User's Manual

RL78 ファミリ

ユーザーズマニュアル　ソフトウェア編

シングルチップ・マイクロコントローラ

ルネサス エレクトロニクス
www.renesas.com

Rev.2.30　2022.04

ご注意書き

1. 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。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、お客様の責任において、お客様の機器・システムを設計ください。これらの使用に起因して生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
2. 当社製品または本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 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。
4. 当社製品を組み込んだ製品の輸出入、製造、販売、利用、配布その他の行為を行うにあたり、第三者保有の技術の利用に関するライセンスが必要となる場合、当該ライセンス取得の判断および取得はお客様の責任において行ってください。
5. 当社製品を、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング等により生じた損害に関し、当社は、一切その責任を負いません。
6. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。

   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、産業用ロボット等

   高品質水準：輸送機器（自動車、電車、船舶等）、交通制御（信号）、大規模通信機器、金融端末基幹システム、各種安全制御装置等

   当社製品は、データシート等により高信頼性、Harsh environment 向け製品と定義しているものを除き、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
7. あらゆる半導体製品は、外部攻撃からの安全性を 100%保証されているわけではありません。当社ハードウェア／ソフトウェア製品にはセキュリティ対策が組み込まれているものもありますが、これによって、当社は、セキュリティ脆弱性または侵害（当社製品または当社製品が使用されているシステムに対する不正アクセス・不正使用を含みますが、これに限りません。）から生じる責任を負うものではありません。当社は、当社製品または当社製品が使用されたあらゆるシステムが、不正な改変、攻撃、ウイルス、干渉、ハッキング、データの破壊または窃盗その他の不正な侵入行為（「脆弱性問題」といいます。）によって影響を受けないことを保証しません。当社は、脆弱性問題に起因しまたはこれに関連して生じた損害について、一切責任を負いません。また、法令において認められる限りにおいて、本資料および当社ハードウェア／ソフトウェア製品について、商品性および特定目的との合致に関する保証ならびに第三者の権利を侵害しないことの保証を含め、明示または黙示のいかなる保証も行いません。
8. 当社製品をご使用の際は、最新の製品情報（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、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合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いますが、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、データシート等において高信頼性、Harsh environment 向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他社会的損害等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、お客様の機器・システムとしての出荷保証を行ってください。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、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
10. 当社製品の環境適合性等の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。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. 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・システムに使用することはできません。当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
12. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。
13. 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禁じます。
14. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

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直接的、間接的に支配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。

(Rev.5.0-1 2020.10)

本社所在地

〒 135-0061 東京都江東区豊洲 3-2-24（豊洲フォレシア）

www.renesas.com

お問合せ窓口

弊社の製品や技術、ドキュメントの最新情報、最寄の営業お問合せ窓口に関する情報などは、弊社ウェブサイトをご覧ください。

www.renesas.com/contact/

商標について

ルネサスおよびルネサスロゴはルネサス エレクトロニクス株式会社の商標です。すべての商標および登録商標は、それぞれの所有者に帰属します。

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本ドキュメントおよびテクニカルアップデートを参照してください。

1. 静電気対策
   CMOS 製品の取り扱いの際は静電気防止を心がけてください。CMOS 製品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、当社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、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、CMOS 製品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. 電源投入時の処置
   電源投入時は、製品の状態は不定です。電源投入時には、LSI の内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。
3. 電源オフ時における入力信号
   当該製品の電源がオフ状態のときに、入力信号や入出力プルアップ電源を入れないでください。入力信号や入出力プルアップ電源からの電流注入により、誤動作を引き起こしたり、異常電流が流れ内部素子を劣化させたりする場合があります。資料中に「電源オフ時における入力信号」についての記載のある製品は、その内容を守ってください。
4. 未使用端子の処理
   未使用端子は、「未使用端子の処理」に従って処理してください。CMOS 製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI 周辺のノイズが印加され、LSI 内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。
5. クロックについて
   リセット時は、クロックが安定した後、リセットを解除してください。プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。
6. 入力端子の印加波形
   入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。CMOS 製品の入力がノイズなどに起因して、V_{IL}（Max.）から V_{IH}（Min.）までの領域にとどまるような場合は、誤動作を引き起こす恐れがあります。入力レベルが固定の場合はもちろん、V_{IL}（Max.）から V_{IH}（Min.）までの領域を通過する遷移期間中にチャタリングノイズなどが入らないように使用してください。
7. リザーブアドレス（予約領域）のアクセス禁止
   リザーブアドレス（予約領域）のアクセスを禁止します。アドレス領域には、将来の拡張機能用に割り付けられている リザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。
8. 製品間の相違について
   型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。同じグループのマイコンでも型名が違うと、フラッシュメモリ、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ幅射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

このマニュアルの使い方

対 象 者　このマニュアルは、RL78マイクロコントローラ製品の機能を理解し、その応用システムを設計するユーザのエンジニアを対象としています。

目　　的　このマニュアルは、RL78マイクロコントローラ製品の持つ各種命令機能を理解していただくことを目的とします。

構　　成　このマニュアルは、大きく分けて次の内容で構成しています。

・CPU機能
・命令セット
・命令の説明

読 み 方　このマニュアルを読むにあたっては、電気、論理回路およびマイクロコントローラの一通りの知識を必要とします。

・ニモニックが分かっていて、命令機能の詳細を確認するとき
「付録A　命令索引（機能別）」および「付録B　命令索引（アルファベット順）」を利用してください。
・ニモニックは分からないが、大体の機能が分かっている命令を確認するとき
「第5章　命令セット」でそのニモニックを調べ、そのあと「第6章　命令の説明」で機能を調べてください。
・一通りRL78マイクロコントローラ製品の各種命令を理解しようとするとき
目次に従って読んでください。
・RL78マイクロコントローラ製品のハードウエア機能について知りたいとき
各製品のユーザーズ・マニュアルを参照してください。

凡　　例　データ表記の重み：左側が上位桁、右側が下位桁
注　　：本文中につけた注の説明
注意　　：特に気をつけていただきたい内容
備考　　：本文の補足説明
数の表記　　：2進数 …xxxxBまたはxxxx
10進数…xxxx
16進数…xxxxH

目次

第1章　概説

RL78マイクロコントローラのCPUコアは，命令フェッチ用のバスとアドレス･データ･バスがそれぞれ独立したハーバード・アーキテクチャを採用しています。さらに，フェッチ，デコード，メモリ・アクセスの3段パイプライン制御を採用することで，従来のCPUコアよりも効率が飛躍的に向上しています。高性能かつ高機能な処理を必要とする様々なアプリケーションに対して，高性能かつ高速な命令処理で応えることができます。

1.1　特徴

RL78マイクロコントローラの主な特徴を以下に示します。

RL78マイクロコントローラは，命令の種類やクロック数，パフォーマンスの違いなどにより，RL78-S1コア，RL78-S2コア，RL78-S3コアの3種類に分類されます。

・3段パイプラインのCISCアーキテクチャ
・アドレス空間：1Mバイト
・最小命令実行時間：1命令1クロック実行
・汎用レジスタ：8ビット・レジスタ×8
・命令の種類：74種類（RL78-S1コア）
　　75種類（RL78-S2コア）
　　81種類（RL78-S3コア）
・データ配置：リトル・エンディアン

1.2　各CPUコアの機能の違い

表1－1に，RL78-S1コア，RL78-S2コア，RL78-S3コアの機能の違いを示します。

表1－1　各CPUコアの機能の違い

項目	RL78-S1コア	RL78-S2コア	RL78-S3コア
CPU	8ビット	16ビット	16ビット
命令の種類	74種類	75種類	81種類
汎用レジスタ	8ビット・レジスタ ×8（バンクなし）	8ビット・レジスタ ×8×4バンク	8ビット・レジスタ ×8×4バンク
乗除算積和演算命令	なし	なし	あり

注意　3種類のCPUコアの命令は共通ですが，RL78-S1コアと他のCPUコアではクロック数の異なる命令があります。詳細は「5.5 オペレーション一覧」を参照してください。

★　備考　製品によって搭載するCPUコアは異なります。以下，CPUコアごとに製品例を示します。下記以外の製品は，各製品のユーザーズ・マニュアルを参照してください。

・RL78-S1コア：RL78/G10，RL78/G1M，RL78/G1N
・RL78-S2コア：RL78/G12，RL78/G13，RL78/G13A，RL78/G1A，RL78/G1C，RL78/G1D，RL78/G1E，RL78/G1P，RL78/I1A，RL78/L12，RL78/L13，RL78/D1A，RL78/F12
・RL78-S3コア：RL78/G23，RL78/G11，RL78/G14，RL78/G1F，RL78/G1G，RL78/G1H，RL78/H1D，RL78/I1B，RL78/I1C，RL78/I1D，RL78/I1E，RL78/L1A，RL78/L1C，RL78/F23，RL78/F24，RL78/F13，RL78/F14，RL78/F15

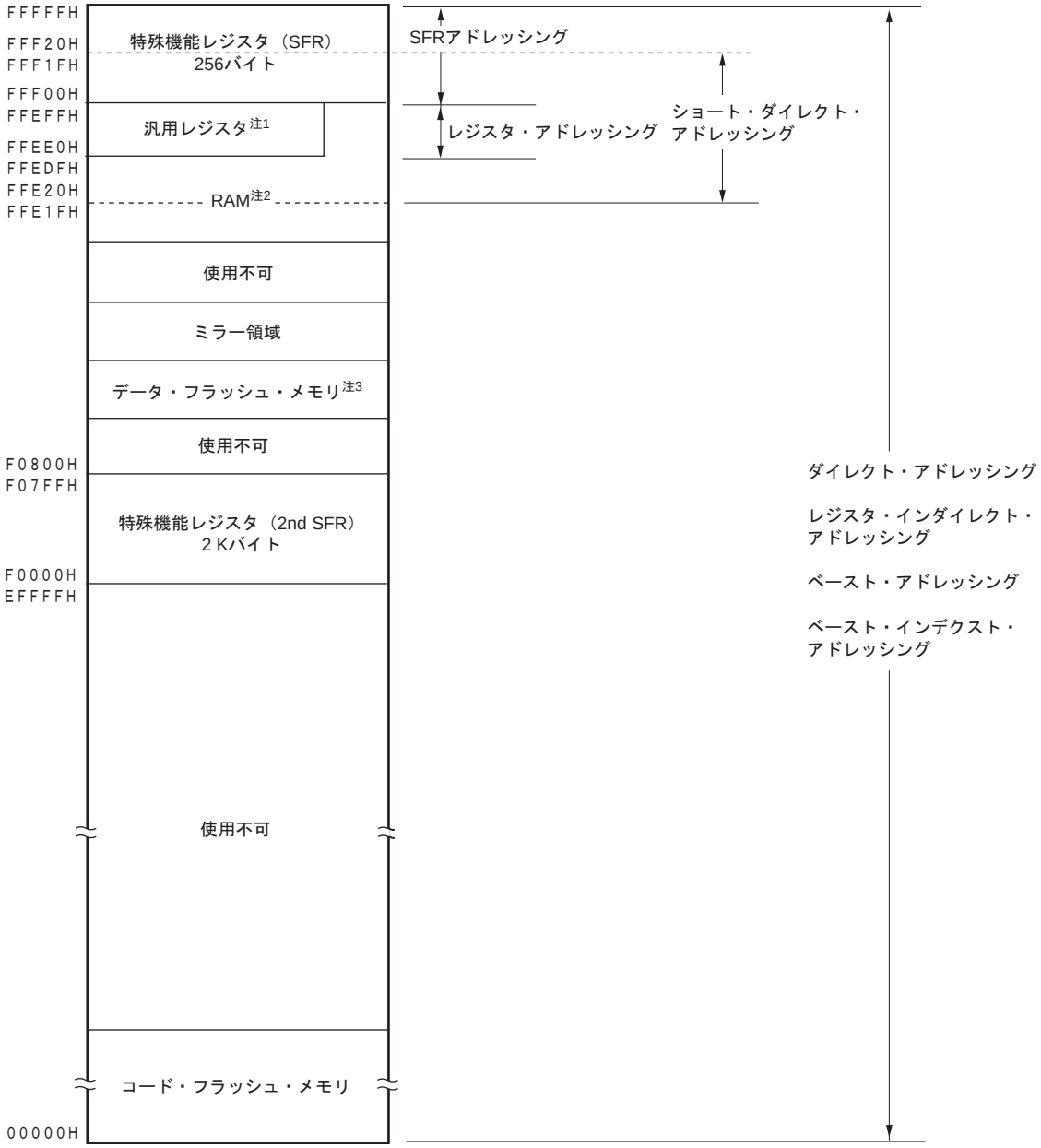
第2章　メモリ空間

2.1　メモリ空間

RL78マイクロコントローラのメモリ空間は，１Mバイトのアドレス空間をアクセスできます。

図2－1にRL78マイクロコントローラのメモリ・マップを示します。

図2－1　RL78マイクロコントローラのメモリ・マップ



注1.　RL78-S1コアの汎用レジスタ領域は8バイトでFFEF8H-FFEFFHです。FFEE0H-FFEF7Hは使用不可領域になります。
RL78-S2コアとRL78-S3コアの汎用レジスタ領域は、32バイトでFFEE0H-FFEFFHです。

2.　セルフ・プログラミング時およびデータ・フラッシュ書き換え時は、各ライブラリで使用するため、使用禁止になる領域があります。各製品によって使用禁止の領域は異なります。詳細は各製品のユーザーズ・マニュアルを参照してください。

3.　データ・フラッシュ・メモリがない製品は使用不可領域になります。各製品によってデータ・フラッシュ・メモリのサイズは異なります。詳細は各製品のユーザーズ・マニュアルを参照してください。

2.2　内部プログラム・メモリ空間

RL78マイクロコントローラでは，00000H-EFFFFHが内部プログラム・メモリ空間となります。

内部ROM（フラッシュ・メモリ）容量の最大値は各製品のユーザーズ・マニュアルを参照してください。

2.2.1　ミラー領域

CPUコアによって以下のとおりミラー領域は異なります。詳細は「3.4.1　プロセッサ・モード・コントロール・レジスタ（PMC）」を参照してください。

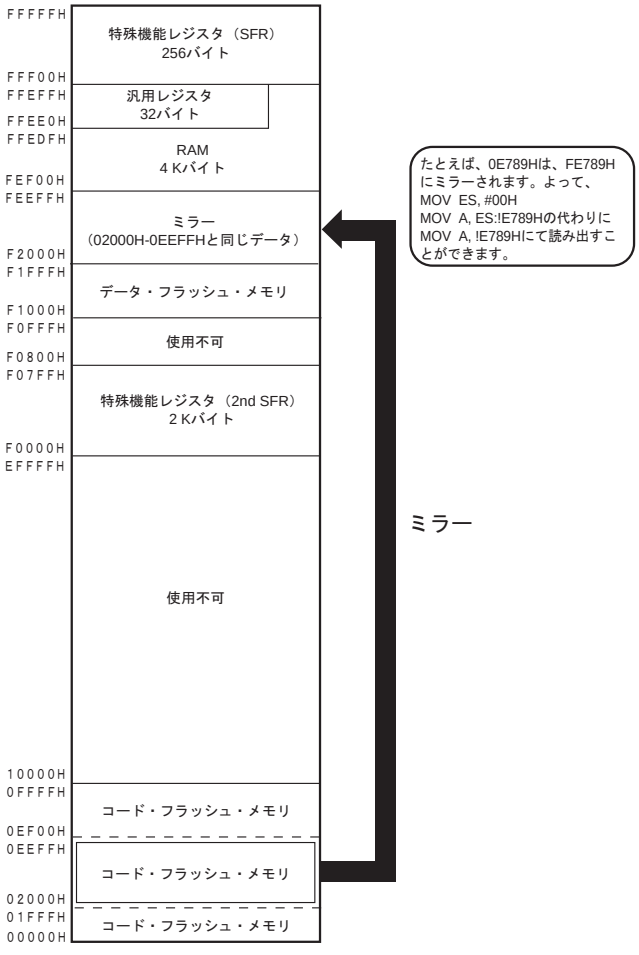
・RL78-S1コア　MAA＝0：00000H-05EFFHをF8000H-FDEFFHへミラー
　　MAA＝1：設定禁止
・RL78-S2コア　MAA＝0：00000H-0FFFFHをF0000H-FFFFFHへミラー
　　MAA＝1：10000H-1FFFFHをF0000H-FFFFFHへミラー
・RL78-S3コア　MAA＝0：00000H-0FFFFHをF0000H-FFFFFHへミラー
　　MAA＝1：10000H-1FFFFHをF0000H-FFFFFHへミラー

ミラー先の領域からデータを読み出すことにより，オペランドにESレジスタを持たない命令を使用することができるため，短いコードでデータ・フラッシュ内容の読み出しを行うことができます。ただし，SFR，拡張SFR（2nd SFR），RAM，データ・フラッシュ・メモリ領域，使用不可領域にはミラーされません。

また，ミラー領域は製品によって異なります。詳細は各製品のユーザーズ・マニュアルを参照してください。なお，ミラー領域は読み出しのみ可能で，命令フェッチはできません。

図2－2に例を示します。

図2－2　RL78-S2コア（64Kバイト・フラッシュ・メモリ，4KバイトRAM）の場合の例



2.2.2　ベクタ・テーブル領域

ベクタ・テーブル領域は，00000H-0007FHの128バイトの領域です。ベクタ・テーブル領域には，リセット，各割り込み要求発生により分岐するときのプログラム・スタート・アドレスを格納しておきます。また，ベクタ・コードは2バイトとしているため，割り込みの飛び先アドレスは00000H-0FFFFHの64 Kアドレスとなります。

16ビット・アドレスのうち，下位8ビットは偶数アドレスに，上位8ビットは奇数アドレスに格納されます。

なお，RL78-S2コアとRL78-S3コアの製品でブート・スワップを使用する際には，01000H-0107FHにもベクタ・テーブルを設定してください。

2.2.3　CALLT命令テーブル領域

CALLT命令テーブル領域は，00080H-000BFHの64バイトの領域です。この領域には，2バイト・コール命令（CALLT）のサブルーチン・エントリ・アドレスを格納できます。サブルーチン・エントリ・アドレスは，00000H-0FFFFH内の値を設定してください（アドレス・コードが2バイトのため）。

なお，RL78-S2コアとRL78-S3コアの製品でブート・スワップを使用する際には，01080H-010BFHにもCALLT命令テーブルを設定してください。

2.2.4　オプション・バイト領域

オプション・バイト領域は，000C0H-000C3Hの4バイトの領域です。

なお，RL78-S2コアとRL78-S3コアの製品でブート・スワップを使用する際には，010C0H-010C3Hにもオプション・バイトを設定してください。

2.2.5　オンチップ・デバッグ・セキュリティID設定領域

オンチップ・デバッグ・セキュリティID設定領域は，000C4H-000CDHの10バイトの領域です。

なお，RL78-S2コアとRL78-S3コアの製品でブート・スワップを使用する際には，010C4H-010CDHにも10バイトのオンチップ・デバッグ・セキュリティIDを設定してください。

2.3　内部データ・メモリ（内部RAM）空間

内部データ・メモリ（内部RAM）空間は，汎用レジスタが割り当てられた領域を除き，データ領域として使用できるほか，プログラム領域として命令を書いて実行することができます。RAM領域は，上限アドレスをFFEFFHで固定し，製品に搭載するRAMサイズに合わせて下限アドレスを伸ばしていきます。下限アドレスは搭載する製品によって変わりますので，各製品のユーザーズ・マニュアルを参照してください。

また，汎用レジスタが割り当てられた領域は，以下のとおりCPUコアによって異なります。詳細は「3.2　汎用レジスタ」を参照してください。

・RL78-S1コア：FFEF8H-FFEFFH
・RL78-S2コア：FFEE0H-FFEFFH
・RL78-S3コア：FFEE0H-FFEFFH

注意1.　スタック領域は，汎用レジスタ領域以外のアドレスを指定してください。汎用レジスタ領域は，命令フェッチやスタック領域としての使用を禁止します。
2.　RAM空間から内部プログラム・メモリ空間への分岐命令は，レラティブ・アドレッシングでは行わないでください。

2.4　特殊機能レジスタ（SFR：Special Function Register）領域

SFRは，汎用レジスタとは異なり，それぞれ特別な機能を持つレジスタです。

SFR空間は，FFF00H-FFFFFHの領域に割り付けられています。

SFRは，演算命令，転送命令，ビット操作命令などにより，汎用レジスタと同じように操作できます。操作可能なビット単位（1，8，16）は，各SFRで異なります。

操作ビット単位ごとの指定方法を次に示します。

・1ビット操作

1ビット操作命令のオペランド（sfr.bit）には，次のような記述をしてください。

ビット名称が定義されている場合：<ビット名称>

ビット名称が定義されていない場合：<レジスタ名>.<ビット番号>または<アドレス>.<ビット番号>

・8ビット操作

8ビット操作命令のオペランド（sfr）にアセンブラで定義されている略号を記述します。アドレスでも指定できます。

・16ビット操作

16ビット操作命令のオペランド（sfrp）にアセンブラで定義されている略号を記述します。アドレスを指定するときは偶数アドレスを記述してください。

2.5　拡張特殊機能レジスタ（2nd SFR: 2nd Special Function Register）領域

拡張SFR（2nd SFR）は，汎用レジスタとは異なり，それぞれ特別な機能を持つレジスタです。

拡張SFR空間は，F0000H-F07FFHの領域です。SFR領域（FFF00H-FFFFFH）以外のSFRが割り付けられています。ただし，拡張SFR領域のアクセス命令はSFR領域より1バイト長くなります。

拡張SFRは，演算命令，転送命令，ビット操作命令などにより，汎用レジスタと同じように操作できます。操作可能なビット単位（1，8，16）は，各拡張SFRで異なります。

操作ビット単位ごとの指定方法を次に示します。

・1ビット操作

1ビット操作命令のオペランド（sfr.bit）には，次のような記述をしてください。

ビット名称が定義されている場合：<ビット名称>

ビット名称が定義されていない場合：<レジスタ名>.<ビット番号>または<アドレス>.<ビット番号>

・8ビット操作

8ビット操作命令のオペランド（sfr）にアセンブラで定義されている略号を記述します。アドレスでも指定できます。

・16ビット操作

16ビット操作命令のオペランド（sfrp）にアセンブラで定義されている略号を記述します。アドレスを指定するときは偶数アドレスを記述してください。

第3章　レジスタ

3.1　制御レジスタ

プログラム・シーケンス，ステータス，スタック・メモリの制御など専用の機能を持ったレジスタです。制御レジスタには，プログラム・カウンタ（PC），プログラム・ステータス・ワード（PSW），スタック・ポインタ（SP）があります。

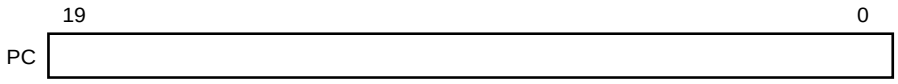
3.1.1　プログラム・カウンタ（PC）

プログラム・カウンタは，次に実行するプログラムのアドレス情報を保持する20ビット・レジスタです。

通常動作時には，フェッチする命令のバイト数に応じて自動的にインクリメントされます。分岐命令実行時には，イミーディエト・データやレジスタの内容がセットされます。

リセット信号の発生により，0000H，0001H番地のリセット・ベクタ・テーブルの値が下位16ビットにセットされます。上位4ビットは0000にクリアされます。

図3－1　プログラム・カウンタの構成



3.1.2　プログラム・ステータス・ワード（PSW）

プログラム・ステータス・ワードは，命令の実行によってセット，リセットされる各種フラグで構成される8ビット・レジスタです。

割り込みレベルが4レベル対応の製品では，ビット2にISP1フラグが追加されます。

プログラム・ステータス・ワードの内容は，ベクタ割り込み要求受け付け発生時およびPUSH PSW命令の実行時にスタック領域に格納され，RETB，RETI命令およびPOP PSW命令の実行時に復帰されます。

リセット信号の発生により06Hになります。

・RL78-S1コアの場合

図3－2　プログラム・ステータス・ワードの構成

	7							0
PSW	IE	Z	0	AC	0	ISP1	ISP0	CY

・RL78-S2コアとRL78-S3コアの場合

図3－3　プログラム・ステータス・ワードの構成

	7							0
PSW	IE	Z	RBS1	AC	RBS0	ISP1	ISP0	CY

（1）割り込み許可フラグ（IE）

CPUの割り込み要求受け付け動作を制御するフラグです。

IE = 0のときは割り込み禁止（DI）状態となり，マスカブル割り込みはすべて禁止されます。

IE = 1のときは割り込み許可（EI）状態となります。このときマスカブル割り込み要求の受け付けは，インサービス・プライオリティ・フラグ（ISP1, ISP0），各割り込み要因に対する割り込みマスク・フラグおよび優先順位指定フラグにより制御されます。

このフラグは，DI命令の実行または割り込みの受け付けでリセット（0）され，EI命令実行によりセット（1）されます。

（2）ゼロ・フラグ（Z）

演算や比較で結果がゼロまたは等しいときセット（1）され，それ以外のときにリセット（0）されるフラグです。

（3）レジスタ・バンク選択フラグ（RBS0, RBS1）

4個のレジスタ・バンクのうちの1つを選択する2ビットのフラグです。

SEL RBn命令の実行によって選択されたレジスタ・バンクを示す2ビットの情報が格納されています。

注意　RL78-S1コアにはありません。

（4）補助キャリー・フラグ（AC）

演算結果で，ビット3からキャリーがあったとき，またはビット3へのボローがあったときセット（1）され，それ以外のときリセット（0）されるフラグです。

（5）インサービス・プライオリティ・フラグ（ISP0, ISP1）

受け付け可能なマスカブル・ベクタ割り込みの優先順位レベルを管理するフラグです。優先順位指定フラグ・レジスタ（PR）でISP0, ISP1の値より低位に指定されたベクタ割り込み要求は受け付け禁止となります。なお，実際に割り込み要求が受け付けられるかどうかは，割り込み許可フラグ（IE）の状態により制御されます。

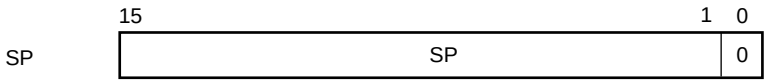
（6）キャリー・フラグ（CY）

加減算命令実行時のオーバフロー，アンダフローを記憶するフラグです。また，ローテート命令実行時はシフト・アウトされた値を記憶し，ビット演算命令実行時にはビット・アキュームレータとして機能します。

3.1.3　スタック・ポインタ（SP）

メモリのスタック領域の先頭アドレスを保持する16ビットのレジスタです。スタック領域としては内部RAM領域のみ設定可能です。

図3－4　スタック・ポインタの構成



スタック・ポインタを用いたスタック・アドレッシングでは，スタック・メモリへの書き込み（退避）動作に先立ってデクリメントされ，スタック・メモリからの読み取り（復帰）動作のあとインクリメントされます。

SPの内容はリセット信号の発生により不定になりますので，必ずスタック使用前にイニシャライズしてください。またSPの設定値は必ず偶数にしてください。奇数を指定すると，最下位ビットは自動的に0が設定されます。表3－1にRL78マイクロコントローラのスタック・サイズを示します。

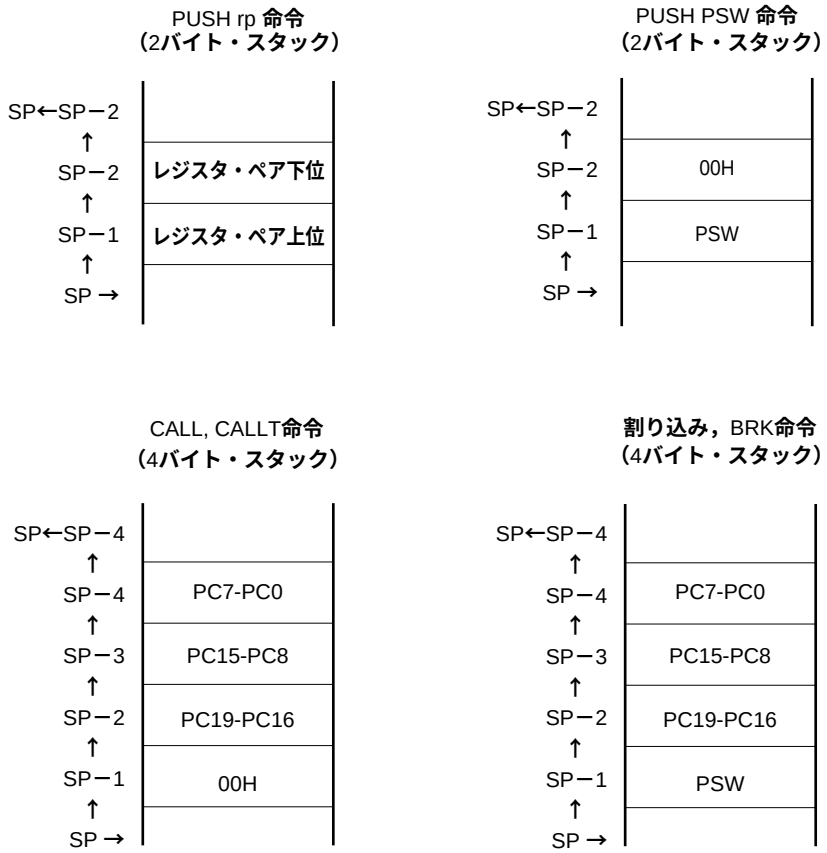
注意　汎用レジスタの空間はスタック領域としての使用を禁止します。

表3－1　RL78マイクロコントローラのスタック・サイズ

退避命令	復帰命令	スタック・サイズ
PUSH rp	POP rp	2バイト
PUSH PSW	POP PSW	2バイト
CALL, CALLT	RET	4バイト
割り込み	RETI	4バイト
BRK	RETB	4バイト

RL78マイクロコントローラでは，各スタック動作によって退避されるデータは図3－5のようになります。

図3－5　スタック・メモリへ退避されるデータ



スタック・ポインタは内部RAMのみ指定可能です。ただし，アドレスはF0000H-FFFFFHの空間を指定可能ですので，内部RAMの空間を越えないようにしてください。内部RAM空間以外を指定した場合，書き込みは無視され，読み出すと不定値が読み出されます。

3.2　汎用レジスタ

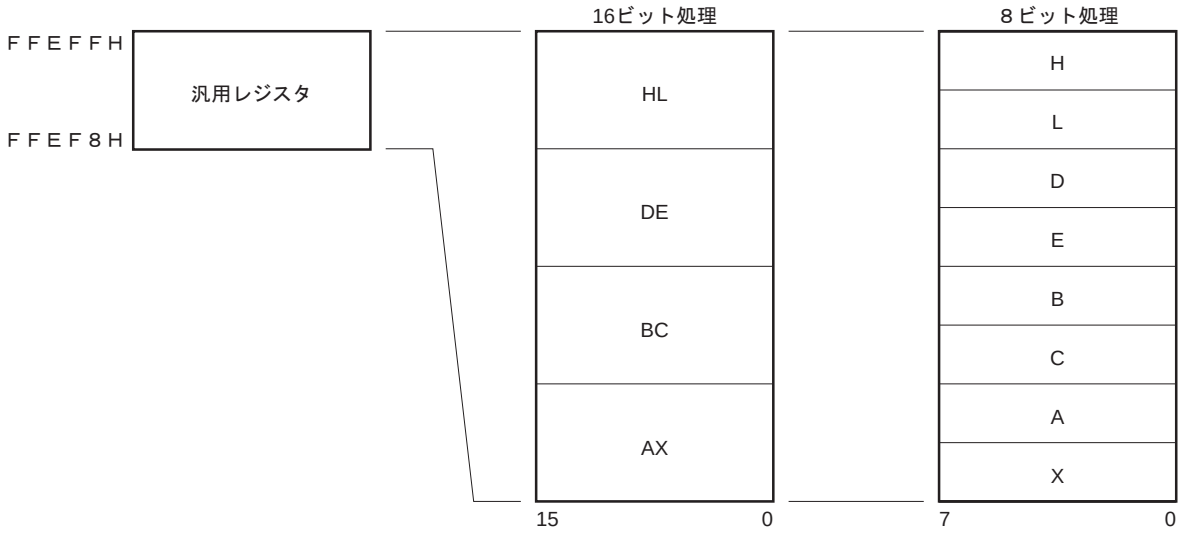
3.2.1　RL78-S1コアの汎用レジスタ

RL78-S1コアの汎用レジスタは，データ・メモリの特定番地（FFEF8H-FFEFFH）にマッピングされており，8ビット・レジスタ8個（X，A，C，B，E，D，L，H）で構成されています。

各レジスタは，それぞれ8ビット・レジスタとして使用できるほか，2個の8ビット・レジスタをペアとして16ビット・レジスタとしても使用できます（AX，BC，DE，HL）。

注意　汎用レジスタ（FFEF8H-FFEFFH）の空間は命令フェッチやスタック領域としての使用を禁止します。

図3－6　RL78-S1コアの汎用レジスタの構成



3.2.2　RL78-S2コアとRL78-S3コアの汎用レジスタ

RL78-S2コアとRL78-S3コアの汎用レジスタは，データ・メモリの特定番地（FFEE0H-FFEFFH）にマッピングされており，8ビット・レジスタ8個（X，A，C，B，E，D，L，H）を1バンクとして4バンクのレジスタで構成されています。

各レジスタは，それぞれ8ビット・レジスタとして使用できるほか，2個の8ビット・レジスタをペアとして16ビット・レジスタとしても使用できます（AX，BC，DE，HL）。

命令実行時に使用するレジスタ・バンクはCPU制御命令（SEL RBn）によって設定します。4レジスタ・バンク構成になっていますので，通常処理で使用するレジスタと割り込み処理で使用するレジスタをバンク切り替えすれば，効率のよいプログラムを作成できます。

注意　汎用レジスタ（FFEE0H-FFEFFH）の空間は命令フェッチやスタック領域としての使用を禁止します。

図3－7　RL78-S2コアとRL78-S3コアの汎用レジスタの構成

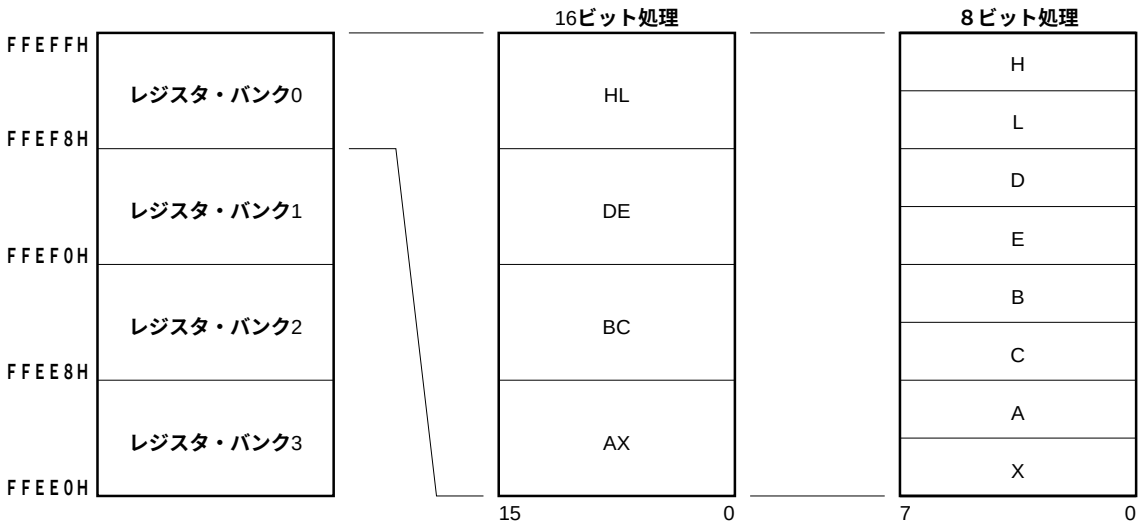


表3−2　汎用レジスター覧

バンク名	レジスタ				絶対アドレス
	機能名称		絶対名称		
	16ビット処理	8ビット処理	16ビット処理	8ビット処理	
BANK0	HL	H	RP3	R7	FFEFFH
		L		R6	FFEFEH
	DE	D	RP2	R5	FFEFDH
		E		R4	FFEFCH
	BC	B	RP1	R3	FFEFBH
		C		R2	FFEFAH
	AX	A	RP0	R1	FFEF9H
		X		R0	FFEF8H
BANK1[注]	HL	H	RP3	R7	FFEF7H
		L		R6	FFEF6H
	DE	D	RP2	R5	FFEF5H
		E		R4	FFEF4H
	BC	B	RP1	R3	FFEF3H
		C		R2	FFEF2H
	AX	A	RP0	R1	FFEF1H
		X		R0	FFEF0H
BANK2[注]	HL	H	RP3	R7	FFEEFH
		L		R6	FFEEEH
	DE	D	RP2	R5	FFEEDH
		E		R4	FFEECH
	BC	B	RP1	R3	FFEEBH
		C		R2	FFEEAH
	AX	A	RP0	R1	FFEE9H
		X		R0	FFEE8H
BANK3[注]	HL	H	RP3	R7	FFEE7H
		L		R6	FFEE6H
	DE	D	RP2	R5	FFEE5H
		E		R4	FFEE4H
	BC	B	RP1	R3	FFEE3H
		C		R2	FFEE2H
	AX	A	RP0	R1	FFEE1H
		X		R0	FFEE0H

注　RL78-S1コアにはありません。

3.3　ES，CSレジスタ

ESレジスタでデータ・アクセス，CSレジスタで（レジスタ・ダイレクト・アドレシング）分岐命令実行時の，それぞれ上位アドレスを指定できます。使用方法は「第4章　アドレッシング」を参照してください。

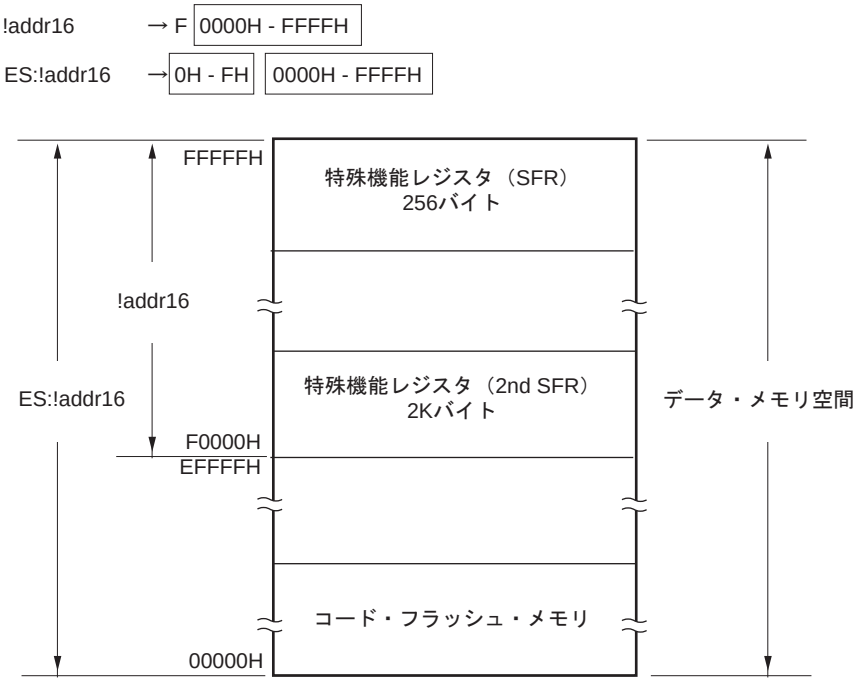
ESレジスタのリセット後の初期値は0FH，CSレジスタのリセット後の初期値は00Hです。

図3－8　ES/CSレジスタの構成

	7	6	5	4	3	2	1	0
ES	0	0	0	0	ES3	ES2	ES1	ES0
CS	0	0	0	0	CS3	CS2	CS1	CS0

16ビット・アドレスでアクセスできるデータ領域はF0000H-FFFFFHの64 Kバイト空間ですが，ES:を付加すると00000H-FFFFFHの1 Mバイト空間に拡張できます。

図3－9　データ・アクセス領域の拡張



3.4　特殊機能レジスタ（SFR）

RL78マイクロコントローラでアドレス固定のSFRを表3－3に示します。

表3－3　固定SFRレジスター覧

アドレス	レジスタ名称
FFFF8H	SPL
FFFF9H	SPH
FFFFAH	PSW
FFFFBH	Reserve
FFFFCH	CS
FFFFDH	ES
FFFFEH	PMC
FFFFFH	MEM

3.4.1　プロセッサ・モード・コントロール・レジスタ（PMC）

プロセッサのモードを制御する8ビット・レジスタです。詳細は「2.2　内部プログラム・メモリ空間」を参照してください。リセット時の初期値は00Hになります。

（1）RL78-S1コアの場合

図3－10　RL78-S1コアのプロセッサ・モード・コントロール・レジスタ（PMC）の構成

アドレス：FFFFEH　　リセット時：00H　　R/W

略号	7	6	5	4	3	2	1	0
PMC	0	0	0	0	0	0	0	MAA

MAA	F8000H-FDEFFH[注]へミラーするフラッシュ・メモリ空間選択
0	00000H-05EFFHをF8000H-FDEFFHへミラー
1	設定禁止

注　F8000H-FDEFFHの中にはSFR，RAM領域があり，この重なった領域はSFR，RAMが優先されます。

注意　PMCレジスタは，基本的には初期値のままで書き込みは不要です。ただし，RL78-S2コアとRL78-S3コアとの互換性を保つため00Hの書き込みのみ許可します。

（2）RL78-S2コアとRL78-S3コアの場合

図3－11　RL78-S2コアとRL78-S3コアのプロセッサ・モード・コントロール・レジスタ（PMC）の構成

アドレス：FFFFEH　　リセット時：00H　　R/W

略号	7	6	5	4	3	2	1	0
PMC	0	0	0	0	0	0	0	MAA

MAA	F0000H-FFFFFH[注]へミラーするフラッシュ・メモリ空間選択
0	00000H-0FFFFHをF0000H-FFFFFHへミラー
1	10000H-1FFFFHをF0000H-FFFFFHへミラー

注　F0000H-FFFFFHの中にはSFR，RAM領域があり，この重なった領域はSFR，RAMが優先されます。

注意　PMCの設定後，1命令以上空けてミラー領域にアクセスしてください。

第4章　アドレッシング

アドレッシングは，処理データ・アドレスに対するアドレッシング，プログラム・アドレスに対するアドレッシング，の2種類から構成されています。次にそれぞれのアドレッシング・モードを示します。

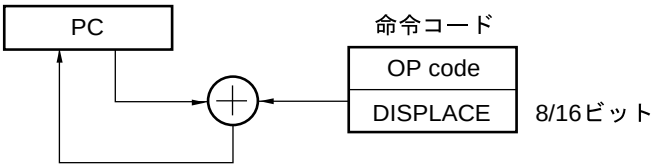
4.1　命令アドレスのアドレッシング

4.1.1　レラティブ・アドレッシング

【機　　能】

プログラム・カウンタ（PC）の値（次に続く命令の先頭アドレス）に対し，命令語に含まれるディスプレースメント値（符号付きの補数データ：－128～＋127または－32768～＋32767）を加算した結果を，プログラム・カウンタ（PC）に格納し分岐先プログラム・アドレスを指定するアドレッシングです。レラティブ・アドレッシングは分岐命令のみに適用されます。

図4－1　レラティブ・アドレッシングの概略



4.1.2　イミーディエト・アドレッシング

【機　　能】

命令語中のイミーディエト・データをプログラム・カウンタに格納し，分岐先プログラム・アドレスを指定するアドレッシングです。

イミーディエト・アドレッシングには20ビットのアドレスを指定するCALL !!addr20/BR !!addr20と，16ビットのアドレスを指定するCALL !addr16/BR !addr16があります。16ビット・アドレスを指定する場合は上位4ビットには0000が入ります。

図4－2　CALL !!addr20/BR !!addr20の例

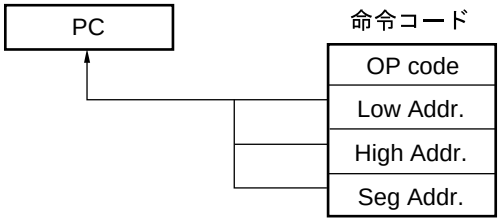
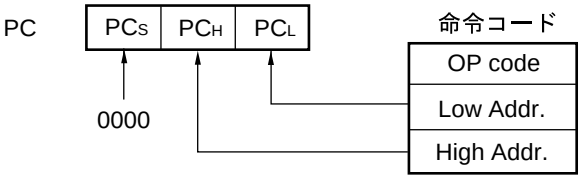


図4－3　CALL !addr16/BR !addr16の例



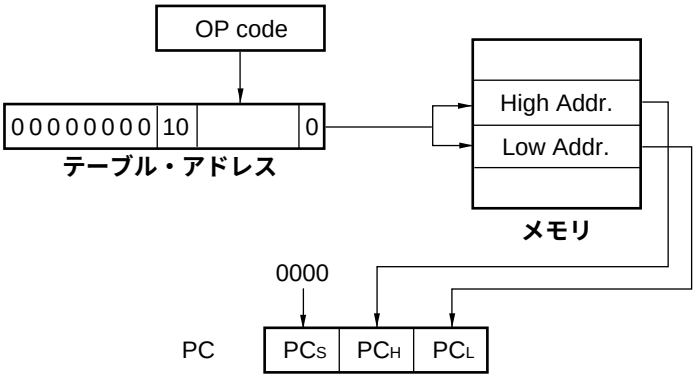
4.1.3　テーブル・インダイレクト・アドレッシング

【機　　能】

命令語中の5ビット・イミーディエト・データによりCALLTテーブル領域（0080H-00BFH）内のテーブル・アドレスを指定し，その内容とそれに続くアドレスの内容を16ビット・データとしてプログラム・カウンタ（PC）に格納し，プログラム・アドレスを指定するアドレッシングです。テーブル・インダイレクト・アドレッシングはCALLT命令にのみ適用されます。

RL78マイクロコントローラでは，00000H-0FFFFHの64 Kバイト空間のみ分岐可能です。

図4－4　テーブル・インダイレクト・アドレッシングの概略

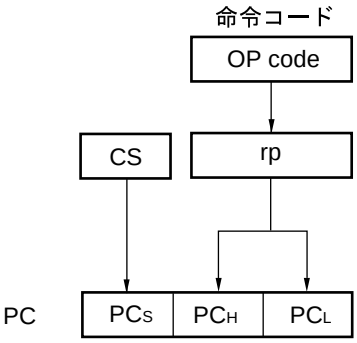


4.1.4　レジスタ・ダイレクト・アドレッシング

【機　　能】

命令語で指定されるカレント・レジスタ・バンク内の汎用レジスタ・ペア（AX/BC/DE/HL）とCSレジスタの内容を20ビット・データとしてプログラム・カウンタ（PC）に格納し，プログラム・アドレスを指定するアドレッシングです。レジスタ・ダイレクト・アドレッシングはCALL AX / BC / DE / HLとBR AX命令にのみ適用されます。

図4－5　レジスタ・ダイレクト・アドレッシングの概略



4.2　処理データ・アドレスに対するアドレッシング

4.2.1　インプライド・アドレッシング

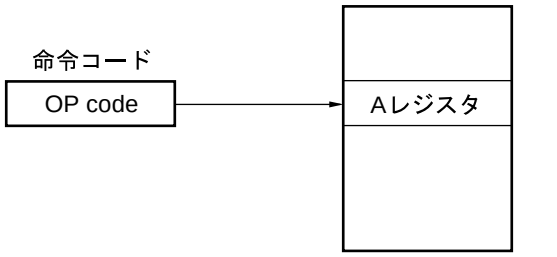
【機　　能】

アキュムレータなどの特別な機能を与えられたレジスタをアクセスする命令は，命令語中にはレジスタ指定フィールドを持たず命令語で直接指定します。

【オペランド形式】

インプライド・アドレッシングはMULU Xのみに適用されます。

図4－6　インプライド・アドレッシングの概略



4.2.2　レジスタ・アドレッシング

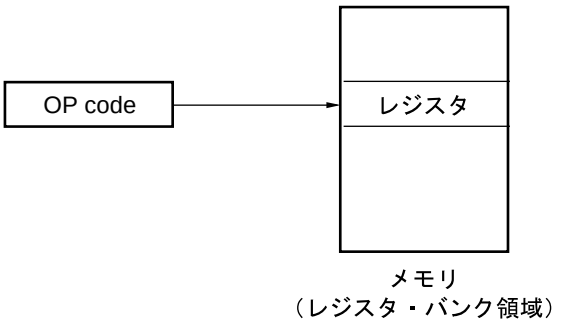
【機　　能】

汎用レジスタをオペランドとしてアクセスするアドレッシングです。8ビット・レジスタを指定する場合は命令語の3ビット，16ビット・レジスタを指定する場合は命令語の2ビットによりレジスタが選択されます。

【オペランド形式】

表現形式	記述方法
r	X，A，C，B，E，D，L，H
rp	AX，BC，DE，HL

図4－7　レジスタ・アドレッシングの概略



4.2.3　ダイレクト・アドレッシング

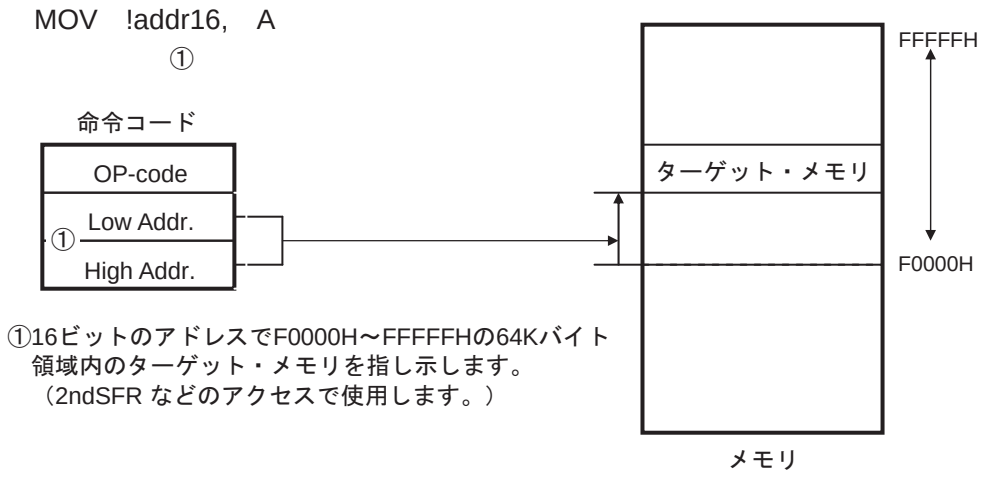
【機　　能】

命令語中のイミーディエト・データがオペランド・アドレスとなり，対象となるアドレスを直接指定するアドレッシングです。

【オペランド形式】

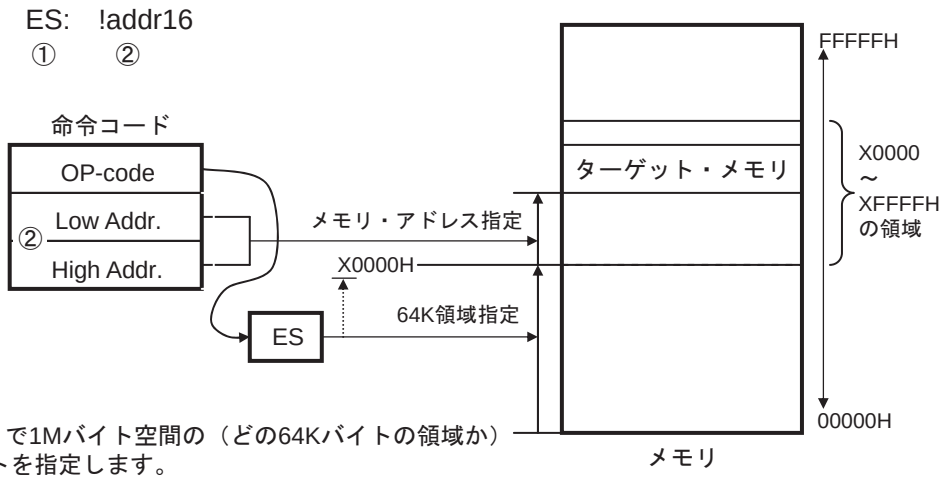
表現形式	記述方法
!addr16	ラベルまたは16ビット・イミーディエト・データ （F0000H-FFFFFH空間のみ指定可能）
ES:!addr16	ラベルまたは16ビット・イミーディエト・データ （ESレジスタにて上位4ビット・アドレス指定）

図4－8　!addr16の例



①16ビットのアドレスでF0000H～FFFFFHの64Kバイト領域内のターゲット・メモリを指し示します。
（2ndSFR などのアクセスで使用します。）

図4－9　ES:!addr16の例



①ESレジスタで1Mバイト空間の（どの64Kバイトの領域か）上位4ビットを指定します。
②16ビットのアドレスで①で指定したX0000H～XFFFFHの領域のターゲット・メモリを指し示します。
（ミラー領域以外の固定データアクセスに使います。）

4.2.4　ショート・ダイレクト・アドレッシング

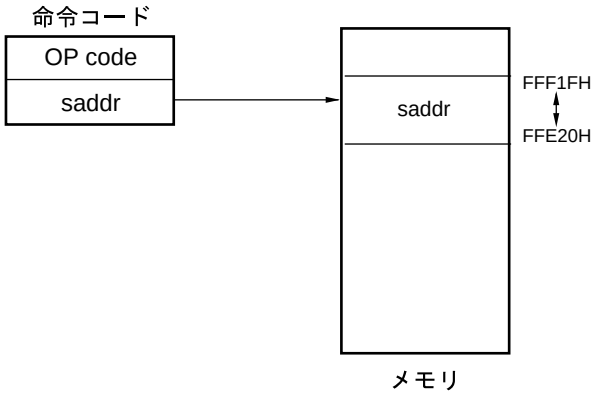
【機　　能】

命令語中の8ビット・データで対象となるアドレスを直接指定するアドレッシングです。このアドレッシングが適用されるのはFFE20H-FFF1FHの空間に限られます。

【オペランド形式】

表現形式	記述方法
SADDR	ラベルまたはFFE20H-FFF1FHのイミーディエト・データまたは0FE20H-0FF1FHのイミーディエト・データ （FFE20H-FFF1FH空間のみ指定可能）
SADDRP	ラベルまたはFFE20H-FFF1FHのイミーディエト・データまたは0FE20H-0FF1FHのイミーディエト・データ（偶数アドレスのみ） （FFE20H-FFF1FH空間のみ指定可能）

図4－10　ショート・ダイレクト・アドレッシングの概略



備考　SADDR，SADDRPは，（実アドレスの上位4ビット・アドレスを省略した）16ビットのイミーディエト・データでFE20H-FF1FHの値を記述することができます。また，20ビットのイミーディエト・データでFFE20H-FFF1FHの値を記述することもできます。
ただし，どちらの形式で書いても，メモリはFFE20H-FFF1FH空間のアドレスが指定されます。

4.2.5　SFRアドレッシング

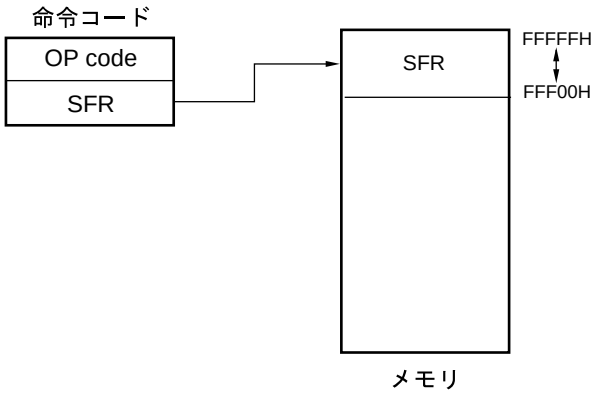
【機　　能】

命令語中の8ビット・データで対象となるSFRアドレスを直接指定するアドレッシングです。このアドレッシングが適用されるのはFFF00H-FFFFFHの空間に限られます。

【オペランド形式】

表現形式	記述方法
SFR	SFRレジスタ名
SFRP	16ビット操作可能なSFRレジスタ名（偶数アドレス）

図4－11　SFRアドレッシングの概略



4.2.6　レジスタ・インダイレクト・アドレッシング

【機　　能】

命令語で指定されたレジスタ・ペアの内容がオペランド・アドレスになり，対象となるアドレスを指定するアドレッシングです。

【オペランド形式】

表現形式	記述方法
−	[DE], [HL] （F0000H-FFFFFH空間のみ指定可能）
−	ES:[DE], ES:[HL] （ESレジスタにて上位4ビット・アドレス指定）

図4－12　[DE], [HL]の例

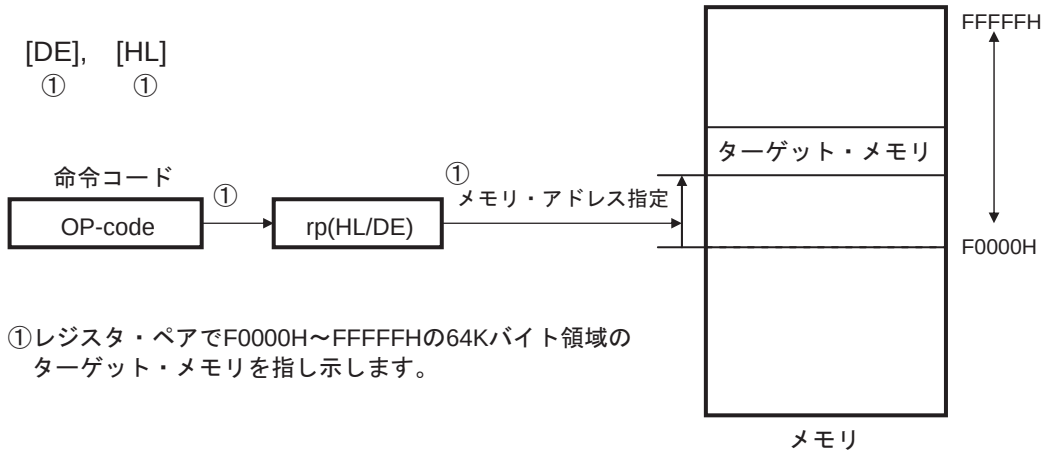
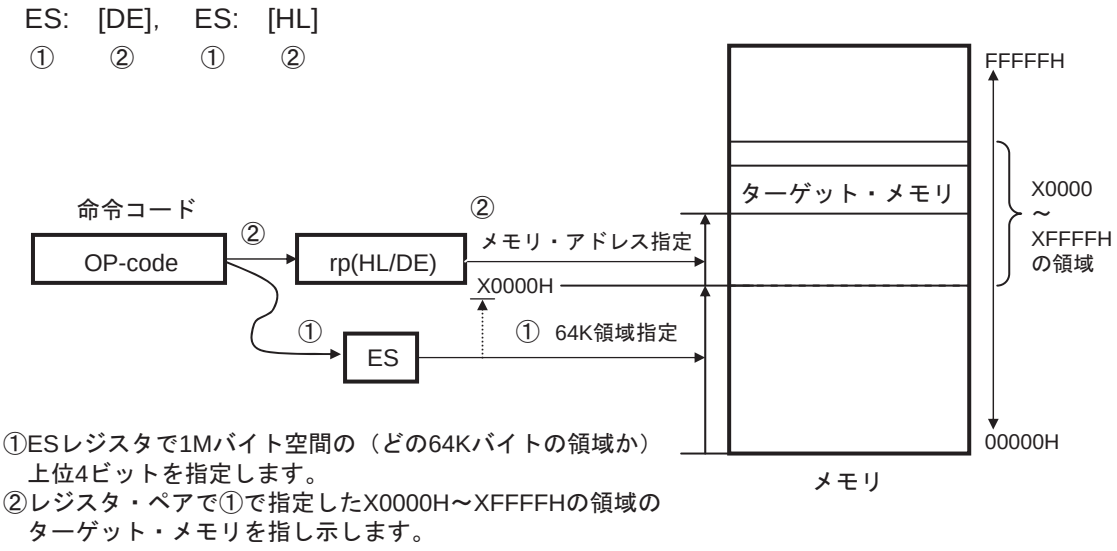


図4－13　ES:[DE], ES:[HL]の例



4.2.7　ベースト・アドレッシング

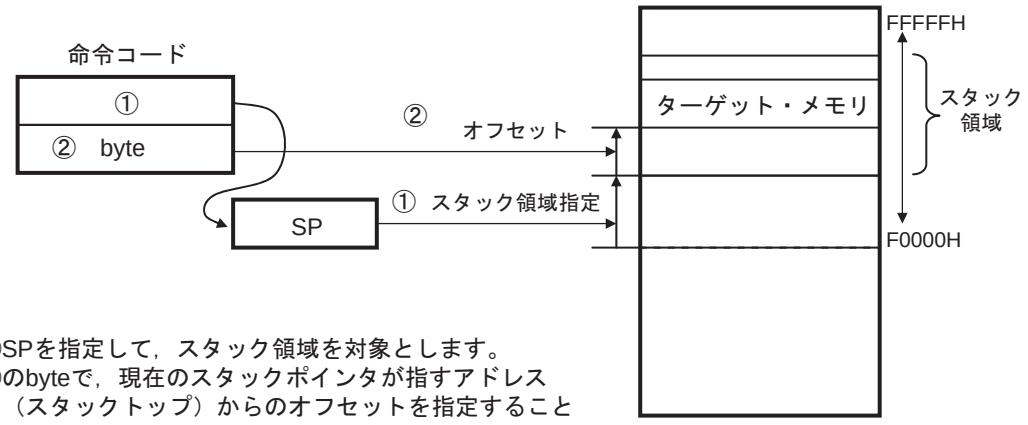
【機　　能】

命令語で指定されるレジスタ・ペアの内容または16ビットのイミーディエト・データをベース・アドレスとし，8ビット・イミーディエト・データまたは16ビット・イミーディエト・データをオフセット・データとしてベース・アドレスに加算した結果で，対象となるアドレスを指定するアドレッシングです。

【オペランド形式】

表現形式	記述方法
−	[HL＋byte], [DE + byte], [SP + byte] （F0000H-FFFFFH空間のみ指定可能）
−	word[B], word[C] （F0000H-FFFFFH空間のみ指定可能）
−	word[BC] （F0000H-FFFFFH空間のみ指定可能）
−	ES:[HL＋byte], ES:[DE + byte] （ESレジスタにて上位4ビット・アドレス指定）
−	ES:word[B], ES:word[C] （ESレジスタにて上位4ビット・アドレス指定）
−	ES:word[BC] （ESレジスタにて上位4ビット・アドレス指定）

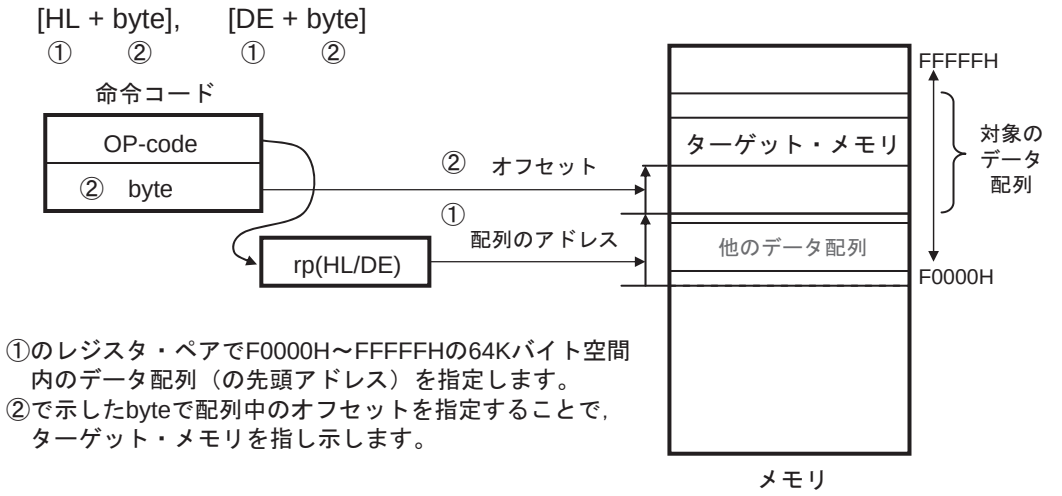
図4－14　[SP+byte]の例



①SPを指定して，スタック領域を対象とします。
②のbyteで，現在のスタックポインタが指すアドレス（スタックトップ）からのオフセットを指定することで,ターゲット・メモリ（SP+byte）を指し示します。

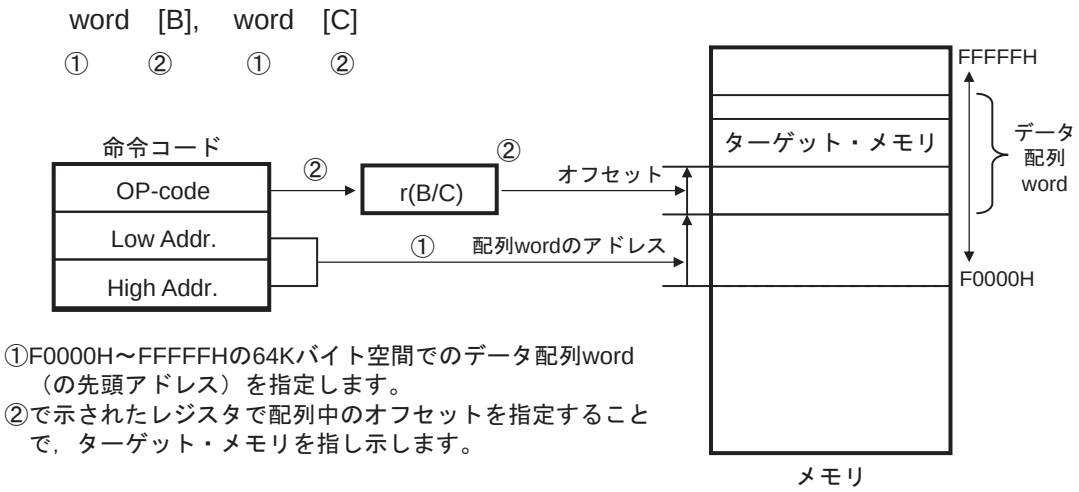
注意　[HL+byte], [DE+byte], word[B], word[C], word[BC]では，足した値がFFFFHを超える使い方は禁止します。
ES:[HL+byte], ES:[DE+byte], ES:word[B], ES:word[C], ES:word[BC]では，足した値がFFFFFHを超える使い方は禁止します。
[SP+byte]においては，SPの値はRAM空間にあること，かつSP+byteの足した値がRAM空間のFFEDFH以下にしてください。

図4－15　[HL+byte], [DE+byte]の例



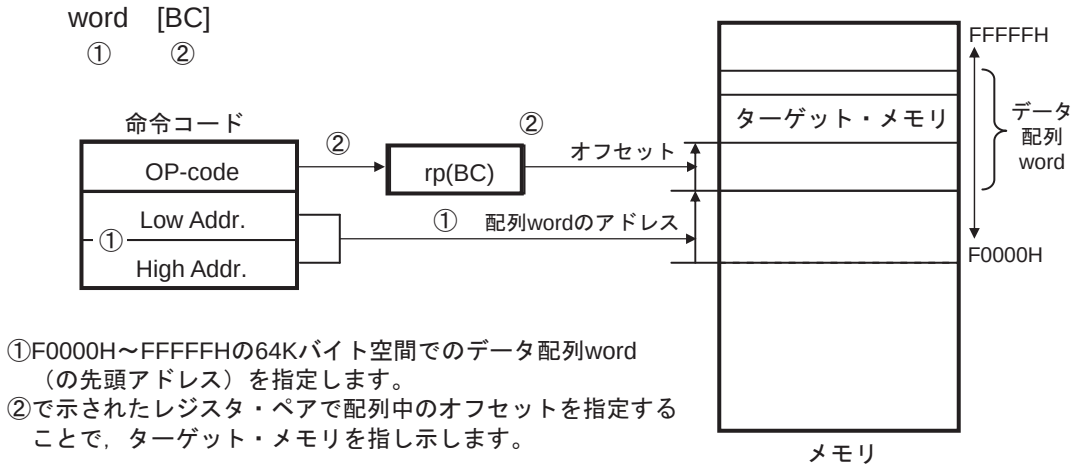
①のレジスタ・ペアでF0000H～FFFFFHの64Kバイト空間内のデータ配列（の先頭アドレス）を指定します。
②で示したbyteで配列中のオフセットを指定することで，ターゲット・メモリを指し示します。

図4－16　word[B], word[C]の例



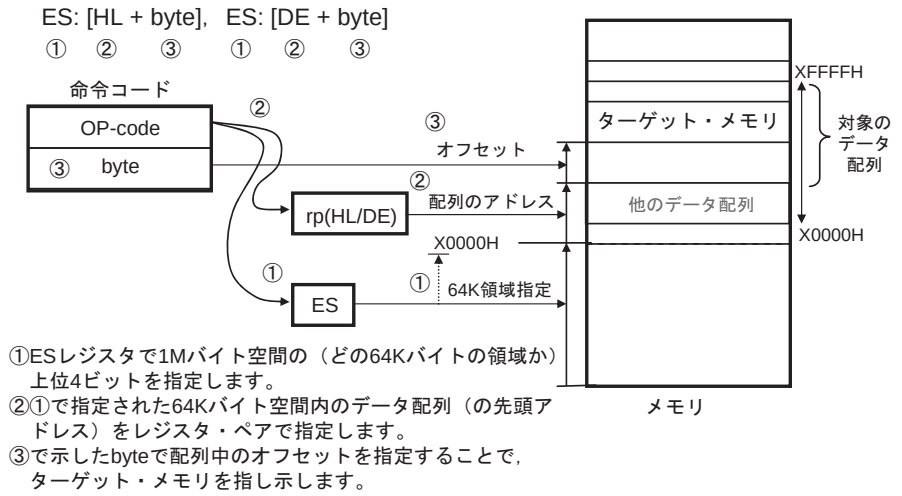
①F0000H～FFFFFHの64Kバイト空間でのデータ配列word（の先頭アドレス）を指定します。
②で示されたレジスタで配列中のオフセットを指定することで，ターゲット・メモリを指し示します。

図4－17　word[BC]の例



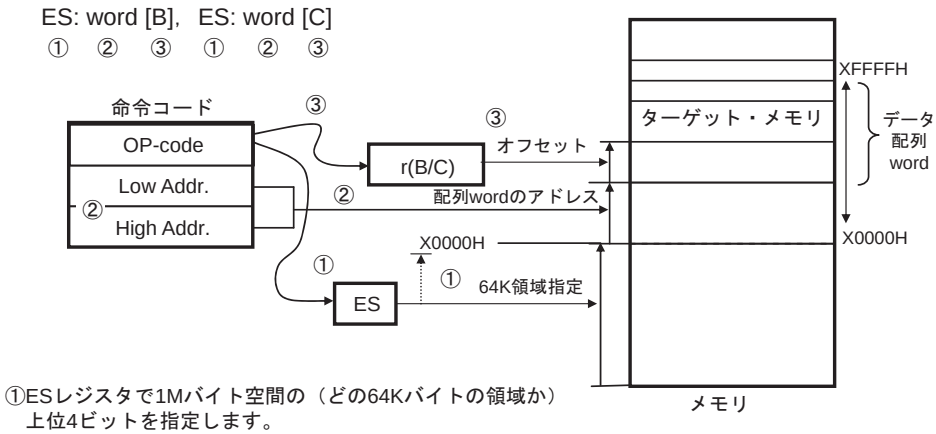
①F0000H～FFFFFHの64Kバイト空間でのデータ配列word（の先頭アドレス）を指定します。
②で示されたレジスタ・ペアで配列中のオフセットを指定することで，ターゲット・メモリを指し示します。

図4－18　ES:[HL+byte], ES:[DE+byte]の例



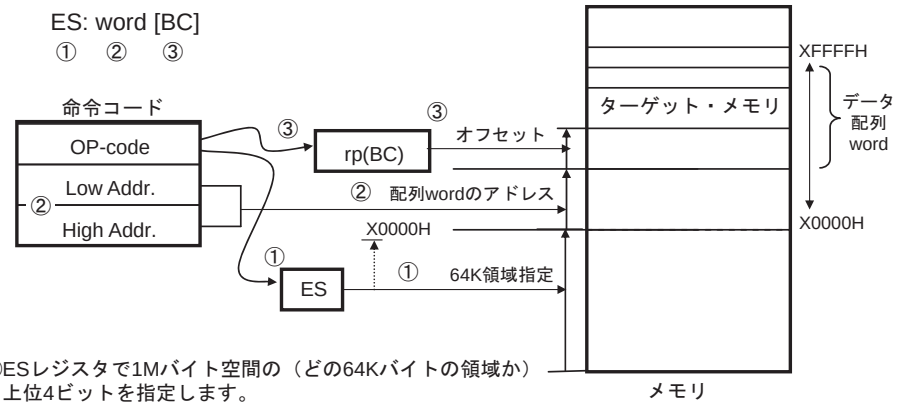
①ESレジスタで1Mバイト空間の（どの64Kバイトの領域か）上位4ビットを指定します。
②①で指定された64Kバイト空間内のデータ配列（の先頭アドレス）をレジスタ・ペアで指定します。
③で示したbyteで配列中のオフセットを指定することで，ターゲット・メモリを指し示します。

図4－19　ES:word[B], ES:word[C]の例



①ESレジスタで1Mバイト空間の（どの64Kバイトの領域か）上位4ビットを指定します。
②①で指定された64Kバイト空間でのデータ配列word（の先頭アドレス）を指定します。
③で示したレジスタで配列中のオフセットを指定することで，ターゲット・メモリを指し示します。

図4－20　ES:word[BC]の例



①ESレジスタで1Mバイト空間の（どの64Kバイトの領域か）上位4ビットを指定します。
②①で指定された64Kバイト空間でのデータ配列word（の先頭アドレス）を指定します。
③で示したレジスタ・ペアで配列中のオフセットを指定することで，ターゲット・メモリを指し示します。

4.2.8　ベースト・インデクスト・アドレッシング

【機　　能】

命令語で指定されるレジスタ・ペアの内容をベース・アドレスとし，同様に命令語で指定されるBレジスタまたはCレジスタの内容をオフセット・アドレスとしてベース・アドレスに加算した結果で，対象となるアドレスを指定するアドレッシングです。

【オペランド形式】

表現形式	記述方法
−	[HL＋B], [HL＋C] （F0000H-FFFFFH空間のみ指定可能）
−	ES:[HL＋B], ES:[HL＋C] （ESレジスタにて上位4ビット・アドレス指定）

図4－21　[HL+B], [HL+C]の例

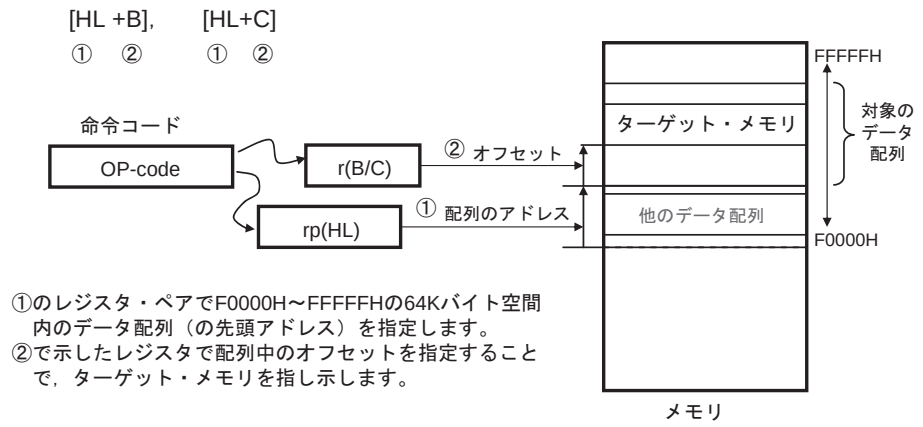
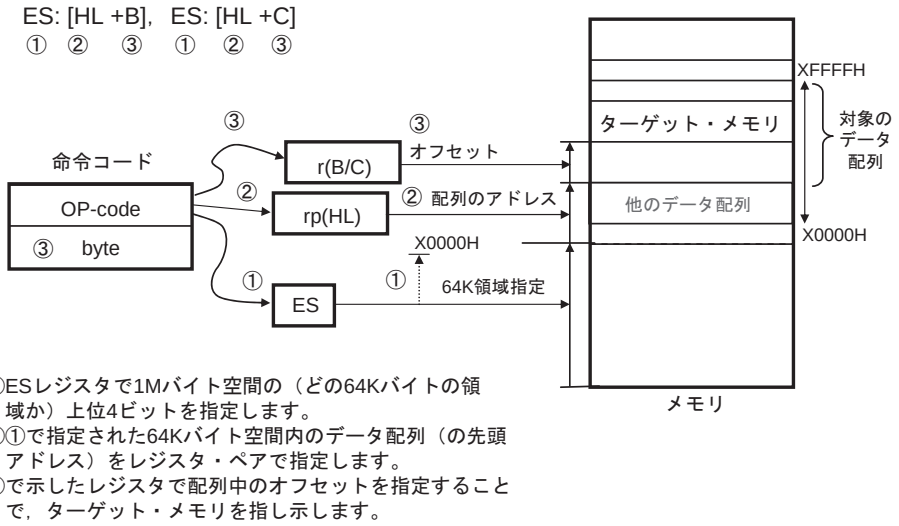


図4－22　ES:[HL+B], ES:[HL+C]の例



注意　[HL+B], [HL+C]では，足した値がFFFFFHを超える使い方は禁止します。

ES:[HL+B], ES:[HL+C]では，足した値がFFFFFHを超える使い方は禁止します。

4.2.9　スタック・アドレッシング

【機　　能】

スタック・ポインタ（SP）の内容によりスタック領域を間接的に指定するアドレッシングです。PUSH，POP，サブルーチン・コール，リターン命令の実行時，および割り込み要求発生によるレジスタの退避／復帰時に自動的に用いられます。

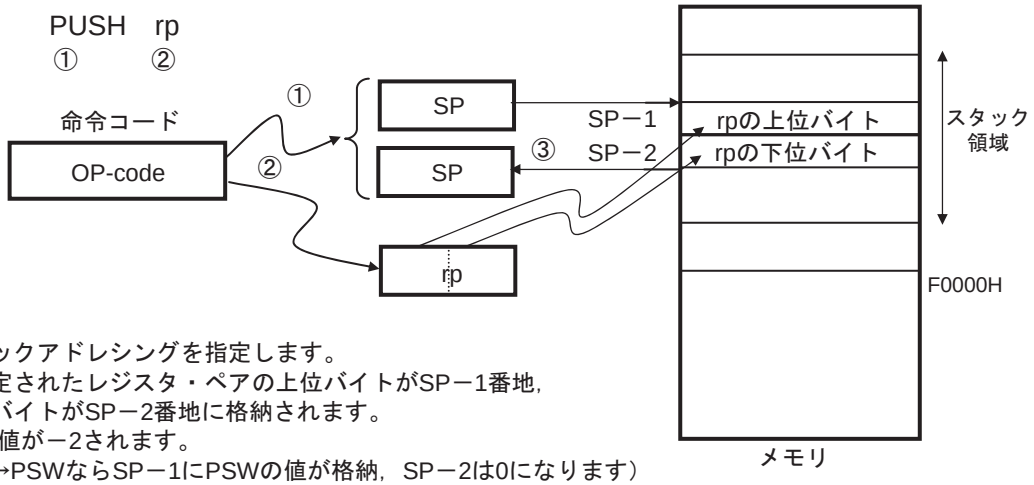
スタック・アクセスは内部RAMのみに用いられます。

【オペランド形式】

表現形式	記述方法
－	PUSH PSW AX/BC/DE/HL POP PSW AX/BC/DE/HL CALL/CALLT RET BRK RETB (割り込み要求発生) RETI

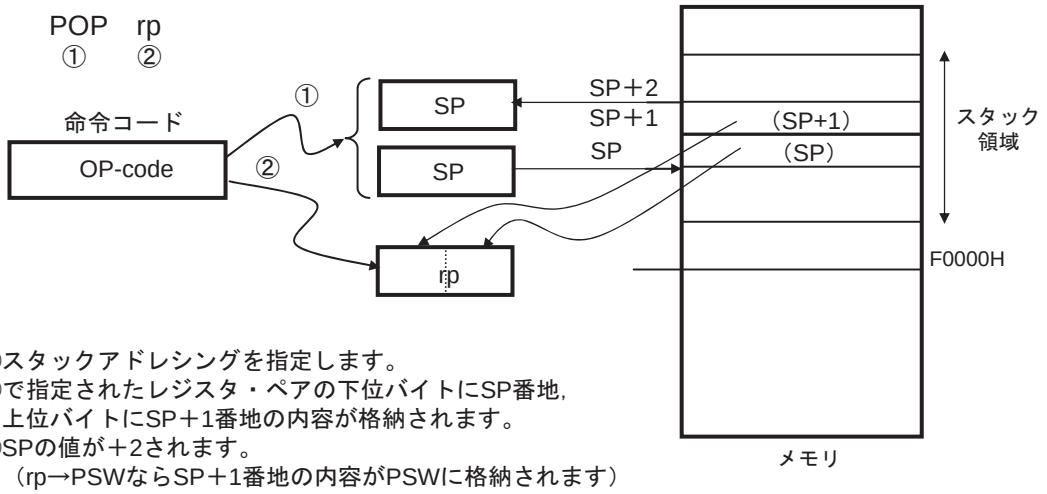
各スタック動作によって退避／復帰されるデータは図4－23～図4－28のようになります。

図4－23　PUSH rpの例



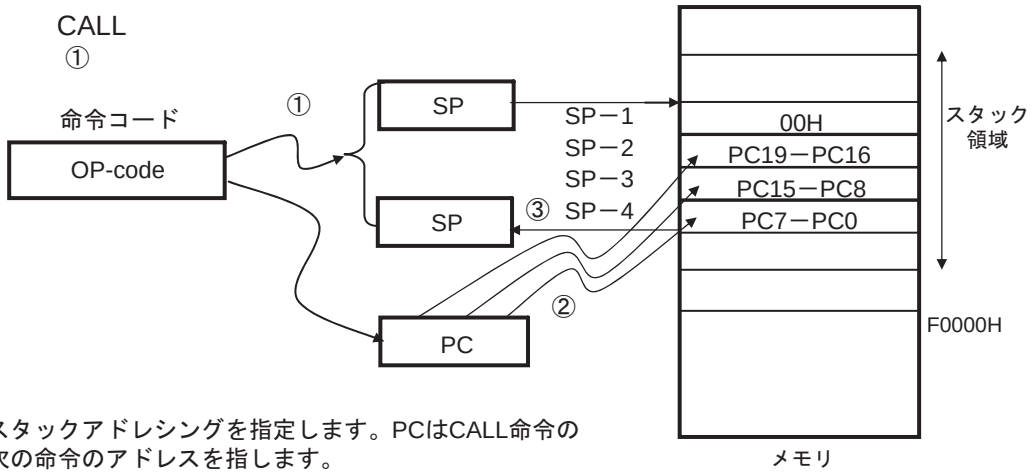
①スタックアドレシングを指定します。
②で指定されたレジスタ・ペアの上位バイトがSP－1番地，
　下位バイトがSP－2番地に格納されます。
③SPの値が－2されます。
　（rp→PSWならSP－1にPSWの値が格納，SP－2は0になります）

図4－24　POPの例



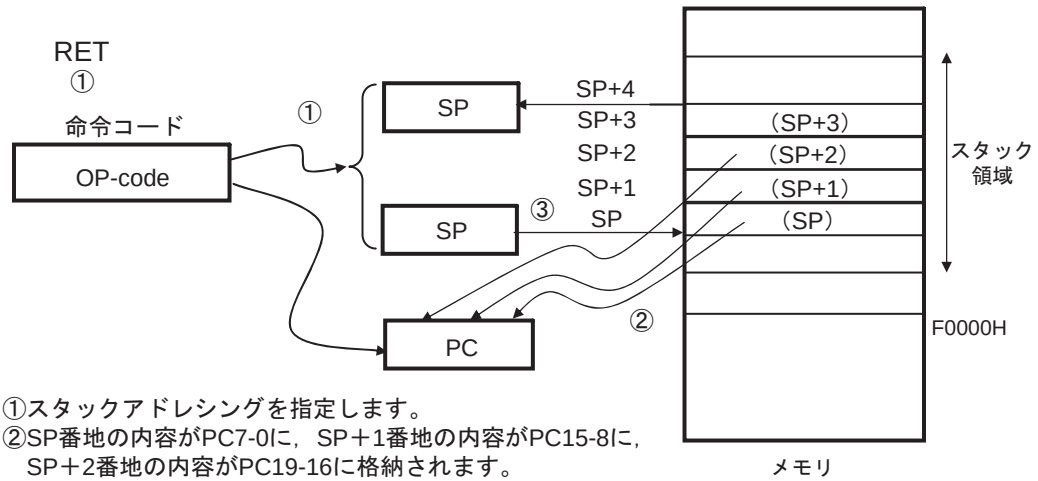
①スタックアドレシングを指定します。
②で指定されたレジスタ・ペアの下位バイトにSP番地，
　上位バイトにSP＋1番地の内容が格納されます。
③SPの値が＋2されます。
　（rp→PSWならSP＋1番地の内容がPSWに格納されます）

図4－25　CALL, CALLTの例



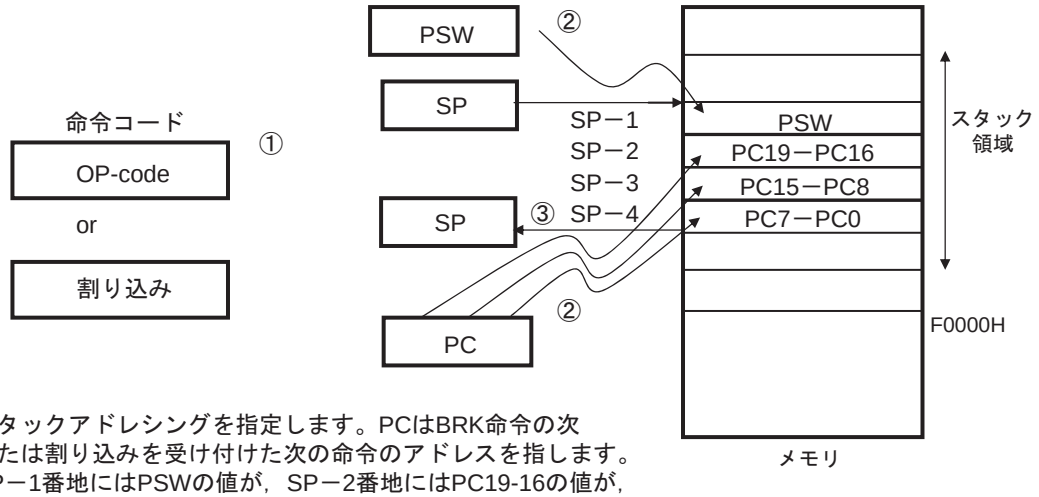
①スタックアドレシングを指定します。PCはCALL命令の
　次の命令のアドレスを指します。
②SP－1番地には00Hが，SP－2番地にはPC19-16の値が，
　SP－3番地にはPC15-8の値が，SP－4番地にはPC7-0の
　値が格納されます。
③SPの値が－4されます。

図4－26　RETの例



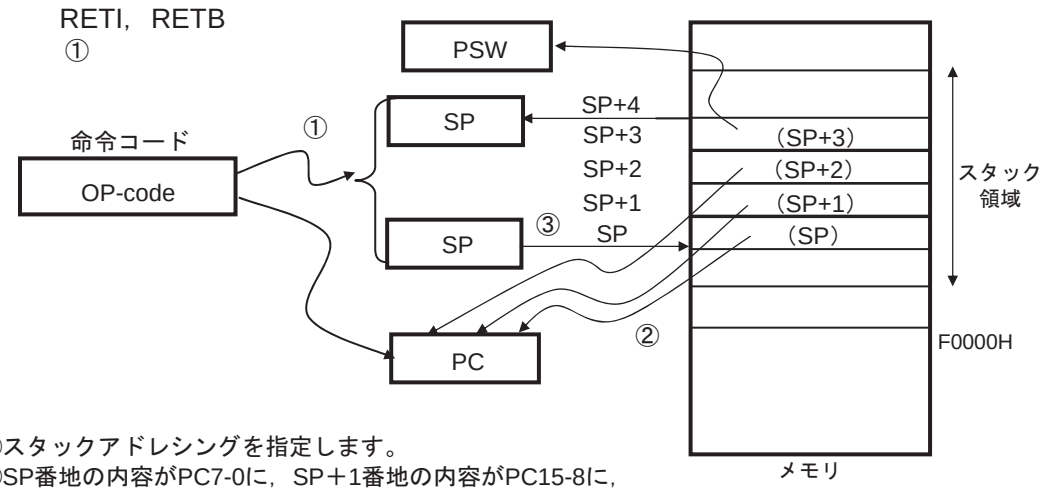
①スタックアドレシングを指定します。
②SP番地の内容がPC7-0に，SP＋1番地の内容がPC15-8に，
　SP＋2番地の内容がPC19-16に格納されます。
③SPの値が＋4されます。

図4－27　割り込み, BRKの例



①スタックアドレシングを指定します。PCはBRK命令の次
　または割り込みを受け付けた次の命令のアドレスを指します。
②SP－1番地にはPSWの値が，SP－2番地にはPC19-16の値が，
　SP－3番地にはPC15-8の値が，SP－4番地にはPC7-0の値が
　格納されます。
③SPの値が－4されます。

図4－28　RETI, RETBの例



①スタックアドレシングを指定します。
②SP番地の内容がPC7-0に，SP＋1番地の内容がPC15-8に，
　SP＋2番地の内容がPC19-16に，SP＋3番地の内容がPSW
　格納されます。
③SPの値が＋4されます。

第5章　命令セット

この章では，RL78マイクロコントローラの命令セットを一覧表にして示します。

RL78マイクロコントローラ製品の命令は共通です。ただし，以下のCPU制御命令はRL78-S1コアにはありません。

・SEL RBn（レジスタ・バンクの選択）

また，以下の乗除積和算命令は拡張命令です。RL78-S3コアのみあります。

・MULHU（符号なし16ビット乗算）

・MULH（符号付き16ビット乗算）

・DIVHU（符号なし16ビット除算）

・DIVWU（符号なし32ビット除算）

・MACHU（符号なし積和算（16ビット×16ビット）＋32ビット）

・MACH（符号付き積和算（16ビット×16ビット）＋32ビット）

なお，以下の命令のクロック数は，RL78-S1コアと他のCPUコアで異なります。詳細は，「5.5　オペレーション一覧」を参照してください。

・16ビット・データ転送（MOVW，XCHW，ONEW，CLRW）

・16ビット演算（ADDW，SUBW，CMPW）

・乗算（MULU）

・16ビット増減（INCW，DECW）

・16ビット・シフト（SHRW，SHLW，SARW）

・16ビット・ローテート（ROLWC）

・コール・リターン（CALL，CALLT，BRK，RET，RETI，RETB）

・スタック操作（PUSH，POP，MOVW，ADDW，SUBW）

5.1　オペランドの表現形式と記述方法

各命令のオペランド欄には，その命令のオペランド表現形式に対する記述方法に従ってオペランドを記述しています（詳細は，アセンブラ仕様によります）。記述方法の中で複数個あるものは，それらの要素の1つを選択します。大文字で書かれた英字および#，!，!!，$，$!，[ ]，ES:の記号はキーワードであり，そのまま記述します。記号の説明は，次のとおりです。

- ・#　　：イミーディエト・データ指定
- ・!　　：16ビット絶対アドレス指定
- ・!!　　：20ビット絶対アドレス指定
- ・$　　：8ビット相対アドレス指定
- ・$!　　：16ビット相対アドレス指定
- ・[ ]　　：間接アドレス指定
- ・ES:　　：拡張アドレス指定

イミーディエト・データのときは，適当な数値またはラベルを記述します。ラベルで記述する際も#，!，!!，$，$!，[ ]，ES:記号は必ず記述してください。

また，オペランドのレジスタの記述形式r，rpには，機能名称（X，A，Cなど），絶対名称（表5－1の中のカッコ内の名称，R0，R1，R2など）のいずれの形式でも記述可能です。

表5－1　オペランドの表現形式と記述方法

表現形式	記述方法
r	X(R0)，A(R1)，C(R2)，B(R3)，E(R4)，D(R5)，L(R6)，H(R7)
rp	AX(RP0)，BC(RP1)，DE(RP2)，HL(RP3)
sfr	特殊機能レジスタ略号（SFR略号）FFF00H-FFFFFH
sfrp	特殊機能レジスタ略号（16ビット操作可能なSFR略号。偶数アドレスのみ注1）FFF00H-FFFFFH
saddr	FFE20H-FFF1FH　イミーディエト・データまたはラベル
saddrp	FFE20H-FFF1FH　イミーディエト・データまたはラベル（偶数アドレスのみ注1）
addr20	00000H-FFFFFH　イミーディエト・データまたはラベル
addr16	0000H-FFFFH　イミーディエト・データまたはラベル （16ビット・データ時は偶数アドレスのみ注1）
addr5	0080H-00BFH　イミーディエト・データまたはラベル（偶数アドレスのみ）
word	16ビット・イミーディエト・データまたはラベル
byte	8ビット・イミーディエト・データまたはラベル
bit	3ビット・イミーディエト・データまたはラベル
RBn注2	RB0-RB3

注1.　奇数アドレスを指定した場合はビット0が“0”になります。

2.　RL78-S1コアにはありません。

備考　特殊機能レジスタは，オペランドsfrに略号で記述することができます。

拡張特殊機能レジスタは，オペランド!addr16に略号で記述することができます。

5.2　オペレーション欄の説明

各命令のオペレーション欄には，その命令実行時の動作を次の記号を用いて表します。

表5－2　オペレーション欄の記号

記　　号	機　　能
A	Aレジスタ：8ビット・アキュムレータ
X	Xレジスタ
B	Bレジスタ
C	Cレジスタ
D	Dレジスタ
E	Eレジスタ
H	Hレジスタ
L	Lレジスタ
ES	ESレジスタ
CS	CSレジスタ
AX	AXレジスタ・ペア：16ビット・アキュムレータ
BC	BCレジスタ・ペア
DE	DEレジスタ・ペア
HL	HLレジスタ・ペア
PC	プログラム・カウンタ
SP	スタック・ポインタ
PSW	プログラム・ステータス・ワード
CY	キャリー・フラグ
AC	補助キャリー・フラグ
Z	ゼロ・フラグ
RBS[注]	レジスタ・バンク選択フラグ
IE	割り込み要求許可フラグ
()	() 内のアドレスまたはレジスタの内容で示されるメモリの内容
X_H，X_L X_S，X_H，X_L	16ビット・レジスタの場合はX_H =上位8ビット，X_L =下位8ビット 20ビット・レジスタの場合はX_S(ビット19-16)，X_H(ビット15-8)，X_L(ビット7-0)
∧	論理積（AND）
∨	論理和（OR）
∀	排他的論理和（exclusive OR）
―	反転データ
addr5	16ビット・イミーディエト・データ（0080H-00BFHの偶数アドレスのみ）
addr16	16ビット・イミーディエト・データ
addr20	20ビット・イミーディエト・データ
jdisp8	符号付き8ビット・データ（ディスプレースメント値）
jdisp16	符号付き16ビット・データ（ディスプレースメント値）

注　RL78-S1コアにはありません。

5.3　フラグ動作欄の説明

各命令のフラグ欄には，その命令実行時のフラグの変化を下記の記号を用いて表します。

表5－3　フラグ欄の記号

記号	フラグ変化
（ブランク）	変化なし
0	0にクリアされる
1	1にセットされる
×	結果に従ってセット／リセットされる
R	以前に退避した値がリストアされる

5.4　PREFIX命令

ES:で示される命令は，PREFIX命令コードを頭に付けることで，アクセスできるデータ領域をF0000H-FFFFFHの64 Kバイト空間から，ESレジスタの値を付加した00000H-FFFFFHの1 Mバイト空間に拡張します。PREFIX命令コードは対象となる命令の先頭に付けることで，PREFIX命令コード直後の1命令だけをESレジスタの値を付加したアドレスとして実行します。

なお，PREFIX命令コードと直後の1命令の間に割り込みやDMA転送を受け付けることはありません。

表5－4　PREFIX命令コードの使用例

命令	命令コード				
	1	2	3	4	5
MOV !addr16, #byte	CFH	!addr16		#byte	－
MOV ES:!addr16, #byte	11H	CFH	!addr16		#byte
MOV A, [HL]	8BH	－	－	－	－
MOV A, ES:[HL]	11H	8BH	－	－	－

注意　ESレジスタの値は，PREFIX命令を実行するまでにMOV ES, Aなどで事前に設定しておいてください。

5.5　オペレーション一覧

5.5.1　RL78-S1コアのオペレーション一覧

表5−5　RL78-S1コアのオペレーション一覧（1/17）

命令群	ニモニック	オペランド	バイト	クロック		オペレーション	フラグ
				注1	注2		Z AC CY
8ビット・データ転送	MOV	r, #byte	2	1	−	r ← byte	
		PSW, #byte	3	3	−	PSW ← byte	× × ×
		CS, #byte	3	1	−	CS ← byte	
		ES, #byte	2	1	−	ES ← byte	
		!addr16, #byte	4	1	−	(addr16) ← byte	
		ES:!addr16, #byte	5	2	−	(ES, addr16) ← byte	
		saddr, #byte	3	1	−	(saddr) ← byte	
		sfr, #byte	3	1	−	sfr ← byte	
		[DE+byte], #byte	3	1	−	(DE＋byte) ← byte	
		ES:[DE+byte],#byte	4	2	−	((ES, DE)＋byte) ← byte	
		[HL+byte], #byte	3	1	−	(HL＋byte) ← byte	
		ES:[HL+byte],#byte	4	2	−	((ES, HL)＋byte) ← byte	
		[SP+byte], #byte	3	1	−	(SP＋byte) ← byte	
		word[B], #byte	4	1	−	(B＋word) ← byte	
		ES:word[B], #byte	5	2	−	((ES, B)＋word) ← byte	
		word[C], #byte	4	1	−	(C＋word) ← byte	
		ES:word[C], #byte	5	2	−	((ES, C)＋word) ← byte	
		word[BC], #byte	4	1	−	(BC＋word) ← byte	
		ES:word[BC], #byte	5	2	−	((ES, BC)＋word) ← byte	
		A, r[注3]	1	1	−	A ← r	
		r, A[注3]	1	1	−	r ← A	
		A, PSW	2	1	−	A ← PSW	
		PSW, A	2	3	−	PSW ← A	× × ×
		A, CS	2	1	−	A ← CS	
		CS, A	2	1	−	CS ← A	
		A, ES	2	1	−	A ← ES	
		ES, A	2	1	−	ES ← A	
		A, !addr16	3	1	4	A ← (addr16)	
		A, ES:!addr16	4	2	5	A ← (ES, addr16)	
		!addr16, A	3	1	−	(addr16) ← A	
		ES:!addr16, A	4	2	−	(ES, addr16) ← A	
		A, saddr	2	1	−	A ← (saddr)	
		saddr, A	2	1	−	(saddr) ← A	

注1.　内部RAM領域，SFR領域および拡張SFR領域をアクセスしたとき，またはデータ・アクセスをしないときのCPUクロック（f_{CLK}）数。

2.　コード・フラッシュ・メモリ領域をアクセスしたときのCPUクロック（f_{CLK}）数。

3.　r = Aを除く。

備考　クロック数は内部ROM（フラッシュ・メモリ）領域にプログラムがある場合です。内部RAM領域から命令フェッチする場合，最大4倍＋6クロックになります。

表5−5　RL78-S1コアのオペレーション一覧（2/17）

命令群	ニモニック	オペランド	バイト	クロック		オペレーション	フラグ
				注1	注2		Z AC CY
8ビット・データ転送	MOV	A, sfr	2	1	−	A ← sfr	
		sfr, A	2	1	−	sfr ← A	
		A, [DE]	1	1	4	A ← (DE)	
		[DE], A	1	1	−	(DE) ← A	
		A, ES:[DE]	2	2	5	A ← (ES, DE)	
		ES:[DE], A	2	2	−	(ES, DE) ← A	
		A, [HL]	1	1	4	A ← (HL)	
		[HL], A	1	1	−	(HL) ← A	
		A, ES:[HL]	2	2	5	A ← (ES, HL)	
		ES:[HL], A	2	2	−	(ES, HL) ← A	
		A, [DE+byte]	2	1	4	A ← (DE＋byte)	
		[DE+byte], A	2	1	−	(DE＋byte) ← A	
		A, ES:[DE+byte]	3	2	5	A ← ((ES, DE)＋byte)	
		ES:[DE+byte], A	3	2	−	((ES, DE)＋byte) ← A	
		A, [HL+byte]	2	1	4	A ← (HL＋byte)	
		[HL+byte], A	2	1	−	(HL＋byte) ← A	
		A, ES:[HL+byte]	3	2	5	A ← ((ES, HL)＋byte)	
		ES:[HL+byte], A	3	2	−	((ES, HL)＋byte) ← A	
		A, [SP+byte]	2	1	−	A ← (SP＋byte)	
		[SP+byte], A	2	1	−	(SP＋byte) ← A	
		A, word[B]	3	1	4	A ← (B＋word)	
		word[B], A	3	1	−	(B＋word) ← A	
		A, ES:word[B]	4	2	5	A ← ((ES, B)＋word)	
		ES:word[B], A	4	2	−	((ES, B)＋word) ← A	
		A, word[C]	3	1	4	A ← (C＋word)	
		word[C], A	3	1	−	(C＋word) ← A	
		A, ES:word[C]	4	2	5	A ← ((ES, C)＋word)	
		ES:word[C], A	4	2	−	((ES, C)＋word) ← A	
		A, word[BC]	3	1	4	A ← (BC＋word)	
		word[BC], A	3	1	−	(BC＋word) ← A	
		A, ES:word[BC]	4	2	5	A ← ((ES, BC)＋word)	
		ES:word[BC], A	4	2	−	((ES, BC)＋word) ← A	

注1.　内部RAM領域，SFR領域および拡張SFR領域をアクセスしたとき，またはデータ・アクセスをしないときのCPUクロック（f_{CLK}）数。

2.　コード・フラッシュ・メモリ領域をアクセスしたときのCPUクロック（f_{CLK}）数。

備考　クロック数は内部ROM（フラッシュ・メモリ）領域にプログラムがある場合です。内部RAM領域から命令フェッチする場合，最大4倍＋6クロックになります。

表5－5　RL78-S1コアのオペレーション一覧（3/17）

命令群	ニモニック	オペランド	バイト	クロック		オペレーション	フラグ
				注1	注2		Z AC CY
8ビット・データ転送	MOV	A, [HL+B]	2	1	4	A ← (HL＋B)	
		[HL+B], A	2	1	－	(HL＋B) ← A	
		A, ES:[HL+B]	3	2	5	A ← ((ES, HL)＋B)	
		ES:[HL+B], A	3	2	－	((ES, HL)＋B) ← A	
		A, [HL+C]	2	1	4	A ← (HL＋C)	
		[HL+C], A	2	1	－	(HL＋C) ← A	
		A, ES:[HL+C]	3	2	5	A ← ((ES, HL)＋C)	
		ES:[HL+C], A	3	2	－	((ES, HL)＋C) ← A	
		X, !addr16	3	1	4	X ← (addr16)	
		X, ES:!addr16	4	2	5	X ← (ES, addr16)	
		X, saddr	2	1	－	X ← (saddr)	
		B, !addr16	3	1	4	B ← (addr16)	
		B, ES:!addr16	4	2	5	B ← (ES, addr16)	
		B, saddr	2	1	－	B ← (saddr)	
		C, !addr16	3	1	4	C ← (addr16)	
		C, ES:!addr16	4	2	5	C ← (ES, addr16)	
		C, saddr	2	1	－	C ← (saddr)	
		ES, saddr	3	1	－	ES ← (saddr)	
	XCH	A, r[注3]	1 (r＝X) 2 (r＝X以外)	1	－	A ⟷ r	
		A, !addr16	4	2	－	A ⟷ (addr16)	
		A, ES:!addr16	5	3	－	A ⟷ (ES, addr16)	
		A, saddr	3	2	－	A ⟷ (saddr)	
		A, sfr	3	2	－	A ⟷ sfr	
		A, [DE]	2	2	－	A ⟷ (DE)	
		A, ES:[DE]	3	3	－	A ⟷ (ES, DE)	
		A, [HL]	2	2	－	A ⟷ (HL)	
		A, ES:[HL]	3	3	－	A ⟷ (ES, HL)	
		A, [DE+byte]	3	2	－	A ⟷ (DE＋byte)	
		A, ES:[DE+byte]	4	3	－	A ⟷ ((ES, DE)＋byte)	
		A, [HL+byte]	3	2	－	A ⟷ (HL＋byte)	
		A, ES:[HL+byte]	4	3	－	A ⟷ ((ES, HL)＋byte)	

注1.　内部RAM領域，SFR領域および拡張SFR領域をアクセスしたとき，またはデータ・アクセスをしないときのCPUクロック（f_{CLK}）数。

2.　コード・フラッシュ・メモリ領域をアクセスしたときのCPUクロック（f_{CLK}）数。

3.　r = Aを除く。

備考　クロック数は内部ROM（フラッシュ・メモリ）領域にプログラムがある場合です。内部RAM領域から命令フェッチする場合，最大4倍＋6クロックになります。

表5－5　RL78-S1コアのオペレーション一覧（4/17）

命令群	ニモニック	オペランド	バイト	クロック		オペレーション	フラグ
				注1	注2		Z AC CY
8ビット・データ転送	XCH	A, [HL+B]	2	2	－	A ⟷ (HL＋B)	
		A, ES:[HL+B]	3	3	－	A ⟷ ((ES, HL)＋B)	
		A, [HL+C]	2	2	－	A ⟷ (HL＋C)	
		A, ES:[HL+C]	3	3	－	A ⟷ ((ES, HL)＋C)	
	ONEB	A	1	1	－	A ← 01H	
		X	1	1	－	X ← 01H	
		B	1	1	－	B ← 01H	
		C	1	1	－	C ← 01H	
		!addr16	3	1	－	(addr16) ← 01H	
		ES:!addr16	4	2	－	(ES, addr16) ← 01H	
		saddr	2	1	－	(saddr) ← 01H	
	CLRB	A	1	1	－	A ← 00H	
		X	1	1	－	X ← 00H	
		B	1	1	－	B ← 00H	
		C	1	1	－	C ← 00H	
		!addr16	3	1	－	(addr16) ← 00H	
		ES:!addr16	4	2	－	(ES,addr16) ← 00H	
		saddr	2	1	－	(saddr) ← 00H	
	MOVS	[HL+byte], X	3	1	－	(HL＋byte) ← X	× ×
		ES:[HL+byte], X	4	2	－	(ES, HL＋byte) ← X	× ×
16ビット・データ転送	MOVW	rp, #word	3	2	－	rp ← word	
		saddrp, #word	4	2	－	(saddrp) ← word	
		sfrp, #word	4	2	－	sfrp ← word	
		AX, rp[注3]	1	2	－	AX ← rp	
		rp, AX[注3]	1	2	－	rp ← AX	
		AX, !addr16	3	2	5	AX ←（addr16）	
		!addr16, AX	3	2	－	（addr16）← AX	
		AX, ES:!addr16	4	3	6	AX ← (ES, addr16)	
		ES:!addr16, AX	4	3	－	(ES, addr16) ← AX	
		AX, saddrp	2	2	－	AX ← (saddrp)	
		saddrp, AX	2	2	－	(saddrp) ← AX	
		AX, sfrp	2	2	－	AX ← sfrp	
		sfrp, AX	2	2	－	sfrp ← AX	

注1.　内部RAM領域，SFR領域および拡張SFR領域をアクセスしたとき，またはデータ・アクセスをしないときのCPUクロック（f_{CLK}）数。

2.　コード・フラッシュ・メモリ領域をアクセスしたときのCPUクロック（f_{CLK}）数。

3.　rp = AXを除く。

備考　クロック数は内部ROM（フラッシュ・メモリ）領域にプログラムがある場合です。内部RAM領域から命令フェッチする場合，最大4倍＋6クロックになります。

表5－5　RL78-S1コアのオペレーション一覧（5/17）

命令群	ニモニック	オペランド	バイト	クロック		オペレーション	フラグ
				注1	注2		Z AC CY
16ビット・データ転送	MOVW	AX, [DE]	1	2	5	AX ← (DE)	
		[DE], AX	1	2	－	(DE) ← AX	
		AX, ES:[DE]	2	3	6	AX ← (ES, DE)	
		ES:[DE], AX	2	3	－	(ES, DE) ← AX	
		AX, [HL]	1	2	5	AX ← (HL)	
		[HL], AX	1	2	－	(HL) ← AX	
		AX, ES:[HL]	2	3	6	AX ← (ES, HL)	
		ES:[HL], AX	2	3	－	(ES, HL) ← AX	
		AX, [DE+byte]	2	2	5	AX ← (DE＋byte)	
		[DE+byte], AX	2	2	－	(DE＋byte) ← AX	
		AX, ES:[DE+byte]	3	3	6	AX ← ((ES, DE)＋byte)	
		ES:[DE+byte], AX	3	3	－	((ES, DE)＋byte) ← AX	
		AX, [HL+byte]	2	2	5	AX ← (HL＋byte)	
		[HL+byte], AX	2	2	－	(HL＋byte) ← AX	
		AX, ES:[HL+byte]	3	3	6	AX ← ((ES, HL)＋byte)	
		ES:[HL+byte], AX	3	3	－	((ES, HL)＋byte) ← AX	
		AX, [SP+byte]	2	2	－	AX ← (SP＋byte)	
		[SP+byte], AX	2	2	－	(SP＋byte) ← AX	
		AX, word[B]	3	2	5	AX ← (B＋word)	
		word[B], AX	3	2	－	(B＋word) ← AX	
		AX, ES:word[B]	4	3	6	AX ← ((ES, B)＋word)	
		ES:word[B], AX	4	3	－	((ES, B)＋word) ← AX	
		AX, word[C]	3	2	5	AX ← (C＋word)	
		word[C], AX	3	2	－	(C＋word) ← AX	
		AX, ES:word[C]	4	3	6	AX ← ((ES, C)＋word)	
		ES:word[C], AX	4	3	－	((ES, C)＋word) ← AX	
		AX, word[BC]	3	2	5	AX ← (BC＋word)	
		word[BC], AX	3	2	－	(BC＋word) ← AX	
		AX, ES:word[BC]	4	3	6	AX ← ((ES, BC)＋word)	
		ES:word[BC], AX	4	3	－	((ES, BC)＋word) ← AX	

注1.　内部RAM領域，SFR領域および拡張SFR領域をアクセスしたとき，またはデータ・アクセスをしないときのCPUクロック（f_{CLK}）数。

2.　コード・フラッシュ・メモリ領域をアクセスしたときのCPUクロック（f_{CLK}）数。

備考　クロック数は内部ROM（フラッシュ・メモリ）領域にプログラムがある場合です。内部RAM領域から命令フェッチする場合，最大4倍＋6クロックになります。

表5－5　RL78-S1コアのオペレーション一覧（6/17）

命令群	ニモニック	オペランド	バイト	クロック		オペレーション	フラグ
				注1	注2		Z AC CY
16ビット・データ転送	MOVW	BC, !addr16	3	2	5	BC ← (addr16)	
		BC, ES:!addr16	4	3	6	BC ← (ES, addr16)	
		DE, !addr16	3	2	5	DE ← (addr16)	
		DE, ES:!addr16	4	3	6	DE ← (ES, addr16)	
		HL, !addr16	3	2	5	HL ← (addr16)	
		HL, ES:!addr16	4	3	6	HL ← (ES, addr16)	
		BC, saddrp	2	2	－	BC ← (saddrp)	
		DE, saddrp	2	2	－	DE ← (saddrp)	
		HL, saddrp	2	2	－	HL ← (saddrp)	
	XCHW	AX, rp[注3]	1	2	－	AX ⟷ rp	
	ONEW	AX	1	2	－	AX ← 0001H	
		BC	1	2	－	BC ← 0001H	
	CLRW	AX	1	2	－	AX ← 0000H	
		BC	1	2	－	BC ← 0000H	
8ビット演算	ADD	A, #byte	2	1	－	A, CY ← A＋byte	× × ×
		saddr, #byte	3	2	－	(saddr), CY ← (saddr)＋byte	× × ×
		A, r[注4]	2	1	－	A, CY ← A＋r	× × ×
		r, A	2	1	－	r, CY ← r＋A	× × ×
		A, !addr16	3	1	4	A, CY ← A＋(addr16)	× × ×
		A, ES:!addr16	4	2	5	A, CY ← A＋(ES, addr16)	× × ×
		A, saddr	2	1	－	A, CY ← A＋(saddr)	× × ×
		A, ［HL］	1	1	4	A, CY ← A＋(HL)	× × ×
		A, ES:[HL]	2	2	5	A,CY ← A＋(ES, HL)	× × ×
		A, ［HL+byte］	2	1	4	A, CY ← A＋(HL＋byte)	× × ×
		A, ES:[HL+byte]	3	2	5	A,CY ← A＋((ES, HL)＋byte)	× × ×
		A, ［HL+B］	2	1	4	A, CY ← A＋(HL＋B)	× × ×
		A, ES:[HL+B]	3	2	5	A,CY ← A＋((ES, HL)＋B)	× × ×
		A, ［HL+C］	2	1	4	A, CY ← A＋(HL＋C)	× × ×
		A, ES:[HL+C]	3	2	5	A,CY ← A＋((ES, HL)＋C)	× × ×

注1.　内部RAM領域，SFR領域および拡張SFR領域をアクセスしたとき，またはデータ・アクセスをしないときのCPUクロック（f_{CLK}）数。

2.　コード・フラッシュ・メモリ領域をアクセスしたときのCPUクロック（f_{CLK}）数。

3.　rp = AXを除く。

4.　r = Aを除く。

備考　クロック数は内部ROM（フラッシュ・メモリ）領域にプログラムがある場合です。内部RAM領域から命令フェッチする場合，最大4倍＋6クロックになります。

表5－5　RL78-S1コアのオペレーション一覧（7/17）

命令群	ニモニック	オペランド	バイト	クロック		オペレーション	フラグ		
				注1	注2		Z	AC	CY
8ビット演算	ADDC	A, #byte	2	1	－	A, CY ← A＋byte＋CY	×	×	×
		saddr, #byte	3	2	－	(saddr), CY ← (saddr)＋byte＋CY	×	×	×
		A, r[注3]	2	1	－	A, CY ← A＋r＋CY	×	×	×
		r, A	2	1	－	r, CY ← r＋A＋CY	×	×	×
		A, !addr16	3	1	4	A, CY ← A＋(addr16)＋CY	×	×	×
		A, ES:!addr16	4	2	5	A, CY ← A＋(ES, addr16)＋CY	×	×	×
		A, saddr	2	1	－	A, CY ← A＋(saddr)＋CY	×	×	×
		A, [HL]	1	1	4	A, CY ← A＋(HL)＋CY	×	×	×
		A, ES:[HL]	2	2	5	A,CY ← A＋(ES, HL)＋CY	×	×	×
		A, [HL+byte]	2	1	4	A, CY ← A＋(HL＋byte)＋CY	×	×	×
		A, ES:[HL+byte]	3	2	5	A,CY ← A＋((ES, HL)＋byte)＋CY	×	×	×
		A, [HL+B]	2	1	4	A, CY ← A＋(HL＋B)＋CY	×	×	×
		A, ES:[HL+B]	3	2	5	A,CY ← A＋((ES, HL)＋B)＋CY	×	×	×
		A, [HL+C]	2	1	4	A, CY ← A＋(HL＋C)＋CY	×	×	×
		A, ES:[HL+C]	3	2	5	A,CY ← A＋((ES, HL)＋C)＋CY	×	×	×
	SUB	A, #byte	2	1	－	A, CY ← A－byte	×	×	×
		saddr, #byte	3	2	－	(saddr), CY ← (saddr) －byte	×	×	×
		A, r[注3]	2	1	－	A, CY ← A－r	×	×	×
		r, A	2	1	－	r, CY ← r－A	×	×	×
		A, !addr16	3	1	4	A, CY ← A－(addr16)	×	×	×
		A, ES:!addr16	4	2	5	A, CY ← A－(ES, addr16)	×	×	×
		A, saddr	2	1	－	A, CY ← A－(saddr)	×	×	×
		A, [HL]	1	1	4	A, CY ← A－(HL)	×	×	×
		A, ES:[HL]	2	2	5	A,CY ← A－(ES, HL)	×	×	×
		A, [HL+byte]	2	1	4	A, CY ← A－(HL＋byte)	×	×	×
		A, ES:[HL+byte]	3	2	5	A,CY ← A－((ES, HL)＋byte)	×	×	×
		A, [HL+B]	2	1	4	A, CY ← A－(HL＋B)	×	×	×
		A, ES:[HL+B]	3	2	5	A,CY ← A－((ES, HL)＋B)	×	×	×
		A, [HL+C]	2	1	4	A, CY ← A－(HL＋C)	×	×	×
		A, ES:[HL+C]	3	2	5	A,CY ← A－((ES, HL)＋C)	×	×	×

注1.　内部RAM領域，SFR領域および拡張SFR領域をアクセスしたとき，またはデータ・アクセスをしないときのCPUクロック（f_{CLK}）数。

2.　コード・フラッシュ・メモリ領域をアクセスしたときのCPUクロック（f_{CLK}）数。

3.　r = Aを除く。

備考　クロック数は内部ROM（フラッシュ・メモリ）領域にプログラムがある場合です。内部RAM領域から命令フェッチする場合，最大4倍＋6クロックになります。

表5－5　RL78-S1コアのオペレーション一覧（8/17）

命令群	ニモニック	オペランド	バイト	クロック		オペレーション	フラグ		
				注1	注2		Z	AC	CY
8ビット演算	SUBC	A, #byte	2	1	－	A, CY ← A－byte－CY	×	×	×
		saddr, #byte	3	2	－	(saddr), CY ← (saddr)－byte－CY	×	×	×
		A, r[注3]	2	1	－	A, CY ← A－r－CY	×	×	×
		r, A	2	1	－	r, CY ← r－A－CY	×	×	×
		A, !addr16	3	1	4	A, CY ← A－(addr16)－CY	×	×	×
		A, ES:!addr16	4	2	5	A, CY ← A－(ES, addr16)－CY	×	×	×
		A, saddr	2	1	－	A, CY ← A－(saddr)－CY	×	×	×
		A, [HL]	1	1	4	A, CY ← A－(HL)－CY	×	×	×
		A, ES:[HL]	2	2	5	A,CY ← A－(ES, HL)－CY	×	×	×
		A, [HL+byte]	2	1	4	A, CY ← A－(HL＋byte)－CY	×	×	×
		A, ES:[HL+byte]	3	2	5	A,CY ← A－((ES, HL)＋byte)－CY	×	×	×
		A, [HL+B]	2	1	4	A, CY ← A－(HL＋B)－CY	×	×	×
		A, ES:[HL+B]	3	2	5	A,CY ← A－((ES, HL)＋B)－CY	×	×	×
		A, [HL+C]	2	1	4	A, CY ← A－(HL＋C)－CY	×	×	×
		A, ES:[HL+C]	3	2	5	A, CY ← A－((ES:HL)＋C)－CY	×	×	×
	AND	A, #byte	2	1	－	A ← A∧byte	×		
		saddr, #byte	3	2	－	(saddr) ← (saddr)∧byte	×		
		A, r[注3]	2	1	－	A ← A∧r	×		
		r, A	2	1	－	r ← r∧A	×		
		A, !addr16	3	1	4	A ← A∧(addr16)	×		
		A, ES:!addr16	4	2	5	A ← A∧(ES:addr16)	×		
		A, saddr	2	1	－	A ← A∧(saddr)	×		
		A, [HL]	1	1	4	A ← A∧(HL)	×		
		A, ES:[HL]	2	2	5	A ← A∧(ES:HL)	×		
		A, [HL+byte]	2	1	4	A ← A∧(HL＋byte)	×		
		A, ES:[HL+byte]	3	2	5	A ← A∧((ES:HL)＋byte)	×		
		A, [HL+B]	2	1	4	A ← A∧(HL＋B)	×		
		A, ES:[HL+B]	3	2	5	A ← A∧((ES:HL)＋B)	×		
		A, [HL+C]	2	1	4	A ← A∧(HL＋C)	×		
		A, ES:[HL+C]	3	2	5	A ← A∧((ES:HL)＋C)	×		

注1.　内部RAM領域，SFR領域および拡張SFR領域をアクセスしたとき，またはデータ・アクセスをしないときのCPUクロック（f_{CLK}）数。

2.　コード・フラッシュ・メモリ領域をアクセスしたときのCPUクロック（f_{CLK}）数。

3.　r = Aを除く。

備考　クロック数は内部ROM（フラッシュ・メモリ）領域にプログラムがある場合です。内部RAM領域から命令フェッチする場合，最大4倍＋6クロックになります。

表5－5　RL78-S1コアのオペレーション一覧（9/17）

命令群	ニモニック	オペランド	バイト	クロック		オペレーション	フラグ
				注1	注2		Z AC CY
8ビット演算	OR	A, #byte	2	1	－	A ← AVbyte	×
		saddr, #byte	3	2	－	(saddr) ← (saddr)Vbyte	×
		A, r[注3]	2	1	－	A ← AVr	×
		r, A	2	1	－	r ← rVA	×
		A, !addr16	3	1	4	A ← AV(addr16)	×
		A, ES:!addr16	4	2	5	A ← AV(ES:addr16)	×
		A, saddr	2	1	－	A ← AV(saddr)	×
		A, [HL]	1	1	4	A ← AV(HL)	×
		A, ES:[HL]	2	2	5	A ← AV(ES:HL)	×
		A, [HL+byte]	2	1	4	A ← AV(HL＋byte)	×
		A, ES:[HL+byte]	3	2	5	A ← AV((ES:HL)＋byte)	×
		A, [HL+B]	2	1	4	A ← AV(HL＋B)	×
		A, ES:[HL+B]	3	2	5	A ← AV((ES:HL)＋B)	×
		A, [HL+C]	2	1	4	A ← AV(HL＋C)	×
		A, ES:[HL+C]	3	2	5	A ← AV((ES:HL)＋C)	×
	XOR	A, #byte	2	1	－	A ← A⊻byte	×
		saddr, #byte	3	2	－	(saddr) ← (saddr)⊻byte	×
		A, r[注3]	2	1	－	A ← A⊻r	×
		r, A	2	1	－	r ← r⊻A	×
		A, !addr16	3	1	4	A ← A⊻(addr16)	×
		A, ES:!addr16	4	2	5	A ← A⊻(ES:addr16)	×
		A, saddr	2	1	－	A ← A⊻(saddr)	×
		A, [HL]	1	1	4	A ← A⊻(HL)	×
		A, ES:[HL]	2	2	5	A ← A⊻(ES:HL)	×
		A, [HL+byte]	2	1	4	A ← A⊻(HL＋byte)	×
		A, ES:[HL+byte]	3	2	5	A ← A⊻((ES:HL)＋byte)	×
		A, [HL+B]	2	1	4	A ← A⊻(HL＋B)	×
		A, ES:[HL+B]	3	2	5	A ← A⊻((ES:HL)＋B)	×
		A, [HL+C]	2	1	4	A ← A⊻(HL＋C)	×
		A, ES:[HL+C]	3	2	5	A ← A⊻((ES:HL)＋C)	×

注1.　内部RAM領域，SFR領域および拡張SFR領域をアクセスしたとき，またはデータ・アクセスをしないときのCPUクロック（f_{CLK}）数。

2.　コード・フラッシュ・メモリ領域をアクセスしたときのCPUクロック（f_{CLK}）数。

3.　r = Aを除く。

備考　クロック数は内部ROM（フラッシュ・メモリ）領域にプログラムがある場合です。内部RAM領域から命令フェッチする場合，最大4倍＋6クロックになります。

表5－5　RL78-S1コアのオペレーション一覧（10/17）

命令群	ニモニック	オペランド	バイト	クロック		オペレーション	フラグ		
				注1	注2		Z	AC	CY
8ビット演算	CMP	A, #byte	2	1	－	A－byte	×	×	×
		!addr16, #byte	4	1	4	(addr16)－byte	×	×	×
		ES:!addr16, #byte	5	2	5	(ES:addr16)－byte	×	×	×
		saddr, #byte	3	1	－	(saddr)－byte	×	×	×
		A, r[注3]	2	1	－	A－r	×	×	×
		r, A	2	1	－	r－A	×	×	×
		A, !addr16	3	1	4	A－(addr16)	×	×	×
		A, ES:!addr16	4	2	5	A－(ES:addr16)	×	×	×
		A, saddr	2	1	－	A－(saddr)	×	×	×
		A, [HL]	1	1	4	A－(HL)	×	×	×
		A, ES:[HL]	2	2	5	A－(ES:HL)	×	×	×
		A, [HL+byte]	2	1	4	A－(HL＋byte)	×	×	×
		A, ES:[HL+byte]	3	2	5	A－((ES:HL)＋byte)	×	×	×
		A, [HL+B]	2	1	4	A－(HL＋B)	×	×	×
		A, ES:[HL+B]	3	2	5	A－((ES:HL)＋B)	×	×	×
		A, [HL+C]	2	1	4	A－(HL＋C)	×	×	×
		A, ES:[HL+C]	3	2	5	A－((ES:HL)＋C)	×	×	×
	CMP0	A	1	1	－	A－00H	×	0	0
		X	1	1	－	X－00H	×	0	0
		B	1	1	－	B－00H	×	0	0
		C	1	1	－	C－00H	×	0	0
		!addr16	3	1	4	(addr16)－00H	×	0	0
		ES:!addr16	4	2	5	(ES:addr16)－00H	×	0	0
		saddr	2	1	－	(saddr)－00H	×	0	0
	CMPS	X, [HL+byte]	3	1	4	X－(HL＋byte)	×	×	×
		X, ES:[HL+byte]	4	2	5	X－((ES:HL)＋byte)	×	×	×

注1.　内部RAM領域，SFR領域および拡張SFR領域をアクセスしたとき，またはデータ・アクセスをしないときのCPUクロック（f_{CLK}）数。

2.　コード・フラッシュ・メモリ領域をアクセスしたときのCPUクロック（f_{CLK}）数。

3.　r = Aを除く。

備考　クロック数は内部ROM（フラッシュ・メモリ）領域にプログラムがある場合です。内部RAM領域から命令フェッチする場合，最大4倍＋6クロックになります。

表5－5　RL78-S1コアのオペレーション一覧（11/17）

命令群	ニモニック	オペランド	バイト	クロック		オペレーション	フラグ		
				注1	注2		Z	AC	CY
16ビット演算	ADDW	AX, #word	3	2	－	AX, CY ← AX＋word	×	×	×
		AX, AX	1	2	－	AX, CY ← AX＋AX	×	×	×
		AX, BC	1	2	－	AX, CY ← AX＋BC	×	×	×
		AX, DE	1	2	－	AX, CY ← AX＋DE	×	×	×
		AX, HL	1	2	－	AX, CY ← AX＋HL	×	×	×
		AX, !addr16	3	2	5	AX, CY ← AX＋(addr16)	×	×	×
		AX, ES:!addr16	4	3	6	AX, CY ← AX＋(ES:addr16)	×	×	×
		AX, saddrp	2	2	－	AX, CY ← AX＋(saddrp)	×	×	×
		AX, [HL+byte]	3	2	5	AX, CY ← AX＋(HL＋byte)	×	×	×
		AX, ES: [HL+byte]	4	3	6	AX, CY ← AX＋((ES:HL)＋byte)	×	×	×
	SUBW	AX, #word	3	2	－	AX, CY ← AX－word	×	×	×
		AX, BC	1	2	－	AX, CY ← AX－BC	×	×	×
		AX, DE	1	2	－	AX, CY ← AX－DE	×	×	×
		AX, HL	1	2	－	AX, CY ← AX－HL	×	×	×
		AX, !addr16	3	2	5	AX, CY ← AX－(addr16)	×	×	×
		AX, ES:!addr16	4	3	6	AX, CY ← AX－(ES:addr16)	×	×	×
		AX, saddrp	2	2	－	AX, CY ← AX－(saddrp)	×	×	×
		AX, [HL+byte]	3	2	5	AX, CY ← AX－(HL＋byte)	×	×	×
		AX, ES: [HL+byte]	4	3	6	AX, CY ← AX－((ES:HL)＋byte)	×	×	×
	CMPW	AX, #word	3	2	－	AX－word	×	×	×
		AX, BC	1	2	－	AX－BC	×	×	×
		AX, DE	1	2	－	AX－DE	×	×	×
		AX, HL	1	2	－	AX－HL	×	×	×
		AX, !addr16	3	2	5	AX－(addr16)	×	×	×
		AX, ES:!addr16	4	3	6	AX－(ES:addr16)	×	×	×
		AX, saddrp	2	2	－	AX－(saddrp)	×	×	×
		AX, [HL+byte]	3	2	5	AX－(HL＋byte)	×	×	×
		AX, ES: [HL+byte]	4	3	6	AX－((ES:HL)＋byte)	×	×	×
乗算	MULU	X	1	2	－	AX ← A×X			

注1.　内部RAM領域，SFR領域および拡張SFR領域をアクセスしたとき，またはデータ・アクセスをしないときのCPUクロック（f_{CLK}）数。

2.　コード・フラッシュ・メモリ領域をアクセスしたときのCPUクロック（f_{CLK}）数。

備考　クロック数は内部ROM（フラッシュ・メモリ）領域にプログラムがある場合です。内部RAM領域から命令フェッチする場合，最大4倍＋6クロックになります。

表5－5　RL78-S1コアのオペレーション一覧（12/17）

命令群	ニモニック	オペランド	バイト	クロック		オペレーション	フラグ
				注1	注2		Z AC CY
増減	INC	r	1	1	－	r ← r＋1	× ×
		!addr16	3	2	－	(addr16) ← (addr16)＋1	× ×
		ES:!addr16	4	3	－	(ES, addr16) ← (ES, addr16)＋1	× ×
		saddr	2	2	－	(saddr) ← (saddr)＋1	× ×
		[HL+byte]	3	2	－	(HL＋byte) ← (HL＋byte)＋1	× ×
		ES: [HL+byte]	4	3	－	((ES:HL)＋byte) ← ((ES:HL)＋byte)＋1	× ×
	DEC	r	1	1	－	r ← r－1	× ×
		!addr16	3	2	－	(addr16) ← (addr16)－1	× ×
		ES:!addr16	4	3	－	(ES, addr16) ← (ES, addr16) －1	× ×
		saddr	2	2	－	(saddr) ← (saddr)－1	× ×
		[HL+byte]	3	2	－	(HL＋byte) ← (HL＋byte) －1	× ×
		ES: [HL+byte]	4	3	－	((ES:HL)＋byte) ← ((ES:HL)＋byte) －1	× ×
	INCW	rp	1	2	－	rp ← rp＋1	
		!addr16	3	4	－	(addr16) ← (addr16)＋1	
		ES:!addr16	4	5	－	(ES, addr16) ← (ES, addr16)＋1	
		saddrp	2	4	－	(saddrp) ← (saddrp)＋1	
		[HL+byte]	3	4	－	(HL＋byte) ← (HL＋byte)＋1	
		ES: [HL+byte]	4	5	－	((ES:HL)＋byte) ← ((ES:HL)＋byte)＋1	
	DECW	rp	1	2	－	rp ← rp－1	
		!addr16	3	4	－	(addr16) ← (addr16)－1	
		ES:!addr16	4	5	－	(ES, addr16) ← (ES, addr16)－1	
		saddrp	2	4	－	(saddrp) ← (saddrp)－1	
		[HL+byte]	3	4	－	(HL＋byte) ← (HL＋byte) －1	
		ES: [HL+byte]	4	5	－	((ES:HL)＋byte) ← ((ES:HL)＋byte) －1	
シフト	SHR	A, cnt	2	1	－	(CY ← A_0, A_{m-1} ← A_m, A_7 ← 0) ×cnt	×
	SHRW	AX, cnt	2	2	－	(CY ← AX_0, AX_{m-1} ← AX_m, AX_{15} ← 0) ×cnt	×
	SHL	A, cnt	2	1	－	(CY ← A_7, A_m ← A_{m-1}, A_0 ← 0) ×cnt	×
		B, cnt	2	1	－	(CY ← B_7, B_m ← B_{m-1}, B_0 ← 0) ×cnt	×
		C, cnt	2	1	－	(CY ← C_7, C_m ← C_{m-1}, C_0 ← 0) ×cnt	×
	SHLW	AX, cnt	2	2	－	(CY ← AX_{15}, AX_m ← AX_{m-1}, AX_0 ← 0) ×cnt	×
		BC, cnt	2	2	－	(CY ← BC_{15}, BC_m ← BC_{m-1}, BC_0 ← 0) ×cnt	×
	SAR	A, cnt	2	1	－	(CY ← A_0, A_{m-1} ← A_m, A_7 ← A_7) ×cnt	×
	SARW	AX, cnt	2	2	－	(CY ← AX_0, AX_{m-1} ← AX_m, AX_{15} ← AX_{15}) ×cnt	×

注1.　内部RAM領域，SFR領域および拡張SFR領域をアクセスしたとき，またはデータ・アクセスをしないときのCPUクロック（f_{CLK}）数。

2.　コード・フラッシュ・メモリ領域をアクセスしたときのCPUクロック（f_{CLK}）数。

備考1.　クロック数は内部ROM（フラッシュ・メモリ）領域にプログラムがある場合です。内部RAM領域から命令フェッチする場合，最大4倍＋6クロックになります。

2.　cntはビット・シフト数です。

表5－5　RL78-S1コアのオペレーション一覧（13/17）

命令群	ニモニック	オペランド	バイト	クロック		オペレーション	フラグ
				注1	注2		Z AC CY
ローテート	ROR	A, 1	2	1	－	(CY, A_7 ← A_0, A_{m-1} ← A_m) × 1	×
	ROL	A, 1	2	1	－	(CY, A_0 ← A_7, A_{m+1} ← A_m) × 1	×
	RORC	A, 1	2	1	－	(CY ← A_0, A_7 ← CY, A_{m-1} ← A_m) × 1	×
	ROLC	A, 1	2	1	－	(CY ← A_7, A_0 ← CY, A_{m+1} ← A_m) × 1	×
	ROLWC	AX,1	2	2	－	(CY ← AX_{15}, AX_0 ← CY, AX_{m+1} ← AX_m) ×1	×
		BC,1	2	2	－	(CY ← BC_{15}, BC_0 ← CY, BC_{m+1} ← BC_m) ×1	×
ビット操作	MOV1	CY, A.bit	2	1	－	CY ← A.bit	×
		A.bit, CY	2	1	－	A.bit ← CY	
		CY, PSW.bit	3	1	－	CY ← PSW.bit	×
		PSW.bit, CY	3	4	－	PSW.bit ← CY	× ×
		CY, saddr.bit	3	1	－	CY ← (saddr).bit	×
		saddr.bit, CY	3	2	－	(saddr).bit ← CY	
		CY, sfr.bit	3	1	－	CY ← sfr.bit	×
		sfr.bit, CY	3	2	－	sfr.bit ← CY	
		CY,[HL].bit	2	1	4	CY ← (HL).bit	×
		[HL].bit, CY	2	2	－	(HL).bit ← CY	
		CY, ES:[HL].bit	3	2	5	CY ← (ES, HL).bit	×
		ES:[HL].bit, CY	3	3	－	(ES, HL).bit ← CY	
	AND1	CY, A.bit	2	1	－	CY ← CY∧A.bit	×
		CY, PSW.bit	3	1	－	CY ← CY∧PSW.bit	×
		CY, saddr.bit	3	1	－	CY ← CY∧(saddr).bit	×
		CY, sfr.bit	3	1	－	CY ← CY∧sfr.bit	×
		CY,[HL].bit	2	1	4	CY ← CY∧(HL).bit	×
		CY, ES:[HL].bit	3	2	5	CY ← CY∧(ES, HL).bit	×
	OR1	CY, A.bit	2	1	－	CY ← CY∨A.bit	×
		CY, PSW.bit	3	1	－	CY ← CY∨PSW.bit	×
		CY, saddr.bit	3	1	－	CY ← CY∨(saddr).bit	×
		CY, sfr.bit	3	1	－	CY ← CY∨sfr.bit	×
		CY, [HL].bit	2	1	4	CY ← CY∨(HL).bit	×
		CY, ES:[HL].bit	3	2	5	CY ← CY∨(ES, HL).bit	×

注1.　内部RAM領域，SFR領域および拡張SFR領域をアクセスしたとき，またはデータ・アクセスをしないときのCPUクロック（f_{CLK}）数。

2.　コード・フラッシュ・メモリ領域をアクセスしたときのCPUクロック（f_{CLK}）数。

備考　クロック数は内部ROM（フラッシュ・メモリ）領域にプログラムがある場合です。内部RAM領域から命令フェッチする場合，最大4倍＋6クロックになります。

表5－5　RL78-S1コアのオペレーション一覧（14/17）

命令群	ニモニック	オペランド	バイト	クロック		オペレーション	フラグ		
				注1	注2		Z	AC	CY
ビット操作	XOR1	CY, A.bit	2	1	－	CY ← CY∀A.bit			×
		CY, PSW.bit	3	1	－	CY ← CY∀PSW.bit			×
		CY, saddr.bit	3	1	－	CY ← CY∀(saddr).bit			×
		CY, sfr.bit	3	1	－	CY ← CY∀sfr.bit			×
		CY, [HL].bit	2	1	4	CY ← CY∀(HL).bit			×
		CY, ES:[HL].bit	3	2	5	CY ← CY∀(ES, HL).bit			×
	SET1	A.bit	2	1	－	A.bit ← 1			
		PSW.bit	3	4	－	PSW.bit ← 1	×	×	×
		!addr16.bit	4	2	－	(addr16).bit ← 1			
		ES:!addr16.bit	5	3	－	(ES, addr16).bit ← 1			
		saddr.bit	3	2	－	(saddr).bit ← 1			
		sfr.bit	3	2	－	sfr.bit ← 1			
		[HL].bit	2	2	－	(HL).bit ← 1			
		ES:[HL].bit	3	3	－	(ES, HL).bit ← 1			
	CLR1	A.bit	2	1	－	A.bit ← 0			
		PSW.bit	3	4	－	PSW.bit ← 0	×	×	×
		!addr16.bit	4	2	－	(addr16).bit ← 0			
		ES:!addr16.bit	5	3	－	(ES, addr16).bit ← 0			
		saddr.bit	3	2	－	(saddr.bit) ← 0			
		sfr.bit	3	2	－	sfr.bit ← 0			
		[HL].bit	2	2	－	(HL).bit ← 0			
		ES:[HL].bit	3	3	－	(ES, HL).bit ← 0			
	SET1	CY	2	1	－	CY ← 1			1
	CLR1	CY	2	1	－	CY ← 0			0
	NOT1	CY	2	1	－	CY ← $\overline{CY}$			×

注1.　内部RAM領域，SFR領域および拡張SFR領域をアクセスしたとき，またはデータ・アクセスをしないときのCPUクロック（f_{CLK}）数。

2.　コード・フラッシュ・メモリ領域をアクセスしたときのCPUクロック（f_{CLK}）数。

備考　クロック数は内部ROM（フラッシュ・メモリ）領域にプログラムがある場合です。内部RAM領域から命令フェッチする場合，最大4倍＋6クロックになります。

表5－5　RL78-S1コアのオペレーション一覧（15/17）

命令群	ニモニック	オペランド	バイト	クロック		オペレーション	フラグ
				注1	注2		Z AC CY
コール・リターン	CALL	rp	2	4	－	(SP－2) ← (PC＋2)S, (SP－3) ← (PC＋2)H, (SP－4) ← (PC＋2)L, PC ← CS, rp, SP ← SP－4	
		$!addr20	3	4	－	(SP－2) ← (PC＋3)S, (SP－3) ← (PC＋3)H, (SP－4) ← (PC＋3)L, PC ← PC＋3＋jdisp16, SP ← SP－4	
		!addr16	3	4	－	(SP－2) ← (PC＋3)S, (SP－3) ← (PC＋3)H, (SP－4) ← (PC＋3)L, PC ← 0000, addr16, SP ← SP－4	
		!!addr20	4	4	－	(SP－2) ← (PC＋4)S, (SP－3) ← (PC＋4)H, (SP－4) ← (PC＋4)L, PC ← addr20, SP ← SP－4	
	CALLT	[addr5]	2	6	－	(SP－2) ← (PC＋2)S , (SP－3) ← (PC＋2)H, (SP－4) ← (PC＋2)L , PCS ← 0000, PCH ← (0000, addr5＋1), PCL ← (0000, addr5), SP ← SP－4	
	BRK	－	2	7	－	(SP－1) ← PSW, (SP－2) ← (PC＋2)S, (SP－3) ← (PC＋2)H, (SP－4) ← (PC＋2)L, PCS ← 0000, PCH ← (0007FH), PCL ← (0007EH), SP ← SP－4, IE ← 0	
	RET	－	1	7	－	PCL ← (SP), PCH ← (SP＋1), PCS ← (SP＋2), SP ← SP＋4	
	RETI	－	2	8	－	PCL ← (SP), PCH ← (SP＋1), PCS ← (SP＋2), PSW ← (SP＋3), SP ← SP＋4	R R R
	RETB	－	2	8	－	PCL ← (SP), PCH ← (SP＋1), PCS ← (SP＋2), PSW ← (SP＋3), SP ← SP＋4	R R R

注1.　内部RAM領域，SFR領域および拡張SFR領域をアクセスしたとき，またはデータ・アクセスをしないときのCPUクロック（f_{CLK}）数。

2.　コード・フラッシュ・メモリ領域をアクセスしたときのCPUクロック（f_{CLK}）数。

備考　クロック数は内部ROM（フラッシュ・メモリ）領域にプログラムがある場合です。内部RAM領域から命令フェッチする場合，最大4倍＋6クロックになります。

表5－5　RL78-S1コアのオペレーション一覧（16/17）

命令群	ニモニック	オペランド	バイト	クロック		オペレーション	フラグ
				注1	注2		Z AC CY
スタック操作	PUSH	PSW	2	2	－	(SP－1) ← PSW, (SP－2) ← 00H, SP ← SP－2	
		rp	1	2	－	(SP－1) ← rp_H, (SP－2) ← rp_L, SP ← SP－2	
	POP	PSW	2	4	－	PSW ← (SP＋1), SP ← SP＋2	R R R
		rp	1	2	－	rp_L ← (SP), rp_H ← (SP＋1), SP ← SP＋2	
	MOVW	SP, #word	4	2	－	SP ← word	
		SP, AX	2	2	－	SP ← AX	
		AX, SP	2	2	－	AX ← SP	
		HL, SP	3	2	－	HL ← SP	
		BC, SP	3	2	－	BC ← SP	
		DE, SP	3	2	－	DE ← SP	
	ADDW	SP, #byte	2	2	－	SP ← SP＋byte	
	SUBW	SP, #byte	2	2	－	SP ← SP－byte	
無条件分岐	BR	AX	2	3	－	PC ← CS, AX	
		$addr20	2	3	－	PC ← PC＋2＋jdisp8	
		$!addr20	3	3	－	PC ← PC＋3＋jdisp16	
		!addr16	3	3	－	PC ← 0000, addr16	
		!!addr20	4	3	－	PC ← addr20	
条件付き分岐	BC	$addr20	2	2/4[注3]	－	PC ← PC＋2＋jdisp8 if CY = 1	
	BNC	$addr20	2	2/4[注3]	－	PC ← PC＋2＋jdisp8 if CY = 0	
	BZ	$addr20	2	2/4[注3]	－	PC ← PC＋2＋jdisp8 if Z = 1	
	BNZ	$addr20	2	2/4[注3]	－	PC ← PC＋2＋jdisp8 if Z = 0	
	BH	$addr20	3	2/4[注3]	－	PC ← PC+3+jdisp8 if (Z∨CY)=0	
	BNH	$addr20	3	2/4[注3]	－	PC ← PC+3+jdisp8 if (Z∨CY)=1	
	BT	saddr.bit, $addr20	4	3/5[注3]	－	PC ← PC＋4＋jdisp8 if (saddr).bit = 1	
		sfr.bit, $addr20	4	3/5[注3]	－	PC ← PC＋4＋jdisp8 if sfr.bit = 1	
		A.bit, $addr20	3	3/5[注3]	－	PC ← PC＋3＋jdisp8 if A.bit = 1	
		PSW.bit, $addr20	4	3/5[注3]	－	PC ← PC＋4＋jdisp8 if PSW.bit = 1	
		[HL].bit, $addr20	3	3/5[注3]	6/7	PC ← PC＋3＋jdisp8 if (HL).bit = 1	
		ES:[HL].bit, $addr20	4	4/6[注3]	7/8	PC ← PC＋4＋jdisp8 if (ES, HL).bit = 1	

注1.　内部RAM領域，SFR領域および拡張SFR領域をアクセスしたとき，またはデータ・アクセスをしないときのCPUクロック（f_{CLK}）数。

2.　コード・フラッシュ・メモリ領域をアクセスしたときのCPUクロック（f_{CLK}）数。

3.　クロック数は“条件不成立時／条件成立時”を表しています。

備考　クロック数は内部ROM（フラッシュ・メモリ）領域にプログラムがある場合です。内部RAM領域から命令フェッチする場合，最大4倍＋6クロックになります。

表5－5　RL78-S1コアのオペレーション一覧（17/17）

命令群	ニモニック	オペランド	バイト	クロック		オペレーション	フラグ
				注1	注2		Z AC CY
条件付き分岐	BF	saddr.bit, $addr20	4	3/5[注3]	－	PC ← PC＋4＋jdisp8 if (saddr).bit = 0	
		sfr.bit, $addr20	4	3/5[注3]	－	PC ← PC＋4＋jdisp8 if sfr.bit = 0	
		A.bit, $addr20	3	3/5[注3]	－	PC ← PC＋3＋jdisp8 if A.bit = 0	
		PSW.bit, $addr20	4	3/5[注3]	－	PC ← PC＋4＋jdisp8 if PSW.bit = 0	
		[HL].bit, $addr20	3	3/5[注3]	6/7	PC ← PC＋3＋jdisp8 if (HL).bit = 0	
		ES:[HL].bit, $addr20	4	4/6[注3]	7/8	PC ← PC＋4＋jdisp8 if (ES, HL).bit = 0	
	BTCLR	saddr.bit, $addr20	4	3/5[注3]	－	PC ← PC＋4＋jdisp8 if (saddr).bit = 1 then reset (saddr).bit	
		sfr.bit, $addr20	4	3/5[注3]	－	PC ← PC＋4＋jdisp8 if sfr.bit = 1 then reset sfr.bit	
		A.bit, $addr20	3	3/5[注3]	－	PC ← PC＋3＋jdisp8 if A.bit = 1 then reset A.bit	
		PSW.bit, $addr20	4	3/5[注3]	－	PC ← PC＋4＋jdisp8 if PSW.bit = 1 then reset PSW.bit	× × ×
		[HL].bit, $addr20	3	3/5[注3]	－	PC ← PC＋3＋jdisp8 if (HL).bit = 1 then reset (HL).bit	
		ES:[HL].bit, $addr20	4	4/6[注3]	－	PC ← PC＋4＋jdisp8 if (ES, HL).bit = 1 then reset (ES, HL).bit	
条件付きスキップ	SKC	－	2	1	－	Next instruction skip if CY = 1	
	SKNC	－	2	1	－	Next instruction skip if CY = 0	
	SKZ	－	2	1	－	Next instruction skip if Z = 1	
	SKNZ	－	2	1	－	Next instruction skip if Z = 0	
	SKH	－	2	1	－	Next instruction skip if (Z∨CY)=0	
	SKNH	－	2	1	－	Next instruction skip if (Z∨CY)=1	
CPU制御	NOP	－	1	1	－	No Operation	
	EI	－	3	4	－	IE ← 1(Enable Interrupt)	
	DI	－	3	4	－	IE ← 0(Disable Interrupt)	
	HALT	－	2	3	－	Set HALT Mode	
	STOP	－	2	3	－	Set STOP Mode	

注1.　内部RAM領域，SFR領域および拡張SFR領域をアクセスしたとき，またはデータ・アクセスをしないときのCPUクロック（f_{CLK}）数。

2.　コード・フラッシュ・メモリ領域をアクセスしたときのCPUクロック（f_{CLK}）数。

3.　クロック数は“条件不成立時／条件成立時”を表しています。

備考　クロック数は内部ROM（フラッシュ・メモリ）領域にプログラムがある場合です。内部RAM領域から命令フェッチする場合，最大4倍＋6クロックになります。

5.5.2　RL78-S2コアのオペレーション一覧

表5－6　RL78-S2コアのオペレーション一覧（1/17）

命令群	ニモニック	オペランド	バイト	クロック		オペレーション	フラグ
				注1	注2		Z AC CY
8ビット・データ転送	MOV	r, #byte	2	1	－	r ← byte	
		PSW, #byte	3	3	－	PSW ← byte	× × ×
		CS, #byte	3	1	－	CS ← byte	
		ES, #byte	2	1	－	ES ← byte	
		!addr16, #byte	4	1	－	(addr16) ← byte	
		ES:!addr16, #byte	5	2	－	(ES, addr16) ← byte	
		saddr, #byte	3	1	－	(saddr) ← byte	
		sfr, #byte	3	1	－	sfr ← byte	
		[DE+byte], #byte	3	1	－	(DE＋byte) ← byte	
		ES:[DE+byte],#byte	4	2	－	((ES, DE)＋byte) ← byte	
		[HL+byte], #byte	3	1	－	(HL＋byte) ← byte	
		ES:[HL+byte],#byte	4	2	－	((ES, HL)＋byte) ← byte	
		[SP+byte], #byte	3	1	－	(SP＋byte) ← byte	
		word[B], #byte	4	1	－	(B＋word) ← byte	
		ES:word[B], #byte	5	2	－	((ES, B)＋word) ← byte	
		word[C], #byte	4	1	－	(C＋word) ← byte	
		ES:word[C], #byte	5	2	－	((ES, C)＋word) ← byte	
		word[BC], #byte	4	1	－	(BC＋word) ← byte	
		ES:word[BC], #byte	5	2	－	((ES, BC)＋word) ← byte	
		A, r[注3]	1	1	－	A ← r	
		r, A[注3]	1	1	－	r ← A	
		A, PSW	2	1	－	A ← PSW	
		PSW, A	2	3	－	PSW ← A	× × ×
		A, CS	2	1	－	A ← CS	
		CS, A	2	1	－	CS ← A	
		A, ES	2	1	－	A ← ES	
		ES, A	2	1	－	ES ← A	
		A, !addr16	3	1	4	A ← (addr16)	
		A, ES:!addr16	4	2	5	A ← (ES, addr16)	
		!addr16, A	3	1	－	(addr16) ← A	
		ES:!addr16, A	4	2	－	(ES, addr16) ← A	
		A, saddr	2	1	－	A ← (saddr)	
		saddr, A	2	1	－	(saddr) ← A	

注1.　内部RAM領域，SFR領域および拡張SFR領域をアクセスしたとき，またはデータ・アクセスをしないときのCPUクロック（f_{CLK}）数。

2.　コード・フラッシュ・メモリ領域および8ビット命令でデータ・フラッシュ・メモリ領域をアクセスしたときのCPUクロック（f_{CLK}）数。

3.　r = Aを除く。

備考　クロック数は内部ROM（フラッシュ・メモリ）領域にプログラムがある場合です。内部RAM領域から命令フェッチする場合，最大2倍＋3クロックになります。

表5－6　RL78-S2コアのオペレーション一覧（2/17）

命令群	ニモニック	オペランド	バイト	クロック		オペレーション	フラグ
				注1	注2		Z AC CY
8ビット・データ転送	MOV	A, sfr	2	1	－	A ← sfr	
		sfr, A	2	1	－	sfr ← A	
		A, [DE]	1	1	4	A ← (DE)	
		[DE], A	1	1	－	(DE) ← A	
		A, ES:[DE]	2	2	5	A ← (ES, DE)	
		ES:[DE], A	2	2	－	(ES, DE) ← A	
		A, [HL]	1	1	4	A ← (HL)	
		[HL], A	1	1	－	(HL) ← A	
		A, ES:[HL]	2	2	5	A ← (ES, HL)	
		ES:[HL], A	2	2	－	(ES, HL) ← A	
		A, [DE+byte]	2	1	4	A ← (DE＋byte)	
		[DE+byte], A	2	1	－	(DE＋byte) ← A	
		A, ES:[DE+byte]	3	2	5	A ← ((ES, DE)＋byte)	
		ES:[DE+byte], A	3	2	－	((ES, DE)＋byte) ← A	
		A, [HL+byte]	2	1	4	A ← (HL＋byte)	
		[HL+byte], A	2	1	－	(HL＋byte) ← A	
		A, ES:[HL+byte]	3	2	5	A ← ((ES, HL)＋byte)	
		ES:[HL+byte], A	3	2	－	((ES, HL)＋byte) ← A	
		A, [SP+byte]	2	1	－	A ← (SP＋byte)	
		[SP+byte], A	2	1	－	(SP＋byte) ← A	
		A, word[B]	3	1	4	A ← (B＋word)	
		word[B], A	3	1	－	(B＋word) ← A	
		A, ES:word[B]	4	2	5	A ← ((ES, B)＋word)	
		ES:word[B], A	4	2	－	((ES, B)＋word) ← A	
		A, word[C]	3	1	4	A ← (C＋word)	
		word[C], A	3	1	－	(C＋word) ← A	
		A, ES:word[C]	4	2	5	A ← ((ES, C)＋word)	
		ES:word[C], A	4	2	－	((ES, C)＋word) ← A	
		A, word[BC]	3	1	4	A ← (BC＋word)	
		word[BC], A	3	1	－	(BC＋word) ← A	
		A, ES:word[BC]	4	2	5	A ← ((ES, BC)＋word)	
		ES:word[BC], A	4	2	－	((ES, BC)＋word) ← A	

注1.　内部RAM領域，SFR領域および拡張SFR領域をアクセスしたとき，またはデータ・アクセスをしないときのCPUクロック（f_{CLK}）数。

2.　コード・フラッシュ・メモリ領域および8ビット命令でデータ・フラッシュ・メモリ領域をアクセスしたときのCPUクロック（f_{CLK}）数。

備考　クロック数は内部ROM（フラッシュ・メモリ）領域にプログラムがある場合です。内部RAM領域から命令フェッチする場合，最大2倍＋3クロックになります。

表5－6　RL78-S2コアのオペレーション一覧（3/17）

命令群	ニモニック	オペランド	バイト	クロック		オペレーション	フラグ
				注1	注2		Z AC CY
8ビット・データ転送	MOV	A, [HL+B]	2	1	4	A ← (HL＋B)	
		[HL+B], A	2	1	－	(HL＋B) ← A	
		A, ES:[HL+B]	3	2	5	A ← ((ES, HL)＋B)	
		ES:[HL+B], A	3	2	－	((ES, HL)＋B) ← A	
		A, [HL+C]	2	1	4	A ← (HL＋C)	
		[HL+C], A	2	1	－	(HL＋C) ← A	
		A, ES:[HL+C]	3	2	5	A ← ((ES, HL)＋C)	
		ES:[HL+C], A	3	2	－	((ES, HL)＋C) ← A	
		X, !addr16	3	1	4	X ← (addr16)	
		X, ES:!addr16	4	2	5	X ← (ES, addr16)	
		X, saddr	2	1	－	X ← (saddr)	
		B, !addr16	3	1	4	B ← (addr16)	
		B, ES:!addr16	4	2	5	B ← (ES, addr16)	
		B, saddr	2	1	－	B ← (saddr)	
		C, !addr16	3	1	4	C ← (addr16)	
		C, ES:!addr16	4	2	5	C ← (ES, addr16)	
		C, saddr	2	1	－	C ← (saddr)	
		ES, saddr	3	1	－	ES ← (saddr)	
	XCH	A, r[注3]	1 (r = X) 2 (r = X以外)	1	－	A ⟷ r	
		A, !addr16	4	2	－	A ⟷ (addr16)	
		A, ES:!addr16	5	3	－	A ⟷ (ES, addr16)	
		A, saddr	3	2	－	A ⟷ (saddr)	
		A, sfr	3	2	－	A ⟷ sfr	
		A, [DE]	2	2	－	A ⟷ (DE)	
		A, ES:[DE]	3	3	－	A ⟷ (ES, DE)	
		A, [HL]	2	2	－	A ⟷ (HL)	
		A, ES:[HL]	3	3	－	A ⟷ (ES, HL)	
		A, [DE+byte]	3	2	－	A ⟷ (DE＋byte)	
		A, ES:[DE+byte]	4	3	－	A ⟷ ((ES, DE)＋byte)	
		A, [HL+byte]	3	2	－	A ⟷ (HL＋byte)	
		A, ES:[HL+byte]	4	3	－	A ⟷ ((ES, HL)＋byte)	

注1.　内部RAM領域，SFR領域および拡張SFR領域をアクセスしたとき，またはデータ・アクセスをしないときのCPUクロック（f_{CLK}）数。

2.　コード・フラッシュ・メモリ領域および8ビット命令でデータ・フラッシュ・メモリ領域をアクセスしたときのCPUクロック（f_{CLK}）数。

3.　r = Aを除く。

備考　クロック数は内部ROM（フラッシュ・メモリ）領域にプログラムがある場合です。内部RAM領域から命令フェッチする場合，最大2倍＋3クロックになります。

表5－6　RL78-S2コアのオペレーション一覧（4/17）

命令群	ニモニック	オペランド	バイト	クロック		オペレーション	フラグ
				注1	注2		Z AC CY
8ビット・データ転送	XCH	A, [HL+B]	2	2	－	A ⟷ (HL＋B)	
		A, ES:[HL+B]	3	3	－	A ⟷ ((ES, HL)＋B)	
		A, [HL+C]	2	2	－	A ⟷ (HL＋C)	
		A, ES:[HL+C]	3	3	－	A ⟷ ((ES, HL)＋C)	
	ONEB	A	1	1	－	A ← 01H	
		X	1	1	－	X ← 01H	
		B	1	1	－	B ← 01H	
		C	1	1	－	C ← 01H	
		!addr16	3	1	－	(addr16) ← 01H	
		ES:!addr16	4	2	－	(ES, addr16) ← 01H	
		saddr	2	1	－	(saddr) ← 01H	
	CLRB	A	1	1	－	A ← 00H	
		X	1	1	－	X ← 00H	
		B	1	1	－	B ← 00H	
		C	1	1	－	C ← 00H	
		!addr16	3	1	－	(addr16) ← 00H	
		ES:!addr16	4	2	－	(ES,addr16) ← 00H	
		saddr	2	1	－	(saddr) ← 00H	
	MOVS	[HL+byte], X	3	1	－	(HL＋byte) ← X	× ×
		ES:[HL+byte], X	4	2	－	(ES, HL＋byte) ← X	× ×
16ビット・データ転送	MOVW	rp, #word	3	1	－	rp ← word	
		saddrp, #word	4	1	－	(saddrp) ← word	
		sfrp, #word	4	1	－	sfrp ← word	
		AX, rp[注3]	1	1	－	AX ← rp	
		rp, AX[注3]	1	1	－	rp ← AX	
		AX, !addr16	3	1	4	AX ←（addr16）	
		!addr16, AX	3	1	－	（addr16）← AX	
		AX, ES:!addr16	4	2	5	AX ← (ES, addr16)	
		ES:!addr16, AX	4	2	－	(ES, addr16) ← AX	
		AX, saddrp	2	1	－	AX ← (saddrp)	
		saddrp, AX	2	1	－	(saddrp) ← AX	
		AX, sfrp	2	1	－	AX ← sfrp	
		sfrp, AX	2	1	－	sfrp ← AX	

注1.　内部RAM領域，SFR領域および拡張SFR領域をアクセスしたとき，またはデータ・アクセスをしないときのCPUクロック（f_{CLK}）数。

2.　コード・フラッシュ・メモリ領域および8ビット命令でデータ・フラッシュ・メモリ領域をアクセスしたときのCPUクロック（f_{CLK}）数。

3.　rp = AXを除く。

備考　クロック数は内部ROM（フラッシュ・メモリ）領域にプログラムがある場合です。内部RAM領域から命令フェッチする場合，最大2倍＋3クロックになります。

表5－6　RL78-S2コアのオペレーション一覧（5/17）

命令群	ニモニック	オペランド	バイト	クロック		オペレーション	フラグ
				注1	注2		Z AC CY
16ビット・データ転送	MOVW	AX, [DE]	1	1	4	AX ← (DE)	
		[DE], AX	1	1	－	(DE) ← AX	
		AX, ES:[DE]	2	2	5	AX ← (ES, DE)	
		ES:[DE], AX	2	2	－	(ES, DE) ← AX	
		AX, [HL]	1	1	4	AX ← (HL)	
		[HL], AX	1	1	－	(HL) ← AX	
		AX, ES:[HL]	2	2	5	AX ← (ES, HL)	
		ES:[HL], AX	2	2	－	(ES, HL) ← AX	
		AX, [DE+byte]	2	1	4	AX ← (DE＋byte)	
		[DE+byte], AX	2	1	－	(DE＋byte) ← AX	
		AX, ES:[DE+byte]	3	2	5	AX ← ((ES, DE)＋byte)	
		ES:[DE+byte], AX	3	2	－	((ES, DE)＋byte) ← AX	
		AX, [HL+byte]	2	1	4	AX ← (HL＋byte)	
		[HL+byte], AX	2	1	－	(HL＋byte) ← AX	
		AX, ES:[HL+byte]	3	2	5	AX ← ((ES, HL)＋byte)	
		ES:[HL+byte], AX	3	2	－	((ES, HL)＋byte) ← AX	
		AX, [SP+byte]	2	1	－	AX ← (SP＋byte)	
		[SP+byte], AX	2	1	－	(SP＋byte) ← AX	
		AX, word[B]	3	1	4	AX ← (B＋word)	
		word[B], AX	3	1	－	(B＋word) ← AX	
		AX, ES:word[B]	4	2	5	AX ← ((ES, B)＋word)	
		ES:word[B], AX	4	2	－	((ES, B)＋word) ← AX	
		AX, word[C]	3	1	4	AX ← (C＋word)	
		word[C], AX	3	1	－	(C＋word) ← AX	
		AX, ES:word[C]	4	2	5	AX ← ((ES, C)＋word)	
		ES:word[C], AX	4	2	－	((ES, C)＋word) ← AX	
		AX, word[BC]	3	1	4	AX ← (BC＋word)	
		word[BC], AX	3	1	－	(BC＋word) ← AX	
		AX, ES:word[BC]	4	2	5	AX ← ((ES, BC)＋word)	
		ES:word[BC], AX	4	2	－	((ES, BC)＋word) ← AX	

注1.　内部RAM領域，SFR領域および拡張SFR領域をアクセスしたとき，またはデータ・アクセスをしないときのCPUクロック（f_{CLK}）数。

2.　コード・フラッシュ・メモリ領域および8ビット命令でデータ・フラッシュ・メモリ領域をアクセスしたときのCPUクロック（f_{CLK}）数。

備考　クロック数は内部ROM（フラッシュ・メモリ）領域にプログラムがある場合です。内部RAM領域から命令フェッチする場合，最大2倍＋3クロックになります。

表5－6　RL78-S2コアのオペレーション一覧（6/17）

命令群	ニモニック	オペランド	バイト	クロック		オペレーション	フラグ
				注1	注2		Z AC CY
16ビット・データ転送	MOVW	BC, !addr16	3	1	4	BC ← (addr16)	
		BC, ES:!addr16	4	2	5	BC ← (ES, addr16)	
		DE, !addr16	3	1	4	DE ← (addr16)	
		DE, ES:!addr16	4	2	5	DE ← (ES, addr16)	
		HL, !addr16	3	1	4	HL ← (addr16)	
		HL, ES:!addr16	4	2	5	HL ← (ES, addr16)	
		BC, saddrp	2	1	－	BC ← (saddrp)	
		DE, saddrp	2	1	－	DE ← (saddrp)	
		HL, saddrp	2	1	－	HL ← (saddrp)	
	XCHW	AX, rp[注3]	1	1	－	AX ⟷ rp	
	ONEW	AX	1	1	－	AX ← 0001H	
		BC	1	1	－	BC ← 0001H	
	CLRW	AX	1	1	－	AX ← 0000H	
		BC	1	1	－	BC ← 0000H	
8ビット演算	ADD	A, #byte	2	1	－	A, CY ← A＋byte	× × ×
		saddr, #byte	3	2	－	(saddr), CY ← (saddr)＋byte	× × ×
		A, r[注4]	2	1	－	A, CY ← A＋r	× × ×
		r, A	2	1	－	r, CY ← r＋A	× × ×
		A, !addr16	3	1	4	A, CY ← A＋(addr16)	× × ×
		A, ES:!addr16	4	2	5	A, CY ← A＋(ES, addr16)	× × ×
		A, saddr	2	1	－	A, CY ← A＋(saddr)	× × ×
		A, [HL]	1	1	4	A, CY ← A＋(HL)	× × ×
		A, ES:[HL]	2	2	5	A,CY ← A＋(ES, HL)	× × ×
		A, [HL+byte]	2	1	4	A, CY ← A＋(HL＋byte)	× × ×
		A, ES:[HL+byte]	3	2	5	A,CY ← A＋((ES, HL)＋byte)	× × ×
		A, [HL+B]	2	1	4	A, CY ← A＋(HL＋B)	× × ×
		A, ES:[HL+B]	3	2	5	A,CY ← A＋((ES, HL)＋B)	× × ×
		A, [HL+C]	2	1	4	A, CY ← A＋(HL＋C)	× × ×
		A, ES:[HL+C]	3	2	5	A,CY ← A＋((ES, HL)＋C)	× × ×

注1.　内部RAM領域，SFR領域および拡張SFR領域をアクセスしたとき，またはデータ・アクセスをしないときのCPUクロック（f_{CLK}）数。

2.　コード・フラッシュ・メモリ領域および8ビット命令でデータ・フラッシュ・メモリ領域をアクセスしたときのCPUクロック（f_{CLK}）数。

3.　rp = AXを除く。

4.　r = Aを除く。

備考　クロック数は内部ROM（フラッシュ・メモリ）領域にプログラムがある場合です。内部RAM領域から命令フェッチする場合，最大2倍＋3クロックになります。

表5－6　RL78-S2コアのオペレーション一覧（7/17）

命令群	ニモニック	オペランド	バイト	クロック		オペレーション	フラグ		
				注1	注2		Z	AC	CY
8ビット演算	ADDC	A, #byte	2	1	－	A, CY ← A＋byte＋CY	×	×	×
		saddr, #byte	3	2	－	(saddr), CY ← (saddr)＋byte＋CY	×	×	×
		A, r[注3]	2	1	－	A, CY ← A＋r＋CY	×	×	×
		r, A	2	1	－	r, CY ← r＋A＋CY	×	×	×
		A, !addr16	3	1	4	A, CY ← A＋(addr16)＋CY	×	×	×
		A, ES:!addr16	4	2	5	A, CY ← A＋(ES, addr16)＋CY	×	×	×
		A, saddr	2	1	－	A, CY ← A＋(saddr)＋CY	×	×	×
		A, [HL]	1	1	4	A, CY ← A＋(HL)＋CY	×	×	×
		A, ES:[HL]	2	2	5	A,CY ← A＋(ES, HL)＋CY	×	×	×
		A, [HL+byte]	2	1	4	A, CY ← A＋(HL＋byte)＋CY	×	×	×
		A, ES:[HL+byte]	3	2	5	A,CY ← A＋((ES, HL)＋byte)＋CY	×	×	×
		A, [HL+B]	2	1	4	A, CY ← A＋(HL＋B)＋CY	×	×	×
		A, ES:[HL+B]	3	2	5	A,CY ← A＋((ES, HL)＋B)＋CY	×	×	×
		A, [HL+C]	2	1	4	A, CY ← A＋(HL＋C)＋CY	×	×	×
		A, ES:[HL+C]	3	2	5	A,CY ← A＋((ES, HL)＋C)＋CY	×	×	×
	SUB	A, #byte	2	1	－	A, CY ← A－byte	×	×	×
		saddr, #byte	3	2	－	(saddr), CY ← (saddr) －byte	×	×	×
		A, r[注3]	2	1	－	A, CY ← A－r	×	×	×
		r, A	2	1	－	r, CY ← r－A	×	×	×
		A, !addr16	3	1	4	A, CY ← A－(addr16)	×	×	×
		A, ES:!addr16	4	2	5	A, CY ← A－(ES, addr16)	×	×	×
		A, saddr	2	1	－	A, CY ← A－(saddr)	×	×	×
		A, [HL]	1	1	4	A, CY ← A－(HL)	×	×	×
		A, ES:[HL]	2	2	5	A,CY ← A－(ES, HL)	×	×	×
		A, [HL+byte]	2	1	4	A, CY ← A－(HL＋byte)	×	×	×
		A, ES:[HL+byte]	3	2	5	A,CY ← A－((ES, HL)＋byte)	×	×	×
		A, [HL+B]	2	1	4	A, CY ← A－(HL＋B)	×	×	×
		A, ES:[HL+B]	3	2	5	A,CY ← A－((ES, HL)＋B)	×	×	×
		A, [HL+C]	2	1	4	A, CY ← A－(HL＋C)	×	×	×
		A, ES:[HL+C]	3	2	5	A,CY ← A－((ES, HL)＋C)	×	×	×

注1.　内部RAM領域，SFR領域および拡張SFR領域をアクセスしたとき，またはデータ・アクセスをしないときのCPUクロック（f_{CLK}）数。

2.　コード・フラッシュ・メモリ領域および8ビット命令でデータ・フラッシュ・メモリ領域をアクセスしたときのCPUクロック（f_{CLK}）数。

3.　r = Aを除く。

備考　クロック数は内部ROM（フラッシュ・メモリ）領域にプログラムがある場合です。内部RAM領域から命令フェッチする場合，最大2倍＋3クロックになります。

表5－6　RL78-S2コアのオペレーション一覧（8/17）

命令群	ニモニック	オペランド	バイト	クロック		オペレーション	フラグ		
				注1	注2		Z	AC	CY
8ビット演算	SUBC	A, #byte	2	1	－	A, CY ← A－byte－CY	×	×	×
		saddr, #byte	3	2	－	(saddr), CY ← (saddr)－byte－CY	×	×	×
		A, r[注3]	2	1	－	A, CY ← A－r－CY	×	×	×
		r, A	2	1	－	r, CY ← r－A－CY	×	×	×
		A, !addr16	3	1	4	A, CY ← A－(addr16)－CY	×	×	×
		A, ES:!addr16	4	2	5	A, CY ← A－(ES, addr16)－CY	×	×	×
		A, saddr	2	1	－	A, CY ← A－(saddr)－CY	×	×	×
		A, [HL]	1	1	4	A, CY ← A－(HL)－CY	×	×	×
		A, ES:[HL]	2	2	5	A,CY ← A－(ES, HL)－CY	×	×	×
		A, [HL+byte]	2	1	4	A, CY ← A－(HL＋byte)－CY	×	×	×
		A, ES:[HL+byte]	3	2	5	A,CY ← A－((ES, HL)＋byte)－CY	×	×	×
		A, [HL+B]	2	1	4	A, CY ← A－(HL＋B)－CY	×	×	×
		A, ES:[HL+B]	3	2	5	A,CY ← A－((ES, HL)＋B)－CY	×	×	×
		A, [HL+C]	2	1	4	A, CY ← A－(HL＋C)－CY	×	×	×
		A, ES:[HL+C]	3	2	5	A, CY ← A－((ES:HL)＋C)－CY	×	×	×
	AND	A, #byte	2	1	－	A ← A∧byte	×		
		saddr, #byte	3	2	－	(saddr) ← (saddr)∧byte	×		
		A, r[注3]	2	1	－	A ← A∧r	×		
		r, A	2	1	－	r ← r∧A	×		
		A, !addr16	3	1	4	A ← A∧(addr16)	×		
		A, ES:!addr16	4	2	5	A ← A∧(ES:addr16)	×		
		A, saddr	2	1	－	A ← A∧(saddr)	×		
		A, [HL]	1	1	4	A ← A∧(HL)	×		
		A, ES:[HL]	2	2	5	A ← A∧(ES:HL)	×		
		A, [HL+byte]	2	1	4	A ← A∧(HL＋byte)	×		
		A, ES:[HL+byte]	3	2	5	A ← A∧((ES:HL)＋byte)	×		
		A, [HL+B]	2	1	4	A ← A∧(HL＋B)	×		
		A, ES:[HL+B]	3	2	5	A ← A∧((ES:HL)＋B)	×		
		A, [HL+C]	2	1	4	A ← A∧(HL＋C)	×		
		A, ES:[HL+C]	3	2	5	A ← A∧((ES:HL)＋C)	×		

注1.　内部RAM領域，SFR領域および拡張SFR領域をアクセスしたとき，またはデータ・アクセスをしないときのCPUクロック（f_{CLK}）数。

2.　コード・フラッシュ・メモリ領域および8ビット命令でデータ・フラッシュ・メモリ領域をアクセスしたときのCPUクロック（f_{CLK}）数。

3.　r = Aを除く。

備考　クロック数は内部ROM（フラッシュ・メモリ）領域にプログラムがある場合です。内部RAM領域から命令フェッチする場合，最大2倍＋3クロックになります。

表5－6　RL78-S2コアのオペレーション一覧（9/17）

命令群	ニモニック	オペランド	バイト	クロック		オペレーション	フラグ
				注1	注2		Z AC CY
8ビット演算	OR	A, #byte	2	1	－	A ← AVbyte	×
		saddr, #byte	3	2	－	(saddr) ← (saddr)Vbyte	×
		A, r[注3]	2	1	－	A ← AVr	×
		r, A	2	1	－	r ← rVA	×
		A, !addr16	3	1	4	A ← AV(addr16)	×
		A, ES:!addr16	4	2	5	A ← AV(ES:addr16)	×
		A, saddr	2	1	－	A ← AV(saddr)	×
		A, [HL]	1	1	4	A ← AV(HL)	×
		A, ES:[HL]	2	2	5	A ← AV(ES:HL)	×
		A, [HL+byte]	2	1	4	A ← AV(HL＋byte)	×
		A, ES:[HL+byte]	3	2	5	A ← AV((ES:HL)＋byte)	×
		A, [HL+B]	2	1	4	A ← AV(HL＋B)	×
		A, ES:[HL+B]	3	2	5	A ← AV((ES:HL)＋B)	×
		A, [HL+C]	2	1	4	A ← AV(HL＋C)	×
		A, ES:[HL+C]	3	2	5	A ← AV((ES:HL)＋C)	×
	XOR	A, #byte	2	1	－	A ← A⊻byte	×
		saddr, #byte	3	2	－	(saddr) ← (saddr)⊻byte	×
		A, r[注3]	2	1	－	A ← A⊻r	×
		r, A	2	1	－	r ← r⊻A	×
		A, !addr16	3	1	4	A ← A⊻(addr16)	×
		A, ES:!addr16	4	2	5	A ← A⊻(ES:addr16)	×
		A, saddr	2	1	－	A ← A⊻(saddr)	×
		A, [HL]	1	1	4	A ← A⊻(HL)	×
		A, ES:[HL]	2	2	5	A ← A⊻(ES:HL)	×
		A, [HL+byte]	2	1	4	A ← A⊻(HL＋byte)	×
		A, ES:[HL+byte]	3	2	5	A ← A⊻((ES:HL)＋byte)	×
		A, [HL+B]	2	1	4	A ← A⊻(HL＋B)	×
		A, ES:[HL+B]	3	2	5	A ← A⊻((ES:HL)＋B)	×
		A, [HL+C]	2	1	4	A ← A⊻(HL＋C)	×
		A, ES:[HL+C]	3	2	5	A ← A⊻((ES:HL)＋C)	×

注1.　内部RAM領域，SFR領域および拡張SFR領域をアクセスしたとき，またはデータ・アクセスをしないときのCPUクロック（f_{CLK}）数。

2.　コード・フラッシュ・メモリ領域および8ビット命令でデータ・フラッシュ・メモリ領域をアクセスしたときのCPUクロック（f_{CLK}）数。

3.　r = Aを除く。

備考　クロック数は内部ROM（フラッシュ・メモリ）領域にプログラムがある場合です。内部RAM領域から命令フェッチする場合，最大2倍＋3クロックになります。

表5－6　RL78-S2コアのオペレーション一覧（10/17）

命令群	ニモニック	オペランド	バイト	クロック		オペレーション	フラグ		
				注1	注2		Z	AC	CY
8ビット演算	CMP	A, #byte	2	1	－	A－byte	×	×	×
		!addr16, #byte	4	1	4	(addr16)－byte	×	×	×
		ES:!addr16, #byte	5	2	5	(ES:addr16)－byte	×	×	×
		saddr, #byte	3	1	－	(saddr)－byte	×	×	×
		A, r[注3]	2	1	－	A－r	×	×	×
		r, A	2	1	－	r－A	×	×	×
		A, !addr16	3	1	4	A－(addr16)	×	×	×
		A, ES:!addr16	4	2	5	A－(ES:addr16)	×	×	×
		A, saddr	2	1	－	A－(saddr)	×	×	×
		A, [HL]	1	1	4	A－(HL)	×	×	×
		A, ES:[HL]	2	2	5	A－(ES:HL)	×	×	×
		A, [HL+byte]	2	1	4	A－(HL＋byte)	×	×	×
		A, ES:[HL+byte]	3	2	5	A－((ES:HL)＋byte)	×	×	×
		A, [HL+B]	2	1	4	A－(HL＋B)	×	×	×
		A, ES:[HL+B]	3	2	5	A－((ES:HL)＋B)	×	×	×
		A, [HL+C]	2	1	4	A－(HL＋C)	×	×	×
		A, ES:[HL+C]	3	2	5	A－((ES:HL)＋C)	×	×	×
	CMP0	A	1	1	－	A－00H	×	0	0
		X	1	1	－	X－00H	×	0	0
		B	1	1	－	B－00H	×	0	0
		C	1	1	－	C－00H	×	0	0
		!addr16	3	1	4	(addr16)－00H	×	0	0
		ES:!addr16	4	2	5	(ES:addr16)－00H	×	0	0
		saddr	2	1	－	(saddr)－00H	×	0	0
	CMPS	X, [HL+byte]	3	1	4	X－(HL＋byte)	×	×	×
		X, ES:[HL+byte]	4	2	5	X－((ES:HL)＋byte)	×	×	×

注1.　内部RAM領域，SFR領域および拡張SFR領域をアクセスしたとき，またはデータ・アクセスをしないときのCPUクロック（f_{CLK}）数。

2.　コード・フラッシュ・メモリ領域および8ビット命令でデータ・フラッシュ・メモリ領域をアクセスしたときのCPUクロック（f_{CLK}）数。

3.　r = Aを除く。

備考　クロック数は内部ROM（フラッシュ・メモリ）領域にプログラムがある場合です。内部RAM領域から命令フェッチする場合，最大2倍＋3クロックになります。

表5－6　RL78-S2コアのオペレーション一覧（11/17）

命令群	ニモニック	オペランド	バイト	クロック		オペレーション	フラグ		
				注1	注2		Z	AC	CY
16ビット演算	ADDW	AX, #word	3	1	－	AX, CY ← AX＋word	×	×	×
		AX, AX	1	1	－	AX, CY ← AX＋AX	×	×	×
		AX, BC	1	1	－	AX, CY ← AX＋BC	×	×	×
		AX, DE	1	1	－	AX, CY ← AX＋DE	×	×	×
		AX, HL	1	1	－	AX, CY ← AX＋HL	×	×	×
		AX, !addr16	3	1	4	AX, CY ← AX＋(addr16)	×	×	×
		AX, ES:!addr16	4	2	5	AX, CY ← AX＋(ES:addr16)	×	×	×
		AX, saddrp	2	1	－	AX, CY ← AX＋(saddrp)	×	×	×
		AX, [HL+byte]	3	1	4	AX, CY ← AX＋(HL＋byte)	×	×	×
		AX, ES: [HL+byte]	4	2	5	AX, CY ← AX＋((ES:HL)＋byte)	×	×	×
	SUBW	AX, #word	3	1	－	AX, CY ← AX－word	×	×	×
		AX, BC	1	1	－	AX, CY ← AX－BC	×	×	×
		AX, DE	1	1	－	AX, CY ← AX－DE	×	×	×
		AX, HL	1	1	－	AX, CY ← AX－HL	×	×	×
		AX, !addr16	3	1	4	AX, CY ← AX－(addr16)	×	×	×
		AX, ES:!addr16	4	2	5	AX, CY ← AX－(ES:addr16)	×	×	×
		AX, saddrp	2	1	－	AX, CY ← AX－(saddrp)	×	×	×
		AX, [HL+byte]	3	1	4	AX, CY ← AX－(HL＋byte)	×	×	×
		AX, ES: [HL+byte]	4	2	5	AX, CY ← AX－((ES:HL)＋byte)	×	×	×
	CMPW	AX, #word	3	1	－	AX－word	×	×	×
		AX, BC	1	1	－	AX－BC	×	×	×
		AX, DE	1	1	－	AX－DE	×	×	×
		AX, HL	1	1	－	AX－HL	×	×	×
		AX, !addr16	3	1	4	AX－(addr16)	×	×	×
		AX, ES:!addr16	4	2	5	AX－(ES:addr16)	×	×	×
		AX, saddrp	2	1	－	AX－(saddrp)	×	×	×
		AX, [HL+byte]	3	1	4	AX－(HL＋byte)	×	×	×
		AX, ES: [HL+byte]	4	2	5	AX－((ES:HL)＋byte)	×	×	×
乗算	MULU	X	1	1	－	AX ← A×X			

注1.　内部RAM領域，SFR領域および拡張SFR領域をアクセスしたとき，またはデータ・アクセスをしないときのCPUクロック（f_{CLK}）数。

2.　コード・フラッシュ・メモリ領域および8ビット命令でデータ・フラッシュ・メモリ領域をアクセスしたときのCPUクロック（f_{CLK}）数。

備考　クロック数は内部ROM（フラッシュ・メモリ）領域にプログラムがある場合です。内部RAM領域から命令フェッチする場合，最大2倍＋3クロックになります。

表5−6　RL78-S2コアのオペレーション一覧（12/17）

命令群	ニモニック	オペランド	バイト	クロック		オペレーション	フラグ
				注1	注2		Z AC CY
増減	INC	r	1	1	−	r ← r＋1	× ×
		!addr16	3	2	−	(addr16) ← (addr16)＋1	× ×
		ES:!addr16	4	3	−	(ES, addr16) ← (ES, addr16)＋1	× ×
		saddr	2	2	−	(saddr) ← (saddr)＋1	× ×
		[HL+byte]	3	2	−	(HL＋byte) ← (HL＋byte)＋1	× ×
		ES: [HL+byte]	4	3	−	((ES:HL)＋byte) ← ((ES:HL)＋byte)＋1	× ×
	DEC	r	1	1	−	r ← r−1	× ×
		!addr16	3	2	−	(addr16) ← (addr16)−1	× ×
		ES:!addr16	4	3	−	(ES, addr16) ← (ES, addr16) −1	× ×
		saddr	2	2	−	(saddr) ← (saddr)−1	× ×
		[HL+byte]	3	2	−	(HL＋byte) ← (HL＋byte) −1	× ×
		ES: [HL+byte]	4	3	−	((ES:HL)＋byte) ← ((ES:HL)＋byte) −1	× ×
	INCW	rp	1	1	−	rp ← rp＋1	
		!addr16	3	2	−	(addr16) ← (addr16)＋1	
		ES:!addr16	4	3	−	(ES, addr16) ← (ES, addr16)＋1	
		saddrp	2	2	−	(saddrp) ← (saddrp)＋1	
		[HL+byte]	3	2	−	(HL＋byte) ← (HL＋byte)＋1	
		ES: [HL+byte]	4	3	−	((ES:HL)＋byte) ← ((ES:HL)＋byte)＋1	
	DECW	rp	1	1	−	rp ← rp−1	
		!addr16	3	2	−	(addr16) ← (addr16)−1	
		ES:!addr16	4	3	−	(ES, addr16) ← (ES, addr16)−1	
		saddrp	2	2	−	(saddrp) ← (saddrp)−1	
		[HL+byte]	3	2	−	(HL＋byte) ← (HL＋byte) −1	
		ES: [HL+byte]	4	3	−	((ES:HL)＋byte) ← ((ES:HL)＋byte) −1	
シフト	SHR	A, cnt	2	1	−	(CY ← A_0, A_{m-1} ← A_m, A_7 ← 0) ×cnt	×
	SHRW	AX, cnt	2	1	−	(CY ← AX_0, AX_{m-1} ← AX_m, AX_{15} ← 0) ×cnt	×
	SHL	A, cnt	2	1	−	(CY ← A_7, A_m ← A_{m-1}, A_0 ← 0) ×cnt	×
		B, cnt	2	1	−	(CY ← B_7, B_m ← B_{m-1}, B_0 ← 0) ×cnt	×
		C, cnt	2	1	−	(CY ← C_7, C_m ← C_{m-1}, C_0 ← 0) ×cnt	×
	SHLW	AX, cnt	2	1	−	(CY ← AX_{15}, AX_m ← AX_{m-1}, AX_0 ← 0) ×cnt	×
		BC, cnt	2	1	−	(CY ← BC_{15}, BC_m ← BC_{m-1}, BC_0 ← 0) ×cnt	×
	SAR	A, cnt	2	1	−	(CY ← A_0, A_{m-1} ← A_m, A_7 ← A_7) ×cnt	×
	SARW	AX, cnt	2	1	−	(CY ← AX_0, AX_{m-1} ← AX_m, AX_{15} ← AX_{15}) ×cnt	×

注1.　内部RAM領域，SFR領域および拡張SFR領域をアクセスしたとき，またはデータ・アクセスをしないときのCPUクロック（f_{CLK}）数。

2.　コード・フラッシュ・メモリ領域および8ビット命令でデータ・フラッシュ・メモリ領域をアクセスしたときのCPUクロック（f_{CLK}）数。

備考1.　クロック数は内部ROM（フラッシュ・メモリ）領域にプログラムがある場合です。内部RAM領域から命令フェッチする場合，最大4倍＋6クロックになります。

2.　cntはビット・シフト数です。

表5－6　RL78-S2コアのオペレーション一覧（13/17）

命令群	ニモニック	オペランド	バイト	クロック		オペレーション	フラグ
				注1	注2		Z AC CY
ローテート	ROR	A, 1	2	1	－	(CY, A_7 ← A_0, A_{m-1} ← A_m) ×1	×
	ROL	A, 1	2	1	－	(CY, A_0 ← A_7, A_{m+1} ← A_m) ×1	×
	RORC	A, 1	2	1	－	(CY ← A_0, A_7 ← CY, A_{m-1} ← A_m) ×1	×
	ROLC	A, 1	2	1	－	(CY ← A_7, A_0 ← CY, A_{m+1} ← A_m) ×1	×
	ROLWC	AX,1	2	1	－	(CY ← AX_{15}, AX_0 ← CY, AX_{m+1} ← AX_m) ×1	×
		BC,1	2	1	－	(CY ← BC_{15}, BC_0 ← CY, BC_{m+1} ← BC_m) ×1	×
ビット操作	MOV1	CY, A.bit	2	1	－	CY ← A.bit	×
		A.bit, CY	2	1	－	A.bit ← CY	
		CY, PSW.bit	3	1	－	CY ← PSW.bit	×
		PSW.bit, CY	3	4	－	PSW.bit ← CY	× ×
		CY, saddr.bit	3	1	－	CY ← (saddr).bit	×
		saddr.bit, CY	3	2	－	(saddr).bit ← CY	
		CY, sfr.bit	3	1	－	CY ← sfr.bit	×
		sfr.bit, CY	3	2	－	sfr.bit ← CY	
		CY,[HL].bit	2	1	4	CY ← (HL).bit	×
		[HL].bit, CY	2	2	－	(HL).bit ← CY	
		CY, ES:[HL].bit	3	2	5	CY ← (ES, HL).bit	×
		ES:[HL].bit, CY	3	3	－	(ES, HL).bit ← CY	
	AND1	CY, A.bit	2	1	－	CY ← CY∧A.bit	×
		CY, PSW.bit	3	1	－	CY ← CY∧PSW.bit	×
		CY, saddr.bit	3	1	－	CY ← CY∧(saddr).bit	×
		CY, sfr.bit	3	1	－	CY ← CY∧sfr.bit	×
		CY,[HL].bit	2	1	4	CY ← CY∧(HL).bit	×
		CY, ES:[HL].bit	3	2	5	CY ← CY∧(ES, HL).bit	×
	OR1	CY, A.bit	2	1	－	CY ← CY∨A.bit	×
		CY, PSW.bit	3	1	－	CY ← CY∨PSW.bit	×
		CY, saddr.bit	3	1	－	CY ← CY∨(saddr).bit	×
		CY, sfr.bit	3	1	－	CY ← CY∨sfr.bit	×
		CY, [HL].bit	2	1	4	CY ← CY∨(HL).bit	×
		CY, ES:[HL].bit	3	2	5	CY ← CY∨(ES, HL).bit	×

注1.　内部RAM領域，SFR領域および拡張SFR領域をアクセスしたとき，またはデータ・アクセスをしないときのCPUクロック（f_{CLK}）数。

2.　コード・フラッシュ・メモリ領域および8ビット命令でデータ・フラッシュ・メモリ領域をアクセスしたときのCPUクロック（f_{CLK}）数。

備考　クロック数は内部ROM（フラッシュ・メモリ）領域にプログラムがある場合です。内部RAM領域から命令フェッチする場合，最大2倍＋3クロックになります。

表5－6　RL78-S2コアのオペレーション一覧（14/17）

命令群	ニモニック	オペランド	バイト	クロック		オペレーション	フラグ
				注1	注2		Z AC CY
ビット操作	XOR1	CY, A.bit	2	1	－	CY ← CY￥A.bit	×
		CY, PSW.bit	3	1	－	CY ← CY￥PSW.bit	×
		CY, saddr.bit	3	1	－	CY ← CY￥(saddr).bit	×
		CY, sfr.bit	3	1	－	CY ← CY￥sfr.bit	×
		CY, [HL].bit	2	1	4	CY ← CY￥(HL).bit	×
		CY, ES:[HL].bit	3	2	5	CY ← CY￥(ES, HL).bit	×
	SET1	A.bit	2	1	－	A.bit ← 1	
		PSW.bit	3	4	－	PSW.bit ← 1	× × ×
		!addr16.bit	4	2	－	(addr16).bit ← 1	
		ES:!addr16.bit	5	3	－	(ES, addr16).bit ← 1	
		saddr.bit	3	2	－	(saddr).bit ← 1	
		sfr.bit	3	2	－	sfr.bit ← 1	
		[HL].bit	2	2	－	(HL).bit ← 1	
		ES:[HL].bit	3	3	－	(ES, HL).bit ← 1	
	CLR1	A.bit	2	1	－	A.bit ← 0	
		PSW.bit	3	4	－	PSW.bit ← 0	× × ×
		!addr16.bit	4	2	－	(addr16).bit ← 0	
		ES:!addr16.bit	5	3	－	(ES, addr16).bit ← 0	
		saddr.bit	3	2	－	(saddr.bit) ← 0	
		sfr.bit	3	2	－	sfr.bit ← 0	
		[HL].bit	2	2	－	(HL).bit ← 0	
		ES:[HL].bit	3	3	－	(ES, HL).bit ← 0	
	SET1	CY	2	1	－	CY ← 1	1
	CLR1	CY	2	1	－	CY ← 0	0
	NOT1	CY	2	1	－	CY ← CY	×

注1.　内部RAM領域，SFR領域および拡張SFR領域をアクセスしたとき，またはデータ・アクセスをしないときのCPUクロック（f_{CLK}）数。

2.　コード・フラッシュ・メモリ領域および8ビット命令でデータ・フラッシュ・メモリ領域をアクセスしたときのCPUクロック（f_{CLK}）数。

備考　クロック数は内部ROM（フラッシュ・メモリ）領域にプログラムがある場合です。内部RAM領域から命令フェッチする場合，最大2倍＋3クロックになります。

表5－6　RL78-S2コアのオペレーション一覧（15/17）

命令群	ニモニック	オペランド	バイト	クロック		オペレーション	フラグ
				注1	注2		Z AC CY
コール・リターン	CALL	rp	2	3	－	(SP－2) ← (PC＋2)S, (SP－3) ← (PC＋2)H, (SP－4) ← (PC＋2)L, PC ← CS, rp, SP ← SP－4	
		$!addr20	3	3	－	(SP－2) ← (PC＋3)S, (SP－3) ← (PC＋3)H, (SP－4) ← (PC＋3)L, PC ← PC＋3＋jdisp16, SP ← SP－4	
		!addr16	3	3	－	(SP－2) ← (PC＋3)S, (SP－3) ← (PC＋3)H, (SP－4) ← (PC＋3)L, PC ← 0000, addr16, SP ← SP－4	
		!!addr20	4	3	－	(SP－2) ← (PC＋4)S, (SP－3) ← (PC＋4)H, (SP－4) ← (PC＋4)L, PC ← addr20, SP ← SP－4	
	CALLT	[addr5]	2	5	－	(SP－2) ← (PC＋2)S , (SP－3) ← (PC＋2)H, (SP－4) ← (PC＋2)L , PCS ← 0000, PCH ← (0000, addr5＋1), PCL ← (0000, addr5), SP ← SP－4	
	BRK	－	2	5	－	(SP－1) ← PSW, (SP－2) ← (PC＋2)S, (SP－3) ← (PC＋2)H, (SP－4) ← (PC＋2)L, PCS ← 0000, PCH ← (0007FH), PCL ← (0007EH), SP ← SP－4, IE ← 0	
	RET	－	1	6	－	PCL ← (SP), PCH ← (SP＋1), PCS ← (SP＋2), SP ← SP＋4	
	RETI	－	2	6	－	PCL ← (SP), PCH ← (SP＋1), PCS ← (SP＋2), PSW ← (SP＋3), SP ← SP＋4	R R R
	RETB	－	2	6	－	PCL ← (SP), PCH ← (SP＋1), PCS ← (SP＋2), PSW ← (SP＋3), SP ← SP＋4	R R R

注1.　内部RAM領域，SFR領域および拡張SFR領域をアクセスしたとき，またはデータ・アクセスをしないときのCPUクロック（f_{CLK}）数。

2.　コード・フラッシュ・メモリ領域および8ビット命令でデータ・フラッシュ・メモリ領域をアクセスしたときのCPUクロック（f_{CLK}）数。

備考　クロック数は内部ROM（フラッシュ・メモリ）領域にプログラムがある場合です。内部RAM領域から命令フェッチする場合，最大2倍＋3クロックになります。

表5－6　RL78-S2コアのオペレーション一覧（16/17）

命令群	ニモニック	オペランド	バイト	クロック		オペレーション	フラグ
				注1	注2		Z AC CY
スタック操作	PUSH	PSW	2	1	－	(SP－1) ← PSW, (SP－2) ← 00H, SP ← SP－2	
		rp	1	1	－	(SP－1) ← rp_H, (SP－2) ← rp_L, SP ← SP－2	
	POP	PSW	2	3	－	PSW ← (SP＋1), SP ← SP＋2	R R R
		rp	1	1	－	rp_L ← (SP), rp_H ← (SP＋1), SP ← SP＋2	
	MOVW	SP, #word	4	1	－	SP ← word	
		SP, AX	2	1	－	SP ← AX	
		AX, SP	2	1	－	AX ← SP	
		HL, SP	3	1	－	HL ← SP	
		BC, SP	3	1	－	BC ← SP	
		DE, SP	3	1	－	DE ← SP	
	ADDW	SP, #byte	2	1	－	SP ← SP＋byte	
	SUBW	SP, #byte	2	1	－	SP ← SP－byte	
無条件分岐	BR	AX	2	3	－	PC ← CS, AX	
		$addr20	2	3	－	PC ← PC＋2＋jdisp8	
		$!addr20	3	3	－	PC ← PC＋3＋jdisp16	
		!addr16	3	3	－	PC ← 0000, addr16	
		!!addr20	4	3	－	PC ← addr20	
条件付き分岐	BC	$addr20	2	2/4[注3]	－	PC ← PC＋2＋jdisp8 if CY = 1	
	BNC	$addr20	2	2/4[注3]	－	PC ← PC＋2＋jdisp8 if CY = 0	
	BZ	$addr20	2	2/4[注3]	－	PC ← PC＋2＋jdisp8 if Z = 1	
	BNZ	$addr20	2	2/4[注3]	－	PC ← PC＋2＋jdisp8 if Z = 0	
	BH	$addr20	3	2/4[注3]	－	PC ← PC+3+jdisp8 if (Z∨CY)=0	
	BNH	$addr20	3	2/4[注3]	－	PC ← PC+3+jdisp8 if (Z∨CY)=1	
	BT	saddr.bit, $addr20	4	3/5[注3]	－	PC ← PC＋4＋jdisp8 if (saddr).bit = 1	
		sfr.bit, $addr20	4	3/5[注3]	－	PC ← PC＋4＋jdisp8 if sfr.bit = 1	
		A.bit, $addr20	3	3/5[注3]	－	PC ← PC＋3＋jdisp8 if A.bit = 1	
		PSW.bit, $addr20	4	3/5[注3]	－	PC ← PC＋4＋jdisp8 if PSW.bit = 1	
		[HL].bit, $addr20	3	3/5[注3]	6/7	PC ← PC＋3＋jdisp8 if (HL).bit = 1	
		ES:[HL].bit, $addr20	4	4/6[注3]	7/8	PC ← PC＋4＋jdisp8 if (ES, HL).bit = 1	

注1.　内部RAM領域，SFR領域および拡張SFR領域をアクセスしたとき，またはデータ・アクセスをしないときのCPUクロック（f_{CLK}）数。

2.　コード・フラッシュ・メモリ領域および8ビット命令でデータ・フラッシュ・メモリ領域をアクセスしたときのCPUクロック（f_{CLK}）数。

3.　クロック数は“条件不成立時／条件成立時”を表しています。

備考　クロック数は内部ROM（フラッシュ・メモリ）領域にプログラムがある場合です。内部RAM領域から命令フェッチする場合，最大2倍＋3クロックになります。

表5－6　RL78-S2コアのオペレーション一覧（17/17）

命令群	ニモニック	オペランド	バイト	クロック		オペレーション	フラグ
				注1	注2		Z AC CY
条件付き分岐	BF	saddr.bit, $addr20	4	3/5[注3]	－	PC ← PC＋4＋jdisp8 if (saddr).bit = 0	
		sfr.bit, $addr20	4	3/5[注3]	－	PC ← PC＋4＋jdisp8 if sfr.bit = 0	
		A.bit, $addr20	3	3/5[注3]	－	PC ← PC＋3＋jdisp8 if A.bit = 0	
		PSW.bit, $addr20	4	3/5[注3]	－	PC ← PC＋4＋jdisp8 if PSW.bit = 0	
		[HL].bit, $addr20	3	3/5[注3]	6/7	PC ← PC＋3＋jdisp8 if (HL).bit = 0	
		ES:[HL].bit, $addr20	4	4/6[注3]	7/8	PC ← PC＋4＋jdisp8 if (ES, HL).bit = 0	
	BTCLR	saddr.bit, $addr20	4	3/5[注3]	－	PC ← PC＋4＋jdisp8 if (saddr).bit = 1 then reset (saddr).bit	
		sfr.bit, $addr20	4	3/5[注3]	－	PC ← PC＋4＋jdisp8 if sfr.bit = 1 then reset sfr.bit	
		A.bit, $addr20	3	3/5[注3]	－	PC ← PC＋3＋jdisp8 if A.bit = 1 then reset A.bit	
		PSW.bit, $addr20	4	3/5[注3]	－	PC ← PC＋4＋jdisp8 if PSW.bit = 1 then reset PSW.bit	× × ×
		[HL].bit, $addr20	3	3/5[注3]	－	PC ← PC＋3＋jdisp8 if (HL).bit = 1 then reset (HL).bit	
		ES:[HL].bit, $addr20	4	4/6[注3]	－	PC ← PC＋4＋jdisp8 if (ES, HL).bit = 1 then reset (ES, HL).bit	
条件付きスキップ	SKC	－	2	1	－	Next instruction skip if CY = 1	
	SKNC	－	2	1	－	Next instruction skip if CY = 0	
	SKZ	－	2	1	－	Next instruction skip if Z = 1	
	SKNZ	－	2	1	－	Next instruction skip if Z = 0	
	SKH	－	2	1	－	Next instruction skip if (Z∨CY)=0	
	SKNH	－	2	1	－	Next instruction skip if (Z∨CY)=1	
CPU制御	SEL[注4]	RBn	2	1	－	RBS[1:0] ← n	
	NOP	－	1	1	－	No Operation	
	EI	－	3	4	－	IE ← 1(Enable Interrupt)	
	DI	－	3	4	－	IE ← 0(Disable Interrupt)	
	HALT	－	2	3	－	Set HALT Mode	
	STOP	－	2	3	－	Set STOP Mode	

注1.　内部RAM領域，SFR領域および拡張SFR領域をアクセスしたとき，またはデータ・アクセスをしないときのCPUクロック（f_{CLK}）数。

2.　コード・フラッシュ・メモリ領域および8ビット命令でデータ・フラッシュ・メモリ領域をアクセスしたときのCPUクロック（f_{CLK}）数。

3.　クロック数は“条件不成立時／条件成立時”を表しています。

4.　nはレジスタ・バンク番号です（n = 0-3）。

備考　クロック数は内部ROM（フラッシュ・メモリ）領域にプログラムがある場合です。内部RAM領域から命令フェッチする場合，最大2倍＋3クロックになります。

5.5.3　RL78-S3コアのオペレーション一覧

表5－7　RL78-S3コアのオペレーション一覧（1/18）

命令群	ニモニック	オペランド	バイト	クロック		オペレーション	フラグ
				注1	注2		Z AC CY
8ビット・データ転送	MOV	r, #byte	2	1	－	r ← byte	
		PSW, #byte	3	3	－	PSW ← byte	× × ×
		CS, #byte	3	1	－	CS ← byte	
		ES, #byte	2	1	－	ES ← byte	
		!addr16, #byte	4	1	－	(addr16) ← byte	
		ES:!addr16, #byte	5	2	－	(ES, addr16) ← byte	
		saddr, #byte	3	1	－	(saddr) ← byte	
		sfr, #byte	3	1	－	sfr ← byte	
		[DE+byte], #byte	3	1	－	(DE＋byte) ← byte	
		ES:[DE+byte],#byte	4	2	－	((ES, DE)＋byte) ← byte	
		[HL+byte], #byte	3	1	－	(HL＋byte) ← byte	
		ES:[HL+byte],#byte	4	2	－	((ES, HL)＋byte) ← byte	
		[SP+byte], #byte	3	1	－	(SP＋byte) ← byte	
		word[B], #byte	4	1	－	(B＋word) ← byte	
		ES:word[B], #byte	5	2	－	((ES, B)＋word) ← byte	
		word[C], #byte	4	1	－	(C＋word) ← byte	
		ES:word[C], #byte	5	2	－	((ES, C)＋word) ← byte	
		word[BC], #byte	4	1	－	(BC＋word) ← byte	
		ES:word[BC], #byte	5	2	－	((ES, BC)＋word) ← byte	
		A, r[注3]	1	1	－	A ← r	
		r, A[注3]	1	1	－	r ← A	
		A, PSW	2	1	－	A ← PSW	
		PSW, A	2	3	－	PSW ← A	× × ×
		A, CS	2	1	－	A ← CS	
		CS, A	2	1	－	CS ← A	
		A, ES	2	1	－	A ← ES	
		ES, A	2	1	－	ES ← A	
		A, !addr16	3	1	4	A ← (addr16)	
		A, ES:!addr16	4	2	5	A ← (ES, addr16)	
		!addr16, A	3	1	－	(addr16) ← A	
		ES:!addr16, A	4	2	－	(ES, addr16) ← A	
		A, saddr	2	1	－	A ← (saddr)	
		saddr, A	2	1	－	(saddr) ← A	

注1.　内部RAM領域，SFR領域および拡張SFR領域をアクセスしたとき，またはデータ・アクセスをしないときのCPUクロック（f_{CLK}）数。

2.　コード・フラッシュ・メモリ領域および8ビット命令でデータ・フラッシュ・メモリ領域をアクセスしたときのCPUクロック（f_{CLK}）数。

3.　r = Aを除く。

備考　クロック数は内部ROM（フラッシュ・メモリ）領域にプログラムがある場合です。内部RAM領域から命令フェッチする場合，最大2倍＋3クロックになります。

表5－7　RL78-S3コアのオペレーション一覧（2/18）

命令群	ニモニック	オペランド	バイト	クロック		オペレーション	フラグ
				注1	注2		Z AC CY
8ビット・データ転送	MOV	A, sfr	2	1	－	A ← sfr	
		sfr, A	2	1	－	sfr ← A	
		A, [DE]	1	1	4	A ← (DE)	
		[DE], A	1	1	－	(DE) ← A	
		A, ES:[DE]	2	2	5	A ← (ES, DE)	
		ES:[DE], A	2	2	－	(ES, DE) ← A	
		A, [HL]	1	1	4	A ← (HL)	
		[HL], A	1	1	－	(HL) ← A	
		A, ES:[HL]	2	2	5	A ← (ES, HL)	
		ES:[HL], A	2	2	－	(ES, HL) ← A	
		A, [DE+byte]	2	1	4	A ← (DE＋byte)	
		[DE+byte], A	2	1	－	(DE＋byte) ← A	
		A, ES:[DE+byte]	3	2	5	A ← ((ES, DE)＋byte)	
		ES:[DE+byte], A	3	2	－	((ES, DE)＋byte) ← A	
		A, [HL+byte]	2	1	4	A ← (HL＋byte)	
		[HL+byte], A	2	1	－	(HL＋byte) ← A	
		A, ES:[HL+byte]	3	2	5	A ← ((ES, HL)＋byte)	
		ES:[HL+byte], A	3	2	－	((ES, HL)＋byte) ← A	
		A, [SP+byte]	2	1	－	A ← (SP＋byte)	
		[SP+byte], A	2	1	－	(SP＋byte) ← A	
		A, word[B]	3	1	4	A ← (B＋word)	
		word[B], A	3	1	－	(B＋word) ← A	
		A, ES:word[B]	4	2	5	A ← ((ES, B)＋word)	
		ES:word[B], A	4	2	－	((ES, B)＋word) ← A	
		A, word[C]	3	1	4	A ← (C＋word)	
		word[C], A	3	1	－	(C＋word) ← A	
		A, ES:word[C]	4	2	5	A ← ((ES, C)＋word)	
		ES:word[C], A	4	2	－	((ES, C)＋word) ← A	
		A, word[BC]	3	1	4	A ← (BC＋word)	
		word[BC], A	3	1	－	(BC＋word) ← A	
		A, ES:word[BC]	4	2	5	A ← ((ES, BC)＋word)	
		ES:word[BC], A	4	2	－	((ES, BC)＋word) ← A	

注1.　内部RAM領域，SFR領域および拡張SFR領域をアクセスしたとき，またはデータ・アクセスをしないときのCPUクロック（f_{CLK}）数。

2.　コード・フラッシュ・メモリ領域および8ビット命令でデータ・フラッシュ・メモリ領域をアクセスしたときのCPUクロック（f_{CLK}）数。

備考　クロック数は内部ROM（フラッシュ・メモリ）領域にプログラムがある場合です。内部RAM領域から命令フェッチする場合，最大2倍＋3クロックになります。

表5－7　RL78-S3コアのオペレーション一覧（3/18）

命令群	ニモニック	オペランド	バイト	クロック		オペレーション	フラグ
				注1	注2		Z AC CY
8ビット・データ転送	MOV	A, [HL+B]	2	1	4	A ← (HL＋B)	
		[HL+B], A	2	1	－	(HL＋B) ← A	
		A, ES:[HL+B]	3	2	5	A ← ((ES, HL)＋B)	
		ES:[HL+B], A	3	2	－	((ES, HL)＋B) ← A	
		A, [HL+C]	2	1	4	A ← (HL＋C)	
		[HL+C], A	2	1	－	(HL＋C) ← A	
		A, ES:[HL+C]	3	2	5	A ← ((ES, HL)＋C)	
		ES:[HL+C], A	3	2	－	((ES, HL)＋C) ← A	
		X, !addr16	3	1	4	X ← (addr16)	
		X, ES:!addr16	4	2	5	X ← (ES, addr16)	
		X, saddr	2	1	－	X ← (saddr)	
		B, !addr16	3	1	4	B ← (addr16)	
		B, ES:!addr16	4	2	5	B ← (ES, addr16)	
		B, saddr	2	1	－	B ← (saddr)	
		C, !addr16	3	1	4	C ← (addr16)	
		C, ES:!addr16	4	2	5	C ← (ES, addr16)	
		C, saddr	2	1	－	C ← (saddr)	
		ES, saddr	3	1	－	ES ← (saddr)	
	XCH	A, r[注3]	1 (r = X) 2 (r = X以外)	1	－	A ⟷ r	
		A, !addr16	4	2	－	A ⟷ (addr16)	
		A, ES:!addr16	5	3	－	A ⟷ (ES, addr16)	
		A, saddr	3	2	－	A ⟷ (saddr)	
		A, sfr	3	2	－	A ⟷ sfr	
		A, [DE]	2	2	－	A ⟷ (DE)	
		A, ES:[DE]	3	3	－	A ⟷ (ES, DE)	
		A, [HL]	2	2	－	A ⟷ (HL)	
		A, ES:[HL]	3	3	－	A ⟷ (ES, HL)	
		A, [DE+byte]	3	2	－	A ⟷ (DE＋byte)	
		A, ES:[DE+byte]	4	3	－	A ⟷ ((ES, DE)＋byte)	
		A, [HL+byte]	3	2	－	A ⟷ (HL＋byte)	
		A, ES:[HL+byte]	4	3	－	A ⟷ ((ES, HL)＋byte)	

注1.　内部RAM領域，SFR領域および拡張SFR領域をアクセスしたとき，またはデータ・アクセスをしないときのCPUクロック（f_{CLK}）数。

2.　コード・フラッシュ・メモリ領域および8ビット命令でデータ・フラッシュ・メモリ領域をアクセスしたときのCPUクロック（f_{CLK}）数。

3.　r = Aを除く。

備考　クロック数は内部ROM（フラッシュ・メモリ）領域にプログラムがある場合です。内部RAM領域から命令フェッチする場合，最大2倍＋3クロックになります。

表5−7　RL78-S3コアのオペレーション一覧（4/18）

命令群	ニモニック	オペランド	バイト	クロック		オペレーション	フラグ
				注1	注2		Z AC CY
8ビット・データ転送	XCH	A, [HL+B]	2	2	−	A ⟷ (HL＋B)	
		A, ES:[HL+B]	3	3	−	A ⟷ ((ES, HL)＋B)	
		A, [HL+C]	2	2	−	A ⟷ (HL＋C)	
		A, ES:[HL+C]	3	3	−	A ⟷ ((ES, HL)＋C)	
	ONEB	A	1	1	−	A ← 01H	
		X	1	1	−	X ← 01H	
		B	1	1	−	B ← 01H	
		C	1	1	−	C ← 01H	
		!addr16	3	1	−	(addr16) ← 01H	
		ES:!addr16	4	2	−	(ES, addr16) ← 01H	
		saddr	2	1	−	(saddr) ← 01H	
	CLRB	A	1	1	−	A ← 00H	
		X	1	1	−	X ← 00H	
		B	1	1	−	B ← 00H	
		C	1	1	−	C ← 00H	
		!addr16	3	1	−	(addr16) ← 00H	
		ES:!addr16	4	2	−	(ES,addr16) ← 00H	
		saddr	2	1	−	(saddr) ← 00H	
	MOVS	[HL+byte], X	3	1	−	(HL＋byte) ← X	× ×
		ES:[HL+byte], X	4	2	−	(ES, HL＋byte) ← X	× ×
16ビット・データ転送	MOVW	rp, #word	3	1	−	rp ← word	
		saddrp, #word	4	1	−	(saddrp) ← word	
		sfrp, #word	4	1	−	sfrp ← word	
		AX, rp[注3]	1	1	−	AX ← rp	
		rp, AX[注3]	1	1	−	rp ← AX	
		AX, !addr16	3	1	4	AX ←（addr16）	
		!addr16, AX	3	1	−	（addr16）← AX	
		AX, ES:!addr16	4	2	5	AX ← (ES, addr16)	
		ES:!addr16, AX	4	2	−	(ES, addr16) ← AX	
		AX, saddrp	2	1	−	AX ← (saddrp)	
		saddrp, AX	2	1	−	(saddrp) ← AX	
		AX, sfrp	2	1	−	AX ← sfrp	
		sfrp, AX	2	1	−	sfrp ← AX	

注1.　内部RAM領域，SFR領域および拡張SFR領域をアクセスしたとき，またはデータ・アクセスをしないときのCPUクロック（f_{CLK}）数。

2.　コード・フラッシュ・メモリ領域および8ビット命令でデータ・フラッシュ・メモリ領域をアクセスしたときのCPUクロック（f_{CLK}）数。

3.　rp = AXを除く。

備考　クロック数は内部ROM（フラッシュ・メモリ）領域にプログラムがある場合です。内部RAM領域から命令フェッチする場合，最大2倍＋3クロックになります。

表5−7　RL78-S3コアのオペレーション一覧（5/18）

命令群	ニモニック	オペランド	バイト	クロック		オペレーション	フラグ
				注1	注2		Z AC CY
16ビット・データ転送	MOVW	AX, [DE]	1	1	4	AX ← (DE)	
		[DE], AX	1	1	−	(DE) ← AX	
		AX, ES:[DE]	2	2	5	AX ← (ES, DE)	
		ES:[DE], AX	2	2	−	(ES, DE) ← AX	
		AX, [HL]	1	1	4	AX ← (HL)	
		[HL], AX	1	1	−	(HL) ← AX	
		AX, ES:[HL]	2	2	5	AX ← (ES, HL)	
		ES:[HL], AX	2	2	−	(ES, HL) ← AX	
		AX, [DE+byte]	2	1	4	AX ← (DE＋byte)	
		[DE+byte], AX	2	1	−	(DE＋byte) ← AX	
		AX, ES:[DE+byte]	3	2	5	AX ← ((ES, DE)＋byte)	
		ES:[DE+byte], AX	3	2	−	((ES, DE)＋byte) ← AX	
		AX, [HL+byte]	2	1	4	AX ← (HL＋byte)	
		[HL+byte], AX	2	1	−	(HL＋byte) ← AX	
		AX, ES:[HL+byte]	3	2	5	AX ← ((ES, HL)＋byte)	
		ES:[HL+byte], AX	3	2	−	((ES, HL)＋byte) ← AX	
		AX, [SP+byte]	2	1	−	AX ← (SP＋byte)	
		[SP+byte], AX	2	1	−	(SP＋byte) ← AX	
		AX, word[B]	3	1	4	AX ← (B＋word)	
		word[B], AX	3	1	−	(B＋word) ← AX	
		AX, ES:word[B]	4	2	5	AX ← ((ES, B)＋word)	
		ES:word[B], AX	4	2	−	((ES, B)＋word) ← AX	
		AX, word[C]	3	1	4	AX ← (C＋word)	
		word[C], AX	3	1	−	(C＋word) ← AX	
		AX, ES:word[C]	4	2	5	AX ← ((ES, C)＋word)	
		ES:word[C], AX	4	2	−	((ES, C)＋word) ← AX	
		AX, word[BC]	3	1	4	AX ← (BC＋word)	
		word[BC], AX	3	1	−	(BC＋word) ← AX	
		AX, ES:word[BC]	4	2	5	AX ← ((ES, BC)＋word)	
		ES:word[BC], AX	4	2	−	((ES, BC)＋word) ← AX	

注1.　内部RAM領域，SFR領域および拡張SFR領域をアクセスしたとき，またはデータ・アクセスをしないときのCPUクロック（f_{CLK}）数。

2.　コード・フラッシュ・メモリ領域および8ビット命令でデータ・フラッシュ・メモリ領域をアクセスしたときのCPUクロック（f_{CLK}）数。

備考　クロック数は内部ROM（フラッシュ・メモリ）領域にプログラムがある場合です。内部RAM領域から命令フェッチする場合，最大2倍＋3クロックになります。

表5－7　RL78-S3コアのオペレーション一覧（6/18）

命令群	ニモニック	オペランド	バイト	クロック		オペレーション	フラグ
				注1	注2		Z AC CY
16ビット・データ転送	MOVW	BC, !addr16	3	1	4	BC ← (addr16)	
		BC, ES:!addr16	4	2	5	BC ← (ES, addr16)	
		DE, !addr16	3	1	4	DE ← (addr16)	
		DE, ES:!addr16	4	2	5	DE ← (ES, addr16)	
		HL, !addr16	3	1	4	HL ← (addr16)	
		HL, ES:!addr16	4	2	5	HL ← (ES, addr16)	
		BC, saddrp	2	1	－	BC ← (saddrp)	
		DE, saddrp	2	1	－	DE ← (saddrp)	
		HL, saddrp	2	1	－	HL ← (saddrp)	
	XCHW	AX, rp[注3]	1	1	－	AX ⟷ rp	
	ONEW	AX	1	1	－	AX ← 0001H	
		BC	1	1	－	BC ← 0001H	
	CLRW	AX	1	1	－	AX ← 0000H	
		BC	1	1	－	BC ← 0000H	
8ビット演算	ADD	A, #byte	2	1	－	A, CY ← A＋byte	× × ×
		saddr, #byte	3	2	－	(saddr), CY ← (saddr)＋byte	× × ×
		A, r[注4]	2	1	－	A, CY ← A＋r	× × ×
		r, A	2	1	－	r, CY ← r＋A	× × ×
		A, !addr16	3	1	4	A, CY ← A＋(addr16)	× × ×
		A, ES:!addr16	4	2	5	A, CY ← A＋(ES, addr16)	× × ×
		A, saddr	2	1	－	A, CY ← A＋(saddr)	× × ×
		A, [HL]	1	1	4	A, CY ← A＋(HL)	× × ×
		A, ES:[HL]	2	2	5	A,CY ← A＋(ES, HL)	× × ×
		A, [HL+byte]	2	1	4	A, CY ← A＋(HL＋byte)	× × ×
		A, ES:[HL+byte]	3	2	5	A,CY ← A＋((ES, HL)＋byte)	× × ×
		A, [HL+B]	2	1	4	A, CY ← A＋(HL＋B)	× × ×
		A, ES:[HL+B]	3	2	5	A,CY ← A＋((ES, HL)＋B)	× × ×
		A, [HL+C]	2	1	4	A, CY ← A＋(HL＋C)	× × ×
		A, ES:[HL+C]	3	2	5	A,CY ← A＋((ES, HL)＋C)	× × ×

注1.　内部RAM領域，SFR領域および拡張SFR領域をアクセスしたとき，またはデータ・アクセスをしないときのCPUクロック（f_{CLK}）数。

2.　コード・フラッシュ・メモリ領域および8ビット命令でデータ・フラッシュ・メモリ領域をアクセスしたときのCPUクロック（f_{CLK}）数。

3.　rp = AXを除く。

4.　r = Aを除く。

備考　クロック数は内部ROM（フラッシュ・メモリ）領域にプログラムがある場合です。内部RAM領域から命令フェッチする場合，最大2倍＋3クロックになります。

表5－7　RL78-S3コアのオペレーション一覧（7/18）

命令群	ニモニック	オペランド	バイト	クロック		オペレーション	フラグ		
				注1	注2		Z	AC	CY
8ビット演算	ADDC	A, #byte	2	1	－	A, CY ← A＋byte＋CY	×	×	×
		saddr, #byte	3	2	－	(saddr), CY ← (saddr)＋byte＋CY	×	×	×
		A, r[注3]	2	1	－	A, CY ← A＋r＋CY	×	×	×
		r, A	2	1	－	r, CY ← r＋A＋CY	×	×	×
		A, !addr16	3	1	4	A, CY ← A＋(addr16)＋CY	×	×	×
		A, ES:!addr16	4	2	5	A, CY ← A＋(ES, addr16)＋CY	×	×	×
		A, saddr	2	1	－	A, CY ← A＋(saddr)＋CY	×	×	×
		A, [HL]	1	1	4	A, CY ← A＋(HL)＋CY	×	×	×
		A, ES:[HL]	2	2	5	A,CY ← A＋(ES, HL)＋CY	×	×	×
		A, [HL+byte]	2	1	4	A, CY ← A＋(HL＋byte)＋CY	×	×	×
		A, ES:[HL+byte]	3	2	5	A,CY ← A＋((ES, HL)＋byte)＋CY	×	×	×
		A, [HL+B]	2	1	4	A, CY ← A＋(HL＋B)＋CY	×	×	×
		A, ES:[HL+B]	3	2	5	A,CY ← A＋((ES, HL)＋B)＋CY	×	×	×
		A, [HL+C]	2	1	4	A, CY ← A＋(HL＋C)＋CY	×	×	×
		A, ES:[HL+C]	3	2	5	A,CY ← A＋((ES, HL)＋C)＋CY	×	×	×
	SUB	A, #byte	2	1	－	A, CY ← A－byte	×	×	×
		saddr, #byte	3	2	－	(saddr), CY ← (saddr) －byte	×	×	×
		A, r[注3]	2	1	－	A, CY ← A－r	×	×	×
		r, A	2	1	－	r, CY ← r－A	×	×	×
		A, !addr16	3	1	4	A, CY ← A－(addr16)	×	×	×
		A, ES:!addr16	4	2	5	A, CY ← A－(ES, addr16)	×	×	×
		A, saddr	2	1	－	A, CY ← A－(saddr)	×	×	×
		A, [HL]	1	1	4	A, CY ← A－(HL)	×	×	×
		A, ES:[HL]	2	2	5	A,CY ← A－(ES, HL)	×	×	×
		A, [HL+byte]	2	1	4	A, CY ← A－(HL＋byte)	×	×	×
		A, ES:[HL+byte]	3	2	5	A,CY ← A－((ES, HL)＋byte)	×	×	×
		A, [HL+B]	2	1	4	A, CY ← A－(HL＋B)	×	×	×
		A, ES:[HL+B]	3	2	5	A,CY ← A－((ES, HL)＋B)	×	×	×
		A, [HL+C]	2	1	4	A, CY ← A－(HL＋C)	×	×	×
		A, ES:[HL+C]	3	2	5	A,CY ← A－((ES, HL)＋C)	×	×	×

注1.　内部RAM領域，SFR領域および拡張SFR領域をアクセスしたとき，またはデータ・アクセスをしないときのCPUクロック（f_{CLK}）数。

2.　コード・フラッシュ・メモリ領域および8ビット命令でデータ・フラッシュ・メモリ領域をアクセスしたときのCPUクロック（f_{CLK}）数。

3.　r = Aを除く。

備考　クロック数は内部ROM（フラッシュ・メモリ）領域にプログラムがある場合です。内部RAM領域から命令フェッチする場合，最大2倍＋3クロックになります。

表5－7　RL78-S3コアのオペレーション一覧（8/18）

命令群	ニモニック	オペランド	バイト	クロック		オペレーション	フラグ		
				注1	注2		Z	AC	CY
8ビット演算	SUBC	A, #byte	2	1	－	A, CY ← A－byte－CY	×	×	×
		saddr, #byte	3	2	－	(saddr), CY ← (saddr)－byte－CY	×	×	×
		A, r[注3]	2	1	－	A, CY ← A－r－CY	×	×	×
		r, A	2	1	－	r, CY ← r－A－CY	×	×	×
		A, !addr16	3	1	4	A, CY ← A－(addr16)－CY	×	×	×
		A, ES:!addr16	4	2	5	A, CY ← A－(ES, addr16)－CY	×	×	×
		A, saddr	2	1	－	A, CY ← A－(saddr)－CY	×	×	×
		A, [HL]	1	1	4	A, CY ← A－(HL)－CY	×	×	×
		A, ES:[HL]	2	2	5	A,CY ← A－(ES, HL)－CY	×	×	×
		A, [HL+byte]	2	1	4	A, CY ← A－(HL＋byte)－CY	×	×	×
		A, ES:[HL+byte]	3	2	5	A,CY ← A－((ES, HL)＋byte)－CY	×	×	×
		A, [HL+B]	2	1	4	A, CY ← A－(HL＋B)－CY	×	×	×
		A, ES:[HL+B]	3	2	5	A,CY ← A－((ES, HL)＋B)－CY	×	×	×
		A, [HL+C]	2	1	4	A, CY ← A－(HL＋C)－CY	×	×	×
		A, ES:[HL+C]	3	2	5	A, CY ← A－((ES:HL)＋C)－CY	×	×	×
	AND	A, #byte	2	1	－	A ← A∧byte	×		
		saddr, #byte	3	2	－	(saddr) ← (saddr)∧byte	×		
		A, r[注3]	2	1	－	A ← A∧r	×		
		r, A	2	1	－	r ← r∧A	×		
		A, !addr16	3	1	4	A ← A∧(addr16)	×		
		A, ES:!addr16	4	2	5	A ← A∧(ES:addr16)	×		
		A, saddr	2	1	－	A ← A∧(saddr)	×		
		A, [HL]	1	1	4	A ← A∧(HL)	×		
		A, ES:[HL]	2	2	5	A ← A∧(ES:HL)	×		
		A, [HL+byte]	2	1	4	A ← A∧(HL＋byte)	×		
		A, ES:[HL+byte]	3	2	5	A ← A∧((ES:HL)＋byte)	×		
		A, [HL+B]	2	1	4	A ← A∧(HL＋B)	×		
		A, ES:[HL+B]	3	2	5	A ← A∧((ES:HL)＋B)	×		
		A, [HL+C]	2	1	4	A ← A∧(HL＋C)	×		
		A, ES:[HL+C]	3	2	5	A ← A∧((ES:HL)＋C)	×		

注1.　内部RAM領域，SFR領域および拡張SFR領域をアクセスしたとき，またはデータ・アクセスをしないときのCPUクロック（f_{CLK}）数。

2.　コード・フラッシュ・メモリ領域および8ビット命令でデータ・フラッシュ・メモリ領域をアクセスしたときのCPUクロック（f_{CLK}）数。

3.　r = Aを除く。

備考　クロック数は内部ROM（フラッシュ・メモリ）領域にプログラムがある場合です。内部RAM領域から命令フェッチする場合，最大2倍＋3クロックになります。

表5－7　RL78-S3コアのオペレーション一覧（9/18）

命令群	ニモニック	オペランド	バイト	クロック		オペレーション	フラグ
				注1	注2		Z AC CY
8ビット演算	OR	A, #byte	2	1	－	A ← AVbyte	×
		saddr, #byte	3	2	－	(saddr) ← (saddr)Vbyte	×
		A, r[注3]	2	1	－	A ← AVr	×
		r, A	2	1	－	r ← rVA	×
		A, !addr16	3	1	4	A ← AV(addr16)	×
		A, ES:!addr16	4	2	5	A ← AV(ES:addr16)	×
		A, saddr	2	1	－	A ← AV(saddr)	×
		A, [HL]	1	1	4	A ← AV(HL)	×
		A, ES:[HL]	2	2	5	A ← AV(ES:HL)	×
		A, [HL+byte]	2	1	4	A ← AV(HL＋byte)	×
		A, ES:[HL+byte]	3	2	5	A ← AV((ES:HL)＋byte)	×
		A, [HL+B]	2	1	4	A ← AV(HL＋B)	×
		A, ES:[HL+B]	3	2	5	A ← AV((ES:HL)＋B)	×
		A, [HL+C]	2	1	4	A ← AV(HL＋C)	×
		A, ES:[HL+C]	3	2	5	A ← AV((ES:HL)＋C)	×
	XOR	A, #byte	2	1	－	A ← A⊻byte	×
		saddr, #byte	3	2	－	(saddr) ← (saddr)⊻byte	×
		A, r[注3]	2	1	－	A ← A⊻r	×
		r, A	2	1	－	r ← r⊻A	×
		A, !addr16	3	1	4	A ← A⊻(addr16)	×
		A, ES:!addr16	4	2	5	A ← A⊻(ES:addr16)	×
		A, saddr	2	1	－	A ← A⊻(saddr)	×
		A, [HL]	1	1	4	A ← A⊻(HL)	×
		A, ES:[HL]	2	2	5	A ← A⊻(ES:HL)	×
		A, [HL+byte]	2	1	4	A ← A⊻(HL＋byte)	×
		A, ES:[HL+byte]	3	2	5	A ← A⊻((ES:HL)＋byte)	×
		A, [HL+B]	2	1	4	A ← A⊻(HL＋B)	×
		A, ES:[HL+B]	3	2	5	A ← A⊻((ES:HL)＋B)	×
		A, [HL+C]	2	1	4	A ← A⊻(HL＋C)	×
		A, ES:[HL+C]	3	2	5	A ← A⊻((ES:HL)＋C)	×

注1.　内部RAM領域，SFR領域および拡張SFR領域をアクセスしたとき，またはデータ・アクセスをしないときのCPUクロック（f_{CLK}）数。

2.　コード・フラッシュ・メモリ領域および8ビット命令でデータ・フラッシュ・メモリ領域をアクセスしたときのCPUクロック（f_{CLK}）数。

3.　r = Aを除く。

備考　クロック数は内部ROM（フラッシュ・メモリ）領域にプログラムがある場合です。内部RAM領域から命令フェッチする場合，最大2倍＋3クロックになります。

表5－7　RL78-S3コアのオペレーション一覧（10/18）

命令群	ニモニック	オペランド	バイト	クロック		オペレーション	フラグ		
				注1	注2		Z	AC	CY
8ビット演算	CMP	A, #byte	2	1	－	A－byte	×	×	×
		!addr16, #byte	4	1	4	(addr16)－byte	×	×	×
		ES:!addr16, #byte	5	2	5	(ES:addr16)－byte	×	×	×
		saddr, #byte	3	1	－	(saddr)－byte	×	×	×
		A, r[注3]	2	1	－	A－r	×	×	×
		r, A	2	1	－	r－A	×	×	×
		A, !addr16	3	1	4	A－(addr16)	×	×	×
		A, ES:!addr16	4	2	5	A－(ES:addr16)	×	×	×
		A, saddr	2	1	－	A－(saddr)	×	×	×
		A, [HL]	1	1	4	A－(HL)	×	×	×
		A, ES:[HL]	2	2	5	A－(ES:HL)	×	×	×
		A, [HL+byte]	2	1	4	A－(HL＋byte)	×	×	×
		A, ES:[HL+byte]	3	2	5	A－((ES:HL)＋byte)	×	×	×
		A, [HL+B]	2	1	4	A－(HL＋B)	×	×	×
		A, ES:[HL+B]	3	2	5	A－((ES:HL)＋B)	×	×	×
		A, [HL+C]	2	1	4	A－(HL＋C)	×	×	×
		A, ES:[HL+C]	3	2	5	A－((ES:HL)＋C)	×	×	×
	CMP0	A	1	1	－	A－00H	×	0	0
		X	1	1	－	X－00H	×	0	0
		B	1	1	－	B－00H	×	0	0
		C	1	1	－	C－00H	×	0	0
		!addr16	3	1	4	(addr16)－00H	×	0	0
		ES:!addr16	4	2	5	(ES:addr16)－00H	×	0	0
		saddr	2	1	－	(saddr)－00H	×	0	0
	CMPS	X, [HL+byte]	3	1	4	X－(HL＋byte)	×	×	×
		X, ES:[HL+byte]	4	2	5	X－((ES:HL)＋byte)	×	×	×

注1.　内部RAM領域，SFR領域および拡張SFR領域をアクセスしたとき，またはデータ・アクセスをしないときのCPUクロック（f_{CLK}）数。

2.　コード・フラッシュ・メモリ領域および8ビット命令でデータ・フラッシュ・メモリ領域をアクセスしたときのCPUクロック（f_{CLK}）数。

3.　r = Aを除く。

備考　クロック数は内部ROM（フラッシュ・メモリ）領域にプログラムがある場合です。内部RAM領域から命令フェッチする場合，最大2倍＋3クロックになります。

表5－7　RL78-S3コアのオペレーション一覧（11/18）

命令群	ニモニック	オペランド	バイト	クロック		オペレーション	フラグ		
				注1	注2		Z	AC	CY
16ビット演算	ADDW	AX, #word	3	1	－	AX, CY ← AX＋word	×	×	×
		AX, AX	1	1	－	AX, CY ← AX＋AX	×	×	×
		AX, BC	1	1	－	AX, CY ← AX＋BC	×	×	×
		AX, DE	1	1	－	AX, CY ← AX＋DE	×	×	×
		AX, HL	1	1	－	AX, CY ← AX＋HL	×	×	×
		AX, !addr16	3	1	4	AX, CY ← AX＋(addr16)	×	×	×
		AX, ES:!addr16	4	2	5	AX, CY ← AX＋(ES:addr16)	×	×	×
		AX, saddrp	2	1	－	AX, CY ← AX＋(saddrp)	×	×	×
		AX, [HL+byte]	3	1	4	AX, CY ← AX＋(HL＋byte)	×	×	×
		AX, ES: [HL+byte]	4	2	5	AX, CY ← AX＋((ES:HL)＋byte)	×	×	×
	SUBW	AX, #word	3	1	－	AX, CY ← AX－word	×	×	×
		AX, BC	1	1	－	AX, CY ← AX－BC	×	×	×
		AX, DE	1	1	－	AX, CY ← AX－DE	×	×	×
		AX, HL	1	1	－	AX, CY ← AX－HL	×	×	×
		AX, !addr16	3	1	4	AX, CY ← AX－(addr16)	×	×	×
		AX, ES:!addr16	4	2	5	AX, CY ← AX－(ES:addr16)	×	×	×
		AX, saddrp	2	1	－	AX, CY ← AX－(saddrp)	×	×	×
		AX, [HL+byte]	3	1	4	AX, CY ← AX－(HL＋byte)	×	×	×
		AX, ES: [HL+byte]	4	2	5	AX, CY ← AX－((ES:HL)＋byte)	×	×	×
	CMPW	AX, #word	3	1	－	AX－word	×	×	×
		AX, BC	1	1	－	AX－BC	×	×	×
		AX, DE	1	1	－	AX－DE	×	×	×
		AX, HL	1	1	－	AX－HL	×	×	×
		AX, !addr16	3	1	4	AX－(addr16)	×	×	×
		AX, ES:!addr16	4	2	5	AX－(ES:addr16)	×	×	×
		AX, saddrp	2	1	－	AX－(saddrp)	×	×	×
		AX, [HL+byte]	3	1	4	AX－(HL＋byte)	×	×	×
		AX, ES: [HL+byte]	4	2	5	AX－((ES:HL)＋byte)	×	×	×

注1.　内部RAM領域，SFR領域および拡張SFR領域をアクセスしたとき，またはデータ・アクセスをしないときのCPUクロック（f_{CLK}）数。

2.　コード・フラッシュ・メモリ領域および8ビット命令でデータ・フラッシュ・メモリ領域をアクセスしたときのCPUクロック（f_{CLK}）数。

備考　クロック数は内部ROM（フラッシュ・メモリ）領域にプログラムがある場合です。内部RAM領域から命令フェッチする場合，最大2倍＋3クロックになります。

表5－7　RL78-S3コアのオペレーション一覧（12/18）

命令群	ニモニック	オペランド	バイト	クロック		オペレーション	フラグ
				注1	注2		Z AC CY
乗除積和算	MULU	X	1	1	－	AX ← A×X	
	MULHU		3	2	－	BCAX ← A X×BC（符号なし）	
	MULH		3	2	－	BCAX ← A X×BC（符号付き）	
	DIVHU		3	9	－	AX（商），DE（余り） ← AX÷DE（符号なし）	
	DIVWU		3	17	－	BCAX（商），HLDE（余り） ← BCAX÷HLDE（符号なし）	
	MACHU		3	3	－	MACR ← MACR＋AX×BC（符号なし）	× ×
	MACH		3	3	－	MACR ← MACR＋AX×BC（符号付き）	× ×

注1.　内部RAM領域，SFR領域および拡張SFR領域をアクセスしたとき，またはデータ・アクセスをしないときのCPUクロック（f_{CLK}）数。

2.　コード・フラッシュ・メモリ領域および8ビット命令でデータ・フラッシュ・メモリ領域をアクセスしたときのCPUクロック（f_{CLK}）数。

注意　　割り込み処理中にDIVHU，DIVWU命令を実行する場合，割り込み禁止状態(DI)で実行してください。
ただし，RAM領域での命令実行を除き，アセンブリ言語ソースにてDIVHU，DIVWU命令の直後にNOP命令を追加した場合は，割り込み許可状態でもDIVHU，DIVWU命令を実行することができます。下記のコンパイラはビルド時にDIVHU，DIVWU命令が出力される場合，その直後に自動でNOP命令が挿入されます。

- CA78K0R(ルネサス エレクトロニクス社 コンパイラ製品)V1.71以降のC言語ソースおよびアセンブリ言語ソース
- EWRL78 (IAR社 コンパイラ製品) Service pack 1.40.6以降のC言語ソース
- GNURL78 (KPIT社 コンパイラ)のC言語ソース

備考1.　クロック数は内部ROM（フラッシュ・メモリ）領域にプログラムがある場合です。内部RAM領域から命令フェッチする場合，最大2倍＋3クロックになります。

2.　MACR：積和演算累計レジスタ（MACRH，MACRL）

表5－7　RL78-S3コアのオペレーション一覧（13/18）

命令群	ニモニック	オペランド	バイト	クロック		オペレーション	フラグ
				注1	注2		Z AC CY
増減	INC	r	1	1	－	r ← r＋1	× ×
		!addr16	3	2	－	(addr16) ← (addr16)＋1	× ×
		ES:!addr16	4	3	－	(ES, addr16) ← (ES, addr16)＋1	× ×
		saddr	2	2	－	(saddr) ← (saddr)＋1	× ×
		[HL+byte]	3	2	－	(HL＋byte) ← (HL＋byte)＋1	× ×
		ES: [HL+byte]	4	3	－	((ES:HL)＋byte) ← ((ES:HL)＋byte)＋1	× ×
	DEC	r	1	1	－	r ← r－1	× ×
		!addr16	3	2	－	(addr16) ← (addr16)－1	× ×
		ES:!addr16	4	3	－	(ES, addr16) ← (ES, addr16) －1	× ×
		saddr	2	2	－	(saddr) ← (saddr)－1	× ×
		[HL+byte]	3	2	－	(HL＋byte) ← (HL＋byte) －1	× ×
		ES: [HL+byte]	4	3	－	((ES:HL)＋byte) ← ((ES:HL)＋byte) －1	× ×
	INCW	rp	1	1	－	rp ← rp＋1	
		!addr16	3	2	－	(addr16) ← (addr16)＋1	
		ES:!addr16	4	3	－	(ES, addr16) ← (ES, addr16)＋1	
		saddrp	2	2	－	(saddrp) ← (saddrp)＋1	
		[HL+byte]	3	2	－	(HL＋byte) ← (HL＋byte)＋1	
		ES: [HL+byte]	4	3	－	((ES:HL)＋byte) ← ((ES:HL)＋byte)＋1	
	DECW	rp	1	1	－	rp ← rp－1	
		!addr16	3	2	－	(addr16) ← (addr16)－1	
		ES:!addr16	4	3	－	(ES, addr16) ← (ES, addr16)－1	
		saddrp	2	2	－	(saddrp) ← (saddrp)－1	
		[HL+byte]	3	2	－	(HL＋byte) ← (HL＋byte) －1	
		ES: [HL+byte]	4	3	－	((ES:HL)＋byte) ← ((ES:HL)＋byte) －1	
シフト	SHR	A, cnt	2	1	－	(CY ← A_0, A_{m-1} ← A_m, A_7 ← 0) ×cnt	×
	SHRW	AX, cnt	2	1	－	(CY ← AX_0, AX_{m-1} ← AX_m, AX_{15} ← 0) ×cnt	×
	SHL	A, cnt	2	1	－	(CY ← A_7, A_m ← A_{m-1}, A_0 ← 0) ×cnt	×
		B, cnt	2	1	－	(CY ← B_7, B_m ← B_{m-1}, B_0 ← 0) ×cnt	×
		C, cnt	2	1	－	(CY ← C_7, C_m ← C_{m-1}, C_0 ← 0) ×cnt	×
	SHLW	AX, cnt	2	1	－	(CY ← AX_{15}, AX_m ← AX_{m-1}, AX_0 ← 0) ×cnt	×
		BC, cnt	2	1	－	(CY ← BC_{15}, BC_m ← BC_{m-1}, BC_0 ← 0) ×cnt	×
	SAR	A, cnt	2	1	－	(CY ← A_0, A_{m-1} ← A_m, A_7 ← A_7) ×cnt	×
	SARW	AX, cnt	2	1	－	(CY ← AX_0, AX_{m-1} ← AX_m, AX_{15} ← AX_{15}) ×cnt	×

注1.　内部RAM領域，SFR領域および拡張SFR領域をアクセスしたとき，またはデータ・アクセスをしないときのCPUクロック（f_{CLK}）数。

2.　コード・フラッシュ・メモリ領域および8ビット命令でデータ・フラッシュ・メモリ領域をアクセスしたときのCPUクロック（f_{CLK}）数。

備考1.　クロック数は内部ROM（フラッシュ・メモリ）領域にプログラムがある場合です。内部RAM領域から命令フェッチする場合，最大4倍＋6クロックになります。

2.　cntはビット・シフト数です。

表5－7　RL78-S3コアのオペレーション一覧（14/18）

命令群	ニモニック	オペランド	バイト	クロック		オペレーション	フラグ
				注1	注2		Z AC CY
ローテート	ROR	A, 1	2	1	－	(CY, A_7 ← A_0, A_{m-1} ← A_m) × 1	×
	ROL	A, 1	2	1	－	(CY, A_0 ← A_7, A_{m+1} ← A_m) × 1	×
	RORC	A, 1	2	1	－	(CY ← A_0, A_7 ← CY, A_{m-1} ← A_m) × 1	×
	ROLC	A, 1	2	1	－	(CY ← A_7, A_0 ← CY, A_{m+1} ← A_m) × 1	×
	ROLWC	AX,1	2	1	－	(CY ← AX_{15}, AX_0 ← CY, AX_{m+1} ← AX_m) ×1	×
		BC,1	2	1	－	(CY ← BC_{15}, BC_0 ← CY, BC_{m+1} ← BC_m) ×1	×
ビット操作	MOV1	CY, A.bit	2	1	－	CY ← A.bit	×
		A.bit, CY	2	1	－	A.bit ← CY	
		CY, PSW.bit	3	1	－	CY ← PSW.bit	×
		PSW.bit, CY	3	4	－	PSW.bit ← CY	× ×
		CY, saddr.bit	3	1	－	CY ← (saddr).bit	×
		saddr.bit, CY	3	2	－	(saddr).bit ← CY	
		CY, sfr.bit	3	1	－	CY ← sfr.bit	×
		sfr.bit, CY	3	2	－	sfr.bit ← CY	
		CY,[HL].bit	2	1	4	CY ← (HL).bit	×
		[HL].bit, CY	2	2	－	(HL).bit ← CY	
		CY, ES:[HL].bit	3	2	5	CY ← (ES, HL).bit	×
		ES:[HL].bit, CY	3	3	－	(ES, HL).bit ← CY	
	AND1	CY, A.bit	2	1	－	CY ← CY∧A.bit	×
		CY, PSW.bit	3	1	－	CY ← CY∧PSW.bit	×
		CY, saddr.bit	3	1	－	CY ← CY∧(saddr).bit	×
		CY, sfr.bit	3	1	－	CY ← CY∧sfr.bit	×
		CY,[HL].bit	2	1	4	CY ← CY∧(HL).bit	×
		CY, ES:[HL].bit	3	2	5	CY ← CY∧(ES, HL).bit	×
	OR1	CY, A.bit	2	1	－	CY ← CY∨A.bit	×
		CY, PSW.bit	3	1	－	CY ← CY∨PSW.bit	×
		CY, saddr.bit	3	1	－	CY ← CY∨(saddr).bit	×
		CY, sfr.bit	3	1	－	CY ← CY∨sfr.bit	×
		CY, [HL].bit	2	1	4	CY ← CY∨(HL).bit	×
		CY, ES:[HL].bit	3	2	5	CY ← CY∨(ES, HL).bit	×

注1.　内部RAM領域，SFR領域および拡張SFR領域をアクセスしたとき，またはデータ・アクセスをしないときのCPUクロック（f_{CLK}）数。

2.　コード・フラッシュ・メモリ領域および8ビット命令でデータ・フラッシュ・メモリ領域をアクセスしたときのCPUクロック（f_{CLK}）数。

備考　クロック数は内部ROM（フラッシュ・メモリ）領域にプログラムがある場合です。内部RAM領域から命令フェッチする場合，最大2倍＋3クロックになります。

表5－7　RL78-S3コアのオペレーション一覧（15/18）

命令群	ニモニック	オペランド	バイト	クロック		オペレーション	フラグ		
				注1	注2		Z	AC	CY
ビット操作	XOR1	CY, A.bit	2	1	－	CY ← CY￥A.bit			×
		CY, PSW.bit	3	1	－	CY ← CY￥PSW.bit			×
		CY, saddr.bit	3	1	－	CY ← CY￥(saddr).bit			×
		CY, sfr.bit	3	1	－	CY ← CY￥sfr.bit			×
		CY, [HL].bit	2	1	4	CY ← CY￥(HL).bit			×
		CY, ES:[HL].bit	3	2	5	CY ← CY￥(ES, HL).bit			×
	SET1	A.bit	2	1	－	A.bit ← 1			
		PSW.bit	3	4	－	PSW.bit ← 1	×	×	×
		!addr16.bit	4	2	－	(addr16).bit ← 1			
		ES:!addr16.bit	5	3	－	(ES, addr16).bit ← 1			
		saddr.bit	3	2	－	(saddr).bit ← 1			
		sfr.bit	3	2	－	sfr.bit ← 1			
		[HL].bit	2	2	－	(HL).bit ← 1			
		ES:[HL].bit	3	3	－	(ES, HL).bit ← 1			
	CLR1	A.bit	2	1	－	A.bit ← 0			
		PSW.bit	3	4	－	PSW.bit ← 0	×	×	×
		!addr16.bit	4	2	－	(addr16).bit ← 0			
		ES:!addr16.bit	5	3	－	(ES, addr16).bit ← 0			
		saddr.bit	3	2	－	(saddr.bit) ← 0			
		sfr.bit	3	2	－	sfr.bit ← 0			
		[HL].bit	2	2	－	(HL).bit ← 0			
		ES:[HL].bit	3	3	－	(ES, HL).bit ← 0			
	SET1	CY	2	1	－	CY ← 1			1
	CLR1	CY	2	1	－	CY ← 0			0
	NOT1	CY	2	1	－	CY ← CY			×

注1.　内部RAM領域，SFR領域および拡張SFR領域をアクセスしたとき，またはデータ・アクセスをしないときのCPUクロック（f_{CLK}）数。

2.　コード・フラッシュ・メモリ領域および8ビット命令でデータ・フラッシュ・メモリ領域をアクセスしたときのCPUクロック（f_{CLK}）数。

備考　クロック数は内部ROM（フラッシュ・メモリ）領域にプログラムがある場合です。内部RAM領域から命令フェッチする場合，最大2倍＋3クロックになります。

表5－7　RL78-S3コアのオペレーション一覧（16/18）

命令群	ニモニック	オペランド	バイト	クロック		オペレーション	フラグ
				注1	注2		Z AC CY
コール・リターン	CALL	rp	2	3	－	(SP－2) ← (PC＋2)S, (SP－3) ← (PC＋2)H, (SP－4) ← (PC＋2)L, PC ← CS, rp, SP ← SP－4	
		$!addr20	3	3	－	(SP－2) ← (PC＋3)S, (SP－3) ← (PC＋3)H, (SP－4) ← (PC＋3)L, PC ← PC＋3＋jdisp16, SP ← SP－4	
		!addr16	3	3	－	(SP－2) ← (PC＋3)S, (SP－3) ← (PC＋3)H, (SP－4) ← (PC＋3)L, PC ← 0000, addr16, SP ← SP－4	
		!!addr20	4	3	－	(SP－2) ← (PC＋4)S, (SP－3) ← (PC＋4)H, (SP－4) ← (PC＋4)L, PC ← addr20, SP ← SP－4	
	CALLT	[addr5]	2	5	－	(SP－2) ← (PC＋2)S , (SP－3) ← (PC＋2)H, (SP－4) ← (PC＋2)L , PCS ← 0000, PCH ← (0000, addr5＋1), PCL ← (0000, addr5), SP ← SP－4	
	BRK	－	2	5	－	(SP－1) ← PSW, (SP－2) ← (PC＋2)S, (SP－3) ← (PC＋2)H, (SP－4) ← (PC＋2)L, PCS ← 0000, PCH ← (0007FH), PCL ← (0007EH), SP ← SP－4, IE ← 0	
	RET	－	1	6	－	PCL ← (SP), PCH ← (SP＋1), PCS ← (SP＋2), SP ← SP＋4	
	RETI	－	2	6	－	PCL ← (SP), PCH ← (SP＋1), PCS ← (SP＋2), PSW ← (SP＋3), SP ← SP＋4	R R R
	RETB	－	2	6	－	PCL ← (SP), PCH ← (SP＋1), PCS ← (SP＋2), PSW ← (SP＋3), SP ← SP＋4	R R R

注1.　内部RAM領域，SFR領域および拡張SFR領域をアクセスしたとき，またはデータ・アクセスをしないときのCPUクロック（f_{CLK}）数。

2.　コード・フラッシュ・メモリ領域および8ビット命令でデータ・フラッシュ・メモリ領域をアクセスしたときのCPUクロック（f_{CLK}）数。

備考　クロック数は内部ROM（フラッシュ・メモリ）領域にプログラムがある場合です。内部RAM領域から命令フェッチする場合，最大2倍＋3クロックになります。

表5－7　RL78-S3コアのオペレーション一覧（17/18）

命令群	ニモニック	オペランド	バイト	クロック		オペレーション	フラグ
				注1	注2		Z AC CY
スタック操作	PUSH	PSW	2	1	－	(SP－1) ← PSW, (SP－2) ← 00H, SP ← SP－2	
		rp	1	1	－	(SP－1) ← rp_H, (SP－2) ← rp_L, SP ← SP－2	
	POP	PSW	2	3	－	PSW ← (SP＋1), SP ← SP＋2	R R R
		rp	1	1	－	rp_L ← (SP), rp_H ← (SP＋1), SP ← SP＋2	
	MOVW	SP, #word	4	1	－	SP ← word	
		SP, AX	2	1	－	SP ← AX	
		AX, SP	2	1	－	AX ← SP	
		HL, SP	3	1	－	HL ← SP	
		BC, SP	3	1	－	BC ← SP	
		DE, SP	3	1	－	DE ← SP	
	ADDW	SP, #byte	2	1	－	SP ← SP＋byte	
	SUBW	SP, #byte	2	1	－	SP ← SP－byte	
無条件分岐	BR	AX	2	3	－	PC ← CS, AX	
		$addr20	2	3	－	PC ← PC＋2＋jdisp8	
		$!addr20	3	3	－	PC ← PC＋3＋jdisp16	
		!addr16	3	3	－	PC ← 0000, addr16	
		!!addr20	4	3	－	PC ← addr20	
条件付き分岐	BC	$addr20	2	2/4[注3]	－	PC ← PC＋2＋jdisp8 if CY = 1	
	BNC	$addr20	2	2/4[注3]	－	PC ← PC＋2＋jdisp8 if CY = 0	
	BZ	$addr20	2	2/4[注3]	－	PC ← PC＋2＋jdisp8 if Z = 1	
	BNZ	$addr20	2	2/4[注3]	－	PC ← PC＋2＋jdisp8 if Z = 0	
	BH	$addr20	3	2/4[注3]	－	PC ← PC+3+jdisp8 if (Z∨CY)=0	
	BNH	$addr20	3	2/4[注3]	－	PC ← PC+3+jdisp8 if (Z∨CY)=1	
	BT	saddr.bit, $addr20	4	3/5[注3]	－	PC ← PC＋4＋jdisp8 if (saddr).bit = 1	
		sfr.bit, $addr20	4	3/5[注3]	－	PC ← PC＋4＋jdisp8 if sfr.bit = 1	
		A.bit, $addr20	3	3/5[注3]	－	PC ← PC＋3＋jdisp8 if A.bit = 1	
		PSW.bit, $addr20	4	3/5[注3]	－	PC ← PC＋4＋jdisp8 if PSW.bit = 1	
		[HL].bit, $addr20	3	3/5[注3]	6/7	PC ← PC＋3＋jdisp8 if (HL).bit = 1	
		ES:[HL].bit, $addr20	4	4/6[注3]	7/8	PC ← PC＋4＋jdisp8 if (ES, HL).bit = 1	

注1.　内部RAM領域，SFR領域および拡張SFR領域をアクセスしたとき，またはデータ・アクセスをしないときのCPUクロック（f_{CLK}）数。

2.　コード・フラッシュ・メモリ領域および8ビット命令でデータ・フラッシュ・メモリ領域をアクセスしたときのCPUクロック（f_{CLK}）数。

3.　クロック数は“条件不成立時／条件成立時”を表しています。

備考　クロック数は内部ROM（フラッシュ・メモリ）領域にプログラムがある場合です。内部RAM領域から命令フェッチする場合，最大2倍＋3クロックになります。

表5－7　RL78-S3コアのオペレーション一覧（18/18）

命令群	ニモニック	オペランド	バイト	クロック		オペレーション	フラグ
				注1	注2		Z AC CY
条件付き分岐	BF	saddr.bit, $addr20	4	3/5[注3]	－	PC ← PC＋4＋jdisp8 if (saddr).bit = 0	
		sfr.bit, $addr20	4	3/5[注3]	－	PC ← PC＋4＋jdisp8 if sfr.bit = 0	
		A.bit, $addr20	3	3/5[注3]	－	PC ← PC＋3＋jdisp8 if A.bit = 0	
		PSW.bit, $addr20	4	3/5[注3]	－	PC ← PC＋4＋jdisp8 if PSW.bit = 0	
		[HL].bit, $addr20	3	3/5[注3]	6/7	PC ← PC＋3＋jdisp8 if (HL).bit = 0	
		ES:[HL].bit, $addr20	4	4/6[注3]	7/8	PC ← PC＋4＋jdisp8 if (ES, HL).bit = 0	
	BTCLR	saddr.bit, $addr20	4	3/5[注3]	－	PC ← PC＋4＋jdisp8 if (saddr).bit = 1 then reset (saddr).bit	
		sfr.bit, $addr20	4	3/5[注3]	－	PC ← PC＋4＋jdisp8 if sfr.bit = 1 then reset sfr.bit	
		A.bit, $addr20	3	3/5[注3]	－	PC ← PC＋3＋jdisp8 if A.bit = 1 then reset A.bit	
		PSW.bit, $addr20	4	3/5[注3]	－	PC ← PC＋4＋jdisp8 if PSW.bit = 1 then reset PSW.bit	× × ×
		[HL].bit, $addr20	3	3/5[注3]	－	PC ← PC＋3＋jdisp8 if (HL).bit = 1 then reset (HL).bit	
		ES:[HL].bit, $addr20	4	4/6[注3]	－	PC ← PC＋4＋jdisp8 if (ES, HL).bit = 1 then reset (ES, HL).bit	
条件付きスキップ	SKC	－	2	1	－	Next instruction skip if CY = 1	
	SKNC	－	2	1	－	Next instruction skip if CY = 0	
	SKZ	－	2	1	－	Next instruction skip if Z = 1	
	SKNZ	－	2	1	－	Next instruction skip if Z = 0	
	SKH	－	2	1	－	Next instruction skip if (Z∨CY)=0	
	SKNH	－	2	1	－	Next instruction skip if (Z∨CY)=1	
CPU制御	SEL[注4]	RBn	2	1	－	RBS[1:0] ← n	
	NOP	－	1	1	－	No Operation	
	EI	－	3	4	－	IE ← 1(Enable Interrupt)	
	DI	－	3	4	－	IE ← 0(Disable Interrupt)	
	HALT	－	2	3	－	Set HALT Mode	
	STOP	－	2	3	－	Set STOP Mode	

注1.　内部RAM領域，SFR領域および拡張SFR領域をアクセスしたとき，またはデータ・アクセスをしないときのCPUクロック（f_{CLK}）数。

2.　コード・フラッシュ・メモリ領域および8ビット命令でデータ・フラッシュ・メモリ領域をアクセスしたときのCPUクロック（f_{CLK}）数。

3.　クロック数は“条件不成立時／条件成立時”を表しています。

4.　nはレジスタ・バンク番号です（n = 0-3）。

備考　クロック数は内部ROM（フラッシュ・メモリ）領域にプログラムがある場合です。内部RAM領域から命令フェッチする場合，最大2倍＋3クロックになります。

5.6　命令フォーマット

命令は，固定したOP codeとそのあとに続くオペランドにより構成されます。その一覧を示します。

表5－8　命令フォーマット一覧（1/30）

ニモニック	オペランド	命令コード				
		1st	2nd	3rd	4th	5th
MOV	X, #byte	50	data	－	－	－
	A, #byte	51	data	－	－	－
	C, #byte	52	data	－	－	－
	B, #byte	53	data	－	－	－
	E, #byte	54	data	－	－	－
	D, #byte	55	data	－	－	－
	L, #byte	56	data	－	－	－
	H, #byte	57	data	－	－	－
	saddr, #byte	CD	saddr	data	－	－
	sfr, #byte	CE	sfr	data	－	－
	!addr16,#byte	CF	adrl	adrh	data	－
	A, X	60	－	－	－	－
	A, C	62	－	－	－	－
	A, B	63	－	－	－	－
	A, E	64	－	－	－	－
	A, D	65	－	－	－	－
	A, L	66	－	－	－	－
	A, H	67	－	－	－	－
	X, A	70	－	－	－	－
	C, A	72	－	－	－	－
	B, A	73	－	－	－	－
	E, A	74	－	－	－	－
	D, A	75	－	－	－	－
	L, A	76	－	－	－	－
	H, A	77	－	－	－	－
	A, saddr	8D	saddr	－	－	－
	saddr, A	9D	saddr	－	－	－
	A, sfr	8E	sfr	－	－	－
	sfr, A	9E	sfr	－	－	－
	A, !addr16	8F	adrl	adrh	－	－
	!addr16, A	9F	adrl	adrh	－	－
	PSW, #byte	CE	FA	data	－	－
	A, PSW	8E	FA	－	－	－
	PSW, A	9E	FA	－	－	－
	ES, #byte	41	data	－	－	－
	ES, saddr	61	B8	saddr	－	－
	A, ES	8E	FD	－	－	－
	ES, A	9E	FD	－	－	－
	CS, #byte	CE	FC	data	－	－

表5－8　命令フォーマット一覧（2/30）

ニモニック	オペランド	命令コード				
		1st	2nd	3rd	4th	5th
MOV	A, CS	8E	FC	－	－	－
	CS, A	9E	FC	－	－	－
	A, [DE]	89	－	－	－	－
	[DE], A	99	－	－	－	－
	[DE+byte],#byte	CA	adr	data	－	－
	A, [DE+byte]	8A	adr	－	－	－
	[DE+byte], A	9A	adr	－	－	－
	A, [HL]	8B	－	－	－	－
	[HL], A	9B	－	－	－	－
	[HL+byte],#byte	CC	adr	data	－	－
	A, [HL+byte]	8C	adr	－	－	－
	[HL+byte], A	9C	adr	－	－	－
	A, [HL+B]	61	C9	－	－	－
	[HL+B], A	61	D9	－	－	－
	A, [HL+C]	61	E9	－	－	－
	[HL+C], A	61	F9	－	－	－
	word[B], #byte	19	adrl	adrh	data	－
	A, word[B]	09	adrl	adrh	－	－
	word[B], A	18	adrl	adrh	－	－
	word[C], #byte	38	adrl	adrh	data	－
	A, word[C]	29	adrl	adrh	－	－
	word[C], A	28	adrl	adrh	－	－
	word[BC], #byte	39	adrl	adrh	data	－
	A, word[BC]	49	adrl	adrh	－	－
	word[BC], A	48	adrl	adrh	－	－
	[SP+byte], #byte	C8	adr	data	－	－
	A, [SP+byte]	88	adr	－	－	－
	[SP+byte], A	98	adr	－	－	－
	B, saddr	E8	saddr	－	－	－
	B, !addr16	E9	adrl	adrh	－	－
	C, saddr	F8	saddr	－	－	－
	C, !addr16	F9	adrl	adrh	－	－
	X, saddr	D8	saddr	－	－	－
	X, !addr16	D9	adrl	adrh	－	－
	ES:!addr16, #byte	11	CF	adrl	adrh	data
	A, ES:!addr16	11	8F	adrl	adrh	－
	ES:!addr16, A	11	9F	adrl	adrh	－
	A, ES:[DE]	11	89	－	－	－
	ES:[DE], A	11	99	－	－	－
	ES:[DE+byte], #byte	11	CA	adr	data	－
	A, ES:[DE+byte]	11	8A	adr	－	－
	ES:[DE+byte], A	11	9A	adr	－	－
	A, ES:[HL]	11	8B	－	－	－

表5－8　命令フォーマット一覧（3/30）

ニモニック	オペランド	命令コード				
		1st	2nd	3rd	4th	5th
MOV	ES:[HL], A	11	9B	－	－	－
	ES:[HL+byte], #byte	11	CC	adr	data	－
	A, ES:[HL+byte]	11	8C	adr	－	－
	ES:[HL+byte], A	11	9C	adr	－	－
	A, ES:[HL+B]	11	61	C9	－	－
	ES:[HL+B], A	11	61	D9	－	－
	A, ES:[HL+C]	11	61	E9	－	－
	ES:[HL+C], A	11	61	F9	－	－
	ES:word[B], #byte	11	19	adrl	adrh	data
	A, ES:word[B]	11	09	adrl	adrh	－
	ES:word[B], A	11	18	adrl	adrh	－
	ES:word[C], #byte	11	38	adrl	adrh	data
	A, ES:word[C]	11	29	adrl	adrh	－
	ES:word[C], A	11	28	adrl	adrh	－
	ES:word[BC], #byte	11	39	adrl	adrh	data
	A, ES:word[BC]	11	49	adrl	adrh	－
	ES:word[BC], A	11	48	adrl	adrh	－
	B, ES:!addr16	11	E9	adrl	adrh	－
	C, ES:!addr16	11	F9	adrl	adrh	－
	X, ES:!addr16	11	D9	adrl	adrh	－
XCH	A, X	08	－	－	－	－
	A, C	61	8A	－	－	－
	A, B	61	8B	－	－	－
	A, E	61	8C	－	－	－
	A, D	61	8D	－	－	－
	A, L	61	8E	－	－	－
	A, H	61	8F	－	－	－
	A, saddr	61	A8	saddr	－	－
	A, sfr	61	AB	sfr	－	－
	A, !addr16	61	AA	adrl	adrh	－
	A, [DE]	61	AE	－	－	－
	A, [DE+byte]	61	AF	adr	－	－
	A, [HL]	61	AC	－	－	－
	A, [HL+byte]	61	AD	adr	－	－
	A, [HL+B]	61	B9	－	－	－
	A, [HL+C]	61	A9	－	－	－
	A, ES:!addr16	11	61	AA	adrl	adrh
	A, ES: [DE]	11	61	AE	－	－
	A, ES: [DE+byte]	11	61	AF	adr	－
	A, ES: [HL]	11	61	AC	－	－
	A, ES: [HL+byte]	11	61	AD	adr	－
	A, ES: [HL+B]	11	61	B9	－	－
	A, ES: [HL+C]	11	61	A9	－	－

表5－8　命令フォーマット一覧（4/30）

ニモニック	オペランド	命令コード				
		1st	2nd	3rd	4th	5th
ONEB	A	E1	－	－	－	－
	X	E0	－	－	－	－
	B	E3	－	－	－	－
	C	E2	－	－	－	－
	saddr	E4	saddr	－	－	－
	!addr16	E5	adrl	adrh	－	－
	ES:!addr16	11	E5	adrl	adrh	－
CLRB	A	F1	－	－	－	－
	X	F0	－	－	－	－
	B	F3	－	－	－	－
	C	F2	－	－	－	－
	saddr	F4	saddr	－	－	－
	!addr16	F5	adrl	adrh	－	－
	ES:!addr16	11	F5	adrl	adrh	－
MOVS	[HL+byte], X	61	CE	adr	－	－
	ES:[HL+byte], X	11	61	CE	adr	－
MOVW	AX, #word	30	datal	datah	－	－
	BC, #word	32	datal	datah	－	－
	DE, #word	34	datal	datah	－	－
	HL, #word	36	datal	datah	－	－
	saddrp,#word	C9	saddr	datal	datah	－
	sfrp,#word	CB	sfr	datal	datah	－
	AX, saddrp	AD	saddr	－	－	－
	saddrp, AX	BD	saddr	－	－	－
	AX, sfrp	AE	sfr	－	－	－
	sfrp, AX	BE	sfr	－	－	－
	AX, BC	13	－	－	－	－
	AX, DE	15	－	－	－	－
	AX, HL	17	－	－	－	－
	BC, AX	12	－	－	－	－
	DE, AX	14	－	－	－	－
	HL, AX	16	－	－	－	－
	AX, !addr16	AF	adrl	adrh	－	－
	!addr16, AX	BF	adrl	adrh	－	－
	AX, [DE]	A9	－	－	－	－
	[DE], AX	B9	－	－	－	－
	AX, [DE+byte]	AA	adr	－	－	－
	[DE+byte], AX	BA	adr	－	－	－
	AX, [HL]	AB	－	－	－	－
	[HL], AX	BB	－	－	－	－
	AX, [HL+byte]	AC	adr	－	－	－
	[HL+byte], AX	BC	adr	－	－	－
	AX,word[B]	59	adrl	adrh	－	－

表5－8　命令フォーマット一覧（5/30）

ニモニック	オペランド	命令コード				
		1st	2nd	3rd	4th	5th
MOVW	word[B], AX	58	adrl	adrh	－	－
	AX,word[C]	69	adrl	adrh	－	－
	word[C], AX	68	adrl	adrh	－	－
	AX,word[BC]	79	adrl	adrh	－	－
	word[BC], AX	78	adrl	adrh	－	－
	AX, [SP+byte]	A8	adr	－	－	－
	[SP+byte], AX	B8	adr	－	－	－
	BC, saddrp	DA	saddr	－	－	－
	BC, !addr16	DB	adrl	adrh	－	－
	DE, saddrp	EA	saddr	－	－	－
	DE, !addr16	EB	adrl	adrh	－	－
	HL, saddrp	FA	saddr	－	－	－
	HL, !addr16	FB	adrl	adrh	－	－
	AX, ES:!addr16	11	AF	adrl	adrh	－
	ES:!addr16, AX	11	BF	adrl	adrh	－
	AX, ES:[DE]	11	A9	－	－	－
	ES:[DE], AX	11	B9	－	－	－
	AX, ES:[DE+byte]	11	AA	adr	－	－
	ES:[DE+byte], AX	11	BA	adr	－	－
	AX, ES:[HL]	11	AB	－	－	－
	ES:[HL], AX	11	BB	－	－	－
	AX, ES:[HL+byte]	11	AC	adr	－	－
	ES:[HL+byte], AX	11	BC	adr	－	－
	AX, ES:word[B]	11	59	adrl	adrh	－
	ES:word[B], AX	11	58	adrl	adrh	－
	AX, ES:word[C]	11	69	adrl	adrh	－
	ES:word[C], AX	11	68	adrl	adrh	－
	AX, ES:word[BC]	11	79	adrl	adrh	－
	ES:word[BC], AX	11	78	adrl	adrh	－
	BC, ES:!addr16	11	DB	adrl	adrh	－
	DE, ES:!addr16	11	EB	adrl	adrh	－
	HL, ES:!addr16	11	FB	adrl	adrh	－
XCHW	AX, BC	33	－	－	－	－
	AX, DE	35	－	－	－	－
	AX, HL	37	－	－	－	－
ONEW	AX	E6	－	－	－	－
	BC	E7	－	－	－	－
CLRW	AX	F6	－	－	－	－
	BC	F7	－	－	－	－

表5－8　命令フォーマット一覧（6/30）

ニモニック	オペランド	命令コード				
		1st	2nd	3rd	4th	5th
ADD	A, #byte	0C	data	－	－	－
	saddr, #byte	0A	saddr	data	－	－
	A, X	61	08	－	－	－
	A, C	61	0A	－	－	－
	A, B	61	0B	－	－	－
	A, E	61	0C	－	－	－
	A, D	61	0D	－	－	－
	A, L	61	0E	－	－	－
	A, H	61	0F	－	－	－
	X, A	61	00	－	－	－
	A, A	61	01	－	－	－
	C, A	61	02	－	－	－
	B, A	61	03	－	－	－
	E, A	61	04	－	－	－
	D, A	61	05	－	－	－
	L, A	61	06	－	－	－
	H, A	61	07	－	－	－
	A, saddr	0B	saddr	－	－	－
	A, !addr16	0F	adrl	adrh	－	－
	A, [HL]	0D	－	－	－	－
	A, [HL+byte]	0E	adr	－	－	－
	A, [HL+B]	61	80	－	－	－
	A, [HL+C]	61	82	－	－	－
	A, ES:!addr16	11	0F	adrl	adrh	－
	A, ES:[HL]	11	0D	－	－	－
	A, ES:[HL+byte]	11	0E	adr	－	－
	A, ES:[HL+B]	11	61	80	－	－
	A, ES:[HL+C]	11	61	82	－	－
ADDC	A, #byte	1C	data	－	－	－
	saddr, #byte	1A	saddr	data	－	－
	A, X	61	18	－	－	－
	A, C	61	1A	－	－	－
	A, B	61	1B	－	－	－
	A, E	61	1C	－	－	－
	A, D	61	1D	－	－	－
	A, L	61	1E	－	－	－
	A, H	61	1F	－	－	－
	X, A	61	10	－	－	－
	A, A	61	11	－	－	－
	C, A	61	12	－	－	－
	B, A	61	13	－	－	－
	E, A	61	14	－	－	－
	D, A	61	15	－	－	－

表5－8　命令フォーマット一覧（7/30）

ニモニック	オペランド	命令コード				
		1st	2nd	3rd	4th	5th
ADDC	L, A	61	16	－	－	－
	H, A	61	17	－	－	－
	A, saddr	1B	saddr	－	－	－
	A, !addr16	1F	adrl	adrh	－	－
	A, [HL]	1D	－	－	－	－
	A, [HL+byte]	1E	adr	－	－	－
	A, [HL+B]	61	90	－	－	－
	A, [HL+C]	61	92	－	－	－
	A, ES:!addr16	11	1F	adrl	adrh	－
	A, ES:[HL]	11	1D	－	－	－
	A, ES:[HL+byte]	11	1E	adr	－	－
	A, ES:[HL+B]	11	61	90	－	－
	A, ES:[HL+C]	11	61	92	－	－
SUB	A, #byte	2C	data	－	－	－
	saddr, #byte	2A	saddr	data	－	－
	A, X	61	28	－	－	－
	A, C	61	2A	－	－	－
	A, B	61	2B	－	－	－
	A, E	61	2C	－	－	－
	A, D	61	2D	－	－	－
	A, L	61	2E	－	－	－
	A, H	61	2F	－	－	－
	X, A	61	20	－	－	－
	A, A	61	21	－	－	－
	C, A	61	22	－	－	－
	B, A	61	23	－	－	－
	E, A	61	24	－	－	－
	D, A	61	25	－	－	－
	L, A	61	26	－	－	－
	H, A	61	27	－	－	－
	A, saddr	2B	saddr	－	－	－
	A, !addr16	2F	adrl	adrh	－	－
	A, [HL]	2D	－	－	－	－
	A, [HL+byte]	2E	adr	－	－	－
	A, [HL+B]	61	A0	－	－	－
	A, [HL+C]	61	A2	－	－	－
	A, ES:!addr16	11	2F	adrl	adrh	－
	A, ES:[HL]	11	2D	－	－	－
	A, ES:[HL+byte]	11	2E	adr	－	－
	A, ES:[HL+B]	11	61	A0	－	－
	A, ES:[HL+C]	11	61	A2	－	－

表5－8　命令フォーマット一覧（8/30）

ニモニック	オペランド	命令コード				
		1st	2nd	3rd	4th	5th
SUBC	A, #byte	3C	data	－	－	－
	saddr, #byte	3A	saddr	data	－	－
	A, X	61	38	－	－	－
	A, C	61	3A	－	－	－
	A, B	61	3B	－	－	－
	A, E	61	3C	－	－	－
	A, D	61	3D	－	－	－
	A, L	61	3E	－	－	－
	A, H	61	3F	－	－	－
	X, A	61	30	－	－	－
	A, A	61	31	－	－	－
	C, A	61	32	－	－	－
	B, A	61	33	－	－	－
	E, A	61	34	－	－	－
	D, A	61	35	－	－	－
	L, A	61	36	－	－	－
	H, A	61	37	－	－	－
	A, saddr	3B	saddr	－	－	－
	A, !addr16	3F	adrl	adrh	－	－
	A, [HL]	3D	－	－	－	－
	A, [HL+byte]	3E	adr	－	－	－
	A, [HL+B]	61	B0	－	－	－
	A, [HL+C]	61	B2	－	－	－
	A, ES:!addr16	11	3F	adrl	adrh	－
	A, ES:[HL]	11	3D	－	－	－
	A, ES:[HL+byte]	11	3E	adr	－	－
	A, ES:[HL+B]	11	61	B0	－	－
	A, ES:[HL+C]	11	61	B2	－	－
AND	A, #byte	5C	data	－	－	－
	saddr, #byte	5A	saddr	data	－	－
	A, X	61	58	－	－	－
	A, C	61	5A	－	－	－
	A, B	61	5B	－	－	－
	A, E	61	5C	－	－	－
	A, D	61	5D	－	－	－
	A, L	61	5E	－	－	－
	A, H	61	5F	－	－	－
	X, A	61	50	－	－	－
	A, A	61	51	－	－	－
	C, A	61	52	－	－	－
	B, A	61	53	－	－	－
	E, A	61	54	－	－	－
	D, A	61	55	－	－	－

表5－8　命令フォーマット一覧（9/30）

ニモニック	オペランド	命令コード				
		1st	2nd	3rd	4th	5th
AND	L, A	61	56	－	－	－
	H, A	61	57	－	－	－
	A, saddr	5B	saddr	－	－	－
	A, !addr16	5F	adrl	adrh	－	－
	A, [HL]	5D	－	－	－	－
	A, [HL+byte]	5E	adr	－	－	－
	A, [HL+B]	61	D0	－	－	－
	A, [HL+C]	61	D2	－	－	－
	A, ES:!addr16	11	5F	adrl	adrh	－
	A, ES:[HL]	11	5D	－	－	－
	A, ES:[HL+byte]	11	5E	adr	－	－
	A, ES:[HL+B]	11	61	D0	－	－
	A, ES:[HL+C]	11	61	D2	－	－
OR	A, #byte	6C	data	－	－	－
	saddr, #byte	6A	saddr	data	－	－
	A, X	61	68	－	－	－
	A, C	61	6A	－	－	－
	A, B	61	6B	－	－	－
	A, E	61	6C	－	－	－
	A, D	61	6D	－	－	－
	A, L	61	6E	－	－	－
	A, H	61	6F	－	－	－
	X, A	61	60	－	－	－
	A, A	61	61	－	－	－
	C, A	61	62	－	－	－
	B, A	61	63	－	－	－
	E, A	61	64	－	－	－
	D, A	61	65	－	－	－
	L, A	61	66	－	－	－
	H, A	61	67	－	－	－
	A, saddr	6B	saddr	－	－	－
	A, !addr16	6F	adrl	adrh	－	－
	A, [HL]	6D	－	－	－	－
	A, [HL+byte]	6E	adr	－	－	－
	A, [HL+B]	61	E0	－	－	－
	A, [HL+C]	61	E2	－	－	－
	A, ES:!addr16	11	6F	adrl	adrh	－
	A, ES:[HL]	11	6D	－	－	－
	A, ES:[HL+byte]	11	6E	adr	－	－
	A, ES:[HL+B]	11	61	E0	－	－
	A, ES:[HL+C]	11	61	E2	－	－

表5−8　命令フォーマット一覧（10/30）

ニモニック	オペランド	命令コード				
		1st	2nd	3rd	4th	5th
XOR	A, #byte	7C	data	−	−	−
	saddr, #byte	7A	saddr	data	−	−
	A, X	61	78	−	−	−
	A, C	61	7A	−	−	−
	A, B	61	7B	−	−	−
	A, E	61	7C	−	−	−
	A, D	61	7D	−	−	−
	A, L	61	7E	−	−	−
	A, H	61	7F	−	−	−
	X, A	61	70	−	−	−
	A, A	61	71	−	−	−
	C, A	61	72	−	−	−
	B, A	61	73	−	−	−
	E, A	61	74	−	−	−
	D, A	61	75	−	−	−
	L, A	61	76	−	−	−
	H, A	61	77	−	−	−
	A, saddr	7B	saddr	−	−	−
	A, !addr16	7F	adrl	adrh	−	−
	A, [HL]	7D	−	−	−	−
	A, [HL+byte]	7E	adr	−	−	−
	A, [HL+B]	61	F0	−	−	−
	A, [HL+C]	61	F2	−	−	−
	A, ES:!addr16	11	7F	adrl	adrh	−
	A, ES:[HL]	11	7D	−	−	−
	A, ES:[HL+byte]	11	7E	adr	−	−
	A, ES:[HL+B]	11	61	F0	−	−
	A, ES:[HL+C]	11	61	F2	−	−
CMP	A, #byte	4C	data	−	−	−
	saddr, #byte	4A	saddr	data	−	−
	A, X	61	48	−	−	−
	A, C	61	4A	−	−	−
	A, B	61	4B	−	−	−
	A, E	61	4C	−	−	−
	A, D	61	4D	−	−	−
	A, L	61	4E	−	−	−
	A, H	61	4F	−	−	−
	X, A	61	40	−	−	−
	A, A	61	41	−	−	−
	C, A	61	42	−	−	−
	B, A	61	43	−	−	−
	E, A	61	44	−	−	−
	D, A	61	45	−	−	−

表5−8　命令フォーマット一覧（11/30）

ニモニック	オペランド	命令コード				
		1st	2nd	3rd	4th	5th
CMP	L, A	61	46	−	−	−
	H, A	61	47	−	−	−
	A, saddr	4B	saddr	−	−	−
	A, !addr16	4F	adrl	adrh	−	−
	A, [HL]	4D	−	−	−	−
	A, [HL+byte]	4E	adr	−	−	−
	A, [HL+B]	61	C0	−	−	−
	A, [HL+C]	61	C2	−	−	−
	!addr16, #byte	40	adrl	adrh	data	−
	A, ES:!addr16	11	4F	adrl	adrh	−
	A, ES:[HL]	11	4D	−	−	−
	A, ES:[HL+byte]	11	4E	adr	−	−
	A, ES:[HL+B]	11	61	C0	−	−
	A, ES:[HL+C]	11	61	C2	−	−
	ES:!addr16, #byte	11	40	adrl	adrh	data
CMP0	A	D1	−	−	−	−
	X	D0	−	−	−	−
	B	D3	−	−	−	−
	C	D2	−	−	−	−
	saddr	D4	saddr	−	−	−
	!addr16	D5	adrl	adrh	−	−
	ES:!addr16	11	D5	adrl	adrh	−
CMPS	X, [HL+byte]	61	DE	adr	−	−
	X, ES:[HL+byte]	11	61	DE	adr	−
ADDW	AX, #word	04	datal	datah	−	−
	AX, AX	01	−	−	−	−
	AX, BC	03	−	−	−	−
	AX, DE	05	−	−	−	−
	AX, HL	07	−	−	−	−
	AX, saddrp	06	saddr	−	−	−
	AX, !addr16	02	adrl	adrh	−	−
	AX, [HL+byte]	61	09	adr	−	−
	AX, ES:!addr16	11	02	adrl	adrh	−
	AX, ES:[HL+byte]	11	61	09	adr	−
SUBW	AX, #word	24	datal	datah	−	−
	AX, BC	23	−	−	−	−
	AX, DE	25	−	−	−	−
	AX, HL	27	−	−	−	−
	AX, saddrp	26	saddr	−	−	−
	AX, !addr16	22	adrl	adrh	−	−
	AX, [HL+byte]	61	29	adr	−	−
	AX, ES:!addr16	11	22	adrl	adrh	−
	AX, ES:[HL+byte]	11	61	29	adr	−

表5－8　命令フォーマット一覧（12/30）

ニモニック	オペランド	命令コード				
		1st	2nd	3rd	4th	5th
CMPW	AX, #word	44	datal	datah	－	－
	AX, BC	43	－	－	－	－
	AX, DE	45	－	－	－	－
	AX, HL	47	－	－	－	－
	AX, saddrp	46	saddr	－	－	－
	AX, !addr16	42	adrl	adrh	－	－
	AX, [HL+byte]	61	49	adr	－	－
	AX, ES:!addr16	11	42	adrl	adrh	－
	AX, ES:[HL+byte]	11	61	49	adr	－
MULU	X	D6	－	－	－	－
★ MULHU[注]		CE	FB	01	－	－
MULH[注]		CE	FB	02	－	－
DIVHU[注]		CE	FB	03	－	－
DIVWU[注]		CE	FB	0B	－	－
MACHU[注]		CE	FB	05	－	－
MACH[注]		CE	FB	06	－	－
INC	X	80	－	－	－	－
	A	81	－	－	－	－
	C	82	－	－	－	－
	B	83	－	－	－	－
	E	84	－	－	－	－
	D	85	－	－	－	－
	L	86	－	－	－	－
	H	87	－	－	－	－
	saddr	A4	saddr	－	－	－
	!addr16	A0	adrl	adrh	－	－
	[HL+byte]	61	59	adr	－	－
	ES:!addr16	11	A0	adrl	adrh	－
	ES:[HL+byte]	11	61	59	adr	－
DEC	X	90	－	－	－	－
	A	91	－	－	－	－
	C	92	－	－	－	－
	B	93	－	－	－	－
	E	94	－	－	－	－
	D	95	－	－	－	－
	L	96	－	－	－	－
	H	97	－	－	－	－
	saddr	B4	saddr	－	－	－
	!addr16	B0	adrl	adrh	－	－
	[HL+byte]	61	69	adr	－	－
	ES:!addr16	11	B0	adrl	adrh	－
	ES:[HL+byte]	11	61	69	adr	－

注　この拡張命令はRL78-S3コアのみあります。

表5−8　命令フォーマット一覧（13/30）

ニモニック	オペランド	命令コード				
		1st	2nd	3rd	4th	5th
INCW	AX	A1	−	−	−	−
	BC	A3	−	−	−	−
	DE	A5	−	−	−	−
	HL	A7	−	−	−	−
	saddrp	A6	saddr	−	−	−
	!addr16	A2	adrl	adrh	−	−
	[HL+byte]	61	79	adr	−	−
	ES:!addr16	11	A2	adrl	adrh	−
	ES:[HL+byte]	11	61	79	adr	−
DECW	AX	B1	−	−	−	−
	BC	B3	−	−	−	−
	DE	B5	−	−	−	−
	HL	B7	−	−	−	−
	saddrp	B6	saddr	−	−	−
	!addr16	B2	adrl	adrh	−	−
	[HL+byte]	61	89	adr	−	−
	ES:!addr16	11	B2	adrl	adrh	−
	ES:[HL+byte]	11	61	89	adr	−
SHR	A, 1	31	1A	−	−	−
	A, 2	31	2A	−	−	−
	A, 3	31	3A	−	−	−
	A, 4	31	4A	−	−	−
	A, 5	31	5A	−	−	−
	A, 6	31	6A	−	−	−
	A, 7	31	7A	−	−	−
SHRW	AX, 1	31	1E	−	−	−
	AX, 2	31	2E	−	−	−
	AX, 3	31	3E	−	−	−
	AX, 4	31	4E	−	−	−
	AX, 5	31	5E	−	−	−
	AX, 6	31	6E	−	−	−
	AX, 7	31	7E	−	−	−
	AX, 8	31	8E	−	−	−
	AX, 9	31	9E	−	−	−
	AX, 10	31	AE	−	−	−
	AX, 11	31	BE	−	−	−
	AX, 12	31	CE	−	−	−
	AX, 13	31	DE	−	−	−
	AX, 14	31	EE	−	−	−
	AX, 15	31	FE	−	−	−

表5−8　命令フォーマット一覧（14/30）

ニモニック	オペランド	命令コード				
		1st	2nd	3rd	4th	5th
SHL	A, 1	31	19	−	−	−
	A, 2	31	29	−	−	−
	A, 3	31	39	−	−	−
	A, 4	31	49	−	−	−
	A, 5	31	59	−	−	−
	A, 6	31	69	−	−	−
	A, 7	31	79	−	−	−
	B, 1	31	18	−	−	−
	B, 2	31	28	−	−	−
	B, 3	31	38	−	−	−
	B, 4	31	48	−	−	−
	B, 5	31	58	−	−	−
	B, 6	31	68	−	−	−
	B, 7	31	78	−	−	−
	C, 1	31	17	−	−	−
	C, 2	31	27	−	−	−
	C, 3	31	37	−	−	−
	C, 4	31	47	−	−	−
	C, 5	31	57	−	−	−
	C, 6	31	67	−	−	−
	C, 7	31	77	−	−	−
SHLW	AX, 1	31	1D	−	−	−
	AX, 2	31	2D	−	−	−
	AX, 3	31	3D	−	−	−
	AX, 4	31	4D	−	−	−
	AX, 5	31	5D	−	−	−
	AX, 6	31	6D	−	−	−
	AX, 7	31	7D	−	−	−
	AX, 8	31	8D	−	−	−
	AX, 9	31	9D	−	−	−
	AX, 10	31	AD	−	−	−
	AX, 11	31	BD	−	−	−
	AX, 12	31	CD	−	−	−
	AX, 13	31	DD	−	−	−
	AX, 14	31	ED	−	−	−
	AX, 15	31	FD	−	−	−
	BC, 1	31	1C	−	−	−
	BC, 2	31	2C	−	−	−
	BC, 3	31	3C	−	−	−
	BC, 4	31	4C	−	−	−
	BC, 5	31	5C	−	−	−
	BC, 6	31	6C	−	−	−
	BC, 7	31	7C	−	−	−

表5−8　命令フォーマット一覧（15/30）

ニモニック	オペランド	命令コード				
		1st	2nd	3rd	4th	5th
SHLW	BC, 8	31	8C	−	−	−
	BC, 9	31	9C	−	−	−
	BC, 10	31	AC	−	−	−
	BC, 11	31	BC	−	−	−
	BC, 12	31	CC	−	−	−
	BC, 13	31	DC	−	−	−
	BC, 14	31	EC	−	−	−
	BC, 15	31	FC	−	−	−
SAR	A, 1	31	1B	−	−	−
	A, 2	31	2B	−	−	−
	A, 3	31	3B	−	−	−
	A, 4	31	4B	−	−	−
	A, 5	31	5B	−	−	−
	A, 6	31	6B	−	−	−
	A, 7	31	7B	−	−	−
SARW	AX, 1	31	1F	−	−	−
	AX, 2	31	2F	−	−	−
	AX, 3	31	3F	−	−	−
	AX, 4	31	4F	−	−	−
	AX, 5	31	5F	−	−	−
	AX, 6	31	6F	−	−	−
	AX, 7	31	7F	−	−	−
	AX, 8	31	8F	−	−	−
	AX, 9	31	9F	−	−	−
	AX, 10	31	AF	−	−	−
	AX, 11	31	BF	−	−	−
	AX, 12	31	CF	−	−	−
	AX, 13	31	DF	−	−	−
	AX, 14	31	EF	−	−	−
	AX, 15	31	FF	−	−	−
ROR	A, 1	61	DB	−	−	−
ROL	A, 1	61	EB	−	−	−
RORC	A, 1	61	FB	−	−	−
ROLC	A, 1	61	DC	−	−	−
ROLWC	AX, 1	61	EE	−	−	−
	BC, 1	61	FE	−	−	−
MOV1	CY, saddr.0	71	04	saddr	−	−
	CY, saddr.1	71	14	saddr	−	−
	CY, saddr.2	71	24	saddr	−	−
	CY, saddr.3	71	34	saddr	−	−
	CY, saddr.4	71	44	saddr	−	−
	CY, saddr.5	71	54	saddr	−	−
	CY, saddr.6	71	64	saddr	−	−

表5－8　命令フォーマット一覧（16/30）

ニモニック	オペランド	命令コード				
		1st	2nd	3rd	4th	5th
MOV1	CY, saddr.7	71	74	saddr	－	－
	CY, sfr.0	71	0C	sfr	－	－
	CY, sfr.1	71	1C	sfr	－	－
	CY, sfr.2	71	2C	sfr	－	－
	CY, sfr.3	71	3C	sfr	－	－
	CY, sfr.4	71	4C	sfr	－	－
	CY, sfr.5	71	5C	sfr	－	－
	CY, sfr.6	71	6C	sfr	－	－
	CY, sfr.7	71	7C	sfr	－	－
	CY, A.0	71	8C	－	－	－
	CY, A.1	71	9C	－	－	－
	CY, A.2	71	AC	－	－	－
	CY, A.3	71	BC	－	－	－
	CY, A.4	71	CC	－	－	－
	CY, A.5	71	DC	－	－	－
	CY, A.6	71	EC	－	－	－
	CY, A.7	71	FC	－	－	－
	CY, PSW.0	71	0C	FA	－	－
	CY, PSW.1	71	1C	FA	－	－
	CY, PSW.2	71	2C	FA	－	－
	CY, PSW.3	71	3C	FA	－	－
	CY, PSW.4	71	4C	FA	－	－
	CY, PSW.5	71	5C	FA	－	－
	CY, PSW.6	71	6C	FA	－	－
	CY, PSW.7	71	7C	FA	－	－
	CY, [HL].0	71	84	－	－	－
	CY, [HL].1	71	94	－	－	－
	CY, [HL].2	71	A4	－	－	－
	CY, [HL].3	71	B4	－	－	－
	CY, [HL].4	71	C4	－	－	－
	CY, [HL].5	71	D4	－	－	－
	CY, [HL].6	71	E4	－	－	－
	CY, [HL].7	71	F4	－	－	－
	saddr.0, CY	71	01	saddr	－	－
	saddr.1, CY	71	11	saddr	－	－
	saddr.2, CY	71	21	saddr	－	－
	saddr.3, CY	71	31	saddr	－	－
	saddr.4, CY	71	41	saddr	－	－
	saddr.5, CY	71	51	saddr	－	－
	saddr.6, CY	71	61	saddr	－	－
	saddr.7, CY	71	71	saddr	－	－

表5－8　命令フォーマット一覧（17/30）

	ニモニック	オペランド	命令コード				
			1st	2nd	3rd	4th	5th
★	MOV1	sfr.0, CY	71	09	sfr	－	－
		sfr.1, CY	71	19	sfr	－	－
		sfr.2, CY	71	29	sfr	－	－
		sfr.3, CY	71	39	sfr	－	－
		sfr.4, CY	71	49	sfr	－	－
		sfr.5, CY	71	59	sfr	－	－
		sfr.6, CY	71	69	sfr	－	－
		sfr.7, CY	71	79	sfr	－	－
		A.0, CY	71	89	－	－	－
		A.1, CY	71	99	－	－	－
		A.2, CY	71	A9	－	－	－
		A.3, CY	71	B9	－	－	－
		A.4, CY	71	C9	－	－	－
		A.5, CY	71	D9	－	－	－
		A.6, CY	71	E9	－	－	－
		A.7, CY	71	F9	－	－	－
		PSW.0, CY	71	09	FA	－	－
		PSW.1, CY	71	19	FA	－	－
		PSW.2, CY	71	29	FA	－	－
		PSW.3, CY	71	39	FA	－	－
		PSW.4, CY	71	49	FA	－	－
		PSW.5, CY	71	59	FA	－	－
		PSW.6, CY	71	69	FA	－	－
		PSW.7, CY	71	79	FA	－	－
		[HL].0, CY	71	81	－	－	－
		[HL].1, CY	71	91	－	－	－
		[HL].2, CY	71	A1	－	－	－
		[HL].3, CY	71	B1	－	－	－
		[HL].4, CY	71	C1	－	－	－
		[HL].5, CY	71	D1	－	－	－
		[HL].6, CY	71	E1	－	－	－
		[HL].7, CY	71	F1	－	－	－
		CY, ES:[HL].0	11	71	84	－	－
		CY, ES:[HL].1	11	71	94	－	－
		CY, ES:[HL].2	11	71	A4	－	－
		CY, ES:[HL].3	11	71	B4	－	－
		CY, ES:[HL].4	11	71	C4	－	－
		CY, ES:[HL].5	11	71	D4	－	－
		CY, ES:[HL].6	11	71	E4	－	－
		CY, ES:[HL].7	11	71	F4	－	－

表5－8　命令フォーマット一覧（18/30）

ニモニック	オペランド	命令コード				
		1st	2nd	3rd	4th	5th
MOV1	ES:[HL].0, CY	11	71	81	－	－
	ES:[HL].1, CY	11	71	91	－	－
	ES:[HL].2, CY	11	71	A1	－	－
	ES:[HL].3, CY	11	71	B1	－	－
	ES:[HL].4, CY	11	71	C1	－	－
	ES:[HL].5, CY	11	71	D1	－	－
	ES:[HL].6, CY	11	71	E1	－	－
	ES:[HL].7, CY	11	71	F1	－	－
AND1	CY, saddr.0	71	05	saddr	－	－
	CY, saddr.1	71	15	saddr	－	－
	CY, saddr.2	71	25	saddr	－	－
	CY, saddr.3	71	35	saddr	－	－
	CY, saddr.4	71	45	saddr	－	－
	CY, saddr.5	71	55	saddr	－	－
	CY, saddr.6	71	65	saddr	－	－
	CY, saddr.7	71	75	saddr	－	－
	CY, sfr.0	71	0D	sfr	－	－
	CY, sfr.1	71	1D	sfr	－	－
	CY, sfr.2	71	2D	sfr	－	－
	CY, sfr.3	71	3D	sfr	－	－
	CY, sfr.4	71	4D	sfr	－	－
	CY, sfr.5	71	5D	sfr	－	－
	CY, sfr.6	71	6D	sfr	－	－
	CY, sfr.7	71	7D	sfr	－	－
	CY, A.0	71	8D	－	－	－
	CY, A.1	71	9D	－	－	－
	CY, A.2	71	AD	－	－	－
	CY, A.3	71	BD	－	－	－
	CY, A.4	71	CD	－	－	－
	CY, A.5	71	DD	－	－	－
	CY, A.6	71	ED	－	－	－
	CY, A.7	71	FD	－	－	－
	CY, PSW.0	71	0D	FA	－	－
	CY, PSW.1	71	1D	FA	－	－
	CY, PSW.2	71	2D	FA	－	－
	CY, PSW.3	71	3D	FA	－	－
	CY, PSW.4	71	4D	FA	－	－
	CY, PSW.5	71	5D	FA	－	－
	CY, PSW.6	71	6D	FA	－	－
	CY, PSW.7	71	7D	FA	－	－

表5－8　命令フォーマット一覧（19/30）

ニモニック	オペランド	命令コード				
		1st	2nd	3rd	4th	5th
AND1	CY, [HL].0	71	85	－	－	－
	CY, [HL].1	71	95	－	－	－
	CY, [HL].2	71	A5	－	－	－
	CY, [HL].3	71	B5	－	－	－
	CY, [HL].4	71	C5	－	－	－
	CY, [HL].5	71	D5	－	－	－
	CY, [HL].6	71	E5	－	－	－
	CY, [HL].7	71	F5	－	－	－
	CY, ES:[HL].0	11	71	85	－	－
	CY, ES:[HL].1	11	71	95	－	－
	CY, ES:[HL].2	11	71	A5	－	－
	CY, ES:[HL].3	11	71	B5	－	－
	CY, ES:[HL].4	11	71	C5	－	－
	CY, ES:[HL].5	11	71	D5	－	－
	CY, ES:[HL].6	11	71	E5	－	－
	CY, ES:[HL].7	11	71	F5	－	－
OR1	CY, saddr.0	71	06	saddr	－	－
	CY, saddr.1	71	16	saddr	－	－
	CY, saddr.2	71	26	saddr	－	－
	CY, saddr.3	71	36	saddr	－	－
	CY, saddr.4	71	46	saddr	－	－
	CY, saddr.5	71	56	saddr	－	－
	CY, saddr.6	71	66	saddr	－	－
	CY, saddr.7	71	76	saddr	－	－
	CY, sfr.0	71	0E	sfr	－	－
	CY, sfr.1	71	1E	sfr	－	－
	CY, sfr.2	71	2E	sfr	－	－
	CY, sfr.3	71	3E	sfr	－	－
	CY, sfr.4	71	4E	sfr	－	－
	CY, sfr.5	71	5E	sfr	－	－
	CY, sfr.6	71	6E	sfr	－	－
	CY, sfr.7	71	7E	sfr	－	－
	CY, A.0	71	8E	－	－	－
	CY, A.1	71	9E	－	－	－
	CY, A.2	71	AE	－	－	－
	CY, A.3	71	BE	－	－	－
	CY, A.4	71	CE	－	－	－
	CY, A.5	71	DE	－	－	－
	CY, A.6	71	EE	－	－	－
	CY, A.7	71	FE	－	－	－

表5－8　命令フォーマット一覧（20/30）

ニモニック	オペランド	命令コード				
		1st	2nd	3rd	4th	5th
OR1	CY, PSW.0	71	0E	FA	－	－
	CY, PSW.1	71	1E	FA	－	－
	CY, PSW.2	71	2E	FA	－	－
	CY, PSW.3	71	3E	FA	－	－
	CY, PSW.4	71	4E	FA	－	－
	CY, PSW.5	71	5E	FA	－	－
	CY, PSW.6	71	6E	FA	－	－
	CY, PSW.7	71	7E	FA	－	－
	CY, [HL].0	71	86	－	－	－
	CY, [HL].1	71	96	－	－	－
	CY, [HL].2	71	A6	－	－	－
	CY, [HL].3	71	B6	－	－	－
	CY, [HL].4	71	C6	－	－	－
	CY, [HL].5	71	D6	－	－	－
	CY, [HL].6	71	E6	－	－	－
	CY, [HL].7	71	F6	－	－	－
	CY, ES:[HL].0	11	71	86	－	－
	CY, ES:[HL].1	11	71	96	－	－
	CY, ES:[HL].2	11	71	A6	－	－
	CY, ES:[HL].3	11	71	B6	－	－
	CY, ES:[HL].4	11	71	C6	－	－
	CY, ES:[HL].5	11	71	D6	－	－
	CY, ES:[HL].6	11	71	E6	－	－
	CY, ES:[HL].7	11	71	F6	－	－
XOR1	CY, saddr.0	71	07	saddr	－	－
	CY, saddr.1	71	17	saddr	－	－
	CY, saddr.2	71	27	saddr	－	－
	CY, saddr.3	71	37	saddr	－	－
	CY, saddr.4	71	47	saddr	－	－
	CY, saddr.5	71	57	saddr	－	－
	CY, saddr.6	71	67	saddr	－	－
	CY, saddr.7	71	77	saddr	－	－
	CY, sfr.0	71	0F	sfr	－	－
	CY, sfr.1	71	1F	sfr	－	－
	CY, sfr.2	71	2F	sfr	－	－
	CY, sfr.3	71	3F	sfr	－	－
	CY, sfr.4	71	4F	sfr	－	－
	CY, sfr.5	71	5F	sfr	－	－
	CY, sfr.6	71	6F	sfr	－	－
	CY, sfr.7	71	7F	sfr	－	－

表5－8　命令フォーマット一覧（21/30）

ニモニック	オペランド	命令コード				
		1st	2nd	3rd	4th	5th
XOR1	CY, A.0	71	8F	－	－	－
	CY, A.1	71	9F	－	－	－
	CY, A.2	71	AF	－	－	－
	CY, A.3	71	BF	－	－	－
	CY, A.4	71	CF	－	－	－
	CY, A.5	71	DF	－	－	－
	CY, A.6	71	EF	－	－	－
	CY, A.7	71	FF	－	－	－
	CY, PSW.0	71	0F	FA	－	－
	CY, PSW.1	71	1F	FA	－	－
	CY, PSW.2	71	2F	FA	－	－
	CY, PSW.3	71	3F	FA	－	－
	CY, PSW.4	71	4F	FA	－	－
	CY, PSW.5	71	5F	FA	－	－
	CY, PSW.6	71	6F	FA	－	－
	CY, PSW.7	71	7F	FA	－	－
	CY, [HL].0	71	87	－	－	－
	CY, [HL].1	71	97	－	－	－
	CY, [HL].2	71	A7	－	－	－
	CY, [HL].3	71	B7	－	－	－
	CY, [HL].4	71	C7	－	－	－
	CY, [HL].5	71	D7	－	－	－
	CY, [HL].6	71	E7	－	－	－
	CY, [HL].7	71	F7	－	－	－
	CY, ES:[HL].0	11	71	87	－	－
	CY, ES:[HL].1	11	71	97	－	－
	CY, ES:[HL].2	11	71	A7	－	－
	CY, ES:[HL].3	11	71	B7	－	－
	CY, ES:[HL].4	11	71	C7	－	－
	CY, ES:[HL].5	11	71	D7	－	－
	CY, ES:[HL].6	11	71	E7	－	－
	CY, ES:[HL].7	11	71	F7	－	－
SET1	saddr.0	71	02	saddr	－	－
	saddr.1	71	12	saddr	－	－
	saddr.2	71	22	saddr	－	－
	saddr.3	71	32	saddr	－	－
	saddr.4	71	42	saddr	－	－
	saddr.5	71	52	saddr	－	－
	saddr.6	71	62	saddr	－	－
	saddr.7	71	72	saddr	－	－

表5−8　命令フォーマット一覧（22/30）

ニモニック	オペランド	命令コード				
		1st	2nd	3rd	4th	5th
SET1	sfr.0	71	0A	sfr	−	−
	sfr.1	71	1A	sfr	−	−
	sfr.2	71	2A	sfr	−	−
	sfr.3	71	3A	sfr	−	−
	sfr.4	71	4A	sfr	−	−
	sfr.5	71	5A	sfr	−	−
	sfr.6	71	6A	sfr	−	−
	sfr.7	71	7A	sfr	−	−
	A.0	71	8A	−	−	−
	A.1	71	9A	−	−	−
	A.2	71	AA	−	−	−
	A.3	71	BA	−	−	−
	A.4	71	CA	−	−	−
	A.5	71	DA	−	−	−
	A.6	71	EA	−	−	−
	A.7	71	FA	−	−	−
	!addr16.0	71	00	adrl	adrh	−
	!addr16.1	71	10	adrl	adrh	−
	!addr16.2	71	20	adrl	adrh	−
	!addr16.3	71	30	adrl	adrh	−
	!addr16.4	71	40	adrl	adrh	−
	!addr16.5	71	50	adrl	adrh	−
	!addr16.6	71	60	adrl	adrh	−
	!addr16.7	71	70	adrl	adrh	−
	PSW.0	71	0A	FA	−	−
	PSW.1	71	1A	FA	−	−
	PSW.2	71	2A	FA	−	−
	PSW.3	71	3A	FA	−	−
	PSW.4	71	4A	FA	−	−
	PSW.5	71	5A	FA	−	−
	PSW.6	71	6A	FA	−	−
	PSW.7	71	7A	FA	−	−
	[HL].0	71	82	−	−	−
	[HL].1	71	92	−	−	−
	[HL].2	71	A2	−	−	−
	[HL].3	71	B2	−	−	−
	[HL].4	71	C2	−	−	−
	[HL].5	71	D2	−	−	−
	[HL].6	71	E2	−	−	−
	[HL].7	71	F2	−	−	−

表5－8　命令フォーマット一覧（23/30）

ニモニック	オペランド	命令コード				
		1st	2nd	3rd	4th	5th
SET1	ES:!addr16.0	11	71	00	adrl	adrh
	ES:!addr16.1	11	71	10	adrl	adrh
	ES:!addr16.2	11	71	20	adrl	adrh
	ES:!addr16.3	11	71	30	adrl	adrh
	ES:!addr16.4	11	71	40	adrl	adrh
	ES:!addr16.5	11	71	50	adrl	adrh
	ES:!addr16.6	11	71	60	adrl	adrh
	ES:!addr16.7	11	71	70	adrl	adrh
	ES:[HL].0	11	71	82	－	－
	ES:[HL].1	11	71	92	－	－
	ES:[HL].2	11	71	A2	－	－
	ES:[HL].3	11	71	B2	－	－
	ES:[HL].4	11	71	C2	－	－
	ES:[HL].5	11	71	D2	－	－
	ES:[HL].6	11	71	E2	－	－
	ES:[HL].7	11	71	F2	－	－
CLR1	saddr.0	71	03	saddr	－	－
	saddr.1	71	13	saddr	－	－
	saddr.2	71	23	saddr	－	－
	saddr.3	71	33	saddr	－	－
	saddr.4	71	43	saddr	－	－
	saddr.5	71	53	saddr	－	－
	saddr.6	71	63	saddr	－	－
	saddr.7	71	73	saddr	－	－
	sfr.0	71	0B	sfr	－	－
	sfr.1	71	1B	sfr	－	－
	sfr.2	71	2B	sfr	－	－
	sfr.3	71	3B	sfr	－	－
	sfr.4	71	4B	sfr	－	－
	sfr.5	71	5B	sfr	－	－
	sfr.6	71	6B	sfr	－	－
	sfr.7	71	7B	sfr	－	－
	A.0	71	8B	－	－	－
	A.1	71	9B	－	－	－
	A.2	71	AB	－	－	－
	A.3	71	BB	－	－	－
	A.4	71	CB	－	－	－
	A.5	71	DB	－	－	－
	A.6	71	EB	－	－	－
	A.7	71	FB	－	－	－

表5－8　命令フォーマット一覧（24/30）

ニモニック	オペランド	命令コード				
		1st	2nd	3rd	4th	5th
CLR1	!addr16.0	71	08	adrl	adrh	－
	!addr16.1	71	18	adrl	adrh	－
	!addr16.2	71	28	adrl	adrh	－
	!addr16.3	71	38	adrl	adrh	－
	!addr16.4	71	48	adrl	adrh	－
	!addr16.5	71	58	adrl	adrh	－
	!addr16.6	71	68	adrl	adrh	－
	!addr16.7	71	78	adrl	adrh	－
	PSW.0	71	0B	FA	－	－
	PSW.1	71	1B	FA	－	－
	PSW.2	71	2B	FA	－	－
	PSW.3	71	3B	FA	－	－
	PSW.4	71	4B	FA	－	－
	PSW.5	71	5B	FA	－	－
	PSW.6	71	6B	FA	－	－
	PSW.7	71	7B	FA	－	－
	[HL].0	71	83	－	－	－
	[HL].1	71	93	－	－	－
	[HL].2	71	A3	－	－	－
	[HL].3	71	B3	－	－	－
	[HL].4	71	C3	－	－	－
	[HL].5	71	D3	－	－	－
	[HL].6	71	E3	－	－	－
	[HL].7	71	F3	－	－	－
	ES:!addr16.0	11	71	08	adrl	adrh
	ES:!addr16.1	11	71	18	adrl	adrh
	ES:!addr16.2	11	71	28	adrl	adrh
	ES:!addr16.3	11	71	38	adrl	adrh
	ES:!addr16.4	11	71	48	adrl	adrh
	ES:!addr16.5	11	71	58	adrl	adrh
	ES:!addr16.6	11	71	68	adrl	adrh
	ES:!addr16.7	11	71	78	adrl	adrh
	ES:[HL].0	11	71	83	－	－
	ES:[HL].1	11	71	93	－	－
	ES:[HL].2	11	71	A3	－	－
	ES:[HL].3	11	71	B3	－	－
	ES:[HL].4	11	71	C3	－	－
	ES:[HL].5	11	71	D3	－	－
	ES:[HL].6	11	71	E3	－	－
	ES:[HL].7	11	71	F3	－	－
SET1	CY	71	80	－	－	－
CLR1	CY	71	88	－	－	－
NOT1	CY	71	C0	－	－	－

表5－8　命令フォーマット一覧（25/30）

ニモニック	オペランド	命令コード				
		1st	2nd	3rd	4th	5th
CALL	AX	61	CA	－	－	－
	BC	61	DA	－	－	－
	DE	61	EA	－	－	－
	HL	61	FA	－	－	－
	$!addr20	FE	adrl	adrh	－	－
	!addr16	FD	adrl	adrh	－	－
	!!addr20	FC	adrl	adrh	adrs	－
CALLT	[0080h]	61	84	－	－	－
	[0082h]	61	94	－	－	－
	[0084h]	61	A4	－	－	－
	[0086h]	61	B4	－	－	－
	[0088h]	61	C4	－	－	－
	[008Ah]	61	D4	－	－	－
	[008Ch]	61	E4	－	－	－
	[008Eh]	61	F4	－	－	－
	[0090h]	61	85	－	－	－
	[0092h]	61	95	－	－	－
	[0094h]	61	A5	－	－	－
	[0096h]	61	B5	－	－	－
	[0098h]	61	C5	－	－	－
	[009Ah]	61	D5	－	－	－
	[009Ch]	61	E5	－	－	－
	[009Eh]	61	F5	－	－	－
	[00A0h]	61	86	－	－	－
	[00A2h]	61	96	－	－	－
	[00A4h]	61	A6	－	－	－
	[00A6h]	61	B6	－	－	－
	[00A8h]	61	C6	－	－	－
	[00AAh]	61	D6	－	－	－
	[00ACh]	61	E6	－	－	－
	[00AEh]	61	F6	－	－	－
	[00B0h]	61	87	－	－	－
	[00B2h]	61	97	－	－	－
	[00B4h]	61	A7	－	－	－
	[00B6h]	61	B7	－	－	－
	[00B8h]	61	C7	－	－	－
	[00BAh]	61	D7	－	－	－
	[00BCh]	61	E7	－	－	－
	[00BEh]	61	F7	－	－	－
BRK	－	61	CC	－	－	－
RET	－	D7	－	－	－	－
RETI	－	61	FC	－	－	－
RETB	－	61	EC	－	－	－

表5−8　命令フォーマット一覧（26/30）

ニモニック	オペランド	命令コード				
		1st	2nd	3rd	4th	5th
PUSH	PSW	61	DD	−	−	−
	AX	C1	−	−	−	−
	BC	C3	−	−	−	−
	DE	C5	−	−	−	−
	HL	C7	−	−	−	−
POP	PSW	61	CD	−	−	−
	AX	C0	−	−	−	−
	BC	C2	−	−	−	−
	DE	C4	−	−	−	−
	HL	C6	−	−	−	−
MOVW	SP, #word	CB	F8	datal	datah	−
	SP, AX	BE	F8	−	−	−
	AX, SP	AE	F8	−	−	−
	BC, SP	DB	adrl	adrh	−	−
	DE, SP	EB	adrl	adrh	−	−
	HL, SP	FB	adrl	adrh	−	−
ADDW	SP, #byte	10	data	−	−	−
SUBW	SP, #byte	20	data	−	−	−
BR	AX	61	CB	−	−	−
	$addr20	EF	adr	−	−	−
	$!addr20	EE	adrl	adrh	−	−
	!addr16	ED	adrl	adrh	−	−
	!!addr20	EC	adrl	adrh	adrs	−
BC	$addr20	DC	adr	−	−	−
BNC	$addr20	DE	adr	−	−	−
BZ	$addr20	DD	adr	−	−	−
BNZ	$addr20	DF	adr	−	−	−
BH	$addr20	61	C3	adr	−	−
BNH	$addr20	61	D3	adr	−	−
BT	saddr.0, $addr20	31	02	saddr	adr	−
	saddr.1, $addr20	31	12	saddr	adr	−
	saddr.2, $addr20	31	22	saddr	adr	−
	saddr.3, $addr20	31	32	saddr	adr	−
	saddr.4, $addr20	31	42	saddr	adr	−
	saddr.5, $addr20	31	52	saddr	adr	−
	saddr.6, $addr20	31	62	saddr	adr	−
	saddr.7, $addr20	31	72	saddr	adr	−
	sfr.0, $addr20	31	82	sfr	adr	−
	sfr.1, $addr20	31	92	sfr	adr	−
	sfr.2, $addr20	31	A2	sfr	adr	−
	sfr.3, $addr20	31	B2	sfr	adr	−
	sfr.4, $addr20	31	C2	sfr	adr	−

表5−8　命令フォーマット一覧（27/30）

ニモニック	オペランド	命令コード				
		1st	2nd	3rd	4th	5th
BT	sfr.5, $addr20	31	D2	sfr	adr	−
	sfr.6, $addr20	31	E2	sfr	adr	−
	sfr.7, $addr20	31	F2	sfr	adr	−
	A.0, $addr20	31	03	adr	−	−
	A.1, $addr20	31	13	adr	−	−
	A.2, $addr20	31	23	adr	−	−
	A.3, $addr20	31	33	adr	−	−
	A.4, $addr20	31	43	adr	−	−
	A.5, $addr20	31	53	adr	−	−
	A.6, $addr20	31	63	adr	−	−
	A.7, $addr20	31	73	adr	−	−
	PSW.0, $addr20	31	82	FA	adr	−
	PSW.1, $addr20	31	92	FA	adr	−
	PSW.2, $addr20	31	A2	FA	adr	−
	PSW.3, $addr20	31	B2	FA	adr	−
	PSW.4, $addr20	31	C2	FA	adr	−
	PSW.5, $addr20	31	D2	FA	adr	−
	PSW.6, $addr20	31	E2	FA	adr	−
	PSW.7, $addr20	31	F2	FA	adr	−
	[HL].0, $addr20	31	83	adr	−	−
	[HL].1, $addr20	31	93	adr	−	−
	[HL].2, $addr20	31	A3	adr	−	−
	[HL].3, $addr20	31	B3	adr	−	−
	[HL].4, $addr20	31	C3	adr	−	−
	[HL].5, $addr20	31	D3	adr	−	−
	[HL].6, $addr20	31	E3	adr	−	−
	[HL].7, $addr20	31	F3	adr	−	−
	ES:[HL].0, $addr20	11	31	83	adr	−
	ES:[HL].1, $addr20	11	31	93	adr	−
	ES:[HL].2, $addr20	11	31	A3	adr	−
	ES:[HL].3, $addr20	11	31	B3	adr	−
	ES:[HL].4, $addr20	11	31	C3	adr	−
	ES:[HL].5, $addr20	11	31	D3	adr	−
	ES:[HL].6, $addr20	11	31	E3	adr	−
	ES:[HL].7, $addr20	11	31	F3	adr	−
BF	saddr.0, $addr20	31	04	saddr	adr	−
	saddr.1, $addr20	31	14	saddr	adr	−
	saddr.2, $addr20	31	24	saddr	adr	−
	saddr.3, $addr20	31	34	saddr	adr	−
	saddr.4, $addr20	31	44	saddr	adr	−
	saddr.5, $addr20	31	54	saddr	adr	−
	saddr.6, $addr20	31	64	saddr	adr	−
	saddr.7,$addr20	31	74	saddr	adr	−

表5－8　命令フォーマット一覧（28/30）

ニモニック	オペランド	命令コード				
		1st	2nd	3rd	4th	5th
BF	sfr.0, $addr20	31	84	sfr	adr	－
	sfr.1, $addr20	31	94	sfr	adr	－
	sfr.2, $addr20	31	A4	sfr	adr	－
	sfr.3,$addr20	31	B4	sfr	adr	－
	sfr.4, $addr20	31	C4	sfr	adr	－
	sfr.5, $addr20	31	D4	sfr	adr	－
	sfr.6,$addr20	31	E4	sfr	adr	－
	sfr.7, $addr20	31	F4	sfr	adr	－
	A.0, $addr20	31	05	adr	－	－
	A.1, $addr20	31	15	adr	－	－
	A.2, $addr20	31	25	adr	－	－
	A.3, $addr20	31	35	adr	－	－
	A.4, $addr20	31	45	adr	－	－
	A.5, $addr20	31	55	adr	－	－
	A.6, $addr20	31	65	adr	－	－
	A.7, $addr20	31	75	adr	－	－
	PSW.0, $addr20	31	84	FA	adr	－
	PSW.1, $addr20	31	94	FA	adr	－
	PSW.2, $addr20	31	A4	FA	adr	－
	PSW.3, $addr20	31	B4	FA	adr	－
	PSW.4, $addr20	31	C4	FA	adr	－
	PSW.5, $addr20	31	D4	FA	adr	－
	PSW.6, $addr20	31	E4	FA	adr	－
	PSW.7, $addr20	31	F4	FA	adr	－
	[HL].0, $addr20	31	85	adr	－	－
	[HL].1, $addr20	31	95	adr	－	－
	[HL].2, $addr20	31	A5	adr	－	－
	[HL].3, $addr20	31	B5	adr	－	－
	[HL].4, $addr20	31	C5	adr	－	－
	[HL].5, $addr20	31	D5	adr	－	－
	[HL].6, $addr20	31	E5	adr	－	－
	[HL].7, $addr20	31	F5	adr	－	－
	ES:[HL].0, $addr20	11	31	85	adr	－
	ES:[HL].1, $addr20	11	31	95	adr	－
	ES:[HL].2, $addr20	11	31	A5	adr	－
	ES:[HL].3, $addr20	11	31	B5	adr	－
	ES:[HL].4, $addr20	11	31	C5	adr	－
	ES:[HL].5, $addr20	11	31	D5	adr	－
	ES:[HL].6, $addr20	11	31	E5	adr	－
	ES:[HL].7, $addr20	11	31	F5	adr	－

表5－8　命令フォーマット一覧（29/30）

ニモニック	オペランド	命令コード				
		1st	2nd	3rd	4th	5th
BTCLR	saddr.0, $addr20	31	00	saddr	adr	－
	saddr.1, $addr20	31	10	saddr	adr	－
	saddr.2, $addr20	31	20	saddr	adr	－
	saddr.3, $addr20	31	30	saddr	adr	－
	saddr.4, $addr20	31	40	saddr	adr	－
	saddr.5, $addr20	31	50	saddr	adr	－
	saddr.6, $addr20	31	60	saddr	adr	－
	saddr.7, $addr20	31	70	saddr	adr	－
	sfr.0, $addr20	31	80	sfr	adr	－
	sfr.1, $addr20	31	90	sfr	adr	－
	sfr.2, $addr20	31	A0	sfr	adr	－
	sfr.3, $addr20	31	B0	sfr	adr	－
	sfr.4, $addr20	31	C0	sfr	adr	－
	sfr.5, $addr20	31	D0	sfr	adr	－
	sfr.6, $addr20	31	E0	sfr	adr	－
	sfr.7, $addr20	31	F0	sfr	adr	－
	A.0, $addr20	31	01	adr	－	－
	A.1, $addr20	31	11	adr	－	－
	A.2, $addr20	31	21	adr	－	－
	A.3, $addr20	31	31	adr	－	－
	A.4, $addr20	31	41	adr	－	－
	A.5, $addr20	31	51	adr	－	－
	A.6, $addr20	31	61	adr	－	－
	A.7, $addr20	31	71	adr	－	－
	PSW.0, $addr20	31	80	FA	adr	－
	PSW.1, $addr20	31	90	FA	adr	－
	PSW.2, $addr20	31	A0	FA	adr	－
	PSW.3, $addr20	31	B0	FA	adr	－
	PSW.4, $addr20	31	C0	FA	adr	－
	PSW.5, $addr20	31	D0	FA	adr	－
	PSW.6, $addr20	31	E0	FA	adr	－
	PSW.7, $addr20	31	F0	FA	adr	－
	[HL].0, $addr20	31	81	adr	－	－
	[HL].1, $addr20	31	91	adr	－	－
	[HL].2, $addr20	31	A1	adr	－	－
	[HL].3, $addr20	31	B1	adr	－	－
	[HL].4, $addr20	31	C1	adr	－	－
	[HL].5, $addr20	31	D1	adr	－	－
	[HL].6, $addr20	31	E1	adr	－	－
	[HL].7, $addr20	31	F1	adr	－	－

表5－8　命令フォーマット一覧（30/30）

ニモニック	オペランド	命令コード				
		1st	2nd	3rd	4th	5th
BTCLR	ES:[HL].0, $addr20	11	31	81	adr	－
	ES:[HL].1, $addr20	11	31	91	adr	－
	ES:[HL].2, $addr20	11	31	A1	adr	－
	ES:[HL].3, $addr20	11	31	B1	adr	－
	ES:[HL].4, $addr20	11	31	C1	adr	－
	ES:[HL].5, $addr20	11	31	D1	adr	－
	ES:[HL].6, $addr20	11	31	E1	adr	－
	ES:[HL].7, $addr20	11	31	F1	adr	－
SKC	－	61	C8	－	－	－
SKNC	－	61	D8	－	－	－
SKZ	－	61	E8	－	－	－
SKNZ	－	61	F8	－	－	－
SKH	－	61	E3	－	－	－
SKNH	－	61	F3	－	－	－
SEL[注]	RB0	61	CF	－	－	－
	RB1	61	DF	－	－	－
	RB2	61	EF	－	－	－
	RB3	61	FF	－	－	－
NOP	－	00	－	－	－	－
EI	－	71	7A	FA	－	－
DI	－	71	7B	FA	－	－
HALT	－	61	ED	－	－	－
STOP	－	61	FD	－	－	－
PREFIX	－	11	－	－	－	－

注　RL78-S1コアにはありません。

5.7　命令マップ

命令マップを表5－9から表5－12に示します。

表5−9　命令マップ（1st MAP）

	0(low)	1(low)	2(low)	3(low)	4(low)	5(low)	6(low)	7(low)	8(low)	9(low)	a(low)	b(low)	c(low)	d(low)	e(low)	f(low)
0	NOP	ADDW AX,AX	ADDW AX,!addr16	ADDW AX,BC	ADDW AX,#word	ADDW AX,DE	ADDW AX,saddrp	ADDW AX,HL	XCH A,X	MOV A,word[B]	ADD saddr,#byte	ADD A,saddr	ADD A,#byte	ADD A,[HL]	ADD A,[HL+byte]	ADD A,!addr16
1	ADDW SP,#byte	PREFIX	MOVW BC,AX	MOVW AX,BC	MOVW DE,AX	MOVW AX,DE	MOVW HL,AX	MOVW AX,HL	MOV word[B],A	MOV word[B],#byte	ADDC saddr,#byte	ADDC A,saddr	ADDC A,#byte	ADDC A,[HL]	ADDC A,[HL+byte]	ADDC A,!addr16
2	SUBW SP,#byte		SUBW AX,!addr16	SUBW AX,BC	SUBW AX,#word	SUBW AX,DE	SUBW AX,saddrp	SUBW AX,HL	MOV word[C],A	MOV A,word[C]	SUB saddr,#byte	SUB A,saddr	SUB A,#byte	SUB A,[HL]	SUB A,[HL+byte]	SUB A,!addr16
3	MOVW AX,#word	4th MAP	MOVW BC,#word	XCHW AX,BC	MOVW DE,#word	XCHW AX,DE	MOVW HL,#word	XCHW AX,HL	MOV word[C],#byte	MOV word[BC],#byte	SUBC saddr,#byte	SUBC A,saddr	SUBC A,#byte	SUBC A,[HL]	SUBC A,[HL+byte]	SUBC A,!addr16
4	CMP !addr16,#byte	MOV ES,#byte	CMPW AX,!addr16	CMPW AX,BC	CMPW AX,#word	CMPW AX,DE	CMPW AX,saddrp	CMPW AX,HL	MOV word[BC],A	MOV A,word[BC]	CMP saddr,#byte	CMP A,saddr	CMP A,#byte	CMP A,[HL]	CMP A,[HL+byte]	CMP A,!addr16
5	MOV X,#byte	MOV A,#byte	MOV C,#byte	MOV B,#byte	MOV E,#byte	MOV D,#byte	MOV L,#byte	MOV H,#byte	MOVW word[B],AX	MOVW AX,word[B]	AND saddr,#byte	AND A,saddr	AND A,#byte	AND A,[HL]	AND A,[HL+byte]	AND A,!addr16
6	MOV A,X	2nd MAP	MOV A,C	MOV A,B	MOV A,E	MOV A,D	MOV A,L	MOV A,H	MOVW word[C],AX	MOVW AX,word[C]	OR saddr,#byte	OR A,saddr	OR A,#byte	OR A,[HL]	OR A,[HL+byte]	OR A,!addr16
7	MOV X,A	3rd MAP	MOV C,A	MOV B,A	MOV E,A	MOV D,A	MOV L,A	MOV H,A	MOVW word[BC],AX	MOVW AX,word[BC]	XOR saddr,#byte	XOR A,saddr	XOR A,#byte	XOR A,[HL]	XOR A,[HL+byte]	XOR A,!addr16
8	INC X	INC A	INC C	INC B	INC E	INC D	INC L	INC H	MOV A,[SP+byte]	MOV A,[DE]	MOV A,[DE+byte]	MOV A,[HL]	MOV A,[HL+byte]	MOV A,saddr	MOV A,sfr	MOV A,!addr16
9	DEC X	DEC A	DEC C	DEC B	DEC E	DEC D	DEC L	DEC H	MOV [SP+byte],A	MOV [DE],A	MOV [DE+byte],A	MOV [HL],A	MOV [HL+byte],A	MOV saddr,A	MOV sfr,A	MOV !addr16,A
a	INC !addr16	INCW AX	INCW !addr16	INCW BC	INC saddr	INCW DE	INCW saddrp	INCW HL	MOVW AX,[SP+byte]	MOVW AX,[DE]	MOVW AX,[DE+byte]	MOVW AX,[HL]	MOVW AX,[HL+byte]	MOVW AX,saddrp	MOVW AX,sfrp	MOVW AX,!addr16
b	DEC !addr16	DECW AX	DECW !addr16	DECW BC	DEC saddr	DECW DE	DECW saddrp	DECW HL	MOVW [SP+byte],AX	MOVW [DE],AX	MOVW [DE+byte],AX	MOVW [HL],AX	MOVW [HL+byte],AX	MOVW saddrp,AX	MOVW sfrp,AX	MOVW !addr16,AX
c	POP AX	PUSH AX	POP BC	PUSH BC	POP DE	PUSH DE	POP HL	PUSH HL	MOV [SP+byte],#byte	MOVW saddrp,#word	MOV [DE+byte],#byte	MOVW sfrp,#word	MOV [HL+byte],#byte	MOV saddr,#byte	注 MOV sfr,#byte	MOV !addr16,#byte
d	CMP0 X	CMP0 A	CMP0 C	CMP0 B	CMP0 saddr	CMP0 !addr16	MULU X	RET	MOV X,saddr	MOV X,!addr16	MOVW BC,saddrp	MOVW BC,!addr16	BC $addr20	BZ $addr20	BNC $addr20	BNZ $addr20
e	ONEB X	ONEB A	ONEB C	ONEB B	ONEB saddr	ONEB !addr16	ONEW AX	ONEW BC	MOV B,saddr	MOV B,!addr16	MOVW DE,saddrp	MOVW DE,!addr16	BR !!addr20	BR !addr16	BR $!addr20	BR $addr20
f	CLRB X	CLRB A	CLRB C	CLRB B	CLRB saddr	CLRB !addr16	CLRW AX	CLRW BC	MOV C,saddr	MOV C,!addr16	MOVW HL,saddrp	MOVW HL,!addr16	CALL !!addr20	CALL !addr16	CALL $!addr20	

注　乗除積和算命令のMULHU, MULH, DIVHU, DIVWU, MACHU, MACHもここにマッピングされます。

表5－10　命令マップ（2nd MAP）

	0(low)	1(low)	2(low)	3(low)	4(low)	5(low)	6(low)	7(low)	8(low)	9(low)	a(low)	b(low)	c(low)	d(low)	e(low)	f(low)
0	ADD X,A	ADD A,A	ADD C,A	ADD B,A	ADD E,A	ADD D,A	ADD L,A	ADD H,A	ADD A,X	ADDW AX,[HL+byte]	ADD A,C	ADD A,B	ADD A,E	ADD A,D	ADD A,L	ADD A,H
1	ADDC X,A	ADDC A,A	ADDC C,A	ADDC B,A	ADDC E,A	ADDC D,A	ADDC L,A	ADDC H,A	ADDC A,X		ADDC A,C	ADDC A,B	ADDC A,E	ADDC A,D	ADDC A,L	ADDC A,H
2	SUB X,A	SUB A,A	SUB C,A	SUB B,A	SUB E,A	SUB D,A	SUB L,A	SUB H,A	SUB A,X	SUBW AX,[HL+byte]	SUB A,C	SUB A,B	SUB A,E	SUB A,D	SUB A,L	SUB A,H
3	SUBC X,A	SUBC A,A	SUBC C,A	SUBC B,A	SUBC E,A	SUBC D,A	SUBC L,A	SUBC H,A	SUBC A,X		SUBC A,C	SUBC A,B	SUBC A,E	SUBC A,D	SUBC A,L	SUBC A,H
4	CMP X,A	CMP A,A	CMP C,A	CMP B,A	CMP E,A	CMP D,A	CMP L,A	CMP H,A	CMP A,X	CMPW AX,[HL+byte]	CMP A,C	CMP A,B	CMP A,E	CMP A,D	CMP A,L	CMP A,H
5	AND X,A	AND A,A	AND C,A	AND B,A	AND E,A	AND D,A	AND L,A	AND H,A	AND A,X	INC [HL+byte]	AND A,C	AND A,B	AND A,E	AND A,D	AND A,L	AND A,H
6	OR X,A	OR A,A	OR C,A	OR B,A	OR E,A	OR D,A	OR L,A	OR H,A	OR A,X	DEC [HL+byte]	OR A,C	OR A,B	OR A,E	OR A,D	OR A,L	OR A,H
7	XOR X,A	XOR A,A	XOR C,A	XOR B,A	XOR E,A	XOR D,A	XOR L,A	XOR H,A	XOR A,X	INCW [HL+byte]	XOR A,C	XOR A,B	XOR A,E	XOR A,D	XOR A,L	XOR A,H
8	ADD A,[HL+B]		ADD A,[HL+C]		CALLT [0080h]	CALLT [0090h]	CALLT [00A0h]	CALLT [00B0h]		DECW [HL+byte]	XCH A,C	XCH A,B	XCH A,E	XCH A,D	XCH A,L	XCH A,H
9	ADDC A,[HL+B]		ADDC A,[HL+C]		CALLT [0082h]	CALLT [0092h]	CALLT [00A2h]	CALLT [00B2h]								
a	SUB A,[HL+B]		SUB A,[HL+C]		CALLT [0084h]	CALLT [0094h]	CALLT [00A4h]	CALLT [00B4h]	XCH A,saddr	XCH A,[HL+C]	XCH A,!addr16	XCH A,sfr	XCH A,[HL]	XCH A,[HL+byte]	XCH A,[DE]	XCH A,[DE+byte]
b	SUBC A,[HL+B]		SUBC A,[HL+C]		CALLT [0086h]	CALLT [0096h]	CALLT [00A6h]	CALLT [00B6h]	MOV ES,saddr	XCH A,[HL+B]						
c	CMP A,[HL+B]		CMP A,[HL+C]	BH $addr20	CALLT [0088h]	CALLT [0098h]	CALLT [00A8h]	CALLT [00B8h]	SKC	MOV A,[HL+B]	CALL AX	BR AX	BRK	POP PSW	MOVS [HL+byte],X	SEL RB0 注
d	AND A,[HL+B]		AND A,[HL+C]	BNH $addr20	CALLT [008Ah]	CALLT [009Ah]	CALLT [00AAh]	CALLT [00BAh]	SKNC	MOV [HL+B],A	CALL BC	ROR A,1	ROLC A,1	PUSH PSW	CMPS X,[HL+byte]	SEL RB1 注
e	OR A,[HL+B]		OR A,[HL+C]	SKH	CALLT [008Ch]	CALLT [009Ch]	CALLT [00ACh]	CALLT [00BCh]	SKZ	MOV A,[HL+C]	CALL DE	ROL A,1	RETB	HALT	ROLWC AX,1	SEL RB2 注
f	XOR A,[HL+B]		XOR A,[HL+C]	SKNH	CALLT [008Eh]	CALLT [009Eh]	CALLT [00AEh]	CALLT [00BEh]	SKNZ	MOV [HL+C],A	CALL HL	RORC A,1	RETI	STOP	ROLWC BC,1	SEL RB3 注

注　RL78-S1コアにはありません。

表5－11　命令マップ（3rd MAP）

	0(low)	1(low)	2(low)	3(low)	4(low)	5(low)	6(low)	7(low)	8(low)	9(low)	a(low)	b(low)	c(low)	d(low)	e(low)	f(low)
0	SET1 !addr16.0	MOV1 saddr.0,CY	SET1 saddr.0	CLR1 saddr.0	MOV1 CY,saddr.0	AND1 CY,saddr.0	OR1 CY,saddr.0	XOR1 CY,saddr.0	CLR1 !addr16.0	MOV1 sfr.0,CY	SET1 sfr.0	CLR1 sfr.0	MOV1 CY,sfr.0	AND1 CY,sfr.0	OR1 CY,sfr.0	XOR1 CY,sfr.0
1	SET1 !addr16.1	MOV1 saddr.1,CY	SET1 saddr.1	CLR1 saddr.1	MOV1 CY,saddr.1	AND1 CY,saddr.1	OR1 CY,saddr.1	XOR1 CY,saddr.1	CLR1 !addr16.1	MOV1 sfr.1,CY	SET1 sfr.1	CLR1 sfr.1	MOV1 CY,sfr.1	AND1 CY,sfr.1	OR1 CY,sfr.1	XOR1 CY,sfr.1
2	SET1 !addr16.2	MOV1 saddr.2,CY	SET1 saddr.2	CLR1 saddr.2	MOV1 CY,saddr.2	AND1 CY,saddr.2	OR1 CY,saddr.2	XOR1 CY,saddr.2	CLR1 !addr16.2	MOV1 sfr.2,CY	SET1 sfr.2	CLR1 sfr.2	MOV1 CY,sfr.2	AND1 CY,sfr.2	OR1 CY,sfr.2	XOR1 CY,sfr.2
3	SET1 !addr16.3	MOV1 saddr.3,CY	SET1 saddr.3	CLR1 saddr.3	MOV1 CY,saddr.3	AND1 CY,saddr.3	OR1 CY,saddr.3	XOR1 CY,saddr.3	CLR1 !addr16.3	MOV1 sfr.3,CY	SET1 sfr.3	CLR1 sfr.3	MOV1 CY,sfr.3	AND1 CY,sfr.3	OR1 CY,sfr.3	XOR1 CY,sfr.3
4	SET1 !addr16.4	MOV1 saddr.4,CY	SET1 saddr.4	CLR1 saddr.4	MOV1 CY,saddr.4	AND1 CY,saddr.4	OR1 CY,saddr.4	XOR1 CY,saddr.4	CLR1 !addr16.4	MOV1 sfr.4,CY	SET1 sfr.4	CLR1 sfr.4	MOV1 CY,sfr.4	AND1 CY,sfr.4	OR1 CY,sfr.4	XOR1 CY,sfr.4
5	SET1 !addr16.5	MOV1 saddr.5,CY	SET1 saddr.5	CLR1 saddr.5	MOV1 CY,saddr.5	AND1 CY,saddr.5	OR1 CY,saddr.5	XOR1 CY,saddr.5	CLR1 !addr16.5	MOV1 sfr.5,CY	SET1 sfr.5	CLR1 sfr.5	MOV1 CY,sfr.5	AND1 CY,sfr.5	OR1 CY,sfr.5	XOR1 CY,sfr.5
6	SET1 !addr16.6	MOV1 saddr.6,CY	SET1 saddr.6	CLR1 saddr.6	MOV1 CY,saddr.6	AND1 CY,saddr.6	OR1 CY,saddr.6	XOR1 CY,saddr.6	CLR1 !addr16.6	MOV1 sfr.6,CY	SET1 sfr.6	CLR1 sfr.6	MOV1 CY,sfr.6	AND1 CY,sfr.6	OR1 CY,sfr.6	XOR1 CY,sfr.6
7	SET1 !addr16.7	MOV1 saddr.7,CY	SET1 saddr.7	CLR1 saddr.7	MOV1 CY,saddr.7	AND1 CY,saddr.7	OR1 CY,saddr.7	XOR1 CY,saddr.7	CLR1 !addr16.7	MOV1 sfr.7,CY	SET1 sfr.7	CLR1 sfr.7	MOV1 CY,sfr.7	AND1 CY,sfr.7	OR1 CY,sfr.7	XOR1 CY,sfr.7
8	SET1 CY	MOV1 [HL].0,CY	SET1 [HL].0	CLR1 [HL].0	MOV1 CY,[HL].0	AND1 CY,[HL].0	OR1 CY,[HL].0	XOR1 CY,[HL].0	CLR1 CY	MOV1 A.0,CY	SET1 A.0	CLR1 A.0	MOV1 CY,A.0	AND1 CY,A.0	OR1 CY,A.0	XOR1 CY,A.0
9		MOV1 [HL].1,CY	SET1 [HL].1	CLR1 [HL].1	MOV1 CY,[HL].1	AND1 CY,[HL].1	OR1 CY,[HL].1	XOR1 CY,[HL].1		MOV1 A.1,CY	SET1 A.1	CLR1 A.1	MOV1 CY,A.1	AND1 CY,A.1	OR1 CY,A.1	XOR1 CY,A.1
a		MOV1 [HL].2,CY	SET1 [HL].2	CLR1 [HL].2	MOV1 CY,[HL].2	AND1 CY,[HL].2	OR1 CY,[HL].2	XOR1 CY,[HL].2		MOV1 A.2,CY	SET1 A.2	CLR1 A.2	MOV1 CY,A.2	AND1 CY,A.2	OR1 CY,A.2	XOR1 CY,A.2
b		MOV1 [HL].3,CY	SET1 [HL].3	CLR1 [HL].3	MOV1 CY,[HL].3	AND1 CY,[HL].3	OR1 CY,[HL].3	XOR1 CY,[HL].3		MOV1 A.3,CY	SET1 A.3	CLR1 A.3	MOV1 CY,A.3	AND1 CY,A.3	OR1 CY,A.3	XOR1 CY,A.3
c	NOT1 CY	MOV1 [HL].4,CY	SET1 [HL].4	CLR1 [HL].4	MOV1 CY,[HL].4	AND1 CY,[HL].4	OR1 CY,[HL].4	XOR1 CY,[HL].4		MOV1 A.4,CY	SET1 A.4	CLR1 A.4	MOV1 CY,A.4	AND1 CY,A.4	OR1 CY,A.4	XOR1 CY,A.4
d		MOV1 [HL].5,CY	SET1 [HL].5	CLR1 [HL].5	MOV1 CY,[HL].5	AND1 CY,[HL].5	OR1 CY,[HL].5	XOR1 CY,[HL].5		MOV1 A.5,CY	SET1 A.5	CLR1 A.5	MOV1 CY,A.5	AND1 CY,A.5	OR1 CY,A.5	XOR1 CY,A.5
e		MOV1 [HL].6,CY	SET1 [HL].6	CLR1 [HL].6	MOV1 CY,[HL].6	AND1 CY,[HL].6	OR1 CY,[HL].6	XOR1 CY,[HL].6		MOV1 A.6,CY	SET1 A.6	CLR1 A.6	MOV1 CY,A.6	AND1 CY,A.6	OR1 CY,A.6	XOR1 CY,A.6
f		MOV1 [HL].7,CY	SET1 [HL].7	CLR1 [HL].7	MOV1 CY,[HL].7	AND1 CY,[HL].7	OR1 CY,[HL].7	XOR1 CY,[HL].7		MOV1 A.7,CY	SET1 A.7	CLR1 A.7	MOV1 CY,A.7	AND1 CY,A.7	OR1 CY,A.7	XOR1 CY,A.7

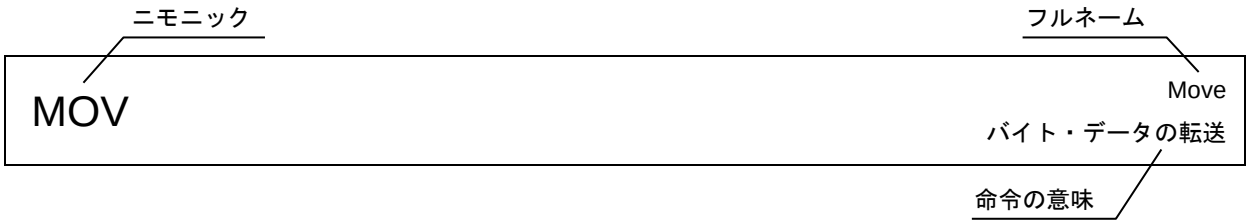
表5－12　命令マップ（4th MAP）

	0(low)	1(low)	2(low)	3(low)	4(low)	5(low)	6(low)	7(low)	8(low)	9(low)	a(low)	b(low)	c(low)	d(low)	e(low)	f(low)
0	BTCLR saddr.0,$addr20	BTCLR A.0,$addr20	BT saddr.0,$addr20	BT A.0,$addr20	BF saddr.0,$addr20	BF A.0,$addr20										
1	BTCLR saddr.1,$addr20	BTCLR A.1,$addr20	BT saddr.1,$addr20	BT A.1,$addr20	BF saddr.1,$addr20	BF A.1,$addr20		SHL C,1	SHL B,1	SHL A,1	SHR A,1	SAR A,1	SHLW BC,1	SHLW AX,1	SHRW AX,1	SARW AX,1
2	BTCLR saddr.2,$addr20	BTCLR A.2,$addr20	BT saddr.2,$addr20	BT A.2,$addr20	BF saddr.2,$addr20	BF A.2,$addr20		SHL C,2	SHL B,2	SHL A,2	SHR A,2	SAR A,2	SHLW BC,2	SHLW AX,2	SHRW AX,2	SARW AX,2
3	BTCLR saddr.3,$addr20	BTCLR A.3,$addr20	BT saddr.3,$addr20	BT A.3,$addr20	BF saddr.3,$addr20	BF A.3,$addr20		SHL C,3	SHL B,3	SHL A,3	SHR A,3	SAR A,3	SHLW BC,3	SHLW AX,3	SHRW AX,3	SARW AX,3
4	BTCLR saddr.4,$addr20	BTCLR A.4,$addr20	BT saddr.4,$addr20	BT A.4,$addr20	BF saddr.4,$addr20	BF A.4,$addr20		SHL C,4	SHL B,4	SHL A,4	SHR A,4	SAR A,4	SHLW BC,4	SHLW AX,4	SHRW AX,4	SARW AX,4
5	BTCLR saddr.5,$addr20	BTCLR A.5,$addr20	BT saddr.5,$addr20	BT A.5,$addr20	BF saddr.5,$addr20	BF A.5,$addr20		SHL C,5	SHL B,5	SHL A,5	SHR A,5	SAR A,5	SHLW BC,5	SHLW AX,5	SHRW AX,5	SARW AX,5
6	BTCLR saddr.6,$addr20	BTCLR A.6,$addr20	BT saddr.6,$addr20	BT A.6,$addr20	BF saddr.6,$addr20	BF A.6,$addr20		SHL C,6	SHL B,6	SHL A,6	SHR A,6	SAR A,6	SHLW BC,6	SHLW AX,6	SHRW AX,6	SARW AX,6
7	BTCLR saddr.7,$addr20	BTCLR A.7,$addr20	BT saddr.7,$addr20	BT A.7,$addr20	BF saddr.7,$addr20	BF A.7,$addr20		SHL C,7	SHL B,7	SHL A,7	SHR A,7	SAR A,7	SHLW BC,7	SHLW AX,7	SHRW AX,7	SARW AX,7
8	BTCLR sfr.0,$addr20	BTCLR [HL].0,$addr20	BT sfr.0,$addr20	BT [HL].0,$addr20	BF sfr.0,$addr20	BF [HL].0,$addr20							SHLW BC,8	SHLW AX,8	SHRW AX,8	SARW AX,8
9	BTCLR sfr.1,$addr20	BTCLR [HL].1,$addr20	BT sfr.1,$addr20	BT [HL].1,$addr20	BF sfr.1,$addr20	BF [HL].1,$addr20							SHLW BC,9	SHLW AX,9	SHRW AX,9	SARW AX,9
a	BTCLR sfr.2,$addr20	BTCLR [HL].2,$addr20	BT sfr.2,$addr20	BT [HL].2,$addr20	BF sfr.2,$addr20	BF [HL].2,$addr20							SHLW BC,10	SHLW AX,10	SHRW AX,10	SARW AX,10
b	BTCLR sfr.3,$addr20	BTCLR [HL].3,$addr20	BT sfr.3,$addr20	BT [HL].3,$addr20	BF sfr.3,$addr20	BF [HL].3,$addr20							SHLW BC,11	SHLW AX,11	SHRW AX,11	SARW AX,11
c	BTCLR sfr.4,$addr20	BTCLR [HL].4,$addr20	BT sfr.4,$addr20	BT [HL].4,$addr20	BF sfr.4,$addr20	BF [HL].4,$addr20							SHLW BC,12	SHLW AX,12	SHRW AX,12	SARW AX,12
d	BTCLR sfr.5,$addr20	BTCLR [HL].5,$addr20	BT sfr.5,$addr20	BT [HL].5,$addr20	BF sfr.5,$addr20	BF [HL].5,$addr20							SHLW BC,13	SHLW AX,13	SHRW AX,13	SARW AX,13
e	BTCLR sfr.6,$addr20	BTCLR [HL].6,$addr20	BT sfr.6,$addr20	BT [HL].6,$addr20	BF sfr.6,$addr20	BF [HL].6,$addr20							SHLW BC,14	SHLW AX,14	SHRW AX,14	SARW AX,14
f	BTCLR sfr.7,$addr20	BTCLR [HL].7,$addr20	BT sfr.7,$addr20	BT [HL].7,$addr20	BF sfr.7,$addr20	BF [HL].7,$addr20							SHLW BC,15	SHLW AX,15	SHRW AX,15	SARW AX,15

第6章　命令の説明

この章では，RL78マイクロコントローラ製品の命令を説明します。

記　述　例



【命　令　形　式】　MOV dst, src　：命令の基本記述形式を示します。

【オペレーション】　dst ← src　　：略号を用いて命令のオペレーションを示します。

【オ ペ ラ ン ド】　：この命令で指定できるオペランドを示します。各オペランドの略号の説明は，「5.2　オペレーション欄の説明」を参照してください。

ニモニック	オペランド（dst, src）
MOV	r, #byte
	PSW, #byte
	A, PSW
	PSW, A

ニモニック	オペランド（dst, src）
MOV	A, saddr
	saddr, A
	A, [HL+byte]
	[HL+byte], A

【フ　ラ　グ】　：命令実行により変化するフラグの動作を示します。
各フラグの動作記号を凡例に示します。

Z	AC	CY

凡　例

記号	フラグ変化
ブランク	変化なし
0	0にクリアされる
1	1にセットされる
×	結果に従ってセットまたはクリアされる
R	以前に退避した値がリストアされる

【説　　明】　：命令のオペレーションの詳細を解説します。

●第1オペランドで指定されるデスティネーション・オペランド（dst）に，第2オペランドで指定されるソース・オペランド（src）の内容を転送します。

【記　述　例】

MOV A, #4DH；Aレジスタに4DHを転送

6.1　8ビット・データ転送命令

8ビット・データ転送命令には次の命令があります。

・MOV
・XCH
・ONEB
・CLRB
・MOVS

MOV

Move
バイト・データの転送

【命　令　形　式】　MOV dst, src

【オペレーション】　dst ← src

★　【オ ペ ラ ン ド】

ニモニック	オペランド（dst, src）
MOV	r, #byte
	PSW, #byte
	CS, #byte
	ES, #byte
	!addr16, #byte
	ES:!addr16, #byte
	saddr, #byte
	sfr, #byte
	[DE+byte], #byte
	ES:[DE+byte],#byte
	[HL+byte], #byte
	ES:[HL+byte],#byte
	[SP+byte], #byte
	word[B], #byte
	ES:word[B], #byte
	word[C], #byte
	ES:word[C], #byte
	word[BC], #byte
	ES:word[BC], #byte
	A, r[注]
	r, A[注]
	A, PSW
	PSW, A
	A, CS
	CS, A
	A, ES
	ES, A
	A, !addr16
	A, ES:!addr16
	!addr16, A
	ES:!addr16, A

ニモニック	オペランド（dst, src）
MOV	A, saddr
	saddr, A
	A, sfr
	sfr, A
	A, [DE]
	[DE], A
	A, ES:[DE]
	ES:[DE], A
	A, [HL]
	[HL], A
	A, ES:[HL]
	ES:[HL], A
	A, [DE+byte]
	[DE+byte], A
	A, ES:[DE+byte]
	ES:[DE+byte], A
	A, [HL+byte]
	[HL+byte], A
	A, ES:[HL+byte]
	ES:[HL+byte], A
	A, [SP+byte]
	[SP+byte], A
	A, word[B]
	word[B], A
	A, ES:word[B]
	ES:word[B], A
	A, word[C]
	word[C], A
	A, ES:word[C]
	ES:word[C], A
	A, word[BC]

ニモニック	オペランド（dst, src）
MOV	word[BC], A
	A, ES:word[BC]
	ES:word[BC], A
	A, [HL+B]
	[HL+B], A
	A, ES:[HL+B]
	ES:[HL+B], A
	A, [HL+C]
	[HL+C], A
	A, ES:[HL+C]
	ES:[HL+C], A
	X, !addr16
	X, ES:!addr16
	X, saddr
	B, !addr16
	B, ES:!addr16
	B, saddr
	C, !addr16
	C, ES:!addr16
	C, saddr
	ES, saddr

注　r = Aを除く。

【フ　ラ　グ】

PSW, #byteとPSW, A のオペランドの場合

Z	AC	CY
×	×	×

左記以外

Z	AC	CY

【説　　明】

●第1オペランドで指定されるデスティネーション・オペランド（dst）に，第2オペランドで指定されるソース・オペランド（src）の内容を転送します。

●MOV PSW, #byte命令，MOV PSW, A命令と次に続く命令の間では，すべての割り込みを受け付けません。

【記　述　例】

MOV A, #4DH；Aレジスタに4DHを転送

XCH

Exchange
バイト・データの交換

【命　令　形　式】　XCH dst, src

【オペレーション】　dst ⟷ src

【オ ペ ラ ン ド】

ニモニック	オペランド（dst, src）
XCH	A, r[注]
	A, !addr16
	A, ES:!addr16
	A, saddr
	A, sfr
	A, [DE]
	A, ES:[DE]
	A, [HL]
	A, ES:[HL]

ニモニック	オペランド（dst, src）
XCH	A, [DE+byte]
	A, ES:[DE+byte]
	A, [HL＋byte]
	A, ES:[HL+byte]
	A, [HL＋B]
	A, ES:[HL+B]
	A, [HL＋C]
	A, ES:[HL+C]

注　r = Aを除く。

【フ　　ラ　　グ】

Z	AC	CY

【説　　明】

●第1オペランドと第2オペランドの内容を交換します。

【記　　述　　例】

XCH A, FFEBCH；Aレジスタの内容とFFEBCH番地の内容を交換

ONEB

One byte
バイト・データの01Hセット

【命　令　形　式】　ONEB　dst

【オペレーション】　dst ← 01H

【オ　ペ　ラ　ン　ド】

ニモニック	オペランド（dst）
ONEB	A
	X
	B
	C
	!addr16
	ES:!addr16
	saddr

【フ　　ラ　　グ】

Z	AC	CY

【説　　明】

●第1オペランドで指定されるデスティネーション・オペランド（dst）に01Hを転送します。

【記　　述　　例】

ONEB　A ; Aレジスタに01Hを転送

CLRB

Clear byte
バイト・データのクリア

【命　令　形　式】　CLRB　dst

【オペレーション】　dst　←　00H

【オ ペ ラ ン ド】

ニモニック	オペランド（dst）
CLRB	A
	X
	B
	C
	!addr16
	ES:!addr16
	saddr

【フ　ラ　グ】

Z	AC	CY

【説　　明】

●第1オペランドで指定されるデスティネーション・オペランド（dst）に00Hを転送します。

【記　述　例】

CLRB　A ; Aレジスタに00Hを転送

MOVS

Move and change PSW
バイト・データの転送とPSW変化

【命　令　形　式】　MOVS dst, src

【オペレーション】　dst ← src

【オ ペ ラ ン ド】

ニモニック	オペランド（dst, src）
MOVS	[HL+byte], X
	ES:[HL+byte], X

【フ　　ラ　　グ】

Z	AC	CY
×		×

【説　　明】

●第1オペランドで指定されるデスティネーション・オペランド（dst）に，第2オペランドで指定されるソース・オペランドの内容を転送します。

●srcの値が0の場合，Zフラグがセット（1），その他の場合はZフラグはクリア（0）されます。

●Aレジスタの値が0，またはsrcの値が0であった場合，CYフラグがセット（1），その他の場合はCYフラグはクリア（0）されます。

【記　　述　　例】

MOVS [HL+2H], X ; HL = FE00H, X = 55H, A = 0Hの場合

Xの55HをFE02H 番地に格納

Zフラグ = 0

CYフラグ = 1（Aレジスタ0のため）

6.2　16ビット・データ転送命令

16ビット・データ転送命令には次の命令があります。

・MOVW
・XCHW
・ONEW
・CLRW

MOVW

Move Word
ワード・データの転送

【命　令　形　式】　MOVW dst, src

【オペレーション】　dst ← src

【オ ペ ラ ン ド】

ニモニック	オペランド（dst, src）
MOVW	rp, #word
	saddrp, #word
	sfrp, #word
	AX, rp[注]
	rp, AX[注]
	AX, !addr16
	!addr16, AX
	AX, ES:!addr16
	ES:!addr16, AX
	AX, saddrp
	saddrp, AX
	AX, sfrp
	sfrp, AX
	AX, [DE]
	[DE], AX
	AX, ES:[DE]
	ES:[DE], AX
	AX, [HL]
	[HL], AX
	AX, ES:[HL]
	ES:[HL], AX
	AX, [DE+byte]
	[DE+byte], AX
	AX, ES:[DE+byte]
	ES:[DE+byte], AX
	AX, [HL+byte]

ニモニック	オペランド（dst, src）
MOVW	[HL+byte], AX
	AX, ES:[HL+byte]
	ES:[HL+byte], AX
	AX, [SP+byte]
	[SP+byte], AX
	AX, word[B]
	word[B], AX
	AX, ES:word[B]
	ES:word[B], AX
	AX, word[C]
	word[C], AX
	AX, ES:word[C]
	ES:word[C], AX
	AX, word[BC]
	word[BC], AX
	AX, ES:word[BC]
	ES:word[BC], AX
	BC, !addr16
	BC, ES:!addr16
	DE, !addr16
	DE, ES:!addr16
	HL, !addr16
	HL, ES:!addr16
	BC, saddrp
	DE, saddrp
	HL, saddrp

注　rp = BC, DE, HLのときのみ。

【フ　ラ　グ】

Z	AC	CY

【説　　明】

●第1オペランドで指定されるデスティネーション・オペランド（dst）に，第2オペランドで指定されるソース・オペランド（src）の内容を転送します。

【記　述　例】

MOVW AX, HL；HLレジスタの内容をAXレジスタに転送

【注　　意】

偶数アドレスのみ指定できます。奇数アドレスは指定できません。

XCHW

Exchange Word
ワード・データの交換

【命　令　形　式】　XCHW dst, src

【オペレーション】　dst ⟷ src

【オ ペ ラ ン ド】

ニモニック	オペランド（dst, src）
XCHW	AX, rp[注]

注　rp = BC, DE, HLのときのみ。

【フ　　ラ　　グ】

Z	AC	CY

【説　　明】

●第1オペランドと第2オペランドの内容を交換します。

【記　　述　　例】

XCHW AX, BC；AXレジスタとBCレジスタの内容を交換

ONEW

One Word
ワード・データの0001Hセット

【命　令　形　式】　ONEW　dst

【オペレーション】　dst ← 0001H

【オ ペ ラ ン ド】

ニモニック	オペランド（dst）
ONEW	AX
	BC

【フ　　ラ　　グ】

Z	AC	CY

【説　　明】

●第1オペランドで指定されるデスティネーション・オペランド（dst）に0001Hを転送します。

【記　　述　　例】

ONEW　AX ; AXレジスタに0001Hを転送

CLRW

Clear Word
ワード・データのクリア

【命　令　形　式】　CLRW　dst

【オペレーション】　dst ← 0000H

【オ ペ ラ ン ド】

ニモニック	オペランド（dst）
CLRW	AX
	BC

【フ　　ラ　　グ】

Z	AC	CY

【説　　明】

●第1オペランドで指定されるデスティネーション・オペランド（dst）に0000Hを転送します。

【記　　述　　例】

CLRW　AX ; AXレジスタに0000Hを転送

6.3　8ビット演算命令

8ビット演算命令には次の命令があります。

・ADD
・ADDC
・SUB
・SUBC
・AND
・OR
・XOR
・CMP
・CMP0
・CMPS

ADD

Add
バイト・データの加算

【命　令　形　式】　ADD dst, src

【オペレーション】　dst, CY ← dst＋src

【オ　ペ　ラ　ン　ド】

ニモニック	オペランド（dst, src）
ADD	A, #byte
	saddr, #byte
	A, r[注]
	r, A
	A !addr16
	A, ES:!addr16
	A, saddr
	A,[HL]

ニモニック	オペランド（dst, src）
ADD	A, ES:[HL]
	A,[HL+byte]
	A, ES:[HL+byte]
	A,[HL+B]
	A, ES:[HL+B]
	A,[HL+C]
	A, ES:[HL+C]

注　r = Aを除く。

【フ　　ラ　　グ】

Z	AC	CY
×	×	×

【説　　明】

●第1オペランドで指定されるデスティネーション・オペランド（dst）と第2オペランドで指定されるソース・オペランド（src）を加算し，その結果をCYフラグとデスティネーション・オペランド（dst）に格納します。

●加算の結果，dstが0になった場合，Zフラグがセット（1），その他の場合はZフラグはクリア（0）されます。

●加算の結果，ビット7からのキャリーが発生した場合は，CYフラグはセット（1），その他の場合はCYフラグはクリア（0）されます。

●加算の結果，ビット3からビット4へのキャリーが発生した場合は，ACフラグはセット（1），その他の場合はACフラグはクリア（0）されます。

【記　　述　　例】

ADD CR10, #56H；CR10レジスタに56Hを加算し，結果をCR10レジスタに格納

ADDC

Add with Carry
キャリーを含むバイト・データの加算

【命　令　形　式】　ADDC dst, src

【オペレーション】　dst, CY ← dst＋src＋CY

【オ ペ ラ ン ド】

ニモニック	オペランド（dst, src）
ADDC	A, #byte
	saddr, #byte
	A, r[注]
	r, A
	A, !addr16
	A, ES:!addr16
	A, saddr
	A,[HL]

ニモニック	オペランド（dst, src）
ADDC	A, ES:[HL]
	A,[HL+byte]
	A, ES:[HL+byte]
	A,[HL+B]
	A, ES:[HL+B]
	A,[HL+C]
	A, ES:[HL+C]

注　r = Aを除く。

【フ　ラ　グ】

Z	AC	CY
×	×	×

【説　　明】

●第1オペランドで指定されるデスティネーション・オペランド（dst）と第2オペランドで指定されるソース・オペランド（src）とCYフラグを加算して，結果をデスティネーション・オペランド（dst）とCYフラグに格納します。

CYフラグは最下位ビットへ加算されます。

この命令は，おもに複数バイトの加算を行うときに使用します。

●加算の結果，dstが0になった場合，Zフラグがセット（1），その他の場合はZフラグはクリア（0）されます。

●加算の結果，ビット7からのキャリーが発生した場合は，CYフラグはセット（1），その他の場合はCYフラグはクリア（0）されます。

●加算の結果，ビット3からビット4へのキャリーが発生した場合は，ACフラグはセット（1），その他の場合はACフラグがクリア（0）されます。

【記　述　例】

ADDC A,[HL+B]；Aレジスタと（HLレジスタ＋（Bレジスタ））番地の内容とCYフラグを加算し，結果をAレジスタに格納

SUB

Subtract
バイト・データの減算

【命　令　形　式】　SUB dst, src

【オペレーション】　dst, CY ← dst－src

【オ　ペ　ラ　ン　ド】

ニモニック	オペランド（dst, src）
SUB	A, #byte
	saddr, #byte
	A, r[注]
	r, A
	A, !addr16
	A, ES:!addr16
	A, saddr
	A,[HL]

ニモニック	オペランド（dst, src）
SUB	A, ES:[HL]
	A,[HL+byte]
	A, ES:[HL+byte]
	A,[HL+B]
	A, ES:[HL+B]
	A,[HL+C]
	A, ES:[HL+C]

注　r = Aを除く。

【フ　　ラ　　グ】

Z	AC	CY
×	×	×

【説　　明】

●第1オペランドで指定されるデスティネーション・オペランド（dst）から第2オペランドで指定されるソース・オペランド（src）を減算し，結果をデスティネーション・オペランド（dst）とCYフラグに格納します。

ソース・オペランド（src）とデスティネーション・オペランド（dst）を同一のものとすることにより，デスティネーション・オペランドの0クリアが可能です。

●減算の結果，dstが0なら，Zフラグはセット（1），その他の場合はZフラグはクリア（0）されます。

●減算の結果，ビット7でボローが発生した場合，CYフラグはセット（1），その他の場合はクリア（0）されます。

●減算の結果，ビット4からビット3へのボローが発生した場合，ACフラグはセット（1），その他の場合はクリア（0）されます。

【記　　述　　例】

SUB D, A；DレジスタからAレジスタを減算し，結果をDレジスタに格納

SUBC

Subtract with Carry
キャリーを含むバイト・データの減算

【命　令　形　式】　SUBC dst, src

【オペレーション】　dst, CY ← dstーsrcーCY

【オ ペ ラ ン ド】

ニモニック	オペランド（dst, src）
SUBC	A, #byte
	saddr, #byte
	A, r注
	r, A
	A, !addr16
	A, ES:!addr16
	A, saddr
	A,[HL]

ニモニック	オペランド（dst, src）
SUBC	A, ES:[HL]
	A,[HL+byte]
	A, ES:[HL+byte]
	A,[HL+B]
	A, ES:[HL+B]
	A,[HL+C]
	A, ES:[HL+C]

注　r = Aを除く。

【フ　ラ　グ】

Z	AC	CY
×	×	×

【説　　明】

●第1オペランドで指定されるデスティネーション・オペランド（dst）から第2オペランドで指定されるソース・オペランド（src）とCYフラグを減算し，結果をデスティネーション・オペランド（dst）に格納します。

CYフラグは最下位ビットから減算します。

この命令は，主として複数バイトの減算を行うときに使用します。

●減算の結果，dstが0ならZフラグはセット（1），その他の場合はZフラグはクリア（0）されます。

●減算の結果，ビットで7でボローが発生した場合，CYフラグはセット（1），その他の場合はクリア（0）されます。

●減算の結果，ビット4からビット3へのボローが発生した場合は，ACフラグはセット（1），その他の場合はクリア（0）されます。

【記　述　例】

SUBC A,[HL]；Aレジスタから（HLレジスタ）番地の内容とCYフラグを減算し，結果をAレジスタに格納

AND

And
バイト・データの論理積

【命　令　形　式】　AND dst, src

【オペレーション】　dst ← dst ∧ src

【オ ペ ラ ン ド】

ニモニック	オペランド（dst, src）
AND	A, #byte
	saddr, #byte
	A, r[注]
	r, A
	A, !addr16
	A, ES:!addr16
	A, saddr
	A,[HL]

ニモニック	オペランド（dst, src）
AND	A, ES:[HL]
	A,[HL+byte]
	A, ES:[HL+byte]
	A,[HL+B]
	A, ES:[HL+B]
	A,[HL+C]
	A, ES:[HL+C]

注　r = Aを除く。

【フ　ラ　グ】

Z	AC	CY
×		

【説　　明】

●第1オペランドで指定されるデスティネーション・オペランド（dst）と第2オペランドで指定されるソース・オペランド（src）のビットごとの論理積をとり，結果をデスティネーション・オペランド（dst）に格納します。

●論理積をとった結果，全ビットが0であればZフラグはセット（1），その他の場合は，Zフラグはクリア（0）されます。

【記　述　例】

AND FFEBAH, #11011100B ; FFEBAHの内容と11011100Bのビットごとの論理積をとり，結果をFFEBAHに格納

OR

Or
バイト・データの論理和

【命　令　形　式】　OR dst, src

【オペレーション】　dst ← dst∨ src

【オ ペ ラ ン ド】

ニモニック	オペランド（dst, src）
OR	A, #byte
	saddr, #byte
	A, r[注]
	r, A
	A, !addr16
	A, ES:!addr16
	A, saddr
	A,[HL]

ニモニック	オペランド（dst, src）
OR	A, ES:[HL]
	A,[HL+byte]
	A, ES:[HL+byte]
	A,[HL+B]
	A, ES:[HL+B]
	A,[HL+C]
	A, ES:[HL+C]

注　r = Aを除く。

【フ　ラ　グ】

Z	AC	CY
×		

【説　　明】

●第1オペランドで指定されるデスティネーション・オペランド（dst）と第2オペランドで指定されるソース・オペランド（src）のビットごとの論理和をとり，結果をデスティネーション・オペランド（dst）に格納します。

●論理和をとった結果，全ビットが0であればZフラグはセット（1），その他の場合はクリア（0）されます。

【記　述　例】

OR A, FFE98H；AレジスタとFFE98Hのビットごとの論理和をとり，結果をAレジスタに格納

XOR

Exclusive Or
バイト・データの排他的論理和

【命　令　形　式】　XOR dst, src

【オペレーション】　dst ← dst ⩛ src

【オ ペ ラ ン ド】

ニモニック	オペランド（dst, src）
XOR	A, #byte
	saddr, #byte
	A, r[注]
	r, A
	A, !addr16
	A, ES:!addr16
	A, saddr
	A,[HL]

ニモニック	オペランド（dst, src）
XOR	A, ES:[HL]
	A,[HL+byte]
	A, ES:[HL+byte]
	A,[HL+B]
	A, ES:[HL+B]
	A,[HL+C]
	A, ES:[HL+C]

注　r = Aを除く。

【フ　ラ　グ】

Z	AC	CY
×		

【説　　明】

●第1オペランドで指定されるデスティネーション・オペランド（dst）と第2オペランドで指定されるソース・オペランド（src）のビットごとの排他的論理和をとり，結果をデスティネーション・オペランド（dst）に格納します。

この命令でソース・オペランド（src）に#0FFHを選択することにより，デスティネーション・オペランド（dst）の全ビットの論理否定がとれます。

●排他的論理和の結果，全ビットが0であればZフラグはセット（1），その他の場合はクリア（0）されます。

【記　述　例】

XOR A, L；AレジスタとLレジスタのビットごとの排他的論理和をとり，結果をAレジスタに格納

CMP

Compare
バイト・データの比較

【命　令　形　式】　CMP dst, src

【オペレーション】　dst－src

【オ ペ ラ ン ド】

ニモニック	オペランド（dst, src）
CMP	A, #byte
	!addr16, #byte
	ES:!addr16, #byte
	saddr, #byte
	A, r[注]
	r, A
	A, !addr16
	A, ES:!addr16
	A, saddr

ニモニック	オペランド（dst, src）
CMP	A,[HL]
	A, ES:[HL]
	A,[HL+byte]
	A, ES:[HL+byte]
	A,[HL+B]
	A, ES:[HL+B]
	A,[HL+C]
	A, ES:[HL+C]

注　r = Aを除く。

【フ　ラ　グ】

Z	AC	CY
×	×	×

【説　　明】

●第1オペランドで指定されるデスティネーション・オペランド（dst）から第2オペランドで指定されるソース・オペランド（src）を減算します。
減算の結果はどこへも格納せずにZ，AC，CYの各フラグだけを変化させます。

●減算の結果，0ならZフラグはセット（1），その他の場合はZフラグはクリア（0）されます。

●減算の結果，ビット7でボローが発生した場合，CYフラグはセット（1），その他の場合はクリア（0）されます。

●減算の結果，ビット4からビット3へのボローが発生した場合，ACフラグはセット（1），その他の場合はクリア（0）されます。

【記　　述　　例】

CMP FFE38H, #38H；FFE38H番地の内容から38Hを減算し，フラグだけを変化（FFE38H番地の内容とイミーディエト・データの比較）

CMP0

Compare 00H
バイト・データの0比較

【命　令　形　式】　CMP0 dst

【オペレーション】　dst－00H

【オ ペ ラ ン ド】

ニモニック	オペランド（dst）
CMP0	A
	X
	B
	C
	!addr16
	ES:!addr16
	saddr

【フ　ラ　グ】

Z	AC	CY
×	0	0

【説　　明】

●第1オペランドで指定されるデスティネーション・オペランド（dst）から00Hを減算します。

●減算結果はどこへも格納せずにZ，AC，CYの各フラグだけ変化させます。

●dstの値が00Hであった場合，Zフラグはセット(1)，その他の場合はZフラグはクリア(0)されます。

●AC，CYフラグは常にクリア(0)されます。

【記　述　例】

CMP0　A ; Aレジスタの内容が0であった場合はZフラグがセット

CMPS

Compare
バイト・データの比較

【命　令　形　式】　CMPS dst, src

【オペレーション】　dst－src

【オ ペ ラ ン ド】

ニモニック	オペランド（dst, src）
CMPS	X, [HL+byte]
	X, ES:[HL+byte]

【フ　ラ　グ】

Z	AC	CY
×	×	×

【説　　明】

●第1オペランドで指定されるデスティネーション・オペランド（[dst]）から，第2オペランドで指定されるソース・オペランド（src）を減算します。
減算の結果はどこへも格納せずZ，AC，CYの各フラグだけを変化させます。

●減算の結果，0ならZフラグはセット（1），その他の場合はZフラグはクリア（0）されます。

●演算の結果が0以外，またはAレジスタの値が0，またはdstの値が0であった場合，CYフラグがセット（1），その他の場合はCYフラグはクリア（0）されます。

●減算の結果，ビット4からビット3へのボローが発生した場合，ACフラグはセット（1），その他の場合はクリア（0）されます。

【記　述　例】

CMPS X, [HL+F0H] ; HL = FD12Hの場合，

Xの値とFFE02H番地の内容を比較し，同じ値であった場合はZフラグをセット
Xの値とFFE02H番地の内容を比較し，違う値であった場合はCYフラグをセット
Aレジスタの値が0だった場合はCYフラグをセット
Xレジスタの値が0だった場合はCYフラグをセット
ACフラグはCMP命令と同様にビット4からビット3へのボローによってセット

6.4　16ビット演算命令

16ビット演算命令には次の命令があります。

・ADDW
・SUBW
・CMPW

ADDW

Add Word
ワード・データの加算

【命　令　形　式】　ADDW dst, src

【オペレーション】　dst, CY ← dst＋src

【オ ペ ラ ン ド】

ニモニック	オペランド（dst, src）
ADDW	AX, #word
	AX, AX
	AX, BC
	AX, DE
	AX, HL
	AX, !addr16
	AX, ES:!addr16
	AX, saddrp
	AX, [HL+byte]
	AX, ES:[HL+byte]

【フ　ラ　グ】

Z	AC	CY
×	×	×

【説　　明】

●第1オペランドで指定されるデスティネーション・オペランド（dst）と第2オペランドで指定されるソース・オペランド（src）の加算を行い，結果をデスティネーション・オペランド（dst）に格納します。

●加算の結果，dstが0になった場合，Zフラグがセット（1），その他の場合はZフラグはクリア（0）されます。

●加算の結果，ビット15からのキャリーが発生した場合は，CYフラグはセット（1），その他の場合はCYフラグはクリア（0）されます。

●加算の結果，ACフラグは不定となります。

【記　述　例】

ADDW AX, #ABCDH；AXレジスタとABCDHを加算し，結果をAXレジスタに格納

SUBW

Subtract Word
ワード・データの減算

【命　令　形　式】　SUBW dst, src

【オペレーション】　dst, CY ← dstーsrc

【オ ペ ラ ン ド】

ニモニック	オペランド（dst, src）
SUBW	AX, #word
	AX, BC
	AX, DE
	AX, HL
	AX, !addr16
	AX, ES:!addr16
	AX, saddrp
	AX, [HL+byte]
	AX, ES:[HL+byte]

【フ　ラ　グ】

Z	AC	CY
×	×	×

【説　　明】

●第1オペランドで指定されるデスティネーション・オペランド（dst）から第2オペランドで指定されるソース・オペランド（src）を減算し，結果をデスティネーション・オペランド（dst）とCYフラグに格納します。

●減算の結果，dstが0ならZフラグはセット（1），その他の場合はZフラグはクリア（0）されます。

●減算の結果，ビット15でボローが発生した場合，CYフラグはセット（1），その他の場合はクリア（0）されます。

●減算の結果，ACフラグは不定となります。

【記　述　例】

SUBW AX, #ABCDH；AXレジスタの内容からABCDHを減算し，結果をAXレジスタに格納

CMPW

Compare Word
ワード・データの比較

【命　令　形　式】　CMPW dst, src

【オペレーション】　dst－src

【オ ペ ラ ン ド】

ニモニック	オペランド（dst, src）
CMPW	AX, #word
	AX, BC
	AX, DE
	AX, HL
	AX, !addr16
	AX, ES:!addr16
	AX, saddrp
	AX, [HL+byte]
	AX, ES:[HL+byte]

【フ　ラ　グ】

Z	AC	CY
×	×	×

【説　　明】

●第1オペランドで指定されるデスティネーション・オペランド（dst）から第2オペランドで指定されるソース・オペランド（src）を減算します。
減算の結果はどこへも格納せずにZ，AC，CYの各フラグだけを変化させます。

●減算の結果，0ならZフラグはセット（1），その他の場合はZフラグはクリア（0）されます。

●減算の結果，ビット15でボローが発生した場合，CYフラグはセット（1），その他の場合はクリア（0）されます。

●減算の結果，ACフラグは不定となります。

【記　述　例】

CMPW AX, #ABCDH ; AXレジスタからABCDHを減算し，フラグだけを変化（AXレジスタとイミーディエト・データとの比較）

6.5　乗除積和算命令

乗除積和算命令には次の命令があります。

・MULU
・MULHU
・MULH
・DIVHU
・DIVWU
・MACHU
・MACH

注意　以下の乗除積和算命令は拡張命令です。RL78-S3コアのみあります。

・MULHU（符号なし16ビット乗算）
・MULH（符号付き16ビット乗算）
・DIVHU（符号なし16ビット除算）
・DIVWU（符号なし32ビット除算）
・MACHU（符号なし積和算（16ビット×16ビット）＋32ビット）
・MACH（符号付き積和算（16ビット×16ビット）＋32ビット）

MULU

Multiply Unsigned
データの符号なし乗算

【命　令　形　式】　MULU src

【オペレーション】　AX ← A×src

【オ ペ ラ ン ド】

ニモニック	オペランド（src）
MULU	X

【フ　ラ　グ】

Z	AC	CY

【説　　明】

●Aレジスタの内容とソース・オペランド（src）のデータを符号なしのデータとして乗算し，結果をAXレジスタに格納します。

【記　述　例】

MULU X；Aレジスタの内容とXレジスタの内容を乗算し，結果をAXレジスタに格納

MULHU

Multiply Unsigned
データの符号なし乗算

【命　令　形　式】　MULHU

【オペレーション】　BCAX ← AX × BC

【オ ペ ラ ン ド】

ニモニック	オペランド（src）
MULHU	

【フ　ラ　グ】

Z	AC	CY

【説　　明】

●AXレジスタの内容とBCレジスタの内容を符号なしのデータとして乗算し，結果の上位16ビットをBCレジスタ，結果の下位16ビットをAXレジスタに格納します。

【記　述　例】

```
MOVW AX, #0C000H
MOVW BC, #1000H
MULHU
MOVW !addr16, AX
MOVW AX, BC
MOVW !addr16, AX    ；C000Hと1000Hを乗算し，結果のC000000Hを!addr16で示すメモリに格納する。
```

MULH

Multiply Signed
データの符号付き乗算

【命　令　形　式】　MULH

【オペレーション】　BCAX ← AX × BC

【オ ペ ラ ン ド】

ニモニック	オペランド（src）
MULH	

【フ　ラ　グ】

Z	AC	CY

【説　　明】

●AXレジスタの内容とBCレジスタの内容を符号付きのデータとして乗算し，結果の上位16ビットをBCレジスタ，結果の下位16ビットをAXレジスタに格納します。

【記　述　例】

```
MOVW AX, #0C000H
MOVW BC, #1000H
MULH
MOVW !addr16, AX
MOVW AX, BC
MOVW !addr16, AX　；C000Hと1000Hを乗算し，結果のFC000000Hを!addr16で示すメモリに格納する。
```

DIVHU

16-bit Unsigned Division
データの符号なし除算

【命　令　形　式】　DIVHU

【オペレーション】　AX（商），DE（余り）　← AX　÷　DE

【オ　ペ　ラ　ン　ド】

ニモニック	オペランド（src）
DIVHU	

【フ　　ラ　　グ】

Z	AC	CY

【説　　明】

●AXレジスタの内容をDEレジスタの内容で除算し，商をAXレジスタに，余りをDEレジスタに格納します。
除算はAXレジスタおよびDEレジスタの内容を符号なしのデータとして行います。
ただしDEレジスタの内容が0のときは，DEレジスタにはAXレジスタの内容が格納され，
AXレジスタ　= 0FFFFHとなります。

【記　　述　　例】

```
MOVW AX, #8081H
MOVW DE, #0002H
DIVHU
MOVW !addr16, AX
MOVW AX, DE
MOVW !addr16, AX    ; 8081Hを0002Hで除算し，AXレジスタの商（4040H）とDEレジスタの余り（0001H）
                      を!addr16で示すメモリに格納する。
```

注意　割り込み処理中にDIVHU，DIVWU命令を実行する場合，割り込み禁止状態(DI)で実行してください。
ただし，RAM領域での命令実行を除き，アセンブリ言語ソースにてDIVHU，DIVWU命令の直後にNOP命令を追加した場合は，割り込み許可状態でもDIVHU，DIVWU命令を実行することができます。下記のコンパイラはビルド時にDIVHU，DIVWU命令が出力される場合，その直後に自動でNOP命令が挿入されます。

- CA78K0R (ルネサス エレクトロニクス社 コンパイラ製品)V1.71以降のC言語ソースおよびアセンブリ言語ソース
- EWRL78 (IAR社 コンパイラ製品) Service pack 1.40.6以降のC言語ソース
- GNURL78 (KPIT社 コンパイラ)のC言語ソース

DIVWU

32-bit Unsigned Division
データの符号なし除算

【命　令　形　式】　DIVWU

【オペレーション】　BCAX（商），HLDE（余り）　←　BCAX　÷　HLDE

【オ　ペ　ラ　ン　ド】

ニモニック	オペランド（src）
DIVWU	

【フ　　ラ　　グ】

Z	AC	CY

【説　　明】

●BCAXレジスタの内容をHLDEレジスタの内容で除算し，商をBCAXレジスタに，余りをHLDEレジスタに格納します。

除算はBCAXレジスタおよびHLDEレジスタの内容を符号なしのデータとして行います。

ただしHLDEレジスタの内容が0のときは，HLDEレジスタにはBCAXレジスタの内容が格納され，

BCAXレジスタ　= 0FFFFFFFFHとなります。

【記　　述　　例】

```
MOVW AX, #8081H
MOVW BC, #8080H
MOVW DE, #0002H
MOVW HL, #0000H
DIVWU
MOVW !addr16, AX
MOVW AX, BC
MOVW !addr16, AX
MOVW AX, DE
MOVW !addr16, AX
MOVW AX, HL
MOVW !addr16, AX    ; 80808081Hを00000002Hで除算し，
                      BCAXレジスタの商（40404040H）とHLDEレジスタの余り（00000001H）を
                      !addr16で示すメモリに格納する。
```

(注意は次ページにあります)

注意　割り込み処理中にDIVHU，DIVWU命令を実行する場合，割り込み禁止状態(DI)で実行してください。
ただし，RAM領域での命令実行を除き，アセンブリ言語ソースにてDIVHU，DIVWU命令の直後にNOP命令を追加した場合は，割り込み許可状態でもDIVHU，DIVWU命令を実行することができます。下記のコンパイラはビルド時にDIVHU，DIVWU命令が出力される場合，その直後に自動でNOP命令が挿入されます。

- CA78K0R (ルネサスエレクトロニクス社 コンパイラ製品)V1.71以降のC言語ソースおよびアセンブリ言語ソース
- EWRL78 (IAR社 コンパイラ製品) Service pack 1.40.6以降のC言語ソース
- GNURL78 (KPIT社 コンパイラ)のC言語ソース

MACHU

Multiply and Accumulation Unsigned
データの符号なし積和演算

【命　令　形　式】　MACHU

【オペレーション】　MACR ← MACR ＋ AX × BC

【オ ペ ラ ン ド】

ニモニック	オペランド（src）
MACHU	

【フ　ラ　グ】

Z	AC	CY
	×	×

【説　　明】

●AXレジスタの内容とBCレジスタの内容を符号なしのデータとして乗算した結果をMACRレジスタと加算を行い，MACRレジスタに格納します。

●加算の結果，オーバフローが発生した場合はCYフラグがセット（1）され，その他の場合はCYフラグがクリア（0）されます。

●ACフラグは0となります。

●MACRレジスタは，積和演算前に初期値を設定してください。また，MACRレジスタは固定であるため，積和演算結果を複数必要とする場合はMACRレジスタを退避して使用してください。

【記　述　例】

```
MOVW AX, #00000H
MOVW !0FFF2H, AX
MOVW !0FFF0H, AX
MOVW AX, #0C000H
MOVW BC, #01000H
MACHU
MOVW AX, !0FFF2H
MOVW !addr16, AX
MOVW AX, !0FFF0H
MOVW !addr16, AX    ; AXレジスタの内容とBCレジスタの内容を乗算し，MACRレジスタの内容と加算した
                      結果をMACRレジスタに格納する。
```

MACH

Multiply and Accumulation Signed
データの符号付き積和演算

【命　令　形　式】　MACH

【オペレーション】　MACR ← MACR ＋ AX × BC

【オ ペ ラ ン ド】

ニモニック	オペランド（src）
MACH	

【フ　ラ　グ】

Z	AC	CY
	×	×

【説　　明】

●AXレジスタの内容とBCレジスタの内容を符号付きのデータとして乗算した結果をMACRレジスタと加算を行い，MACRレジスタに格納します。

●加算の結果，オーバフローが発生した場合はCYフラグがセット（1）され，その他の場合はCYフラグがクリア（0）されます。オーバフローとは，正の累計値に正の積を加算した結果が7FFFFFFFHを超えた場合と，負の累計値に負の積を加算した結果が80000000Hを超えた場合となります。

●演算の結果，MACRレジスタの値が正の場合はACフラグがクリア（0），負の場合はACフラグがセット（1）されます。

●MACRレジスタは，積和演算前に初期値を設定してください。また，MACRレジスタは固定であるため，積和演算結果を複数必要とする場合はMACRレジスタを退避して使用してください。

【記　　述　　例】

```
MOVW AX, #00000H
MOVW !0FFF0H, AX
MOVW AX, #08000H
MOVW !0FFF2H, AX
MOVW AX, #00001H
MOVW !0FFF0H, AX
MOVW AX, #07FFFH
MOVW BC, #0FFFFH
MACH
MOVW AX, !0FFF2H
MOVW !addr16, AX
MOVW AX, !0FFF0H
MOVW !addr16, AX    ; AXレジスタの内容とBCレジスタの内容を乗算し，MACRレジスタの内容と加算した
                      結果をMACRレジスタに格納する。
```

6.6　増減命令

増減命令には次の命令があります。

・INC
・DEC
・INCW
・DECW

INC

Increment
バイト・データのインクリメント

【命　令　形　式】　INC dst

【オペレーション】　dst ← dst＋1

【オ　ペ　ラ　ン　ド】

ニモニック	オペランド（dst）
INC	r
	!addr16
	ES:!addr16
	saddr
	[HL+byte]
	ES:[HL+byte]

【フ　　ラ　　グ】

Z	AC	CY
×	×	

【説　　明】

●デスティネーション・オペランド（dst）の内容を1だけインクリメントします。

●インクリメントした結果が0になればZフラグはセット（1），その他の場合はクリア（0）されます。

●インクリメントした結果，ビット3からビット4へのキャリーがあれば，ACフラグはセット（1），その他の場合はクリア（0）されます。

●繰り返し処理のカウンタやインデクスト・アドレッシングのオフセット・レジスタのインクリメントに使用することが多いため，CYフラグの内容は変化させません（複数バイトの演算時に，CYフラグの内容を保持させるため）。

【記　　述　　例】

INC B；Bレジスタをインクリメント

DEC

Decrement
バイト・データのデクリメント

【命　令　形　式】　DEC dst

【オペレーション】　dst ← dst－1

【オ ペ ラ ン ド】

ニモニック	オペランド（dst）
DEC	r
	!addr16
	ES:!addr16
	saddr
	[HL+byte]
	ES:[HL+byte]

【フ　ラ　グ】

Z	AC	CY
×	×	

【説　　明】

●デスティネーション・オペランド（dst）の内容を1だけデクリメントします。

●デクリメントした結果が0であれば，Zフラグはセット（1），その他の場合はクリア（0）されます。

●デクリメントした結果がビット4からビット3へのキャリーがあれば，ACフラグはセット（1），その他の場合はクリア（0）されます。

●繰り返し処理のカウンタに使用することが多いため，CYフラグの内容は変化させません（複数バイトの演算時にCYフラグを保持させるため）。

●dstがBレジスタ，Cレジスタ，またはsaddrの場合でAC，CYの各フラグを変化させたくない場合，DBNZ命令を使用できます。

【記　述　例】

DEC FFE92H；FFE92H番地の内容をデクリメント

INCW

Increment Word
ワード・データのインクリメント

【命　令　形　式】　INCW dst

【オペレーション】　dst ← dst＋1

【オ ペ ラ ン ド】

ニモニック	オペランド（dst）
INCW	rp
	!addr16
	ES:!addr16
	saddrp
	[HL+byte]
	ES:[HL+byte]

【フ　ラ　グ】

Z	AC	CY

【説　　明】

●デスティネーション・オペランド（dst）の内容を1だけインクリメントします。

●レジスタを使用するアドレッシングで，使用するレジスタ（ポインタ）のインクリメントに使用することが多いため，Z，AC，CYの各フラグを変化させません。

【記　述　例】

INCW HL；HLレジスタをインクリメント

DECW

Decrement Word
ワード・データのデクリメント

【命　令　形　式】　DECW dst

【オペレーション】　dst ← dst－1

【オ ペ ラ ン ド】

ニモニック	オペランド（dst）
DECW	rp
	!addr16
	ES:!addr16
	saddrp
	[HL+byte]
	ES:[HL+byte]

【フ　ラ　グ】

Z	AC	CY

【説　　明】

●デスティネーション・オペランド（dst）の内容を1だけデクリメントします。

●レジスタを使用するアドレッシングで，使用するレジスタ（ポインタ）のデクリメントに使用することが多いため，Z，AC，CYの各フラグを変化させません。

【記　述　例】

DECW DE；DEレジスタをデクリメント

6.7　シフト命令

シフト命令には次の命令があります。

・SHR
・SHRW
・SHL
・SHLW
・SAR
・SARW

SHR

Shift Right
右方向の論理シフト

【命　令　形　式】　SHR　dst , cnt

【オペレーション】　(CY ← dst_0, dst_{m-1} ← dst_m, dst_7 ← 0)×cnt

【オ ペ ラ ン ド】

ニモニック	オペランド（dst, cnt）
SHR	A, cnt

【フ　ラ　グ】

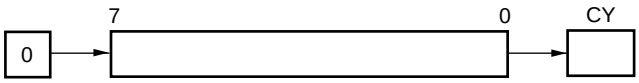
Z	AC	CY
		×

【説　　明】

●第1オペランドで指定されるデスティネーション・オペランド（dst）をcnt回右にシフトします。

●上位ビットには0が埋め込まれ，CYへは最後にビット0からシフトした値が入ります。

●cntは1-7を指定できます。



【記　述　例】

SHR　A, 3 ; Aレジスタの内容がF5Hであった場合の結果は，A = 1EH，CY = 1

A = 1111_0101B　CY = 0

A = 0111_1010B　CY = 1　1回

A = 0011_1101B　CY = 0　2回

A = 0001_1110B　CY = 1　3回

SHRW

Shift Right Word
右方向の論理シフト

【命　令　形　式】　SHRW　dst , cnt

【オペレーション】　(CY ← dst_0, dst_{m-1} ← dst_m, dst_{15} ← 0) × cnt

【オ ペ ラ ン ド】

ニモニック	オペランド（dst, cnt）
SHRW	AX, cnt

【フ　ラ　グ】

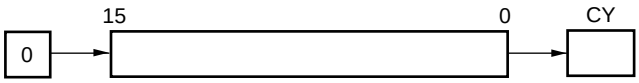
Z	AC	CY
		×

【説　　明】

●第1オペランドで指定されるデスティネーション・オペランド（dst）をcnt回右にシフトします。

●上位ビットには0が埋め込まれ，CYへは最後にビット0からシフトした値が入ります。

●cntは1-15を指定できます。



【記　述　例】

SHRW　AX , 3 ; AXレジスタの内容がAAF5Hであった場合の結果は，AX = 155EH，CY = 1

AX = 1010_1010_1111_0101B　CY = 0

AX = 0101_0101_0111_1010B　CY = 1　1回

AX = 0010_1010_1011_1101B　CY = 0　2回

AX = 0001_0101_0101_1110B　CY = 1　3回

SHL

Shift Left
左方向の論理シフト

【命　令　形　式】　SHL　dst , cnt

【オペレーション】　(CY ← dst_7, dst_m ← dst_{m-1}, dst_0 ← 0)×cnt

【オ ペ ラ ン ド】

ニモニック	オペランド（dst, cnt）
SHL	A, cnt
	B, cnt
	C, cnt

【フ　ラ　グ】

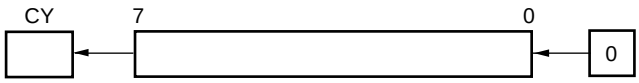
Z	AC	CY
		×

【説　　明】

●第1オペランドで指定されるデスティネーション・オペランド（dst）をcnt回左にシフトします。

●下位ビットには0が埋め込まれ，CYへは最後にビット7からシフトした値が入ります。

●cntは1-7を指定できます。



【記　述　例】

SHL　A , 3 ; Aレジスタの内容が5DHであった場合の結果は，A = E8H，CY = 0

CY = 0　A = 0101_1101B

CY = 0　A = 1011_1010B　1回

CY = 1　A = 0111_0100B　2回

CY = 0　A = 1110_1000B　3回

SHLW

Shift Left Word
左方向の論理シフト

【命　令　形　式】　SHLW　dst , cnt

【オペレーション】　(CY ← dst_{15}, dst_m ← dst_{m-1}, dst_0 ← 0) × cnt

【オ ペ ラ ン ド】

ニモニック	オペランド（dst, cnt）
SHLW	AX, cnt
	BC, cnt

【フ　ラ　グ】

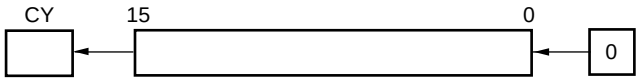
Z	AC	CY
		×

【説　　明】

●第1オペランドで指定されるデスティネーション・オペランド（dst）をcnt回左にシフトします。

●下位ビットには0が埋め込まれ，CYへは最後にビット15からシフトした値が入ります。

●cntは1-15を指定できます。



【記　述　例】

SHLW　BC , 3 ; BCレジスタの内容がC35DHであった場合の結果は，BC = 1AE8H，CY = 0

CY = 0　BC = 1100_0011_0101_1101B

CY = 1　BC = 1000_0110_1011_1010B　1回

CY = 1　BC = 0000_1101_0111_0100B　2回

CY = 0　BC = 0001_1010_1110_1000B　3回

SAR

Shift Arithmetic Right
右方向の算術シフト

【命　令　形　式】　SAR　dst , cnt

【オペレーション】　(CY ← dst_0, dst_{m-1} ← dst_m, dst_7 ← dst_7) × cnt

【オ ペ ラ ン ド】

ニモニック	オペランド（dst, cnt）
SAR	A, cnt

【フ　　ラ　　グ】

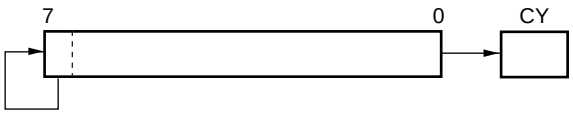
Z	AC	CY
		×

【説　　明】

●第1オペランドで指定されるデスティネーション・オペランド（dst）をcnt回右にシフトします。

●上位ビットには同じ値を保持し，CYへは最後にビット0からシフトした値が入ります。

●cntは1-7を指定できます。



【記　　述　　例】

SAR　A , 4 ; Aレジスタの内容が8CHであった場合の結果は，A = F8H，CY = 1

A = 1000_1100B　CY = 0
A = 1100_0110B　CY = 0　1回
A = 1110_0011B　CY = 0　2回
A = 1111_0001B　CY = 1　3回
A = 1111_1000B　CY = 1　4回

SARW

Shift Arithmetic Right Word
右方向の算術シフト

【命　令　形　式】　SARW　dst , cnt

【オペレーション】　(CY ← dst_0, dst_{m-1} ← dst_m, dst_{15} ← dst_{15}) × cnt

【オ ペ ラ ン ド】

ニモニック	オペランド（dst, cnt）
SARW	AX, cnt

【フ　ラ　グ】

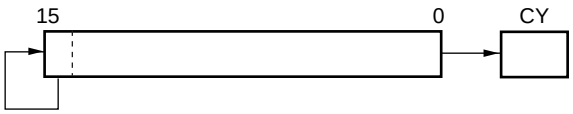
Z	AC	CY
		×

【説　　明】

●第1オペランドで指定されるデスティネーション・オペランド（dst）をcnt回右にシフトします。

●上位ビットには同じ値を保持し，CYへは最後にビット0からシフトした値が入ります。

●cntは1-15を指定する事ができます。



【記　述　例】

SAR　AX , 4 ; AXレジスタの内容がA28CHであった場合の結果は，AX = FA28H，CY = 1

AX = 1010_0010_1000_1100B　CY = 0

AX = 1101_0001_0100_0110B　CY = 0　1回

AX = 1110_1000_1010_0011B　CY = 0　2回

AX = 1111_0100_0101_0001B　CY = 1　3回

AX = 1111_1010_0010_1000B　CY = 1　4回

6.8　ローテート命令

ローテート命令には次の命令があります。

・ROR
・ROL
・RORC
・ROLC
・ROLWC

ROR

Rotate Right
バイト・データの右方向のローテート

【命　令　形　式】　ROR dst, cnt

【オペレーション】　(CY, dst_7 ← dst_0, dst_{m-1} ← dst_m) × 1回

【オ ペ ラ ン ド】

ニモニック	オペランド（dst, cnt）
ROR	A, 1

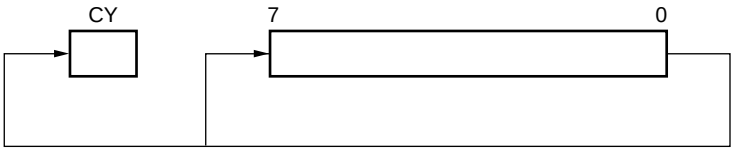
【フ　ラ　グ】

Z	AC	CY
		×

【説　　明】

●第1オペランドで指定されるデスティネーション・オペランド（dst）の内容を，1回だけ右方向へ回転させます。

●LSB（ビット0）の内容はMSB（ビット7）へ回転されると同時にCYフラグへも転送されます。



【記　述　例】

ROR A, 1；Aレジスタの内容を右へ1ビット回転

ROL

Rotate Left
バイト・データの左方向のローテート

【命　令　形　式】　ROL dst, cnt

【オペレーション】　(CY, dst_0 ← dst_7, dst_{m+1} ← dst_m) × 1回

【オ ペ ラ ン ド】

ニモニック	オペランド（dst, cnt）
ROL	A, 1

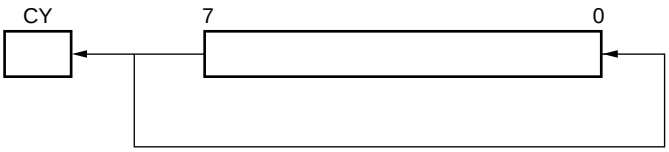
【フ　ラ　グ】

Z	AC	CY
		×

【説　　明】

●第1オペランドで指定されるデスティネーション・オペランド（dst）の内容を1回だけ左方向へ回転させます。

●MSB（ビット7）の内容は，LSB（ビット0）へ回転されると同時にCYフラグへも転送されます。



【記　述　例】

ROL A, 1；Aレジスタの内容を左へ1ビット回転

RORC

Rotate Right with Carry
キャリーを含むバイト・データの右方向のローテート

【命　令　形　式】　RORC dst, cnt

【オペレーション】　(CY ← dst_0, dst_7 ← CY, dst_{m-1} ← dst_m) × 1回

【オ ペ ラ ン ド】

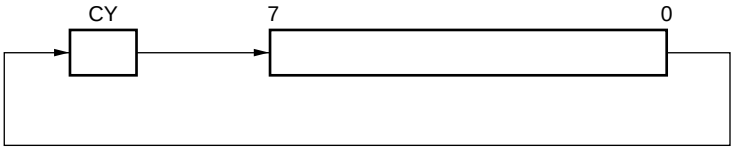
ニモニック	オペランド（dst, cnt）
RORC	A, 1

【フ　ラ　グ】

Z	AC	CY
		×

【説　　明】

●第1オペランドで指定されるデスティネーション・オペランド（dst）の内容を，CYフラグを含め，1回だけ右方向へ回転します。



【記　述　例】

RORC A, 1；Aレジスタの内容を，CYフラグを含めて1ビット右方向へ回転

ROLC

Rotate Left with Carry
キャリーを含むバイト・データの左方向のローテート

【命　令　形　式】　ROLC dst, cnt

【オペレーション】　(CY ← dst_7, dst_0 ← CY, dst_{m+1} ← dst_m) × 1回

【オ ペ ラ ン ド】

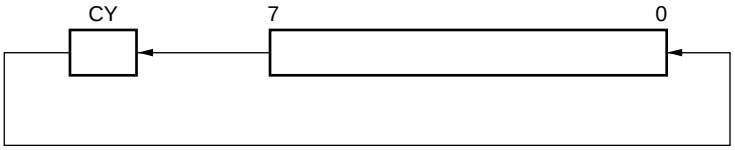
ニモニック	オペランド（dst, cnt）
ROLC	A, 1

【フ　ラ　グ】

Z	AC	CY
		×

【説　　明】

●第1オペランドで指定されるデスティネーション・オペランド（dst）の内容を，CYフラグを含め，1回だけ左方向へ回転させます。



【記　　述　　例】

ROLC A, 1；Aレジスタの内容を，CYフラグを含めて1ビット左へ回転

ROLWC

Rotate Left word with Carry
キャリーを含むワード・データの左方向ローテート

【命　令　形　式】　ROLWC dst, cnt

【オペレーション】　(CY ← dst_{15}, dst_0 ← CY, dst_{m+1} ← dst_m) × 1回

【オ ペ ラ ン ド】

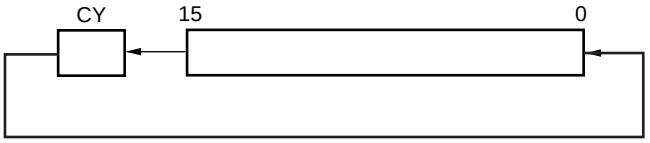
ニモニック	オペランド（dst, cnt）
ROLWC	AX, 1
	BC, 1

【フ　ラ　グ】

Z	AC	CY
		×

【説　　明】

●第1オペランドで指定されるデスティネーション・オペランド（dst）を，CYフラグを含め，1回だけ左方向へ回転させます。



【記　述　例】

ROLWC BC , 1 ; BCレジスタの内容を，CYフラグを含めて1ビット左へ回転

6.9　ビット操作命令

ビット操作命令には次の命令があります。

・MOV1
・AND1
・OR1
・XOR1
・SET1
・CLR1
・NOT1

MOV1

Move Single Bit
1ビット・データの転送

【命　令　形　式】　MOV1 dst, src

【オペレーション】　dst ← src

【オ ペ ラ ン ド】

ニモニック	オペランド（dst, src）
MOV1	CY, A.bit
	A.bit, CY
	CY, PSW.bit
	PSW.bit, CY
	CY, saddr.bit
	saddr.bit, CY

ニモニック	オペランド（dst, src）
MOV1	CY, sfr.bit
	sfr.bit, CY
	CY, [HL].bit
	[HL].bit, CY
	CY, ES:[HL].bit
	ES:[HL].bit, CY

【フ　ラ　グ】

dstがCY

Z	AC	CY
		×

dstがPSW.bit

Z	AC	CY
×	×	

左記以外

Z	AC	CY

【説　　明】

●第1オペランドで指定されたデスティネーション・オペランド（dst）に第2オペランドで指定されたソース・オペランド（src）のビット・データを転送します。

●デスティネーション・オペランド（dst）がCY，またはPSW.bitの場合，該当するフラグのみが変化します。

●MOV1 PSW.bit, CY命令と次に続く命令の間では，すべての割り込みを受け付けません。

【記　述　例】

MOV1 P3.4, CY；CYフラグの内容をポート3のビット4に転送

AND1

And Single Bit
1ビット・データの論理積

【命　令　形　式】　AND1 dst, src

【オペレーション】　dst ← dst∧src

【オ ペ ラ ン ド】

ニモニック	オペランド（dst, src）
AND1	CY, A.bit
	CY, PSW.bit
	CY, saddr.bit
	CY, sfr.bit
	CY, [HL].bit
	CY, ES:[HL].bit

【フ　ラ　グ】

Z	AC	CY
		×

【説　　明】

●第1オペランドで指定されるデスティネーション・オペランド（dst）と第2オペランドで指定されるソース・オペランド（src）のビット・データとの論理積をとり，結果をデスティネーション・オペランド（dst）に格納します。

●CYフラグは，演算結果が格納されます（デスティネーション・オペランド（dst）であるため）。

【記　述　例】

AND1 CY, FFE7FH.3；FFE7FHのビット3とCYフラグの論理積をとり，結果をCYフラグに格納

OR1

Or Single Bit
1ビット・データの論理和

【命　令　形　式】　OR1 dst, src

【オペレーション】　dst ← dst∨src

【オ ペ ラ ン ド】

ニモニック	オペランド（dst, src）
OR1	CY, A.bit
	CY, PSW.bit
	CY, saddr.bit
	CY, sfr.bit
	CY, [HL].bit
	CY, ES:[HL].bit

【フ　ラ　グ】

Z	AC	CY
		×

【説　　明】

●第1オペランドで指定されるデスティネーション・オペランド（dst）と第2オペランドで指定されるソース・オペランド（src）のビット・データとの論理和をとり，結果をデスティネーション・オペランド（dst）に格納します。

●CYフラグは，演算結果が格納されます（デスティネーション・オペランド（dst）であるため）。

【記　述　例】

OR1 CY, P2.5；ポート2のビット5とCYフラグの論理和をとり，結果をCYフラグに格納

XOR1

Exclusive Or Single Bit
1ビット・データの排他的論理和

【命　令　形　式】　XOR1 dst, src

【オペレーション】　dst ← dst ⩛ src

【オ ペ ラ ン ド】

ニモニック	オペランド（dst, src）
XOR1	CY, A.bit
	CY, PSW.bit
	CY, saddr.bit
	CY, sfr.bit
	CY, [HL].bit
	CY, ES:[HL].bit

【フ　ラ　グ】

Z	AC	CY
		×

【説　　明】

●第1オペランドで指定されるデスティネーション・オペランド（dst）と，第2オペランドで指定されるソース・オペランド（src）のビット・データとの排他的論理和をとり，結果をデスティネーション・オペランド（dst）に格納します。

●CYフラグは，演算結果が格納されます（デスティネーション・オペランド（dst）であるため）。

【記　述　例】

XOR1 CY, A.7；Aレジスタのビット7とCYフラグの排他的論理和をとり，結果をCYフラグに格納

SET1

Set Single Bit（Carry Flag）
1ビット・データのセット

【命　令　形　式】　SET1 dst

【オペレーション】　dst ← 1

【オ ペ ラ ン ド】

ニモニック	オペランド（dst）
SET1	A.bit
	PSW.bit
	!addr16.bit
	ES:!addr16.bit
	saddr.bit
	sfr.bit
	[HL].bit
	ES:[HL].bit
	CY

【フ　ラ　グ】

dstがPSW.bit

Z	AC	CY
×	×	×

dstがCY

Z	AC	CY
		1

左記以外

Z	AC	CY

【説　　明】

●デスティネーション・オペランド（dst）をセット（1）します。

●デスティネーション・オペランド（dst）がCY，またはPSW.bitの場合，該当するフラグのみがセット（1）されます。

●SET1 PSW.bit命令と次に続く命令の間では，すべての割り込みを受け付けません。

【記　述　例】

SET1 FFE55H.1；FFE55Hのビット1をセット（1）

CLR1

Clear Single Bit（Carry Flag）
1ビット・データのクリア

【命　令　形　式】　CLR1 dst

【オペレーション】　dst ← 0

【オ ペ ラ ン ド】

ニモニック	オペランド（dst）
CLR1	A.bit
	PSW.bit
	!addr16.bit
	ES:!addr16.bit
	saddr.bit
	sfr.bit
	[HL].bit
	ES:[HL].bit
	CY

【フ　ラ　グ】

dstがPSW.bit

Z	AC	CY
×	×	×

dstがCY

Z	AC	CY
		0

左記以外

Z	AC	CY

【説　　明】

●デスティネーション・オペランド（dst）をクリア（0）します。

●デスティネーション・オペランド（dst）がCY，またはPSW.bitの場合，該当するフラグのみがクリア（0）されます。

●CLR1 PSW.bit命令と次に続く命令の間では，すべての割り込みを受け付けません。

【記　述　例】

CLR1 P3.7；ポート3のビット7をクリア（0）

NOT1

Not Single Bit（Carry Flag）
1ビット・データの論理否定

【命　令　形　式】　NOT1 dst

【オペレーション】　dst ← $\overline{\text{dst}}$

【オ ペ ラ ン ド】

ニモニック	オペランド（dst）
NOT1	CY

【フ　　ラ　　グ】

Z	AC	CY
		×

【説　　明】

●CYフラグを反転します。

【記　　述　　例】

NOT1 CY；CYフラグを反転

6.10　コール・リターン命令

コール・リターン命令には次の命令があります。

・CALL
・CALLT
・BRK
・RET
・RETI
・RETB

CALL

Call
サブルーチン・コール

【命　令　形　式】　CALL　target

【オペレーション】　(SP－2) ← (PC＋n)$_S$
(SP－3) ← (PC＋n)$_H$
(SP－4) ← (PC＋n)$_L$
SP ← SP－4
PC ← target

備考　nは!!addr20のときは4，!addr16/$!addr20のときは3，AX/BC/DE/HLのときは2となります。

【オ　ペ　ラ　ン　ド】

ニモニック	オペランド（target）
CALL	AX
	BC
	DE
	HL
	$!addr20
	!addr16
	!!addr20

【フ　　ラ　　グ】

Z	AC	CY

【説　　明】

●20/16ビットの絶対アドレスまたはレジスタ間接アドレスによるサブルーチン・コールです。

●次の命令の先頭アドレス（PC+n）をスタックに退避し，ターゲット・オペランド（target）で指定されるアドレスに分岐します。

【記　　述　　例】

CALL　!!3E000H : 3E000H番地にサブルーチン・コール

CALLT

Call Table
サブルーチン・コール（コール・テーブル参照）

【命　令　形　式】　CALLT [addr5]

【オペレーション】　(SP－2) ← (PC＋2)S,
(SP－3) ← (PC＋2)H,
(SP－4) ← (PC＋2)L,
PCS ← 0000,
PCH ← (0000, addr5＋1),
PCL ← (0000, addr5)
SP ← SP－4

【オ ペ ラ ン ド】

ニモニック	オペランド（[addr5]）
CALLT	[addr5]

【フ　ラ　グ】

Z	AC	CY

【説　　明】

●コール・テーブル参照のサブルーチン・コールです。

●次の命令の先頭アドレス（PC＋2）をスタックに退避し，コール・テーブル（アドレスの上位4ビットは0000Bに固定で下位16ビットをaddr5で示し，00080H-000BFHの偶数アドレスを指定）のワード・データで示されるアドレスに分岐します。

【記　述　例】

CALLT [80H] ; 00080H, 00081H番地にあるワード・データをアドレスとして，そのアドレスにサブルーチン・コール

【備　　考】

アドレスの指定は偶数アドレスのみです。奇数アドレスは指定できません。

addr5：0080H-00BFHのイミーディエト・データまたはラベル（偶数アドレスのみ）
（ビット15-6は0000000010Bに固定，ビット0は0Bに固定で，ビット5-1の5ビットを可変した0080H-00BFHの16ビット偶数アドレス）

BRK

Break
ソフトウエア・ベクタ割り込み

【命　令　形　式】　BRK

【オペレーション】　(SP－1) ← PSW,
(SP－2) ← (PC＋2)S,
(SP－3) ← (PC＋2)H,
(SP－4) ← (PC＋2)L,
PCS ← 0000,
PCH ← (0007FH),
PCL ← (0007EH),
SP ← SP－4,
IE ← 0

【オ ペ ラ ン ド】

なし

【フ　ラ　グ】

Z	AC	CY

【説　　明】

●ソフトウエア割り込み命令です。

●PSWと次の命令のアドレス（PC＋2）をスタックに退避し，次にIEフラグをクリア（0）して，ベクタ・アドレス（0007EH, 0007FH）のワード・データで指示されるアドレスに分岐します。
IEフラグがクリア（0）されるため，以後のマスカブル・ベクタ割り込みは禁止されます。

●この命令で発生したソフトウエア・ベクタ割り込みからの復帰には，RETB命令を使用します。

RET

Return
サブルーチンからの復帰

【命　令　形　式】　RET

【オペレーション】　PC_L ← (SP),
PC_H ← (SP＋1),
PC_S ← (SP＋2),
SP　← SP＋4

【オ ペ ラ ン ド】

なし

【フ　　ラ　　グ】

Z	AC	CY

【説　　明】

●CALL, CALLT命令でコールされたサブルーチン・コールからのリターン命令です。

●スタックに退避されているワード・データをPCに復帰し，サブルーチンからリターンします。

RETI

Return from Interrupt
ハードウエア・ベクタ割り込みからの復帰

【命　令　形　式】　RETI

【オペレーション】　PC_L ← (SP),
PC_H ← (SP＋1),
PC_S ← (SP＋2),
PSW ← (SP＋3),
SP ← SP＋4,

【オ ペ ラ ン ド】

なし

【フ　ラ　グ】

Z	AC	CY
R	R	R

【説　　明】

●ベクタ割り込みからの復帰命令です。
●スタックに退避されているデータをPCとPSWに復帰し，割り込み処理ルーチンからリターンします。
●BRK命令によるソフトウエア割り込みからの復帰には使用できません。
●この命令と次に実行する命令の間では，すべての割り込みを受け付けません。

【注　　意】

ノンマスカブル割り込みからの復帰には，必ずRETI命令を使用してください。

RETB

Return from Break
ハードウエア・ベクタ割り込みからの復帰

【命　令　形　式】　RETB

【オペレーション】　PC_L ← (SP),
PC_H ← (SP＋1),
PC_S ← (SP＋2),
PSW ← (SP＋3),
SP ← SP＋4

【オ ペ ラ ン ド】

なし

【フ　ラ　グ】

Z	AC	CY
R	R	R

【説　　明】

●BRK命令で発生したソフトウエア割り込みからの復帰命令です。

●スタックに退避されているPCとPSWを復帰し，割り込み処理ルーチンからリターンします。

●この命令と次に実行する命令の間では，すべての割り込みを受け付けません。

6.11　スタック操作命令

スタック操作命令には次の命令があります。

- PUSH
- POP
- MOVW SP, src
- MOVW rp, SP
- ADDW SP, #byte
- SUBW SP, #byte

PUSH

Push
プッシュ

【命　令　形　式】　PUSH src

【オペレーション】

srcがrpの場合	srcがPSWの場合
(SP－1) ← rp_H,	(SP－1) ← PSW
(SP－2) ← rp_L,	(SP－2) ← 00H
SP ← SP－2	SP ← SP－2

【オ ペ ラ ン ド】

ニモニック	オペランド（src）
PUSH	PSW
	rp

【フ　ラ　グ】

Z	AC	CY

【説　　明】

●ソース・オペランド（src）で指定されたレジスタのデータをスタックに退避します。

【記　述　例】

PUSH AX；AXレジスタの内容をスタックに退避

POP

Pop
ポップ

【命　令　形　式】　POP dst

【オペレーション】　dstがrpの場合

rp_L ← (SP),
rp_H ← (SP＋1),
SP ← SP＋2

dstがPSWの場合

PSW ← (SP＋1)
SP ← SP＋2

【オ ペ ラ ン ド】

ニモニック	オペランド（dst）
POP	PSW
	rp

【フ　ラ　グ】

dstがrp

Z	AC	CY

dstがPSW

Z	AC	CY
R	R	R

【説　　明】

●デスティネーション・オペランド（dst）で指定されたレジスタに，データをスタックから復帰します。

●オペランドがPSWの場合，各フラグはスタックのデータで置き換わります。

●POP PSW命令と次に続く命令の間では，すべての割り込みを受け付けません。

【記　述　例】

POP AX；AXレジスタにスタックのデータを復帰

MOVW SP, src
MOVW rp, SP

Move Word
スタック・ポインタとのワード・データの転送

【命　令　形　式】　MOVW dst, src

【オペレーション】　dst ← src

【オ ペ ラ ン ド】

ニモニック	オペランド（dst, src）
MOVW	SP, #word
	SP, AX
	AX, SP
	HL, SP
	BC, SP
	DE, SP

【フ　ラ　グ】

Z	AC	CY

【説　　明】

●スタック・ポインタの内容を操作するための命令です。

●第1オペランドで指定されるデスティネーション・オペランド（dst）に第2オペランドで指定されるソース・オペランド（src）を格納します。

【記　述　例】

MOVW SP, #FE20H ; スタック・ポインタにFE20Hを格納

ADDW SP, #byte

Add stack pointer
スタック・ポインタの加算

【命　令　形　式】　ADDW　SP, src

【オペレーション】　SP ← SP+src

【オ ペ ラ ン ド】

ニモニック	オペランド（src）
ADDW	SP, #byte

【フ　　ラ　　グ】

Z	AC	CY

【説　　明】

●第1オペランドで指定されるスタック・ポインタと第2オペランドで指定されるソース・オペランド（src）の加算を行い，結果をスタック・ポインタに格納します。

【記　　述　　例】

ADDW SP, #12H ; スタック・ポインタと12Hを加算し，結果をスタック・ポインタに格納

SUBW SP, #byte

Sub stack pointer
スタック・ポインタの減算

【命　令　形　式】　SUBW　SP, src

【オペレーション】　SP ← SP－src

【オ ペ ラ ン ド】

ニモニック	オペランド（src）
SUBW	SP, #byte

【フ　　ラ　　グ】

Z	AC	CY

【説　　明】

●第1オペランドで指定されるスタック・ポインタから第2オペランドで指定されるソース・オペランド（src）を減算し，結果をスタック・ポインタに格納します。

【記　　述　　例】

SUBW SP, #12H ; スタック・ポインタから12Hを減算し，結果をスタック・ポインタに格納

6.12　無条件分岐命令

無条件分岐命令には次の命令があります。

・BR

BR

Branch
無条件分岐

【命　令　形　式】　BR　target

【オペレーション】　PC ← target

【オ ペ ラ ン ド】

ニモニック	オペランド（target）
BR	AX
	$addr20
	$!addr20
	!addr16
	!!addr20

【フ　　ラ　　グ】

Z	AC	CY

【説　　明】

●無条件に分岐を行う命令です。

●ターゲット・アドレス・オペランド（target）のワード・データをPCに転送し，分岐します。

【記　　述　　例】

BR !!12345H ; 12345H番地に分岐

6.13　条件付き分岐命令

条件付き分岐命令には次の命令があります。

- BC
- BNC
- BZ
- BNZ
- BH
- BNH
- BT
- BF
- BTCLR

BC

Branch if Carry
キャリー・フラグによる条件分岐（CY = 1）

【命　令　形　式】　BC $addr20

【オペレーション】　PC ← PC＋2＋jdisp8 if CY = 1

【オ ペ ラ ン ド】

ニモニック	オペランド（$addr20）
BC	$addr20

【フ　ラ　グ】

Z	AC	CY

【説　　明】

●CY = 1の場合に，オペランドで指定されたアドレスに分岐します。
　CY = 0の場合，何も処理を行わず，次に続く命令を実行します。

【記　述　例】

BC $00300H; CY = 1なら00300H番地に分岐（ただし，この命令の先頭は0027FH-0037EH番地内にあります）。

BNC

Branch if Not Carry
キャリー・フラグによる条件分岐（CY = 0）

【命　令　形　式】　BNC $addr20

【オペレーション】　PC ← PC＋2＋jdisp8 if CY = 0

【オ ペ ラ ン ド】

ニモニック	オペランド（$addr20）
BNC	$addr20

【フ　ラ　グ】

Z	AC	CY

【説　　明】

●CY = 0の場合に，オペランドで指定されたアドレスに分岐します。
　CY = 1の場合，何も処理を行わず，次に続く命令を実行します。

【記　述　例】

BNC $00300H；CY = 0なら00300H番地に分岐（ただし，この命令の先頭は0027FH-0037EH番地内にあります）。

BZ

Branch if Zero
ゼロ・フラグによる条件分岐（Z = 1）

【命　令　形　式】　BZ $addr20

【オペレーション】　PC ← PC＋2＋jdisp8 if Z = 1

【オ ペ ラ ン ド】

ニモニック	オペランド（$addr20）
BZ	$addr20

【フ　ラ　グ】

Z	AC	CY

【説　　明】

●Z = 1の場合に，オペランドで指定されたアドレスに分岐します。
　Z = 0の場合は，何も処理を行わず，次に続く命令を実行します。

【記　述　例】

DEC B
BZ $003C5H；Bレジスタが0なら003C5H番地に分岐（ただし，この命令の先頭は，00344H-00443H番地内にあります）。

BNZ

Branch if Not Zero
ゼロ・フラグによる条件分岐（Z = 0）

【命　令　形　式】　BNZ $addr20

【オペレーション】　PC ← PC＋2＋jdisp8 if Z = 0

【オ ペ ラ ン ド】

ニモニック	オペランド（$addr20）
BNZ	$addr20

【フ　ラ　グ】

Z	AC	CY

【説　　明】

●Z = 0の場合に，オペランドで指定されたアドレスに分岐します。
　Z = 1の場合は，何も処理を行わず，次に続く命令を実行します。

【記　述　例】

CMP A, #55H
BNZ $00A39H;Aレジスタが55Hでないとき, 00A39H番地に分岐（ただし, この命令の先頭は009B8H-00AB7H
　　番地内にあります）。

BH

Branch if Higher than
数値の大小による条件分岐（(Z∨CY) = 0）

【命　令　形　式】　BH $addr20

【オペレーション】　PC ← PC＋3＋jdisp8 if (Z∨CY) = 0

【オ ペ ラ ン ド】

ニモニック	オペランド（$addr20）
BH	$addr20

【フ　　ラ　　グ】

Z	AC	CY

【説　　明】

●(Z∨CY) = 0の場合に，オペランドで指定されたアドレスに分岐します。
(Z∨CY) = 1の場合は，何も処理を行わず，次に続く命令を実行します。

●この命令は，符号なしの大小を判定するのに使用します。直前のCMP命令の第1オペランドが第2オペランドの値より大きいことを調べます。

【記　　述　　例】

```
CMP A, C
BH $00356H；Aレジスタの内容がCレジスタの内容より大きい場合，00356H番地に分岐（ただし，BH命令
　　の先頭は002D4H-003D3H番地内にあります）。
```

BNH

Branch if Not Higher than
数値の大小による条件分岐（(Z∨CY) = 1）

【命　令　形　式】　BNH $addr20

【オペレーション】　PC ← PC＋3＋jdisp8 if (Z∨CY) = 1

【オ ペ ラ ン ド】

ニモニック	オペランド（$addr20）
BNH	$addr20

【フ　ラ　グ】

Z	AC	CY

【説　　明】

●(Z∨CY) = 1の場合に，オペランドで指定されたアドレスに分岐します。
(Z∨CY) = 0の場合は，何も処理を行わず，次に続く命令を実行します。

●この命令は，符号なしの大小を判定するのに使用します。直前のCMP命令の第1オペランドが第2オペランドの値より大きくない（第1オペランドが第2オペランド以下）ことを調べます。

【記　述　例】

```
CMP A, C
BNH $00356H ; Aレジスタの内容がCレジスタの内容より小さいか等しい場合，00356H番地に分岐（ただし，
              BNH命令の先頭は002D4H-003D3H番地内にあります）。
```

BT

Branch if True

ビット・テストによる条件分岐（バイト・データのビット= 1）

【命　令　形　式】　BT bit, $addr20

【オペレーション】　PC ← PC＋b＋jdisp8 if bit = 1

【オ ペ ラ ン ド】

ニモニック	オペランド（bit, $addr20）	b（バイト数）
BT	saddr.bit, $addr20	4
	sfr.bit, $addr20	4
	A.bit, $addr20	3
	PSW.bit, $addr20	4
	[HL].bit, $addr20	3
	ES:[HL].bit, $addr20	4

【フ　ラ　グ】

Z	AC	CY

【説　　明】

●第1オペランド（bit）の内容がセット（1）されているとき，第2オペランド（$addr20）で指定されるアドレスに分岐します。

第1オペランド（bit）の内容がセット（1）されていないときは，何も処理を行わず，次に続く命令を実行します。

【記　述　例】

BT FFE47H.3, $0055CH；FFE47H番地のビット3が1のとき，0055CH番地に分岐（ただし，この命令の先頭は，004DAH-005D9H番地内にあります）。

BF

Branch if False

ビット・テストによる条件分岐（バイト・データのビット= 0）

【命　令　形　式】　BF bit, $addr20

【オペレーション】　PC ← PC＋b＋jdisp8 if bit = 0

【オ ペ ラ ン ド】

ニモニック	オペランド（bit, $addr20）	b（バイト数）
BF	saddr.bit, $addr20	4
	sfr.bit, $addr20	4
	A.bit, $addr20	3
	PSW.bit, $addr20	4
	[HL].bit, $addr20	3
	ES:[HL].bit, $addr20	4

【フ　ラ　グ】

Z	AC	CY

【説　　明】

●第1オペランド（bit）の内容がクリア（0）されているとき，第2オペランド（$addr20）で指定されるアドレスに分岐します。

第1オペランド（bit）の内容がクリア（0）されていないときは，何も処理を行わず，次に続く命令を実行します。

【記　述　例】

BF P2.2, $01549H；ポート2のビット2が0のとき，01549H番地に分岐（ただし，この命令の先頭は，014C6H-015C5H番地内にあります）。

BTCLR

Branch if True and Clear
ビット・テストによる条件分岐とクリア（バイト・データのビット= 1）

【命　令　形　式】　BTCLR bit, $addr20

【オペレーション】　PC ← PC＋b＋jdisp8 if bit = 1, then bit ← 0

【オ ペ ラ ン ド】

ニモニック	オペランド（bit, $addr20）	b（バイト数）
BTCLR	saddr.bit, $addr20	4
	sfr.bit, $addr20	4
	A.bit, $addr20	3
	PSW.bit, $addr20	4
	[HL].bit, $addr20	3
	ES:[HL].bit, $addr20	4

【フ　ラ　グ】

bitがPSW.bit

Z	AC	CY
×	×	×

左記以外

Z	AC	CY

【説　　明】

●第1オペランド（bit）の内容がセット（1）されているとき，第1オペランド（bit）の内容をクリア（0）し，第2オペランドで指定されたアドレスに分岐します。
第1オペランド（bit）の内容がセット（1）されていないときは，何も処理を行わず，次に続く命令を実行します。

●第1オペランド（bit）がPSW.bitの場合，該当するフラグの内容がクリア（0）されます。

●BTCLR PSW.bit, $addr20命令と次に続く命令の間では，すべての割り込みを受け付けません。

【記　述　例】

BTCLR PSW.0, $00356H；PSWのビット0（CYフラグ）が1の場合，CYフラグをクリアして，00356H番地に分岐（ただし，この命令の先頭は，002D4H-003D3H番地内にあります）。

6.14　条件付きスキップ命令

条件付きスキップ命令には次の命令があります。

- SKC
- SKNC
- SKZ
- SKNZ
- SKH
- SKNH

SKC

Skip if CY
キャリー・フラグによるスキップ（CY = 1）

【命　令　形　式】　SKC

【オペレーション】　Next instruction skip if CY = 1

【オ ペ ラ ン ド】

なし

【フ　　ラ　　グ】

Z	AC	CY

【説　　明】

●CY = 1のときには次の命令をスキップします。次に続く命令はNOPになり，1クロックの実行時間は消費します。ただし，ES:で示されるPREFIX命令が次の命令であった場合は，2クロックの実行時間を消費します。

●CY = 0のときには次の命令を実行します。

●この命令と次に続く命令の間では，すべての割り込みを受け付けません。

【記　　述　　例】

```
MOV A, #55H
SKC
ADD A, #55H ; CY = 0のときはAレジスタ= AAH，CY = 1のときはAレジスタ= 55H
```

SKNC

Skip if not CY
キャリー・フラグによるスキップ（CY = 0）

【命　令　形　式】　SKNC

【オペレーション】　Next instruction skip if CY = 0

【オ ペ ラ ン ド】

なし

【フ　　ラ　　グ】

Z	AC	CY

【説　　明】

●CY = 0のときには次の命令をスキップします。次に続く命令はNOPになり，1クロックの実行時間は消費します。ただし，ES:で示されるPREFIX命令が次の命令であった場合は，2クロックの実行時間を消費します。

●CY = 1のときには次の命令を実行します。

●この命令と次に続く命令の間では，すべての割り込みを受け付けません。

【記　　述　　例】

```
MOV A, #55H
SKNC
ADD A, #55H ; CY = 1のときはAレジスタ= AAH，CY = 0のときはAレジスタ= 55H
```

SKZ

Skip if Z
ゼロ・フラグによるスキップ（Z = 1）

【命　令　形　式】　SKZ

【オペレーション】　Next instruction skip if Z = 1

【オ ペ ラ ン ド】

なし

【フ　　ラ　　グ】

Z	AC	CY

【説　　明】

●Z = 1のときには次の命令をスキップします。次に続く命令はNOPになり，1クロックの実行時間は消費します。ただし, ES:で示されるPREFIX命令が次の命令であった場合は, 2クロックの実行時間を消費します。

●Z = 0のときには次の命令を実行します。

●この命令と次に続く命令の間では，すべての割り込みを受け付けません。

【記　　述　　例】

```
MOV A, #55H
SKZ
ADD A, #55H ; Z = 0のときはAレジスタ= AAH，Z = 1のときはAレジスタ= 55H
```

SKNZ

Skip if not Z
ゼロ・フラグによるスキップ（Z = 0）

【命　令　形　式】　SKNZ

【オペレーション】　Next instruction skip if Z = 0

【オ ペ ラ ン ド】

なし

【フ　　ラ　　グ】

Z	AC	CY

【説　　明】

●Z = 0のときには次の命令をスキップします。次に続く命令はNOPになり，1クロックの実行時間は消費します。ただし, ES:で示されるPREFIX命令が次の命令であった場合は, 2クロックの実行時間を消費します。

●Z = 1のときには次の命令を実行します。

●この命令と次に続く命令の間では，すべての割り込みを受け付けません。

【記　　述　　例】

```
MOV A, #55H
SKNZ
ADD A, #55H ; Z = 1のときはAレジスタ= AAH，Z = 0のときはAレジスタ= 55H
```

SKH

Skip if Higher than
数値の大小によるスキップ（(Z∨CY) = 0）

【命　令　形　式】　SKH

【オペレーション】　Next instruction skip if (Z∨CY) = 0

【オ ペ ラ ン ド】

なし

【フ　ラ　グ】

Z	AC	CY

【説　　明】

●(Z∨CY) = 0のときには次の命令をスキップします。次に続く命令はNOPになり，1クロックの実行時間は消費します。ただし，ES:で示されるPREFIX命令が次の命令であった場合は，2クロックの実行時間を消費します。

●(Z∨CY) = 1のときには次の命令を実行します。

●この命令と次に続く命令の間では，すべての割り込みを受け付けません。

【記　述　例】

```
CMP A, #80H
SKH
CALL !!TARGET ; Aレジスタの内容が80Hより大きい場合はCALL命令をスキップし，その次の命令を実行，
                Aレジスタの内容が80H以下の場合は次のCALL命令を実行し，TARGET番地へ分岐
```

SKNH

Skip if Not Higher than
数値の大小によるスキップ（(Z∨CY) = 1）

【命　令　形　式】　SKNH

【オペレーション】　Next instruction skip if (Z∨CY) = 1

【オ ペ ラ ン ド】

なし

【フ　　ラ　　グ】

Z	AC	CY

【説　　明】

●(Z∨CY) = 1のときには次の命令をスキップします。次に続く命令はNOPになり，1クロックの実行時間は消費します。ただし，ES:で示されるPREFIX命令が次の命令であった場合は，2クロックの実行時間を消費します。

●(Z∨CY) = 0のときには次の命令を実行します。

●この命令と次に続く命令の間では，すべての割り込みを受け付けません。

【記　　述　　例】

```
CMP A, #80H
SKNH
CALL !!TARGET ; Aレジスタの内容が80H以下の場合はCALL命令をスキップし，その次の命令を実行，
                Aレジスタの内容が80Hより大きい場合は次のCALL命令を実行し，TARGET番地へ分岐
```

6.15　CPU制御命令

CPU制御命令には次の命令があります。

・SEL RBn
・NOP
・EI
・DI
・HALT
・STOP

注意　以下のCPU制御命令はRL78-S1コアにはありません。

・SEL RBn（レジスタ・バンクの選択）

SEL RBn

Select Register Bank
レジスタ・バンクの選択

【命　令　形　式】　SEL RBn

【オペレーション】　RBS0, RBS1 ← n；（n = 0-3)

【オ ペ ラ ン ド】

ニモニック	オペランド（RBn）
SEL	RBn

【フ　ラ　グ】

Z	AC	CY

【説　　明】

●オペランド（RBn）で指定されたレジスタ・バンクを次命令以降で使用するレジスタ・バンクとします。

●RBnには，RB0-RB3まであります。

【記　述　例】

SEL RB2；次命令以降で使用するレジスタ・バンクとして，レジスタ・バンク2を選択

NOP

No Operation
ノー・オペレーション

【命　令　形　式】　NOP

【オペレーション】　No Operation

【オ ペ ラ ン ド】

なし

【フ　　ラ　　グ】

Z	AC	CY

【説　　明】

●何も処理をせずに時間だけを消費します。

EI

Enable Interrupt
割り込みの許可

【命　令　形　式】　EI

【オペレーション】　IE ← 1

【オ ペ ラ ン ド】

なし

【フ　　ラ　　グ】

Z	AC	CY

【説　　明】

●マスカブル割り込みの受け付け可能な状態にします（割り込み許可フラグ（IE）をセット（1）します）。

●この命令と次に続く1命令の間では，すべての割り込みを受け付けません。

●この命令を実行しても，他の要因によりベクタ割り込みの受け付けを行わないようにすることができます。
詳細については，各製品のユーザーズ・マニュアルの割り込み機能を参照してください。

DI

Disable Interrupt
割り込みの禁止

【命　令　形　式】　DI

【オペレーション】　IE ← 0

【オ ペ ラ ン ド】

なし

【フ　　ラ　　グ】

Z	AC	CY

【説　　明】

●マスカブル割り込みのベクタ割り込みによる受け付けを禁止します（割り込み許可フラグ（IE）をクリア（0）します）。

●この命令と次に続く1命令の間では，すべての割り込みを受け付けません。

●割り込み処理の詳細については，各製品のユーザーズ・マニュアルの割り込み機能を参照してください。

HALT

Halt
ホルト・モードの設定

【命　令　形　式】　HALT

【オペレーション】　Set HALT Mode

【オ ペ ラ ン ド】

なし

【フ　　ラ　　グ】

Z	AC	CY

【説　　明】

●HALTモードになります。CPUの動作クロックを停止させるモードです。通常動作モードとの組み合わせによる間欠動作により，システムのトータル消費電力を低下させることができます。

STOP

Stop
ストップ・モードの設定

【命　令　形　式】　STOP

【オペレーション】　Set STOP Mode

【オ ペ ラ ン ド】

なし

【フ　　ラ　　グ】

Z	AC	CY

【説　　明】

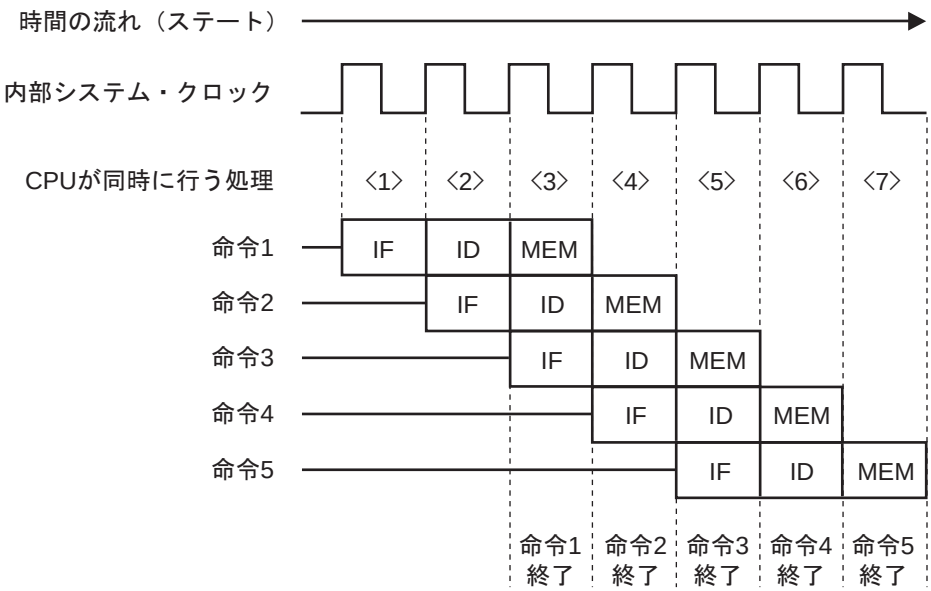
●STOPモードになります。メイン・システム・クロック発振回路を停止させ，システム全体が停止するモードです。リーク電流だけの超低消費電力にすることができます。

第7章　パイプライン

7.1　特徴

RL78マイクロコントローラは，3段パイプラインの制御により，ほとんどの命令を1クロックで実行します。命令実行手順は，インストラクション・フェッチ（IF），インストラクション・デコード（ID），メモリ・アクセス（MEM）の3つのステージで構成されています。

図7－1　標準的な命令を5つ続けて実行する例



・IF（インストラクション・フェッチ）　：命令のフェッチを行い，フェッチ・ポインタをインクリメントします。

・ID（インストラクション・デコード）　：命令をデコードし，アドレス計算を行います。

・MEM（メモリ・アクセス）　：デコードした命令を実行し，対象となるアドレスのメモリにアクセスします。

7.2　動作クロック数

RL78マイクロコントローラでは，他のパイプライン・マイコンに見られるクロック数が数えられないという問題が解決され，常に同じクロック数で動作することにより，安定したプログラムを提供できます。

次の場合を除き，**「5.5　オペレーション一覧」**に記載してある動作クロック数となります。

7.2.1　フラッシュ・メモリの内容をデータ・アクセス

フラッシュ・メモリの内容をデータとしてアクセスした場合は，MEMステージでパイプラインが停止しますので，一覧にある動作クロック数より増加します。詳細は「5.5　オペレーション一覧」を参照してください。

7.2.2　RAMからの命令フェッチ

RAMの内容をフェッチ・データとした場合は，RAMの読み出しが間に合わず，命令キューが空になりますので，命令キューにデータが揃うまでCPUはウエイトします。また，RAMからフェッチ中にRAMへのアクセスが発生した場合もCPUのフェッチ動作はウエイトします。

内部RAM領域からの命令フェッチ時のクロック数は，RL78-S2コアとRL78-S3コアは内部ROM（フラッシュ・メモリ）領域から命令フェッチする場合の最大2倍＋3クロックになります。また，RL78-S1コアは，内部ROM（フラッシュ・メモリ）領域から命令フェッチする場合の最大4倍＋
6クロックになります。

7.2.3　命令の組み合わせによるハザード

間接アクセスに使用するレジスタへの書き込み直後に，そのレジスタの内容のデータを間接アクセスする場合は，RL78-S1コアは1～2クロックのウエイト，RL78-S2コアとRL78-S3コアは1クロックのウエイトが入ります。

レジスタ名	前命令	次の命令のオペランドまたは命令
DE	Dレジスタへのライト命令[注1] Eレジスタへのライト命令[注1] DEレジスタへのライト命令[注1] SEL RBn[注2]	[DE]，[DE+byte]
HL	Hレジスタへのライト命令[注1] Lレジスタへのライト命令[注1] HLレジスタへのライト命令[注1] SEL RBn[注2]	[HL]，[HL+byte]，[HL+B]，[HL+C]，[HL].bit
B	Bレジスタへのライト命令[注1] SEL RBn[注2]	word[B]，[HL+B]
C	Cレジスタへのライト命令[注1] SEL RBn[注2]	word[C]，[HL+C]
BC	Bレジスタへのライト命令[注1] Cレジスタへのライト命令[注1] BCレジスタへのライト命令[注1] SEL RBn[注2]	word[BC]，[HL+B]，[HL+C]

SP	MOVW SP, #word
	MOVW SP, AX
	ADDW SP, #byte
	SUBW SP, #byte
		[SP+byte]
		CALL命令，CALLT命令，BRK命令，SOFT命令，
		RET命令，RETI命令，RETB命令，割り込み，
		PUSH命令，POP命令

CS	MOV CS, #byte MOV CS, A	CALL rp BR AX
AX	Aレジスタへのライト命令[注1] Xレジスタへのライト命令[注1] AXレジスタへのライト命令[注1] SEL RBn[注2]	BR AX
AX BC DE HL	Aレジスタへのライト命令[注1] Xレジスタへのライト命令[注1] Bレジスタへのライト命令[注1] Cレジスタへのライト命令[注1] Dレジスタへのライト命令[注1] Eレジスタへのライト命令[注1] Hレジスタへのライト命令[注1] Lレジスタへのライト命令[注1] AXレジスタへのライト命令[注1] BCレジスタへのライト命令[注1] DEレジスタへのライト命令[注1] HLレジスタへのライト命令[注1] SEL RBn[注2]	CALL rp

注1.　レジスタへのライト命令は，ダイレクト・アドレッシング，ショート・ダイレクト・アドレッシング，レジスタ・インダイレクト・アドレッシング，ベースト・アドレッシング，ベースト・インデクスト・アドレッシングにて，対象となるレジスタの値を書き換えたときにもウエイトが入ります。

2.　RL78-S1コアにはありません。

付録A　命令索引（機能別）

表A－1　機能別の命令早見表（1/3）

機能	ニモニック	内容	機能の掲載ページ	クロック数の掲載ページ		
				RL78-S1	RL78-S2	RL78-S3
8ビット・データ転送命令	MOV	バイト・データの転送	126	37	53	70
	XCH	バイト・データの交換	128	38	55	72
	ONEB	バイト・データの01Hセット	129	39	56	73
	CLRB	バイト・データのクリア	130	39	56	73
	MOVS	バイト・データの転送とPSW変化	131	39	56	73
16ビット・データ転送命令	MOVW	ワード・データの転送	133	39	56	73
	XCHW	ワード・データの交換	135	41	58	75
	ONEW	ワード・データの0001Hセット	136	41	58	75
	CLRW	ワード・データのクリア	137	41	58	75
8ビット演算命令	ADD	バイト・データの加算	139	41	58	75
	ADDC	キャリーを含むバイト・データの加算	140	42	59	76
	SUB	バイト・データの減算	141	42	59	76
	SUBC	キャリーを含むバイト・データの減算	142	43	60	77
	AND	バイト・データの論理積	143	43	60	77
	OR	バイト・データの論理和	144	44	61	78
	XOR	バイト・データの排他的論理和	145	44	61	78
	CMP	バイト・データの比較	146	45	62	79
	CMP0	バイト・データの0比較	147	45	62	79
	CMPS	バイト・データの比較	148	45	62	79
16ビット演算命令	ADDW	ワード・データの加算	150	46	63	80
	SUBW	ワード・データの減算	151	46	63	80
	CMPW	ワード・データの比較	152	46	63	80
乗除積和算命令	MULU	データの符号なし乗算	154	46	63	81
	MULHU	データの符号なし乗算	155	－	－	81
	MULH	データの符号付き乗算	156	－	－	81
	DIVHU	データの符号なし除算	157	－	－	81
	DIVWU	データの符号なし除算	158	－	－	81
	MACHU	データの符号なし積和演算	159	－	－	81
	MACH	データの符号付き積和演算	160	－	－	81
増減命令	INC	バイト・データのインクリメント	162	47	64	82
	DEC	バイト・データのデクリメント	163	47	64	82
	INCW	ワード・データのインクリメント	164	47	64	82
	DECW	ワード・データのデクリメント	165	47	64	82

表A－1　機能別の命令早見表（2/3）

名称	ニモニック	機能	機能の掲載ページ	クロック数の掲載ページ		
				RL78-S1	RL78-S2	RL78-S3
シフト命令	SHR	右方向の論理シフト	167	47	64	82
	SHRW	右方向の論理シフト	168	47	64	82
	SHL	左方向の論理シフト	169	47	64	82
	SHLW	左方向の論理シフト	170	47	64	82
	SAR	右方向の算術シフト	171	47	64	82
	SARW	右方向の算術シフト	172	47	64	82
ローテート命令	ROR	バイト・データの右方向のローテート	174	48	65	83
	ROL	バイト・データの左方向のローテート	175	48	65	83
	RORC	キャリーを含むバイト・データの右方向のローテート	176	48	65	83
	ROLC	キャリーを含むバイト・データの左方向のローテート	177	48	65	83
	ROLWC	キャリーを含むワード・データの左方向ローテート	178	48	65	83
ビット操作命令	MOV1	1ビット・データの転送	180	48	65	83
	AND1	1ビット・データの論理積	181	48	65	83
	OR1	1ビット・データの論理和	182	48	65	83
	XOR1	1ビット・データの排他的論理和	183	49	66	84
	SET1	1ビット・データのセット	184	49	66	84
	CLR1	1ビット・データのクリア	185	49	66	84
	NOT1	1ビット・データの論理否定	186	49	66	84
コール・リターン命令	CALL	サブルーチン・コール	188	50	67	85
	CALLT	サブルーチン・コール（コール・テーブル参照）	189	50	67	85
	BRK	ソフトウエア・ベクタ割り込み	190	50	67	85
	RET	サブルーチンからの復帰	191	50	67	85
	RETI	ハードウエア・ベクタ割り込みからの復帰	192	50	67	85
	RETB	ハードウエア・ベクタ割り込みからの復帰	193	50	67	85
スタック操作命令	PUSH	プッシュ	195	51	68	86
	POP	ポップ	196	51	68	86
	MOVW SP, src	スタック・ポインタとのワード・データの転送	197	51	68	86
	MOVW rp, SP	スタック・ポインタとのワード・データの転送	197	51	68	86
	ADDW SP, #byte	スタック・ポインタの加算	198	51	68	86
	SUBW SP, #byte	スタック・ポインタの減算	199	51	68	86
無条件分岐命令	BR	無条件分岐	201	51	68	86
条件付き分岐命令	BC	キャリー・フラグによる条件分岐（CY = 1）	203	51	68	86
	BNC	キャリー・フラグによる条件分岐（CY = 0）	204	51	68	86
	BZ	ゼロ・フラグによる条件分岐（Z = 1）	205	51	68	86
	BNZ	ゼロ・フラグによる条件分岐（Z = 0）	206	51	68	86
	BH	数値の大小による条件分岐（(Z∨CY) = 0）	207	51	68	86
	BNH	数値の大小による条件分岐（(Z∨CY) = 1）	208	51	68	86
	BT	ビット・テストによる条件分岐 （バイト・データのビット= 1）	209	51	68	86
	BF	ビット・テストによる条件分岐 （バイト・データのビット= 0）	210	52	69	87
	BTCLR	ビット・テストによる条件分岐とクリア （バイト・データのビット= 1）	211	52	69	87

表A－1　機能別の命令早見表（3/3）

名称	ニモニック	機能	機能の掲載ページ	クロック数の掲載ページ		
				RL78-S1	RL78-S2	RL78-S3
条件付きスキップ命令	SKC	キャリー・フラグによるスキップ（CY = 1）	213	52	69	87
	SKNC	キャリー・フラグによるスキップ（CY = 0）	214	52	69	87
	SKZ	ゼロ・フラグによるスキップ（Z = 1）	215	52	69	87
	SKNZ	ゼロ・フラグによるスキップ（Z = 0）	216	52	69	87
	SKH	数値の大小によるスキップ（(Z∨CY) = 0）	217	52	69	87
	SKNH	数値の大小によるスキップ（(Z∨CY) = 1）	218	52	69	87
CPU制御命令	SEL RBn	レジスタ・バンクの選択	220	－	69	87
	NOP	ノー・オペレーション	221	52	69	87
	EI	割り込みの許可	222	52	69	87
	DI	割り込みの禁止	223	52	69	87
	HALT	ホルト・モードの設定	224	52	69	87
	STOP	ストップ・モードの設定	225	52	69	87

付録B　命令索引（アルファベット順）

表B－1　アルファベット順の命令早見表（1/3）

ニモニック	内容	機能の掲載ページ	クロック数の掲載ページ		
			RL78-S1	RL78-S2	RL78-S3
ADD	バイト・データの加算	139	41	58	75
ADDC	キャリーを含むバイト・データの加算	140	42	59	76
ADDW	ワード・データの加算	150	46	63	80
ADDW SP, #byte	スタック・ポインタの加算	198	51	68	86
AND	バイト・データの論理積	143	43	60	77
AND1	1ビット・データの論理積	181	48	65	83
BC	キャリー・フラグによる条件分岐（CY = 1）	203	51	68	86
BF	ビット・テストによる条件分岐（バイト・データのビット= 0）	210	52	69	87
BH	数値の大小による条件分岐（(Z∨CY) = 0）	207	51	68	86
BNC	キャリー・フラグによる条件分岐（CY = 0）	204	51	68	86
BNH	数値の大小による条件分岐（(Z∨CY) = 1）	208	51	68	86
BNZ	ゼロ・フラグによる条件分岐（Z = 0）	206	51	68	86
BR	無条件分岐	201	51	68	86
BRK	ソフトウエア・ベクタ割り込み	190	50	67	85
BT	ビット・テストによる条件分岐（バイト・データのビット= 1）	209	51	68	86
BTCLR	ビット・テストによる条件分岐とクリア（バイト・データのビット= 1）	211	52	69	87
BZ	ゼロ・フラグによる条件分岐（Z = 1）	205	51	68	86
CALL	サブルーチン・コール	188	50	67	85
CALLT	サブルーチン・コール（コール・テーブル参照）	189	50	67	85
CLRB	バイト・データのクリア	130	39	56	73
CLRW	ワード・データのクリア	137	41	58	75
CLR1	1ビット・データのクリア	185	49	66	84
CMP	バイト・データの比較	146	45	62	79
CMPS	バイト・データの比較	148	45	62	79
CMPW	ワード・データの比較	152	46	63	80
CMP0	バイト・データの0比較	147	45	62	79
DEC	バイト・データのデクリメント	163	47	64	82
DECW	ワード・データのデクリメント	165	47	64	82
DI	割り込みの禁止	223	52	69	87
DIVHU	データの符号なし除算	157	－	－	81
DIVWU	データの符号なし除算	158	－	－	81
EI	割り込みの許可	222	52	69	87
HALT	ホルト・モードの設定	224	52	69	87
INC	バイト・データのインクリメント	162	47	64	82
INCW	ワード・データのインクリメント	164	47	64	82
MACH	データの符号付き積和演算	160	－	－	81
MACHU	データの符号なし積和演算	159	－	－	81

表B－1　アルファベット順の命令早見表（2/3）

ニモニック	内容	機能の掲載ページ	クロック数の掲載ページ		
			RL78-S1	RL78-S2	RL78-S3
MOV	バイト・データの転送	126	37	53	70
MOVS	バイト・データの転送とPSW変化	131	39	56	73
MOVW	ワード・データの転送	133	39	56	73
MOVW rp, SP	スタック・ポインタとのワード・データの転送	197	51	68	86
MOVW SP, src	スタック・ポインタとのワード・データの転送	197	51	68	86
MOV1	1ビット・データの転送	180	48	65	83
MULH	データの符号付き乗算	156	－	－	81
MULHU	データの符号なし乗算	155	－	－	81
MULU	データの符号なし乗算	154	46	63	81
NOP	ノー・オペレーション	221	52	69	87
NOT1	1ビット・データの論理否定	186	49	66	84
ONEB	バイト・データの01Hセット	129	39	56	73
ONEW	ワード・データの0001Hセット	136	41	58	75
OR	バイト・データの論理和	144	44	61	78
OR1	1ビット・データの論理和	182	48	65	83
POP	ポップ	196	51	68	86
PUSH	プッシュ	195	51	68	86
RET	サブルーチンからの復帰	191	50	67	85
RETB	ハードウエア・ベクタ割り込みからの復帰	193	50	67	85
RETI	ハードウエア・ベクタ割り込みからの復帰	192	50	67	85
ROL	バイト・データの左方向のローテート	175	48	65	83
ROLC	キャリーを含むバイト・データの左方向のローテート	177	48	65	83
ROLWC	キャリーを含むワード・データの左方向ローテート	178	48	65	83
ROR	バイト・データの右方向のローテート	174	48	65	83
RORC	キャリーを含むバイト・データの右方向のローテート	176	48	65	83
SAR	右方向の算術シフト	171	47	64	82
SARW	右方向の算術シフト	172	47	64	82
SEL RBn	レジスタ・バンクの選択	220	－	69	87
SET1	1ビット・データのセット	184	49	66	84
SHL	左方向の論理シフト	169	47	64	82
SHLW	左方向の論理シフト	170	47	64	82
SHR	右方向の論理シフト	167	47	64	82
SHRW	右方向の論理シフト	168	47	64	82
SKC	キャリー・フラグによるスキップ（CY = 1）	213	52	69	87
SKH	数値の大小によるスキップ（(Z∨CY) = 0）	217	52	69	87
SKNC	キャリー・フラグによるスキップ（CY = 0）	214	52	69	87
SKNH	数値の大小によるスキップ（(Z∨CY) = 1）	218	52	69	87
SKNZ	ゼロ・フラグによるスキップ（Z = 0）	216	52	69	87
SKZ	ゼロ・フラグによるスキップ（Z = 1）	215	52	69	87
STOP	ストップ・モードの設定	225	52	69	87
SUB	バイト・データの減算	141	42	59	76
SUBC	キャリーを含むバイト・データの減算	142	43	60	77

表B－1　アルファベット順の命令早見表（3/3）

ニモニック	内容	機能の掲載ページ	クロック数の掲載ページ		
			RL78-S1	RL78-S2	RL78-S3
SUBW	ワード・データの減算	151	46	63	80
SUBW SP, #byte	スタック・ポインタの減算	199	51	68	86
XCH	バイト・データの交換	128	38	55	72
XCHW	ワード・データの交換	135	41	58	75
XOR	バイト・データの排他的論理和	145	44	61	78
XOR1	1ビット・データの排他的論理和	183	49	66	84

付録C　改版履歴

C.1　本版で改訂された主な箇所

箇所	内容	分類
第1章　概説		
p.2	表1－1 各CPUコアの機能の違いの備考を変更	(b)
第5章　命令セット		
p.99	表5－8 命令フォーマット一覧（12/30）の誤記を訂正	(a)
p.104	表5－8 命令フォーマット一覧（17/30）の誤記を訂正	(a)
第6章　命令の説明		
p.126	6.1 8ビット・データ転送命令 MOVの【オペランド】を追加	(b)

備考　表中の「分類」により、改訂内容を次のように区分しています。

（a）：誤記訂正、（b）：仕様（スペック含む）の追加／変更、（c）：説明、注意事項の追加／変更、

（d）：パッケージ、オーダ名称、管理区分の追加／変更、（e）：関連資料の追加／変更

C.2　前版までの改版履歴

これまでの改版履歴を次に示します。

版数	内容	適用箇所
Rev.2.20	表5－7　RL78-S3コアのオペレーション一覧（12/18）に注意を追加	第5章　命令セット
	6.5　乗除積和算命令に注意を追加	第6章　命令の説明
Rev.2.10	3.1.2（3）レジスタ・バンク選択フラグ（RBS0, RBS1）の誤記を訂正	第3章　レジスタ
	表5－8　命令フォーマット一覧の誤記を訂正	第5章　命令セット
Rev.2.00	説明を変更	第1章　概説
	説明を変更	第2章　メモリ空間
	3.1.1　プログラム・カウンタ（PC）の説明を変更	第3章　レジスタ
	3.1.2　プログラム・ステータス・ワード（PSW）の説明を変更	
	3.1.2（3）レジスタ・バンク選択フラグ（RBS0, RBS1）に注意を追加	
	3.1.3　スタック・ポインタ（SP）の説明を変更	
	3.2　汎用レジスタの説明を変更	
	表3－2　汎用レジスター覧に注を追加	
	3.3　ES、CSレジスタの説明を変更	
	3.4.1　プロセッサ・モード・コントロール・レジスタ（PMC）の説明を変更	
	4.1.1　レラティブ・アドレッシングの注意を削除	第4章　アドレッシング
	4.2.3　ダイレクト・アドレッシングの誤記を訂正	
	4.2.9　スタック・アドレッシングの説明を変更	
	命令セットの説明を変更	第5章　命令セット
	表5－1　オペランドの表現形式と記述方法に注を追加	
	表5－2　オペレーション欄の記号に注を追加	
	5.5.1　RL78-S1コアのオペレーション一覧として説明を追加	
	5.5.2　RL78-S2コアのオペレーション一覧として説明を追加	
	5.5.3　RL78-S3コアのオペレーション一覧として説明を追加	
	表5－8　命令フォーマット一覧（12/30）に注を追加	
	表5－8　命令フォーマット一覧（30/30）に注を追加	
	表5－10　命令マップ（2nd MAP）に注を追加	
	CMP0（バイト・データの0比較）のフラグの誤記を訂正	第6章　命令の説明
	6.5　乗除積和算命令に注意を追加	
	6.15　CPU制御命令に注意を追加	
	7.2　動作クロック数の説明を変更	第7章　パイプライン
	7.2.3　命令の組み合わせによるハザードの説明を変更、注を追加	
	命令索引（機能別）を早見表の形式に変更	付録A　命令索引（機能別）
	命令索引（アルファベット順）を早見表の形式に変更	付録B　命令索引(アルファベット順)

RL78 ファミリ ユーザーズマニュアル
ソフトウェア編

発行年月日　2011 年 1 月 31 日　Rev.1.00
2022 年 4 月 22 日　Rev.2.30

発行　ルネサス エレクトロニクス株式会社
〒135-0061 東京都江東区豊洲 3-2-24（豊洲フォレシア）

RL78 ファミリ

ルネサス エレクトロニクス株式会社

R01US0015JJ0230