑似命令，
    領域確保疑似命令（DW），
    PUBLIC/EXTRN疑似命令）
  - 汎用コントロール
  - マクロ命令ステートメント

保守／廃止

- データROM用セグメント
  - 疑似命令ステートメント（セグメント開始疑似命令（XROMSEG, YROMSEG），シンボル定義疑似命令，ロケーション・カウンタ制御疑似命令，領域確保疑似命令（DW, DWL），PUBLIC/EXTRN疑似命令）
  - 汎用コントロール
  - マクロ命令ステートメント
- データRAM用セグメント
  - 疑似命令ステートメント（セグメント開始疑似命令（XRAMSEG, YRAMSEG），シンボル定義疑似命令，ロケーション・カウンタ制御疑似命令，領域確保疑似命令（DS），PUBLIC/EXTRN疑似命令）
  - 汎用コントロール
  - マクロ命令ステートメント

3.1.3　アセンブル終了疑似命令部

ソース・モジュールの終わりには必ずEND疑似命令を記述します。ソース・モジュール・ファイルの途中にEND疑似命令があった場合，ワーニング・メッセージを一度出力し，それ以降の命令はアセンブルの対象としません。

保守／廃止

３.２　ステートメント

ステートメントにはアセンブリ命令ステートメント，疑似命令ステートメント，汎用コントロール，マクロ命令ステートメントがあります。

ステートメント１行には最大220文字（CRおよびLFは除く）まで記述できます。

３.２.１　アセンブリ命令ステートメントの構成

アセンブリ命令ステートメントは，μPD77016ファミリの命令を記述するためのステートメントであり，シンボル欄，ニモニック欄，コメント欄で構成されています。アセンブリ命令のターミナル・マークは“；”です。

[△][シンボル欄 [△]:[△]] ニモニック１欄 [△ニモニック２欄 [△ニモニック３欄]] [△];[[△] コメント欄] [△] ↵

また，ニモニック１欄とニモニック２欄のあとにそれぞれ↵を入れてもかまいません。

・記述例

① R2 = R2＋R0＊R1H
　R0L = ＊DP0＋＋　R1H = ＊DP4＃＃；

② R2 = R2＋R0＊R1H　R0L = ＊DP0＋＋
　R1H = ＊DP4＃＃；

③ R2 = R2＋R0＊R1H
　R0L = ＊DP0＋＋
　R1H = ＊DP4＃＃；

（１）シンボル欄

シンボル欄には必要に応じてシンボルを記述します。シンボルの詳細については，**４.５　シンボル**を参照してください。

シンボル欄とニモニック欄は，コロン“：”で区切り，コロンの前後には任意の空白（またはTAB）を記述することができます。

（２）ニモニック欄（ニモニック１欄-ニモニック３欄）

ニモニック欄にはμPD77016ファミリのニモニックを記述します。

ニモニックはインストラクション・セットの規則に従い，１-３個まで記述することができます。このとき，ニモニックどうしの区切りとして，１つ以上の空白，TABまたは改行を置きます。ただし改行で区切った場合，ニモニックどうしの間にコメントは記述できません。

（３）コメント欄

コメント欄を記述するときは，コメント欄の前にセミコロン“；”を記述するか，コメント欄の前後に“/＊”と“＊/”を記述します。詳細については**３.３　コメント**を参照してください。

保守／廃止

3.2.2　疑似命令ステートメント

疑似命令ステートメントは疑似命令を記述するためのステートメントで，シンボル欄，疑似命令欄，オペランド欄，コメント欄で構成されています。１ステートメントはLFで終了します。１つの疑似命令を複数行にわたって記述することはできません。

[△] [シンボル欄 [△] [:] [△]] 疑似命令欄 [△オペランド欄] [△] [;[[△]コメント欄] [△]] ↵

（1）シンボル欄

シンボルの詳細については，**4.5　シンボル**を参照してください。

（2）疑似命令欄

疑似命令欄には，疑似命令を１つだけ記述します。

疑似命令の詳細については，**第６章　疑似命令**を参照してください。

（3）オペランド欄

オペランド欄には，疑似命令欄に記述した命令に必要なオペランドを記述します。疑似命令には，オペランドを記述してはいけないもの，１個以上のオペランドを記述するものがあります。複数個のオペランドを記述する場合は，各オペランドをカンマ“ , ”で区切ります。

（4）コメント欄

コメント欄を記述する場合は，コメント欄の前に“ ; ”を記述するか，コメント欄の前後に“ /* ”と“ */ ”を記述します。詳細については**3.3　コメント**を参照してください。

保守／廃止

３.２.３　汎用コントロール

汎用コントロールはソース・モジュール・ファイル中の任意の位置に記述します。記述形式は次のように２通りあります。＄，＃の区別および汎用コントロールの詳細については，**第７章　汎用コントロール**を参照してください。

・リスト制御用式

[△]＄[△] 汎用コントロール欄 [△][;[[△] コメント欄][△]] ↵

・インクルード・ファイル指定，条件付きアセンブル制御，リテラル・ネーム指定用式

[△]＃[△] 汎用コントロール欄 [△][;[[△] コメント欄][△]] ↵

・＄，＃は空白またはタブを除いた最初に指定します。
・汎用コントロールは，LFで終了します。

（１）汎用コントロール欄

汎用コントロール欄には，汎用コントロールを１つだけ記述します。

（２）コメント欄

疑似命令ステートメントと同様にコメントを記述することができます。

コメント欄を記述する場合は，コメント欄の前にセミコロン“ ； ”を記述するか，コメント欄の前後に“ /＊ ”と“ ＊/ ”を記述します。詳細については**３.３　コメント**を参照してください。

３.２.４　マクロ命令ステートメント

マクロ命令ステートメントはマクロ定義／参照を記述するためのステートメントで，ソース・モジュール・ファイル中の任意の位置に記述します。マクロ命令ステートメントは，シンボル欄，マクロ命令欄，マクロ名欄，仮パラメータ・リスト欄，ローカル・リスト欄，マクロ・ボディ欄および実パラメータ・リスト欄で構成されています。シンボル記述形式は次のとおりです。

・マクロ定義形式

[△]%[△] マクロ命令欄 [▽]([▽] マクロ名欄 [▽][([▽] 仮パラメータ・リスト欄 [▽])[▽]])[▽]
[LOCAL▽ローカル・リスト欄 [▽]](マクロ・ボディ欄)[△][;[[△] コメント欄][△]] ↵

・マクロ参照形式

[△][シンボル欄 [△][:][△]][△]%[△] マクロ名欄 [[▽]([▽] 実パラメータ・リスト欄 [▽])]
[△][;[[△] コメント欄][△]] ↵

保守／廃止

（1）シンボル欄

シンボルの詳細については，4.5　シンボルを参照してください。

（2）マクロ命令欄，マクロ名欄，仮パラメータ・リスト欄，ローカル・リスト欄，マクロ・ボディ欄，実パラメータ・リスト欄

詳細については，第８章　マ ク ロを参照してください。

（3）コメント欄

コメント欄を記述する場合は，コメント欄の前にセミコロン“ ； ”を記述します。

保守／廃止

3.3　コメント

コメントの記述形式は，次の２通りです。

3.3.1　“；”（セミコロン）記述形式

“；”から↵までの文字列をコメント記述とみなします。ソース・モジュール・ファイルの任意の位置に記述することができますが，各命令ステートメントまたは，汎用コントロールが存在する行は，その命令の記述形式に従います。**3.2　ステートメント**を参照してください。

3.3.2　/＊　＊/記述形式

（1）“/＊”から“＊/”までの文字列を記述とみなします。
（2）コメント内には複数の↵を記述できます。
（3）コメント内の“/＊”はコメント開始記号とはみなされません（/＊　＊/のネスティングはできません）。
（4）“/＊　＊/”のコメント記述は，ソース・モジュール・ファイルの任意の位置に記述できます。各命令ステートメント内でも，空白，タブの記述可能な位置ならば，“/＊　＊/”のコメントを記述することができます。
（5）アセンブラは“/＊　＊/”のコメント記述を１個の空白として解釈します。
　これにより，“/＊　＊/”のコメント記述の外の左右の文字列は，区切られることになります。

例

```
E01 EQU/＊　COMMENT　＊/100　…　EQUと100を区切られた文字列として扱う
```

ただし，コメント内の↵は改行として認識するため，改行できない位置に記述された“/＊　＊/”コメント内に，↵を記述した場合，エラーになります。

例

```
E02 EQU/＊　COMMENT↵
　　＊/100　…　エラー
```

保守／廃止

第４章　言語要素

ソース・モジュールを記述するための，基本単位となる言語要素には，次の７種類があります。

- 定数
- 演算子
- シンボル
- ニモニック（アセンブリ命令）
- 疑似命令
- キーワード
- 区切子

これらの言語要素はすべて文字で記述します。**4.1　文字セット**にアセンブラに対して入力可能な文字の種類を，**4.2　定数**-**4.6　オペランド**に各言語要素の説明を示します。

4.1　文字セット

文字は使用目的により，次の２種類に分けられます。それ以外の文字を不当文字といいます。

- 言語文字
- 文字データ

4.1.1　言語文字

（1）目　　的

ソース・モジュール上で命令の記述（コメントおよび，文字データの定義以外）に使用します。

（2）言語文字の文字セット

言語文字の文字セットには，数字，英字，英字相当文字および特殊文字があります。

表４－１にそれらの分類を示します。

保守／廃止

表４－１　言語文字

名　称		文　字
数字		０　１　２　３　４　５　６　７　８　９
英字	大文字	Ａ　Ｂ　Ｃ　Ｄ　Ｅ　Ｆ　Ｇ　Ｈ　Ｉ　Ｊ　Ｋ　Ｌ　Ｍ Ｎ　Ｏ　Ｐ　Ｑ　Ｒ　Ｓ　Ｔ　Ｕ　Ｖ　Ｗ　Ｘ　Ｙ　Ｚ
	小文字	ａ　ｂ　ｃ　ｄ　ｅ　ｆ　ｇ　ｈ　ｉ　ｊ　ｋ　ｌ　ｍ ｎ　ｏ　ｐ　ｑ　ｒ　ｓ　ｔ　ｕ　ｖ　ｗ　ｘ　ｙ　ｚ
英字相当文字		？[注1]　＿[注2]
特殊文字	特殊１	．　，　：　；　＊　／　＋　－　＇　〈　〉　（　） ＄　＝　！　□[注3]　＆　｛　｝　～[注4]　｜[注5] ＾　＃　％　＠
	特殊２	ＣＲ[注6]
	特殊３	ＨＴ[注7]　ＬＦ[注8]

注１．シンボル名の先頭文字には使えません。

２．下線（0x5F）

３．空白（0x20）

４．ティルド（0x7E）

５．縦棒（0x7C）

６．復帰（0x0D）

７．タブ（0x09）

８．改行（0x0A）

（３）説　明

・予約語，キー・ワード，または基数表現文字（Ｂ，Ｘ）に記述された英小文字は，対応する英大文字（例：ａに対応するのはＡ）と同一文字として解釈されます。

たとえば，２進数を表す場合，“ 0b1011 ”と記述してもかまいません。なお，入力イメージはアセンブル・リスト上にそのまま出力されます。

・表４－１に示した文字以外は不当文字とみなし，エラー・メッセージを出力します。アセンブル・リストのソース・イメージ欄には“ ＾ ”に置き換えて表示します。

・予約語以外（ユーザ定義シンボル等）の文字を英小文字で記述した場合，アセンブラは英小文字と解釈し，英大文字，小文字の区別をします。ただし，アセンブラ・オプション“ －CA ”を指定した場合，英大文字，小文字の区別をしないで，英小文字を英大文字と解釈します。

・特殊文字の扱いを表４－２に示します。

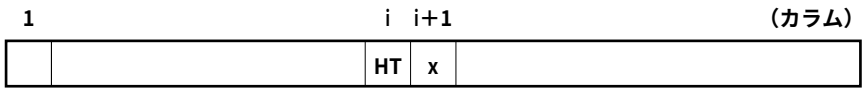
保守／廃止

表４－２　特殊文字の扱い（言語文字）

特殊文字	オブジェクト出力	リスト出力
HT	なし	８カラムの倍数に見合う数の空白[注1]
CR	なし	なし
LF	なし	0x0D0A[注2]

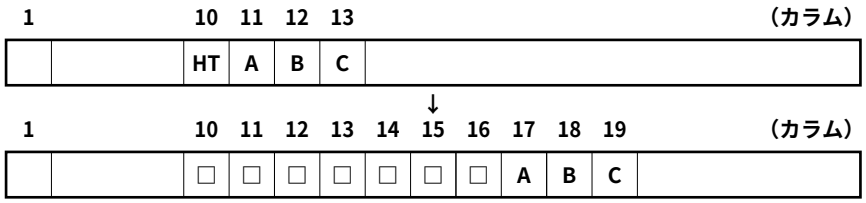
注１．ソース・ステートメント（**３.２　ステートメント**を参照）のカラム位置を移動させる文字です。

HTの次の文字を，現在のカラム位置から最も近い８の倍数カラム＋１のカラムに変更します。途中の空いたカラムには空白を入れます。



xのカラム＝［((i＋7)/8)＊8］＋1

［例］



Aのカラム位置＝［((10＋7)/8)＊8］＋1
＝　2＊8＋1＝17

２．行の区切りを示す文字です。

保守／廃止

４.１.２　文字データ

（１）目　　的

ソース・モジュール上で，コメントや文字データの定義（文字列定義）などに使用します。次の記述に使用します。

・文字列定数
・汎用コントロール部（TITLE）
・コメントの記述[注]

注　コメントにだけ漢字が使えます。

（２）文字データの文字セット

文字データの文字セットについては，**４.１.１（２）言語文字の文字セット**を参照してください。
言語文字セットには，次の文字を追加します。

・片仮名（0xA6-0xAF, 0xB1-0xDD）
・　”（0x22）
・　。（句点：0xA1）
・　」（0xA3）
・　・（中点：0xA5）
・　゛（濁点：0xDE）
・　￥（0x5C）
・　「（0xA2）
・　、（読点：0xA4）
・　ー（0xB0）
・　゜（半濁点：0xDF）
・８ビットJISコードで表される文字とシフトJISコードで表される文字。

コード値の解釈を次に示します。

① 0x00, 0x01-0x08, 0x0B, 0x0C, 0x0E-0x19, 0x1B-0x1F, 0x7F, 0x80, 0xA0, 0xFD-0xFFのコードが指定された場合，不当文字とします。
② 0x81-0x9F, 0xE0-0xFCのコードが指定された場合，次の８ビット・コードも取り込み，0x40-0x7E, 0x80-0xFCであればシフトJISコードで表される文字とみなし，それ以外であれば16ビット分不当文字であるとみなします。ただし，LFは特別に有効とし，最初の８ビット分のみ不当文字とみなします。

（３）説　　明

特殊文字の扱いを表４－３に示します。

表４－３　特殊文字の扱い（文字データ）

特殊文字	オブジェクト出力	リスト出力
HT	なし	８カラムの倍数に見合う数の空白[注1]
CR	なし	なし
LF	なし	0x0D0A[注2]
不当文字	なし	^

注１．**４.１.１　言語文字**を参照してください。
　２．行の区切りを示す文字です。

保守／廃止

4.2　定　　数

定数とは，固定した値を表すものであり，次の２種類があります。

・数値定数
・文字列定数

4.2.1　数値定数

数値定数は数値を表し，２進定数，８進定数，10進定数，16進定数，固定小数点数が記述できます。

（1）種類と形式

［種類］

・数値定数：｛２進定数，８進定数，10進定数，16進定数，固定小数点数｝

［形式］

・２進定数：｛0b[注]n，0b[注]n... n｝　（n = 0, 1）

０，１の２文字で構成し，数字列の先頭に0bを付加します。

［例］

0b11……10進で３の数値を表す。

・８進定数：｛0n，0n..... n｝　（n = 0, 1, 2, 3, 4, 5, 6, 7）

０，１，２，３，４，５，６，７の８文字で構成し，数字列の先頭に０（ゼロ）を付加します。

［例］

04……10進で４の数値を表す。
010……10進で８の数値を表す。

保守／廃止

・10進定数：{ n / n..... n } （n = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9）

０，１，２，３，４，５，６，７，８，９の10文字で構成されます。

数字列が１桁の場合は数字列の先頭に０を記述することができます。２桁以上の場合は数字列の先頭に０を記述することはできません。

・16進定数：{ 0x[注] / 0x[注]n..... n } （n = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F[注]）

０，１，２，３，４，５，６，７，８，９，Ａ，Ｂ，Ｃ，Ｄ，Ｅ，Ｆの16文字で構成し，数字列の先頭に0xを付加します。

[例]

0x13……10進で19の数値を表す。

0xFF……10進で255の数値を表す。

・固定小数点数については，**4.4.1　固定小数点型**を参照してください。

注　基数表現文字（ｂ，ｘ）および16進数の英字は小文字でも記述できます。アセンブル・リストには，小文字は小文字のまま出力します。

（2）評 価 値

記述された値を，評価値とします。

詳細については，**4.4　数　　値**を参照してください。

保守／廃止

４.２.２　文字列定数

８ビットJISコードとシフトJISコードを表します。

文字の列（DWL, SET, EQU疑似命令の場合の文字列の長さは１-４，それ以外は長さ１または２）を引用符“ ' ”で囲んで表します。

（１）形　　式

文字列定数：'文字〔...〕'…最高４文字まで

“ ' ”そのものを文字として指定する場合は，１つの“ ' ”に対して，“ '' ”と２文字連続して指定する必要があります。

[例]

''''　　：１個の“ ' ”

''''''　：２個の“ ' ”

文字については，**４.１.２　文字データ**を参照してください。

長さ０の文字列（''）は，ヌル文字（0x00）とします。

長さ３以上の文字列は，フェイタル・エラーを出力します。ただし，DWL, SET, EQU疑似命令のオペランドとして記述した場合は，長さ５以上の文字列のときフェイタル・エラーを出力します。

（２）評 価 値

（ａ）各文字を８ビットJISコード値で評価します。

（ｂ）文字列の長さは，１または２とします。ただし，DWL, SET, EQU疑似命令で使用する場合は，１から４までとします。

（ｃ）値は，コード値右詰め32ビット値とします。左側が空いている場合は，０を詰めます。例を次に示します。

[例]　文字列定数の評価値

通常の場合

文字列定数	評価値
'ab'	0x00006162
'A'	0x00000041
'_'注1	0x0000005F
' '注2	0x00000009
' '注3	0x0000000A

DWL, SET, EQU疑似命令の場合

文字列定数	評価値
'abcd'	0x61626364
'A'	0x00000041
'_'注1	0x0000005F
' '注2	0x00000009
' '注3	0x0000000A

注１．スキップしません

２．タブ

３．復帰

保守／廃止

4.3　演 算 子

μPD77016ファミリ・アセンブリ言語の式に記述できる演算子には，次の３種類があります。

・算術演算子
・論理演算子
・比較演算子

（1）演算子の種類

表４－４に演算子の種類を示します。

表４－４　演算子の種類

演算子の種類		意　　味
算術演算子	＋	符号の＋，または加算演算子を表す。
	－	符号の－，または加算演算子を表す。
	＊	乗算演算子を表す。
	／	除算演算子を表す。
	MOD	剰余演算子を表す。
論理演算子	AND	論理積を求める。
	OR	論理和を求める。
	XOR	排他的論理和を求める。
	NOT	論理否定（ビット反転）を求める。
比較演算子	EQ, =	左辺と右辺の値が等しいかを調べる。
	NE, ！=	左辺と右辺の値が等しくないかを調べる。
	LT, <	左辺が右辺の値より小さいかを調べる。
	LE, < =	左辺が右辺の値以下であるかを調べる。
	GT, >	左辺が右辺の値より大きいかを調べる。
	GE, > =	左辺が右辺の値以上であるかを調べる。

（2）演算子の優先順位

演算子には優先順位が決められています。式の中で使用する場合，優先順位の高い演算子から評価されます。

同一の優先順位を持つ演算子が式の中に複数あるときは，式の左側から演算します。

表４－５に演算子の優先順位を示します。なお，優先順位の高い方から１とします。

表４－５　演算子の優先順位

優先順位	演　算　子
1	（）
2	＋，－（単項演算子），NOT
3	＊，／，MOD
4	＋，－
5	EQ, =, NE, ！=, LT, <, LE, < =, GT, >, GE, > =
6	AND
7	OR, XOR

保守／廃止

（３）演算子の制御

リロケータブルなセグメント中で定義されたレーベルなど，アセンブル時に値の定まらないシンボル（リロケータブルな値）に対する演算は制限されています。たとえば，リロケータブルな値どうしの加算を行うことはできません。

リロケータブルな値とは，次のものを示します。

- 配置属性がリロケータブルであるセグメントで定義されたレーベル
- 配置属性がリロケータブルであるセグメントで定義されたレーベルを含む式を値とするSET, EQUで定義されたシンボル
- 配置属性がリロケータブルであるセグメントのロケーション・カウンタ（“ $ ”）

アブソリュートな値とは，次のものを示します。

- 配置属性がアブソリュートであるセグメントで定義されたレーベル
- 配置属性がアブソリュートであるセグメントで定義されたレーベルを含む式を値とするSET, EQUで定義されたシンボル
- 配置属性がアブソリュートであるセグメントのロケーション・カウンタ（“ $ ”）
- 定数

アブソリュート値はセグメント属性を持ちません。

どのような演算が可能なのか，また，演算結果のアブソリュート／リロケータブルの区別を表４－６に示します。

表４－６　セグメント属性における項と演算子の組み合わせ

演算子 ＼ セグメント属性	＋	－	＊，／	MOD	論理演算子	比較演算子
R，R	×	A[注1]	×	×	×	A[注1]
R，A	R	R[注2]	×	×	×	×
A，A	A	A	A	A	A	A

注１．同じセグメントに所属していれば演算可能（結果はアブソリュートな値）。

２．Ａ－Ｒは演算できません。

備考　Ｒ：リロケータブルな値。

Ａ：アブソリュートな値。

×：組み合わせることはできません。

保守／廃止

・単項演算子は項についてのみ演算できます。

一次式：｛符号なし定数／シンボル名｝

項　　：｛１次式／ロケーション・カウンタ／算術式｝

シンボルは，SET, EQU, レーベル名のみです。算術式については，**付録A　構 文 図**を参照してください。

・ただし，単項演算子の“＋”または“－”に符号なし定数がつづく場合は，全体を符号付き定数とみなします。例を次に示します。

[例]

－1.0は符号付き定数である。

・単項演算子の“＋”および“－”はアブソリュートなオペランドだけが記述できます。演算結果はアブソリュートな値となります。

・外部参照名を含む式の形式

｛外部参照名値±アブソリュートな式／アブソリュートな式＋外部参照名値｝

数値（整数型，固定小数点型）間の算術２項演算結果を表４－７に示します（演算子は，“／”，“＊”，“－”，“＋”の場合）。

表４－７　演算結果

数値の型 ＼ 数値の型	整数型[注]	固定小数点型
整数型[注]	整数型[注]	固定小数点型
固定小数点型	固定小数点型	固定小数点型

注　整数型は，文字列定数を含みます。

備考１．演算結果がオーバフローの場合は，フェイタル・エラーとし，演算結果を０（ゼロ）として処理します。

２．MOD演算子は，整数型と整数型のみの演算ができ，結果は整数型になります。

３．数値間の２項論理演算は，すべての数値を整数型として演算を行い，結果も整数型となります。

４．比較演算は，固定小数点型と整数型の演算も行うことが可能です。また，この評価値は，FALSEのとき０，TRUEのとき－１となります。

保守／廃止

4.4　数　　値

μPD77016ファミリ・アセンブリ言語で扱う数値の種類を次に示します。

- 数値
  - 整数型
  - 固定小数点型

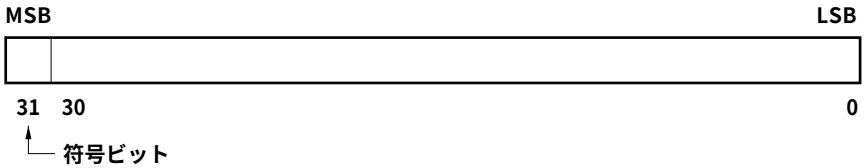
4.4.1　固定小数点型

（1）記述形式

－[０[・・・]]１[．０[・・・]] または [{±}][０[・・・]]０[．小数部]

小数部は10進数とみなします。32ビットの内部表現形式に変換したときに，小数部の桁数がμPD77016ファミリで扱うことのできる桁数よりも多い場合は，扱うことのできる桁数に切り捨てます。

（2）内部表現形式



（3）評価値の範囲

評価値の範囲と例について次に示します。

$-1.0 \leqq$ 評価値 $\leqq 1.0-2^{-32}$

［例］固定小数点型の内部表現形式

記述形式	内部表現形式
－0.5	11000000000000000000000000000000
－1.0	10000000000000000000000000000000

式中の固定小数点型を含む演算については，**4.3　演 算 子**を参照してください。

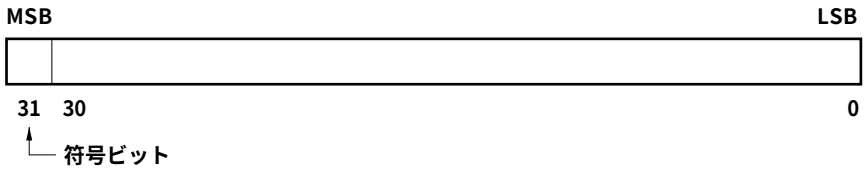
保守／廃止

4.4.2　整 数 型

（1）記述形式

空でない数字の列で表し，先頭に符号を付けることができます。記述した値は10進数とみなします。

（2）内部表現形式



（3）評価値の範囲

評価値の範囲と例について次に示します。

$-2^{31} \leqq$ 評価値 $\leqq 2^{31}-1$

[例] 整数型の内部表現形式

記述形式	内部表現形式
2	00000000000000000000000000000010
−0x0FFF	11111111111111111111000000000001 （0x0FFFの２の補数）
−2	11111111111111111111111111111110

式中の固定小数点型を含む演算については，**4.3　演 算 子**を参照してください。

上記の範囲を越える数値に対してはフェイタル・エラーを出力します。

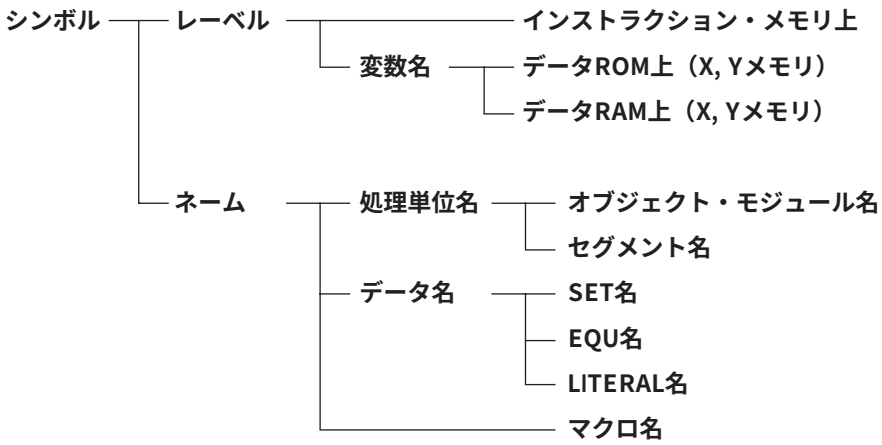
保守／廃止

４.５　シンボル

ソース・モジュール上で各種データ（数値，アドレス等）を表現するために，前述の定数以外にシンボルが使えます。

シンボルを一度定義すると，そのシンボルを使ったデータ参照ができます。

シンボルには，次のような種類があります。



１つのソース・モジュール・ファイル中で，同一名称のシンボルを２回以上定義すると，エラーとなります。

保守／廃止

（１）シンボルの定義

表４－８にシンボルが定義するステートメントを示します。

表４－８　ステートメント

シンボルの種類		定義するステートメント
レーベル	インストラクション・メモリ上	インストラクション・メモリ用セグメント内ステートメント
	データROM上（X，Yメモリ）	データROM用セグメント内の領域確保疑似命令ステートメント（DW, DWL）
	データRAM上（X，Yメモリ）	データRAM用セグメント内の領域確保疑似命令ステートメント（DS）
ネーム	オブジェクト・モジュール名	NAME疑似命令ステートメント
	セグメント名	セグメント開始疑似命令ステートメント
	SET名	SET疑似命令ステートメント
	EQU名	EQU疑似命令ステートメント
	LITERAL名	DEFINE汎用コントロール
	マクロ名	%DEFINE命令ステートメント

（２）ターミネータ

ターミネータには，次の２種類があります。

・レーベル：“ ： ”

・ネーム　：空白，TAB，コメント。

（３）シンボルの記述上の規則

・シンボルとして，予約語と同等のものを記述することはできません。

・英大小文字，数字，英字相当文字（？，＿）で構成します。
ただし，先頭文字として数字（０-９）と？は使用できません。

・レーベルはシンボル名の後ろにコロン（“ ： ”）を記述します。

・シンボルとして定義できる文字列の長さは１-８文字はまた１-31文字で，どちらにするかはアセンブラの－Ｓオプション（シンボル名長指定）で指定します。その字数を越えた文字は無視されます。

・同一シンボルを２回以上定義することはできません。ただし，SET疑似命令で定義したネームは，SET疑似命令で再定義できます。また，MODULE属性のシンボルは他属性のシンボルとして再定義できます。

・シンボルを英小文字で記述した場合，アセンブラは英小文字として解釈し，英大文字，小文字の区別をします。ただし，アセンブラ・オプション（－CA）を指定した場合，英大文字，小文字の区別をせず，英小文字を英大文字として解釈します。

保守／廃止

（４）シンボルの属性

シンボルの属性には次の９種類があります。

① NUMBER属性 …… 数値定数を割り付けたネーム，EXTRN疑似命令で定義されたシンボル。
② ADDRESS属性…… レーベルとして定義されたシンボル。また，そのシンボルをEQU/SET疑似命令でシンボル定義したネーム。
③ IM属性 …… インストラクション・メモリのセグメント名。
④ XROM属性 …… データROM（Ｘメモリ）のセグメント名。
⑤ YROM属性 …… データROM（Ｙメモリ）のセグメント名。
⑥ XRAM属性 …… データRAM（Ｘメモリ）のセグメント名。
⑦ YRAM属性 …… データRAM（Ｙメモリ）のセグメント名。
⑧ MODULE属性 …… NAME疑似命令によって定義されたモジュール名。指定されなかった場合は，ソース・モジュール・ファイル名のプライマリ・ネームから作成されます。
⑨ MACRO属性 …… %DEFINE命令によって定義されたマクロ名。

シンボルの属性と，シンボルの範囲を，表４－９に示します。

表４－９　シンボルの属性と範囲

属　　性	範　　囲
NUMBER属性	16進表現：0x00000000-0x0000FFFF
ADDRESS属性	10進表現：0-65535 16進表現：0x0000-0xFFFF（符号なし）
IM属性 XROM属性 YROM属性 XRAM属性 YRAM属性 MODULE属性 MACRO属性	値は持たない。

保守／廃止

4.6　オペランド

オペランドとして記述できるものを，次に示します。

・ニモニック（アセンブリ命令）
・キー・ワード
・式（シンボル，定数，ロケーション・カウンタ参照，またはそれらを演算子で結合したもの）

4.6.1　キー・ワード

キー・ワードはセグメント開始疑似命令のAT, 汎用コントロール（短縮型を含む）です。キー・ワードを他の目的（シンボルなど）に用いることはできません。

4.6.2　式

μPD77016ファミリ・アセンブリ言語で式と呼ぶものを，次に示します。

・シンボル（レーベル，SET, EQUで定義したもの）
・定数
・ロケーション・カウンタ参照記号（“ $ ”で表す。記述セグメントに対するロケーション・カウンタの値をもつ。）
・シンボル，定数，ロケーション・カウンタに演算子を付加したもの。または，演算子で結合したもの。演算子については，**4.3　演 算 子**を参照してください。
・外部参照名を含む式の形式：具体的には，

{外部参照名値±アブソリュートな式
 アブソリュートな式＋外部参照名値}

式に含まれるオペランドによってアブソリュートな式とリロケータブルな式を区別します（**4.3　演 算 子**を参照してください）。

[例]

3/5＋2
SYM＋4（SYMはシンボル）

保守／廃止

第５章　アセンブリ命令

５.１　アセンブリ命令セット

μPD77016ファミリの命令は32ビットで構成されています。
複数の処理を同一インストラクションに記述できます。また，条件付加できる命令があります。

５.１.１　命令セットの種類

命令セットは次の９種類に分けられます。

（１）３項演算命令

乗累算器での演算を指定する命令です。演算対象（入力）は汎用レジスタから３レジスタを，任意に指定できます。出力先は，入力に指定した汎用レジスタの１レジスタです。

（２）２項演算命令

乗累算器，ALUまたはシフタでの演算を指定する命令です。演算対象（入力）は汎用レジスタから２レジスタを任意に指定できます。また，汎用レジスタの代わりにイミーディエト値を１入力に指定できる命令もあります。出力先は汎用レジスタから１レジスタを任意に指定できます。

（３）単項演算命令

ALUでの演算を指定する命令です。演算対象（入力）は汎用レジスタから１レジスタを，任意に指定できます。出力先は汎用レジスタから１レジスタを，任意に指定できます。

演算対象の汎用レジスタが出力先となるクリア命令，出力先に指定したレジスタも演算対象（入力）とする累加算命令，累減算命令および除算命令も単項演算として扱います。

（４）ロード／ストア命令

メモリと汎用レジスタ間の16ビット・データ転送を指定する命令です。同時に演算命令を指定できる並列ロード／ストア，ロード／ストアの対象となる汎用レジスタの範囲を指定できる部分ロード／ストア，命令でアドレスを指定するダイレクト・アドレシング・ロード／ストア，および命令でモディファイ値を指定する即値インデクス・ロード／ストアなどがあります。

転送対象（入出力）は汎用レジスタから，任意に指定できます。

（５）レジスタ間転送命令

汎用レジスタとほかのレジスタ間の転送を指定する命令です。

（６）即値設定命令

汎用レジスタとアドレス演算ユニットの各レジスタにイミーディエト値を設定する命令です。

（７）分岐命令

プログラムの分岐を指定する命令です。ジャンプ，サブルーチン・コールおよびリターン命令があります。

（８）ハードウエア・ループ命令

命令の繰り返し実行を指定する命令です。１命令を繰り返すリピート命令と，２命令以上繰り返すループ命令があります。

（９）制御命令

プログラム制御を指定する命令です。

保守／廃止

5.1.2　ニモニック

（1）ニモニックの種類

ニモニックの種類を次に示します。

① ３項演算命令
② ２項演算命令（イミーディエト値を使用しない）
③ ２項演算命令（イミーディエト値を使用する）
④ 単項演算命令
⑤ ロード／ストア命令（並列ロード／ストア）
⑥ 〃 （部分ロード／ストア）
⑦ 〃 （そのほかのロード／ストア）
⑧ レジスタ間転送命令
⑨ 即値設定命令
⑩ 分岐命令
⑪ ハードウエア・ループ命令（リピート）
⑫ 〃 （ループ）
⑬ 制御命令（条件命令）
⑭ 〃 （条件命令以外）

（2）組み合わせ可能な命令セット

１ステートメントで組み合わせ可能な命令セットを次に示します。

ただし，命令セット24は１ステートメントに記述可能な組み合わせではなく，リピート命令と次に記述可能な命令との組み合わせです。

命令セット１ … ①
命令セット２ … ②
命令セット３ … ③
命令セット４ … ④
命令セット５ … ⑤
命令セット６ … ⑥
命令セット７ … ⑦
命令セット８ … ⑧
命令セット９ … ⑨
命令セット10 … ⑩
命令セット11 … ⑪
命令セット12 … ⑫

保守／廃止

命令セット13　…　⑬
命令セット14　…　⑭
命令セット15　…　①＋⑤（ＸメモリまたはＹメモリ）
命令セット16　…　①＋⑤（Ｘメモリ）＋⑤（Ｙメモリ）
命令セット17　…　②＋⑤（ＸメモリまたはＹメモリ）
命令セット18　…　②＋⑤（Ｘメモリ）＋⑤（Ｙメモリ）
命令セット19　…　④＋⑤（ＸメモリまたはＹメモリ）
命令セット20　…　④＋⑤（Ｘメモリ）＋⑤（Ｙメモリ）
命令セット21　…　⑬＋④
命令セット22　…　⑬＋⑧
命令セット23　…　⑬＋⑩
命令セット24　…　⑪
　　＋①，②，③，④，⑤，⑥，⑦，⑧，⑨，⑬，⑭のいずれか

（３）ニモニックの記述制限

各命令セットの記述には，次のような共通の制限があります。

・各ニモニックは，算術子（=，±等）の前後に任意の空白，またはTABを記述できます。
・演算命令（３項演算，２項演算，単項演算）で，（）を使って演算に優先順位をつけることはできません。
・イミーディエト値としてシンボル，式も記述できます。

保守／廃止

５.２　命令セットの詳細

ニモニックに記述できる数値（imm, addr, count）の表現はすべて16進数で示していますが，記述するときは，２進数，８進数，10進数，16進数のいずれでも記述できます。

また，データ・ポインタのモディファイは，メモリ・アクセス後に行います。結果は直後の命令から有効になります。データ・ポインタのモディファイのみはできません。記述例とオペレーションについて，以下に示します。

記述例	オペレーション
DPn	何も行いません（DPnの値を変化させません）。
DPn＋＋	DPn←DPn＋１
DPn－－	DPn←DPn－１
DPn＃＃	DPn←DPn＋DNn （DP0-DP7に対応するDN0-DN7の値を加算します。） 例：DP0←DP0＋DN0
DPn%%	（n = 0-3）DPn =（（DP_L＋DNn）mod（DMX＋１））＋DP_H
	（n = 4-7）DPn =（（DP_L＋DNn）mod（DMY＋１））＋DP_H
！DPn＃＃	DPnをビット・リバース後メモリ・アクセスします。 メモリ・アクセス後　DPn←DPn＋DNn
DPn＃＃ imm	DPn←DPn＋imm

保守／廃止

５.２.１　３項演算命令　（命令セット１）

[記述方法]

命令名称	ニモニック	オペランドに記述できるもの
マルチプライ・アド	ro = ro＋rh＊rh'	rh, rh' = {R0H-R7H} ro = {R0-R7}
マルチプライ・サブ	ro = ro－rh＊rh'	rh, rh' = {R0H-R7H} ro = {R0-R7}
サイン・アンサイン・マルチプライ・アド	ro = ro＋rh＊rl （rlは正の整数フォーマット）	rh = {R0H-R7H} rl = {R0L-R7L} ro = {R0-R7}
アンサイン・アンサイン・マルチプライ・アド	ro = ro＋rl＊rl' （rl, rl'は正の整数フォーマット）	rl, rl' = {R0L-R7L} ro = {R0-R7}
１ビット・シフト・マルチプライ・アド	ro =（ro＞＞１）＋rh＊rh'	rh, rh' = {R0H-R7H} ro = {R0-R7}
16ビット・シフト・マルチプライ・アド	ro =（ro＞＞16）＋rh＊rh'	rh, rh' = {R0H-R7H} ro = {R0-R7}

[記述上の注意]

各ニモニックのro, rh, rlなどの記述順序は**５.２.１　３項演算命令**の表のとおりで，入れ替えることはできません。ただし，サイン・アンサイン・マルチプライ・アドにかぎり，被乗数と乗数の入れ替えができます。

例：

c＝c＋a＊b …… 正しい

aとbを入れ替えたとき

c＝c＋b＊a …… 正しい

bとcを入れ替えたとき

c＝b＋a＊c …… 誤り

a＊bとcを入れ替えたとき

c＝a＊b＋c …… 誤り

保守／廃止

５.２.２　２項演算命令（イミーディエト値を使用しない）　（命令セット２）

[記述方法]

命令名称	ニモニック	オペランドに記述できるもの
マルチプライ	ro = rh＊rh'	rh, rh' = {R0H-R7H} ro = {R0-R7}
アド	ro'' = ro＋ro'	ro, ro', ro'' = {R0-R7}
サブ	ro'' = ro－ro'	ro, ro', ro'' = {R0-R7}
算術右シフト	ro' = ro SRA rl	ro, ro' = {R0-R7} rl = {R0L-R7L}
論理右シフト	ro' = ro SRL rl	ro, ro' = {R0-R7} rl = {R0L-R7L}
論理左シフト	ro' = ro SLL rl	ro, ro' = {R0-R7} rl = {R0L-R7L}
アンド	ro'' = ro & ro'	ro, ro', ro'' = {R0-R7}
オア	ro'' = ro ｜ ro'	ro, ro', ro'' = {R0-R7}
イクスクルーシブ・オア	ro'' = ro ^ ro'	ro, ro', ro'' = {R0-R7}
レスザン	ro'' = LT（ro, ro'）	ro, ro', ro'' = {R0-R7}

[記述上の注意]

並列ロード／ストア命令との同時記述が可能です。同時記述については，**５.２.16　２項演算（イミーディエト値を使用しない）＋並列ロード／ストア命令**を参照してください。

保守／廃止

５.２.３　２項演算命令（イミーディエト値を使用する）　（命令セット３）

[記述方法]

命令名称	ニモニック	オペランドに記述できるもの
イミーディエト算術右シフト	ro' = ro SRA imm	ro, ro' = {R0-R7} imm = {0-0x3F}
イミーディエト論理右シフト	ro' = ro SRL imm	ro, ro' = {R0-R7} imm = {0-0x3F}
イミーディエト論理左シフト	ro' = ro SLL imm	ro, ro' = {R0-R7} imm = {0-0x3F}
イミーディエト・アンド	ro' = ro & imm	ro, ro' = {R0-R7} imm = {0-0xFFFF}
イミーディエト・オア	ro' = ro ｜ imm	ro, ro' = {R0-R7} imm = {0-0xFFFF}
イミーディエト・イクスクルーシブ・オア	ro' = ro ＾ imm	ro, ro' = {R0-R7} imm = {0-0xFFFF}
イミーディエト・アド	ro' = ro＋imm	ro, ro' = {R0-R7} imm = {0, 2-0xFFFF}
イミーディエト・サブ	ro' = ro－imm	ro, ro' = {R0-R7} imm = {0, 2-0xFFFF}

[記述上の注意]

これらの命令は，並列ロード／ストア命令との同時記述はできません。

シフト命令のイミーディエト値に負の数値，小数点数を記述した場合，フェイタル・エラーを出力します。

保守／廃止

５.２.４　単項演算命令　（命令セット４）

[記述方法]

命令名称	ニモニック	オペランドに記述できるもの
クリア	CLR（ro）	ro = {R0-R7}
インクリメント	ro' = ro＋1	ro, ro' = {R0-R7}
デクリメント	ro' = ro－1	ro, ro' = {R0-R7}
絶対値	ro' = ABS（ro）	ro, ro' = {R0-R7}
１の補数	ro' = ~ro	ro, ro' = {R0-R7}
２の補数	ro' = －ro	ro, ro' = {R0-R7}
クリップ	ro' = CLIP（ro）	ro, ro' = {R0-R7}
丸め	ro' = ROUND（ro）	ro, ro' = {R0-R7}
指数	ro' = EXP（ro）	ro, ro' = {R0-R7}
代入	ro' = ro	ro, ro' = {R0-R7}
累加算	ro'＋ = ro	ro, ro' = {R0-R7}
累減算	ro'－ = ro	ro, ro' = {R0-R7}
除算	ro'/ = ro	ro, ro' = {R0-R7}

[記述上の注意]

特になし

保守／廃止

５.２.５　ロード／ストア命令（並列ロード／ストア）　（命令セット５）

[記述方法]

命令名称	ニモニック	オペランドに記述できるもの
並列ロード／ストア	ro = ＊dpx_mod　ro' = ＊dpy_mod	ro, ro'= {R0-R7} ,
	ro = ＊dpx_mod　＊dpy_mod = rh	rh, rh'= {R0H-R7H} ,
	＊dpx_mod = rh　ro = ＊dpy_mod	dpx= {DP0-DP3} ,
	＊dpx_mod = rh　＊dpy_mod = rh'	dpy= {DP4-DP7}

[記述上の注意]

（１）転送後，modで指定されたモディファイを行います。

（２）Xメモリ転送，Yメモリ転送の記述順序はどちらからでもかまいません。

（３）Xメモリ転送，Yメモリ転送どちらか１つのみの記述もできます。

（４）１ステートメント中にXメモリ転送同士あるいは，Yメモリ転送同士を記述することはできません。

例：

R2 = R3＋R4　R0 = ＊DP1　＊DP0 = R1；…… フェイタル・エラーを出力します。

~~~~はXメモリへのアクセスです。

（５）Xメモリからロードする汎用レジスタと，Yメモリからロードする汎用レジスタが重複するような記述はできません。

例：

R2 = R3＋R4　R0 = ＊DP1　R0 = ＊DP4；…… フェイタル・エラーを出力します。

（６）演算命令（３項演算，２項演算，単項演算）の同時記述がない場合，部分ロード／ストアの命令コードを出力します。つまり，この命令セットだけの命令コードを出力することはありません。

（７）レジスタ間転送命令および，即値設定命令で設定したデータ・ポインタ・レジスタを直後の並列ロード／ストア命令で指定した場合，フェイタル・エラーを出力します。

また，セグメントの先頭に並列ロード／ストア命令を記述した場合は，ワーニング・エラーを出力します。
~~~~

保守／廃止

例：

```
SEG1  IMSEG
      R2 = R3＋R4   R0 = ＊DP0   R5 = ＊DP5；  …… ワーニング・エラーを出力
                    ⋮                          します。
      DP0 = 0x0100；
      R2 = R3＋R4   R0 = ＊DP0   R5 = ＊DP5；  …… フェイタル・エラーを出力
                    ⋮                          します。
      IF（R0 = = 0）  DP0 = R1L；
      R2 = R3＋R4   R0 = ＊DP0   R5 = ＊DP5；  …… フェイタル・エラーを出力
                    ⋮                          します。
      LOOP 10 ｛
            R2 = R3＋R4   R0 = ＊DP0   R5 = ＊DP5；…… フェイタル・エラーを出力
                    ⋮                          します。
            DP0 = 0x0100；
      ｝ ；
      R2 = R3＋R4   R0 = ＊DP0   R5 = ＊DP5；  …… フェイタル・エラーを出力
                    ⋮                          します。
```

保守／廃止

５.２.６　ロード／ストア命令（部分ロード／ストア）　（命令セット６）

[記述方法]

命令名称	ニモニック	オペランドに記述できるもの
部分ロード／ストア	dest = ＊dpx_mod　dest' = ＊dpy_mod	dest, dest' = {R0-R7, R0EH-R7EH, R0E-R7E, R0H-R7H, R0L-R7L}, source, source' = {R0E-R7E, R0H-R7H, R0L-R7L}, dpx = DP0-DP3, dpy = DP4-DP7
	dest = ＊dpx_mod　＊dpy_mod = source	
	＊dpx_mod = source　dest = ＊dpy_mod	
	＊dpx_mod = source　＊dpy_mod = source'	

[記述上の注意]

（１）転送後，modで指定されたモディファイを行います。

（２）Xメモリ転送，Yメモリ転送の記述順序はどちらからでもかまいません。

（３）Xメモリ転送，Yメモリ転送どちらか１つのみの記述もできます。

（４）１ステートメント中にXメモリ転送同士あるいは，Yメモリ転送同士を記述することはできません。

（５）Xメモリ，Yメモリからロードする汎用レジスタが同じでビットが重複するような組み合わせの記述はできません（R0とR0EH, R0E, R0H, R0Lは同時記述できません。R1-R7についても同様です）。

例：

R0 = ＊DP1　R0 = ＊DP4；
R0H = ＊DP1　R0 = ＊DP4；
……どちらの場合もフェイタル・エラーを出力します。

（６）レジスタ間転送命令および，即値設定命令で設定したデータ・ポインタ・レジスタを直後の部分ロード／ストア命令で指定した場合，フェイタル・エラーを出力します。

また，セグメントの先頭に部分ロード／ストア命令を記述した場合は，ワーニング・エラーを出力します。

例：

```
SEG1  IMSEG
      R0 = ＊DP0   R5 = ＊DP5；  ……  ワーニング・エラーを出力します。
                  ⋮
      DP0 = 0x0100；
      R0 = ＊DP0   R5 = ＊DP5；  ……  フェイタル・エラーを出力します。
                  ⋮
      IF（R0 = = 0）   DP0 = R1L；
      R0 = ＊DP0   R5 = ＊DP5；  ……  フェイタル・エラーを出力します。
                  ⋮
      LOOP 10 ｛
            R0 = ＊DP0   R5 = ＊DP5；  ……  フェイタル・エラーを出力します。
                  ⋮
            DP0 = 0x0100；
      ｝ ；
      R0 = ＊DP0   R5 = ＊DP5；  ……  フェイタル・エラーを出力します。
                  ⋮
```

保守／廃止

５.２.７　ロード／ストア命令（その他のロード／ストア）　（命令セット７）

[記述方法]

命令名称	ニモニック	オペランドに記述できるもの
ダイレクト・アドレシング・ロード／ストア	dest = ＊addr	dest = ｛R0-R7, R0EH-R7EH, R0E-R7E, R0H-R7H, R0L-R7L｝, source = ｛R0E-R7E, R0H-R7H, R0L-R7L｝, addr= ｛0：X-0xFFFF：Xメモリ 0：Y-0xFFFF：Yメモリ｝
	＊addr = source	
即値インデクス・ロード／ストア	dest = ＊dp_imm	dest = ｛R0-R7, R0EH-R7EH, R0E-R7E, R0H-R7H, R0L-R7L｝, source = ｛R0E-R7E, R0H-R7H, R0L-R7L｝, dp = ｛DP0-DP7｝
	＊dp_imm = source	

[記述上の注意]

（１）レジスタ間転送命令および，即値設定命令で設定したデータ・ポインタ・レジスタを直後の即値インデクス・ロード／ストア命令で指定した場合，フェイタル・エラーを出力します。

　また，セグメントの先頭に即値インデクス・ロード／ストア命令を記述した場合は，ワーニング・エラーを出力します。

例：

```
SEG1  IMSEG
      R0 = ＊DP0＃＃1；  …… ワーニング・エラーを出力します。
                ⋮
      DP0 = 0x0100；
      R0 = ＊DP0＃＃1；  …… フェイタル・エラーを出力します。
                ⋮
```

（２）ダイレクト・アドレシング・ロード／ストア命令のaddrに小数点数を記述した場合，フェイタル・エラーを出力します。

保守／廃止

５.２.８　レジスタ間転送命令　（命令セット８）

[記述方法]

命令名称	ニモニック	オペランドに記述できるもの
レジスタ間転送	dest = rl	dest, source： 汎用レジスタ以外のすべてのレジスタから選択 rl = {R0L-R7L}
	rl = source	

[記述上の注意]

（１）destに指定したデータ・ポインタ・レジスタを直後のロード／ストア命令でポインタに指定した場合，フェイタル・エラーを出力します。

具体的な例については，**５.２.５　ロード／ストア命令（並列ロード／ストア）-５.２.７　ロード／ストア命令（その他のロード／ストア）**を参照してください。

また，セグメントの最後にレジスタ間転送命令を記述した場合は，ワーニング・エラーを出力します。

例：

```
        ⋮
DP0 = R0L；……　ワーニング・エラーを出力します。
END
```

（２）dest, sourceにスタック（STK），スタック・ポインタ（SP）を指定した場合，直後にRET, RETIを記述しないでください。

保守／廃止

５.２.９　即値設定命令　（命令セット９）

[記述方法]

命令名称	ニモニック	オペランドに記述できるもの
即値設定	rl = imm	rl = {R0L-R7L}， dp = {DP0-DP7}， dn = {DN0-DN7}， dm = {DMX, DMY}， imm = {0-0xFFFF}
	dp = imm	
	dn = imm	
	dm = imm	

[記述上の注意]

destに指定したデータ・ポインタ・レジスタを直後のロード／ストア命令でデータ・ポインタとして使用した場合，フェイタル・エラーを出力します。

具体的な例については，**５.２.５　ロード／ストア命令（並列ロード／ストア）-５.２.７　ロード／ストア命令（その他のロード／ストア）**を参照してください。

また，セグメントの最後に即値設定命令を記述した場合は，ワーニング・エラーを出力します。

例：

```
        ⋮
DP0 = 0x0100；  …… ワーニング・エラーを出力します。
END
```

保守／廃止

５.２.10　分岐命令　（命令セット10）

[記述方法]

命令名称	ニモニック	オペランドに記述できるもの
ジャンプ[注]	JMP imm	imm = {0-0xFFFF}
レジスタ間接ジャンプ	JMP dp	dp = {DP0-DP7}
サブルーチン・コール[注]	CALL imm	imm = {0-0xFFFF}
レジスタ間接サブルーチン・コール	CALL dp	dp = {DP0-DP7}
リターン	RET	
割り込みリターン	RETI	

注　ジャンプ命令とサブルーチン・コール命令は，実際には±32 Kワードの範囲に分岐する相対ジャンプ命令です。しかし，相対アドレスの計算はアセンブラとリンカが行うため，ユーザが相対ジャンプを意識する必要はありません。ソース・プログラムには絶対アドレス（実際の分岐先アドレス）を記述してください。もし相対アドレス形式で記述したいときは，以下のように分岐命令のオペランドの先頭に＄を記述します。

JMP ＄＋2　…　２つ先のアドレスへジャンプ
JMP ＄－3　…　３つ手前のアドレスへジャンプ

[記述上の注意]

（１）ジャンプ，サブルーチン・コール命令のイミーディエト値に小数点数を記述した場合，フェイタル・エラーを出力します。

（２）IF条件付きサブルーチン・コール命令の次にリターン命令（IF条件なし）または割り込みリターン命令（IF条件なし）を記述するとフェイタル・エラーを出力します。

例：（ａ）if（r0 ! = 0）call dp0；
　　ret；　……フェイタル・エラーを出力します。
（ｂ）if（r0 ! = 0）call 0x200；
　　reti；　……フェイタル・エラーを出力します。

（３）リターン命令，割り込みリターン命令の直前に，スタック（STK），スタック・ポインタ（SP）をdest, sourceとしたレジスタ間転送命令を記述しないでください。

例：

　　⋮
R0L = STK；……　フェイタル・エラーを出力します。
RET；

保守／廃止

５.２.11　ハードウエア・ループ命令（リピート）　（命令セット11）

[記述方法]

命令名称	ニモニック	オペランドに記述できるもの
リピート	REP count	count = {1-0x7FFF, R0L-R7L}

[記述上の注意]

（１）countに0および0x8000以上を指定した場合，フェイタル・エラーを出力します。

（２）リピート対象命令に分岐命令，リピート命令，ループ命令，ループ終端の命令，ループ・ポップ命令，HALT命令およびSTOP命令を記述した場合，フェイタル・エラーを出力します。

例：

次の（ａ）-（ｄ）のような記述をした場合，フェイタル・エラーを出力します。

```
（ａ）REP count；
      REP count；
（ｂ）REP count；
      LOOP count ｛
                 ⋮
（ｃ）LOOP count ｛
                 ⋮
                   REP count；
                   リピート対象命令
                   ｝ ；
（ｄ）REP count；
      HALT；
```

（３）リピート命令のカウンタ値に負の数値，または小数点数を記述した場合，フェイタル・エラーを出力します。

保守／廃止

５.２.12　ハードウエア・ループ命令（ループ）　（命令セット12）

[記述方法]

命令名称	ニモニック	オペランドに記述できるもの
ループ	LOOP count （２桁以上の命令）	count = {1-0x7FFF, R0L-R7L}

[記述上の注意]

（１）countに0および0x8000以上を指定した場合，フェイタル・エラーを出力します。

（２）ループ開始命令からループ終了命令までの間には，２行-255行のアセンブリ命令ステートメントが記述できます。

（３）LOOP命令の“ {”は，LOOP命令記述行の最後か，改行後に記述できます。

例：

```
        LOOP count    {
または
        LOOP count
         {
```

（４）LOOP命令の“ ；”は次の例のように記述してください。

例１.

```
    LOOP   count  {
                ⋮
    }  ;   ←この位置に必ず；を記述してください。
```

２.

```
    LOOP count    ;   ←この位置に；を記述してはいけません。
     {            ;   ←この位置に；を記述してもかまいません。
```

３.

```
    LOOP count    {;   ←この位置に；を記述してもかまいません。
```

（５）ループ開始アドレスからループ終端アドレスまでの命令ステートメントが０または１行の場合，または256行以上の場合，フェイタル・エラーを出力します。

例：

```
    LOOP count    {
                “１”命令
               }  ;
```

保守／廃止

（６）ソース・モジュール上のループ・スタックのネスト管理は最大４レベルまでとします。４レベルを越えた場合，ワーニング・エラーを出力します。

★ （７）ループ先頭にループ・カウンタからの転送命令を記述することはできません。

（８）ループ終端３命令以内に記述できない命令を次に示します。

分岐命令，リピート命令，リピート対象命令（リピート命令とのセット），ループ命令，ほかのループ命令の終端命令，ループ・ポップ命令，HALT命令，STOP命令およびループ・カウンタからの転送命令

次のような記述をした場合は，フェイタル・エラーとなります。

例：分岐命令，リピート命令，リピート対象命令（リピート命令とのセット）

```
LOOP count  {                          LOOP count  {
            ⋮                                      ⋮
            REP count                              REP count
            } ；（ループ終了命令を                    リピート対象命令
                 リピートしている）                 } ；
```

例：ループ命令およびほかのループ命令の終端命令

```
LOOP count  {          LOOP count  {          ：ループ１
            ⋮                      ⋮
            LOOP count             LOOP count  {   ：ループ２
            } ；                   ⋮
                                          } ；    ：ループ2END
                                   } ；            ：ループ1END
```

（９）ループ命令のカウンタ値に負の数値，または小数点数を記述した場合，フェイタル・エラーを出力します。

保守／廃止

５.２.13　制御命令（条件命令）　（命令セット13）

[記述方法]

命令名称	ニモニック	オペランドに記述できるもの
条件	IF（ro cond）	ro = {R0-R7}， cond：下記参照

[記述上の注意]

（１）condに記述できる条件を次に示します。

EVER　：無条件（このときは，roを省略可能）
==0　：=0
!=0　：≠0
＞0　：＞0
＜0　：＜0
<=0　：≦0
>=0　：≧0
==EX　：エクステンション（ビット39-31に０と１が混在する）
!=EX　：エクステンションなし（ビット39-31がすべて０または１）

（２）condを省略したとき，およびEVERを記述した場合は，無条件分岐とみなします。

（３）条件のみを記述した場合（単項演算命令，レジスタ間転送命令または，分岐命令が同時記述されていない場合）は，フェイタル・エラーを出力します。

（４）IF命令は複数行にわたって記述することはできません。

例：誤った記述形式
```
IF（R0 != 0）↵
CLR（R0）；
```

正しい記述形式
```
IF（R0 != 0）　CLR（R0）；
```

保守／廃止

５.２.14　制御命令（条件命令以外）　（命令セット14）

[記述方法]

命令名称	ニモニック	オペランドに記述できるもの
ノー・オペレーション	NOP	
ホルト	HALT	
ストップ	STOP （μPD77016にはありません。）	
ループ・ポップ	LPOP	
フォーゲット・インタラプト	FINT	

[記述上の注意]

（１）HALT命令，STOP命令をリピートの対象命令として記述しないでください。
また，ループ終端３命令以内に記述しないでください。

★ （２）μPD77111ファミリで$\overline{\text{WAKEUP}}$端子によるストップ・モードからの復帰を行う場合は，STOP命令の直前にNOP命令を記述してください。

例：nop；←必要
stop；$\overline{\text{WAKEUP}}$端子による復帰を前提としたSTOP命令

保守／廃止

５.２.15　３項演算＋並列ロード／ストア命令　（命令セット15，命令セット16）

[記述方法]

５.２.１　３項演算命令，および**５.２.５　ロード／ストア命令（並列ロード／ストア）**を参照してください。

[記述上の注意]

（１）３項演算と並列ロード／ストアの間には１個以上の空白（またはTAB）を記述してください。
また，必ず３項演算を先に（左側に）記述します。

３項演算　　並列ロード／ストア……正しい
並列ロード／ストア　　３項演算……誤り（フェイタル・エラーを出力します。）

（２）並列ロード／ストアのXメモリ転送部，Yメモリ転送部の記述順序はどちらからでも可能です。

（３）使用するレジスタが重複（３項演算，並列ロード／ストア部）するような場合には，次のような処理を行います。

例：

① R0 = R0＋R1＊R2　　＊DP0 = R0；
３項演算部 …… R0には演算結果が入ります。
並列ロード／ストア部 …… R0には３項演算前のR0の値が入ります。

② R0 = R0＋R1＊R2　　R0 = ＊DP0； …… フェイタル・エラーを出力します。

保守／廃止

５.２.16　２項演算命令（イミーディエト値を使用しない）＋並列ロード／ストア命令（命令セット17，命令セット18）

[記述方法]

５.２.２　２項演算命令（イミーディエト値を使用しない），および**５.２.５　ロード／ストア命令（並列ロード／ストア）**を参照してください。

[記述上の注意]

（１）並列ロード／ストア命令と同時記述できる２項演算命令は，イミーディエト値を使用しないものにかぎります。イミーディエト値を使用した２項演算命令は，並列ロード／ストア命令と同時記述することはできません。

（２）２項演算と並列ロード／ストアの間には１個以上の空白（またはTAB）を記述します。また，必ず２項演算を先に（左側に）記述します。

２項演算　　並列ロード／ストア……正しい
並列ロード／ストア　　２項演算……誤り（フェイタル・エラーを出力します。）

（３）並列ロード／ストアのXメモリ転送部，Yメモリ転送部の記述順序はどちらからでも可能です。

（４）使用するレジスタが重複（２項演算部，並列ロード／ストア部）するような場合には，次のような処理を行います。

例：

① R0 = R0＋R1　　＊DP0 = R0；
　２項演算部 …… R0には演算結果が入ります。
　並列ロード／ストア部 …… R0には２項演算前のR0の値が入ります。
② R0 = R0＋R1　　R0 = ＊DP0；　…… フェイタル・エラーを出力します。

保守／廃止

５.２.17　単項演算命令＋並列ロード／ストア命令　（命令セット19，命令セット20）

[記述方法]

５.２.４　単項演算命令，および**５.２.５　ロード／ストア命令（並列ロード／ストア）**を参照してください。

[記述上の注意]

（１）単項演算とロード／ストアの間には１個以上の空白（またはTAB）を記述します。また，必ず単項演算を先に（左側に）記述します。

単項演算　　並列ロード／ストア……正しい
並列ロード／ストア　　単項演算……誤り（フェイタル・エラーを出力します。）

（２）並列ロード／ストアのXメモリ転送部，Yメモリ転送部の記述順序はどちらからでも可能です。

（３）使用するレジスタが重複（単項演算部，並列ロード／ストア部）するような場合には，次のような処理を行います。

例：

① R0 = R0＋1　　＊DP0 = R0；
　単項演算部 …… R0には演算結果が入ります。
　並列ロード／ストア部 …… R0には単項演算前のR0の値が入ります。
② R0 = R0＋1　　R0 = ＊DP0； …… フェイタル・エラーを出力します。

５.２.18　条件＋単項演算命令　（命令セット21）

[記述方法]

５.２.４　単項演算命令，および**５.２.13　制御命令（条件命令）**を参照してください。

[記述上の注意]

IF条件命令と単項演算命令の間には１個以上の空白（またはTAB）を記述します。

保守／廃止

５.２.19　条件＋レジスタ間転送命令　（命令セット22）

[記述方法]

５.２.８　レジスタ間転送命令，および**５.２.13　制御命令（条件命令）**を参照してください。

[記述上の注意]

（１）IF条件命令とレジスタ間転送命令の間には１個以上の空白（またはTAB）を記述します。

（２）次のレジスタは，IF条件命令と同時に記述することはできません。

SR, EIR, STK, SP, LSA, LEA, LC, LSP, LSR1, LSR2, LSR3, ESR

５.２.20　条件＋分岐命令　（命令セット23）

[記述方法]

５.２.10　分岐命令，および**５.２.13　制御命令（条件命令）**を参照してください。

[記述上の注意]

IF条件命令と分岐命令の間には１個以上の空白（またはTAB）を記述します。

５.２.21　リピート命令に記述可能な命令　（命令セット24）

[記述方法]

５.２.11　ハードウエア・ループ命令（リピート）を参照してください。

[記述上の注意]

（１）リピート対象命令には，３項演算命令，２項演算命令，単項演算命令，ロード／ストア命令，レジスタ間転送命令，即値設定命令，および制御命令（ただし，条件命令＋分岐命令（命令セット23）の組み合わせおよび制御命令のループ・ポップ命令を除く）が記述できます。

（２）リピート対象命令に分岐命令，リピート命令，ループ命令，ループ終端の命令およびループ・ポップ命令を記述した場合，フェイタル・エラーを出力します。

第６章　疑似命令

６.１　疑似命令の種類

疑似命令の種類を表６－１に示します。

表６－１　疑似命令の種類

疑似命令の種類		意　　味
セグメント開始	IMSEG	インストラクション・メモリ用セグメントを定義します。
	XROMSEG	XメモリROM用セグメントを定義します。
	XRAMSEG	XメモリRAM用セグメントを定義します。
	YROMSEG	YメモリROM用セグメントを定義します。
	YRAMSEG	YメモリRAM用セグメントを定義します。
シンボル定義	EQU	式にネームを定義します。（再定義不可）
	SET	式にネームを定義します。（再定義可）
	%DEFINE	マクロ名を定義する。（再定義不可）
ロケーション・カウンタ制御	ORG	ロケーション・カウンタの値を指定します。
領域確保	DS	領域を確保します。
	DW	データ付きで領域（データROM：16ビット，インストラクション・メモリ：32ビット）を確保します。
	DWL	データ付きで領域（32ビット）を確保します。
プログラム・リンケージ	NAME	モジュール名を定義します。
	PUBLIC	外部定義名を宣言します。
	EXTRN	外部定義名の参照を宣言します。
アセンブル終了	END	アセンブルの終了をアセンブラに知らせます。

保守／廃止

６．１．１　セグメント開始疑似命令

セグメントの開始を指定します。

μPD77016ファミリ・アセンブリ言語は，セグメントの終了を記述するための疑似命令を持たないので，セグメントの開始とともに，直前のセグメントの終了が指定されたことになります。

セグメントには，次の３タイプ５種類があり，それぞれ異なるセグメント開始疑似命令で指定します。

・インストラクション・メモリ用	IMSEG
・データROM用	XROMSEG, YROMSEG
・データRAM用	XRAMSEG, YRAMSEG

配置属性指定は，リンカに対し，そのセグメントをどこに配置するかを指示します。

リロケータブル属性を指定する場合は，オペランド欄に“ INTERNAL ”または“ EXTERNAL ”を記述して，内部メモリまたは外部メモリのどちらに配置するかを指定します。

アブソリュート属性を指定する場合は，オペランド欄に“ AT ”で始まるアブソリュート属性指定を記述します。

詳しい説明は**表６－２**を参照してください。

保守／廃止

IMSEG etc

IMSEG etc

[記述形式]

シンボル欄	ニモニック欄	オペランド欄
セグメント名	IMSEG	[配置属性指定]
セグメント名	XROMSEG	[配置属性指定]
セグメント名	XRAMSEG	[配置属性指定]
セグメント名	YROMSEG	[配置属性指定]
セグメント名	YRAMSEG	[配置属性指定]

[機　　能]

セグメントの開始を指定します。

[説　　明]

・同名のセグメントが存在した場合はアボート・エラーとなります。

・セグメント開始疑似命令ステートメントは，そのオペランドにより，セグメントに属性を付与します。

・配置属性指定を表６－２に示します。

IMSEG/XROMSEG/XRAMSEG/YROMSEG/YRAMSEGで配置属性指定を省略した場合は，INTERNAL属性とします。

表６－２　配置属性指定

配置属性	記述形式	説　　明
INTERNAL	INTERNAL	指定セグメントを内部メモリに配置します。 リロケータブル属性です。
EXTERNAL	EXTERNAL	指定セグメントを外部メモリに配置します。 リロケータブル属性です。
AT	AT△n	指定セグメントを絶対番地で指定します。アブソリュート属性です。

保守／廃止

IMSEG etc

IMSEG etc

AT△nで記述するnにはアブソリュートな属性の整数型の式を記述します。このとき“ $ ”を含めることはできません。

セグメントを配置する範囲を表６－３に示します。

表６－３　セグメントを配置する範囲

疑似命令	オペランド	
	形　式	範　　囲
		nはアブソリュートな整数式で，値の範囲は，アセンブル環境ディレクティブ・ファイルの以下の命令で指定したアドレス範囲です。
IMSEG	AT△n[注]	EIMEM/IROM/IRAMで指定したアドレス範囲
XROMSEG	AT△n[注]	XROM/EXROMで指定したアドレス範囲
YROMSEG	AT△n[注]	YROM/EYROMで指定したアドレス範囲
XRAMSEG	AT△n[注]	XRAM/EXRAMで指定したアドレス範囲
YRAMSEG	AT△n[注]	YRAM/EYRAMで指定したアドレス範囲

注　AT△nの前は，空白相当文字で区切ります。

注意　アセンブリ命令または疑似命令により，セグメントのサイズを越えた場合，セグメントのサイズを越えた命令に対してのみ，フェイタル・エラーを出力します。それ以降の命令に対してはエラーを出力しません。いずれもセグメントのサイズを越えたアドレスを有効として，処理を続行します。

保守／廃止

６.１.２　シンボル定義疑似命令

シンボルを定義し，値を割り当てます。

シンボル定義疑似命令には，次の３種類があります。

なお，リテラル・ネームについては**第７章　汎用コントロール**を参照してください。

- EQU
- SET
- %DEFINE[注]

注　%DEFINEはマクロ定義命令です。詳しい説明については**８.２　マクロの機能**を参照してください。

保守／廃止

EQU　　EQU

（1）EQU

[記述形式]

シンボル欄	ニモニック欄	オペランド欄
ネーム	EQU	式

[機　　能]

ネームを定義して，値を割り当てます。同じネームに対して１回だけ定義することができます。

[説　　明]

・ネームおよび式は必ず記述してください。記述しない場合はフェイタル・エラーとなります。

・同じネームに対して２回以上EQU定義を行った場合は，フェイタル・エラーとなります。

・式にシンボルが含まれる場合には，すでにその値が定義されている必要があります（シンボルの前方参照および外部参照はできません）。

・EQUで定義したシンボルをEXTRN宣言した場合は，フェイタル・エラーとなります。また，EXTRN宣言されたシンボルをEQUで定義した場合には，フェイタル・エラーとなります。

・ここで定義したネームは，アセンブリ命令ステートメントまたは疑似命令ステートメントに記述でき，オペランド欄に記述した式の値が割り当てられます。

ネームの参照は，定義の前でもあとでもかまいません。

保守／廃止

SET　　SET

（2）SET

[記述形式]

シンボル欄	ニモニック欄	オペランド欄
ネーム	SET	式

[機　　能]

ネームの値を割り当てます。同じネームに対して複数回定義できます。

[説　　明]

・ネームおよび式は必ず記述してください。記述しない場合には，フェイタル・エラーになります。

・式にシンボルが含まれる場合には，すでにその値が定義されている必要があります（シンボルの前方参照および外部参照はできません）。

・SETで定義したシンボルをPUBLIC宣言した場合には，フェイタル・エラーになります。

・SETで定義したシンボルをEXTRN宣言した場合には，フェイタル・エラーになります。また，EXTRN宣言されたシンボルをSETで定義した場合も，フェイタル・エラーになります。

・ここで定義したネームは，アセンブリ命令ステートメントまたは疑似命令ステートメントに記述することができます。オペランド欄に記述した式の値が割り当てられます。
ネームの参照は，その直前に与えられた値が割り当てられます。ネームを最初に定義する以前のネームの参照は，フェイタル・エラーとなります。

例：

```
        ⋮
        DW ABC   …フェイタル・エラーとなります。
        ⋮
ABC     SET 123
        ⋮
        DW ABC   …DW 123と解釈されます。
        ⋮
ABC     SET 456
        ⋮
        DW ABC   …DW 456と解釈されます。
```

保守／廃止

６.１.３　ロケーション・カウンタ制御疑似命令

ロケーション・カウンタとは，次のセグメントごとに異なるポインタを指定します。

・インストラクション・メモリ用セグメント
・データROM用セグメント
・データRAM用セグメント

ロケーション・カウンタ制御疑似命令には，次のものがあります。

・ORG

保守／廃止

ORG　　ORG

[記述形式]

シンボル欄	ニモニック欄	オペランド欄
[ネーム]	ORG	式

[機　　能]

アブソリュートなセグメントならば，このステートメントの次のステートメントに対して，絶対番地を指定します。リロケータブルなセグメントならば，このステートメントの次のステートメントに対して，セグメントの先頭からの相対番地を指定します。

[説　　明]

・ロケーション・カウンタの値は，オペランド欄の式で指定します。

・式の値は，このステートメント出現時にすでに確定されており，現在のロケーション・カウンタの値よりも，大きいか，等しくなければなりません。この条件を満たさない場合はフェイタル・エラーとなります。

・ORG疑似命令は，リロケータブルなセグメントと，アブソリュートなセグメントのどちらにも用いることができます。

このとき，リロケータブルなセグメントの場合には，ロケーション・アドレスはセグメントの先頭アドレスにオペランドで記述された式の値を加えた値となります。また，アブソリュートなセグメントの場合には，オペランドで記述された式の値となります。

保守／廃止

６.１.４　領域確保疑似命令

領域確保疑似命令には，次の３種類があります。

・DS　（領域確保）
・DW　（データ付き領域確保）
・DWL（データ付き領域確保）

保守／廃止

DS　　DS

（1）DS

[記述形式]

シンボル欄	ニモニック欄	オペランド欄
[レーベル：]	DS	式

[機　　能]

データRAMの領域を確保し，その先頭番地をシンボルに割り付けます。

[説　　明]

・式にシンボルが含まれる場合は，すでにその値が絶対形式で定義されている必要があります（シンボルの前方参照および外部参照はできません）。
また，式の値には正の整数のみ記述することができます。

・ワード単位（16ビット）で領域を確保します。オペランド欄の式には，領域数を記述します。

例：

```
DS    5
```

5ワードの領域が確保されます。

保守／廃止

DW　　DW

（2）DW

[記述形式]

シンボル欄	ニモニック欄	オペランド欄
[レーベル：]	DW	式 [，式・・・]

[機　　能]

データROMまたはインストラクション・メモリの領域をデータ定義（初期化）し，その先頭番地をシンボルに割り付けます。

[説　　明]

・ワード単位（データROM：16ビット）で領域を確保し，その領域へデータを書き込みます。オペランド欄の式には，データを記述します。

・データを複数個記述するときは，カンマ“，”で区切ります。このときは式の数だけ領域を確保します。

例：

```
DW　１，２，３
```

３ワードの領域を確保し，それぞれ0x0001, 0x0002, 0x0003を書き込みます。

保守／廃止

DWL

DWL

（３）DWL

[記述形式]

シンボル欄	ニモニック欄	オペランド欄
[レーベル：]	DWL	式 [，式・・・]

[機　　能]

データROMの領域をデータ定義（初期化）し，その先頭番地をシンボルに割り付けます。

[説　　明]

・２ワード単位（32ビット）で領域を確保し，その領域へデータを書き込みます。オペランド欄の式には，データを記述します。

例：

（0.3 = 0x26666666）

```
XSEG    XROMSEG
        DW          0.3
        DWL         0.3
```

Xメモリ	Yメモリ
2666	
2666	
6666	

```
XSEG    XROMSEG
        DW          0.3
YSEG    YROMSEG
        DWL         0.3
```

Xメモリ	Yメモリ
2666	2666
	6666

保守／廃止

６.１.５　プログラム・リンケージ疑似命令

オブジェクト・モジュール名，外部定義名，外部参照名を宣言します。

プログラム・リンケージ疑似命令には，次の３種類があります。

- NAME
- PUBLIC
- EXTRN

保守／廃止

NAME　　NAME

（１）NAME

[記述形式]

シンボル欄	ニモニック欄	オペランド欄
	NAME	オブジェクト・モジュール名

[機　　能]

アセンブラで生成するオブジェクト・モジュールにネームを割り付けます。

[説　　明]

・オブジェクト・モジュール名は，シンボル名と同じ形式で記述します。アセンブラは記述されたオブジェクト・モジュール名の先頭から８文字をシンボル・テーブルに登録します。
登録するオブジェクト・モジュール名の長さは，シンボルが31文字認識である場合でも８文字です。
-CA/-NCAオプションは，オブジェクト・モジュール名についても有効です。

・オブジェクト・モジュール名を機械語命令，または疑似命令のオペランドに記述することはできません。

・この疑似命令はソース・モジュール・ヘッダ部にのみ記述することができます。

・この疑似命令がソース・モジュール内に複数記述された場合は，ワーニング・エラーを出力してその行は無視します。このとき，最初の疑似命令だけが有効となります。

・この疑似命令がソース・モジュール内に存在しない場合は，ソース・モジュール・ファイル名のプライマリ・ネームの先頭から８文字をオブジェクト・モジュール名としてシンボル・テーブルに登録します。このとき，プライマリ・ネームが８文字未満であった場合には，登録されるモジュール名はプライマリ・ネームと同じ長さにします。また，プライマリ・ネームは大文字に変換されて取り出されます。
シンボルとして不適当な文字が含まれていた場合には，モジュール名を決定することができないので，アボート・エラーとなります。

保守／廃止

PUBLIC　　PUBLIC

（２）PUBLIC

[記述形式]

シンボル欄	ニモニック欄	オペランド欄
	PUBLIC	シンボル［，・・・］

[機　　能]

オペランドで記述したシンボルを，ほかのモジュールでも参照できるようになります。

[説　　明]

・シンボルが２つ以上ある場合はカンマで区切る必要があります。
・シンボルは全モジュール中で１回だけPUBLIC宣言することが可能です。同一シンボルの２回目以後の宣言に対しては，ワーニング・エラーを出力します。
・オペランドで記述するシンボルは同一モジュール内で定義されていなければなりません。定義されていないシンボルに対しては，フェイタル・エラーを出力し，PUBLIC宣言を取り消します。
・この疑似命令はソース・モジュール・ファイル中のどこにでも記述できます。
・シンボルはカンマで区切って，１行に最大20個まで記述できます。最大個数を越えた場合はフェイタル・エラーを出力し，21個目からのPUBLIC宣言を取り消します。
・次に示すシンボルを指定すると，フェイタル・エラーになります。
　：SETシンボル
　：外部参照シンボル（すでにEXTRN宣言された外部参照シンボル）
　：セグメント名
　：モジュール名
　：モジュール内で定義されていないシンボル
・オペランドにエラーがあった場合は，そのシンボル名をエラー・メッセージ中に出力します。

保守／廃止

EXTRN

EXTRN

（３）EXTRN

[記述形式]

シンボル欄	ニモニック欄	オペランド欄
	EXTRN	シンボル名［，・・・］

[機　　能]

オペランドで記述するシンボルを，このモジュールで参照できるようになります。

[説　　明]

- オペランドで記述するシンボルは，リンカでリンクするほかのモジュール内でPUBLIC疑似命令により宣言されている必要があります。
- この疑似命令はソース・モジュール・ファイル中どこにでも記述できます。
- シンボルは１つのモジュール内で１回だけEXTRN宣言することが可能です。同一シンボルの２回目以後の宣言に対しては，ワーニング・エラーを出力します。
- シンボルはカンマで区切って１行に最大20個まで記述できます。
  最大個数を越えた場合はフェイタル・エラーを出力し，21個目からのEXTRN宣言を取り消します。
- すでに定義されているシンボル，すでにPUBLIC宣言されたシンボル，またはすでにNAME疑似命令により宣言されたモジュール名をオペランドに記述すると，フェイタル・エラーとなり，そのシンボルのEXTRN宣言を取り消します。
- 宣言されたシンボルはNUMBER属性として扱います。
- オペランドにエラーがあった場合は，そのシンボル名をエラー・メッセージ中に出力します。
- すでにEXTRN宣言されたシンボルを．他の疑似命令により再度定義または宣言した場合は，そこでフェイタル・エラーとなります。
- EXTRN宣言されたシンボルを，モジュール中で必ずしも参照する必要はありません。

保守／廃止

６.１.６　アセンブル終了疑似命令

アセンブル終了疑似命令には次のものがあります。

・END

保守／廃止

END　　END

[記述形式]

シンボル欄	ニモニック欄	オペランド欄
	END	

[機　　能]

アセンブルを終了します。

[説　　明]

・ソース・モジュールの最後に必ず記述します。

・記述されていない場合は，フェイタル・エラーとし，ファイルの終了でENDがあったものとします。

・ENDのあとの行に空白，TAB，改行コード，コメント以外のステートメントがあった場合，一度だけワーニング・エラーを出力します。

なお，ENDのあとのステートメントは，すべてアセンブル・リストに出力します。ただし，アセンブル対象とはなりません。

〔メ　モ〕

第７章　汎用コントロール

汎用コントロールは，ソース・モジュール・ファイルおよびインクルード・ファイル中に記述し，アセンブラに対して動作の詳細を表示するものです。

７.１　記述形式

[形　　式]

（１）リスト制御用形式

[△] $ [△] 汎用コントロール [▲] ↵

（２）インクルード・ファイル指定，条件付きアセンブル制御，リテラル・ネーム指定用形式

[△] # [△] 汎用コントロール [▲] ↵

[説　　明]

・$，#は，ソース・ステートメントの先頭に記述してください。ただし$，#の前に空白またはタブは記述できます。

・$，#に続いて，任意の数の空白またはタブのあとに汎用コントロールを記述します。

・汎用コントロールは，ソース・モジュール・ファイルの中のみ記述します。

・汎用コントロール指定に誤りがあった場合，フェイタル・エラーを出力し，その汎用コントロール指定がなかったものとしてアセンブル処理は続行されます。

・汎用コントロール行に記述できる文字セットは，アセンブラの文字セットと同等とします。

７.２　汎用コントロールの種類

汎用コントロールは，ソース・モジュール・ファイル中の任意の箇所に記述可能で，アセンブラに動作の詳細を指示するものです。

ただし，１行には１つの汎用コントロールのみとします。

汎用コントロールには次のものがあります。

（１）インクルード・ファイル指定

・INCLUDE

（２）リスト制御

・EJECT

・LIST/NOLIST

・TITLE

・GEN/NOGEN/GENONLY

（３）条件付きアセンブル制御

・IF/IFDEF/IFNDEF/ELSE/ENDIF

（４）リテラル・ネーム指定

・DEFINE

保守／廃止

７.３　汎用コントロールの初期状態

汎用コントロールの初期状態を表７－１に示します。ソース・モジュール・ファイルおよびインクルード・ファイル中で指定しないかぎり，この初期状態でアセンブル処理は実行されます。

表７－１　汎用コントロールの初期状態

	項　番	汎用コントロール	初期状態
	1	#INCLUDE	なし
	2	$EJECT	なし
	3	$LIST/NOLIST	アセンブル・リスト出力
	4	$TITLE	アセンブル・リストのヘッダのタイトル部は空白
	5	#IF/IFDEF/IFNDEF/ELSE/ENDIF	なし
	6	#DEFINE	なし
	7	$GEN/$NOGEN/$GENONLY	GEN（マクロ定義行／参照行／展開行をアセンブル・リストに出力）
★	8	$PAGELENGTH	63
★	9	$PAGEWIDTH	132
★	10	$RADIX	10

保守／廃止

7.4　インクルード・ファイル指定

（1）INCLUDE

[形　　式]

[△]＃[△]INCLUDE[△]"[△] ディスク型ファイル名 [△]"[▲] ↵

[△]＃[△]INCLUDE[△]＜[△] ディスク型ファイル名 [△]＞[▲] ↵

[短 縮 形]

IC

[説　　明]

・INCLUDEは " ディスク型ファイル名 " で指定したファイルの内容をアセンブル実行時に展開します。

・＃INCLUDE＜ファイル名＞形式は，-Iオプションでインクルード・ファイルのパス名あるいはドライブ名を指定することができます。

・インクルード・ファイルの読み出しパスを探す順番は次のとおりです。

（a）＃INCLUDE＜ファイル名＞形式で，インクルード・ファイルがパス名なしで指定されているとき

① ソース・モジュール・ファイルのあるパス

② -Iオプションで指定されたパス

③ 環境変数 ' INC77016 ' に設定されているパス

（b）＃INCLUDE＜ファイル名＞形式で，インクルード・ファイルがパス名付きで指定されていて，さらに次のような条件があるとき

（i）インクルード・ファイルがドライブ名または￥から始まるパス名付きで指定されているとき

① インクルード・ファイル名に付いているパス

（ii）インクルード・ファイルが相対パス（先頭が￥でない）付きで指定されているとき

指定されたインクルード・ファイル名の前に前記（a）の順でパス名を付加します。

（c）＃INCLUDE " ファイル名 " 形式で，インクルード・ファイルがパス名なしで指定されているとき

① ソース・モジュール・ファイルのあるパス

（d）＃INCLUDE " ファイル名 " 形式で，インクルード・ファイルがパス名付きで指定されていて，さらに次のような条件があるとき

（i）インクルード・ファイルがドライブ名または￥から始まるパス名付きで指定されているとき

① インクルード・ファイル名に付いているパス

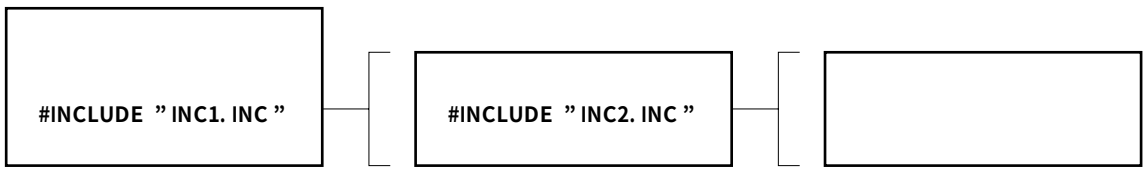
（ii）インクルード・ファイルが相対パス（先頭が￥でない）付きで指定されているとき

指定されたインクルード・ファイル名の前にソース・モジュール・ファイルのあるパス名を付加します。

保守／廃止

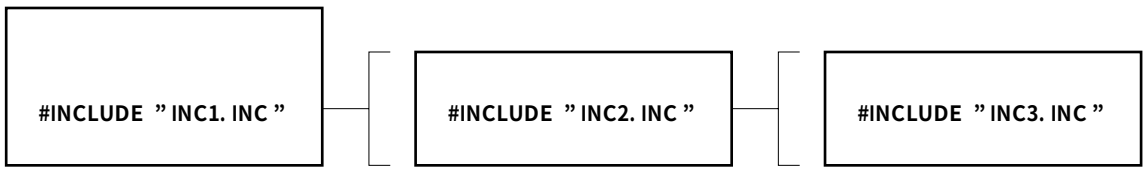
・インクルードのネストは最大２レベルまでです。２レベルを越えるとフェイタル・エラーになります。
・インクルード・ファイルにEND疑似命令を記述すると，アセンブラはそこで実行を終了します。
・インクルード・ファイルがオープンできないときは，アボート・エラーとなります。
・インクルード・ファイルと同じ名前で出力ファイルを作成しないよう注意してください。同じ名前で出力ファイルを作成した場合，インクルード・ファイルのオープン時にアボート・エラーとなり，インクルード・ファイルを失ってしまう可能性があります。
・１つのインクルード・ファイル中では，IF/IFDEF/IFNDEF-ENDIFの対応がとれている必要があります。
アセンブラは，インクルード・ファイルの展開時にIF/IFDEF/IFNDEFの対応をチェックします。インクルード・ファイルの展開入口のIF/IFDEF/IFNDEFのレベルと，インクルード・ファイルの展開終了直後のIF/IFDEF/IFNDEFのレベルが等しくなければ，フェイタル・エラーになり，展開直後のレベルを，強制的に展開入口でのレベルに戻して，アセンブルを続けます。
なお，アセンブル対象とならないINCLUDEは，インクルード・ファイルの展開をしないため，インクルード・ファイル内でチェックされません。

例１．



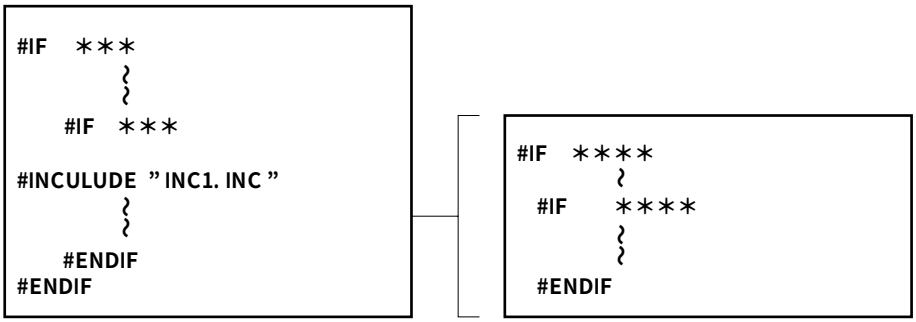
ソース・モジュール・ファイル　　インクルード・ファイル１　　インクルード・ファイル２

２．



ソース・モジュール・ファイル　　インクルード・ファイル１　　インクルード・ファイル２
フェイタル・エラーとなり，インクルード・ファイルを展開しません。

例３．



備考　インクルード・ファイルの展開入り口ではIFのレベルは２レベルで，展開終了後は３レベルとなり，フェイタル・エラーとします。

保守／廃止

７.５　リスト制御

リスト制御は，アセンブル対象行ならば有効とします。ただし，汎用コントロールと起動コマンドのオプションの間で優先順位があります。

また，出力を制御させる汎用コントロールが有効で，リストに出力されない行であってもエラーが発生した場合には，エラー・メッセージとともにその行を出力します。

（１）EJECT

[形　　式]

[△] $ [△] EJECT [▲] ↵

[短 縮 形]

EJ

[説　　明]

① EJECTは，アセンブル・リストの改ページを指定します。

② EJECTが１ページの最終行になった場合は，改ページは１回だけ行います。

③ 起動行上で-NP,-LL0オプションの指定があった場合，あるいはソース・モジュール中のリスト制御指定によりリスト出力禁止状態のときは，EJECTは無効となります。

（２）LIST/NOLIST

[形　　式]

[△] $ [△] LIST [▲] ↵

[△] $ [△] NOLIST [▲] ↵

[短 縮 形]

LI/NOLI

[説　　明]

・LISTは，アセンブル・リストの出力を開始する場所を指定します。

・NOLISTは，アセンブル・リストの出力を中止する場所を指定します。

・LISTは，リストの出力量を制限するためのもので，リスト出力中止後もアセンブルは続けられ，行番号もカウントされます。リスト・ファイルに出力したくない部分の前に$NOLIST，後ろに$LISTを記述することにより，リスト・ファイルの量を減らすことができます。

・NOLIST指定後，LIST指定があれば，再びリストの出力は開始され，LIST自身は印字されます。

・このコントロールの有効範囲内にエラー行があった場合，エラー・メッセージを出力します。

保守／廃止

（３）TITLE

[形　　式]

[△] $ [△] TITLE [△] ”[△] タイトル・ストリング [△] ”[▲] ↵

[短 縮 形]

TT

[説　　明]

・TITLEは，アセンブル・リストのヘッダのタイトル部に印字するタイトル・ストリングを指定します。このコントロールが指定されると，それ以降の改ページのリスト・ヘッダのタイトル部に，指定されたタイトル・ストリングが印字されます。
ただし，ページの先頭にTITLEが記述された場合は，そのページのリスト・ヘッダのタイトル部にもタイトル・ストリングを出力します。

・タイトル・ストリングは，展開イメージで72文字以内とします。73文字以上記述すると，先頭の72文字を有効とし，エラー・メッセージは出力しません。
全角文字は２文字，タブは１文字と数えます。なお，タブ（0x09）は常に空白（0x20）１個と等価として処理を行います。

・タイトル・ストリングに引用符“ ”を記述したいときは２個続けます。

例　”AB” ”C” →AB”Cと印字します。

・文字数が０の場合はタイトル欄に“↵”１個が指定されたものとします。

・指定されたタイトル・ストリングの中に不当文字（0x00～0x08, 0x0B, 0x0C, 0x0E～0x19, 0x1B～0x1F, 0x7F）が記述された場合は，“ ^ ”に置き換えてタイトル欄を表示します。
CR（0x0D）が記述されると，リスト上には何も出力されません。
ヌル・コード（0x00）が現れた場合は，フェイタル・エラーを出力し，それ以降“ ”で閉じるまでの文字列は，タイトル・ストリングとして表示されません。

・$NOLISTによりリストが出力されないときも，$TITLEによるタイトル・ストリングの指定は有効です。このときは，$LISTによりリスト出力が再開されたあとタイトル・ストリングを印字します。

保守／廃止

（4）GEN/NOGEN/GENONLY

[形　　式]

[△] $ [△] GEN [▲] ↵

[△] $ [△] NOGEN [▲] ↵

[△] $ [△] GENONLY [▲] ↵

[短 縮 形]

なし

[説　　明]

- GEN/NOGEN/GENONLYは，マクロ定義行，マクロ参照行およびマクロ展開行のアセンブル・リスト出力を制御します。
  どれもコントロール自身のイメージを出力したあとに，リストを制御します。
- GENを指定すると，このコントロールを指定したあとのマクロ定義行，マクロ参照行およびマクロ展開行を，そのままアセンブル・リストへ出力します。
- NOGENを指定すると，このコントロールを指定したあとのマクロ定義行およびマクロ参照行をアセンブル・リストへ出力し，マクロ展開行は出力しません。
  GENONLYを指定すると，このコントロールを指定したあとのマクロ展開行だけをアセンブル・リストへ出力し，マクロ定義行およびマクロ参照行は出力しません。
- これらのコントロールは，リストの出力量を制限するためのものです。したがって，リスト出力中止後もアセンブルは続行し，STNOはカウント・アップされます。
- NOGEN指定後，GEN指定があれば，再びマクロ展開行の出力を開始します。また，GENONLY指定後，GEN指定があれば，再びマクロ定義行とマクロ参照行の出力を開始します。
  NOGENとGENONLYが指定され，どちらも有効状態のときは，マクロ定義行，マクロ参照行およびマクロ展開行ともに出力しません。
- NOGENまたはGENONLYによるマクロのリスト出力中止中にエラーが発生した場合，エラー行とエラー・メッセージはアセンブル・リストへ出力します。

保守／廃止

★　（５）PAGELENGTH

[形　　式]

[△] $ [△] PAGELENGTH△ページあたりの行数 [▲] ↵

[短 縮 形]

PGLEN

[説　　明]

・プリント・ファイルおよびエラー・ファイルのページあたりの行数を指定します。

・このコントロールによって指定された値にはヘッダ行が含まれます。

・値として指定できる数字は，8-65535です。ディフォールトは63です。

★　（６）PAGEWIDTH

[形　　式]

[△] $ [△] PAGEWIDTH△行あたりの桁数 [▲] ↵

[短 縮 形]

PGWI

[説　　明]

・プリント・ファイルおよびエラー・ファイルの行あたりの桁数を指定します。

・指定された桁数に達すると改行し，次の行の先頭から印字を再開します。

・値として指定できる数字は，60-65535です。ディフォールトは132です。

保守／廃止

7.6　条件付きアセンブル制御

（1）IF/IFDEF/IFNDEF/ELSE/ENDIF

[形　　式]

[△]＃[△] IF△ 条件式 [▲] ↵

[△]＃[△] IFDEF [△]([△] リテラル・ネーム／マクロ名 [△])[▲] ↵

[△]＃[△] IFNDEF [△]([△] リテラル・ネーム／マクロ名 [△])[▲] ↵

[△]＃[△] ELSE [▲] ↵

[△]＃[△] ENDIF [▲] ↵

[短 縮 形]

なし

[説　　明]

・IF/ELSE/ENDIFコントロールは指定された条件式の評価値により，アセンブルを行う範囲を限定します。IFDEF/IFNDEF/ELSE/ENDIFコントロールは指定されたリテラル・ネームまたはマクロ名の評価値により，アセンブルする範囲を限定します。

・IFは指定された条件式の評価値により，アセンブル実行の対象部分が決まります。IFDEF/IFNDEFは，指定されたリテラル・ネームまたはマクロ名の評価により，アセンブル実行の対象部分が決まります。

IF/IFDEF/IFNDEFそれぞれのオペランドの評価が真になるのは，次の条件のときです。偽はこの条件にあてはまらない場合です。

IF	：条件式の評価値が0以外のとき。
IFDEF	：オペランドに指定されたリテラル・ネームまたはマクロ名がこのコントロール指定行以前に，すでに定義されているとき。
IFNDEF	：オペランドに指定されたリテラル・ネームまたはマクロ名がこのコントロール指定行以前に，定義されていないとき。

アセンブル実行の対象になる制御の詳細は次のとおりです。

（a）IF/IFDEF/IFNDEFのオペランドの評価が真の場合，次に現れるELSEまたはENDIFまでアセンブルを実行します。ELSEまでの実行であれば，次に現れるENDIFまでスキップし，条件アセンブルの終了となります。

（b）評価が偽の場合，次に現れるELSEまたはENDIFまでスキップします。

（ⅰ）ELSEが現れたときは，ENDIFまでをアセンブルします。

（ⅱ）ENDIFが現れたときは，条件アセンブルの終了になります。

このような処理をIF/IFDEF/IFNDEFから順に条件アセンブルが終了するまで行います。

・条件式に記述できるのは絶対式です。

・IF/IFDEF/IFNDEF汎用コントロールは８レベルまでネスティングできます。
レベル・オーバした場合は，フェイタル・エラーとし，オーバしたレベルのIF/IFDEF/IFNDEF-ENDIFまでスキップして以降の処理を続行します。このエラー・チェックは，条件アセンブルにより対象行でなくなっても行います。

・IF/IFDEF/IFNDEF-ENDIFの対応で，ソース・モジュール・ファイルの最後で対応がとれない場合，次のようになります。

（ａ）ENDIFが足りない場合はその場でフェイタル・エラーとし，ソース・モジュールの最後の足りないENDIFを補って処理を続けます。

（ｂ）ENDIFが余った場合はその場でフェイタル・エラーとし，ENDIFを無視してそのまま処理を続けます。

例１．

```
#IF   条件式α        ;条件式αの値が真のとき，text部のアセンブルが実行されます。
        ⋮
      text              条件式αの値が偽の場合には，text部のアセンブルは実行されません。
        ⋮
#ENDIF                ;IFコントロールで操作対象となるtext部の終了を示します。
```

２．

```
#IF   条件式α        ;条件式αの値が真のとき，text1部のアセンブルが実行されます。
        ⋮
      text1
        ⋮             text1部のアセンブル実行が終わるとENDIFの次のステートメントの実行に
        ⋮             移ります。
#ELSE                 ;IFコントロールで指定した条件式αの値が偽の場合，text2部のアセンブル
        ⋮             が実行されます。
      test2
        ⋮
#ENDIF
```

３．

```
#IFDEF（LITERAL）  ;LITERALがすでにリテラル・ネームまたはマクロ名として定義されていれ
        ⋮             ば，text部のアセンブルが実行されます。
      test
        ⋮
#ENDIF                ;IFDEFコントロールで操作対象となるtextの終了を示します。
```

保守／廃止

★ ７.７　基数指定

（１）RADIX

[形　　式]

[△] $ [△] RADIX△基数 [▲] ↵

[短 縮 形]

なし

[説　　明]

・ディフォールトの基数を指定します。基数表現文字（0b, 0q, 0t, 0xなど）を省略した数字はディフォールトの基数をもとに取り扱われます。

・基数として指定できる値は，２，８，10，16です。ディフォールトは10です。

例

```
$RADIX 10    ; 1234という値は0t1234として扱われます。
$RADIX 2     ; 11という値は0b11として扱われます。
```

保守／廃止

７.８　リスト制御の優先順位

（１）リスト制御の優先順位

リスト制御コントロールの優先順位を表７－２に示します。

表７－２　リスト制御の優先順位

先に指定 / あとに指定	$LIST	$NOLIST	$GEN	$GENONLY	$NOGEN
$LIST	○	○	○	○	×[注1]
$NOLIST	○	○	○	○	○
$TITLE	○	△	○	○	△
$EJECT	○	×	○	○	×[注2]
$GEN	○	×	○	○	○
$GENONLY	○	×	○	○	○
$NOGEN	○	×	○	○	○

注１．マクロ展開行内では無効。

２．マクロ展開行内の$EJECTが無効。

備考　○…有効

×…無効

△…タイトル・ストリングの置き換えのみ行い，出力が開始されたときに改ページをして初めてタイトル部に指定されたストリングを印字します。

○印の場合は，先に指定された汎用コントロールとあとに指定された汎用コントロールの両方とも，有効になります。

×印の場合は，先に指定された汎用コントロールを指定するとあとに指定された汎用コントロールは無効になります。ただし，アセンブル対象となった汎用コントロールにかぎります。また，先に指定された汎用コントロールが有効であっても，あとに指定された汎用コントロールの状態は保持されます。

（２）リスト制御コントロールのリスト出力の有無

表７－３のようにリスト制御コントロール自身のリストへの出力にも優先順位があります。

保守／廃止

表７－３　リスト制御コントロールのリスト出力の有無

先に指定／あとに指定	＄LIST	＄NOLIST	＄GEN	＄GENONLY	＄NOGEN
			上記状態時のマクロ展開行内のリスト制御コントロールに対して		
＄LIST	○	○	○	○	×
＄NOLIST	○	×	○	○	×
＄GEN	○	×	○	○	○
＄GENONLY	○	×	○	○	×
＄NOGEN	○	×	○	○	×

備考　○…印字する　　×…印字しない

×印の場合は，先に指定された汎用コントロールが有効ならば，あとに指定された汎用コントロール自身は出力されません。

保守／廃止

7.9　リテラル・ネーム指定

（1）DEFINE

[形　　式]

[△]＃[△] DEFINE { △リテラル・ネーム ― / [△]([△] リテラル・ネーム [△]) ― } ―[{ △言語文字列 / ([△]([△] 言語文字列 [△])) }] ―[[△] 言語文字列] [▲] ↵

[短 縮 形]

なし

[説　　明]

- 指定されたリテラル・ネームがDEFINEコントロール以降のシンボル欄，ニモニック欄およびオペランド欄に現れた場合，これを指定された文字列に置き換えます。同じリテラル・ネームに対して１回だけ置き換えられます。
- オペランドに指定できる言語文字列は，空白，タブ，；，CR, LF以外の言語文字列です。
- リテラル・ネームは必ず指定してください。指定しないと，フェイタル・エラーになります。
- 置き換え対象となる言語文字列は省略できます。このとき，リテラル・ネームは有効ですが，置き換え対象になる言語文字列は０文字とします。
- 同じリテラル・ネームに対してDEFINE定義をした場合は，フェイタル・エラーになります。
- 置き換えられた文字列を再びDEFINE定義により置き換えることはしないでください。

例

```
＃DEFINE ADD ＋
＃DEFINE PLUS ADD
                ・
                ・
                ・
R0 = R1 ADD R2；
          ↑
        ＋と解釈する
R0 = R1 PLUS R2
          ↑
        ADDと解釈し，＋とは解釈しない
```

- プリント・ファイルには置き換える前の言語文字列が出力されます。
- リテラル・ネームの前方参照はできません。
- ここで定義したリテラル・ネームは，アセンブリ命令ステートメントまたは疑似命令ステートメントに記述でき，オペランド欄に記述した文字列に展開されます。
- 置き換えられた文字列が予約語の場合，その予約語の処理をします。

・言語文字列をかっこで囲み，言語文字列中にもかっこを記述するとき，かっこの対応がとれていないとエラーになります。

保守／廃止

第8章　マ ク ロ

同一プログラム中で同じ命令群を何回も実行する場合，一般にサブルーチン化して，サブルーチン・コール命令を使用するとメモリ・サイズを節約できます。この命令群の占めるメモリ・サイズが比較的小さく，かつパラメータが複雑な場合にマクロ機能を使用すると，簡単にプログラムを作成できます。

8.1　マクロの利用

マクロ定義命令として“ %DEFINE ”があります。マクロは次のような手順で使用します。

（1）マクロの定義

まず，使用頻度の高い一連のステートメントをソース・プログラムの作成段階で，“ %DEFINE ”命令により，マクロ定義します。

マクロ定義により，一連のステートメント（マクロ・ボディ）はマクロ名で代表されます。

（2）マクロの参照

ソース・プログラム内で，以前に定義されたマクロを参照したい場合は，そのマクロ名を“ %マクロ名 ”の形式で，通常の命令と同じように“ ニモニック欄 ”に記述してください。マクロ・ボディ全体が参照されます。

（3）マクロの展開

マクロの定義および参照が行われているソース・プログラムをアセンブルすると，マクロを参照しているところにマクロを展開してアセンブルします。このとき，パラメータを用いたマクロでは，参照時に指定されたパラメータを用いて展開します。

8.1.1　マクロとサブルーチン

（1）サブルーチン

サブルーチン本体は１度だけ機械語に変換され，そのサブルーチン参照に対して，参照回数だけサブルーチン・コール命令を実行します。したがってサブルーチンを活用するとメモリを効率よく利用できます。

また，サブルーチン化することでプログラムの設計が容易になり，プログラム全体の構造が理解しやすくなります（プログラムの構造化）。

保守／廃止

（2）マ ク ロ

マクロの基本的な機能は命令群の置き換えです。アセンブラは，マクロ参照を検出すると，あらかじめ定義されていた命令群に置き換えます。その際にマクロ・ボディの仮パラメータを，参照時の実パラメータに置き換えます。

サブルーチン化の手法がメモリ・サイズの削減やプログラム構造化を図るために用いられるのに対し，マクロはコーディングの能率を向上させるため，あるいはプログラムの読みやすさを向上させるために用いられます。

たとえば，処理手順は同じですが，使用するデータとオペランドに現れるレーベルが異なる命令群を使う場合，データとレーベルに対して仮パラメータを割り当ててマクロを定義しておけば，コーディング時にはマクロ名と実パラメータ（該当するデータとレーベル）を記述するだけで種々の命令群を記述できます。

保守／廃止

8.2　マクロの機能

8.2.1 マクロの定義

（1）%DEFINE

[記述形式]

```
%DEFINE      （マクロ名［（仮パラメータ・リスト）］）
             [LOCAL　ローカル・リスト]（マクロ・ボディ）
```

備考　“ %DEFINE ”の記述形式は，シンボル欄，ニモニック欄，オペランド欄のフィールドに分けられません。

詳細な記述形式を示す構文については**3.2.4　マクロ命令ステートメント**を参照してください。

[機　　能]

マクロ・ボディに記述されている一連のステートメントに対して，マクロ名を割り付けます。このマクロ名はあとでマクロ・ボディを参照する場合の代表名になります。

[説　　明]

（a）マクロ名

① マクロ名はシンボルを構成する文字セットおよび構成規則に従います。

② マクロ名の取り扱いもシンボルと同じです。シンボル・テーブルに登録すると，ユーザ定義シンボルになります。予約語と同じ名前にしないでください。

③ マクロの最大登録数は，特に定められていません。ただし，マクロ名はシンボルのひとつとして登録されますので，ほかのシンボル定義との合計がシンボルの最大登録数を越えることはできません。

④ ある特定のマクロが定義されているかどうかを知るには，#IFDEF/#IFNDEFコントロールを使用します。マクロ名は，#IFDEF/#IFNDEFコントロールのオペランドで指定されるネームの検索対象になります。

#IFDEF/#IFNDEFコントロールのオペランドでマクロ名を指定する場合，マクロ名の先頭に“ % ”を記述する必要はありません。

#IFDEF/#IFNDEFコントロール以外のコントロールまたは，命令のオペランドでマクロ名を参照することはできません。

保守／廃止

（b）マクロ・ボディ

① マクロ・ボディは“（”の次の文字から，対応する“）”の直前文字までになります。コメント内の“）”もマクロ・ボディとして扱われます。

また，マクロ・ボディ（コメントも含みます）内で，“（”または“）”の記述がある場合は，かっこが対応していなければなりません。マクロ・ボディ開始の“（”に対応する“）”が現れたところまでがマクロ・ボディになります。

② マクロ・ボディは，“レーベルまたはネーム”，“機械語命令”，“疑似命令”，“汎用コントロール”，“コメント”で構成されます。

③ マクロ・ボディ内でマクロの参照（マクロのネスティング）はできません。

④ マクロ・ボディ内の“/＊”と“＊/”のコメント記述はマクロ・ボディ内で対応していなくてもかまいません。マクロ・ボディを展開したときに“/＊”と“＊/”が対応するようにしてください。

⑤ マクロ・ボディ内に記述可能な文字数は，アセンブラが実行時に確保可能なメモリ領域の範囲内に限ります。

使用例　マクロ名とマクロ・ボディのみ指定したマクロ定義／参照

```
定義    %DEFINE   (init)    (
                    clr（r0）；
                    r0l = 0x100；
                    rl = r0；
                )

参照            %init
                （展開した結果）
                clr（r0）；
                r0l = 0x100；
                rl = r0；
```

保守／廃止

（c）仮パラメータ・リスト

① 仮パラメータを構成する文字セットおよび構成規則は，シンボルの場合と同様です。シンボルとして定義できる文字列の長さおよび英大文字／小文字の区別はオプションで指定できます。ディフォールト時および“ -S ”を指定したときは1-8文字までを，“ -NS ”を指定したときは1-31文字までを認識できます。

“ -CA ”を指定したときは英大文字／小文字の区別はせず，すべて大文字として解釈されます。ディフォールト時および“ -NCA ”を指定したときは英小文字も区別されます。

仮パラメータは，カンマ“ , ”で区切って16個まで記述できます。カンマの前後の任意の数の空白，タブ，改行文字は無視されます。

仮パラメータとして予約語を記述するとエラーになります。

仮パラメータとして，ユーザ定義シンボルと同一名を用いることはできます。その場合，マクロ内では仮パラメータとして認識します。

注意　仮パラメータが17個以上記述されたとき，あるいは同じ名前の仮パラメータが２個以上指定された場合，エラー・メッセージを出力し，そのマクロ定義に空のマクロ・ボディを登録します。そのマクロが参照された場合，またはマクロ定義時にエラーがあった場合，エラー・メッセージを出力します。

② 仮パラメータは，マクロ・ボディ内でのみ有効で，シンボル・テーブルに登録しません。

③ マクロ・ボディ内に記述された仮パラメータは，そのマクロが参照されたときに実パラメータに置き換えられます。

また，仮パラメータの数と実パラメータの数は同じにしてください。異なる場合は，参照時にワーニング・メッセージを出力します。処理については**8.2.2　マクロの参照**を参照してください。

④ マクロ・ボディ内の引用符“ ’ ”内およびコメント中も，仮パラメータと同じ文字の列があれば，仮パラメータとして認識します。

⑤ シンボル・リストおよびクロスレファレンス・リストに仮パラメータのシンボルは出力されません。また，シンボル・テーブルにも出力されません。

⑥ 仮パラメータはマクロ・ボディの外からは参照できません。

保守／廃止

使用例　仮パラメータを指定したマクロ定義／参照

```
定義　　%DEFINE　（initregs（p0, p1, p2））　（
            clr（r0）；
            r0l = p0；
            clr =（r1）；
            r1l = p1；
            clr =（r2）；
            r2l = p2；
        )

参照　　%initregs（0x100，0x200，0x300）
        （展開した結果）
        clr（r0）；
        r0l = 0x100；
        clr =（r1）；
        r1l = 0x200；
        clr =（r2）；
        r2l = 0x300；
```

（d）ローカル・リスト

① ローカル・リストに記述されたシンボルを，マクロ内ローカル・シンボルとして宣言します。

② ローカル・リストは，ローカル・シンボルをカンマ“，”で区切って，最大64個まで指定できます。65個以上のローカル・シンボルが宣言された場合，エラー・メッセージを出力し，そのマクロ定義を空のマクロ・ボディとして登録します。そのマクロが参照された場合，マクロ定義時にエラーがあったことを示すメッセージを出力し，マクロの展開は行いません。

③ ローカル・リストに記述可能なシンボルの構成規則は，シンボルの場合と同様です。シンボルとして定義できる文字列の長さおよび英大文字／小文字の区別はオプションで指定できます。ディフォールト時および“-S”を指定したときは1-8文字までを，“-NS”を指定したときは1-31文字までを認識できます。

“-CA”を指定したときは，英大文字／小文字の区別をしないで，すべて大文字として解釈します。ディフォールト時および“-NCA”を指定したときは英小文字も区別されます。

ローカル・シンボルとして予約語を記述するとエラーになります。

ローカル・シンボルとして，ユーザ定義シンボルと同一名を用いることはできます。その場合，マクロ内ではローカル・シンボルとして認識します。

1つのローカル・リスト内で同じシンボルが記述された場合，エラー・メッセージを出力します。

④ ローカル・リストに宣言されたシンボルは，シンボル・テーブルに登録されません。よって，ローカル・リストに宣言されたシンボルは，クロスレファレンス・リスト，シンボル・リストに出力されません。

⑤　他のマクロで宣言されたローカル・シンボルと同じ名前のシンボルを，ローカル・リストに記述することによって，別のローカル・シンボルとして使用できます。

⑥　マクロ定義の仮パラメータと同名のシンボルを，同じマクロ定義内のローカル・リストに記述した場合，エラー・メッセージを出力し，ローカル・シンボルと見なさずに仮パラメータとして処理します。

⑦　ローカル・リストに宣言されたシンボルは，そのマクロ・ボディ内で使用されなくてもエラーにはなりません。

⑧　マクロ内シンボルの詳細については，**8.2.4　マクロ内シンボルの有効範囲**を参照してください。

使用例　マクロ内のローカル・シンボル宣言を含むマクロ定義／参照例

```
定義    %DEFINE（wait（waitcnt））LOCAL back
        （      clr（r0）；
               r0l = waitcnt
               clr =（r1）；
               r1l = 0；
        back：
               r1 = r1＋1；
               r2 = r0－r1；
               if（r2＜＝0）jmp back；
        ）

参照           %wait（0x10）
               （展開した結果）
               clr =（r0）；
               r0l = 0x10
               clr（r1）；
               r1l = 0；
        _0_back：
               r1 = r1＋1；
               r2 = r0－r1；
               if（r2＜＝0）jmp _0_back；
```

（e）マクロ定義時のマクロ・ボディ構文チェックは，展開時にソース・プログラムの一部として初めて行われます。マクロ・ボディ中に構文エラーがあるとき，参照した回数だけエラー・メッセージを出力します。

保守／廃止

（ｆ）コンカティネート記号“＠”

① マクロ・ボディ内の“＠”は，文字列を連結する記号です。マクロ展開時には，“＠”の左右の文字を連結し，“＠”は消滅します。

また，区切り記号としても機能し，文字列中の“＠”の前後を仮パラメータ，あるいはローカル・シンボルとして認識できます。マクロ展開時に文字列中の“＠”前後の仮パラメータ，あるいはローカル・シンボルを評価して文字列に連結します。

たとえば，仮パラメータP1をマクロ・ボディ中に，“R@P1@H”のように指定するとき，マクロ参照のP1の実パラメータに０を与えると，“R0H”に展開されます。

② “＠”の機能は，マクロ・ボディ内のみ有効です。

③ 引用符で囲まれた文字列の中およびコメント中に“＠”を記述しても，区切り文字として扱われます。

④ 連結された結果が仮パラメータと一致しても，仮パラメータとして認識しません。

⑤ 連結された結果が，ローカル・シンボルと一致しても，ローカル・シンボルとして認識しません。

使用例　マクロ・ボディ内に記述する仮パラメータを“＠”によって，区切ったとき。

```
定義    %DEFINE   (init（p0，p1）)   (
                r@p0@l = 0x100；
                r@p1 = r1；
        )

参照            %init（0，1）
                （展開した結果）
                r0l = 0x100；
                r1 = r1；
```

保守／廃止

8.2.2　マクロの参照

（1）%マクロ名

[記述形式]

シンボル欄	ニモニック欄	オペランド欄
[レーベル：]	%マクロ名	[（実パラメータ・リスト）]

詳細な記述形式を示す構文については**3.2.4　マクロ命令ステートメント**を参照してください。

[機　　能]

%DEFINE文で定義されたマクロ・ボディを参照します。

参照されるとマクロ・ボディ内の仮パラメータをすべて対応する実パラメータに置き換えて展開します（文字列の置き換え）。

[説　　明]

・マクロ名は%DEFINE文で定義された“マクロ名”で，参照する以前に定義されているものでなければなりません。

・実パラメータの規則は次のとおりです。

① 実パラメータはストリング（文字の並び）として認識され，カンマ“，”で区切って16個まで指定できます。

ユーザ定義シンボル，予約語の記述が可能です。

② 区切り記号のカンマ，“，”の前後にある空白，タブ，改行文字は無視されます。

また，カンマではなく，任意数の空白，タブ，改行文字によって，区切られている場合，任意数の空白，タブ，改行文字をスキップし，区切られている実パラメータの文字列は１つの文字列として，連結して展開します。

③ 実パラメータ中に記述可能な文字は，“＠”と“，”を除いた言語文字です。空白，タブ，改行文字はスキップされます。

また，“？”は実パラメータの２文字目以降から記述できます。記述できない文字が存在した場合，または先頭文字が“？”の場合，ワーニング・メッセージを出力します。その文字を無視し，その文字の前後の文字列を連結した結果を展開します。

・仮パラメータから実パラメータへの置き換えは，それぞれの記述順に対応して左から順に行います。仮パラメータと実パラメータの数が一致しない場合，ワーニング・メッセージを出力します。実際の処理は次のとおりです。

仮パラメータの数をN，実パラメータの数をMとします。

N = Mの場合…１：１に対応します。

N＜Mの場合…余分な実パラメータを無視し，ワーニング・メッセージを出力します。

N＞Mの場合…不足の実パラメータに対する仮パラメータを展開せずに，ワーニング・メッセージを出力します。

・実パラメータの先頭あるいは最後に書かれたカンマは無視します。また，実パラメータの途中で２つ以上続いて書かれたカンマは，１つのカンマと認識します。

使用例　8.2.1　マクロの定義の使用例を参照してください。

8.2.3　マクロの展開

マクロを使用したソース・プログラムをアセンブルすると，アセンブラは，次のように動作します。使用例については**8.2.1　マクロの定義の使用例**を参照してください。

・マクロの参照を見つけると，それに対応するマクロ・ボディをマクロ名参照行の次行から展開します。そのとき，仮パラメータを実パラメータに置き換えます。展開されるリスト・イメージは，実パラメータを出力します。

・展開したステートメントをほかのステートメントと同様にアセンブルします。

保守／廃止

8.2.4　マクロ内シンボルの有効範囲

マクロ内で定義するシンボルには，グローバル・シンボルとローカル・シンボルの２種類があります。

（1）グローバル・シンボル

ソース・プログラム内のすべてのステートメントから参照できます。したがって，グローバル・シンボルが定義されているマクロを２回以上参照し，一連のステートメントが展開されると，シンボルは二重定義エラーになります。

（2）ローカル・シンボル

ローカル・シンボルは，%DEFINE文内のLOCAL宣言で宣言します。マクロ内でシンボルをLOCAL宣言なしで定義した場合，そのシンボルはグローバル・シンボルになります。LOCAL宣言したシンボルは，LOCAL宣言したマクロ内でだけ参照することができます。

マクロ・ボディ内のローカル・シンボルは，展開されるたびに，“ _n_ユーザ定義シンボル名 ”（n = 0－0xFFF，）というグローバル・シンボルに自動的に置き換えられます（nは展開されるたびに16進数でインクリメント（＋１）されます。シンボル最大処理数（最大1200個）から，nが0xFFFを越えることはありません）。

展開時に自動生成した“ _n_ユーザ定義シンボル名 ”というシンボルは，グローバル・シンボルと同じ扱いになります。同じマクロが２回以上参照され，一連のステートメントが生成されても，シンボルは二重定義エラーになりません。

保守／廃止

使用例　マクロ・ボディ内のローカル・シンボルの定義／参照

```
定義    %DEFINE  (wait)  LOCAL back
            (           r1l = 0；
              back：
                  r1 = r1＋1；
                  r2 = r0－r1；
                  if（r0＜＝0）jmp back；
        )

参照    r0l = 0x10；
        %wait
        r0l = 0x20；
        %wait
        （展開した結果）

        r0l = 0x10；
        r1l = 0；
        _0_back：
                r1 = r1＋1；
                r2 = r0－r1；
                if（r0＜＝0）jmp_0_back；
                r0l = 0x20；
                r1l = 0；
        _1_back：
                r1 = r1＋1；
                r2 = r0－r1；
                if（r0＜＝0）jmp_1_back；
```

この例で，“ back ”をLOCALシンボルとして宣言しない場合は，２回の参照でそれぞれ“ back： ”というレーベルが定義されてしまうため，エラーになります。このエラーを防止するために，“ back ”をLOCAL文中でローカル・シンボルとして宣言し，参照するたびに別名のレーベルが定義されるようにしています。

付録A　構 文 図

μPD77016ファミリ・アセンブリ言語の構文図を次に示します。

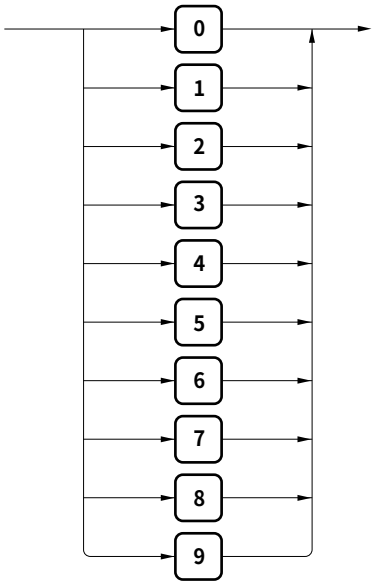
記述規則

□ または □ ：文字を示します。

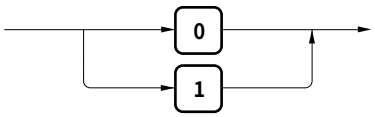
□ ：さらに別の箇所で定義されています。

保守／廃止

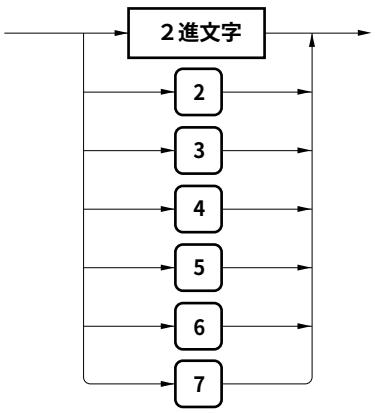
数字



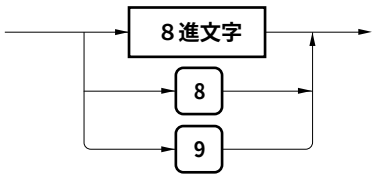
2進文字



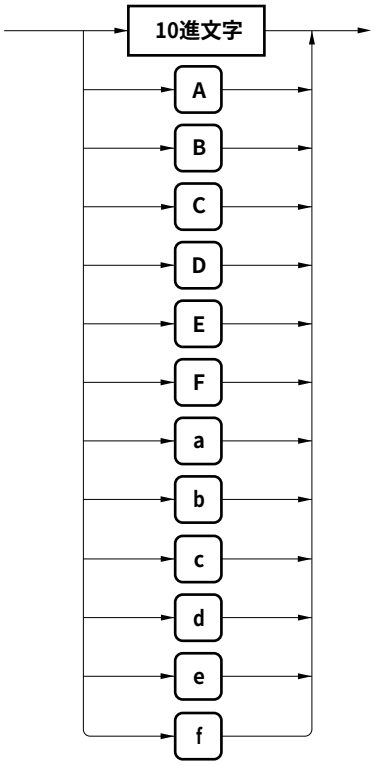
8進文字



10進文字

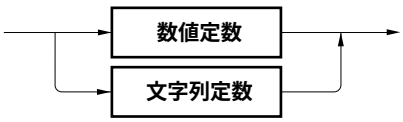


16進文字

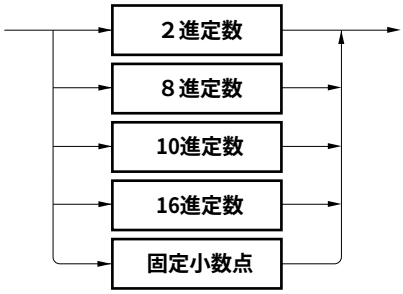


保守／廃止

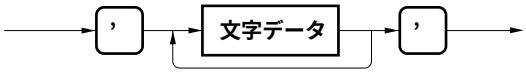
定数



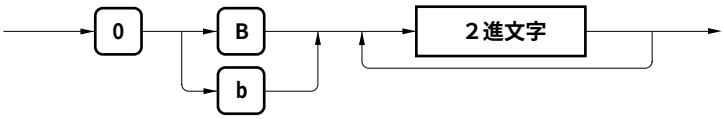
数値定数



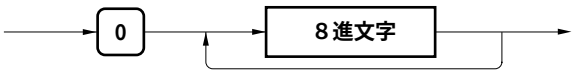
文字列定数



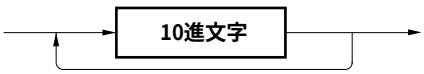
２進定数



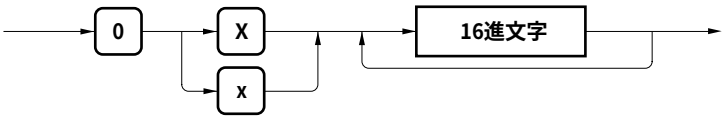
８進定数



10進定数

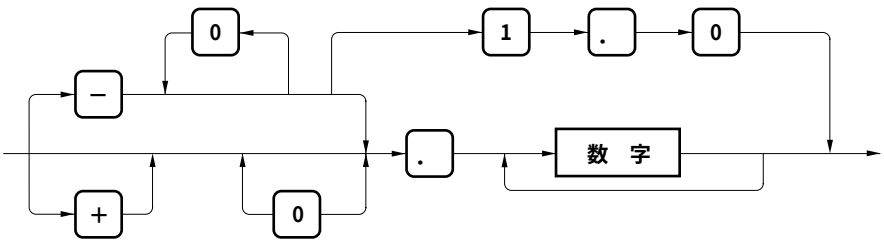


16進定数



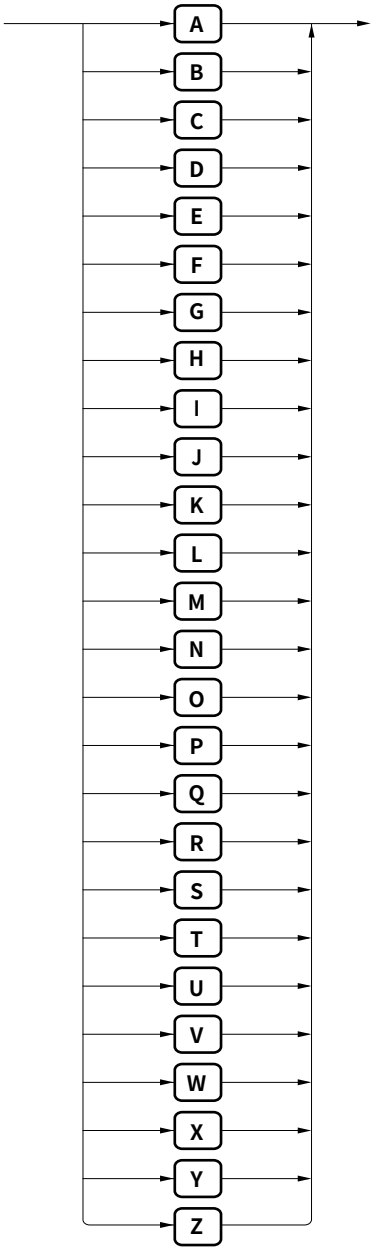
注意　１桁の場合は数字列の先頭に０を記述することができます。２桁以上の場合は先頭に０を記述することはできません。

固定小数点定数

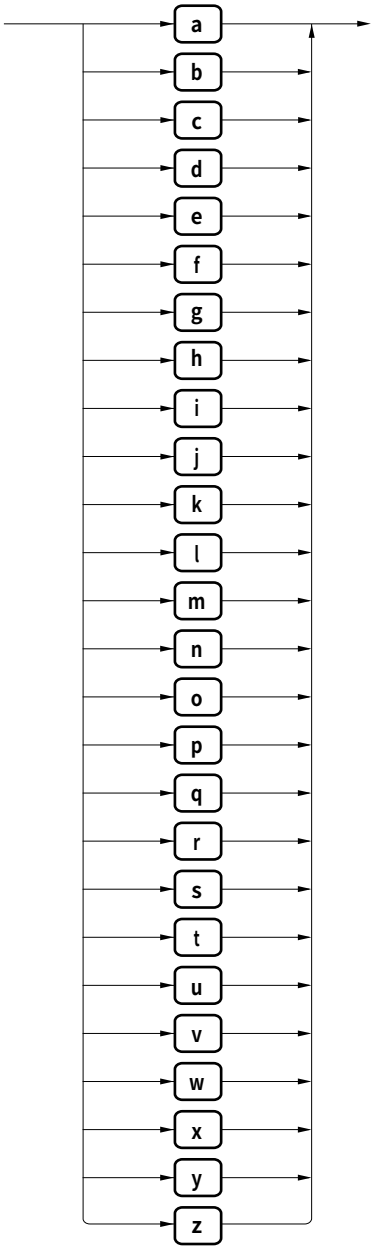


保守／廃止

英大文字

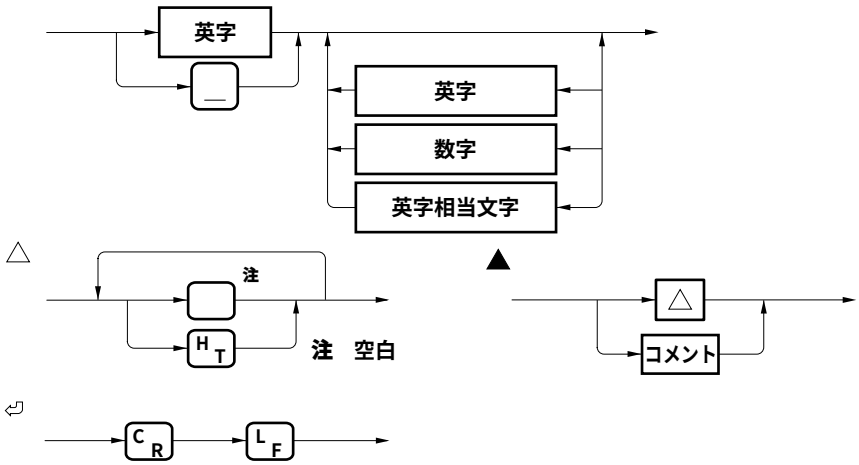


英小文字

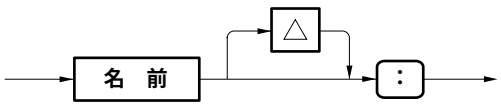


保守／廃止

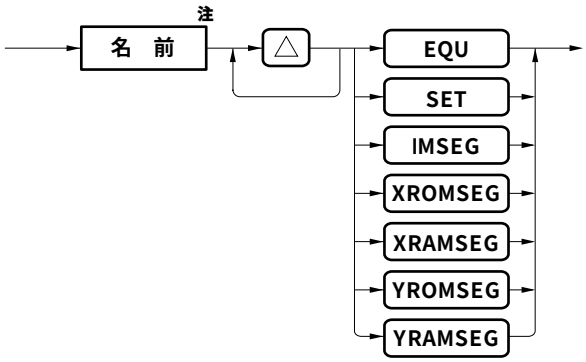
名前



レーベル定義

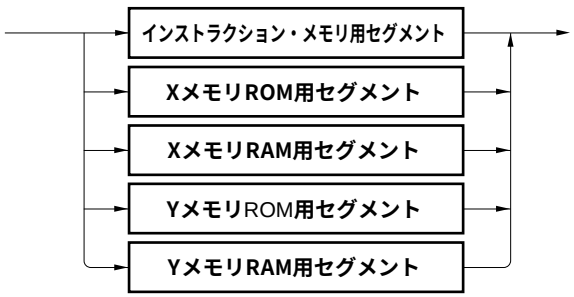


ネーム定義

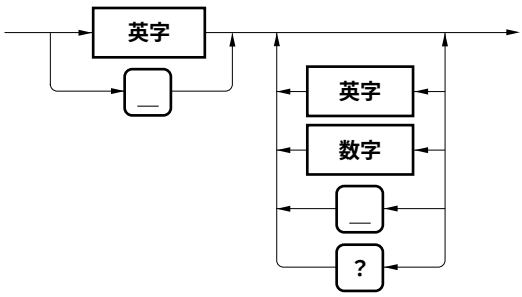


注　リテラル・ネームについては**7.9　リテラル・ネーム指定**を参照してください。マクロ名については**3.2.4　マクロ命令ステートメント**を参照してください。

セグメント

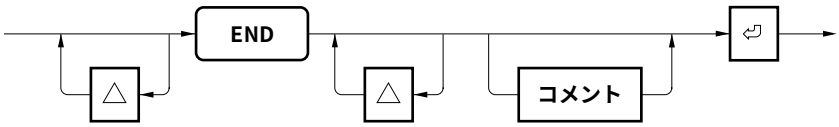


モジュール名

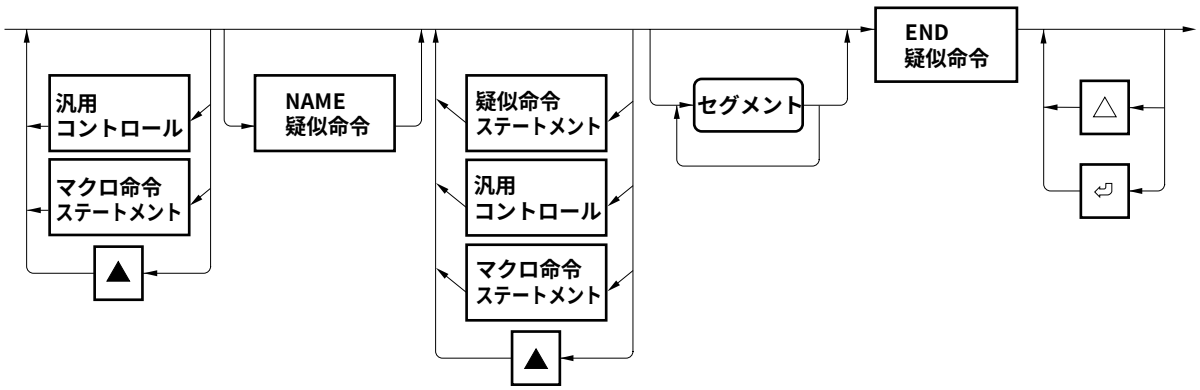


保守／廃止

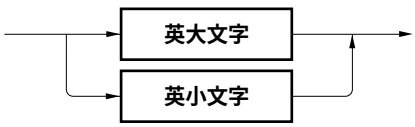
アセンブル終了疑似命令ステートメント



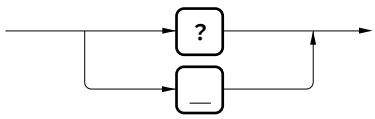
ソース・モジュール



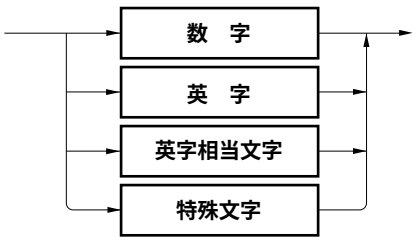
英字



英字相当文字

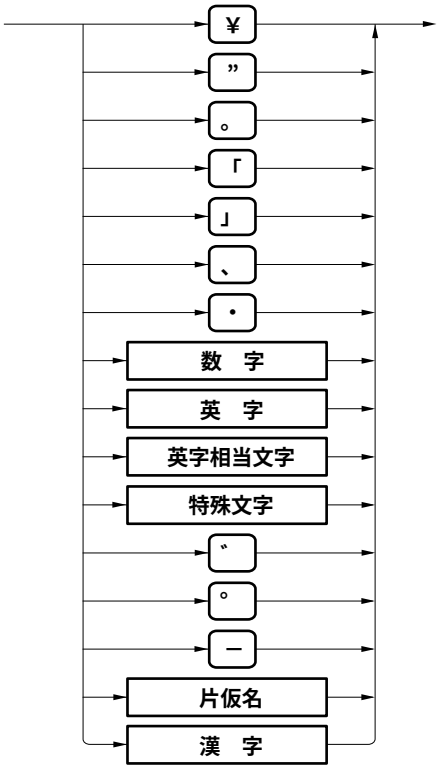


言語文字

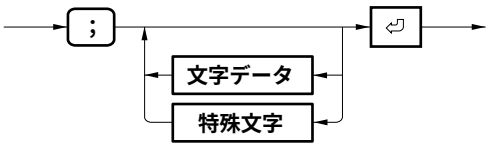


保守／廃止

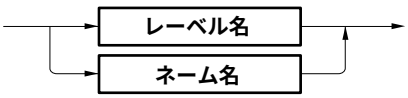
文字データ



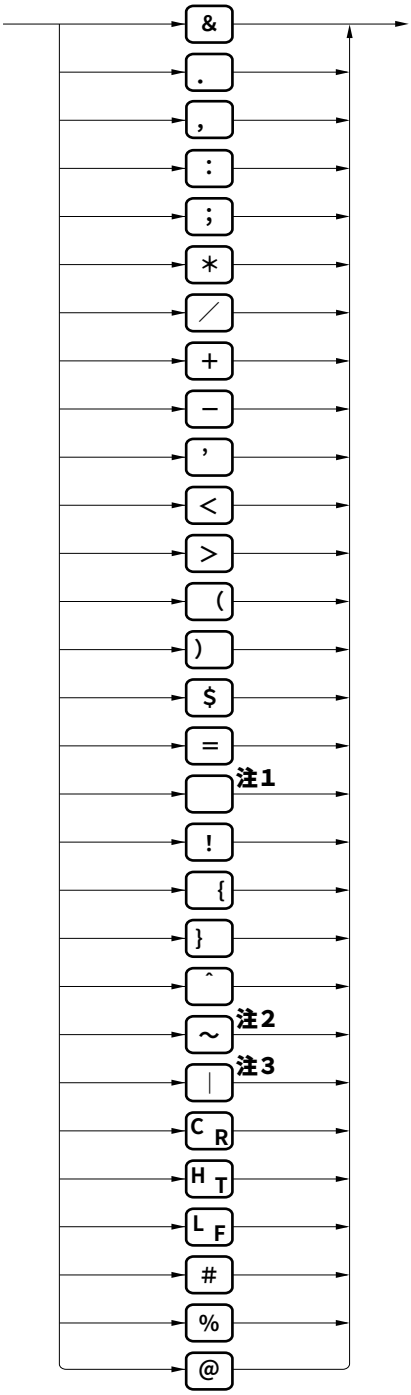
コメント



シンボル名



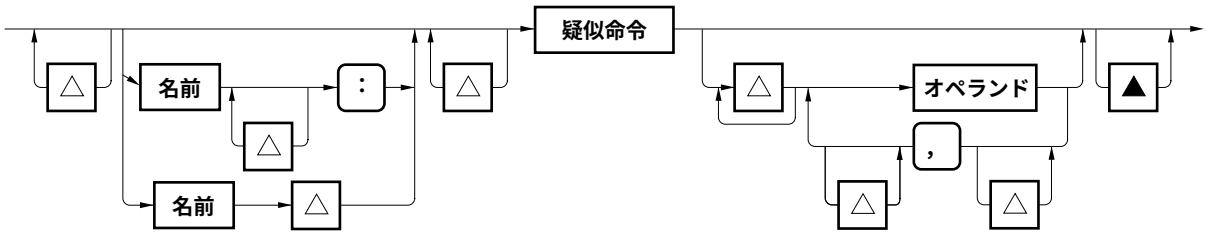
特殊文字



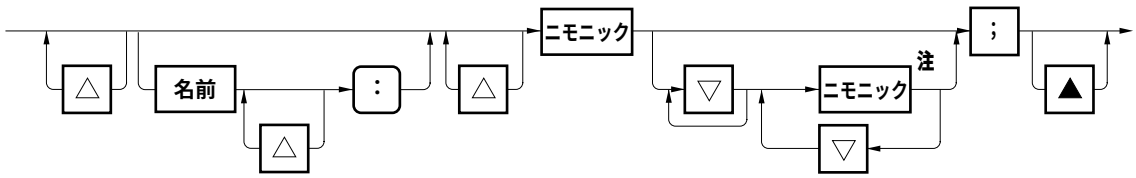
注1．空白
　2．ティルド
　3．縦棒

保守／廃止

疑似命令ステートメント



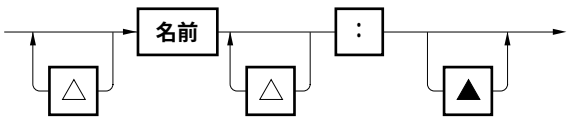
アセンブリ命令ステートメント



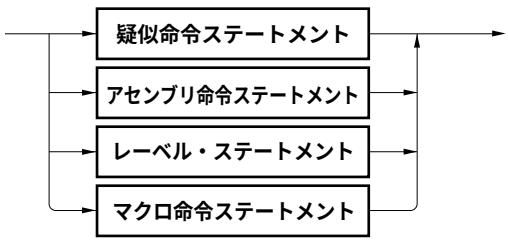
備考　▽は１個以上の空白，⏎を示します。

注　同時記述可能なもの

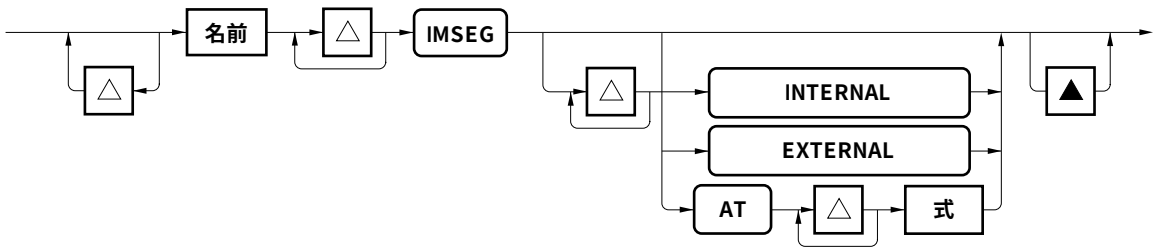
レーベル・ステートメント



ステートメント

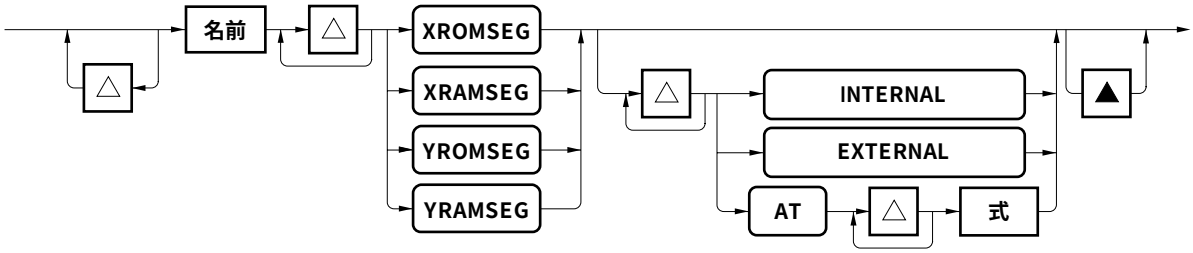


インストラクション・メモリ用セグメント開始疑似命令ステートメント

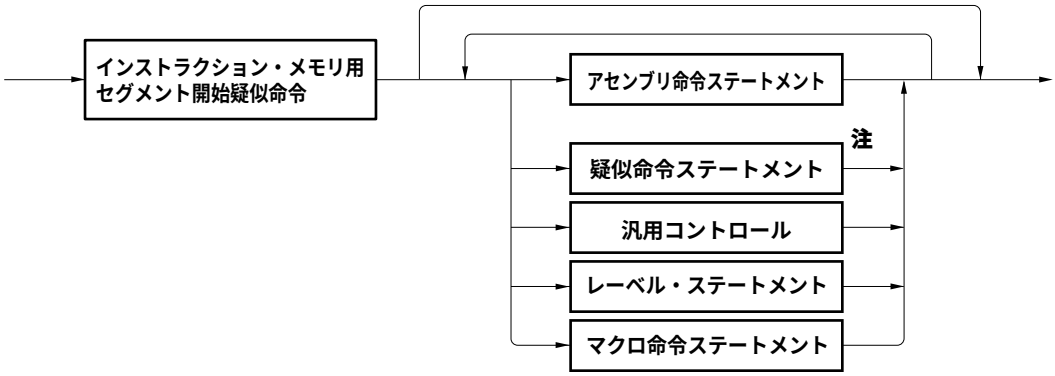


保守／廃止

データROM，データRAM（X/Y）用セグメント開始疑似命令ステートメント

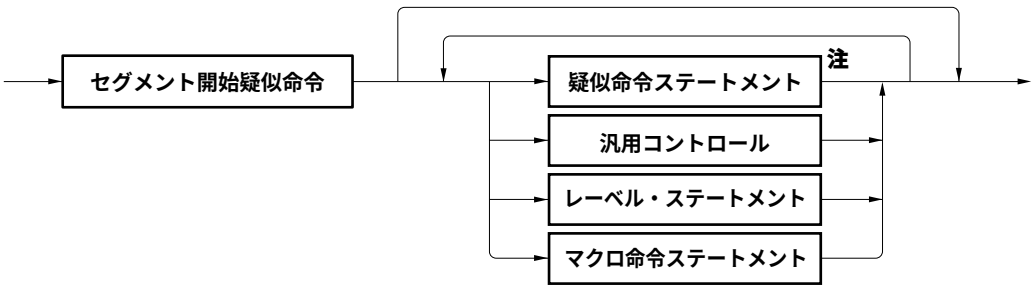


インストラクション・メモリ用セグメント



注　記述可能な疑似命令については**3.1.2　セグメント部**を参照してください。

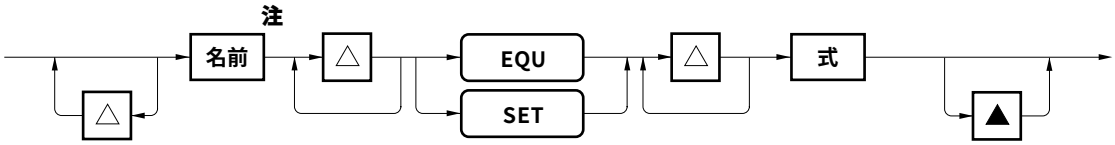
データROM，データRAM（X/Y）用セグメント



注　記述可能な疑似命令については**3.1.2　セグメント部**を参照してください。

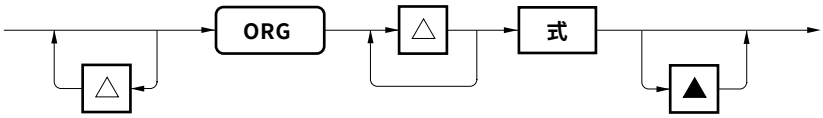
保守／廃止

シンボル定義疑似命令ステートメント

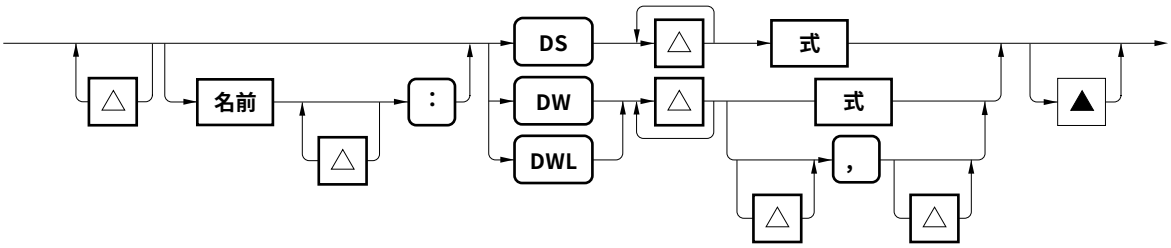


注　リテラル・ネームについては**7.9　リテラル・ネーム指定**を参照してください。
　　マクロ名については**3.2.4　マクロ命令ステートメント**を参照してください。

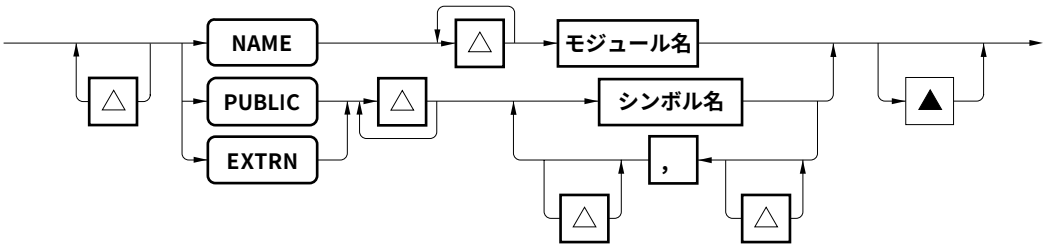
ロケーション・カウンタ制御疑似命令



領域確保疑似命令ステートメント



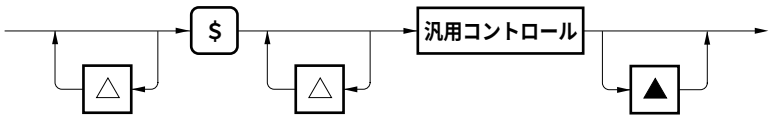
プログラム・リンケージ疑似命令ステートメント



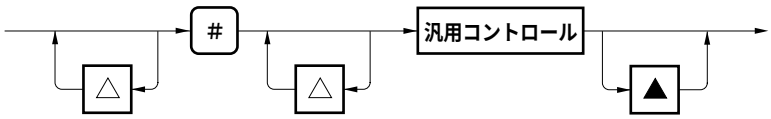
保守／廃止

汎用コントロール

①　リスト制御

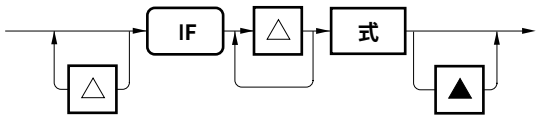


②　条件付きアセンブル制御，インクルード・ファイル指定，リテラル・ネーム指定

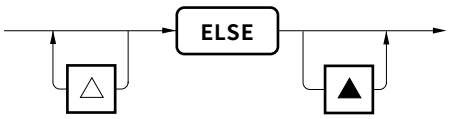


条件付きアセンブル制御汎用コントロール

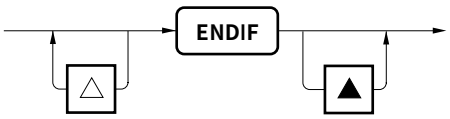
①　IFコントロール



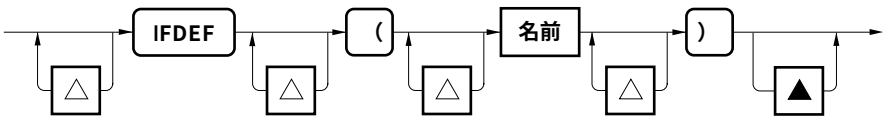
②　ELSEコントロール



③　ENDIFコントロール

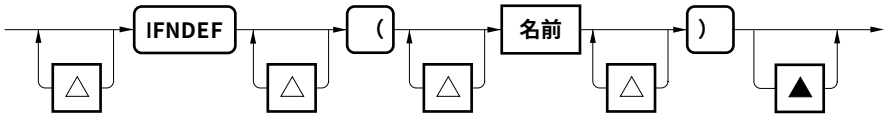


④　IFDEFコントロール

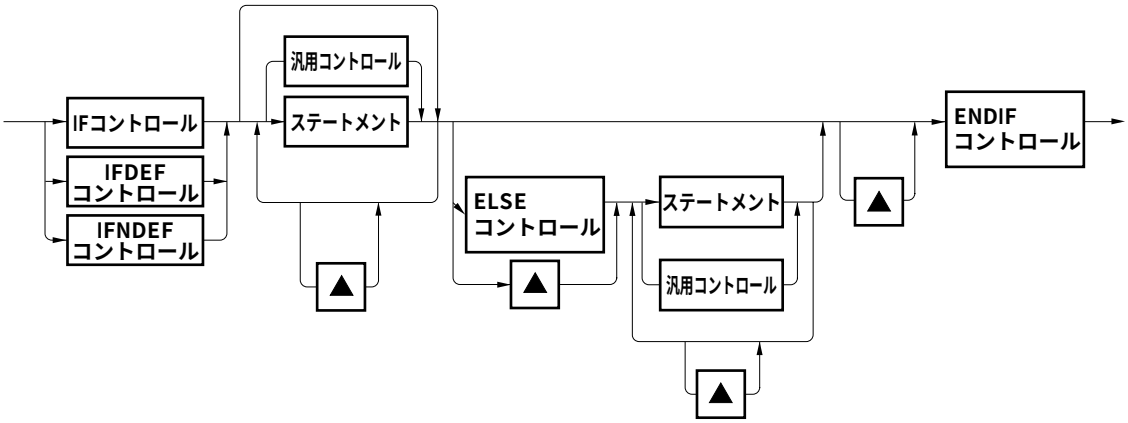


保守／廃止

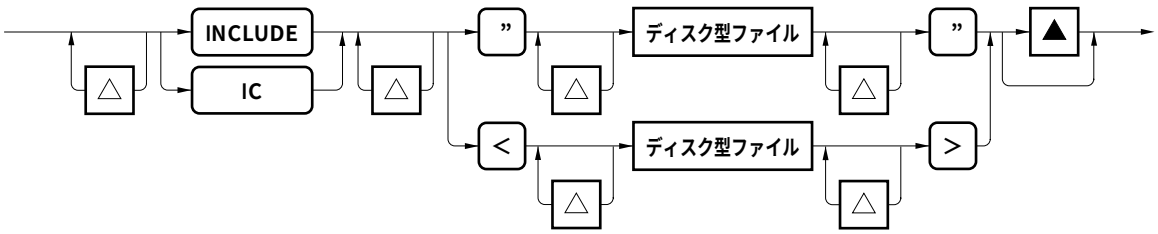
⑤ IFNDEFコントロール



アセンブル制御部

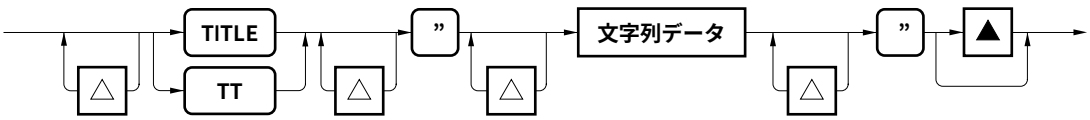


インクルード・ファイル指定汎用コントロール



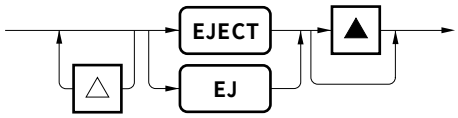
リスト制御汎用コントロール

① TITLEコントロール

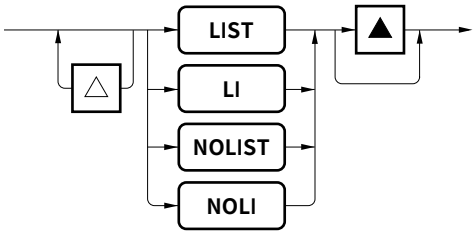


保守／廃止

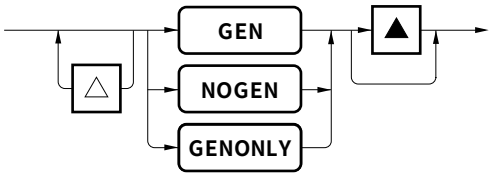
② EJECTコントロール



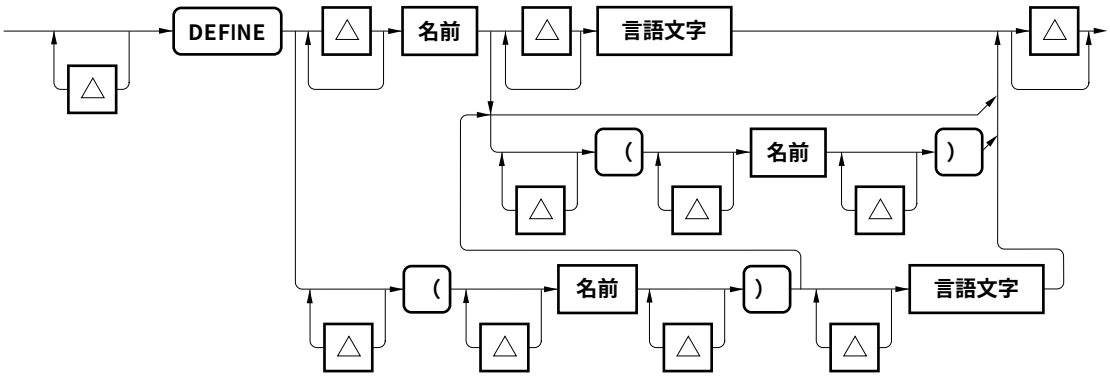
③ LISTコントロール



④ GENコントロール



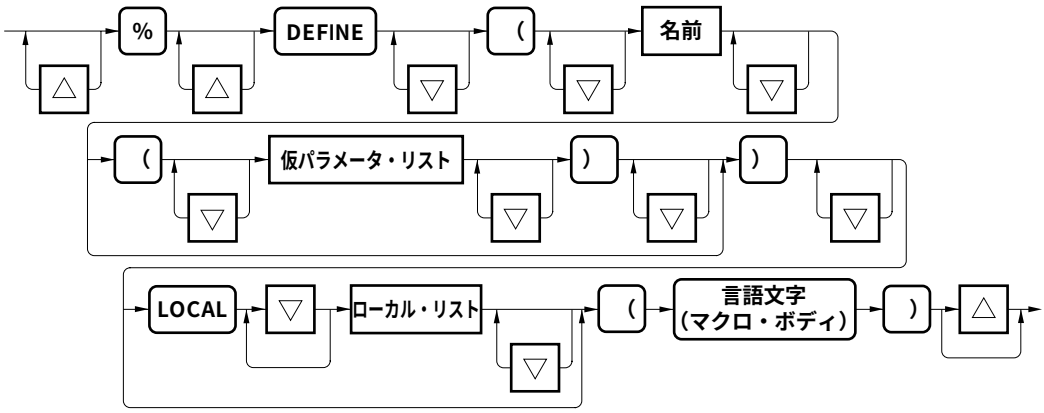
リテラル・ネーム指定汎用コントロール



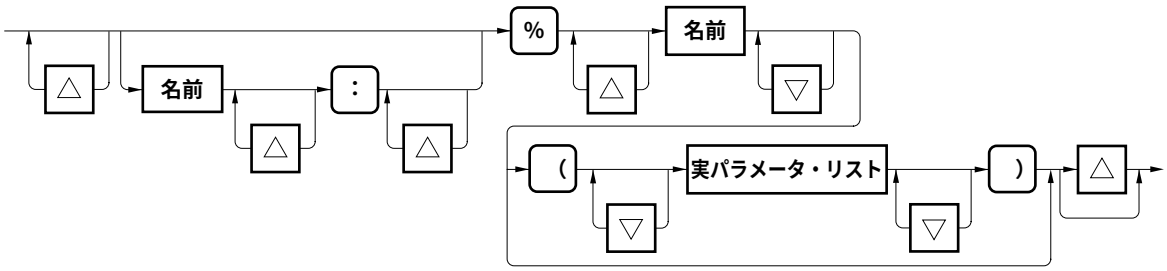
保守／廃止

マクロ命令ステートメント

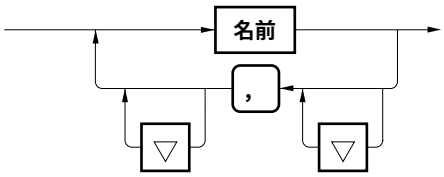
①　マクロ定義



②　マクロ参照

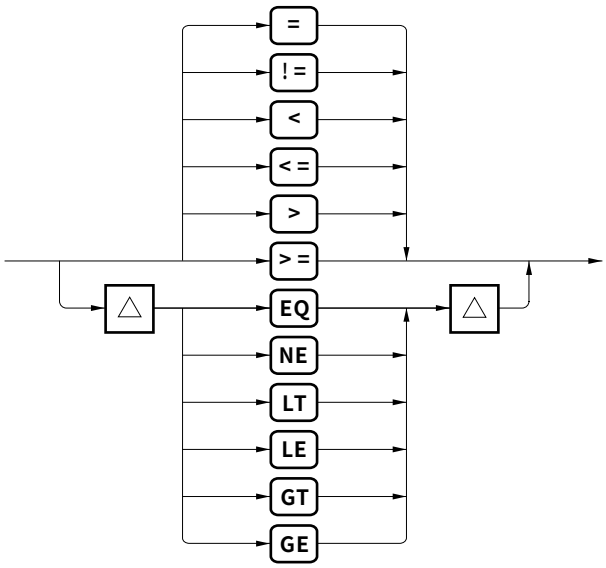


③　仮パラメータ・リスト／ローカル・リスト／実パラメータ・リスト

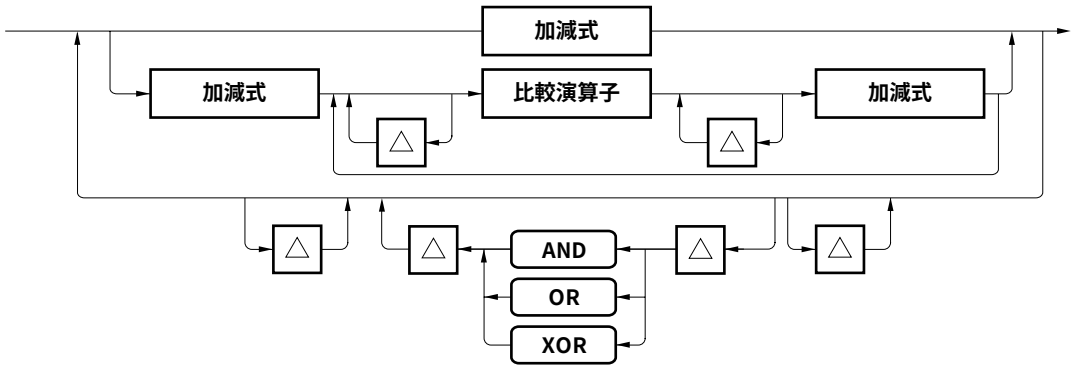


保守／廃止

比較演算子

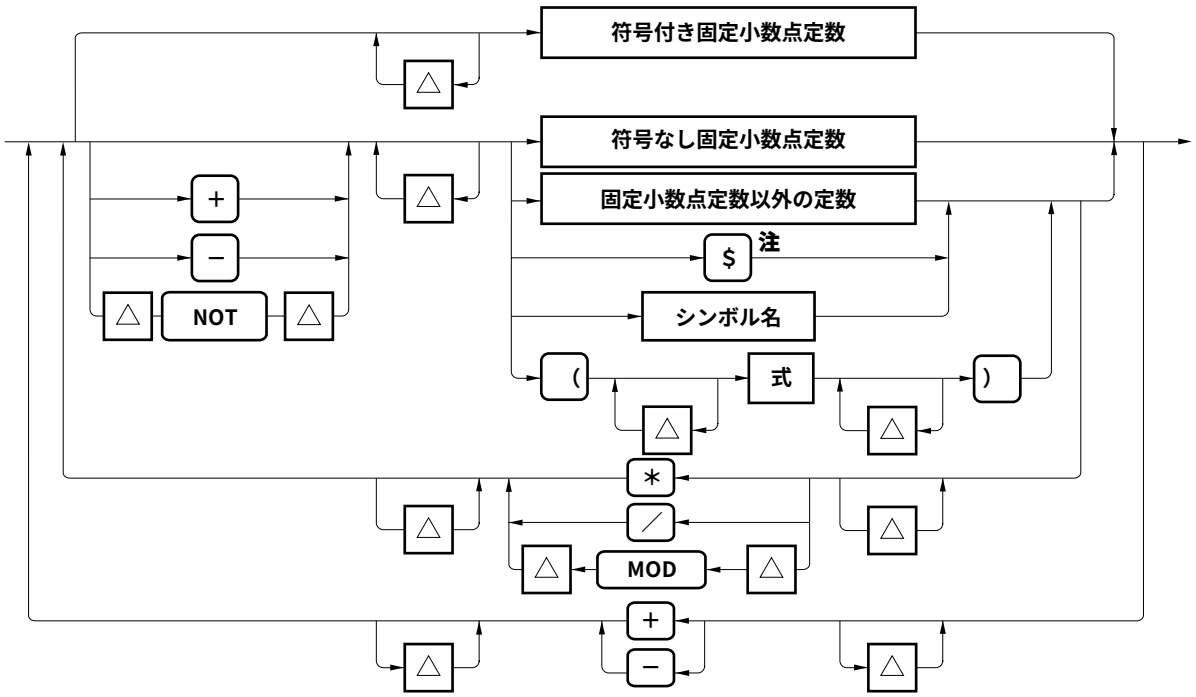


式



保守／廃止

加減式



注　ロケーション・カウンタ値

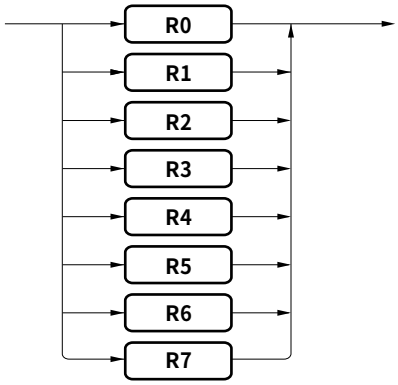
注意　演算子（＋，－）と符号（＋，－）の組み合わせで，“＋＋”，“－－”になった場合はフェイタル・エラーとなります。

保守／廃止

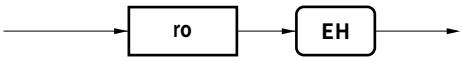
アセンブリ命令ステートメント

汎用レジスタ

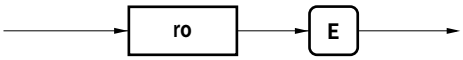
① ro



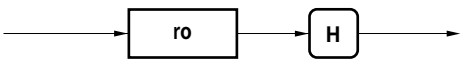
② reh



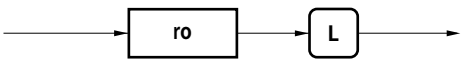
③ re



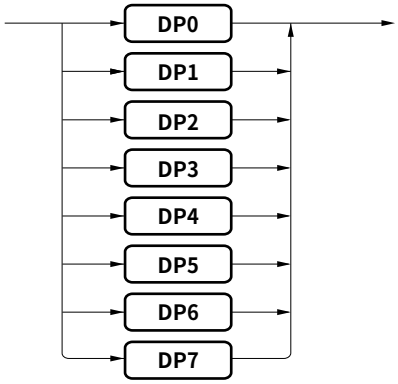
④ rh



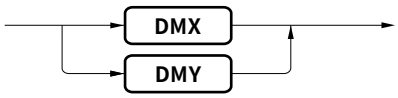
⑤ rl



データ・ポインタ・レジスタ

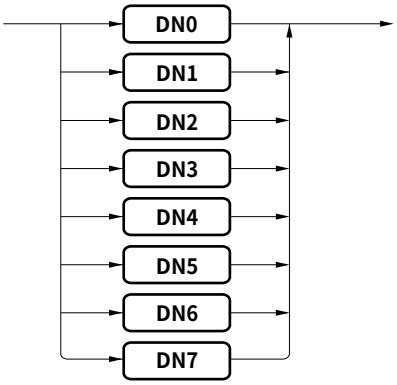


モジュロ・レジスタ

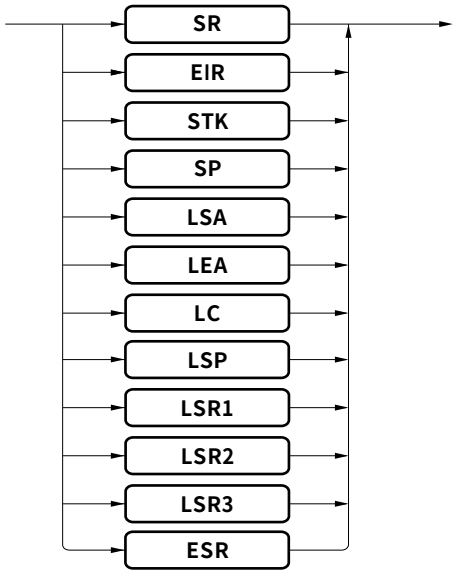


保守／廃止

インデクス・レジスタ

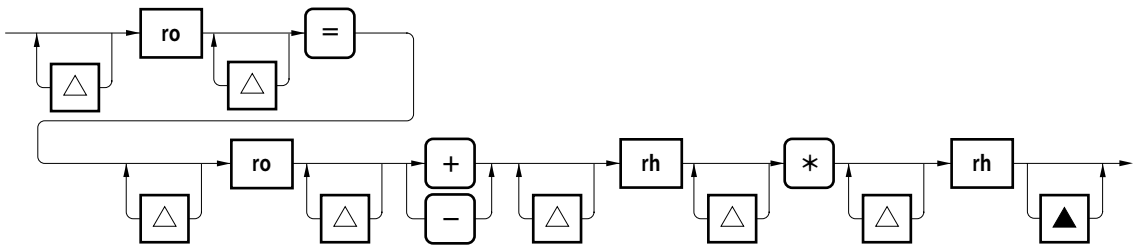


そのほかのレジスタ

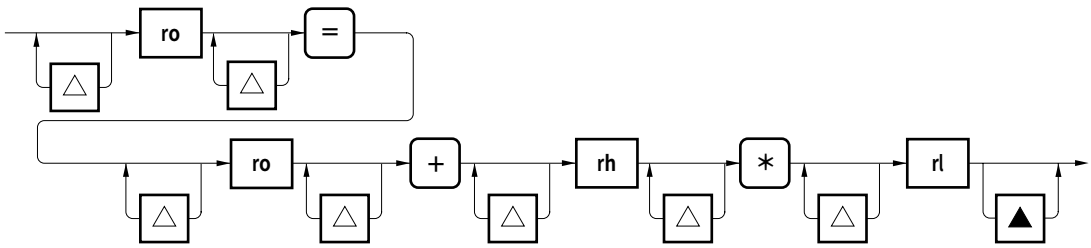


3項演算命令

① マルチプライ・アド，マルチプライ・サブ

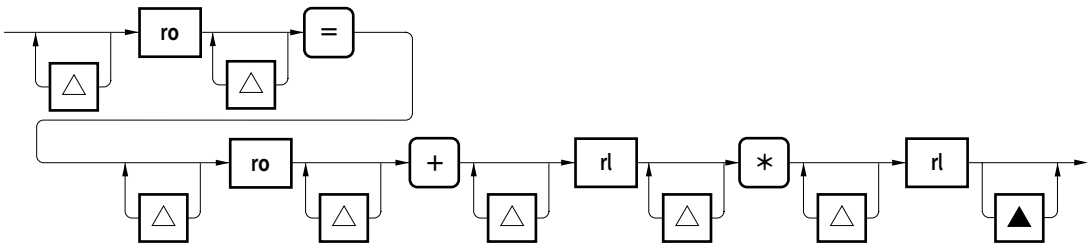


② サイン・アンサイン・マルチプライ・アド

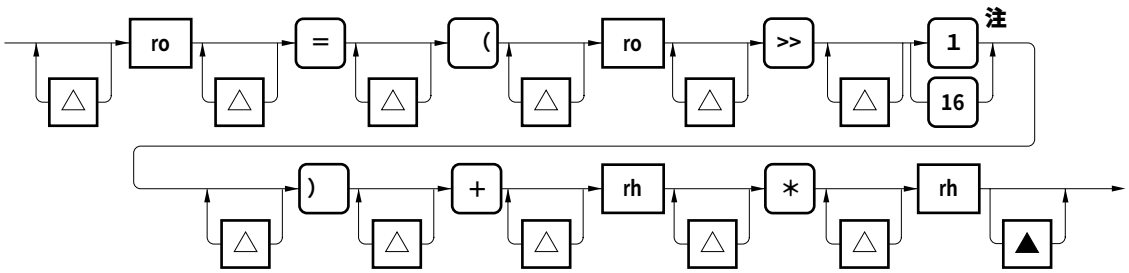


保守／廃止

③　アンサイン・アンサイン・マルチプライ・アド



④　1ビット・シフト・マルチプライ・アド，16ビット・シフト・マルチプライ・アド

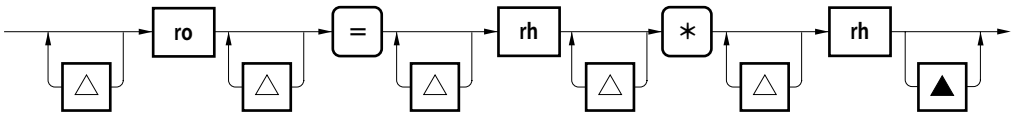


注　2進数，8進数，10進数，16進数のいずれも記述することができます。

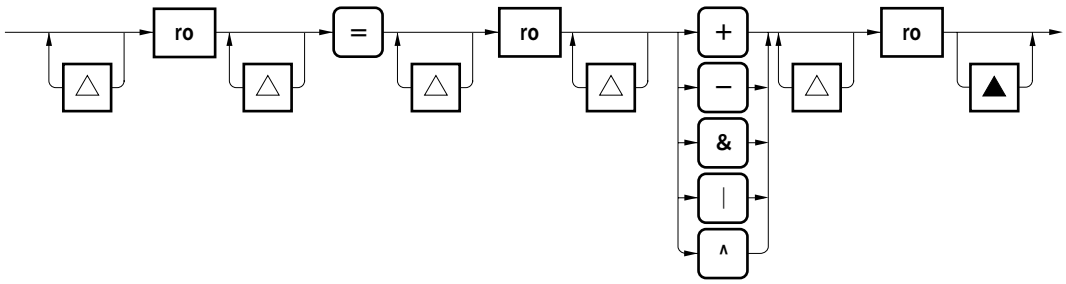
保守／廃止

2項演算命令

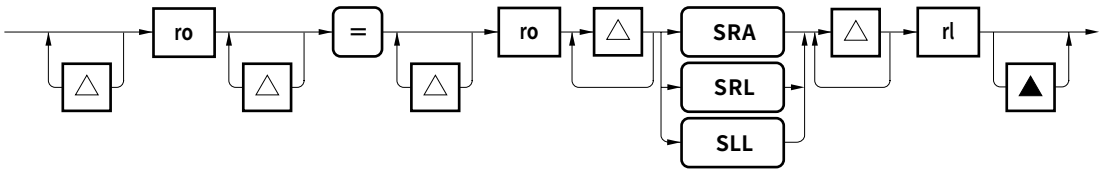
①　マルチプライ



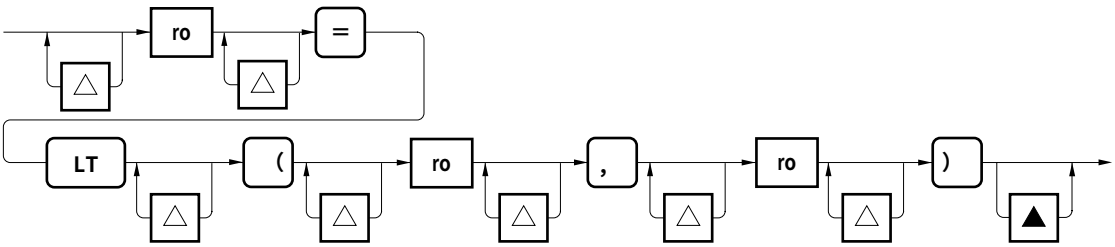
②　アド，サブ，アンド，オア，イクスクルーシブ・オア



③　算術右シフト，論理右シフト，論理左シフト

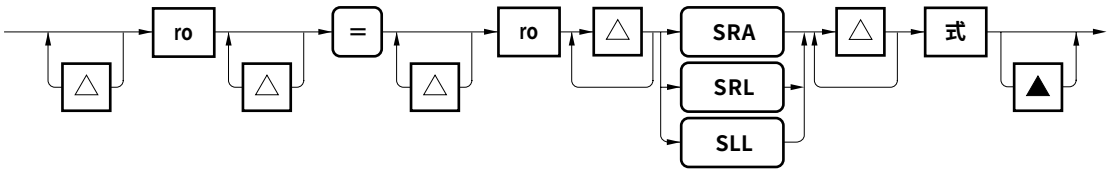


④　レスザン

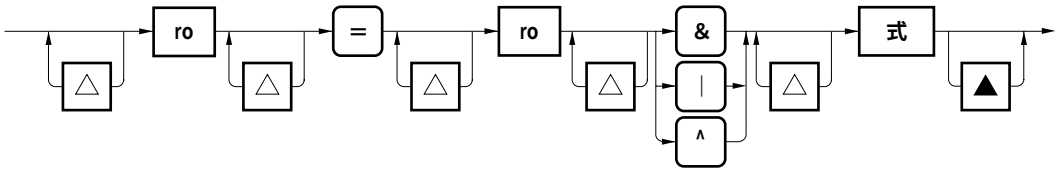


保守／廃止

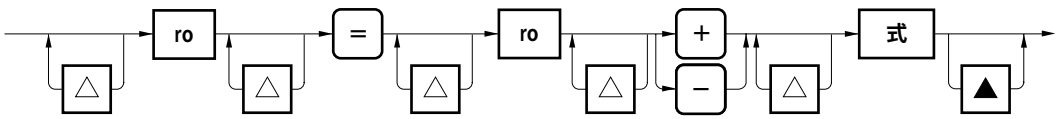
⑤ イミーディエト算術右シフト，イミーディエト論理右シフト，イミーディエト論理左シフト



⑥ イミーディエト・アンド，イミーディエト・オア，イミーディエト・イクスクルーシブ・オア



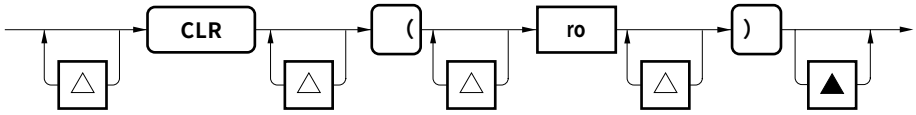
⑦ イミーディエト・アド，イミーディエト・サブ



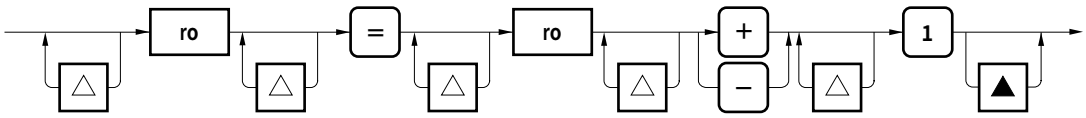
保守／廃止

単項演算命令

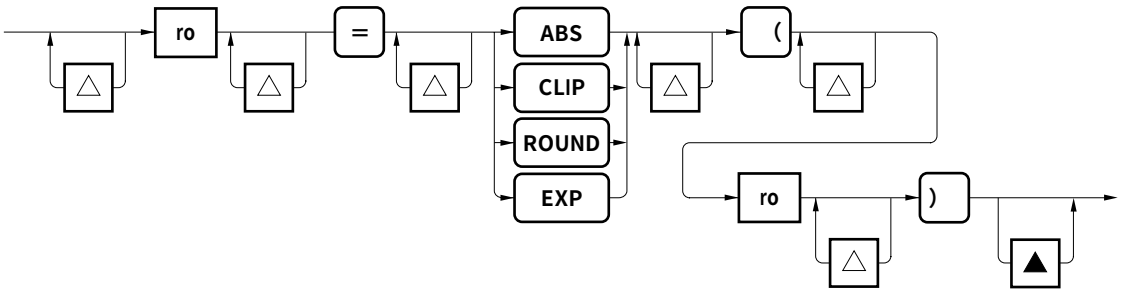
①　クリア



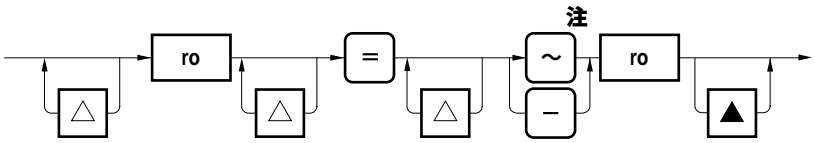
②　インクリメント，デクリメント



③　絶対値，クリップ，丸め，指数

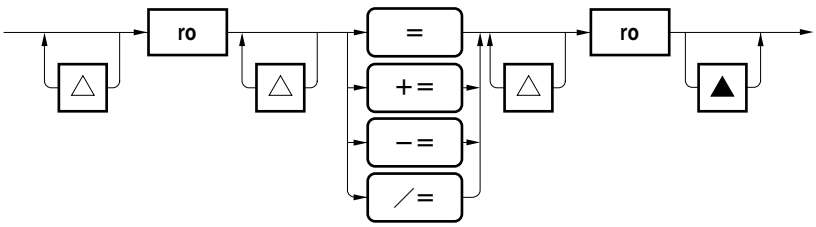


④　１の補数，２の補数



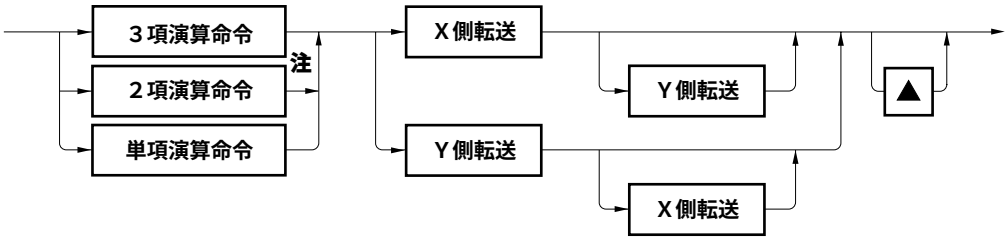
注　ティルド

⑤　代入，累加算，累減算，除算



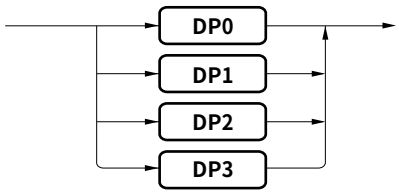
保守／廃止

並列ロード／ストア命令

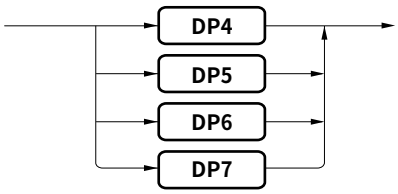


注　イミーディエト値を使用しないもの。

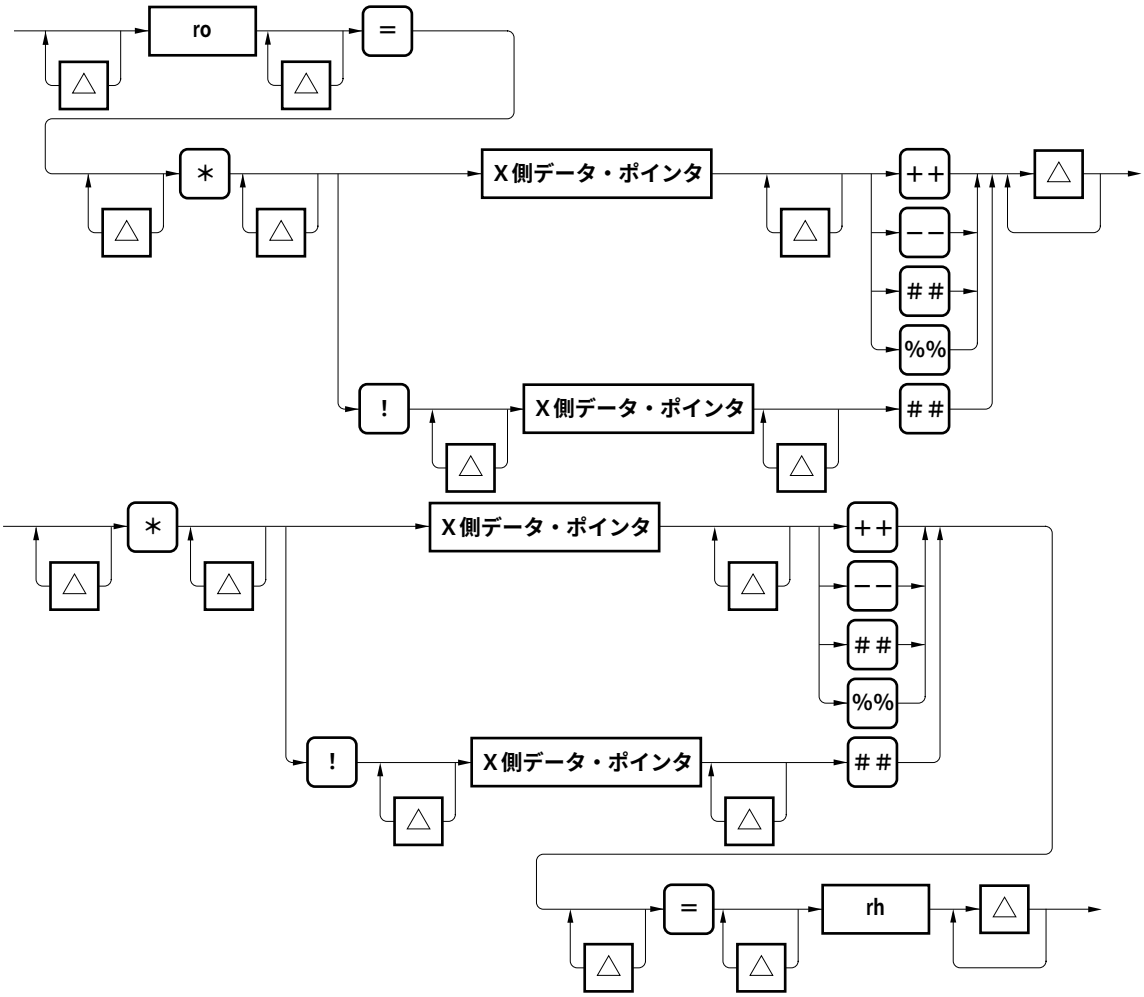
X側データ・ポインタ・レジスタ



Y側データ・ポインタ・レジスタ

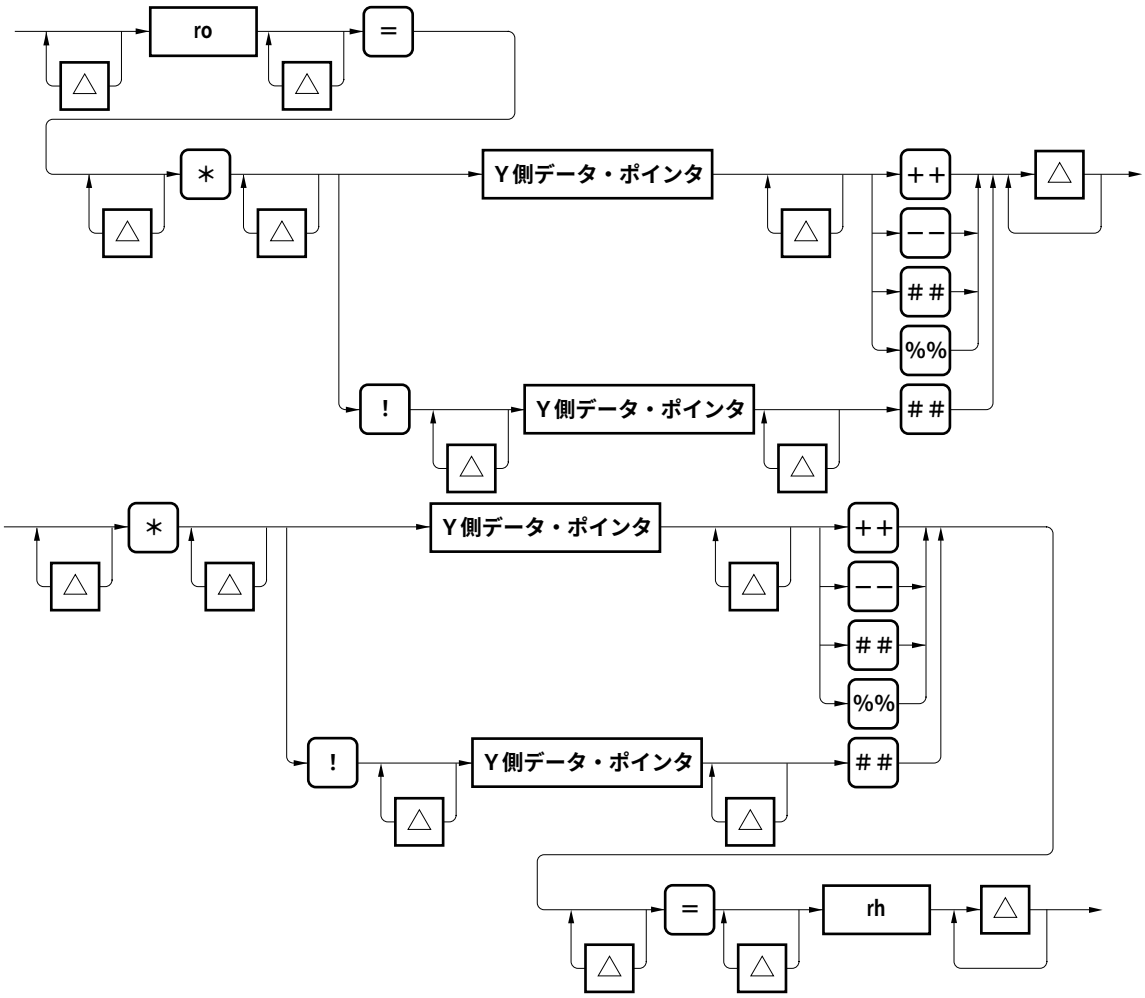


①　X側転送



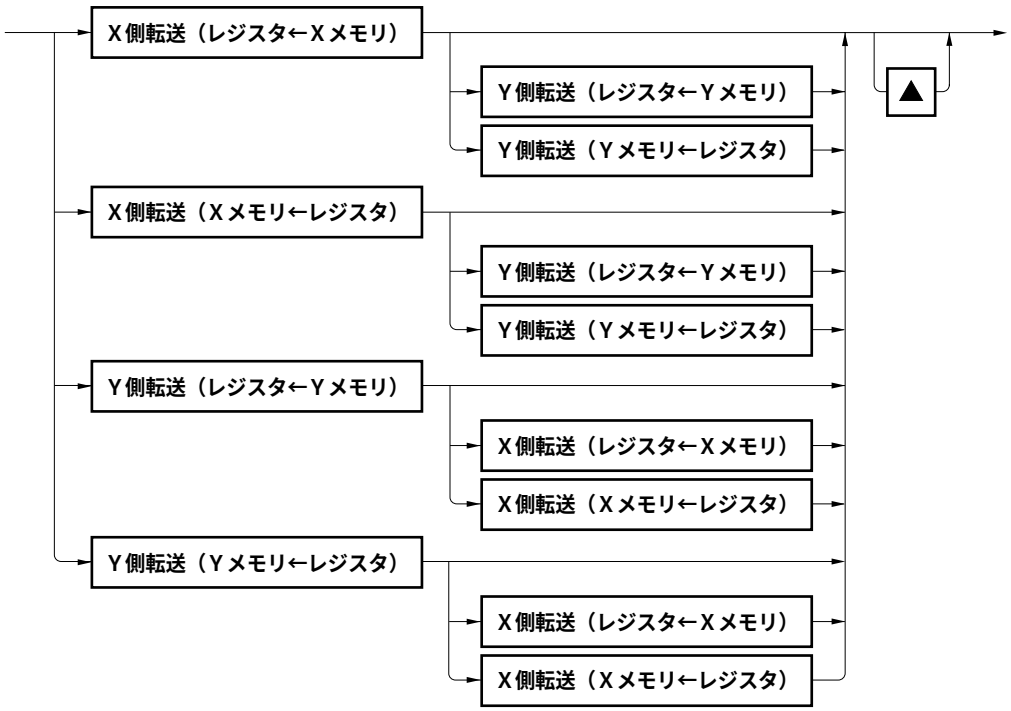
保守／廃止

② Y側転送

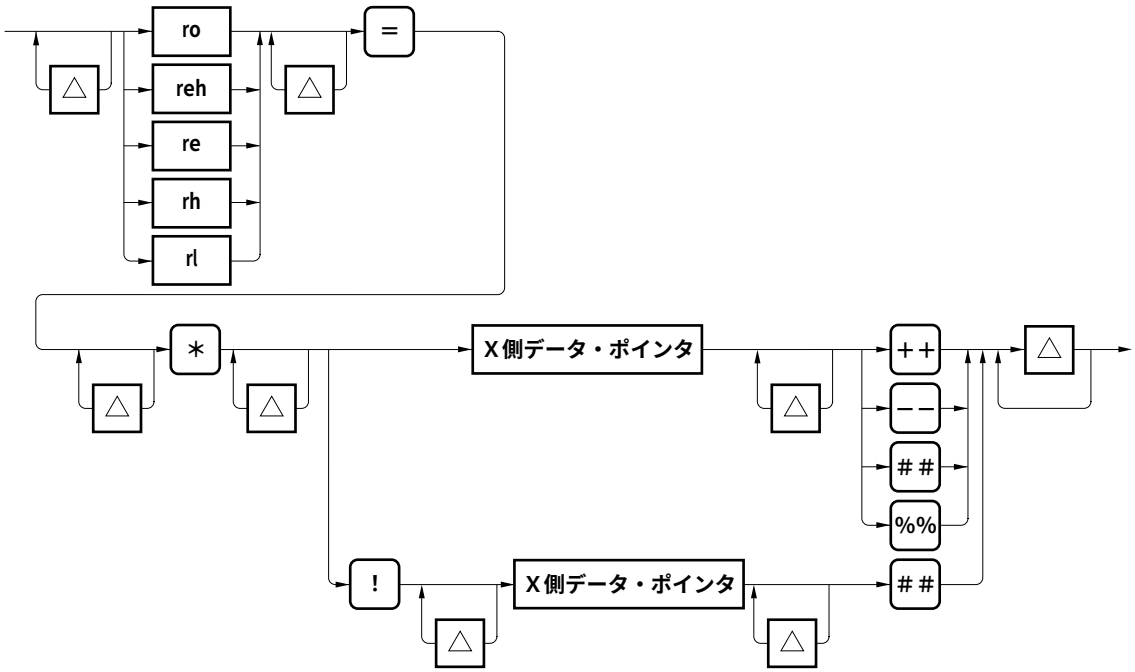


保守／廃止

部分ロード／ストア命令

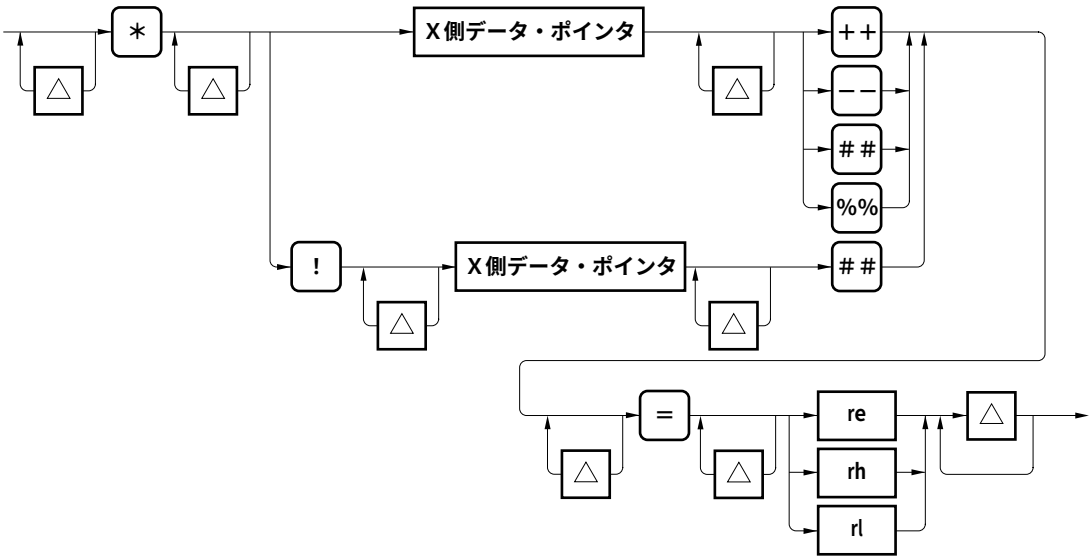


①　X側転送（レジスタ←Xメモリ）

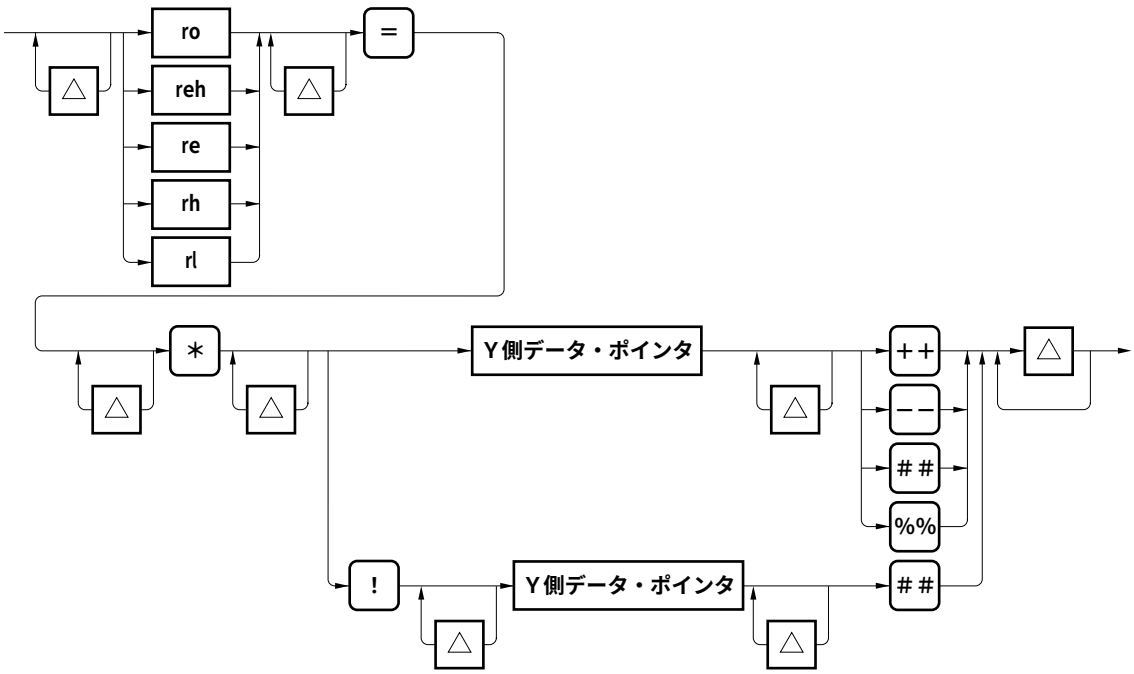


保守／廃止

② X側転送（Xメモリ←レジスタ）

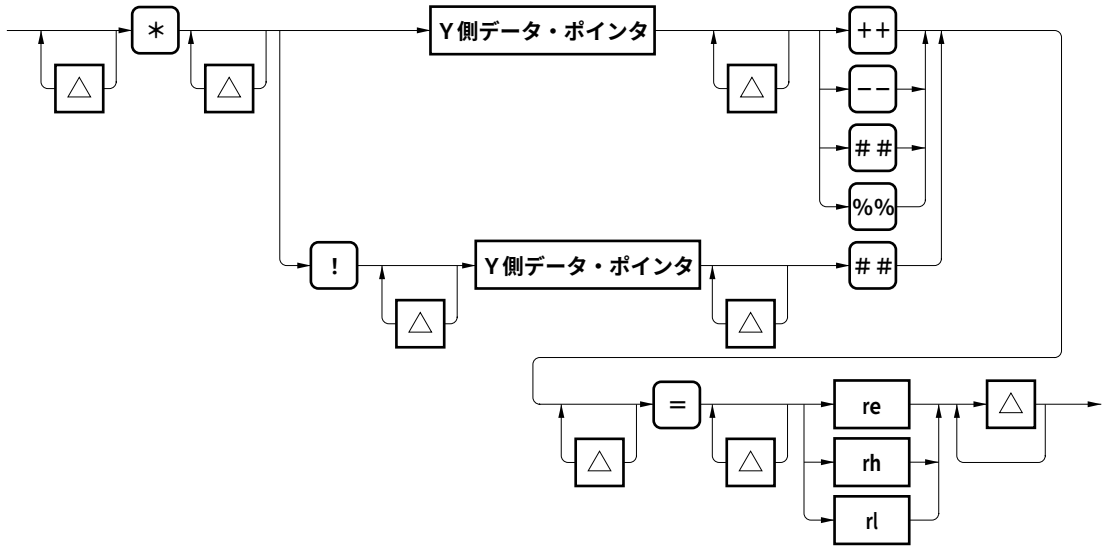


③ Y側転送（レジスタ←Yメモリ）

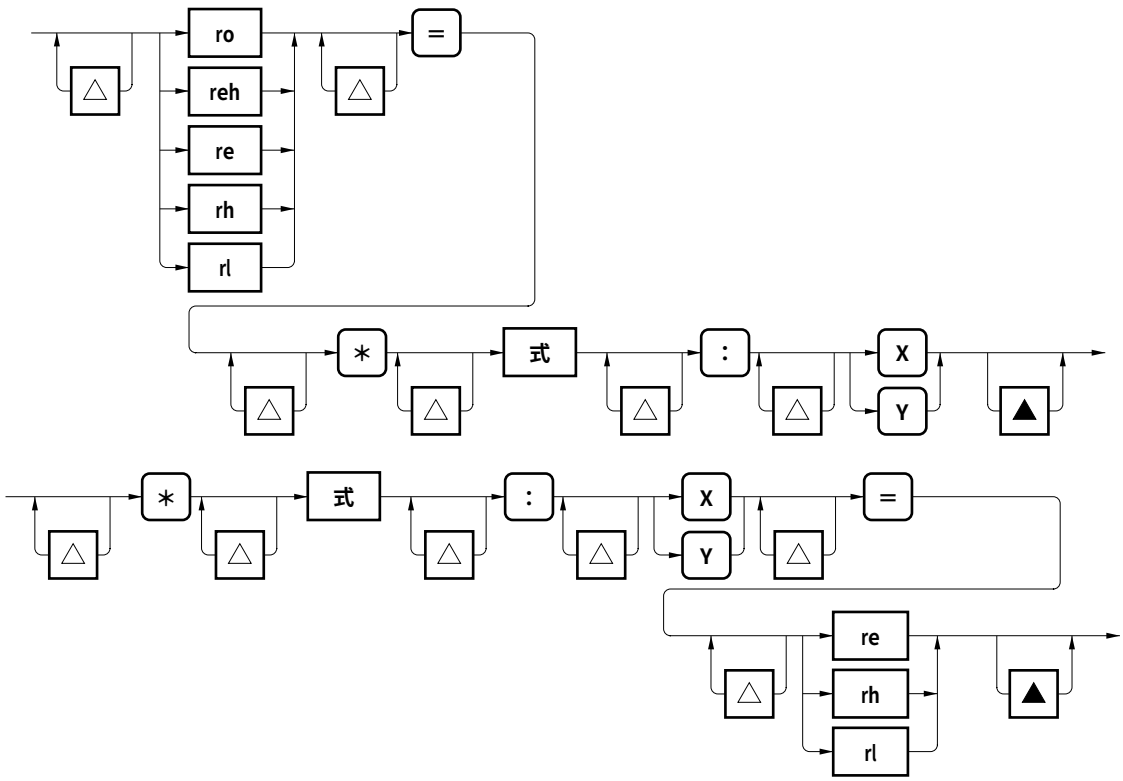


保守／廃止

④　Y側転送（Yメモリ←レジスタ）

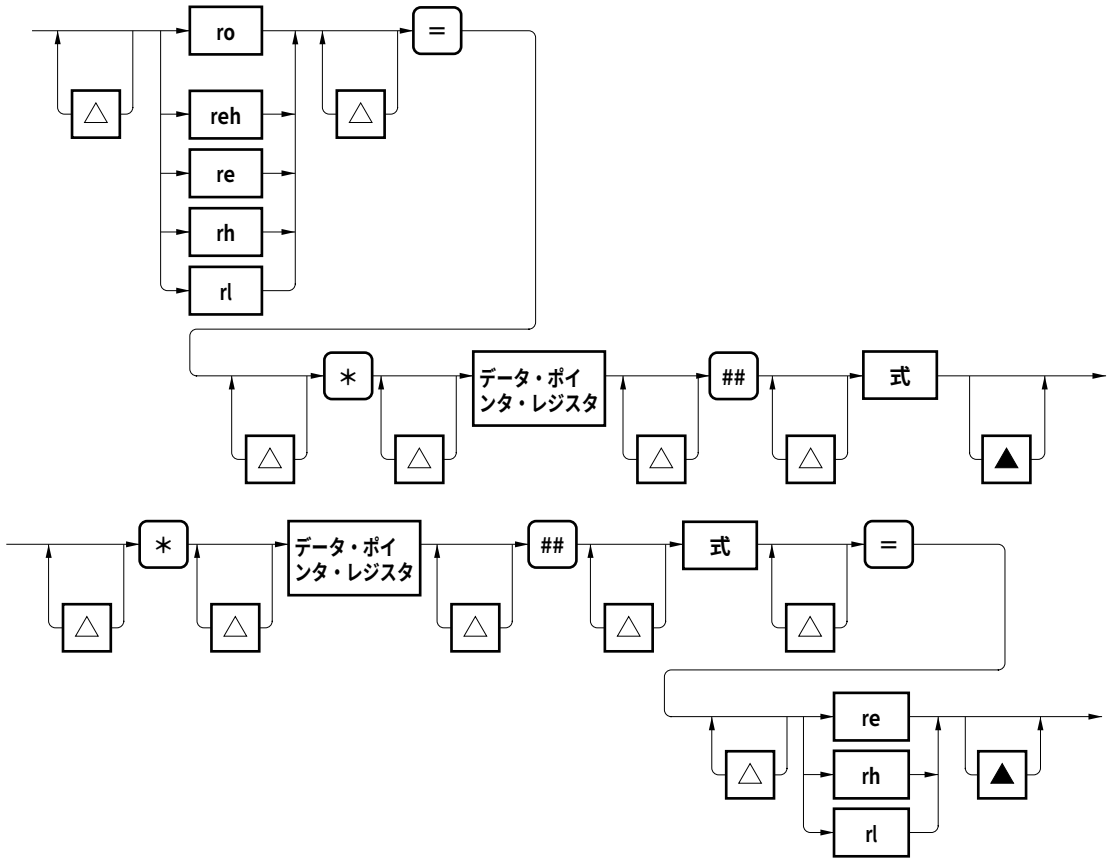


ダイレクト・アドレシング・ロード／ストア命令

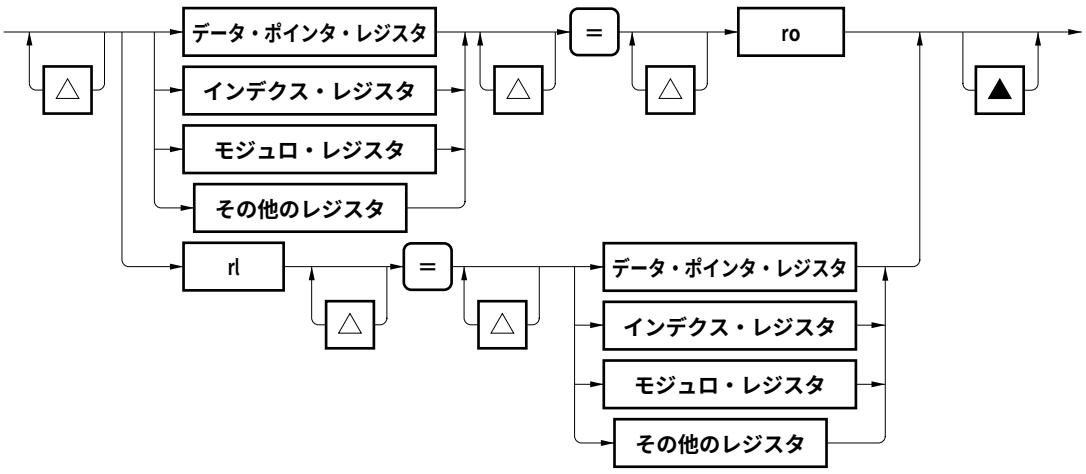


保守／廃止

即値インデクス・ロード／ストア命令

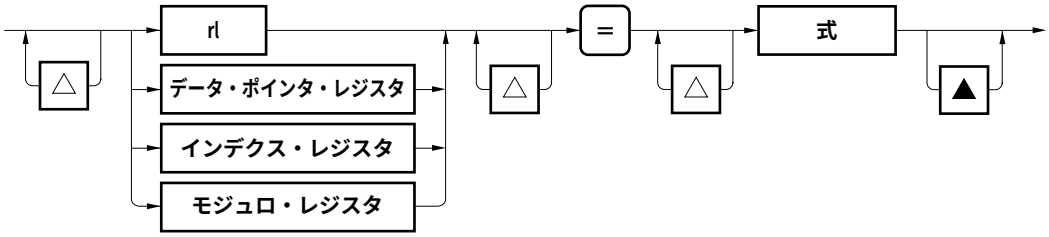


レジスタ間転送命令



保守／廃止

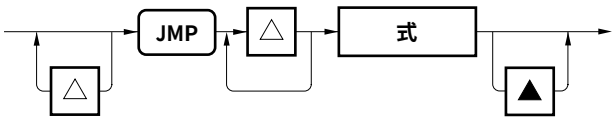
即値設定命令



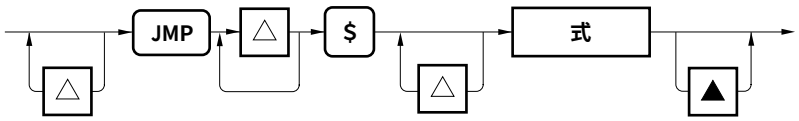
分岐命令

①　ジャンプ

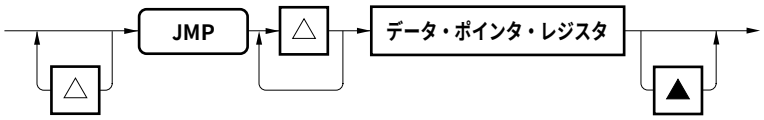
（a）絶対ジャンプ



（b）相対ジャンプ



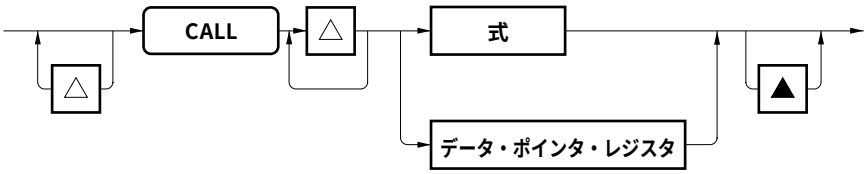
②　レジスタ間接ジャンプ



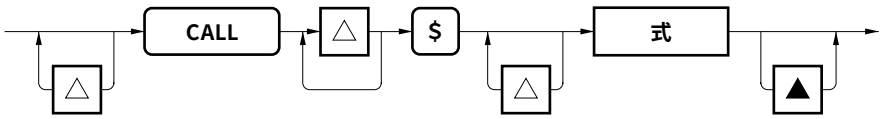
保守／廃止

③　サブルーチン・コール

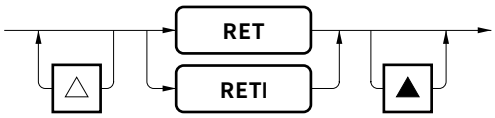
（a）絶対サブルーチン・コール，レジスタ間接サブルーチン・コール



（b）相対サブルーチン・コール



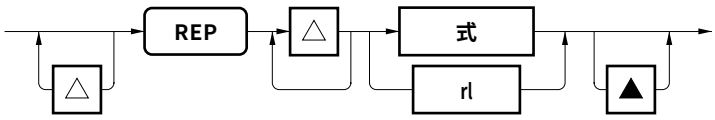
④　リターン，割り込みリターン



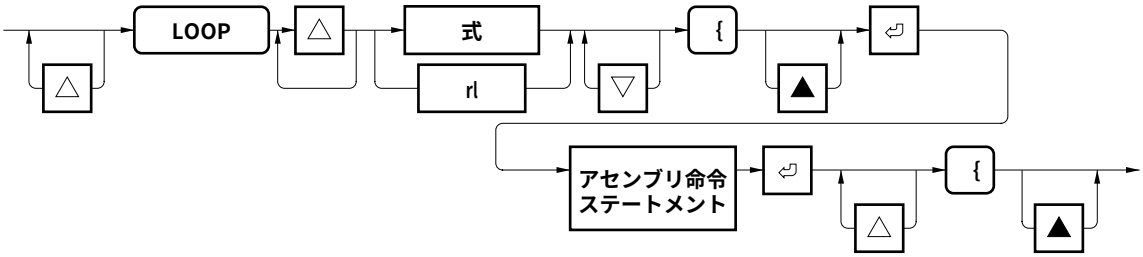
保守／廃止

ハードウエア・ループ命令

①　リピート

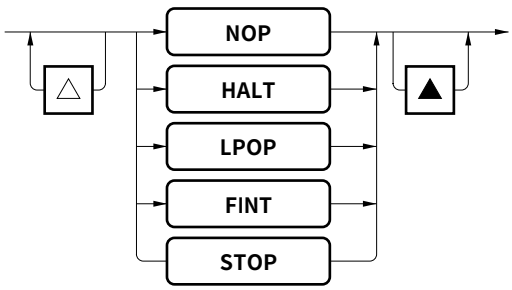


②　ループ



備考　▽は１個以上の空白，⏎を示します。

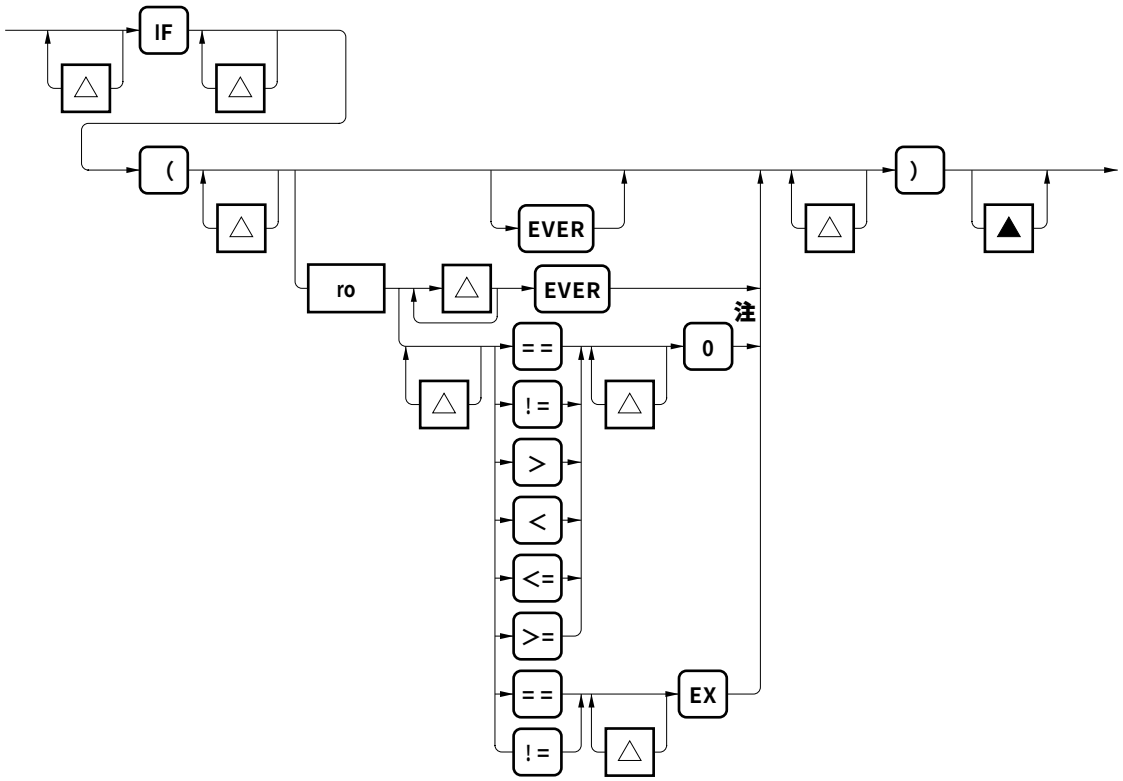
制御命令



備考　STOP命令はμPD77016にはありません。

保守／廃止

制御命令（条件）

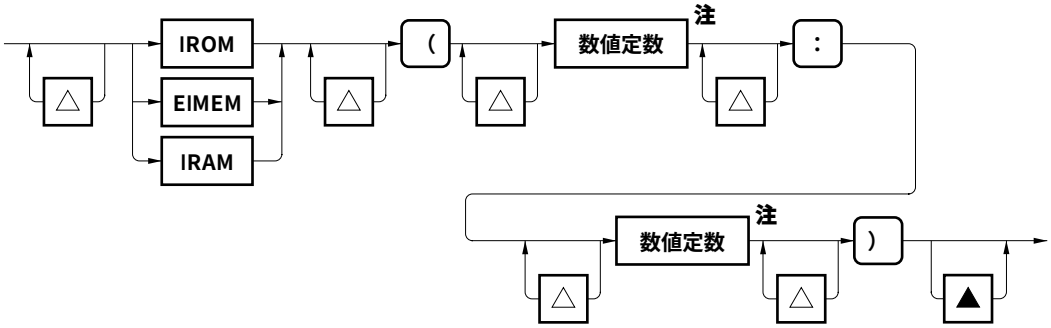


注　２進数，８進数，10進数，16進数のいずれも記述することができます。

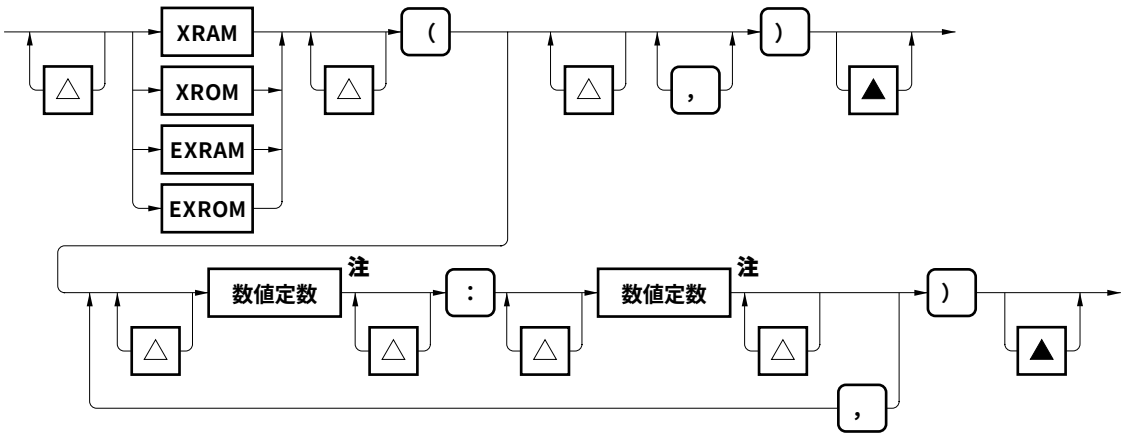
保守／廃止

メモリ領域指定ディレクティブ

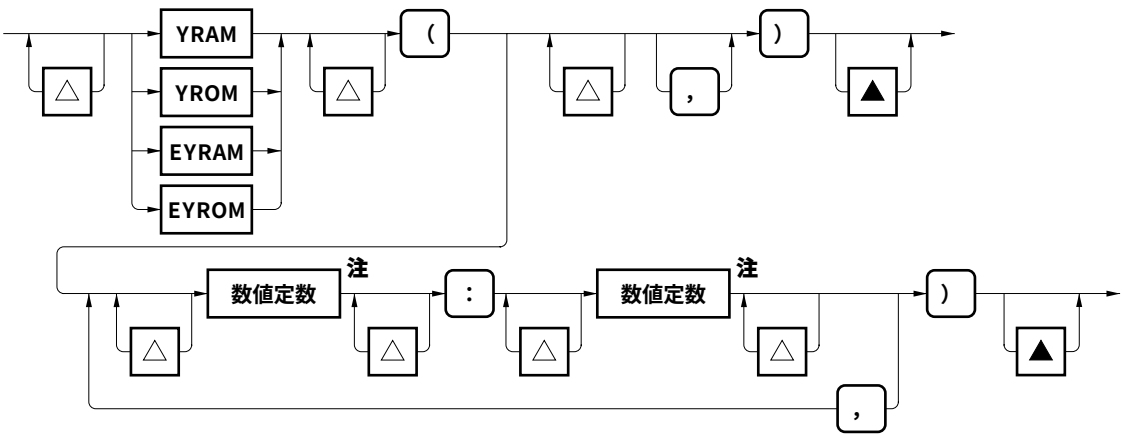
①　インストラクション・メモリ



②　Xメモリ



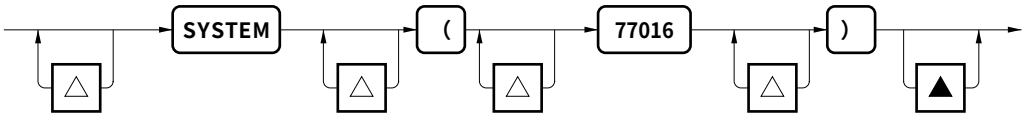
③　Yメモリ



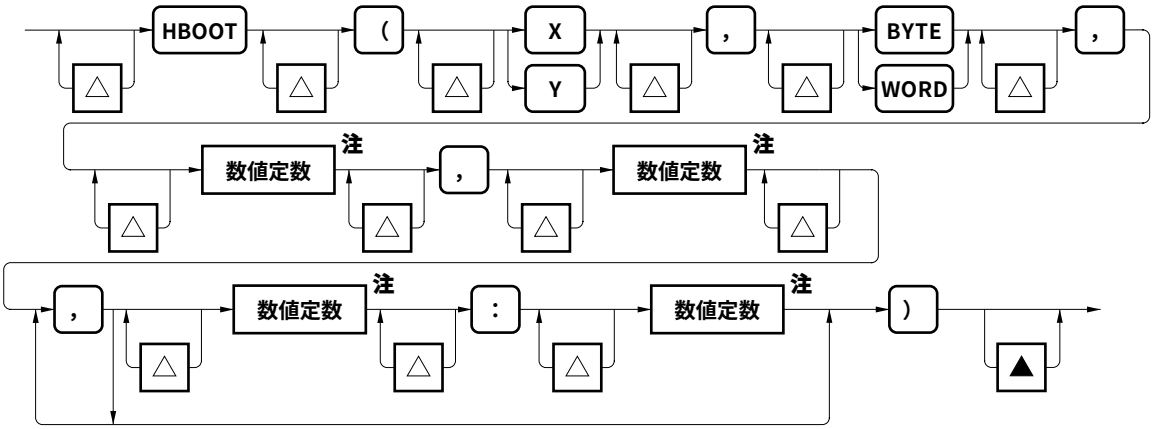
注　数値定数は固定小数点を除いたものです。

保守／廃止

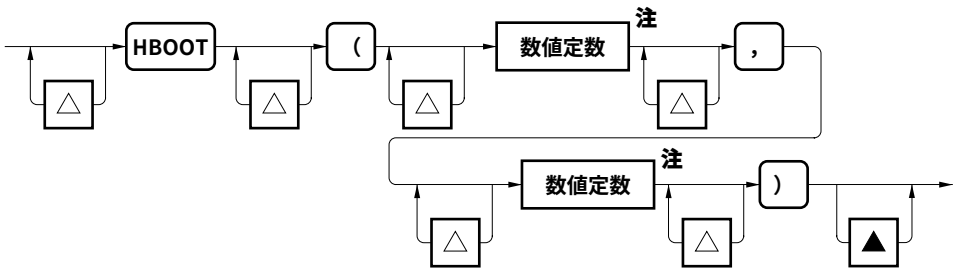
④　使用デバイス名



⑤　セルフブートアップ情報



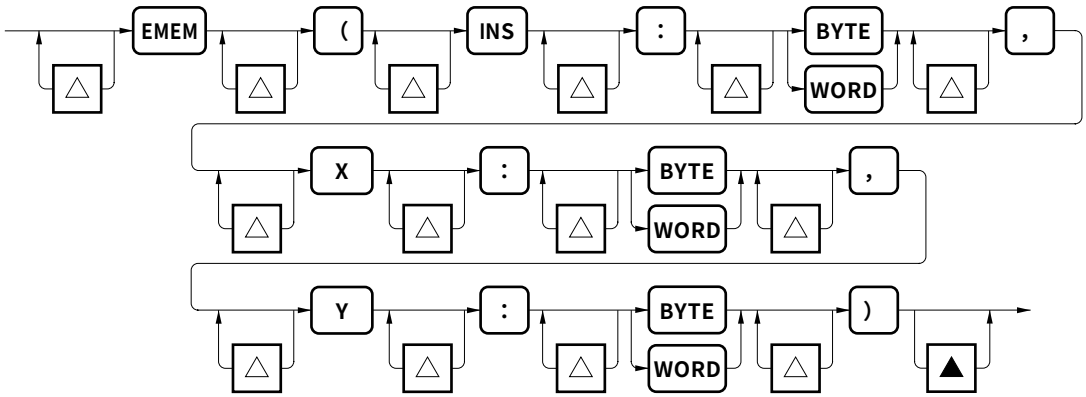
⑥　ホスト・ブートアップ情報



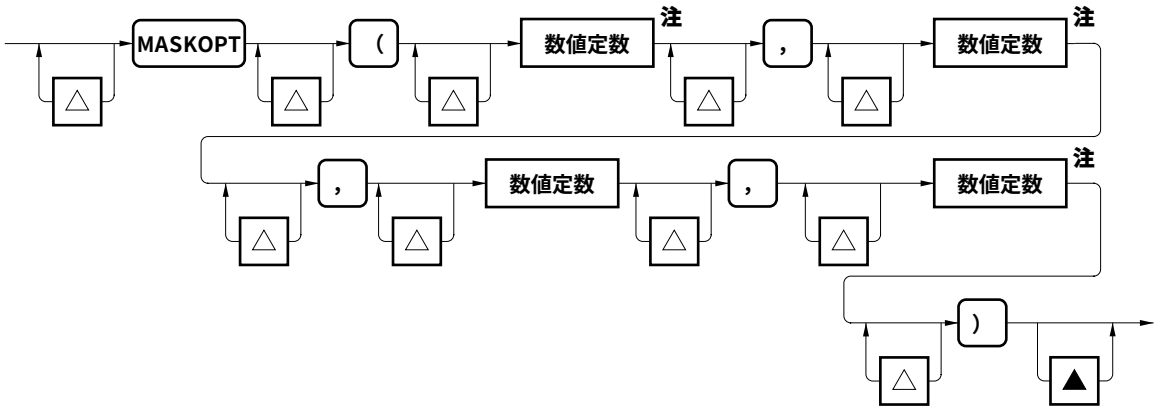
注　数値定数は固定小数点を除いたものです。

保守／廃止

⑦　外部メモリ種類別



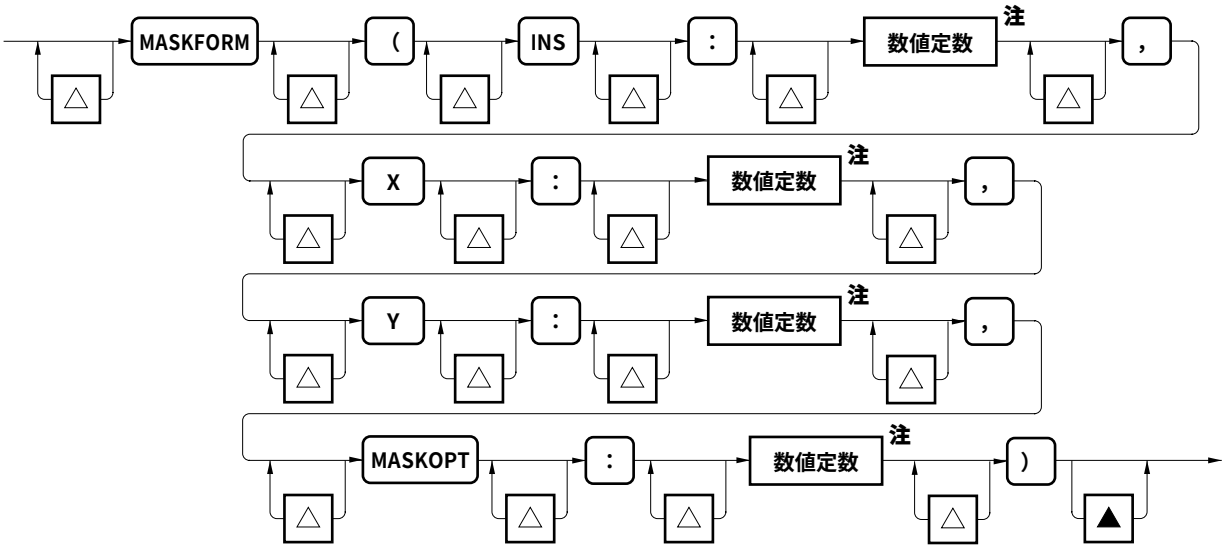
⑧　マスク・オプション情報



注　数値定数は固定小数点を除いたものです。

保守／廃止

⑨　マスクROMフォーマット情報



注　数値定数は固定小数点を除いたものです。

保守／廃止

付録B　予約語一覧

レジスタ名

R0	R1	R2	R3	R4	R5	R6	R7
R0EH	R1EH	R2EH	R3EH	R4EH	R5EH	R6EH	R7EH
R0E	R1E	R2E	R3E	R4E	R5E	R6E	R7E
R0H	R1H	R2H	R3H	R4H	R5H	R6H	R7H
R0L	R1L	R2L	R3L	R4L	R5L	R6L	R7L
DP0	DP1	DP2	DP3	DP4	DP5	DP6	DP7
DN0	DN1	DN2	DN3	DN4	DN5	DN6	DN7
DMX	DMY						
SR	EIR	STK	SP	LSP	LSR1	LSR2	LSR3
ESR	LSA	LEA	LC				

命令語

＞＞	SRA	SRL	SLL	&	｜注1	^	CLR
ABS	~注2	CLIP	ROUND	EXP			
＋＝	－＝	／＝	＋＋	－－			
JMP	CALL	RET	RETI	REP	LOOP		
NOP	HALT	LPOP	IF	FINT	THALT		
EVER	＝＝	！＝	EX	STOP注3			

注1．縦棒（0x7C）

2．ティルド（0x7E）

3．μPD77016にはありません。

疑似命令

IMSEG	XROMSEG	XRAMSEG	YROMSEG	YRAMSEG	AT	INTERNAL	EXTERNAL
EQU	SET	ORG	DS	DW	DWL		
NAME	PUBLIC	EXTRN	END				

汎用コントロール

	INCLUDE	IC	EJECT	EJ	LIST	LI	NOLIST	NOLI
	TITLE	TT	IF	ELSE	ENDIF	DEFINE		
	LOCAL	GEN	NOGEN	GENONLY	IFDEF	IFNDEF	/＊	＊/
★	PAGELENGTH		PAGEWIDTH	RADIX				

保守／廃止

演算子

＋	－	＊	／	MOD	AND	OR	XOR
NOT	EQ	＝	NE	！＝	LT	＜	LE
＜＝	GT	＞	GE	＞＝	X	Y	

メモリ名

IROM	EIMEM	IRAM	XROM	EXROM	XRAM	EXRAM	YROM
EYROM	YRAM	EYRAM	SYSTEM	SBOOT	HBOOT	EMEM	BYTE
WORD	INS	MASKOPT	MASKFORM				

備考　予約語には，大文字と小文字の区別はありません。

〔メ　モ〕

保守／廃止

お問い合わせ先

【技術的なお問い合わせ先】

ＮＥＣ半導体テクニカルホットライン
（電話：午前 9:00～12:00，午後 1:00～5:00）

電　話　：044−435−9494
FAX　：044−435−9608
E-mail　：s-info@saed.tmg.nec.co.jp

【営業関係お問い合わせ先】

第一販売事業部
東　京　(03)3798−6106, 6107, 6108
名古屋　(052)222−2375
大　阪　(06)6945−3178, 3200, 3208, 3212
仙　台　(022)267−8740
郡　山　(024)923−5591
千　葉　(043)238−8116

第二販売事業部
東　京　(03)3798−6110, 6111, 6112
立　川　(042)526−5981, 6167
松　本　(0263)35−1662
静　岡　(054)254−4794
金　沢　(076)232−7303
松　山　(089)945−4149

第三販売事業部
東　京　(03)3798−6151, 6155, 6586, 1622, 1623, 6156
水　戸　(029)226−1702
広　島　(082)242−5504
高　崎　(027)326−1303
鳥　取　(0857)27−5313
太　田　(0276)46−4014
名古屋　(052)222−2170, 2190
福　岡　(092)261−2806

【資料の請求先】

上記営業関係お問い合わせ先またはNEC特約店へお申しつけください。

【インターネット電子デバイス・ニュース】

NECエレクトロンデバイスの情報がインターネットでご覧になれます。　URL(アドレス)　http://www.ic.nec.co.jp/

保守／廃止

アンケート記入のお願い

お手数ですが，このドキュメントに対するご意見をお寄せください。今後のドキュメント作成の参考にさせていただきます。

［ドキュメント名］WB77016　ユーザーズ・マニュアル　言語編
（U10078JJ6V0UMJ1（第６版））

［お名前など］（さしつかえのない範囲で）

御社名（学校名，その他）　（　　）
ご住所　（　　）
お電話番号　（　　）
お仕事の内容　（　　）
お名前　（　　）

１．ご評価（各欄に○をご記入ください）

項　　目	大変良い	良　い	普　通	悪　い	大変悪い
全体の構成					
説明内容					
用語解説					
調べやすさ					
デザイン，字の大きさなど					
その他（　　）					
（　　）					

２．わかりやすい所（第　章，第　章，第　章，第　章，その他　　）

理由［　　］

３．わかりにくい所（第　章，第　章，第　章，第　章，その他　　）

理由［　　］

４．ご意見，ご要望

５．このドキュメントをお届けしたのは

NEC販売員，特約店販売員，その他（　　）

ご協力ありがとうございました。
下記あてにFAXで送信いただくか，最寄りの販売員にコピーをお渡しください。

日本電気（株）NECエレクトロンデバイス
半導体テクニカルホットライン
FAX：（044）435-9608

2000.6

キリトリ