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

μPD77016ファミリ

ディジタル・シグナル・プロセッサ

命令編

μPD77015
μPD77016
μPD77017
μPD77018
μPD77018A
μPD77019
μPD77110
μPD77111
μPD77112
μPD77113
μPD77114

資料番号　U13116JJ2V0UMJ1（第２版）
発行年月　October 2000 N　CP(K)

〔メ　モ〕

目次要約

CMOSデバイスの一般的注意事項

①静電気対策（MOS全般）

注意　MOSデバイス取り扱いの際は静電気防止を心がけてください。

MOSデバイスは強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には，NECが出荷梱包に使用している導電性のトレーやマガジン・ケース，または導電性の緩衝材，金属ケースなどを利用し，組み立て工程にはアースを施してください。プラスチック板上に放置したり，端子を触ったりしないでください。

また，MOSデバイスを実装したボードについても同様の扱いをしてください。

②未使用入力の処理（CMOS特有）

注意　CMOSデバイスの入力レベルは固定してください。

バイポーラやNMOSのデバイスと異なり，CMOSデバイスの入力に何も接続しない状態で動作させると，ノイズなどに起因する中間レベル入力が生じ，内部で貫通電流が流れて誤動作を引き起こす恐れがあります。プルアップかプルダウンによって入力レベルを固定してください。また，未使用端子が出力となる可能性（タイミングは規定しません）を考慮すると，個別に抵抗を介してV_{DD}またはGNDに接続することが有効です。

資料中に「未使用端子の処理」について記載のある製品については，その内容を守ってください。

③初期化以前の状態（MOS全般）

注意　電源投入時，MOSデバイスの初期状態は不定です。

分子レベルのイオン注入量等で特性が決定するため，初期状態は製造工程の管理外です。電源投入時の端子の出力状態や入出力設定，レジスタ内容などは保証しておりません。ただし，リセット動作やモード設定で定義している項目については，これらの動作ののちに保証の対象となります。

リセット機能を持つデバイスの電源投入後は，まずリセット動作を実行してください。

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

非該当品：μPD77016, 77019-013, 77110

ユーザ判定品：μPD77015, 77017, 77018, 77018A, 77019, 77111, 77112, 77113, 77114

- **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。
- 当社は品質，信頼性の向上に努めていますが，半導体製品はある確率で故障が発生します。当社半導体製品の故障により結果として，人身事故，火災事故，社会的な損害等を生じさせない冗長設計，延焼対策設計，誤動作防止設計等安全設計に十分ご注意願います。
- 当社は，当社製品の品質水準を「標準水準」，「特別水準」およびお客様に品質保証プログラムを指定して頂く「特定水準」に分類しております。また，各品質水準は以下に示す用途に製品が使われることを意図しておりますので，当社製品の品質水準をご確認の上ご使用願います。

  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

  特別水準：輸送機器（自動車，列車，船舶等），交通用信号機器，防災／防犯装置，各種安全装置，生命維持を直接の目的としない医療機器

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

当社製品のデータ・シート／データ・ブック等の資料で，特に品質水準の表示がない場合は標準水準製品であることを表します。当社製品を上記の「標準水準」の用途以外でご使用をお考えのお客様は，必ず事前に当社販売窓口までご相談頂きますようお願い致します。

M7　98.8

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. 75	**３．４　ロード／ストア命令** 並列ロード／ストア命令の注意①に説明を追加。
p. 81	**３．４　ロード／ストア命令** 部分ロード／ストア命令の注意①に説明を追加。
p. 107	**３．８　ハードウエア・ループ命令** リピート命令の注意②を変更。
p. 110	**３．８　ハードウエア・ループ命令** ループ命令に注意②を追加，注意③を変更。
p. 116	**３．９　制御命令** ストップ命令の説明を変更，注意③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。

はじめに

対象者　このマニュアルは，ディジタル・シグナル・プロセッサμPD77016ファミリの機能を理解し，これを用いたソフトウエア，ハードウエアなどのアプリケーション・システムを設計するエンジニアを対象としています。

目　的　このマニュアルは，μPD77016ファミリの持つ命令機能をユーザに理解していただき，これらのデバイスを使用するシステムのハードウエア，ソフトウエア開発の参照用資料として役立つことを目的としています。

構　成　このマニュアルは，大きく分けて以下の内容で構成します。

- ●第１章　概要
- ●第２章　命令カテゴリと命令機能
- ●第３章　機能別命令解説
- ●付録Ａ　命令語のカテゴリ分類
- ●付録Ｂ　命令セット一覧
- ●付録Ｃ　命令索引

読み方　このマニュアルを読むにあたっては，電気，論理回路，マイクロコンピュータの一般的な知識を必要とします。

μPD77016ファミリは，μPD7701×ファミリ（μPD77016, 77015, 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名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。

凡　例

データ表記の重み	：左が上位桁，右が下位桁
注	：本文中に付けた注の説明
注意	：気を付けて読んでいただきたい内容
備考	：本文の補足説明
重要事項，強調	：太字で表記
数の表記	：２進数…0b××××
	10進数…××××
	16進数…0x××××
｛　｝	：｛　｝のうちいずれか１つを選択可能。

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。

デバイスに関する資料

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	このマニュアル	U11958J	U12021J
μPD77015		U10902J				
μPD77017						
μPD77018						
μPD77018A		U11849J				
μPD77019						
μPD77019-013		U13053J				
μPD77110		U12801J	U14623J			
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	U10078J
	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
IE-77016-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。

目　　次

図の目次

表の目次

〔メ　モ〕

第１章　概　　要

μPD77016ファミリの命令語は，１語32ビットで構成されます。命令語は合理的にフィールド分割され，１語中に複数の機能命令を含むことができます。演算系の命令と転送系の命令は１インストラクション・サイクルで実行し，ループ系の命令，分岐系の命令は２または３インストラクション・サイクルで実行します。次に命令の特徴を要約します。

●１語32ビット構成
●命令機能は９グループに分類される
●１語中に複数の機能命令記述が可能
●独立の条件命令で，柔軟な条件記述が可能。

１.１　アセンブリ記述

１.１.１　命令の記述方法

μPD77016ファミリのアセンブリ言語は，ADD, MOVといった表記のニモニックを使わず，＋，－，＊，／，=などの算術記号を使います。このため，アセンブリ言語で記述されたプログラムが読みやすくなっています。

また，各行の終わりにはＣ言語と同じようにセミコロン“；”を記述して，行の終わりを明示する必要があります。“；”を記述しないかぎり，アセンブラは記述が数行にわたっていても１つの行として解釈します。

記述例を次に示します。

記述例　●R1 = R0；　　…R0レジスタの内容をR1レジスタに代入します。
●R5 = ＊DP0；　…DP0レジスタをポインタとして，Ｘメモリの内容をR5レジスタに代入します。
●R4 = R2＋R3；…R2レジスタとR3レジスタの内容を加算し，結果をR4レジスタに代入します。

1.1.2　並列記述

μPD77016ファミリの並列動作を記述するため，演算を行う命令（三項演算命令，二項演算命令，単項演算命令）と，2つのデータ・メモリ空間（Xメモリ，Yメモリ）の転送命令を1つずつ，合わせて3つの命令を並列に記述することができます。

記述例　R1 = R0　　R2 = ＊DP0　　＊DP4 = R3H；

R0レジスタの内容をR1レジスタに代入，DP0レジスタをポインタとしてXメモリの内容をR2レジスタに代入，R3Hレジスタの内容をDP4レジスタをポインタとしてYメモリに代入します。

備考　このとき，μPD77016ファミリは演算命令を実行し，次に転送命令を実行します。

1.1.3　三項演算の記述

μPD77016ファミリが内蔵している累乗算器（ハードウエアで乗算を行い，その結果をほかのレジスタに加算する）の動作を記述するため，オペランドを3つ持つ三項演算命令をサポートしています。

記述例　R0 = R0 ＋ R1L＊R2L；

R1LレジスタとR2Lレジスタの内容を乗算し，その結果をR0レジスタに加算します。

1.1.4　ポインタ操作

（2）および（3）で説明した転送命令において，それぞれの転送命令を実行したのちポインタの値を加減する命令があります。

ポインタ操作の詳細については，**μPD7701×ファミリ　ユーザーズ・マニュアル　アーキテクチャ編**または**μPD77111ファミリ　ユーザーズ・マニュアル　アーキテクチャ編**を参照してください。

記述例　R1 = ＊DP0＋＋　　R2 = ＊DP4＃＃；

DP0レジスタとDP4レジスタをポインタとして転送命令を実行したあと，DP0に1を加算し，DP4にDN4（補助レジスタ）の値を加算します。

1.1.5　条件命令の記述

レジスタの値を条件として，その条件が成立したときだけ条件命令を実行させることができます。この命令を使うと，条件実行に必要な分岐条件（命令条件）を減らすことができます。

条件命令の詳細については，**3.9　制御命令**の条件命令（COND）を参照してください。

記述例　if（R0＜0）　R1L＋＝R2L；

R0レジスタの値が負のとき，R1レジスタにR2レジスタの内容を加算します。

1.1.6　ハードウエア・ループ命令の記述

μPD77016ファミリには，1-255行の命令を1-32767回まで繰り返し実行するハードウエア・ループ命令があります。この命令を記述するため，リピート命令とループ命令があります。

●１行の命令を複数回実行させるとき

```
REPEAT   32
         R0 = R0 ＋ R1L＊R2L     R1 = ＊DP0＋＋     R2 = ＊DP4＋＋；
```

●2-255行の命令を複数回実行させるとき

```
LOOP     R1L｛
         R0 = R0 ＋ R1L＊R2L；
              ⋮
         R4 = R4－１；
｝；
```

●ループ命令を入れ子（ネスティング）で記述するとき

```
LOOP     64｛
         R0 = R0 ＋ R1L＊R2L；
                       ⋮
         LOOP          R1L｛
                       R3 = R0＋R4；
                       ⋮
         ｝；
         R4 = R4－１；
         R5 = R5＋１；
         NOP；
｝；
```

1.2 凡　例

1.2.1 表現形式と選択できるレジスタの対応

次の節ではレジスタを一般的に表現します。レジスタの一般表現された表記と対応する具体的レジスタを表１－１に示します。

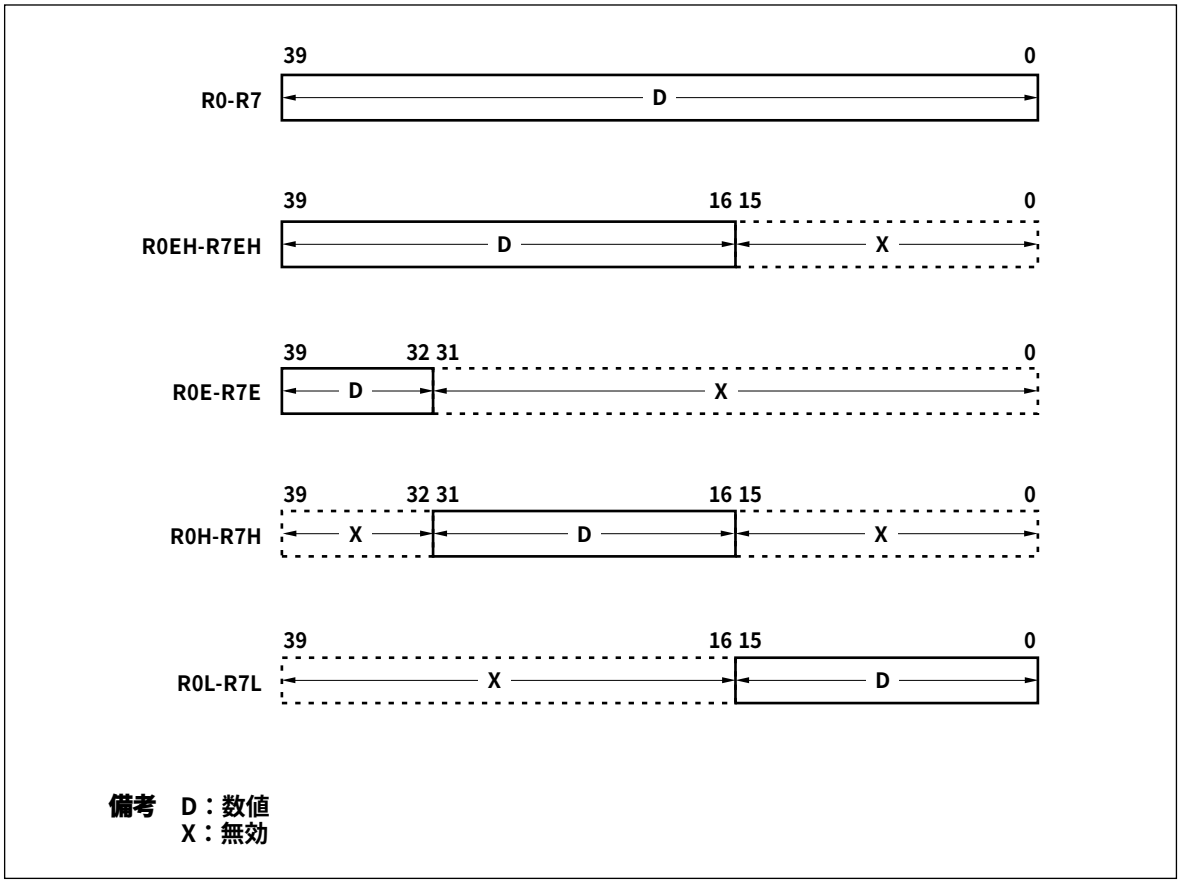
表１－１　表現形式と対応レジスタ

表現形式	対 応 レ ジ ス タ
ro, ro', ro"	R0-R7
rl, rl'	R0L-R7L
rh, rh'	R0H-R7H
re	R0E-R7E
reh	R0EH-R7EH
dp	DP0-DP7
dn	DN0-DN7
dm	DMX, DMY
dpx	DP0-DP3
dpy	DP4-DP7
dpx_mod	DPn, DPn＋＋, DPn－－, DPn＃＃, DPn%%，！DPn＃＃（n = 0-3）
dpy_mod	DPn, DPn＋＋, DPn－－, DPn＃＃, DPn%%，！DPn＃＃（n = 4-7）
dp_imm	DPn＃＃imm（n = 0-7）
＊××××	×××をアドレスとするメモリの内容 例　DP0レジスタの内容が1000のとき，＊DP0はメモリの1000番地の内容を表します。

1.2.2　汎用レジスタの分割フォーマット

汎用レジスタは，演算や転送にあたってその部分のみを対象にすることがあります。図１－１に汎用レジスタの分割フォーマットを示します。

図１－１　汎用レジスタの分割フォーマット



備考　D：数値
X：無効

1.2.3　データ・ポインタのモディファイ

データ・ポインタ（DPn）を使用した間接アドレシングでは，多くの場合メモリ・アクセス後DPnをモディファイします。表１－２にモディファイ表記とオペレーションを示します。

表１－２　データ・ポインタのモディファイ

記 述 例	オペレーション
DPn	何もしません（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） $DPn = ((DP_L + DNn) \bmod (DMX + 1)) + DP_H$
	（n = 4-7） $DPn = ((DP_L + DNn) \bmod (DMY + 1)) + DP_H$
！DPn＃＃	DPnをビット・リバース後メモリ・アクセスする。 メモリ・アクセス後　DPn←DPn＋DNn
DPn ＃＃imm	DPn←DPn＋imm

1.3 記 述 例

命令の略号

MADD

命令の種類

MADD

三項演算

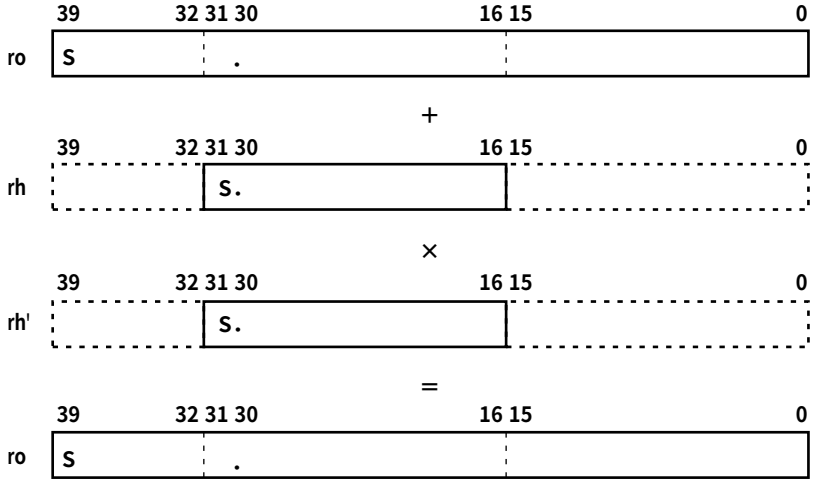
ro = ro ＋ rh＊rh'

命令名称	**マルチプライ・アド**
ニモニック	**ro = ro ＋ rh＊rh'**

[記 述 例]　R1 = R1＋R0H＊R4H

[説　明]　：命令のオペレーションの詳細を解説します。

２つの16ビット・データを乗算して得た積と，40ビット・データを加算する命令です。



[実行サイクル]　１：実行サイクル数を表します。

[同時記述できる命令]：○のついている命令がこの命令と同時記述できます。

三項演算	二項演算	単項演算	並列ロード／ストア	レジスタ間転送	分　岐	条　件
×	×	×	○	×	×	×

[注　意]　：特に注意するべきことについて説明します。

〔メ　モ〕

第２章　命令カテゴリと命令機能

命令は１語32ビットで構成されますが，デバイス内に分布する多数の資源ブロックを平行して管理するために，多様なフォーマットでフィールド分割されています。また，それらフォーマットは，いくつかのカテゴリに分類整理され，簡潔な命令機能グループとして合理的な統一が図られています。

２.１　命令語の分類

この章では，命令を分類整理して説明しますが，これには２つの視点があります。

２.１.１　命令語のフォーマットに基づく分類（カテゴリ分類）

命令語をフィールド分割するフォーマットの相違で分類するもので，カテゴリ分類と呼びます。したがって，分類された各カテゴリは相互に排他的で，あるカテゴリのフォーマットで記述された命令は，同時に別のフォーマットを採ることはできません。

この分類にしたがえば，１語の命令中にどのような機能が平行して動作するのかをただちに見ることができます。したがって，それら要素的なフィールドを合成して命令語のビット・パターンを作成したり，逆に命令語のビット・パターンから機能を分析するといったことも可能です。

この分類は，アセンブラなどの言語処理系に密接に関係した視点を与えるものですが，アプリケーション・プログラミングでは，通常直接にこれらフォーマットを意識することはありません。

この章では，カテゴリ分類について詳細に説明することはしません。詳しくは，**付録Ａ　命令語のカテゴリ分類**をご覧ください。

２.１.２　命令の機能に注目した分類（機能分類）

命令が実現するグローバルな機能に焦点を当てて分類するもので，アプリケーション記述ではこの分類にしたがって命令を整理，認識すると便利です。このマニュアルでは，次の９つの機能グループに，命令を分類整理しています。

- ●三項演算命令
- ●二項演算命令
- ●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
- ●分岐命令
- ●ハードウエア・ループ命令
- ●制御命令

備考　これらのグループは，１語の命令中で相互に排他的というものではなく，１語中に同時記述可能なものもあります。詳しくは，**第３章　機能別命令解説**をご覧ください。

第３章　機能別命令解説

ここではμPD77016ファミリの命令を機能別に分類して，個々の命令について詳しく説明します。
μPD77016ファミリの命令は次の９種類があります。

３.１　三項演算命令

累乗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・ファイルのうちの３レジスタを任意に指定できます。

三項演算命令には，次の命令があります。（　）内は命令の略号です。

マルチプライ・アド（MADD）
マルチプライ・サブ（MSUB）
サイン・アンサイン・マルチプライ・アド（SUMA）
アンサイン・アンサイン・マルチプライ・アド（UUMA）
１ビット・シフト・マルチプライ・アド（MAS1）
16ビット・シフト・マルチプライ・アド（MAS16）

MADD　　MADD

三項演算

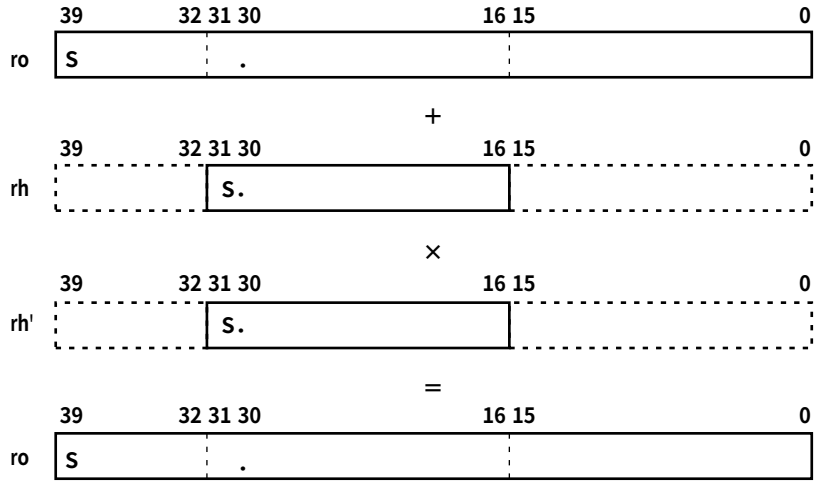
ro = ro ＋ rh＊rh'

命令名称	マルチプライ・アド
ニモニック	ro = ro ＋ rh＊rh'

［記 述 例］　R1 = R1＋R0H＊R4H

［説　　明］　２つの16ビット・データを乗算して得た積と，40ビット・データを加算する命令です。

roで指定した汎用レジスタのビット39-０の値に，rhとrh'で指定した汎用レジスタのビット31-16の値どうしの積を加算します。結果をroで指定した汎用レジスタのビット39-０に格納します。rh, rh'で指定した16ビットの値およびroで指定した40ビットの値は，２の補数表現のデータ・フォーマットです。



たとえば，rhおよびrh'で指定した汎用レジスタの値の，ビット31とビット30の間に小数点があるとすると，roで指定した汎用レジスタの値と加算する積は，ビット31とビット30の間に小数点があり，ビット39-32に符号拡張，ビット０を０にした値です。

MADD　　MADD

[実行サイクル]　１

[同時記述できる命令]

三項演算	二項演算	単項演算	並列ロード／ストア	レジスタ間転送	分　岐	条　件
×	×	×	○	×	×	×

[注　　意]　　加算の結果オーバフローが発生した場合，エラー・ステータス・レジスタのオーバフロー・フラグは“１”にセットされます。

MSUB　　MSUB

三項演算

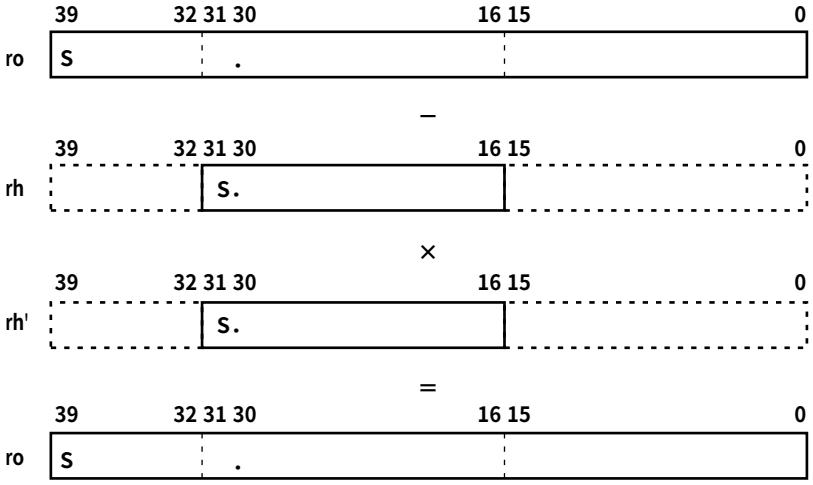
ro = ro － rh＊rh'

命令名称	マルチプライ・サブ
ニモニック	ro = ro － rh＊rh'

[記 述 例]　R1 = R1－R0H＊R4H

[説　　明]　２つの16ビット・データを乗算して得た積を，40ビット・データから減算する命令です。

roで指定した汎用レジスタのビット39-０の値から，rhとrh'で指定した汎用レジスタのビット31-16の値どうしの積を減算します。結果をroで指定した汎用レジスタのビット39-０に格納します。rh, rh'で指定した16ビットの値およびroで指定した40ビットの値は，２の補数表現のデータ・フォーマットです。



たとえば，rh, rh'およびroで指定した汎用レジスタの値の，ビット31とビット30の間に小数点があるとすると，roで指定した汎用レジスタの値から減算する積は，ビット31とビット30の間に小数点があり，ビット39-32に符号拡張，ビット０を０にした値です。

MSUB　　MSUB

[実行サイクル]　１

[同時記述できる命令]

三項演算	二項演算	単項演算	並列ロード／ストア	レジスタ間転送	分　岐	条　件
×	×	×	○	×	×	×

[注　　意]　減算の結果オーバフローが発生した場合，エラー・ステータス・レジスタのオーバフロー・フラグは“１”にセットされます。

SUMA　　SUMA

三項演算

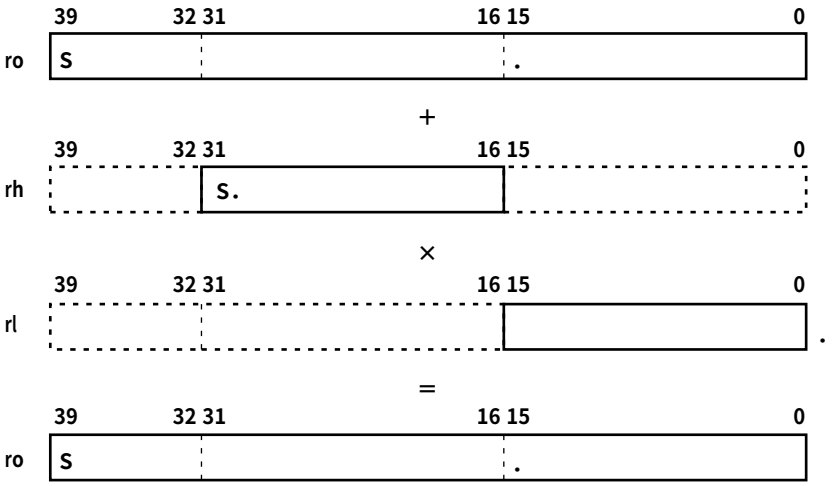
ro = ro ＋ rh＊rl

命令名称	サイン・アンサイン・マルチプライ・アド
ニモニック	ro = ro ＋ rh＊rl

［記 述 例］　R1 = R1＋R0H＊R4L

［説　　明］　２つの16ビット・データを乗算して得た積と，40ビット・データを加算する命令です。

roで指定した汎用レジスタのビット39-０の値に，rhで指定した汎用レジスタのビット31-16の値と，rlで指定した汎用レジスタのビット15-０の値とを乗算した値を加算します。結果をroで指定した汎用レジスタのビット39-０に格納します。rhで指定した16ビットの値およびroで指定した40ビットの値は２の補数表現，rlで指定した16ビットの値は正の整数値のデータ・フォーマットです。



たとえば，rhで指定した汎用レジスタの値の，ビット31とビット30の間に小数点があり，rlで指定した汎用レジスタの値のビット０の下位に小数点があるとすると，roで指定した汎用レジスタの値と加算する積は，ビット16とビット15の間に小数点があり，ビット39-32に符号拡張，ビット０を０にした値です。

SUMA　　SUMA

[実行サイクル]　１

[同時記述できる命令]

三項演算	二項演算	単項演算	並列ロード／ストア	レジスタ間転送	分　岐	条　件
×	×	×	○	×	×	×

[注　　意]　　加算の結果オーバフローが発生した場合，エラー・ステータス・レジスタのオーバフロー・フラグは“１”にセットされます。

UUMA　　UUMA

三項演算

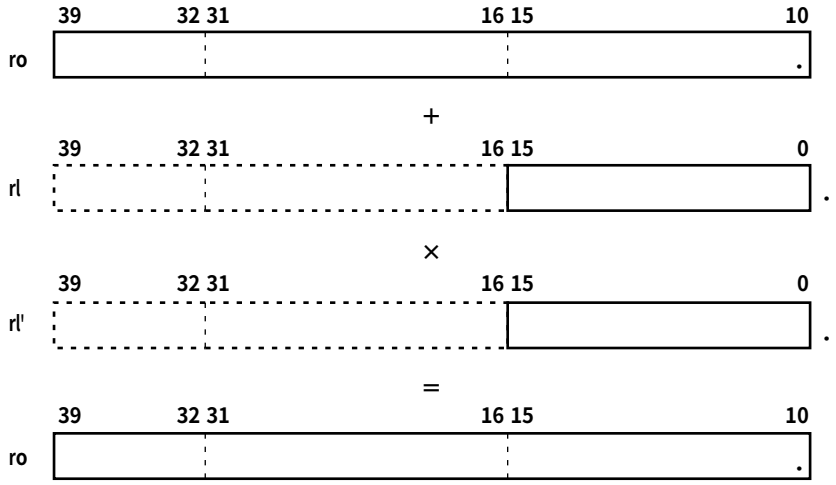
ro = ro ＋ rl＊rl'

命令名称	アンサイン・アンサイン・マルチプライ・アド
ニモニック	ro = ro ＋ rl＊rl'

［記 述 例］　R1 = R1＋R0L＊R4L

［説　　明］　２つの16ビット・データを乗算して得た積と，40ビット・データを加算する命令です。

roで指定した汎用レジスタのビット39-０の値に，rlとrl'で指定した汎用レジスタのビット15-０の値どうしを乗算した値を加算します。結果をroで指定した汎用レジスタに格納します。rlおよびrl'で指定した16ビットの値は正の整数値，roで指定した40ビットの値は２の補数表現のデータ・フォーマットです。



たとえば，rlおよびrl'で指定した汎用レジスタの値のビット０の下位に小数点があるとすると，roで指定した汎用レジスタの値と加算する積は，ビット１とビット０の間に小数点があり，ビット39-32とビット０を０にした値です。

UUMA　　UUMA

[実行サイクル]　１

[同時記述できる命令]

三項演算	二項演算	単項演算	並列ロード／ストア	レジスタ間転送	分　岐	条　件
×	×	×	○	×	×	×

[注　　意]　① オーバフローの判断は，40ビットの２の補数表現のデータ・フォーマットとして行います。

② 加算の結果オーバフローが発生した場合，エラー・ステータス・レジスタのオーバフロー・フラグは“１”にセットされます。

MAS1　　MAS1

三項演算

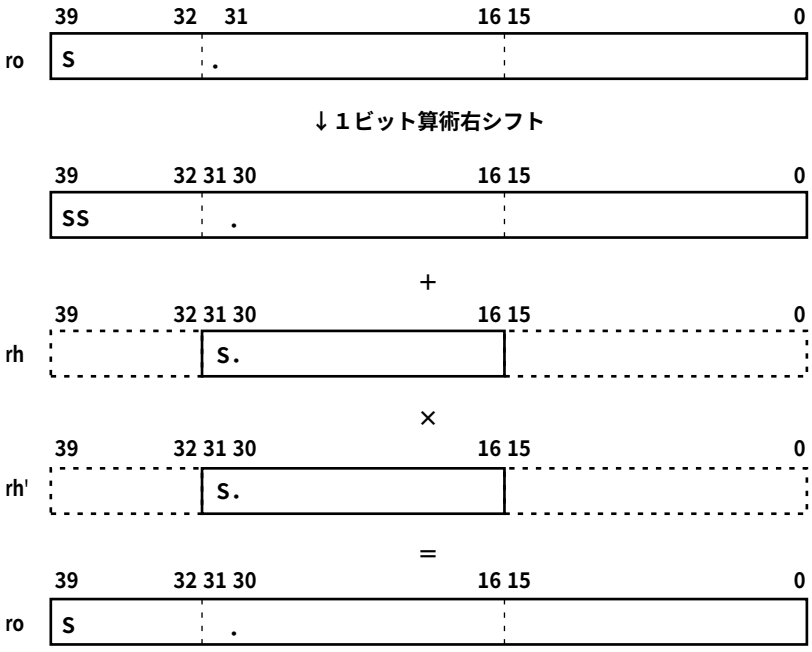
ro＝（ro＞＞1）＋rh＊rh'

命令名称	１ビット・シフト・マルチプライ・アド
ニモニック	ro＝（ro＞＞1）＋ rh＊rh'

[記 述 例]　R1=（R1＞＞１）＋R0H＊R4H

[説　　明]　２つの16ビット・データを乗算して得た積と，40ビット・データを加算する命令です。

roで指定した汎用レジスタのビット39-０の値を１ビット算術右シフトした値に，rhとrh' で指定した汎用レジスタのビット31-16の値どうしの積を加算します。結果をroで指定した汎用レジスタのビット39-０に格納します。rh, rh'で指定した16ビットの値およびroで指定した40ビットの値は，２の補数表現のデータ・フォーマットです。



たとえば，rhおよびrh'で指定した汎用レジスタの値の，ビット31とビット30の間に小数点があるとすると，roで指定した汎用レジスタの値と加算する積は，ビット31とビット30の間に小数点があり，ビット39-32に符号拡張，ビット０を０にした値です。

備考　FFTを高速に実行するために有効な機能です。

MAS1　　MAS1

[実行サイクル]　1

[同時記述できる命令]

三項演算	二項演算	単項演算	並列ロード／ストア	レジスタ間転送	分　岐	条　件
×	×	×	○	×	×	×

[注　　意]　算術右シフトでアンダフローが発生しても，値は補正されません。
たとえば0xFF' FFFF' FFFFを算術右シフトしても，値は変化しません（０になりません）。

MAS16　　MAS16

三項演算

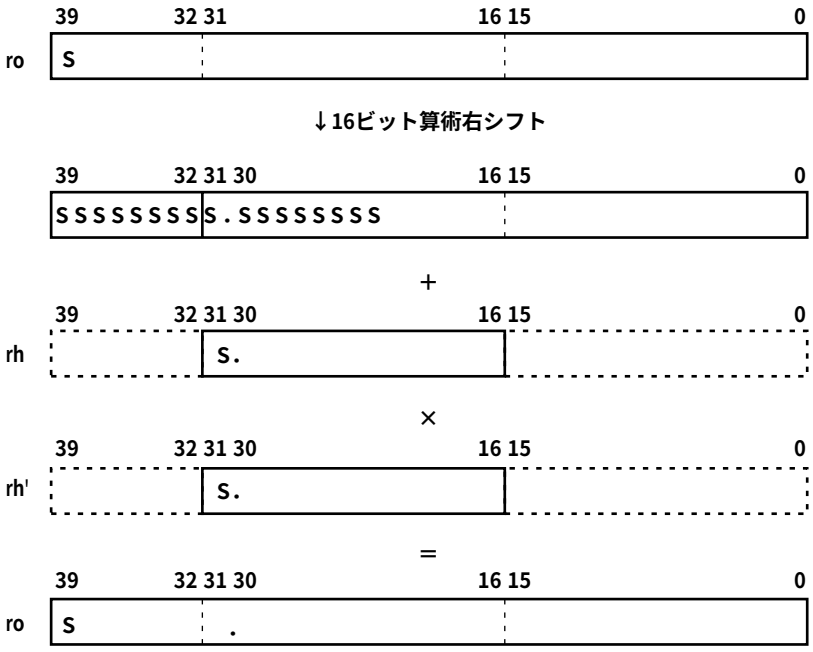
ro =（ro＞＞16）＋rh＊rh'

命令名称	**16ビット・シフト・マルチプライ・アド**
ニモニック	**ro=（ro＞＞16）＋rh＊rh'**

[記 述 例]　R1 =（R1＞＞16）＋R0H＊R4H

[説　　明]　２つの16ビット・データを乗算して得た積と，40ビット・データを加算する命令です。

roで指定した汎用レジスタのビット39-０の値を16ビット算術右シフトした値に，rhとrh'で指定した汎用レジスタのビット31-16の値どうしの積を加算します。結果をroで指定した汎用レジスタのビット39-０に格納します。rh, rh'で指定した16ビットの値およびroで指定した40ビットの値は，２の補数表現のデータ・フォーマットです。



たとえば，rhおよびrh'で指定した汎用レジスタの値の，ビット31とビット30の間に小数点があるとすると，roで指定した汎用レジスタの値と加算する積は，ビット31とビット30の間に小数点があり，ビット39-32に符号拡張，ビット０を０にした値です。

備考　倍精度での乗算を高速に行うための機能です。

MAS16　　MAS16

[実行サイクル]　１

[同時記述できる命令]

三項演算	二項演算	単項演算	並列ロード／ストア	レジスタ間転送	分　岐	条　件
×	×	×	○	×	×	×

[注　　意]　算術右シフトの結果アンダフローが発生しても，値は補正されません。
たとえば算術右シフトの過程で0xFF' FFFF' FFFFを算術右シフトしても，値は変化しません（０にはなりません）。

３.２　二項演算命令

累乗算器，ALUまたはバレル・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・ファイルのうちから２つのレジスタを任意に指定できます。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があります。出力先は汎用レジスタ・ファイルのうちから１つのレジスタを，任意に指定できます。

二項演算命令には，次の命令があります。

マルチプライ（MPY）
アド（ADD）
イミーディエト・アド（IADD）
サブ（SUB）
イミーディエト・サブ（ISUB）
算術右シフト（SRA）
イミーディエト算術右シフト（ISRA）
論理右シフト（SRL）
イミーディエト論理右シフト（ISRL）
論理左シフト（SLL）
イミーディエト論理左シフト（ISLL）
アンド（AND）
イミーディエト・アンド（IAND）
オア（OR）
イミーディエト・オア（IOR）
イクスクルーシブ・オア（XOR）
イミーディエト・イクスクルーシブ・オア（IXOR）
レスザン（LT）

MPY　　MPY

二項演算

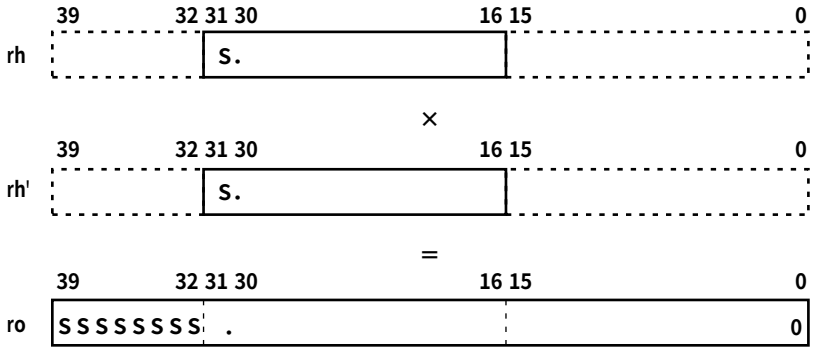
ro= rh＊rh'

命令名称	**マルチプライ**
ニモニック	**ro = rh＊rh'**

[記 述 例]　　R1 = R0H＊R4H

[説　　明]　　２つの16ビット・データを乗算する命令です。

rhとrh'で指定した，汎用レジスタのビット31-16の値どうしを乗算します。乗算結果をroで指定した汎用レジスタのビット39-０に格納します。rh, rh'で指定した16ビットの値およびroで指定した40ビットの値は，２つの補数表現のデータ・フォーマットです。



たとえば，rhとrh'で指定した汎用レジスタの値の，ビット31とビット30の間に小数点があるとすると，roで指定した汎用レジスタに格納する値は，ビット31とビット30の間に小数点があり，ビット39-32に符号拡張，ビット０を０にした値です。

[実行サイクル]　１

[同時記述できる命令]

三項演算	二項演算	単項演算	並列ロード／ストア	レジスタ間転送	分　岐	条　件
×	×	×	○	×	×	×

[注　　意]　　次の場合，演算結果のビット31は，符号ではありません。

0x8000×0x8000 = 0x00' 8000' 0000

ADD　　ADD

二項演算

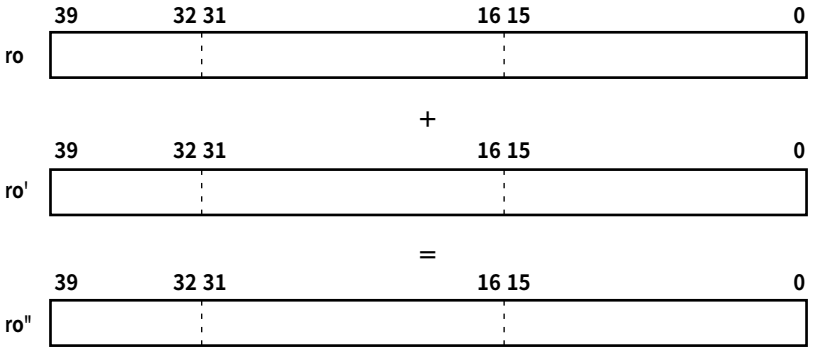
ro'' = ro＋ro'

命令名称	アド
ニモニック	ro'' = ro＋ro'

[記 述 例]　R1 = R0＋R4

[説　　明]　２つの40ビット・データを加算する命令です。

roとro'で指定した汎用レジスタのビット39-０の値どうしを加算します。結果をro''で指定した汎用レジスタのビット39-０に格納します。ro, ro'およびro''で指定した40ビットの値は，２の補数表現のデータ・フォーマットです。



[実行サイクル]　１

[同時記述できる命令]

三項演算	二項演算	単項演算	並列ロード／ストア	レジスタ間転送	分　岐	条　件
×	×	×	○	×	×	×

[注　　意]　加算の結果オーバフローが発生した場合，エラー・ステータス・レジスタのオーバフロー・フラグは“１”にセットされますが値は補正されません。

IADD　　IADD

二項演算

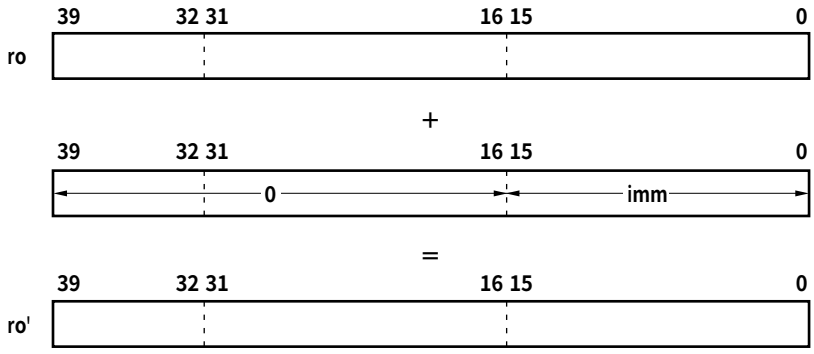
ro' = ro＋imm

命令名称	**イミーディエト・アド**
ニモニック	**ro' = ro＋imm（imm = 0-0xFFFF（0-65535））**

[記 述 例]　R1 = R0＋0x10

[説　　明]　２つの40ビット・データを加算する命令です。

roで指定した汎用レジスタのビット39-０の値と，ビット15-0の値をimmで指定しビット39-16に０拡張した40ビットのイミーディエト値とを加算します。結果をro'で指定した汎用レジスタのビット39-０に格納します。roおよびro'で指定した40ビットの値は，２の補数表現のデータ・フォーマットです。



[実行サイクル]　１

[同時記述できる命令]

三項演算	二項演算	単項演算	並列ロード／ストア	レジスタ間転送	分　岐	条　件
×	×	×	×	×	×	×

[注　　意]　加算の結果，オーバフローが発生した場合，エラー・ステータス・レジスタのオーバフロー・フラグは“１”にセットされます。

SUB　　SUB

二項演算

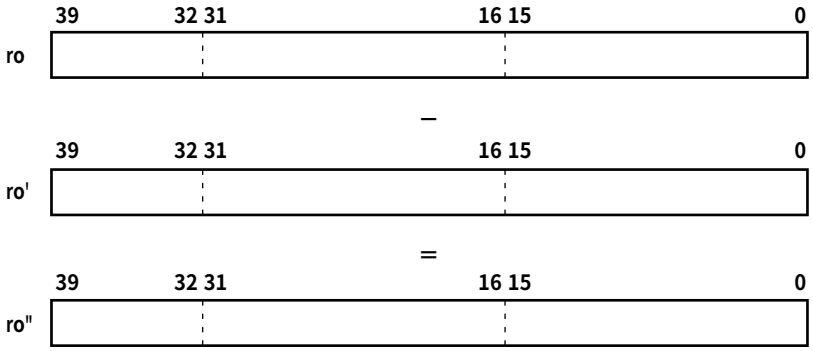
ro'' = ro－ro'

命令名称	**サブ**
ニモニック	**ro'' = ro－ro'**

[記 述 例]　　R1 = R0－R4

[説　　明]　　40ビット・データから40ビット・データを減算する命令です。

roで指定した汎用レジスタのビット39-０の値から，ro'で指定した汎用レジスタのビット39-0の値を減算します。結果をro''で指定した汎用レジスタのビット39-０に格納します。ro, ro'およびro''で指定した40ビットの値は，２の補数表現のデータ・フォーマットです。



[実行サイクル]　１

[同時記述できる命令]

三項演算	二項演算	単項演算	並列ロード／ストア	レジスタ間転送	分　岐	条　件
×	×	×	○	×	×	×

[注　　意]　　減算の結果オーバフローが発生した場合はエラー・ステータス・レジスタのオーバフロー・フラグは“１”にセットされます。

ISUB　　ISUB

二項演算

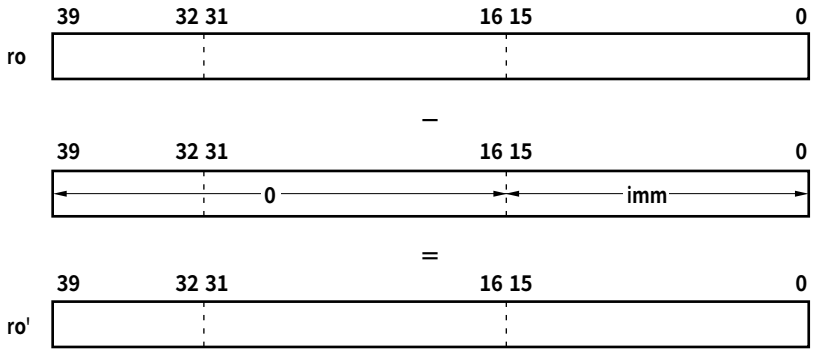
ro' = ro－imm

命令名称	**イミーディエト・サブ**
ニモニック	**ro' = ro－imm（imm = 0-0xFFFF（0-65535））**

[記 述 例]　R1 = R0－0x10

[説　　明]　40ビット・データから40ビット・データを減算する命令です。

roで指定した汎用レジスタのビット39-０の値から，ビット15-0の値を命令で設定し，ビット39-16に０拡張した40ビットのイミーディエト値を減算します。結果をro'で指定した汎用レジスタのビット39-０に格納します。roおよびro'で指定した40ビットの値は，２の補数表現のデータ・フォーマットです。



[実行サイクル]　１

[同時記述できる命令]

三項演算	二項演算	単項演算	並列ロード／ストア	レジスタ間転送	分　岐	条　件
×	×	×	×	×	×	×

[注　　意]　減算の結果オーバフローが発生した場合，エラー・ステータス・レジスタのオーバフロー・フラグは“１”にセットされます。

SRA　　SRA

二項演算

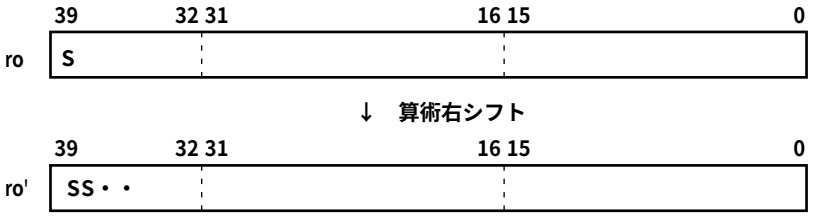
ro' = ro SRA rl

命令名称	**算術右シフト**
ニモニック	**ro' = ro SRA rl**

[記 述 例]　　R1 = R0　SRA　R4L

[説　　明]　　40ビット・データを算術右シフトする命令です。

roで指定した汎用レジスタのビット39-０の値を，rlで指定した汎用レジスタのビット5-0の値だけ算術右シフトします。結果をro'で指定した汎用レジスタのビット39-０に格納します。roおよびro'で指定した40ビットの値は２の補数表現，rlで指定した６ビットの値は正の整数値のデータ・フォーマットです。



算術右シフトとは符号拡張をともなう右シフトです。rlで指定した汎用レジスタのビット15-6は無視されます。

[実行サイクル]　１

[同時記述できる命令]

三項演算	二項演算	単項演算	並列ロード／ストア	レジスタ間転送	分　岐	条　件
×	×	×	○	×	×	×

[注　　意]　① 算術右シフトの結果アンダフローが発生しても，値は補正されません。

たとえば算術右シフトの過程で0xFF' FFFF' FFFFを算術右シフトしても，値は変化しません（０にはなりません）。

② 40ビット以上の算術右シフトをすると，符号がビット39-0に拡張されます。

ISRA　　ISRA

二項演算

ro' = ro SRA imm

命令名称	**イミーディエト算術右シフト**
ニモニック	**ro' = ro SRA imm（imm = 0-0x27（0-39））**

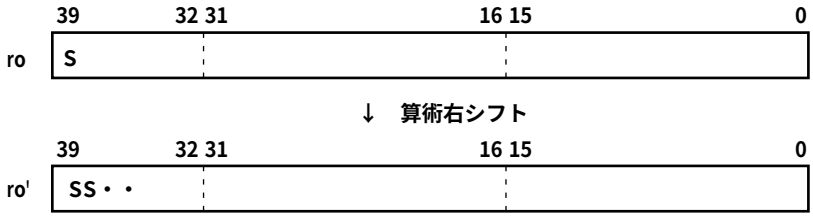
[記 述 例]　R1 = R0　SRA　0x10

[説　　明]　40ビット・データを算術右シフトする命令です。

roで指定した汎用レジスタのビット39-０の値を，６ビットのイミーディエト値だけ算術右シフトします。

算術右シフトとは符号拡張をともなう右シフトです。シフト量は，イミーディエト値のビット５-０が有効で，ビット15-６は無視されます。

結果をro'で指定した汎用レジスタのビット39-０に格納します。roおよびro'で指定した40ビットの値は２の補数表現，６ビットのイミーディエト値は，正の整数値のデータ・フォーマットです。



[実行サイクル]　１

[同時記述できる命令]

三項演算	二項演算	単項演算	並列ロード／ストア	レジスタ間転送	分　岐	条　件
×	×	×	×	×	×	×

[注　　意]　算術右シフトの結果アンダフローが発生しても，値は補正されません。

たとえば算術右シフトの過程で0xFF' FFFF' FFFFを算術右シフトしても，値は変化しません（０にはなりません）。

SRL　　SRL

二項演算

ro' = ro SRL rl

命令名称	**論理右シフト**
ニモニック	**ro' = ro SRL rl**

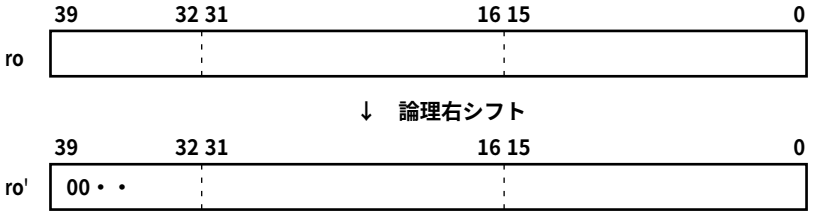
[記 述 例]　　R1 = R0　SRL　R4L

[説　　明]　　40ビット・データを論理右シフトする命令です。

roで指定した汎用レジスタのビット39-０の値を，rlで指定した汎用レジスタのビット５-０の値だけ論理右シフトします。

論理右シフトとはシフト量だけMSBから０を挿入する右シフトです。rlで指定した汎用レジスタのビット39-６は無視されます。

結果をro'で指定した汎用レジスタのビット39-０に格納します。roおよびro'で指定した40ビットの値は論理値，rlで指定した６ビットの値は，正の整数値のデータ・フォーマットです。



[実行サイクル]　１

[同時記述できる命令]

三項演算	二項演算	単項演算	並列ロード／ストア	レジスタ間転送	分　岐	条　件
×	×	×	○	×	×	×

[注　　意]　　40ビット以上の論理右シフトをしたとき，演算結果は０になります。

ISRL　　ISRL

二項演算

ro' = ro SRL imm

命令名称	**イミーディエト論理右シフト**
ニモニック	**ro' = ro SRL imm（imm = 0-0x27（0-39））**

[記 述 例]　R1 = R0　SRL　0x10

[説　　明]　40ビット・データを論理右シフトする命令です。

roで指定した汎用レジスタのビット39-０の値を，６ビットのイミーディエト値だけ論理右シフトします。

論理右シフトとはシフト量だけMSBから０を挿入する右シフトです。シフト量は，イミーディエト値のビット５-０が有効で，ビット15-６は無視されます。

結果をro'で指定した汎用レジスタのビット39-０に格納します。roおよびro'で指定した40ビットの値は論理値，６ビットのイミーディエト値は，正の整数値のデータ・フォーマットです。

[実行サイクル]　１

[同時記述できる命令]

三項演算	二項演算	単項演算	並列ロード／ストア	レジスタ間転送	分　岐	条　件
×	×	×	×	×	×	×

SLL　　SLL

二項演算

ro' = ro SLL rl

命令名称	**論理左シフト**
ニモニック	**ro' = ro SLL rl**

[記 述 例]　　R1 = R0　SLL　R4L

[説　　明]　　40ビット・データを論理左シフトする命令です。

roで指定した汎用レジスタのビット39-０の値を，rlで指定した汎用レジスタのビット5-0の値だけ論理左シフトします。

算術左シフトとはシフト量だけLSBから０を挿入する左シフトです。rlで指定した汎用レジスタのビット39-６は無視されます。

結果をro'で指定した汎用レジスタのビット39-０に格納します。roおよびro'で指定した40ビットの値は論理値，rlで指定した６ビットの値は，正の整数値のデータ・フォーマットです。

[実行サイクル]　１

[同時記述できる命令]

三項演算	二項演算	単項演算	並列ロード／ストア	レジスタ間転送	分　岐	条　件
×	×	×	○	×	×	×

[注　　意]　① 40ビット以上の論理左シフトをしたとき，演算結果は０になります。

② 論理左シフトの結果，オーバフローは発生しません。

ISLL　　ISLL

二項演算

ro' = ro SLL imm

命令名称	**イミーディエト論理左シフト**
ニモニック	**ro' = ro SLL imm（0-0x27（0-39））**

[記 述 例]　R1 = R0　SLL　0x10

[説　　明]　40ビット・データを論理左シフトする命令です。

roで指定した汎用レジスタのビット39-０の値を，６ビットのイミーディエト値だけ論理左シフトします。

論理左シフトとはシフト量だけLSBから０を挿入する左シフトです。シフト量は，イミーディエト値のビット5-0が有効で，ビット15-６は無視されます。

結果をro'で指定した汎用レジスタのビット39-０に格納します。roおよびro'で指定した40ビットの値は論理値，６ビットのイミーディエト値は，正の整数値のデータ・フォーマットです。

[実行サイクル]　1

[同時記述できる命令]

三項演算	二項演算	単項演算	並列ロード／ストア	レジスタ間転送	分　岐	条　件
×	×	×	×	×	×	×

[注　　意]　論理左シフトの結果，オーバフローは発生しません。

AND　　AND

二項演算

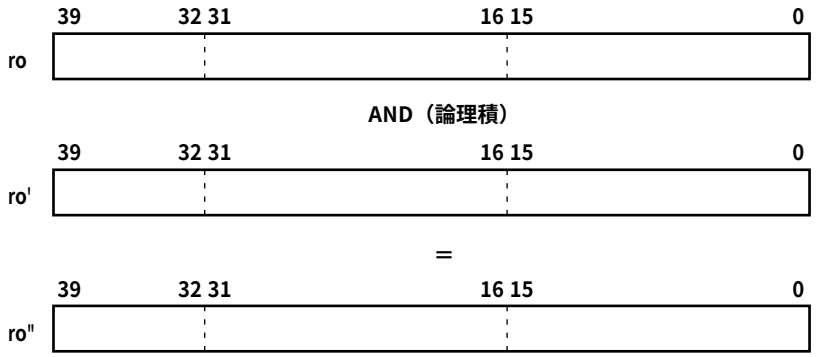
ro'' = ro & ro'

命令名称	**アンド**
ニモニック	**ro'' = ro & ro'**

[記 述 例]　　R1 = R0　&　R4

[説　　明]　　40ビット・データの論理積をとる命令です。

roで指定した汎用レジスタのビット39-０の値と，ro'で指定した汎用レジスタのビット39-0の値との論理積をとります。結果をro''で指定した汎用レジスタのビット39-０に格納します。ro, ro'およびro''で指定した40ビットの値は論理値です。



[実行サイクル]　１

[同時記述できる命令]

三項演算	二項演算	単項演算	並列ロード／ストア	レジスタ間転送	分　岐	条　件
×	×	×	○	×	×	×

IAND　　IAND

二項演算

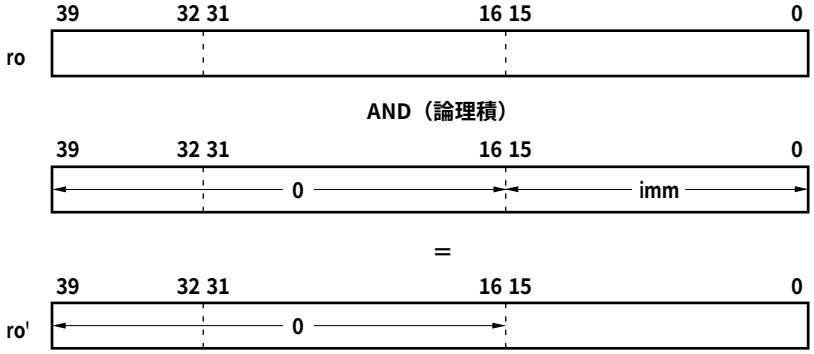
ro' = ro & imm

命令名称	**イミーディエト・アンド**
ニモニック	**ro' = ro & imm（imm = 0-0xFFFF（0-65535））**

[記 述 例]　　R1 = R0　&　0x10

[説　　明]　　40ビット・データの論理積をとる命令です。

roで指定した汎用レジスタのビット39-０の値と，ビット15-０の値を命令で設定しビット39-16に０拡張した40ビットのイミーディエト値との論理積をとります。結果をro'で指定した汎用レジスタのビット39-０に格納します。ro'で指定した汎用レジスタのビット39-16は０になります。roおよびro'で指定した40ビットの値，および16ビットのイミーディエト値は論理値です。



[実行サイクル]　１

[同時記述できる命令]

三項演算	二項演算	単項演算	並列ロード／ストア	レジスタ間転送	分　岐	条　件
×	×	×	×	×	×	×

OR　　OR

二項演算

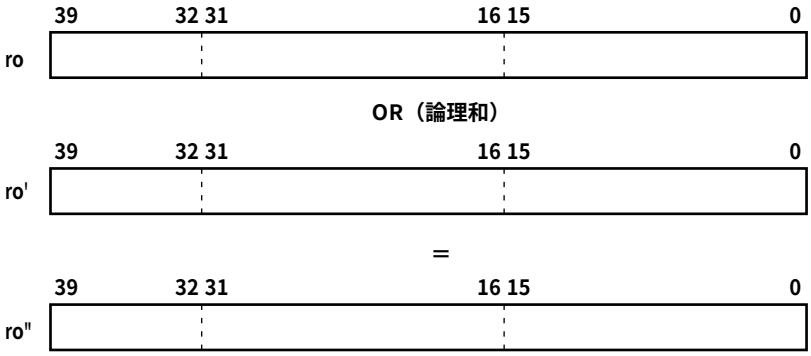
ro'' = ro　｜　ro'

命令名称	オア
ニモニック	ro'' = ro　｜　ro'

[記 述 例]　R1 = R0　｜　R4

[説　　明]　40ビット・データの論理和をとる命令です。

roで指定した汎用レジスタのビット39-０の値と，ro'で指定した汎用レジスタのビット39-０の値との論理和をとります。結果をro''で指定した汎用レジスタのビット39-０に格納します。ro, ro'およびro''で指定した40ビットの値は論理値です。



[実行サイクル]　１

[同時記述できる命令]

三項演算	二項演算	単項演算	並列ロード／ストア	レジスタ間転送	分　岐	条　件
×	×	×	○	×	×	×

IOR　　IOR

二項演算

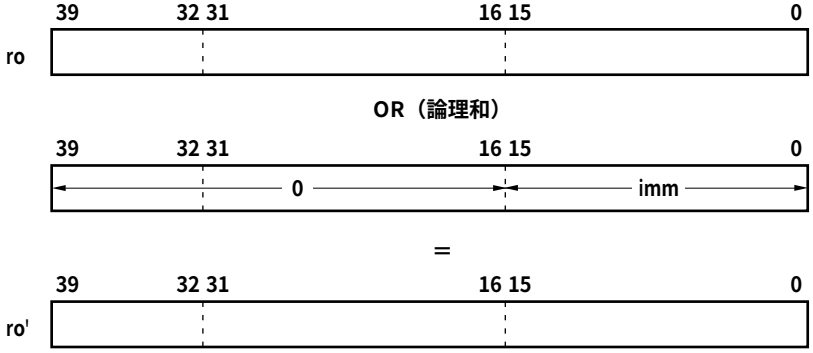
ro' = ro　｜　imm

命令名称	**イミーディエト・オア**
ニモニック	**ro' = ro　｜　imm（imm = 0-0xFFFF（0-65535））**

[記 述 例]　R1 = R0　｜　0x10

[説　　明]　40ビット・データの論理和をとる命令です。

roで指定した汎用レジスタのビット39-０の値と，ビット15-０の値を命令で設定しビット39-16に０拡張した40ビットのイミーディエト値との論理和をとります。結果をro'で指定した汎用レジスタのビット39-０に格納します。ro'で指定した汎用レジスタのビット39-16はroで指定した汎用レジスタのビット39-16と同じになります。roおよびro'で指定した40ビットの値，および16ビットのイミーディエト値は論理値です。



[実行サイクル]　1

[同時記述できる命令]

三項演算	二項演算	単項演算	並列ロード／ストア	レジスタ間転送	分　岐	条　件
×	×	×	×	×	×	×

XOR　　XOR

二項演算

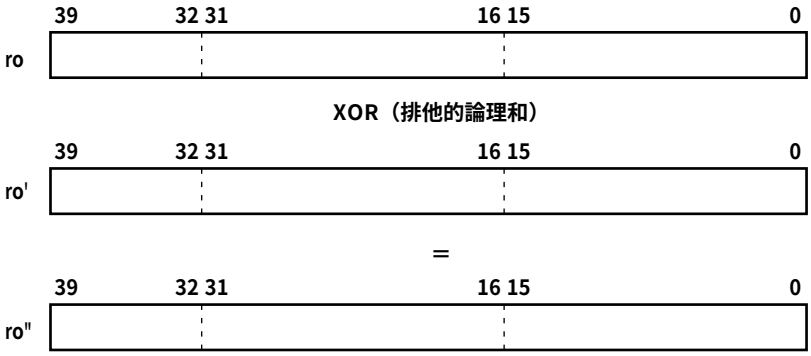
ro'' = ro∧ro'

命令名称	**イクスクルーシブ・オア**
ニモニック	**ro'' = ro∧ro'**

[記 述 例]　　R1 = R0∧R4

[説　　明]　　40ビット・データの排他的論理和をとる命令です。

roで指定した汎用レジスタのビット39-０の値と，ro'で指定した汎用レジスタのビット39-0の値との排他的論理和をとります。結果をro''で指定した汎用レジスタのビット39-0に格納します。ro, ro'およびro''で指定した40ビットの値は論理値です。



[実行サイクル]　１

[同時記述できる命令]

三項演算	二項演算	単項演算	並列ロード／ストア	レジスタ間転送	分　岐	条　件
×	×	×	○	×	×	×

IXOR　　IXOR

二項演算

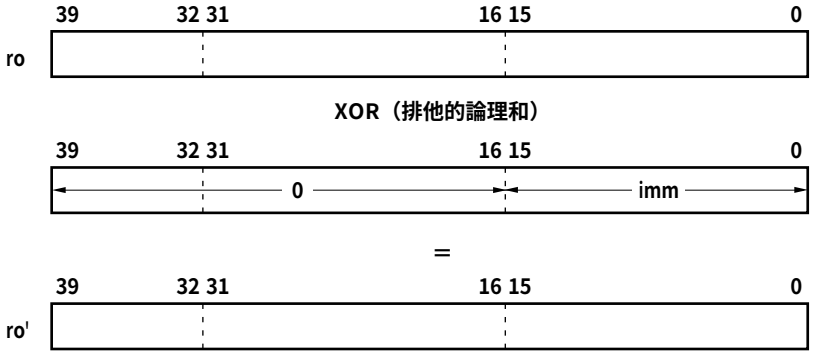
ro' = ro∧imm

命令名称	**イミーディエト・イクスクルーシブ・オア**
ニモニック	**ro' = ro∧imm（imm = 0-0xFFFF（0-65535））**

[記 述 例]　R1 = R0∧0x10

[説　　明]　40ビット・データの排他的論理和をとる命令です。

roで指定した汎用レジスタのビット39-０の値と，ビット15-０の値を命令で設定しビット39-16に０拡張した40ビットのイミーディエト値との排他的論理和をとります。結果をro'で指定した汎用レジスタのビット15-０に格納します。ro'で指定した汎用レジスタのビット39-16はroで指定した汎用レジスタのビット39-16と同じになります。roおよびro'で指定した40ビットの値，および16ビットのイミーディエト値は論理値です。



[実行サイクル]　1

[同時記述できる命令]

三項演算	二項演算	単項演算	並列ロード／ストア	レジスタ間転送	分　岐	条　件
×	×	×	×	×	×	×

LT　　LT

二項演算

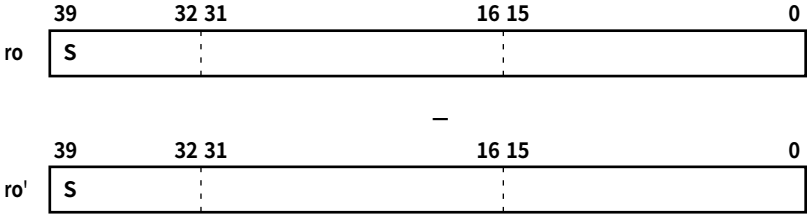
ro'' = LT（ro, ro'）

命令名称	**レスザン**
ニモニック	**ro'' = LT（ro, ro'）**

[記 述 例]　R2 = LT（R1, R4）

[説　　明]　２つの40ビット・データの大小比較をする命令です。

roで指定した汎用レジスタのビット39-０の値が，ro'で指定した汎用レジスタのビット39-０の値より小さい場合，ro''で指定した汎用レジスタのビット39-０に0x00' 0000' 0001を格納します。roで指定した汎用レジスタのビット39-０の値が，ro'で指定した汎用レジスタのビット39-０の値以上の場合，ro''で指定した汎用レジスタのビット39-０に0x00' 0000' 0000を格納します。ro, ro'で指定した40ビットの値は２の補数表現のデータ・フォーマットです。



[算術式表現]　if（ro＜ro'）　{ro'' = １；}

else　{ro'' = ０；}

[実行サイクル]　１

[同時記述できる命令]

三項演算	二項演算	単項演算	並列ロード／ストア	レジスタ間転送	分　岐	条　件
×	×	×	○	×	×	×

[注　　意]　ro－ro'のとき，オーバフローが発生するデータ同士の大小比較も正しく行います。このとき，オーバフロー・フラグは変化しません。

３.３　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・ファイルのうちから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・ファイルのうちから１レジスタを，任意に指定できます。

単項演算命令には，次の命令があります。

- クリア（CLR）
- インクリメント（INC）
- デクリメント（DEC）
- 絶対値（ABS）
- １の補数（NOT）
- ２の補数（NEG）
- クリップ（CLIP）
- 丸め（RND）
- 指数（EXP）
- 代入（PUT）
- 累加算（ACA）
- 累減算（ACS）
- 除算（DIV）

CLR

CLR

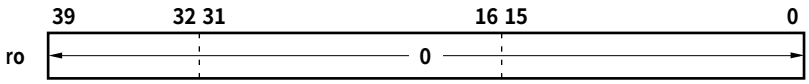
単項演算

CLR（ro）

命令名称	**クリア**
ニモニック	**CLR（ro）**

[記 述 例]　CLR（R0）

[説　　明]　roで指定した汎用レジスタのすべてのビットを０にする命令です。



[実行サイクル]　1

[同時記述できる命令]

三項演算	二項演算	単項演算	並列ロード／ストア	レジスタ間転送	分　岐	条　件
×	×	×	○	×	×	○

INC　　INC

単項演算

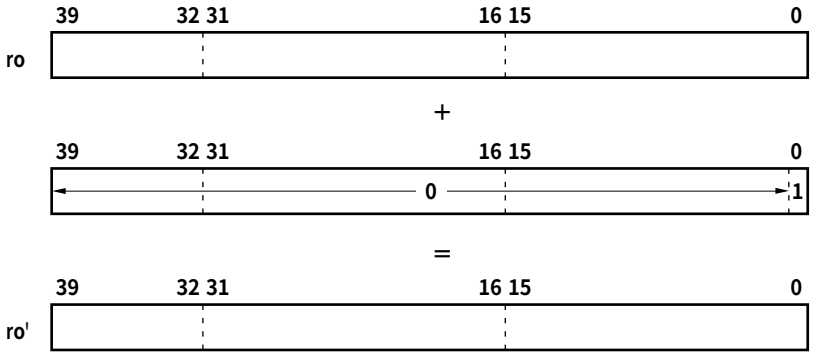
ro' = ro＋１

命令名称	**インクリメント**
ニモニック	**ro' = ro＋１**

[記 述 例]　　R1 = R1＋１

[説　　明]　　40ビット・データに１加算する命令です。

roで指定した汎用レジスタのビット39-０の値をインクリメントした結果をro'で指定した汎用レジスタのビット39-０に格納します。roおよびro'で指定した40ビット値は，２の補数表現のデータ・フォーマットです。



[実行サイクル]　１

[同時記述できる命令]

三項演算	二項演算	単項演算	並列ロード／ストア	レジスタ間転送	分　岐	条　件
×	×	×	○	×	×	○

[注　　意]　　インクリメントしてオーバフローが発生した場合，エラー・ステータス・レジスタのオーバフロー・フラグは“１”にセットされます（0x7F' FFFF' FFFF＋１→0x80' 0000' 0000）。

DEC　　DEC

単項演算

ro' = ro－１

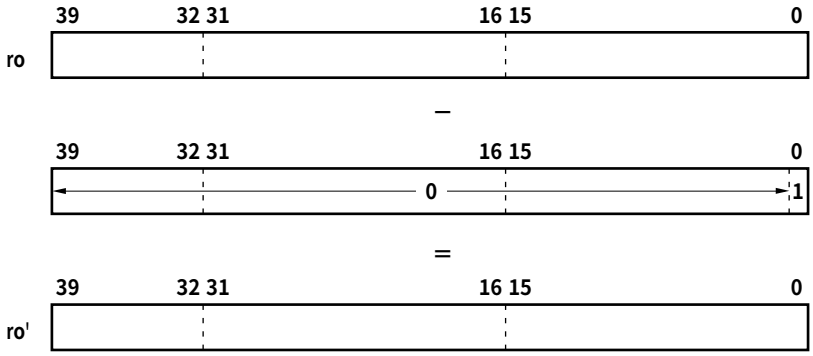
命令名称	デクリメント
ニモニック	ro' = ro－１

[記 述 例]　R1 = R1－１

[説　　明]　40ビット・データから１減算する命令です。

roで指定した汎用レジスタのビット39-０の値をデクリメントした結果をro'で指定した汎用レジスタのビット39-０に格納します。

roおよびro'で指定した40ビットの値は，２の補数表現のデータ・フォーマットです。



[実行サイクル]　１

[同時記述できる命令]

三項演算	二項演算	単項演算	並列ロード／ストア	レジスタ間転送	分　岐	条　件
×	×	×	○	×	×	○

[注　　意]　デクリメントしてオーバフローが発生した場合，エラー・ステータス・レジスタのオーバフロー・フラグは“１”にセットされます（0x80' 0000' 0000－1→0x7F' FFFF' FFFF）。

ABS　　ABS

単項演算

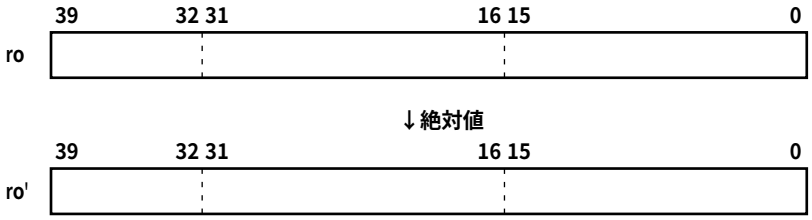
ro' = ABS（ro）

命令名称	**絶対値**
ニモニック	**ro' = ABS（ro）**

[記 述 例]　R2 = ABS（R1）

[説　　明]　40ビット・データの絶対値をとる命令です。

roで指定した汎用レジスタのビット39-０の値の絶対値を，ro'で指定した汎用レジスタのビット39-０に格納します。roおよびro'で指定した40ビットの値は，２の補数表現のデータ・フォーマットです。



[算術式表現]　if（ro＜０）　{ro' =－ro ; }

else　{ro' =ro ; }

[実行サイクル]　１

[同時記述できる命令]

三項演算	二項演算	単項演算	並列ロード／ストア	レジスタ間転送	分　岐	条　件
×	×	×	○	×	×	○

[注　　意]　roで指定した汎用レジスタの値が0x80' 0000' 0000の場合，エラー・ステータス・レジスタのオーバフロー・フラグは“１”にセットされます。

NOT　　NOT

単項演算

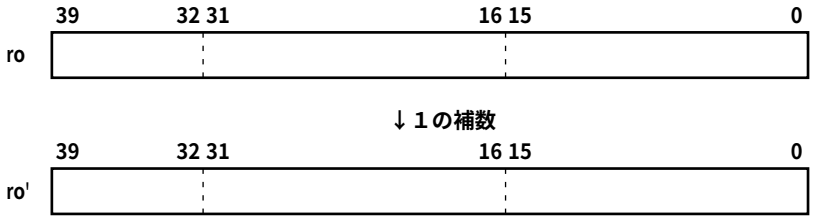
ro' = ~ro

命令名称	**１の補数**
ニモニック	**ro' = ~ro**

[記 述 例]　　R2 = ~R1

[説　　明]　　40ビット・データの１の補数をとる命令です。

roで指定した汎用レジスタのビット39-０の値の１の補数（各ビットの０と１を反転）をro'で指定した汎用レジスタのビット39-０に格納します。roおよびro'で指定した40ビットの値は，２の補数表現のデータ・フォーマットです。



[実行サイクル]　１

[同時記述できる命令]

三項演算	二項演算	単項演算	並列ロード／ストア	レジスタ間転送	分　岐	条　件
×	×	×	○	×	×	○

NEG　　NEG

単項演算

ro' = －ro

命令名称	**２の補数**
ニモニック	**ro' = －ro**

[記 述 例]　　R2 = －R1

[説　　明]　　40ビット・データの２の補数をとる命令です。

roで指定した汎用レジスタのビット39-０の値の２の補数（算術否定）をro'で指定した汎用レジスタのビット39-０に格納します。roおよびro'で指定した40ビットの値は，２の補数表現のデータ・フォーマットです。

[実行サイクル]　１

[同時記述できる命令]

三項演算	二項演算	単項演算	並列ロード／ストア	レジスタ間転送	分　岐	条　件
×	×	×	○	×	×	○

[注　　意]　　roで指定した汎用レジスタの値が0x80' 0000' 0000の場合，エラー・ステータス・レジスタのオーバフロー・フラグは“１”にセットされます。

CLIP

CLIP

単項演算

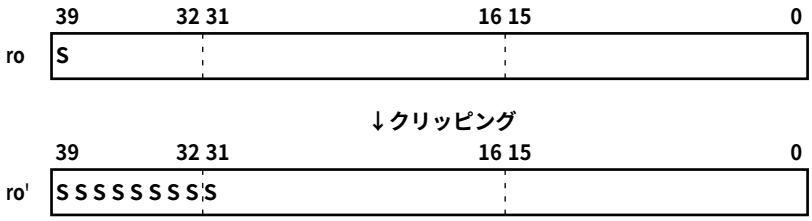
ro' = CLIP（ro）

命令名称	**クリップ**
ニモニック	**ro' = CLIP（ro）**

[記 述 例]　R2 = CLIP（R1）

[説　　明]　40ビット・データを32ビットにクリップする命令です。

roで指定した汎用レジスタのビット39-０の値を32ビット（ビット31-0）にクリップした値を，ro'で指定した汎用レジスタのビット39-０に格納します。ro'で指定した汎用レジスタのビット39-32には符号を拡張します。roおよびro'で指定した40ビットの値は，２の補数表現のデータ・フォーマットです。



[算術式表現]

```
if        (ro＞0x00' 7FFF' FFFF)      {ro' =0x00' 7FFF' FFFF ; }
else if   (ro＜0xFF' 8000' 0000)      {ro' =0xFF' 8000' 0000 ; }
else      {ro' =ro ; }
```

[実行サイクル]　１

[同時記述できる命令]

三項演算	二項演算	単項演算	並列ロード／ストア	レジスタ間転送	分　岐	条　件
×	×	×	○	×	×	○

[注　　意]　クリップによりサチュレーションが起こっても，オーバフロー・フラグは変化しません。

RND　　RND

単項演算

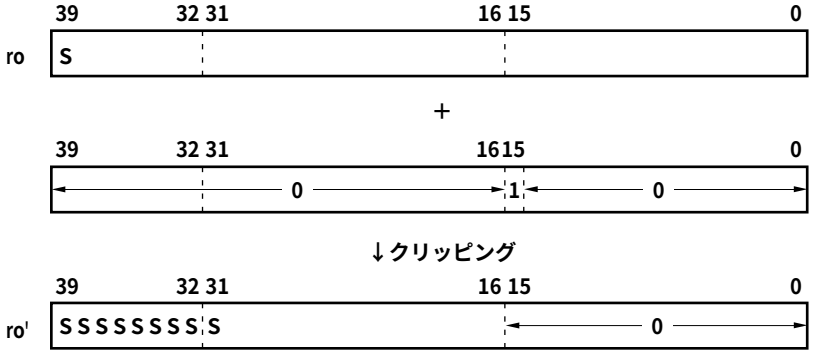
ro' = ROUND（ro）

命令名称	**丸め**
ニモニック	**ro' = ROUND（ro）**

[記 述 例]　R2 = ROUND（R1）

[説　　明]　40ビット・データを丸めおよびクリップし，16ビット・データにする命令です。

roで指定した汎用レジスタのビット15-０を四捨五入（丸め）した後16ビット（ビット31-16）にクリップした値を，ro'で指定した汎用レジスタのビット39-０に格納します。ここで，ビット15-０の四捨五入とは，ビット15のみ１の値を加算することです。ro'で指定した汎用レジスタのビット39-32には符号を拡張し，ビット15-０は０にします。roおよびro'で指定した40ビットの値は，２の補数表現のデータ・フォーマットです。



[算術式表現]　if（ro＞0x00' 7FFF' 0000）　{ro' =0x00' 7FFF' 0000；}

else if（ro＜0xFF' 8000' 0000）　{ro' =0xFF' 8000' 0000；}

else　{ro' =（ro＋0x8000）＆0xFF' FFFF' 0000；}

[実行サイクル]　１

[同時記述できる命令]

三項演算	二項演算	単項演算	並列ロード／ストア	レジスタ間転送	分　岐	条　件
×	×	×	○	×	×	○

[注　　意]　丸めによりサチュレーションが起こってもオーバフロー・フラグは変化しません。

EXP　　EXP

単項演算

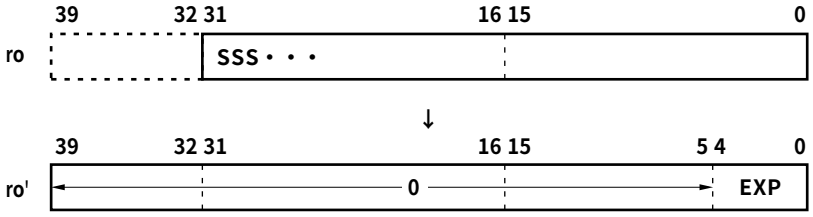
ro' = EXP（ro）

命令名称	指数
ニモニック	ro' = EXP（ro）

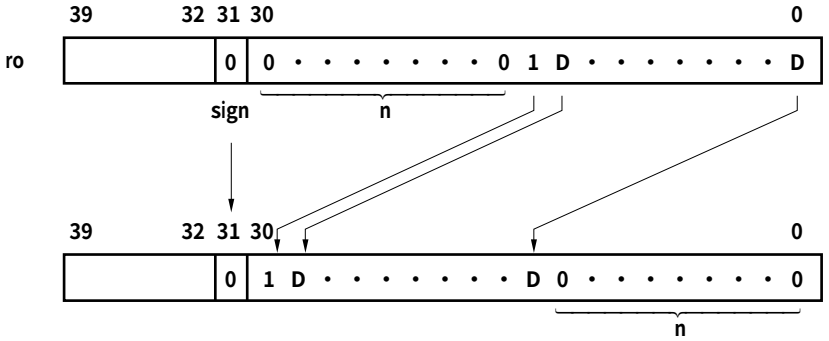
[記 述 例]　R2 = EXP（R1）

[説　明]　32ビットに正規化するための左シフト量を求める命令です。

roで指定した汎用レジスタのビット31-０の値を正規化するための左シフト量（０-31）を求めます。結果をro'で指定した汎用レジスタのビット39-０に格納します。ro'で指定した汎用レジスタのビット39-５は０にします。roで指定した32ビットの値およびro'で指定した40ビットの値は，２の補数表現のデータ・フォーマットです。



ここで正規化とは次のような操作のことをいいます。



上図の場合（ビット31 = ０：正）は，ビット30から下位側に向かって最初に（符号ビットと異なる）１が現れるまでの０の値を持つビット数を数えます。この命令では，ro'にこの数ｎを格納します。このあと，ｎビット左シフトする操作を正規化といいます。

```
r2 = exp（r1）　　;
r3 = r1 sll r2l　　;
```

EXP　　EXP

r1はこのr3に対してr1 = r3×2^{-n}となりますが，このｎを考慮すれば実質上浮動小数点としてr1の値を持つことができ，これによって小さな値の乗算も精度を保ったまま行うことができます。

[実行サイクル]　1

[同時記述できる命令]

三項演算	二項演算	単項演算	並列ロード／ストア	レジスタ間転送	分　岐	条　件
×	×	×	○	×	×	○

[注　　意]　① roで指定した汎用レジスタの値が0x00' 8000' 0000以上の場合，および0xFF' 7FFF' FFFF以下の場合，ro'の値は保証されません。

② roで指定した汎用レジスタの値が0x00' 0000' 0000の場合，ro'の値は0x00' 0000' 0000とします。

③ roで指定した汎用レジスタの値が0xFF' FFFF' FFFFの場合，ro'の値は0x1Fとします。

④ roで指定した汎用レジスタのビット31-０を評価対象とします。

PUT

PUT

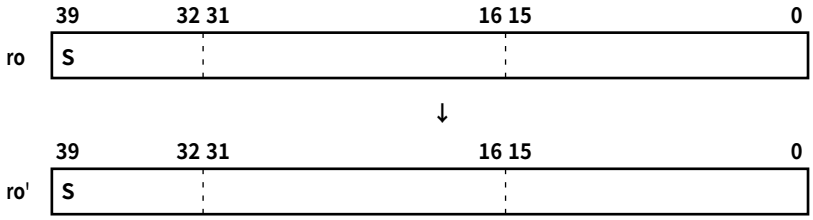
単項演算

ro' = ro

命令名称	**代入**
ニモニック	**ro' = ro**

[記 述 例]　R2 = R1

[説　　明]　40ビット・データをレジスタからレジスタへ代入する命令です。roで指定した汎用レジスタのビット39-０の値を，ro'で指定した汎用レジスタのビット39-０に格納します。roおよびro' で指定した40ビットの値は，２の補数表現のデータ・フォーマットです。



[実行サイクル]　１

[同時記述できる命令]

三項演算	二項演算	単項演算	並列ロード／ストア	レジスタ間転送	分　岐	条　件
×	×	×	○	×	×	○

ACA

ACA

単項演算

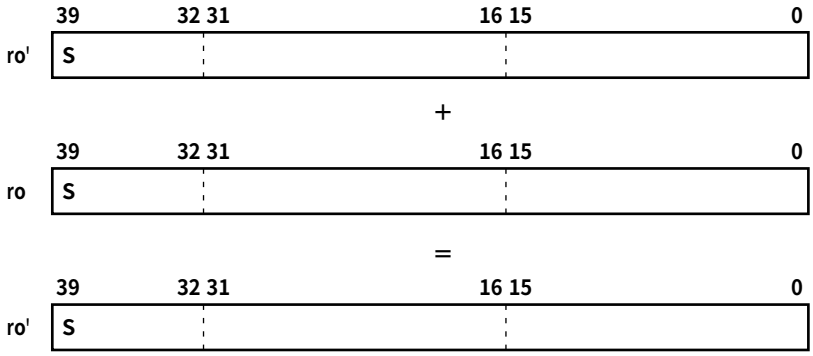
ro' ＋ = ro

命令名称	**累加算**
ニモニック	**ro' ＋ = ro**

[記 述 例]　R2＋ = R1

[説　　明]　２つの40ビット・データを加算する命令です。

roとro'で指定した汎用レジスタのビット39-0の値どうしを加算します。結果をro'で指定した汎用レジスタのビット39-0に格納します。roおよびro'で指定した40ビットの値は，２の補数表現のデータ・フォーマットです。



[実行サイクル]　１

[同時記述できる命令]

三項演算	二項演算	単項演算	並列ロード／ストア	レジスタ間転送	分　岐	条　件
×	×	×	○	×	×	○

[注　　意]　加算の結果オーバフローが発生した場合，エラー・ステータス・レジスタのオーバフロー・フラグは“１”にセットされます。

ACS　　ACS

単項演算

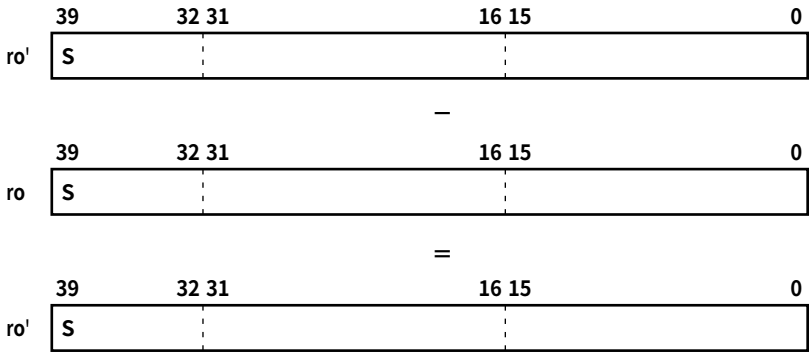
ro'− = ro

命令名称	**累減算**
ニモニック	**ro'− = ro**

[記 述 例]　R2− = R1

[説　　明]　40ビット・データから40ビット・データを減算する命令です。

ro'で指定した汎用レジスタのビット39-０の値から，roで指定した汎用レジスタのビット39-0の値を減算します。結果をro'で指定した汎用レジスタのビット39-０に格納します。roおよびro'で指定した40ビットの値は，２の補数表現のデータ・フォーマットです。



[実行サイクル]　１

[同時記述できる命令]

三項演算	二項演算	単項演算	並列ロード／ストア	レジスタ間転送	分　岐	条　件
×	×	×	○	×	×	○

[注　　意]　減算によりオーバフローが発生した場合，エラー・ステータス・レジスタのオーバフロー・フラグは“１”にセットされます。

DIV　　DIV

単項演算

ro'/ = ro

命令名称	**除算（１bit）**
ニモニック	**ro'/ = ro**

[記 述 例]　　**REP　16（16 bitレンジでの除算例）**

R2/ = R1

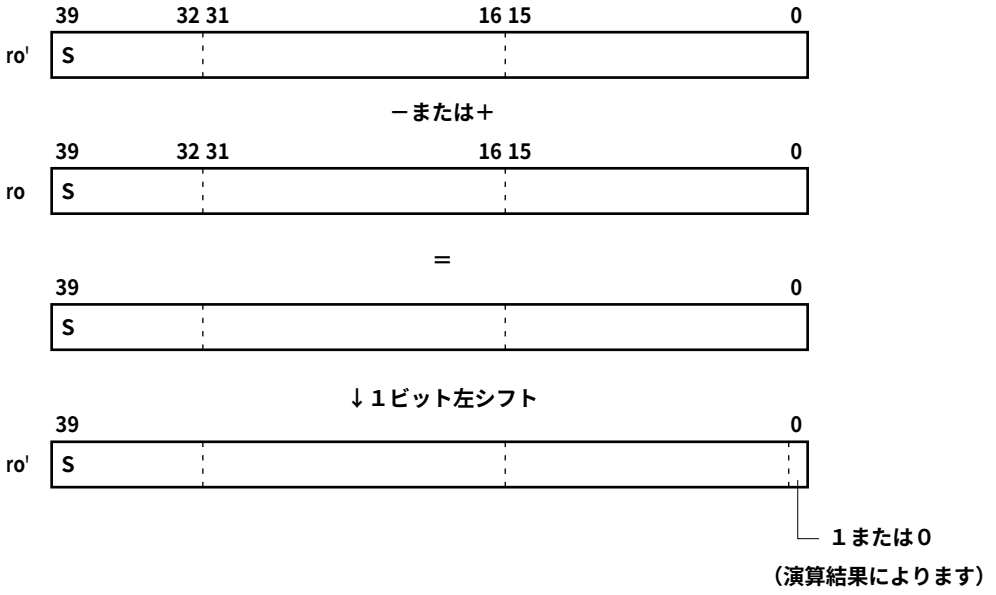
[説　　明]　　**１回の演算で，商を１ビットずつ求める，除算用の命令です。**

ro'で指定した汎用レジスタのビット39-0の値と，roで指定した汎用レジスタのビット39-0の値とで以下の演算をします。

ro'，roの値が同符合：ro'－ro

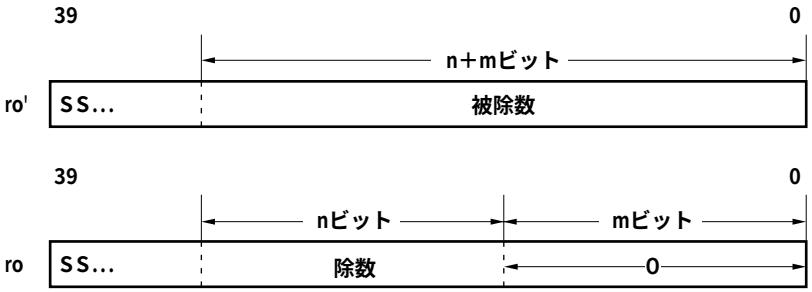
ro'，roの値が異符合：ro'＋ro

演算後の値を１ビット左シフトし，LSBには商の１ビットを挿入します。結果をro'で指定した汎用レジスタのビット39-0に格納します。roおよびro'で指定した40ビットの値は，２の補数表現のデータ・フォーマットです。

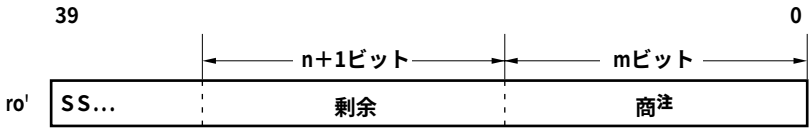


ro'およびroの値を以下のように設定します。

DIV　　DIV



除算命令をm回繰り返すと，ro'の以下の位置に商と剰余が求められます。



注　商 $= \dfrac{\text{被除数}}{|\text{除数}|}$

備考　roで指定した汎用レジスタのビット39-0の値は演算前と同じです。

[算術式表現]　if（sign（ro'）= =sign（ro））　{ro' = ro'－ro；}
else　{ro' = ro'＋ro}
if（sign（ro'）= =０）　{ro' = （ro'＜＜１）＋１；}
else　{ro' = ro'＜＜１；}

[実行サイクル]　１

[同時記述できる命令]

三項演算	二項演算	単項演算	並列ロード／ストア	レジスタ間転送	分　岐	条　件
×	×	×	○	×	×	○

[注　　意]　①　roと，ro'で同一の汎用レジスタを指定しないでください。
②　被除数の絶対値は，除数の絶対値より小さい値にしてください。

３.４　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／ストアがあります。

転送対象（入出力）は汎用レジスタ・ファイルから，任意に指定できます。

ロード／ストア命令には次の命令があります。

並列ロード／ストア（LSPA）－間接アドレシング

部分ロード／ストア（LSSE）

ダイレクト・アドレシング・ロード／ストア（LSDA）

即値インデクス・ロード／ストア（LSIM）

LSPA　　LSPA

ロード／ストア

[ro = ＊dpx_mod]　[ro' = ＊dpy_mod]
[＊dpx_mod = rh]　[＊dpy_mod = rh']
[ro = ＊dpx_mod]　[＊dpy_mod = rh]
[＊dpx_mod = rh]　[ro = ＊dpy_mod]

命令名称	**並列ロード／ストア**
ニモニック	**[ro = ＊dpx_mod]　[ro' = ＊dpy_mod]** **[＊dpx_mod = rh]　[＊dpy_mod = rh']** **[ro = ＊dpx_mod]　[＊dpy_mod = rh]** **[＊dpx_mod = rh]　[ro = ＊dpy_mod]**

[記 述 例]　R0 =＊DP0＋＋　　R1 =＊DP4＋＋

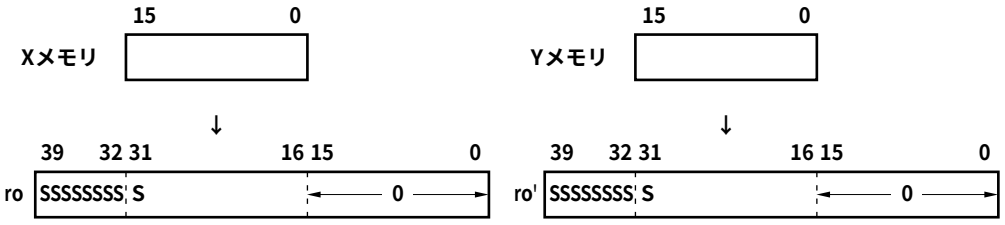
[説　　明]　間接アドレス指定したＸメモリ，Ｙメモリと汎用レジスタ間の16ビット・データのロード／ストアを行う命令です。演算命令（即値演算を除く）と同時記述できます。

記述方法には，次の４種類があります。

（1）[ro = ＊dpx_mod]　[ro' = ＊dpy_mod]

dpx_modで指定したＸメモリの番地の内容を，roで指定した汎用レジスタにロードします。同時に，dpy_modで指定したＹメモリの番地の内容を，ro'で指定した汎用レジスタにロードします。

メモリの内容は，汎用レジスタのビット31-16にロードします。ビット39-32には符合を拡張します（ビット31に入った値をビット39-32にコピーします）。ビット15-０はすべて０になります。

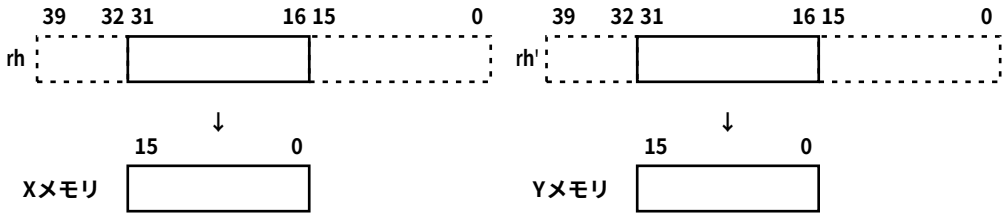


LSPA　　LSPA

（２）［＊dpx_mod = rh］［＊dpy_mod = rh'］

rhで指定した汎用レジスタの内容を，dpx_modでアドレス指定したＸメモリにロードします。同時に，rh'で指定した汎用レジスタの内容を，dpy_modでアドレス指定したＹメモリにロードします。

汎用レジスタの内容は，Ｘ，Ｙメモリのビット15-0にストアします。汎用レジスタのビット39-32とビット15-０は無視します。

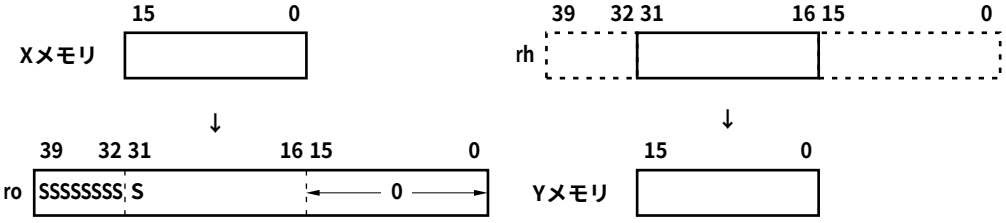


（３）［ro = ＊dpx_mod］［＊dpy_mod = rh］

dpx_modで指定したＸメモリの番地の内容を，roで指定した汎用レジスタにロードします。同時に，rhで指定した汎用レジスタの内容を，dpy_modでアドレス指定したＹメモリにロードします。

Ｘメモリの内容は汎用レジスタのビット31-16にロードします。ビット39-32には符合を拡張します。ビット15-０はすべて０になります。

汎用レジスタの内容はＹメモリのビット15-０にストアします。汎用レジスタのビット39-32とビット15-０は無視します。



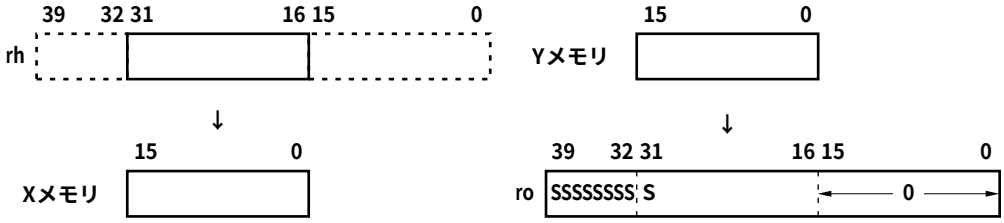
LSPA　　LSPA

（４）［＊dpx_mod = rh］［ro = ＊dpy_mod］

rhで指定した汎用レジスタの内容を，dpx_modでアドレス指定したＸメモリにロードします。同時に，dpy_modで指定したＹメモリの番地の内容を，roで指定した汎用レジスタにロードします。

汎用レジスタの内容はＸメモリのビット15-０にストアします。汎用レジスタのビット39-32およびビット15-０は無視されます。

Ｙメモリの内容は汎用レジスタのビット31-16にロードします。ビット39-32には符合を拡張します。ビット15-０はすべて０になります。



備考１．Ｘメモリと汎用レジスタ間のロード／ストアと，Ｙメモリと汎用レジスタ間のロード／ストアは同時に記述できるほか，それぞれ単独でも記述できます。データ・ポインタおよびデータ・ポインタのモディファイの方法はdpx_modとdpy_modを別々に指定できます。

２．ロード／ストアの対象となるアドレスは，！DPn＃＃を除いて，モディファイ前のデータ・ポインタの値です。ただし，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・ポインタをポインタに指定できるのは，設定した命令の２命令後からなので，設定した命令の直後の命令で指定しないでください。データ・ポインタのモディファイの結果は，直後の命令から有効になります。

［実行サイクル］　１

［同時記述できる命令］

三項演算	二項演算[注]	単項演算	並列ロード／ストア	レジスタ間転送	分　岐	条　件
○	○	○	×	×	×	×

注　即値（イミーディエト値）を使った演算を除く

LSPA　　LSPA

［注　　意］　①　Ｘメモリと汎用レジスタ間のロード／ストアと，Ｙメモリと汎用レジスタ間のロード／ストアを同時に指定する場合は，ＸメモリとＹメモリの以下の組み合わせを禁止します。

★　以下の組み合わせが発生した場合は，μPD7701×ファミリでは，バス・アクセス・エラー・フラグが“１”にセットされます。μPD77111ファミリには，バス・アクセス・エラー・フラグが存在しないので注意してください。

［μPD77016の場合］

外部領域と外部領域
ペリフェラル領域とペリフェラル領域　×

［μPD77015，77017，77018，77018A，77019の場合］

内部ROMと内部ROM
内部ROMと外部領域
外部領域と外部領域
ペリフェラル領域とペリフェラル領域　×

★　［μPD77110, 77111, 77112, 77113, 77114の場合］

外部領域と外部領域
ペリフェラル領域とペリフェラル領域　×

②　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は重複しないようにしてください。

③　同時記述した演算命令の演算結果を書き戻す汎用レジスタと，メモリからロードする汎用レジスタは重複しないでください。

④　データ・ポインタに値を設定した直後の命令で，同じデータ・ポインタの値をアドレスとしたロード／ストアを禁止します。

禁止例１：レジスタ間転送命令

```
DP0 = R4L；
R0 = ＊DP0 ＋＋；
```
×

禁止例２：即値設定命令

```
DP0 = 0x1234；
R0 = ＊DP0 ＋＋；
```
×

LSSE　　LSSE

ロード／ストア

[dest = ＊dpx_mod]　[dest' = ＊dpy_mod]
[dest = ＊dpx_mod]　[＊dpy_mod = source]
[＊dpx_mod = source]　[dest = ＊dpy_mod]
[＊dpx_mod = source]　[＊dpy_mod = source']

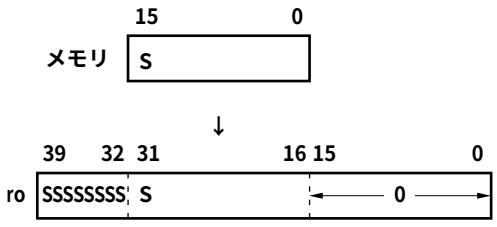
命令名称	部分ロード／ストア
ニモニック	[dest = ＊dpx_mod]　[dest' = ＊dpy_mod] [dest = ＊dpx_mod]　[＊dpy_mod = source] [＊dpx_mod = source]　[dest = ＊dpy_mod] [＊dpx_mod = source]　[＊dpy_mod = source'] 指定できるレジスタは以下のとおりです。複数のレジスタのうちいずれかひとつを記述してください。 dest，dest' =　{ro，reh，re，rh，rl} source，source' =　{re，rh，rl}

[記 述 例]　R0EH = ＊DP0＋＋　R0L = ＊DP4＋＋

[説　　明]　間接アドレス指定したＸメモリ，Ｙメモリと汎用レジスタ間の16ビット・データのロード／ストアを行う命令です。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ます。

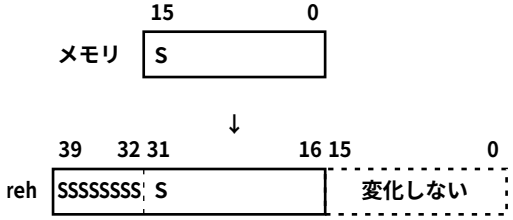
（１）dpx_modまたはdpy_modで指定したメモリから，destまたはdest'で指定した汎用レジスタへのロードの方法には次の５種類があります。

（ａ）destまたはdest'にR0-R7を指定した場合には，dpx_modまたはdpy_modで指定したメモリの値を汎用レジスタのビット31-16にロードし，ビット39-32には符合を拡張し，ビット15-０はすべて０になります。

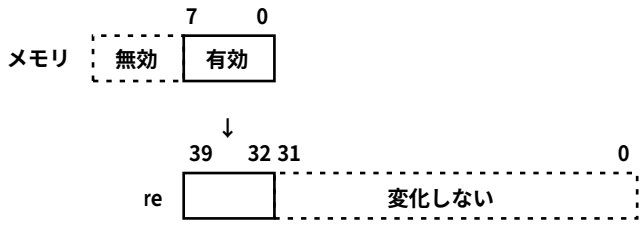


LSSE　　LSSE

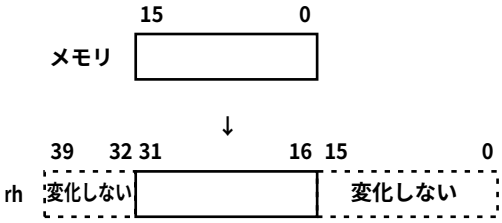
（ｂ）destまたはdest'にR0EH-R7EHを指定した場合には，dpx_modまたはdpy_modで指定したメモリの値を汎用レジスタのビット31-16にロードし，ビット39-32には符合を拡張します。汎用レジスタのビット15-０は変化しません。



（ｃ）destまたはdest'にR0E-R7Eを指定した場合には，dpx_modまたはdpy_modで指定したメモリの値の下位８ビットを汎用レジスタのビット39-32にロードします。汎用レジスタのビット31-０は変化しません。メモリの値の上位８ビットは無視します。

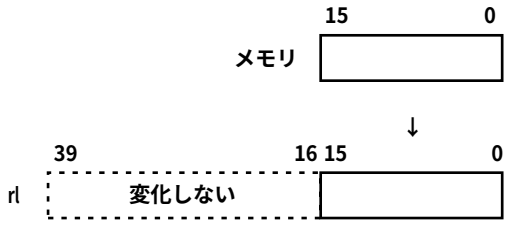


（ｄ）destまたはdest'にR0H-R7Hを指定した場合には，dpx_modまたはdpy_modで指定したメモリの値を汎用レジスタのビット31-16にロードします。汎用レジスタのビット39-32とビット15-０は変化しません。



LSSE　　LSSE

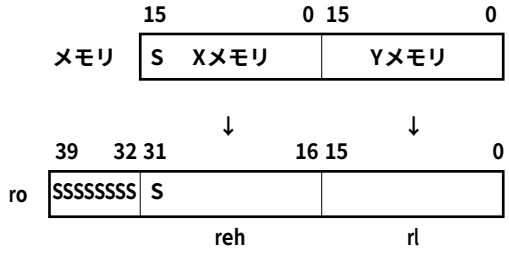
（ｅ）destまたはdest'にR0L-R7Lを指定した場合には，dpx_modまたはdpy_modで指定したメモリの値を汎用レジスタのビット15-０にロードします。汎用レジスタのビット39-16は変化しません。



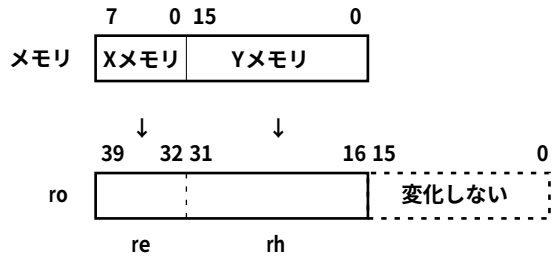
１つのレジスタ（ro）に対して，ＸメモリとＹメモリの両方からの転送を同時に行うことができます。ただし，転送先の汎用レジスタは，以下の組み合わせに限られます。

また，sourceとなる汎用レジスタのＸメモリとＹメモリを入れ替えることも可能です。

●RnEHとRnLの場合（例　[reh = ＊dpx_mod]　[rl = ＊dpy_mod]）

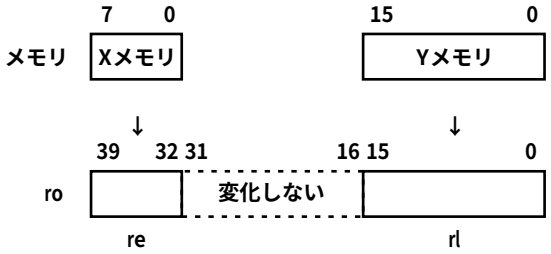


●RnEとRnHの場合（例　[re = ＊dpx_mod]　[rh = ＊dpy_mod]）

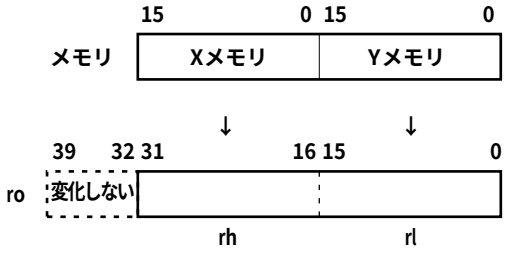


LSSE　　LSSE

●RnEとRnLの場合（例　[re = ＊dpx_mod]　[rl = ＊dpy_mod]）



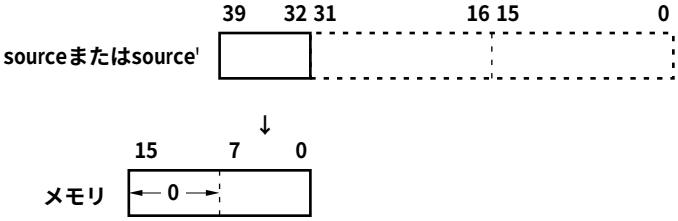
●RnHとRnLの場合（例　[rh = ＊dpx_mod]　[rl = ＊dpy_mod]）



注意　他の組み合わせは指定しないでください。

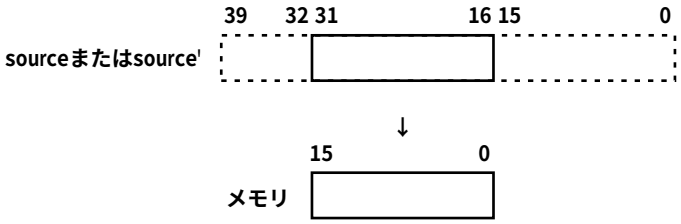
（２）sourceまたはsource'で指定した汎用レジスタから，dpx_modまたはdpy_modで指定したメモリへストアする方法は次の３種類です。

（ａ）sourceまたはsource'にR0E-R7Eを指定した場合には，汎用レジスタのビット39-32の値を，dpx_modまたはdpy_modで指定したメモリの下位８ビットにストアします。メモリの上位８ビットは０にします。汎用レジスタのビット31-０は無視します。

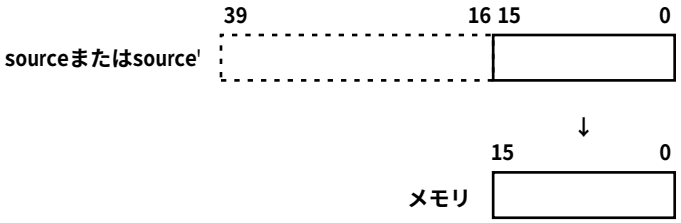


LSSE　　LSSE

（ｂ）sourceまたはsource'にR0H-R7Hを指定した場合には，汎用レジスタのビット31-16の値を，dpx_modまたはdpy_modで指定したメモリにストアにします。汎用レジスタのビット39-32とビット16-０は無視します。



（ｃ）sourceまたはsource'にR0L-R7Lを指定した場合には，汎用レジスタのビット15-０の値を，dpx_modまたはdpy_modで指定したメモリにストアします。汎用レジスタのビット39-16は無視します。



備考１．Ｘメモリと汎用レジスタ間のロード／ストアと，Ｙメモリと汎用レジスタ間のロード／ストアは同時に記述できるほか，それぞれ単独でも記述できます。

２．データ・ポインタおよびデータ・ポインタのモディファイの方法はdpx_modとdpy_modを独立に指定できます。

３．ロード／ストアの対象となるアドレスは，！DPn＃＃を除いて，モディファイ前のデータ・ポインタの値です。データ・ポインタのモディファイ結果は，直後の命令から有効になります。

[実行サイクル]　１

[同時記述できる命令]

三項演算	二項演算	単項演算	並列ロード／ストア	レジスタ間転送	分　岐	条　件
×	×	×	×	×	×	×

LSSE　　LSSE

［注　　意］　①　Ｘメモリと汎用レジスタ間のロード／ストアと，Ｙメモリと汎用レジスタ間のロード／ストアを同時に指定する場合，ＸメモリとＹメモリの以下の組み合わせを禁止します。以下の
★ 組み合わせが発生した場合は，μPD7701×ファミリでは，バス・アクセス・エラー・フラグが“１”にセットされます。μPD77111ファミリには，バス・アクセス・エラー・フラグが存在しないので注意してください。

［μPD77016の場合］
外部領域と外部領域
ペリフェラル領域とペリフェラル領域　×

［μPD77015，77017，77018，77018A，77019の場合］
内部ROMと内部ROM
内部ROMと外部領域
外部領域と外部領域　×
ペリフェラル領域とペリフェラル領域

★ ［μPD77110, 77111, 77112, 77113, 77114の場合］
外部領域と外部領域
ペリフェラル領域とペリフェラル領域　×

②　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・ポインタをポインタに指定できるのは，設定した命令の２命令後からなので，設定した命令の直後の命令で指定しないでください。データ・ポインタに値を設定した直後の命令で，同じデータ・ポインタの値をアドレスとしたロード／ストアは禁止します。

禁止例１：レジスタ間転送命令

```
DP0 = R4L；
R0E = ＊DP0 ＋＋；
```
×

禁止例２：即値設定命令

```
DP0 = 0x1234；
R0H = ＊DP0 ＋＋；
```
×

LSDA　　LSDA

ロード／ストア

dest = ＊addr

＊addr = source

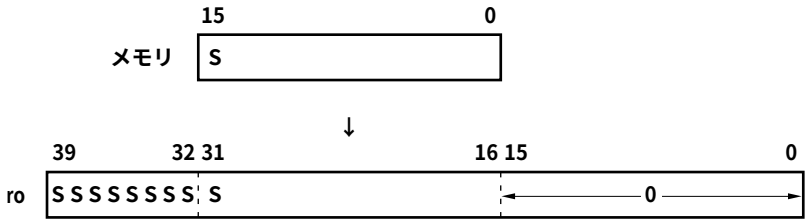
命令名称	ダイレクト・アドレシング・ロード／ストア
ニモニック	dest = ＊addr ＊addr = source addrに指定できるアドレスの範囲は以下のとおりです。 Ｘメモリ0-0xFFFF（0-65535）：Ｘ（Ｘメモリ） Ｙメモリ0-0xFFFF（0-65535）：Ｙ（Ｙメモリ） 指定できるレジスタは以下のとおりです。複数のレジスタのうちいずれかひとつを記述してください。 dest = {ro，reh，re，rh，rl} source = {re，rh，rl}

[記 述 例]　R0H = ＊0x1234：X

[説　　明]　直接アドレス指定したＸメモリまたはＹメモリと，汎用レジスタ間の16ビット・データのロード／ストアを行う命令です。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ます。

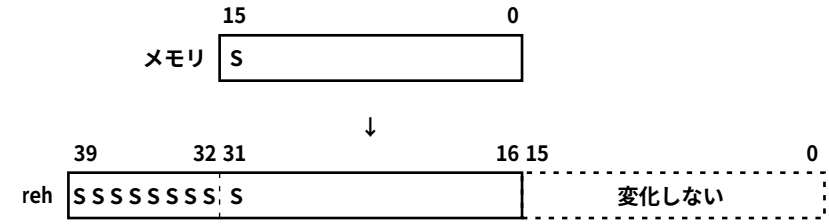
（１）addrで指定したメモリからdestで指定した汎用レジスタへのロードの方法には次の５種類があります。いずれの場合も汎用レジスタにロードした値は，直後の命令から有効になります。

（ａ）destにR0-R7を指定した場合には，addrで指定したメモリの値を汎用レジスタのビット31-16にロードします。ビット39-32には符合を拡張し，ビット15-０はすべて０になります。

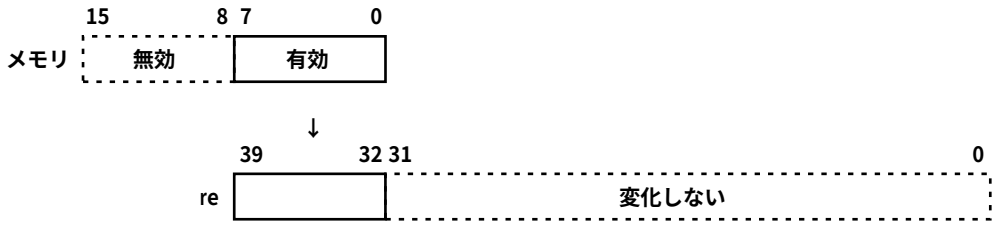


LSDA　　LSDA

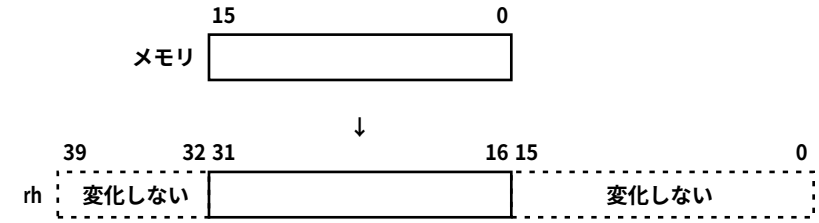
（ｂ）destにR0EH-R7EHを指定した場合には，addrで指定したメモリの値を汎用レジスタのビット31-16にロードします。ビット39-32には符合を拡張します。汎用レジスタのビット15-０は変化しません。



（ｃ）destにR0E-R7Eを指定した場合には，addrで指定したメモリの値の下位８ビットを汎用レジスタのビット39-32にロードします。汎用レジスタのビット31-０は変化しません。メモリの値の上位８ビットは無視されます。

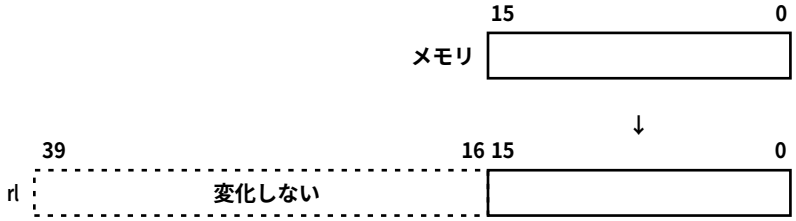


（ｄ）destにR0H-R7Hを指定した場合には，addrで指定したメモリの値を汎用レジスタのビット31-16にロードします。汎用レジスタのビット39-32および15-０は変化しません。



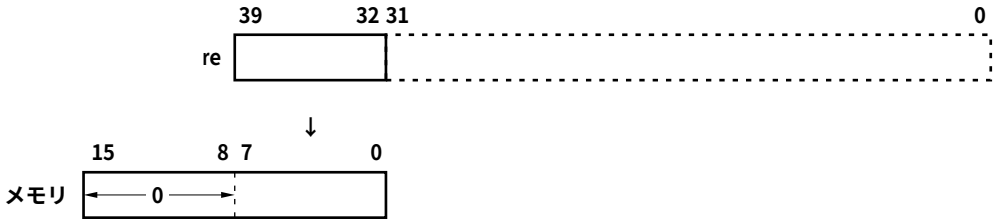
LSDA　　LSDA

（ｅ）destにR0L-R7Lを指定した場合には，addrで指定したメモリの値を汎用レジスタのビット15-０にロードします。汎用レジスタのビット39-16は変化しません。

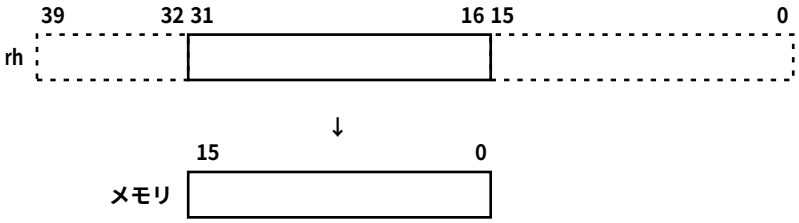


（２）sourceで指定した汎用レジスタから，addrで指定したメモリへのストア方法には次の３種類があります。

（ａ）sourceにR0E-R7Eを指定した場合には，汎用レジスタのビット39-32の値を，addrで指定したメモリの下位８ビットにストアします。メモリの上位８ビットは０にします。汎用レジスタのビット31-０は無視されます。

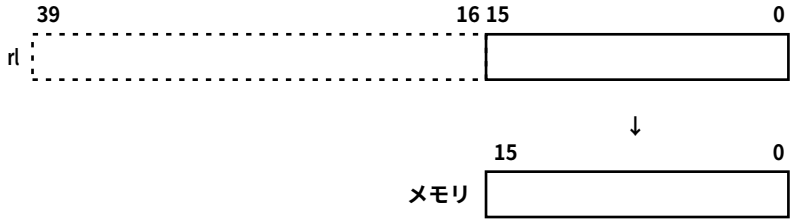


（ｂ）sourceにR0H-R7Hを指定した場合には，汎用レジスタのビット31-16の値を，addrで指定したメモリにストアします。汎用レジスタのビット39-32とビット15-０は無視されます。



LSDA　　LSDA

（ｃ）sourceにR0L-R7Lを指定した場合には，汎用レジスタのビット15-０の値を，addrで指定したメモリにストアします。汎用レジスタのビット39-16は無視されます。



[実行サイクル]　１

[同時記述できる命令]

三項演算	二項演算	単項演算	並列ロード／ストア	レジスタ間転送	分　岐	条　件
×	×	×	×	×	×	×

LSIM　　LSIM

ロード／ストア

dest = ＊dp_imm

＊dp_imm = source

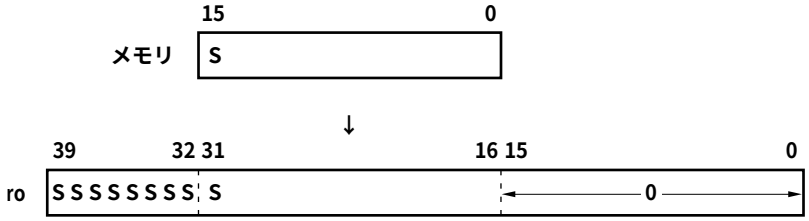
命令名称	**即値インデクス・ロード／ストア**
ニモニック	**dest = ＊dp_imm** **＊dp_imm = source（imm = 0-0xFFFF）** **指定できるレジスタは以下のとおりです。複数のレジスタのうちいずれかひとつを記述してください。** dest =　{ro，reh，re，rh，rl} source =　{re，rh，rl}

[記 述 例]　R0 = ＊DP0＃＃0x10

[説　　明]　間接アドレス指定したＸメモリまたはＹメモリと，汎用レジスタ間の16ビット・データのロード／ストアを行う命令です。

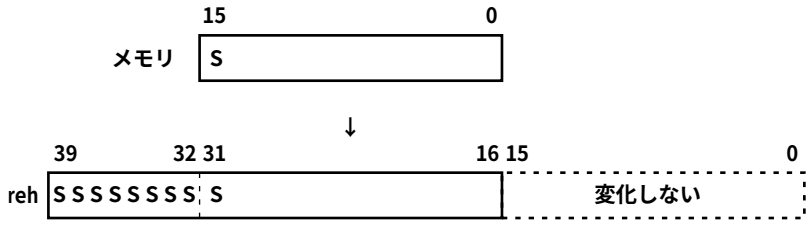
（１）dp_immで指定したメモリからdestで指定した汎用レジスタへのロードの方法には次の５種類があります。

（ａ）destにR0-R7を指定した場合には，dp_immで指定したメモリの値を汎用レジスタのビット31-16にロードします。ビット39-32には符合を拡張し，ビット15-０は０にします。

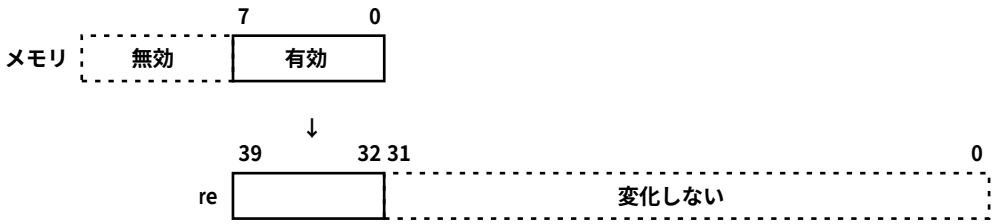


LSIM　　LSIM

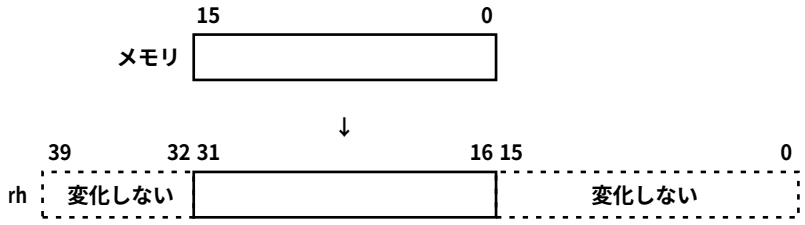
（ｂ）destにR0EH-R7EHを指定した場合には，dp_immで指定したメモリの値を汎用レジスタのビット31-16にロードします。ビット39-32には符合を拡張します。汎用レジスタのビット15-０は変化しません。



（ｃ）destにR0E-R7Eを指定した場合には，dp_immで指定したメモリの値の下位８ビットを汎用レジスタのビット39-32にロードします。汎用レジスタのビット31-０は変化しません。メモリの値の上位８ビットは無視します。

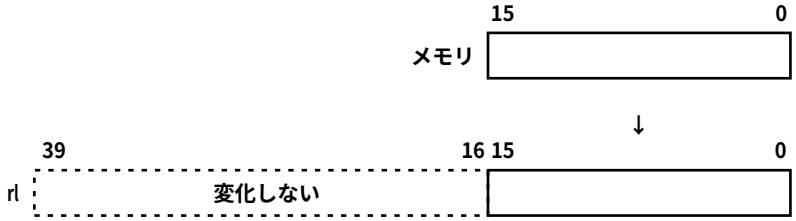


（ｄ）destにR0H-R7Hを指定した場合には，dp_immで指定したメモリの値を汎用レジスタのビット31-16にロードします。汎用レジスタのビット39-32および15-０は変化しません。



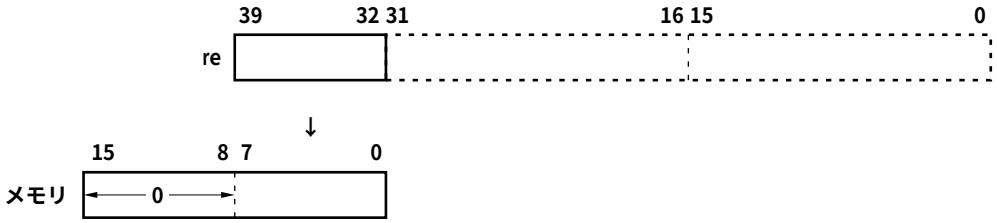
LSIM　　LSIM

（ｅ）destにR0L-R7Lを指定した場合には，dp_immで指定したメモリの値を汎用レジスタのビット15-０にロードします。汎用レジスタのビット39-16は変化しません。

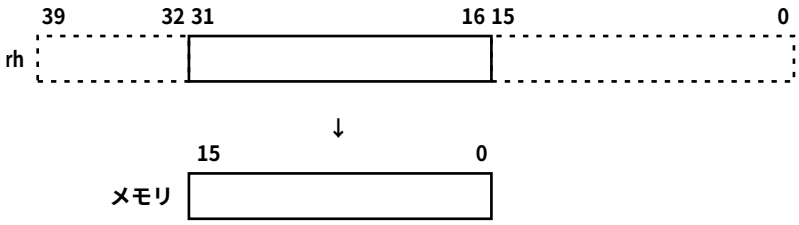


（２）sourceで指定した汎用レジスタから，dp_immで指定したメモリへのストア方法には次の３種類があります。

（ａ）sourceにR0E-R7Eを指定した場合には，汎用レジスタのビット39-32の値を，dp_immで指定したメモリの下位８ビットにストアします。メモリの上位８ビットは０にします。汎用レジスタのビット31-０は無視されます。

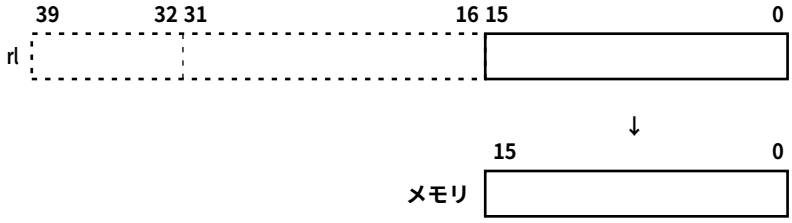


（ｂ）sourceにR0H-R7Hを指定した場合には，汎用レジスタのビット31-16の値を，dp_immで指定したメモリにストアします。汎用レジスタのビット39-32およびビット15-０は無視します。



LSIM　　LSIM

（ｃ）sourceにR0L-R7Lを指定した場合には，汎用レジスタのビット15-０の値を，dp_imm で指定したメモリにストアします。汎用レジスタのビット39-16は無視します。



備考　ロード／ストアの対象となるアドレスは，!DPn##を除いて，モディファイ前のデータ・ポインタの値です。ただし，レジスタ間転送命令または即値設定命令で設定したデータ・ポインタをポインタに指定できるのは，設定した命令の２命令後からなので，設定した命令の直後の命令で指定しないでください。データ・ポインタのモディファイ結果は，直後の命令から有効になります。

[実行サイクル]　１

[同時記述できる命令]

三項演算	二項演算	単項演算	並列ロード／ストア	レジスタ間転送	分　岐	条　件
×	×	×	×	×	×	×

[注　　意]　データ・ポインタに値を設定した直後の命令で，同じデータ・ポインタの値をアドレスとしたロード／ストアは禁止します。

禁止例１：レジスタ間転送命令

```
DP0 = R4L；
R0 = ＊DP0##0xABCD；
```

×

禁止例２：即値設定命令

```
DP0 =0 x1234；
R0 = ＊DP0##0xABCD；
```

×

３.５　レジスタ間転送命令

メイン・バスを通じて汎用レジスタとほかのレジスタとの転送を指定する命令です。

MOV　　MOV

レジスタ間転送

dest = rl

rl = source

命令名称	**レジスタ間転送**
ニモニック	**dest = rl** **rl = source** dest, sourceには汎用レジスタ以外の全レジスタのうちいずれかを指定してください。

[記 述 例]　DP0 = R0L

[説　　明]　メイン・バスを通じて汎用レジスタとその他のレジスタ間の転送を行う命令です。

sourceに指定したレジスタの値をメイン・バス経由でdestに指定したレジスタに下詰めで転送します。

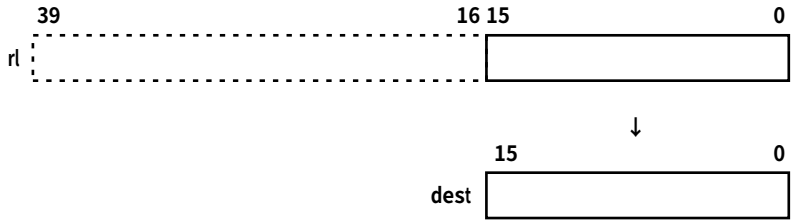
対象となるレジスタを次に示します。

レジスタ名	アセンブラ予約名
汎用レジスタ	R0L-R7L（R0-R7のＬパート）
データ・ポインタ	DP0-DP7
インデクス・レジスタ	DN0-DN7
モジュロ・レジスタ	DMX, DMY
スタック	STK
スタック・ポインタ	SP
ループ・カウンタ注	LC
ループ・スタック（LSTK）	LSR1, LSR2, LSR3
ループ・スタック・ポインタ	LSP
ステータス・レジスタ	SR
割り込み許可フラグ・スタック・レジスタ	EIR
エラー・ステータス・フラグ	ESR

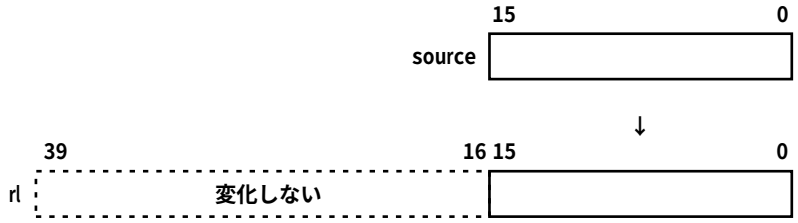
注　destに指定することはできません。

MOV　　MOV

（１）dest = rlのときは，汎用レジスタのビット15-0の値を，destに指定したレジスタに転送します。destに指定したレジスタの有効ビット長がmビット（m＜16）の場合，汎用レジスタのビット15-mは無視されます。



（２）rl = sourceのときは，sourceに指定したレジスタの値を汎用レジスタのビット15-0に転送します。汎用レジスタのビット39-16は変化しません。sourceに指定したレジスタのビット長がエラー・ステータス・レジスタのようにｎビット（n＜16）の場合，汎用レジスタのビット15-nは不定です。



[実行サイクル]　1

[同時記述できる命令]

三項演算	二項演算	単項演算	並列ロード／ストア	レジスタ間転送	分　岐	条　件[注]
×	×	×	×	×	×	○

注　DP0-DP7,DN0-DN7,DMX,DMYをdestまたはsourceに指定した場合に限ります。

MOV　　MOV

［注　　意］　① destにDP0-DP7を指定した場合には，転送結果をロード／ストアのポインタに指定できるのは２命令後からです。

② DMXおよびDMYに0x8000-0xFFFFの値を転送したあとは，モジュロ・インデクス加算をしないでください。

③ モジュロ・インデクス加算（DPn%%）を行うときは，DMX, DMYに1-0x7FFFの範囲の値を転送してください。０または0x8000以上の値を転送したとき，DPn%%は正しく動作しません。

④ LCのビット15はループ・フラグです。このフラグを書き換えないでください。

⑤ SPに設定できる値は0-0xFまでです。このレジスタに0x10-0xFFFFを設定しないでください。

⑥ LSPに設定できる値は0-4までです。このレジスタに0x5-0xFFFF を設定しないでください。

⑦ RETおよびRETI命令を，STKまたはSPにロード／ストアするレジスタ間転送命令の直後に記述しないでください。

３.６　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即値（イミーディエト値）を設定する命令です。

LDI　　LDI

即値設定

dest = imm

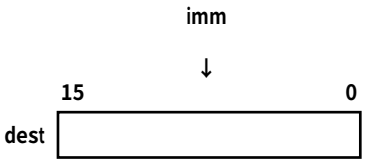
命令名称	**即値設定**
ニモニック	**dest = imm（imm = 0-0xFFFF（0-65535））** **dest = {rl, dp, dn, dm}**

[記 述 例]　DP0 = 0x1234

[説　　明]　destに指定したレジスタに，16ビットのイミーディエト値を下詰めで設定する命令です。

destにR0L-R7Lを指定した場合，汎用レジスタのビット15-0にイミーディエト値を設定します。ビット39-16は変化しません。

destにDP0-DP7,DN0-DN7,DMXまたはDMYを指定した場合，16ビットのイミーディエト値がそのままレジスタの値になります。



[実行サイクル]　1

[同時記述できる命令]

三項演算	二項演算	単項演算	並列ロード／ストア	レジスタ間転送	分　岐	条　件
×	×	×	×	×	×	×

[注　　意]　① 設定した値は直後の命令から有効になります。ただし，destにDP0-DP7を指定した場合には，設定結果をロード／ストアのポインタに指定できるのは，２命令後からであり，直後のロード／ストア命令のポインタに指定することはできません。

② モジュロ・インデクス加算をするときにDMXおよびDMYに０および0x8000-0xFFFFを設定しないでください。

３.７　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。分岐命令には，次の命令があります。

ジャンプ（JMP）
レジスタ間接ジャンプ（JREG）
サブルーチン・コール（CALL）
レジスタ間接サブルーチン・コール（CREG）
リターン（RET）
割り込みリターン（RETI）

JMP　　JMP

分岐

JMP imm

命令名称	**ジャンプ**
ニモニック	**JMP imm（imm = 0-0xFFFF（0-65535））**

[記 述 例]　JMP 0x1234

[説　　明]　μPD77016ファミリのインストラクション・メモリ空間のすべてのアドレスに分岐できる命令です。実行には２サイクルが必要です。

μPD77016ファミリには条件ジャンプ命令（フラグの値が０または１なら分岐など）はありません。条件ジャンプ命令を行いたいときは，以下のように条件命令と分岐命令を組み合わせて記述します。命令実行に必要なサイクル数は，条件命令の条件判断が成立して分岐を行うときは３サイクル，条件が成立しないときは分岐を行わないので１サイクルとなります。

IF（R0 = = 0）JMP LABEL　または
IF（R2＞0）JMP DP0　（JMP DP0についてはこのあとの**レジスタ間接ジャンプ**で説明します。）

備考　μPD77016ファミリのジャンプ命令およびサブルーチン・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およびリンカが行いますので，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場合は，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

[実行サイクル]　２
ただし，条件命令と組み合わせたときは
分岐するとき　　３
分岐しないとき　１

JMP　　JMP

[同時記述できる命令]

三項演算	二項演算	単項演算	並列ロード／ストア	レジスタ間転送	分　岐	条　件
×	×	×	×	×	×	○

[注　　意]　　ジャンプ命令をリピート対象命令，またはループの終端３命令以内に記述しないでください。

JREG　　JREG

分岐

JMP dp

命令名称	**レジスタ間接ジャンプ**
ニモニック	**JMP dp**

[記 述 例]　JMP DP0

[説　　明]　dpで指定したレジスタの値を分岐先とする命令（PC←dp）です。実行には３サイクルが必要です。

この命令を使った条件ジャンプ命令の記述方法については，前出のジャンプ命令を参照してください。このとき，命令実行に必要なサイクル数は，条件命令の条件判断が成立して分岐を行うときは３サイクル，条件が成立しないときは分岐を行わないので１サイクルとなります。ジャンプ命令とは異なり，レジスタ間接ジャンプは単独で使っても，条件命令と組み合わせて使っても，実行サイクル数は同じです。

[実行サイクル]　３

ただし，条件命令と組み合わせたときは

分岐するとき　３

分岐しないとき　１

[同時記述できる命令]

三項演算	二項演算	単項演算	並列ロード／ストア	レジスタ間転送	分　岐	条　件
×	×	×	×	×	×	○

[注　　意]　レジスタ間接ジャンプ命令をリピート対象命令，またはループの終端３命令以内に記述しないでください。

CALL　　CALL

分岐

CALL imm

命令名称	サブルーチン・コール
ニモニック	CALL imm（imm = 0-0xFFFF（0-65535））

［記 述 例］　CALL 0x1234

［説　　明］　μPD77016ファミリのインストラクション・メモリ空間のすべてのアドレスへサブルーチン・コールを行う命令です。実行には２サイクルが必要です。

スタック・ポインタをインクリメントして，スタックに直後の命令のアドレスを格納したあと，オペランドで指定したアドレスへ分岐します（SP←SP＋1, STACK←PC＋1, PC←PC＋imm）。

ジャンプ命令と同様に条件命令と組み合わせて，条件サブルーチン・コールを行うことができます。

IF（R0! = 0）　CALL LABEL

命令実行に必要なサイクル数は，条件命令の条件判断が成立して分岐を行うときは３サイクル，条件が成立しないときは分岐を行わないので１サイクルとなります。このサイクル数はレジスタ間接サブルーチン・コールの場合も同じです。

備考　サブルーチン・コール命令はジャンプ命令と同様に，相対サブルーチン・コールを行います。ユーザが相対サブルーチン・コールを意識する必要はありません（ジャンプ命令の**備考**を参照）。相対アドレス形式で記述したい場合は，以下のようにサブルーチン・コール命令のオペランドの先頭に＄を記述します。

CALL　＄＋２　…　２つ先のアドレスをサブルーチン・コール
CALL　＄－３　…　３つ手前のアドレスをサブルーチン・コール

［実行サイクル］　２

ただし，条件命令と組み合わせたときは
分岐するとき　　３
分岐しないとき　１

CALL　　CALL

[同時記述できる命令]

三項演算	二項演算	単項演算	並列ロード／ストア	レジスタ間転送	分　岐	条　件
×	×	×	×	×	×	○

[注　　意]　① サブルーチン・コール命令をリピート命令，またはループの終端３命令以内に記述しないでください。

② SPの値が0xF-0x1Fのとき，サブルーチン・コール命令を実行しないでください。実行した場合，スタック・アンダフローまたはスタック・オーバフローになります。このとき，サブルーチン・コールは正常に行われますが，リターン命令による戻りアドレスは不定となります。

③ CALL命令を条件命令と組み合わせた場合は，CALL命令の直後の命令に無条件リターン命令または無条件割り込みリターン命令を記述しないでください。

CREG　　CREG

分岐

CALL dp

命令名称	レジスタ間接サブルーチン・コール
ニモニック	CALL dp

［記 述 例］　CALL DP0

［説　　明］　サブルーチン・コール・アドレスをdpで指定すること，および実行に３サイクルを必要とすることを除いては，サブルーチン・コール命令と同じです（SP←SP＋１，STACK←PC＋１，PC←reg）。

［実行サイクル］　３

ただし，条件命令と組み合わせたときは

分岐するとき　　３

分岐しないとき　１

［同時記述できる命令］

三項演算	二項演算	単項演算	並列ロード／ストア	レジスタ間転送	分　岐	条　件
×	×	×	×	×	×	○

［注　　意］　① レジスタ間接サブルーチン・コール命令をリピート命令，またはループの終端３命令以内に記述しないでください。

② SPの値が0xF-0x1Fのとき，レジスタ間接サブルーチン・コール命令を実行しないでください。実行した場合，スタック・アンダフローまたはスタック・オーバフローになります。このとき，サブルーチン・コールは正常に行われますが，リターン命令による戻りアドレスは不定となります。

③ 条件命令と組み合わせた場合は，直後の命令に無条件リターン命令または無条件割り込みリターン命令を記述しないでください。

RET　　RET

分岐

RET

命令名称	**リターン**
ニモニック	**RET**

[記 述 例]　　RET

[説　　明]　　サブルーチンから復帰するための命令です。

スタック・ポインタの指しているスタックの値のアドレスに分岐したあと，スタック・ポインタをデクリメントします（PC←STACK, SP←SP-1）。実行には２サイクルが必要です。

ジャンプ命令，サブルーチン・コール命令と同様に条件命令と組み合わせて，条件リターン（IF（R0 = = 0）RETなど）を記述できます。このときの実行サイクル数もジャンプ命令と同じです。

[実行サイクル]　２

ただし，条件命令と組み合わせたときは

分岐するとき　　３

分岐しないとき　１

[同時記述できる命令]

三項演算	二項演算	単項演算	並列ロード／ストア	レジスタ間転送	分　岐	条　件
×	×	×	×	×	×	○

[注　　意]　① サブルーチン処理以外でリターン命令を実行しても，分岐，スタック・ポインタのデクリメントは行われます。

② SPの値が０または0x10-0x1Fのとき，リターン命令を実行しないでください。実行した場合，スタック・アンダフローまたはスタック・オーバフローになります。このとき分岐アドレスは不定となります。

③ RET命令を，STKまたはSPにロード／ストアするレジスタ間転送命令の直後に記述しないでください。

④ 条件命令と組み合わせたサブルーチン・コール命令またはレジスタ間接サブルーチン・コール命令の直後に記述しないでください。

RETI　　RETI

分岐

RETI

命令名称　　割り込みリターン
ニモニック　　RETI

[記 述 例]　　RETI

[説　　明]　　割り込み処理から復帰するための命令です。割り込み許可フラグの復帰をあわせて行う以外，リターン命令と同じです（PC←STACK，SP←SP-１，割り込み許可フラグの復帰）。

[実行サイクル]　２

ただし，条件命令と組み合わせたときは

分岐するとき　　３

分岐しないとき　１

[同時記述できる命令]

三項演算	二項演算	単項演算	並列ロード／ストア	レジスタ間転送	分　岐	条　件
×	×	×	×	×	×	○

[注　　意]　① 割り込みリターン命令をリピート対象命令またはループの終端３命令以内に記述しないでください。

② 割り込み処理以外で割り込みリターン命令を実行しても，分岐，スタック・ポインタのデクリメント，割り込み許可フラグの変更（SRのビット15-13,EIRの左シフト）は行います。

③ SPの値が０または0x10-0x1Fのとき，割り込みリターン命令を実行しないでください。実行した場合，スタック・アンダフローまたはスタック・オーバフローになります。このとき分岐アドレスは不定となります。

④ RETI命令を，STKまたはSPにロード／ストアするレジスタ間転送命令の直後に記述しないでください。

⑤ 条件命令と組み合わせたサブルーチン・コール命令またはレジスタ間接サブルーチン・コール命令の直後に記述しないでください。

⑥ ベクタ領域の各割り込み要因のエントリの先頭には記述しないでください。

３.８　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。

ハードウエア・ループ命令には，次の命令があります。

リピート（REP）

ループ（LOOP）

ループ・ポップ（LPOP）

REP　　REP

ハードウエア・ループ

REP count

命令名称	**リピート**
ニモニック	**REP count** **count = {1-0x7FFF, rl}**

[記 述 例]　REP 0x1234

[説　　明]　１命令を繰り返す命令です。

リピート対象となる直後の命令をcountで指定した回数繰り返します。リピート動作中，直後の命令のフェッチを繰り返します。

命令実行時の動作は以下のとおりです。

●開始

RC←count

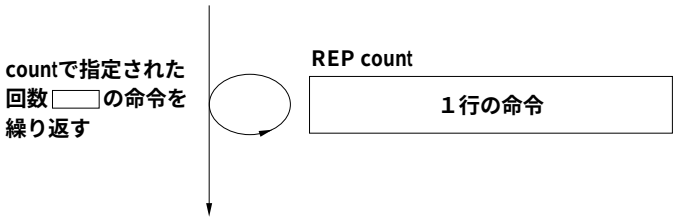
●リピート中

PC←PC

RC←RC－1

●終了

PC←PC＋1



リピート動作を終了して，リピート対象命令の直後の命令を実行する場合，オーバヘッドは発生しません。

[実行サイクル]　2

[同時記述できる命令]

三項演算	二項演算	単項演算	並列ロード／ストア	レジスタ間転送	分　岐	条　件
×	×	×	○	×	×	×

REP　　REP

［注　　意］　① 分岐命令（JMP, JREG, CALL, CREG, RET, RETI），リピート命令，ループ命令，ループ・ポップ命令，ストップ命令およびホールト命令を，リピート対象命令に指定しないでください。

★　② ループ命令を，リピート対象命令に記述しないでください。

③ リピート命令またはリピート対象命令をデコード中または実行中は，割り込みを受け付けません。

④ 繰り返し回数に０または0x8000以上を設定しないでください。

LOOP　　LOOP

ハードウエア・ループ

```
LOOP count ｛
        2-225行の命令
    ｝ ；
```

命令名称	ループ
ニモニック	LOOP count ｛ 2-225行の命令 ｝ ； count = ｛1-0x7FFF, rl｝
注意	“ ｛ ”“ ｝ ”はそのまま記述します。

［記 述 例］

```
LOOP 0x1234 ｛
    R1 = R0＋R4　R0 = ＊DP0＋＋　R4 = ＊DP4＋＋；
    ＊DP1＋＋ = R1
        ｝ ；
```

［説　　明］　複数の命令を繰り返す命令です。１つの命令を繰り返すときは，リピート命令を使ってください。

直後の命令（ループ開始アドレス）からループ・エンド疑似命令（ ｝ ）の直前の命令（ループ終端アドレス）までの命令をcountで指定した回数繰り返します。ループ開始アドレスとループ終端アドレスの差は，２≦ループ終端アドレス-ループ開始アドレス≦255の範囲で設定できます。命令実行時の動作は以下のとおりです。

●開始

LSP←LSP＋１
LSR1←LSA
LSR2←LEA
LSR3←LC
LSA←PC＋１
LEA←PC＋RLE（ループ終端相対アドレス）
LC←count

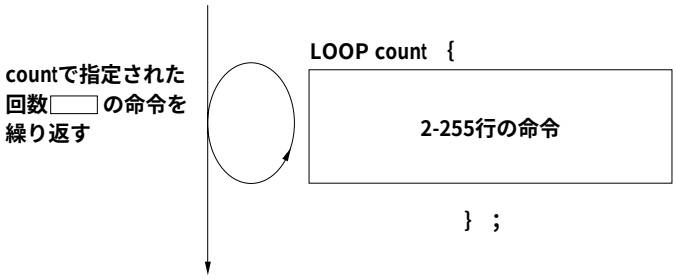
●ループ終端の命令フェッチ（ループ中）

LC←LC－１
PC←LSA

●ループ終端命令フェッチ（ループ終了）

PC←PC＋１
LC←LSR3
LEA←LSR2
LSA←LSR1
LSP←LSP－１

LOOP　　LOOP



　ループ終端アドレスからループ開始アドレスに分岐するとき，およびループ動作を終了してループ・エンド疑似命令の直後の命令を実行するとき，オーバヘッドは発生しません。

　ループ命令のデコード中および実行中，また，ループ終端命令をフェッチしている間は，割り込みを受け付けません。

[実行サイクル]　2

[同時記述できる命令]

三項演算	二項演算	単項演算	並列ロード／ストア	レジスタ間転送	分　岐	条　件
×	×	×	×	×	×	×

LOOP　　LOOP

［注　　意］　① 分岐命令（JMP, JREG, CALL, CREG, RET, RETI），リピート命令，ループ命令，ループ・ポップ命令，LCからの転送命令，ストップ命令およびホールト命令を，ループの終端３命令以内に記述しないでください。

ループ終端が，ほかのループ命令のループ終端と重複しないようにしてください。

禁止例１．

```
LOOP   0x1234 {
          :
          :
       [          ]
       [          ]      分岐命令
       [          ]
             } ;
```

禁止例２．

```
LOOP   0x1234 {
          :
          :
       LOOP  0x2345  {
          :
          :
                    } ;
             } ;
```

★　② LCからの転送命令をループの先頭に記述しないでください。

★　③ リピート命令を，ループの終端３命令以内に記述しないでください。

④ LSPが４-７の場合，LOOPを実行しないでください（ループ・スタック・アンダフローまたはループ・スタック・オーバフローが発生します）。

⑤ 繰り返し回数に０および0x8000以上を設定しないでください。

LPOP

LPOP

ハードウエア・ループ

LPOP

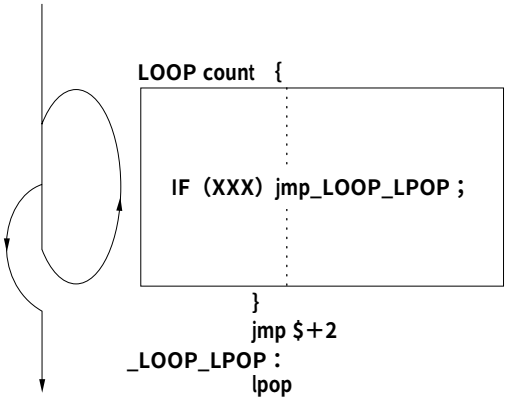
命令名称	ループ・ポップ
ニモニック	LPOP

[記 述 例]　LPOP

[説　　明]　現在のループの情報を破棄し，ループ・スタック・ポインタが指定しているループ・スタックの値をLC，LEAおよびLSAに転送します。ループ・スタック・ポインタの値をデクリメントします。

命令実行時の動作は以下のとおりです。

LC←LSR3
LEA←LSR2
LSA←LSR1
LSP←LSP－1



[実行サイクル]　1

[同時記述できる命令]

三項演算	二項演算	単項演算	並列ロード／ストア	レジスタ間転送	分　岐	条　件
×	×	×	×	×	×	×

LPOP　　LPOP

［注　　意］　① ループ・ポップ命令を，ループの終端３命令以内に記述しないでください。

② ループ・ポップ命令を，リピート対象命令に記述しないでください。

③ LSPが０および５-７の場合，LPOPを実行しないでください（ループ・スタック・アンダフローまたはループ・スタック・オーバフローが起こります）。

④ ループ・ポップ命令でループ命令を終えることはできません。この命令はループ・スタック・ポインタ（LSP）の値をデクリメントするだけです。

３.９　制御命令

プログラム制御を指定する命令です。

制御命令には，次の命令があります。

ノー・オペレーション（NOP）

ホールト（HALT）

ストップ（STOP）

フォーゲット・インタラプト（FINT）

条件（COND）

NOP　　NOP

制御

NOP

命令名称	ノー・オペレーション
ニモニック	NOP

[記 述 例]　NOP

[説　　明]　何も行いません。PCの値を更新し，１サイクル時間を消費するための命令です。

[実行サイクル]　１

[同時記述できる命令]

三項演算	二項演算	単項演算	並列ロード／ストア	レジスタ間転送	分　岐	条　件
×	×	×	×	×	×	×

HALT　　HALT

制御

HALT

命令名称	ホールト
ニモニック	HALT

[記 述 例]　HALT

[説　　明]　μPD77016ファミリの動作を停止する命令です。

備考１．ホールト・モード時，レジスタおよび内部メモリは，ホールト・モードに入る直前の状態を保持します（**μPD7701×ファミリ　ユーザーズ・マニュアル　アーキテクチャ編**または**μPD77111ファミリ　ユーザーズ・マニュアル　アーキテクチャ編**を参照してください）。

２．ホールト・モードの解除は，マスクされていない外部／内部割り込み，またはハードウエア・リセットで行います。

[実行サイクル]　１

[同時記述できる命令]

三項演算	二項演算	単項演算	並列ロード／ストア	レジスタ間転送	分　岐	条　件
×	×	×	×	×	×	×

[注　　意]　HALT命令を，リピート対象命令またはループ終端３命令以内に記述しないでください。

★ STOP　　STOP

制御

STOP

命令名称	ストップ
ニモニック	STOP

[記 述 例]　STOP

[説　　明]　μPD77016ファミリ（μPD77016を除く）のクロック回路およびPLLを停止する命令です。

備考１．ストップ・モード時，デバイスの端子は，ストップ・モードに入る直前の状態を保持します（**μPD7701×ファミリ　ユーザーズ・マニュアル　アーキテクチャ編**または**μPD77111ファミリ　ユーザーズ・マニュアル　アーキテクチャ編**を参照してください）。

２．ストップ・モードの解除は，ハードウエア・リセット，および$\overline{\text{WAKEUP}}$端子[注]で行います。

注　$\overline{\text{WAKEUP}}$端子によるストップ・モードの解除は，μPD77111ファミリのみ有効です。

[実行サイクル]　1

[同時記述できる命令]

三項演算	二項演算	単項演算	並列ロード／ストア	レジスタ間転送	分　岐	条　件
×	×	×	×	×	×	×

[注　　意]　① ストップ命令をリピート対象命令またはループ終端３命令以内に記述しないでください。

② ストップ・モードの解除は，ハードウエア・リセットで行います。このとき出力端子は初期化されますが，デバイスのPLLが安定するまでの間，出力端子の状態は不定となり，この間の状態を保証できません。

③ μPD77111ファミリで$\overline{\text{WAKEUP}}$端子によるストップ・モードの解除を行う場合には，STOP命令の直前にNOP命令を入れる必要があります。

例）nop；←必要
　　stop；$\overline{\text{WAKEUP}}$端子による解除を前提としたSTOP命令

FINT

FINT

制御

FINT

命令名称	フォーゲット・インタラプト
ニモニック	FINT

[記 述 例]　FINT

[説　　明]　既存の割り込み要求をすべて破棄する命令です。

起動する割り込み処理の順序に制限がある場合，特定の割り込み要因の発生にプログラムを同期させるために使用します。

[実行サイクル]　1

[同時記述できる命令]

三項演算	二項演算	単項演算	並列ロード／ストア	レジスタ間転送	分　岐	条　件
×	×	×	×	×	×	×

COND　　COND

制御

IF（ro cond）

命令名称	条件
ニモニック	IF（ro cond）

condに記述できる条件

EVER ：無条件（topの記述不要）
= = 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）

[記 述 例]　IF（R1＞= 0）　R0 = R0＋１

[説　明]　条件判定する命令です。

roで指定した汎用レジスタを40ビットの２の補数表現のデータとして評価します。条件が真のとき，同一ステートメントに記述した命令を実行します。偽のとき，ノー・オペレーションとします。

備考　条件判定が成立したときに実行する命令として三項演算，二項演算を記述することはできません。しかし，単項演算として代入（ro' = ro），累加算（ro'＋ = ro），累減算（ro'－ = ro）を記述することができます。したがって，これらの命令を使って加減算を行うことができます。

[実行サイクル]　同時記述した命令のサイクル数（条件が真）または１（条件が偽）

[同時記述できる命令]

三項演算	二項演算	単項演算	並列ロード／ストア	レジスタ間転送	分　岐	条　件
×	×	○	×	○	○	×

付録A　命令語のカテゴリ分類

命令語のフォーマット一覧を表A－1に示し，各カテゴリを説明します。

表A－1　命令語のフォーマット

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	op1			opcode				op2			op3			dx	dy	dpx		modix			rx (rxh)			dpy		modiy			ry (ryh)		
0	1	1	1	opcode				op1			op2			dx	dy	dpx		modix			rx (rxh)			dpy		modiy			ry (ryh)		
0	1	1	0	opcode				op1			op2			–	–	–	–	–	–	–	–	–	–	cond					top		
0	1	0	1	opcode				op1			op2			–	–	imm															
0	1	0	0	1	1	sufx			sufy			–	–	dx	dy	dpx		modix			regx			dpy		modiy			regy		
0	1	0	0	1	0	reg			suf			xy	–	–	d	direct															
0	1	0	0	0	1	reg			suf			dp			d	imm															
0	0	1	1	1	1	source1			0	dest1					0	–	–	–	–	–	–	–	–	cond					top		
0	0	1	1	1	1	source1			1	dest1					0	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
0	0	1	1	1	1	dest2			0	source2					1	–	–	–	–	–	–	–	–	cond					top		
0	0	1	1	1	1	dest2			1	source2					1	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
0	0	1	1	1	0	–	–	–	dest1						0	imm															
0	0	1	1	1	0	dest2			–	–	–	–	–	–	1	imm															
0	0	1	0	1	1	jc	–	–	**相対アドレス**															cond					top		
0	0	1	0	1	0	jc	–	–	–	–	–	dp			–	–	–	–	–	–	–	–	–	cond					top		
0	0	1	0	0	1	rt	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	cond					top		
0	0	0	1	1	1	–	–	–	–	–	–	–	–	–	–	0	**imm (回数)**														
0	0	0	1	1	0	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	rl		
0	0	0	1	0	1	–	–	**ループ終端相対アドレス**								0	**imm (回数)**														
0	0	0	1	0	0	–	–	**ループ終端相対アドレス**								–	–	–	–	–	–	–	–	–	–	–	–	–	rl		
0	0	0	0																												

備考　“ － ”は未使用のビットを示します。

A.1　３オペランド命令

３オペランド命令は，MACとALUの演算についてオペランドを３つ使用する命令です。このカテゴリには，三項演算命令と二項演算命令があり，それぞれ次のような演算形式となります。

●三項演算命令の形式

OP3 = OP3 opm op1 OP1 op2 OP2 ;

ここで，

OP1 ：第１オペランド（演算オペランド）

OP2 ：第２オペランド（演算オペランド）

OP3 ：第３オペランド（演算オペランドかつ結果オペランド）

opm ：MSFT演算指示で，OP3を局所的に修飾します。“ 指示なし ”，“ ＞＞１ ”（１ビット算術右シフト），“ ＞＞16 ”（16ビット算術右シフト）のいずれかです。

op1 ：ALU演算指示で，“ ＋ ”または“ － ”があります。

op2 ：MAC演算指示で，常に“ ＊ ”です。

●二項演算命令の形式

（１）OP3 = OP1 op OP2 ;

（２）OP3 = oplt （OP1, OP2） ;

ここで，

OP1 ：第１オペランド（演算オペランド）

OP2 ：第２オペランド（演算オペランド）

OP3 ：第３オペランド（結果オペランド）

op　：ALU，MAC演算指示。

oplt ：ALU演算指示で，常に“ LT ”です。

“ LT（Less Than） ”命令を除く通常の二項演算はすべて（１）の形式で記述します。“ LT ”命令のみ（２）の形式で記述します。

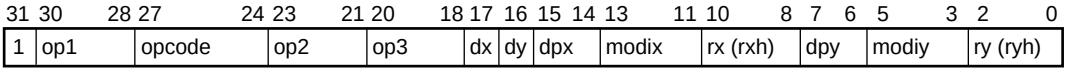
また３オペランド命令のカテゴリでは，並列ロード／ストア命令を同時記述可能ですから，ALU，MACの演算を実行しながら同時にX/Yメモリと汎用レジスタとの間でデータ交換が可能です。

３オペランド命令の命令語フォーマットを図Ａ－１に示します。

図A－1　3オペランド命令フォーマット

XY並列ロード／ストア（演算レジスタ⟷メモリ）同時記述可

op [[rx = *dp (*dp++, *dp– –, *dp##, *dp%%, *!dp##)]
[*dp (*dp++, *dp– –, *dp##, *dp%%, *!dp##)] = rxh]] [[ry = *dp (*dp++, *dp– –, *dp##, *dp%%, *!dp##)]
[*dp (*dp++, *dp– –, *dp##, *dp%%, *!dp##)] = ryh]]



opcode

op3 = lt (op1,op2)
op3 = op1h*op2h
op3 = op3 + op1h*op2h
op3 = op3 – op1h*op2h
op3 = op3 + op1h*op2l
op3 = op3 + op1l*op2l
op3 = op3 >> 1 + op1h*op2h
op3 = op3 >> 16 + op1h*op2h
op3 = op1 + op2
op3 = op1 – op2
op3 = op1 & op2
op3 = op1 | op2
op3 = op1 ^ op2
op3 = op1 sra op2l
op3 = op1 srl op2l
op3 = op1 sll op2l

op1, op2, op3

r0 (r0h, r0l)
r1 (r1h, r1l)
r2 (r2h, r2l)
r3 (r3h, r3l)
r4 (r4h, r4l)
r5 (r5h, r5l)
r6 (r6h, r6l)
r7 (r7h, r7l)

A.2　２オペランド命令

２オペランド命令には，ALUの単項演算のみが分類されます（NOPを含む）。

●単項演算命令の形式

（１）opn

（２）OP2 = OP1 op 1 ;

（３）OP2 opu = OP1 ;

（４）OP2 = opl OP1 ;

（５）OP2 = opf （OP1） ;

ここで，

OP1 ：第１オペランド（演算オペランド）

OP2 ：第２オペランド（結果オペランド）

opn ：“ NOP ”のみがあります。

注意　ここでいう“ NOP ”はALUに対する指示のことで，命令語全体が“ NOP ”という意味ではありません。

op　：ALUの演算指示で“ ＋ ”または“ － ”。

opu ：ALU演算指示です。次の図を参照してください。

opl　：ALU演算指示で“ － ”または“ ～ ”です。

opf　：ALU演算指示で“ CLIP ”，“ ROUND ”，“ EXP ”があります。

また２オペランド命令のカテゴリでは，並列ロード／ストア命令または条件命令のどちらかを同時記述可能ですから，ALU，MACの演算を実行しながら同時にX/Yメモリと汎用レジスタとの間でデータ交換を可能とするか，または，条件付きでALU，MAC演算を可能とします。

２オペランド命令の命令語フォーマットを図Ａ－２に示します。

図A－2　2オペランド命令フォーマット

XY並列ロード／ストア（演算レジスタ←→メモリ）同時記述可

op [[rx = *dp (*dp++, *dp– –, *dp##, *dp%%, *!dp##)] [*dp (*dp++, *dp– –, *dp##, *dp%%, *!dp##)] = rxh] [[ry = *dp (*dp++, *dp– –, *dp##, *dp%%, *!dp##)] [*dp (*dp++, *dp– –, *dp##, *dp%%, *!dp##)] = ryh]

31 28	27 24	23 21	20 18	17	16	15 14	13 11	10 8	7 6	5 3	2 0
0 1 1 1	opcode	op1	op2	dx	dy	dpx	modix	rx (rxh)	dpy	modiy	ry (ryh)

条件同時記述可

[if (top cond)] op

31 28	27 24	23 21	20 18	17 7	6 3	2 0
0 1 1 0	opcode	op1	op2	– – – – – – – – – –	cond	top

opcode

nop
clr (op2)
op2 = op1 + 1
op2 = op1 – 1
op2 = abs (op1)
op2 = ~op1
op2 = –op1
op2 = clip (op1)
op2 = round (op1)
op2 = exp (op1)
op2 = op1
op2 / = op1
op2 + = op1
op2 – = op1

op1, op2

r0
r1
r2
r3
r4
r5
r6
r7

A.3　即値演算命令

即値演算命令には，ALUのイミーディエト二項演算のみがあります。

●二項演算命令（イミーディエト演算）

OP2 = OP1 op imm ;

ここで，

OP1 ：第１オペランド（演算オペランド）

OP2 ：第２オペランド（結果オペランド）

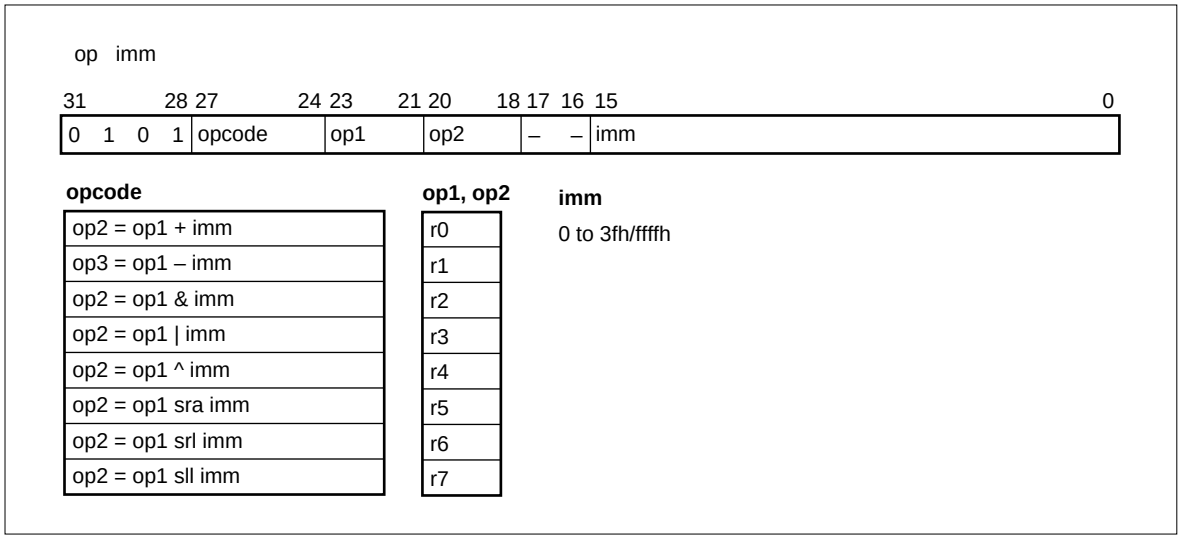
op　：ALU演算指示です。次の図を参照してください。

imm ：16ビットのイミーディエト・データです。

このカテゴリの命令には，同時記述できるほかの機能命令はありません。

即値演算命令の命令語フォーマットを図Ａ－３に示します。

図Ａ－３　即値演算命令フォーマット



A.4　ロード／ストア命令

ロード／ストア命令のカテゴリには，次の４つの命令があります。

●並列ロード／ストア命令

Xメモリ，Yメモリと汎用レジスタとの間で，16ビット・データをロード／ストアします。メモリ・アクセスにあたっては，DPn（n = 0-7）による間接アドレシングのみが有効です。この命令の重要なポイントは，次の演算命令のいずれかと同時記述できるところです。

・三項演算命令
・二項演算命令（ただし，イミーディエト二項演算は除く）
・単項演算

したがって，演算を実行しながら次の演算のために次のデータを汎用レジスタに準備するといった処理を１命令で実行できます。

●部分ロード／ストア命令

Xメモリ，Yメモリと汎用レジスタとの間で，16ビット・データをロード／ストアします。メモリ・アクセスにあたっては，DPn（n = 0-7）による間接アドレシングのみが有効です。この命令の重要なポイントを次に示します。

・ロード／ストアの対象として，汎用レジスタの分割フォーマットにしたがった任意の部分を指定できます。
・１つの汎用レジスタの２つの部分に，X, Yメモリから同時にそれぞれの16ビット・データをロードすることが可能です（32ビット・データのロード）。
・同時記述できるほかの機能命令はありません。

●ダイレクト・アドレシング・ロード／ストア命令

命令語のなかで，直接データ・メモリ区分とアドレスを指定するロード／ストア形式です。この命令の重要なポイントを次に示します。

・命令中に，直接メモリ区分とメモリ・アドレスを記述します。
・ロード（メモリ→汎用レジスタ）では，汎用レジスタの分割フォーマットによる任意の部分と，２種類の組み合わせ（Rn, RnEH：n = 0-7）を対象にできます。
・ストア（汎用レジスタ→メモリ）では，汎用レジスタの分割フォーマットによる任意の部分を対象にできます。
・X，Yメモリを同時にアクセスすることはできません。
・同時記述できるほかの機能命令はありません。

●即値モディファイ・ロード／ストア命令

DPnによる間接アドレシングですが，アクセス後のDPn修飾をDNnではなく，イミーディエト・データで行います。この命令の重要なポイントを次に示します。

- 命令中に，DPn修飾データを直接記述します。
- ロード（メモリ→汎用レジスタ）では，汎用レジスタの分割フォーマットによる任意の部分と，2種類の組み合わせ（Rn, RnEH：n = 0-7）を対象にできます。
- ストア（汎用レジスタ→メモリ）では，汎用レジスタの分割フォーマットによる任意の部分を対象にできます。
- X，Yメモリを同時にアクセスすることはできません。
- 同時記述できるほかの機能命令はありません。

ロード／ストア命令の命令語フォーマットを図A－4に示します。

図A－4 ロード／ストア命令フォーマット（1/2）

XY並列ロード／ストア（演算レジスタ←→メモリ）

[op] [x load/store nop
rx = *dp (*dp++, *dp– –, *dp##, *dp%%, *!dp##)
*dp (*dp++, *dp– –, *dp##, *dp%%, *!dp##) = rxh] [y load/store nop
ry = *dp (*dp++, *dp – –, *dp##, *dp%%, *!dp##)
*dp (*dp++, *dp– –, *dp##, *dp%%, *!dp##) = ryh]

31 30 28	27 24	23 21	20 18	17	16	15 14	13 11	10 8	7 6	5 3	2 0
1 op1	opcode	op2	op3	dx	dy	dpx	modix	rx (rxh)	dpy	modiy	ry (ryh)
0 1 1 1	opcode	op1	op2	dx	dy	dpx	modix	rx (rxh)	dpy	modiy	ry (ryh)

dx, dy

*dp = rxh (ryh)
rx (ry) =*dp

rx (rxh), ry (ryh)

r0 (r0h)
r1 (r1h)
r2 (r2h)
r3 (r3h)
r4 (r4h)
r5 (r5h)
r6 (r6h)
r7 (r7h)

dpx, dpy

dp0, dp4
dp1, dp5
dp2, dp6
dp3, dp7

modix, modiy

nop
*dpx (y)
*dpx (y)++
*dpx (y)– –
*dpx (y)##
*dpx (y)%%
*idpx (y)##

部分ロード／ストア

[x load/store nop
regxs = *dp (*dp++, *dp– –, *dp##, *dp%%, *!dp##)
*dp (*dp++, *dp– –, *dp##, *dp%%, *!dp##) = regxs] [y load/store nop
regys = *dp (*dp++, *dp– –, *dp##, *dp%%, *!dp##)
*dp (*dp++, *dp– –, *dp##, *dp%%, *!dp##) = regys]

31 28	27 26	25 23	22 20	19 18	17	16	15 14	13 11	10 8	7 6	5 3	2 0
0 1 0 0	1 1	sufx	sufy	– –	dx	dy	dpx	modix	regx	dpy	modiy	regy

sufx + dx, sufy + dy

*dp = rx (y) l
*dp = rx (y) h
*dp = rx (y) e
rx (y) l = *dp
rx (y) h = *dp
rx (y) e = *dp
rx (y) eh = *dp
rx (y) = *dp

regx, regy

r0suf
r1suf
r2suf
r3suf
r4suf
r5suf
r6suf
r7suf

dpx, dpy

dp0, dp4
dp1, dp5
dp2, dp6
dp3, dp7

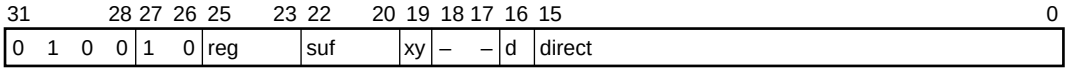
modix, modiy

nop
*dpx (y)
*dpx (y)++
*dpx (y)– –
*dpx (y)##
*dpx (y)%%
*idpx (y)##

図A－4 ロード／ストア命令フォーマット（2/2）

ダイレクト・アドレシング・ロード／ストア

[regs = *direct: x (y)
*direct: x (y) = regs]



suf + d
*dp = regl
*dp = regh
*dp = rege
regl = *dp
regh = *dp
rege = *dp
regeh = *dp
reg = *dp

reg
r0suf
r1suf
r2suf
r3suf
r4suf
r5suf
r6suf
r7suf

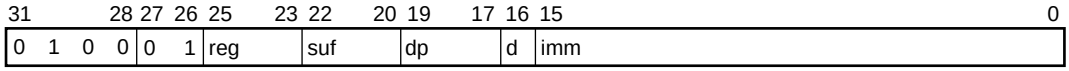
xy
x
y

direct

0 to ffffh

即値モディファイ・ロード／ストア

[rs = *dp##imm
*dp##imm = rs]



suf + d
*dp##imm = regl
*dp##imm = regh
*dp##imm = rege
regl = *dp##imm
regh = *dp##imm
rege = *dp##imm
regeh = *dp##imm
reg = *dp##imm

reg
r0suf
r1suf
r2suf
r3suf
r4suf
r5suf
r6suf
r7suf

dp
dp0
dp1
dp2
dp3
dp4
dp5
dp6
dp7

imm

0 to ffffh

A.5　レジスタ間転送命令

汎用レジスタと，メイン・バスに接続されたその他のレジスタとの転送を実行します。この命令の重要なポイントを次に示します。

- 転送の片側（デスティネーションまたはソース）には，常に汎用レジスタが記述されます。
- 転送対象には，メイン・バスに接続された任意のレジスタを指定できます。
- 条件命令を同時に記述できるので，条件付き転送が可能です。

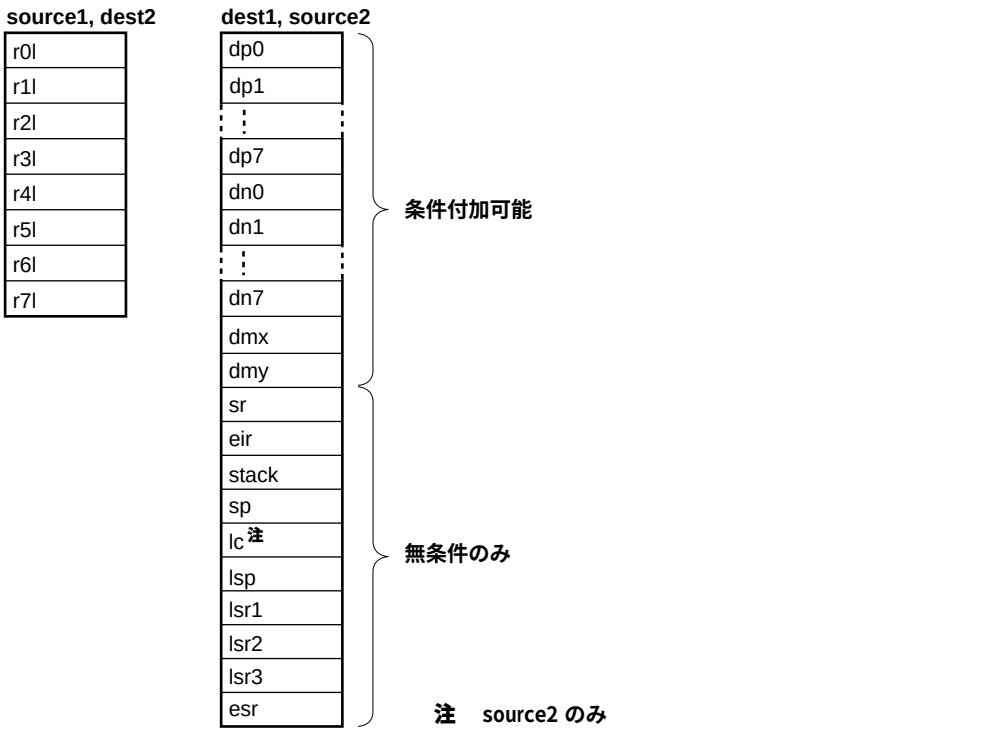
備考　この命令は汎用レジスタ間の転送には適用できません。汎用レジスタ間でデータを転送する場合は，単項演算の“ PUT ”命令を使用します。

レジスタ間転送命令の命令語フォーマットを図A－5に示します。

図A－5　レジスタ間転送命令フォーマット

[if cond] dest = source

31 28	27	26	25 23	22	21 17	16	15 7	6 3	2 0
0 0 1 1	1	1	source1	0	dest1	0	– – – – – – – – –	cond	top
0 0 1 1	1	1	source1	1	dest1	0	– – – – – – – – – – – – – – – –		
0 0 1 1	1	1	dest2	0	source2	1	– – – – – – – – –	cond	top
0 0 1 1	1	1	dest2	1	source2	1	– – – – – – – – – – – – – – – –		



A.6　即値設定命令

汎用レジスタ，またはアドレシング用レジスタにイミーディエト・データを設定する命令です。この命令の要点を次に示します。

- 設定対象となるのは，汎用レジスタのＬパート（R0L-R7L），またはDPn，DNn，DMX/Yです。
- 同時記述できるほかの機能命令はありません。

即値設定命令の命令語フォーマットを図Ａ－６に示します。

図Ａ－６ 即値設定命令フォーマット

reg = imm

31 28	27	26	25 23	22 17	16	15 0
0 0 1 1	1	0	– – –	dest1	0	imm
0 0 1 1	1	0	dest2	– – – – – –	1	imm

dest2
r0l
r1l
r2l
r3l
r4l
r5l
r6l
r7l

dest1
dp0
dp1
⋮
dp7
dn0
dn1
⋮
dn7
dmx
dmy

imm

0 to ffffh

A.7　分岐命令

分岐命令のカテゴリには，次の３種類があります。

●相対アドレス・ジャンプ／相対アドレス・コール

これらの分岐は，現在のPC値に目的アドレスまでの差分値を加算（減算）して分岐を実現するものです。差分値は16ビット２の補数形式で与えられますから，結局64 Kワード命令メモリ空間全域に分岐できます。次にこの命令の重要なポイントを示します。

- オペランドは16ビット２の補数形式で，分岐先アドレスまでの差分を表します（ただし，アセンブリ記述では目的アドレスの数値を直接に，またはラベルで記述します）。
- 条件命令を同時に記述できます。

●間接ジャンプ／間接コール

分岐先アドレスを，レジスタ（DPn）で与えます。このとき，PCには指定したDPnの値が直接設定されます。次にこの命令の重要なポイントを示します。

- オペランドはDPn（n = 0-7）です。
- 条件命令を同時に記述できます。

●リターン／割り込みリターン

分岐先アドレスは，STK（スタック）に退避されていたデータを分岐先までの差分値として現在のPCに加算（減算）します。

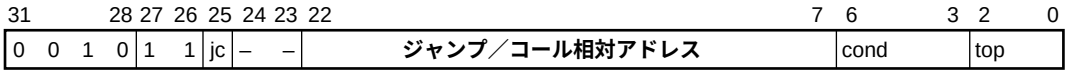
- 明示的なオペランドはなく，暗黙のオペランドとしてSTKが指定されています。STKの値は２の補数として現在のPC値に加算（減算）されます。
- 条件命令を同時に記述できます。

分岐命令の命令語フォーマットを図A－7に示します。

図A－7　分岐命令フォーマット

相対アドレス・ジャンプ／相対アドレス・コール

[if cond] jmp/call imm

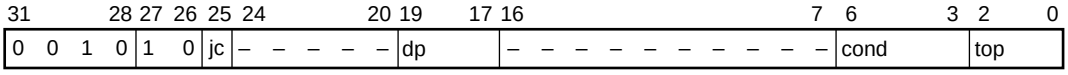


jc

br
call

間接ジャンプ／間接コール

[if cond] jmp/call dp



jc

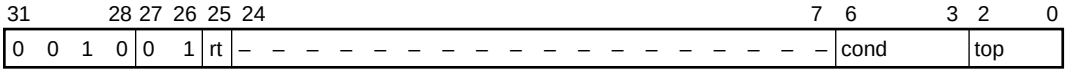
jp
callr

dp

dp0
dp1
dp2
dp3
dp4
dp5
dp6
dp7

リターン／割り込みリターン

[if cond] return/returni



rt

ret
reti

A.8　ハードウエア・ループ

ハードウエア・ループのカテゴリには次の２つの命令があります。

●リピート命令

１命令[注]を繰り返す命令です。次に要点を示します。

・オペランドとして，命令中に直接繰り返し回数を記述する形式と，汎用レジスタのＬパート（RnL：n = 0-7）で与える形式があります。
・同時記述できるほかの機能命令はありません。

●ループ命令

２命令[注]以上255命令[注]までを，グルーピングして繰り返す命令です。
次に要点を示します。

・オペランドとして，命令中に直接繰り返し回数を記述する形式と，汎用レジスタのＬパート（RnL：n = 0-7）で与える形式があります。
・同時記述できるほかの機能命令はありません。

注　ここでいう“命令”とは，１命令語を単位として表したもので，命令語の中のフィールド要素としての機能命令のことではありません。

ハードウエア・ループ命令の命令語フォーマットを図A－8に示します。

図A－8　ハードウエア・ループ命令のフォーマット

リピート命令

repeat count

31	28	27	26	25	16	15	14	0
0 0 0 1		1	1	– – – – – – – – – –		0	imm	

imm

1to7fffh

repeat rl

31	28	27	26	25	3	2	0
0 0 0 1		1	0	– –		rl	

rl

r0l
r1l
r2l
r3l
r4l
r5l
r6l
r7l

ループ命令

loop imm { }

31	28	27	26	25	24	23	16	15	14	0
0 0 0 1		0	1	–	–	**ループ終端相対アドレス**		0	imm	

imm

1to7fffh

loop rl { }

31	28	27	26	25	24	23	16	15	3	2	0
0 0 0 1		0	0	–	–	**ループ終端相対アドレス**		– – – – – – – – – – – – –		rl	

rl

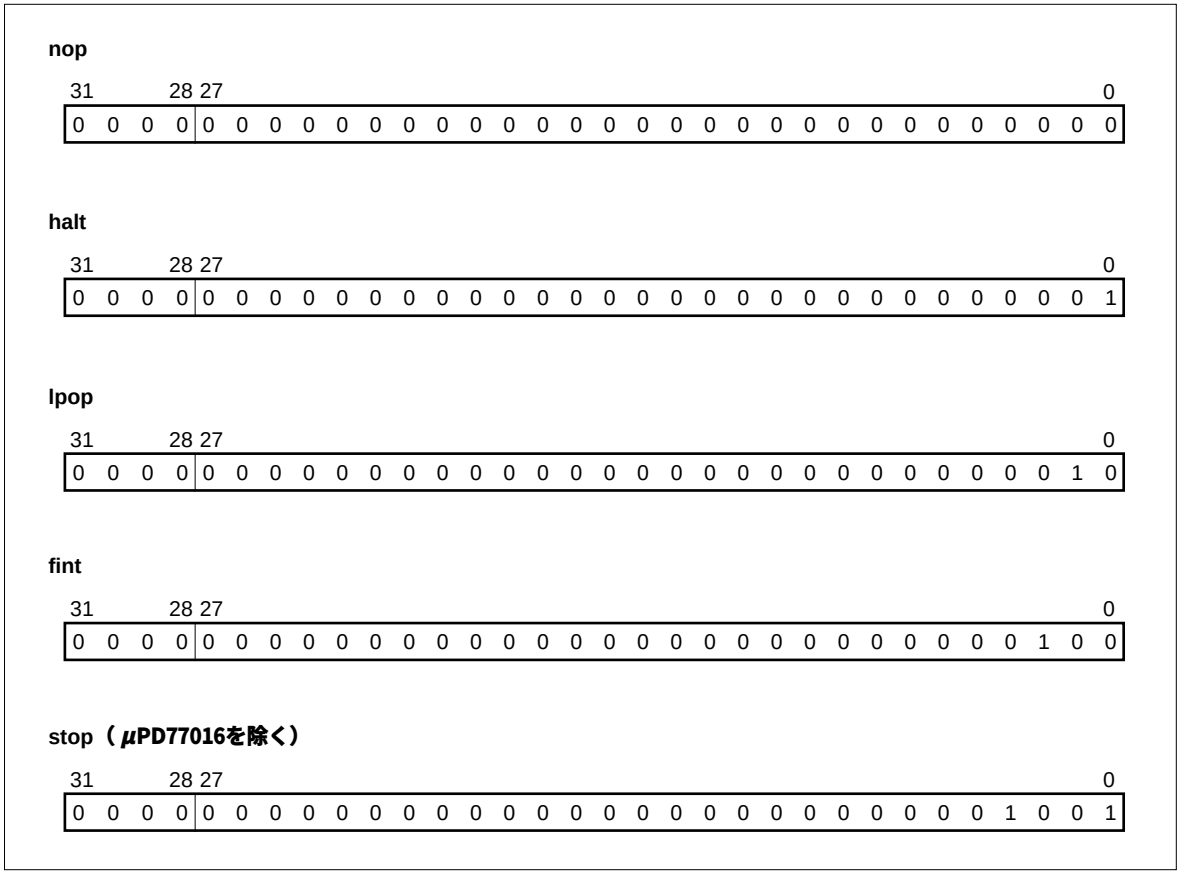
r0l
r1l
r2l
r3l
r4l
r5l
r6l
r7l

A.9　CPU制御命令

このカテゴリの命令はオペランドを使用しません。また，同時記述できる他の機能命令もありません。

CPU制御命令の命令語フォーマットを図A－9に示します。

図A－9　CPU制御命令フォーマット



A.10　条件命令

条件命令は，ほかの機能命令と組み合わせて意味を持ちます。

●記述形式

IF　（ro　cond）　ほかの命令　；

ここで，

ro　：任意の汎用レジスタ（R0-R7）

cond：条件記述で，次のものがあります。

= = 0　：= 0を判定

! = 0　：≠0を判定

＞0　：＞0を判定

＜0　：＜0を判定

＞ = 0：≧0を判定

＜ = 0：≦0を判定

= = ex：拡張あり（ビット31-ビット39に０と１が混在）

! = ex ：拡張なし（ビット31-ビット39がすべて０かすべて１）

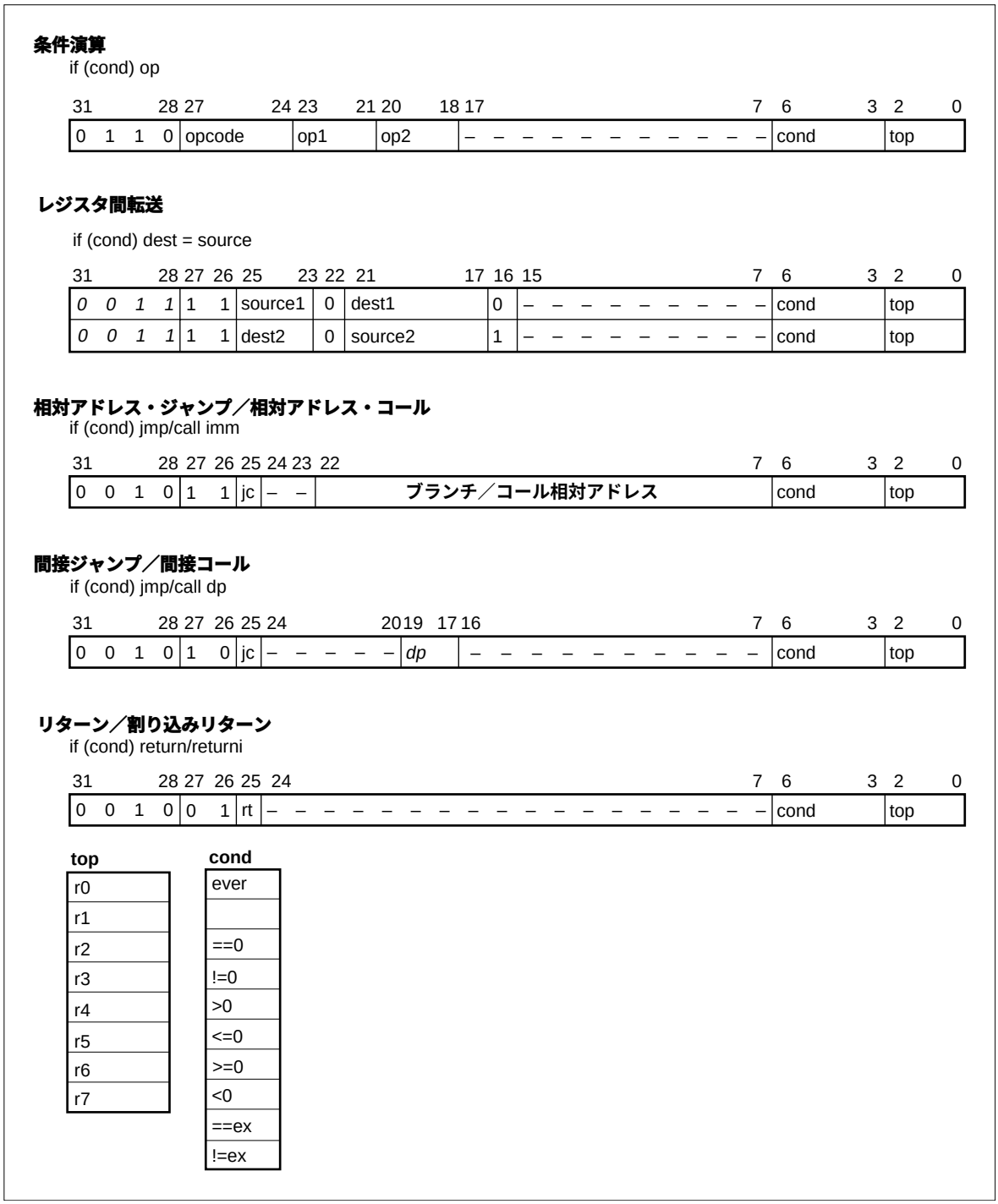
●組み合わせ可能なほかの機能命令

次のいずれかの機能命令と同時記述が可能です。

- 単項演算命令
- レジスタ間転送命令
- 分岐命令

条件命令の命令語フォーマットを図A－10に示します。

図A－10 条件命令フォーマット



〔メ　モ〕

付録B　命令セット一覧

命令群	命令名称	ニモニック	オペレーション	同時記述できる命令									フラグ	ページ
				三項	二項	単項	ロード／ストア	転送	即値	分岐	ループ	条件	OV	
三項演算	マルチプライ・アド	ro = ro＋rh＊rh'	ro←ro＋rh＊rh'				○						↕	24
	マルチプライ・サブ	ro = ro－rh＊rh'	ro←ro－rh＊rh'				○						↕	26
	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）	ro←ro＋rh＊rl				○						↕	28
	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）	ro←ro＋rl＊rl'				○						↕	30
	1ビット・シフト・マルチプライ・アド	ro =（ro＞＞1）＋rh＊rh'	ro←$\frac{ro}{2}$＋rh＊rh'				○						↕	32
	16ビット・シフト・マルチプライ・アド	ro =（ro＞＞16）＋rh＊rh'	ro←$\frac{ro}{2^{16}}$＋rh＊rh'				○						●	34
二項演算	マルチプライ	ro = rh＊rh'	ro←rh＊rh'				○						●	37
	アド	ro" = ro＋ro'	ro"←ro＋ro'				○						↕	38
	イミーディエト・アド	ro' = ro＋imm	ro'←ro＋imm （ただしimm ≠ 1）										↕	39
	サブ	ro" = ro－ro'	ro"←ro－ro'				○						↕	40
	イミーディエト・サブ	ro' = ro－imm	ro'←ro－imm （ただしimm ≠ 1）										↕	41
	算術右シフト	ro' = ro SRA rl	ro'←ro＞＞rl				○						●	42
	イミーディエト算術右シフト	ro' = ro SRA imm	ro'←ro＞＞imm										●	43
	論理右シフト	ro' = ro SRL rl	ro'←ro＞＞rl				○						●	44
	イミーディエト論理右シフト	ro' =ro SRL imm	ro'←ro＞＞imm										●	45
	論理左シフト	ro' = ro SLL rl	ro'←ro＜＜rl				○						●	46
	イミーディエト・論理左シフト	ro' = ro SLL imm	ro'←ro＜＜imm										●	47
	アンド	ro" = ro & ro'	ro"←ro & ro'				○						●	48
	イミーディエト・アンド	ro' = ro & imm	ro'←ro & imm										●	49

オーバフロー・フラグ（OV）の状態

●：変化なし

↕：オーバフローが起きたとき，1にセットされます。

命令群	命令名称	ニモニック	オペレーション	同時記述できる命令									フラグ	ページ
				三項	二項	単項	ロード／ストア	転送	即値	分岐	ループ	条件	OV	
二項演算	オア	ro" = ro ｜ ro'	ro"←ro ｜ ro'				○						●	50
	イミーディエト・オア	ro' = ro ｜ imm	ro'←ro ｜ imm										●	51
	イクスクルーシブ・オア	ro" = ro^ro'	ro"←ro^ro'				○						●	52
	イミーディエト・イクスクルーシブ・オア	ro' = ro^imm	ro'←ro^imm										●	53
	レスザン	ro" = LT（ro, ro ′ ）	if（ro＜ro'） {ro"←0x0000000001} else {ro"←0x0000000000}				○						●	54
単項演算	クリア	CLR（ro）	ro←0x0				○					○	●	56
	インクリメント	ro' = ro＋1	ro'←ro＋1				○					○	↕	57
	デクリメント	ro' = ro－1	ro'←ro－1				○					○	↕	58
	絶対値	ro' = ABS（ro）	if（ro＜0） {ro'←－ro} else {ro'←ro}				○					○	↕	59
	１の補数	ro' = ~ro	ro'←~ro				○					○	●	60
	２の補数	ro' = －ro	ro'←－ro				○					○	↕	61
	クリップ	ro' = CLIP（ro）	if（ro＞0x007FFFFFFF） {ro'←0x007FFFFFFF} else if（ro＜0xFF80000000） {ro'←0xFF80000000} else {ro'←ro}				○					○	↕	62
	丸め	ro' = ROUND（ro）	if（ro＞0x007FFF0000） {ro'←0x007FFF0000} else if（ro＞0xFF80000000） {ro'←0xFF80000000} else {ro'←（ro＋0x8000） & 0xFFFFFF0000}				○					○	↕	63
	指数	ro' = EXP（ro）	ro'←$\log_2\left(\frac{1}{ro}\right)$				○					○	●	64
	代入	ro' = ro	ro'←ro				○					○	●	66
	累加算	ro'＋ = ro	ro'←ro'＋ro				○					○	↕	67
	累減算	ro'－ = ro	ro'←ro'－ro				○					○	↕	68

オーバフロー・フラグ（OV）の状態

●：変化なし

↕：オーバフローが起きたとき，１にセットされます。

命令群	命令名称	ニモニック	オペレーション	同時記述できる命令									フラグ	ページ
				三項	二項	単項	ロード／ストア	転送	即値	分岐	ループ	条件	OV	
単項演算	除算	ro'/ = ro	if（sign（ro'）= = sign（ro）） {ro'←（ro'−ro）＜＜1} else {ro'←（ro'＋ro）＜＜1} if（sign（ro'）= = 0） {ro'←ro'＋1}				○					○	↕	69
ロード／ストア	並列ロード／ストア	ro = ＊dpx_mod ro' = ＊dpy_mod	ro←＊dpx, ro'←＊dpy	○	○	○							●	72
		ro = ＊dpx_mod ＊dpy_mod = rh	ro←＊dpx, ＊dpy←rh											
		＊dpx_mod = rh ro = ＊dpy_mod	＊dpx←rh, ro←＊dpy											
		＊dpx_mod = rh ＊dpy_mod = rh'	＊dpx←rh, ＊dpy←rh'											
	部分ロード／ストア	dest = ＊dpx_mod dest' = ＊dpy_mod	dest←＊dpx, dest'←＊dpy										●	76
		dest = ＊dpx_mod ＊dpy_mod = source	dest←＊dpx, ＊dpy←source											
		＊dpx_mod = source dest = ＊dpy_mod	＊dpx←source, dest←＊dpy											
		＊dpx_mod = source ＊dpy_mod = source'	＊dpx←source, ＊dpy←source'											
	ダイレクト・アドレシング・ロード／ストア	dest = ＊addr	dest←＊addr										●	82
		＊addr = source	＊addr←source											
	即値インデクス・ロード／ストア	dest = ＊dp_imm	dest←＊dp										●	86
		＊dp_imm = source	＊dp←source											
レジスタ間転送	レジスタ間転送	dest = rl	dest←rl									○	●	91
		rl = source	rl←source											

オーバフロー・フラグ（OV）の状態

●：変化なし

↕：オーバフローが起きたとき，1にセットされます。

命令群	命令名称	ニモニック	オペレーション	同時記述できる命令									フラグ	ページ
				三項	二項	単項	ロードストア／	転送	即値	分岐	ループ	条件	OV	
即値設定	即値設定	rl = imm （ただし，imm = 0-0xFFFF）	rl←imm										●	95
		dp = imm （ただし，imm = 0-0xFFFF）	dp←imm											
		dn = imm （ただし，imm = 0-0xFFFF）	dn←imm											
		dm = imm （ただし，imm = 1-0xFFFF）	dm←imm											
分岐	ジャンプ	JMP imm	PC←imm									○	●	97
	レジスタ間接ジャンプ	JMP dp	PC←dp									○	●	99
	サブルーチン・コール	CALL imm	SP←SP＋1 STK←PC＋1 PC←imm									○	●	100
	レジスタ間接サブルーチン・コール	CALL dp	SP←SP＋1 STK←PC＋1 PC←dp									○	●	102
	リターン	RET	PC←STK SP←SP－1									○	●	103
	割り込みリターン	RETI	PC←STK STK←SP－1 割り込み許可フラグの復帰									○	●	104
ハードウエア・ループ	リピート	REP count	開始　RC←count RF←0 リピート中　PC←PC RC←RC－1 終了　PC←PC＋1 RF←1										●	106

オーバフロー・フラグ（OV）の状態

●：変化なし

↕：オーバフローが起きたとき，1にセットされます。

命令群	命令名称	ニモニック	オペレーション	同時記述できる命令									フラグ	ページ
				三項	二項	単項	ロードストア	転送	即値	分岐	ループ	条件	OV	
ハードウエア・ループ	ループ	LOOP count （２行以上の命令）	開始　RC←count RF←０ リピート中　PC←PC RC←RC－１ 終了　PC←PC＋１ RF←１										●	108
	ループ・ポップ	LPOP	LC←LSR3 LEA←LSR2 LSA←LSR1 LSP←LSP－１										●	111
制御	ノー・オペレーション	NOP	PC←PC＋１										●	114
	ホールト	HALT	CPU停止										●	115
	ストップ	STOP	CPU, PLL, OSC停止										●	116
	フォーゲット・インタラプト	FINT	割り込み要求を破棄										●	117
条件	条件	IF（ro cond）	条件判定			○		○		○			●	118

オーバフロー・フラグ（OV）の状態

●：変化なし

↕：オーバフローが起きたとき，１にセットされます。

〔メ　モ〕

付録C　命令索引

C.1　五十音順

C.2　アルファベット順

〔メ　モ〕

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/

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］μPD77016ファミリ　ユーザーズ・マニュアル　命令編
（U13116JJ2V0UMJ1（第２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