お客様各位

カタログ等資料中の旧社名の扱いについて

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

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

2010年4月1日
ルネサスエレクトロニクス株式会社

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

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

ご注意書き

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

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

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

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

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

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

RENESAS

ユーザーズ・マニュアル

保守／廃止

V25™, V35™ ファミリ

16/8, 16 ビット・シングルチップ・マイクロコンピュータ

命令編

対象デバイス

V25

V35

V25 ＋ ™

V35 ＋ ™

資料番号　U12120JJ3V0UM00（第３版）
発行年月　September 1997　N

保守／廃止

〔メ　モ〕

CMOSデバイスの一般的注意事項

①静電気対策（MOS全般）

注意　MOSデバイス取り扱いの際は静電気防止を心がけてください。

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

また，MOSデバイスを実装したボードについても同様の扱いをしてください。

②未使用入力の処理（CMOS特有）

注意　CMOSデバイスの入力レベルは固定してください。

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

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

③初期化以前の状態（MOS全般）

注意　電源投入時，MOSデバイスの初期状態は不定です。

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

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

M7　94.11

保守／廃止

本版で改訂された主な箇所

箇　所	内　容
はじめに	μPD70320(A), μPD70330(A), μPD70325(A), μPD70335(A)を削除

本文欄外の★印は，本版で改訂された主な箇所を示しています。

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

保守／廃止

はじめに

対象者　このマニュアルは，次のV25, V35ファミリの機能を理解し，それを用いたアプリケーション・システムを設計するユーザを対象とします。

★

製品名	別名称
μPD70320	V25
μPD70330	V35
μPD70325	V25＋
μPD70335	V35＋

目　的　このマニュアルは，上記V25, V35ファミリの持つ各種命令機能をユーザに理解していただくことを目的とします。

構　成　上記V25, V35ファミリのユーザーズ・マニュアルは，ハードウエア編と命令編（このマニュアル）の２冊に分かれています。

ハードウエア編	命　令　編
概　説	概　説
端子機能	命令の説明
CPU機能	V20, V30に対する追加命令
内部ブロック機能	命令マップ
バス制御機能	μPD8086, 8088とのニモニック対応表
割り込み機能	プログラムの実行クロック数
スタンバイ機能	86系Cコンパイラ，アセンブラでのプログラム開発
リセット機能	
その他	

読 み 方　このマニュアルの読者には，電気，論理回路，マイクロコンピュータの一般知識を必要とします。特に断りのないかぎり，各製品に共通の説明です。
なお，このマニュアルでは「μPD70…」という製品を，「V…」の名称に統一して説明してあります。

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
→ **第2章　命　　令**を参照してください（ニモニックのアルファベット順に記載してあります）。

一通り命令機能の詳細を理解しようとするとき
→ 目次に従って読んでください。

各製品のハードウエア機能を理解しようとするとき
→ 各製品の**ユーザーズ・マニュアル　ハードウエア編**（別冊）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
アクティブ・ロウの表記	：$\overline{\times\times\times}$（端子，信号名称に上線）
メモリ・マップのアドレス	：上側－上位，下側－下位
アドレスの表記	：次の場合，xがセグメント値，yがオフセット値を表します。 x：yH
注	：本文中につけた注の説明
注　　意	：気をつけて読んでいただきたい内容
備　　考	：本文の補足説明
数の表記	：２進数…××××または××××B 10進数…×××× 16進数…××××H

関連資料

資料名／品名	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
		ハードウエア編	命令編	
V25	U10090J	IEM-967	このマニュアル	IEM-5077 IEA-604 IEA-701
V35	U10170J			IEA-709
V25＋	U12850J	IEU-706		－
V35＋	U12884J			IEA-709

備考　上記関連資料は予告なしに内容を変更することがあります。資料を請求される際には必ず最新の資料かどうかを確認してください。

保守／廃止

目　　次

図　の　目　次

表　の　目　次

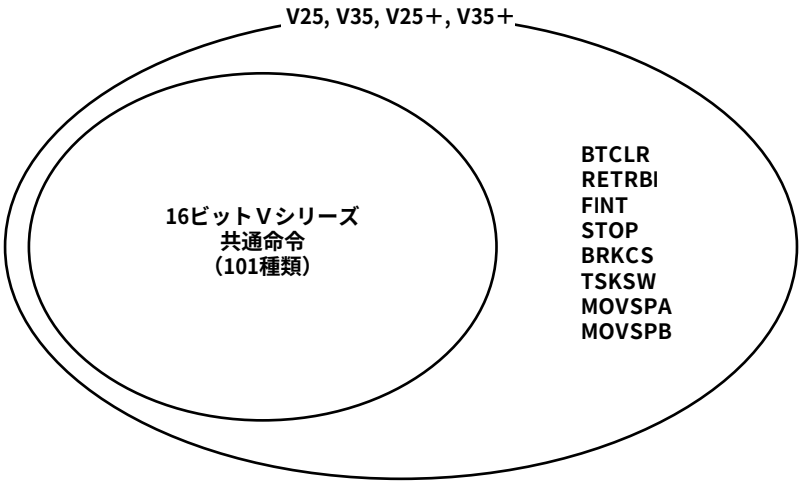
保守／廃止

第１章　概　　説

16ビットVシリーズ™には，完全なソフトウエア互換性を持つ101種類の共通命令があり，ソフトウエア資産の有効活用ができます。

この共通命令に対し，V25, V35, V25＋, V35＋には８つの命令が追加されています。

図１－１　共通命令と各製品の専用命令の関係



保守／廃止

1.1　機能別命令一覧

V25, V35ファミリの各命令を機能別に大別すると次の26種類になります。

表1－1　機能別命令一覧

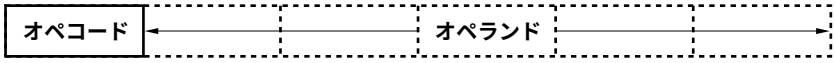
命　令　群	ニモニック（アルファベット順）
データ転送命令	LDEA, MOV, MOVSPA, MOVSPB, TRANS, TRANSB, XCH
リピート・プリフィクス	REP, REPC, REPE, REPNC, REPNE, REPNZ, REPZ
プリミティブ・ブロック転送命令	CMPBK, CMPBKB, CMPBKW, CMPM, CMPMB, CMPMW, LDM, LDMB, LDMW, MOVBK, MOVBKB, MOVBKW, STM, STMB, STMW
ビット・フィールド操作命令	EXT, INS
入出力命令	IN, OUT
プリミティブ入出力命令	INM, OUTM
加減算命令	ADD, ADDC, SUB, SUBC
BCD演算命令	ADD4S, CMP4S, ROL4, ROR4, SUB4S
増減命令	DEC, INC
乗除算命令	DIV, DIVU, MUL, MULU
BCD補正命令	ADJ4A, ADJ4S, ADJBA, ADJBS
データ変換命令	CVTBD, CVTBW, CVTDB, CVTWL
比較命令	CMP
補数演算命令	NEG, NOT
論理演算命令	AND, OR, TEST, XOR
ビット操作命令	CLR1, NOT1, SET1, TEST1
シフト命令	SHL, SHR, SHRA
ローテート命令	ROL, ROLC, ROR, RORC
サブルーチン制御命令	CALL, RET
スタック操作命令	DISPOSE, POP, PREPARE, PUSH
ブランチ命令	BR
条件付きブランチ命令	BC, BCWZ, BE, BGE, BGT, BH, BL, BLE, BLT, BN, BNC, BNE, BNH, BNL, BNV, BNZ, BP, BPE, BPO, BTCLR, BZ, BV, DBNZ, DBNZE, DBNZNE
割り込み命令	BRK, BRKV, CHKIND, FINT, RETI, RETRBI
CPU制御命令	BUSLOCK, DI, EI, FPO1, FPO2, HALT, NOP, POLL, STOP
セグメント・オーバライド・プリフィクス	DS0：, DS1：, PS：, SS：
レジスタ・バンク切り替え命令	BRKCS, TSKSW

保守／廃止

1.2　命令語の形式

命令語（オブジェクト・コード）は基本的に次の形式で表されます。

図１－２　命令語形式



備考　オペコード：命令の種類を表す８ビットのコードです。
オペランド：命令の処理の対象となるレジスタ，メモリ・アドレスを示すフィールドです。
0-5バイトのフィールドで表されます。

1.3　命令の概要

1.3.1　データ転送

データ転送命令により，データの操作なしで，レジスタとレジスタ間またはレジスタとメモリ間のデータ転送ができます。次の５種の命令に分類できます。

汎用データの転送（MOV）	：第２オペランドから第１オペランドへのバイト／ワード転送をします。また，レジスタやメモリに直接数値を転送することもできます。
レジスタ・バンク内データの転送（MOVSPA, MOVSPB）	：レジスタ・バンク切り替え前後，SS, SP間でデータを転送します。
実効アドレスの転送（LDEA）	：第２オペランドのオフセット・アドレス（実効アドレス）を第１オペランドへ転送します。
変換テーブルの転送（TRANS/TRANSB）	：変換テーブルの１バイトを転送します。
汎用データの交換（XCH）	：第１オペランドと第２オペランドの内容を交換します。

1.3.2　ブロック操作

リピート・プリフィクスとプリミティブ・ブロック転送命令によりバイトまたはワードのブロック（連続したデータ）に対する転送や比較ができます。

プリミティブ・ブロック転送命令には，アキュームレータとの間で行うブロック単位の転送に対する命令と同様，転送，ある値との比較，走査に対しての命令があります。また，１バイトのリピート・プリフィクスを前置するとハードウエアによる反復処理が可能で，連続したデータ操作ができます。

1.3.3　ビット・フィールド操作

ビット・フィールド操作命令により連続するメモリをビット・フィールドとみなして，指定した長さのデータを（指定した）ビット・フィールド領域とAWレジスタの間で転送することができます。

この命令は，命令実行終了後にワード・オフセット（IXレジスタまたはIYレジスタ）およびビット・オフセット（８ビット汎用レジスタ）を更新して，連続したビット・フィールド・データを自動的に指定します。コンピュータ・グラフィクスや高級言語に有効で，たとえば，Pascalのパクト・アレイやレコード・タイプのデータ構造に対応できます。

保守／廃止

1.3.4 入 出 力

入出力命令，プリミティブ入出力命令によりI/Oデバイスに対してリード／ライトができます。

I/Oデバイスは，この命令によりデータ・バスを通じて，CPUとのデータ転送をします。

1.3.5 演　　算

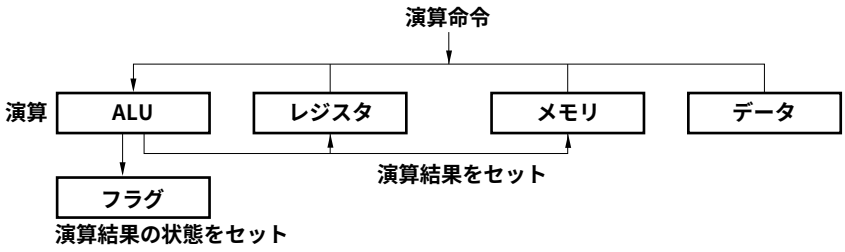
次の命令により8/16ビット・データの演算ができます。

加減算命令，増減命令，乗算命令，除算命令，比較命令，補数演算命令，論理演算命令

また，増減命令により汎用レジスタまたはメモリの8/16ビット・データをインクリメント（＋１）／デクリメント（－１）することもできます。

各演算命令は対象となるレジスタやメモリの中で実行されるのではなく，実際は内部のALUで実行され，プログラム・ステータス・ワード（PSW）のフラグに，演算の結果がセット（１），リセット（０）されます。

図１－３　演算命令でのALUの働き



1.3.6 BCD演算

BCD演算命令により16進数を用いて10進数を表現し，計算することができます。

この命令により，メモリ上のBCDストリングに対して算術演算，比較もできます。

また，BCDストリングに対してのローテートをサポートする命令も含まれています。

演算，比較命令は，使用するレジスタが決まっているため，パックトBCDストリングを指定するオペランドはありません。

ソース・ストリングの先頭アドレス（LSDを含むバイト・データのアドレス）は，データ・セグメント０（DS0）内でIXレジスタの内容で指定します。

デスティネーション・ストリングの先頭アドレス（LSDを含むバイト・データのアドレス）は，データ・セグメント１（DS1）内でIYレジスタの内容で指定します。

桁数は，CLレジスタの内容で指定します。

デスティネーション・ストリングとソース・ストリングは，同一の長さ（桁数）でなければならないので，長さが異なる場合は長い方に合わせて０を拡張します。

1.3.7 BCD補正

算術演算の実行前あるいは実行後にBCD補正命令を実行することで，BCD演算をサポートします。

BCD補正命令はALレジスタに対して行うので，オペランドはありません。加算，減算の場合，パックトBCDおよびアンパックトBCDのどちらでも補正できますが，乗除算の場合，アンパックトBCD表現の演算でしか補正できません。

保守／廃止

1.3.8　データ変換

データ変換命令により２進数と10進数の型変換，語長の変換ができます。

CVTBD命令とCVTDB命令は，２進数と２桁のアンパックトBCDの型変換をする命令です。

CVTBW命令とCVTWL命令は，レジスタ内の符号拡張をする命令です。

1.3.9　ビット操作

ビット操作命令により汎用レジスタあるいはメモリのビット・データに対して，論理演算ができます。

命令形式のオペランドは，“ reg, bit ”または，“ mem, bit ”となります。

第１オペランドのregまたはmemは，ビット・データを含んでいる８/16ビット・データを指定するもので，汎用レジスタまたは実効アドレスを記述します。

第２オペランドのbitは，ビット・データのバイト／ワード内アドレスを示すもので，CLの内容または８ビット・イミーディエト・データを使用します。

ただし，regまたはmemが８ビット・データの場合は，下位３ビットのみが有効なビット・アドレスとなり，16ビット・データの場合は，下位４ビットのみが有効なビット・アドレスで，上位ビットは無視されます。

1.3.10　シフト，ローテート

シフト命令，ローテート命令により汎用レジスタあるいはメモリの８/16ビット・データを１ビットまたは複数ビット（0-255），シフトまたはローテートできます。

シフトには，算術シフトと論理シフトがあります。シフトする桁数は通常では１ですが，命令のカウント・オペランドで指定すれば，CLレジスタの値によって実行するたびに変更できます（最大255）。算術シフトは，左に１ビットずらすときにLSBに０が入り，右に１ビットずらすとMSBに０が入ります。論理シフトの場合は，１ビットずらしてもLSBビットまたはMSBの値は変化しません。

ローテートもシフトと同様，ローテートする桁数はカウント・オペランドでCLレジスタに入れた値を指定します。この命令では，CYフラグとVフラグだけが変化します。CYフラグには，常にローテートして最後に押し出されたビットが入ります。Vフラグは，２桁以上のローテートの場合は必ず不定となり，１桁の場合はデスティネーションのMSB（延長）が変化するとセット（１），変化しなければリセット（０）されます。CYフラグは，ROLC命令とRORC命令でデスティネーションの延長として使用できます。

1.3.11　スタック操作

スタック操作命令によりメモリ上のスタック操作ができます。

スタック操作命令には次の４種の命令があります。

PUSH命令	：スタックへデータを退避させる命令です。
POP命令	：スタックからデータを復帰させる命令です。
PREPARE命令	：スタック・フレームを生成する命令で，ローカル変数の領域確保と，グローバル変数への参照を可能とするためにフレーム・ポインタをコピーします。
DISPOSE命令	：スタック・ポインタ（SP）およびベース・ポインタ（BP）をPREPARE命令を実行する直前の状態に戻す命令です。

保守／廃止

1.3.12　プログラム分岐

指定されたアドレスに分岐できます。

次の４種に大別できます。

サブルーチン制御命令	：プログラム・カウンタ（PC）の内容をスタックに退避させる命令（CALL）と，スタックからPCの内容をリストアする命令（RET）があります。
ブランチ命令	：命令の流れを他のアドレスに移します。
条件付きブランチ命令	：フラグの値によって命令の実行の流れを他のアドレスに移します。
割り込み命令	：外部デバイスからの割り込み要求や演算エラーが発生した際に，プログラムの実行を一時中止し，ソフトウエア割り込みによりプログラム実行の流れを制御します。

1.3.13　CPU制御

CPU制御命令により，フラグの操作や，外部デバイスとの同期，データ転送ができます。また，CPUに何もさせない命令（NOP）もあります。

1.3.14　レジスタ・バンク切り替え

レジスタ・バンクを切り替える命令です。高速なサブルーチン・コールなどに使用します。

第2章　命　　令

2.1　命令の説明（ニモニックのアルファベット順）

この節では各命令を次の項目に分けて説明します。

【命　令　形　式】
【オペレーション】
【オ　ペ　ラ　ン　ド】
【フ　　ラ　　グ】
【説　　　　明】
【記　　述　　例】
【バ　イ　ト　数】
【命　令　語　形　式】

【命令形式】，【オペレーション】，【オペランド】の各項目では説明上いくつもの識別子を用いています。表２－２から表２－４に識別子とその意味を，表２－５から表２－７にメモリ・アドレシング，汎用レジスタ／セグメント・レジスタの選択について示します。

【フラグ】の項目は命令実行により変化するフラグの動作を識別子により示します。各フラグの動作の凡例を表２－１に示します。

表２－１　フラグ動作の凡例

識別子	説　　明
空白	変化なし
0	リセット（0）される
1	セット（1）される
×	結果に従ってセット（1）またはリセット（0）される
U	不定
R	以前に退避した値がリストアされる

保守／廃止

表2－2　オペランド・タイプの凡例

識　別　子	説　　明
reg	8/16ビット汎用レジスタ （8/16ビット汎用レジスタを２つ用いる命令における，デスティネーション側レジスタ）
reg'	8/16ビット汎用レジスタを２つ用いる命令における，ソース側レジスタ
reg8	８ビット汎用レジスタ （８ビット汎用レジスタを２つ用いる命令における，デスティネーション側レジスタ）
reg8'	８ビット汎用レジスタを２つ用いる命令における，ソース側レジスタ
reg16	16ビット汎用レジスタ （16ビット汎用レジスタを２つ用いる命令における，デスティネーション側レジスタ）
reg16'	16ビット汎用レジスタを２つ用いる命令における，ソース側レジスタ
mem	8/16ビット・メモリ・アドレス
mem8	８ビット・メモリ・アドレス
mem16	16ビット・メモリ・アドレス
mem32	32ビット・メモリ・アドレス
sfr	８ビット特殊機能レジスタ・ロケーション
dmem	16ビット・ダイレクト・メモリ・アドレス
imm	8/16ビット・イミーディエト・データ
imm3	３ビット・イミーディエト・データ
imm4	４ビット・イミーディエト・データ
imm8	８ビット・イミーディエト・データ
imm16	16ビット・イミーディエト・データ
acc	アキュームレータ（AWまたはAL）
sreg	セグメント・レジスタ
src-table	256バイト変換テーブルの名称
src-block	IXレジスタでアドレスされるソース・ブロックの名称
dst-block	IYレジスタでアドレスされるデスティネーション・ブロックの名称
near-proc	現在のプログラム・セグメント内のプロシージャ
far-proc	別のプログラム・セグメント内のプロシージャ
near-label	現在のプログラム・セグメント内のレーベル
short-label	命令の終わりから－128～＋127バイトの範囲のレーベル
far-label	別のプログラム・セグメント内のレーベル
regptr16	現在のプログラム・セグメント内のコール・アドレスのオフセットを持つ16ビット汎用レジスタ
memptr16	現在のプログラム・セグメント内のコール・アドレスのオフセットを持つ16ビット・メモリ・アドレス
memptr32	別のプログラム・セグメント内のコール・アドレスのオフセットとセグメント・データを持つ32ビット・メモリ・アドレス
pop-value	スタックから捨てるバイト数（０-64 K，通常は偶数）
fp-op	浮動小数点演算用コプロセッサの命令コードを判別するイミーディエト値
R	レジスタ・セット（AW, BW, CW, DW, SP, BP, IX, IY）
DS1-spec	DS1, またはDS1にASSUMEしてあるセグメント名／グループ名
Seg-spec	任意のセグメント・レジスタ名，またはセグメント・レジスタにASSUMEしてあるセグメント名／グループ名
〔　〕	省略可能

保守／廃止

表２－３　命令語の凡例

識　別　子	説　　明
W	バイト／ワード・フィールド（０，１）
reg	レジスタ・フィールド（000-111）
reg'	レジスタ・フィールド（000-111）（レジスタを２つ用いる命令における，ソース側レジスタ）
mod, mem	メモリ・アドレシング指定ビット（mod：00-10, mem：000-111）
（disp-low）	オプション16ビット・ディスプレースメント下位バイト
（disp-high）	〃　　上位バイト
disp-low	PCレラティブ加算用16ビット・ディスプレースメント下位バイト
disp-high	〃　　上位バイト
imm3	３ビット・イミーディエト・データ
imm4	４ビット　　〃
imm8	８ビット　　〃
imm16-low	16ビット・イミーディエト・データの下位バイト
imm16-high	〃　　上位バイト
addr-low	16ビット・ダイレクト・アドレスの下位バイト
addr-high	〃　　上位バイト
sreg	セグメント・レジスタ指定ビット（00-11）
s	サイン拡張指定ビット（１：サイン拡張あり，０：サイン拡張なし）
offset-low	PCにロードされる16ビット・オフセット・データの下位バイト
offset-high	〃　　上位バイト
seg-low	PSにロードされる16ビット・セグメント・データの下位バイト
seg-high	〃　　上位バイト
pop-value-low	スタックの捨てるバイト数を指定する16ビット・データの下位バイト
pop-value-high	〃　　上位バイト
disp8	PCにレラティブ加算される８ビット・ディスプレースメント
X	浮動小数点演算用コプロセッサのオペレーション・コード
XXX	
YYY	
ZZZ	

表２－４　命令形式またはオペレーション説明上の凡例（1/2）

識　別　子	説　　明
dst	デスティネーション・オペランド
dst1	〃
dst2	〃
src	ソース・オペランド
src1	〃
src2	〃
target	ターゲット・オペランド
AW	アキュームレータ（16ビット）
AH	〃　　（上位バイト）
AL	〃　　（下位バイト）
BW	BWレジスタ（16ビット）
CW	CWレジスタ（　〃　）
CL	〃　　（下位バイト）
DW	DWレジスタ（16ビット）
BP	ベース・ポインタ（16ビット）

保守／廃止

表２－４　命令形式またはオペレーション説明上の凡例（2/2）

識　別　子	説　　明
SP	スタック・ポインタ（16ビット）
PC	プログラム・カウンタ（16ビット）
PSW	プログラム・ステータス・ワード（16ビット）
IX	インデクス・レジスタ（ソース）（16ビット）
IY	〃　　（デスティネーション）（16ビット）
PS	プログラム・セグメント・レジスタ（16ビット）
SS	スタック・セグメント・レジスタ（16ビット）
DS0	データ・セグメント０レジスタ（16ビット）
DS1	データ・セグメント１レジスタ（16ビット）
AC	補助キャリー・フラグ
CY	キャリー・フラグ
P	パリティ・フラグ
S	サイン・フラグ
Z	ゼロ・フラグ
DIR	方向フラグ
IE	割り込み許可フラグ
V	オーバフロー・フラグ
BRK	ブレーク・フラグ
$\overline{\text{IBRK}}$	I/Oブレーク・フラグ
RB0	レジスタ・バンク・フラグ
RB1	〃
RB2	〃
F0	ユーザ・フラグ
F1	〃
(…)	（　）内で示されるメモリの内容
disp	ディスプレースメント（8/16ビット）
temp	テンポラリ・レジスタ（8/16/32ビット）
temp1	テンポラリ・レジスタ（16ビット）
temp2	〃
TA	テンポラリ・レジスタA（16ビット）
TB	テンポラリ・レジスタB（16ビット）
TC	テンポラリ・レジスタC（16ビット）
ext-disp8	８ビット・ディスプレースメントをサイン拡張した16ビット
seg	イミーディエト・セグメント・データ（16ビット）
offset	イミーディエト・オフセット・データ（16ビット）
←	転送方向
＋	加　算
－	減　算
×	乗　算
÷	除　算
%	モジュロ
∧	論理積（AND）
∨	論理和（OR）
⊻	排他的論理和（XOR）
××H	16進数２けたの数値
××××H	16進数４けたの数値

表２－５　メモリ・アドレシング

mod / mem	00	01	10
000	BW＋IX	BW＋IX＋disp8	BW＋IX＋disp16
001	BW＋IY	BW＋IY＋disp8	BW＋IY＋disp16
010	BP＋IX	BP＋IX＋disp8	BP＋IX＋disp16
011	BP＋IY	BP＋IY＋disp8	BP＋IY＋disp16
100	IX	IX＋disp8	IX＋disp16
101	IY	IY＋disp8	IY＋disp16
110	ダイレクト・アドレス	BP＋disp8	BP＋disp16
111	BW	BW＋disp8	BW＋disp16

表２－６　8/16ビット汎用レジスタの選択

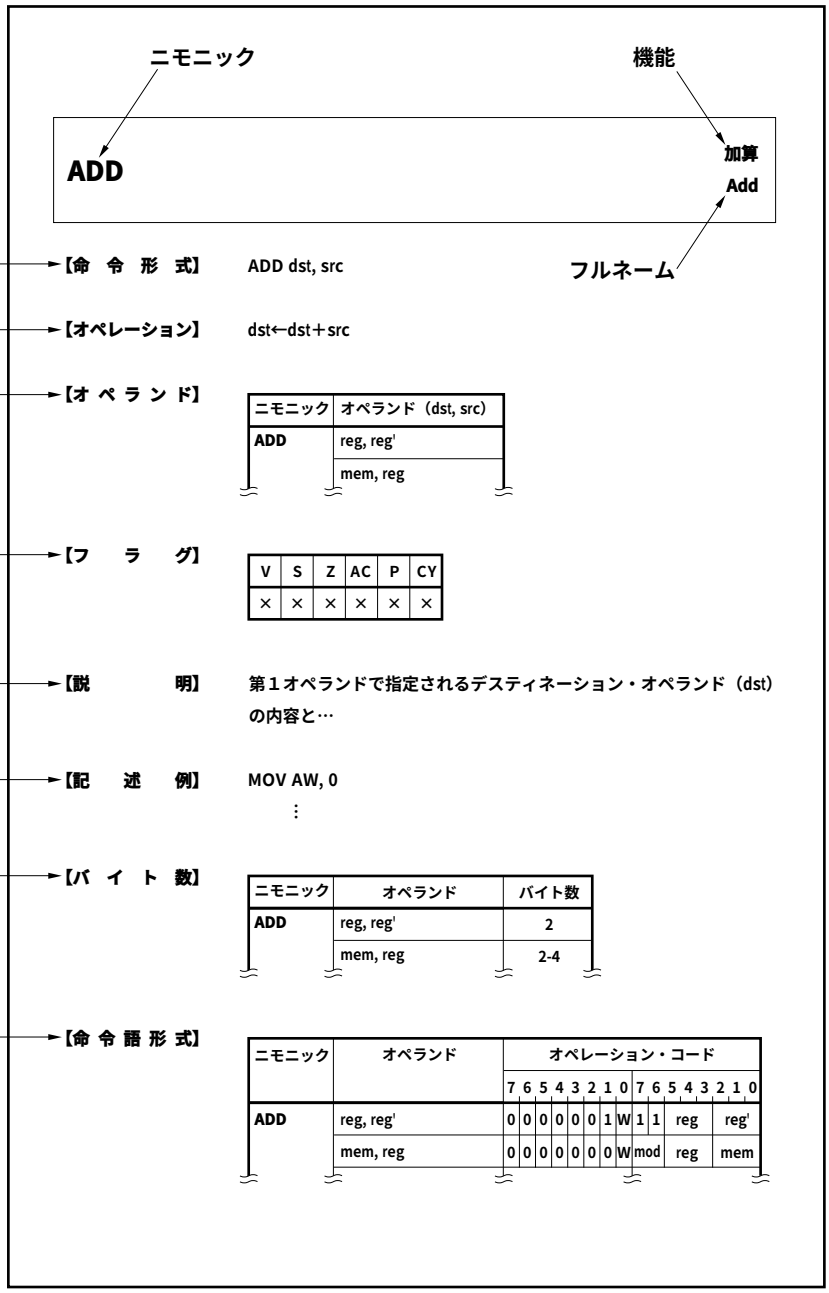
reg, reg'	W = 0	W = 1
000	AL	AW
001	CL	CW
010	DL	DW
011	BL	BW
100	AH	SP
101	CH	BP
110	DH	IX
111	BH	IY

表２－７　セグメント・レジスタの選択

sreg	
00	DS1
01	PS
10	SS
11	DS0

保守／廃止

図2－1 記 述 例



オペレーション・コード	
7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
１バイト目	２バイト目
３バイト目	４バイト目
５バイト目	６バイト目

保守／廃止

ADD

加算
Add

【命 令 形 式】　ADD dst, src

【オペランド，オペレーション】

ニモニック	オペランド（dst, src）	オペレーション
ADD	reg, reg'	dst←dst＋src
	mem, reg	
	reg, mem	
	reg, imm	
	mem, imm	
	acc, imm	［W = 0のとき］AL←AL＋imm8 ［W = 1のとき］AW←AW＋imm16

【フ　ラ　グ】

V	S	Z	AC	P	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）の内容と第２オペランドで指定されるソース・オペランド（src）の内容を加算し，結果をデスティネーション・オペランド（dst）に格納します。

【記　述　例】　メモリ０：50Hの内容（ワード・データ）とDWレジスタの内容を加算して０：50Hへストアする。

```
MOV    AW, 0
MOV    DS1, AW
MOV    IY, 50H
ADD    DS1：WORD PTR［IY］, DW
```

保守／廃止

【バ イ ト 数】

ニモニック	オペランド	バイト数
ADD	reg, reg'	2
	mem, reg	2-4
	reg, mem	2-4
	reg, imm	3, 4
	mem, imm	3-6
	acc, imm	2, 3

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
ADD	reg, reg'	0	0	0	0	0	0	1	W	1	1	reg			reg'		
	mem, reg	0	0	0	0	0	0	0	W	mod		reg			mem		
		(disp-low)								(disp-high)							
	reg, mem	0	0	0	0	0	0	1	W	mod		reg			mem		
		(disp-low)								(disp-high)							
	reg, imm	1	0	0	0	0	0	s	W	1	1	0	0	0	reg		
		imm8 or imm16-low								imm16-high							
	mem, imm	1	0	0	0	0	0	s	W	mod		0	0	0	mem		
		(disp-low)								(disp-high)							
		imm8 or imm16-low								imm16-high							
	acc, imm	0	0	0	0	0	1	0	W	imm8 or imm16-low							
		imm16-high								—							

保守／廃止

ADD4S

10進加算
Add Nibble String

【命 令 形 式】　ADD4S［DS1-spec：］dst-string,［Seg-spec：］src-string
ADD4S

【オペレーション】　BCDストリング（IY, CL）←BCDストリング（IY, CL）＋BCDストリング（IX, CL）

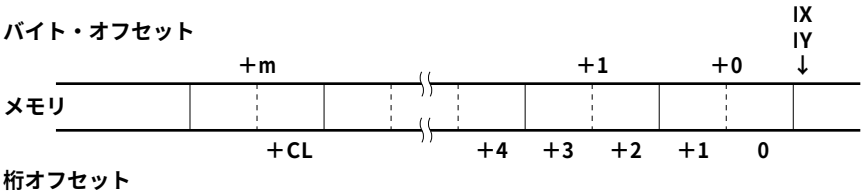
【オ ペ ラ ン ド】

ニモニック	オペランド（dst, src）
ADD4S	［DS1-spec：］dst-string,［Seg-spec：］src-string
	な し

【フ ラ グ】

V	S	Z	AC	P	CY
U	U	×	U	U	×

【説　　明】　IXレジスタでアドレスされるパックトBCDストリングとIYレジスタでアドレスされるパックトBCDストリングを加算し，結果をIYレジスタでアドレスされるストリングにストアします。ストリング長（BCD桁数）は，CLレジスタ（CLの内容がdならばd桁）によって決定され1-254桁まで可能です。
デスティネーション・ストリングは，常にDS1レジスタで指されるセグメント内にロケートされる必要があり，セグメント・オーバライドはできません。一方，ソース・ストリングは，ディフォールト・セグメント・レジスタはDS0レジスタとなっていますが，セグメント・オーバライド可能で，任意のセグメント・レジスタで指定されるセグメント内にロケートできます。
パックトBCDストリングのフォーマットを次に示します。



注意　BCDストリング命令は常に偶数桁単位で動作します。このため桁数として偶数を指定したときは演算結果，および各フラグは正しく動きますが，桁数として奇数を指定した場合は奇数＋１の偶数桁演算を実行し，演算結果，および各フラグは偶数桁の演算結果を示しますので，桁数として奇数を指定する場合は次のようにしてください。
桁数が奇数のときのBCD加算命令は，最上位バイトの上位４ビットを“０”にしてから実行してください。この結果，キャリーは最上位バイトのビット４に示され，フラグには反映されません。

保守／廃止

【記　述　例】　MOV　IX, OFFSET　VAR_1
MOV　IY, OFFSET　VAR_2
MOV　CL, 4
ADD4S

【バ イ ト 数】　2

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
ADD4S	［DS1-spec：］dst-string, ［Seg-spec：］src-string	0	0	0	0	1	1	1	1	0	0	1	0	0	0	0	0
	な　し																

保守／廃止

ADDC

キャリーを含む加算
Add with Carry

【命 令 形 式】　ADDC dst, src

【オペランド，オペレーション】

ニモニック	オペランド（dst, src）	オペレーション
ADDC	reg, reg'	dst←dst＋src＋CY
	mem, reg	
	reg, mem	
	reg, imm	
	mem, imm	
	acc, imm	［W = 0のとき］AL←AL＋imm8＋CY ［W = 1のとき］AW←AW＋imm16＋CY

【フ ラ グ】

V	S	Z	AC	P	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）の内容と第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CYフラグの内容も含めて加算し，結果をデスティネーション・オペランド（dst）に格納します。

【記 述 例】
```
SET1   CY        ；CYフラグをセット（１）する。
XOR    AW, AW    ；AW = 0
MOV    BW, 0FFH  ；BW = 0FFH
ADDC   AW, BW    ；AWレジスタの内容 = 100Hとなる。
```

【バ イ ト 数】

ニモニック	オペランド	バイト数
ADDC	reg, reg'	2
	mem, reg	2-4
	reg, mem	2-4
	reg, imm	3, 4
	mem, imm	3-6
	acc, imm	2, 3

保守／廃止

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
ADDC	reg, reg'	0	0	0	1	0	0	1	W	1	1	reg			reg'		
	mem, reg	0	0	0	1	0	0	0	W	mod		reg			mem		
		(disp-low)								(disp-high)							
	reg, mem	0	0	0	1	0	0	1	W	mod		reg			mem		
		(disp-low)								(disp-high)							
	reg, imm	1	0	0	0	0	0	s	W	1	1	0	1	0	reg		
		imm8 or imm16-low								imm16-high							
	mem, imm	1	0	0	0	0	0	s	W	mod		0	1	0	mem		
		(disp-low)								(disp-high)							
		imm8 or imm16-low								imm16-high							
	acc, imm	0	0	0	1	0	1	0	W	imm8 or imm16-low							
		imm16-high								—							

保守／廃止

ADJ4A

加算結果のパックト10進補正
Adjust Nibble Add

【命 令 形 式】　ADJ4A

【オペレーション】　AL∧0FH＞9またはAC = 1のとき
AL←AL＋6
AC←1
AL＞9FHまたはCY = 1のとき
AL←AL＋60H
CY←1

【オ ペ ラ ン ド】

ニモニック	オペランド
ADJ4A	な　し

【フ　ラ　グ】

V	S	Z	AC	P	CY
U	×	×	×	×	×

【説　　明】　２つのパックト10進数の加算のALレジスタ内の結果を１つのパックト10進数に補正します。

【記　述　例】　ADJ4A

【バ イ ト 数】　１

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード							
		7	6	5	4	3	2	1	0
ADJ4A	な　し	0	0	1	0	0	1	1	1

保守／廃止

ADJ4S

減算結果のパックト10進補正
Adjust Nibble Subtract

【命 令 形 式】　ADJ4S

【オペレーション】　AL∧0FH＞9またはAC = 1のとき
AL←AL－6
AC←1
AL＞9FHまたはCY = 1のとき
AL←AL－60H
CY←1

【オ ペ ラ ン ド】

ニモニック	オペランド
ADJ4S	な　し

【フ　ラ　グ】

V	S	Z	AC	P	CY
U	×	×	×	×	×

【説　　明】　２つのパックト10進数の減算のALレジスタ内の結果を１つのパックト10進数に補正します。

【記　述　例】　SUB　AW, BW
ADJ4S

【バ イ ト 数】　１

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード							
		7	6	5	4	3	2	1	0
ADJ4S	な　し	0	0	1	0	1	1	1	1

保守／廃止

ADJBA

加算結果のアンパックト10進補正
Adjust Byte Add

【命 令 形 式】　ADJBA

【オペレーション】　AL∧0FH＞9またはAC = 1のとき
AL←AL＋6
AH←AH＋1
AC←1
CY←AC
AL←AL∧0FH

【オ ペ ラ ン ド】

ニモニック	オペランド
ADJBA	な　し

【フ　ラ　グ】

V	S	Z	AC	P	CY
U	U	U	×	U	×

【説　　明】　２つのアンパックト10進数の加算のALレジスタ内の結果を１つのアンパックト10進数に補正します。上位４ビットは０になります。

【記　述　例】　ADJBA

【バ イ ト 数】　１

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード							
		7	6	5	4	3	2	1	0
ADJBA	な　し	0	0	1	1	0	1	1	1

保守／廃止

ADJBS

減算結果のアンパックト10進補正
Adjust Byte Subtract

【命 令 形 式】　ADJBS

【オペレーション】　AL∧0FH＞9またはAC = 1のとき
AL←AL－6
AH←AH－1
AC←1
CY←AC
AL←AL∧0FH

【オ ペ ラ ン ド】

ニモニック	オペランド
ADJBS	な　し

【フ　ラ　グ】

V	S	Z	AC	P	CY
U	U	U	×	U	×

【説　　明】　２つのアンパックト10進数の減算のALレジスタ内の結果を１つのアンパックト10進数に補正します。上位４ビットは０になります。

【記　述　例】
```
SUB    AW, BW
ADJBS
```

【バ イ ト 数】　１

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード							
		7	6	5	4	3	2	1	0
ADJBS	な　し	0	0	1	1	1	1	1	1

保守／廃止

AND

論理積
And

【命 令 形 式】　AND dst, src

【オペランド，オペレーション】

ニモニック	オペランド（dst, src）	オペレーション
AND	reg, reg'	dst←dst∧src
	mem, reg	
	reg, mem	
	reg, imm	
	mem, imm	
	acc, imm	［W = 0のとき］AL←AL∧imm8 ［W = 1のとき］AW←AW∧imm16

【フ ラ グ】

V	S	Z	AC	P	CY
0	×	×	U	×	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）と第２オペランドで指定されるソース・オペランド（src）の論理積（AND）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。

【記 述 例】　MOV　DW, IY
AND　DW, 7FFFH

【バ イ ト 数】

ニモニック	オペランド	バイト数
AND	reg, reg'	2
	mem, reg	2-4
	reg, mem	2-4
	reg, imm	3, 4
	mem, imm	3-6
	acc, imm	2, 3

保守／廃止

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
AND	reg, reg'	0	0	1	0	0	0	1	W	1	1	reg			reg'		
	mem, reg	0	0	1	0	0	0	0	W	mod		reg			mem		
		(disp-low)								(disp-high)							
	reg, mem	0	0	1	0	0	0	1	W	mod		reg			mem		
		(disp-low)								(disp-high)							
	reg, imm[注]	1	0	0	0	0	0	0	W	1	1	1	0	0	reg		
		imm8 or imm16-low								imm16-high							
	mem, imm	1	0	0	0	0	0	0	W	mod		1	0	0	mem		
		(disp-low)								(disp-high)							
		imm8 or imm16-low								imm16-high							
	acc, imm	0	0	1	0	0	1	0	W	imm8 or imm16-low							
		imm16-high								—							

注 アセンブラ，コンパイラによっては，次に示すようなコードが生成されることがあります。

オペレーション・コード																	
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0		
1	0	0	0	0	0	1	W	1	1	1	0	0	reg				
imm8								—									

このような場合でも正常に命令を実行します。ただし，エミュレータによっては，この命令に対する逆アセンブラ機能，ライン・アセンブラ機能がサポートされていない場合がありますのでご注意ください。

保守／廃止

BC
BL

CY = 1による条件分岐
Branch if Carry
Branch if Lower

【命 令 形 式】 **BC short-label**
BL short-label

【オペレーション】 CY = 1のとき：PC←PC＋ext－disp8

【オ ペ ラ ン ド】

ニモニック	オペランド
BC	short-label
BL	

【フ ラ グ】

V	S	Z	AC	P	CY

【説　　明】 CYフラグが１のとき，現在のPCの値に８ビット・ディスプレースメント（実際にはサイン拡張された16ビット）を加えた値をロードします。
この命令の置かれているセグメント内でかつ，－128～＋127バイトのアドレス内にブランチできます。

【記　述　例】

```
      TEST  AL, BL
      BC    SHORT LP4 ；LP4 = レーベル
      ⋮
      TEST  AL, BL
      BL    SHORT LP5；LP5 = レーベル
      ⋮
LP4：
```

【バ イ ト 数】 2

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
BC	short-label	0	1	1	1	0	0	1	0	disp8							
BL																	

保守／廃止

BCWZ

CW = 0による条件分岐
Branch if CW equals Zero

【命 令 形 式】　BCWZ　short-label

【オペレーション】　CW = 0のとき：PC←PC＋ext-disp8

【オ ペ ラ ン ド】

ニモニック	オペランド
BCWZ	short-label

【フ　ラ　グ】

V	S	Z	AC	P	CY

【説　　明】　CWレジスタの値が０ならば，現在のPCの値に８ビット・ディスプレースメント（実際にはサイン拡張された16ビット）を加えた値をロードします。
この命令の置かれているセグメント内でかつ，－128～＋127バイトのアドレス内にブランチできます。
条件不成立のときは次の命令に進みます。

【記　述　例】
```
LP22：
        ⋮
        ADD     AL, BL
        BCWZ    SHORT LP22 ；LP22 = レーベル
```

【バ イ ト 数】　２

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
BCWZ	short-label	1	1	1	0	0	0	1	1	disp8							

保守／廃止

BE
BZ

Z = 1による条件分岐
Branch if Equal
Branch if Zero

【命 令 形 式】 **BE short-label**
BZ short-label

【オペレーション】 Z = 1のとき：PC←PC＋ext-disp8

【オ ペ ラ ン ド】

ニモニック	オペランド
BE	short-label
BZ	

【フ ラ グ】

V	S	Z	AC	P	CY

【説 明】 Zフラグが１のとき，現在のPCの値に８ビット・ディスプレースメント（実際にはサイン拡張された16ビット）を加えた値をロードします。
この命令の置かれているセグメント内でかつ，－128～＋127バイトのアドレス内にブランチできます。

【記 述 例】

```
      AND   AL, 2
      BE    SHORT LOOP ；LOOP = レーベル
      ⋮
      OR    AH, BH
      BZ    SHORT LOOP1；LOOP1 = レーベル
      ⋮
LOOP：
```

【バ イ ト 数】 2

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
BE	short-label	0	1	1	1	0	1	0	0	disp8							
BZ																	

保守／廃止

BGE

S⊽V = 0による条件分岐
Branch if Greater Than or Equal

【命 令 形 式】　BGE　short-label

【オペレーション】　S⊽V = 0のとき：PC←PC＋ext-disp8

【オ ペ ラ ン ド】

ニモニック	オペランド
BGE	short-label

【フ　ラ　グ】

V	S	Z	AC	P	CY

【説　　明】　SフラグとVフラグの排他的論理和（XOR）が0のとき，現在のPCの値に8ビット・ディスプレースメント（実際にはサイン拡張された16ビット）を加えた値をロードします。
この命令の置かれているセグメント内でかつ，－128～＋127バイトのアドレス内にブランチできます。
条件不成立のときは次の命令に進みます。

【記　述　例】

```
        SHL   AL, 1
        BGE   SHORT LP16 ；LP16 = レーベル
        ⋮
LP16：
```

【バ イ ト 数】　2

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
BGE	short-label	0	1	1	1	1	1	0	1	disp8							

保守／廃止

BGT

(S⊻V)∨Z = 0による条件分岐
Branch if Greater Than

【命 令 形 式】　BGT short-label

【オペレーション】　(S⊻V)∨Z = 0のとき：PC←PC＋ext-disp8

【オ ペ ラ ン ド】

ニモニック	オペランド
BGT	short-label

【フ　ラ　グ】

V	S	Z	AC	P	CY

【説　　明】　Sフラグ とVフラグの排他的論理和（XOR）をとった結果とZフラグの論理和（OR）が0のとき，現在のPCの値に８ビット・ディスプレースメント（実際にはサイン拡張された16ビット）を加えた値をロードします。
この命令の置かれているセグメント内でかつ，−128〜＋127バイトのアドレス内にブランチできます。
条件不成立のときは次の命令に進みます。

【記　述　例】
```
LP18：
        ⋮
        SHL   AL, 1
        BGT   LP18
```

【バ イ ト 数】　2

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
BGT	short-label	0	1	1	1	1	1	1	1	disp8							

保守／廃止

BH

CY∨Z = 0による条件分岐
Branch if Higher

【命 令 形 式】　BH short-label

【オペレーション】　CY∨Z = 0のとき：PC←PC＋ext-disp8

【オ ペ ラ ン ド】

ニモニック	オペランド
BH	short-label

【フ　ラ　グ】

V	S	Z	AC	P	CY

【説　　明】　CYフラグとZフラグの論理和（OR）が0のとき，現在のPCの値に8ビット・ディスプレースメント（実際にはサイン拡張された16ビット）を加えた値をロードします。
この命令の置かれているセグメント内でかつ，－128～＋127バイトのアドレス内にブランチできます。

【記　述　例】

```
        ROL  AL, 1
        BH   SHORT LP10 ；LP10 = レーベル
        ⋮
LP10：
```

【バ イ ト 数】　2

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
BH	short-label	0	1	1	1	0	1	1	1	disp8							

保守／廃止

BLE

(S⨁V)∨Z = 1による条件分岐
Branch if Less than or Equal

【命 令 形 式】　BLE short-label

【オペレーション】　(S⨁V)∨Z = 1のとき：PC←PC＋ext-disp8

【オ ペ ラ ン ド】

ニモニック	オペランド
BLE	short-label

【フ　ラ　グ】

V	S	Z	AC	P	CY

【説　　明】　SフラグとVフラグの排他的論理和（XOR）をとった結果とZフラグの論理和（OR）が1のとき，現在のPCの値に8ビット・ディスプレースメント（実際にはサイン拡張された16ビット）を加えた値をロードします。
この命令の置かれているセグメント内でかつ，−128〜＋127バイトのアドレス内にブランチできます。
条件不成立のときは次の命令に進みます。

【記　述　例】
```
LP17：
        ⋮
        SHR   AL, 1
        BLE   SHORT LP17
```

【バ イ ト 数】　2

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
BLE	short-label	0	1	1	1	1	1	1	0	disp8							

保守／廃止

BLT

S▽V = 1による条件分岐
Branch if Less Than

【命 令 形 式】　BLT short-label

【オペレーション】　S▽V = 1のとき：PC←PC＋ext-disp8

【オ ペ ラ ン ド】

ニモニック	オペランド
BLT	short-label

【フ　ラ　グ】

V	S	Z	AC	P	CY

【説　　明】　SフラグとVフラグの排他的論理和（XOR）が１のとき，現在のPCの値に８ビット・ディスプレースメント（実際にはサイン拡張された16ビット）を加えた値をロードします。
この命令の置かれているセグメント内でかつ，－128～＋127バイトのアドレス内にブランチできます。
条件不成立のときは次の命令に進みます。

【記　述　例】

```
        ADD   AL, BL
        BLT   SHORT LP15 ；LP15 = レーベル
        ⋮
LP15：
```

【バ イ ト 数】　２

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
BLT	short-label	0	1	1	1	1	1	0	0	disp8							

保守／廃止

BN

S = 1による条件分岐
Branch if Negative

【命 令 形 式】　BN short-label

【オペレーション】　S = 1のとき：PC←PC＋ext-disp8

【オ ペ ラ ン ド】

ニモニック	オペランド
BN	short-label

【フ　ラ　グ】

V	S	Z	AC	P	CY

【説　　明】　Sフラグが１のとき，現在のPCの値に８ビット・ディスプレースメント（実際にはサイン拡張された16ビット）を加えた値をロードします。
この命令の置かれているセグメント内でかつ，－128～＋127バイトのアドレス内にブランチできます。

【記　述　例】

```
        ADD   AL, BL
        BN    LP11 ；LP11 = レーベル
        ⋮
LP11：
```

【バ イ ト 数】　２

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
BN	short-label	0	1	1	1	1	0	0	0	disp8							

保守／廃止

BNC
BNL

CY = 0による条件分岐
Branch if Not Carry
Branch if Not Lower

【命 令 形 式】　BNC short-label
BNL　short-label

【オペレーション】　CY = 0のとき：PC←PC＋ext-disp8

【オ ペ ラ ン ド】

ニモニック	オペランド
BNC	short-label
BNL	

【フ　ラ　グ】

V	S	Z	AC	P	CY

【説　　明】　CYフラグが0のとき，現在のPCの値に8ビット・ディスプレースメント（実際にはサイン拡張された16ビット）を加えた値をロードします。
この命令の置かれているセグメント内でかつ，−128〜＋127バイトのアドレス内にブランチできます。

【記　述　例】

```
        ROR    AL, 1
        BNC    SHORT LP6 ；LP6 = レーベル
        ⋮
        ROR    AL, 1
        BNL    SHORT LP7 ；LP7 = レーベル
        ⋮
LP6：
```

【バ イ ト 数】　2

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
BNC	short-label	0	1	1	1	0	0	1	1	disp8							
BNL																	

保守／廃止

BNE
BNZ

Z = 0による条件分岐
Branch if Not Equal
Branch if Not Zero

【命 令 形 式】 **BNE short-label**
BNZ short-label

【オペレーション】 Z = 0のとき：PC←PC＋ext-disp8

【オ ペ ラ ン ド】

ニモニック	オペランド
BNE	short-label
BNZ	

【フ ラ グ】

V	S	Z	AC	P	CY

【説 明】 Zフラグが０のとき，現在のPCの値に８ビット・ディスプレースメント（実際にはサイン拡張された16ビット）を加えた値をロードします。
この命令の置かれているセグメント内でかつ，－128～＋127バイトのアドレス内にブランチできます。

【記 述 例】

```
      OR    AL, BL
      BNE   SHORT LP8 ；LP8 = レーベル
      ⋮
      AND   SH, BH
      BNZ   SHORT LP9 ；LP9 = レーベル
      ⋮
LP8：
```

【バ イ ト 数】 2

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
BNE	short-label	0	1	1	1	0	1	0	1	disp8							
BNZ																	

保守／廃止

BNH

CY∨Z = 1による条件分岐
Branch if Not Higher

【命 令 形 式】　BNH short-label

【オペレーション】　CY∨Z = 1のとき：PC←PC＋ext-disp8

【オ ペ ラ ン ド】

ニモニック	オペランド
BNH	short-label

【フ　ラ　グ】

V	S	Z	AC	P	CY

【説　　明】　CYフラグとZフラグの論理和（OR）が１のとき，現在のPCの値に８ビット・ディスプレースメント（実際にはサイン拡張された16ビット）を加えた値をロードします。
この命令の置かれているセグメント内でかつ，－128～＋127バイトのアドレス内にブランチできます。

【記　述　例】

```
        ROR   AL, 1
        BNH   SHORT LP9 ；LP9 = レーベル
        ⋮
LP9：
```

【バ イ ト 数】　２

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード													
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
BNH	short-label	0	1	1	1	0	1	1	0	disp8							

保守／廃止

BNV

V = 0による条件分岐
Branch if Not Overflow

【命 令 形 式】　BNV short-label

【オペレーション】　V = 0のとき：PC←PC＋ext-disp8

【オ ペ ラ ン ド】

ニモニック	オペランド
BNV	short-label

【フ　ラ　グ】

V	S	Z	AC	P	CY

【説　　明】　Vフラグが０のとき，現在のPCの値に８ビット・ディスプレースメント（実際にはサイン拡張された16ビット）を加えた値をロードします。
この命令の置かれているセグメント内でかつ，－128～＋127バイトのアドレス内にブランチできます。

【記　述　例】

```
        ROR   AL, 1
        BNV   LP3
        ⋮
LP3：
```

【バ イ ト 数】　２

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
BNV	short-label	0	1	1	1	0	0	0	1	disp8							

保守／廃止

BP

S = 0による条件分岐
Branch if Positive

【命 令 形 式】　BP short-label

【オペレーション】　S = 0のとき：PC←PC＋ext-disp8

【オ ペ ラ ン ド】

ニモニック	オペランド
BP	short-label

【フ　ラ　グ】

V	S	Z	AC	P	CY

【説　　明】　Sフラグが０のとき，現在のPCの値に８ビット・ディスプレースメント（実際にはサイン拡張された16ビット）を加えた値をロードします。
この命令の置かれているセグメント内でかつ，－128～＋127バイトのアドレス内にブランチできます。

【記　述　例】

```
        SHR   AL, 1
        BP    SHORT LP12 ；LP12 = レーベル
        ⋮
LP12：
```

【バ イ ト 数】　２

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
BP	short-label	0	1	1	1	1	0	0	1	disp8							

保守／廃止

BPE

P = 1による条件分岐
Branch if Parity Even

【命 令 形 式】　BPE short-label

【オペレーション】　P = 1のとき：PC←PC＋ext-disp8

【オ ペ ラ ン ド】

ニモニック	オペランド
BPE	short-label

【フ　ラ　グ】

V	S	Z	AC	P	CY

【説　　明】　Pフラグが１のとき，現在のPCの値に８ビット・ディスプレースメント（実際にはサイン拡張された16ビット）を加えた値をロードします。
この命令の置かれているセグメント内でかつ，－128～＋127バイトのアドレス内にブランチできます。

【記　述　例】

```
        ADD   AL, BL
        BPE   SHORT LP13 ；LP13 = レーベル
        ⋮
LP13：
```

【バ イ ト 数】　２

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
BPE	short-label	0	1	1	1	1	0	1	0	disp8							

保守／廃止

BPO

P = 0による条件分岐
Branch if Parity Odd

【命 令 形 式】 **BPO short-label**

【オペレーション】 P = 0のとき：PC←PC＋ext-disp8

【オ ペ ラ ン ド】

ニモニック	オペランド
BPO	short-label

【フ　ラ　グ】

V	S	Z	AC	P	CY

【説　　明】 Pフラグが０のとき，現在のPCの値に８ビット・ディスプレースメント（実際にはサイン拡張された16ビット）を加えた値をロードします。
この命令の置かれているセグメント内でかつ，－128～＋127バイトのアドレス内にブランチできます。

【記　述　例】

```
        ADD  AL, BL
        BPO  SHORT LP14 ；LP14 = レーベル
        ⋮
LP14：
```

【バ イ ト 数】 ２

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
BPO	short-label	0	1	1	1	1	0	1	1	disp8							

保守／廃止

BR

無条件分岐
Branch

【命 令 形 式】　BR target

【オペランド，オペレーション】

ニモニック	オペランド（target）	オペレーション
BR	near-label	PC←PC＋disp
	short-label	PC←PC＋ext-disp8
	regptr16	PC←target
	memptr16	
	far-label	PS←seg PC←offset
	memptr32	PS←（memptr32＋3, memptr32＋2） PC←（memptr32＋1, memptr32）

【フ ラ グ】

V	S	Z	AC	P	CY

【説　　明】　○target = near-labelのとき

現在のPCの値に16ビット・ディスプレースメント（disp）を加えた値をPCに転送します。
ブランチ・アドレスがこの命令の置かれているセグメント内にあれば，アセンブラが自動的にこの命令を実行します。

○target = short-labelのとき

現在のPCの値に８ビット・ディスプレースメント（実際にはサイン拡張された16ビット（ext-disp8））を加えた値をPCに転送します。
ブランチ・アドレスがこの命令の置かれているセグメント内でかつ±127バイト内にあれば，アセンブラが自動的にこの命令を実行します。

○target = regptr16またはtarget = memptr16のとき

ターゲット・オペランド（target）の内容をPCに転送します。
この命令の置かれているセグメント内の任意のアドレスにブランチできます。

○target = far-labelのとき

PCに命令の２，３バイト目の16ビット・オフセット・データを，PSに命令の４，５バイト目の16ビット・セグメント・データを転送します。
任意のセグメントの任意のアドレスにブランチできます。

○target = memptr32のとき

32ビット・メモリの上位２バイトをPSに，下位２バイトをPCにロードします。任意のセグメントの任意のアドレスにブランチできます。

【記　述　例】　BR $－8

【バ イ ト 数】

ニモニック	オペランド	バイト数
BR	near-label	3
	short-label	2
	regptr16	2
	memptr16	2-4
	far-label	5
	memptr32	2-4

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
BR	near-label	1	1	1	0	1	0	0	1	disp-low							
		(disp-high)								—							
	short-label	1	1	1	0	1	0	1	1	disp8							
	regptr16	1	1	1	1	1	1	1	1	1	1	1	0	0	reg		
	memptr16	1	1	1	1	1	1	1	1	mod		1	0	0	mem		
		(disp-low)								(disp-high)							
	far-label	1	1	1	0	1	0	1	0	offset-low							
		offset-high								seg-low							
		seg-high								—							
	memptr32	1	1	1	1	1	1	1	1	mod		1	0	1	mem		
		(disp-low)								(disp-high)							

保守／廃止

BRK

ソフトウエア・トラップ
Break

【命 令 形 式】　BRK target

【オペランド，オペレーション】

ニモニック	オペランド（target）	オペレーション
BRK	3	TA←（00DH, 00CH） TC←（00FH, 00EH） SP←SP－2,（SP＋1, SP）←PSW IE←0, BRK←0 SP←SP－2,（SP＋1, SP）←PS PS←TC SP←SP－2,（SP＋1, SP）←PC PC←TA
	imm8（≠3）	TA←（imm8×4＋1, imm8×4） TC←（imm8×4＋3, imm8×4＋2） SP←SP－2,（SP＋1, SP）←PSW IE←0, BRK←0 SP←SP－2,（SP＋1, SP）←PS PS←TC SP←SP－2,（SP＋1, SP）←PC PC←TA

【フ ラ グ】

V	IE	BRK	S	Z	AC	P	CY
	0	0					

【説　　明】　PSW, PS, PCの値をスタックに退避し，IEフラグとBRKフラグをリセット（0）します。
続いて，target = 3の場合は割り込みベクタ・テーブルのベクタ３の下位２バイトをPCに，上位２バイトをPSにロードします。
target = imm8の場合は８ビット・イミーディエト・データで指定される割り込みベクタ・テーブル（４ビット）の下位２バイトをPCに，上位２バイトをPSにロードします。

【記 述 例】　●BRK 3
●BRK 5

【バ イ ト 数】

ニモニック	オペランド	バイト数
BRK	3	1
	imm8	2

保守／廃止

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
BRK	3	1	1	0	0	1	1	0	0	—							
	imm8	1	1	0	0	1	1	0	1	imm8							

保守／廃止

BRKCS 【V20, V30に対して追加】

レジスタ・バンク切り替え
Break Context Switch

【命 令 形 式】　BRKCS reg16

【オペレーション】　RB2-0←reg16の下位３ビット
PSW退避←PSW, PC退避←PC
IE←0, BRK←0
PC←ベクタPC

【オ ペ ラ ン ド】

ニモニック	オペランド
BRKCS	reg16

【フ　ラ　グ】

RB2	RB1	RB0	V	DIR	IE	BRK	S	Z	F1	AC	F0	P	$\overline{\text{IBRK}}$	CY
×	×	×			0	0								

【説　　明】　レジスタ・バンクを切り替える命令です。高速なサブルーチン・コールなどに使用します。切り替え後のレジスタ・バンクからの復帰には，RETRBI命令を使用します。その際，FINT命令の実行は必要ありません。

【記　述　例】　BRKCS AW

【バ イ ト 数】　３

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
BRKCS	reg16	0	0	0	0	1	1	1	1	0	0	1	0	1	1	0	1
		1	1	0	0	0	reg			−							

保守／廃止

BRKV

オーバフロー例外
Break if Overflow

【命 令 形 式】　BRKV

【オペレーション】　V = 1のとき　TA←（011H, 010H）
TC←（013H, 012H）
SP←SP－2,（SP＋1, SP）←PSW
IE←0, BRK←0
SP←SP－2,（SP＋1, SP）←PS
PS←TC
SP←SP－2,（SP＋1, SP）←PC
PC←TA

【オ ペ ラ ン ド】

ニモニック	オペランド
BRKV	な　し

【フ　ラ　グ】

V	IE	BRK	S	Z	AC	P	CY
	0	0					

【説　　明】　Vフラグがセット（１）されていれば，PSW, PS, PCの値をスタックに退避し，IEフラグとBRKフラグをリセット（０）します。
続いて，target = 3の場合は割り込みベクタ・テーブルのベクタ４の下位２バイトをPCに，上位２バイトをPSにロードします。
Vフラグがリセット（０）されていれば次の命令に進みます。

【記　述　例】　BRKV

【バ イ ト 数】　１

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード							
		7	6	5	4	3	2	1	0
BRKV	な　し	1	1	0	0	1	1	1	0

保守／廃止

BTCLR 【V20, V30に対して追加】

条件付きブランチ
Branch if True and Clear

【命 令 形 式】 BTCLR sfr, imm3, short-label

【オペレーション】 （sfr）のビットNo.imm3 = 1のとき
PC←PC＋ext-disp8
（sfr）のビットNo.imm3←0

【オ ペ ラ ン ド】

ニモニック	オペランド
BTCLR	sfr, imm3, short-label

【フ ラ グ】

V	S	Z	AC	P	CY

【説 明】 指定された特殊機能レジスタのビットが１のとき，そのビットをクリアして現在のPCの値に８ビット・ディスプレースメント（実際にはサイン拡張された16ビット）を加えた値をPCにロードします。
この命令の置かれているセグメント内でかつ，－128～＋127バイトのアドレス内にブランチできます。

【記 述 例】 BTCLR　EXIC0, 7, 45

【バ イ ト 数】 5

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
BTCLR	sfr, imm3, short-label	0	0	0	0	1	1	1	1	1	0	0	1	1	1	0	0
		sfr								imm3							
		disp8								－							

保守／廃止

BUSLOCK

バス・ロック・プリフィクス
Bus Lock Prefix

【命 令 形 式】　BUSLOCK

【オペレーション】　Bus Lock Prefix

【オ ペ ラ ン ド】

ニモニック	オペランド
BUSLOCK	な　し

【フ　ラ　グ】

V	S	Z	AC	P	CY

【説　　明】　この命令に続く１命令を実行している間，ホールド要求が禁止されます。したがって，上述のブロック処理等の途中でホールド要求を受け付けたくない場合，この命令を用いると有効です。

注意１．この命令はPOLL命令の直前に置かないでください。
２．この命令と次の命令の間では，ハードウエア割り込み（マスカブル割り込み，ノンマスカブル割り込み）要求およびシングルステップ・ブレークは受け付けられません。

【記　述　例】　BUSLOCK　REP MOVBKB

【バ イ ト 数】　１

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード							
		7	6	5	4	3	2	1	0
BUSLOCK	な　し	1	1	1	1	0	0	0	0

保守／廃止

BV

V = 1による条件分岐
Branch if Overflow

【命 令 形 式】 BV short-label

【オペレーション】 V = 1のとき：PC←PC＋ext-disp8

【オ ペ ラ ン ド】

ニモニック	オペランド
BV	short-label

【フ　ラ　グ】

V	S	Z	AC	P	CY

【説　　明】 Vフラグが１のとき，現在のPCの値に８ビット・ディスプレースメント（実際にはサイン拡張された16ビット）を加えた値をロードします。
この命令の置かれているセグメント内でかつ，－128～＋127バイトのアドレス内にブランチできます。

【記　述　例】

```
LP2：
        ⋮
        SHL   AL, 1
        BV    SHORT LP2
```

【バ イ ト 数】 ２

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
BV	short-label	0	1	1	1	0	0	0	0	disp8							

保守／廃止

CALL

サブルーチン・コール
Call

【命 令 形 式】　CALL target

【オペランド，オペレーション】

ニモニック	オペランド（target）	オペレーション
CALL	near-proc	SP←SP－2 （SP＋1, SP）←PC PC←PC＋disp
	regptr16	SP←SP－2 （SP＋1, SP）←PC PC←regptr16
	memptr16	TA←（memptr16＋1, memptr16） SP←SP－2 （SP＋1, SP）←PC PC←TA
	far-proc	SP←SP－2 （SP＋1, SP）←PS PS←seg SP←SP－2 （SP＋1, SP）←PC PS←offset
	memptr32	TA←（memptr32＋1, memptr32） TB←（memptr32＋3, memptr32＋2） SP←SP－2 （SP＋1, SP）←PS PS←TB SP←SP－2 （SP＋1, SP）←PC PC←TA

【フ ラ グ】

V	S	Z	AC	P	CY

【説　　明】　○target = near-procまたはtarget = regptr16のとき

PCの値がスタックに退避されたあと，ターゲット・オペランド（target）の次の内容がPCに転送されます。

target = near-procのとき　：16ビット相対アドレス

target = regptr16のとき　：16ビット・レジスタの値（オフセット）

○target = memptr16のとき

PCの値がスタックに退避されたあと，ターゲット・オペランド（target）でアドレスされる16ビット・メモリの内容（オフセット）がPCに転送されます。

この命令の置かれているセグメント内の任意のアドレスを呼ぶことが可能です。

保守／廃止

○target = far-procのとき

PCとPSの値がスタックに退避されたあと，PCには命令の２，３バイト目が，PSには命令の４，５バイト目が転送されます。

この命令によって任意のセグメントの任意のアドレスを呼ぶことが可能です。

○target = memptr32のとき

PCとPSの値がスタックに退避されたあと，ターゲット・オペランド（target）でアドレスされる32ビット・メモリの上位２バイトがPSに，下位２バイトがPCに転送されます。

この命令によって任意のセグメントの任意のアドレスを呼ぶことが可能です。

【記　述　例】　●CALL ＄＋10

●CALL SUB1 ；SUB1はレーベル

【バ イ ト 数】

ニモニック	オペランド	バイト数
CALL	near-proc	3
	regptr16	2
	memptr16	2-4
	far-proc	5
	memptr32	2-4

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
CALL	near-proc	1	1	1	0	1	0	0	0	disp-low							
		disp-high								—							
	regptr16	1	1	1	1	1	1	1	1	1	1	0	1	0	reg		
	memptr16	1	1	1	1	1	1	1	1	mod		0	1	0	mem		
		(disp-low)								(disp-high)							
	far-proc	1	0	0	1	1	0	1	0	offset-low							
		offset-high								seg-low							
		seg-high								—							
	memptr32	1	1	1	1	1	1	1	1	mod		0	1	1	mem		
		(disp-low)								(disp-high)							

保守／廃止

CHKIND

インデクス値チェック
Check Index

【命 令 形 式】　CHKIND reg16, mem32

【オペレーション】　（mem32）＞reg16または（mem32＋2）＜reg16のとき

TA←（015H, 014H）

TC←（017H, 016H）

SP←SP－2,（SP＋1, SP）←PSW

IE←0, BRK←0

SP←SP－2,（SP＋1, SP）←PS

PS←TC

SP←SP－2,（SP＋1, SP）←PC

PC←TA

【オ ペ ラ ン ド】

ニモニック	オペランド
CHKIND	reg16, mem32

【フ　ラ　グ】　割り込み条件成立のとき

V	IE	BRK	S	Z	AC	P	CY
						0	0

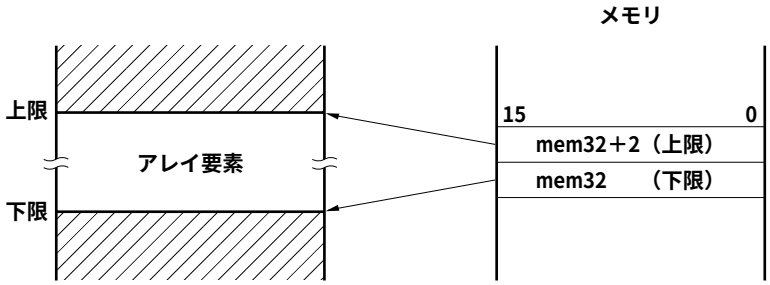
割り込み条件不成立のとき

V	IE	BRK	S	Z	AC	P	CY

保守／廃止

【説　　明】　アレイ・タイプのデータ構造において，要素を指定するインデクス値が定義域内にあるか否かをチェックするための命令です。インデクスが定義域を越える場合には，BRK 5命令を起動します。定義域値は，あらかじめメモリ中の２ワード（１ワード目は下限値，２ワード目は上限値）にセットしておきます。

インデクス値は，アレイ操作プログラムが使用しているレジスタ（任意の16ビット・レジスタ）を対象とします。



【記　述　例】　CHKIND　AW, DWORD_VAR

【バ　イ　ト　数】　2-4

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード													
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
CHKIND	reg16, mem32	0	1	1	0	0	0	1	0	mod		reg			mem		
		(disp-low)								(disp-high)							

保守／廃止

CLR1

ビットのリセット
Clear Bit

【命 令 形 式】　① CLR1 dst, src
② CLR1 dst

【オペレーション】　命令形式①：dstのビットn（nはsrcで指定）←0
命令形式②：dst←0

【オ ペ ラ ン ド】　命令形式①

ニモニック	オペランド（dst, src）
CLR1	reg8, CL
	mem8, CL
	reg16, CL
	mem16, CL
	reg8, imm3
	mem8, imm3
	reg16, imm4
	mem16, imm4

命令形式②

ニモニック	オペランド（dst）
CLR1	CY
	DIR

【フ ラ グ】　命令形式①

V	S	Z	AC	P	CY

命令形式②（dst = CYの場合）

V	S	Z	AC	P	CY
					0

命令形式②（dst = DIRの場合）

V	DIR	S	Z	AC	P	CY
	0					

【説　　明】　命令形式①：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）のビットn（nは第２オペランドで指定されるソース・オペランド（src）の内容）をリセット（０）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。
オペランドがreg8, CLまたはmem8, CLのとき，CLの値は下位３ビット（0-7）のみ有効です。
オペランドがreg16, CLまたはmem16, CLのとき，CLの値は下位４ビット（0-15）のみ有効です。
オペランドがreg8, imm3のとき，命令の４バイト目のイミーディエト・データは下位３ビットのみ有効です。
オペランドがmem8, imm3のとき，命令の最終バイトのイミーディエト・データは下位３ビットのみ有効です。

オペランドがreg16, imm4のとき，命令の４バイト目のイミーディエト・データは下位４ビットのみ有効です。
オペランドがmem16, imm4のとき，命令の最終バイトのイミーディエト・データは下位４ビットのみ有効です。

命令形式②：dst = CYの場合，CYフラグをリセット（0）します。
dst = DIRの場合，DIRフラグをリセット（0）します。また，MOVBK, CMPBK, CMPM, LDM, STM, INM, OUTMの各命令の実行時にインデクス・レジスタ（IX, IY）をオートインクリメントするように設定します。

【記　述　例】

```
CLR1  CY
SHL   AL, 1
BC    $+6
```

【バ　イ　ト　数】

ニモニック	オペランド	バイト数
CLR1	reg8, CL	3
	mem8, CL	3-5
	reg16, CL	3
	mem16, CL	3-5
	reg8, imm3	4
	mem8, imm3	4-6
	reg16, imm4	4
	mem16, imm4	4-6
	CY	1
	DIR	1

保守／廃止

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
CLR1	reg8, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	0
		1	1	0	0	0	reg			—							
	mem8, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	0
		mod		0	0	0	mem			(disp-low)							
		(disp-high)								—							
	reg16, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1
		1	1	0	0	0	reg			—							
	mem16, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1
		mod		0	0	0	mem			(disp-low)							
		(disp-high)								—							
	reg8, imm3	0	0	0	0	1	1	1	1	0	0	0	1	1	0	1	0
		1	1	0	0	0	reg			imm3							
	mem8, imm3	0	0	0	0	1	1	1	1	0	0	0	1	1	0	1	0
		mod		0	0	0	mem			(disp-low)							
		(disp-high)								imm3							
	reg16, imm4	0	0	0	0	1	1	1	1	0	0	0	1	1	0	1	1
		1	1	0	0	0	reg			imm4							
	mem16, imm4	0	0	0	0	1	1	1	1	0	0	0	1	1	0	1	1
		mod		0	0	0	mem			(disp-low)							
		(disp-high)								imm4							
	CY	1	1	1	1	1	0	0	0	—							
	DIR	1	1	1	1	1	1	0	0	—							

保守／廃止

CMP

比較
Compare

【命 令 形 式】　CMP dst, src

【オペランド，オペレーション】

ニモニック	オペランド（dst, src）	オペレーション
CMP	reg, reg'	dst－src
	mem, reg	
	reg, mem	
	reg, imm	
	mem, imm	
	acc, imm	［W = 0のとき］AL－imm8 ［W = 1のとき］AW－imm16

【フ　ラ　グ】

V	S	Z	AC	P	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，フラグを変化させます。

【記　述　例】　●CMP　BL, BYTE PTR［IX］
●CMP　CW, ［BP＋4］

【バ イ ト 数】

ニモニック	オペランド	バイト数
CMP	reg, reg'	2
	mem, reg	2-4
	reg, mem	
	reg, imm	3, 4
	mem, imm	3-6
	acc, imm	2, 3

保守／廃止

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
CMP	reg, reg'	0	0	1	1	1	0	1	W	1	1	reg			reg'		
	mem, reg	0	0	1	1	1	0	0	W	mod		reg			mem		
		(disp-low)								(disp-high)							
	reg, mem	0	0	1	1	1	0	1	W	mod		reg			mem		
		(disp-low)								(disp-high)							
	reg, imm	1	0	0	0	0	0	s	W	1	1	1	1	1	reg		
		imm8 or imm16-low								imm16-high							
	mem, imm	1	0	0	0	0	0	s	W	mod		1	1	1	mem		
		(disp-low)								(disp-high)							
		imm8 or imm16-low								imm16-high							
	acc, imm	0	0	1	1	1	1	0	W	imm8 or imm16-low							
		imm16-high								—							

保守／廃止

CMP4S

10進比較
Compare Nibble String

【命 令 形 式】　CMP4S［DS1-spec：］dst-string,［Seg-spec：］src-string
CMP4S

【オペレーション】　BCDストリング（IY, CL）－BCDストリング（IX, CL）

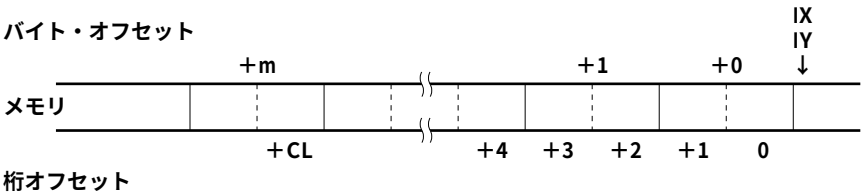
【オ ペ ラ ン ド】

ニモニック	オペランド（dst, src）
CMP4S	［DS1-spec：］dst-string,［Seg-spec：］src-string
	な し

【フ ラ グ】

V	S	Z	AC	P	CY
U	U	×	U	U	×

【説　　明】　IYレジスタでアドレスされるパックトBCDストリングからIXレジスタでアドレスされるパックトBCDストリングを減算し，結果はストアされずフラグのみ影響を受けます。ストリング長（BCD桁数）は，CLレジスタ（CLの内容がdならばd桁）によって決定され，1-254桁まで可能です。
デスティネーション・ストリングは，常にDS1レジスタで指されるセグメント内にロケートされる必要があり，セグメント・オーバライドはできません。一方，ソース・ストリングは，ディフォールト・セグメント・レジスタはDS0レジスタとなっていますが，セグメント・オーバライド可能で，任意のセグメント・レジスタで指定されるセグメント内にロケートできます。パックトBCDストリングのフォーマットを次に示します。



注意　BCDストリング命令は常に偶数桁単位で動作します。このため桁数として偶数を指定したときは演算結果，および各フラグは正しく動きますが，桁数として奇数を指定した場合は奇数＋１の偶数桁演算を実行し，演算結果，および各フラグは偶数桁の演算結果を示しますので，桁数として奇数を指定する場合は次のようにしてください。
桁数が奇数のときのBCD比較命令は，最上位バイトの上位４ビットを“０”にしてから実行してください。

保守／廃止

【記　述　例】　MOV　IX, OFFSET VAR_1
MOV　IY, OFFSET VAR_2
MOV　CL, 4
CMP4S

【バ イ ト 数】　2

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
CMP4S	［DS1-spec：］dst-string, ［Seg-spec：］src-string	0	0	0	0	1	1	1	1	0	0	1	0	0	1	1	0
	な　し																

保守／廃止

CMPBK
CMPBKB
CMPBKW

ブロック比較
Compare Block
Compare Block Byte
Compare Block Word

【命 令 形 式】　(repeat) CMPBK［Seg-spec：］src-block,［DS1-spec：］dst-block
(repeat) CMPBKB
(repeat) CMPBKW

【オペレーション】　［W = 0のとき］（IX）－（IY）
DIR = 0：IX←IX＋1, IY←IY＋1
DIR = 1：IX←IX－1, IY←IY－1
［W = 1のとき］（IX＋1, IX）－（IY＋1, IY）
DIR = 0：IX←IX＋2, IY←IY＋2
DIR = 1：IX←IX－2, IY←IY－2

【オ ペ ラ ン ド】

ニモニック	オペランド
CMPBK	［Seg-spec：］src-block,［DS1-spec：］dst-block
CMPBKB	な　し
CMPBKW	

【フ　ラ　グ】

V	S	Z	AC	P	CY
×	×	×	×	×	×

【説　　明】　IXレジスタでアドレスされるブロックからIYレジスタでアドレスされるブロックをバイトまたはワード・データ単位に繰り返し減算を行い，結果をフラグに反映させます。
IXレジスタ，IYレジスタは，次のバイト／ワード処理のために１バイト／ワード・データが処理されるごとに自動的にインクリメント（＋１/＋２）またはデクリメント（－１/－２）されます。ブロックの方向は，DIRフラグの状態によって決定されます。
バイトかワードかの指定は，CMPBK命令を用いる場合はオペランドの属性によって行われ，CMPBKB命令またはCMPBKW命令を用いる場合はそれぞれバイト，ワード・タイプに直接指定されます。
デスティネーション・ブロックは常にDS1レジスタで指定されるセグメント内にロケートされる必要があり，セグメント・オーバライドはできません。一方，ソース・ブロックは，ディフォールト・セグメント・レジスタはDS0レジスタとなっていますが，セグメント・オーバライド可能で，任意のセグメント・レジスタで指定されるセグメント内にロケートできます。

保守／廃止

【記　述　例】　CMPBK BYTE_VAR1, BYTE_VAR2

【バ イ ト 数】　1

【命 令 語 形 式】

ニモニック	オペランド（dst, src）	オペレーション・コード							
		7	6	5	4	3	2	1	0
CMPBK	［Seg-spec：］src-block, ［DS1-spec：］dst-block	1	0	1	0	0	1	1	W
CMPBKB	な　し								
CMPBKW									

保守／廃止

CMPM
CMPMB
CMPMW

アキュームレータとのブロック比較
Compare Multiple
Compare Multiple Byte
Compare Multiple Word

【命 令 形 式】　(repeat) CMPM［DS1-spec：］dst-block
(repeat) CMPMB
(repeat) CMPMW

【オペレーション】　［W = 0のとき］AL－（IY）
DIR = 0：IY←IY＋1,
DIR = 1：IY←IY－1
［W = 1のとき］AW－（IY＋1, IY）
DIR = 0：IY←IY＋2
DIR = 1：IY←IY－2

【オ ペ ラ ン ド】

ニモニック	オペランド
CMPM	［DS1-spec：］dst-block
CMPMB	な　し
CMPMW	

【フ　ラ　グ】

V	S	Z	AC	P	CY
×	×	×	×	×	×

【説　　明】　アキュームレータ（AL/AW）の値からIYレジスタでアドレスされるブロックをバイトまたはワード・データ単位に繰り返し減算を行い，結果をフラグに反映させます。IYレジスタは，次のバイト／ワード処理のために１バイト／ワード・データが処理されるごとに自動的にインクリメント（＋１/＋２）またはデクリメント（－１/－２）されます。ブロックの方向は，DIRフラグの状態によって決定されます。
バイトかワードかの指定は，CMPM命令を用いる場合はオペランドの属性によって行われ，CMPMB命令またはCMPMW命令を用いる場合はそれぞれバイト，ワード・タイプに直接指定されます。
デスティネーション・ブロックは，常にDS1レジスタで指定されるセグメント内にロケートされる必要があり，セグメント・オーバライドはできません。

保守／廃止

【記　述　例】　●MOV　AW, 5555H
　　MOV　BW, 1000H
　　MOV　IY, BW
　　REPC　CMPM WORD PTR［IY］
　　●REPNC CMPMW
　　●REPZ　CMPMB

【バ イ ト 数】　1

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード							
		7	6	5	4	3	2	1	0
CMPM	［DS1-spec：］dst-block	1	0	1	0	1	1	1	W
CMPMB	な　し								
CMPMW									

保守／廃止

CVTBD

２進→アンパックト10進変換
Convert Binary to Decimal

【命 令 形 式】　CVTBD

【オペレーション】　AH←AL÷0AH
AL←AL%0AH

【オ ペ ラ ン ド】

ニモニック	オペランド
CVTBD	な　し

【フ　ラ　グ】

V	S	Z	AC	P	CY
U	×	×	U	×	U

【説　　明】　ALレジスタの２進数８ビットを２桁のアンパックト10進数に変換します。
AHレジスタはALレジスタの値を10で割った商で置き換えられ，次にALレジスタがその乗算の剰余で置き換えられます。

【記　述　例】　MOV AL, 30H
CVTBD

【バ イ ト 数】　２

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
CVTBD	な　し	1	1	0	1	0	1	0	0	0	0	0	0	1	0	1	0

保守／廃止

CVTBW

ワード符号拡張
Convert Byte to Word

【命 令 形 式】　CVTBW

【オペレーション】　AL＜80Hのとき：AH←0
AL≧80Hのとき：AH←FFH

【オ ペ ラ ン ド】

ニモニック	オペランド
CVTBW	な　し

【フ　ラ　グ】

V	S	Z	AC	P	CY

【説　　明】　ALレジスタ内のバイトの符号をAHレジスタに拡張します。バイト除算を実行する前に，あるバイトから倍長（ワード）の被除数を得るのに有効です。

【記　述　例】
```
MOV     AL, BUF1 ; BUF1はバイト変数
CVTBW
MOV     DL, 60
DIV     DL
```

【バ イ ト 数】　1

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード							
		7	6	5	4	3	2	1	0
CVTBW	な　し	1	0	0	1	1	0	0	0

保守／廃止

CVTDB

アンパックト10進→２進変換
Convert Decimal to Binary

【命 令 形 式】　CVTDB

【オペレーション】　AL←AH×0AH＋AL
AH←0

【オ ペ ラ ン ド】

ニモニック	オペランド
CVTDB	な　し

【フ　ラ　グ】

V	S	Z	AC	P	CY
U	×	×	U	×	U

【説　　明】　AHレジスタおよびALレジスタの２桁のアンパックト10進数を16ビットの２進数に変換します。
ALレジスタはAHレジスタの値に10を掛けた結果にALレジスタの値を加えたもので置き換えられ，AHレジスタは０で置き換えられます。

【記　述　例】　MOV AW,［BW］
CVTDB

【バ イ ト 数】　２

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
CVTDB	な　し	1	1	0	1	0	1	0	1	0	0	0	0	1	0	1	0

保守／廃止

CVTWL

ダブル・ワード符号拡張
Convert Word to Long Word

【命 令 形 式】　CVTWL

【オペレーション】　AW＜8000Hのとき：DW←0
AW≧8000Hのとき：DW←FFFFH

【オ ペ ラ ン ド】

ニモニック	オペランド
CVTWL	な　し

【フ　ラ　グ】

V	S	Z	AC	P	CY

【説　　明】　AWレジスタのワードの符号をDWレジスタに拡張します。ワード除算を実行する前に，あるワードから倍長（ダブル・ワード）の被除数を得るのに有効です。

【記　述　例】
```
MOV     AW, BUFFER
CVTWL
DIV     CW
```

【バ イ ト 数】　1

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード							
		7	6	5	4	3	2	1	0
CVTWL	な　し	1	0	0	1	1	0	0	1

保守／廃止

DBNZ

CW≠0による条件ループ
Decrement and Branch if Not Zero

【命 令 形 式】　DBNZ short-label

【オペレーション】　CW←CW－1
CW≠0のとき：PC←PC＋ext-disp8

【オ ペ ラ ン ド】

ニモニック	オペランド
DBNZ	short-label

【フ　ラ　グ】

V	S	Z	AC	P	CY

【説　　明】　CWレジスタの値をデクリメント（－１）し，その結果CWレジスタの値が０でなければ，現在のPCの値に８ビット・ディスプレースメント（実際にはサイン拡張された16ビット）を加えた値をロードします。
この命令に置かれているセグメント内でかつ，－128～＋127バイトのアドレス内にブランチできます。
条件不成立のときは次の命令に進みます。

【記　述　例】　LP21：
　　⋮
　　SHL　AL, 1
　　DBNZ　LP21；LP21 = レーベル

【バ イ ト 数】　２

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード													
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
DBNZ	short-label	1	1	1	0	0	0	1	0	disp8							

保守／廃止

DBNZE

CW≠0かつZ = 1による条件ループ
Decrement and Branch if Not Zero and Equal

【命 令 形 式】　DBNZE short-label

【オペレーション】　CW←CW－1
CW≠0でZ = 1のとき：PC←PC＋ext-disp8

【オ ペ ラ ン ド】

ニモニック	オペランド
DBNZE	short-label

【フ　ラ　グ】

V	S	Z	AC	P	CY

【説　　明】　CWレジスタの値をデクリメント（－１）し，その結果CWレジスタの値が０でなく，かつZフラグがセット（１）されていれば，現在のPCの値に８ビット・ディスプレースメント（実際にはサイン拡張された16ビット）を加えた値をロードします。この命令の置かれているセグメント内でかつ，－128～＋127バイトのアドレス内にブランチできます。
条件不成立のときは次の命令に進みます。

【記　述　例】
```
LP20：
        ⋮
        AND     AL, BL
        DBNZE   LP20   ; LP20 = レーベル
```

【バ イ ト 数】　２

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
DBNZE	short-label	1	1	1	0	0	0	0	1	disp8							

保守／廃止

DBNZNE

CW≠0かつZ = 0による条件ループ
Decrement and Branch if Not Zero and Not Equal

【命 令 形 式】　DBNZNE short-label

【オペレーション】　CW←CW－1
CW≠0でZ = 0のとき：PC←PC＋ext-disp8

【オ ペ ラ ン ド】

ニモニック	オペランド
DBNZNE	short-label

【フ　ラ　グ】

V	S	Z	AC	P	CY

【説　　明】　CWレジスタの値をデクリメント（－１）したあと，CWレジスタの値が０でなくZフラグがクリアされていれば，現在のPCの値に８ビット・ディスプレースメント（実際にはサイン拡張された16ビット）を加えた値をロードします。
この命令の置かれているセグメント内でかつ，－128～＋127バイトのアドレス内にブランチできます。
条件不成立のときは次の命令に進みます。

【記　述　例】　LP19：
　　⋮
　　AND　　AL, 0FFH
　　DBNZNE　SHORT LP19；LP19 = レーベル

【バ イ ト 数】　２

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
DBNZNE	short-label	1	1	1	0	0	0	0	0	disp8							

保守／廃止

DEC

デクリメント
Decrement

【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
DEC	reg8
	mem
	reg16

【フ　ラ　グ】

V	S	Z	AC	P	CY
×	×	×	×	×	

【説　　明】　デスティネーション・オペランド（dst）の内容をデクリメント（－１）します。

【記　述　例】
●DEC BW
●DEC BP
●DEC IX
●DEC IY

【バ イ ト 数】

ニモニック	オペランド	バイト数
DEC	reg8	2
	mem	2-4
	reg16	1

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
DEC	reg8	1	1	1	1	1	1	1	0	1	1	0	0	1	reg		
	mem	1	1	1	1	1	1	1	W	mod		0	0	1	mem		
		(disp-low)								(disp-high)							
	reg16	0	1	0	0	1	reg			－							

保守／廃止

DI

マスカブル割り込みの禁止
Disable Interrupt

【命 令 形 式】　DI

【オペレーション】　IE←0

【オ ペ ラ ン ド】

ニモニック	オペランド
DI	な　し

【フ　ラ　グ】

V	IE	S	Z	AC	P	CY
	0					

【説　　明】　IEフラグをリセット（0）し，マスカブル割り込み要求を禁止します。
この命令によってノンマスカブル割り込み要求およびソフトウエア割り込み要求は禁止されません。

【記　述　例】
```
DI
PUSH R
```

【バ イ ト 数】　1

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード							
		7	6	5	4	3	2	1	0
DI	な　し	1	1	1	1	1	0	1	0

保守／廃止

DISPOSE

スタック・フレームの削除
Dispose a Stack Frame

【命 令 形 式】　DISPOSE

【オペレーション】　SP←BP
BP←（SP＋1, SP）
SP←SP＋2

【オ ペ ラ ン ド】

ニモニック	オペランド
DISPOSE	な　し

【フ　ラ　グ】

V	S	Z	AC	P	CY

【説　　明】　この命令は，PREPARE命令で生成されたスタック・フレームを１フレーム分リリースします。
BPには，１つ前のフレームを指すポインタ値をロードし，SPには，フレームの最下位を示すポインタ値をロードします。

【記　述　例】　DISPOSE

【バ イ ト 数】　１

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード							
		7	6	5	4	3	2	1	0
DISPOSE	な　し	1	1	0	0	1	0	0	1

保守／廃止

DIV

符号付き除算
Divide Signed

【命 令 形 式】　DIV dst

【オペランド，オペレーション】

ニモニック	オペランド（dst）	オペレーション
DIV	reg8	temp←AW temp÷reg8＞0でtemp÷reg8≦7FHまたは temp÷reg8＜0でtemp÷reg8＞0－7FH－1のとき 　AH←temp%dst 　AL←temp÷dst temp÷reg8＞0でtemp÷reg8＞7FHまたは temp÷reg8＜0でtemp÷reg8≦0－7FH－1のとき 　商と剰余は不定
	mem8	TA←（001H, 000H） 　TC←（003H, 002H） 　SP←SP－2,（SP＋1, SP）←PSW 　IE←0, BRK←0 　SP←SP－2,（SP＋1, SP）←PS 　PS←TC 　SP←SP－2,（SP＋1, SP）←PC 　PC←TA
	reg16	temp←DW, AW temp÷dst＞0でtemp÷dst≦7FFFHまたは temp÷dst＜0でtemp÷dst＞0－7FFFH－1のとき 　DW←temp%dst 　AW←temp÷dst temp÷dst＞0でtemp÷dst＞7FFFHまたは temp÷dst＜0でtemp÷dst≦0－7FFFH－1のとき 　商と剰余は不定
	mem16	TA←（001H, 000H） 　TC←（003H, 002H） 　SP←SP－2,（SP＋1, SP）←PSW 　IE←0, BRK←0 　SP←SP－2,（SP＋1, SP）←PS 　PS←TC 　SP←SP－2,（SP＋1, SP）←PC 　PC←TA

保守／廃止

【フ　ラ　グ】

V	S	Z	AC	P	CY
U	U	U	U	U	U

【説　　明】　○src = reg8またはsrc = mem8のとき

AWレジスタの値をデスティネーション・オペランド（dst）の内容で符号付き除算します。商はALレジスタに格納し，余りはAHレジスタに格納します。

正の商の最大値は＋127（7FH）で，負の商の最小値は－127（81H）です。商が正で，最大値を越えたとき，または商が負で最小値より小さくなったときは，ベクタ０割り込みが発生し，商と余りは不定となります（特にsrc = 00Hのときに発生します）。また，整数でない商は整数に切り縮められ，余りは被除数と同じ符号を持ちます。

○src = reg16またはsrc = mem16のとき

AWレジスタの値とDWレジスタの値をデスティネーション・オペランド（dst）の内容で符号付き除算します。商はAWレジスタに格納し，余りはDWレジスタに格納します。

正の商の最大値は＋32767（7FFFH）で，負の商の最小値は－32767（8001H）です。商が正で，最大値を越えたとき，または商が負で最小値より小さくなったときは，ベクタ０割り込みが発生し，商と余りは不定となります（特にsrc = 0000Hのときに発生します）。また，整数でない商は整数に切り縮められ，余りは被除数と同じ符号を持ちます。

【記　述　例】　32ビット・データDW：AWをメモリ０：50の内容で割る。

```
MOV   BW, 0
MOV   DS0, BW
MOV   IX, 50H
DIV   DS0：WORD PTR［IX］
```

【バ イ ト 数】

ニモニック	オペランド	バイト数
DIV	reg8	2
	mem8	2-4
	reg16	2
	mem16	2-4

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
DIV	reg8	1	1	1	1	0	1	1	0	1	1	1	1	1	reg		
	mem8	1	1	1	1	0	1	1	0	mod			1	1	1	mem	
		(disp-low)								(disp-high)							
	reg16	1	1	1	1	0	1	1	1	1	1	1	1	1	reg		
	mem16	1	1	1	1	0	1	1	1	mod			1	1	1	mem	
		(disp-low)								(disp-high)							

保守／廃止

DIVU

符号なし除算
Divide Unsigned

【命 令 形 式】　DIVU dst

【オペランド，オペレーション】

ニモニック	オペランド（dst）	オペレーション
DIVU	reg8	temp←AW temp÷dst≦FFHのとき： AH←temp%dst AL←temp÷dst temp÷dst＞FFHのとき： TA←（001H, 000H） TC←（003H, 002H） SP←SP－2,（SP＋1, SP）←PSW IE←0, BRK←0 SP←SP－2,（SP＋1, SP）←PS RS←TC SP←SP－2,（SP＋1, SP）←PC PC←TA
	mem8	
	reg16	temp←DW, AW temp÷dst≦FFFFHのとき： DW←temp%dst AW←temp÷dst temp÷dst＞FFFFHのとき： TA←（001H, 000H） TC←（003H, 002H） SP←SP－2,（SP＋1, SP）←PSW IE←0, BRK←0 SP←SP－2,（SP＋1, SP）←PS RS←TC SP←SP－2,（SP＋1, SP）←PC PC←TA
	mem16	

【フ　ラ　グ】

V	S	Z	AC	P	CY
U	U	U	U	U	U

保守／廃止

【説　　明】　○src = reg8またはsrc = mem8のとき

AWレジスタの値をデスティネーション・オペランド（dst）の内容で符号なし除算します。商はALレジスタに格納し，余りはAHレジスタに格納します。

商がALレジスタの容量（FFH）を越えた場合は，ベクタ０割り込みが発生し，商と余りは不定となります（特にsrc = 00Hのときに発生します）。また，整数でない商は整数に切り縮められます。

○src = reg16またはsrc = mem16のとき

AWレジスタの値とDWレジスタの値をデスティネーション・オペランド（dst）の内容で符号なし除算します。商はAWレジスタに格納し，余りはDWレジスタに格納します。

商がAWレジスタの容量（FFFFH）を越えた場合，ベクタ０割り込みが発生し，商と余りは不定となります（特にsrc = 0000Hのときに発生します）。また，整数でない商は整数に切り縮められます。

【記　述　例】　５÷３の除算をする。

```
MOV     AW, 5
MOV     DL, 3
DIVU    DL
；AH = 2  AL = 1
```

【バ イ ト 数】

ニモニック	オペランド	バイト数
DIVU	reg8	2
	mem8	2-4
	reg16	2
	mem16	2-4

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
DIVU	reg8	1	1	1	1	0	1	1	0	1	1	1	1	0	reg		
	mem8	1	1	1	1	0	1	1	0	mod		1	1	0	mem		
		(disp-low)								(disp-high)							
	reg16	1	1	1	1	0	1	1	1	1	1	1	1	0	reg		
	mem16	1	1	1	1	0	1	1	1	mod		1	1	0	mem		
		(disp-low)								(disp-high)							

保守／廃止

DS0：
DS1：
PS：
SS：

セグメント・オーバライド・プリフィクス
Data Segment 0
Data Segment 1
Program Segment
Stack Segment

【命 令 形 式】　DS0：
DS1：
PS：
SS：

【オペレーション】　セグメント・オーバライド・プリフィクス

【オ ペ ラ ン ド】

ニモニック	オペランド
DS0：	な　し
DS1：	
PS：	
SS：	

【フ　ラ　グ】

V	S	Z	AC	P	CY

【説　　明】　セグメント・オーバライド可能なメモリ・オペランドをアクセスする際に，オペランドに付記してそのとき使用されるセグメント・レジスタを指定します。この命令を直接記述しなくとも，ASSUME（アセンブラ疑似命令）を使用することによって，セグメント・オーバライド指定をアセンブラに行わせることができます。

注意　この命令と次の命令との間では，ハードウエア割り込み（マスカブル割り込み，ノンマスカブル割り込み）要求およびシングルステップ・ブレークは受け付けられません。

【記　述　例】　MOV　DW, DS1：［BW］；ディフォールトのセグメント・レジスタはDS0

【バ イ ト 数】　1

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード							
		7	6	5	4	3	2	1	0
DS0：	な　し	0	0	1	sreg		1	1	0
DS1：									
PS：									
SS：									

保守／廃止

EI

マスカブル割り込みの許可
Enable Interrupt

【命 令 形 式】　EI

【オペレーション】　IE←1

【オ ペ ラ ン ド】

ニモニック	オペランド
EI	な　し

【フ　ラ　グ】

V	IE	S	Z	AC	P	CY
	1					

【説　　明】　IEフラグをセット（1）し，マスカブル割り込み要求を許可します。
ただし，実際に割り込み許可状態になるのはEI命令に続く１命令を実行したあとです。

【記　述　例】
```
POP　R
EI
```

【バ イ ト 数】　１

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード							
		7	6	5	4	3	2	1	0
EI	な　し	1	1	1	1	1	0	1	1

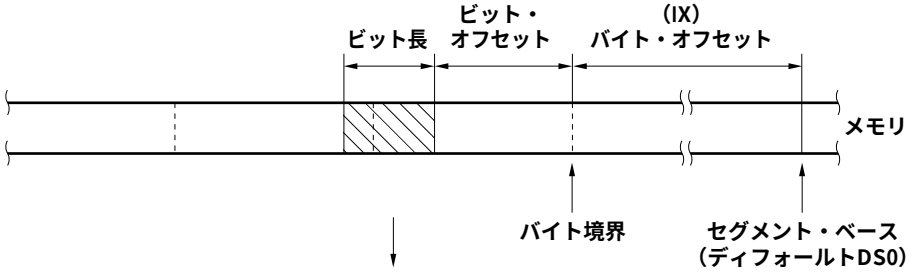
保守／廃止

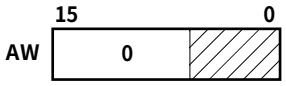
EXT

ビット・フィールドの抽出
Extract Bit Field

【命 令 形 式】　EXT dst, src

【オペレーション】　AW←16ビット・フィールド





【オ ペ ラ ン ド】

ニモニック	オペランド（dst, src）
EXT	reg8, reg8'
	reg8, imm4

【フ　ラ　グ】

V	S	Z	AC	P	CY
U	U	U	U	U	U

【説　　明】　IXレジスタでアドレスされるバイト・オフセットおよび第１オペランドの８ビット・レジスタで指定されるビット・オフセットによって決定されるメモリ領域から，ソース・オペランド（src）によって指定されるビット長のビット・フィールド・データをAWレジスタへロードします。このときAWレジスタの余りの上位ビットには０がロードされます。
転送終了後，IXレジスタおよび第１オペランドで指定される８ビット・レジスタは次のビット・フィールドを示すために自動的に次のように更新されます。

```
reg8←reg8＋src＋1
if reg8＞15 then
        {
    reg8←reg8－16
    IX←IX＋２
    }
```

ビット・オフセット（最長15ビット）を指定する第１オペランドの８ビット・レジスタの値は０-15のみ有効です。また，ビット長（最長16ビット）を指定するソース・オペランド（src）の値は０-15のみが有効で，０が１ビット長を，15が16ビット長を指定します。

ビット・フィールド・データはメモリのバイト境界にまたがることができます。ソースのビット・フィールドはディフォールト・セグメント・レジスタがDS0レジスタとなっていますが，セグメント・オーバライド可能で，任意のセグメント・レジスタで指定されるセグメント内にロケートできます。

注意　reg8またはreg8'の上位４ビットは０にしてください。

【記　述　例】　●EXT　CL, DL

●EXT　CL, 8

【バ　イ　ト　数】

ニモニック	オペランド	バイト数
EXT	reg8, reg8'	3
	reg8, imm4	4

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
EXT	reg8, reg8'	0	0	0	0	1	1	1	1	0	0	1	1	0	0	1	1
		1	1	reg'			reg			—							
	reg8, imm4	0	0	0	0	1	1	1	1	0	0	1	1	1	0	1	1
		1	1	0	0	0	reg			imm4							

保守／廃止

FINT 【V20, V30に対して追加】

割り込み
Finish Interrupt

【命 令 形 式】　FINT

【オペレーション】　CPUに内蔵される割り込みコントローラへ割り込み処理ルーチンが終了したことを示す。

【オ ペ ラ ン ド】

ニモニック	オペランド
FINT	な　し

【フ　ラ　グ】

V	S	Z	AC	P	CY

【説　　明】　ISPRレジスタのセットされているビットのうち最下位（最高優先順位）の１ビットをリセットします。これにより，内蔵する割り込みコントローラに対して，NMIとINT，およびソフトウエア割り込みを除く割り込みが終了したことを知らせます。NMIとINT，およびソフトウエア割り込みを除く割り込み処理プログラムが終了する直前（RETI命令またはRETRBI命令の直前）に使用してください。それ以外では使用しないでください。

【記　述　例】　FINT

【バ イ ト 数】　２

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
FINT	な　し	0	0	0	0	1	1	1	1	1	0	0	1	0	0	1	0

保守／廃止

FPO1

浮動小数点演算用コプロセッサ制御
Floating Point Operation 1

【命 令 形 式】　① FPO1 fp-op
② FPO1 fp-op, mem

【オペランド，オペレーション】

命令形式①，②

ニモニック	オペランド	オペレーション
FPO1	fp-op	（SP－1, SP－2）←PSW （SP－3, SP－4）←PS （SP－5, SP－6）←PC－x[注]
	fp-op, mem	SP←SP－6 IE←0, BRK←0 PC←（01DH, 01CH） PS←（01FH, 01EH）

注　xは命令のバイト数＋プリフィクスの数

【フ ラ グ】

V	IE	BRK	S	Z	AC	P	CY
	0	0					

【説　　明】　PSW, PS, PC－xをスタックに退避し，IEとBRKフラグをリセット（0）します。次に割り込みベクタ・テーブルのベクタ7の下位2バイトをPCに，上位2バイトをPSにロードします。

この命令はV20, V30との互換性を保つために用意された命令です。V25, V35ファミリとしては特に機能的に意味を持たず，割り込みのみ発生します。

【記 述 例】
●FPO1　010101010B
●FPO1　0FFH
●FPO1　6，BYTE PTR［IX］
●FPO1　4，WORD_VAR

【バ イ ト 数】

ニモニック	オペランド	バイト数
FPO1	fp-op	2
	fp-op, mem	2-4

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
FPO1	fp-op	1	1	0	1	1	X	X	X	1	1	Y	Y	Y	Z	Z	Z
	fp-op, mem	1	1	0	1	1	X	X	X	mod		Y	Y	Y	mem		
		(disp-low)								(disp-high)							

保守／廃止

FPO2

浮動小数点演算用コプロセッサ制御
Floating Point Operation 2

【命 令 形 式】　① FPO2 fp-op
② FPO2 fp-op, mem

【オペランド，オペレーション】

命令形式①，②

ニモニック	オペランド	オペレーション
FPO2	fp-op	(SP－1, SP－2) ←PSW (SP－3, SP－4) ←PS (SP－5, SP－6) ←PC－x[注]
	fp-op, mem	SP←SP－6 IE←0, BRK←0 PC← (01DH, 01CH) PS← (01FH, 01EH)

注　xは命令のバイト数＋プリフィクスの数

【フ ラ グ】

V	IE	BRK	S	Z	AC	P	CY
	0	0					

【説　　明】　PSW, PS, PC－xをスタックに退避し，IEとBRKフラグをリセット（0）します。次に割り込みベクタ・テーブルのベクタ7の下位2バイトをPCに，上位2バイトをPSにロードします。

この命令はV20, V30との互換性を保つために用意された命令です。V25, V35ファミリとしては特に機能的に意味を持たず，割り込みのみ発生します。

【記 述 例】
●FPO2　010101010B
●FPO2　0FFH
●FPO2　0101B, BYTE PTR［IY］
●FPO2　1010B, WORD_VAR

【バ イ ト 数】

ニモニック	オペランド	バイト数
FPO2	fp-op	2
	fp-op, mem	2-4

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
FPO2	fp-op	0	1	1	0	0	1	1	X	1	1	Y	Y	Y	Z	Z	Z
	fp-op, mem	0	1	1	0	0	1	1	X	mod		Y	Y	Y	mem		
		(disp-low)								(disp-high)							

保守／廃止

HALT

ホールト
Halt

【命 令 形 式】　HALT

【オペレーション】　CPU Halt

【オ ペ ラ ン ド】

ニモニック	オペランド
HALT	な　し

【フ　ラ　グ】

V	S	Z	AC	P	CY

【説　　明】　CPUへのクロック供給を停止し，スタンバイ状態にします。
スタンバイ状態は次の要因によって解除されます。

- ●リセット入力
- ●マスカブル割り込み要求入力
- ●ノンマスカブル割り込み要求入力

【記　述　例】　HALT

【バ イ ト 数】　1

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード							
		7	6	5	4	3	2	1	0
HALT	な　し	1	1	1	1	0	1	0	0

保守／廃止

IN

I/Oデバイスからのデータ入力
Input

【命 令 形 式】　IN dst, src

【オペランド，オペレーション】

$\overline{IBRK}$ = 1のとき

ニモニック	オペランド（dst, src）	オペレーション
IN	acc, imm8	［W = 0のとき］AL←（imm8）
		［W = 1のとき］AH←（imm8＋1），AL←（imm8）
	acc, DW	［W = 0のとき］AL←（DW）
		［W = 1のとき］AH←（DW＋1），AL←（DW）

$\overline{IBRK}$ = 0のとき

ニモニック	オペランド（dst, src）	オペレーション
IN	acc, imm8	（SP－1, SP－2）←PSW
		（SP－3, SP－4）←PS
		（SP－5, SP－6）←PC－x[注]
	acc, DW	IE←0, BRK←0, $\overline{IBRK}$←1
		PC←（04DH, 04CH）
		PS←（04FH, 04EH）

注　xは命令のバイト数＋プリフィックスの数

【フ ラ グ】

$\overline{IBRK}$ = 1のとき

V	IE	BRK	S	Z	AC	P	$\overline{IBRK}$	CY

$\overline{IBRK}$ = 0のとき

V	IE	BRK	S	Z	AC	P	$\overline{IBRK}$	CY
	0	0					1	

【説　　明】　$\overline{IBRK}$ = 1のとき，デスティネーション・オペランド（dst）で指定されるアキュームレータ（ALレジスタまたはAWレジスタ）にソース・オペランド（src）で指定されるI/Oデバイスのレジスタ内容を転送します。

$\overline{IBRK}$ = 0のとき，PSW, PS,PC－xをスタックに退避し，IEとBRKフラグをリセット（0）し，$\overline{IBRK}$をセット（1）します。次に割り込みベクタ・テーブルのベクタ19の下位2バイトをPCに，上位2バイトをPSにロードします。

この機能は，ソフトウエアの移植を容易にするために用意された機能です。

【記　述　例】　ポート・アドレス0DAHの内容をALレジスタに転送する。

```
MOV   DW, 0DAH
IN    AL, DW
```

保守／廃止

【バ イ ト 数】

ニモニック	オペランド	バイト数
IN	acc, imm8	2
	acc, DW	1

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード													
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
IN	acc, imm8	1	1	1	0	0	1	0	W	imm8							
	acc, DW	1	1	1	0	1	1	0	W	—							

保守／廃止

INC

インクリメント
Increment

【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	reg8
	mem
	reg16

【フ　ラ　グ】

V	S	Z	AC	P	CY
×	×	×	×	×	

【説　　明】　デスティネーション・オペランド（dst）の内容をインクリメント（＋１）します。

【記　述　例】
●INC　DW
●INC　BP
●INC　SP

【バ イ ト 数】

ニモニック	オペランド	バイト数
INC	reg8	2
	mem	2-4
	reg16	1

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
INC	reg8	1	1	1	1	1	1	1	0	1	1	0	0	0	reg		
	mem	1	1	1	1	1	1	1	W	mod		0	0	0	mem		
		(disp-low)								(disp-high)							
	reg16	0	1	0	0	0	reg			—							

保守／廃止

INM

I/O-メモリのブロック転送
Input Multiple

【命 令 形 式】　（repeat）INM［DS1-spec：］dst-block, DW

【オペレーション】　$\overline{IBRK}$ = 1のとき　［W = 0のとき］（IY）←（DW）
DIR = 0：IY←IY＋1
DIR = 1：IY←IY－1
［W = 1のとき］（IY＋1, IY）←（DW＋1, DW）
DIR = 0：IY←IY＋2
DIR = 1：IY←IY－2

$\overline{IBRK}$ = 0のとき　（SP－1, SP－2）←PSW
（SP－3, SP－4）←PS
（SP←5, SP－6）←PC－x注
IE←0, BRK←0, $\overline{IBRK}$←1
PC←（04DH, 04CH）
PS←（04FH, 04EH）

注　xは命令のバイト数＋プリフィクスの数

【オ ペ ラ ン ド】

ニモニック	オペランド
INM	［DS1-spec：］dst-block, DW

【フ ラ グ】

$\overline{IBRK}$ = 1のとき

V	IE	BRK	S	Z	AC	P	$\overline{IBRK}$	CY

$\overline{IBRK}$ = 0のとき

V	IE	BRK	S	Z	AC	P	$\overline{IBRK}$	CY
	0	0					1	

【説　　明】　$\overline{IBRK}$ = 1のとき，DWレジスタでアドレスされるI/Oデバイスのレジスタ内容を，IYレジスタでアドレスされるメモリに転送します。繰り返し転送回数は，ペアで使用されるリピート・プリフィクスのREP命令によって制御されます。繰り返し転送の際，DWレジスタの内容（I/Oデバイスのアドレス）は固定ですが，IYレジスタは次のバイト／ワード転送のために１バイト／ワード・データが転送されるごとに自動的にインクリメント（＋１/＋２）またはデクリメント（－１/－２）されます。ブロックの方向は，DIRフラグの状態によって決定されます。バイトかワードかの指定は，オペランドの属性によって行われます。
INM命令は，リピート・プリフィクスのREP命令とともに使用されます。
デスティネーション・ブロックは，常にDS1レジスタで指定されるセグメント内にロケートされる必要があり，セグメント・オーバライドはできません。

$\overline{IBRK}$ = 0のとき，PSW, PS, PC－xをスタックに退避し，IEとBRKフラグをリセット（０）

し，$\overline{\text{IBRK}}$をセット（1）します。次に割り込みベクタ・テーブルのベクタ19の下位2バイトをPCに，上位2バイトをPSにロードします。
この機能は，ソフトウエアの移植を容易にするために用意された機能です。

【記　述　例】　●メモリ・ワーク・エリアにポート・アドレス0DAHの内容（バイト・データ）をロードする。

```
MOV   AW, 0
MOV   DS1, AW
MOV   IY, 50H
MOV   DW, 0DAH
INM   DS1：BYTE PTR［IY］, DW
```

●メモリ0：0H～0：FFHにポート・アドレス0DAHの内容（バイト・データ）をロードする。

```
MOV   AW, 0
MOV   DS1, AW
MOV   IY,0
MOV   DW, 0DAH
MOV   CW, 0FFH
REP   INM   DS1：BYTE PTR［IY］, DW
```

【バ　イ　ト　数】　1

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード							
		7	6	5	4	3	2	1	0
INM	［DS1-spec：］dst-block, DW	0	1	1	0	1	1	0	W

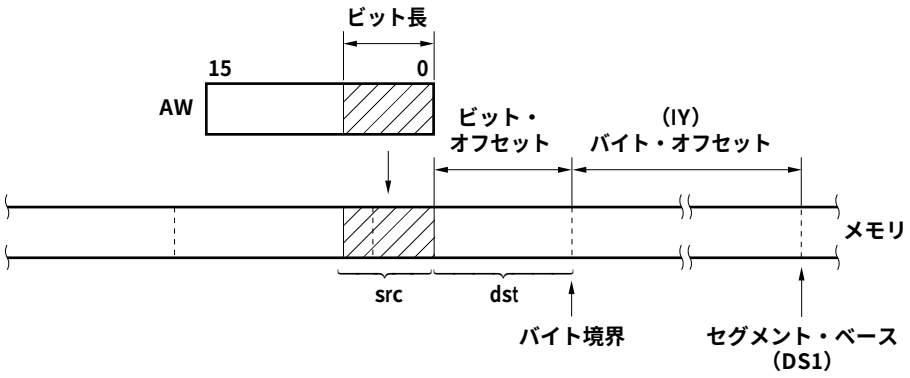
保守／廃止

INS

ビット・フィールドの挿入
Insert Bit Field

【命 令 形 式】　INS dst, src

【オペレーション】　16ビット・フィールド←AW



【オ ペ ラ ン ド】

ニモニック	オペランド（dst, src）
INS	reg8, reg8'
	reg8, imm4

【フ ラ グ】

V	S	Z	AC	P	CY
U	U	U	U	U	U

【説　　明】　AWレジスタの16ビット・データのうち，ソース・オペランド（src）で指定される長さの下位ビット・データをDS1レジスタとIYレジスタでアドレスされるバイト・オフセットおよび第１オペランドの８ビット・レジスタで指定されるビット・オフセットによって決定されるメモリ領域へ転送します。
転送終了後，IYレジスタおよび第１オペランドで指定される８ビット・レジスタは，次のビット・フィールドを示すために自動的に次のように更新されます。

```
reg8←reg8＋src＋1
if reg8＞15 then
        {
    reg8←reg8－16
    IY←IY＋２
    }
```

ビット・オフセット（最長15ビット）を指定する第１オペランドの８ビット・レジスタの値は0-15だけが有効です。また，ビット長（最長16ビット）を指定するソース・オペランド（src）の値は0-15だけが有効で，０が１ビット長を，15が16ビット長を指定します。

ビット・フィールド・データはメモリのバイト境界にまたがることができます。デスティネーションのビット・フィールドは常にDS1レジスタで指定されるセグメントにロケートされる必要があり，セグメント・オーバライドはできません。

注意　reg8またはreg8'の上位４ビットは０にしてください。

【記　述　例】

●INS DL, CL

●INS DL, 12

【バ　イ　ト　数】

ニモニック	オペランド	バイト数
INS	reg8, reg8'	3
	reg8, imm4	4

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
INS	reg8, reg8'	0	0	0	0	1	1	1	1	0	0	1	1	0	0	0	1
		1	1	reg'			reg			—							
	reg8, imm4	0	0	0	0	1	1	1	1	0	0	1	1	1	0	0	1
		1	1	0	0	0	reg			imm4							

保守／廃止

LDEA

実効アドレスのロード
Load Effective Address

【命 令 形 式】　LDEA reg16, mem16

【オペレーション】　reg16←mem16

【オ ペ ラ ン ド】

ニモニック	オペランド
LDEA	reg16, mem16

【フ　ラ　グ】

V	S	Z	AC	P	CY

【説　　明】　第１オペランドで指定される16ビット汎用レジスタに，第２オペランドで生成される実効アドレス（オフセット）をロードします。
TRANS命令やプリミティブ・ブロック転送命令でオペランドを指定するために自動的に用いられるレジスタなどにオペランド・アドレスの先頭値をセット（１）するときなどに用いられます。

【記　述　例】　AWレジスタにINT_PROCという手続きの実行アドレスのオフセットをロードします。

```
LDEA   AW, INT_PROC
LDEA   AW, [BP] VAR01+2
```

【バ イ ト 数】　2-4

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード													
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
LDEA	reg16, mem16	1	0	0	0	1	1	0	1	mod		reg			mem		
		(disp-low)								(disp-high)							

保守／廃止

LDM
LDMB
LDMW

ブロック・ロード
Load Multiple
Load Multiple Byte
Load Multiple Word

【命 令 形 式】　(repeat) LDM［Seg-spec：］src-block
(repeat) LDMB
(repeat) LDMW

【オペレーション】　［W = 0のとき］AL←（IX）
DIR = 0：IX←IX＋1
DIR = 1：IX←IX－1
［W = 1のとき］AW←（IX＋1, IX）
DIR = 0：IX←IX＋2
DIR = 1：IX←IX－2

【オ ペ ラ ン ド】

ニモニック	オペランド
LDM	［Seg-spec：］src-block
LDMB	な　し
LDMW	

【フ　ラ　グ】

V	S	Z	AC	P	CY

【説　　明】　IXレジスタでアドレスされるブロックをバイトまたはワード単位にアキュームレータ（AL/AW）に繰り返し転送します。
IXレジスタは次のバイト／ワード処理のために１バイト／ワード・データが処理されるごとに自動的にインクリメント（＋１/＋２）またはデクリメント（－１/－２）されます。ブロックの方向は，DIRフラグの状態によって決定されます。
バイトかワードかの指定は，LDM命令を用いる場合はオペランドの属性によって行われ，LDMB命令またはLDMW命令を用いる場合はそれぞれバイト，ワード・タイプに直接指定されます。
ソース・ブロックは，ディフォールト・セグメント・レジスタはDS0レジスタとなっていますが，セグメント・オーバライド可能で，任意のセグメント・レジスタで指定されるセグメント内にロケートできます。

保守／廃止

【記　述　例】　●REP　LDM　DS1：BYTE_VAR ；DS1セグメント
●REP　LDMB　　；DS0セグメント

【バ イ ト 数】　1

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード							
		7	6	5	4	3	2	1	0
LDM	［Seg-spec：］src-block	1	0	1	0	1	1	0	W
LDMB	な　し								
LDMW									

保守／廃止

MOV

データ転送
Move

【命 令 形 式】　① Mov dst, src
② Mov dst1, dst2, src

【オペランド，オペレーション】

命令形式①

ニモニック	オペランド（dst, src）	オペレーション
MOV	reg, reg'	dst←src
	mem, reg	
	reg, mem	
	mem, imm	
	reg, imm	
	acc, dmem	［W = 0のとき］AL←（dmem）
		［W = 1のとき］AH←（dmem＋1）, AL←（dmem）
	dmem, acc	［W = 0のとき］（dmem）←AL
		［W = 1のとき］（dmem＋1）←AH,（dmem）←AL
	sreg, reg16	dst←src
	sreg, mem16	
	reg16, sreg	
	mem16, sreg	
	AH, PSW	AH←S, Z, F1, AC, F0, P, $\overline{\text{IBRK}}$, CY
	PSW, AH	S, Z, F1, AC, F0, P, $\overline{\text{IBRK}}$, CY←AH

命令形式②

ニモニック	オペランド（dst1, dst2, src）	オペレーション
MOV	DS0, reg16, mem32	reg16←（mem32）
		DS0←（mem32＋2）
	DS1, reg16, mem32	reg16←（mem32）
		DS1←（mem32＋2）

【フ ラ グ】　オペランドがPSW, AHの場合

V	S	Z	F1	AC	F0	P	$\overline{\text{IBRK}}$	CY
	×	×	×	×	×	×	×	×

左記以外

V	S	Z	F1	AC	F0	P	$\overline{\text{IBRK}}$	CY

保守／廃止

【説　　明】　命令形式①：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。
オペランドがAH, PSWの場合は，S, Z, F1, AC, F0, P, $\overline{\text{IBRK}}$, CYの各フラグをAHレジスタに転送します。
オペランドがPSW, AHの場合は，AHレジスタのビット0-ビット7をそれぞれPSWのS, Z, F1, AC, F0, P, $\overline{\text{IBRK}}$, CYの各フラグに転送します。

注意　dst = sregまたはsrc = sregの場合，次の命令との間にハードウエア割り込み（マスカブル割り込み，ノンマスカブル割り込み）要求およびシングルステップ・ブレークは受け付けられません。

命令形式②：ソース・オペランド（src）でアドレスされる32ビット・メモリの下位16ビット（32ビット・ポインタ変数のオフセット・ワード）を，デスティネーション・オペランド2（dst2）の16ビット・レジスタに，上位16ビット（セグメント・ワード）をデスティネーション・オペランド1（dst1）のセグメント・レジスタ（DS0レジスタまたはDS1レジスタ）に転送します。

【記　述　例】　メモリ0：50Hに55Hを書く動作

```
MOV   AW, 0
MOV   DS1, AW
MOV   IY, 50H
MOV   DL, 55H
MOV   DS1：［IY］, DL
```

【バ　イ　ト　数】

ニモニック	オペランド	バイト数
MOV	reg, reg'	2
	mem, reg	2-4
	reg, mem	
	mem, imm	3-6
	reg, imm	2, 3
	acc, dmem	3
	dmem, acc	
	sreg, reg16	2
	sreg, mem16	2-4
	reg16, sreg	2
	mem16, sreg	2-4
	DS0, reg16, mem32	
	DS1, reg16, mem32	
	AH, PSW	1
	PSW, AH	

保守／廃止

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
MOV	reg, reg'	1	0	0	0	1	0	1	W	1	1	reg			reg'		
	mem, reg	1	0	0	0	1	0	0	W	mod		reg			mem		
		(disp-low)								(disp-high)							
	reg, mem	1	0	0	0	1	0	1	W	mod		reg			mem		
		(disp-low)								(disp-high)							
	mem, imm	1	1	0	0	0	1	1	W	mod		0	0	0	mem		
		(disp-low)								(disp-high)							
		imm8 or imm16-low								imm16-high							
	reg, imm	1	0	1	1	W	reg			imm8 or imm16-low							
		imm16-high								—							
	acc, dmem	1	0	1	0	0	0	0	W	addr-low							
		addr-high								—							
	dmem, acc	1	0	1	0	0	0	1	W	addr-low							
		addr-high								—							
	sreg, reg16	1	0	0	0	1	1	1	0	1	1	0	sreg		reg		
	sreg, mem16	1	0	0	0	1	1	1	0	mod		0	sreg		mem		
		(disp-low)								(disp-high)							
	reg16, sreg	1	0	0	0	1	1	0	0	1	1	0	sreg		reg		
	mem16, sreg	1	0	0	0	1	1	0	0	mod		0	sreg		mem		
		(disp-low)								(disp-high)							
	DS0, reg16, mem32	1	1	0	0	0	1	0	1	mod		reg			mem		
		(disp-low)								(disp-high)							
	DS1, reg16, mem32	1	1	0	0	0	1	0	0	mod		reg			mem		
		(disp-low)								(disp-high)							
	AH, PSW	1	0	0	1	1	1	1	1	—							
	PSW, AH	1	0	0	1	1	1	1	0	—							

保守／廃止

MOVBK
MOVBKB
MOVBKW

ブロック転送
Move Block
Move Block Byte
Move Block Word

【命 令 形 式】　(repeat) MOVBK［DS1-spec：］dst-block,［Seg-spec：］src-block
(repeat) MOVBKB
(repeat) MOVBKW

【オペレーション】　［W = 0のとき］（IY）←（IX）
DIR = 0：IX←IX＋1, IY←IY＋1
DIR = 1：IX←IX－1, IY←IY－1
［W = 1のとき］（IY＋1，IY）←（IX＋1, IX）
DIR = 0：IX←IX＋2, IY←IY＋2
DIR = 1：IX←IX－2, IY←IY－2

【オ ペ ラ ン ド】

ニモニック	オペランド
MOVBK	［DS1-spec：］dst-block,［Seg-spec：］src-block
MOVBKB	な し
MOVBKW	

【フ ラ グ】

V	S	Z	AC	P	CY

【説　　明】　IXレジスタでアドレスされるブロックを，IYレジスタでアドレスされるブロックに，バイトまたはワード・データの繰り返しで転送します。
IXレジスタ，IYレジスタは，次のバイト／ワード転送のために1バイト／ワード転送ごとに自動的にインクリメント（＋1/＋2），またはデクリメント（－1/－2）されます。ブロックの方向は，DIRフラグの状態によって決定されます。
バイトかワードかの指定は，MOVBK命令を用いる場合はオペランドの属性によって行われ，MOVBKB命令またはMOVBKW命令を用いる場合は，それぞれバイト，ワード・タイプに直接指定されます。
デスティネーション・ブロックは常にDS1レジスタで指定されるセグメント内にロケートされる必要があり，セグメント・オーバライドはできません。
一方，ソース・ブロックは，ディフォールト・セグメント・レジスタはDS0レジスタとなっていますが，セグメント・オーバライド可能で，任意のセグメント・レジスタで指定されるセグメント内にロケートできます。

保守／廃止

【記　述　例】　MOVBK BYTE_VAR1, BYTE_VAR2
MOVBK WORD_VAR1, WORD_VAR2

【バ イ ト 数】　1

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード							
		7	6	5	4	3	2	1	0
MOVBK	［DS1-spec：］dst-blcok, ［Seg-spec：］src-block	1	0	1	0	0	1	0	W
MOVBKB	な　し								
MOVBKW									

保守／廃止

MOVSPA 【V20, V30に対して追加】

データ転送
Move Stack Pointer After Context Switch

【命 令 形 式】　MOVSPA

【オペレーション】　新レジスタ・バンクのSS，SP←旧レジスタ・バンクのSS，SP

【オ ペ ラ ン ド】

ニモニック	オペランド
MOVSPA	な　し

【フ　ラ　グ】

V	S	Z	AC	P	CY

【説　　明】　レジスタ・バンクの切り替わる前のSSの値を現在（切り替え後）のレジスタ・バンクのSSに転送します。同じように，SPの値も転送します。
BRKCS命令または割り込み要求でレジスタ・バンクを切り替えた場合，切り替えの前後でスタックを連続させるときに使用します。

【記　述　例】　MOVSPA

【バ イ ト 数】　2

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
MOVSPA	な　し	0	0	0	0	1	1	1	1	0	0	1	0	0	1	0	1

保守／廃止

MOVSPB 【V20, V30に対して追加】

データ転送
Move Stack Pointer Before Task Switch

【命 令 形 式】　MOVSPB reg16

【オペレーション】　reg16で示される新レジスタ・バンクのSS，SP←旧レジスタ・バンクのSS，SP

【オ ペ ラ ン ド】

ニモニック	オペランド
MOVSPB	reg16

【フ　ラ　グ】

V	S	Z	AC	P	CY

【説　　明】　現在レジスタ・バンク内のSSの値を，オペランドに記述した16ビット・レジスタの内容の下位3ビットで示される切り替え先のレジスタ・バンク内のSSに転送します。同じようにSPの値も転送します。
TSKSW命令でレジスタ・バンクを切り替える場合，切り替えの前後でスタックを連続させるときに使用します。

【記　述　例】　MOVSPB　AW

【バ イ ト 数】　3

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
MOVSPB	reg16	0	0	0	0	1	1	1	1	1	0	0	1	0	1	0	1
		1	1	1	1	1	reg			—							

保守／廃止

MUL

符号付き乗算
Multiply Signed

【命 令 形 式】 ① MUL src
② MUL dst，src
③ MUL dst, src1, src2

【オペランド，オペレーション】

命令形式①

ニモニック	オペランド	オペレーション
MUL	reg8	AW←AL×src AH = ALのサイン拡張 ：CY←0, V←0 AH≠ALのサイン拡張 ：CY←1, V←1
	mem8	
	reg16	DW, AW←AW×src DW = AWのサイン拡張：CY←0, V←0 DW≠AWのサイン拡張：CY←1, V←1
	mem16	

命令形式②

ニモニック	オペランド	オペレーション
MUL	reg16, imm8	dst←dst×src 積≦16ビット：CY←0, V←0 積＞16ビット：CY←1, V←1
	reg16, imm16	

命令形式③

ニモニック	オペランド	オペレーション
MUL	reg16, reg16', imm8	dst←src1×src2 積≦16ビット：CY←0, V←0 積＞16ビット：CY←1, V←1
	reg16, mem16, imm8	
	reg16, reg16', imm16	
	reg16, mem16, imm16	

【フ ラ グ】

V	S	Z	AC	P	CY
×	U	U	U	U	×

保守／廃止

【説　　明】　命令形式①：○src = reg8またはsrc = mem8のとき

ALレジスタの値とソース・オペランド（src）の符号付き乗算を行い，倍長結果をAWレジスタに格納します。このとき結果の上位半分（AHレジスタ）が下位半分（ALレジスタ）の符号拡張でなければCYフラグとVフラグがセット（1）されます。AHレジスタは拡張レジスタです。

○src = reg16またはsrc = mem16のとき

AWレジスタの値とソース・オペランド（src）の符合付き乗算を行い，倍長結果をAWレジスタとDWレジスタに格納します。このとき結果の上位半分（DWレジスタ）が下位半分（AWレジスタ）の符合拡張でなければCYフラグとVフラグがセット（1）されます。DWレジスタは拡張レジスタです。

命令形式②：デスティネーション・オペランド（dst）とソース・オペランド（src）の符号付き乗算を行い，結果をデスティネーション・オペランド（dst）に格納します。

命令形式③：第１ソース・オペランド（src1）と第2ソース・オペランド（src2）の符号付き乗算を行い，結果をデスティネーション・オペランド（dst）に格納します。

【記　述　例】　AWレジスタの値とメモリ0：50Hの内容（ワード・データ）を乗算する。

```
MOV   BW, 0
MOV   DS0, BW
MOV   IX, 50H
MUL   WORD PTR  [IX]
```

保守／廃止

【バ イ ト 数】

ニモニック	オペランド	バイト数
MUL	reg8	2
	mem8	2-4
	reg16	2
	mem16	2-4
	reg16, imm8	3
	reg16, imm16	4
	reg16, reg16', imm8	3
	reg16, mem16, imm8	3-5
	reg16, reg16', imm16	4
	reg16, mem16, imm16	4-6

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
MUL	reg8	1	1	1	1	0	1	1	0	1	1	1	0	1	reg		
	mem8	1	1	1	1	0	1	1	0	mod		1	0	1	mem		
		(disp-low)								(disp-high)							
	reg16	1	1	1	1	0	1	1	1	1	1	1	0	1	reg		
	mem16	1	1	1	1	0	1	1	1	mod		1	0	1	mem		
		(disp-low)								(disp-high)							
	reg16, imm8	0	1	1	0	1	0	1	1	1	1	reg			reg'		
		imm8								—							
	reg16, imm16	0	1	1	0	1	0	0	1	1	1	reg			reg'		
		imm16-low								imm16-high							
	reg16, reg16', imm8	0	1	1	0	1	0	1	1	1	1	reg			reg'		
		imm8								—							
	reg16, mem16, imm8	0	1	1	0	1	0	1	1	mod		reg			mem		
		(disp-low)								(disp-high)							
		imm8								—							
	reg16, reg16', imm16	0	1	1	0	1	0	0	1	1	1	reg			reg'		
		imm16-low								imm16-high							
	reg16, mem16, imm16	0	1	1	0	1	0	0	1	mod		reg			mem		
		(disp-low)								(disp-high)							
		imm16-low								imm16-high							

保守／廃止

MULU

符号なし乗算
Multiply Unsigned

【命 令 形 式】　MULU src

【オペランド，オペレーション】

ニモニック	オペランド（src）	オペレーション
MULU	reg8	AW←AL×src AH = 0：CY←0, V←0 AH≠0：CY←1, V←1
	mem8	
	reg16	DW, AW←AW×src DW = 0：CY←0, V←0 DW≠0：CY←1, V←1
	mem16	

【フ ラ グ】

V	S	Z	AC	P	CY
×	U	U	U	U	×

【説　　明】　○src = reg8またはsrc = mem8のとき

ALレジスタの値とソース・オペランド（src）の符号なし乗算を行い，倍長結果をAWレジスタに格納します。このとき結果の上位半分（AHレジスタ）がゼロでなければCYフラグとVフラグがセット（1）されます。AHレジスタは拡張レジスタです。

○src = reg16またはsrc = mem16のとき

AWレジスタの値とソース・オペランド（src）の符号なし乗算を行い，倍長結果をAWレジスタとDWレジスタに格納します。このとき結果の上位半分（DWレジスタ）がゼロでなければCYフラグとVフラグがセット（1）されます。DWレジスタは拡張レジスタです。

【記 述 例】　ALレジスタの内容とCLレジスタの内容を乗算する。

```
MULU　CL
```

【バ イ ト 数】

ニモニック	オペランド	バイト数
MULU	reg8	2
	mem8	2-4
	reg16	2
	mem16	2-4

保守／廃止

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
MULU	reg8	1	1	1	1	0	1	1	0	1	1	1	0	0	reg		
	mem8	1	1	1	1	0	1	1	0	mod		1	0	0	mem		
		(disp-low)								(disp-high)							
	reg16	1	1	1	1	0	1	1	1	1	1	1	0	0	reg		
	mem16	1	1	1	1	0	1	1	1	mod		1	0	0	mem		
		(disp-low)								(disp-high)							

保守／廃止

NEG

２の補数演算
Negate

【命 令 形 式】　NEG dst

【オペレーション】　dst←$\overline{dst}$＋1

【オ ペ ラ ン ド】

ニモニック	オペランド（dst）
NEG	reg
	mem

【フ　ラ　グ】

V	S	Z	AC	P	CY
×	×	×	×	×	注

注　CY = 1。ただし実行前のdstが0のときはCY = 0。

【説　　明】　デスティネーション・オペランド（dst）の内容の2の補数をとります。

【記　述　例】
- ●NEG　DL
- ●NEG　CW
- ●NEG　IX
- ●NEG　BP

【バ イ ト 数】

ニモニック	オペランド	バイト数
NEG	reg	2
	mem	2-4

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
NEG	reg	1	1	1	1	0	1	1	W	1	1	0	1	1	reg		
	mem	1	1	1	1	0	1	1	W	mod		0	1	1	mem		
		(disp-low)								(disp-high)							

保守／廃止

NOP

ノー・オペレーション
No Operation

【命 令 形 式】　NOP

【オペレーション】　ノー・オペレーション

【オ ペ ラ ン ド】

ニモニック	オペランド
NOP	な　し

【フ　ラ　グ】

V	S	Z	AC	P	CY

【説　　明】　何も処理せず，4クロック費やします。

【記　述　例】　NOP

【バ イ ト 数】　1

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード							
		7	6	5	4	3	2	1	0
NOP	な　し	1	0	0	1	0	0	0	0

保守／廃止

NOT

論理否定
Not

【命 令 形 式】　NOT dst

【オペレーション】　dst←$\overline{dst}$

【オ ペ ラ ン ド】

ニモニック	オペランド（dst）
NOT	reg
	mem

【フ　ラ　グ】

V	S	Z	AC	P	CY

【説　　明】　デスティネーション・オペランド（dst）で指定されるビットを反転し（論理否定），結果をデスティネーション・オペランド（dst）へ格納します。

【記　述　例】
●NOT　AL
●NOT　CW
●NOT　IX

【バ イ ト 数】

ニモニック	オペランド	バイト数
NOT	reg	2
	mem	2-4

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
NOT	reg	1	1	1	1	0	1	1	W	1	1	0	1	0	reg		
	mem	1	1	1	1	0	1	1	W	mod		0	1	0	mem		
		(disp-low)								(disp-high)							

保守／廃止

NOT1

ビットの反転
Not Bit

【命 令 形 式】 ① NOT1 dst, src
② NOT1 dst

【オペレーション】 命令形式①：dstのビットn（nはsrcで指定）←$\overline{\text{dstのビットn（nはsrcで指定）}}$
命令形式②：dst←$\overline{\text{dst}}$

【オ ペ ラ ン ド】 命令形式①

ニモニック	オペランド（dst, src）
NOT1	reg8, CL
	mem8, CL
	reg16, CL
	mem16, CL
	reg8, imm3
	mem8, imm3
	reg16, imm4
	mem16, imm4

命令形式②

ニモニック	オペランド（dst）
NOT1	CY

【フ ラ グ】 命令形式①

V	S	Z	AC	P	CY

命令形式②

V	S	Z	AC	P	CY
					×

【説 明】 命令形式①：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）のビットn（nは第2オペランドで指定されるソース・オペランド（src）の内容）の論理否定をとり，結果をデスティネーション・オペランド（dst）に格納します。
オペランドがreg8，CLまたはmem8，CLのとき，CLの値は下位3ビット（0-7）のみ有効です。
オペランドがreg16，CLまたはmem16，CLのとき，CLの値は下位4ビット（0-15）のみ有効です。
オペランドがreg8，imm3のとき，命令の4バイト目のイミーディエト・データは下位3ビットのみ有効です。
オペランドがmem8，imm3のとき，命令の最終バイトのイミーディエト・データは下位3ビットのみ有効です。
オペランドがreg16，imm4のとき，命令の4バイト目のイミーディエト・データは下位4ビットのみ有効です。
オペランドがmem16，imm4のとき，命令の最終バイトのイミーディエト・データは下位4ビットのみ有効です。

命令形式②：CYフラグの内容の論理否定をとり，CYフラグに格納します。

保守／廃止

【記　述　例】　IN　AL, 0

NOT1　AL, 7

【バ　イ　ト　数】

ニモニック	オペランド	バイト数
NOT1	reg8, CL	3
	mem8, CL	3-5
	reg16, CL	3
	mem16, CL	3-5
	reg8, imm3	4
	mem8, imm3	4-6
	reg16, imm4	4
	mem16, imm4	4-6
	CY	1

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
NOT1	reg8, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	1	1	0
		1	1	0	0	0	reg			—							
	mem8, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	1	1	0
		mod		0	0	0	mem			(disp-low)							
		(disp-high)								—							
	reg16, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1
		1	1	0	0	0	reg			—							
	mem16, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1
		mod		0	0	0	mem			(disp-low)							
		(disp-high)								—							
	reg8, imm3	0	0	0	0	1	1	1	1	0	0	0	1	1	1	1	0
		1	1	0	0	0	reg			imm3							
	mem8, imm3	0	0	0	0	1	1	1	1	0	0	0	1	1	1	1	0
		mod		0	0	0	mem			(disp-low)							
		(disp-high)								imm3							
	reg16, imm4	0	0	0	0	1	1	1	1	0	0	0	1	1	1	1	1
		1	1	0	0	0	reg			imm4							
	mem16, imm4	0	0	0	0	1	1	1	1	0	0	0	1	1	1	1	1
		mod		0	0	0	mem			(disp-low)							
		(disp-high)								imm4							
	CY	1	1	1	1	0	1	0	1	—							

保守／廃止

OR

論理和
Or

【命 令 形 式】　OR dst, src

【オペランド，オペレーション】

ニモニック	オペランド（dst, src）	オペレーション
OR	reg, reg'	dst←dst∨src
	mem, reg	
	reg, mem	
	reg, imm	
	mem, imm	
	acc, imm	［W = 0のとき］AL←AL∨imm8 ［W = 1のとき］AW←AW∨imm16

【フ ラ グ】

V	S	Z	AC	P	CY
0	×	×	U	×	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）と第２オペランドで指定されるソース・オペランド（src）の論理和（OR）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。

【記 述 例】　OR　AW, WORD PTR［IX］

【バ イ ト 数】

ニモニック	オペランド	バイト数
OR	reg, reg'	2
	mem, reg	2-4
	reg, mem	2-4
	reg, imm	3, 4
	mem, imm	3-6
	acc, imm	2, 3

保守／廃止

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
OR	reg, reg'	0	0	0	0	1	0	1	W	1	1	reg			reg'		
	mem, reg	0	0	0	0	1	0	0	W	mod		reg			mem		
		(disp-low)								(disp-high)							
	reg, mem	0	0	0	0	1	0	1	W	mod		reg			mem		
		(disp-low)								(disp-high)							
	reg, imm[注]	1	0	0	0	0	0	0	W	1	1	0	0	1	reg		
		imm8 or imm16-low								imm16-high							
	mem, imm	1	0	0	0	0	0	0	W	mod		0	0	1	mem		
		(disp-low)								(disp-high)							
		imm8 or imm16-low								imm16-high							
	acc, imm	0	0	0	0	1	1	0	W	imm8 or imm16-low							
		imm16-high								—							

注 アセンブラ，コンパイラによっては，次に示すようなコードが生成されることがあります。

オペレーション・コード															
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
1	0	0	0	0	0	1	W	1	1	0	0	1	reg		
imm8								—							

このような場合でも正常に命令を実行します。ただし，エミュレータによっては，この命令に対する逆アセンブラ機能，ライン・アセンブラ機能がサポートされていない場合がありますのでご注意ください。

保守／廃止

OUT

I/Oデバイスへのデータ出力
Output

【命 令 形 式】　OUT dst, src

【オペランド，オペレーション】

$\overline{IBRK}$ = 1のとき

ニモニック	オペランド（dst, src）	オペレーション
OUT	imm8, acc	[W = 0のとき]（imm8）←AL [W=1のとき]（imm8＋1）←AH，（imm8）←AL
	DW, acc	[W = 0のとき]（DW）←AL [W = 1のとき]（DW＋1）←AH，（DW）←AL

$\overline{IBRK}$ = 0のとき

ニモニック	オペランド（dst, src）	オペレーション
OUT	imm8, acc	（SP－1，SP－2）←PSW （SP－3，SP－4）←PS （SP－5，SP－6）←PC－x[注]
	DW, acc	IE←0，BRK←0，$\overline{IBRK}$←1 PC←（04DH, 04CH） PS←（04FH, 04EH）

注　xは命令のバイト数＋プリフィクスの数

【フ ラ グ】　$\overline{IBRK}$ = 1のとき

V	IE	BRK	S	Z	AC	P	$\overline{IBRK}$	CY

$\overline{IBRK}$ = 0のとき

V	IE	BRK	S	Z	AC	P	$\overline{IBRK}$	CY
	0	0					1	

【説 明】　$\overline{IBRK}$ = 1のとき，デスティネーション・オペランド（dst）で指定されるI/Oデバイスのレジスタにアキュームレータ（ALレジスタまたはAWレジスタ）の内容を転送します。

$\overline{IBRK}$ = 0のとき，PSW，PS，PC－xをスタックに退避し，IEとBRKフラグをリセット（0）し，$\overline{IBRK}$をセット（1）します。次に割り込みベクタ・テーブルのベクタ19の下位2バイトをPCに，上位2バイトをPSにロードします。

この機能は，ソフトウエアの移植を容易にするために用意された機能です。

【記 述 例】　ポート・アドレス0D8HにALレジスタの内容を転送する。

```
MOV    DW, 0D8H
OUT    DW, AL
```

保守／廃止

【バ イ ト 数】

ニモニック	オペランド	バイト数
OUT	imm8, acc	2
	DW, acc	1

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード													
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
OUT	imm8, acc	1	1	1	0	0	1	1	W	imm8							
	DW, acc	1	1	1	0	1	1	1	W	—							

保守／廃止

OUTM

メモリ-I/Oのブロック転送
Output Multiple

【命 令 形 式】　(repeat) OUTM DW, [Seg-spec：] src-block

【オペレーション】　$\overline{IBRK}$ = 1のとき［W = 0のとき］（DW）←（IX）
DIR = 0：IX←IX＋1
DIR = 1：IX←IX－1
［W = 1のとき］（DW＋1, DW）←（IX＋1, IX）
DIR = 0：IX←IX＋2
DIR = 1：IX←IX－2

$\overline{IBRK}$ = 0のとき（SP－1, SP－2）←PSW
（SP－3, SP－4）←PS
（SP－5, SP－6）←PC－x[注]
IE←0, BRK←0, $\overline{IBRK}$←1
PC←（04DH, 04CH）
PS←（04FH, 04EH）

注　xは命令のバイト数＋プリフィクスの数

【オ ペ ラ ン ド】

ニモニック	オペランド
OUTM	DW, [Seg-spec：] src-block

【フ ラ グ】　$\overline{IBRK}$ = 1のとき

V	IE	BRK	S	Z	AC	P	$\overline{IBRK}$	CY

$\overline{IBRK}$ = 0のとき

V	IE	BRK	S	Z	AC	P	$\overline{IBRK}$	CY
	0	0					1	

【説 明】　$\overline{IBRK}$ = 1のとき，IXレジスタでアドレスされるメモリの内容を，DWレジスタでアドレスされるI/Oデバイスのレジスタに転送します。繰り返し転送回数は，ペアで使用されるリピート・プリフィクスのREP命令によって制御されます。繰り返し転送の際，DWレジスタの内容（I/Oデバイスのアドレス）は固定ですが，IXレジスタは次のバイト／ワード転送のために1バイト／ワード・データが転送されるごとに自動的にインクリメント（＋1/＋2）またはデクリメント（－1/－2）されます。ブロックの方向は，DIRフラグの内容によって決定されます。

バイトかワードかの指定は，オペランド属性によって行われます。

OUTM命令は，リピート・プリフィクスのREP命令とともに使用されます。

ソース・ブロックは，ディフォールト・セグメント・レジスタはDS0レジスタとなっていますが，セグメント・オーバライド可能で，任意のセグメント・レジスタで指定されるセグメント内にローケートできます。

保守／廃止

$\overline{IBRK}$ = 0のとき，PSW, PS, PC－xをスタックに退避し，IEとBRKフラグをリセット（0）し，$\overline{IBRK}$をセット（1）します。次に割り込みベクタ・テーブルのベクタ19の下位2バイトをPCに，上位2バイトをPSにロードします。
この機能は，ソフトウエアの移植を容易にするために用意された機能です。

【記　述　例】　●メモリ0：50Hの内容をポート・アドレス0D8H（バイト・データ）に転送する。

```
MOV   AW, 0
MOV   DS0, AW
MOV   IX, 50H
MOV   DW, 0D8H
OUTM  DW, DS0：WORD PTR［IX］
```

●メモリ0：0H～0FFHの内容をポート・アドレス0D8H（バイト・データ）に転送する。

```
MOV   AW, 0
MOV   DS0, AW
MOV   IX, 0H
MOV   DW, 0D8H
MOV   CW, 0FFH
REP   OUTM DW, DS0：PTR［IX］
```

【バ イ ト 数】　1

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード							
		7	6	5	4	3	2	1	0
OUTM	DW, ［Seg-spec：］ src-block	0	1	1	0	1	1	1	W

保守／廃止

POLL

浮動小数点演算用コプロセッサ待ち
Poll and wait

【命 令 形 式】　POLL

【オペレーション】　POLL and wait

【オ ペ ラ ン ド】

ニモニック	オペランド
POLL	な　し

【フ　ラ　グ】

V	S	Z	AC	P	CY

【説　　明】　$\overline{\text{POLL}}$端子がアクティブ（ロウ・レベル）になるまでCPUをウエイト状態にします。

注意１．この命令の直前にBUSLOCK命令を置くことはできません。
２．P14が入力ポートとして設定されているときPOLL命令は有効です。その他の場合は，$\overline{\text{POLL}}$端子は常時アクティブ（ロウ・レベル）とみなされます。

【記　述　例】　POLL

【バ イ ト 数】　１

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード							
		7	6	5	4	3	2	1	0
POLL	な　し	1	0	0	1	1	0	1	1

保守／廃止

POP

スタックからの復帰
Pop

【命 令 形 式】　POP dst

【オペランド，オペレーション】

ニモニック	オペランド（dst）	オペレーション
POP	mem16	SP←SP＋2 （mem16）←（SP－1, SP－2）
	reg16	SP←SP＋2 dst←（SP－1, SP－2）
	sreg	
	PSW	
	R	IY←（SP＋1, SP） IX←（SP＋3, SP＋2） BP←（SP＋5, SP＋4） BW←（SP＋9, SP＋8） DW←（SP＋11, SP＋10） CW←（SP＋13, SP＋12） AW←（SP＋15, SP＋14） SP←SP＋16

【フ ラ グ】　○dst = PSWの場合

V	DIR	IE	BRK	S	Z	F1	AC	F0	P	$\overline{\text{IBRK}}$	CY
R	R	R	R	R	R	R	R	R	R	R	R

備考　RB0-2フラグは変化しません。

○dst = PSW以外の場合

V	S	Z	AC	P	CY

【説　　明】　デスティネーション・オペランド（dst）の内容にスタックの内容を転送します（ただし，dst = sregの場合，PSへは転送されません）。

注意　dst = sregの場合，次の命令との間にハードウエア割り込み（マスカブル割り込み，ノンマスカブル割り込み）要求およびシングルステップ・ブレークは受け付けられません。

【記 述 例】　●POP　AW
●POP　BW
●POP　IY
●POP　SP
　MOV　BP, SP

保守／廃止

【バ イ ト 数】

ニモニック	オペランド	バイト数
POP	mem16	2-4
	reg16	1
	sreg	
	PSW	
	R	1

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
POP	mem16	1	0	0	0	1	1	1	1	mod		0	0	0	mem		
		(disp-low)								(disp-high)							
	reg16	0	1	0	1	1	reg			—							
	sreg	0	0	0	sreg		1	1	1	—							
	PSW	1	0	0	1	1	1	0	1	—							
	R	0	1	1	0	0	0	0	1	—							

保守／廃止

PREPARE

スタック・フレームの生成
Prepare New Stack Frame

【命　令　形　式】　PREPARE imm16, imm8

【オペレーション】　（SP－1, SP－2）←BP
SP←SP－2
temp←SPを行ったあと
imm8＞0のとき次の動作を，
　（SP－1, SP－2）←（BP－1, BP－2）
　SP←SP－2
　BP←BP－2
　“ imm8－1 ”回繰り返したあとに，　＊1
　（SP－1, SP－2）←temp
　SP←SP－2　＊2
を実行します。
このあと次の処理を行います。
BP←temp
SP←SP－imm16
imm8 = 1のとき，＊1の繰り返し動作を行いません。
imm8 = 0のとき，＊1および＊2の動作を行いません。

【オ　ペ　ラ　ン　ド】

ニモニック	オペランド
PREPARE	imm16, imm8

【フ　　ラ　　グ】

V	S	Z	AC	P	CY

【説　　明】　この命令は，ブロック構造の高級言語（たとえばPascal, Adaなど）において必要な“ スタック・フレーム ”を生成するために用いられます。スタック・フレームには，そのプロシージャから参照してもよい変数を含むフレームを指すポインタ群とローカル変数の領域が含まれます。

この命令は，ローカル変数の領域確保と，グローバル変数への参照を可能とするため，フレーム・ポインタのコピーを行います。第１オペランドに記述された16ビット・イミーディエト・データでローカル変数用として確保する領域の大きさ（バイト単位）が指定され，第２オペランドに記述された８ビット・イミーディエト・データでプロシージャ・ブロックの深さ（この深さをレキシカル・レベルといいます）が示されます。

この命令によって生成されるフレームのベース・アドレスは，BPにセットされます。

まずBPをスタックへ退避します。これは，プロシージャの終了時点で，コールした側のプロシージャのBPを復帰するためです。次に，コールされたプロシージャから参照可能な範囲の

保守／廃止

フレーム・ポインタ（退避されたBP）をスタックに積みます。参照可能な範囲は，そのプロシージャのレキシカル・レベルより１減じた値となります。
レキシカル・レベルが１以上の場合には，自分自身のフレーム・ポインタもスタックへ積みます。これは，このプロシージャからコールされたプロシージャにおいてフレーム・ポインタのコピーを行う場合に，コールしたプロシージャのフレーム・ポインタもコピーするためです。
次にBPに新しいフレーム・ポインタの値をセットし，そのプロシージャで使用されるローカル変数の領域をスタックに確保します。つまり，SPをローカル変数分だけ減らします。

【記　述　例】

```
MOV       SP, 60H
MOV       BP, SP
CALL      CHK
PREPARE 0006, 04
MOV       AW, ［BP＋0FAH］
ADD       AW, ［BP＋0F8A］
MOV        ［BP＋0FCH］, AW
```

【バ イ ト 数】　４

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
PREPARE	imm16, imm8	1	1	0	0	1	0	0	0	imm16-low							
		imm16-high								imm8							

保守／廃止

PUSH

スタックへの退避
Push

【命 令 形 式】　PUSH src

【オペランド，オペレーション】

ニモニック	オペランド（src）	オペレーション
PUSH	mem16	SP←SP－2 （SP＋1, SP）←（mem16＋1, mem16）
	reg16	SP←SP－2 （SP＋1, SP）←src
	sreg	
	PSW	
	R	TEMP←SP （SP－1, SP－2）←AW （SP－3, SP－4）←CW （SP－5, SP－6）←DW （SP－7, SP－8）←BW （SP－9, SP－10）←TEMP （SP－11, SP－12）←BP （SP－13, SP－14）←IX （SP－15, SP－16）←IY SP←SP－16
	imm8	（SP－1, SP－2）←imm8のサイン拡張 SP←SP－2
	imm16	（SP－1, SP－2）←imm16 SP←SP－2

【フ ラ グ】

V	S	Z	AC	P	CY

【説　　明】　ソース・オペランド（src）の内容をスタックに退避します。
オペランドに８ビット・イミーディエト・データ（imm8）を記述した場合は，imm8はサイン拡張され，16ビット・データとしてSPでアドレスされるスタックに退避されます。

【記 述 例】　●PUSH　DS0
●PUSH　SS
●PUSH　DS1

保守／廃止

【バ イ ト 数】

ニモニック	オペランド	バイト数
PUSH	mem16	2-4
	reg16	1
	sreg	
	PSW	
	R	1
	imm8	2
	imm16	3

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
PUSH	mem16	1	1	1	1	1	1	1	1	mod		1	1	0	mem		
		(disp-low)								(disp-high)							
	reg16	0	1	0	1	0	reg			—							
	sreg	0	0	0	sreg		1	1	0	—							
	PSW	1	0	0	1	1	1	0	0	—							
	R	0	1	1	0	0	0	0	0	—							
	imm8	0	1	1	0	1	0	1	0	imm8							
	imm16	0	1	1	0	1	0	0	0	imm16-low							
		imm16-high								—							

保守／廃止

REP
REPE
REPZ

Z = 1によるリピート・プリフィクス
Repeat
Repeat while Equal
Repeat while Zero

【命 令 形 式】　REP
REPE
REPZ

【オペレーション】　［CW≠0のとき］PS：PC＋1のバイト命令実行
CW←CW－1
Z≠1のとき：PC←PC＋2
Z = 1のとき：再実行
［CW = 0のとき］PC←PC＋2

【オ ペ ラ ン ド】

ニモニック	オペランド
REP	な　し
REPE	
REPZ	

【フ　ラ　グ】

V	S	Z	AC	P	CY

【説　　明】　CW≠0の間，続くバイトのブロック転送／比較／入出力命令を実行し，CWレジスタの値をデクリメント（－1）します。
REPはMOVBK, LDM, STM, OUTM, INMの各命令とともに用いられ，Zフラグの値に関係なくCW≠0の間繰り返し処理を行います。
REPZ命令およびREPE命令はCMPBK, CMPMの各命令とともに用いられ，各ブロック命令による比較の結果Z≠1になるか，またはCW = 0になるとループを抜けます。
CWレジスタのチェックは，ブロック命令の実行前，すなわちREP/REPE/REPZ命令実行直前の状態に対して行われます。したがって，最初CW = 0でREP/REPE/REPZ命令実行に入ると，続くブロック命令は１回も実行されずにその次の命令に移ります。
Zフラグのチェックは，あくまで続くブロック命令の結果に対して行われ，最初REPE/REPZ命令に入る直前の内容は関係ありません。

注意　この命令と次の命令との間では，ハードウエア割り込み（マスカブル割り込み，ノンマスカブル割り込み）要求およびシングルステップ・ブレークは受け付けられません。

保守／廃止

【記　述　例】　●REP MOVBKW

●REPZ CMPBKW

【バ イ ト 数】　1

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード							
		7	6	5	4	3	2	1	0
REP	な　し	1	1	1	1	0	0	1	1
REPE									
REPZ									

保守／廃止

REPC

CY = 1によるリピート・プリフィクス
Repeat while Carry

【命 令 形 式】　REPC

【オペレーション】　［CW≠0のとき］PS：PC＋1のバイト命令実行
CW←CW－1
CY≠1のとき：PC←PC＋2
CY = 1のとき：再実行
［CW = 0のとき］PC←PC＋2

【オ ペ ラ ン ド】

ニモニック	オペランド
REPC	な　し

【フ　ラ　グ】

V	S	Z	AC	P	CY

【説　　明】　CW≠0の間，続くバイトのブロック比較命令（CMPBK, CMPM）を実行し，CWレジスタの値をデクリメント（－1）します。
ブロック比較命令の結果，CY≠1になるとループを抜けます。
CWレジスタのチェックは，ブロック比較命令の実行前，すなわちREPC命令実行の直前の状態に対して行われます。したがって，最初CW = 0でREPC命令実行に入ると，続くブロック比較命令は１回も実行されずにその次の命令に移ります。
CYフラグのチェックは，あくまで続くブロック比較命令の結果に対して行われ，最初REPC命令に入る直前の内容は関係ありません。

注意　この命令と次の命令との間では，ハードウエア割り込み（マスカブル割り込み，ノンマスカブル割り込み）要求およびシングルステップ・ブレークは受け付けられません。

【記　述　例】　REPC CMPBKW

【バ イ ト 数】　１

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード							
		7	6	5	4	3	2	1	0
REPC	な　し	0	1	1	0	0	1	0	1

保守／廃止

REPNC

CY = 0によるリピート・プリフィクス
Repeat while Not Carry

【命 令 形 式】　REPNC

【オペレーション】　［CW≠0のとき］PS：PC＋1のバイト命令実行
CW←CW－1
CY≠1のとき：再実行
CY = 1のとき：PC←PC＋2
［CW = 0のとき］PC←PC＋2

【オ ペ ラ ン ド】

ニモニック	オペランド
REPNC	な　し

【フ　ラ　グ】

V	S	Z	AC	P	CY

【説　　明】　CW≠0の間，続くバイトのブロック比較命令（CMPBK, CMPM）を実行し，CWレジスタの値をデクリメント（－1）します。
ブロック比較命令の結果，CY = 1になるとループを抜けます。
CWレジスタのチェックは，ブロック比較命令の実行前，すなわちREPNC命令実行の直前の状態に対して行われます。したがって，最初CW = 0でREPNC命令実行に入ると，続くブロック比較命令は１回も実行されずにその次の命令に移ります。
CYフラグのチェックは，あくまで続くブロック比較命令の結果に対して行われ，最初REPNC命令に入る直前の内容は関係ありません。

注意　この命令と次の命令との間では，ハードウエア割り込み（マスカブル割り込み，ノンマスカブル割り込み）要求およびシングルステップ・ブレークは受け付けられません。

【記　述　例】　REPNC　CMPMB

【バ イ ト 数】　１

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード							
		7	6	5	4	3	2	1	0
REPNC	な　し	0	1	1	0	0	1	0	0

保守／廃止

REPNE
REPNZ

Z = 0によるリピート・プリフィクス
Repeat while Not Equal
Repeat while Not Zero

【命 令 形 式】　REPNE
REPNZ

【オペレーション】　［CW≠0のとき］PS：PC＋1のバイト命令実行
CW←CW－1
Z≠1のとき：再実行
Z = 1のとき：PC←PC＋2
［CW = 0のとき］PC←PC＋2

【オ ペ ラ ン ド】

ニモニック	オペランド
REPNE	な　し
REPNZ	

【フ　ラ　グ】

V	S	Z	AC	P	CY

【説　　明】　CW≠0の間，続くバイトのブロック比較命令（CMPBK, CMPM）を実行し，CWレジスタの値をデクリメント（－1）します。
ブロック比較命令の結果，Z≠0になるか，またはCW = 0になるとループを抜けます。
CWレジスタのチェックは，ブロック比較命令の実行前，すなわちREPNE/REPNZ命令実行直前に対して行われます。したがって，最初CW = 0でREPNE/REPNZ命令実行に入ると，続くブロック比較命令は１回も実行されずにその次の命令に移ります。
Zフラグのチェックは，あくまで続くブロック比較命令の結果に対して行われ，最初REPNE/REPNZ命令に入る直前の内容は関係ありません。

注意　この命令と次の命令との間では，ハードウエア割り込み（マスカブル割り込み，ノンマスカブル割り込み）要求およびシングルステップ・ブレークは受け付けられません。

【記　述　例】　●REPNE　CMPMB
●REPNZ　CMPBKW

【バ イ ト 数】　１

保守／廃止

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード							
		7	6	5	4	3	2	1	0
REPNE	な　し	1	1	1	1	0	0	1	0
REPNZ									

保守／廃止

RET

サブルーチンからの復帰
Return from Procedure

【命 令 形 式】　① RET
② RET pop-value

【オペランド，オペレーション】

○セグメント内コールからのリターンのとき

ニモニック	オペランド	オペレーション
RET	な　し	PC←（SP＋1, SP） SP←SP＋2
	pop-value	PC←（SP＋1, SP） SP←SP＋2 SP←SP＋pop-value

○セグメント外コールからのリターンのとき

ニモニック	オペランド	オペレーション
RET	な　し	PC←（SP＋1, SP） PS←（SP＋3, SP＋2） SP←SP＋4
	pop-value	PC←（SP＋1, SP） PS←（SP＋3, SP＋2） SP←SP＋4 SP←SP＋pop-value

【フ　ラ　グ】

V	S	Z	AC	P	CY

【説　　明】　○セグメント内コールからのリターンのとき

PCをスタックからリストアします。オペランドにpop-valueを記述したときは16ビットのpop-valueからSPに加算されます（PCに続いて退避してあったパラメータが不要になったとき，不要なパラメータの分だけSPの値を飛ばすのに有効です）。
セグメント外コールからのRET命令との区別はアセンブラが自動的に行います。

○セグメント外コールからのリターンのとき

PCとPSをスタックからリストアします。オペランドにpop-valueを記述したときは16ビットのpop-valueがSPに加算されます（PCとPSに続いて退避してあったパラメータが不要になったとき，不要なパラメータの分だけSPの値を飛ばすのに有効です）。
セグメント内コールからのRET命令との区別はアセンブラが自動的に行います。

【記　述　例】
```
POP R
RET
```

保守／廃止

【バ　イ　ト　数】

ニモニック	オペランド	バイト数
RET	な　し	1
	pop-value	3

【命 令 語 形 式】　○セグメント内コールからのリターンのとき

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
RET	な　し	1	1	0	0	0	0	1	1	—							
	pop-value	1	1	0	0	0	0	1	0	pop-value-low							
		pop-value-high								—							

○セグメント外コールからのリターンのとき

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
RET	な　し	1	1	0	0	1	0	1	1	—							
	pop-value	1	1	0	0	1	0	1	0	pop-value-low							
		pop-value-high								—							

保守／廃止

RETI

割り込みからの復帰
Return from Interrupt

【命 令 形 式】　RETI

【オペレーション】　PC←（SP＋1, SP）
PS←（SP＋3, SP＋2）
PSW←（SP＋5, SP＋4）
SP←SP＋6

【オ ペ ラ ン ド】

ニモニック	オペランド
RETI	な　し

【フ　ラ　グ】

RB2	RB1	RB0	V	DIR	IE	BRK	S	Z	F1	AC	F0	P	$\overline{\text{IBRK}}$	CY
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

【説　　明】　PC, PS, PSWにスタックの内容をリストアします。割り込み処理から戻るときに用いられます。

【記　述　例】
```
POP R
RETI
```

【バ イ ト 数】　1

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード							
		7	6	5	4	3	2	1	0
RETI	な　し	1	1	0	0	1	1	1	1

保守／廃止

RETRBI 【V20, V30に対して追加】

レジスタ・バンク割り込みまたはBRKCS命令からの復帰
Return from Register Bank Switching

【命 令 形 式】 RETRBI

【オペレーション】 PC←PC退避
PSW←PSW退避

【オ ペ ラ ン ド】

ニモニック	オペランド
RETRBI	な し

【フ ラ グ】

RB2	RB1	RB0	V	DIR	IE	BRK	S	Z	F1	AC	F0	P	$\overline{\text{IBRK}}$	CY
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

【説 明】 レジスタ・バンク切り替え機能を使用した割り込みまたはBRKCS命令からの復帰に使用します。RETI命令では正常にこの割り込みからの復帰はできません。また，ほかの割り込みからの復帰にこの命令を使用することはできません。
レジスタ・バンク切り替え機能を使用した割り込みからの復帰では，この命令の実行前にFINT命令を実行する必要があります。

【記 述 例】 RETRBI

【バ イ ト 数】 2

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
RETRBI	な し	0	0	0	0	1	1	1	1	1	0	0	1	0	0	0	1

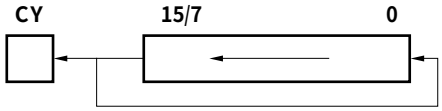
保守／廃止

ROL

左方向のローテート
Rotate Left

【命 令 形 式】　ROL dst, src

【オペレーション】



【オ ペ ラ ン ド】

ニモニック	オペランド（dst, src）
ROL	reg, 1
	mem, 1
	reg, CL
	mem, CL
	reg, imm8
	mem, imm8

【フ　ラ　グ】

src = 1の場合

V	S	Z	AC	P	CY
×					×

左記以外

V	S	Z	AC	P	CY
U					×

【説　　明】　○src = 1のとき

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ビットだけ左シフトします。dstの内容のMSB（ビット７またはビット15）のデータはLSB（ビット０）へシフトされると同時に，CYフラグへも転送されます。

また，MSBが変化したときはVフラグがセット（１）され，元のままのときはリセット（０）されます。

○src = CLまたはsrc = imm8のとき

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）の内容のビット数分だけ左シフトします。dstの内容のMSB（ビット７またはビット15）のデータはLSB（ビット０）へシフトされると同時に，CYフラグへも転送されます。

【記　述　例】

```
MOV［IX］, BL
ROL  BYTE PTR［IX］, 1
```

保守／廃止

【バ イ ト 数】

ニモニック	オペランド	バイト数
ROL	reg, 1	2
	mem, 1	2-4
	reg, CL	2
	mem, CL	2-4
	reg, imm8	3
	mem, imm8	3-5

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
ROL	reg, 1	1	1	0	1	0	0	0	W	1	1	0	0	0	reg		
	mem, 1	1	1	0	1	0	0	0	W	mod		0	0	0	mem		
		(disp-low)								(disp-high)							
	reg, CL	1	1	0	1	0	0	1	W	1	1	0	0	0	reg		
	mem, CL	1	1	0	1	0	0	1	W	mod		0	0	0	mem		
		(disp-low)								(disp-high)							
	reg, imm8	1	1	0	0	0	0	0	W	1	1	0	0	0	reg		
		imm8								—							
	mem, imm8	1	1	0	0	0	0	0	W	mod		0	0	0	mem		
		(disp-low)								(disp-high)							
		imm8								—							

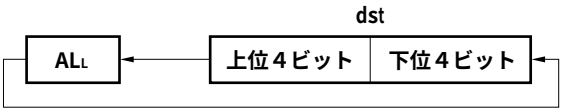
保守／廃止

ROL4

左方向のニブル・ローテート
Rotate Left Nibble

【命 令 形 式】　ROL4 dst

【オペレーション】



【オ ペ ラ ン ド】

ニモニック	オペランド（dst）
ROL4	reg8
	mem8

【フ　ラ　グ】

V	S	Z	AC	P	CY

【説　　明】　デスティネーション・オペランド（dst）の内容を２桁のパックトBCDとして扱い，ALレジスタの下位４ビット（ALL）を介して，１桁分左に回転します。
この命令の結果，ALレジスタの上位４ビットは保証されません。

【記　述　例】
●MOV　AL, 24H
　ROL4　AL
●MOV　AL, BYTE PTR［IX］
　ROL4　AL

【バ イ ト 数】

ニモニック	オペランド	バイト数
ROL4	reg8	3
	mem8	3-5

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
ROL4	reg8	0	0	0	0	1	1	1	1	0	0	1	0	1	0	0	0
		1	1	0	0	0	reg			—							
	mem8	0	0	0	0	1	1	1	1	0	0	1	0	1	0	0	0
		mod		0	0	0	mem			(disp-low)							
		(disp-high)								—							

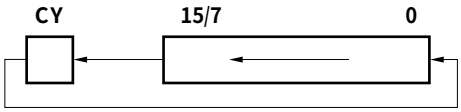
保守／廃止

ROLC

キャリーを含む左方向のローテート
Rotate Left with Carry

【命 令 形 式】　ROLC dst, src

【オペレーション】



【オ ペ ラ ン ド】

ニモニック	オペランド（dst, src）
ROLC	reg, 1
	mem, 1
	reg, CL
	mem, CL
	reg, imm8
	mem, imm8

【フ　ラ　グ】

src = 1の場合

V	S	Z	AC	P	CY
×					×

左記以外

V	S	Z	AC	P	CY
U					×

【説　　明】

○src = 1のとき

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フラグを通して１ビットだけ左シフトします。dstの内容のMSB（ビット７またはビット15）のデータはCYフラグへ転送されCYフラグのデータはLSB（ビット０）へ転送されます。
また，MSBが変化したときはVフラグがセット（１）され，元のままのときはリセット（０）されます。

○src = CLまたはsrc = imm8のとき

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）の内容のビット数分だけCYフラグを通して左シフトします。dstの内容のMSB（ビット７またはビット15）のデータはCYフラグへ転送されCYフラグのデータはLSB（ビット０）へ転送されます。

【記　述　例】

●ROLC　AL, 1
●ROLC　CL, 1
●ROLC　DW, 1
●ROLC　AW, 1

保守／廃止

【バ イ ト 数】

ニモニック	オペランド	バイト数
ROLC	reg, 1	2
	mem, 1	2-4
	reg, CL	2
	mem, CL	2-4
	reg, imm8	3
	mem, imm8	3-5

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
ROLC	reg, 1	1	1	0	1	0	0	0	W	1	1	0	1	0	reg		
	mem, 1	1	1	0	1	0	0	0	W	mod		0	1	0	mem		
		(disp-low)								(disp-high)							
	reg, CL	1	1	0	1	0	0	1	W	1	1	0	1	0	reg		
	mem, CL	1	1	0	1	0	0	1	W	mod		0	1	0	mem		
		(disp-low)								(disp-high)							
	reg, imm8	1	1	0	0	0	0	0	W	1	1	0	1	0	reg		
		imm8								—							
	mem, imm8	1	1	0	0	0	0	0	W	mod		0	1	0	mem		
		(disp-low)								(disp-high)							
		imm8								—							

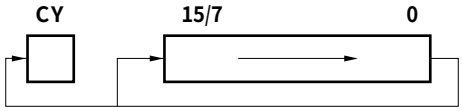
保守／廃止

ROR

右方向のローテート
Rotate Right

【命 令 形 式】　ROR dst, src

【オペレーション】



【オ ペ ラ ン ド】

ニモニック	オペランド（dst, src）
ROR	reg, 1
	mem, 1
	reg, CL
	mem, CL
	reg, imm8
	mem, imm8

【フ　ラ　グ】

src = 1の場合

V	S	Z	AC	P	CY
×					×

左記以外

V	S	Z	AC	P	CY
U					×

【説　　明】

○src = 1のとき

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ビットだけ右シフトします。dstの内容のLSB（ビット０）のデータはMSB（ビット７またはビット15）へシフトされると同時に，CYフラグへも転送されます。

また，MSBが変化したときはVフラグがセット（１）され，元のままのときはリセット（０）されます。

○src = CLまたはsrc = imm8のとき

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）の内容のビット数分だけ右シフトします。dstの内容のLSB（ビット０）のデータはMSB（ビット７またはビット15）へシフトされると同時に，CYフラグへも転送されます。

【記　述　例】

●ROR　AL, 3

●ROR　CW, 6

●ROR　IY, 2

保守／廃止

【バ イ ト 数】

ニモニック	オペランド	バイト数
ROR	reg, 1	2
	mem, 1	2-4
	reg, CL	2
	mem, CL	2-4
	reg, imm8	3
	mem, imm8	3-5

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
ROR	reg, 1	1	1	0	1	0	0	0	W	1	1	0	0	1	reg		
	mem, 1	1	1	0	1	0	0	0	W	mod		0	0	1	mem		
		(disp-low)								(disp-high)							
	reg, CL	1	1	0	1	0	0	1	W	1	1	0	0	1	reg		
	mem, CL	1	1	0	1	0	0	1	W	mod		0	0	1	mem		
		(disp-low)								(disp-high)							
	reg, imm8	1	1	0	0	0	0	0	W	1	1	0	0	1	reg		
		imm8								—							
	mem, imm8	1	1	0	0	0	0	0	W	mod		0	0	1	mem		
		(disp-low)								(disp-high)							
		imm8								—							

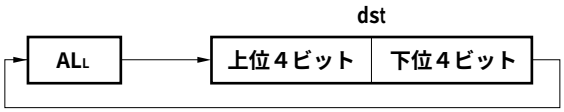
保守／廃止

ROR4

右方向のニブル・ローテート
Rotate Right Nibble

【命 令 形 式】　ROR4 dst

【オペレーション】



【オ ペ ラ ン ド】

ニモニック	オペランド（dst）
ROR4	reg8
	mem8

【フ　ラ　グ】

V	S	Z	AC	P	CY

【説　　明】　デスティネーション・オペランド（dst）の内容を２桁のパックトBCDとして扱い，ALレジスタの下位４ビット（ALL）を介して，１桁分右に回転します。
この命令の結果，ALレジスタの上位４ビットは保証されません。

【記　述　例】
●MOV　AL, 24H
　ROR4　AL
●MOV　AL, BYTE PTR［IX］
　ROR4　AL

【バ イ ト 数】

ニモニック	オペランド	バイト数
ROR4	reg8	3
	mem8	3-5

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
ROR4	reg8	0	0	0	0	1	1	1	1	0	0	1	0	1	0	1	0
		1	1	0	0	0	reg			—							
	mem8	0	0	0	0	1	1	1	1	0	0	1	0	1	0	1	0
		mod		0	0	0	mem			(disp-low)							
		(disp-high)								—							

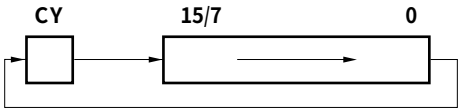
保守／廃止

RORC

キャリーを含む右方向のローテート
Rotate Right with Carry

【命 令 形 式】　RORC dst, src

【オペレーション】



【オ ペ ラ ン ド】

ニモニック	オペランド（dst, src）
RORC	reg, 1
	mem, 1
	reg, CL
	mem, CL
	reg, imm8
	mem, imm8

【フ　ラ　グ】　src = 1の場合

V	S	Z	AC	P	CY
×					×

左記以外

V	S	Z	AC	P	CY
U					×

【説　　明】　○src = 1のとき

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フラグを通して１ビットだけ右シフトします。dstの内容のLSB（ビット０）のデータはCYフラグへ転送されCYフラグのデータはMSB（ビット７またはビット15）へ転送されます。
また，MSBが変化したときはVフラグがセット（１）され，元のままのときはリセット（０）されます。

○src = CLまたはsrc = imm8のとき

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）の内容のビット数分だけCYフラグを通して右シフトします。dstの内容のLSB（ビット０）のデータはCYフラグへ転送されCYフラグのデータはMSB（ビット７またはビット15）へ転送されます。

【記　述　例】　●RORC　AL, 1
●RORC　BL, 1
●RORC　CW, 1
●RORC　IX, 1

保守／廃止

【バ イ ト 数】

ニモニック	オペランド	バイト数
RORC	reg, 1	2
	mem, 1	2-4
	reg, CL	2
	mem, CL	2-4
	reg, imm8	3
	mem, imm8	3-5

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
RORC	reg, 1	1	1	0	1	0	0	0	W	1	1	0	1	1	reg		
	mem, 1	1	1	0	1	0	0	0	W	mod		0	1	1	mem		
		(disp-low)								(disp-high)							
	reg, CL	1	1	0	1	0	0	1	W	1	1	0	1	1	reg		
	mem, CL	1	1	0	1	0	0	1	W	mod		0	1	1	mem		
		(disp-low)								(disp-high)							
	reg, imm8	1	1	0	0	0	0	0	W	1	1	0	1	1	reg		
		imm8								—							
	mem, imm8	1	1	0	0	0	0	0	W	mod		0	1	1	mem		
		(disp-low)								(disp-high)							
		imm8								—							

保守／廃止

SET1

ビットのセット
Set Bit

【命 令 形 式】　① SET1 dst, src
② SET1 dst

【オペレーション】　命令形式①：dstのビットn（nはsrcで指定）←1
命令形式②：dst←1

【オ ペ ラ ン ド】　命令形式①

ニモニック	オペランド（dst, src）
SET1	reg8, CL
	mem8, CL
	reg16, CL
	mem16, CL
	reg8, imm3
	mem8, imm3
	reg16, imm4
	mem16, imm4

命令形式②

ニモニック	オペランド（dst）
SET1	CY
	DIR

【フ ラ グ】　命令形式①

V	S	Z	AC	P	CY

命令形式②（dst = CYの場合）

V	S	Z	AC	P	CY
					1

命令形式②（dst = DIRの場合）

V	DIR	S	Z	AC	P	CY
	1					

保守／廃止

【説　　明】　命令形式①：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）のビットn（nは第２オペランドで指定されるソース・オペランド（src）の内容）をセット（１）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。
オペランドがreg8, CLまたはmem8, CLのとき，CLの値は下位３ビット（0-7）のみ有効です。
オペランドがreg16, CLまたはmem16, CLのとき，CLの値は下位４ビット（0-15）のみ有効です。
オペランドがreg8, imm3のとき，命令の４バイト目のイミーディエト・データは下位３ビットのみ有効です。
オペランドがmem8, imm3のとき，命令の最終バイトのイミーディエト・データは下位３ビットのみ有効です。
オペランドがreg16, imm4のとき，命令の４バイト目のイミーディエト・データは下位４ビットのみ有効です。
オペランドがmem16, imm4のとき，命令の最終バイトのイミーディエト・データは下位４ビットのみ有効です。

命令形式②：dst = CYの場合，CYフラグをセット（１）します。
dst = DIRの場合，DIRフラグをセット（１）します。また，MOVBK, CMPBK, CMPM, LDM, STM, INM, OUTMの各命令の実行時にインデクス・レジスタ（IX, IY）をオートデクリメントするように設定します。

【記　述　例】
```
MOV   CL, 4
SET1  AL, CL
OUT   0DAH, AL
```

【バ　イ　ト　数】

ニモニック	オペランド	バイト数
SET1	reg8, CL	3
	mem8, CL	3-5
	reg16, CL	3
	mem16, CL	3-5
	reg8, imm3	4
	mem8, imm3	4-6
	reg16, imm4	4
	mem16, imm4	4-6
	CY	1
	DIR	1

保守／廃止

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
SET1	reg8, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	1	0	0
		1	1	0	0	0	reg			—							
	mem8, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	1	0	0
		mod		0	0	0	mem			(disp-low)							
		(disp-high)								—							
	reg16, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	1	0	1
		1	1	0	0	0	reg			—							
	mem16, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	1	0	1
		mod		0	0	0	mem			(disp-low)							
		(disp-high)								—							
	reg8, imm3	0	0	0	0	1	1	1	1	0	0	0	1	1	1	0	0
		1	1	0	0	0	reg			imm3							
	mem8, imm3	0	0	0	0	1	1	1	1	0	0	0	1	1	1	0	0
		mod		0	0	0	mem			(disp-low)							
		(disp-high)								imm3							
	reg16, imm4	0	0	0	0	1	1	1	1	0	0	0	1	1	1	0	1
		1	1	0	0	0	reg			imm4							
	mem16, imm4	0	0	0	0	1	1	1	1	0	0	0	1	1	1	0	1
		mod		0	0	0	mem			(disp-low)							
		(disp-high)								imm4							
	CY	1	1	1	1	1	0	0	1	—							
	DIR	1	1	1	1	1	1	0	1	—							

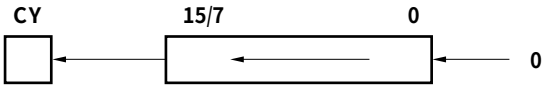
保守／廃止

SHL

左方向のシフト
Shift Left

【命 令 形 式】　SHL dst, src

【オペレーション】



【オ ペ ラ ン ド】

ニモニック	オペランド（dst, src）
SHL	reg, 1
	mem, 1
	reg, CL
	mem, CL
	reg, imm8
	mem, imm8

【フ　ラ　グ】　src = 1の場合

V	S	Z	AC	P	CY
×	×	×	U	×	×

左記以外

V	S	Z	AC	P	CY
U	×	×	U	×	×

【説　　明】　○src = 1のとき

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ビットだけ左シフトします。

dstの内容のLSB（ビット０）には０がシフト・インされ，MSB（ビット７またはビット15）のデータはCYフラグにセットされます。

シフト後，符号ビット（ビット７またはビット15）が元のままであれば，Vフラグがクリアされます。

○src = CLまたはsrc = imm8のとき

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）の内容のビット数分だけ左シフトします。１ビット，シフトするごとにdstの内容のLSB（ビット０）には０がシフト・インされ，MSB（ビット７またはビット15）のデータはCYフラグにセットされます。

【記　述　例】

```
IN      AW, 0C8H
MOV     [IY], AW
SHL     WORD PTR [IY], 12
```

保守／廃止

【バ イ ト 数】

ニモニック	オペランド	バイト数
SHL	reg, 1	2
	mem, 1	2-4
	reg, CL	2
	mem, CL	2-4
	reg, imm8	3
	mem, imm8	3-5

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
SHL	reg, 1	1	1	0	1	0	0	0	W	1	1	1	0	0	reg		
	mem, 1	1	1	0	1	0	0	0	W	mod		1	0	0	mem		
		(disp-low)								(disp-high)							
	reg, CL	1	1	0	1	0	0	1	W	1	1	1	0	0	reg		
	mem, CL	1	1	0	1	0	0	1	W	mod		1	0	0	mem		
		(disp-low)								(disp-high)							
	reg, imm8	1	1	0	0	0	0	0	W	1	1	1	0	0	reg		
		imm8								—							
	mem, imm8	1	1	0	0	0	0	0	W	mod		1	0	0	mem		
		(disp-low)								(disp-high)							
		imm8								—							

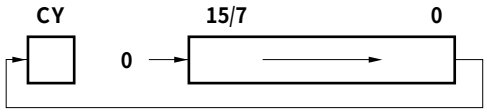
保守／廃止

SHR

右方向のシフト
Shift Right

【命 令 形 式】　SHR dst, src

【オペレーション】



【オ ペ ラ ン ド】

ニモニック	オペランド（dst, src）
SHR	reg, 1
	mem, 1
	reg, CL
	mem, CL
	reg, imm8
	mem, imm8

【フ　ラ　グ】　src = 1の場合

V	S	Z	AC	P	CY
×	×	×	U	×	×

左記以外

V	S	Z	AC	P	CY
U	×	×	U	×	×

【説　　明】　○src = 1のとき

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ビットだけ右シフトします。

dstの内容のMSB（ビット７またはビット15）には０がシフト・インされ，LSB（ビット０）のデータはCYフラグにセットされます。

シフト後，符号ビット（ビット７またはビット15）が元のままであれば，Vフラグがクリアされます。

○src = CLまたはsrc = imm8のとき

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）の内容のビット数分だけ右シフトします。１ビット，シフトするごとにdstの内容のMSB（ビット７またはビット15）には０がシフト・インされ，LSB（ビット０）のデータはCYフラグにセットされます。

【記　述　例】　●RCV：IN　AL, 0DAH

```
SHR   AL, 3
BC    RCV
```

●SHR　CW, 8

保守／廃止

【バ イ ト 数】

ニモニック	オペランド	バイト数
SHR	reg, 1	2
	mem, 1	2-4
	reg, CL	2
	mem, CL	2-4
	reg, imm8	3
	mem, imm8	3-5

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
SHR	reg, 1	1	1	0	1	0	0	0	W	1	1	1	0	1	reg		
	mem, 1	1	1	0	1	0	0	0	W	mod		1	0	1	mem		
		(disp-low)								(disp-high)							
	reg, CL	1	1	0	1	0	0	1	W	1	1	1	0	1	reg		
	mem, CL	1	1	0	1	0	0	1	W	mod		1	0	1	mem		
		(disp-low)								(disp-high)							
	reg, imm8	1	1	0	0	0	0	0	W	1	1	1	0	1	reg		
		imm8								—							
	mem, imm8	1	1	0	0	0	0	0	W	mod		1	0	1	mem		
		(disp-low)								(disp-high)							
		imm8								—							

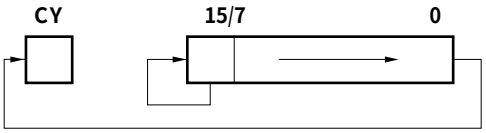
保守／廃止

SHRA

右方向の算術シフト
Shift Right Arithmetic

【命 令 形 式】　SHRA dst, src

【オペレーション】



【オ ペ ラ ン ド】

ニモニック	オペランド（dst, src）
SHRA	reg, 1
	mem, 1
	reg, CL
	mem, CL
	reg, imm8
	mem, imm8

【フ　ラ　グ】　src = 1の場合

V	S	Z	AC	P	CY
0	×	×	U	×	×

左記以外

V	S	Z	AC	P	CY
U	×	×	U	×	×

【説　　明】　○src = 1のとき

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ビットだけ算術的に右シフトします。

dstの内容のMSB（ビット７またはビット15）には元と同じ値がシフト・インされ，シフト後も符号は変化しません。LSB（ビット０）のデータはCYフラグにセットされます。

○src = CLまたはsrc = imm8のとき

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）の内容のビット数分だけ右シフトします。dstの内容のMSB（ビット７またはビット15）には元と同じ値がシフト・インされ，シフト後も符号は変化しません。LSB（ビット０）のデータはCYフラグにセットされます。

【記　述　例】

```
●MOV    CL, 2
  SHRA   BL, CL
●MOV    CL, 9
  SHRA   DW, CL
```

保守／廃止

【バ イ ト 数】

ニモニック	オペランド	バイト数
SHRA	reg, 1	2
	mem, 1	2-4
	reg, CL	2
	mem, CL	2-4
	reg, imm8	3
	mem, imm8	3-5

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
SHRA	reg, 1	1	1	0	1	0	0	0	W	1	1	1	1	1	reg		
	mem, 1	1	1	0	1	0	0	0	W	mod		1	1	1	mem		
		(disp-low)								(disp-high)							
	reg, CL	1	1	0	1	0	0	1	W	1	1	1	1	1	reg		
	mem, CL	1	1	0	1	0	0	1	W	mod		1	1	1	mem		
		(disp-low)								(disp-high)							
	reg, imm8	1	1	0	0	0	0	0	W	1	1	1	1	1	reg		
		imm8								—							
	mem, imm8	1	1	0	0	0	0	0	W	mod		1	1	1	mem		
		(disp-low)								(disp-high)							
		imm8								—							

保守／廃止

STM
STMB
STMW

ブロック・ストア
Store Multiple
Store Multiple Byte
Store Multiple Word

【命 令 形 式】　(repeat) STM［DS1-spec：］dst-block
(repeat) STMB
(repeat) STMW

【オペレーション】　［W = 0のとき］（IX）←AL
DIR = 0：IY←IY＋1
DIR = 1：IY←IY－1
［W = 1のとき］（IY＋1, IY）←AW
DIR = 0：IY←IY＋2
DIR = 1：IY←IY－2

【オ ペ ラ ン ド】

ニモニック	オペランド
STM	［DS1-spec：］dst-block
STMB	な　し
STMW	

【フ　ラ　グ】

V	S	Z	AC	P	CY

【説　　明】　ALレジスタの値，またはAWレジスタの値をIYレジスタでアドレスされるブロックにバイト／ワード・データ単位に繰り返し転送します。
IYレジスタの値は次のバイト／ワード処理のために１バイト／ワード・データが処理されるごとに自動的にインクリメント（＋1/＋2）またはデクリメント（－1/－2）されます。ブロックの方向は，DIRフラグの状態によって決定されます。
バイトかワードかの指定は，STM命令を用いる場合はオペランドの属性によって，STMB命令またはSTMW命令を用いる場合は，それぞれバイト，ワード・タイプに直接指定されます。
デスティネーション・ブロックは，常にDS1レジスタで指定されるセグメント内にロケートされる必要があり，セグメント・オーバライドはできません。

【記　述　例】　●REP　STM DS1：WORD_VAR　：DS1セグメント
●REP　STMB　　：DS1セグメント

【バ イ ト 数】　１

保守／廃止

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード							
		7	6	5	4	3	2	1	0
STM	［DS1-spec：］dst-block	1	0	1	0	1	0	1	W
STMB	なし								
STMW									

保守／廃止

STOP ［V20, V30に対して追加］

CPU制御
Stop

【命 令 形 式】　STOP

【オペレーション】　CPU Stop

【オ ペ ラ ン ド】

ニモニック	オペランド
STOP	な　し

【フ　ラ　グ】

V	S	Z	AC	P	CY

【説　　明】　ストップ状態にします。

【記　述　例】　STOP

【バ イ ト 数】　2

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
STOP	な　し	0	0	0	0	1	1	1	1	1	0	0	1	1	1	1	0

保守／廃止

SUB

減算
Subtract

【命 令 形 式】　SUB dst, src

【オペランド，オペレーション】

ニモニック	オペランド（dst, src）	オペレーション
SUB	reg, reg'	dst←dst－src
	mem, reg	
	reg, mem	
	reg, imm	
	mem, imm	
	acc, imm	［W = 0のとき］AL←AL－imm8 ［W = 1のとき］AW←AW－imm16

【フ　ラ　グ】

V	S	Z	AC	P	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）の内容から第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。

【記　述　例】　DLレジスタの内容からメモリ０：50Hの内容を減算してDLレジスタへストアする。

```
MOV   AW, 0
MOV   DS0, AW
MOV   IX, 50H
SUB   DL, DS0：BYTE PTR［IX］
```

保守／廃止

【バ イ ト 数】

ニモニック	オペランド	バイト数
SUB	reg, reg'	2
	mem, reg	2-4
	reg, mem	2-4
	reg, imm	3, 4
	mem, imm	3-6
	acc, imm	2, 3

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
SUB	reg, reg'	0	0	1	0	1	0	1	W	1	1	reg			reg'		
	mem, reg	0	0	1	0	1	0	0	W	mod		reg			mem		
		(disp-low)								(disp-high)							
	reg, mem	0	0	1	0	1	0	1	W	mod		reg			mem		
		(disp-low)								(disp-high)							
	reg, imm	1	0	0	0	0	0	s	W	1	1	1	0	1	reg		
		imm8 or imm16-low								imm16-high							
	mem, imm	1	0	0	0	0	0	s	W	mod		1	0	1	mem		
		(disp-low)								(disp-high)							
		imm8 or imm16-low								imm16-high							
	acc, imm	0	0	1	0	1	1	0	W	imm8 or imm16-low							
		imm16-high								—							

保守／廃止

SUB4S

10進減算
Subtract Nibble String

【命 令 形 式】　SUB4S［DS1-spec：］dst-string,［Seg-spec：］src-string
SUB4S

【オペレーション】　BCDストリング（IY, CL）←BCDストリング（IY, CL）－BCDストリング（IX, CL）

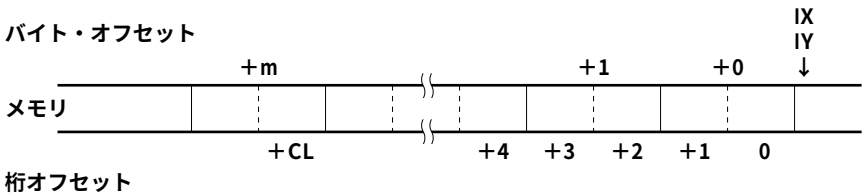
【オ ペ ラ ン ド】

ニモニック	オペランド（dst, src）
SUB4S	［DS1-spec：］dst-string,［Seg-spec：］src-string
	な　し

【フ　ラ　グ】

V	S	Z	AC	P	CY
U	U	×	U	U	×

【説　　明】　IYレジスタでアドレスされるパックトBCDストリングからIXレジスタでアドレスされるパックトBCDストリングを減算し，結果をIYレジスタでアドレスされるストリングにストアします。ストリング長（BCD桁数）は，CLレジスタ（CLの内容がdならばd桁）によって決定され，1-254桁まで可能です。
デスティネーション・ストリングは，常にDS1レジスタで指されるセグメント内にロケートされる必要があり，セグメント・オーバライドはできません。一方，ソース・ストリングは，ディフォールト・セグメント・レジスタはDS0レジスタとなっていますが，セグメント・オーバライド可能で，任意のセグメント・レジスタで指定されるセグメント内にロケートできます。
パックトBCDストリングのフォーマットを次に示します。



注意　BCDストリング命令は常に偶数桁単位で動作します。このため桁数として偶数を指定したときは演算結果，および各フラグは正しく動きますが，桁数として奇数を指定した場合は奇数＋１の偶数桁演算を実行し，演算結果，および各フラグは偶数桁の演算結果を示しますので，桁数として奇数を指定する場合は次のようにしてください。
桁数が奇数のときのBCD減算命令は，最上位バイトの上位４ビットを“０”にしてから実行してください。この結果ボローが生じた場合は最上位バイトの上位４ビットは“９”になります。

保守／廃止

【記　述　例】　MOV　IX, OFFSET VAR_1
MOV　IY OFFSET VAR_2
MOV　CL, 4
SUB4S

【バ イ ト 数】　2

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
SUB4S	［DS1-spec：］dst-string, ［Seg-spec：］src-string	0	0	0	0	1	1	1	1	0	0	1	0	0	0	1	0
	な　し																

保守／廃止

SUBC

キャリーを含む減算
Subtract with Carry

【命 令 形 式】　SUBC dst, src

【オペランド，オペレーション】

ニモニック	オペランド（dst, src）	オペレーション
SUBC	reg, reg'	dst←dst－src－CY
	mem, reg	
	reg, mem	
	reg, imm	
	mem, imm	
	acc, imm	［W = 0のとき］AL←AL－imm8－CY ［W = 1のとき］AW←AW－imm16－CY

【フ ラ グ】

V	S	Z	AC	P	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）の内容から第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CYフラグの内容も含めて減算し，結果をデスティネーション・オペランド（dst）に格納します。

【記 述 例】　SUBC　DL, BYTE PTR［IX］

【バ イ ト 数】

ニモニック	オペランド	バイト数
SUBC	reg, reg'	2
	mem, reg	2-4
	reg, mem	2-4
	reg, imm	3, 4
	mem, imm	3-6
	acc, imm	2, 3

保守／廃止

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
SUBC	reg, reg'	0	0	0	1	1	0	1	W	1	1	reg			reg'		
	mem, reg	0	0	0	1	1	0	0	W	mod		reg			mem		
		(disp-low)								(disp-high)							
	reg, mem	0	0	0	1	1	0	1	W	mod		reg			mem		
		(disp-low)								(disp-high)							
	reg, imm	1	0	0	0	0	0	s	W	1	1	0	1	1	reg		
		imm8 or imm16-low								imm16-high							
	mem, imm	1	0	0	0	0	0	s	W	mod		0	1	1	mem		
		(disp-low)								(disp-high)							
		imm8 or imm16-low								imm16-high							
	acc, imm	0	0	0	1	1	1	0	W	imm8 or imm16-low							
		imm16-high								—							

保守／廃止

TEST

テスト
Test

【命 令 形 式】　TEST dst, src

【オペランド，オペレーション】

ニモニック	オペランド（dst, src）	オペレーション
TEST	reg, reg'	dst∧src
	mem, reg	
	reg, mem	
	reg, imm	
	mem, imm	
	acc, imm	［W = 0のとき］AL∧imm8 ［W = 1のとき］AW∧imm16

【フ ラ グ】

V	S	Z	AC	P	CY
0	×	×	U	×	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）と第２オペランドで指定されるソース・オペランド（src）の論理積（AND）をとります。
結果はどこへも格納せず，フラグを変化させます。

【記 述 例】　IN　AL, 0D8H
TEST AL, ‘ A ’

【バ イ ト 数】

ニモニック	オペランド	バイト数
TEST	reg, reg'	2
	mem, reg	2-4
	reg, mem	
	reg, imm	3, 4
	mem, imm	3-6
	acc, imm	2, 3

保守／廃止

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
TEST	reg, reg'	1	0	0	0	0	1	0	W	1	1	reg'			reg		
	mem, reg	1	0	0	0	0	1	0	W	mod		reg			mem		
		(disp-low)								(disp-high)							
	reg, mem	1	0	0	0	0	1	0	W	mod		reg			mem		
		(disp-low)								(disp-high)							
	reg, imm	1	1	1	1	0	1	1	W	1	1	0	0	0	reg		
		imm8 or imm16-low								imm16-high							
	mem, imm	1	1	1	1	0	1	1	W	mod		0	0	0	mem		
		(disp-low)								(disp-high)							
		imm8 or imm16-low								imm16-high							
	acc, imm	1	0	1	0	1	0	0	W	imm8 or imm16-low							
		imm16-high								—							

保守／廃止

TEST1

ビットのテスト
Test Bit

【命 令 形 式】　TEST1 dst, src

【オペレーション】　dstのビットn = 0（nはsrcで指定）のとき：Z←1
dstのビットn = 1（nはsrcで指定）のとき：Z←0

【オ ペ ラ ン ド】

ニモニック	オペランド（dst, src）
TEST1	reg8, CL
	mem8, CL
	reg16, CL
	mem16, CL
	reg8, imm3
	mem8, imm3
	reg16, imm4
	mem16, imm4

【フ ラ グ】

V	S	Z	AC	P	CY
0	U	×	U	U	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）のビットn（nは第２オペランドで指定されるソース・オペランド（src）の内容）が０のときはZフラグがセット（１）され，ビットnが１のときはリセット（０）されます。
オペランドがreg8, CLまたはmem8, CLのとき，CLの値は下位３ビット（0-7）のみ有効です。
オペランドがreg16, CLまたはmem16, CLのとき，CLの値は下位４ビット（0-15）のみ有効です。
オペランドがreg8, imm3のとき，命令の４バイト目のイミーディエト・データは下位３ビットのみ有効です。
オペランドがmem8, imm3のとき，命令の最終バイトのイミーディエト・データは下位３ビットのみ有効です。
オペランドがreg16, imm4のとき，命令の４バイト目のイミーディエト・データは下位４ビットのみ有効です。
オペランドがmem16, imm4のとき，命令の最終バイトのイミーディエト・データは下位４ビットのみ有効です。

【記　述　例】
```
MOV    CL, 01
IN     AL, 0DAH
TEST1  AL, CL；ビット１をテスト
```

保守／廃止

【バ イ ト 数】

ニモニック	オペランド	バイト数
TEST1	reg8, CL	3
	mem8, CL	3-5
	reg16, CL	3
	mem16, CL	3-5
	reg8, imm3	4
	mem8, imm3	4-6
	reg16, imm4	4
	mem16, imm4	4-6

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
TEST1	reg8, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	0	0	0
		1	1	0	0	0	reg			—							
	mem8, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	0	0	0
		mod		0	0	0	mem			(disp-low)							
		(disp-high)								—							
	reg16, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	0	0	1
		1	1	0	0	0	reg			—							
	mem16, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	0	0	1
		mod		0	0	0	mem			(disp-low)							
		(disp-high)								—							
	reg8, imm3	0	0	0	0	1	1	1	1	0	0	0	1	1	0	0	0
		1	1	0	0	0	reg			imm3							
	mem8, imm3	0	0	0	0	1	1	1	1	0	0	0	1	1	0	0	0
		mod		0	0	0	mem			(disp-low)							
		(disp-high)								imm3							
	reg16, imm4	0	0	0	0	1	1	1	1	0	0	0	1	1	0	0	1
		1	1	0	0	0	reg			imm4							
	mem16, imm4	0	0	0	0	1	1	1	1	0	0	0	1	1	0	0	1
		mod		0	0	0	mem			(disp-low)							
		(disp-high)								imm4							

保守／廃止

TRANS
TRANSB

変換テーブルの転送
Translate
Translate Byte

【命 令 形 式】　TRANS src-table
TRANS
TRANSB

【オペレーション】　AL←（BW＋AL）

【オ ペ ラ ン ド】

ニモニック	オペランド
TRANS	src-table
	な　し
TRANSB	な　し

【フ　ラ　グ】

V	S	Z	AC	P	CY

【説　　明】　BWレジスタとALレジスタによってアドレスされる256バイトの変換テーブルの１バイトをALレジスタに転送します。このときBWレジスタはテーブルの先頭アドレスを指し，先頭アドレスから256バイト内のオフセット値をALレジスタが指定します。

【記　述　例】　TRANS　SIN_TBL

【バ イ ト 数】　１

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード							
		7	6	5	4	3	2	1	0
TRANS	src-table	1	1	0	1	0	1	1	1
	な　し								
TRANSB	な　し								

保守／廃止

TSKSW 【V20, V30に対して追加】

レジスタ・バンク切り替え
Task Switch

【命 令 形 式】　TSKSW reg16

【オペレーション】　RB0-2←reg16の下位３ビット
旧レジスタ・バンクのPSW退避，PC退避←PSW, PC
PSW, PC←新レジスタ・バンクのPSW退避，PC退避

【オ ペ ラ ン ド】

ニモニック	オペランド
TSKSW	reg16

【フ　ラ　グ】

RB2	RB1	RB0	V	DIR	IE	BRK	S	Z	F1	AC	F0	P	$\overline{\text{IBRK}}$	CY
×	×	×	×	×	×	×	×	×	×	×	×	×	×	×

【説　　明】　レジスタ・バンクを切り替える命令です。高速なタスク切り替えなどに使用します。現在のレジスタ・バンク内のPC退避エリアにPCの内容を退避します。同じように，PSW退避エリアにPSWの内容を退避します。

【記　述　例】　TSKSW　BP

【バ イ ト 数】　３

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
TSKSW	reg16	0	0	0	0	1	1	1	1	1	0	0	1	0	1	0	0
		1	1	1	1	1	reg			−							

保守／廃止

XCH

データ交換
Exchange

【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	reg, reg'
	mem, reg
	reg, mem
	AW, reg16
	reg16, AW

【フ　ラ　グ】

V	S	Z	AC	P	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）の内容を交換します。

【記　述　例】
```
MOV　AW，100H
MOV　BW，50H
XCH　AW，BW
；AW = 50H, BW = 100Hとなる。
```

【バ イ ト 数】

ニモニック	オペランド	バイト数
XCH	reg, reg'	2
	mem, reg	2-4
	reg, mem	
	AW, reg16	1
	reg16, AW	

保守／廃止

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
XCH	reg, reg'	1	0	0	0	0	1	1	W	1	1	reg			reg'		
	mem, reg	1	0	0	0	0	1	1	W	mod		reg			mem		
		(disp-low)								(disp-high)							
	reg, mem	1	0	0	0	0	1	1	W	mod		reg			mem		
		(disp-low)								(disp-high)							
	AW, reg16	1	0	0	1	0	reg			—							
	reg16, AW	1	0	0	1	0	reg			—							

保守／廃止

XOR

排他的論理和
Exclusive Or

【命 令 形 式】　XOR dst, src

【オペランド，オペレーション】

ニモニック	オペランド（dst, src）	オペレーション
XOR	reg, reg'	dst←dst▽src
	mem, reg	
	reg, mem	
	reg, imm	
	mem, imm	
	acc, imm	［W = 0のとき］AL←AL▽imm8 ［W = 1のとき］AW←AW▽imm16

【フ ラ グ】

V	S	Z	AC	P	CY
0	×	×	U	×	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）と第２オペランドで指定されるソース・オペランド（src）の排他的論理和（XOR）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。

【記 述 例】　●XOR　CL，DL
●XOR　CW，CW；CWレジスタのクリア
●XOR　AW，DW

【バ イ ト 数】

ニモニック	オペランド	バイト数
XOR	reg, reg'	2
	mem, reg	2-4
	reg, mem	2-4
	reg, imm	3, 4
	mem, imm	3-6
	acc, imm	2, 3

保守／廃止

【命 令 語 形 式】

ニモニック	オペランド	オペレーション・コード															
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
XOR	reg, reg'	0	0	1	1	0	0	1	W	1	1	reg			reg'		
	mem, reg	0	0	1	1	0	0	0	W	mod		reg			mem		
		(disp-low)								(disp-high)							
	reg, mem	0	0	1	1	0	0	1	W	mod		reg			mem		
		(disp-low)								(disp-high)							
	reg, imm[注]	1	0	0	0	0	0	0	W	1	1	1	1	0	reg		
		imm8 or imm16-low								imm16-high							
	mem, imm	1	0	0	0	0	0	0	W	mod		1	1	0	mem		
		(disp-low)								(disp-high)							
		imm8 or imm16-low								imm16-high							
	acc, imm	0	0	1	1	0	1	0	W	imm8 or imm16-low							
		imm16-high								—							

注 アセンブラ，コンパイラによっては，次に示すようなコードが生成されることがあります。

オペレーション・コード															
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
1	0	0	0	0	0	1	W	1	1	1	1	0	reg		
imm8								—							

このような場合でも正常に命令を実行します。ただし，エミュレータによっては，この命令に対する逆アセンブラ機能，ライン・アセンブラ機能がサポートされていない場合がありますのでご注意ください。

保守／廃止

2.2　命令実行クロック数

各命令の実行クロック数とワード転送回数をニモニックのアルファベット順に表２－８に示します

（1）クロック数

表の中で示されている数値は，実行ユニットが命令実行に必要とする時間で，次の条件に基づいています。

（a）プリフェッチ時間，プリデコード時間，バス使用のための待ち時間などは含みません。

（b）命令コードのフェッチ状態，パイプラインの処理状態によってクロック数は長くなることがあります。**付録C　プログラムの実行クロック数**を参照してください。

（c）I/Oアクセスは０ウエイトを想定しています。

（d）プリミティブ・ブロック転送命令，プリミティブ入出力命令はリピート・プリフィクスも含んでいます。

（e）奇数アドレスへのワード・データの場合，バス・サイクルを２回起動します。

（f）外部データ・バス幅は次のとおりです。

●V25, V25＋：８ビット

●V35, V35＋：16ビット

（g）メモリ・オペランドの場合，クロック数がアドレシング・モードによって異なります。“ EA ”と表記しているものについては次に示す数値を適用してください。

mod / mem	00	クロック数	01	クロック数	10	クロック数
000	BW＋IX	3	BW＋IX＋disp8	3	BW＋IX＋disp16	4
001	BW＋IY	3	BW＋IY＋disp8	3	BW＋IY＋disp16	4
010	BP＋IX	3	BP＋IX＋disp8	3	BP＋IX＋disp16	4
011	BP＋IY	3	BP＋IY＋disp8	3	BP＋IY＋disp16	4
100	IX	3	IX＋disp8	3	IX＋disp16	4
101	IY	3	IY＋disp8	3	IY＋disp16	4
110	ダイレクト・アドレス	3	BP＋disp8	3	BP＋disp16	4
111	BW	3	BW＋disp8	3	BW＋disp16	4

（h）“Ｔ”と表記しているものは，ウエイト・ステート数を示しています。“０”（ウエイトなし）から任意のウエイト・ステート数を適用してください。

（i）ブランチ命令でクロック数は次のように示します。

／の左側：ブランチするときのクロック数

／の右側：ブランチしないときのクロック数

（2）ワード転送回数

ワード転送回数とは命令実行によって発生するワード・データ（16ビット）のバス・アクセスを必要とする回数です。

表２－８　命令実行クロック数一覧（1/15）

ニモニック	オペランド	ワード転送回数	V25, V25＋				V35, V35＋			
			バイト処理		ワード処理		バイト処理		ワード処理	
			内蔵RAMアクセス許可	内蔵RAMアクセス禁止	内蔵RAMアクセス許可	内蔵RAMアクセス禁止	内蔵RAMアクセス許可	内蔵RAMアクセス禁止	内蔵RAMアクセス許可	内蔵RAMアクセス禁止
ADD	reg, reg ′	0	2	2	2	2	2	2	2	2
	mem, reg	2	EA＋8＋2・T	EA＋6＋T	EA＋12＋4・T	EA＋8＋2・T	EA＋10＋2・T	EA＋7＋T	EA＋10＋2・T	EA＋7＋T
	reg, mem	1	EA＋6＋T	EA＋6＋T	EA＋8＋2・T	EA＋8＋2・T	EA＋7＋T	EA＋7＋T	EA＋7＋T	EA＋7＋T
	reg, imm	0	5	5	6	6	5	5	6	6
	mem, imm	2	EA＋9＋2・T	EA＋7＋2・T	EA＋14＋4・T	EA＋10＋4・T	EA＋11＋2・T	EA＋9＋2・T	EA＋12＋2・T	EA＋8＋2・T
	acc, imm	0	5	5	6	6	5	5	6	6
ADD4S[注]	［DS1-spec：］dst-string, ［Seg-spec：］src-string	0	22＋(27＋3・T)・m	22＋(25＋3・T)・m	－	－	22＋(30＋3・T)・m	22＋(28＋3・T)・m	－	－
	なし	0	22＋(27＋3・T)・m	22＋(25＋3・T)・m	－	－	22＋(30＋3・T)・m	22＋(28＋3・T)・m	－	－
ADDC	reg, reg'	0	2	2	2	2	2	2	2	2
	mem, reg	2	EA＋8＋2・T	EA＋6＋T	EA＋12＋4・T	EA＋8＋2・T	EA＋10＋2・T	EA＋7＋T	EA＋10＋2・T	EA＋7＋T
	reg, mem	1	EA＋6＋T	EA＋6＋T	EA＋8＋2・T	EA＋8＋2・T	EA＋7＋T	EA＋7＋T	EA＋7＋T	EA＋7＋T
	reg, imm	0	5	5	6	6	5	5	6	6
	mem, imm	2	EA＋9＋2・T	EA＋7＋2・T	EA＋14＋4・T	EA＋10＋4・T	EA＋10＋2・T	EA＋9＋2・T	EA＋12＋2・T	EA＋8＋2・T
	acc, imm	0	5	5	6	6	5	5	6	6
ADJ4A	なし	0	9	9	－	－	9	9	－	－
ADJ4S	なし	0	9	9	－	－	9	9	－	－
ADJBA	なし	0	17	17	－	－	17	17	－	－
ADJBS	なし	0	17	17	－	－	17	17	－	－
AND	reg, reg'	0	2	2	2	2	2	2	2	2
	mem, reg	2	EA＋8＋2・T	EA＋6＋T	EA＋12＋4・T	EA＋8＋2・T	EA＋10＋2・T	EA＋7＋T	EA＋10＋2・T	EA＋7＋T
	reg, mem	1	EA＋6＋T	EA＋6＋T	EA＋8＋2・T	EA＋8＋2・T	EA＋7＋T	EA＋7＋T	EA＋7＋T	EA＋7＋T
	reg, imm	0	5	5	6	6	5	5	6	6
	mem, imm	2	EA＋9＋2・T	EA＋7＋2・T	EA＋14＋4・T	EA＋10＋4・T	EA＋11＋2・T	EA＋9＋2・T	EA＋12＋2・T	EA＋8＋2・T
	acc, imm	0	5	5	6	6	5	5	6	6

注　m：BCD桁数×1/2

表２－８　命令実行クロック数一覧（2/15）

ニモニック	オペランド	ワード転送回数	V25, V25＋				V35, V35＋			
			バイト処理		ワード処理		バイト処理		ワード処理	
			内蔵RAMアクセス許可	内蔵RAMアクセス禁止	内蔵RAMアクセス許可	内蔵RAMアクセス禁止	内蔵RAMアクセス許可	内蔵RAMアクセス禁止	内蔵RAMアクセス許可	内蔵RAMアクセス禁止
BC	short-label	0	－	－	15/8	15/8	－	－	15/8	15/8
BCWZ	short-label	0	－	－	15/8	15/8	－	－	15/8	15/8
BE	short-label	0	－	－	15/8	15/8	－	－	15/8	15/8
BGE	short-label	0	－	－	15/8	15/8	－	－	15/8	15/8
BGT	short-label	0	－	－	15/8	15/8	－	－	15/8	15/8
BH	short-label	0	－	－	15/8	15/8	－	－	15/8	15/8
BL	short-label	0	－	－	15/8	15/8	－	－	15/8	15/8
BLE	short-label	0	－	－	15/8	15/8	－	－	15/8	15/8
BLT	short-label	0	－	－	15/8	15/8	－	－	15/8	15/8
BN	short-label	0	－	－	15/8	15/8	－	－	15/8	15/8
BNC	short-label	0	－	－	15/8	15/8	－	－	15/8	15/8
BNE	short-label	0	－	－	15/8	15/8	－	－	15/8	15/8
BNH	short-label	0	－	－	15/8	15/8	－	－	15/8	15/8
BNL	short-label	0	－	－	15/8	15/8	－	－	15/8	15/8
BNV	short-label	0	－	－	15/8	15/8	－	－	15/8	15/8
BNZ	short-label	0	－	－	15/8	15/8	－	－	15/8	15/8
BP	short-label	0	－	－	15/8	15/8	－	－	15/8	15/8
BPE	short-label	0	－	－	15/8	15/8	－	－	15/8	15/8
BPO	short-label	0	－	－	15/8	15/8	－	－	15/8	15/8
BR	near-label	0	－	－	12	12	－	－	12	12
	short-label	0	－	－	12	12	－	－	12	12
	regptr16	0	－	－	13	13	－	－	13	13
	memptr16	1	－	－	EA＋17＋2･T	EA＋17＋2･T	－	－	EA＋16＋T	EA＋16＋T
	far-label	0	－	－	15	15	－	－	15	15
	memptr32	2	－	－	EA＋25＋4・T	EA＋25＋4・T	－	－	EA＋23＋2・T	EA＋23＋2・T

表２－８　命令実行クロック数一覧（3/15）

ニモニック	オペランド	ワード転送回数	V25, V25＋				V35, V35＋			
			バイト処理		ワード処理		バイト処理		ワード処理	
			内蔵RAMアクセス許可	内蔵RAMアクセス禁止	内蔵RAMアクセス許可	内蔵RAMアクセス禁止	内蔵RAMアクセス許可	内蔵RAMアクセス禁止	内蔵RAMアクセス許可	内蔵RAMアクセス禁止
BRK	3	5	－	－	55＋10・T	43＋10・T	－	－	50＋5・T	38＋5・T
	imm8（≠3）	5	2	2	56＋10・T	44＋10・T	2	2	51＋5・T	39＋5・T
BRKCS	reg16	0	－	－	15	15	－	－	15	15
BRKV	なし	5	－	－	55＋10・T	43＋10・T	－	－	50＋5・T	38＋5・T
BTCLR	sfr, imm3, short-label	0	29/21	29/21	－	－	29/21	29/21	－	－
BUSLOCK	なし	0	2	2	2	2	2	2	2	2
BV	short-label	0	－	－	15/8	15/8	－	－	15/8	15/8
BZ	short-label	0	－	－	15/8	15/8	－	－	15/8	15/8
CALL	near-proc	1	－	－	22＋2・T	18＋2・T	－	－	21＋T	17＋T
	regptr16	1	－	－	22＋2・T	18＋2・T	－	－	21＋T	17＋T
	memptr16	2	－	－	EA＋26＋4・T	EA＋24＋4・T	－	－	EA＋24＋2・T	EA＋22＋2・T
	far-proc	2	－	－	38＋4・T	34＋4・T	－	－	36＋2・T	32＋2・T
	memptr32	4	－	－	EA＋36＋8・T	EA＋24＋8・T	－	－	EA＋32＋4・T	EA＋20＋4・T
CHKIND	reg16, mem32（割り込み条件成立のとき）	7	－	－	EA＋26＋4・T	EA＋26＋4・T	－	－	EA＋24＋2・T	EA＋24＋2・T
	reg16, mem32（割り込み条件不成立のとき）	2	－	－	EA＋26＋4・T	EA＋26＋4・T	－	－	EA＋24＋2・T	EA＋24＋2・T
CLR1	reg8, CL	0	8	8	－	－	8	8	－	－
	mem8, CL	0	EA＋14＋2・T	EA＋12＋T	－	－	EA＋16＋2·T	EA＋13＋T	－	－
	reg16, CL	0	－	－	8	8	－	－	8	8
	mem16, CL	2	－	－	EA＋18＋4・T	EA＋14＋2・T	－	－	EA＋16＋2・T	EA＋13＋T
	reg8, imm3	0	7	7	－	－	7	7	－	－
	mem8, imm3	0	EA＋11＋2・T	EA＋9＋T	－	－	EA＋13＋2·T	EA＋10＋T	－	－
	reg16, imm4	0	－	－	7	7	－	－	7	7
	mem16, imm4	3	－	－	EA＋15＋4・T	EA＋10＋2・T	－	－	EA＋13＋2・T	EA＋9＋T

表２－８　命令実行クロック数一覧（4/15）

ニモニック	オペランド	ワード転送回数	V25, V25＋				V35, V35＋			
			バイト処理		ワード処理		バイト処理		ワード処理	
			内蔵RAMアクセス許可	内蔵RAMアクセス禁止	内蔵RAMアクセス許可	内蔵RAMアクセス禁止	内蔵RAMアクセス許可	内蔵RAMアクセス禁止	内蔵RAMアクセス許可	内蔵RAMアクセス禁止
CLR1	CY	0	2	2	2	2	2	2	2	2
	DIR	0	2	2	2	2	2	2	2	2
CMP	reg, reg ′	0	2	2	2	2	2	2	2	2
	mem, reg	1	EA＋6＋T	EA＋6＋T	EA＋8＋2・T	EA＋8＋2・T	EA＋7＋T	EA＋7＋T	EA＋7＋T	EA＋7＋T
	reg, mem	1	EA＋6＋T	EA＋6＋T	EA＋8＋2・T	EA＋8＋2・T	EA＋7＋T	EA＋7＋T	EA＋7＋T	EA＋7＋T
	reg, imm	0	5	5	6	6	5	5	6	6
	mem, imm	1	EA＋7＋T	EA＋7＋T	EA＋10＋2・T	EA＋10＋2・T	EA＋8＋T	EA＋8＋T	EA＋9＋T	EA＋9＋T
	acc, imm	0	5	5	6	6	5	5	6	6
CMP4S[注1]	［DS1-spec：］dst-string, ［Seg-spec：］src-string	0	22＋(23＋3・T)・m	22＋(23＋2・T)・m	－	－	22＋(25＋2・T)・m	22＋(25＋2・T)・m	－	－
	なし	0	22＋(23＋3・T)・m	22＋(23＋2・T)・m	－	－	22＋(25＋2・T)・m	22＋(25＋2·T)・m	－	－
CMPBK[注2]	［Seg-spec：］src-block, ［DS1-spec：］dst-block	2×rep	16＋(21＋2·T)・n	16＋(21＋2・T)・n	16＋(25＋4・T)・n	16＋(25＋4・T)・n	16＋(23＋2・T)・n	16＋(23＋2·T)・n	16＋(23＋2·T)・n	16＋(23＋2・T)・n
		(2)[注3]	23＋2・T	19＋2・T	27＋4・T	21＋4・T	25＋2・T	21＋2・T	25＋2・T	19＋2・T
CMPBKB[注2]	なし	2×rep	16＋(21＋2・T)・n	16＋(21＋2·T)・n	16＋(25＋4・T)・n	16＋(25＋4・T)・n	16＋(23＋2・T)・n	16＋(23＋2·T)・n	16＋(23＋2·T)・n	16＋(23＋2·T)・n
		(2)[注3]	23＋2・T	19＋2・T	27＋4・T	21＋4·T	25＋2・T	21＋2・T	25＋2・T	19＋2・T
CMPBKW[注2]	なし	2×rep	16＋(21＋2・T)・n	16＋(21＋2・T)・n	16＋(25＋4・T)・n	16＋(25＋4・T)・n	16＋(23＋2・T)・n	16＋(23＋2·T)・n	16＋(23＋2·T)・n	16＋(23＋2·T)・n
		(2)[注3]	23＋2・T	19＋2・T	27＋4・T	21＋4·T	25＋2・T	21＋2・T	25＋2・T	19＋2・T
CMPM[注2]	［DS1-spec：］dst-block	1×rep	16＋(15＋T)・n	16＋(15＋T)・n	16＋(17＋2・T)・n	16＋(17＋2・T)・n	16＋(16＋T)・n	16＋(16＋T)・n	16＋(16＋T)・n	16＋(16＋T)・n
		(1)[注3]	17＋T	17＋T	19＋2・T	19＋2・T	18＋T	18＋T	19＋2・T	19＋2・T
CMPMB[注2]	なし	1×rep	16＋(15＋T)・n	16＋(15＋T)・n	16＋(17＋2・T)・n	16＋(17＋2・T)・n	16＋(16＋T)・n	16＋(16＋T)・n	16＋(16＋T)・n	16＋(16＋T)・n
		(1)[注3]	17＋T	17＋T	19＋2・T	19＋2・T	18＋T	18＋T	19＋2・T	19＋2·T
CMPMW[注2]	なし	1×rep	16＋(15＋T)・n	16＋(15＋T)・n	16＋(17＋2・T)・n	16＋(17＋2・T)・n	16＋(16＋T)・n	16＋(16＋T)・n	16＋(16＋T)・n	16＋(16＋T)・n
		(1)[注3]	17＋T	17＋T	19＋2・T	19＋2・T	18＋T	18＋T	19＋2・T	19＋2·T

注１．m：BCD桁数×1/2

２．n：リピート数（n≧1）

３．（ ）：一回だけの処理に適用

表２－８　命令実行クロック数一覧（5/15）

ニモニック	オペランド	ワード転送回数	V25, V25＋				V35, V35＋			
			バイト処理		ワード処理		バイト処理		ワード処理	
			内蔵RAMアクセス許可	内蔵RAMアクセス禁止	内蔵RAMアクセス許可	内蔵RAMアクセス禁止	内蔵RAMアクセス許可	内蔵RAMアクセス禁止	内蔵RAMアクセス許可	内蔵RAMアクセス禁止
CVTBD	なし	0	19	19	—	—	19	19	—	—
CVTBW	なし	0	3	3	—	—	3	3	—	—
CVTDB	なし	0	20	20	—	—	20	20	—	—
CVTWL	なし	0	—	—	8	8	—	—	8	8
DBNZ	short-label	0	—	—	17/8	17/8	—	—	17/8	17/8
DBNZE	short-label	0	—	—	17/8	17/8	—	—	17/8	17/8
DBNZNE	short-label	0	—	—	17/8	17/8	—	—	17/8	17/8
DEC	reg8	0	5	5	—	—	5	5	—	—
	mem	2	EA＋11＋2・T	EA＋9＋2・T	EA＋15＋4・T	EA＋11＋4・T	EA＋13＋2・T	EA＋11＋2･T	EA＋13＋2・T	EA＋9＋2･T
	reg16	0	—	—	2	2	—	—	2	2
DI	なし	0	4	4	4	4	4	4	4	4
DISPOSE	なし	1	—	—	12＋2・T	12＋2・T	—	—	11＋T	11＋T
DIV	reg8	0	45-56	45-56	—	—	45-56	45-56	—	—
	mem8	0	EA＋48＋T ～EA＋58＋T	EA＋48＋T ～EA＋58＋T	—	—	EA＋49＋T ～EA＋59＋T	EA＋49＋T ～EA＋59＋T	—	—
	reg16	0	—	—	54-64	54-64	—	—	54-64	54-64
	mem16	1	—	—	EA＋58＋2・T ～EA＋68＋2・T	EA＋58＋2・T ～EA＋68＋2・T	—	—	EA＋57＋T ～EA＋67＋T	EA＋57＋T ～EA＋67＋T
DIVU	reg8	0	31	31	—	—	31	31	—	—
	mem8	0	EA＋33＋T	EA＋33＋T	—	—	EA＋34＋T	EA＋34＋T	—	—
	reg16	0	—	—	39	39	—	—	39	39
	mem16	1	—	—	EA＋43＋2･T	EA＋43＋2･T	—	—	EA＋43＋2・T	EA＋43＋2・T
DS0：	なし	0	2	2	2	2	2	2	2	2
DS1：	なし	0	2	2	2	2	2	2	2	2
EI	なし	0	12	12	12	12	12	12	12	12

表２－８　命令実行クロック数一覧（6/15）

ニモニック	オペランド	ワード転送回数	V25, V25＋				V35, V35＋			
			バイト処理		ワード処理		バイト処理		ワード処理	
			内蔵RAMアクセス許可	内蔵RAMアクセス禁止	内蔵RAMアクセス許可	内蔵RAMアクセス禁止	内蔵RAMアクセス許可	内蔵RAMアクセス禁止	内蔵RAMアクセス許可	内蔵RAMアクセス禁止
EXT	reg8, reg8'	１または２	41-121（ビット長によって異なります）							
	reg8, imm4	１または２	42-122（ビット長によって異なります）							
FINT	なし	0	2	2	2	2	2	2	2	2
FPO1	fp-op	5	－	－	60＋10・T	48＋10・T	－	－	55＋5·T	43＋5·T
	fp-op, mem	5	－	－	60＋10・T	48＋10・T	－	－	55＋5·T	43＋5·T
FPO2	fp-op	5	－	－	60＋10・T	48＋10・T	－	－	55＋5·T	43＋5·T
	fp-op, mem	5	－	－	60＋10・T	48＋10・T	－	－	55＋5·T	43＋5·T
HALT	なし	0	－	－	－	－	－	－	－	－
IN[注1]	acc, imm8	1	14＋T	14＋T	16＋2・T	16＋2・T	15＋T	15＋T	15＋T	15＋T
	acc, DW	1	13＋T	13＋T	15＋2・T	15＋2・T	14＋T	14＋T	14＋T	14＋T
INC	reg8	0	5	5	－	－	5	5	－	－
	mem	2	EA＋11＋2・T	EA＋9＋2・T	EA＋15＋4・T	EA＋11＋4・T	EA＋13＋2・T	EA＋11＋2・T	EA＋13＋2・T	EA＋9＋2・T
	reg16	0	－	－	2	2	－	－	2	2
INM[注1，2]	[DS1-spec：]dst-block, DW	2×rep	18＋(13＋2・T)・n	18＋(11＋2・T)・n	18＋(15＋4・T)・n	18＋(11＋4・T)・n	18＋(15＋2・T)・n	18＋(13＋2・T)・n	18＋(13＋2・T)・n	18＋(9＋2・T)・n
		(2)[注3]	19＋2・T	17＋2・T	21＋4・T	17＋4・T	21＋2・T	19＋2・T	19＋2・T	15＋2・T
INS	reg8, reg8 '	２または４	63-155(ビット長によって異なります。)							
	reg8, imm4	２または４	64-156(ビット長によって異なります。)				－	－	64-156	64-156
LDEA	reg16, mem16	0	－	－	EA＋2	EA＋2	－	－	EA＋2	EA＋2
LDM[注2]	[Seg-spec：]src-block	1×rep	16＋(10＋T)・n	16＋(10＋T)・n	16＋(12＋2・T)・n	16＋(12＋2・T)・n	16＋(11＋T)・n	16＋(11＋T)・n	16＋(11＋T)・n	16＋(11＋T)・n
		(1)[注3]	12＋T	12＋T	14＋2·T	14＋2・T	13＋T	13＋T	13＋T	13＋T
LDMB[注2]	なし	1×rep	16＋(10＋T)・n	16＋(10＋T)・n	16＋(12＋2・T)・n	16＋(12＋2・T)・n	16＋(11＋T)・n	16＋(11＋T)・n	16＋(11＋T)・n	16＋(11＋T)・n
		(1)[注3]	12＋T	12＋T	14＋2・T	14＋2・T	13＋T	13＋T	13＋T	13＋T
LDMW[注2]	なし	1×rep	16＋(10＋T)・n	16＋(10＋T)・n	16＋(12＋2・T)・n	16＋(12＋2・T)・n	16＋(11＋T)・n	16＋(11＋T)・n	16＋(11＋T)・n	16＋(11＋T)・n
		(1)[注3]	12＋T	12＋T	14＋2・T	14＋2・T	13＋T	13＋T	13＋T	13＋T

注１．$\overline{\text{IBRK}}$ = 1のとき

２．n：リピート数（n≧1）

３．（ ）：一回だけの処理に適用

表２－８　命令実行クロック数一覧（7/15）

ニモニック	オペランド	ワード転送回数	V25, V25＋				V35, V35＋			
			バイト処理		ワード処理		バイト処理		ワード処理	
			内蔵RAMアクセス許可	内蔵RAMアクセス禁止	内蔵RAMアクセス許可	内蔵RAMアクセス禁止	内蔵RAMアクセス許可	内蔵RAMアクセス禁止	内蔵RAMアクセス許可	内蔵RAMアクセス禁止
MOV	reg, reg'	0	2	2	2	2	2	2	2	2
	mem, reg	1	EA＋4＋T	EA＋2	EA＋6＋2・T	EA＋2	EA＋5＋T	EA＋2	EA＋5＋T	EA＋2
	reg, mem	1	EA＋6＋T	EA＋6＋T	EA＋8＋2・T	EA＋8＋2・T	EA＋7＋T	EA＋7＋T	EA＋7＋T	EA＋7＋T
	mem, imm	1	EA＋5＋T	EA＋5＋T	EA＋5＋2・T	EA＋5＋2・T	EA＋6＋T	EA＋6＋T	EA＋6＋T	EA＋6＋T
	reg, imm	0	5	5	6	6	5	5	6	6
	acc, dmem	1	9＋T	9＋T	11＋2・T	11＋2・T	10＋T	10＋T	10＋T	10＋T
	dmem, acc	1	7＋T	5	9＋2・T	5	8＋T	5	8＋T	5
	sreg, reg16	0	－	－	4	4	－	－	4	4
	sreg, mem16	1	－	－	EA＋10＋2・T	EA＋10＋2・T	－	－	EA＋9＋T	EA＋9＋T
	reg16, sreg	0	－	－	3	3	－	－	3	3
	mem16, sreg	1	－	－	EA＋7＋2・T	EA＋3	－	－	EA＋6＋T	EA＋3
	DS0, reg16, mem32	2	－	－	EA＋19＋4・T	EA＋19＋4・T	－	－	EA＋17＋2・T	EA＋17＋2・T
	DS1, reg16, mem32	2	－	－	EA＋19＋4・T	EA＋19＋4・T	－	－	EA＋17＋2・T	EA＋17＋2・T
	AH, PSW	0	2	2	－	－	2	2	－	－
	PSW, AH	0	3	3	－	－	3	3	－	－
MOVBK[注1]	［DS1-spec：］dst-block,	2×rep	16＋(16＋2・T)・n	16＋(12＋T)・n	16＋(20＋4・T)・n	16＋(12＋2・T)・n	16＋(18＋2・T)・n	16＋(13＋2)・n	16＋(18＋2・T)・n	16＋(10＋T)・n
	［Seg-spec：］src-block	(2)[注2]	20＋2・T	16＋T	24＋4・T	20＋2・T	22＋2・T	17＋T	22＋2・T	19＋T
MOVBKB[注1]	なし	2×rep	16＋(16＋2・T)・n	16＋(12＋T)・n	16＋(20＋4・T)・n	16＋(12＋2・T)・n	16＋(18＋2・T)・n	16＋(13＋T)・n	16＋(18＋2・T)・n	16＋(10＋T)・n
		(2)[注2]	20＋2・T	16＋T	24＋4・T	20＋2・T	22＋2・T	17＋T	22＋2・T	19＋T
MOVBKW[注1]	なし	2×rep	16＋(16＋2・T)・n	16＋(12＋T)・n	16＋(20＋4・T)・n	16＋(12＋2・T)・n	16＋(18＋2・T)・n	16＋(13＋T)・n	16＋(18＋2・T)・n	16＋(10＋T)・n
		(2)[注2]	20＋2・T	16＋T	24＋4・T	20＋2・T	22＋2・T	17＋T	22＋2・T	19＋T
MOVSPA	なし	0	－	－	16	16	－	－	16	16
MOVSPB	reg16	0	－	－	11	11	－	－	11	11

注１．n：リピート数（n≧1）

２．（ ）：一回だけの処理に適用

表２－８　命令実行クロック数一覧（8/15）

ニモニック	オペランド	ワード転送回数	V25, V25＋				V35, V35＋			
			バイト処理		ワード処理		バイト処理		ワード処理	
			内蔵RAMアクセス許可	内蔵RAMアクセス禁止	内蔵RAMアクセス許可	内蔵RAMアクセス禁止	内蔵RAMアクセス許可	内蔵RAMアクセス禁止	内蔵RAMアクセス許可	内蔵RAMアクセス禁止
MUL	reg8	0	31-40	31-40	－	－	31-40	31-40	－	－
	mem8	0	EA＋33＋T ～EA＋42＋T	EA＋33＋T ～EA＋42＋T	－	－	EA＋34＋T ～EA＋43＋T	EA＋34＋T ～EA＋43＋T	－	－
	reg16	0	－	－	39-48	39-48	－	－	39-48	39-48
	mem16	1	－	－	EA＋43＋2・T ～EA＋52＋2・T	EA＋43＋2・T ～EA＋52＋2・T	－	－	EA＋42＋T ～EA＋51＋T	EA＋42＋T ～EA＋51＋T
	reg16, imm8	0	－	－	39-49	39-49	－	－	39-49	39-49
	reg16, imm16	0	－	－	40-50	40-50	－	－	40-50	40-50
	reg16, reg16', imm8	0	－	－	39-49	39-49	－	－	39-49	39-49
	reg16, mem16, imm8	1	－	－	EA＋43＋2･T ～EA＋53＋2･T	EA＋43＋2･T ～EA＋53＋2･T	－	－	EA＋42＋T ～EA＋52＋T	EA＋42＋T ～EA＋52＋T
	reg16, reg16', imm16	0	－	－	40-50	40-50	－	－	40-50	40-50
	reg16, mem16, imm16	1	－	－	EA＋44＋2･T ～EA＋54＋2･T	EA＋44＋2･T ～EA＋54＋2･T	－	－	EA＋43＋T ～EA＋53＋T	EA＋43＋T ～EA＋53＋T
MULU	reg8	0	24	24	－	－	24	24	－	－
	mem8	1	EA＋26＋T	EA＋26＋T	－	－	EA＋27＋T	EA＋27＋T	－	－
	reg16	0	－	－	32	32	－	－	32	32
	mem16	1	－	－	EA＋34＋2･T	EA＋34＋2･T	－	－	EA＋33＋T	EA＋33＋T
NEG	reg	0	5	5	5	5	5	5	5	5
	mem	2	EA＋11＋2・T	EA＋9＋T	EA＋15＋4･T	EA＋11＋2･T	EA＋13＋2･T	EA＋10＋T	EA＋13＋2・T	EA＋10＋T
NOP	なし	0	4	4	4	4	4	4	4	4
NOT	reg	0	5	5	5	5	5	5	5	5
	mem	2	EA＋11＋2・T	EA＋9＋T	EA＋15＋4･T	EA＋11＋2･T	EA＋10＋2･T	EA＋10＋T	EA＋13＋2・T	EA＋10＋T

表２－８　命令実行クロック数一覧（9/15）

ニモニック	オペランド	ワード転送回数	V25, V25＋				V35, V35＋			
			バイト処理		ワード処理		バイト処理		ワード処理	
			内蔵RAMアクセス許可	内蔵RAMアクセス禁止	内蔵RAMアクセス許可	内蔵RAMアクセス禁止	内蔵RAMアクセス許可	内蔵RAMアクセス禁止	内蔵RAMアクセス許可	内蔵RAMアクセス禁止
NOT1	reg8, CL	0	7	7	－	－	7	7	－	－
	mem8, CL	0	EA＋13＋2・T	EA＋11＋2・T	－	－	EA＋15＋2・T	EA＋12＋T	－	－
	reg16, CL	0	－	－	7	7	－	－	7	7
	mem16, CL	2	－	－	EA＋17＋4・T	EA＋13＋2・T	－	－	EA＋15＋2・T	EA＋12＋T
	reg8, imm3	0	6	6	－	－	6	6	－	－
	mem8, imm3	0	EA＋10＋2・T	EA＋8＋T	－	－	EA＋12＋2・T	EA＋9＋T	－	－
	reg16, imm4	0	－	－	6	6	－	－	6	6
	mem16, imm4	2	－	－	EA＋14＋4・T	EA＋10＋2・T	－	－	EA＋12＋2・T	EA＋9＋T
	CY	0	2	2	2	2	2	2	2	2
OR	reg, reg'	0	2	2	2	2	2	2	2	2
	mem, reg	2	EA＋8＋2・T	EA＋6＋T	EA＋12＋4・T	EA＋8＋2・T	EA＋10＋2・T	EA＋7＋T	EA＋10＋2・T	EA＋7＋T
	reg, mem	1	EA＋6＋T	EA＋6＋T	EA＋8＋2・T	EA＋8＋2・T	EA＋7＋T	EA＋7＋T	EA＋7＋T	EA＋7＋T
	reg, imm	0	5	5	6	6	5	5	6	6
	mem, imm	2	EA＋9＋2・T	EA＋7＋2・T	EA＋14＋4・T	EA＋10＋4・T	EA＋11＋2・T	EA＋9＋2・T	EA＋12＋12・T	EA＋8＋2・T
	acc, imm	0	5	5	6	6	5	5	6	6
OUT[注1]	imm8, acc	1	10＋T	10＋T	10＋2・T	10＋2・T	11＋T	11＋T	9＋T	9＋T
	DW, acc	1	9＋T	9＋T	9＋2・T	9＋2・T	9＋T	9＋T	9＋2・T	9＋2·T
OUTM[注2]	DW, ［Seg-spec］src-block	2×rep	18＋(13＋2・T)・n	18＋(11＋2・T)・n	18＋(15＋4・T)・n	18＋(11＋4・T)・n	18＋(15＋2・T)・n	18＋(13＋2・T)・n	18＋(13＋2・T)・n	18＋(9＋2・T)・n
		(2)[注3]	19＋2・T	17＋2・T	21＋4・T	17＋4・T	21＋2・T	19＋2・T	19＋2・T	15＋2・T
POLL	なし	0	－	－	－	－	－	－	－	－

注１．$\overline{IBRK}$ = 1のとき

２．n：リピート数（n≧1）

３．（ ）：一回だけの処理に適用

表２－８　命令実行クロック数一覧（10/15）

ニモニック	オペランド	ワード転送回数	V25, V25＋				V35, V35＋			
			バイト処理		ワード処理		バイト処理		ワード処理	
			内蔵RAMアクセス許可	内蔵RAMアクセス禁止	内蔵RAMアクセス許可	内蔵RAMアクセス禁止	内蔵RAMアクセス許可	内蔵RAMアクセス禁止	内蔵RAMアクセス許可	内蔵RAMアクセス禁止
POP	mem16	2	－	－	EA＋16＋4・T	EA＋12＋2・T	－	－	EA＋14＋2・T	EA＋11＋T
	reg16	1	－	－	12＋2・T	12＋2・T	－	－	11＋T	11＋T
	sreg	1	－	－	13＋2・T	13＋2・T	－	－	12＋T	12＋T
	PSW	1	－	－	14＋2・T	14＋2・T	－	－	13＋T	13＋T
	R	7	－	－	82＋16・T	58	－	－	74＋8・T	58
PREPARE	imm16, imm8（imm8 = 0のとき）	0	27＋2・T				26＋T			
	imm16, imm8（imm8 = 1のとき）	1＋2（imm8-1）	39＋4・T				37＋2・T			
	imm16, imm8（imm8≧n,n＞1のとき）		46＋19（n－1）＋4・T				44＋19（n－1）＋2・T			
PS：	なし	0	2	2	2	2	2	2	2	2
PUSH	mem16	2	－	－	EA＋18＋4・T	EA＋14＋4・T	－	－	EA＋16＋2・T	EA＋12＋2・T
	reg16	1	－	－	10＋2・T	6	－	－	13＋T	9＋T
	sreg	1	－	－	11＋2・T	7	－	－	10＋T	7
	PSW	1	－	－	10＋2・T	6	－	－	9＋T	6
	R	8	－	－	82＋16・T	50	－	－	74＋8・T	50
	imm8	1	－	－	13＋2・T	9	－	－	12＋T	9
	imm16	1	－	－	14＋2・T	10	－	－	13＋T	10
REP	なし	0	2	2	2	2	2	2	2	2
REPC	なし	0	2	2	2	2	2	2	2	2
REPE	なし	0	2	2	2	2	2	2	2	2
REPNC	なし	0	2	2	2	2	2	2	2	2
REPNE	なし	0	2	2	2	2	2	2	2	2
REPNZ	なし	0	2	2	2	2	2	2	2	2
REPZ	なし	0	2	2	2	2	2	2	2	2

表2－8　命令実行クロック数一覧（11/15）

ニモニック	オペランド	ワード転送回数	V25, V25＋				V35, V35＋			
			バイト処理		ワード処理		バイト処理		ワード処理	
			内蔵RAMアクセス許可	内蔵RAMアクセス禁止	内蔵RAMアクセス許可	内蔵RAMアクセス禁止	内蔵RAMアクセス許可	内蔵RAMアクセス禁止	内蔵RAMアクセス許可	内蔵RAMアクセス禁止
RET	なし（セグメント内コール）	1	－	－	20＋2・T	20＋2・T	－	－	19＋T	19＋T
	pop-value（セグメント内コール）	1	－	－	20＋2・T	20＋2・T	－	－	19＋T	19＋T
	なし（セグメント外コール）	2	－	－	29＋4・T	29＋4・T	－	－	27＋2・T	27＋2・T
	pop-value（セグメント外コール）	2	－	－	30＋4・T	30＋4・T	－	－	28＋2・T	28＋2・T
RETI	なし	3	－	－	45＋6・T	37＋2・T	－	－	42＋3・T	36＋T
RETRBI	なし	0	－	－	12	12	－	－	12	12
ROL[注]	reg, 1	0	8	8	8	8	8	8	8	8
	mem, 1	2	EA＋14＋2・T	EA＋12＋T	EA＋18＋4・T	EA＋14＋2・T	EA＋16＋2・T	EA＋13＋T	EA＋16＋2・T	EA＋13＋T
	reg, CL	0	11＋2・n	11＋2・n	11＋2・n	11＋2・n	11＋2・n	11＋2・n	11＋2・n	11＋2・n
	mem, CL	2	EA+17+2・T+2・n	EA+15+T+2・n	EA+21+4・T+2・n	EA+17+2・T+2・n	EA+19+2・T+2・n	EA+16+T+2・n	EA+19+2・T+2・n	EA+16+T+2・n
	reg, imm8	0	9＋2・n	9＋2・n	9＋2・n	9＋2・n	9＋2・n	9＋2・n	9＋2・n	9＋2・n
	mem, imm8	2	EA+13+2・T+2・n	EA+11+T+2・n	EA+17+4・T+2・n	EA+13+2・T+2・n	EA+15+2・T+2・n	EA+12+T+2・n	EA+15+2・T+2・n	EA+12+T+2・n
ROL4	reg8	0	17	17	－	－	17	17	－	－
	mem8	0	EA＋18＋2・T	EA＋16＋2・T	－	－	EA＋20＋2・T	EA＋18＋2・T	－	－
ROLC[注]	reg, 1	0	8	8	8	8	8	8	8	8
	mem, 1	2	EA＋14＋2・T	EA＋12＋T	EA＋18＋4・T	EA＋14＋2・T	EA＋16＋2・T	EA＋13＋T	EA＋16＋2・T	EA＋13＋T
	reg, CL	0	11＋2・n	11＋2・n	11＋2・n	11＋2・n	11＋2・n	11＋2・n	11＋2・n	11＋2・n
	mem, CL	2	EA+17+2・T+2・n	EA+15+T+2・n	EA+21+4・T+2・n	EA+17+2・T+2・n	EA+19+2・T+2・n	EA+16+T+2・n	EA+19+2・T+2・n	EA+16+T+2・n
	reg, imm8	0	9＋2・n	9＋2・n	9＋2・n	9＋2・n	9＋2・n	9＋2・n	9＋2・n	9＋2・n
	mem, imm8	2	EA+13+2・T+2・n	EA+11+T+2・n	EA+17+4・T+2・n	EA+13+2・T+2・n	EA+15+2・T+2・n	EA+12+T+2・n	EA+15+2・T+2・n	EA+12+T+2・n
ROR[注]	reg, 1	0	8	8	8	8	8	8	8	8
	mem, 1	2	EA＋14＋2・T	EA＋12＋T	EA＋18＋4・T	EA＋14＋2・T	EA＋16＋2・T	EA＋13＋T	EA＋16＋2・T	EA＋13＋T
	reg, CL	0	11＋2・n	11＋2・n	11＋2・n	11＋2・n	11＋2・n	11＋2・n	11＋2・n	11＋2・n
	mem, CL	2	EA+17+2・T+2・n	EA+15+T+2・n	EA+21+4・T+2・n	EA+17+2・T+2・n	EA+19+2・T+2・n	EA+16+T+2・n	EA+19+2・T+2・n	EA+16+T+2・n
	reg, imm8	0	9＋2・n	9＋2・n	9＋2・n	9＋2・n	9＋2・n	9＋2・n	9＋2・n	9＋2・n
	mem, imm8	2	EA+13+2・T+2・n	EA+11+T+2・n	EA+17+4・T+2・n	EA+13+2・T+2・n	EA+15+2・T+2・n	EA+12+T+2・n	EA+15+2・T+2・n	EA+12+T+2・n

注　n：シフト数

表２－８　命令実行クロック数一覧（12/15）

ニモニック	オペランド	ワード転送回数	V25, V25＋				V35, V35＋			
			バイト処理		ワード処理		バイト処理		ワード処理	
			内蔵RAMアクセス許可	内蔵RAMアクセス禁止	内蔵RAMアクセス許可	内蔵RAMアクセス禁止	内蔵RAMアクセス許可	内蔵RAMアクセス禁止	内蔵RAMアクセス許可	内蔵RAMアクセス禁止
ROR4	reg8	0	21	21	－	－	21	21	－	－
	mem8	0	EA＋24＋2・T	EA＋22＋2・T	－	－	EA＋26＋2・T	EA＋24＋2・T	－	－
RORC[注]	reg, 1	0	8	8	8	8	8	8	8	8
	mem, 1	2	EA＋14＋2・T	EA＋12＋2・T	EA＋18＋4・T	EA＋14＋2・T	EA＋16＋2・T	EA＋13＋T	EA＋16＋2・T	EA＋13＋T
	reg, CL	0	11＋2・n	11＋2・n	11＋2・n	11＋2・n	11＋2・n	11＋2・n	11＋2・n	11＋2・n
	mem, CL	2	EA+17+2・T+2・n	EA+15+T+2・n	EA+21+4・T+2・n	EA+17+2・T+2・n	EA+19+2・T+2・n	EA+16+T+2・n	EA+19+2・T+2・n	EA+16+T+2・n
	reg, imm8	0	9＋2・n	9＋2・n	9＋2・n	9＋2・n	9＋2・n	9＋2・n	9＋2・n	9＋2・n
	mem, imm8	2	EA+13+2・T+2・n	EA+11+T+2・n	EA+17+4・T+2・n	EA+13+2・T+2・n	EA+15+2・T+2・n	EA+12+T+2・n	EA+15+2・T+2・n	EA+12+T+2・n
SET1	reg8, CL	0	7	7	－	－	7	7	－	－
	mem8, CL	0	EA＋13＋2・T	EA＋11＋T	－	－	EA＋15＋2・T	EA＋12＋T	－	－
	reg16, CL	0	－	－	7	7	－	－	7	7
	mem16, CL	2	－	－	EA＋17＋4・T	EA＋13＋2・T	－	－	EA＋15＋2・T	EA＋12＋T
	reg8, imm3	0	6	6	－	－	6	6	－	－
	mem8, imm3	0	EA＋10＋2・T	EA＋8＋T	－	－	EA＋12＋2・T	EA＋9＋T	－	－
	reg16, imm4	0	－	－	6	6	－	－	6	6
	mem16, imm4	2	－	－	EA＋14＋4・T	EA＋10＋2・T	－	－	EA＋12＋2・T	EA＋9＋T
	CY	0	2	2	2	2	2	2	2	2
	DIR	0	2	2	2	2	2	2	2	2
SHL[注]	reg, 1	0	8	8	8	8	8	8	8	8
	mem, 1	2	EA＋14＋2・T	EA＋12＋2・T	EA＋18＋4・T	EA＋14＋2・T	EA＋16＋2・T	EA＋13＋T	EA＋16＋2・T	EA＋13＋T
	reg, CL	0	11＋2・n	11＋2・n	11＋2・n	11＋2・n	11＋2・n	11＋2・n	11＋2・n	11＋2・n
	mem, CL	2	EA+17+2・T+2・n	EA+15+T+2・n	EA+21+4・T+2・n	EA+17+2・T+2・n	EA+19+2・T+2・n	EA+16+T+2・n	EA+19+2・T+2・n	EA+16+T+2・n
	reg, imm8	0	9＋2・n	9＋2・n	9＋2・n	9＋2・n	9＋2・n	9＋2・n	9＋2・n	9＋2・n
	mem, imm8	2	EA+13+2・T+2・n	EA+11+T+2・n	EA+17+4・T+2・n	EA+13+2・T+2・n	EA+15+2・T+2・n	EA+12+T+2・n	EA+15+2・T+2・n	EA+12+T+2・n

注　n：シフト数

表２－８　命令実行クロック数一覧（13/15）

ニモニック	オペランド	ワード転送回数	V25, V25＋				V35, V35＋			
			バイト処理		ワード処理		バイト処理		ワード処理	
			内蔵RAMアクセス許可	内蔵RAMアクセス禁止	内蔵RAMアクセス許可	内蔵RAMアクセス禁止	内蔵RAMアクセス許可	内蔵RAMアクセス禁止	内蔵RAMアクセス許可	内蔵RAMアクセス禁止
SHR注1	reg, 1	0	8	8	8	8	8	8	8	8
	mem, 1	2	EA＋14＋2・T	EA＋12＋T	EA＋18＋4・T	EA＋14＋2・T	EA＋16＋2・T	EA＋13＋T	EA＋16＋2・T	EA＋13＋T
	reg, CL	0	11＋2・n	11＋2・n	11＋2・n	11＋2・n	11＋2・n	11＋2・n	11＋2・n	11＋2・n
	mem, CL	2	EA+17+2・T+2・n	EA+15+T+2・n	EA+21+4・T+2・n	EA+17+2・T+2・n	EA+19+2・T+2・n	EA+16+T+2・n	EA+19+2・T+2・n	EA+16+T+2・n
	reg, imm8	0	9＋2・n	9＋2・n	9＋2・n	9＋2・n	9＋2・n	9＋2・n	9＋2・n	9＋2・n
	mem, imm8	2	EA+13+2・T+2・n	EA+11+T+2・n	EA+17+4・T+2・n	EA+13+2・T+2・n	EA+15+2・T+2・n	EA+12+T+2・n	EA+15+2・T+2・n	EA+12+T+2・n
SHRA注1	reg, 1	0	8	8	8	8	8	8	8	8
	mem, 1	2	EA＋14＋2・T	EA＋12＋T	EA＋18＋4・T	EA＋14＋2・T	EA＋16＋2・T	EA＋13＋T	EA＋16＋2・T	EA＋13＋T
	reg, CL	0	11＋2・n	11＋2・n	11＋2・n	11＋2・n	11＋2・n	11＋2・n	11＋2・n	11＋2・n
	mem, CL	2	EA+17+2・T+2・n	EA+15+T+2・n	EA+21+4・T+2・n	EA+17+2・T+2・n	EA+19+2・T+2・n	EA+16+T+2・n	EA+19+2・T+2・n	EA+16+T+2・n
	reg, imm8	0	9＋2・n	9＋2・n	9＋2・n	9＋2・n	9＋2・n	9＋2・n	9＋2・n	9＋2・n
	mem, imm8	2	EA+13+2・T+2・n	EA+11+T+2・n	EA+17+4・T+2・n	EA+13+2・T+2・n	EA+15+2・T+2・n	EA+12+T+2・n	EA+15+2・T+2・n	EA+12+T+2・n
SS：	なし	0	2	2	2	2	2	2	2	2
STM注2	[DS1-spec：]dst-block	1×rep	16＋(8＋T)・n	16＋(6＋T)・n	16＋(10＋2・T)・n	16＋(6＋2・T)・n	16＋(9＋T)・n	16＋(7＋T)・n	16＋(9＋T)・n	16＋(5＋T)・n
		(1)注3	12＋T	10	14＋2・T	10	13＋T	10	13＋T	10
STMB注2	なし	1×rep	16＋(8＋T)・n	16＋(6＋T)・n	16＋(10＋2・T)・n	16＋(6＋2・T)・n	16＋(9＋T)・n	16＋(7＋T)・n	16＋(9＋T)・n	16＋(5＋T)・n
		(1)注3	12＋T	10	14＋2・T	10	13＋T	10	13＋T	10
STMW注2	なし	1×rep	16＋(8＋T)・n	16＋(6＋T)・n	16＋(10＋2・T)・n	16＋(6＋2・T)・n	16＋(9＋T)・n	16＋(7＋T)・n	16＋(9＋T)・n	16＋(5＋T)・n
		(1)注3	12＋T	10	14＋2・T	10	13＋T	10	13＋T	10
STOP	なし	0	－	－	－	－	－	－	－	－

注１．n：シフト数

２．n：リピート数（n≧1）

３．（ ）：一回だけの処理に適用

表２－８　命令実行クロック数一覧（14/15）

ニモニック	オペランド	ワード転送回数	V25, V25＋				V35, V35＋			
			バイト処理		ワード処理		バイト処理		ワード処理	
			内蔵RAMアクセス許可	内蔵RAMアクセス禁止	内蔵RAMアクセス許可	内蔵RAMアクセス禁止	内蔵RAMアクセス許可	内蔵RAMアクセス禁止	内蔵RAMアクセス許可	内蔵RAMアクセス禁止
SUB	reg, reg ′	0	2	2	2	2	2	2	2	2
	mem, reg	2	EA＋8＋2・T	EA＋6＋T	EA＋12＋4・T	EA＋8＋2・T	EA＋10＋2・T	EA＋7＋T	EA＋10＋2・T	EA＋7＋T
	reg, mem	1	EA＋6＋T	EA＋6＋T	EA＋8＋2・T	EA＋8＋2・T	EA＋7＋T	EA＋7＋T	EA＋7＋T	EA＋7＋T
	reg, imm	0	5	5	6	6	5	5	6	6
	mem, imm	2	EA＋9＋2・T	EA＋7＋2・T	EA＋14＋4・T	EA＋10＋4・T	EA＋11＋2・T	EA＋9＋2・T	EA＋12＋2・T	EA＋8＋2・T
	acc, imm	0	5	5	6	6	5	5	6	6
SUB4S[注]	[DS1-spec：]dst-string, [Seg-spec：]src-string	0	22＋(27＋3・T)・m	22＋(25＋3・T)・m	－	－	22＋(30＋3・T)・m	22＋(28＋3・T)・m	－	－
	なし	0	22＋(27＋3・T)・m	22＋(25＋3・T)・m	－	－	22＋(30＋3・T)・m	22＋(28＋3・T)・m	－	－
SUBC	reg, reg ′	0	2	2	2	2	2	2	2	2
	mem, reg	2	EA＋8＋2・T	EA＋6＋T	EA＋12＋4・T	EA＋8＋2・T	EA＋10＋2・T	EA＋7＋T	EA＋10＋2・T	EA＋7＋T
	reg, mem	1	EA＋6＋T	EA＋6＋T	EA＋8＋2・T	EA＋8＋2・T	EA＋7＋T	EA＋7＋T	EA＋7＋T	EA＋7＋T
	reg, imm	0	5	5	6	6	5	5	6	6
	mem, imm	2	EA＋9＋2・T	EA＋7＋2・T	EA＋14＋4・T	EA＋10＋4・T	EA＋11＋2・T	EA＋9＋2・T	EA＋12＋2・T	EA＋8＋2・T
	acc, imm	0	5	5	6	6	5	5	6	6
TEST	reg, reg ′	0	4	4	4	4	4	4	4	4
	mem, reg	1	EA＋8＋T	EA＋8＋T	EA＋10＋2・T	EA＋10＋2・T	EA＋12＋T	EA＋12＋T	EA＋11＋2・T	EA＋11＋2・T
	reg, mem	1	EA＋8＋T	EA＋8＋T	EA＋10＋2・T	EA＋10＋2・T	EA＋12＋T	EA＋12＋T	EA＋11＋2・T	EA＋11＋2・T
	reg, imm	0	7	7	8	8	7	7	8	8
	mem, imm	1	EA＋11＋T	EA＋11＋T	EA＋11＋2・T	EA＋11＋2・T	EA＋9＋T	EA＋9＋T	EA＋10＋T	EA＋10＋T
	acc, imm	0	5	5	6	6	5	5	6	6

注　m：BCD桁数×1/2

表2－8　命令実行クロック数一覧（15/15）

ニモニック	オペランド	ワード転送回数	V25, V25＋				V35, V35＋			
			バイト処理		ワード処理		バイト処理		ワード処理	
			内蔵RAMアクセス許可	内蔵RAMアクセス禁止	内蔵RAMアクセス許可	内蔵RAMアクセス禁止	内蔵RAMアクセス許可	内蔵RAMアクセス禁止	内蔵RAMアクセス許可	内蔵RAMアクセス禁止
TEST1	reg8, CL	0	7	7	－	－	7	7	－	－
	mem8, CL	0	EA＋11＋T	EA＋11＋T	－	－	EA＋12＋T	EA＋12＋T	－	－
	reg16, CL	0	－	－	7	7	－	－	7	7
	mem16, CL	1	－	－	EA＋13＋2・T	EA＋13＋2・T	－	－	EA＋12＋T	EA＋12＋T
	reg8,imm3	0	6	6	－	－	6	6	－	－
	mem8, imm3	0	EA＋8＋T	EA＋8＋T	－	－	EA＋9＋T	EA＋9＋T	－	－
	reg16, imm4	0	－	－	6	6	－	－	6	6
	mem16, imm4	1	－	－	EA＋10＋2・T	EA＋10＋2・T	－	－	EA＋9＋T	EA＋9＋T
TRANS	src-table	1	10＋T	10＋T	－	－	11＋T	11＋T	－	－
	なし	1	10＋T	10＋T	－	－	11＋T	11＋T	－	－
TRANSB	なし	1	10＋T	10＋T	－	－	11＋T	11＋T	－	－
TSKSW	reg16	0	－	－	20	20	－	－	20	20
XCH	reg, reg ′	0	3	3	3	3	3	3	3	3
	mem, reg ′	2	EA＋10＋2・T	EA＋8＋2・T	EA＋14＋4・T	EA＋10＋4・T	EA＋12＋2・T	EA＋9＋T	EA＋12＋2・T	EA＋9＋T
	reg, mem	2	EA＋10＋2・T	EA＋8＋2・T	EA＋14＋4・T	EA＋10＋4・T	EA＋12＋2・T	EA＋9＋T	EA＋12＋2・T	EA＋9＋T
	AW, reg16	0	－	－	4	4	－	－	4	4
	reg16, AW	0	－	－	4	4	－	－	4	4
XOR	reg, reg ′	0	2	2	2	2	2	2	2	2
	mem, reg	2	EA＋8＋2・T	EA＋6＋T	EA＋12＋4・T	EA＋8＋2・T	EA＋10＋2・T	EA＋7＋T	EA＋10＋2・T	EA＋7＋T
	reg, mem	1	EA＋6＋T	EA＋6＋T	EA＋8＋2・T	EA＋8＋2・T	EA＋7＋T	EA＋7＋T	EA＋7＋T	EA＋7＋T
	reg, imm	0	5	5	6	6	5	5	6	6
	mem, imm	2	EA＋9＋2・T	EA＋7＋2・T	EA＋14＋4・T	EA＋10＋4・T	EA＋11＋2・T	EA＋9＋2・T	EA＋12＋2・T	EA＋8＋2・T
	acc, imm	0	5	5	6	6	5	5	6	6

第３章　V20, V30に対する追加命令

V25, V35ファミリの命令セットは，V20, V30の命令セットと上位互換性を持っています。
V20, V30に対して追加された命令を次に示します。

（1）条件付きブランチ命令

●BTCLR…特殊機能レジスタのビット・テスト命令

BTCLRの実行により，対象となる特殊機能レジスタのビットの状態が１の場合，そのビットをリセット（0）し，オペランドに記述したshort-labelへ分岐します。
対象ビットが０の場合，次の命令に移ります。なお，PSWは変化しません。

（記述形式）

ニモニック	オペランド		
	特殊機能レジスタのアドレス	特殊機能レジスタの対象ビット	分　岐　先
BTCLR	sfr	imm3	short-label

（2）割り込み命令

●RETRBI…レジスタ・バンクの復帰命令

レジスタ・バンク切り替え機能を使用した割り込み処理ルーチンから復帰するときに使用します。ベクタ割り込みからの復帰には使用できません。

（記述形式）

ニモニック	オペランド
RETRBI	な　し

●FINT…割り込みコントローラに対して割り込み処理を終了したことを示す命令

NMI, INT，ソフトウエア割り込みを除く割り込み処理からの復帰命令の前に実行してください。
NMI, INT，ソフトウエア割り込みでは使用しないでください。

（記述形式）

ニモニック	オペランド
FINT	な　し

保守／廃止

（３）CPU制御命令

●STOP…STOP状態への移行命令

（記述形式）

ニモニック	オペランド
STOP	な　し

（４）レジスタ・バンク切り替え命令

●BRKCS…レジスタ・バンクを切り替える命令

BRKCSの実行により，レジスタ・バンクがオペランドに記述した16ビット・レジスタの内容の下位３ビットで示されるレジスタ・バンクに切り替わります。
また，新しいレジスタ・バンク内にあらかじめストアしておいたPSとベクタPCから得られるアドレスに分岐します。
新しいレジスタ・バンクからの復帰には，RETRBI命令を使用します。

（記述形式）

ニモニック	オペランド
BRKCS	reg16

●TSKSW…レジスタ・バンクを切り替える命令

BRKCS命令と同様にレジスタ・バンクを切り替え，新しいレジスタ・バンク内にあらかじめストアしておいたPSとPC退避エリアから得られるアドレスに分岐します。

（記述形式）

ニモニック	オペランド
TSKSW	reg16

（５）データ転送命令

●MOVSPA…SS, SPの転送命令

レジスタ・バンクの切り替わる前のSS, SPの値を，現在（切り替え後）のレジスタ・バンクのSS, SPに転送します。

（記述形式）

ニモニック	オペランド
MOVSPA	な　し

●MOVSPB…SS, SPの転送命令

現在（切り替え前）のレジスタ・バンクSS, SPの値を，オペランドに記述した16ビット・レジスタの内容の下位３ビットで示される切り替え先のレジスタ・バンクのSS, SPに転送します。

（記述形式）

ニモニック	オペランド
MOVSPB	reg16

保守／廃止

その他，V20, V30の命令セットに対して，V25, V35ファミリでは使用上注意すべき以下の命令セットがあります。

●入出力命令（IN, OUT），プリミティブ入出力命令（INM, OUTM）

PSWの$\overline{\text{IBRK}}$フラグがリセット（０）されていると命令を実行しないで割り込みが発生します。

実行前に$\overline{\text{IBRK}}$フラグをセット（１）してください。

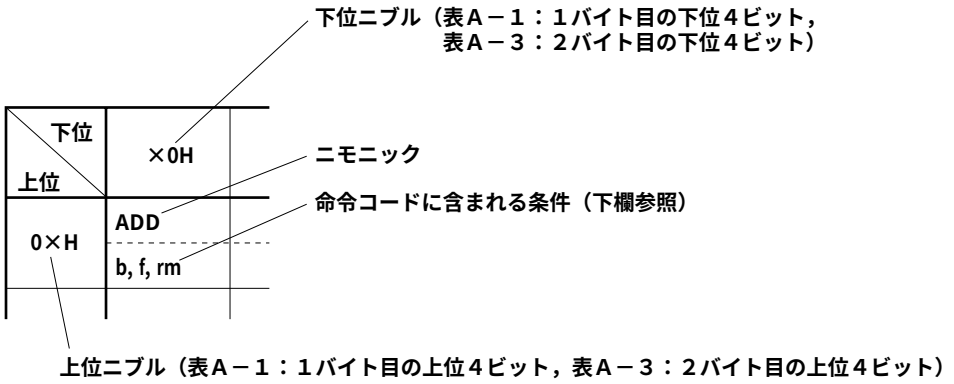
●FP01, FP02

命令を実行しないで，割り込みが発生します。

〔メ　モ〕

付録A　命令マップ

【凡　　例】



【命令コードに含まれる条件】

b　：バイト動作を行う
d　：ダイレクト・アドレシングを用いる
f　：CPU内のレジスタからのリード動作を伴う
i　：イミーディエト・データを用いる
ia　：イミーディエト・データを用い，アキュムレータへの書き戻しがある
id　：インダイレクト・アドレシングを用いる
l　：セグメント間の制御を伴う
m　：メモリ・データを用いる
reg8：８ビット・レジスタを用いる
rm　：２バイト目に実効アドレス・フィールドを持つ
s　：符号拡張した16ビット・イミーディエト・データを用いる
sr　：セグメント・レジスタを使用する
t　：CPU内のレジスタへのライト動作
v　：インダイレクトでポート番号を指定する
w　：ワード動作を行う

上記以外の記号については**表２－４　命令形式またはオペレーション説明上の凡例**を参照してください。

保守／廃止

表A－1　命令マップ

下位 / 上位	×0H	×1H	×2H	×3H	×4H	×5H	×6H	×7H	×8H	×9H	×AH	×BH	×CH	×DH	×EH	×FH
0×H	ADD	ADD	ADD	ADD	ADD	ADD	PUSH	POP	OR	OR	OR	OR	OR	OR	PUSH	Group3
	b,f,rm	w,f,rm	b,t,rm	w,t,rm	b,ia	w,ia	DS1	DS1	b,f,rm	w,f,rm	b,t,rm	w,t,rm	b,ia	w,ia	PS	
1×H	ADDC	ADDC	ADDC	ADDC	ADDC	ADDC	PUSH	POP	SUBC	SUBC	SUBC	SUBC	SUBC	SUBC	PUSH	POP
	b,f,rm	w,f,rm	b,t,rm	w,t,rm	b,ia	w,ia	SS	SS	b,f,rm	w,f,rm	b,t,rm	w,t,rm	b,ia	w,ia	DS0	DS0
2×H	AND	AND	AND	AND	AND	AND	DS1：	ADJ4A	SUB	SUB	SUB	SUB	SUB	SUB	PS：	ADJ4S
	b,f,rm	w,f,rm	b,t,rm	w,t,rm	b,ia	w,ia			b,f,rm	w,f,rm	b,t,rm	w,t,rm	b,ia	w,ia		
3×H	XOR	XOR	XOR	XOR	XOR	XOR	SS：	ADJBA	CMP	CMP	CMP	CMP	CMP	CMP	DS0：	ADJBS
	b,f,rm	w,f,rm	b,t,rm	w,t,rm	b,ia	w,ia			b,f,rm	w,f,rm	b,t,rm	w,t,rm	b,ia	w,ia		
4×H	INC	INC	INC	INC	INC	INC	INC	INC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC
	AW	CW	DW	BW	SP	BP	IX	IY	AW	CW	DW	BW	SP	BP	IX	IY
5×H	PUSH	PUSH	PUSH	PUSH	PUSH	PUSH	PUSH	PUSH	POP	POP	POP	POP	POP	POP	POP	POP
	AW	CW	DW	BW	SP	BP	IX	IY	AW	CW	DW	BW	SP	BP	IX	IY
6×H	PUSH	POP	CHKIND	未定義	REPNC	REPC	FPO2	FPO2	PUSH	MUL	PUSH	MUL	INM	INM	OUTM	OUTM
	R	R					0	1	w,i	w,i	s,i	s,i	b	w	b	w
7×H	BV	BNV	BC BL	BNC BNL	BE BZ	BNE BNZ	BNH	BH	BN	BP	BPE	BPO	BLT	BGE	BLE	BGT
8×H	Imm	Imm	Imm	Imm	TEST	TEST	XCH	XCH	NOV	MOV	MOV	MOV	MOV	LDEA	MOV	POP
	b,rm	w,rm	b,s,rm	w,s,rm	b,rm	w,rm	b,rm	w,rm	b,f,rm	w,f,rm	b,t,rm	w,t,rm	sr,f,rm		sr,t,rm	rm
9×H	XCH	XCH	XCH	XCH	XCH	XCH	XCH	XCH	CVTBW	CVTWL	CALL	POLL	PUSH	POP	MOV	MOV
	AW	CW	DW	BW	SP	BP	IX	IY			l,d		PSW	PSW	PSW,AH	AH,PSW
A×H	MOV	MOV	MOV	MOV	MOVBK MOVBKB MOVBKW	MOVBK MOVBKB MOVBKW	CMPBK CMPBKB CMPBKW	CMPBK CMPBKB CMPBKW	TEST	TEST	STM STMB STMW	STM STMB STMW	LDM LDMB LDMW	LDM LDMB LDMW	CMPM CMPMB CMPMW	CMPM CMPMB CMPMW
	AL,m	AW,m	m,AL	m,AW	b	w	b	w	b,ia	w,ia	b	w	b	w	b	w
B×H	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV
	AL,i	CL,i	DL,i	BL,i	AH,i	CH,i	DH,i	BH,i	AW,i	CW,i	DW,i	BW,i	SP,i	BP,i	IX,i	IY,i
C×H	Shift	Shift	RET	RET	MOV	MOV	MOV	MOV	PREPARE	DISPOSE	RET	RET	BRK	BRK	BRKV	RETI
	b,i	w,i	(SP)		DS1	DS0	b,i,rm	w,i,rm			l,(SP)	l	3	i		
D×H	Shift	Shift	Shift	Shift	CVTBD	CVTDB	未定義	TRANS TRANSB	FPO1	FPO1	FPO1	FPO1	FPO1	FPO1	FPO1	FPO1
	b	w	b,v	w,v					0	1	2	3	4	5	6	7
E×H	DBNZE	DBNZE	DBNZ	BCWZ	IN	IN	OUT	OUT	CALL	BR	BR	BR	IN	IN	OUT	OUT
					b	w	b	w	d	d	l,d	si,d	b,v	w,v	b,v	w,v
F×H	BUSLOCK	未定義	REPNE REPNZ	REP REPE REPZ	HALT	NOT1	Group1	Group1	CLR1	SET1	DI	EI	CLR1	SET1	Group2	Group2
						CY	b	w	CY	CY			DIR	DIR	b	w

注意 ：**Group1, Group2, Imm, Shiftは命令コードの２バイト目のビット３-ビット５によって命令が定まります（表A－２参照）。**

Group3は命令コードの２バイト目によって命令が定まります（表A－３参照）。

保守／廃止

表A－2　Group1, Group2, Imm, Shiftコード表

注	000	001	010	011	100	101	110	111
Imm	ADD	OR	ADDC	SUBC	AND	SUB	XOR	CMP
Shift	ROL	ROR	ROLC	RORC	SHL	SHR	未定義	SHRA
Group1	TEST rm	未定義	NOT rm	NEG rm	MULU rm	MUL rm	DIVU rm	DIV rm
Group2	INC rm	DEC rm	CALL id	CALL l, id	BR id	BR l, id	PUSH rm	未定義

注　2バイト目のビット5-ビット3

表A－3　Group3コード表

下位 上位	×0H	×1H	×2H	×3H	×4H	×5H	×6H	×7H	×8H	×9H	×AH	×BH	×CH	×DH	×EH	×FH
0×H																
1×H	TEST1 b	TEST1 w	CLR1 b	CLR1 w	SET1 b	SET1 w	NOT1 b	NOT1 w	TEST1 i, b	TEST1 i, w	CLR1 i, b	CLR1 i, w	SET1 i, b	SET1 i, w	NOT1 i, b	NOT1 i, w
2×H	ADD4S		SUB4S			MOVSPA	CMP4S		ROL4		ROR4			BRKCS		
3×H		INS reg8		EXT reg8						INS i		EXT i				
9×H		RETRBI	FINT		TSKSW	MOVSPB							BTCLR		STOP	

備考　空白の欄は未定義コードです。

〔メ　モ〕

付録Ｂ　μPD8086, 8088とのニモニック対応表

V25, V35ファミリの命令セットは，オブジェクト・コード・レベルで，μPD8086, 8088に対して上位互換性があります。

表Ｂ－１にμPD8086, 8088とV25, V35ファミリのニモニックの対応を示します。

保守／廃止

表B－1　μPD8086, 8088とのニモニック対応表

μPD8086, 8088	V25, V35 ファミリ	μPD8086, 8088	V25, V35 ファミリ	μPD8086, 8088	V25, V35 ファミリ	μPD8086, 8088	V25, V35 ファミリ
AAA	ADJBA	JBE	BNH	LOOPNE	DBNZNE	STD	SET1 DIR
AAD	CVTDB	JC	BC/BL	LOOPNZ	DBNZNE	STI	EI
AAM	CVTBD	JCXZ	BCWZ	LOOPZ	DBNZE	STOS	STM/STMB/
AAS	ADJBS	JE	BE/BZ	MOV	MOV		STMW
ADC	ADDC	JG	BGT	MOVS	MOVBK	SUB	SUB
ADD	ADD	JGE	BGE	MOVSB	MOVBKB	TEST	TEST
AND	AND	JL	BLT	MOVSW	MOVBKW	WAIT	POLL
CALL	CALL	JLE	BLE	MUL	MULU	XCHG	XCH
CBW	CVTBW	JMP	BR	NEG	NEG	XLAT	TRANS
CLC	CLR1 CY	JNA	BNH	NOP	NOP	XLATB	TRANSB
CLD	CLR1 DIR	JNAE	BC/BL	NOT	NOT	XOR	XOR
CLI	DI	JNB	BNC/BNL	OR	OR	－	ADD4S
CMC	NOT1 CY	JNBE	BH	OUT	OUT	－	BRKCS
CMP	CMP	JNC	BNC/BNL	POP	POP	－	BTCLR
CMPS	CMPBK/	JNE	BNE/BNZ	POPF	POP PSW	－	CHKIND
	CMPBKB/	JNG	BLE	PUSH	PUSH	－	CMP4S
	CMPBKW	JNGE	BLT	PUSHF	PUSH PSW	－	DISPOSE
CS：	PS：	JNL	BGE	RCL	ROLC	－	EXT
CWD	CVTWL	JNLE	BGT	RCR	RORC	－	FINT
DAA	ADJ4A	JNO	BNV	REP	REP	－	FPO2
DAS	ADJ4S	JNP	BPO	REPE	REPE	－	INM
DEC	DEC	JNS	BP	REPNE	REPNE	－	INS
CIV	DIVU	JNZ	BNE/BNZ	REPNZ	REPNZ	－	MOVSPA
DS：	DS0：	JO	BV	REPZ	REPZ	－	MOVSPB
ES：	DS1：	JP	BPE	RET	RET	－	OUTM
ESC	FPO1	JPE	BPE	ROL	ROL	－	PREPARE
HLT	HALT	JPO	BPO	ROR	ROR	－	REPC
IDIV	DIV	JS	BN	SAHF	MOV PSW, AH	－	REPNC
IMUL	MUL	JZ	BE/BZ	SAL	SHL	－	RETRBI
IN	IN	LAHF	MOV AH, PSW	SAR	SHRA	－	ROL4
INC	INC	LDS	MOV DS0,	SBB	SUBC	－	ROR4
INT	BRK	LEA	LDEA	SCAS	CMPM/	－	STOP
INT 3	BRK 3	LES	MOV DS1,		CMPMB/	－	SUB4S
INTO	BRKV	LOCK	BUSLOCK		CMPMW	－	TEST1
IRET	RETI	LODS	LDM/LDMB/	SHL	SHL	－	TSKSW
JA	BH		LDMW	SHR	SHR		
JAE	BNC/BNL	LOOP	DBNZ	SS：	SS：		
JB	BC/BL	LOOPE	DBNZE	STC	SET1 CY		

備考　－：該当する命令なし

保守／廃止

付録C　プログラムの実行クロック数

V25, V35ファミリの実行クロック数は，厳密にはパイプラインの処理状態などによって以下に示すようにクロック数が変化します。また，CPUパイプラインの動作を予測することは非常に困難です。したがって，プログラムの正確な実行クロック数を机上で計算することも困難になります。プログラム実行時間はインサーキット・エミュレータなどを利用し，実測して求めてください。

次にV25, V35ファミリのCPUパイプラインについて示します。

C.1　同期式パイプライン

V25, V35ファミリのCPUパイプラインは，各ユニットが同期して動作する同期式を採用しています。同時に起動する各ユニットの１パイプライン処理がすべてのユニットで終了したあとに，次のパイプライン処理を一斉に起動します。

また，V25, V35ファミリの各命令は２-４クロックのパイプライン・ステージの組み合わせで実行されています。したがって，実行ユニットでの命令の実行クロック数は，ほかのユニットの処理状態によって変化します。

たとえば，実行ユニットの１パイプライン・ステージが２クロックで終了しても，同時に起動されるバス・コントロール・ユニットの処理時間が３クロックの場合，実行ユニットでは次のパイプライン・ステージに移行するまでに１クロックの待ち時間が発生します。

この場合，命令実行クロック数は，**表２－８　命令実行クロック数一覧**で示した各命令のクロック数より１クロック多くなります。

例　V25の場合

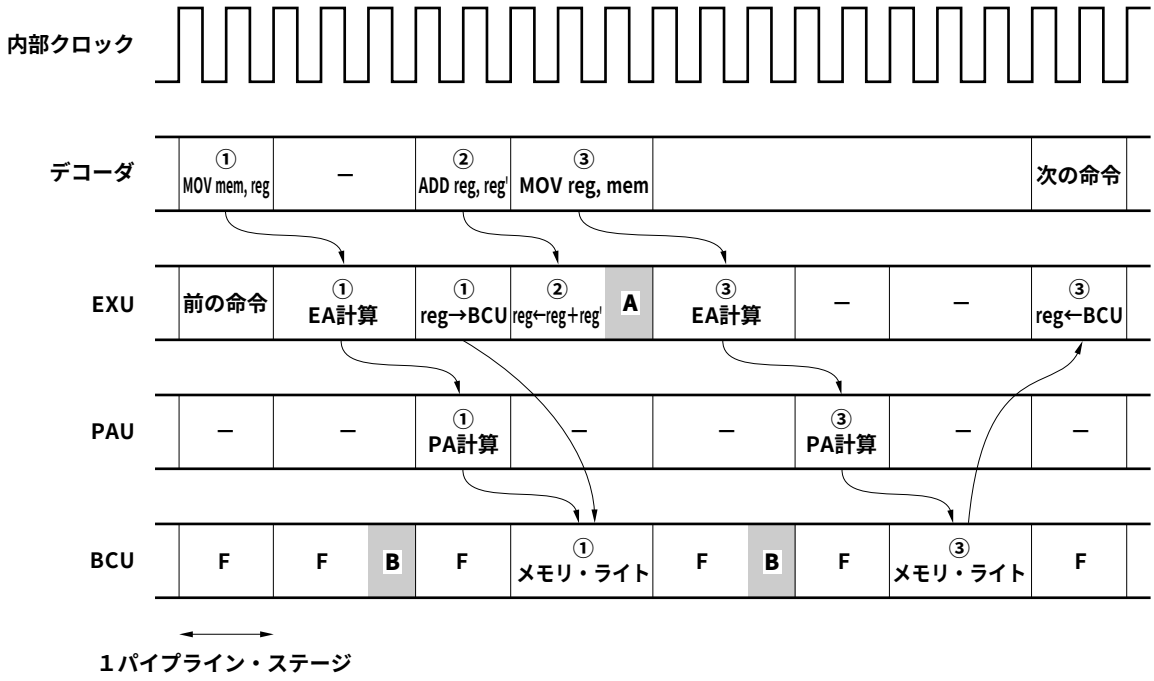
次に示す命令を実行した場合のパイプラインの動きを示します。

```
～
MOV     mem, reg        …①
ADD     reg, reg'       …②
MOV     reg, mem        …③
～
```

例は次の条件を想定しています。

- プリフェッチ・サイクル：２クロック（０ウエイト）
- メモリ・リード／ライト：３クロック（１ウエイト）
- 内部RAMアクセス禁止
- プリフェッチ・キューに常に１バイト以上の空きがあるとき

保守／廃止



A ：BCUのパイプライン・ステージ終了待ち。

（EXUにおける命令実行クロック数に影響を与えます。）

B ：EXUのパイプライン・ステージ終了待ち。

（EXUにおける命令実行クロック数に影響はありません。）

備考　EXU：実行ユニット

PAU：アドレス演算ユニット

BCU：バス・コントロール・ユニット

F：プリフェッチ

C.2　強制プリフェッチ・サイクル

V25, V35ファミリには６バイトのプリフェッチ・キューがあります。通常，キューに１バイト以上の空きがあるとプリフェッチを行います。

キュー内の命令コードが２バイト以下となり，さらに，命令コードのプリフェッチが行われていないときに，次のパイプライン・ステージでは命令の実行を中止し，プリフェッチ・キュー内に３バイト以上の命令コードが格納されるまで強制的にプリフェッチ・サイクルが起動されます。

また，分岐命令，CALL命令などのアドレスが不連続となる場合は，プリフェッチ・キューの内容はクリアされ，新しいロケーションの命令コードを２バイト・フェッチしてから命令実行を開始します。

CPU機能に関しては**各製品のユーザーズ・マニュアル　ハードウエア編**を参照してください。

付録D　86系Cコンパイラ，アセンブラでのプログラム開発

他社86系Cコンパイラとアセンブラを使って，V25, V35ファミリ用プログラムを開発する方法を以下に示します。

D.1　C言語の場合

（1）V25, V35ファミリ・オリジナル命令

V25, V35ファミリの命令は，V20, V30に対して上位互換性があるため，他社の86系Cコンパイラが使用できます。

V25, V35ファミリのオリジナル命令（８種類）を使う場合には，アセンブラで作成した関数を呼び出します。また，インライン展開するCコンパイラであれば，Cソースの中にV25, V35ファミリのオリジナル命令を記述することができます。

（2）特殊機能レジスタ

特殊機能レジスタ（SFR）群は，メモリ上にマッピングされているので，ポインタ変数を使って参照，代入することにより操作することができます。

```
例　struct IDB ＊sfradr；
    sfradr = setidb（0xfe）；　/＊　setidb関数は，idbレジスタに値はセットし，
                                    sfr配置アドレスを返却する関数　　＊/
    sfradr->port0 = 0x10；
```

（3）レジスタ・バンク操作

レジスタ・バンクは，メモリ上にマッピングされているので，ポインタ変数を使って参照，代入することにより操作することができます。

```
例　struct REGBNK ＊ bnk；
    long  tmp；
    bnk =（struct REGBNK ＊）BNK_ADR；
    tmp =（long）int_handler；　　/＊ int_handler is a function ＊/
    bnk. vec_pc =（int）（tmp & 0x0000ffff）；
    bnk. ps　　=（int）（（tmp & 0xffff0000）＞＞16）；
```

保守／廃止

D.2　アセンブラの場合

（1）V25, V35オリジナル命令

V25, V35オリジナル命令は，マクロ機能を使って記述します。

```
例          BTCLR NO_P0 7 dummy
            ..........
    dummy： nop
```

（2）特殊機能レジスタ

特殊機能レジスタ（SFR）群は，C言語と同じく構造体を宣言して操作します。

```
例　MOV al, sfr, p0
```

D.3　インクルード・ファイル／マクロ・ファイル例

C言語用ヘッダ・ファイルとアセンブラ用マクロ・ファイルの記述例を次ページ以降に示します。次ページ以降のリストを作成したら，C言語の場合にはC言語用ファイルをインクルードしてください。アセンブラの場合にはアセンブラ用ファイルをインクルードしてください。

注意1．添付リストでは，C言語用ヘッダ・ファイルはV25, V35用になっています。アセンブラ用ヘッダ・ファイルはV25＋, V35＋用になっています。
C言語でV25＋, V35＋のプログラム開発を行うときは，V25, V35用のC言語用ヘッダ・ファイルをV25＋, V35＋用に修正してください。
アセンブラでV25, V35のプログラム開発を行うときは，V25＋, V35＋用のアセンブラ用ヘッダ・ファイルをV25, V35用に修正してください。
修正する箇所は，特殊機能レジスタ（SFR）構造体のメンバです。

2．添付リストは，厳密な検査を行っていません。したがって，使用するコンパイラで十分な評価を行って使用してください。特に，最適化を行うコンパイラでは，SFRへ値を代入しても（それを参照しない場合）Cコンパイラが冗長な命令と解釈して，コード生成を行わない場合があります。

```
/*
        THIS STRUCTURE IS DEFINED TO THE SPECIAL FACULTY REGISTER OF V25/V35.

        DATE    08 JULY88
        Copyright (C) NEC Corporation 1988
*/

struct  SFR
        {
        char    port0;
        char    portm0;
        char    portmc0;
        char    dummy1[5];
        char    port1;
        char    portm1;
        char    portmc1;
        char    dummy2[5];
        char    port2;
        char    portm2;
        char    portmc2;
        char    dummy3[37];
        char    portT;
        char    dummy4[2];
        char    portmT;
        char    dummy5[4];
        char    intm;
        char    dummy6[3];
        char    ems0;
        char    ems1;
        char    ems2;
        char    dummy7[5];
        char    exic0;
        char    exic1;
        char    exic2;
        char    dummy8[17];
        char    rxb0;
        char    dummy9;
        char    txb0;
        char    dummy10[2];
        char    srms0;
        char    stms0;
        char    dummy11;
        char    scm0;
        char    scc0;
        char    brg0;
        char    sce0;
        char    seic0;
        char    sric0;
        char    stic0;
        char    dummy12;
        char    rxb1;
        char    dummy13;
        char    txb1;
        char    dummy14[2];
        char    srms1;
        char    stms1;
        char    dummy15;
        char    scm1;
        char    scc1;
        char    brg1;
        char    sce1;
```

```
        char    seic1;
        char    sric1;
        char    stic1;
        char    dummy16;
        int     tm0;
        int     md0;
        int     dummy17[2];
        int     tm1;
        int     md1;
        INT     dummy18[2];
        char    tmc0;
        char    tmc1;
        char    dummy19[2];
        char    tmms0;
        char    tmms1;
        char    tmms2;
        char    dummy20[5];
        char    tmic0;
        char    tmic1;
        char    tmic2;
        char    dummy21;
        char    dmac0;
        char    dmam0;
        char    dmac1;
        char    dmam1;
        char    dummy22[8];
        char    dic0;
        char    dic1;
        char    dummy23[50];
        char    stbc;
        char    rfm;
        char    dummy24[6];
        int     wtc;
        char    flag;
        char    prc;
        char    tbic;
        char    dummy25[15];
        char    ispr;
        char    dummy26[2]
        char    idb;
        } *sfr;
/*
        THIS DEFINE NAME IS SPECIAL FACULTY REGISTER NAME.

        DATE    22 AUG 88
        Copyright (C) NEC Corporation 1988
*/


#define         P0              sfr->port0
#define         PM0             sfr->portm0
#define         PMC0            sfr->portmc0
#define         P1              sfr->port1
#define         PM1             sfr->portm1
#define         PMC1            sfr->portmc1
#define         P2              sfr->port2
#define         PM2             sfr->portm2
#define         PMC2            sfr->portmc2
#define         PT              sfr->portT
#define         PMT             sfr->portmT
#define         INTM            sfr->intm
```

```
#define         EMS0            sfr->ems0
#define         EMS1            sfr->ems1
#define         EXIC0           sfr->exic0
#define         EXIC1           sfr->exic1
#define         EXIC2           sfr->exic2
#define         RXB0            sfr->rxb0
#define         TXB0            sfr->txb0
#define         SRMS0           sfr->srms0
#define         STMS0           sfr->stms0
#define         SCM0            sfr->scm0
#define         SCC0            sfr->scc0
#define         BRG0            sfr->brg0
#define         SCE0            sfr->sce0
#define         SEIC0           sfr->seic0
#define         SRIC0           sfr->sric0
#define         STIC0           sfr->stic0
#define         RXB1            sfr->rxb1
#define         TXB1            sfr->txb1
#define         SRMS1           sfr->srms1
#define         STMS1           sfr->stms1
#define         SCM1            sfr->scm1
#define         SCC1            sfr->scc1
#define         BRG1            sfr->brg1
#define         SCE1            sfr->sce1
#define         SEIC1           sfr->seic1
#define         SRIC1           sfr->sric1
#define         STIC1           sfr->stic1
#define         TM0             sfr->tm0
#define         MD0             sfr->md0
#define         TM1             sfr->tm1
#define         MD1             sfr->md1
#define         TMC0            sfr->tmc0
#define         TMC1            sfr->tmc1
#define         TMMS0           sfr->tmms0
#define         TMMS1           sfr->tmms1
#define         TMMS2           sfr->tmms2
#define         TMIC0           sfr->tmic0
#define         TMIC1           sfr->tmic1
#define         TMIC2           sfr->tmic2
#define         DMAC0           sfr->dmac0
#define         DMAM0           sfr->dmam0
#define         DMAC1           sfr->dmac1
#define         DMAM1           sfr->dmam1
#define         DIC0            sfr->dic0
#define         DIC1            sfr->dic1
#define         STBC            sfr->stbc
#define         RFM             sfr->rfm
#define         WTC             sfr->wtc
#define         FLAG            sfr->flag
#define         PRC             sfr->prc
#define         TBIC            sfr->tbic
#define         ISPR            sfr->ispr
#define         IDB             sfr->idb
/**/
/*
        THIS DEFINE NAME ARE FUCTIONS TO GIVE ACCESS
                                        TO SPECIAL FACULTY REGISTER
        DATE    22 AUG 88
        Copyright (C) NEC Corporation 1988
*/
#define         c_dis0()        (SEIC0 |=0x40)
```

```
#define         c_dis1()        (SEIC1 |=0x40)
#define         c_ena0()        (SEIC0 &=0xbf)
#define         c_ena1()        (SEIC1 &=0xbf)
#define         c_rdis0()       (SRIC0 |=0x40)
#define         c_rdis1()       (SRIC1 |=0x40)
#define         c_rena0()       (SRIC0 &=0xbf)
#define         c_rena1()       (SRIC1 &=0xbf)
#define         c_read0()       (RXB0)
#define         c_read1()       (RXB1)
#define         c_tdis0()       (STIC0 |=0x40)
#define         c_tdis1()       (STIC1 |=0x40)
#define         c_tena0()       (STIC0 &=0xbf)
#define         c_tena1()       (STIC1 &=0xbf)
#define         c_trns0(data)   (TXB0  = data)
#define         c_trns1(data)   (TXB1  = data)
#define         c_tstrt0()      (SCM0  |=0x40)
#define         c_tstrt1()      (SCM1  |=0x40)
#define         c_tstop0()      (SCM0  &=0xbf)
#define         c_tstop1()      (SCM1  &=0xbf)
#define         d_disa0()       (DIC0  |=0x40)
#define         d_disa1()       (DIC1  |=0x40)
#define         d_ena0()        (DIC0  &=0xbf)
#define         d_ena1()        (DIC1  &=0xbf)
#define         p_read0()       (P0)
#define         p_read1()       (P1)
#define         p_read2()       (P2)
#define         p_readt0()      (PT)
#define         p_write0(data)  (P0  =data)
#define         p_write1(data)  (P1  =data)
#define         t_disa0()       (TMIC0 |=0x40)
#define         t_disa1()       (TMIC1 |=0x40)
#define         t_ena0()        (TMIC0 &=0x40)
#define         t_ena1()        (TMIC1 &=0x40)
#define         t_start0()      (TMC0 |=0x80)
#define         t_start1()      (TMC1 |=0x80)
#define         t_stop0()       (TMC0 &=0x7f)
#define         t_stop1()       (TMC1 &=0x7f)
#define         d_hold0()       (DMAM0 =(DMAM0 | 0x08) & 0xfc)
#define         d_hold1()       (DMAM1 =(DMAM1 | 0x08) & 0xfc)
#define         d_start0()      (DMAM0 =DMAM0 | 0xc)
#define         d_start1()      (DMAM1 =DMAM1 | 0xc)
/*
*/
#define         ON              1
#define         OFF             0

struct  REGBNK  {
        int     reserve ;
        int     vec_pc ;
        int     save_psw ;
        int     save_pc ;
        int     ds0 ;
        int     ss ;
        int     ps ;
        int     ds1 ;
        int     iy ;
        int     ix ;
        int     bp ;
        int     sp ;
        int     bw ;
        int     dw ;
```


```
        int     cw ;
        int     aw ;
};
```

```
;  ************************************************************************
;  *            Sample for V25 programing                                 *
;  *                                  for MASM V5.1                       *
;  *                                       (C) NEC Corp. 1990             *
;  ************************************************************************
;
; *********************************
;       SFR MACRO(for uPD70320)
; *********************************
;
SFR_320 struc
        P0      db      ?       ;   BYTE  1   xxF00H
        PM0     db      ?       ;   BYTE  1   xxF01H
        PMC0    db      ?       ;   BYTE  1   xxF02H
        DMY03   db      5       dup(?)

        P1      db      ?       ;   BYTE  1   xxF08H
        PM1     db      ?       ;   BYTE  1   xxF09H
        PMC1    db      ?       ;   BYTE  1   xxF0AH

        DMY0B   db      5       dup(?)

        P2      db      ?       ;   BYTE  1   xxF10H
        PM2     db      ?       ;   BYTE  1   xxF11H
        PMC2    db      ?       ;   BYTE  1   xxF12H

        DMY13   db      37      dup(?)

        PT      db      ?       ;   BYTE  1   xxF38H
        DMY39   db      2       dup(?)
        PMT     db      ?       ;   BYTE  1   xxF3BH

        DMY3C   db      4       dup(?)

        INTM    db      ?       ;   BYTE  1   xxF40H
        DMY41   db      3       dup(?)  ;   BYTE  1   xxF41H

        EMS0    db      ?       ;   BYTE  1   xxF44H
        EMS1    db      ?       ;   BYTE  1   xxF45H
        EMS2    db      ?       ;   BYTE  1   xxF46H

        DMY47   db      5       dup(?)  ;   BYTE  1   xxF47H

        EXIC0   db      ?       ;   BYTE  1   xxF4CH
        EXIC1   db      ?       ;   BYTE  1   xxF4DH
        EXIC2   db      ?       ;   BYTE  1   xxF4EH
        DMY4F   db      17      dup(?)  ;   BYTE  1   xxF4FH

        RXB0    db      ?       ;   BYTE  1   xxF60H
        DMY61   db      ?       ;   BYTE  1   xxF61H
        TXB0    db      ?       ;   BYTE  1   xxF62H
        DMY63   db      2       dup(?)  ;   BYTE  1   xxF63H
        SRMS0   db      ?       ;   BYTE  1   xxF65H
        STMS0   db      ?       ;   BYTE  1   xxF66H

        DMY67   db      ?       ;   BYTE  1   xxF67H

        SCM0    db      ?       ;   BYTE  1   xxF68H
        SCC0    db      ?       ;   BYTE  1   xxF69H
        BRG0    db      ?       ;   BYTE  1   xxF6AH
        SCE0    db      ?       ;*  BYTE  1   xxF6BH
```

```
SEIC0   db      ?       ;   BYTE  1   xxF6CH
SRIC0   db      ?       ;   BYTE  1   xxF6DH
STIC0   db      ?       ;   BYTE  1   xxF6EH
DMY6F   db      ?       ;   BYTE  1   xxF6FH

RXB1    db      ?       ;   BYTE  1   xxF70H
DMY71   db      ?       ;   BYTE  1   xxF71H
TXB1    db      ?       ;   BYTE  1   xxF72H
DMY73   db      ?       ;   BYTE  1   xxF73H
DMY74   db      ?       ;   BYTE  1   xxF74H
SRMS1   db      ?       ;   BYTE  1   xxF75H
STMS1   db      ?       ;   BYTE  1   xxF76H
DMY77   db      ?       ;   BYTE  1   xxF77H
SCM1    db      ?       ;   BYTE  1   xxF78H
SCC1    db      ?       ;   BYTE  1   xxF79H
BRG1    db      ?       ;   BYTE  1   xxF7AH
SCE1    db      ?       ;*  BYTE  1   xxF7BH
SEIC1   db      ?       ;   BYTE  1   xxF7CH
SRIC1   db      ?       ;   BYTE  1   xxF7DH
STIC1   db      ?       ;   BYTE  1   xxF7EH
DMY7F   db      ?       ;   BYTE  1   xxF7FH

TM0     dw      ?       ;   WORD  1   xxF80H
MD0     dw      ?       ;   WORD  1   xxF82H
DMY84   db      4       dup(?)  ;   BYTE  1   xxF84H
TM1     dw      ?       ;   WORD  1   xxF88H
MD1     dw      ?       ;   WORD  1   xxF8AH
DMY8C   db      4       dup(?)  ;   BYTE  1   xxF8CH
TMC0    db      ?       ;   BYTE  1   xxF90H
TMC1    db      ?       ;   BYTE  1   xxF91H
DMY92   db      2       dup(?)  ;   BYTE  1   xxF92H
TMMS0   db      ?       ;   BYTE  1   xxF94H
TMMS1   db      ?       ;   BYTE  1   xxF95H
TMMS2   db      ?       ;   BYTE  1   xxF96H
DMY97   db      5       dup(?)  ;   BYTE  1   xxF97H
TMIC0   db      ?       ;   BYTE  1   xxF9CH
TMIC1   db      ?       ;   BYTE  1   xxF9DH
TMIC2   db      ?       ;   BYTE  1   xxF9EH
DMY9F   db      ?       ;   BYTE  1   xxF9FH

DMAC0   db      ?       ;   BYTE  1   xxFA0H
DMAM0   db      ?       ;   BYTE  1   xxFA1H
DMAC1   db      ?       ;   BYTE  1   xxFA2H
DMAM1   db      ?       ;   BYTE  1   xxFA3H
DMYA4   db      8       dup(?)  ;   BYTE  1   xxFA4H
DIC0    db      ?       ;   BYTE  1   xxFACH
DIC1    db      ?       ;   BYTE  1   xxFADH
DMYAE   db      50      dup(?)  ;   BYTE  1   xxFAEH

STBC    db      ?       ;   BYTE  1   xxFE0H
RFM     db      ?       ;   BYTE  1   xxFE1H
DMYE2   db      6       dup(?)  ;   BYTE  1   xxFE2H
WTC     db      ?       ;   BYTE  1   xxFE8H
DMYE9   db      ?       ;   BYTE  1   xxFE9H
FLAG    db      ?       ;   BYTE  1   xxFEAH
PRC     db      ?       ;   BYTE  1   xxFEBH
TBIC    db      ?       ;   BYTE  1   xxFECH
DMYED   db      15      dup(?)  ;   BYTE  1   xxFEDH
ISPR    db      ?       ;*  BYTE  1   xxFFCH
DMYFD   db      2       dup(?)  ;   BYTE  1   xxFFDH
IDB     db      ?       ;   BYTE  1   xxFFFH
```

```
SFR_320 ends

;
;  ************************************
;  *  SFR Number for BTCLR instruction
;  ************************************
;

NO_P0           EQU     00h
NO_PM0          EQU     01h
NO_PMC0         EQU     02h

NO_P1           EQU     08h
NO_PM1          EQU     09h
NO_PMC1         EQU     0ah

NO_P2           EQU     010h
NO_PM2          EQU     011h
NO_PMC2         EQU     012h

NO_PT           EQU     038h
NO_PMT          EQU     03bh

NO_INTM         EQU     040h
NO_EMS0         EQU     044h
NO_EMS1         EQU     045h
NO_EMS2         EQU     046h
NO_EXIC0        EQU     04ch
NO_EXIC1        EQU     04dh
NO_EXIC2        EQU     04eh

NO_RXB0         EQU     060h
NO_TXB0         EQU     062h
NO_SRMS0        EQU     065h
NO_STMS0        EQU     066h
NO_SCM0         EQU     068h
NO_SCC0         EQU     069h
NO_BRG0         EQU     06ah
NO_SCE0         EQU     06bh
NO_SEIC0        EQU     06ch
NO_SRIC0        EQU     06dh
NO_STIC0        EQU     06eh

NO_RXB1         EQU     070h
NO_TXB1         EQU     072h
NO_SRMS1        EQU     075h
NO_STMS1        EQU     076h
NO_SCM1         EQU     078h
MP_SCC1         EQU     079h
NO_BRG1         EQU     07ah
NO_SCE1         EQU     07bh
NO_SEIC1        EQU     07ch
NO_SRIC1        EQU     07dh
NO_STIC1        EQU     07eh

NO_TM0          EQU     080h
NO_MD0          EQU     082h
NO_TM1          EQU     088h
NO_MD1          EQU     08ah
NO_TMC0         EQU     090h
NO_TMC1         EQU     091h
NO_TMMS0        EQU     094h
```

```
NO_TMMS1        EQU     095h
NO_TMMS2        EQU     096h
NO_TMIC0        EQU     09ch
NO_TMIC1        EQU     09dh
NO_TMIC2        EQU     09eh

NO_DMAC0        EQU     0a0h
NO_DMAM0        EQU     0a1h
NO_DMAC1        EQU     0a2h
NO_DMAM1        EQU     0a3h
NO_DIC0         EQU     0ach
NO_DIC1         EQU     0adh

NO_STBC         EQU     0e0h
NO_RFM          EQU     0e1h
NO_WTC          EQU     0e8h
NO_FLAG         EQU     0eah
NO_PRC          EQU     0ebh
NO_TBIC         EQU     0ech
NO_ISPR         EQU     0fch
NO_IDB          EQU     0ffh

;
;

; *********************************
;      V25/V35  Extend instructions
; *********************************

BTCLR   macro sfr,imm3,br_label
        db      0fh,9ch
        db      sfr,imm3,offset br_label - $  -1
        endm

RETRBI  macro
        db      0fh,91h
        endm

FINT    macro
        db      0fh,92h
        endm

STOP    macro
        db      0fh,9eh
        endm

BRKCS   macro   reg16
ifidni  <reg16>,<ax>    ;; if reg16 is ax or AX
        db      0fh, 2dh, 0c0h
endif
ifidni  <reg16>,<bx>    ;; if reg16 is bx or BX
        db      0fh, 2dh, 0c3h
endif
ifidni  <reg16>,<cx>    ;; if reg16 is cx or CX
        db      0fh, 2dh, 0c1h
endif
ifidni  <reg16>,<dx>    ;; if reg16 is dx or DX
        db      0fh, 2dh, 0c2h
endif
        endm
```


```
TSKSW   macro   reg16
ifidni  <reg16>,<ax>    ;; if reg16 is ax or AX
        db      0fh,94h,0f8h
endif
ifidni  <reg16>,<bx>    ;; if reg16 is bx or BX
        db      0fh,94h,0ffh
endif
ifidni  <reg16>,<cx>    ;; if reg16 is cx or CX
        db      0fh,94h,0f9h
endif
ifidni  <reg16>,<dx>    ;; if reg16 is dx or DX
        db      0fh,94h,0fah
endif
        endm

MOVSPA  macro
        db      0fh,25h
        endm

MOVSPB  macro  reg16
ifidni  <reg16>,<ax>    ;; if reg16 is ax or AX
        db      0fh,95h,0f8h
endif
ifidni  <reg16>,<bx>    ;; if reg16 is bx or BX
        db      0fh,95h,0ffh
endif
ifidni  <reg16>,<cx>    ;; if reg16 is cx or CX
        db      0fh,95h,0f9h
endif
ifidni  <reg16>,<dx>    ;; if reg16 is dx or DX
        db      0fh,95h,0fah
endif
        endm
```

保守／廃止

付録Ｅ　命令索引（ニモニック：機能別）

保守／廃止

【セグメント・オーバライド・プリフィクス】

【レジスタ・バンク切り替え】

〔メ　モ〕

保守／廃止

付録Ｆ　命令索引（ニモニック：アルファベット順）

保守／廃止

〔メ　モ〕

保守／廃止

お問い合わせは，最寄りのNECへ

【営業関係お問い合わせ先】

半導体第一販売事業部 半導体第二販売事業部 半導体第三販売事業部	〒108-01　東京都港区芝五丁目７番１号（NEC本社ビル）	東　京	(03)3454−1111　(大代表)
中部支社 半導体第一販売部 半導体第二販売部	〒460　名古屋市中区錦一丁目17番１号（NEC中部ビル）	名古屋 名古屋	(052)222−2170 (052)222−2190
関西支社 半導体第一販売部 半導体第二販売部 半導体第三販売部	〒540　大阪市中央区城見一丁目４番24号（NEC関西ビル）	大　阪 大　阪 大　阪	(06) 945−3178 (06) 945−3200 (06) 945−3208

北海道支社	札　幌 (011)251−5599	太田支店	太　田 (0276)46−4011	福井支店	福　井 (0776)22−1866
東北支社	仙　台 (022)267−8740	宇都宮支店	宇都宮 (028)621−2281	富山支店	富　山 (0764)31−8461
岩手支店	盛　岡 (019)651−4344	小山支店	小　山 (0285)24−5011	三重支店	津　 (0592)25−7341
郡山支店	郡　山 (0249)23−5511	長野支社	松　本 (0263)35−1662	京都支社	京　都 (075)344−7824
いわき支店	いわき (0246)21−5511	甲府支店	甲　府 (0552)24−4141	神戸支社	神　戸 (078)333−3854
長岡支店	長　岡 (0258)36−2155	埼玉支社	大　宮 (048)649−1415	中国支社	広　島 (082)242−5504
土浦支店	土　浦 (0298)23−6161	立川支社	立　川 (0425)26−5981	鳥取支店	鳥　取 (0857)27−5311
水戸支店	水　戸 (029)226−1717	千葉支社	千　葉 (043)238−8116	岡山支店	岡　山 (086)225−4455
神奈川支社	横　浜 (045)682−4524	静岡支社	静　岡 (054)254−4794	松山支店	松　山 (089)945−4149
群馬支店	高　崎 (0273)26−1255	北陸支社	金　沢 (076)232−7303	九州支社	福　岡 (092)261−2806

【本資料に関する技術お問い合わせ先】

半導体ソリューション技術本部 マイクロコンピュータ技術部	〒210　川崎市幸区塚越三丁目484番地	川　崎	(044)548−7924
半導体販売技術本部 東日本販売技術部	〒108-01　東京都港区芝五丁目7番1号（NEC本社ビル）	東　京	(03)3798−9619
半導体販売技術本部 中部販売技術部	〒460　名古屋市中区錦一丁目17番１号（NEC中部ビル）	名古屋	(052)222−2125
半導体販売技術本部 西日本販売技術部	〒540　大阪市中央区城見一丁目４番24号（NEC関西ビル）	大　阪	(06) 945−3383

半導体
インフォメーションセンター
FAX(044)548−7900
（FAXにてお願い致します）

保守／廃止

アンケート記入のお願い

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

［ドキュメント名］V25, V35ファミリ　ユーザーズ・マニュアル　命令編
（U12120JJ3V0UM00（第３版））

［お名前など］（さしつかえのない範囲で）

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

１．ご評価（各欄に○をご記入ください）

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

２．わかりやすい所（第　章，第　章，第　章，第　章，その他　　）
理由［　　］

３．わかりにくい所（第　章，第　章，第　章，第　章，その他　　）
理由［　　］

４．ご意見，ご要望

５．このドキュメントをお届けしたのは
NEC販売員，特約店販売員，NEC半導体ソリューション技術本部員，
その他（　　）

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

NEC半導体インフォメーションセンター
FAX：（044）548-7900

キリトリ