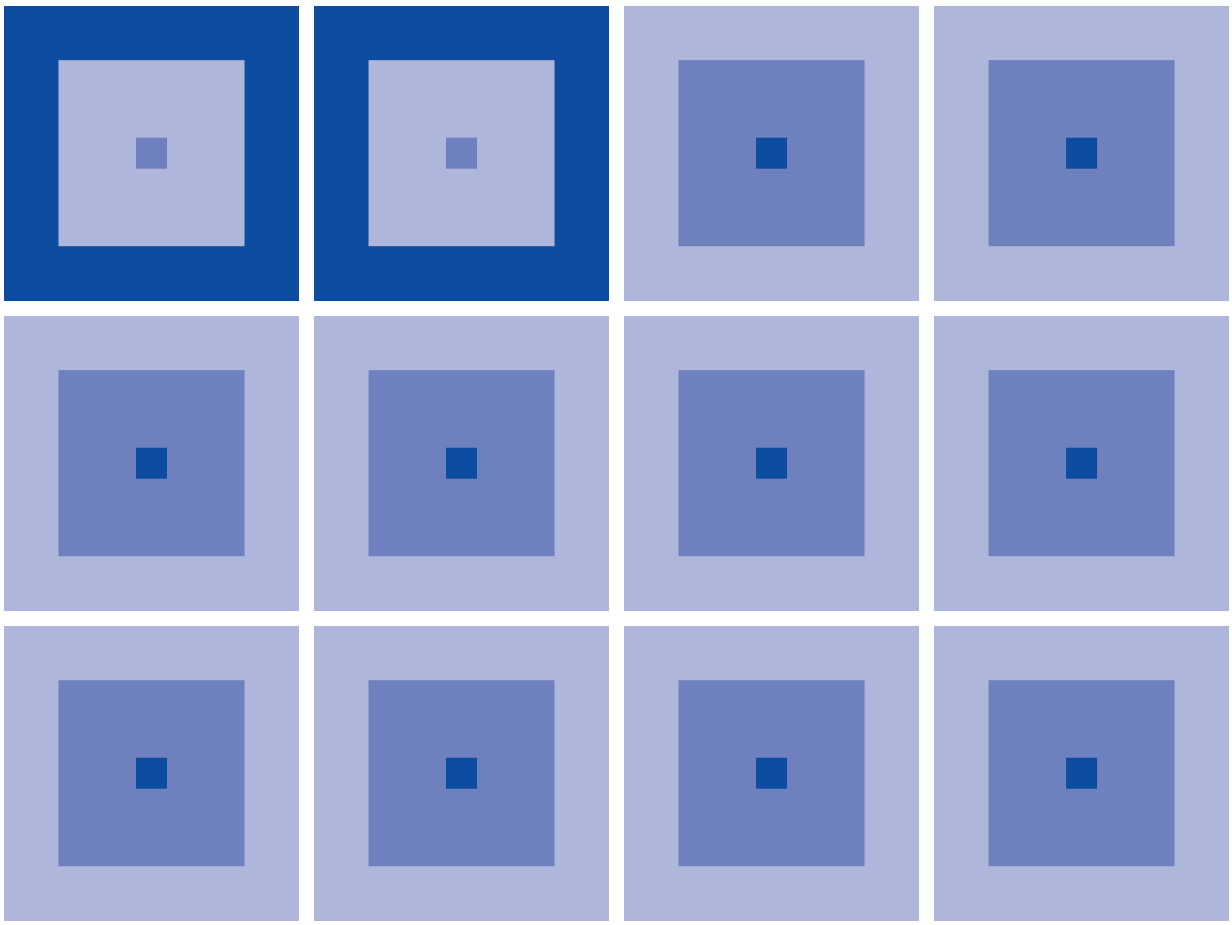
MF657-05

EPSON

CMOS 8-BIT SINGLE CHIP MICROCOMPUTER

S1C88
コアCPUマニュアル



セイコーエプソン株式会社

本資料のご使用につきましては、次の点にご留意願います。

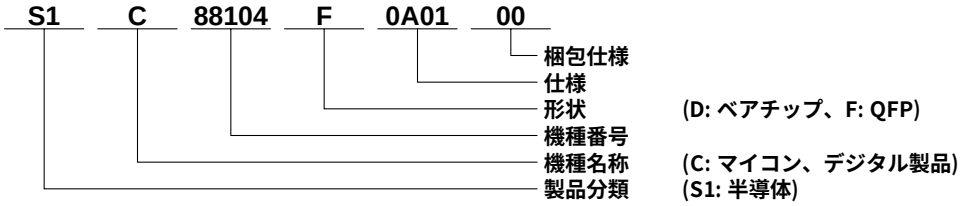
1. 本資料の内容については、予告なく変更することがあります。
2. 本資料の一部、または全部を弊社に無断で転載、または、複製など他の目的に使用することは堅くお断りします。
3. 本資料に掲載される応用回路、プログラム、使用方法等はあくまでも参考情報であり、これらに起因する第三者の権利(工業所有権を含む)侵害あるいは損害の発生に対し、弊社は如何なる保証を行うものではありません。また、本資料によって第三者または弊社の工業所有権の実施権の許諾を行うものではありません。
4. 特性表の数値の大小は、数直線上の大小関係で表しています。
5. 本資料に掲載されている製品のうち、「外国為替および外国貿易法」に定める戦略物資に該当するものについては、輸出する場合、同法に基づく輸出許可が必要です。
6. 本資料に掲載されている製品は、一般民生用です。生命維持装置その他、きわめて高い信頼性が要求される用途を前提としていません。よって、弊社は本(当該)製品をこれらの用途に用いた場合の如何なる責任についても負いかねます。

製品型番変更のご案内

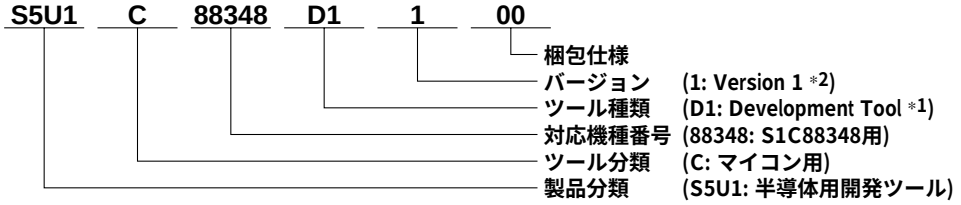
2001年4月1日より、弊社半導体製品の製品型番が以下のとおり変更となりますので、4月1日以降のご発注につきましては変更後の製品型番にてお願い申し上げます。
なお、製品型番の詳細仕様につきましては、弊社営業担当にお問い合わせください。

製品型番体系

●デバイス



●開発ツール



*1: ツールの種類は、新旧型番対応表を参照してください。(マニュアル類には一桁で記載されているものもあります。)
*2: マニュアル類には、実際のバージョンは記載されておりません。

新旧型番対応表

●S1C88ファミリ

旧型番	新型番
E0C88104	S1C88104
E0C88112	S1C88112
E0C88308	S1C88308
E0C88316	S1C88316
E0C88317	S1C88317
E0C88348	S1C88348
E0C88P348	S1C8P348
E0C88349	S1C88349

旧型番	新型番
E0C88365	S1C88365
E0C88F360	S1C8F360
E0C88408	S1C88408
E0C88409	S1C88409
E0C88816	S1C88816
E0C88832	S1C88832
E0C88862	S1C88862
E0C88F816	S1C8F816

開発ツール新旧型番対応表

●S1C88ファミリ関係の開発ツール

旧型番	新型番
88ISAIF	S5U1C88000H4
ADP88348	S5U1C88348X
ADP88360	S5U1C88360X
DEV88104	S5U1C88104D
DEV88112	S5U1C88112D
DEV88308	S5U1C88308D
DEV88316	S5U1C88316D
DEV88317	S5U1C88317D
DEV88348	S5U1C88348D
DEV88365	S5U1C88365D
DEV88408	S5U1C88408D
DEV88409	S5U1C88409D

旧型番	新型番
DEV88816	S5U1C88816D
DEV88832	S5U1C88832D
DEV88862	S5U1C88862D
DMT88348-DB	S5U1C88348T
ICE88UR	S5U1C88000H5
PRC88316	S5U1C88316P
PRC88348	S5U1C88348P
PRC88365	S5U1C88365P
PRC88409	S5U1C88409P
PRC88816	S5U1C88816P
SAP88	S5U1C88000S
URS88348	S5U1C88348Y

●S1C63/88ファミリ関係の開発ツール

旧型番	新型番
ADS00002	S5U1C88000X1
GWH00002	S5U1C88000W2
URM00002	S5U1C88000W1

CMOS 8-bit Single Chip Microcomputer

S1C88 Core CPU Manual

はじめに

本書はCMOS 8ビットシングルチップマイクロコンピュータS1C88 FamilyのコアCPU S1C88のアーキテクチャとその動作および命令セットについて解説します。

なお、S1C88 Familyの各デバイスごとにメモリ構成や周辺回路の構成が異なりますので、基本機能以外の内容についてはそれぞれのマニュアルをご覧ください。

- 目 次 -

1 概要

S1C88はEPSONオリジナルのアーキテクチャを採用したCMOS 8ビットシングルチップマイクロコンピュータS1C88 Familyのコアcpuです。
最大16Mバイトのアドレス空間と高速・豊富な命令セットを持ち、広範囲な動作電圧に対応するとともに低消費電力を特長としています。
また、統一アーキテクチャのもとにメモリマップドI/O方式の周辺回路インタフェースを採用しており、今後のFamily拡充にも柔軟に対応します。

1.1 特長

S1C88は以下の特長を持っています。

アドレス空間 最大16Mバイト
命令サイクル 1～15サイクル(1サイクル=2クロック)
命令セット 608種類
レジスタ構成 データレジスタ 2
インデックスレジスタ 3
(1つはデータレジスタとして使用可)
プログラムカウンタ
スタックポインタ
システムコンディションフラグ
カスタマイズコンディションフラグ
例外処理要因 リセット、ゼロ除算、割り込み
例外処理ベクタ 最大128ベクタ
スタンバイ機能 HALT/SLEEP
周辺回路インタフェース メモリマップドI/O方式

1.2 命令セットの特長

(1) マシンサイクル効率の高い、高速かつ豊富な命令セットを採用しています。

(2) 12種類のアドレッシングモードにより、メモリ管理が容易に行えます。

(3) アドレス計算などに有効な16ビット演算機能を持っています。

(4) 10進演算モード、パック/アンパック命令など、強力な10進演算機能があります。

(5) カストマイズドフラグ命令により、各種の特定用途向けマイコンの実現をサポートしています。

(6) リロケータブルプログラミングが可能な命令体系となっており、ソフトウェアのライブラリ化が容易に実現できます。

1.3 ブロック図

図1.3.1にS1C88のブロック図を示します。

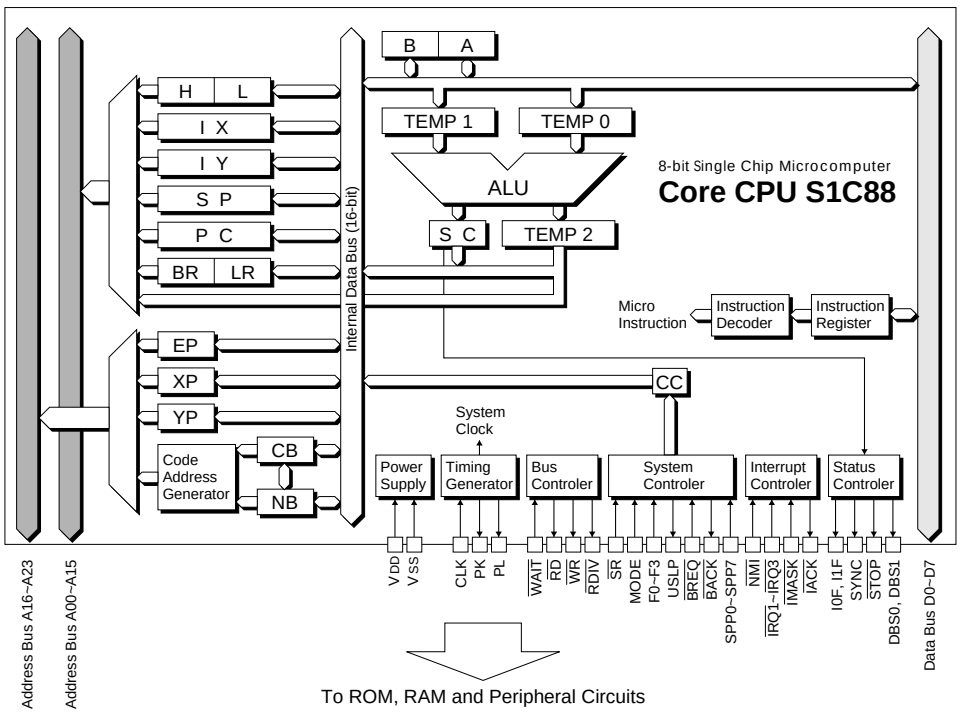
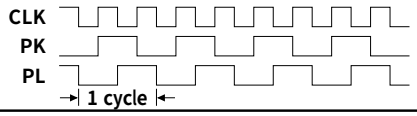


図1.3.1 S1C88ブロック図

1.4 入出力信号

表1.4.1(a)、表1.4.1(b)にS1C88と周辺回路間の入出力信号を示します。

表1.4.1(a) 入出力信号一覧 (1)

種 別	名 称	信 号 名	入出力	機 能
電源	電源	V_{DD}	入力	＋側電源を入力します。
	グランド	V_{SS}	入力	－側電源(GND)を入力します。
クロック	クロック入力	CLK	入力	周辺回路からのシステムクロックを入力します。
	クロック出力	PK PL	出力	CLK端子に入力されるシステムクロックの2相分周信号を以下の位相で出力します。 
アドレスバス	アドレスバス	A00~A23	出力	24ビットのアドレスバスです。
データバス	データバス	D0~D7	入出力	8ビットの双方向データバスです。
バス制御信号	ウェイト	$\overline{WAIT}$	入力	メモリアクセス時のアクセスタイム伸長用ウェイトステートの挿入を制御します。 LOWレベルの入力で有効です。
	リード	$\overline{RD}$	出力	メモリ(周辺回路)リード信号です。 データの読み出し時にLOWレベルとなります。
	ライト	$\overline{WR}$	出力	メモリ(周辺回路)ライト信号です。 データの書き込み時にLOWレベルとなります。
	リード割り込みベクタアドレス	$\overline{RDIV}$	出力	割り込みベクタアドレスリード信号です。ベクタアドレスの読み出し時にLOWレベルとなります。
システム制御信号	リセット	$\overline{SR}$	入力	LOWレベルの入力によりCPUがリセット状態となります。
	モード設定	MODE	入力	CPUの動作モードを周辺回路より設定します。 LOW: ミニマムモード HIGH: マキシマムモード
	カスタマイズコンディションフラグ	F0~F3	入力	周辺回路より入力されるステータス信号です。 周辺回路により信号の意味が異なります。
	マイクロスリープ	USLP	出力	CPUがSLP(SLEEP)命令によりSLEEP状態に入る1サイクル前にHIGHレベルとなります。本信号をもとに周辺回路は発振停止の制御を行います。
	バス権要求	$\overline{BREQ}$	入力	周辺回路がDMA転送を行う場合のバス権要求信号です。本端子へのLOWレベル入力によりCPUはバスを解放します。アドレスバス、データバス、およびリード/ライト信号がハイインピーダンス状態となります。
	バス権要求アクノリッジ	$\overline{BACK}$	出力	バス権を周辺回路に解放したことを示す応答信号です。バス権解放時にLOWレベルとなります。
	スタックポインタページ	SPP0~SPP7	入力	周辺回路により指定されるスタックポインタのページアドレスです。スタックポインタがメモリをアクセスする際に、このアドレスがアドレスバスのページ部(AD16～AD23)に出力されます。

各信号のタイミング等については、3章の"CPUの動作と処理状態"を参照してください。

表1.4.1(b) 入出力信号一覧 (2)

種 別	名 称	信 号 名	入出力	機 能
割り込み信号	ノンマスカブル割り込み	$\overline{\text{NMI}}$	入力	ソフトウェアによるマスクが不可能な割り込み信号です。入力は立ち下がりエッジにてセンスされます。
	割り込み要求3	$\overline{\text{IRQ3}}$	入力	ソフトウェアによるマスクが可能な割り込み信号です。割り込み優先度はレベル3で、入力はLOWレベルにてセンスされます。
	割り込み要求2	$\overline{\text{IRQ2}}$	入力	ソフトウェアによるマスクが可能な割り込み信号です。割り込み優先度はレベル2で、入力はLOWレベルにてセンスされます。
	割り込み要求1	$\overline{\text{IRQ1}}$	入力	ソフトウェアによるマスクが可能な割り込み信号です。割り込み優先度はレベル1で、入力はLOWレベルにてセンスされます。
	割り込みマスク	$\overline{\text{IMASK}}$	入力	周辺回路より入力される割り込みマスク信号です。周辺回路部に構成されるスタックポインタのページ部等をアクセスする際に本端子にLOWレベルが入力され、以下の割り込みがマスクされます。 $\overline{\text{NMI}}$、$\overline{\text{IRQ3}}$、$\overline{\text{IRQ2}}$、$\overline{\text{IRQ1}}$
	割り込みアクノリッジ	$\overline{\text{IACK}}$	出力	割り込み要求を受け付けたことを示す応答信号です。割り込み受け付け時にLOWレベルとなります。周辺回路は本信号を受けてベクタアドレスをホールドします。なお、本信号はリセットおよびゼロ除算による例外処理の実行時においてもLOWレベルとなります。
	インタラプトフラグ	I0F I1F	出力	システムコンディションフラグ(SC)内のインタラプトフラグ(I0、I1)をモニタ出力しています。
ステータス信号	第1オペコードフェッチ信号	SYNC	出力	CPUが第1オペコードをフェッチする際にアクティブとなる信号です。第1オペコードフェッチのバスサイクル期間中、HIGHレベルとなります。 また、本信号の立ち上がりエッジにて割り込みがサンプリングされます。
	停止信号	$\overline{\text{STOP}}$	出力	CPUが以下の状態に入ったとき、LOWレベルとなります。 ・HALT命令によるCPUの停止 ・SLP命令によるCPUの停止 ・$\overline{\text{BREQ}}$へのLOWレベル入力によるバスの解放
	データバスステータス	DBS0 DBS1	出力	データバスの状態を以下のとおり示す2ビットのステータス信号です。 DSB1 DSB0 ステート 0 0 ハイインピーダンス 0 1 割り込みベクタアドレスリード 1 0 メモリライト 1 1 メモリリード

注! 入出力信号はS1C88 Familyの各デバイスによって周辺回路の信号の追加など、表の内容と異なる場合があります。それらの信号については各デバイスのマニュアルを参照してください。

2 アーキテクチャ

S1C88は最大16Mバイトのアドレス空間を持ち、大規模なアプリケーションにも対応できます。ここでは、このアドレス空間とメモリ管理、レジスタの構成等について説明します。

2.1 アドレス空間とCPUモデル

S1C88にはアドレス空間の大きさと乗除算命令のありなしにより、**MODEL0**～**MODEL3**の4種類のCPUモデルが設定されています。各モデルの違いは表2.1.1のとおりで、マイコンの用途、アプリケーションの規模によって選択できるようになっています。

表2.1.1 CPUモデル

CPUモデル	アドレス空間	乗除算命令
MODEL0	64Kバイト	なし
MODEL1	64Kバイト	あり
MODEL2	16Mバイト	なし
MODEL3	16Mバイト	あり

MODEL2、MODEL3については、さらにプログラミング領域を最大64Kバイトとする**ミニマムモード**と最大8Mバイトとする**マキシマムモード**のいずれかがコアCPUの**MODE**端子の設定によって選択できます。

表2.1.2 動作モードの設定(MODEL2、MODEL3)

MODE	動作モード	プログラミング領域
0	ミニマムモード	最大64Kバイト
1	マキシマムモード	最大8Mバイト

各CPUモデルに対するメモリマップの概念を図2.1.1に示します。

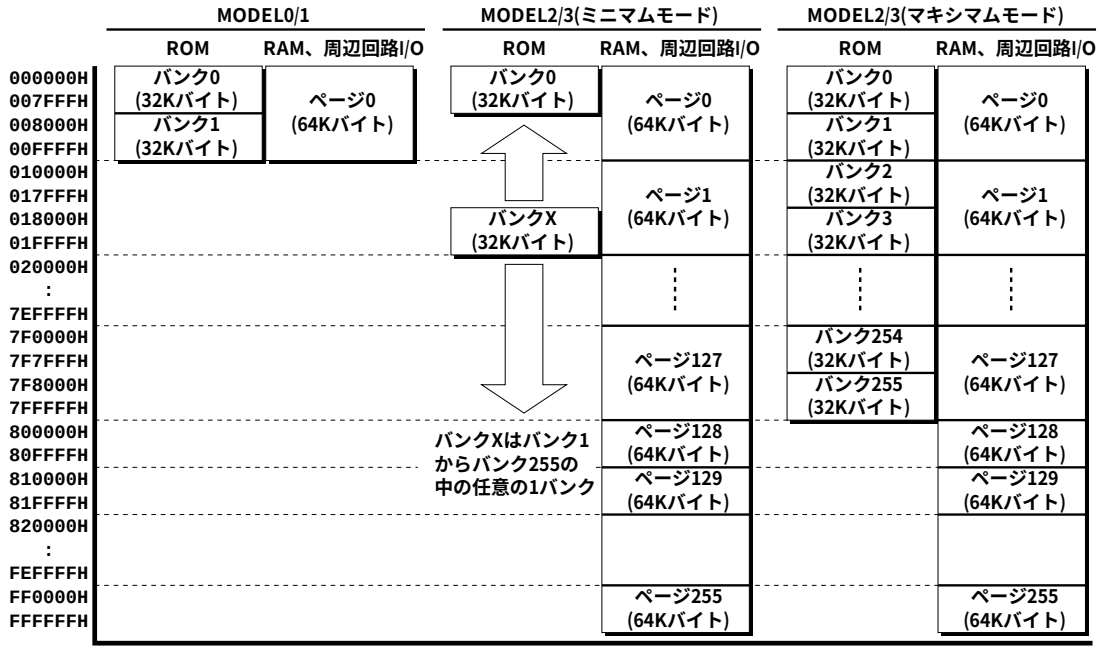


図2.1.1 メモリマップ

プログラムメモリは32Kバイトごとの**バンク**、データメモリは64Kバイトごとの**ページ**に分けて管理されます。

☞ **"2.3 プログラムメモリ"、"2.4 データメモリ"**

注! S1C88 Familyの各デバイスによってメモリの構成が異なります。各デバイスのマニュアルを参照してください。

2.2　演算部とレジスタ

2.2.1 ALU

ALU(算術論理演算ユニット)は2種類のテンポラリレジスタTEMP 0とTEMP 1にストアされた8ビットまたは16ビットデータ間の演算を行います。
ALUの機能は表2.2.1.1のとおりです。

演算結果は16ビットのテンポラリレジスタTEMP 2に一旦ストアされた後、演算命令にしたがったレジスタ/メモリにストアされるか、アドレスデータとして使用されます。
また、演算結果によりZ(ゼロ)フラグ、C(キャリー)フラグ、V(オーバーフロー)フラグ、N(ネガティブ)フラグがセット/リセットされます。
☞ **"2.2.3 フラグ"**

表2.2.1.1　ALUの演算機能

演算機能	演算命令	16ビット演算
加算	ADD, ADC	○
減算	SUB, SBC	○
論理積	AND	
論理和	OR	
排他的論理和	XOR	
比較	CP	○
ビットテスト	BIT	
1加算/減算	INC, DEC	○
乗算	MLT	
除算	DIV	
補数	CPL, NEG	
ローテート	RL, RLC, RR, RRC	
シフト	SLA, SLL, SRA, SRL	
パック/アンパック	PACK, UPCK	
符号拡張	SEP	

2.2.2 レジスタ構成

S1C88のレジスタ構成を図2.2.2.1に示します。

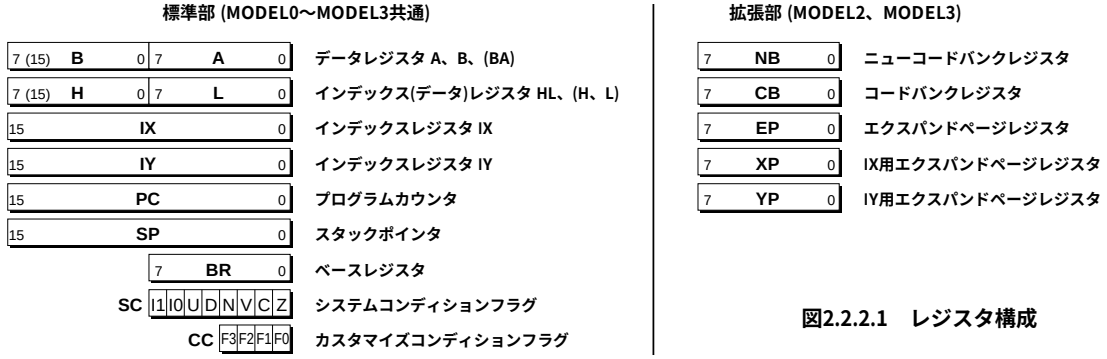


図2.2.2.1　レジスタ構成

■Aレジスタ、Bレジスタ

Aレジスタ、Bレジスタはそれぞれ8ビットのデータレジスタで、他のレジスタやデータメモリとのデータ転送や演算、即値データの転送、演算が行えます。8ビット転送/演算においてはそれぞれ単独に、16ビット転送/演算においてはBレジスタを上位8ビットとするBAペアで使用します。

■HLレジスタ

HLレジスタは16ビットのインデックスレジスタで、データメモリの間接アドレッシングに使用します(ページ内のアドレスを指定)。他のレジスタやデータメモリとの16ビットデータ転送や演算が行えます。また、Hレジスタ、Lレジスタの8ビットずつに分けてデータレジスタとしても使用できます。この場合のLレジスタはIXレジスタ、IYレジスタによる間接アドレッシングの際のディスプレースメントとして使用することもできます。
☞ **"2.4 データメモリ"**
☞ **"4.1 アドレッシングモード"**

■IXレジスタ、IYレジスタ

IXレジスタ、IYレジスタはそれぞれ16ビットのインデックスレジスタで、データメモリの間接アドレッシングに使用します(ページ内のアドレスを指定)。他のレジスタやデータメモリとの16ビットデータ転送や演算が行えます。
☞ **"2.4 データメモリ"**
☞ **"4.1 アドレッシングモード"**

■PC(プログラムカウンタ)

PCはプログラムメモリのアドレッシングを行う16ビットのカウンタレジスタで、次に実行する命令のアドレスを示します。
☞ **"2.3 プログラムメモリ"**

■SP(スタックポインタ)

SPは16ビットのカウンタレジスタでスタックアドレス(スタックページ内のアドレス)を示します。他のレジスタやデータメモリとの16ビットデータ転送や演算が行えます。
☞ **"2.4.3 スタック"**

■BR(ベースレジスタ)

BRは8ビットのインデックスレジスタで、8ビット絶対アドレッシング(アドレスの下位8ビットを指定)の際にページ内の上位8ビットのアドレス指定に使用されます。
☞ "4.1 アドレッシングモード"

■SC(システムコンディションフラグ)

SCは8ビットのフラグで、演算結果を示すZ、C、V、Nフラグ、演算モードを設定するD、Uフラグ、割り込み優先レベルを設定するI0、I1フラグで構成されます。
☞ "2.2.3 フラグ"

■CC(カスタマイズコンディションフラグ)

CCは周辺回路によって設定された各種の状態を示す4ビットのフラグで、周辺回路によってセット/リセットされ、分岐命令の条件の1つとして使用されます。
☞ "2.2.3 フラグ"

■NB(ニューコードバンクレジスタ)

NBレジスタはプログラムメモリのバンクを指定する8ビットのレジスタです。NBレジスタはCPUモデルのMODEL2およびMODEL3に設定されています。
☞ "2.3 プログラムメモリ"

■CB(コードバンクレジスタ)

CBレジスタはプログラムメモリの現在選択されているバンクを示す8ビットのレジスタです。NBレジスタにデータを設定した場合、そのデータがCBレジスタにロードされ、新たなバンクが選択されます。CBレジスタはCPUモデルのMODEL2およびMODEL3に設定されています。
☞ "2.3 プログラムメモリ"

■EP、XP、YP(エクスパンドページレジスタ)

これらのレジスタはデータメモリのページを指定する8ビットのレジスタです。
EPレジスタはHLレジスタによる間接アドレッシング、あるいは即値データによる絶対アドレッシングの際に使用されます。
XPレジスタはIXレジスタによる間接アドレッシング、YPレジスタはIYレジスタによる間接アドレッシングの際に使用されます。
これらのレジスタはCPUモデルのMODEL2およびMODEL3に設定されています。
☞ "2.4.2 ページレジスタEP、XP、YP"
☞ "4.1 アドレッシングモード"

2.2.3 フラグ

S1C88にはCPU内部の演算結果の状態などを表す**システムコンディションフラグ(SC)**と周辺回路の状態を表す**カスタマイズコンディションフラグ(CC)**が設定されています。

■システムコンディションフラグ (SC)

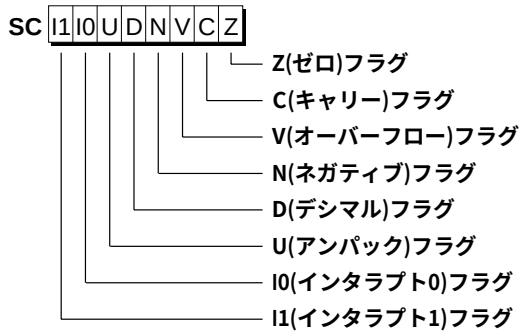


図2.2.3.1 システムコンディションフラグ (SC)

システムコンディションフラグは図2.2.3.1に示すとおり、8ビットのフラグで構成されたレジスタSCとなっています。
システムコンディションフラグの**Z(ゼロ)**、**C(キャリー)**、**V(オーバーフロー)**、**N(ネガティブ)フラグ**は演算の結果によりセット/リセットされ、**I0**、**I1(インタラプト)フラグ**は割り込みによりセット/リセットされます。また、これらのフラグは以下の命令によっても操作できます。

```
AND   SC,#nn    (任意のフラグをリセット)
OR    SC,#nn    (任意のフラグをセット)
XOR   SC,#nn    (任意のフラグを反転)
LD    SC,#nn    (フラグのライト)
LD    SC,A      (フラグのライト)
POP   SC        (フラグの復帰)
RETE            (フラグの復帰)
```

Z、C、V、NフラグはJRS命令やCARS命令などの条件ジャンプ/コール命令実行の際の条件判定に使用されます。
☞ "4.4 命令の詳細説明"

以下、各フラグについて説明します。

(1) Z(ゼロ)フラグ

Zフラグは演算命令の**実行結果が'0'**になった場合に'1'にセット、結果が'0'以外の場合に'0'にリセットされます。

(2) C(キャリー)フラグ

Cフラグは加算命令の実行により**キャリー**(最上位ビットから桁上げ)が発生した場合、または減算命令/比較命令の実行により**ボロー**(最上位ビットに借り)が発生した場合に'1'にセット、それ以外の場合に'0'にリセットされます。ただし、1加減算命令(INC、DEC命令)の実行ではCフラグは変化しません。
また、ローテート/シフト命令の実行によってもCフラグは変化します。
乗除算命令(MLT、DIV命令)を実行した場合は'0'にリセットされます。

(3) V(オーバーフロー)フラグ

Vフラグは演算を行った結果が8ビットまたは16ビットで**補数表現可能な範囲を越えた**場合に'1'にセット、範囲内の場合に'0'にリセットされます。補数表現の範囲は8ビットが**-128～127**、16ビットが**-32768～32767**になります。
ただし、演算命令の中でも論理演算命令(デスティネーションがSCの場合を除くAND、OR、XOR命令)、ビット操作命令(BIT命令)、1加減算命令(INC、DEC命令)の実行ではVフラグは変化しません。乗算命令(MLT命令)を実行した場合は'0'にリセットされます。除算命令(DIV命令)を実行した場合は、商が8ビットを越えるときに'1'にセットされます。
Cフラグが絶対値演算のオーバー(アンダー)フローを示すのに対し、Vフラグは補数演算のオーバーフローを示します。オーバーフローが起こりうる補数演算を行う場合はVフラグをチェックして、'1'のときには演算結果を補正する必要があります。
☞ **"2.2.4 補数演算とオーバーフロー"**

(4) N(ネガティブ)フラグ

Nフラグは演算を行った結果が**マイナス**(最上位ビットが'1')になった場合に'1'にセット、プラス(最上位ビットが'0')の場合に'0'にリセットされます。ただし、1加減算命令(INC、DEC命令)の実行ではNフラグは変化しません。

(5) D(デシマル)フラグ

Dフラグは8ビット加減算命令の実行の際に**10進演算**(演算結果が10進補正される)を行うように設定するビットで、'1'を設定することにより10進演算を行い、'0'で16進演算を行います。
☞ **"2.2.5 10進演算とアンパック演算"**

(6) U(アンパック)フラグ

Uフラグは8ビット加減算の実行の際に**アンパック演算**(上位4ビットを'0'として演算)を行うように設定するビットで、'1'を設定することによりアンパック演算を行い、'0'で通常の8ビット演算を行います。
☞ **"2.2.5 10進演算とアンパック演算"**

(7) I0、I1(インタラプト)フラグ

I0およびI1フラグは**割り込み優先レベル**を設定するビットで、この2ビットで設定したレベルより高いレベルの割り込みのみCPUは受け付けます。
また、割り込みが発生した際にも、そのレベル以下の割り込みをマスクするよう、新たな値が自動設定されます。
☞ **"3.5.3 割り込み"**

命令セット一覧表等では命令の実行により変化するフラグを"↕"を付けて示しています。Dフラグ、Uフラグは"★"を付けて、その命令が10進演算、アンパック演算可能であることを示しています。

■カスタマイズコンディションフラグ (CC)

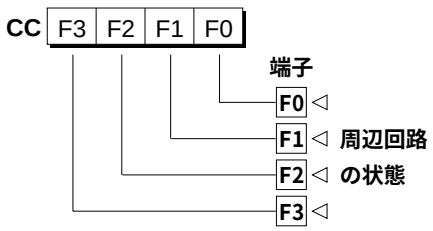


図2.2.3.2　カスタマイズコンディションフラグ (CC)

カスタマイズコンディションフラグは図2.2.3.2に示すとおり、4ビットのフラグで構成されたレジスタCCとなっています。
CCの各フラグは**F0～F3**の名称となっており、周辺回路からS1C88のF0～F3端子に入力される信号によって変化します。ここには**周辺回路の状態を示す**信号が入力されますので、各フラグに反映された周辺回路の状態によってプログラムを分岐させることができます。
S1C88はCCフラグによって**特定用途向けマイコン**が容易に実現できるように配慮されています。
CCフラグはJRS命令やCARS命令の条件ジャンプ/コール命令実行の際の条件判定に使用されます。
☞ **"4.4 命令の詳細説明"**

2.2.4 補数演算とオーバーフロー

S1C88内部ではマイナスのデータを扱うために**補数表現**が用いられます。以下に補数表現と補数を用いた演算について説明します。

■補数

マイクロコンピュータでマイナス(負)の数を扱う場合、一般的には補数表現が用いられます。補数には**2の補数**と**1の補数**の2種類の表現方法があり、通常単に補数といった場合、2の補数を指します。S1C88でも**マイナスの数は2の補数**で表しています。

任意の数Nの補数は以下の式で表され、2の補数表現が可能な範囲は8ビットの場合が**-128～127**、16ビットの場合が**-32768～32767**になります。1の補数表現が可能な範囲は8ビットの場合が-127～127、16ビットの場合が-32767～32767になります。

補数表現した場合、マイナスの数の最上位ビットは必ず'1'になります。この最上位ビットの内容がN(ネガティブ)フラグに反映されます。

なお、8ビットのデータを補数に変換するための命令として"**CPL**"**命令**(1の補数に変換)と"**NEG**"**命令**(2の補数に変換)が、8ビットの補数を16ビットに拡張するための命令として"**SEP**"**命令**が用意されています。

例: NEG命令とSEP命令

命令	B reg.	A reg.	Nフラグ
LD A,#127	0000 0000	0111 1111	0
NEG A	0000 0000	1000 0001	1
SEP	1111 1111	1000 0001	1

---------- 2の補数 ----------

8ビット **$-N = 2^8 - N = 256 - N$**

127 = 0111 1111b
126 = 0111 1110b
:
2 = 0000 0010b
1 = 0000 0001b
0 = 0000 0000b
-1 = 1111 1111b (= 1 0000 0000b - 0000 0001b)
-2 = 1111 1110b (= 1 0000 0000b - 0000 0010b)
:
-127 = 1000 0001b (= 1 0000 0000b - 0111 1111b)
-128 = 1000 0000b (= 1 0000 0000b - 1000 0000b)

16ビット **$-N = 2^{16} - N = 65536 - N$**

32767 = 0111 1111 1111 1111b
32766 = 0111 1111 1111 1110b
:
2 = 0000 0000 0000 0010b
1 = 0000 0000 0000 0001b
0 = 0000 0000 0000 0000b
-1 = 1111 1111 1111 1111b
(= 1 0000 0000 0000 0000b - 0000 0000 0000 0001b)
-2 = 1111 1111 1111 1110b
(= 1 0000 0000 0000 0000b - 0000 0000 0000 0010b)
:
-32767 = 1000 0000 0000 0001b
(= 1 0000 0000 0000 0000b - 0111 1111 1111 1111b)
-32768 = 1000 0000 0000 0000b
(= 1 0000 0000 0000 0000b - 1000 0000 0000 0000b)

---------- 1の補数 ----------

8ビット **$-N = 2^8 - 1 - N = 255 - N (= \bar{N})$**

127 = 0111 1111b
126 = 0111 1110b
:
2 = 0000 0010b
1 = 0000 0001b
0 = 0000 0000b
-1 = 1111 1110b (= 1111 1111b - 0000 0001b)
-2 = 1111 1101b (= 1111 1111b - 0000 0010b)
:
-126 = 1000 0001b (= 1111 1111b - 0111 1110b)
-127 = 1000 0000b (= 1111 1111b - 0111 1111b)

16ビット **$-N = 2^{16} - 1 - N = 65535 - N (= \bar{N})$**

32767 = 0111 1111 1111 1111b
32766 = 0111 1111 1111 1110b
:
2 = 0000 0000 0000 0010b
1 = 0000 0000 0000 0001b
0 = 0000 0000 0000 0000b
-1 = 1111 1111 1111 1110b
(= 1111 1111 1111 1111b - 0000 0000 0000 0001b)
-2 = 1111 1111 1111 1101b
(= 1111 1111 1111 1111b - 0000 0000 0000 0010b)
:
-32766 = 1000 0000 0000 0001b
(= 1111 1111 1111 1111b - 0111 1111 1111 1110b)
-32767 = 1000 0000 0000 0000b
(= 1111 1111 1111 1111b - 0111 1111 1111 1111b)

■補数演算とV(オーバーフロー)フラグ

アドレス計算などの絶対値による演算の場合、8ビットでは0～255、16ビットでは0～65535の範囲で正しい演算結果が得られます。演算によりオーバーフローあるいはアンダーフローが発生し範囲を外れた場合は、C(キャリー)フラグが'1'にセットされます。
演算数、被演算数が補数になっている場合の正しい演算結果の範囲は、8ビットが-128～127、16ビットが-32768～32767となり、Cフラグのみでは演算結果が正しいかどうか判断できません。この判断を行うために**V(オーバーフロー)フラグ**が設定されており、補数表現の範囲を越えた場合にVフラグが'1'にセットされます。

なお、ALUは絶対値演算、補数演算を区別してはいませんので、Cフラグ、Vフラグのセット/リセットは単に演算結果が上記の範囲内にあるかどうかで行われます。したがって、絶対値演算においてもVフラグが'1'にセットされる場合があります。この場合のVフラグは意味を持ちませんので、プログラムでVフラグを確認する必要はありません。Vフラグでオーバーフローが判断できるのは補数演算の場合に限られますので、アプリケーションで扱うデータが符号付きかどうかを判断してください。

以下に8ビットの演算例とその演算結果によるVフラグ、Cフラグの変化を示します。

例: "ADD A,B"命令による加算例

A reg.	B reg.	結果(A reg.)	Vフラグ	Cフラグ
0101 1010	1010 0101	1111 1111	0	0
0101 1011	1010 0101	0000 0000	0	1
0101 1011	0010 0101	1000 0000	1	0

"SUB A,B"命令による減算例

A reg.	B reg.	結果(A reg.)	Vフラグ	Cフラグ
0101 1010	0101 1010	0000 0000	0	0
0101 1010	0101 1011	1111 1111	0	1
0101 1010	1101 1010	1000 0000	1	1

2.2.5 10進演算とアンパック演算

S1C88では以下の8ビット演算命令実行時、通常の16進演算の他に**10進演算**、**アンパック演算**、およびその組み合わせによって演算を行うように設定することができます。この設定は**D(デシマル)フラグ**、**U(アンパック)フラグ**によって行います。

10進演算とアンパック演算が可能な演算命令

ADD	ADC	SUB	SBC	NEG

いずれも8ビット演算命令で、命令セット一覧表等ではDフラグ、Uフラグの部分に"★"を付けて、10進演算、アンパック演算が可能であることを示しています。

■10進演算

Dフラグを'1'にセットして8ビット算術演算(ADD、ADC、SUB、SBC、NEG)命令を実行すると、2桁の10進演算が行われます。演算結果はBCD(2進化10進数)コードで得られます。
10進演算を行う場合は演算命令の実行前に"OR SC,#00010000B"命令等でDフラグを'1'に設定します。また、演算数および被演算数はBCDコードが対象となりますので、BCDコードを設定してから演算を実行してください。BCDコードになっていない場合は、正しい結果が得られません。

10進演算時のSCフラグ

10進演算実行後、演算結果によってSCのN/V/C/Zフラグは下記のように設定されます。

N: 常時　リセット(0)
V: 常時　リセット(0)
C: 2桁の10進数に対して
　桁上げまたは桁借りがあったとき セット(1)
　桁上げまたは桁借りがないとき　リセット(0)
Z: 演算結果=0のとき　セット(1)
　演算結果≠0のとき　リセット(0)

例: "ADD A,B"命令による10進演算例

設定値		結果	SC			
A reg.	B reg.	A reg.(和)	N	V	C	Z
55	28	83	0	0	0	0
74	98	72	0	0	1	0

"SUB A,B"命令による10進演算例

設定値		結果	SC			
A reg.	B reg.	A reg.(差)	N	V	C	Z
55	55	00	0	0	0	1
55	28	27	0	0	0	0
74	98	76	0	0	1	0

■アンパック演算

Uフラグを'1'にセットして8ビット算術演算(ADD、ADC、SUB、SBC、NEG)命令を実行すると、下記に示すアンパック形式での演算を行うことができます。
アンパック演算は、上位4ビットのデータを無視し、下位4ビットのみを対象に演算を行います。実行後は下位4ビットのみの演算結果が出力され、上位4ビットは'0'が出力されます。
アンパック演算は、メモリ1アドレスに対して1桁のデータを格納するため、演算項の桁合わせが容易に行えます。(この場合の桁合わせはメモリアドレスのポインティングのみとなります。)

<ADD命令の例>

アンパック演算時のSCフラグ

アンパック演算は下位4ビットデータのみを対象にしますので、SCフラグも下位4ビットの演算結果によって変化します。
アンパック演算実行後、演算結果によってSCのN/V/C/Zフラグは下記のように設定されます。

N: 2^3ビットが'1'のとき　セット(1)
　2^3ビットが'0'のとき　リセット(0)
V: 4ビットの補数表現(-8～7)
　を越えたとき　セット(1)
　以内のとき　リセット(0)
C: 2^3ビットからの桁上げ、2^3ビットへの桁借りが
　あったとき　セット(1)
　ないとき　リセット(0)
Z: 下位4ビット=0のとき　セット(1)
　下位4ビット≠0のとき　リセット(0)

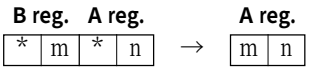
例: "ADD A,B"命令によるアンパック演算例

設定値		結果	SC			
A reg.	B reg.	A reg.(和)	N	V	C	Z
20H	D0H	00H	0	0	0	1
2EH	53H	01H	0	0	1	0
C7H	52H	09H	1	1	0	0

アンパック演算補助命令

アンパック形式とパック形式(通常の8ビットデータ形式)を相互に変換する"PACK"、"UPCK"命令が用意されており、容易にフォーマット変換ができます。

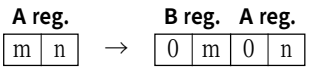
PACK命令: BAレジスタのアンパック形式データをパック形式に変換し、Aレジスタに格納します。



例: PACKの実行例

設定値		結果	SC			
BA reg.		A reg.	N	V	C	Z
38C4H		84H	不変			

UPCK命令: Aレジスタの8ビットデータをアンパック形式に変換し、BAレジスタに格納します。



例: UPCKの実行例

設定値		結果	SC			
A reg.		BA reg.	N	V	C	Z
84H		0804H	不変			

2.2.6 乗除算

S1C88のMODEL1およびMODEL3は乗除算機能を持っています。MODEL0およびMODEL2ではこの機能および以下に説明する乗除算命令は使用できません。

■乗算

乗算はMLT命令によって行います。
MLT命令を実行すると、Lレジスタ× Aレジスタの演算が行われ、積がHLレジスタに格納されます。
この演算結果により、SCのN/V/C/Zフラグは次のように設定されます。

N: HLレジスタ(積)の最上位ビットが
　'1'のとき　　セット(1)
　'0'のとき　　リセット(0)
V: 常時　　リセット(0)
C: 常時　　リセット(0)
Z: HLレジスタ(積)が
　0000Hのとき　　セット(1)
　0000H以外のとき　リセット(0)

以下にMLT命令の実行例を示します。

設定値		結果	SC			
L reg.	A reg.	HL reg.(積)	N	V	C	Z
00H	64H	0000H	0	0	0	1
64H	58H	2260H	0	0	0	0
C8H	58H	44C0H	0	0	0	0
A5H	93H	5EBFH	0	0	0	0
C8H	A5H	80E8H	1	0	0	0

乗算は上記の設定値を符号なし8ビットデータとして扱い、符号なしの演算を行いますので、演算結果により設定されるNフラグは符号を表しません。
したがって、上記例のC8H×A5Hのように負の数どうしの乗算を行った場合でも、Nフラグが'0'に設定されないことがあります。

■除算

除算はDIV命令によって行います。
DIV命令を実行すると、HLレジスタ÷ Aレジスタの演算が行われ、商がLレジスタに、剰余がHレジスタに格納されます。
商が8ビットを越える場合はV(オーバーフロー)フラグがセットされ、HLレジスタの内容は被除数が保持されます。
Aレジスタに'0'を設定してDIV命令を実行した場合はゼロ除算例外処理が発生します。

この演算結果により、SCのN/V/C/Zフラグは次のように設定されます。

N: Lレジスタ(商)の最上位ビットが
　'1'のとき　　セット(1)
　'0'のとき　　リセット(0)
V: 商が8ビット以内に
　納まらないとき　セット(1)
　納まるとき　　リセット(0)
C: 常時　　リセット(0)
Z: Lレジスタ(商)が
　00Hのとき　　セット(1)
　00H以外のとき　リセット(0)

SCの動作例

設定値		SC				備考
HL reg.	A reg.	N	V	C	Z	
nz	nz	↕	↕	0	↕	
0000H	nz	0	0	0	1	
nz	00H	1	1	0	0	ゼロ除算
0000H	00H	1	1	0	0	例外処理発生

nzは'0'以外の8ビットまたは16ビットデータを表します。

除算実行例

以下にDIV命令の実行例を示します。

設定値		結果			SC			
HL reg.	A reg.	L(商)	H(剰余)	A reg.	N	V	C	Z
1A16H	64H	42H	4EH	64H	0	0	0	0
332CH	64H	83H	00H	64H	1	0	0	0
0000H	58H	00H	00H	58H	0	0	0	1
0301H	02H	01H	03H	02H	1	1	0	0

上記例の0301H÷02Hは商が8ビットを越えるため、HLレジスタの値が保持され結果が出力されません。このような場合は、下記のように被除数を上位8ビット、下位8ビットに分離して除算を行います。

<0301H÷02Hの実行例>

```
LD   HL,#0003H   ;被除数=上位8ビット
LD   A,#02H      ;除数
DIV              ;L=商, H=剰余
LD   [hhll],L    ;商(上位8ビット)をメモリに格納
LD   L,#01H      ;被除数=Hレジスタ+下位8ビット
DIV              ;
```

設定値		結果			SC			
HL reg.	A reg.	L(商)	H(剰余)	A reg.	N	V	C	Z
0003H	02H	01H	01H	02H	0	0	0	0
0101H	02H	80H	01H	02H	1	0	0	0

剰余: 01H
商　: 0180H

2.3 プログラムメモリ

2.3.1 プログラムメモリの構成

S1C88のアドレス空間16Mバイトの中で前半の**8Mバイト(アドレス000000H～7FFFFFH)**がプログラミング領域として使用できるようになっています。ただし、MODEL0とMODEL1についてはアドレス空間が最大64Kバイトのため、プログラミング領域もそれ以内に制限されます。

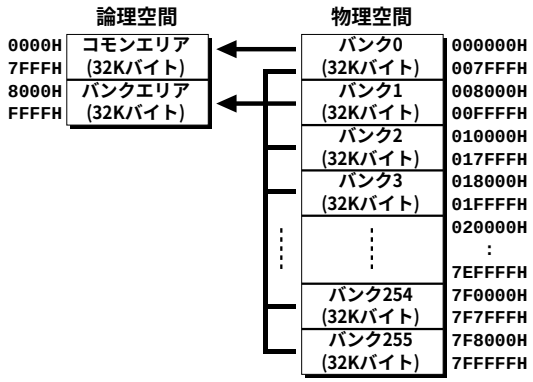


図2.3.1.1 プログラムメモリの構成

8ビットCPUの**論理空間**64Kバイトを越えるメモリを管理するため、S1C88では**バンクマッピング方式**を採用しています。最大8Mバイトのプログラムメモリは**バンク0～バンク255**まで、それぞれ32Kバイトのバンクに分割されます。

64Kバイトの論理空間上には2つのバンクがアドレス0000H～FFFFHとして論理的に連続するように配置され、そのアドレス空間内でプログラムを実行します。論理空間内のアドレッシングは**PC(プログラムカウンタ)**が行います。

論理空間のアドレス0000H～7FFFHには**コモンバンク**として**バンク0(アドレス000000H～007FFFH)**が配置され、この物理アドレスは常時固定となります。アドレス000000H～0000FFHは例外処理(割り込み等)のベクタが割り付けられます。

☞ **"3.5.2 例外処理要因とベクタ"**

コモンエリアが固定のため、各バンクに例外処理ベクタを割り付ける必要はありません。また、汎用的なサブルーチン等もコモンエリアに記述することができます。

後半のアドレス8000H～FFFFHの**バンクエリア**には**CB(コードバンク)レジスタ**で選択されるバンクが配置されます。この部分に配置するバンクはプログラムで任意に選択できます。ただし、MODEL0とMODEL1についてはバンク1に固定、MODEL2とMODEL3のミニマムモードについては任意に選択された1つのバンクに固定されます。

2.3.2 PC(プログラムカウンタ)とCB(コードバンク)レジスタ

PC(プログラムカウンタ)は実行するプログラムのアドレスを保持しています。PCの内容は64Kバイトの**論理空間内のアドレス**で、物理アドレスが連続しない32Kずつのコモンエリアとバンクエリアを論理的に連続したプログラムメモリとしてアドレッシングしています。

コモンエリアは物理アドレスのバンク0に固定されていますが、バンクエリアは256バンクの中から任意の1バンクを選択できるようになっています。(MODEL2とMODEL3)

このバンクエリアに割り当てるバンクアドレス(0～255)を示すレジスタが**CB(コードバンク)**です。

実際にメモリをアクセスするためにアドレスバスに出力される物理アドレスはCPU内部で図2.3.2.1のように生成されます。

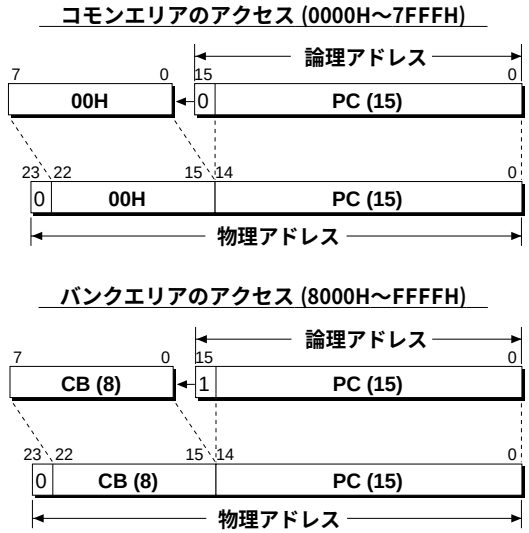


図2.3.2.1 論理アドレスと物理アドレス(MODEL2/3)

図に示すとおり、16ビットのPCの中でアドレスバスに出力されるのは最上位ビットを除く15ビットです。その内容はアドレスバスのA00～A14に出力されます。PCの最上位ビットは'0'でコモンエリア、'1'でバンクエリアを示しており、この内容によってCBをアドレスバスに出力するかどうかを決定します。コモンエリアの場合はアドレスバスのA15～A22に00Hが出力され、バンクエリアの場合はCBの内容8ビットが出力されます。アドレスバスのA23はデータメモリ領域専用で、最大8Mバイトのプログラムメモリアクセス時には常時'0'を出力します。

以上のように**PCの最上位ビットはアドレスバスに出力されません**ので、システム開発時には注意してください。

MODEL0とMODEL1についてはアドレスバスが16ビットのため、PCの内容がそのまま出力されます。

"LD BA,PC"命令、"LD HL,PC"命令のPC値
"LD BA,PC"命令、"LD HL,PC"命令はPC値をBAレジスタおよびHLレジスタに読み込む命令です。読み込むPC値を"PC"、本命令の置かれている先頭アドレスを"PC'"とすると、

PC←PC'+2

で示されるPC値がBAおよびHLレジスタに読み込まれます。たとえば、"LD BA,PC"命令が100H番地に置かれているとすると、BAレジスタには102Hが読み込まれます。

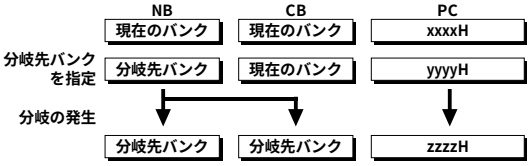


図2.3.3.1　バンクの変更

2.3.3 バンク管理

プログラムの実行は基本的に論理空間に割り当てられたバンク内に限られています。バンクが変更されるのは、プログラムによって分岐命令が実行される時点で他のバンクが指定されている場合です。

注!　プログラムの実行によりPCのカウントがオーバーフローした場合でもCBは更新されません。再びコモンエリアの先頭からの実行となります。

以下にバンク指定の方法と、分岐命令実行時の動作について説明します。
なお、バンク変更にかかわる記載事項はMODEL2およびMODEL3にのみ摘要されます。

■リセット時のバンク設定

イニシャルリセット時にCBは'1'に初期化され、バンクエリアには**バンク1**が割り当てられます。
コモンエリアはバンク0に固定されますので、論理アドレスは物理アドレスと同様になります。
この設定は別のバンクがプログラムで指定され、分岐命令によって実際に分岐が行われるまで変更されません。

■バンクの指定

選択されているバンクを示すCBはプログラムによって直接変更することはできません。
バンク指定用には**NB(ニューコードバンク)レジスタ**が用意されており、分岐命令実行直前に分岐先のバンクアドレス(0～255)を以下の命令によって書き込んでおきます。

```
LD  NB,A   (Aレジスタによる指定)
LD  NB,#bb(8ビット即値データによる指定)
```

NBの内容はその直後の分岐命令の実行により実際に分岐が行われる時点でCBにロードされ、新たなバンクがバンクエリアに選択されます。条件ジャンプなどにおいて条件が合わずに分岐が行われない場合は、CBの内容が逆にNBにロードされます。
したがって、NBに値を設定せずに分岐命令を実行した場合はその時点の論理空間内に分岐するようになっています。

2.3.4 分岐命令

分岐命令はPCとCBを変更してプログラムを任意のアドレスに分岐させます。
分岐命令の種類は動作の違いにより、以下のように分類されます。

表2.3.4.1　分岐命令の種類

種　類	条　件	命　令
PC相対ジャンプ	条件/無条件	JRS、JRL、DJR
間接ジャンプ	無条件	JP
PC相対コール	条件/無条件	CARS、CARL
間接コール	無条件	CALL
リターン	無条件	RET、RETS、RETE
ソフトウェア割り込み	無条件	INT

上記の分岐命令それぞれの中にも無条件に分岐する**無条件分岐命令**とフラグの状態によって分岐する数種類の**条件分岐命令**があります。
条件分岐命令の条件が満たされていない場合は、分岐が行われずにその分岐命令の次の命令を実行します。
☞ **"4.4 命令の詳細説明"**

■PC相対ジャンプ命令(JRS、JRL、DJR)

PC相対ジャンプはPCにオペランドで指定された相対アドレスを加え、そのアドレスに分岐させる命令で、リロケータブルなプログラミングを可能にしています。
相対アドレスは分岐実行時のアドレスから分岐先アドレスまでのディスプレースメントで、1または2バイトで指定します。
指定できる相対アドレスは**"JRS"**命令、**"DJR"**命令が8ビット補数表現の**-128～127**、**"JRL"**命令が16ビット補数表現の**-32768～32767**の範囲です。
なお、この相対アドレスはPCに加算されるため分岐先のアドレスは論理アドレスとなります。直前のNBの設定によって他のバンクにも分岐できますが、分岐先はあくまでも論理空間内で物理的な相対アドレスの指定はできません。
図2.3.4.1にPC相対ジャンプの動作を示します。

"JRS"命令は無条件ジャンプと20種類の条件ジャンプ命令が設定されています。
"JRL"命令は無条件ジャンプと4種類の条件ジャンプ命令が設定されています。

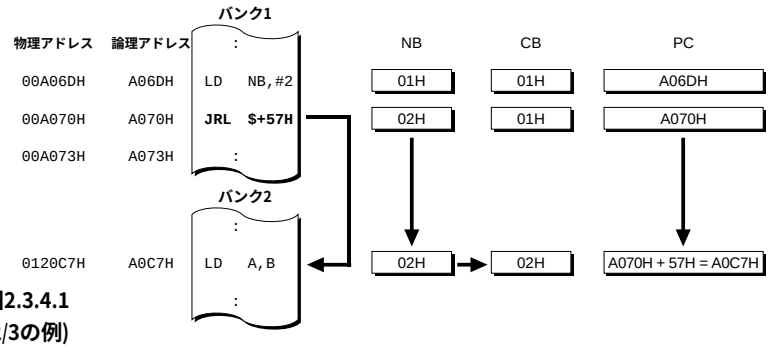


図2.3.4.1
PC相対ジャンプの動作 (MODEL2/3の例)

"**DJR NZ,rr**"命令はBレジスタの'1'減算を行い、その結果が'0'以外の場合に"JRS"の無条件ジャンプ命令を実行します。この命令によって、Bレジスタをカウンタとしたその初期値分の繰り返しルーチンが簡単に記述できます。

例: 50 cycle時間のウェイトルーチン
```
LD   B,#12   ;初期値をBレジスタにセット(2 cycle)
DJR  NZ,$    ;Bレジスタが0になるまで繰り返す(48 cycle)
```

■間接ジャンプ命令(JP)

間接ジャンプは分岐先のアドレスを間接的に指定する命令です。
"**JP [kk]**"命令はアドレス00kk(kk=00H～FFH、ページは'0'固定)のメモリの内容をPCの下位8ビットに、アドレス00kk+1のメモリの内容をPCの上位8ビットにそれぞれロードして、そのアドレスに無条件に分岐します。ここで指定するアドレス00kkは例外処理やソフトウェア割り込みの**ベクタ領域**となっています。
"**JP HL**"命令はHLレジスタの内容をアドレスとして無条件に分岐します。この命令は演算結果等をそのまま分岐先アドレスにすることができるため、ジャンプテーブルの作成などに有効です。

■PC相対コール命令(CARS、CARL)

PC相対コールはPCにオペランドで指定された相対アドレスを加え、そのアドレスからのサブルーチンをコールする命令です。
相対アドレスは分岐実行時のアドレスから分岐先アドレスまでのディスプレースメントで、1または2バイトで指定します。
指定できる相対アドレスは"**CARS**"命令が8ビット補数表現の-128～127、"**CARL**"命令が16ビット補数表現の-32768～32767の範囲です。
なお、この相対アドレスはPCに加算されるため分岐先のアドレスは論理アドレスとなります。直前のNBの設定によって他のバンクにも分岐できますが、分岐先はあくまでも論理空間内で物理的な相対アドレスの指定はできません。

サブルーチンコール実行時にはリターン情報としてPC値(コール命令の次の命令の先頭アドレス)がスタックに退避されます。
MODEL2/3のマキシマムモードではPC値以外にCB値もスタックに退避され、リターン時にはコール元のバンクに戻るようになっています。
MODEL2/3のミニマムモードではMODEL0/1と同様、PC値のみのスタックとなるため、64Kバイト以上のプログラムメモリを使用することはできません。
図2.3.4.2にPC相対コールの動作を示します。

"CARS"命令は無条件コールと20種類の条件コール命令が設定されています。
"CARL"命令は無条件コールと4種類の条件コール命令が設定されています。

■間接コール命令(CALL)

間接コールはサブルーチンのアドレスを間接的に指定するコール命令です。
"**CALL [hhll]**"命令はアドレスhhll(hhll=0000H～FFFFH、ページはEPレジスタによって指定)のメモリの内容をPCの下位8ビットに、アドレスhhll+1のメモリの内容をPCの上位8ビットにそれぞれロードして、そのアドレスのサブルーチンを無条件にコールします。
サブルーチンコール実行時にはPC相対コールと同様、リターン情報としてPC値(コール命令の次の命令の先頭アドレス)とCB値(MODEL2/3マキシマムモードの場合)がスタックに退避されます。

■リターン命令(RET、RETS、RETE)

リターン命令はコール命令によって呼び出されたサブルーチンからコール元のルーチンに戻るための命令です。
リターン命令はサブルーチンコール実行時にスタックに退避されているPC値(コール命令の次の命令の先頭アドレス)を元に戻します。MODEL2/3のマキシマムモードではCB値の復帰も行われ、コールしたバンクに戻ります。
MODEL2/3のミニマムモードではMODEL0/1と同様、PC値のみ復帰します。

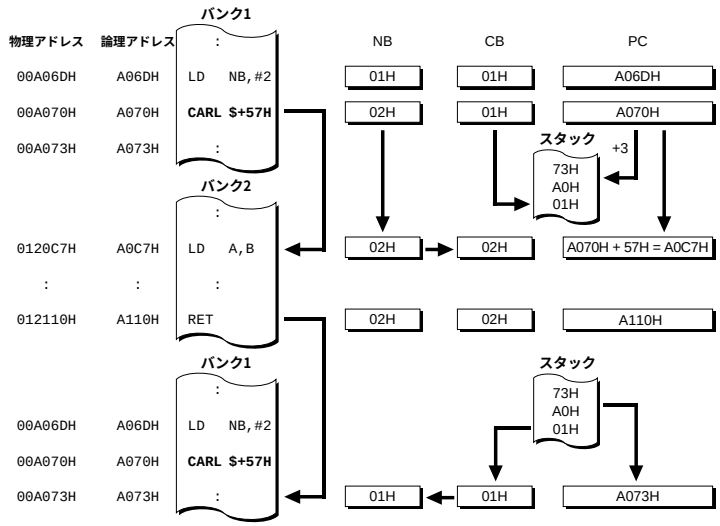


図2.3.4.2　PC相対コールの動作 (MODEL2/3マキシマムモードの例)

ミニマムモードでコール命令の実行時、または実行後にバンクの変更を行っている場合は、リターン命令を実行しても正常なアドレスへ戻ることはできません。

"**RET**"命令はリターン情報そのままに、コール命令の次の命令の先頭アドレスに処理を戻します。
"**RETS**"命令はリターン情報のPC値に'2'を加えてリターンしますので、コール命令の次の**2バイト命令をスキップ**することができます。

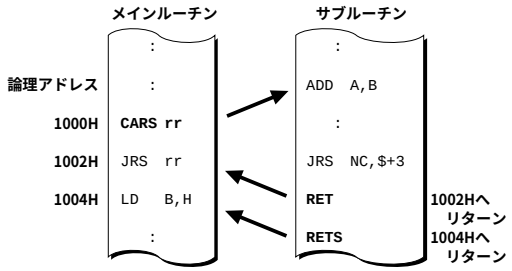


図2.3.4.3　サブルーチンからのリターン

"**RETE**"命令はソフトウェア割り込みルーチンや例外処理ルーチン専用のリターン命令で、リターン情報に**SC**(システムコンディションフラグ)の内容が含まれていることが"RET"命令との違いです。
☞ **"3.5 例外処理状態"**

■ソフトウェア割り込み命令(INT)

ソフトウェア割り込み"**INT　[kk]**"はアドレス00kk(kk=00H～FFH、ページは'0'固定)の**ベクタアドレス**を指定して、その割り込みルーチンを実行させる命令です。間接コール命令の一種ですが、分岐前にSC(システムコンディションフラグ)もスタックに退避されます。したがって、この命令により実行される割り込みルーチンは必ず"**RETE**"命令でリターンする必要があります。
☞ **"3.5 例外処理状態"**

相対分岐命令のPC値

JRS命令、CARS命令、DJR命令
JRS命令、CARS命令、DJR命令は符号付き8ビットPC相対ジャンプ命令で、PC値に相対値rr(-128～127)を符号付きで加算し分岐先を決定します。分岐先アドレスを"PC"、本命令の置かれている先頭アドレスを"PC'"とすると、

PC←PC'+rr+(n-1)

となります。
ここでnは命令のバイト数を示します。
たとえば、"JRS LE,rr"命令が100H番地に置かれているとすると分岐先アドレスは102H+rr番地となります。

JRL命令、CARL命令
JRL命令、CARL命令は符号付き16ビットPC相対ジャンプ命令で、PC値に相対値qqrr(-32768～32767)を符号付きで加算し分岐先を決定します。分岐先アドレスを"PC"、本命令の置かれている先頭アドレスを"PC'"とすると、

PC←PC'+qqrr+2

となります。
たとえば、"JRL　C,qqrr"命令が100H番地に置かれているとすると分岐先アドレスは102H+qqrr番地となります。

2.4 データメモリ

2.4.1 データメモリの構成

S1C88のアドレス空間(最大16Mバイト)の中でプログラムメモリとして使用する領域を除くすべてをデータメモリとして使用することができます。
データメモリ領域には**RAM**や**表示メモリ**、周辺回路を制御する**I/Oメモリ**などが配置されます。
データメモリは64Kバイトを1つの**ページ**として管理します。
図2.4.1.1にデータメモリの構成を示します。

MODEL0/1においてはアドレス空間が64Kバイトのため、ページによる管理を考慮する必要はありません。MODEL2/3は最大で**256ページ**の構成となります。

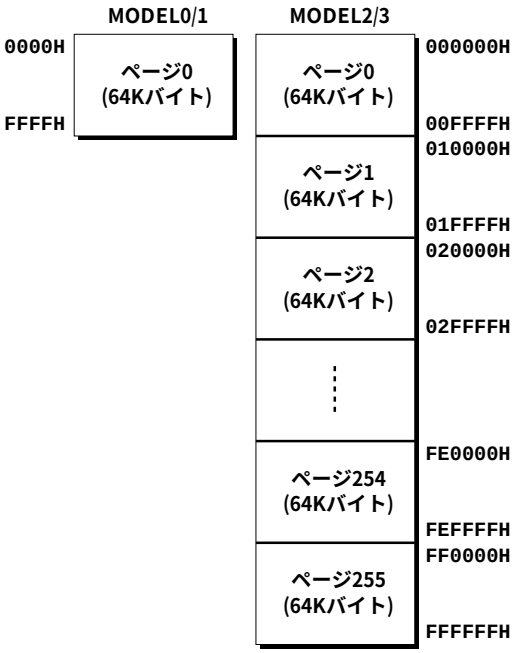


図2.4.1.1 データメモリの構成

2.4.2 ページレジスタEP、XP、YP

データメモリの物理空間は64Kバイトのページに論理的に区切られています。したがって、物理アドレスの上位8ビットがページ部、下位16ビットが論理アドレスとして管理されます。ページ内のアドレス指定はアドレッシングモードにしたがってインデックスレジスタ、即値データ等によって行います。
MODEL2とMODEL3にはページ部の指定用として3組の**ページレジスタEP**、**XP**、**YP**が設定されており、アドレッシングモードの指定にしたがって使い分けられます。図2.4.2.1にアドレッシングモードとページレジスタの対応を示します。
☞ **"4.1 アドレッシングモード"**

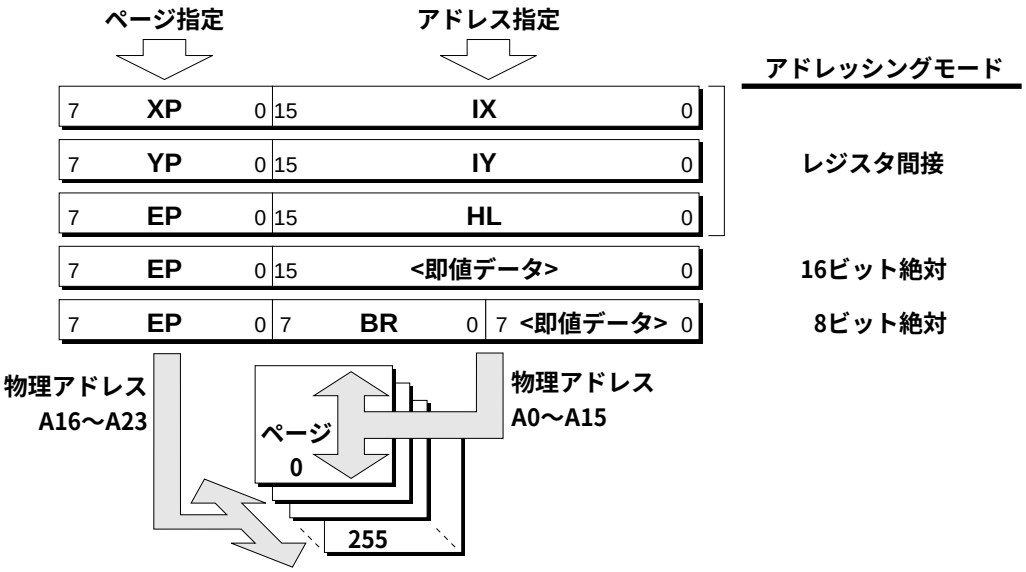


図2.4.2.1 データメモリのアドレッシング

2.4.3 スタック

スタックは**LIFO(Last In, First Out)**形式でアクセスされるメモリで、S1C88ではデータメモリのRAM領域に割り当てられます。
スタックはサブルーチンコールや例外処理(割り込み)などが発生した場合にCPUのレジスタ情報を退避するのに使用されます。またプログラムの任意の場所でレジスタの退避等を行うことができます。
以下、スタックにデータを記憶させることを**"プッシュ"**、記憶したデータを取り出すことを**"ポップ"**と記述します。

■スタックポインタSP

データはスタックの最上位アドレスから順番にプッシュされ、逆にデータを取り出すときには最後にプッシュされたデータから順にポップされます。このプッシュとポップを行うスタックのアドレスを示すレジスタが**SP (スタックポインタ)**です。
SPは1バイトデータのプッシュにより'1'減算(プリデクリメント)され、1バイトデータのポップにより'1'加算(ポストインクリメント)されます。

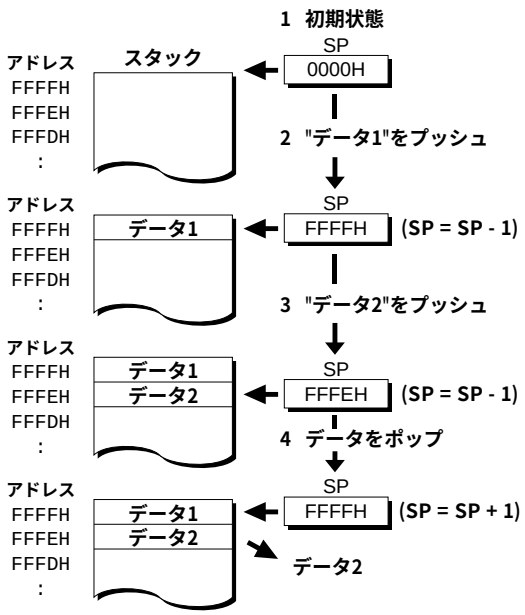


図2.4.3.1　スタックの動作

物理メモリ内でのスタックの位置は周辺回路からコアCPUに入力される**SPP0～SPP7(スタックポインタページ)**信号によってページアドレスが決定され、スタックアクセス時にはSPP0～SPP7の内容がそのままアドレスバスのページ部(A16～A23)に出力されます。そのページ内のアドレスはSPによって指定されます。一般的にはSPの初期値としてアドレス0000Hを設定することにより、そのページの最終アドレスFFFFHから下位のアドレスに向かってデータが順にプッシュされます。

注!　SP(スタックポインタ)はイニシャルリセット時は不定ですので、スタックが使用される前に必ずプログラム("LD SP,"命令)で初期化を行ってください。**

■サブルーチンコールとスタック

コール命令を実行すると、サブルーチンに分岐する前に**リターンアドレス**としてコール命令の次の命令の先頭アドレスとCB(MODEL2/3のマキシマムモードの場合)がスタックにプッシュされます。
スタックにプッシュされたリターン情報はリターン命令の実行によりポップされ、PCとCBに再設定されます。
サブルーチンの中からさらに別のサブルーチンをコールするようなネスティングはスタックが割り当てられたページの使用可能なメモリの範囲内で何レベルまででも可能です。

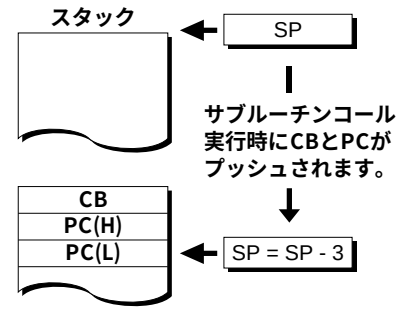


図2.4.3.2　サブルーチンコール実行時のスタック消費

MODEL2/3のマキシマムモードでは、サブルーチンコールによりスタックを**3バイト**(CB、PC)消費します。MODEL0/1およびMODEL2/3のミニマムモードではCBを除くPCの2バイト分を消費します。

■例外処理とスタック

例外処理(割り込み等)発生時もサブルーチンコールと同様にリターン情報がスタックにプッシュされます。このときのリターン情報にはリターンアドレスPCとCB(MODEL2/3のマキシマムモード)のほかに**SC**も含まれます。

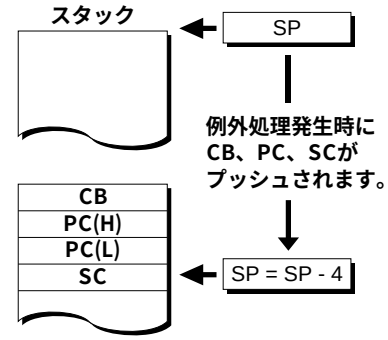


図2.4.3.3　例外処理発生時のスタック消費

MODEL2/3のマキシマムモードでは、例外処理の発生によりスタックを**4バイト**(CB、PC、SC)消費します。MODEL0/1およびMODEL2/3のミニマムモードではCBを除くPCとSCの3バイト分を消費します。

■その他のスタック操作

サブルーチンコールや例外処理(割り込み等)でのリターン情報のプッシュは自動的に行われますが、汎用レジスタのプッシュは行われません。サブルーチンや例外処理ルーチンから汎用レジスタの内容を分岐前の状態にして戻りたい場合は、ルーチンの最初と最後にそれぞれレジスタの内容をプッシュ/ポップする命令を配置する必要があります。
レジスタのプッシュ/ポップは"**PUSH**"命令と"**POP**"命令によって行えます。この命令によってプッシュ/ポップできるレジスタは以下のとおりです。

A、B、L、H、BR、SC 、EP*、IP(XPとYP)*、BA、HL、IX、IY

*はMODEL0/1にはありません。

また、上記のSCを除くすべてのレジスタを1命令でプッシュする"**PUSH ALL**"、"**PUSH ALE**"命令と、1命令でポップする"**POP ALL**"、"**POP ALE**"命令も用意されています。

PUSH ALL =	PUSH BA	POP ALL =	POP BR
	PUSH HL		POP IY
	PUSH IX		POP IX
	PUSH IY		POP HL
	PUSH BR		POP BA
PUSH ALE =	PUSH BA	POP ALE =	POP IP
	PUSH HL		POP EP
	PUSH IX		POP BR
	PUSH IY		POP IY
	PUSH BR		POP IX
	PUSH EP		POP HL
	PUSH IP		POP BA

オペランドの"ALL"はMODEL0/1用で、"ALE"はMODEL2/3で拡張されているレジスタEPとIP(XPとYP)のプッシュ/ポップも行います。

構造化プログラミングにおいてはサブルーチンに渡す**引数**などをスタック領域にストアするといったことがよく行われますが、上記の"PUSH"命令や"POP"命令を使用せずにSPを操作したりスタック領域の直接アクセスが容易に行える命令等も用意されています。

ADD、SUB、CP、INC、DEC、LD

☞ **"4.4 命令の詳細説明"**

注! スタックは汎用RAM領域に割り当てられますので、データ領域とスタック領域が重ならないよう注意してください。

2.4.4 メモリマップドI/O

S1C88 FamilyはS1C88をコアCPUとして、その周辺に入出力ポート等、各種の回路を内蔵します。
S1C88では、それらの周辺回路の制御に**メモリマップドI/O方式**を採用しており、周辺回路の制御ビットやデータをやり取りするためのレジスタがデータメモリ領域に配置されます。
このメモリ領域を汎用RAMと区別するために**I/Oメモリ**と呼びますが、ページの管理やアクセスの方法はデータメモリと共通ですので、通常のメモリアクセスの命令を使って周辺回路を制御することができます。

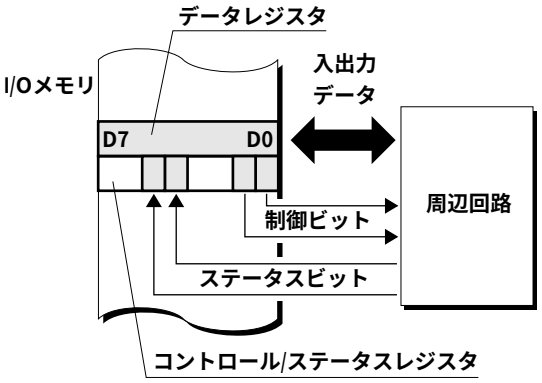


図2.4.4.1 周辺回路とメモリマップドI/O

LCDドライバ内蔵の機種ではデータメモリの一部をセグメントデータ用の**表示メモリ**として使用します。表示メモリ領域の各ビットはセグメントと1対1に対応し、ビットをオン/オフさせることにより対応したセグメントが点灯/消灯します。
このセグメントの制御も通常のメモリアクセスの命令で行うことができます。

注! 機種によってはI/Oメモリの一部や表示メモリがライトオンリーになっている場合があります。その場合、その部分を論理演算および算術演算命令等で直接ビット操作(リード/モディファイ/ライト)を行うことはできません。ビット操作を行う場合は同じ内容のバッファをリード/ライト可能なメモリ上に設定しておき、バッファ上でデータを変更してから本来のメモリに書き込みを行う必要があります。

周辺回路とI/Oメモリおよび表示メモリの詳細についてはS1C88 Familyの各機種のマニュアルを参照してください。

3　CPUの動作と処理状態

CPUはシステムクロックに同期して動作します。また、CPUの処理にはプログラムをシーケンシャルに実行している状態やスタンバイ状態等各種の状態があります。
ここでは割り込みなども含む各種の処理状態と、動作のタイミングについて説明します。

3.1　タイミングジェネレータとバス制御

はじめに、CPU動作の基準となるクロックとバスの制御について説明します。

3.1.1 バスサイクル

S1C88のタイミングジェネレータは入力されたクロックCLKから2相分周した信号PK、PLを作りCLKを**ステート**に分解します。1ステートはCLKの1/2サイクルとなり、命令実行の単位となる**1バスサイクル**は4ステートの構成となります。

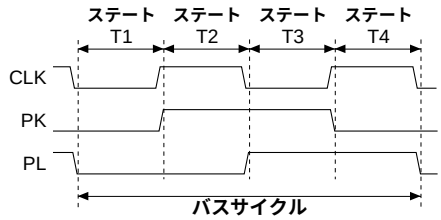


図3.1.1.1　ステートとバスサイクル

命令セット一覧表などでサイクルとして示された数値はこのバスサイクル数を表しています。

S1C88では各バスサイクルでのデータバスの状態を2ビットのステータス信号**DBS0**、**DBS1**で外部に出力しています。周辺回路はこの信号によってバスドライバの方向制御等を容易に行うことができます。
DBS0、DBS1により示されるデータバスのステートは表3.1.1.1のとおりです。

表3.1.1.1　データバスのステート

DBS1	DBS0	ステート
0	0	ハイインピーダンス
0	1	割り込みベクタアドレスリード
1	0	メモリライト
1	1	メモリリード

以下に各バスサイクルのタイミングチャートを示します。

■ハイインピーダンス

内部レジスタアクセス時にはデータバスがハイインピーダンスとなります。また、読み出し信号$\overline{RD}$と書き込み信号WRが共にHIGHレベルに固定され、アドレスバスはバスサイクルの期間中ダミーアドレスを出力します。

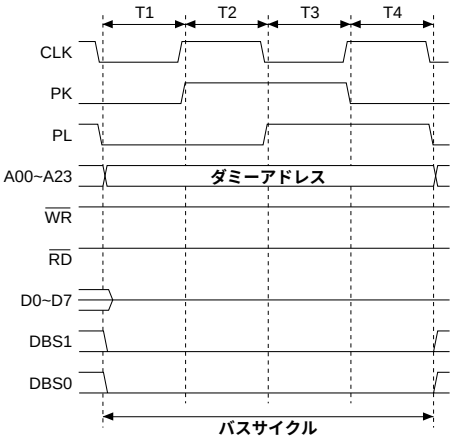


図3.1.1.2　内部レジスタアクセス時のバスサイクル

■割り込みベクタアドレスリード

割り込みベクタアドレスはT2～T3ステートの間にデータバスから読み込まれます。
このリードに際しては読み出し信号$\overline{RD}$が出力されず、割り込みベクタアドレス専用のリード信号**$\overline{RDIV}$**が出力されます。アドレスバスはバスサイクルの期間中ダミーアドレスを出力します。

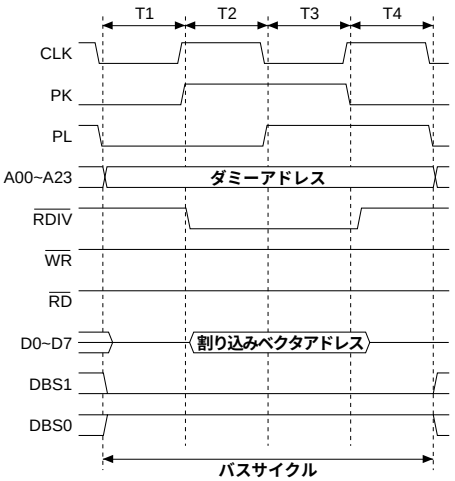


図3.1.1.3 割り込みベクタアドレスリード時のバスサイクル

■メモリライト

メモリライト時はT2～T4ステートの間データバスに書き込みデータが出力され、T3ステートにおいて書き込み信号$\overline{WR}$が出力されます。アドレスバスはバスサイクルの期間中対象アドレスを出力します。

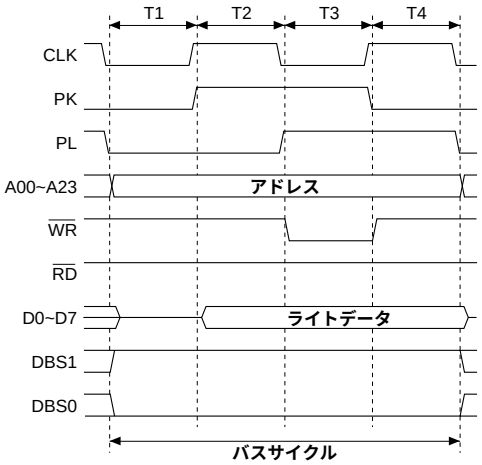


図3.1.1.4　メモリライト時のバスサイクル

■メモリリード

メモリリード時はT2～T3ステートの間読み出し信号$\overline{RD}$が出力され、データバス上のデータを読み込みます。アドレスバスはバスサイクルの期間中対象アドレスを出力します。

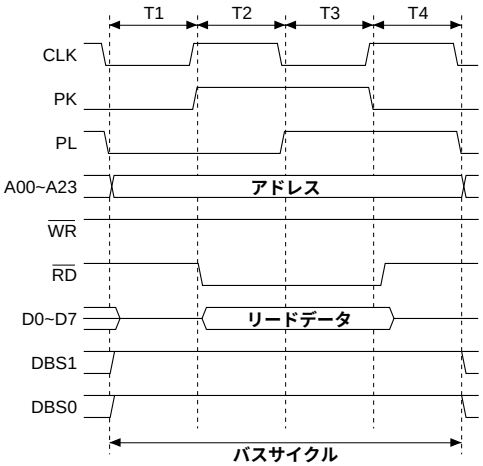


図3.1.1.5　メモリリード時のバスサイクル

3.1.2 ウェイトステート

S1C88はバスラインに接続された**低速デバイス**のアクセスを確実に行うために、**ウェイトステート**を挿入してバスサイクルを伸ばすことができます。S1C88ではウェイトステートとしてアクセスタイム伸長用の**ウェイト**を挿入する機能を持っており、**$\overline{WAIT}$**端子の入力信号によってその制御を行います。

$\overline{WAIT}$信号のサンプリングは、T3ステートのCLKの立ち上がりエッジで行われます。このときの$\overline{WAIT}$信号がLOWレベルの場合にT3ステートとT4ステートの間にウェイトステート**Tw1**、**Tw2**を挿入し、アクセスタイムを伸ばします。$\overline{WAIT}$信号がHIGHレベルの場合、ウェイトステートは挿入されません。ウェイトステートTw1、Tw2は$\overline{WAIT}$信号がLOWレベルの間、連続して挿入されます。ウェイトステートの挿入を解除するためのサンプリングはTw2ステートのCLKの立ち上がりエッジで行われ、$\overline{WAIT}$信号がHIGHレベルに戻っている場合は以降のウェイトステートが挿入されずにT4ステートを開始します。

なお、**ウェイトステートはメモリ空間上に接続されたデバイスをアクセスするときにのみ挿入**され、内部レジスタをアクセスするときには挿入されません。

以下に、割り込みベクタアドレスリード、メモリライト、メモリリードの各サイクルにおけるウェイトステート挿入のタイミングチャートを示します。

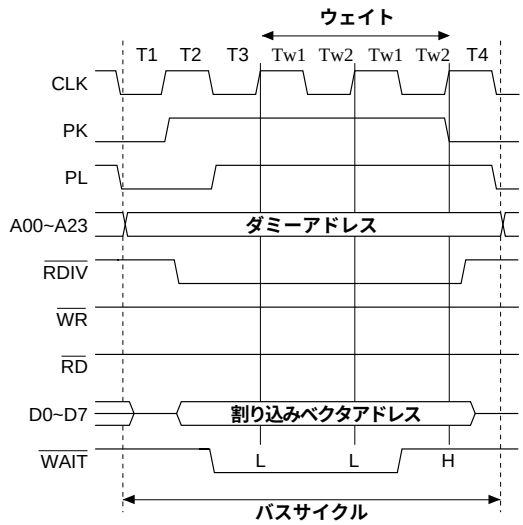


図3.1.2.1　割り込みベクタアドレス
リードサイクルのウェイト挿入

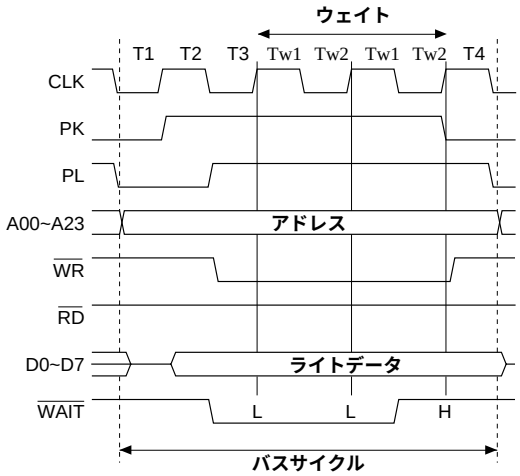


図3.1.2.2　メモリライトサイクルのウェイト挿入

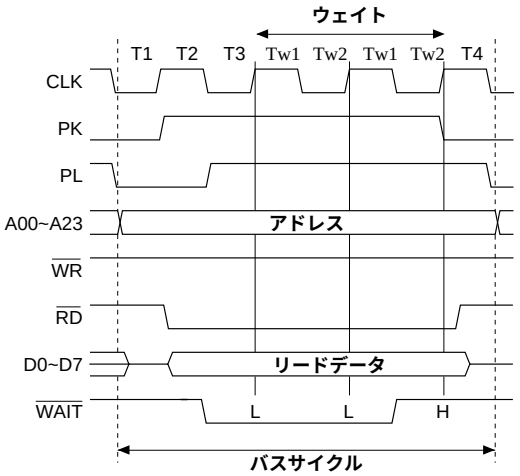


図3.1.2.3　メモリリードサイクルのウェイト挿入

3.2　処理状態の概要

S1C88の動作をその処理の内容により分類すると**リセット状態**、**プログラム実行状態**、**例外処理状態**、**バス権解放状態**、**スタンバイ状態**の5種類になります。
表3.2.1に処理状態の分類を、図3.2.1に状態遷移図を示します。

表3.2.1　処理状態の分類

処理状態		概　　要
リセット状態		CPUがリセットされ、停止している状態
プログラム実行状態		CPUがプログラムを順次実行している状態
例外処理状態		リセット、割り込み等の例外処理要因によって起動される例外処理(ベクタアドレスのフェッチ、PCとSCの退避、PCに対する分岐アドレスの設定)を実行している過渡的な状態
バス権解放状態		外部からのバス権要求信号により外部バスを解放している状態
スタンバイ状態	HALT	CPUを停止し、消費電力を低減させている状態
	SLEEP	CPUおよび周辺回路を停止し、消費電力を低減させている状態

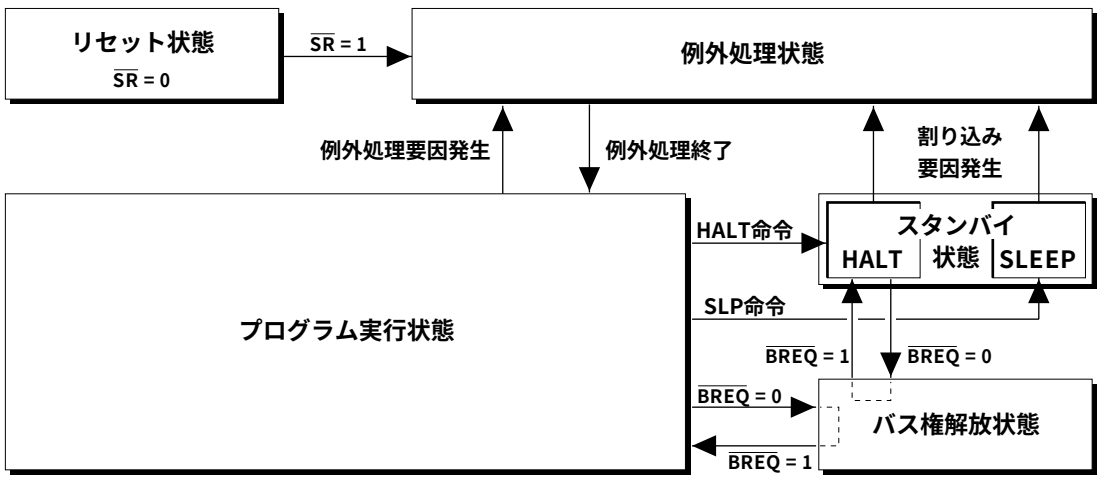


図3.2.1　状態遷移図

3.3 リセット状態

リセット状態とは、S1C88がリセットされ停止している状態を示します。**SR**端子に**LOWレベル**を入力することによりS1C88がリセットされます。リセットはCLKと非同期に行われますので、すべての処理状態から即時リセット状態に移行します。
リセットにより一部の内部レジスタが初期化されます。表3.3.1にレジスタの初期設定値を示します。

図3.3.1にリセット状態とリセット解除後のシーケンスを示します。
SR端子がLOWレベルのリセット期間中はアドレスバス、データバス、リード/ライト信号がハイインピーダンスになります。ただし、アドレスバスとリード/ライト信号はCPU内部でプルアップされるため、HIGHレベルが出力されます。
SR端子がHIGHレベルになるとリセットが解除され、CLKの立ち下がりエッジが2回入力された時点で最初のバスサイクルを開始します。このバスサイクルではアドレスバスにダミーアドレスが出力され、次のバスサイクルで割り込みアクノリッジIACKがイネーブルになります。
これにより、ベクタテーブルに格納されているスタートアドレスを、不定の状態にあるPC(プログラムカウンタ)にロードするリセットの例外処理を開始します。また、このときにNB(ニューコードバンクレジスタ)の初期値01HをCB(コードバンクレジスタ)にロードする処理も同時に行います。このため、**リセット後のバンクエリアにはバンク1**(008000H～00FFFFH)が選択されます。

以上の処理により、**イニシャルリセット後は000000H～000001Hのメモリに格納されたスタートアドレスからプログラムの実行を開始**します。

表3.3.1 内部レジスタの初期設定値

レジスタ名称	記号	ビット長	初期値
データレジスタA	A	8	不定
データレジスタB	B	8	不定
インデックス(データ)レジスタL	L	8	不定
インデックス(データ)レジスタH	H	8	不定
インデックスレジスタIX	IX	16	不定
インデックスレジスタIY	IY	16	不定
プログラムカウンタ	PC	16	不定*
スタックポインタ	SP	16	不定
ベースレジスタ	BR	8	不定
ゼロフラグ	Z	1	0
キャリーフラグ	C	1	0
オーバーフローフラグ	V	1	0
ネガティブフラグ	N	1	0
デシマルフラグ	D	1	0
アンパックフラグ	U	1	0
インタラプトフラグ0	I0	1	1
インタラプトフラグ1	I1	1	1
ニューコードバンクレジスタ	NB	8	01H
コードバンクレジスタ	CB	8	不定*
エクスパンドページレジスタ	EP	8	00H
IX用エクスパンドページレジスタ	XP	8	00H
IY用エクスパンドページレジスタ	YP	8	00H

* リセット例外処理によって、0バンクのメモリの先頭(000000H～000001H)に格納されている値がPCにロードされます。また、このとき同時にNBの初期値01HがCBにロードされます。

レジスタNB、CB、EP、XPおよびYPはMODEL2/3にのみ設定されており、MODEL0/1にはありません。

注! リセットで初期化されないレジスタは、必要に応じてプログラムで初期化を行う必要があります。

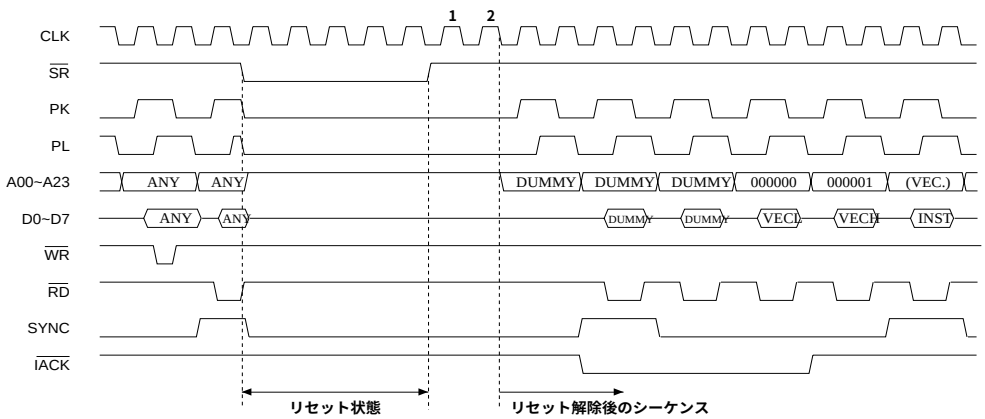


図3.3.1 リセット状態とリセット解除後のシーケンス

3.4 プログラム実行状態

プログラム実行状態とは、S1C88がプログラムを順次実行している状態を示します。

S1C88では、命令の第1オペコードのフェッチが直前の命令の最終サイクルに重複して行われます。

したがって、S1C88の1命令の実行サイクルは第2オペコードのフェッチサイクル、第1オペランドのリードサイクル、第1実行サイクルのいずれか(命令により異なる)から始まり、次の命令の第1オペコードのフェッチサイクルで完了します。1サイクル命令については、次の命令の第1オペコードのフェッチサイクルのみとなります。また、オペランドのリードサイクル後、実行サイクルをはさまずに第1オペコードのフェッチサイクルに移行する場合もあります。

第1オペコードのフェッチサイクルでは、その期間中**SYNC**信号がHIGHレベルになります。

図3.4.1に以下のプログラムと条件による命令の実行サイクルの例を示します。

プログラムリスト

```
001000    44 6E       LD    A,[BR:6EH]
001002    CE 10 34    SUB   A,[IX+34H]
001005    50          LD    L,A
001006    69          LD    [HL],B
```

レジスタおよびメモリの条件

B	=	7FH
H	=	81H
BR	=	83H
IX	=	8000H
EP	=	00H
XP	=	00H
M(008034H)	=	27H
M(00836EH)	=	9BH

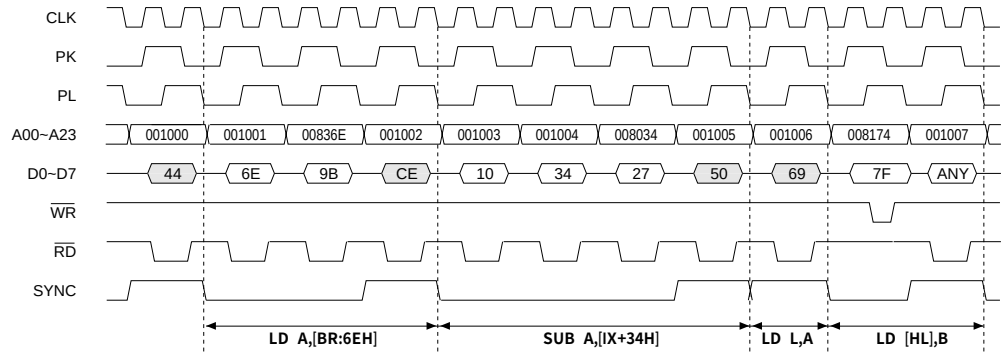


図3.4.1 命令の実行サイクル例

3.5 例外処理状態

例外処理状態とは、割り込みなどの例外処理要因によってS1C88が通常のプログラム実行を中断して処理の流れを変える過渡的な状態を示します。
例外処理のシーケンスを図3.5.1に示します。

例外処理は例外処理要因が発生した時点で実行している命令のサイクル終了時に開始されます。
例外処理フローに示すとおり、中断したルーチンを再開するためのリターン情報をスタックに退避後、例外処理要因に対応したベクタアドレスから**例外処理ルーチン(ユーザーが設定した処理ルーチン)**のスタートアドレスをPCにロードしてその処理ルーチンに分岐します。ただし、リセット例外処理についてはリターン情報の退避は行われません。
この例外処理ルーチンに分岐するまでの過渡的な状態が例外処理状態で、分岐後は通常のプログラム実行状態になります。

ユーザーが作成する例外処理ルーチンはサブルーチンの形式を取りますが、スタックに**SC**が退避されるためリターン命令は必ず**"RETE"**命令を使用する必要があります。"RETE"命令により、例外処理で中断されていたルーチンの実行を再開します。

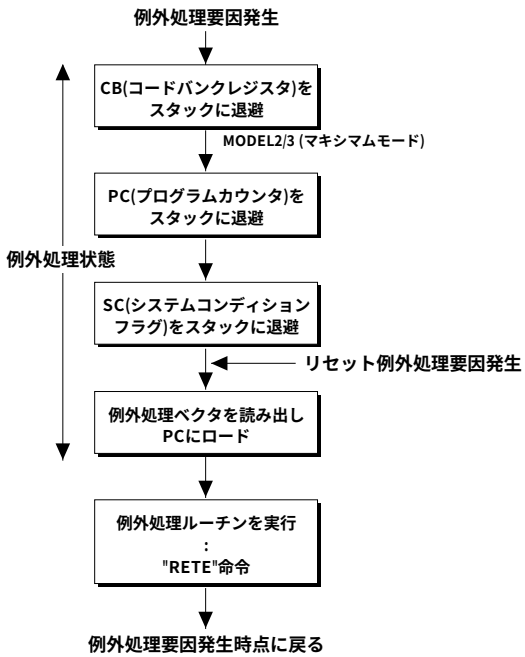


図3.5.1 例外処理フロー

3.5.1 例外処理の種類と優先度

例外処理の種類と優先度を表3.5.1.1に示します。

表3.5.1.1 例外処理の種類と優先度

優先度	種 類	例外処理の開始タイミング
高	リセット	$\overline{SR}$端子がLOWレベルからHIGHレベルに変化後の最初のフェッチサイクル
↑	ゼロ除算	除数ゼロでDIV命令(除算)を実行した場合のDIV命令実行直後
	$\overline{NMI}$	<ノンマスカブル割り込み> $\overline{NMI}$端子に立ち下がりエッジが入力された時点で実行中の命令または例外処理終了時
	$\overline{IRQ3}$	<割り込み要求3> $\overline{IRQ3}$端子にLOWレベルが入力された時点で実行中の命令または例外処理終了時
	$\overline{IRQ2}$	<割り込み要求2> $\overline{IRQ2}$端子にLOWレベルが入力された時点で実行中の命令または例外処理終了時
低	$\overline{IRQ1}$	<割り込み要求1> $\overline{IRQ1}$端子にLOWレベルが入力された時点で実行中の命令または例外処理終了時
なし	INT命令	<ソフトウェア割り込み> INT命令の実行

例外処理要因には優先順位が設定されており、複数の要因が同時に発生した場合は優先度の高い例外処理が先に実行されます。
また、例外処理状態において新たな例外処理要因が発生した場合は、その時点の例外処理終了後(例外処理ルーチンの実行前)に新たな例外処理が実行されます。
たとえば、$\overline{IRQ3}$の例外処理中に$\overline{NMI}$が発生した場合、$\overline{IRQ3}$の例外処理の最後に$\overline{NMI}$のサンプリングが行われ、ユーザーの処理ルーチンとしては$\overline{NMI}$処理ルーチンが$\overline{IRQ3}$処理ルーチンより先に実行されます。$\overline{IRQ3}$の処理ルーチンは$\overline{NMI}$の処理ルーチン終了後に実行されます。
このため、割り込みによる例外処理ではその割り込みより優先度の低い割り込みがマスクされるようになっています。
INT命令による例外処理はプログラムによって開始できますので、優先度は設定されません。

3.5.2 例外処理要因とベクタ

例外処理ルーチンのスタートアドレスは、各例外処理要因に対応したベクタアドレスに**ベクタ**として設定しておきます。
このベクタが例外処理の最後にPCにロードされ、例外処理ルーチンへ分岐します。
表3.5.2.1に例外処理要因とベクタアドレスの対応を示します。

ベクタはCPUモデルにかかわらず論理アドレスを示す2バイトのアドレス情報に固定されています。MODEL2/3においてもユーザーが作成した例外処理ルーチンのバンクをベクタにより指定することはできません。したがって、**どのバンクのプログラムを実行している場合でも共通の例外処理ルーチンに分岐させるために、ベクタにより指定される例外処理ルーチンの先頭をコモンエリア内(000000H～007FFFH)に配置する必要があります。**

表3.5.2.1 例外処理要因とベクタアドレスの対応

例外処理要因	ベクタアドレス	ベクタアドレス発生元
リセット	000000H～000001H	CPU内部
ゼロ除算	000002H～000003H	CPU内部
$\overline{NMI}$	000004H～000005H	CPU内部
$\overline{IRQ1}$～$\overline{IRQ3}$	000006H～0000FFH	周辺回路
INT命令	000000H～0000FFH	命令のオペランド

$\overline{IRQ1}$～$\overline{IRQ3}$のベクタアドレスは周辺回路によって設定されます。INT命令の場合は命令のオペランドがそのままベクタアドレスとなり、他の例外処理要因も含めると最大128ベクタまで確保できます。

3.5.3 割り込み

割り込みには**$\overline{NMI}$**、**$\overline{IRQ3}$**、**$\overline{IRQ2}$**、**$\overline{IRQ1}$**の4種類があり、それぞれには表3.5.3.1に示す割り込み優先レベルが設定されています。

表3.5.3.1 割り込み優先レベル

優先度	割り込み優先レベル	割り込み要因
高	4	$\overline{NMI}$
↑	3	$\overline{IRQ3}$
	2	$\overline{IRQ2}$
低	1	$\overline{IRQ1}$

割り込みはインタラプトフラグI0、I1によってマスク(割り込みを受け付けないように設定)することができます。プログラムによってI0、I1の2ビットに割り込み優先レベルを設定すると、それより高い優先レベルの割り込みのみが受け付けられます。このうち、レベル4の優先度を持っている$\overline{NMI}$はI0とI1の設定にかかわらず常時受け付けられます。
また、割り込み要因の発生により例外処理が実行されるとI0とI1は受け付けた割り込みと同じレベルに設定され、同一レベル以下の割り込みがマスクされます。このマスクの設定はSC(システムコンディションフラグ)のスタック退避後に行われますので、割り込み処理ルーチンを"RETE"命令で終了した時点でSCが元の状態に戻り、割り込みマスクも元の優先レベルに戻ります。
割り込み処理ルーチン内で同一レベル以下の多重割り込みを許可したい場合は、そのルーチンの中で再度優先レベルの設定を行ってください。

なお、NBおよびSCの内容を変更する命令を実行している最中は割り込みを受け付けません。その間に発生した割り込みの例外処理は、次の命令終了後に開始されます。

表3.5.3.2 割り込みマスクの設定

I1	I0	受け付け可能な割り込み
1	1	$\overline{NMI}$
1	0	$\overline{NMI}$ $\overline{IRQ3}$
0	1	$\overline{NMI}$ $\overline{IRQ3}$ $\overline{IRQ2}$
0	0	$\overline{NMI}$ $\overline{IRQ3}$ $\overline{IRQ2}$ $\overline{IRQ1}$

表3.5.3.3 割り込み受け付け後のI0とI1

受け付けた割り込み要因	I1	I0
$\overline{NMI}$	1	1
$\overline{IRQ3}$	1	1
$\overline{IRQ2}$	1	0
$\overline{IRQ1}$	0	1

3.5.4 例外処理のシーケンス

例外処理のサンプリングはSYNC信号の立ち上がりエッジ(命令の第1オペコードフェッチサイクル開始時)で行われます。ここで例外処理要因が発生している場合、CPUは割り込みアクノリッジ信号$\overline{IACK}$を出力し例外処理を開始します。$\overline{IRQ1}$～$\overline{IRQ3}$割り込みの場合、割り込みを発生させた周辺回路は$\overline{IACK}$信号を受けてベクタアドレスをホールドします。
以下に各例外処理のシーケンスを示します。

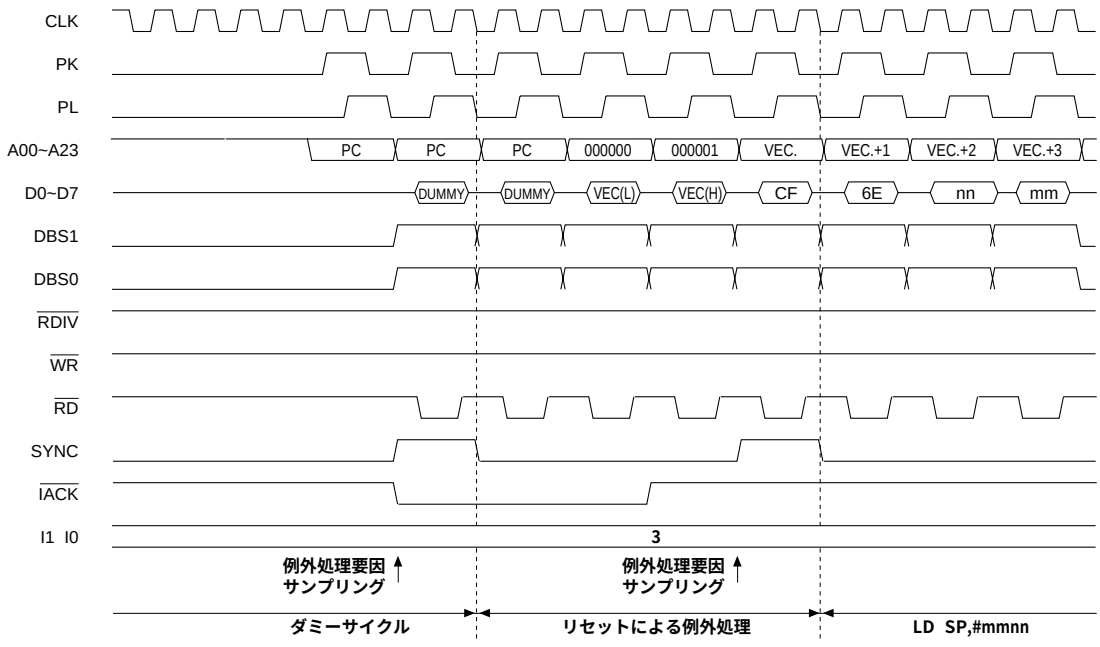


図3.5.4.1 例外処理のシーケンス －リセット－

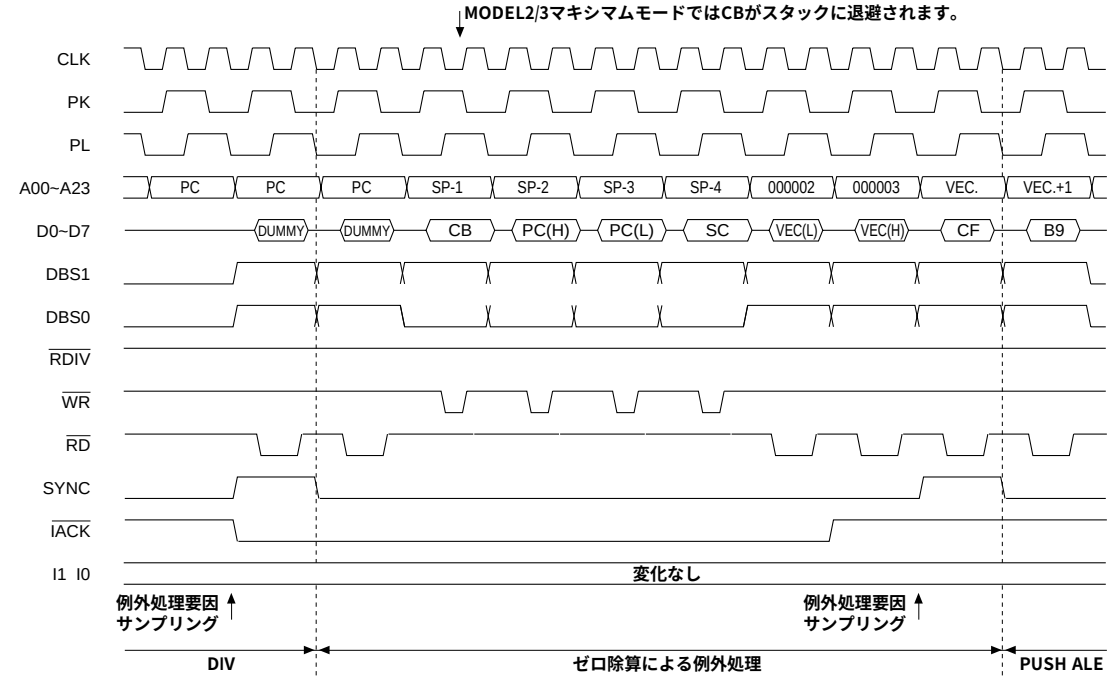


図3.5.4.2　例外処理のシーケンス　－ゼロ除算－

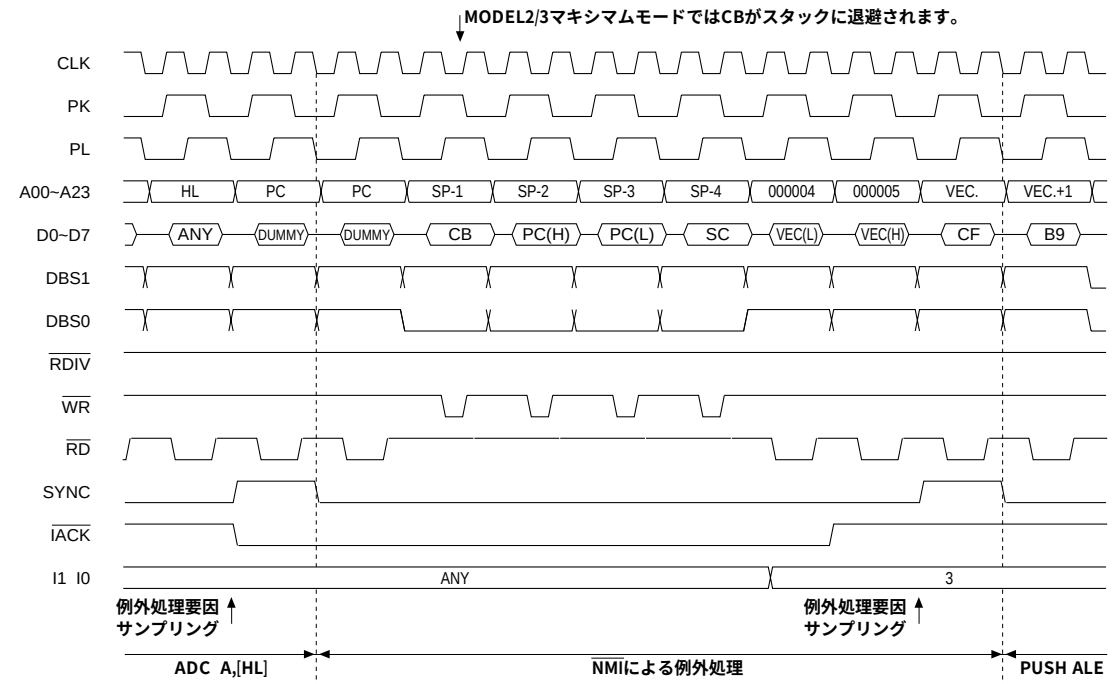


図3.5.4.3　例外処理のシーケンス　－NMI－

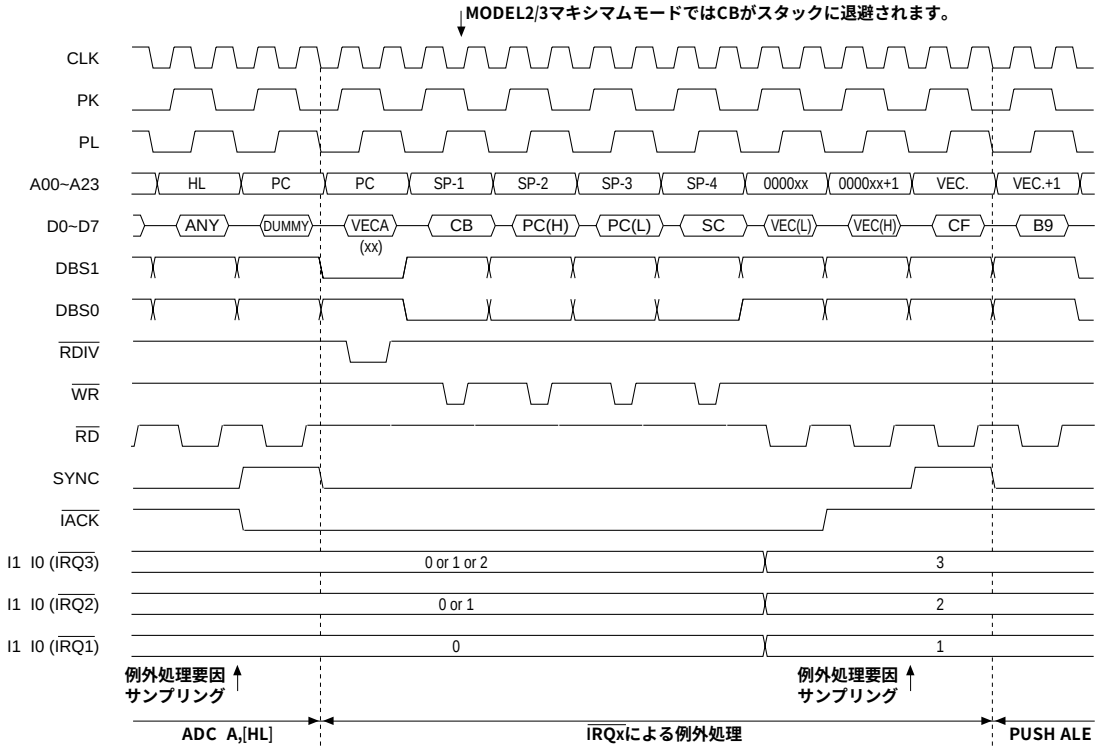


図3.5.4.4　例外処理のシーケンス　－IRQ1～IRQ3－

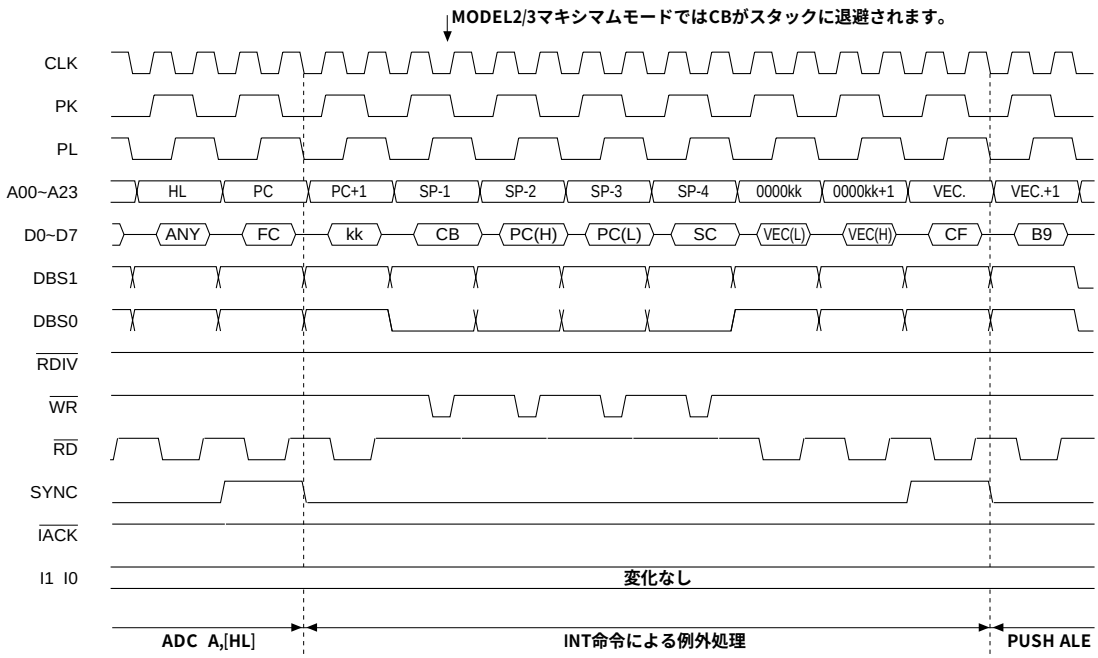


図3.5.4.5　例外処理のシーケンス　－INT命令－

3.6 バス権解放状態

S1C88では**DMA(Direct Memory Access)転送**など、CPU外部からのバス権要求に対してバスを解放する機能を持っています。この外部からの要求に応答してバスを解放している状態を**バス権解放状態**といいます。
バス権解放状態では**アドレスバス(A00～A23)、データバス(D0～D7)、およびリード/ライト信号(RD/WR)がハイインピーダンス**となり、バスマスタ(バス権解放要求を出したCPU外部のデバイス)がバスに接続されたメモリなどの他のデバイスを直接アクセスすることができます。

図3.6.1にプログラム実行状態からのバス権解放のシーケンスを示します。
バスマスタとなるデバイスはCPUの**BREQ**端子にLOWレベルを入力して、バス権の解放を要求します。
この信号に対してCPUは各バスサイクルのT2ステート(ウェイトが挿入されている場合はTw1ステート)のCLKの立ち下がりエッジおよびT4ステートのCLKの立ち下がりエッジで**2度のサンプリング**を行います。T4ステートのサンプリングの時点でT2ステートに引き続きBREQ信号がLOWレベルだった場合、CPUは実行中の命令をそのバスサイクルで中断し、**BACK**信号をLOWレベルにしてバス権解放状態に移行します。外部バスマスタはこのBACK信号を受けてバスの制御を開始します。

なお、外部バスマスタはバス権解放要求からバスの使用が終了するまでBREQ信号をLOWレベルに保持しておく必要があります。
バス権解放状態移行後、CPUはTz1およびTz2ステートを挿入し、Tz2ステートのCLKの立ち下がりエッジでBREQ信号のサンプリングを行います。このサンプリングでHIGHレベルが検出されるまでTz1/Tz2ステートは連続して挿入されます。HIGHレベルを検出した場合、CPUは次のTz2ステートのCLKの立ち上がりエッジでBACK信号をHIGHレベルに戻し、そのTz2ステート終了直後に通常のバスサイクルに復帰して中断していた処理を再開します。

前記の例外処理状態が各命令の区切りで挿入できるのに対し、バス権解放状態はバスサイクルの区切りで挿入することができます。ただし、IACK信号を出力する例外処理(INT命令以外)を実行中で、IACK信号がLOWレベルとなっている期間はバス権解放要求を受け付けません。

以上、プログラム実行状態からバス権解放状態への移行について説明しましたが、スタンバイ状態でもHALT状態からはバス権解放状態に移行することができます。
HALT状態でのバス解放要求信号のサンプリングが、Th2ステート(後述)のCLKの立ち下がりエッジで行われる点がプログラム実行状態とは異なります。
図3.6.2にHALT状態からのバス権解放のシーケンスを示します。

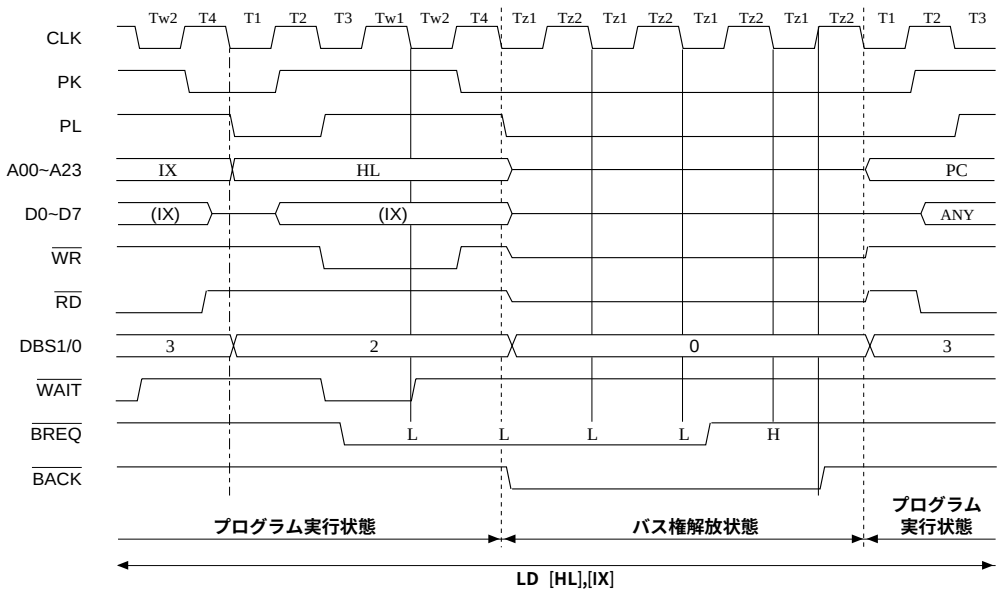


図3.6.1 プログラム実行状態からのバス権解放シーケンス

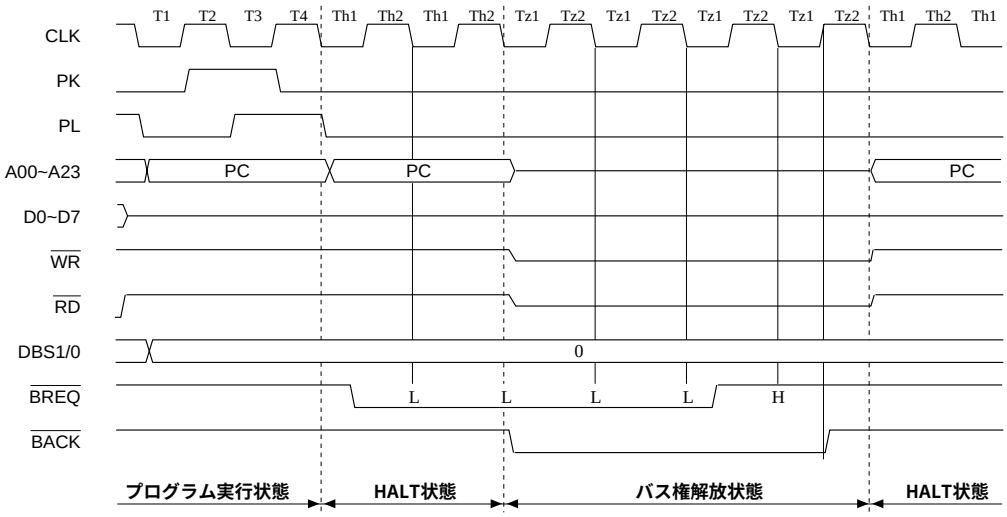


図3.6.2　HALT状態からのバス権解放シーケンス

3.7　スタンバイ状態

S1C88にはCPUの動作を停止させる機能があり、これによって消費電力を大幅に低減することができます。アプリケーションプログラムがある期間CPUに実行させる処理がない場合にこの機能を用いてCPUを停止させることができます。低消費電力化のためにCPUを停止させた状態が**スタンバイ状態**で、この状態にも以下に説明する**HALT状態**と**SLEEP状態**の2種類があります。

3.7.1 HALT状態

HALT状態はCPUのみが停止するもので、"**HALT**"命令によって移行することができます。
HALT状態からはリセットまたは任意の割り込み($\overline{NMI}$、$\overline{IRQ1}$～$\overline{IRQ3}$)によって例外処理に移行することができ、割り込みによる再起動の場合は例外処理ルーチン実行後"RETE"命令によって"HALT"命令の次の命令からプログラムの実行を再開できます。
発振回路などの周辺回路はHALT状態の間も動作しているため、MCU(S1C88 Family)の外部にCPU再起動用の割り込み回路などを設ける必要もなく、再起動も瞬時に行われます。
CPU内のレジスタ等は"HALT"命令実行時点の内容がHALT状態の間も保持されます。

図3.7.1.1にHALT状態への移行と再起動のシーケンスを示します。
HALT状態ではTh1およびTh2ステートが連続して挿入されます。この間Th2ステートのCLKの立ち下がりエッジで割り込みのサンプリングが行われ、割り込み要因の発生により即時例外処理に移行します。

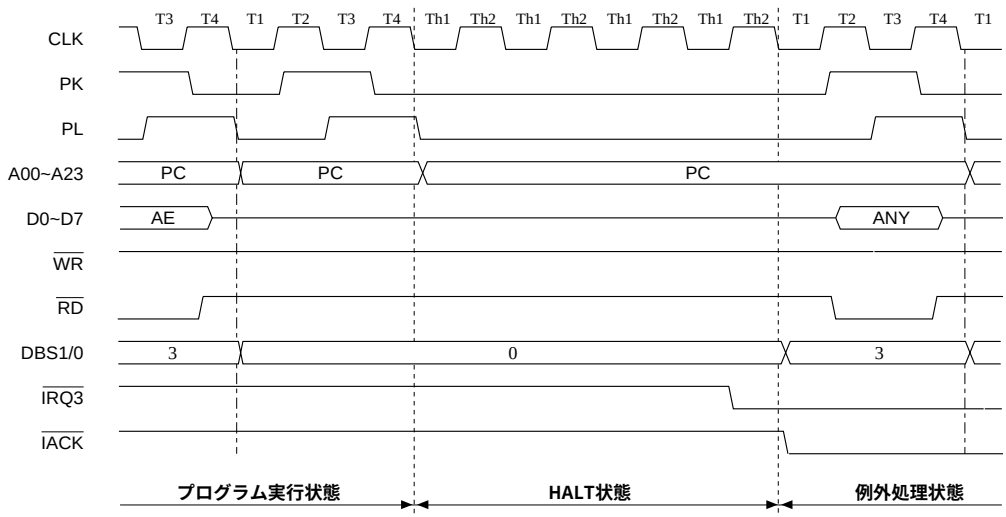


図3.7.1.1　HALT状態への移行と再起動のシーケンス

3.7.2 SLEEP状態

SLEEP状態はCPUとMCU内部の周辺回路の動作が停止するもので、"**SLP**"命令によって移行することができます。

SLEEP状態からはリセットまたはMCU外部からの割り込み($\overline{NMI}$、$\overline{IRQ1}$～$\overline{IRQ3}$)によって例外処理に移行することができ、割り込みによる再起動の場合は例外処理ルーチン実行後"RETE"命令によって"SLP"命令の次の命令からプログラムの実行を再開できます。

SLEEP状態では発振回路などの周辺回路も停止するため、HALT状態に比べて消費電力が大幅に低減できます。ただし、再起動の際に発振回路の安定待ち時間が必要となりますので、瞬時の立ち上げの必要のない長時間のスタンバイに有効です。

SLEEP状態でも規定電圧の印加によりCPU内のレジスタ等は"SLP"命令実行時点の内容が保持されます。

図3.7.2.1にSLEEP状態への移行と再起動のシーケンスを示します。

SLEEP状態で外部割り込みが発生すると、周辺回路が動作を開始し発振回路も発振を開始します。発振開始時はCPUへのCLK入力が周辺回路によってマスクされ、ある程度の安定待ち時間(数10msec～数sec)経過後にCPUへの入力が開始されます。CPUは最初に入力されたCLKの立ち下がりエッジで割り込みをサンプリングし例外処理を開始します。

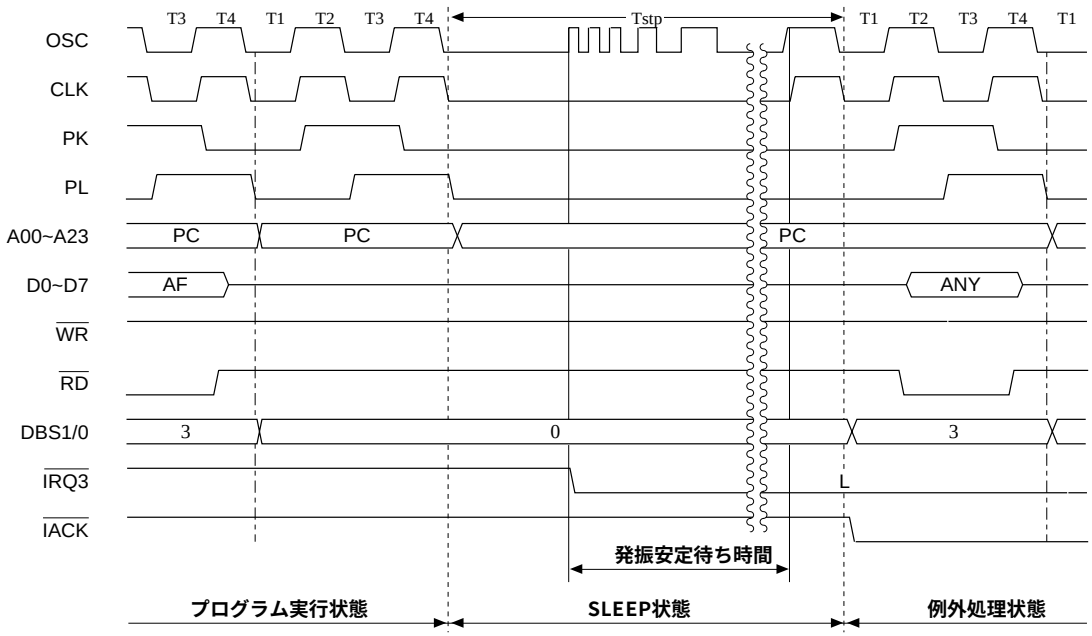


図3.7.2.1　SLEEP状態への移行と再起動のシーケンス

4 命令セット

S1C88はマシンサイクル効率が高い、高速かつ豊富な命令セットを持っています。
命令の数は608個(MODEL3)でリロケータブルプログラミングが可能な命令体系となっています。
ここでは、メモリ管理のためのアドレッシングモードと各命令の詳細について説明します。

4.1 アドレッシングモード

S1C88は以下に説明する12種類のアドレッシングモードを持っており、いろいろな状況に応じたアドレス指定が簡潔かつ的確に行えます。
以下の説明と例は、基本的にソース側を対象としています。

表4.1.1 アドレッシングモードの種類

No.	アドレッシングモード
1	即値データアドレッシング
2	レジスタ直接アドレッシング
3	レジスタ間接アドレッシング
4	ディスプレースメント付きレジスタ間接アドレッシング
5	インデックスレジスタ付きレジスタ間接アドレッシング
6	8ビット絶対アドレッシング
7	16ビット絶対アドレッシング
8	8ビット間接アドレッシング
9	16ビット間接アドレッシング
10	符号付き8ビットPC相対アドレッシング
11	符号付き16ビットPC相対アドレッシング
12	インプライドレジスタアドレッシング

■即値データアドレッシング

即値データアドレッシングは演算や転送のソースデータに即値データを用いる場合のアドレッシングモードです。命令のソースオペランドを"#"に続く**8ビット即値データ**、または**16ビット即値データ**で直接ソースデータとして指定します。
命令セットの表記においては、以下の記号によって即値データを表します。

表4.1.2 即値データのシンボル

記号	用途	サイズ	指定範囲
#nn	汎用データ	8ビット	0~255
#hh	BR設定用	8ビット	0~255
#bb	NB設定用	8ビット	0~255
#pp	ページ設定用	8ビット	0~255
#mmnn	汎用データ	16ビット	0~65535

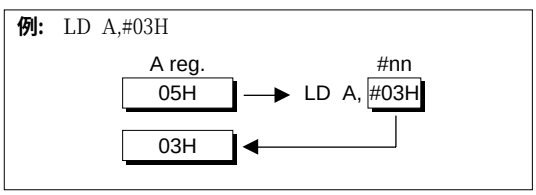


図4.1.1 即値データアドレッシング

■レジスタ直接アドレッシング

レジスタ直接アドレッシングは、ソースまたはディスティネーションにレジスタを指定する場合のアドレッシングモードです。
命令セットの表記においては、オペランドに以下のレジスタ名をそのまま使用します。

レジスタ表記の種類

8ビット: A、B、L、H、BR、SC、NB、EP、XP、YP
16ビット: BA、HL、IX、IY、PC、SP、IP(XP&YP)

NB、EP、XP、YP、IPについてはMODEL2/3のみ使用可能です。

ソースオペランドにこのモードを用いた場合、指定したレジスタの内容が演算や転送のソースデータになります。ディスティネーションオペランドに用いた場合は、そのレジスタに対してデータのストアや演算などの操作が行われます。

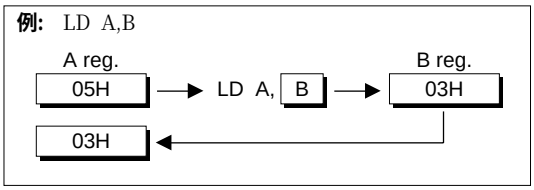


図4.1.2 レジスタ直接アドレッシング

■レジスタ間接アドレッシング

レジスタ間接アドレッシングはデータメモリをアクセスするためのアドレッシングモードで、インデックスレジスタによってデータメモリのアドレスを間接的に指定します。
アドレス指定に使用するインデックスレジスタは**HL**、**IX**、**IY**の3種類で、その内容がアクセスされるデータメモリのアドレスとなります。
命令セットにおいては、インデックスレジスタ名を[ ]で囲んで**[HL]**、**[IX]**、**[IY]**と表記します。

ソースオペランドにこのモードを用いた場合、指定したインデックスレジスタの内容がデータメモリのアドレスとなり、そのアドレスにストアされている内容がソースデータとなります。ディスティネーションオペランドに用いた場合は、指定されたデータメモリに対してデータのストアや演算などの操作が行われます。

MODEL2/3ではページ部の指定も必要になり、これにはエクスパンドページレジスタ**EP**(HL用)、**XP**(IX用)、**YP**(IY用)が使用されます。

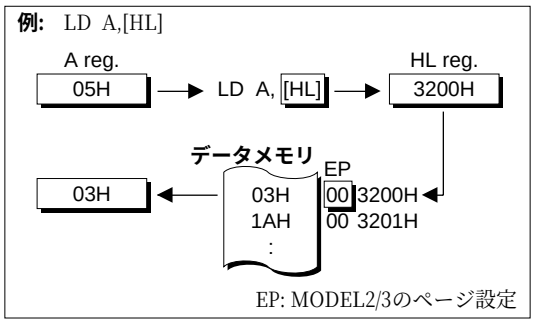


図4.1.3 レジスタ間接アドレッシング

■ディスプレースメント付きレジスタ間接アドレッシング

ディスプレースメント付きレジスタ間接アドレッシングはデータメモリをアクセスするためのアドレッシングモードで、レジスタとディスプレースメントによってデータメモリのアドレスを指定します。データメモリのアドレスは、指定した**レジスタの内容にディスプレースメント(符号付き8ビットデータ、-128～127)を加算**した値となります。
アドレス指定に使用するレジスタは**IX**、**IY**、**SP**で、命令セットにおいては、符号付き8ビットデータによるディスプレースメントに記号**dd**を使用し、**[IX+dd]**、**[IY+dd]**、**[SP+dd]**と表記します。

ソースオペランドにこのモードを用いた場合、指定されたレジスタの内容にディスプレースメントを加算した値がデータメモリのアドレスとなり、そのアドレスにストアされている内容がソースデータとなります。ディスティネーションオペランドに用いた場合は、指定されたデータメモリに対してデータのストアや演算などの操作が行われます。

MODEL2/3ではページ部の指定も必要になり、これにはエクスパンドページレジスタ**XP**(IX用)、**YP**(IY用)が使用されます。SP(スタックポインタ)を使用する場合のページ指定は各機種ごとに周辺回路上で設定されるSP用のページレジスタの内容がそのまま用いられます。

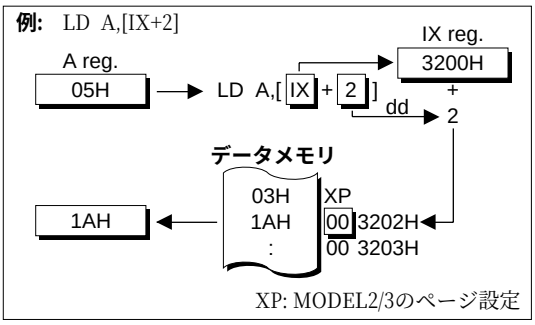


図4.1.4 ディスプレースメント付き レジスタ間接アドレッシング

■インデックスレジスタ付きレジスタ間接アドレッシング

インデックスレジスタ付きレジスタ間接アドレッシングは、ディスプレースメント付きレジスタ間接アドレッシングと同様のモードで、ディスプレースメントに8ビットデータではなく**Lレジスタ**の内容を使用します。
この場合、Lレジスタの内容は符号付き8ビットデータ(-128～127)として扱われます。
アドレス指定にはインデックスレジスタ**IX**と**IY**を使用し、ディスプレースメントとして使用されるレジスタはLレジスタに固定されます。命令セットにおいては、**[IX+L]**、**[IY+L]**と表記します。

MODEL2/3ではページ部の指定も必要になり、これにはエクスパンドページレジスタ**XP**(IX用)、**YP**(IY用)が使用されます。

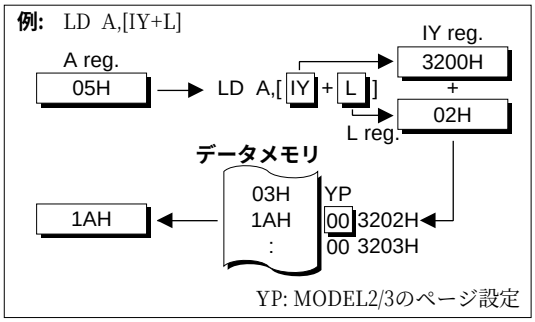


図4.1.5 インデックスレジスタ付き レジスタ間接アドレッシング

■8ビット絶対アドレッシング

8ビット絶対アドレッシングはデータメモリをアクセスするためのアドレッシングモードで、**8ビット絶対アドレスによってアドレスの下位8ビットを直接指定**します。アドレスの上位8ビットは**BRレジスタの内容により間接的に**指定されます。
命令セットにおいては、アドレス指定を行う8ビット絶対アドレス(0～255)に記号*ll*を使用し、[**BR:***ll*]と表記します。
ソースオペランドにこのモードを用いた場合、BRレジスタの内容をアドレスの上位8ビット、指定した8ビット絶対アドレスを下位8ビットとしてアドレス指定されたデータメモリにストアされている内容がソースデータとなります。ディスティネーションオペランドに用いた場合は、指定されたデータメモリに対してデータのストアや演算などの操作が行われます。

MODEL2/3ではページ部の指定も必要になり、これにはエクスパンドページレジスタ**EP**が使用されます。

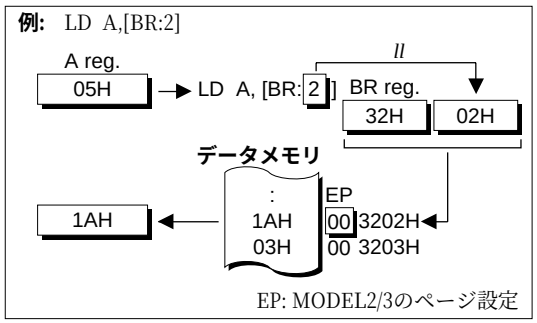


図4.1.6 8ビット絶対アドレッシング

■16ビット絶対アドレッシング

16ビット絶対アドレッシングはデータメモリをアクセスするためのアドレッシングモードで、**16ビット絶対アドレスによってアドレスを直接指定**します。命令セットにおいては、アドレス指定を行う16ビット絶対アドレス(0～65535)に記号**hh***ll*を使用し、[**hh***ll*]と表記します。
ソースオペランドにこのモードを用いた場合、指定した16ビット絶対アドレスが直接データメモリのアドレスとなり、そのアドレスにストアされている内容がソースデータとなります。ディスティネーションオペランドに用いた場合は、指定されたデータメモリに対してデータのストアや演算などの操作が行われます。

MODEL2/3ではページ部の指定も必要になり、これにはエクスパンドページレジスタ**EP**が使用されます。

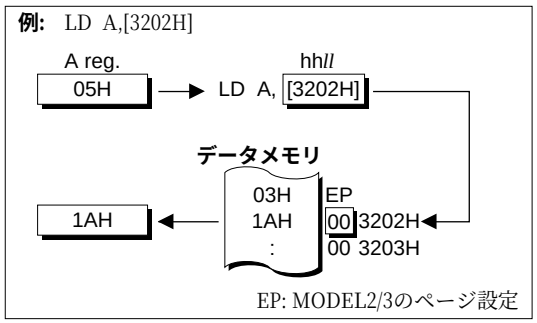


図4.1.7 16ビット絶対アドレッシング

■8ビット間接アドレッシング

8ビット間接アドレッシングは分岐命令における分岐先アドレスに**ベクタ領域**(000000H～0000FFH)の内容を使用するアドレッシングモードで、**ベクタアドレスを8ビット絶対アドレスで指定**します。指定したメモリアドレスの内容をPC(プログラムカウンタ)の下位8ビット、次のアドレスの内容をPCの上位8ビットにそれぞれロードして分岐を行います。

MODEL2/3ではNBレジスタの設定により分岐先バンクの選択も行われます。
命令セットにおいては、アドレス指定を行う8ビット絶対アドレス(0～255)に記号**kk**を使用し、[**kk**]と表記します。
このアドレッシングモードの命令は"**JP [kk]**"、"**INT [kk]**"の2種類です。

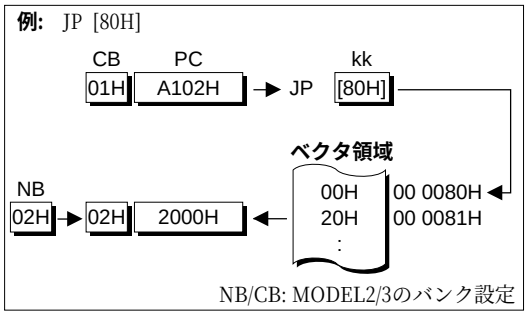


図4.1.8 8ビット間接アドレッシング

■16ビット間接アドレッシング

16ビット間接アドレッシングは"**CALL [hh***ll*]"命令のアドレッシングモードで、**分岐先アドレスを16ビット絶対アドレス(0～65535)で間接的に指定**します。指定したデータメモリアドレスの内容をPC(プログラムカウンタ)の下位8ビット、次のアドレスの内容をPCの上位8ビットにそれぞれロードして分岐を行います。

MODEL2/3ではページ部の指定も必要になり、これにはエクスパンドページレジスタ**EP**が使用されます。また、NBレジスタの設定により分岐先バンクの選択も行われます。

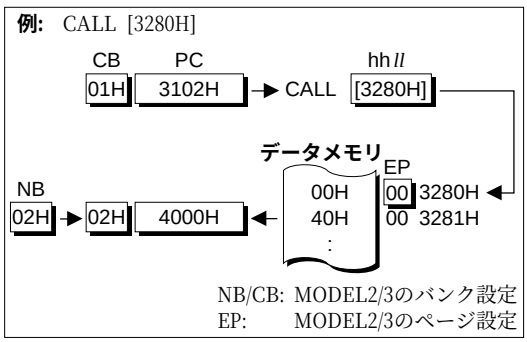


図4.1.9 16ビット間接アドレッシング

■符号付き8ビットPC相対アドレッシング

符号付き8ビットPC相対アドレッシングは分岐命令で使用されるアドレッシングモードです。オペランドで指定された**符号付き8ビット相対アドレス(-128～127)が分岐実行時のPCに加算され、そのアドレスに分岐**します。
分岐実行時のPC値は次のようになっています。
2バイト命令: PC = 命令の先頭アドレス + 1
3バイト命令: PC = 命令の先頭アドレス + 2

命令セットの表記においては、符号付き8ビット相対アドレス(-128～127)に記号**rr**を使用します。
MODEL2/3ではNBレジスタの設定により分岐先バンクの選択も行われます。

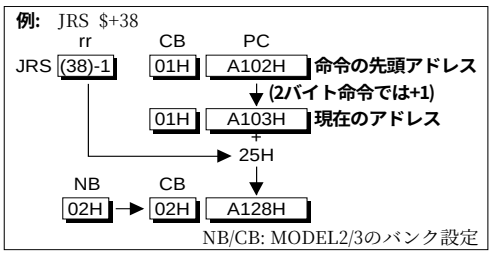


図4.1.10 符号付き8ビットPC相対アドレッシング

■符号付き16ビットPC相対アドレッシング

符号付き16ビットPC相対アドレッシングは分岐命令で使用されるアドレッシングモードです。オペランドで指定された**符号付き16ビット相対アドレス(-32768～32767)が分岐実行時のPCに加算され、そのアドレスに分岐**します。
分岐実行時のPC値は命令の先頭アドレス+2となります。
命令セットの表記においては、符号付き16ビット相対アドレス(-32768～32767)に記号**qqrr**を使用します。
MODEL2/3ではNBレジスタの設定により分岐先バンクの選択も行われます。

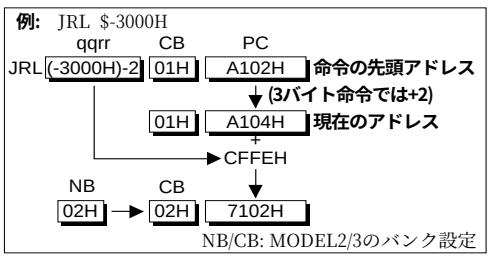


図4.1.11 符号付き16ビットPC相対アドレッシング

■インプライドレジスタアドレッシング

インプライドレジスタアドレッシングはオペランドを持たずに、暗黙的にレジスタが指定されるレジスタ直接アドレッシングモードです。
このアドレッシングモードの命令はMLT、DIV、SEP、PACK、UPCKの5種類です。

4.2 命令のフォーマット

S1C88の1命令は1バイトから4バイトのコードで次のように構成されています。

1 **オペコード**

2 **オペコード** **オペコード**
(CEH/CFH)

3 **オペコード** **オペランド**

4 **オペコード** **オペコード** **オペランド**
(CEH/CFH)

5 **オペコード** **オペランド** **オペランド**

6 **オペコード** **オペコード** **オペランド** **オペランド**
(CEH/CFH)

図4.2.1 命令のフォーマット

■オペコード

S1C88の命令セットは608種類(MODEL3)あり、1バイトのオペコードではすべての命令を表現できません。そこで、コードの**CEH**と**CFH**を**拡張コード**として第1オペコードに使用し、続く1バイトを第２オペコードとして命令を拡張しています。16ビット演算/転送命令およびスタック操作命令をコードCFHで、その他の命令をコードCEHでそれぞれ拡張しています。

また、各命令のアドレッシングモードは、第1または第2オペコードの下位3ビットで指定されます。レジスタ直接アドレッシング、レジスタ間接アドレッシング、インデックスレジスタ付きレジスタ間接アドレッシングの命令がオペコードのみで構成されます。

■オペランド

8ビットの即値データアドレッシング、ディスプレースメント付きレジスタ間接アドレッシング、8ビット絶対アドレッシング(ソースをレジスタで指定している場合)、8ビット間接アドレッシング、符号付き8ビットPC相対アドレッシングの命令は1バイトのオペランドを持っており、**8ビットデータによる指定値がそのままオペランドとなります**。

16ビットの即値データアドレッシング、8ビット絶対アドレッシング(ソースを即値で指定している場合)、16ビット絶対アドレッシング、16ビット間接アドレッシング、符号付き16ビットPC相対アドレッシングの命令は2バイトのオペランドを持っており、**16ビットデータによる指定値の下位8ビットが第1オペランド、上位8ビットが第2オペランドとなります**。(8ビット絶対アドレッシングの場合は、アドレス指定が第1オペランド、即値データが第2オペランドとなります。)

4.3 命令セット一覧表

ここではS1C88の命令セットを機能別に分類した一覧表を掲載しています。
なお、"APPENDIX"にもアドレッシングモード別一覧表が掲載されていますので、必要に応じて参照してください。

4.3.1 機能分類

命令の機能分類を表4.3.1.1に示します。

表4.3.1.1 命令の機能分類

機能分類	ニーモニック	オペレーション
8ビット算術論理演算	ADD	加算
	ADC	キャリー付き加算
	SUB	減算
	SBC	キャリー付き減算
	AND	論理積
	OR	論理和
	XOR	排他的論理和
	CP	比較
	BIT	ビットテスト
	INC	1加算
	DEC	1減算
	MLT	乗算
	DIV	除算
	CPL	1の補数
	NEG	2の補数
8ビット転送	LD	ロード
	EX	バイト交換
	SWAP	ニブル交換
ローテート/シフト	RL	キャリー付き左ローテート
	RLC	左ローテート
	RR	キャリー付き右ローテート
	RRC	右ローテート
	SLA	算術左シフト
	SLL	論理左シフト
	SRA	算術右シフト
	SRL	論理右シフト
演算補助	PACK	パック
	UPCK	アンパック
	SEP	符号拡張
16ビット算術演算	ADD	加算
	ADC	キャリー付き加算
	SUB	減算
	SBC	キャリー付き減算
	CP	比較
	INC	1加算
	DEC	1減算
16ビット転送	LD	ロード
	EX	ワード交換
スタック制御	PUSH	プッシュ
	POP	ポップ
分岐	JRS	相対ショートジャンプ
	JRL	相対ロングジャンプ
	JP	間接ジャンプ
	DJR	ループ
	CARS	相対ショートコール
	CARL	相対ロングコール
	CALL	間接コール
	RET	リターン
	RETE	例外処理リターン
	RETS	スキップリターン
	INT	ソフトウェア割り込み
システム制御	NOP	ノーオペレーション
	HALT	HALT状態移行
	SLP	SLEEP状態移行

4.3.2 記号の意味

次項の機能別命令一覧表で使われている記号の意味を表4.3.2.1に示します。

表4.3.2.1 記号の意味

レジスタ関係	
A	データレジスタA
A(H)	Aレジスタの上位4ビット
A(L)	Aレジスタの下位4ビット
B	データレジスタB
BA	BAペアレジスタ
H	データレジスタH
L	データレジスタL
HL	インデックスレジスタHL
IX	インデックスレジスタIX
IX(H)	IXレジスタの上位8ビット
IX(L)	IXレジスタの下位8ビット
IY	インデックスレジスタIY
IY(H)	IYレジスタの上位8ビット
IY(L)	IYレジスタの下位8ビット
SP	スタックポインタSP
BR	ベースレジスタBR
SC	システムコンディションフラグSC
CC	カスタマイズコンディションフラグCC
PC	プログラムカウンタPC
PC(H)	PCの上位8ビット
PC(L)	PCの下位8ビット
NB	ニューコードバンクレジスタNB
CB	コードバンクレジスタCB
EP	エクスパンドページレジスタEP
XP	IX用エクスパンドページレジスタXP
YP	IY用エクスパンドページレジスタYP
IP	XPおよびYPレジスタ

即値データ	
nn	8ビット即値データ(符号なし)
hh	絶対アドレス上位8ビット設定データ(符号なし)
ll	絶対アドレス下位8ビット設定データ(符号なし)
pp	ページ設定データ(符号なし)
bb	バンク設定データ(符号なし)
dd	符号付き8ビットディスプレースメント
rr	8ビット相対アドレス設定データ(符号付き)
kk	ベクタアドレス設定データ(符号なし)
mmnn	16ビット即値データ(符号なし)
hh*ll*	16ビット絶対アドレス設定データ(符号なし)
qqrr	16ビット相対アドレス設定データ(符号付き)

メモリ関係	
[HL]	HLレジスタで指定されるメモリ
[HL](H)	[HL]の上位4ビット
[HL](L)	[HL]の下位4ビット
[IX]	IXレジスタで指定されるメモリ
[IX+dd]	IXレジスタ+ddで指定されるメモリ
[IX+L]	IXレジスタ+Lレジスタで指定されるメモリ
[IY]	IYレジスタで指定されるメモリ
[IY+dd]	IYレジスタ+ddで指定されるメモリ
[IY+L]	IYレジスタ+Lレジスタで指定されるメモリ
[BR:*ll*]	BRレジスタと"*ll*"で指定されるメモリ
[hh*ll*]	"hh*ll*"で指定されるメモリ
[kk]	"kk"で指定されるベクタ
[SP]	SPで指定されるスタック
[SP+dd]	SP+ddで指定されるスタック

フラグ関係	
Z	ゼロフラグ
C	キャリーフラグ
V	オーバーフローフラグ
N	ネガティブフラグ
D	デシマルフラグ
U	アンパックフラグ
I0	インタラプトフラグ0
I1	インタラプトフラグ1
↕	フラグのセット/リセット
–	変化なし
0	フラグのリセット
F0	カスタマイズコンディションフラグF0
F1	カスタマイズコンディションフラグF1
F2	カスタマイズコンディションフラグF2
F3	カスタマイズコンディションフラグF3

演算、その他	
+	加算
-	減算
*	乗算
/	除算
∧	論理積
∨	論理和
∀	排他的論理和
★	10進演算/アンパック演算可能な命令

4.3.3 機能別命令一覧表

表4.3.3.1(a) 機能別命令一覧表 (8ビット転送命令 1/3)

8ビット転送命令

ニーモニック		コード	オペレーション	サイクル	バイト	SC I1	SC I0	SC U	SC D	SC N	SC V	SC C	SC Z	備 考	ページ
LD	A,A	40	A←A	1	1	–	–	–	–	–	–	–	–		*101*
	A,B	41	A←B	1	1	–	–	–	–	–	–	–	–		*101*
	A,L	42	A←L	1	1	–	–	–	–	–	–	–	–		*101*
	A,H	43	A←H	1	1	–	–	–	–	–	–	–	–		*101*
	A,BR	CE,C0	A←BR	2	2	–	–	–	–	–	–	–	–		*101*
	A,SC	CE,C1	A←SC	2	2	–	–	–	–	–	–	–	–		*101*
	A,#nn	B0,nn	A←nn	2	2	–	–	–	–	–	–	–	–		*108*
	A,[BR:*ll*]	44,*ll*	A←[BR:*ll*]	3	2	–	–	–	–	–	–	–	–		*111*
	A,[hh*ll*]	CE,D0,*ll*,hh	A←[hh*ll*]	5	4	–	–	–	–	–	–	–	–		*113*
	A,[HL]	45	A←[HL]	2	1	–	–	–	–	–	–	–	–		*113*
	A,[IX]	46	A←[IX]	2	1	–	–	–	–	–	–	–	–		*115*
	A,[IY]	47	A←[IY]	2	1	–	–	–	–	–	–	–	–		*116*
	A,[IX+dd]	CE,40,dd	A←[IX+dd]	4	3	–	–	–	–	–	–	–	–		*118*
	A,[IY+dd]	CE,41,dd	A←[IY+dd]	4	3	–	–	–	–	–	–	–	–		*119*
	A,[IX+L]	CE,42	A←[IX+L]	4	2	–	–	–	–	–	–	–	–		*121*
	A,[IY+L]	CE,43	A←[IY+L]	4	2	–	–	–	–	–	–	–	–		*122*
	A,NB	CE,C8	A←NB	2	2	–	–	–	–	–	–	–	–		*102*
	A,EP	CE,C9	A←EP	2	2	–	–	–	–	–	–	–	–	MODEL2/3	*102*
	A,XP	CE,CA	A←XP	2	2	–	–	–	–	–	–	–	–	のみ	*102*
	A,YP	CE,CB	A←YP	2	2	–	–	–	–	–	–	–	–		*102*
LD	B,A	48	B←A	1	1	–	–	–	–	–	–	–	–		*101*
	B,B	49	B←B	1	1	–	–	–	–	–	–	–	–		*101*
	B,L	4A	B←L	1	1	–	–	–	–	–	–	–	–		*101*
	B,H	4B	A←H	1	1	–	–	–	–	–	–	–	–		*101*
	B,#nn	B1,nn	B←nn	2	2	–	–	–	–	–	–	–	–		*108*
	B,[BR:*ll*]	4C,*ll*	B←[BR:*ll*]	3	2	–	–	–	–	–	–	–	–		*111*
	B,[hh*ll*]	CE,D1,*ll*,hh	B←[hh*ll*]	5	4	–	–	–	–	–	–	–	–		*113*
	B,[HL]	4D	B←[HL]	2	1	–	–	–	–	–	–	–	–		*113*
	B,[IX]	4E	B←[IX]	2	1	–	–	–	–	–	–	–	–		*115*
	B,[IY]	4F	B←[IY]	2	1	–	–	–	–	–	–	–	–		*116*
	B,[IX+dd]	CE,48,dd	B←[IX+dd]	4	3	–	–	–	–	–	–	–	–		*118*
	B,[IY+dd]	CE,49,dd	B←[IY+dd]	4	3	–	–	–	–	–	–	–	–		*119*
	B,[IX+L]	CE,4A	B←[IX+L]	4	2	–	–	–	–	–	–	–	–		*121*
	B,[IY+L]	CE,4B	B←[IY+L]	4	2	–	–	–	–	–	–	–	–		*122*
LD	L,A	50	L←A	1	1	–	–	–	–	–	–	–	–		*101*
	L,B	51	L←B	1	1	–	–	–	–	–	–	–	–		*101*
	L,L	52	L←L	1	1	–	–	–	–	–	–	–	–		*101*
	L,H	53	L←H	1	1	–	–	–	–	–	–	–	–		*101*
	L,#nn	B2,nn	L←nn	2	2	–	–	–	–	–	–	–	–		*108*
	L,[BR:*ll*]	54,*ll*	L←[BR:*ll*]	3	2	–	–	–	–	–	–	–	–		*111*
	L,[hh*ll*]	CE,D2,*ll*,hh	L←[hh*ll*]	5	4	–	–	–	–	–	–	–	–		*113*
	L,[HL]	55	L←[HL]	2	1	–	–	–	–	–	–	–	–		*113*
	L,[IX]	56	L←[IX]	2	1	–	–	–	–	–	–	–	–		*115*
	L,[IY]	57	L←[IY]	2	1	–	–	–	–	–	–	–	–		*116*
	L,[IX+dd]	CE,50,dd	L←[IX+dd]	4	3	–	–	–	–	–	–	–	–		*118*
	L,[IY+dd]	CE,51,dd	L←[IY+dd]	4	3	–	–	–	–	–	–	–	–		*119*
	L,[IX+L]	CE,52	L←[IX+L]	4	2	–	–	–	–	–	–	–	–		*121*
	L,[IY+L]	CE,53	L←[IY+L]	4	2	–	–	–	–	–	–	–	–		*122*

* ニューコードバンクレジスタNBおよびエクスパンドページレジスタEP/XP/YPはMODEL2/3にのみ設定されています。したがって、MODEL0/1ではこれらのレジスタをアクセスする命令は使用できません。

表4.3.3.1(b) 機能別命令一覧表 (8ビット転送命令 2/3)

8ビット転送命令

ニーモニック		コード	オペレーション	サイクル	バイト	SC								備 考	ページ
						I1	I0	U	D	N	V	C	Z		
LD	H,A	58	H←A	1	1	–	–	–	–	–	–	–	–		*101*
	H,B	59	H←B	1	1	–	–	–	–	–	–	–	–		*101*
	H,L	5A	H←L	1	1	–	–	–	–	–	–	–	–		*101*
	H,H	5B	H←H	1	1	–	–	–	–	–	–	–	–		*101*
	H,#nn	B3,nn	H←nn	2	2	–	–	–	–	–	–	–	–		*108*
	H,[BR:*ll*]	5C,*ll*	H←[BR:*ll*]	3	2	–	–	–	–	–	–	–	–		*111*
	H,[hh*ll*]	CE,D3,*ll*,hh	H←[hh*ll*]	5	4	–	–	–	–	–	–	–	–		*113*
	H,[HL]	5D	H←[HL]	2	1	–	–	–	–	–	–	–	–		*113*
	H,[IX]	5E	H←[IX]	2	1	–	–	–	–	–	–	–	–		*115*
	H,[IY]	5F	H←[IY]	2	1	–	–	–	–	–	–	–	–		*116*
	H,[IX+dd]	CE,58,dd	H←[IX+dd]	4	3	–	–	–	–	–	–	–	–		*118*
	H,[IY+dd]	CE,59,dd	H←[IY+dd]	4	3	–	–	–	–	–	–	–	–		*119*
	H,[IX+L]	CE,5A	H←[IX+L]	4	2	–	–	–	–	–	–	–	–		*121*
	H,[IY+L]	CE,5B	H←[IY+L]	4	2	–	–	–	–	–	–	–	–		*122*
LD	BR,A	CE,C2	BR←A	2	2	–	–	–	–	–	–	–	–		*102*
	BR,#hh	B4,hh	BR←hh	2	2	–	–	–	–	–	–	–	–		*108*
LD	SC,A	CE,C3	SC←A	3	2	↕	↕	↕	↕	↕	↕	↕	↕		*102*
	SC,#nn	9F,nn	SC←nn	3	2	↕	↕	↕	↕	↕	↕	↕	↕		*108*
LD	[BR:*ll*],A	78,*ll*	[BR:*ll*]←A	3	2	–	–	–	–	–	–	–	–		*103*
	[BR:*ll*],B	79,*ll*	[BR:*ll*]←B	3	2	–	–	–	–	–	–	–	–		*103*
	[BR:*ll*],L	7A,*ll*	[BR:*ll*]←L	3	2	–	–	–	–	–	–	–	–		*103*
	[BR:*ll*],H	7B,*ll*	[BR:*ll*]←H	3	2	–	–	–	–	–	–	–	–		*103*
	[BR:*ll*],#nn	DD,*ll*,nn	[BR:*ll*]←nn	4	3	–	–	–	–	–	–	–	–		*110*
	[BR:*ll*],[HL]	7D,*ll*	[BR:*ll*]←[HL]	4	2	–	–	–	–	–	–	–	–		*114*
	[BR:*ll*],[IX]	7E,*ll*	[BR:*ll*]←[IX]	4	2	–	–	–	–	–	–	–	–		*115*
	[BR:*ll*],[IY]	7F,*ll*	[BR:*ll*]←[IY]	4	2	–	–	–	–	–	–	–	–		*117*
LD	[hh*ll*],A	CE,D4,*ll*,hh	[hh*ll*]←A	5	4	–	–	–	–	–	–	–	–		*104*
	[hh*ll*],B	CE,D5,*ll*,hh	[hh*ll*]←B	5	4	–	–	–	–	–	–	–	–		*104*
	[hh*ll*],L	CE,D6,*ll*,hh	[hh*ll*]←L	5	4	–	–	–	–	–	–	–	–		*104*
	[hh*ll*],H	CE,D7,*ll*,hh	[hh*ll*]←H	5	4	–	–	–	–	–	–	–	–		*104*
LD	[HL],A	68	[HL]←A	2	1	–	–	–	–	–	–	–	–		*104*
	[HL],B	69	[HL]←B	2	1	–	–	–	–	–	–	–	–		*104*
	[HL],L	6A	[HL]←L	2	1	–	–	–	–	–	–	–	–		*104*
	[HL],H	6B	[HL]←H	2	1	–	–	–	–	–	–	–	–		*104*
	[HL],#nn	B5,nn	[HL]←nn	3	2	–	–	–	–	–	–	–	–		*110*
	[HL],[BR:*ll*]	6C,*ll*	[HL]←[BR:*ll*]	4	2	–	–	–	–	–	–	–	–		*112*
	[HL],[HL]	6D	[HL]←[HL]	3	1	–	–	–	–	–	–	–	–		*114*
	[HL],[IX]	6E	[HL]←[IX]	3	1	–	–	–	–	–	–	–	–		*115*
	[HL],[IY]	6F	[HL]←[IY]	3	1	–	–	–	–	–	–	–	–		*117*
	[HL],[IX+dd]	CE,60,dd	[HL]←[IX+dd]	5	3	–	–	–	–	–	–	–	–		*118*
	[HL],[IY+dd]	CE,61,dd	[HL]←[IY+dd]	5	3	–	–	–	–	–	–	–	–		*120*
	[HL],[IX+L]	CE,62	[HL]←[IX+L]	5	2	–	–	–	–	–	–	–	–		*121*
	[HL],[IY+L]	CE,63	[HL]←[IY+L]	5	2	–	–	–	–	–	–	–	–		*123*
LD	[IX],A	60	[IX]←A	2	1	–	–	–	–	–	–	–	–		*105*
	[IX],B	61	[IX]←B	2	1	–	–	–	–	–	–	–	–		*105*
	[IX],L	62	[IX]←L	2	1	–	–	–	–	–	–	–	–		*105*
	[IX],H	63	[IX]←H	2	1	–	–	–	–	–	–	–	–		*105*
	[IX],#nn	B6,nn	[IX]←nn	3	2	–	–	–	–	–	–	–	–		*111*

表4.3.3.1(c) 機能別命令一覧表 (8ビット転送命令 3/3)

8ビット転送命令

ニーモニック		コード	オペレーション	サイクル	バイト	SC								備 考	ページ
						I1	I0	U	D	N	V	C	Z		
LD	[IX],[BR:*ll*]	64,*ll*	[IX]←[BR:*ll*]	4	2	–	–	–	–	–	–	–	–		*112*
	[IX],[HL]	65	[IX]←[HL]	3	1	–	–	–	–	–	–	–	–		*114*
	[IX],[IX]	66	[IX]←[IX]	3	1	–	–	–	–	–	–	–	–		*116*
	[IX],[IY]	67	[IX]←[IY]	3	1	–	–	–	–	–	–	–	–		*117*
	[IX],[IX+dd]	CE,68,dd	[IX]←[IX+dd]	5	3	–	–	–	–	–	–	–	–		*119*
	[IX],[IY+dd]	CE,69,dd	[IX]←[IY+dd]	5	3	–	–	–	–	–	–	–	–		*120*
	[IX],[IX+L]	CE,6A	[IX]←[IX+L]	5	2	–	–	–	–	–	–	–	–		*122*
	[IX],[IY+L]	CE,6B	[IX]←[IY+L]	5	2	–	–	–	–	–	–	–	–		*123*
LD	[IY],A	70	[IY]←A	2	1	–	–	–	–	–	–	–	–		*105*
	[IY],B	71	[IY]←B	2	1	–	–	–	–	–	–	–	–		*105*
	[IY],L	72	[IY]←L	2	1	–	–	–	–	–	–	–	–		*105*
	[IY],H	73	[IY]←H	2	1	–	–	–	–	–	–	–	–		*105*
	[IY],#nn	B7,nn	[IY]←nn	3	2	–	–	–	–	–	–	–	–		*111*
	[IY],[BR:*ll*]	74,ll	[IY]←[BR:*ll*]	4	2	–	–	–	–	–	–	–	–		*112*
	[IY],[HL]	75	[IY]←[HL]	3	1	–	–	–	–	–	–	–	–		*114*
	[IY],[IX]	76	[IY]←[IX]	3	1	–	–	–	–	–	–	–	–		*116*
	[IY],[IY]	77	[IY]←[IY]	3	1	–	–	–	–	–	–	–	–		*117*
	[IY],[IX+dd]	CE,78,dd	[IY]←[IX+dd]	5	3	–	–	–	–	–	–	–	–		*119*
	[IY],[IY+dd]	CE,79,dd	[IY]←[IY+dd]	5	3	–	–	–	–	–	–	–	–		*120*
	[IY],[IX+L]	CE,7A	[IY]←[IX+L]	5	2	–	–	–	–	–	–	–	–		*122*
	[IY],[IY+L]	CE,7B	[IY]←[IY+L]	5	2	–	–	–	–	–	–	–	–		*123*
LD	[IX+dd],A	CE,44,dd	[IX+dd]←A	4	3	–	–	–	–	–	–	–	–		*106*
	[IX+dd],B	CE,4C,dd	[IX+dd]←B	4	3	–	–	–	–	–	–	–	–		*106*
	[IX+dd],L	CE,54,dd	[IX+dd]←L	4	3	–	–	–	–	–	–	–	–		*106*
	[IX+dd],H	CE,5C,dd	[IX+dd]←H	4	3	–	–	–	–	–	–	–	–		*106*
LD	[IY+dd],A	CE,45,dd	[IY+dd]←A	4	3	–	–	–	–	–	–	–	–		*106*
	[IY+dd],B	CE,4D,dd	[IY+dd]←B	4	3	–	–	–	–	–	–	–	–		*106*
	[IY+dd],L	CE,55,dd	[IY+dd]←L	4	3	–	–	–	–	–	–	–	–		*106*
	[IY+dd],H	CE,5D,dd	[IY+dd]←H	4	3	–	–	–	–	–	–	–	–		*106*
LD	[IX+L],A	CE,46	[IX+L]←A	4	2	–	–	–	–	–	–	–	–		*107*
	[IX+L],B	CE,4E	[IX+L]←B	4	2	–	–	–	–	–	–	–	–		*107*
	[IX+L],L	CE,56	[IX+L]←L	4	2	–	–	–	–	–	–	–	–		*107*
	[IX+L],H	CE,5E	[IX+L]←H	4	2	–	–	–	–	–	–	–	–		*107*
LD	[IY+L],A	CE,47	[IY+L]←A	4	2	–	–	–	–	–	–	–	–		*107*
	[IY+L],B	CE,4F	[IY+L]←B	4	2	–	–	–	–	–	–	–	–		*107*
	[IY+L],L	CE,57	[IY+L]←L	4	2	–	–	–	–	–	–	–	–		*107*
	[IY+L],H	CE,5F	[IY+L]←H	4	2	–	–	–	–	–	–	–	–		*107*
LD	NB,A	CE,CC	NB←A	3	2	–	–	–	–	–	–	–	–		*103*
	NB,#bb	CE,C4,bb	NB←bb	4	3	–	–	–	–	–	–	–	–		*109*
LD	EP,A	CE,CD	EP←A	2	2	–	–	–	–	–	–	–	–		*103*
	EP,#pp	CE,C5,pp	EP←pp	3	3	–	–	–	–	–	–	–	–	MODEL2/3	*109*
LD	XP,A	CE,CE	XP←A	2	2	–	–	–	–	–	–	–	–	のみ	*103*
	XP,#pp	CE,C6,pp	XP←pp	3	3	–	–	–	–	–	–	–	–		*109*
LD	YP,A	CE,CF	YP←A	2	2	–	–	–	–	–	–	–	–		*103*
	YP,#pp	CE,C7,pp	YP←pp	3	3	–	–	–	–	–	–	–	–		*110*
EX	A,B	CC	A↔B	2	1	–	–	–	–	–	–	–	–		*93*
	A,[HL]	CD	A↔[HL]	3	1	–	–	–	–	–	–	–	–		*93*
SWAP	A	F6	$A_{(H)}$↔$A_{(L)}$	2	1	–	–	–	–	–	–	–	–		*172*
	[HL]	F7	$[HL]_{(H)}$↔$[HL]_{(L)}$	3	1	–	–	–	–	–	–	–	–		*173*

* ニューコードバンクレジスタNBおよびエクスパンドページレジスタEP/XP/YPはMODEL2/3にのみ設定されています。したがって、MODEL0/1ではこれらのレジスタをアクセスする命令は使用できません。

表4.3.3.1(d) 機能別命令一覧表 (16ビット転送命令 1/2)

16ビット転送命令

ニーモニック		コード	オペレーション	サイクル	バイト	SC								備 考	ページ
						I1	I0	U	D	N	V	C	Z		
LD	BA,BA	CF,E0	BA←BA	2	2	–	–	–	–	–	–	–	–		*124*
	BA,HL	CF,E1	BA←HL	2	2	–	–	–	–	–	–	–	–		*124*
	BA,IX	CF,E2	BA←IX	2	2	–	–	–	–	–	–	–	–		*124*
	BA,IY	CF,E3	BA←IY	2	2	–	–	–	–	–	–	–	–		*124*
	BA,SP	CF,F8	BA←SP	2	2	–	–	–	–	–	–	–	–		*124*
	BA,PC	CF,F9	BA←PC+2	2	2	–	–	–	–	–	–	–	–		*124*
	BA,#mmnn	C4,nn,mm	BA←mmnn	3	3	–	–	–	–	–	–	–	–		*129*
	BA,[hh*ll*]	B8,*ll*,hh	A←[hh*ll*], B←[hh*ll*+1]	5	3	–	–	–	–	–	–	–	–		*130*
	BA,[HL]	CF,C0	A←[HL], B←[HL+1]	5	2	–	–	–	–	–	–	–	–		*131*
	BA,[IX]	CF,D0	A←[IX], B←[IX+1]	5	2	–	–	–	–	–	–	–	–		*132*
	BA,[IY]	CF,D8	A←[IY], B←[IY+1]	5	2	–	–	–	–	–	–	–	–		*132*
	BA,[SP+dd]	CF,70,dd	A←[SP+dd], B←[SP+dd+1]	6	3	–	–	–	–	–	–	–	–		*133*
LD	HL,BA	CF,E4	HL←BA	2	2	–	–	–	–	–	–	–	–		*124*
	HL,HL	CF,E5	HL←HL	2	2	–	–	–	–	–	–	–	–		*124*
	HL,IX	CF,E6	HL←IX	2	2	–	–	–	–	–	–	–	–		*124*
	HL,IY	CF,E7	HL←IY	2	2	–	–	–	–	–	–	–	–		*124*
	HL,SP	CF,F4	HL←SP	2	2	–	–	–	–	–	–	–	–		*125*
	HL,PC	CF,F5	HL←PC+2	2	2	–	–	–	–	–	–	–	–		*125*
	HL,#mmnn	C5,nn,mm	HL←mmnn	3	3	–	–	–	–	–	–	–	–		*129*
	HL,[hh*ll*]	B9,*ll*,hh	L←[hh*ll*], H←[hh*ll*+1]	5	3	–	–	–	–	–	–	–	–		*130*
	HL,[HL]	CF,C1	L←[HL], H←[HL+1]	5	2	–	–	–	–	–	–	–	–		*131*
	HL,[IX]	CF,D1	L←[IX], H←[IX+1]	5	2	–	–	–	–	–	–	–	–		*132*
	HL,[IY]	CF,D9	L←[IY], H←[IY+1]	5	2	–	–	–	–	–	–	–	–		*132*
	HL,[SP+dd]	CF,71,dd	L←[SP+dd], H←[SP+dd+1]	6	3	–	–	–	–	–	–	–	–		*133*
LD	IX,BA	CF,E8	IX←BA	2	2	–	–	–	–	–	–	–	–		*124*
	IX,HL	CF,E9	IX←HL	2	2	–	–	–	–	–	–	–	–		*124*
	IX,IX	CF,EA	IX←IX	2	2	–	–	–	–	–	–	–	–		*124*
	IX,IY	CF,EB	IX←IY	2	2	–	–	–	–	–	–	–	–		*124*
	IX,SP	CF,FA	IX←SP	2	2	–	–	–	–	–	–	–	–		*125*
	IX,#mmnn	C6,nn,mm	IX←mmnn	3	3	–	–	–	–	–	–	–	–		*129*
	IX,[hh*ll*]	BA,*ll*,hh	IX(L)←[hh*ll*], IX(H)←[hh*ll*+1]	5	3	–	–	–	–	–	–	–	–		*130*
	IX,[HL]	CF,C2	IX(L)←[HL], IX(H)←[HL+1]	5	2	–	–	–	–	–	–	–	–		*131*
	IX,[IX]	CF,D2	IX(L)←[IX], IX(H)←[IX+1]	5	2	–	–	–	–	–	–	–	–		*132*
	IX,[IY]	CF,DA	IX(L)←[IY], IX(H)←[IY+1]	5	2	–	–	–	–	–	–	–	–		*132*
	IX,[SP+dd]	CF,72,dd	IX(L)←[SP+dd], IX(H)←[SP+dd+1]	6	3	–	–	–	–	–	–	–	–		*133*
LD	IY,BA	CF,EC	IY←BA	2	2	–	–	–	–	–	–	–	–		*124*
	IY,HL	CF,ED	IY←HL	2	2	–	–	–	–	–	–	–	–		*124*
	IY,IX	CF,EE	IY←IX	2	2	–	–	–	–	–	–	–	–		*124*
	IY,IY	CF,EF	IY←IY	2	2	–	–	–	–	–	–	–	–		*124*
	IY,SP	CF,FE	IY←SP	2	2	–	–	–	–	–	–	–	–		*125*
	IY,#mmnn	C7,nn,mm	IY←mmnn	3	3	–	–	–	–	–	–	–	–		*129*
	IY,[hh*ll*]	BB,*ll*,hh	IY(L)←[hh*ll*], IY(H)←[hh*ll*+1]	5	3	–	–	–	–	–	–	–	–		*130*
	IY,[HL]	CF,C3	IY(L)←[HL], IY(H)←[HL+1]	5	2	–	–	–	–	–	–	–	–		*131*
	IY,[IX]	CF,D3	IY(L)←[IX], IY(H)←[IX+1]	5	2	–	–	–	–	–	–	–	–		*132*
	IY,[IY]	CF,DB	IY(L)←[IY], IY(H)←[IY+1]	5	2	–	–	–	–	–	–	–	–		*132*
	IY,[SP+dd]	CF,73,dd	IY(L)←[SP+dd], IY(H)←[SP+dd+1]	6	3	–	–	–	–	–	–	–	–		*133*

表4.3.3.1(e) 機能別命令一覧表 (16ビット転送命令 2/2)

16ビット転送命令

ニーモニック		コード	オペレーション	サイクル	バイト	SC								備 考	ページ
						I1	I0	U	D	N	V	C	Z		
LD	SP,BA	CF,F0	SP←BA	2	2	–	–	–	–	–	–	–	–		*126*
	SP,[hh*ll*]	CF,78,*ll*,hh	SP(L)←[hh*ll*], SP(H)←[hh*ll*+1]	6	4	–	–	–	–	–	–	–	–		*131*
	SP,HL	CF,F1	SP←HL	2	2	–	–	–	–	–	–	–	–		*126*
	SP,IX	CF,F2	SP←IX	2	2	–	–	–	–	–	–	–	–		*126*
	SP,IY	CF,F3	SP←IY	2	2	–	–	–	–	–	–	–	–		*126*
	SP,#mmnn	CF,6E,nn,mm	SP←mmnn	4	4	–	–	–	–	–	–	–	–		*130*
LD	[hh*ll*],BA	BC,*ll*,hh	[hh*ll*]←A, [hh*ll*+1]←B	5	3	–	–	–	–	–	–	–	–		*126*
	[hh*ll*],HL	BD,*ll*,hh	[hh*ll*]←L, [hh*ll*+1]←H	5	3	–	–	–	–	–	–	–	–		*126*
	[hh*ll*],IX	BE,*ll*,hh	[hh*ll*]←IX(L), [hh*ll*+1]←IX(H)	5	3	–	–	–	–	–	–	–	–		*126*
	[hh*ll*],IY	BF,*ll*,hh	[hh*ll*]←IY(L), [hh*ll*+1]←IY(H)	5	3	–	–	–	–	–	–	–	–		*126*
	[hh*ll*],SP	CF,7C,*ll*,hh	[hh*ll*]←SP(L), [hh*ll*+1]←SP(H)	6	4	–	–	–	–	–	–	–	–		*127*
LD	[HL],BA	CF,C4	[HL]←A, [HL+1]←B	5	2	–	–	–	–	–	–	–	–		*127*
	[HL],HL	CF,C5	[HL]←L, [HL+1]←H	5	2	–	–	–	–	–	–	–	–		*127*
	[HL],IX	CF,C6	[HL]←IX(L), [HL+1]←IX(H)	5	2	–	–	–	–	–	–	–	–		*127*
	[HL],IY	CF,C7	[HL]←IY(L), [HL+1]←IY(H)	5	2	–	–	–	–	–	–	–	–		*127*
LD	[IX],BA	CF,D4	[IX]←A, [IX+1]←B	5	2	–	–	–	–	–	–	–	–		*128*
	[IX],HL	CF,D5	[IX]←L, [IX+1]←H	5	2	–	–	–	–	–	–	–	–		*128*
	[IX],IX	CF,D6	[IX]←IX(L), [IX+1]←IX(H)	5	2	–	–	–	–	–	–	–	–		*128*
	[IX],IY	CF,D7	[IX]←IY(L), [IX+1]←IY(H)	5	2	–	–	–	–	–	–	–	–		*128*
LD	[IY],BA	CF,DC	[IY]←A, [IY+1]←B	5	2	–	–	–	–	–	–	–	–		*128*
	[IY],HL	CF,DD	[IY]←L, [IY+1]←H	5	2	–	–	–	–	–	–	–	–		*128*
	[IY],IX	CF,DE	[IY]←IX(L), [IY+1]←IX(H)	5	2	–	–	–	–	–	–	–	–		*128*
	[IY],IY	CF,DF	[IY]←IY(L), [IY+1]←IY(H)	5	2	–	–	–	–	–	–	–	–		*128*
LD	[SP+dd],BA	CF,74,dd	[SP+dd]←A, [SP+dd+1]←B	6	3	–	–	–	–	–	–	–	–		*129*
	[SP+dd],HL	CF,75,dd	[SP+dd]←L, [SP+dd+1]←H	6	3	–	–	–	–	–	–	–	–		*129*
	[SP+dd],IX	CF,76,dd	[SP+dd]←IX(L), [SP+dd+1]←IX(H)	6	3	–	–	–	–	–	–	–	–		*129*
	[SP+dd],IY	CF,77,dd	[SP+dd]←IY(L), [SP+dd+1]←IY(H)	6	3	–	–	–	–	–	–	–	–		*129*
EX	BA,HL	C8	BA↔HL	3	1	–	–	–	–	–	–	–	–		*93*
	BA,IX	C9	BA↔IX	3	1	–	–	–	–	–	–	–	–		*93*
	BA,IY	CA	BA↔IY	3	1	–	–	–	–	–	–	–	–		*93*
	BA,SP	CB	BA↔SP	3	1	–	–	–	–	–	–	–	–		*93*

表4.3.3.1(f) 機能別命令一覧表 (8ビット算術論理演算命令 1/4)

8ビット算術論理演算命令

ニーモニック		コード	オペレーション	サイクル	バイト	SC I1	SC I0	SC U	SC D	SC N	SC V	SC C	SC Z	備考	ページ
ADD	A,A	00	A←A+A	2	1	–	–	★	★	↕	↕	↕	↕		*60*
	A,B	01	A←A+B	2	1	–	–	★	★	↕	↕	↕	↕		*60*
	A,#nn	02,nn	A←A+nn	2	2	–	–	★	★	↕	↕	↕	↕		*60*
	A,[BR:*ll*]	04,*ll*	A←A+[BR:*ll*]	3	2	–	–	★	★	↕	↕	↕	↕		*61*
	A,[hh*ll*]	05,*ll*,hh	A←A+[hh*ll*]	4	3	–	–	★	★	↕	↕	↕	↕		*61*
	A,[HL]	03	A←A+[HL]	2	1	–	–	★	★	↕	↕	↕	↕		*61*
	A,[IX]	06	A←A+[IX]	2	1	–	–	★	★	↕	↕	↕	↕		*62*
	A,[IY]	07	A←A+[IY]	2	1	–	–	★	★	↕	↕	↕	↕		*62*
	A,[IX+dd]	CE,00,dd	A←A+[IX+dd]	4	3	–	–	★	★	↕	↕	↕	↕		*62*
	A,[IY+dd]	CE,01,dd	A←A+[IY+dd]	4	3	–	–	★	★	↕	↕	↕	↕		*62*
	A,[IX+L]	CE,02	A←A+[IX+L]	4	2	–	–	★	★	↕	↕	↕	↕		*63*
	A,[IY+L]	CE,03	A←A+[IY+L]	4	2	–	–	★	★	↕	↕	↕	↕		*63*
	[HL],A	CE,04	[HL]←[HL]+A	4	2	–	–	★	★	↕	↕	↕	↕		*63*
	[HL],#nn	CE,05,nn	[HL]←[HL]+nn	5	3	–	–	★	★	↕	↕	↕	↕		*63*
	[HL],[IX]	CE,06	[HL]←[HL]+[IX]	5	2	–	–	★	★	↕	↕	↕	↕		*64*
	[HL],[IY]	CE,07	[HL]←[HL]+[IY]	5	2	–	–	★	★	↕	↕	↕	↕		*64*
ADC	A,A	08	A←A+A+C	2	1	–	–	★	★	↕	↕	↕	↕		*55*
	A,B	09	A←A+B+C	2	1	–	–	★	★	↕	↕	↕	↕		*55*
	A,#nn	0A,nn	A←A+nn+C	2	2	–	–	★	★	↕	↕	↕	↕		*55*
	A,[BR:*ll*]	0C,*ll*	A←A+[BR:*ll*]+C	3	2	–	–	★	★	↕	↕	↕	↕		*55*
	A,[hh*ll*]	0D,*ll*,hh	A←A+[hh*ll*]+C	4	3	–	–	★	★	↕	↕	↕	↕		*56*
	A,[HL]	0B	A←A+[HL]+C	2	1	–	–	★	★	↕	↕	↕	↕		*56*
	A,[IX]	0E	A←A+[IX]+C	2	1	–	–	★	★	↕	↕	↕	↕		*56*
	A,[IY]	0F	A←A+[IY]+C	2	1	–	–	★	★	↕	↕	↕	↕		*56*
	A,[IX+dd]	CE,08,dd	A←A+[IX+dd]+C	4	3	–	–	★	★	↕	↕	↕	↕		*57*
	A,[IY+dd]	CE,09,dd	A←A+[IY+dd]+C	4	3	–	–	★	★	↕	↕	↕	↕		*57*
	A,[IX+L]	CE,0A	A←A+[IX+L]+C	4	2	–	–	★	★	↕	↕	↕	↕		*57*
	A,[IY+L]	CE,0B	A←A+[IY+L]+C	4	2	–	–	★	★	↕	↕	↕	↕		*57*
	[HL],A	CE,0C	[HL]←[HL]+A+C	4	2	–	–	★	★	↕	↕	↕	↕		*58*
	[HL],#nn	CE,0D,nn	[HL]←[HL]+nn+C	5	3	–	–	★	★	↕	↕	↕	↕		*58*
	[HL],[IX]	CE,0E	[HL]←[HL]+[IX]+C	5	2	–	–	★	★	↕	↕	↕	↕		*58*
	[HL],[IY]	CE,0F	[HL]←[HL]+[IY]+C	5	2	–	–	★	★	↕	↕	↕	↕		*58*
SUB	A,A	10	A←A-A	2	1	–	–	★	★	↕	↕	↕	↕		*165*
	A,B	11	A←A-B	2	1	–	–	★	★	↕	↕	↕	↕		*165*
	A,#nn	12,nn	A←A-nn	2	2	–	–	★	★	↕	↕	↕	↕		*165*
	A,[BR:*ll*]	14,*ll*	A←A-[BR:*ll*]	3	2	–	–	★	★	↕	↕	↕	↕		*166*
	A,[hh*ll*]	15,*ll*,hh	A←A-[hh*ll*]	4	3	–	–	★	★	↕	↕	↕	↕		*166*
	A,[HL]	13	A←A-[HL]	2	1	–	–	★	★	↕	↕	↕	↕		*166*
	A,[IX]	16	A←A-[IX]	2	1	–	–	★	★	↕	↕	↕	↕		*167*
	A,[IY]	17	A←A-[IY]	2	1	–	–	★	★	↕	↕	↕	↕		*167*
	A,[IX+dd]	CE,10,dd	A←A-[IX+dd]	4	3	–	–	★	★	↕	↕	↕	↕		*167*
	A,[IY+dd]	CE,11,dd	A←A-[IY+dd]	4	3	–	–	★	★	↕	↕	↕	↕		*167*
	A,[IX+L]	CE,12	A←A-[IX+L]	4	2	–	–	★	★	↕	↕	↕	↕		*168*
	A,[IY+L]	CE,13	A←A-[IY+L]	4	2	–	–	★	★	↕	↕	↕	↕		*168*
	[HL],A	CE,14	[HL]←[HL]-A	4	2	–	–	★	★	↕	↕	↕	↕		*168*
	[HL],#nn	CE,15,nn	[HL]←[HL]-nn	5	3	–	–	★	★	↕	↕	↕	↕		*168*
	[HL],[IX]	CE,16	[HL]←[HL]-[IX]	5	2	–	–	★	★	↕	↕	↕	↕		*169*
	[HL],[IY]	CE,17	[HL]←[HL]-[IY]	5	2	–	–	★	★	↕	↕	↕	↕		*169*

表4.3.3.1(g) 機能別命令一覧表 (8ビット算術論理演算命令 2/4)

8ビット算術論理演算命令

ニーモニック		コード	オペレーション	サイクル	バイト	SC								備考	ページ
						I1	I0	U	D	N	V	C	Z		
SBC	A,A	18	A←A-A-C	2	1	–	–	★	★	↕	↕	↕	↕		*153*
	A,B	19	A←A-B-C	2	1	–	–	★	★	↕	↕	↕	↕		*153*
	A,#nn	1A,nn	A←A-nn-C	2	2	–	–	★	★	↕	↕	↕	↕		*153*
	A,[BR:*ll*]	1C,*ll*	A←A-[BR:*ll*]-C	3	2	–	–	★	★	↕	↕	↕	↕		*153*
	A,[hh*ll*]	1D,*ll*,hh	A←A-[hh*ll*]-C	4	3	–	–	★	★	↕	↕	↕	↕		*154*
	A,[HL]	1B	A←A-[HL]-C	2	1	–	–	★	★	↕	↕	↕	↕		*154*
	A,[IX]	1E	A←A-[IX]-C	2	1	–	–	★	★	↕	↕	↕	↕		*154*
	A,[IY]	1F	A←A-[IY]-C	2	1	–	–	★	★	↕	↕	↕	↕		*154*
	A,[IX+dd]	CE,18,dd	A←A-[IX+dd]-C	4	3	–	–	★	★	↕	↕	↕	↕		*155*
	A,[IY+dd]	CE,19,dd	A←A-[IY+dd]-C	4	3	–	–	★	★	↕	↕	↕	↕		*155*
	A,[IX+L]	CE,1A	A←A-[IX+L]-C	4	2	–	–	★	★	↕	↕	↕	↕		*155*
	A,[IY+L]	CE,1B	A←A-[IY+L]-C	4	2	–	–	★	★	↕	↕	↕	↕		*155*
	[HL],A	CE,1C	[HL]←[HL]-A-C	4	2	–	–	★	★	↕	↕	↕	↕		*156*
	[HL],#nn	CE,1D,nn	[HL]←[HL]-nn-C	5	3	–	–	★	★	↕	↕	↕	↕		*156*
	[HL],[IX]	CE,1E	[HL]←[HL]-[IX]-C	5	2	–	–	★	★	↕	↕	↕	↕		*156*
	[HL],[IY]	CE,1F	[HL]←[HL]-[IY]-C	5	2	–	–	★	★	↕	↕	↕	↕		*156*
AND	A,A	20	A←A∧A	2	1	–	–	–	–	↕	–	–	↕		*67*
	A,B	21	A←A∧B	2	1	–	–	–	–	↕	–	–	↕		*67*
	A,#nn	22,nn	A←A∧nn	2	2	–	–	–	–	↕	–	–	↕		*68*
	A,[BR:*ll*]	24,*ll*	A←A∧[BR:*ll*]	3	2	–	–	–	–	↕	–	–	↕		*68*
	A,[hh*ll*]	25,*ll*,hh	A←A∧[hh*ll*]	4	3	–	–	–	–	↕	–	–	↕		*68*
	A,[HL]	23	A←A∧[HL]	2	1	–	–	–	–	↕	–	–	↕		*69*
	A,[IX]	26	A←A∧[IX]	2	1	–	–	–	–	↕	–	–	↕		*69*
	A,[IY]	27	A←A∧[IY]	2	1	–	–	–	–	↕	–	–	↕		*69*
	A,[IX+dd]	CE,20,dd	A←A∧[IX+dd]	4	3	–	–	–	–	↕	–	–	↕		*69*
	A,[IY+dd]	CE,21,dd	A←A∧[IY+dd]	4	3	–	–	–	–	↕	–	–	↕		*69*
	A,[IX+L]	CE,22	A←A∧[IX+L]	4	2	–	–	–	–	↕	–	–	↕		*70*
	A,[IY+L]	CE,23	A←A∧[IY+L]	4	2	–	–	–	–	↕	–	–	↕		*70*
	B,#nn	CE,B0,nn	B←B∧nn	3	3	–	–	–	–	↕	–	–	↕		*70*
	L,#nn	CE,B1,nn	L←L∧nn	3	3	–	–	–	–	↕	–	–	↕		*70*
	H,#nn	CE,B2,nn	H←H∧nn	3	3	–	–	–	–	↕	–	–	↕		*71*
	SC,#nn	9C,nn	SC←SC∧nn	3	2	↓	↓	↓	↓	↓	↓	↓	↓		*71*
	[BR:*ll*],#nn	D8,*ll*,nn	[BR:*ll*]←[BR:*ll*]∧nn	5	3	–	–	–	–	↕	–	–	↕		*71*
	[HL],A	CE,24	[HL]←[HL]∧A	4	2	–	–	–	–	↕	–	–	↕		*72*
	[HL],#nn	CE,25,nn	[HL]←[HL]∧nn	5	3	–	–	–	–	↕	–	–	↕		*72*
	[HL],[IX]	CE,26	[HL]←[HL]∧[IX]	5	2	–	–	–	–	↕	–	–	↕		*72*
	[HL],[IY]	CE,27	[HL]←[HL]∧[IY]	5	2	–	–	–	–	↕	–	–	↕		*72*
OR	A,A	28	A←A∨A	2	1	–	–	–	–	↕	–	–	↕		*135*
	A,B	29	A←A∨B	2	1	–	–	–	–	↕	–	–	↕		*135*
	A,#nn	2A,nn	A←A∨nn	2	2	–	–	–	–	↕	–	–	↕		*135*
	A,[BR:*ll*]	2C,*ll*	A←A∨[BR:*ll*]	3	2	–	–	–	–	↕	–	–	↕		*136*
	A,[hh*ll*]	2D,*ll*,hh	A←A∨[hh*ll*]	4	3	–	–	–	–	↕	–	–	↕		*136*
	A,[HL]	2B	A←A∨[HL]	2	1	–	–	–	–	↕	–	–	↕		*136*
	A,[IX]	2E	A←A∨[IX]	2	1	–	–	–	–	↕	–	–	↕		*137*
	A,[IY]	2F	A←A∨[IY]	2	1	–	–	–	–	↕	–	–	↕		*137*
	A,[IX+dd]	CE,28,dd	A←A∨[IX+dd]	4	3	–	–	–	–	↕	–	–	↕		*137*
	A,[IY+dd]	CE,29,dd	A←A∨[IY+dd]	4	3	–	–	–	–	↕	–	–	↕		*137*
	A,[IX+L]	CE,2A	A←A∨[IX+L]	4	2	–	–	–	–	↕	–	–	↕		*138*

表4.3.3.1(h) 機能別命令一覧表 (8ビット算術論理演算命令 3/4)

8ビット算術論理演算命令

ニーモニック		コード	オペレーション	サイクル	バイト	SC								備 考	ページ
						I1	I0	U	D	N	V	C	Z		
OR	A,[IY+L]	CE,2B	A←A∨[IY+L]	4	2	–	–	–	–	↕	–	–	↕		*138*
	B,#nn	CE,B4,nn	B←B∨nn	3	3	–	–	–	–	↕	–	–	↕		*138*
	L,#nn	CE,B5,nn	L←L∨nn	3	3	–	–	–	–	↕	–	–	↕		*138*
	H,#nn	CE,B6,nn	H←H∨nn	3	3	–	–	–	–	↕	–	–	↕		*139*
	SC,#nn	9D,nn	SC←SC∨nn	3	2	↑	↑	↑	↑	↑	↑	↑	↑		*139*
	[BR:*ll*],#nn	D9,*ll*,nn	[BR:*ll*]←[BR:*ll*]∨nn	5	3	–	–	–	–	↕	–	–	↕		*139*
	[HL],A	CE,2C	[HL]←[HL]∨A	4	2	–	–	–	–	↕	–	–	↕		*140*
	[HL],#nn	CE,2D,nn	[HL]←[HL]∨nn	5	3	–	–	–	–	↕	–	–	↕		*140*
	[HL],[IX]	CE,2E	[HL]←[HL]∨[IX]	5	2	–	–	–	–	↕	–	–	↕		*140*
	[HL],[IY]	CE,2F	[HL]←[HL]∨[IY]	5	2	–	–	–	–	↕	–	–	↕		*140*
XOR	A,A	38	A←A∀A	2	1	–	–	–	–	↕	–	–	↕		*173*
	A,B	39	A←A∀B	2	1	–	–	–	–	↕	–	–	↕		*173*
	A,#nn	3A,nn	A←A∀nn	2	2	–	–	–	–	↕	–	–	↕		*174*
	A,[BR:*ll*]	3C,*ll*	A←A∀[BR:*ll*]	3	2	–	–	–	–	↕	–	–	↕		*174*
	A,[hh*ll*]	3D,*ll*,hh	A←A∀[hh*ll*]	4	3	–	–	–	–	↕	–	–	↕		*174*
	A,[HL]	3B	A←A∀[HL]	2	1	–	–	–	–	↕	–	–	↕		*175*
	A,[IX]	3E	A←A∀[IX]	2	1	–	–	–	–	↕	–	–	↕		*175*
	A,[IY]	3F	A←A∀[IY]	2	1	–	–	–	–	↕	–	–	↕		*175*
	A,[IX+dd]	CE,38,dd	A←A∀[IX+dd]	4	3	–	–	–	–	↕	–	–	↕		*175*
	A,[IY+dd]	CE,39,dd	A←A∀[IY+dd]	4	3	–	–	–	–	↕	–	–	↕		*175*
	A,[IX+L]	CE,3A	A←A∀[IX+L]	4	2	–	–	–	–	↕	–	–	↕		*176*
	A,[IY+L]	CE,3B	A←A∀[IY+L]	4	2	–	–	–	–	↕	–	–	↕		*176*
	B,#nn	CE,B8,nn	B←B∀nn	3	3	–	–	–	–	↕	–	–	↕		*176*
	L,#nn	CE,B9,nn	L←L∀nn	3	3	–	–	–	–	↕	–	–	↕		*176*
	H,#nn	CE,BA,nn	H←H∀nn	3	3	–	–	–	–	↕	–	–	↕		*177*
	SC,#nn	9E,nn	SC←SC∀nn	3	2	↕	↕	↕	↕	↕	↕	↕	↕		*177*
	[BR:*ll*],#nn	DA,*ll*,nn	[BR:*ll*]←[BR:*ll*]∀nn	5	3	–	–	–	–	↕	–	–	↕		*177*
	[HL],A	CE,3C	[HL]←[HL]∀A	4	2	–	–	–	–	↕	–	–	↕		*178*
	[HL],#nn	CE,3D,nn	[HL]←[HL]∀nn	5	3	–	–	–	–	↕	–	–	↕		*178*
	[HL],[IX]	CE,3E	[HL]←[HL]∀[IX]	5	2	–	–	–	–	↕	–	–	↕		*178*
	[HL],[IY]	CE,3F	[HL]←[HL]∀[IY]	5	2	–	–	–	–	↕	–	–	↕		*178*
CP	A,A	30	A-A	2	1	–	–	–	–	↕	↕	↕	↕		*80*
	A,B	31	A-B	2	1	–	–	–	–	↕	↕	↕	↕		*80*
	A,#nn	32,nn	A-nn	2	2	–	–	–	–	↕	↕	↕	↕		*80*
	A,[BR:*ll*]	34,*ll*	A-[BR:*ll*]	3	2	–	–	–	–	↕	↕	↕	↕		*80*
	A,[hh*ll*]	35,*ll*,hh	A-[hh*ll*]	4	3	–	–	–	–	↕	↕	↕	↕		*81*
	A,[HL]	33	A-[HL]	2	1	–	–	–	–	↕	↕	↕	↕		*81*
	A,[IX]	36	A-[IX]	2	1	–	–	–	–	↕	↕	↕	↕		*81*
	A,[IY]	37	A-[IY]	2	1	–	–	–	–	↕	↕	↕	↕		*81*
	A,[IX+dd]	CE,30,dd	A-[IX+dd]	4	3	–	–	–	–	↕	↕	↕	↕		*82*
	A,[IY+dd]	CE,31,dd	A-[IY+dd]	4	3	–	–	–	–	↕	↕	↕	↕		*82*
	A,[IX+L]	CE,32	A-[IX+L]	4	2	–	–	–	–	↕	↕	↕	↕		*82*
	A,[IY+L]	CE,33	A-[IY+L]	4	2	–	–	–	–	↕	↕	↕	↕		*82*
	B,#nn	CE,BC,nn	B-nn	3	3	–	–	–	–	↕	↕	↕	↕		*83*
	L,#nn	CE,BD,nn	L-nn	3	3	–	–	–	–	↕	↕	↕	↕		*83*
	H,#nn	CE,BE,nn	H-nn	3	3	–	–	–	–	↕	↕	↕	↕		*83*
	BR,#hh	CE,BF,hh	BR-hh	3	3	–	–	–	–	↕	↕	↕	↕		*84*
	[BR:*ll*],#nn	DB,*ll*,nn	[BR:*ll*]-nn	4	3	–	–	–	–	↕	↕	↕	↕		*84*

表4.3.3.1(i) 機能別命令一覧表 (8ビット算術論理演算命令 4/4)

8ビット算術論理演算命令

ニーモニック		コード	オペレーション	サイクル	バイト	SC								備 考	ページ
						I1	I0	U	D	N	V	C	Z		
CP	[HL],A	CE,34	[HL]-A	3	2	–	–	–	–	↕	↕	↕	↕		*84*
	[HL],#nn	CE,35,nn	[HL]-nn	4	3	–	–	–	–	↕	↕	↕	↕		*85*
	[HL],[IX]	CE,36	[HL]-[IX]	4	2	–	–	–	–	↕	↕	↕	↕		*85*
	[HL],[IY]	CE,37	[HL]-[IY]	4	2	–	–	–	–	↕	↕	↕	↕		*85*
BIT	A,B	94	A∧B	2	1	–	–	–	–	↕	–	–	↕		*73*
	A,#nn	96,nn	A∧nn	2	2	–	–	–	–	↕	–	–	↕		*73*
	B,#nn	97,nn	B∧nn	2	2	–	–	–	–	↕	–	–	↕		*73*
	[BR:*ll*],#nn	DC,*ll*,nn	[BR:*ll*]∧nn	4	3	–	–	–	–	↕	–	–	↕		*74*
	[HL],#nn	95,nn	[HL]∧nn	3	2	–	–	–	–	↕	–	–	↕		*74*
INC	A	80	A←A+1	2	1	–	–	–	–	–	–	–	↕		*94*
	B	81	B←B+1	2	1	–	–	–	–	–	–	–	↕		*94*
	L	82	L←L+1	2	1	–	–	–	–	–	–	–	↕		*94*
	H	83	H←H+1	2	1	–	–	–	–	–	–	–	↕		*94*
	BR	84	BR←BR+1	2	1	–	–	–	–	–	–	–	↕		*94*
	[BR:*ll*]	85,*ll*	[BR:*ll*]←[BR:*ll*]+1	4	2	–	–	–	–	–	–	–	↕		*95*
	[HL]	86	[HL]←[HL]+1	3	1	–	–	–	–	–	–	–	↕		*95*
DEC	A	88	A←A-1	2	1	–	–	–	–	–	–	–	↕		*90*
	B	89	B←B-1	2	1	–	–	–	–	–	–	–	↕		*90*
	L	8A	L←L-1	2	1	–	–	–	–	–	–	–	↕		*90*
	H	8B	H←H-1	2	1	–	–	–	–	–	–	–	↕		*90*
	BR	8C	BR←BR-1	2	1	–	–	–	–	–	–	–	↕		*90*
	[BR:*ll*]	8D,*ll*	[BR:*ll*]←[BR:*ll*]-1	4	2	–	–	–	–	–	–	–	↕		*91*
	[HL]	8E	[HL]←[HL]-1	3	1	–	–	–	–	–	–	–	↕		*91*
CPL	A	CE,A0	A←$\overline{A}$	3	2	–	–	–	–	↕	–	–	↕		*89*
	B	CE,A1	B←$\overline{B}$	3	2	–	–	–	–	↕	–	–	↕		*89*
	[BR:*ll*]	CE,A2,*ll*	[BR:*ll*]←$\overline{[BR:ll]}$	5	3	–	–	–	–	↕	–	–	↕		*89*
	[HL]	CE,A3	[HL]←$\overline{[HL]}$	4	2	–	–	–	–	↕	–	–	↕		*90*
NEG	A	CE,A4	A←0-A	3	2	–	–	★	★	↕	↕	↕	↕		*134*
	B	CE,A5	B←0-B	3	2	–	–	★	★	↕	↕	↕	↕		*134*
	[BR:*ll*]	CE,A6,*ll*	[BR:*ll*]←0-[BR:*ll*]	5	3	–	–	★	★	↕	↕	↕	↕		*134*
	[HL]	CE,A7	[HL]←0-[HL]	4	2	–	–	★	★	↕	↕	↕	↕		*134*
MLT		CE,D8	HL←L*A	12	2	–	–	–	–	↕	0	0	↕	MODEL1/3	*133*
DIV		CE,D9	L←HL/A, H←剰余	13	2	–	–	–	–	↕	↕	0	↕	のみ	*92*

* 乗除算命令はMODEL1/3にのみ設定されています。したがって、MODEL0/2ではこれらの命令を使用することはできません。

表4.3.3.1(j) 機能別命令一覧表 (16ビット算術演算命令 1/2)

16ビット算術演算命令

ニーモニック		コード	オペレーション	サイクル	バイト	SC								備 考	ページ
						I1	I0	U	D	N	V	C	Z		
ADD	BA,BA	CF,00	BA←BA+BA	4	2	–	–	–	–	↕	↕	↕	↕		*64*
	BA,HL	CF,01	BA←BA+HL	4	2	–	–	–	–	↕	↕	↕	↕		*64*
	BA,IX	CF,02	BA←BA+IX	4	2	–	–	–	–	↕	↕	↕	↕		*64*
	BA,IY	CF,03	BA←BA+IY	4	2	–	–	–	–	↕	↕	↕	↕		*64*
	BA,#mmnn	C0,nn,mm	BA←BA+mmnn	3	3	–	–	–	–	↕	↕	↕	↕		*64*
	HL,BA	CF,20	HL←HL+BA	4	2	–	–	–	–	↕	↕	↕	↕		*65*
	HL,HL	CF,21	HL←HL+HL	4	2	–	–	–	–	↕	↕	↕	↕		*65*
	HL,IX	CF,22	HL←HL+IX	4	2	–	–	–	–	↕	↕	↕	↕		*65*
	HL,IY	CF,23	HL←HL+IY	4	2	–	–	–	–	↕	↕	↕	↕		*65*
	HL,#mmnn	C1,nn,mm	HL←HL+mmnn	3	3	–	–	–	–	↕	↕	↕	↕		*65*
	IX,BA	CF,40	IX←IX+BA	4	2	–	–	–	–	↕	↕	↕	↕		*65*
	IX,HL	CF,41	IX←IX+HL	4	2	–	–	–	–	↕	↕	↕	↕		*65*
	IX,#mmnn	C2,nn,mm	IX←IX+mmnn	3	3	–	–	–	–	↕	↕	↕	↕		*66*
	IY,BA	CF,42	IY←IY+BA	4	2	–	–	–	–	↕	↕	↕	↕		*66*
	IY,HL	CF,43	IY←IY+HL	4	2	–	–	–	–	↕	↕	↕	↕		*66*
	IY,#mmnn	C3,nn,mm	IY←IY+mmnn	3	3	–	–	–	–	↕	↕	↕	↕		*66*
	SP,BA	CF,44	SP←SP+BA	4	2	–	–	–	–	↕	↕	↕	↕		*67*
	SP,HL	CF,45	SP←SP+HL	4	2	–	–	–	–	↕	↕	↕	↕		*67*
	SP,#mmnn	CF,68,nn,mm	SP←SP+mmnn	4	4	–	–	–	–	↕	↕	↕	↕		*67*
ADC	BA,BA	CF,04	BA←BA+BA+C	4	2	–	–	–	–	↕	↕	↕	↕		*59*
	BA,HL	CF,05	BA←BA+HL+C	4	2	–	–	–	–	↕	↕	↕	↕		*59*
	BA,IX	CF,06	BA←BA+IX+C	4	2	–	–	–	–	↕	↕	↕	↕		*59*
	BA,IY	CF,07	BA←BA+IY+C	4	2	–	–	–	–	↕	↕	↕	↕		*59*
	BA,#mmnn	CF,60,nn,mm	BA←BA+mmnn+C	4	4	–	–	–	–	↕	↕	↕	↕		*59*
	HL,BA	CF,24	HL←HL+BA+C	4	2	–	–	–	–	↕	↕	↕	↕		*59*
	HL,HL	CF,25	HL←HL+HL+C	4	2	–	–	–	–	↕	↕	↕	↕		*59*
	HL,IX	CF,26	HL←HL+IX+C	4	2	–	–	–	–	↕	↕	↕	↕		*59*
	HL,IY	CF,27	HL←HL+IY+C	4	2	–	–	–	–	↕	↕	↕	↕		*59*
	HL,#mmnn	CF,61,nn,mm	HL←HL+mmnn+C	4	4	–	–	–	–	↕	↕	↕	↕		*60*
SUB	BA,BA	CF,08	BA←BA-BA	4	2	–	–	–	–	↕	↕	↕	↕		*169*
	BA,HL	CF,09	BA←BA-HL	4	2	–	–	–	–	↕	↕	↕	↕		*169*
	BA,IX	CF,0A	BA←BA-IX	4	2	–	–	–	–	↕	↕	↕	↕		*169*
	BA,IY	CF,0B	BA←BA-IY	4	2	–	–	–	–	↕	↕	↕	↕		*169*
	BA,#mmnn	D0,nn,mm	BA←BA-mmnn	3	3	–	–	–	–	↕	↕	↕	↕		*169*
	HL,BA	CF,28	HL←HL-BA	4	2	–	–	–	–	↕	↕	↕	↕		*170*
	HL,HL	CF,29	HL←HL-HL	4	2	–	–	–	–	↕	↕	↕	↕		*170*
	HL,IX	CF,2A	HL←HL-IX	4	2	–	–	–	–	↕	↕	↕	↕		*170*
	HL,IY	CF,2B	HL←HL-IY	4	2	–	–	–	–	↕	↕	↕	↕		*170*
	HL,#mmnn	D1,nn,mm	HL←HL-mmnn	3	3	–	–	–	–	↕	↕	↕	↕		*170*
	IX,BA	CF,48	IX←IX-BA	4	2	–	–	–	–	↕	↕	↕	↕		*170*
	IX,HL	CF,49	IX←IX-HL	4	2	–	–	–	–	↕	↕	↕	↕		*170*
	IX,#mmnn	D2,nn,mm	IX←IX-mmnn	3	3	–	–	–	–	↕	↕	↕	↕		*171*
	IY,BA	CF,4A	IY←IY-BA	4	2	–	–	–	–	↕	↕	↕	↕		*171*
	IY,HL	CF,4B	IY←IY-HL	4	2	–	–	–	–	↕	↕	↕	↕		*171*
	IY,#mmnn	D3,nn,mm	IY←IY-mmnn	3	3	–	–	–	–	↕	↕	↕	↕		*171*
	SP,BA	CF,4C	SP←SP-BA	4	2	–	–	–	–	↕	↕	↕	↕		*172*
	SP,HL	CF,4D	SP←SP-HL	4	2	–	–	–	–	↕	↕	↕	↕		*172*
	SP,#mmnn	CF,6A,nn,mm	SP←SP-mmnn	4	4	–	–	–	–	↕	↕	↕	↕		*172*

表4.3.3.1(k) 機能別命令一覧表 (16ビット算術演算命令 2/2)

16ビット算術演算命令

ニーモニック		コード	オペレーション	サイクル	バイト	SC I1	SC I0	SC U	SC D	SC N	SC V	SC C	SC Z	備考	ページ
SBC	BA,BA	CF,0C	BA←BA-BA-C	4	2	–	–	–	–	↕	↕	↕	↕		*157*
	BA,HL	CF,0D	BA←BA-HL-C	4	2	–	–	–	–	↕	↕	↕	↕		*157*
	BA,IX	CF,0E	BA←BA-IX-C	4	2	–	–	–	–	↕	↕	↕	↕		*157*
	BA,IY	CF,0F	BA←BA-IY-C	4	2	–	–	–	–	↕	↕	↕	↕		*157*
	BA,#mmnn	CF,62,nn,mm	BA←BA-mmnn-C	4	4	–	–	–	–	↕	↕	↕	↕		*157*
	HL,BA	CF,2C	HL←HL-BA-C	4	2	–	–	–	–	↕	↕	↕	↕		*157*
	HL,HL	CF,2D	HL←HL-HL-C	4	2	–	–	–	–	↕	↕	↕	↕		*157*
	HL,IX	CF,2E	HL←HL-IX-C	4	2	–	–	–	–	↕	↕	↕	↕		*157*
	HL,IY	CF,2F	HL←HL-IY-C	4	2	–	–	–	–	↕	↕	↕	↕		*157*
	HL,#mmnn	CF,63,nn,mm	HL←HL-mmnn-C	4	4	–	–	–	–	↕	↕	↕	↕		*158*
CP	BA,BA	CF,18	BA-BA	4	2	–	–	–	–	↕	↕	↕	↕		*86*
	BA,HL	CF,19	BA-HL	4	2	–	–	–	–	↕	↕	↕	↕		*86*
	BA,IX	CF,1A	BA-IX	4	2	–	–	–	–	↕	↕	↕	↕		*86*
	BA,IY	CF,1B	BA-IY	4	2	–	–	–	–	↕	↕	↕	↕		*86*
	BA,#mmnn	D4,nn,mm	BA-mmnn	3	3	–	–	–	–	↕	↕	↕	↕		*86*
	HL,BA	CF,38	HL-BA	4	2	–	–	–	–	↕	↕	↕	↕		*87*
	HL,HL	CF,39	HL-HL	4	2	–	–	–	–	↕	↕	↕	↕		*87*
	HL,IX	CF,3A	HL-IX	4	2	–	–	–	–	↕	↕	↕	↕		*87*
	HL,IY	CF,3B	HL-IY	4	2	–	–	–	–	↕	↕	↕	↕		*87*
	HL,#mmnn	D5,nn,mm	HL-mmnn	3	3	–	–	–	–	↕	↕	↕	↕		*87*
	IX,#mmnn	D6,nn,mm	IX-mmnn	3	3	–	–	–	–	↕	↕	↕	↕		*88*
	IY,#mmnn	D7,nn,mm	IY-mmnn	3	3	–	–	–	–	↕	↕	↕	↕		*88*
	SP,BA	CF,5C	SP-BA	4	2	–	–	–	–	↕	↕	↕	↕		*88*
	SP,HL	CF,5D	SP-HL	4	2	–	–	–	–	↕	↕	↕	↕		*88*
	SP,#mmnn	CF,6C,nn,mm	SP-mmnn	4	4	–	–	–	–	↕	↕	↕	↕		*89*
INC	BA	90	BA←BA+1	2	1	–	–	–	–	–	–	–	↕		*95*
	HL	91	HL←HL+1	2	1	–	–	–	–	–	–	–	↕		*95*
	IX	92	IX←IX+1	2	1	–	–	–	–	–	–	–	↕		*95*
	IY	93	IY←IY+1	2	1	–	–	–	–	–	–	–	↕		*95*
	SP	87	SP←SP+1	2	1	–	–	–	–	–	–	–	↕		*96*
DEC	BA	98	BA←BA-1	2	1	–	–	–	–	–	–	–	↕		*91*
	HL	99	HL←HL-1	2	1	–	–	–	–	–	–	–	↕		*91*
	IX	9A	IX←IX-1	2	1	–	–	–	–	–	–	–	↕		*91*
	IY	9B	IY←IY-1	2	1	–	–	–	–	–	–	–	↕		*91*
	SP	8F	SP←SP-1	2	1	–	–	–	–	–	–	–	↕		*92*

表4.3.3.1(l) 機能別命令一覧表 (演算補助命令)

演算補助命令

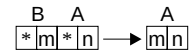
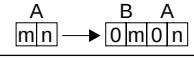
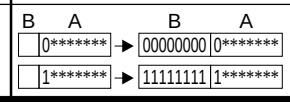
ニーモニック		コード	オペレーション	サイクル	バイト	SC I1	SC I0	SC U	SC D	SC N	SC V	SC C	SC Z	備考	ページ
PACK		DE		2	1	–	–	–	–	–	–	–	–		*141*
UPCK		DF		2	1	–	–	–	–	–	–	–	–		*173*
SEP		CE,A8		3	2	–	–	–	–	–	–	–	–		*158*

表4.3.3.1(m) 機能別命令一覧表 (ローテート/シフト 1/2)

ローテート/シフト命令

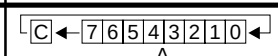
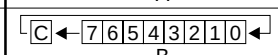
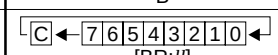
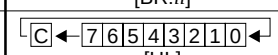
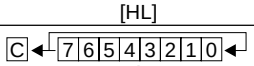
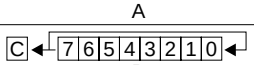
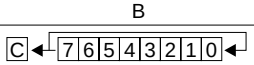
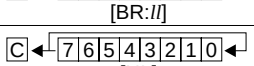
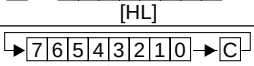
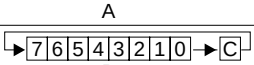
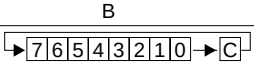
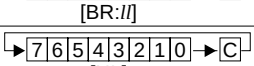
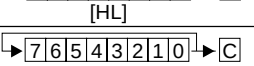
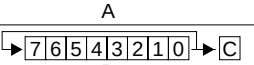
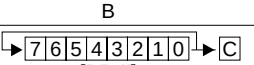
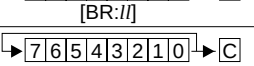
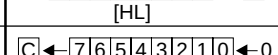
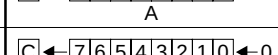
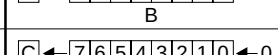
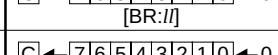
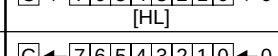
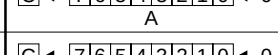
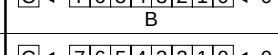
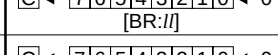
ニーモニック		コード	オペレーション	サイクル	バイト	SC I1	SC I0	SC U	SC D	SC N	SC V	SC C	SC Z	備考	ページ
RL	A	CE,90		3	2	–	–	–	–	↕	–	↕	↕		*147*
	B	CE,91		3	2	–	–	–	–	↕	–	↕	↕		*147*
	[BR:*ll*]	CE,92,*ll*		5	3	–	–	–	–	↕	–	↕	↕		*148*
	[HL]	CE,93		4	2	–	–	–	–	↕	–	↕	↕		*148*
RLC	A	CE,94		3	2	–	–	–	–	↕	–	↕	↕		*149*
	B	CE,95		3	2	–	–	–	–	↕	–	↕	↕		*149*
	[BR:*ll*]	CE,96,*ll*		5	3	–	–	–	–	↕	–	↕	↕		*149*
	[HL]	CE,97		4	2	–	–	–	–	↕	–	↕	↕		*149*
RR	A	CE,98		3	2	–	–	–	–	↕	–	↕	↕		*150*
	B	CE,99		3	2	–	–	–	–	↕	–	↕	↕		*150*
	[BR:*ll*]	CE,9A,*ll*		5	3	–	–	–	–	↕	–	↕	↕		*150*
	[HL]	CE,9B		4	2	–	–	–	–	↕	–	↕	↕		*151*
RRC	A	CE,9C		3	2	–	–	–	–	↕	–	↕	↕		*151*
	B	CE,9D		3	2	–	–	–	–	↕	–	↕	↕		*151*
	[BR:*ll*]	CE,9E,*ll*		5	3	–	–	–	–	↕	–	↕	↕		*152*
	[HL]	CE,9F		4	2	–	–	–	–	↕	–	↕	↕		*152*
SLA	A	CE,80		3	2	–	–	–	–	↕	↕	↕	↕		*159*
	B	CE,81		3	2	–	–	–	–	↕	↕	↕	↕		*159*
	[BR:*ll*]	CE,82,*ll*		5	3	–	–	–	–	↕	↕	↕	↕		*159*
	[HL]	CE,83		4	2	–	–	–	–	↕	↕	↕	↕		*160*
SLL	A	CE,84		3	2	–	–	–	–	↕	–	↕	↕		*160*
	B	CE,85		3	2	–	–	–	–	↕	–	↕	↕		*160*
	[BR:*ll*]	CE,86,*ll*		5	3	–	–	–	–	↕	–	↕	↕		*161*
	[HL]	CE,87		4	2	–	–	–	–	↕	–	↕	↕		*161*

表4.3.3.1(n) 機能別命令一覧表 (ローテート/シフト 2/2)

ローテート/シフト命令

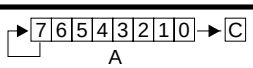
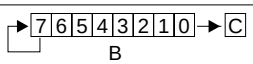
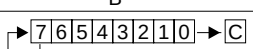
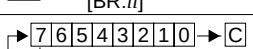
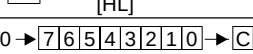
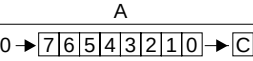
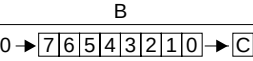
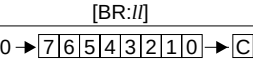
ニーモニック		コード	オペレーション	サイクル	バイト	SC I1	SC I0	SC U	SC D	SC N	SC V	SC C	SC Z	備考	ページ
SRA	A	CE,88		3	2	–	–	–	–	↕	0	↕	↕		*162*
	B	CE,89		3	2	–	–	–	–	↕	0	↕	↕		*162*
	[BR:*ll*]	CE,8A,*ll*		5	3	–	–	–	–	↕	0	↕	↕		*163*
	[HL]	CE,8B		4	2	–	–	–	–	↕	0	↕	↕		*163*
SRL	A	CE,8C		3	2	–	–	–	–	0	–	↕	↕		*164*
	B	CE,8D		3	2	–	–	–	–	0	–	↕	↕		*164*
	[BR:*ll*]	CE,8E,*ll*		5	3	–	–	–	–	0	–	↕	↕		*164*
	[HL]	CE,8F		4	2	–	–	–	–	0	–	↕	↕		*165*

表4.3.3.1(o) 機能別命令一覧表 (スタック制御命令)

スタック制御命令

ニーモニック		コード	オペレーション	サイクル	バイト	SC I1	SC I0	SC U	SC D	SC N	SC V	SC C	SC Z	備考	ページ
PUSH	A	CF,B0	[SP-1]←A, SP←SP-1	3	2	–	–	–	–	–	–	–	–		*143*
	B	CF,B1	[SP-1]←B, SP←SP-1	3	2	–	–	–	–	–	–	–	–		*143*
	L	CF,B2	[SP-1]←L, SP←SP-1	3	2	–	–	–	–	–	–	–	–		*143*
	H	CF,B3	[SP-1]←H, SP←SP-1	3	2	–	–	–	–	–	–	–	–		*143*
	BR	A4	[SP-1]←BR, SP←SP-1	3	1	–	–	–	–	–	–	–	–		*144*
	SC	A7	[SP-1]←SC, SP←SP-1	3	1	–	–	–	–	–	–	–	–		*145*
	BA	A0	[SP-1]←B, [SP-2]←A, SP←SP-2	4	1	–	–	–	–	–	–	–	–		*144*
	HL	A1	[SP-1]←H, [SP-2]←L, SP←SP-2	4	1	–	–	–	–	–	–	–	–		*144*
	IX	A2	[SP-1]←IX(H), [SP-2]←IX(L), SP←SP-2	4	1	–	–	–	–	–	–	–	–		*144*
	IY	A3	[SP-1]←IY(H), [SP-2]←IY(L), SP←SP-2	4	1	–	–	–	–	–	–	–	–		*144*
	EP	A5	[SP-1]←EP, SP←SP-1	3	1	–	–	–	–	–	–	–	–	MODEL2/3のみ	*144*
	IP	A6	[SP-1]←XP, [SP-2]←YP, SP←SP-2	4	1	–	–	–	–	–	–	–	–		*145*
PUSH	ALL	CF,B8	PUSH BA, HL, IX, IY, BR	12	2	–	–	–	–	–	–	–	–		*145*
	ALE	CF,B9	PUSH BA, HL, IX, IY, BR, EP, IP	15	2	–	–	–	–	–	–	–	–	MODEL2/3のみ	*145*
POP	A	CF,B4	A←[SP], SP←SP+1	3	2	–	–	–	–	–	–	–	–		*141*
	B	CF,B5	B←[SP], SP←SP+1	3	2	–	–	–	–	–	–	–	–		*141*
	L	CF,B6	L←[SP], SP←SP+1	3	2	–	–	–	–	–	–	–	–		*141*
	H	CF,B7	H←[SP], SP←SP+1	3	2	–	–	–	–	–	–	–	–		*141*
	BR	AC	BR←[SP], SP←SP+1	2	1	–	–	–	–	–	–	–	–		*142*
	SC	AF	SC←[SP], SP←SP+1	2	1	↕	↕	↕	↕	↕	↕	↕	↕		*142*
	BA	A8	A←[SP], B←[SP+1], SP←SP+2	3	1	–	–	–	–	–	–	–	–		*141*
	HL	A9	L←[SP], H←[SP+1], SP←SP+2	3	1	–	–	–	–	–	–	–	–		*141*
	IX	AA	IX(L)←[SP], IX(H)←[SP+1], SP←SP+2	3	1	–	–	–	–	–	–	–	–		*141*
	IY	AB	IY(L)←[SP], IY(H)←[SP+1], SP←SP+2	3	1	–	–	–	–	–	–	–	–		*141*
	EP	AD	EP←[SP], SP←SP+1	2	1	–	–	–	–	–	–	–	–	MODEL2/3のみ	*142*
	IP	AE	YP←[SP], XP←[SP+1], SP←SP+2	3	1	–	–	–	–	–	–	–	–		*142*
POP	ALL	CF,BC	POP BR, IY, IX, HL, BA	11	2	–	–	–	–	–	–	–	–		*143*
	ALE	CF,BD	POP IP, EP, BR, IY, IX, HL, BA	14	2	–	–	–	–	–	–	–	–	MODEL2/3のみ	*143*

* エクスパンドページレジスタEP/XP/YPはMODEL2/3にのみ設定されています。したがって、MODEL0/1ではこれらのレジスタをアクセスする命令は使用できません。

表4.3.3.1(p) 機能別命令一覧表 (分岐命令 1/4)

分岐命令

ニーモニック		コード	分岐条件	オペレーション	サイクル	バイト	SC I1 I0 U D N V C Z	ページ
JRS	rr	F1,rr	無条件	*MODEL0/1* PC←PC+rr+1 *MODEL2/3* PC←PC+rr+1, CB←NB	2	2	– – – – – – – –	*99*
JRS	C,rr	E4,rr	C=1	*MODEL0/1* If Condition is true, then PC←PC+rr+1 else PC←PC+2 *MODEL2/3* If Condition is true, then PC←PC+rr+1, CB←NB else PC←PC+2, NB←CB	2	2	– – – – – – – –	*99*
	NC,rr	E5,rr	C=0					
	Z,rr	E6,rr	Z=1					
	NZ,rr	E7,rr	Z=0					
JRS	LT,rr	CE,E0,rr	[N∀V]=1	*MODEL0/1* If Condition is true, then PC←PC+rr+2 else PC←PC+3 *MODEL2/3* If Condition is true, then PC←PC+rr+2, CB←NB else PC←PC+3, NB←CB	3	3	– – – – – – – –	*100*
	LE,rr	CE,E1,rr	Z∨[N∀V]=1					
	GT,rr	CE,E2,rr	Z∨[N∀V]=0					
	GE,rr	CE,E3,rr	[N∀V]=0					
	V,rr	CE,E4,rr	V=1					
	NV,rr	CE,E5,rr	V=0					
	P,rr	CE,E6,rr	N=0					
	M,rr	CE,E7,rr	N=1					
	F0,rr	CE,E8,rr	F0=1					
	F1,rr	CE,E9,rr	F1=1					
	F2,rr	CE,EA,rr	F2=1					
	F3,rr	CE,EB,rr	F3=1					
	NF0,rr	CE,EC,rr	F0=0					
	NF1,rr	CE,ED,rr	F1=0					
	NF2,rr	CE,EE,rr	F2=0					
	NF3,rr	CE,EF,rr	F3=0					
JRL	qqrr	F3,rr,qq	無条件	*MODEL0/1* PC←PC+qqrr+2 *MODEL2/3* PC←PC+qqrr+2, CB←NB	3	3	– – – – – – – –	*98*
JRL	C,qqrr	EC,rr,qq	C=1	*MODEL0/1* If Condition is true, then PC←PC+qqrr+2 else PC←PC+3 *MODEL2/3* If Condition is true, then PC←PC+qqrr+2, CB←NB else PC←PC+3, NB←CB	3	3	– – – – – – – –	*98*
	NC,qqrr	ED,rr,qq	C=0					
	Z,qqrr	EE,rr,qq	Z=1					
	NZ,qqrr	EF,rr,qq	Z=0					
DJR	NZ,rr	F5,rr	B=0	*MODEL0/1* B←B-1, If B=0, then PC←PC+rr+1 else PC←PC+2 *MODEL2/3* B←B-1, If B=0, then PC←PC+rr+1, CB←NB else PC←PC+2, NB←CB	4	2	– – – – – – – ↕	*92*

表4.3.3.1(q) 機能別命令一覧表 (分岐命令 2/4)

分岐命令

ニーモニック		コード	分岐条件	オペレーション	サイクル	バイト	SC I1 I0 U D N V C Z	ページ
JP	HL	F4	無条件	*MODEL0/1* PC←HL	2	1	– – – – – – – –	*97*
				MODEL2/3 PC←HL, CB←NB				
	[kk]	FD,kk	無条件	*MODEL0/1* $PC_{(L)}$←[00kk] $PC_{(H)}$←[00kk+1]	4	2	– – – – – – – –	*97*
				MODEL2/3 $PC_{(L)}$←[00kk] $PC_{(H)}$←[00kk+1], CB←NB				
CARS	rr	F0,rr	無条件	*MODEL0/1* [SP-1]←$PC_{(H)}$, [SP-2]←$PC_{(L)}$, SP←SP-2, PC←PC+rr+1	4	2	– – – – – – – –	*77*
				MODEL2/3 (Minimum mode) [SP-1]←$PC_{(H)}$, [SP-2]←$PC_{(L)}$, SP←SP-2, PC←PC+rr+1, CB←NB				
				MODEL2/3 (Maximum mode) [SP-1]←CB, [SP-2]←$PC_{(H)}$, [SP-3]←$PC_{(L)}$, SP←SP-3, PC←PC+rr+1, CB←NB	5			
CARS	C,rr	E0,rr	C=1	*MODEL0/1* If Condition is true then [SP-1]←$PC_{(H)}$, [SP-2]←$PC_{(L)}$, SP←SP-2, PC←PC+rr+1	4	2	– – – – – – – –	*77*
	NC,rr	E1,rr	C=0	else PC←PC+2	2			
	Z,rr	E2,rr	Z=1	*MODEL2/3 (Minimum mode)* If Condition is true then [SP-1]←$PC_{(H)}$, [SP-2]←$PC_{(L)}$, SP←SP-2, PC←PC+rr+1, CB←NB	4			
				else PC←PC+2, NB←CB	2			
	NZ,rr	E3,rr	Z=0	*MODEL2/3 (Maximum mode)* If Condition is true then [SP-1]←CB, [SP-2]←$PC_{(H)}$, [SP-3]←$PC_{(L)}$, SP←SP-3, PC←PC+rr+1, CB←NB	5			
				else PC←PC+2, NB←CB	2			
CARS	LT,rr	CE,F0,rr	[N∀V]=1	*MODEL0/1* If Condition is true then [SP-1]←$PC_{(H)}$, [SP-2]←$PC_{(L)}$, SP←SP-2, PC←PC+rr+2	5	3	– – – – – – – –	*78*
	LE,rr	CE,F1,rr	Z∨[N∀V]=1	else PC←PC+3	3			
	GT,rr	CE,F2,rr	Z∨[N∀V]=0	*MODEL2/3 (Minimum mode)* If Condition is true then [SP-1]←$PC_{(H)}$, [SP-2]←$PC_{(L)}$, SP←SP-2, PC←PC+rr+2, CB←NB	5			
	GE,rr	CE,F3,rr	[N∀V]=0	else PC←PC+3, NB←CB	3			
	V,rr	CE,F4,rr	V=1	*MODEL2/3 (Maximum mode)* If Condition is true then [SP-1]←CB, [SP-2]←$PC_{(H)}$, [SP-3]←$PC_{(L)}$, SP←SP-3, PC←PC+rr+2, CB←NB	6			
	NV,rr	CE,F5,rr	V=0	else PC←PC+3, NB←CB	3			
	P,rr	CE,F6,rr	N=0					
	M,rr	CE,F7,rr	N=1					
	F0,rr	CE,F8,rr	F0=1					
	F1,rr	CE,F9,rr	F1=1					
	F2,rr	CE,FA,rr	F2=1					
	F3,rr	CE,FB,rr	F3=1					
	NF0,rr	CE,FC,rr	F0=0					
	NF1,rr	CE,FD,rr	F1=0					
	NF2,rr	CE,FE,rr	F2=0					
	NF3,rr	CE,FF,rr	F3=0					

表4.3.3.1(r) 機能別命令一覧表 (分岐命令 3/4)

分岐命令

ニーモニック		コード	分岐条件	オペレーション	サイクル	バイト	SC I1 I0 U D N V C Z	ページ
CARL	qqrr	F2,rr,qq	無条件	*MODEL0/1* [SP-1]←PC(H), [SP-2]←PC(L), SP←SP-2, PC←PC+qqrr+2	5	3	– – – – – – – –	*76*
				MODEL2/3 (Minimum mode) [SP-1]←PC(H), [SP-2]←PC(L), SP←SP-2, PC←PC+qqrr+2, CB←NB	5			
				MODEL2/3 (Maximum mode) [SP-1]←CB, [SP-2]←PC(H), [SP-3]←PC(L), SP←SP-3, PC←PC+qqrr+2, CB←NB	6			
CARL	C,qqrr	E8,rr,qq	C=1	*MODEL0/1* If Condition is true then [SP-1]←PC(H), [SP-2]←PC(L), SP←SP-2, PC←PC+qqrr+2	5	3	– – – – – – – –	*76*
				else PC←PC+3	3			
	NC,qqrr	E9,rr,qq	C=0	*MODEL2/3 (Minimum mode)* If Condition is true then [SP-1]←PC(H), [SP-2]←PC(L), SP←SP-2, PC←PC+qqrr+2, CB←NB	5			
				else PC←PC+3, NB←CB	3			
	Z,qqrr	EA,rr,qq	Z=1	*MODEL2/3 (Maximum mode)* If Condition is true then [SP-1]←CB, [SP-2]←PC(H), [SP-3]←PC(L), SP←SP-3, PC←PC+qqrr+2, CB←NB	6			
				else PC←PC+3, NB←CB	3			
	NZ,qqrr	EB,rr,qq	Z=0					
CALL	[hh*ll*]	FB,*ll*,hh	無条件	*MODEL0/1* [SP-1]←PC(H), [SP-2]←PC(L), SP←SP-2, PC(L)←[hh*ll*], PC(H)←[hh*ll*+1]	7	3	– – – – – – – –	*75*
				MODEL2/3 (Minimum mode) [SP-1]←PC(H), [SP-2]←PC(L), SP←SP-2, PC(L)←[hh*ll*], PC(H)←[hh*ll*+1], CB←NB	8			
				MODEL2/3 (Maximum mode) [SP-1]←CB, [SP-2]←PC(H), [SP-3]←PC(L), SP←SP-3, PC(L)←[hh*ll*], PC(H)←[hh*ll*+1], CB←NB				

表4.3.3.1(s) 機能別命令一覧表 (分岐命令 4/4)

分岐命令

ニーモニック		コード	オペレーション	サイクル	バイト	SC I1 I0 U D N V C Z	備考	ページ
INT	[kk]	FC,kk	*MODEL0/1* [SP-1]←$PC_{(H)}$, [SP-2]←$PC_{(L)}$, [SP-3]←SC, SP←SP-3, $PC_{(L)}$←[00kk], $PC_{(H)}$←[00kk+1]	7	2	– – – – – – – –		*96*
			MODEL2/3 (Minimum mode) [SP-1]←$PC_{(H)}$, [SP-2]←$PC_{(L)}$, [SP-3]←SC, SP←SP-3, $PC_{(L)}$←[00kk], $PC_{(H)}$←[00kk+1], CB←NB					
			MODEL2/3 (Maximum mode) [SP-1]←CB, [SP-2]←$PC_{(H)}$, [SP-3]←$PC_{(L)}$, [SP-4]←SC, SP←SP-4, $PC_{(L)}$←[00kk], $PC_{(H)}$←[00kk+1], CB←NB	8				
RET		F8	*MODEL0/1,MODEL2/3 (Minimum mode)* $PC_{(L)}$←[SP], $PC_{(H)}$←[SP+1], SP←SP+2	3	1	– – – – – – – –		*146*
			MODEL2/3 (Maximum mode) $PC_{(L)}$←[SP], $PC_{(H)}$←[SP+1], CB←[SP+2], NB←CB, SP←SP+3	4				
RETE		F9	*MODEL0/1,MODEL2/3 (Minimum mode)* SC←[SP], $PC_{(L)}$←[SP+1], $PC_{(H)}$←[SP+2], SP←SP+3	4	1	↕ ↕ ↕ ↕ ↕ ↕ ↕ ↕		*146*
			MODEL2/3 (Maximum mode) SC←[SP], $PC_{(L)}$←[SP+1], $PC_{(H)}$←[SP+2], CB←[SP+3], NB←CB, SP←SP+4	5				
RETS		FA	*MODEL0/1,MODEL2/3 (Minimum mode)* $PC_{(L)}$←[SP], $PC_{(H)}$←[SP+1], SP←SP+2, PC←PC+2	5	1	– – – – – – – –		*147*
			MODEL2/3 (Maximum mode) $PC_{(L)}$←[SP], $PC_{(H)}$←[SP+1], CB←[SP+2], NB←CB, SP←SP+3, PC←PC+2	6				

表4.3.3.1(t) 機能別命令一覧表 (システム制御命令)

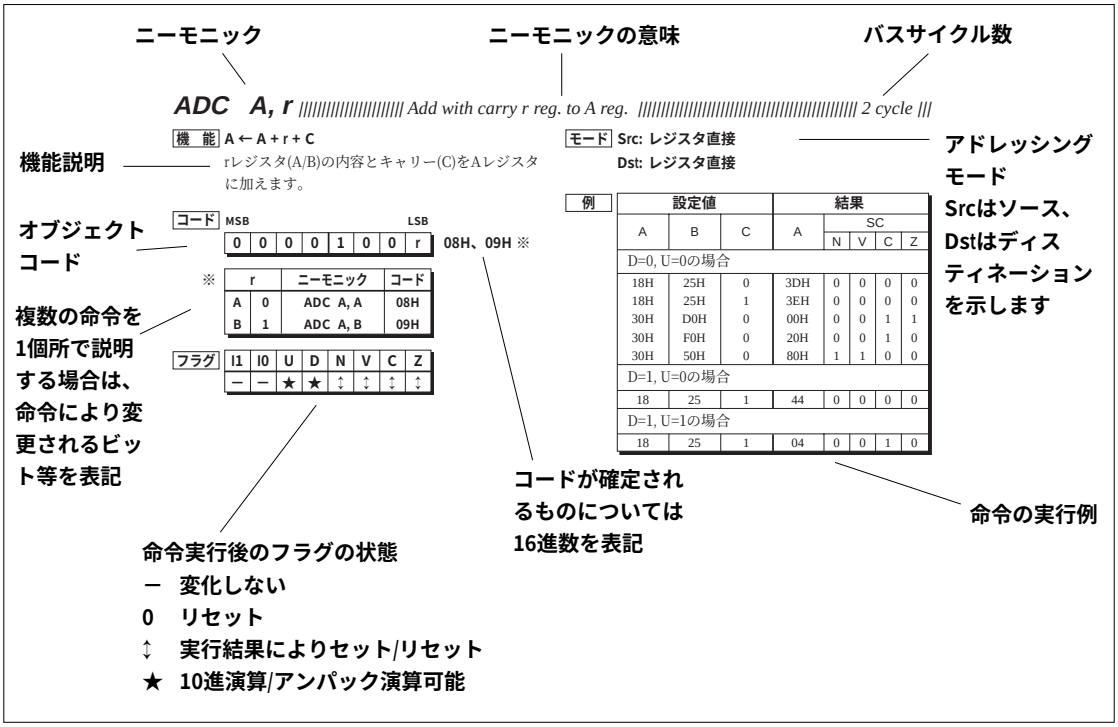
システム制御命令

ニーモニック		コード	オペレーション	サイクル	バイト	SC I1 I0 U D N V C Z	備考	ページ
NOP		FF	No Operation	2	1	– – – – – – – –		*135*
HALT		CE,AE	HALT	3	2	– – – – – – – –		*94*
SLP		CE,AF	SLEEP	3	2	– – – – – – – –		*162*

4.4 命令の詳細説明

ここでは各命令を個々に説明します。
なお、説明は以下のフォーマットにしたがって行います。

説明の見方



記号の意味は命令一覧表と同様です。☞ **"4.3.2 記号の意味"**

なお、複数のレジスタを一括して説明する際には以下の記号を使用します。

r ..... データレジスタA/B、またはA/B/L/H
ir ..... インデックスレジスタIX/IY
rp .... 16ビット(ペア)レジスタBA/ HL、または16ビットレジスタ(BA)/HL/IX/IY/(SP)
er .... ニューコードバンクレジスタNB、およびエクスパンドページレジスタEP/XP/YP

cc1 . 分岐条件C/NC/Z/NZ
cc2 . 分岐条件LT/LE/GT/GE/V/NV/P/M/F0/F1/F2/F3/NF0/NF1/NF2/NF3

マキシマムモードとミニマムモードでバスサイクル数の異なるものについてはサイクル数に(MAX)と(MIN)を付けて記載します。MINにはMODEL0/1も含まれます。

ADC A, r *Add with carry r reg. to A reg.* 2 cycle

機 能 A ← A + r + C

rレジスタ(A/B)の内容とキャリー(C)をAレジスタに加えます。

コード

MSB							LSB	
0	0	0	0	1	0	0	r	08H、09H ※

※

r		ニーモニック	コード
A	0	ADC A, A	08H
B	1	ADC A, B	09H

フラグ

I1	I0	U	D	N	V	C	Z
−	−	★	★	↕	↕	↕	↕

モード Src: レジスタ直接
Dst: レジスタ直接

例

設定値			結果				
A	B	C	A	SC			
				N	V	C	Z
D=0, U=0の場合							
18H	25H	0	3DH	0	0	0	0
18H	25H	1	3EH	0	0	0	0
30H	D0H	0	00H	0	0	1	1
30H	F0H	0	20H	0	0	1	0
30H	50H	0	80H	1	1	0	0
D=1, U=0の場合							
18	25	1	44	0	0	0	0
D=1, U=1の場合							
18	25	1	04	0	0	1	0

ADC A, #nn *Add with carry immediate data nn to A reg.* 2 cycle

機 能 A ← A + nn + C

8ビット即値データnnとキャリー(C)をAレジスタに加えます。

コード

MSB							LSB	
0	0	0	0	1	0	1	0	0AH
				n	n			nn

フラグ

I1	I0	U	D	N	V	C	Z
−	−	★	★	↕	↕	↕	↕

モード Src: 即値データ
Dst: レジスタ直接

例

設定値			結果				
A	nn	C	A	SC			
				N	V	C	Z
D=0, U=0の場合							
18H	25H	0	3DH	0	0	0	0
18H	25H	1	3EH	0	0	0	0
30H	D0H	0	00H	0	0	1	1
30H	F0H	0	20H	0	0	1	0
30H	50H	0	80H	1	1	0	0
D=1, U=0の場合							
18	25	1	44	0	0	0	0
D=1, U=1の場合							
18	25	1	04	0	0	1	0

ADC A, [BR:ll] *Add with carry location [BR:ll] to A reg.* 3 cycle

機 能 A ← A + [BR:*ll*] + C

BRレジスタの内容を上位バイト、8ビット絶対アドレス*ll*を下位バイトとしてアドレス指定されたデータメモリの内容とキャリー(C)をAレジスタに加えます。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

コード

MSB							LSB	
0	0	0	0	0	1	0	0	04H
				l	*l*			*ll*

フラグ

I1	I0	U	D	N	V	C	Z
−	−	★	★	↕	↕	↕	↕

モード Src: 8ビット絶対
Dst: レジスタ直接

例

設定値			結果				
A	[BR:*ll*]	C	A	SC			
				N	V	C	Z
D=0, U=0の場合							
18H	25H	0	3DH	0	0	0	0
18H	25H	1	3EH	0	0	0	0
30H	D0H	0	00H	0	0	1	1
30H	F0H	0	20H	0	0	1	0
30H	50H	0	80H	1	1	0	0
D=1, U=0の場合							
18	25	1	44	0	0	0	0
D=1, U=1の場合							
18	25	1	04	0	0	1	0

ADC A, [hhll] /////////////// Add with carry location [hhll] to A reg. ////////////////////////////////// 4 cycle ///

機 能 A ← A + [hh*ll*] + C

16ビット絶対アドレスhh*ll*でアドレス指定されたデータメモリの内容とキャリー(C)をAレジスタに加えます。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: 16ビット絶対
Dst: レジスタ直接

コード

MSB							LSB	
0	0	0	0	1	1	0	1	0DH
l	*l*							*ll*
h	h							hh

フラグ

I1	I0	U	D	N	V	C	Z
−	−	★	★	↕	↕	↕	↕

例

設定値			結果				
A	[hh*ll*]	C	A	SC			
				N	V	C	Z
D=0, U=0の場合							
18H	25H	0	3DH	0	0	0	0
18H	25H	1	3EH	0	0	0	0
30H	D0H	0	00H	0	0	1	1
30H	F0H	0	20H	0	0	1	0
30H	50H	0	80H	1	1	0	0
D=1, U=0の場合							
18	25	1	44	0	0	0	0
D=1, U=1の場合							
18	25	1	04	0	0	1	0

ADC A, [HL] /////////////// Add with carry location [HL] to A reg. ////////////////////////////////// 2 cycle ///

機 能 A ← A + [HL] + C

HLレジスタでアドレス指定されたデータメモリの内容とキャリー(C)をAレジスタに加えます。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: レジスタ間接
Dst: レジスタ直接

コード

MSB							LSB	
0	0	0	0	1	0	1	1	0BH

フラグ

I1	I0	U	D	N	V	C	Z
−	−	★	★	↕	↕	↕	↕

例

設定値			結果				
A	[HL]	C	A	SC			
				N	V	C	Z
D=0, U=0の場合							
18H	25H	0	3DH	0	0	0	0
18H	25H	1	3EH	0	0	0	0
30H	D0H	0	00H	0	0	1	1
30H	F0H	0	20H	0	0	1	0
30H	50H	0	80H	1	1	0	0
D=1, U=0の場合							
18	25	1	44	0	0	0	0
D=1, U=1の場合							
18	25	1	04	0	0	1	0

ADC A, [ir] ////////////////// Add with carry location [ir reg.] to A reg. ////////////////////////////// 2 cycle ///

機 能 A ← A + [ir] + C

irレジスタ(IX/IY)でアドレス指定されたデータメモリの内容とキャリー(C)をAレジスタに加えます。
XPレジスタ(IX指定時)、YPレジスタ(IY指定時)の内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: レジスタ間接
Dst: レジスタ直接

コード

MSB							LSB	
0	0	0	0	1	1	1	ir	0EH、0FH ※

※

ir		ニーモニック	コード
IX	0	ADC A, [IX]	0EH
IY	1	ADC A, [IY]	0FH

フラグ

I1	I0	U	D	N	V	C	Z
−	−	★	★	↕	↕	↕	↕

例

設定値			結果				
A	[ir]	C	A	SC			
				N	V	C	Z
D=0, U=0の場合							
18H	25H	0	3DH	0	0	0	0
18H	25H	1	3EH	0	0	0	0
30H	D0H	0	00H	0	0	1	1
30H	F0H	0	20H	0	0	1	0
30H	50H	0	80H	1	1	0	0
D=1, U=0の場合							
18	25	1	44	0	0	0	0
D=1, U=1の場合							
18	25	1	04	0	0	1	0

ADC A, [ir+dd] //////// *Add with carry location [ir reg. + dd] to A reg.* ////////////////////// *4 cycle* ///

機 能 A ← A + [ir+dd] + C

irレジスタ(IX/IY)の内容とディスプレースメントddの和でアドレス指定されたデータメモリの内容とキャリー(C)をAレジスタに加えます。
ddは符号付きデータとして扱われ、範囲は-128～127です。
XPレジスタ(IX指定時)、YPレジスタ(IY指定時)の内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: ディスプレースメント付きレジスタ間接
Dst: レジスタ直接

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	0	0	1	0	0	ir	08H、09H ※
d	d	d	d	d	d	d	d	dd

※

ir		ニーモニック	コード
IX	0	ADC A, [IX+dd]	08H
IY	1	ADC A, [IY+dd]	09H

フラグ

I1	I0	U	D	N	V	C	Z
−	−	★	★	↕	↕	↕	↕

例

設定値			結果				
A	[ir+dd]	C	A	SC			
				N	V	C	Z
D=0, U=0の場合							
18H	25H	0	3DH	0	0	0	0
18H	25H	1	3EH	0	0	0	0
30H	D0H	0	00H	0	0	1	1
30H	F0H	0	20H	0	0	1	0
30H	50H	0	80H	1	1	0	0
D=1, U=0の場合							
18	25	1	44	0	0	0	0
D=1, U=1の場合							
18	25	1	04	0	0	1	0

ADC A, [ir+L] ////////// *Add with carry location [ir reg. + L] to A reg.* ////////////////////// *4 cycle* ///

機 能 A ← A + [ir+L] + C

irレジスタ(IX/IY)の内容とLレジスタの内容の和でアドレス指定されたデータメモリの内容とキャリー(C)をAレジスタに加えます。
Lレジスタの内容は符号付きデータとして扱われ、範囲は-128～127です。
XPレジスタ(IX指定時)、YPレジスタ(IY指定時)の内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: インデックスレジスタ付きレジスタ間接
Dst: レジスタ直接

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	0	0	1	0	1	ir	0AH、0BH ※

※

ir		ニーモニック	コード
IX	0	ADC A, [IX+L]	0AH
IY	1	ADC A, [IY+L]	0BH

フラグ

I1	I0	U	D	N	V	C	Z
−	−	★	★	↕	↕	↕	↕

例

設定値			結果				
A	[ir+L]	C	A	SC			
				N	V	C	Z
D=0, U=0の場合							
18H	25H	0	3DH	0	0	0	0
18H	25H	1	3EH	0	0	0	0
30H	D0H	0	00H	0	0	1	1
30H	F0H	0	20H	0	0	1	0
30H	50H	0	80H	1	1	0	0
D=1, U=0の場合							
18	25	1	44	0	0	0	0
D=1, U=1の場合							
18	25	1	04	0	0	1	0

ADC [HL], A /////////////// Add with carry A reg. to location [HL] ////////////////////////////////// 4 cycle ///

機 能 [HL] ← [HL] + A + C

Aレジスタの内容とキャリー(C)をHLレジスタでアドレス指定されたデータメモリに加えます。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: レジスタ直接
Dst: レジスタ間接

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	0	0	1	1	0	0	0CH

フラグ

I1	I0	U	D	N	V	C	Z
−	−	★	★	↕	↕	↕	↕

例

設定値			結果				
[HL]	A	C	[HL]	SC			
				N	V	C	Z
D=0, U=0の場合							
18H	25H	0	3DH	0	0	0	0
18H	25H	1	3EH	0	0	0	0
30H	D0H	0	00H	0	0	1	1
30H	F0H	0	20H	0	0	1	0
30H	50H	0	80H	1	1	0	0
D=1, U=0の場合							
18	25	1	44	0	0	0	0
D=1, U=1の場合							
18	25	1	04	0	0	1	0

ADC [HL], #nn //////// Add with carry immediate data nn to location [HL] ////////////// 5 cycle ///

機 能 [HL] ← [HL] + nn + C

8ビット即値データnnとキャリー(C)をHLレジスタでアドレス指定されたデータメモリに加えます。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: 即値データ
Dst: レジスタ間接

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	0	0	1	1	0	1	0DH
n	n							nn

フラグ

I1	I0	U	D	N	V	C	Z
−	−	★	★	↕	↕	↕	↕

例

設定値			結果				
[HL]	nn	C	[HL]	SC			
				N	V	C	Z
D=0, U=0の場合							
18H	25H	0	3DH	0	0	0	0
18H	25H	1	3EH	0	0	0	0
30H	D0H	0	00H	0	0	1	1
30H	F0H	0	20H	0	0	1	0
30H	50H	0	80H	1	1	0	0
D=1, U=0の場合							
18	25	1	44	0	0	0	0
D=1, U=1の場合							
18	25	1	04	0	0	1	0

ADC [HL], [ir] ////////// Add with carry location [ir reg.] to location [HL] ////////////////// 5 cycle ///

機 能 [HL] ← [HL] + [ir] + C

irレジスタ(IX/IY)でアドレス指定されたデータメモリの内容とキャリー(C)をHLレジスタでアドレス指定されるデータメモリに加えます。
EPレジスタの内容がデータメモリ[HL]のページアドレス、XPレジスタ(IX指定時)、YPレジスタ(IY指定時)の内容がデータメモリ[ir]のページアドレスになります(MODEL2/3)。

モード Src: レジスタ間接
Dst: レジスタ間接

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	0	0	1	1	1	ir	0EH、0FH ※

※

ir		ニーモニック	コード
IX	0	ADC [HL], [IX]	0EH
IY	1	ADC [HL], [IY]	0FH

フラグ

I1	I0	U	D	N	V	C	Z
−	−	★	★	↕	↕	↕	↕

例

設定値			結果				
[HL]	[ir]	C	[HL]	SC			
				N	V	C	Z
D=0, U=0の場合							
18H	25H	0	3DH	0	0	0	0
18H	25H	1	3EH	0	0	0	0
30H	D0H	0	00H	0	0	1	1
30H	F0H	0	20H	0	0	1	0
30H	50H	0	80H	1	1	0	0
D=1, U=0の場合							
18	25	1	44	0	0	0	0
D=1, U=1の場合							
18	25	1	04	0	0	1	0

ADC BA, rp /////////////// Add with carry rp reg. to BA reg. ////////////////////////////////////// 4 cycle ///

機 能 BA ← BA + rp + C

rpレジスタ(BA/HL/IX/IY)の内容とキャリー(C)をBAレジスタに加えます。

モード Src: レジスタ直接
Dst: レジスタ直接

コード

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
0	0	0	0	0	1	rp		04H～07H ※

※

rp		ニーモニック	コード
BA	00	ADC BA, BA	04H
HL	01	ADC BA, HL	05H
IX	10	ADC BA, IX	06H
IY	11	ADC BA, IY	07H

フラグ

I1	I0	U	D	N	V	C	Z
−	−	−	−	↕	↕	↕	↕

例

設定値			結果				
BA	rp	C	BA	SC			
				N	V	C	Z
1380H	3546H	0	48C6H	0	0	0	0
1380H	3546H	1	48C7H	0	0	0	0
1380H	EC80H	0	0000H	0	0	1	1
5218H	4174H	0	938CH	1	1	0	0
5342H	C32AH	1	166DH	0	0	1	0

(rp≠BA)

ADC BA, #mmnn // Add with carry immediate data mmnn to BA reg. ////////////////// 4 cycle ///

機 能 BA ← BA + mmnn + C

16ビット即値データmmnnとキャリー(C)をBAレジスタに加えます。

モード Src: 即値データ
Dst: レジスタ直接

コード

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
0	1	1	0	0	0	0	0	60H
n	n							nn
m	m							mm

フラグ

I1	I0	U	D	N	V	C	Z
−	−	−	−	↕	↕	↕	↕

例

設定値			結果				
BA	mmnn	C	BA	SC			
				N	V	C	Z
1380H	3546H	0	48C6H	0	0	0	0
1380H	3546H	1	48C7H	0	0	0	0
1380H	EC80H	0	0000H	0	0	1	1
5218H	4174H	0	938CH	1	1	0	0
5342H	C32AH	1	166DH	0	0	1	0

ADC HL, rp ////////////////// Add with carry rp reg. to HL reg. ////////////////////////////////////// 4 cycle ///

機 能 HL ← HL + rp + C

rpレジスタ(BA/HL/IX/IY)の内容とキャリー(C)をHLレジスタに加えます。

モード Src: レジスタ直接
Dst: レジスタ直接

コード

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
0	0	1	0	0	1	rp		24H～27H ※

※

rp		ニーモニック	コード
BA	00	ADC HL, BA	24H
HL	01	ADC HL, HL	25H
IX	10	ADC HL, IX	26H
IY	11	ADC HL, IY	27H

フラグ

I1	I0	U	D	N	V	C	Z
−	−	−	−	↕	↕	↕	↕

例

設定値			結果				
HL	rp	C	HL	SC			
				N	V	C	Z
1380H	3546H	0	48C6H	0	0	0	0
1380H	3546H	1	48C7H	0	0	0	0
1380H	EC80H	0	0000H	0	0	1	1
5218H	4174H	0	938CH	1	1	0	0
5342H	C32AH	1	166DH	0	0	1	0

(rp≠HL)

ADC HL, #mmnn /// Add with carry immediate data mmnn to HL reg. //////////////////// 4 cycle ///

機 能 HL ← HL + mmnn + C

16ビット即値データmmnnとキャリー(C)をHLレジスタに加えます。

モード Src: 即値データ
Dst: レジスタ直接

コード

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
0	1	1	0	0	0	0	1	61H
				n	n			nn
				m	m			mm

フラグ

I1	I0	U	D	N	V	C	Z
−	−	−	−	↕	↕	↕	↕

例

設定値			結果				
HL	mmnn	C	HL	SC			
				N	V	C	Z
1380H	3546H	0	48C6H	0	0	0	0
1380H	3546H	1	48C7H	0	0	0	0
1380H	EC80H	0	0000H	0	0	1	1
5218H	4174H	0	938CH	1	1	0	0
5342H	C32AH	1	166DH	0	0	1	0

ADD A, r ////////////////////// Add r reg. to A reg. // 2 cycle ///

機 能 A ← A + r

rレジスタ(A/B)の内容をAレジスタに加えます。

モード Src: レジスタ直接
Dst: レジスタ直接

コード

MSB							LSB	
0	0	0	0	0	0	0	r	00H、01H ※

※

r		ニーモニック	コード
A	0	ADD A, A	00H
B	1	ADD A, B	01H

フラグ

I1	I0	U	D	N	V	C	Z
−	−	★	★	↕	↕	↕	↕

例

設定値		結果				
A	B	A	SC			
			N	V	C	Z
D=0, U=0の場合						
18H	25H	3DH	0	0	0	0
30H	D0H	00H	0	0	1	1
30H	F0H	20H	0	0	1	0
30H	50H	80H	1	1	0	0
D=1, U=0の場合						
18	24	42	0	0	0	0
D=1, U=1の場合						
18	24	02	0	0	1	0

ADD A, #nn ///////////////// Add immediate data nn to A reg. // 2 cycle ///

機 能 A ← A + nn

8ビット即値データnnをAレジスタに加えます。

モード Src: 即値データ
Dst: レジスタ直接

コード

MSB							LSB	
0	0	0	0	0	0	1	0	02H
				n	n			nn

フラグ

I1	I0	U	D	N	V	C	Z
−	−	★	★	↕	↕	↕	↕

例

設定値		結果				
A	nn	A	SC			
			N	V	C	Z
D=0, U=0の場合						
18H	25H	3DH	0	0	0	0
30H	D0H	00H	0	0	1	1
30H	F0H	20H	0	0	1	0
30H	50H	80H	1	1	0	0
D=1, U=0の場合						
18	24	42	0	0	0	0
D=1, U=1の場合						
18	24	02	0	0	1	0

ADD A, [BR:ll] //////// Add location [BR:ll] to A reg. // 3 cycle ///

機 能 A ← A + [BR:*ll*]

BRレジスタの内容を上位バイト、8ビット絶対アドレス*ll*を下位バイトとしてアドレス指定されるデータメモリの内容をAレジスタに加えます。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

コード

MSB							LSB	
0	0	0	0	0	1	0	0	04H
l	*l*							*ll*

フラグ

I1	I0	U	D	N	V	C	Z
−	−	★	★	↕	↕	↕	↕

モード Src: 8ビット絶対
Dst: レジスタ直接

例

設定値		結果				
A	[BR:*ll*]	A	SC			
			N	V	C	Z
D=0, U=0の場合						
18H	25H	3DH	0	0	0	0
30H	D0H	00H	0	0	1	1
30H	F0H	20H	0	0	1	0
30H	50H	80H	1	1	0	0
D=1, U=0の場合						
18	24	42	0	0	0	0
D=1, U=1の場合						
18	24	02	0	0	1	0

ADD A, [hhll] ////////////// Add location [hhll] to A reg. // 4 cycle ///

機 能 A ← A + [hh*ll*]

16ビット絶対アドレスhh*ll*でアドレス指定されたデータメモリの内容をAレジスタに加えます。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

コード

MSB							LSB	
0	0	0	0	0	1	0	1	05H
l	*l*							*ll*
h	h							hh

フラグ

I1	I0	U	D	N	V	C	Z
−	−	★	★	↕	↕	↕	↕

モード Src: 16ビット絶対
Dst: レジスタ直接

例

設定値		結果				
A	[hh*ll*]	A	SC			
			N	V	C	Z
D=0, U=0の場合						
18H	25H	3DH	0	0	0	0
30H	D0H	00H	0	0	1	1
30H	F0H	20H	0	0	1	0
30H	50H	80H	1	1	0	0
D=1, U=0の場合						
18	24	42	0	0	0	0
D=1, U=1の場合						
18	24	02	0	0	1	0

ADD A, [HL] ////////////// Add location [HL] to A reg. // 2 cycle ///

機 能 A ← A + [HL]

HLレジスタでアドレス指定されたデータメモリの内容をAレジスタに加えます。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

コード

MSB							LSB	
0	0	0	0	0	0	1	1	03H

フラグ

I1	I0	U	D	N	V	C	Z
−	−	★	★	↕	↕	↕	↕

モード Src: レジスタ間接
Dst: レジスタ直接

例

設定値		結果				
A	[HL]	A	SC			
			N	V	C	Z
D=0, U=0の場合						
18H	25H	3DH	0	0	0	0
30H	D0H	00H	0	0	1	1
30H	F0H	20H	0	0	1	0
30H	50H	80H	1	1	0	0
D=1, U=0の場合						
18	24	42	0	0	0	0
D=1, U=1の場合						
18	24	02	0	0	1	0

ADD A, [ir] //////////////////// *Add location [ir reg.] to A reg.* // *2 cycle ///*

機 能 **A ← A + [ir]**

irレジスタ(IX/IY)でアドレス指定されたデータメモリの内容をAレジスタに加えます。
XPレジスタ(IX指定時)、YPレジスタ(IY指定時)の内容がデータメモリのページアドレスになります(MODEL2/3)。

コード

MSB							LSB	
0	0	0	0	0	1	1	ir	06H、07H ※

※

ir		ニーモニック	コード
IX	0	ADD A, [IX]	06H
IY	1	ADD A, [IY]	07H

フラグ

I1	I0	U	D	N	V	C	Z
−	−	★	★	↕	↕	↕	↕

モード **Src: レジスタ間接**
Dst: レジスタ直接

例

設定値		結果				
A	[ir]	A	SC			
			N	V	C	Z
D=0, U=0の場合						
18H	25H	3DH	0	0	0	0
30H	D0H	00H	0	0	1	1
30H	F0H	20H	0	0	1	0
30H	50H	80H	1	1	0	0
D=1, U=0の場合						
18	24	42	0	0	0	0
D=1, U=1の場合						
18	24	02	0	0	1	0

ADD A, [ir+dd] //////// *Add location [ir reg. + dd] to A reg.* ////////////////////////////////////// *4 cycle ///*

機 能 **A ← A + [ir+dd]**

irレジスタ(IX/IY)の内容とディスプレースメントddの和でアドレス指定されたデータメモリの内容をAレジスタに加えます。
ddは符号付きデータとして扱われ、範囲は-128～127です。
XPレジスタ(IX指定時)、YPレジスタ(IY指定時)の内容がデータメモリのページアドレスになります(MODEL2/3)。

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	0	0	0	0	0	ir	00H、01H※
				d	d			dd

※

ir		ニーモニック	コード
IX	0	ADD A, [IX+dd]	00H
IY	1	ADD A, [IY+dd]	01H

フラグ

I1	I0	U	D	N	V	C	Z
−	−	★	★	↕	↕	↕	↕

モード **Src: ディスプレースメント付きレジスタ間接**
Dst: レジスタ直接

例

設定値		結果				
A	[ir+dd]	A	SC			
			N	V	C	Z
D=0, U=0の場合						
18H	25H	3DH	0	0	0	0
30H	D0H	00H	0	0	1	1
30H	F0H	20H	0	0	1	0
30H	50H	80H	1	1	0	0
D=1, U=0の場合						
18	24	42	0	0	0	0
D=1, U=1の場合						
18	24	02	0	0	1	0

ADD A, [ir+L] *Add location [ir reg. + L] to A reg.* 4 cycle

機 能 A ← A + [ir+L]

irレジスタ(IX/IY)の内容とLレジスタの内容の和でアドレス指定されたデータメモリの内容をAレジスタに加えます。
Lレジスタの内容は符号付きデータとして扱われ、範囲は-128～127です。
XPレジスタ(IX指定時)、YPレジスタ(IY指定時)の内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: インデックスレジスタ付きレジスタ間接
Dst: レジスタ直接

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	0	0	0	0	1	ir	02H、03H ※

※

ir		ニーモニック	コード
IX	0	ADD A, [IX+L]	02H
IY	1	ADD A, [IY+L]	03H

フラグ

I1	I0	U	D	N	V	C	Z
−	−	★	★	↕	↕	↕	↕

例

設定値		結果				
A	[ir+L]	A	SC			
			N	V	C	Z
D=0, U=0の場合						
18H	25H	3DH	0	0	0	0
30H	D0H	00H	0	0	1	1
30H	F0H	20H	0	0	1	0
30H	50H	80H	1	1	0	0
D=1, U=0の場合						
18	24	42	0	0	0	0
D=1, U=1の場合						
18	24	02	0	0	1	0

ADD [HL], A *Add A reg. to location [HL]* 4 cycle

機 能 [HL] ← [HL] + A

Aレジスタの内容をHLレジスタでアドレス指定されたデータメモリに加えます。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: レジスタ直接
Dst: レジスタ間接

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	0	0	0	1	0	0	04H

フラグ

I1	I0	U	D	N	V	C	Z
−	−	★	★	↕	↕	↕	↕

例

設定値		結果				
[HL]	A	[HL]	SC			
			N	V	C	Z
D=0, U=0の場合						
18H	25H	3DH	0	0	0	0
30H	D0H	00H	0	0	1	1
30H	F0H	20H	0	0	1	0
30H	50H	80H	1	1	0	0
D=1, U=0の場合						
18	24	42	0	0	0	0
D=1, U=1の場合						
18	24	02	0	0	1	0

ADD [HL], #nn *Add immediate data nn to location [HL]* 5 cycle

機 能 [HL] ← [HL] + nn

8ビット即値データnnをHLレジスタでアドレス指定されたデータメモリに加えます。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: 即値データ
Dst: レジスタ間接

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	0	0	0	1	0	1	05H
				n	n			nn

フラグ

I1	I0	U	D	N	V	C	Z
−	−	★	★	↕	↕	↕	↕

例

設定値		結果				
[HL]	nn	[HL]	SC			
			N	V	C	Z
D=0, U=0の場合						
18H	25H	3DH	0	0	0	0
30H	D0H	00H	0	0	1	1
30H	F0H	20H	0	0	1	0
30H	50H	80H	1	1	0	0
D=1, U=0の場合						
18	24	42	0	0	0	0
D=1, U=1の場合						
18	24	02	0	0	1	0

ADD [HL], [ir] ////////// Add location [ir reg.] to location [HL] ////////////////////////////////// 5 cycle ///

機 能 [HL] ← [HL] + [ir]

irレジスタ(IX/IY)でアドレス指定されたデータメモリの内容をHLレジスタでアドレス指定されるデータメモリに加えます。
EPレジスタの内容がデータメモリ[HL]のページアドレス、XPレジスタ(IX指定時)、YPレジスタ(IY指定時)の内容がデータメモリ[ir]のページアドレスになります(MODEL2/3)。

モード Src: レジスタ間接
Dst: レジスタ間接

例

設定値		結果				
[HL]	[ir]	[HL]	SC			
			N	V	C	Z
D=0, U=0の場合						
18H	25H	3DH	0	0	0	0
30H	D0H	00H	0	0	1	1
30H	F0H	20H	0	0	1	0
30H	50H	80H	1	1	0	0
D=1, U=0の場合						
18	24	42	0	0	0	0
D=1, U=1の場合						
18	24	02	0	0	1	0

コード MSB LSB

1	1	0	0	1	1	1	0	CEH
0	0	0	0	0	1	1	ir	06H、07H ※

※

ir		ニーモニック	コード
IX	0	ADD [HL], [IX]	06H
IY	1	ADD [HL], [IY]	07H

フラグ

I1	I0	U	D	N	V	C	Z
−	−	★	★	↕	↕	↕	↕

ADD BA, rp ////////////// Add rp reg. to BA reg. // 4 cycle ///

機 能 BA ← BA + rp

rpレジスタ(BA/HL/IX/IY)の内容をBAレジスタに加えます。

モード Src: レジスタ直接
Dst: レジスタ直接

例

設定値		結果				
BA	rp	BA	SC			
			N	V	C	Z
1380H	3546H	48C6H	0	0	0	0
1380H	EC80H	0000H	0	0	1	1
5218H	4174H	938CH	1	1	0	0
5342H	C32AH	166CH	0	0	1	0

(rp≠BA)

コード MSB LSB

1	1	0	0	1	1	1	1	CFH
0	0	0	0	0	0	rp		00H〜03H ※

※

rp		ニーモニック	コード
BA	00	ADD BA, BA	00H
HL	01	ADD BA, HL	01H
IX	10	ADD BA, IX	02H
IY	11	ADD BA, IY	03H

フラグ

I1	I0	U	D	N	V	C	Z
−	−	−	−	↕	↕	↕	↕

ADD BA, #mmnn /// Add immediate data mmnn to BA reg. ////////////////////////////////// 3 cycle ///

機 能 BA ← BA + mmnn

16ビット即値データmmnnをBAレジスタに加えます。

モード Src: 即値データ
Dst: レジスタ直接

例

設定値		結果				
BA	mmnn	BA	SC			
			N	V	C	Z
1380H	3546H	48C6H	0	0	0	0
1380H	EC80H	0000H	0	0	1	1
5218H	4174H	938CH	1	1	0	0
5342H	C32AH	166CH	0	0	1	0

コード MSB LSB

1	1	0	0	0	0	0	0	C0H
				n	n			nn
				m	m			mm

フラグ

I1	I0	U	D	N	V	C	Z
−	−	−	−	↕	↕	↕	↕

ADD HL, rp /////////////// Add rp reg. to HL reg. // 4 cycle ///

機 能 HL ← HL + rp

rpレジスタ(BA/HL/IX/IY)の内容をHLレジスタに加えます。

モード Src: レジスタ直接
Dst: レジスタ直接

コード

MSB LSB

1	1	0	0	1	1	1	1	CFH

0	0	1	0	0	0	rp		20H～23H ※

※

rp		ニーモニック	コード
BA	00	ADD HL, BA	20H
HL	01	ADD HL, HL	21H
IX	10	ADD HL, IX	22H
IY	11	ADD HL, IY	23H

フラグ

I1	I0	U	D	N	V	C	Z
−	−	−	−	↕	↕	↕	↕

例

設定値		結果				
HL	rp	HL	SC			
			N	V	C	Z
1380H	3546H	48C6H	0	0	0	0
1380H	EC80H	0000H	0	0	1	1
5218H	4174H	938CH	1	1	0	0
5342H	C32AH	166CH	0	0	1	0

(rp≠HL)

ADD HL, #mmnn /// Add immediate data mmnn to HL reg. ////////////////////////////////////// 3 cycle ///

機 能 HL ← HL + mmnn

16ビット即値データmmnnをHLレジスタに加えます。

モード Src: 即値データ
Dst: レジスタ直接

コード

MSB LSB

1	1	0	0	0	0	0	1	C1H

n	n	nn

m	m	mm

フラグ

I1	I0	U	D	N	V	C	Z
−	−	−	−	↕	↕	↕	↕

例

設定値		結果				
HL	mmnn	HL	SC			
			N	V	C	Z
1380H	3546H	48C6H	0	0	0	0
1380H	EC80H	0000H	0	0	1	1
5218H	4174H	938CH	1	1	0	0
5342H	C32AH	166CH	0	0	1	0

ADD IX, rp ////////////////// Add rp reg. to IX reg. // 4 cycle ///

機 能 IX ← IX + rp

rpレジスタ(BA/HL)の内容をIXレジスタに加えます。

モード Src: レジスタ直接
Dst: レジスタ直接

コード

MSB LSB

1	1	0	0	1	1	1	1	CFH

0	1	0	0	0	0	0	rp	40H、41H ※

※

rp		ニーモニック	コード
BA	0	ADD IX, BA	40H
HL	1	ADD IX, HL	41H

フラグ

I1	I0	U	D	N	V	C	Z
−	−	−	−	↕	↕	↕	↕

例

設定値		結果				
IX	rp	IX	SC			
			N	V	C	Z
1380H	3546H	48C6H	0	0	0	0
1380H	EC80H	0000H	0	0	1	1
5218H	4174H	938CH	1	1	0	0
5342H	C32AH	166CH	0	0	1	0

ADD IX, #mmnn ///// Add immediate data mmnn to IX reg. /////////////////////////////////// 3 cycle ///

機 能 IX ← IX + mmnn

16ビット即値データmmnnをIXレジスタに加えます。

モード Src: 即値データ
Dst: レジスタ直接

コード

MSB							LSB	
1	1	0	0	0	0	1	0	C2H
				n	n			nn
				m	m			mm

フラグ

I1	I0	U	D	N	V	C	Z
−	−	−	−	↕	↕	↕	↕

例

設定値		結果				
IX	mmnn	IX	SC			
			N	V	C	Z
1380H	3546H	48C6H	0	0	0	0
1380H	EC80H	0000H	0	0	1	1
5218H	4174H	938CH	1	1	0	0
5342H	C32AH	166CH	0	0	1	0

ADD IY, rp ////////////////////// Add rp reg. to IY reg. // 4 cycle ///

機 能 IY ← IY + rp

rpレジスタ(BA/HL)の内容をIYレジスタに加えます。

モード Src: レジスタ直接
Dst: レジスタ直接

コード

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
0	1	0	0	0	0	1	rp	42H、43H ※

※

rp		ニーモニック	コード
BA	0	ADD IY, BA	42H
HL	1	ADD IY, HL	43H

フラグ

I1	I0	U	D	N	V	C	Z
−	−	−	−	↕	↕	↕	↕

例

設定値		結果				
IY	rp	IY	SC			
			N	V	C	Z
1380H	3546H	48C6H	0	0	0	0
1380H	EC80H	0000H	0	0	1	1
5218H	4174H	938CH	1	1	0	0
5342H	C32AH	166CH	0	0	1	0

ADD IY, #mmnn ////// Add immediate data mmnn to IY reg. /////////////////////////////////// 3 cycle ///

機 能 IY ← IY + mmnn

16ビット即値データmmnnをIYレジスタに加えます。

モード Src: 即値データ
Dst: レジスタ直接

コード

MSB							LSB	
1	1	0	0	0	0	1	1	C3H
				n	n			nn
				m	m			mm

フラグ

I1	I0	U	D	N	V	C	Z
−	−	−	−	↕	↕	↕	↕

例

設定値		結果				
IY	mmnn	IY	SC			
			N	V	C	Z
1380H	3546H	48C6H	0	0	0	0
1380H	EC80H	0000H	0	0	1	1
5218H	4174H	938CH	1	1	0	0
5342H	C32AH	166CH	0	0	1	0

ADD SP, rp /////////////////// Add rp reg. to SP // 4 cycle ///

機 能 SP ← SP + rp

rpレジスタ(BA/HL)の内容をスタックポインタ(SP)に加えます。

コード

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
0	1	0	0	0	1	0	rp	44H、45H ※

※

rp		ニーモニック	コード
BA	0	ADD SP, BA	44H
HL	1	ADD SP, HL	45H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	↕	↕	↕

モード Src: レジスタ直接
Dst: レジスタ直接

例

設定値		結果				
SP	rp	SP	SC			
			N	V	C	Z
1380H	3546H	48C6H	0	0	0	0
1380H	EC80H	0000H	0	0	1	1
5218H	4174H	938CH	1	1	0	0
5342H	C32AH	166CH	0	0	1	0

ADD SP, #mmnn //// Add immediate data mmnn to SP // 4 cycle ///

機 能 SP ← SP + mmnn

16ビット即値データmmnnをスタックポインタ(SP)に加えます。

コード

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
0	1	1	0	1	0	0	0	68H
			n	n				nn
			m	m				mm

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	↕	↕	↕

モード Src: 即値データ
Dst: レジスタ直接

例

設定値		結果				
SP	mmnn	SP	SC			
			N	V	C	Z
1380H	3546H	48C6H	0	0	0	0
1380H	EC80H	0000H	0	0	1	1
5218H	4174H	938CH	1	1	0	0
5342H	C32AH	166CH	0	0	1	0

AND A, r ////////////////////// Logical AND of r reg. and A reg. // 2 cycle ///

機 能 A ← A ∧ r

rレジスタ(A/B)の内容とAレジスタの内容との論理積をとり、結果をAレジスタにストアします。

コード

MSB							LSB	
0	0	1	0	0	0	0	r	20H、21H ※

※

r		ニーモニック	コード
A	0	AND A, A	20H
B	1	AND A, B	21H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	—	↕

モード Src: レジスタ直接
Dst: レジスタ直接

例

設定値		結果				
A	B	A	SC			
			N	V	C	Z
3BH	61H	21H	0	—	—	0
5AH	A5H	00H	0	—	—	1
D6H	93H	92H	1	—	—	0

AND A, #nn /////////////// Logical AND of immediate data nn and A reg. ////////////////////////// 2 cycle ///

機 能 A ← A ∧ nn

8ビット即値データnnとAレジスタの内容との論理積をとり、結果をAレジスタにストアします。

コード

MSB							LSB	
0	0	1	0	0	0	1	0	22H
n	n							nn

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	—	↕

モード Src: 即値データ
Dst: レジスタ直接

例

設定値		結果				
A	nn	A	SC			
			N	V	C	Z
3BH	61H	21H	0	—	—	0
5AH	A5H	00H	0	—	—	1
D6H	93H	92H	1	—	—	0

AND A, [BR:ll] /////// Logical AND of location [BR:ll] and A reg. ////////////////////////////// 3 cycle ///

機 能 A ← A ∧ [BR:*ll*]

BRレジスタの内容を上位バイト、8ビット絶対アドレス*ll*を下位バイトとしてアドレス指定されるデータメモリの内容とAレジスタの内容との論理積をとり、結果をAレジスタにストアします。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

コード

MSB							LSB	
0	0	1	0	0	1	0	0	24H
l	*l*							*ll*

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	—	↕

モード Src: 8ビット絶対
Dst: レジスタ直接

例

設定値		結果				
A	[BR:*ll*]	A	SC			
			N	V	C	Z
3BH	61H	21H	0	—	—	0
5AH	A5H	00H	0	—	—	1
D6H	93H	92H	1	—	—	0

AND A, [hhll] /////////// Logical AND of location [hhll] and A reg. ////////////////////////////// 4 cycle ///

機 能 A ← A ∧ [hh*ll*]

16ビット絶対アドレスhh*ll*でアドレス指定されたデータメモリの内容とAレジスタの内容との論理積をとり、結果をAレジスタにストアします。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

コード

MSB							LSB	
0	0	1	0	0	1	0	1	25H
l	*l*							*ll*
h	h							hh

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	—	↕

モード Src: 16ビット絶対
Dst: レジスタ直接

例

設定値		結果				
A	[hh*ll*]	A	SC			
			N	V	C	Z
3BH	61H	21H	0	—	—	0
5AH	A5H	00H	0	—	—	1
D6H	93H	92H	1	—	—	0

AND A, [HL] /////////////// Logical AND of location [HL] and A reg. ////////////////////////////// 2 cycle ///

機 能 A ← A ∧ [HL]

HLレジスタでアドレス指定されたデータメモリの内容とAレジスタの内容との論理積をとり、結果をAレジスタにストアします。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: レジスタ間接
Dst: レジスタ直接

例

設定値		結果				
A	[HL]	A	SC			
			N	V	C	Z
3BH	61H	21H	0	—	—	0
5AH	A5H	00H	0	—	—	1
D6H	93H	92H	1	—	—	0

コード

MSB							LSB	
0	0	1	0	0	0	1	1	23H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	—	↕

AND A, [ir] ////////////////// Logical AND of location [ir reg.] and A reg. ////////////////////////// 2 cycle ///

機 能 A ← A ∧ [ir]

irレジスタ(IX/IY)でアドレス指定されたデータメモリの内容とAレジスタの内容との論理積をとり、結果をAレジスタにストアします。
XPレジスタ(IX指定時)、YPレジスタ(IY指定時)の内容がページアドレスになります(MODEL2/3)。

モード Src: レジスタ間接
Dst: レジスタ直接

例

設定値		結果				
A	[ir]	A	SC			
			N	V	C	Z
3BH	61H	21H	0	—	—	0
5AH	A5H	00H	0	—	—	1
D6H	93H	92H	1	—	—	0

コード

MSB							LSB	
0	0	1	0	0	1	1	ir	26H、27H ※

※

ir		ニーモニック	コード
IX	0	AND A, [IX]	26H
IY	1	AND A, [IY]	27H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	—	↕

AND A, [ir+dd] /////// Logical AND of location [ir reg. + dd] and A reg. ////////////////// 4 cycle ///

機 能 A ← A ∧ [ir+dd]

irレジスタ(IX/IY)の内容とディスプレースメントddの和でアドレス指定されたデータメモリの内容とAレジスタの内容との論理積をとり、結果をAレジスタにストアします。ddは符号付きデータとして扱われ、範囲は-128～127です。
XPレジスタ(IX指定時)、YPレジスタ(IY指定時)の内容がページアドレスになります(MODEL2/3)。

モード Src: ディスプレースメント付きレジスタ間接
Dst: レジスタ直接

例

設定値		結果				
A	[ir+dd]	A	SC			
			N	V	C	Z
3BH	61H	21H	0	—	—	0
5AH	A5H	00H	0	—	—	1
D6H	93H	92H	1	—	—	0

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	1	0	0	0	0	ir	20H、21H※
d	d							dd

※

ir		ニーモニック	コード
IX	0	AND A, [IX+dd]	20H
IY	1	AND A, [IY+dd]	21H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	—	↕

AND A, [ir+L] ////////// *Logical AND of location [ir reg. + L] and A reg.* ////////////////// *4 cycle* ///

機 能 A ← A ∧ [ir+L]

irレジスタ(IX/IY)の内容とLレジスタの内容の和でアドレス指定されたデータメモリの内容とAレジスタの内容との論理積をとり、結果をAレジスタにストアします。
Lレジスタの内容は符号付きデータとして扱われ、範囲は-128～127です。
XPレジスタ(IX指定時)、YPレジスタ(IY指定時)の内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: インデックスレジスタ付きレジスタ間接
Dst: レジスタ直接

例

設定値		結果				
A	[ir+L]	A	SC			
			N	V	C	Z
3BH	61H	21H	0	—	—	0
5AH	A5H	00H	0	—	—	1
D6H	93H	92H	1	—	—	0

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	1	0	0	0	1	ir	22H、23H ※

※

ir		ニーモニック	コード
IX	0	AND A, [IX+L]	22H
IY	1	AND A, [IY+L]	23H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	—	↕

AND B, #nn ////////////// *Logical AND of immediate data nn and B reg.* ////////////////////// *3 cycle* ///

機 能 B ← B ∧ nn

8ビット即値データnnとBレジスタの内容との論理積をとり、結果をBレジスタにストアします。

モード Src: 即値データ
Dst: レジスタ直接

例

設定値		結果				
B	nn	B	SC			
			N	V	C	Z
3BH	61H	21H	0	—	—	0
5AH	A5H	00H	0	—	—	1
D6H	93H	92H	1	—	—	0

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	1	1	0	0	0	0	B0H
				n	n			nn

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	—	↕

AND L, #nn ////////////// *Logical AND of immediate data nn and L reg.* ////////////////////// *3 cycle* ///

機 能 L ← L ∧ nn

8ビット即値データnnとLレジスタの内容との論理積をとり、結果をLレジスタにストアします。

モード Src: 即値データ
Dst: レジスタ直接

例

設定値		結果				
L	nn	L	SC			
			N	V	C	Z
3BH	61H	21H	0	—	—	0
5AH	A5H	00H	0	—	—	1
D6H	93H	92H	1	—	—	0

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	1	1	0	0	0	1	B1H
				n	n			nn

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	—	↕

AND H, #nn /////////////// Logical AND of immediate data nn and H reg. ////////////////////// 3 cycle ///

機 能 H ← H ∧ nn

8ビット即値データnnとHレジスタの内容との論理積をとり、結果をHレジスタにストアします。

モード Src: 即値データ
Dst: レジスタ直接

例

設定値		結果				
H	nn	H	SC			
			N	V	C	Z
3BH	61H	21H	0	—	—	0
5AH	A5H	00H	0	—	—	1
D6H	93H	92H	1	—	—	0

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	1	1	0	0	1	0	B2H
				n	n			nn

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	—	↕

AND SC, #nn ////////// Logical AND of immediate data nn and SC ////////////////////////////// 3 cycle ///

機 能 SC ← SC ∧ nn

8ビット即値データnnとシステムコンディションフラグ(SC)の内容との論理積をとり、結果をシステムコンディションフラグ(SC)にセットします。

モード Src: 即値データ
Dst: レジスタ直接

例

設定値		結果							
SC	nn	SC	SC						
			I1	I0	U	D	N	V	C	Z
3BH	61H	21H	0	0	1	0	0	0	0	1
5AH	A5H	00H	0	0	0	0	0	0	0	0
D6H	93H	92H	1	0	0	1	0	0	1	0

コード

MSB							LSB	
1	0	0	1	1	1	0	0	9CH
				n	n			nn

フラグ

I1	I0	U	D	N	V	C	Z
↓	↓	↓	↓	↓	↓	↓	↓

AND [BR:ll], #nn // Logical AND of immediate data nn and location [BR:ll] /////// 5 cycle ///

機 能 [BR:*ll*] ← [BR:*ll*] ∧ nn

8ビット即値データnnとBRレジスタの内容を上位バイト、8ビット絶対アドレス*ll*を下位バイトとしてアドレス指定されるデータメモリの内容との論理積をとり、結果をそのアドレスにストアします。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: 即値データ
Dst: 8ビット絶対

例

設定値		結果				
[BR:*ll*]	nn	[BR:*ll*]	SC			
			N	V	C	Z
3BH	61H	21H	0	—	—	0
5AH	A5H	00H	0	—	—	1
D6H	93H	92H	1	—	—	0

コード

MSB							LSB	
1	1	0	1	1	0	0	0	D8H
			l	*l*				*ll*
				n	n			nn

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	—	↕

AND [HL], A /////////////// Logical AND of A reg. and location [HL] ////////////////////////////// 4 cycle ///

機 能 [HL] ← [HL] ∧ A

Aレジスタの内容とHLレジスタでアドレス指定されたデータメモリの内容との論理積をとり、結果をそのアドレスにストアします。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: レジスタ直接
Dst: レジスタ間接

例

設定値		結果				
[HL]	A	[HL]	SC			
			N	V	C	Z
3BH	61H	21H	0	—	—	0
5AH	A5H	00H	0	—	—	1
D6H	93H	92H	1	—	—	0

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	1	0	0	1	0	0	24H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	—	↕

AND [HL], #nn /////// Logical AND of immediate data nn and location [HL] /////////// 5 cycle ///

機 能 [HL] ← [HL] ∧ nn

8ビット即値データnnとHLレジスタでアドレス指定されたデータメモリの内容との論理積をとり、結果をそのアドレスにストアします。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: 即値データ
Dst: レジスタ間接

例

設定値		結果				
[HL]	nn	[HL]	SC			
			N	V	C	Z
3BH	61H	21H	0	—	—	0
5AH	A5H	00H	0	—	—	1
D6H	93H	92H	1	—	—	0

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	1	0	0	1	0	1	25H
n	n							nn

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	—	↕

AND [HL], [ir] ///////// Logical AND of location [ir reg.] and location [HL] ////////////// 5 cycle ///

機 能 [HL] ← [HL] ∧ [ir]

irレジスタ(IX/IY)とHLレジスタでそれぞれアドレス指定されたデータメモリの内容の論理積をとり、結果をデータメモリ[HL]にストアします。
EPレジスタの内容がデータメモリ[HL]のページアドレス、XPレジスタ(IX指定時)、YPレジスタ(IY指定時)の内容がデータメモリ[ir]のページアドレスになります(MODEL2/3)。

モード Src: レジスタ間接
Dst: レジスタ間接

例

設定値		結果				
[HL]	[ir]	[HL]	SC			
			N	V	C	Z
3BH	61H	21H	0	—	—	0
5AH	A5H	00H	0	—	—	1
D6H	93H	92H	1	—	—	0

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	1	0	0	1	1	ir	26H、27H ※

※

ir		ニーモニック	コード
IX	0	AND [HL], [IX]	26H
IY	1	AND [HL], [IY]	27H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	—	↕

BIT A, B ////////////////////// *Test bit of A reg. with B reg.* // *2 cycle* ///

機 能 A ∧ B

Bレジスタの内容とAレジスタの内容との論理積をとり、Aレジスタのビットをチェックします。その結果によりフラグ(N/Z)が変更されますが、レジスタの内容は変更されません。

モード Src: レジスタ直接
Dst: レジスタ直接

例

設定値		結果				
A	B	A	SC			
			N	V	C	Z
3BH	61H	3BH	0	—	—	0
5AH	A5H	5AH	0	—	—	1
D6H	93H	D6H	1	—	—	0

コード

MSB							LSB	
1	0	0	1	0	1	0	0	94H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	—	↕

BIT A, #nn ////////////////// *Test bit of A reg. with immediate data nn* ////////////////////////////// *2 cycle* ///

機 能 A ∧ nn

8ビット即値データnnとAレジスタの内容との論理積をとり、Aレジスタのビットをチェックします。その結果によりフラグ(N/Z)が変更されますが、レジスタの内容は変更されません。

モード Src: 即値データ
Dst: レジスタ直接

例

設定値		結果				
A	nn	A	SC			
			N	V	C	Z
3BH	61H	3BH	0	—	—	0
5AH	A5H	5AH	0	—	—	1
D6H	93H	D6H	1	—	—	0

コード

MSB							LSB	
1	0	0	1	0	1	1	0	96H
			n	n				nn

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	—	↕

BIT B, #nn ////////////////// *Test bit of B reg. with immediate data nn* ////////////////////////////// *2 cycle* ///

機 能 B ∧ nn

8ビット即値データnnとBレジスタの内容との論理積をとり、Bレジスタのビットをチェックします。その結果によりフラグ(N/Z)が変更されますが、レジスタの内容は変更されません。

モード Src: 即値データ
Dst: レジスタ直接

例

設定値		結果				
B	nn	B	SC			
			N	V	C	Z
3BH	61H	3BH	0	—	—	0
5AH	A5H	5AH	0	—	—	1
D6H	93H	D6H	1	—	—	0

コード

MSB							LSB	
1	0	0	1	0	1	1	1	97H
			n	n				nn

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	—	↕

BIT [BR:ll], #nn ///// Test bit of location [BR:ll] with immediate data nn /////////////// 4 cycle ///

機 能 [BR:*ll*] ∧ nn

8ビット即値データnnとBRレジスタの内容を上位バイト、8ビット絶対アドレス*ll*を下位バイトとしてアドレス指定されるデータメモリの内容との論理積をとり、そのメモリのビットをチェックします。その結果によりフラグ(N/Z)が変更されますが、メモリの内容は変更されません。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: 即値データ
Dst: 8ビット絶対

例

設定値		結果				
[BR:*ll*]	nn	[BR:*ll*]	SC			
			N	V	C	Z
3BH	61H	3BH	0	—	—	0
5AH	A5H	5AH	0	—	—	1
D6H	93H	D6H	1	—	—	0

コード

MSB							LSB	
1	1	0	1	1	1	0	0	DCH
			l	*l*				*ll*
			n	n				nn

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	—	↕

BIT [HL], #nn ////////// Test bit of location [HL] with immediate data nn ////////////////// 3 cycle ///

機 能 [HL] ∧ nn

8ビット即値データnnとHLレジスタでアドレス指定されるデータメモリの内容との論理積をとり、そのメモリのビットをチェックします。その結果によりフラグ(N/Z)が変更されますが、メモリの内容は変更されません。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: 即値データ
Dst: レジスタ間接

例

設定値		結果				
[HL]	nn	[HL]	SC			
			N	V	C	Z
3BH	61H	3BH	0	—	—	0
5AH	A5H	5AH	0	—	—	1
D6H	93H	D6H	1	—	—	0

コード

MSB							LSB	
1	0	0	1	0	1	0	1	95H
			n	n				nn

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	—	↕

CALL [hhll] ////////////// *Call subroutine at location [hhll]* ////////////////////// *7(MIN)/8(MAX) cycle* ///

機 能 **■MODEL0/1**

[SP-1]←PC(H), [SP-2]←PC(L), SP←SP-2, PC(L)←[hh*ll*], PC(H)←[hh*ll*+1]

■MODEL2/3(ミニマムモード)

[SP-1]←PC(H), [SP-2]←PC(L), SP←SP-2, PC(L)←[hh*ll*], PC(H)←[hh*ll*+1], CB←NB

■MODEL2/3(マキシマムモード)

[SP-1]←CB, [SP-2]←PC(H), [SP-3]←PC(L), SP←SP-3, PC(L)←[hh*ll*], PC(H)←[hh*ll*+1], CB←NB

本命令の先頭アドレス+3の値をリターンアドレスとしてスタックに退避後、サブルーチンを無条件にコールします。分岐先アドレス(サブルーチンの先頭アドレス)としては、16ビット絶対アドレスhh*ll*で指定されたデータメモリの内容が下位バイト、その次のアドレスの内容が上位バイトとなります。

MODEL2/3のマキシマムモードでは、リターンアドレスの退避時に現在選択されているバンクアドレス(CBの内容)の退避も行われます。また、MODEL2/3では、分岐時にNBに設定されているバンクアドレスがCBにロードされ、バンクの変更も行われます。

EPレジスタの内容がデータメモリのページアドレスになります。

コード

MSB							LSB	
1	1	1	1	1	0	1	1	FBH
l	*l*							*ll*
h	h							hh

フラグ

I1	I0	U	D	N	V	C	Z
−	−	−	−	−	−	−	−

モード **16ビット間接**

例 MODEL2/3においてNB=02Hの場合、物理アドレス9000H番地にあるCALL [2000H]命令を実行。

	NB	CB	PC(論理アドレス)	EP	M(032000H)	SP
実行前	02H	01H	9000H	03H	ABCDH	0000H
実行後	02H	02H	ABCDH	03H	ABCDH	FFFDH

上記の例では物理アドレス012BCDHへ分岐します。MODEL0/1ではEPがないためM(2000H)の内容がPCに転送され、M(2000H)の内容がABCDHであればNB, CBもないため物理アドレスABCDHへ分岐します。

実行後のスタック内容

(1) MODEL2/3(マキシマムモード)

00FFFDH	03H (PC(L))
00FFFEH	90H (PC(H))
00FFFFH	01H (CB)

(2) MODEL2/3(ミニマムモード) MODEL0/1

00FFFEH	03H (PC(L))
00FFFFH	90H (PC(H))

CARL qqrr //////////////// Call subroutine at relative location qqrr //////////////// 5(MIN)/6(MAX) cycle ///

機 能 ■MODEL0/1

[SP-1]←PC(H), [SP-2]←PC(L), SP←SP-2, PC←PC+qqrr+2

■MODEL2/3(ミニマムモード)

[SP-1]←PC(H), [SP-2]←PC(L), SP←SP-2, PC←PC+qqrr+2, CB←NB

■MODEL2/3(マキシマムモード)

[SP-1]←CB, [SP-2]←PC(H), [SP-3]←PC(L), SP←SP-3, PC←PC+qqrr+2, CB←NB

本命令の先頭アドレス+3の値をリターンアドレスとしてスタックに退避後、サブルーチンを無条件にコールします。分岐先(サブルーチンの先頭アドレス)は、本命令の先頭アドレス+2に符号付き16ビット相対アドレスqqrr(-32768～32767)を加算したアドレスになります。

MODEL2/3のマキシマムモードでは、リターンアドレスの退避時に現在選択されているバンクアドレス(CBの内容)の退避も行われます。また、MODEL2/3では、分岐時にNBに設定されているバンクアドレスがCBにロードされ、バンクの変更も行われます。

コード

MSB							LSB	
1	1	1	1	0	0	1	0	F2H
				r	r			rr
				q	q			qq

フラグ

I1	I0	U	D	N	V	C	Z
−	−	−	−	−	−	−	−

モード 符号付き16ビットPC相対

例 MODEL2/3においてNB=02Hの場合、物理アドレス9000H番地にあるCARL $+2000H命令を実行。

	NB	CB	PC(論理アドレス)	SP
実行前	02H	01H	9000H	0000H
			9002H+(2000H-2)	
実行後	02H	02H	B000H	FFFDH

上記の例では物理アドレス013000Hへ分岐します。MODEL0/1ではNB, CBがないため物理アドレスB000Hへ分岐します。

実行後のスタック内容

(1) MODEL2/3(マキシマムモード)

00FFFDH	03H (PC(L))
00FFFEH	90H (PC(H))
00FFFFH	01H (CB)

(2) MODEL2/3(ミニマムモード) MODEL0/1

00FFFEH	03H (PC(L))
00FFFFH	90H (PC(H))

CARL cc1, qqrr //// Call subroutine at relative location qqrr if condition cc1 is true ////////////

機 能 ■MODEL0/1

If cc1 is true then CARL qqrr
else PC←PC+3

■MODEL2/3

If cc1 is true then CARL qqrr
else PC←PC+3, NB←CB

条件cc1が成立している場合に"CARL qqrr"命令を実行し、条件不成立の場合は次の命令を実行します。
☞ **"CARL qqrr"命令**

MODEL2/3では、分岐先バンクを指定するNBの内容が条件不成立の場合に現在のバンクアドレス(CBの内容)に戻されます。

条件cc1は以下の4種類です。

cc1	条 件	
C	Carry	(キャリーフラグC=1)
NC	Non Carry	(キャリーフラグC=0)
Z	Zero	(ゼロフラグZ=1)
NZ	Non Zero	(ゼロフラグZ=0)

バスサイクルはミニマムモード/マキシマムモード、条件成立/不成立により以下のようになります。

モード	条件	バスサイクル
ミニマム	成立	5 cycle
ミニマム	不成立	3 cycle
マキシマム	成立	6 cycle
マキシマム	不成立	3 cycle

コード

MSB							LSB	
1	1	1	0	1	0	cc1		E8H～EBH ※
				r	r			rr
				q	q			qq

※

cc1		ニーモニック	コード
C	00	CARL C,qqrr	E8H
NC	01	CARL NC,qqrr	E9H
Z	10	CARL Z,qqrr	EAH
NZ	11	CARL NZ,qqrr	EBH

フラグ

I1	I0	U	D	N	V	C	Z
−	−	−	−	−	−	−	−

モード 符号付き16ビットPC相対

例 条件成立時はCARL qqrr命令と同じ動作をします。条件不成立時、物理アドレス9000H番地のCARL cc1, qqrrの動作は下記のとおりです。スタックは行いません。

	NB	CB	PC(論理アドレス)	SP
実行前	02H	01H	9000H	0000H
実行後	01H	01H	9003H	0000H

MODEL0/1ではNB, CBはありません。

CARS rr /////////////////////// Call subroutine at relative location rr ////////////////// 4(MIN)/5(MAX) cycle ///

機 能 ■MODEL0/1

[SP-1]←PC(H), [SP-2]←PC(L), SP←SP-2, PC←PC+rr+1

■MODEL2/3(ミニマムモード)

[SP-1]←PC(H), [SP-2]←PC(L), SP←SP-2, PC←PC+rr+1, CB←NB

■MODEL2/3(マキシマムモード)

[SP-1]←CB, [SP-2]←PC(H), [SP-3]←PC(L), SP←SP-3, PC←PC+rr+1, CB←NB

本命令の先頭アドレス+2の値をリターンアドレスとしてスタックに退避後、サブルーチンを無条件にコールします。分岐先(サブルーチンの先頭アドレス)は、本命令の先頭アドレス+1に符号付き8ビット相対アドレスrr(-128～127)を加算したアドレスになります。

MODEL2/3のマキシマムモードでは、リターンアドレスの退避時に現在選択されているバンクアドレス(CBの内容)の退避も行われます。また、MODEL2/3では、分岐時にNBに設定されているバンクアドレスがCBにロードされ、バンクの変更も行われます。

コード

MSB							LSB	
1	1	1	1	0	0	0	0	F0H
r	r							rr

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

モード 符号付き8ビットPC相対

例 MODEL2/3においてNB=02Hの場合、物理アドレス9000H番地にあるCARS $+20H命令を実行。

	NB	CB	PC(論理アドレス)	SP
実行前	02H	01H	9000H	0000H
			9001H+(20H-1)	
実行後	02H	02H	9020H	FFFDH

上記の例では物理アドレス011020Hへ分岐します。MODEL0/1ではNB, CBがないため物理アドレス9020Hへ分岐します。

実行後のスタック内容

(1) MODEL2/3(マキシマムモード)

00FFFDH	02H (PC(L))
00FFFEH	90H (PC(H))
00FFFFH	01H (CB)

(2) MODEL2/3(ミニマムモード) MODEL0/1

00FFFEH	02H (PC(L))
00FFFFH	90H (PC(H))

CARS cc1, rr //////////// Call subroutine at relative location rr if condition cc1 is true ////////////////

機 能 ■MODEL0/1

If cc1 is true then CARS rr
else PC←PC+2

■MODEL2/3

If cc1 is true then CARS rr
else PC←PC+2, NB←CB

条件cc1が成立している場合に"CARS rr"命令を実行し、条件不成立の場合は次の命令を実行します。
☞ **"CARS rr"命令**

MODEL2/3のマキシマムモードでは、分岐先バンクを指定するNBの内容が条件不成立の場合に現在のバンクアドレス(CBの内容)に戻されます。

条件cc1は以下の4種類です。

cc1	条 件	
C	Carry	(キャリーフラグC=1)
NC	Non Carry	(キャリーフラグC=0)
Z	Zero	(ゼロフラグZ=1)
NZ	Non Zero	(ゼロフラグZ=0)

バスサイクルはミニマムモード/マキシマムモード、条件成立/不成立により以下のようになります。

モード	条件	バスサイクル
ミニマム	成立	4 cycle
ミニマム	不成立	2 cycle
マキシマム	成立	5 cycle
マキシマム	不成立	2 cycle

コード

MSB							LSB	
1	1	1	0	0	0	cc1		E0H～E3H ※
r	r							rr

※

cc1		ニーモニック	コード
C	00	CARS C,rr	E0H
NC	01	CARS NC,rr	E1H
Z	10	CARS Z,rr	E2H
NZ	11	CARS NZ,rr	E3H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

モード 符号付き8ビットPC相対

例 条件成立時はCARS rr命令と同じ動作をします。条件不成立時、物理アドレス9000H番地のCARS cc1, rrの動作は下記のとおりです。スタックは行いません。

	NB	CB	PC(論理アドレス)	SP
実行前	02H	01H	9000H	0000H
実行後	01H	01H	9002H	0000H

MODEL0/1ではNB, CBはありません。

CARS cc2, rr ////////// *Call subroutine at relative location rr if condition cc2 is true* //////////

機 能 **■MODEL0/1**

If cc2 is true
then [SP-1]←PC(H), [SP-2]←PC(L),
SP←SP-2, PC←PC+rr+2
else PC←PC+3

■MODEL2/3(ミニマムモード)

If cc2 is true
then [SP-1]←PC(H), [SP-2]←PC(L),
SP←SP-2, PC←PC+rr+2,
CB←NB
else PC←PC+3, NB←CB

■MODEL2/3(マキシマムモード)

If cc2 is true
then [SP-1]←CB, [SP-2]←PC(H),
[SP-3]←PC(L), SP←SP-3,
PC←PC+rr+2, CB←NB
else PC←PC+3, NB←CB

条件cc2が成立している場合に本命令の先頭アドレス+3の値をリターンアドレスとしてスタックに退避後、サブルーチンをコールします。分岐先(サブルーチンの先頭アドレス)は、本命令の先頭アドレス+2に符号付き8ビット相対アドレスrr(-128～127)を加算したアドレスになります。

条件不成立の場合は次の命令を実行します。

MODEL2/3のマキシマムモードでは、リターンアドレスの退避時に現在選択されているバンクアドレス(CBの内容)の退避も行われます。また、MODEL2/3では、分岐時にNBに設定されているバンクアドレスがCBにロードされ、バンクの変更も行われます。
条件不成立の場合には、NBの内容が現在のバンクアドレス(CBの内容)に戻されます。

条件cc2は以下の16種類です。

cc2	条 件	
LT	Less Than	([N∀V]=1)
LE	Less or Equal	(Z∨[N∀V]=1)
GT	Greater Than	(Z∨[N∀V]=0)
GE	Greater or Equal	([N∀V]=0)
V	Overflow	(V=1)
NV	Non Overflow	(V=0)
P	Plus	(N=0)
M	Minus	(N=1)
F0	F0 is set	(F0=1)
F1	F1 is set	(F1=1)
F2	F2 is set	(F2=1)
F3	F3 is set	(F3=1)
NF0	F0 is reset	(F0=0)
NF1	F1 is reset	(F1=0)
NF2	F2 is reset	(F2=0)
NF3	F3 is reset	(F3=0)

バスサイクルはミニマムモード/マキシマムモード、条件成立/不成立により以下のようになります。

モード	条件	バスサイクル
ミニマム	成立	5 cycle
ミニマム	不成立	3 cycle
マキシマム	成立	6 cycle
マキシマム	不成立	3 cycle

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	1	1	1	cc2				F0H～FFH ※
r	r							rr

※

cc2		ニーモニック	コード
LT	0000	CARS LT,rr	F0H
LE	0001	CARS LE,rr	F1H
GT	0010	CARS GT,rr	F2H
GE	0011	CARS GE,rr	F3H
V	0100	CARS V,rr	F4H
NV	0101	CARS NV,rr	F5H
P	0110	CARS P,rr	F6H
M	0111	CARS M,rr	F7H
F0	1000	CARS F0,rr	F8H
F1	1001	CARS F1,rr	F9H
F2	1010	CARS F2,rr	FAH
F3	1011	CARS F3,rr	FBH
NF0	1100	CARS NF0,rr	FCH
NF1	1101	CARS NF1,rr	FDH
NF2	1110	CARS NF2,rr	FEH
NF3	1111	CARS NF3,rr	FFH

フラグ

I1	I0	U	D	N	V	C	Z
−	−	−	−	−	−	−	−

モード **符号付き8ビットPC相対**

例 MODEL2/3においてNB=02Hの場合、条件成立時の物理アドレス9000H番地にあるCARS cc2, $+20Hの動作は下記のとおりです。

	NB	CB	PC(論理アドレス)	SP
実行前	02H	01H	9000H	0000H
			9002H+(20H-2)	
実行後	02H	02H	9020H	FFFDH

上記の例では物理アドレス011020Hへ分岐します。MODEL0/1ではNB, CBがないため物理アドレス9020Hへ分岐します。

条件不成立時、物理アドレス9000H番地のCARS cc2, $+20Hの動作は下記のとおりです。スタックは行いません。

	NB	CB	PC(論理アドレス)	SP
実行前	02H	01H	9000H	0000H
実行後	01H	01H	9003H	0000H

MODEL0/1ではNB, CBはありません。

実行後のスタック内容

(1) MODEL2/3(マキシマムモード)

00FFFDH	03H (PC(L))
00FFFEH	90H (PC(H))
00FFFFH	01H (CB)

(2) MODEL2/3(ミニマムモード) MODEL0/1

00FFFEH	03H (PC(L))
00FFFFH	90H (PC(H))

CP A, r ////////////////////////// Compare r reg. with A reg. // 2 cycle ///

機 能 A - r

Aレジスタの内容からrレジスタ(A/B)の内容を減じ、その結果によりフラグ(N/V/C/Z)の内容を変更します。Aレジスタの内容は変更されません。

モード Src: レジスタ直接
Dst: レジスタ直接

例

設定値		結果				
A	B	A	SC			
			N	V	C	Z
74H	2AH	74H	0	0	0	0
1DH	1DH	1DH	0	0	0	1
3CH	59H	3CH	1	0	1	0
C3H	62H	C3H	0	1	0	0

コード

MSB							LSB	
0	0	1	1	0	0	0	r	30H、31H ※

※

r		ニーモニック	コード
A	0	CP A, A	30H
B	1	CP A, B	31H

フラグ

I1	I0	U	D	N	V	C	Z
−	−	−	−	↕	↕	↕	↕

CP A, #nn ////////////////////// Compare immediate data nn with A reg. ////////////////////////////////// 2 cycle ///

機 能 A - nn

Aレジスタの内容から8ビット即値データnnを減じ、その結果によりフラグ(N/V/C/Z)の内容を変更します。Aレジスタの内容は変更されません。

モード Src: 即値データ
Dst: レジスタ直接

例

設定値		結果				
A	nn	A	SC			
			N	V	C	Z
74H	2AH	74H	0	0	0	0
1DH	1DH	1DH	0	0	0	1
3CH	59H	3CH	1	0	1	0
C3H	62H	C3H	0	1	0	0

コード

MSB							LSB	
0	0	1	1	0	0	1	0	32H
			n	n				nn

フラグ

I1	I0	U	D	N	V	C	Z
−	−	−	−	↕	↕	↕	↕

CP A, [BR:ll] //////////// Compare location [BR:ll] with A reg. ////////////////////////////////////// 3 cycle ///

機 能 A - [BR:*ll*]

Aレジスタの内容からBRレジスタの内容を上位バイト、8ビット絶対アドレス*ll*を下位バイトとしてアドレス指定されるデータメモリの内容を減じ、その結果によりフラグ(N/V/C/Z)の内容を変更します。Aレジスタの内容は変更されません。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: 8ビット絶対
Dst: レジスタ直接

例

設定値		結果				
A	[BR:*ll*]	A	SC			
			N	V	C	Z
74H	2AH	74H	0	0	0	0
1DH	1DH	1DH	0	0	0	1
3CH	59H	3CH	1	0	1	0
C3H	62H	C3H	0	1	0	0

コード

MSB							LSB	
0	0	1	1	0	1	0	0	34H
			l	*l*				*ll*

フラグ

I1	I0	U	D	N	V	C	Z
−	−	−	−	↕	↕	↕	↕

CP A, [hhll] /////////////// Compare location [hhll] with A reg. ////////////////////////////////////// 4 cycle ///

機 能 A - [hh*ll*]

Aレジスタの内容から16ビット絶対アドレスhh*ll*でアドレス指定されたデータメモリの内容を減じ、その結果によりフラグ(N/V/C/Z)の内容を変更します。Aレジスタの内容は変更されません。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: 16ビット絶対
Dst: レジスタ直接

例

設定値		結果				
A	[hh*ll*]	A	SC			
			N	V	C	Z
74H	2AH	74H	0	0	0	0
1DH	1DH	1DH	0	0	0	1
3CH	59H	3CH	1	0	1	0
C3H	62H	C3H	0	1	0	0

コード

MSB							LSB	
0	0	1	1	0	1	0	1	35H
				l	*l*			*ll*
				h	h			hh

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	↕	↕	↕

CP A, [HL] ////////////////// Compare location [HL] with A reg. ////////////////////////////////////// 2 cycle ///

機 能 A - [HL]

Aレジスタの内容からHLレジスタでアドレス指定されたデータメモリの内容を減じ、その結果によりフラグ(N/V/C/Z)の内容を変更します。Aレジスタの内容は変更されません。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: レジスタ間接
Dst: レジスタ直接

例

設定値		結果				
A	[HL]	A	SC			
			N	V	C	Z
74H	2AH	74H	0	0	0	0
1DH	1DH	1DH	0	0	0	1
3CH	59H	3CH	1	0	1	0
C3H	62H	C3H	0	1	0	0

コード

MSB							LSB	
0	0	1	1	0	0	1	1	33H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	↕	↕	↕

CP A, [ir] ////////////////////// Compare location [ir reg.] with A reg. ////////////////////////////////////// 2 cycle ///

機 能 A - [ir]

Aレジスタの内容からirレジスタ(IX/IY)でアドレス指定されたデータメモリの内容を減じ、その結果によりフラグ(N/V/C/Z)の内容を変更します。Aレジスタの内容は変更されません。
XPレジスタ(IX指定時)、YPレジスタ(IY指定時)の内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: レジスタ間接
Dst: レジスタ直接

例

設定値		結果				
A	[ir]	A	SC			
			N	V	C	Z
74H	2AH	74H	0	0	0	0
1DH	1DH	1DH	0	0	0	1
3CH	59H	3CH	1	0	1	0
C3H	62H	C3H	0	1	0	0

コード

MSB							LSB	
0	0	1	1	0	1	1	ir	36H、37H ※

※

ir		ニーモニック	コード
IX	0	CP A, [IX]	36H
IY	1	CP A, [IY]	37H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	↕	↕	↕

CP A, [ir+dd] ////////// *Compare location [ir reg. + dd] with A reg.* ////////////////////// *4 cycle* ///

機 能 A - [ir+dd]

Aレジスタの内容から、irレジスタ(IX/IY)の内容とディスプレースメントddの和でアドレス指定されたデータメモリの内容を減じ、その結果によりフラグ(N/V/C/Z)の内容を変更します。Aレジスタの内容は変更されません。

ddは符号付きデータとして扱われ、範囲は-128～127です。

XPレジスタ(IX指定時)、YPレジスタ(IY指定時)の内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: ディスプレースメント付きレジスタ間接
Dst: レジスタ直接

例

設定値		結果				
A	[ir+dd]	A	SC			
			N	V	C	Z
74H	2AH	74H	0	0	0	0
1DH	1DH	1DH	0	0	0	1
3CH	59H	3CH	1	0	1	0
C3H	62H	C3H	0	1	0	0

コード

MSB LSB

1	1	0	0	1	1	1	0	CEH

0	0	1	1	0	0	0	ir	30H、31H※

d d	dd

※

ir		ニーモニック	コード
IX	0	CP A, [IX+dd]	30H
IY	1	CP A, [IY+dd]	31H

フラグ

I1	I0	U	D	N	V	C	Z
−	−	−	−	↕	↕	↕	↕

CP A, [ir+L] ////////////// *Compare location [ir reg. + L] and A reg.* ////////////////////////////// *4 cycle* ///

機 能 A - [ir+L]

Aレジスタの内容から、irレジスタ(IX/IY)の内容とLレジスタの内容の和でアドレス指定されたデータメモリの内容を減じ、その結果によりフラグ(N/V/C/Z)の内容を変更します。Aレジスタの内容は変更されません。

Lレジスタの内容は符号付きデータとして扱われ、範囲は-128～127です。

XPレジスタ(IX指定時)、YPレジスタ(IY指定時)の内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: インデックスレジスタ付きレジスタ間接
Dst: レジスタ直接

例

設定値		結果				
A	[ir+L]	A	SC			
			N	V	C	Z
74H	2AH	74H	0	0	0	0
1DH	1DH	1DH	0	0	0	1
3CH	59H	3CH	1	0	1	0
C3H	62H	C3H	0	1	0	0

コード

MSB LSB

1	1	0	0	1	1	1	0	CEH

0	0	1	1	0	0	1	ir	32H、33H ※

※

ir		ニーモニック	コード
IX	0	CP A, [IX+L]	32H
IY	1	CP A, [IY+L]	33H

フラグ

I1	I0	U	D	N	V	C	Z
−	−	−	−	↕	↕	↕	↕

CP B, #nn /////////////////// Compare immediate data nn with B reg. ////////////////////////////////// 3 cycle ///

機 能 B - nn

Bレジスタの内容から、8ビット即値データnnを減じ、その結果によりフラグ(N/V/C/Z)の内容を変更します。Bレジスタの内容は変更されません。

モード Src: 即値データ
Dst: レジスタ直接

例

設定値		結果				
B	nn	B	SC			
			N	V	C	Z
74H	2AH	74H	0	0	0	0
1DH	1DH	1DH	0	0	0	1
3CH	59H	3CH	1	0	1	0
C3H	62H	C3H	0	1	0	0

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	1	1	1	1	0	0	BCH
				n	n			nn

フラグ

I1	I0	U	D	N	V	C	Z
−	−	−	−	↕	↕	↕	↕

CP L, #nn /////////////////// Compare immediate data nn with L reg. ////////////////////////////////// 3 cycle ///

機 能 L - nn

Lレジスタの内容から、8ビット即値データnnを減じ、その結果によりフラグ(N/V/C/Z)の内容を変更します。Lレジスタの内容は変更されません。

モード Src: 即値データ
Dst: レジスタ直接

例

設定値		結果				
L	nn	L	SC			
			N	V	C	Z
74H	2AH	74H	0	0	0	0
1DH	1DH	1DH	0	0	0	1
3CH	59H	3CH	1	0	1	0
C3H	62H	C3H	0	1	0	0

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	1	1	1	1	0	1	BDH
				n	n			nn

フラグ

I1	I0	U	D	N	V	C	Z
−	−	−	−	↕	↕	↕	↕

CP H, #nn /////////////////// Compare immediate data nn with H reg. ////////////////////////////////// 3 cycle ///

機 能 H - nn

Hレジスタの内容から、8ビット即値データnnを減じ、その結果によりフラグ(N/V/C/Z)の内容を変更します。Hレジスタの内容は変更されません。

モード Src: 即値データ
Dst: レジスタ直接

例

設定値		結果				
H	nn	H	SC			
			N	V	C	Z
74H	2AH	74H	0	0	0	0
1DH	1DH	1DH	0	0	0	1
3CH	59H	3CH	1	0	1	0
C3H	62H	C3H	0	1	0	0

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	1	1	1	1	1	0	BEH
				n	n			nn

フラグ

I1	I0	U	D	N	V	C	Z
−	−	−	−	↕	↕	↕	↕

CP BR, #hh ////////////// Compare immediate data hh with BR reg. ////////////////////////////// 3 cycle ///

機 能 **BR - hh**

BRレジスタの内容から、8ビット即値データhhを減じ、その結果によりフラグ(N/V/C/Z)の内容を変更します。BRレジスタの内容は変更されません。

モード **Src: 即値データ**
Dst: レジスタ直接

例

設定値		結果				
BR	hh	BR	SC			
			N	V	C	Z
74H	2AH	74H	0	0	0	0
1DH	1DH	1DH	0	0	0	1
3CH	59H	3CH	1	0	1	0
C3H	62H	C3H	0	1	0	0

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	1	1	1	1	1	1	BFH
h	h							hh

フラグ

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	↕	↕	↕

CP [BR:ll], #nn ////// Compare immediate data nn with location [BR:ll] ////////////////// 4 cycle ///

機 能 **[BR:*ll*] - nn**

BRレジスタの内容を上位バイト、8ビット絶対アドレス*ll*を下位バイトとしてアドレス指定されるデータメモリの内容から8ビット即値データnnを減じ、その結果によりフラグ(N/V/C/Z)の内容を変更します。データメモリの内容は変更されません。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード **Src: 即値データ**
Dst: 8ビット絶対

例

設定値		結果				
[BR:*ll*]	nn	[BR:*ll*]	SC			
			N	V	C	Z
74H	2AH	74H	0	0	0	0
1DH	1DH	1DH	0	0	0	1
3CH	59H	3CH	1	0	1	0
C3H	62H	C3H	0	1	0	0

コード

MSB							LSB	
1	1	0	1	1	0	1	1	DBH
l	*l*							*ll*
n	n							nn

フラグ

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	↕	↕	↕

CP [HL], A ////////////////// Compare A reg. with location [HL] ////////////////////////////////////// 3 cycle ///

機 能 **[HL] - A**

HLレジスタでアドレス指定されたデータメモリの内容からAレジスタの内容を減じ、その結果によりフラグ(N/V/C/Z)の内容を変更します。データメモリの内容は変更されません。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード **Src: レジスタ直接**
Dst: レジスタ間接

例

設定値		結果				
[HL]	A	[HL]	SC			
			N	V	C	Z
74H	2AH	74H	0	0	0	0
1DH	1DH	1DH	0	0	0	1
3CH	59H	3CH	1	0	1	0
C3H	62H	C3H	0	1	0	0

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	1	1	0	1	0	0	34H

フラグ

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	↕	↕	↕

CP [HL], #nn /////////// Compare immediate data nn with location [HL] ////////////////////// 4 cycle ///

機 能 [HL] - nn

HLレジスタでアドレス指定されたデータメモリの内容から8ビット即値データnnを減じ、その結果によりフラグ(N/V/C/Z)の内容を変更します。データメモリの内容は変更されません。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: 即値データ
Dst: レジスタ間接

例

設定値		結果				
[HL]	nn	[HL]	SC			
			N	V	C	Z
74H	2AH	74H	0	0	0	0
1DH	1DH	1DH	0	0	0	1
3CH	59H	3CH	1	0	1	0
C3H	62H	C3H	0	1	0	0

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	1	1	0	1	0	1	35H
n	n							nn

フラグ

I1	I0	U	D	N	V	C	Z
−	−	−	−	↕	↕	↕	↕

CP [HL], [ir] ////////////// Compare location [ir reg.] with location [HL] ////////////////////// 4 cycle ///

機 能 [HL] - [ir]

HLレジスタでアドレス指定されたデータメモリの内容からirレジスタ(IX/IY)でアドレス指定されるデータメモリの内容を減じ、その結果によりフラグ(N/V/C/Z)の内容を変更します。データメモリの内容は変更されません。
EPレジスタの内容がデータメモリ[HL]のページアドレス、XPレジスタ(IX指定時)、YPレジスタ(IY指定時)の内容がデータメモリ[ir]のページアドレスになります(MODEL2/3)。

モード Src: レジスタ間接
Dst: レジスタ間接

例

設定値		結果				
[HL]	[ir]	[HL]	SC			
			N	V	C	Z
74H	2AH	74H	0	0	0	0
1DH	1DH	1DH	0	0	0	1
3CH	59H	3CH	1	0	1	0
C3H	62H	C3H	0	1	0	0

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	1	1	0	1	1	ir	36H、37H ※

※

ir		ニーモニック	コード
IX	0	CP [HL], [IX]	36H
IY	1	CP [HL], [IY]	37H

フラグ

I1	I0	U	D	N	V	C	Z
−	−	−	−	↕	↕	↕	↕

CP BA, rp ////////////////// *Compare rp reg. with BA reg.* // *4 cycle* ///

機 能 **BA - rp**

BAレジスタの内容からrpレジスタ(BA/HL/IX/IY)の内容を減じ、その結果によりフラグ(N/V/C/Z)の内容を変更します。BAレジスタの内容は変更されません。

モード **Src: レジスタ直接**
Dst: レジスタ直接

例

設定値		結果				
BA	rp	BA	SC			
			N	V	C	Z
3F71H	145AH	3F71H	0	0	0	0
53D1H	53D1H	53D1H	0	0	0	1
A291H	632EH	A291H	0	1	0	0
2862H	4C25H	2862H	1	0	1	0

(rp≠BA)

コード

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
0	0	0	1	1	0	rp		18H〜1BH ※

※

rp		ニーモニック	コード
BA	00	CP BA, BA	18H
HL	01	CP BA, HL	19H
IX	10	CP BA, IX	1AH
IY	11	CP BA, IY	1BH

フラグ

I1	I0	U	D	N	V	C	Z
−	−	−	−	↕	↕	↕	↕

CP BA, #mmnn ////// *Compare immediate data mmnn with BA reg.* ////////////////////////// *3 cycle* ///

機 能 **BA - mmnn**

BAレジスタの内容から16ビット即値データmmnnを減じ、その結果によりフラグ(N/V/C/Z)の内容を変更します。BAレジスタの内容は変更されません。

モード **Src: 即値データ**
Dst: レジスタ直接

例

設定値		結果				
BA	mmnn	BA	SC			
			N	V	C	Z
3F71H	145AH	3F71H	0	0	0	0
53D1H	53D1H	53D1H	0	0	0	1
A291H	632EH	A291H	0	1	0	0
2862H	4C25H	2862H	1	0	1	0

コード

MSB							LSB	
1	1	0	1	0	1	0	0	D4H
				n	n			nn
				m	m			mm

フラグ

I1	I0	U	D	N	V	C	Z
−	−	−	−	↕	↕	↕	↕

CP HL, rp //////////////////// *Compare rp reg. with HL reg.* // *4 cycle* ///

機 能 **HL - rp**

HLレジスタの内容からrpレジスタ(BA/HL/IX/IY)の内容を減じ、その結果によりフラグ(N/V/C/Z)の内容を変更します。HLレジスタの内容は変更されません。

モード **Src: レジスタ直接**
Dst: レジスタ直接

例

設定値		結果				
HL	rp	HL	SC			
			N	V	C	Z
3F71H	145AH	3F71H	0	0	0	0
53D1H	53D1H	53D1H	0	0	0	1
A291H	632EH	A291H	0	1	0	0
2862H	4C25H	2862H	1	0	1	0

(rp≠HL)

コード MSB LSB

1	1	0	0	1	1	1	1	CFH

0	0	1	1	1	0	rp		38H〜3BH ※

※

rp		ニーモニック	コード
BA	00	CP HL, BA	38H
HL	01	CP HL, HL	39H
IX	10	CP HL, IX	3AH
IY	11	CP HL, IY	3BH

フラグ

I1	I0	U	D	N	V	C	Z
−	−	−	−	↕	↕	↕	↕

CP HL, #mmnn /////// *Compare immediate data mmnn with HL reg.* ////////////////////////// *3 cycle* ///

機 能 **HL - mmnn**

HLレジスタの内容から16ビット即値データmmnnを減じ、その結果によりフラグ(N/V/C/Z)の内容を変更します。HLレジスタの内容は変更されません。

モード **Src: 即値データ**
Dst: レジスタ直接

例

設定値		結果				
HL	mmnn	HL	SC			
			N	V	C	Z
3F71H	145AH	3F71H	0	0	0	0
53D1H	53D1H	53D1H	0	0	0	1
A291H	632EH	A291H	0	1	0	0
2862H	4C25H	2862H	1	0	1	0

コード MSB LSB

1	1	0	1	0	1	0	1	D5H

n	n	nn

m	m	mm

フラグ

I1	I0	U	D	N	V	C	Z
−	−	−	−	↕	↕	↕	↕

CP IX, #mmnn ////////// Compare immediate data mmnn with IX reg. ////////////////////////// 3 cycle ///

機 能 IX - mmnn

IXレジスタの内容から16ビット即値データmmnnを減じ、その結果によりフラグ(N/V/C/Z)の内容を変更します。IXレジスタの内容は変更されません。

モード Src: 即値データ
Dst: レジスタ直接

例

設定値		結果				
IX	mmnn	IX	SC			
			N	V	C	Z
3F71H	145AH	3F71H	0	0	0	0
53D1H	53D1H	53D1H	0	0	0	1
A291H	632EH	A291H	0	1	0	0
2862H	4C25H	2862H	1	0	1	0

コード

MSB							LSB	
1	1	0	1	0	1	1	0	D6H
				n	n			nn
				m	m			mm

フラグ

I1	I0	U	D	N	V	C	Z
−	−	−	−	↕	↕	↕	↕

CP IY, #mmnn ////////// Compare immediate data mmnn with IY reg. ////////////////////////// 3 cycle ///

機 能 IY - mmnn

IYレジスタの内容から16ビット即値データmmnnを減じ、その結果によりフラグ(N/V/C/Z)の内容を変更します。IYレジスタの内容は変更されません。

モード Src: 即値データ
Dst: レジスタ直接

例

設定値		結果				
IY	mmnn	IY	SC			
			N	V	C	Z
3F71H	145AH	3F71H	0	0	0	0
53D1H	53D1H	53D1H	0	0	0	1
A291H	632EH	A291H	0	1	0	0
2862H	4C25H	2862H	1	0	1	0

コード

MSB							LSB	
1	1	0	1	0	1	1	1	D7H
				n	n			nn
				m	m			mm

フラグ

I1	I0	U	D	N	V	C	Z
−	−	−	−	↕	↕	↕	↕

CP SP, rp //////////////////// Compare rp reg. with SP // 4 cycle ///

機 能 SP - rp

スタックポインタ(SP)の内容からrpレジスタ(BA/HL)の内容を減じ、その結果によりフラグ(N/V/C/Z)の内容を変更します。SPの内容は変更されません。

モード Src: レジスタ直接
Dst: レジスタ直接

例

設定値		結果				
SP	rp	SP	SC			
			N	V	C	Z
3F71H	145AH	3F71H	0	0	0	0
53D1H	53D1H	53D1H	0	0	0	1
A291H	632EH	A291H	0	1	0	0
2862H	4C25H	2862H	1	0	1	0

コード

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
0	1	0	1	1	1	0	rp	5CH、5DH ※

※

rp		ニーモニック	コード
BA	0	CP SP, BA	5CH
HL	1	CP SP, HL	5DH

フラグ

I1	I0	U	D	N	V	C	Z
−	−	−	−	↕	↕	↕	↕

CP SP, #mmnn //////// Compare immediate data mmnn with SP //////////////////////////////// 4 cycle ///

機 能 SP - mmnn

スタックポインタ(SP)の内容から16ビット即値データmmnnを減じ、その結果によりフラグ(N/V/C/Z)の内容を変更します。SPの内容は変更されません。

モード Src: 即値データ
Dst: レジスタ直接

例

設定値		結果				
SP	mmnn	SP	SC			
			N	V	C	Z
3F71H	145AH	3F71H	0	0	0	0
53D1H	53D1H	53D1H	0	0	0	1
A291H	632EH	A291H	0	1	0	0
2862H	4C25H	2862H	1	0	1	0

コード

MSB LSB

1	1	0	0	1	1	1	1	CFH
0	1	1	0	1	1	0	0	6CH
				n	n			nn
				m	m			mm

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	↕	↕	↕

CPL r ////////////////////////////// Complement r reg. // 3 cycle ///

機 能 r ← r̄

rレジスタ(A/B)の各ビットを反転し、1の補数を作成します。

モード レジスタ直接

例

設定値	結果				
r	r	SC			
		N	V	C	Z
11111111	00000000	0	—	—	1
10100101	01011010	1	—	—	0

コード

MSB LSB

1	1	0	0	1	1	1	0	CEH
1	0	1	0	0	0	0	r	A0H、A1H ※

※

r		ニーモニック	コード
A	0	CPL A	A0H
B	1	CPL B	A1H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	—	↕

CPL [BR:ll] ////////////// Complement location [BR:ll] // 5 cycle ///

機 能 [BR:*ll*] ← [BR:*ll*]

BRレジスタの内容を上位バイト、8ビット絶対アドレス*ll*を下位バイトとしてアドレス指定されるデータメモリの各ビットを反転し、1の補数を作成します。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード 8ビット絶対

例

設定値	結果				
[BR:*ll*]	[BR:*ll*]	SC			
		N	V	C	Z
11111111	00000000	0	—	—	1
10100101	01011010	1	—	—	0

コード

MSB LSB

1	1	0	0	1	1	1	0	CEH
1	0	1	0	0	0	1	0	A2H
				l	*l*			*ll*

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	—	↕

CPL [HL] ////////////////////// Complement location [HL] // 4 cycle ///

機 能 [HL] ← [HL]

HLレジスタでアドレス指定されたデータメモリの各ビットを反転し、1の補数を作成します。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード レジスタ間接

例

設定値	結果				
[HL]	[HL]	SC			
		N	V	C	Z
11111111	00000000	0	—	—	1
10100101	01011010	1	—	—	0

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	1	0	0	0	1	1	A3H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	—	↕

DEC r ////////////////////////////// Decrement r reg. // 2 cycle ///

機 能 r ← r - 1

rレジスタ(A/B/L/H)の内容をデクリメント(-1)します。

モード レジスタ直接

例

設定値	結果				
r	r	SC			
		N	V	C	Z
63H	62H	—	—	—	0
01H	00H	—	—	—	1

コード

MSB							LSB	
1	0	0	0	1	0	r		88H～8BH ※

※

r		ニーモニック	コード
A	00	DEC A	88H
B	01	DEC B	89H
L	10	DEC L	8AH
H	11	DEC H	8BH

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	↕

DEC BR ////////////////////// Decrement BR reg. // 2 cycle ///

機 能 BR ← BR - 1

BRレジスタの内容をデクリメント(-1)します。

モード レジスタ直接

例

設定値	結果				
BR	BR	SC			
		N	V	C	Z
63H	62H	—	—	—	0
01H	00H	—	—	—	1

コード

MSB							LSB	
1	0	0	0	1	1	0	0	8CH

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	↕

DEC [BR:ll] ////////////// Decrement location [BR:ll] // 4 cycle ///

機 能 [BR:*ll*] ← [BR:*ll*] - 1

BRレジスタの内容を上位バイト、8ビット絶対アドレス*ll*を下位バイトとしてアドレス指定されるデータメモリの内容をデクリメント(-1)します。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード 8ビット絶対

例

設定値		結果				
[BR:*ll*]	[BR:*ll*]	SC				
		N	V	C	Z
63H	62H	—	—	—	0
01H	00H	—	—	—	1

コード

MSB							LSB	
1	0	0	0	1	1	0	1	8DH
				l	*l*			*ll*

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	↕

DEC [HL] ////////////////////// Decrement location [HL] // 3 cycle ///

機 能 [HL] ← [HL] - 1

HLレジスタでアドレス指定されたデータメモリの内容をデクリメント(-1)します。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード レジスタ間接

例

設定値		結果				
[HL]	[HL]	SC				
		N	V	C	Z
63H	62H	—	—	—	0
01H	00H	—	—	—	1

コード

MSB							LSB	
1	0	0	0	1	1	1	0	8EH

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	↕

DEC rp ////////////////////////// Decrement rp reg. // 2 cycle ///

機 能 rp ← rp - 1

rpレジスタ(BA/HL/IX/IY)の内容をデクリメント(-1)します。

モード レジスタ直接

例

設定値		結果				
rp	rp	SC				
		N	V	C	Z
4285H	4284H	—	—	—	0
0001H	0000H	—	—	—	1

コード

MSB							LSB	
1	0	0	1	1	0	rp		98H～9BH ※

※

rp		ニーモニック	コード
BA	00	DEC BA	98H
HL	01	DEC HL	99H
IX	10	DEC IX	9AH
IY	11	DEC IY	9BH

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	↕

DEC SP ////////////////////// Decrement SP // 2 cycle ///

機 能 SP ← SP - 1

スタックポインタ(SP)の内容をデクリメント(-1)します。

コード

MSB							LSB	
1	0	0	0	1	1	1	1	8FH

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	↕

モード レジスタ直接

例

設定値	結果				
SP	SP	SC			
		N	V	C	Z
4285H	4284H	—	—	—	0
0001H	0000H	—	—	—	1

DIV ////////////////////////////////////// Divide HL reg. by A reg. // 13 cycle ///

機 能 L ← HL/A　H ← 剰余

HLレジスタの内容をAレジスタの内容で除算し、商をLレジスタに、剰余をHレジスタにストアします。除数'0'で除算を行うと、ゼロ除算例外処理が発生します。

除数'0'以外で除算を行い、商が8ビットを越えた場合はVフラグが'1'にセットされ、被除数(HLレジスタの内容)は保存されます。

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	1	0	1	1	0	0	1	D9H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	↕	0	↕

注 意 MODEL0/2では本命令が使用できません。

モード インプライド (レジスタ直接)

例

設定値		結果					
HL	A	L	H	SC			
				N	V	C	Z
1A16H	64H	42H	4EH	0	0	0	0
332CH	64H	83H	00H	1	0	0	0
0000H	58H	00H	00H	0	0	0	1
0301H	02H	01H	03H	1	1	0	0

DJR NZ, rr ////////////////// Decrement B reg. & Jump relative if B reg. is not zero ////////// 4 cycle ///

機 能 ■MODEL0/1

B ← B - 1,
If B ≠ 0 then JRS rr
else PC ← PC + 2

■MODEL2/3

B ← B - 1,
If B ≠ 0 then JRS rr
else PC ← PC + 2, NB ← CB

Bレジスタをデクリメント(-1)し、その結果、Bレジスタが'0'以外の場合に分岐命令"JRS rr"を実行します。Bレジスタが'0'になった場合には次の命令を実行します。

☞ "JRS rr"命令

MODEL2/3では、分岐時にNBに設定されているバンクアドレスがCBにロードされ、バンクの変更も行われます。分岐せずに次の命令に移行する際にはNBの内容を現在のバンクアドレス(CBの内容)に戻します。

コード

MSB							LSB	
1	1	1	1	0	1	0	1	F5H
				r	r			rr

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	↕

モード 符号付き8ビットPC相対

例 MODEL2/3においてNB=02H, B=05Hの場合、条件成立時の物理アドレス9000H番地にあるDJR NZ, $-05Hの動作は下記のとおりです。

	NB	CB	PC(論理アドレス)	B
実行前	02H	01H	9000H	05H
			9001H+(FFFBH-1)	
実行後	02H	02H	8FFBH	04H

上記の例では物理アドレス010FFBHへ分岐します。MODEL0/1ではNB, CBがないため物理アドレス8FFBHへ分岐します。

条件不成立時(B=01H)、物理アドレス9000H番地のDJR NZ, $-05Hの動作は下記のとおりです。

	NB	CB	PC(論理アドレス)	B
実行前	01H	02H	9000H	01H
実行後	02H	02H	9002H	00H

MODEL0/1ではNB, CBはありません。

EX A, B ////////////////////////// Exchange A reg. and B reg. // 2 cycle ///

機 能 A ↔ B

AレジスタとBレジスタの内容を交換します。

モード Src: レジスタ直接
Dst: レジスタ直接

コード

MSB							LSB	
1	1	0	0	1	1	0	0	CCH

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

例

設定値		結果					
A	B	A	B	SC			
				N	V	C	Z
82H	49H	49H	82H	—	—	—	—

EX A, [HL] ////////////////// Exchange A reg. and location [HL] ////////////////////////////////////// 3 cycle ///

機 能 A ↔ [HL]

AレジスタとHLレジスタでアドレス指定されたデータメモリの内容を交換します。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: レジスタ間接
Dst: レジスタ直接

コード

MSB							LSB	
1	1	0	0	1	1	0	1	CDH

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

例

設定値		結果					
A	[HL]	A	[HL]	SC			
				N	V	C	Z
82H	49H	49H	82H	—	—	—	—

EX BA, rp ////////////////// Exchange BA reg. and rp reg. // 3 cycle ///

機 能 BA ↔ rp

BAレジスタとrpレジスタ(HL/IX/IY/SP)の内容を交換します。

モード Src: レジスタ直接
Dst: レジスタ直接

コード

MSB							LSB	
1	1	0	0	1	0	rp		C8H〜CBH ※

※

rp		ニーモニック	コード
HL	00	EX BA, HL	C8H
IX	01	EX BA, IX	C9H
IY	10	EX BA, IY	CAH
SP	11	EX BA, SP	CBH

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

例

設定値		結果					
BA	rp	BA	rp	SC			
				N	V	C	Z
35D6H	C284H	C284H	35D6H	—	—	—	—

HALT ////////////////////////////////// Set CPU to HALT mode // 3 cycle ///

機 能 **HALT**

CPUをHALT状態にします。
HALT状態ではCPUが動作を停止し、消費電力を低減できます。発振回路などの周辺回路は動作します。
HALT状態からは割り込みにより通常のプログラム実行状態に戻ります。
☞ **"3.7.1 HALT状態"**

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	1	0	1	1	1	0	AEH

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

INC r ////////////////////////////////// Increment r reg. // 2 cycle ///

機 能 **r ← r + 1**

rレジスタ(A/B/L/H)の内容をインクリメント(+1)します。

コード

MSB							LSB	
1	0	0	0	0	0	r		80H～83H ※

※

r		ニーモニック	コード
A	00	INC A	80H
B	01	INC B	81H
L	10	INC L	82H
H	11	INC H	83H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	↕

モード **レジスタ直接**

例

設定値	結果				
r	r	SC			
		N	V	C	Z
52H	53H	—	—	—	0
FFH	00H	—	—	—	1

INC BR ////////////////////////////// Increment BR reg. // 2 cycle ///

機 能 **BR ← BR + 1**

BRレジスタの内容をインクリメント(+1)します。

コード

MSB							LSB	
1	0	0	0	0	1	0	0	84H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	↕

モード **レジスタ直接**

例

設定値	結果				
BR	BR	SC			
		N	V	C	Z
52H	53H	—	—	—	0
FFH	00H	—	—	—	1

INC [BR:ll] ////////////////// Increment location [BR:ll] // 4 cycle ///

機 能 [BR:*ll*] ← [BR:*ll*] + 1

BRレジスタの内容を上位バイト、8ビット絶対アドレス*ll*を下位バイトとしてアドレス指定されるデータメモリの内容をインクリメント(+1)します。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード 8ビット絶対

例

設定値		結果			
[BR:*ll*]	[BR:*ll*]	SC			
		N	V	C	Z
52H	53H	—	—	—	0
FFH	00H	—	—	—	1

コード

MSB							LSB	
1	0	0	0	0	1	0	1	85H
				l	*l*			*ll*

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	↕

INC [HL] ////////////////////// Increment location [HL] // 3 cycle ///

機 能 [HL] ← [HL] + 1

HLレジスタでアドレス指定されたデータメモリの内容をインクリメント(+1)します。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード レジスタ間接

例

設定値		結果			
[HL]	[HL]	SC			
		N	V	C	Z
52H	53H	—	—	—	0
FFH	00H	—	—	—	1

コード

MSB							LSB	
1	0	0	0	0	1	1	0	86H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	↕

INC rp ////////////////////////////// Increment rp reg. // 2 cycle ///

機 能 rp ← rp + 1

rpレジスタ(BA/HL/IX/IY)の内容をインクリメント(+1)します。

モード レジスタ直接

例

設定値		結果			
rp	rp	SC			
		N	V	C	Z
3259H	325AH	—	—	—	0
FFFFH	0000H	—	—	—	1

コード

MSB							LSB	
1	0	0	1	0	0	rp		90H～93H ※

※

rp		ニーモニック	コード
BA	00	INC BA	90H
HL	01	INC HL	91H
IX	10	INC IX	92H
IY	11	INC IY	93H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	↕

INC SP ////////////////////// Increment SP ////////////////////// 2 cycle ///

機 能 SP ← SP + 1

スタックポインタ(SP)の内容をインクリメント(+1)します。

コード

MSB							LSB	
1	0	0	0	0	1	1	1	87H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	↕

モード レジスタ直接

例

設定値	結果				
SP	SP	SC			
		N	V	C	Z
3259H	325AH	—	—	—	0
FFFFH	0000H	—	—	—	1

INT [kk] ////////////////////// Software Interrupt ////////////////////// 7(MIN)/8(MAX) cycle ///

機 能 ■MODEL0/1

[SP-1] ← PC(H), [SP-2] ← PC(L),
[SP-3] ← SC, SP ← SP-3, PC(L) ← [00kk],
PC(H) ← [00kk+1]

■MODEL2/3(ミニマムモード)

[SP-1] ← PC(H), [SP-2] ← PC(L),
[SP-3] ← SC, SP ← SP-3, PC(L) ← [00kk],
PC(H) ← [00kk+1], CB ← NB

■MODEL2/3(マキシマムモード)

[SP-1] ← CB, [SP-2] ← PC(H),
[SP-3] ← PC(L), [SP-4] ← SC, SP ← SP-4,
PC(L) ← [00kk], PC(H) ← [00kk+1], CB ← NB

本命令の先頭アドレス+2の値とシステムコンディションコード(SC)をスタックに退避後、プログラムメモリの00kk番地をベクタアドレスとするソフトウェア割り込みルーチンを実行します。

MODEL2/3のマキシマムモードでは、リターンアドレスの退避時に現在選択されているバンクアドレス(CBの内容)の退避も行われます。また、MODEL2/3では、分岐時にNBに設定されているバンクアドレスがCBにロードされ、バンクの変更も行われます。

ベクタ領域は0ページに固定です。

注 意 "INT [kk]"命令により実行される割り込みルーチンからのリターンにはSCの内容も復帰させる"RETE"命令を使用してください。

コード

MSB							LSB	
1	1	1	1	1	1	0	0	FCH
				k	k			kk

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

モード 8ビット間接

例 MODEL2/3においてNB=02Hの場合、物理アドレス9000H番地にあるINT [20H]命令を実行。

	NB	CB	PC(論理アドレス)	EP	M(000020H)	SP
実行前	02H	01H	9000H	03H	ABCDH	0000H
実行後	02H	02H	ABCDH	03H	ABCDH	FFFCH

EPは無視されベクタ領域(000000H～0000FFH)が指定されます。

上記の例では物理アドレス012BCDHへ分岐します。

MODEL0/1ではNB, CBがないため物理アドレスABCDHへ分岐します。

実行後のスタック内容

(1) MODEL2/3(マキシマムモード)

00FFFCH	SCの内容
00FFFDH	02H (PC(L))
00FFFEH	90H (PC(H))
00FFFFH	01H (CB)

(2) MODEL2/3(ミニマムモード) MODEL0/1

00FFFDH	SCの内容
00FFFEH	02H (PC(L))
00FFFFH	90H (PC(H))

JP HL ////////////////////////////// Indirect Jump using HL reg. // 2 cycle ///

機 能 ■MODEL0/1

PC ← HL

■MODEL2/3

PC ← HL, CB ← NB

HLレジスタの内容をプログラムカウンタ(PC)にロードして無条件に分岐します。
MODEL2/3では、分岐時にNBに設定されているバンクアドレスがCBにロードされ、バンクの変更も行われます。

コード

MSB							LSB	
1	1	1	1	0	1	0	0	F4H

フラグ

I1	I0	U	D	N	V	C	Z
−	−	−	−	−	−	−	−

モード レジスタ直接

例 MODEL2/3においてNB=02H, HL=8765Hの場合、物理アドレス9000H番地にあるJP HLの動作は下記のとおりです。

	NB	CB	PC(論理アドレス)	HL
実行前	02H	01H	9000H	8765H
実行後	02H	02H	8765H	8765H

上記の例では物理アドレス010765Hへ分岐します。
MODEL0/1ではNB, CBがないため物理アドレス8765Hへ分岐します。

JP [kk] ////////////////////////////// Indirect Jump using vector // 4 cycle ///

機 能 ■MODEL0/1

$PC_{(L)}$ ← [00kk], $PC_{(H)}$ ← [00kk+1]

■MODEL2/3

CB ← NB, $PC_{(L)}$ ← [00kk], $PC_{(H)}$ ← [00kk+1]

8ビット間接アドレスkkで示される、プログラムメモリの00kk〜00kk+1番地に書き込まれているベクタをプログラムカウンタ(PC)にロードして無条件に分岐します。
MODEL2/3では、分岐時にNBに設定されているバンクアドレスがCBにロードされ、バンクの変更も行われます。
ベクタ領域は0ページに固定です。

コード

MSB							LSB	
1	1	1	1	1	1	0	1	FDH
				k	k			kk

フラグ

I1	I0	U	D	N	V	C	Z
−	−	−	−	−	−	−	−

モード 8ビット間接

例 MODEL2/3においてNB=02Hの場合、物理アドレス9000H番地にあるJP [20H]命令を実行。

	NB	CB	PC(論理アドレス)	EP	M(000020H)
実行前	02H	01H	9000H	03H	ABCDH
実行後	02H	02H	ABCDH	03H	ABCDH

EPは無視されベクタ領域(000000H〜0000FFH)が指定されます。
上記の例では物理アドレス012BCDHへ分岐します。
MODEL0/1ではNB, CBがないため物理アドレスABCDHへ分岐します。

JRL qqrr ////////////////////// Jump to relative location qqrr // 3 cycle ///

機 能 **■MODEL0/1**

PC ← PC + qqrr + 2

■MODEL2/3

PC ← PC + qqrr + 2, CB ← NB

16ビット相対アドレスqqrr(-32768～32767)を本命令の先頭アドレス+2からのオフセットとしてプログラムカウンタ(PC)に加え、そのアドレスに無条件に分岐します。

MODEL2/3では、分岐時にNBに設定されているバンクアドレスがCBにロードされ、バンクの変更も行われます。

モード **符号付き16ビットPC相対**

例 MODEL2/3においてNB=02Hの場合、物理アドレス9000H番地にあるJRL $+2000H命令を実行。

	NB	CB	PC(論理アドレス)
実行前	02H	01H	9000H
			9002H+(2000H-2)
実行後	02H	02H	B000H

上記の例では物理アドレス013000Hへ分岐します。MODEL0/1ではNB, CBがないため物理アドレスB000Hへ分岐します。

コード

MSB							LSB	
1	1	1	1	0	0	1	1	F3H
r	r							rr
q	q							qq

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

JRL cc1, qqrr //////////// Jump to relative location qqrr if condition cc1 is true /////////// 3 cycle ///

機 能 **■MODEL0/1**

If cc1 is true then JRL qqrr

else PC ← PC + 3

■MODEL2/3

If cc1 is true then JRL qqrr

else PC ← PC + 3, NB ← CB

条件cc1が成立している場合に"JRL qqrr"命令を実行し、条件不成立の場合は次の命令を実行します。

☞ **"JRL qqrr"命令**

MODEL2/3では、分岐先バンクを指定するNBの内容が条件不成立の場合に現在のバンクアドレス(CBの内容)に戻されます。

条件cc1は以下の4種類です。

cc1	条 件	
C	Carry	(キャリーフラグC=1)
NC	Non Carry	(キャリーフラグC=0)
Z	Zero	(ゼロフラグZ=1)
NZ	Non Zero	(ゼロフラグZ=0)

コード

MSB						LSB	
1	1	1	0	1	1	cc1	ECH～EFH ※
r	r						rr
q	q						qq

※

cc1		ニーモニック	コード
C	00	JRL C,qqrr	ECH
NC	01	JRL NC,qqrr	EDH
Z	10	JRL Z,qqrr	EEH
NZ	11	JRL NZ,qqrr	EFH

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

モード **符号付き16ビットPC相対**

例 条件成立時はJRL qqrr命令と同じ動作をします。条件不成立時、物理アドレス9000H番地のJRL cc1, qqrrの動作は下記のとおりです。

	NB	CB	PC(論理アドレス)
実行前	02H	01H	9000H
実行後	01H	01H	9003H

MODEL0/1ではNB, CBはありません。

JRS rr ////////////////////////// Jump to relative location rr // 2 cycle ///

機 能 ■MODEL0/1

PC ← PC + rr + 1

■MODEL2/3

PC ← PC + rr + 1, CB ← NB

8ビット相対アドレスrr(-128～127)を本命令の先頭アドレス+1からのオフセットとしてプログラムカウンタ(PC)に加え、そのアドレスに無条件に分岐します。

MODEL2/3では、分岐時にNBに設定されているバンクアドレスがCBにロードされ、バンクの変更も行われます。

コード

MSB							LSB	
1	1	1	1	0	0	0	1	F1H
r	r							rr

フラグ

I1	I0	U	D	N	V	C	Z
−	−	−	−	−	−	−	−

モード 符号付き8ビットPC相対

例 MODEL2/3においてNB=02Hの場合、物理アドレス9000H番地にあるJRS $+20H命令を実行。

	NB	CB	PC(論理アドレス)
実行前	02H	01H	9000H
			9001H+(20H-1)
実行後	02H	02H	9020H

上記の例では物理アドレス011020Hへ分岐します。MODEL0/1ではNB, CBがないため物理アドレス9020Hへ分岐します。

JRS cc1, rr ////////////////// Jump to relative location rr if condition cc1 is true ////////////////// 2 cycle ///

機 能 ■MODEL0/1

If cc1 is true then JRS rr

else PC ← PC + 2

■MODEL2/3

If cc1 is true then JRS rr

else PC ← PC + 2, NB ← CB

条件cc1が成立している場合に"JRS rr"命令を実行し、条件不成立の場合は次の命令を実行します。

☞ **"JRS rr"命令**

MODEL2/3では、分岐先バンクを指定するNBの内容が条件不成立の場合に現在のバンクアドレス(CBの内容)に戻されます。

条件cc1は以下の4種類です。

cc1	条 件	
C	Carry	(キャリーフラグC=1)
NC	Non Carry	(キャリーフラグC=0)
Z	Zero	(ゼロフラグZ=1)
NZ	Non Zero	(ゼロフラグZ=0)

コード

MSB						LSB	
1	1	1	0	0	1	cc1	E4H～E7H ※
r	r						rr

※

cc1		ニーモニック	コード
C	00	JRS C,rr	E4H
NC	01	JRS NC,rr	E5H
Z	10	JRS Z,rr	E6H
NZ	11	JRS NZ,rr	E7H

フラグ

I1	I0	U	D	N	V	C	Z
−	−	−	−	−	−	−	−

モード 符号付き8ビットPC相対

例 条件成立時はJRS rr命令と同じ動作をします。条件不成立時、物理アドレス9000H番地のJRS cc1, rrの動作は下記のとおりです。

	NB	CB	PC(論理アドレス)
実行前	02H	01H	9000H
実行後	01H	01H	9002H

MODEL0/1ではNB, CBはありません。

JRS cc2, rr //////////////// *Jump to relative location rr if condition cc2 is true* //////////////// *3 cycle* ///

機 能 ■MODEL0/1

If cc2 is true then PC ← PC + rr + 2
else PC ← PC + 3

■MODEL2/3

If cc2 is true then PC ← PC + rr + 2, CB ← NB
else PC ← PC + 3, NB ← CB

条件cc2が成立している場合、8ビット相対アドレスrr(-128～127)を本命令の先頭アドレス+2からのオフセットとしてプログラムカウンタ(PC)に加え、そのアドレスに分岐します。条件不成立の場合は次の命令を実行します。

MODEL2/3では、分岐時にNBに設定されているバンクアドレスがCBにロードされ、バンクの変更も行われます。
また、条件不成立の場合には分岐先バンクを指定するNBの内容が現在のバンクアドレス(CBの内容)に戻されます。

条件cc2は以下の16種類です。

cc2	条 件	
LT	Less Than	([N∀V]=1)
LE	Less or Equal	(Z∨[N∀V]=1)
GT	Greater Than	(Z∨[N∀V]=0)
GE	Greater or Equal	([N∀V]=0)
V	Overflow	(V=1)
NV	Non Overflow	(V=0)
P	Plus	(N=0)
M	Minus	(N=1)
F0	F0 is set	(F0=1)
F1	F1 is set	(F1=1)
F2	F2 is set	(F2=1)
F3	F3 is set	(F3=1)
NF0	F0 is reset	(F0=0)
NF1	F1 is reset	(F1=0)
NF2	F2 is reset	(F2=0)
NF3	F3 is reset	(F3=0)

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	1	1	0	cc2				E0H～EFH ※
r	r							rr

※

cc2		ニーモニック	コード
LT	0000	JRS LT,rr	E0H
LE	0001	JRS LE,rr	E1H
GT	0010	JRS GT,rr	E2H
GE	0011	JRS GE,rr	E3H
V	0100	JRS V,rr	E4H
NV	0101	JRS NV,rr	E5H
P	0110	JRS P,rr	E6H
M	0111	JRS M,rr	E7H
F0	1000	JRS F0,rr	E8H
F1	1001	JRS F1,rr	E9H
F2	1010	JRS F2,rr	EAH
F3	1011	JRS F3,rr	EBH
NF0	1100	JRS NF0,rr	ECH
NF1	1101	JRS NF1,rr	EDH
NF2	1110	JRS NF2,rr	EEH
NF3	1111	JRS NF3,rr	EFH

フラグ

I1	I0	U	D	N	V	C	Z
−	−	−	−	−	−	−	−

モード **符号付き8ビットPC相対**

例 MODEL2/3においてNB=02Hの場合、条件成立時の物理アドレス9000H番地にあるJRS cc2, $+20Hの動作は下記のとおりです。

	NB	CB	PC(論理アドレス)
実行前	02H	01H	9000H
			9002H+(20H-2)
実行後	02H	02H	9020H

上記の例では物理アドレス011020Hへ分岐します。
MODEL0/1ではNB, CBがないため物理アドレス9020Hへ分岐します。

条件不成立時、物理アドレス9000H番地のJRS cc2, rrの動作は下記のとおりです。

	NB	CB	PC(論理アドレス)
実行前	02H	01H	9000H
実行後	01H	01H	9003H

MODEL0/1ではNB, CBはありません。

LD r, r' ////////////////////////// Load r' reg. into r reg. // 1 cycle ///

機 能 r ← r'

r'レジスタ(A/B/L/H)の内容をrレジスタ(A/B/L/H)にロードします。

コード

MSB						LSB	
0	1	0	r	0	r'		※

※

r \ r'	A (0,0)	B (0,1)	L (1,0)	H (1,1)
A(0,0)	40H	41H	42H	43H
B(0,1)	48H	49H	4AH	4BH
L(1,0)	50H	51H	52H	53H
H(1,1)	58H	59H	5AH	5BH

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

モード Src: レジスタ直接
Dst: レジスタ直接

例

設定値		結果					
r	r'	r	r'	SC			
				N	V	C	Z
5AH	42H	42H	42H	—	—	—	—

(r≠r')

LD A, BR ////////////////////// Load BR reg. into A reg. // 2 cycle ///

機 能 A ← BR

BRレジスタの内容をAレジスタにロードします。

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	1	0	0	0	0	0	0	C0H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

モード Src: レジスタ直接
Dst: レジスタ直接

例

設定値		結果					
A	BR	A	BR	SC			
				N	V	C	Z
5AH	42H	42H	42H	—	—	—	—

LD A, SC ////////////////////// Load SC into A reg. // 2 cycle ///

機 能 A ← SC

システムコンディションフラグ(SC)の内容をAレジスタにロードします。

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	1	0	0	0	0	0	1	C1H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

モード Src: レジスタ直接
Dst: レジスタ直接

例

設定値		結果					
A	SC	A	SC	SC			
				N	V	C	Z
5AH	42H	42H	42H	—	—	—	—

LD A, er ////////////////////// Load er reg. into A reg. // 2 cycle ///

機 能 A ← er

erレジスタ(NB/EP/XP/YP)の内容をAレジスタにロードします。

モード Src: レジスタ直接
Dst: レジスタ直接

例

設定値		結果					
A	er	A	er	SC			
				N	V	C	Z
5AH	42H	42H	42H	—	—	—	—

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	1	0	0	1	0	er		C8H〜CBH ※

※

er		ニーモニック	コード
NB	00	LD A, NB	C8H
EP	01	LD A, EP	C9H
XP	10	LD A, XP	CAH
YP	11	LD A, YP	CBH

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

注 意 MODEL0/1では本命令が使用できません。

LD BR, A ////////////////////// Load A reg. into BR reg. // 2 cycle ///

機 能 BR ← A

Aレジスタの内容をBRレジスタにロードします。

モード Src: レジスタ直接
Dst: レジスタ直接

例

設定値		結果					
BR	A	BR	A	SC			
				N	V	C	Z
5AH	42H	42H	42H	—	—	—	—

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	1	0	0	0	0	1	0	C2H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

LD SC, A ////////////////////// Load A reg. into SC // 3 cycle ///

機 能 SC ← A

Aレジスタの内容をシステムコンディションフラグ(SC)にロードします。

モード Src: レジスタ直接
Dst: レジスタ直接

例

設定値		結果									
SC	A	SC	A	SC							
				I1	I0	U	D	N	V	C	Z
5AH	42H	42H	42H	0	1	0	0	0	0	1	0

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	1	0	0	0	0	1	1	C3H

フラグ

I1	I0	U	D	N	V	C	Z
↕	↕	↕	↕	↕	↕	↕	↕

LD er, A ////////////////////// Load A reg. into er reg. // 2 or 3 cycle ///

機 能 er ← A

Aレジスタの内容をerレジスタ(NB/EP/XP/YP)にロードします。

モード Src: レジスタ直接
Dst: レジスタ直接

例

設定値		結果					
er	A	er	A	SC			
				N	V	C	Z
5AH	42H	42H	42H	—	—	—	—

コード

MSB LSB

1	1	0	0	1	1	1	0	CEH

1	1	0	0	1	1	er		CCH～CFH ※

※

er		ニーモニック	コード	
NB	00	LD NB, A	CCH	(3 cycle)
EP	01	LD EP, A	CDH	(2 cycle)
XP	10	LD XP, A	CEH	(2 cycle)
YP	11	LD YP, A	CFH	(2 cycle)

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

注 意 MODEL0/1では本命令が使用できません。

LD [BR:ll], r ////////////// Load r reg. into location [BR:ll] // 3 cycle ///

機 能 [BR:*ll*] ← r

rレジスタ(A/B/L/H)の内容をBRレジスタの内容を上位バイト、8ビット絶対アドレス*ll*を下位バイトとしてアドレス指定されるデータメモリにロードします。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: レジスタ直接
Dst: 8ビット絶対

例

設定値		結果					
[BR:*ll*]	r	[BR:*ll*]	r	SC			
				N	V	C	Z
5AH	42H	42H	42H	—	—	—	—

コード

MSB LSB

0	1	1	1	1	0	r		78H～7BH ※

						l	*l*	*ll*

※

r		ニーモニック	コード
A	00	LD [BR:*ll*], A	78H
B	01	LD [BR:*ll*], B	79H
L	10	LD [BR:*ll*], L	7AH
H	11	LD [BR:*ll*], H	7BH

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

LD [hhll], r //////////////// *Load r reg. into location [hhll]* // *5 cycle* ///

機 能 [hh*ll*] ← r

rレジスタ(A/B/L/H)の内容を16ビット絶対アドレスhh*ll*でアドレス指定されるデータメモリにロードします。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: レジスタ直接
Dst: 16ビット絶対

例

設定値		結果					
[hh*ll*]	r	[hh*ll*]	r	SC			
				N	V	C	Z
5AH	42H	42H	42H	—	—	—	—

コード

MSB LSB

1	1	0	0	1	1	1	0	CEH

1	1	0	1	0	1	r		D4H～D7H ※

l	*l*							*ll*

h	h							hh

※

r		ニーモニック	コード
A	00	LD [hh*ll*], A	D4H
B	01	LD [hh*ll*], B	D5H
L	10	LD [hh*ll*], L	D6H
H	11	LD [hh*ll*], H	D7H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

LD [HL], r //////////////////// *Load r reg. into location [HL]* // *2 cycle* ///

機 能 [HL] ← r

rレジスタ(A/B/L/H)の内容をHLレジスタでアドレス指定されるデータメモリにロードします。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: レジスタ直接
Dst: レジスタ間接

例

設定値		結果					
[HL]	r	[HL]	r	SC			
				N	V	C	Z
5AH	42H	42H	42H	—	—	—	—

コード

MSB LSB

0	1	1	0	1	0	r		68H～6BH ※

※

r		ニーモニック	コード
A	00	LD [HL], A	68H
B	01	LD [HL], B	69H
L	10	LD [HL], L	6AH
H	11	LD [HL], H	6BH

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

LD [IX], r ////////////////////// Load r reg. into location [IX] // 2 cycle ///

機 能 [IX] ← r

rレジスタ(A/B/L/H)の内容をIXレジスタでアドレス指定されるデータメモリにロードします。
XPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: レジスタ直接
Dst: レジスタ間接

例

設定値		結果					
[IX]	r	[IX]	r	SC			
				N	V	C	Z
5AH	42H	42H	42H	—	—	—	—

コード

MSB						LSB	
0	1	1	0	0	0	r	60H～63H ※

※

r		ニーモニック	コード
A	00	LD [IX], A	60H
B	01	LD [IX], B	61H
L	10	LD [IX], L	62H
H	11	LD [IX], H	63H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

LD [IY], r ////////////////////// Load r reg. into location [IY] // 2 cycle ///

機 能 [IY] ← r

rレジスタ(A/B/L/H)の内容をIYレジスタでアドレス指定されるデータメモリにロードします。
YPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: レジスタ直接
Dst: レジスタ間接

例

設定値		結果					
[IY]	r	[IY]	r	SC			
				N	V	C	Z
5AH	42H	42H	42H	—	—	—	—

コード

MSB						LSB	
0	1	1	1	0	0	r	70H～73H ※

※

r		ニーモニック	コード
A	00	LD [IY], A	70H
B	01	LD [IY], B	71H
L	10	LD [IY], L	72H
H	11	LD [IY], H	73H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

LD [IX+dd], r ////////////// *Load r reg. into location [IX + dd]* ////////////////////////////////////// *4 cycle* ///

機 能 **[IX+dd] ← r**

rレジスタ(A/B/L/H)の内容をIXレジスタの内容とディスプレースメントddの和でアドレス指定されたデータメモリにロードします。
ddは符号付きデータとして扱われ、範囲は-128～127です。
XPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード **Src: レジスタ直接**
Dst: ディスプレースメント付きレジスタ間接

例

設定値		結果					
[IX+dd]	r	[IX+dd]	r	SC			
				N	V	C	Z
5AH	42H	42H	42H	—	—	—	—

コード

MSB LSB

1	1	0	0	1	1	1	0	CEH

0	1	0	r		1	0	0	44H、4CH、54H、5CH ※

d	d	dd

※

r		ニーモニック	コード
A	00	LD [IX+dd], A	44H
B	01	LD [IX+dd], B	4CH
L	10	LD [IX+dd], L	54H
H	11	LD [IX+dd], H	5CH

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

LD [IY+dd], r ////////////// *Load r reg. into location [IY + dd]* ////////////////////////////////////// *4 cycle* ///

機 能 **[IY+dd] ← r**

rレジスタ(A/B/L/H)の内容をIYレジスタの内容とディスプレースメントddの和でアドレス指定されたデータメモリにロードします。
ddは符号付きデータとして扱われ、範囲は-128～127です。
YPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード **Src: レジスタ直接**
Dst: ディスプレースメント付きレジスタ間接

例

設定値		結果					
[IY+dd]	r	[IY+dd]	r	SC			
				N	V	C	Z
5AH	42H	42H	42H	—	—	—	—

コード

MSB LSB

1	1	0	0	1	1	1	0	CEH

0	1	0	r		1	0	1	45H、4DH、55H、5DH ※

d	d	dd

※

r		ニーモニック	コード
A	00	LD [IY+dd], A	45H
B	01	LD [IY+dd], B	4DH
L	10	LD [IY+dd], L	55H
H	11	LD [IY+dd], H	5DH

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

LD [IX+L], r *//////////////// Load r reg. into location [IX + L] // 4 cycle ///*

機 能 **[IX+L] ← r**

rレジスタ(A/B/L/H)の内容をIXレジスタの内容とLレジスタの内容の和でアドレス指定されたデータメモリにロードします。
Lレジスタの内容は符号付きデータとして扱われ、範囲は-128～127です。
XPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード **Src: レジスタ直接**
Dst: インデックスレジスタ付きレジスタ間接

例

設定値		結果					
[IX+L]	r	[IX+L]	r	SC			
				N	V	C	Z
5AH	42H	42H	42H	—	—	—	—

コード MSB LSB

1	1	0	0	1	1	1	0	CEH

0	1	0	r		1	1	0	46H、4EH、56H、5EH ※

※

r		ニーモニック	コード
A	00	LD [IX+L], A	46H
B	01	LD [IX+L], B	4EH
L	10	LD [IX+L], L	56H
H	11	LD [IX+L], H	5EH

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

LD [IY+L], r *//////////////// Load r reg. into location [IY + L] // 4 cycle ///*

機 能 **[IY+L] ← r**

rレジスタ(A/B/L/H)の内容をIYレジスタの内容とLレジスタの内容の和でアドレス指定されたデータメモリにロードします。
Lレジスタの内容は符号付きデータとして扱われ、範囲は-128～127です。
YPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード **Src: レジスタ直接**
Dst: インデックスレジスタ付きレジスタ間接

例

設定値		結果					
[IY+L]	r	[IY+L]	r	SC			
				N	V	C	Z
5AH	42H	42H	42H	—	—	—	—

コード MSB LSB

1	1	0	0	1	1	1	0	CEH

0	1	0	r		1	1	1	47H、4FH、57H、5FH ※

※

r		ニーモニック	コード
A	00	LD [IY+L], A	47H
B	01	LD [IY+L], B	4FH
L	10	LD [IY+L], L	57H
H	11	LD [IY+L], H	5FH

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

LD r, #nn ////////////////////// Load immediate data nn into r reg. ////////////////////////////////////// 2 cycle ///

機 能 r ← nn

8ビット即値データnnをrレジスタ(A/B/L/H)にロードします。

モード Src: 即値データ
Dst: レジスタ直接

コード

MSB							LSB	
1	0	1	1	0	0	r		B0H〜B3H ※
n	n							nn

※

r		ニーモニック	コード
A	00	LD A, #nn	B0H
B	01	LD B, #nn	B1H
L	10	LD L, #nn	B2H
H	11	LD H, #nn	B3H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

例

設定値		結果					
r	nn	r	nn	SC			
				N	V	C	Z
5AH	42H	42H	42H	—	—	—	—

LD BR, #hh ////////////////// Load immediate data hh into BR reg. ////////////////////////////////////// 2 cycle ///

機 能 BR ← hh

8ビット即値データhhをBRレジスタにロードします。

モード Src: 即値データ
Dst: レジスタ直接

コード

MSB							LSB	
1	0	1	1	0	1	0	0	B4H
h	h							hh

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

例

設定値		結果					
BR	hh	BR	hh	SC			
				N	V	C	Z
5AH	42H	42H	42H	—	—	—	—

LD SC, #nn ////////////////// Load immediate data nn into SC // 3 cycle ///

機 能 SC ←nn

8ビット即値データnnをシステムコンディションフラグ(SC)にセットします。

モード Src: 即値データ
Dst: レジスタ直接

コード

MSB							LSB	
1	0	0	1	1	1	1	1	9FH
n	n							nn

フラグ

I1	I0	U	D	N	V	C	Z
↕	↕	↕	↕	↕	↕	↕	↕

例

設定値		結果								
SC	nn	SC	nn	SC						
				I1	I0	U	D	N	V	C	Z
5AH	42H	42H	42H	0	1	0	0	0	0	1	0

LD NB, #bb /////////////// *Load immediate data bb into NB reg.* ////////////////////////////////// *4 cycle* ///

機 能 **NB ← bb**

8ビット即値データbbをニューコードバンクレジスタ(NB)にロードします。

モード **Src: 即値データ**
Dst: レジスタ直接

例

設定値		結果					
NB	bb	NB	bb	SC			
				N	V	C	Z
5AH	42H	42H	42H	—	—	—	—

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	1	0	0	0	1	0	0	C4H
b	b	b	b	b	b	b	b	bb

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

注 意 **MODEL0/1では本命令が使用できません。**

LD EP, #pp ///////////////// *Load immediate data pp into EP reg.* ////////////////////////////////// *3 cycle* ///

機 能 **EP ← pp**

8ビット即値データppをエクスパンドページレジスタ(EP)にロードします。

モード **Src: 即値データ**
Dst: レジスタ直接

例

設定値		結果					
EP	pp	EP	pp	SC			
				N	V	C	Z
5AH	42H	42H	42H	—	—	—	—

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	1	0	0	0	1	0	1	C5H
p	p	p	p	p	p	p	p	pp

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

注 意 **MODEL0/1では本命令が使用できません。**

LD XP, #pp ///////////////// *Load immediate data pp into XP reg.* ////////////////////////////////// *3 cycle* ///

機 能 **XP ← pp**

8ビット即値データppをIX用エクスパンドページレジスタ(XP)にロードします。

モード **Src: 即値データ**
Dst: レジスタ直接

例

設定値		結果					
XP	pp	XP	pp	SC			
				N	V	C	Z
5AH	42H	42H	42H	—	—	—	—

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	1	0	0	0	1	1	0	C6H
p	p	p	p	p	p	p	p	pp

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

注 意 **MODEL0/1では本命令が使用できません。**

LD YP, #pp /////////////// Load immediate data pp into YP reg. ////////////////////////////////// 3 cycle ///

機 能 YP ← pp

8ビット即値データppをIY用エクスパンドページレジスタ(YP)にロードします。

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	1	0	0	0	1	1	1	C7H
p	p							pp

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

モード Src: 即値データ
Dst: レジスタ直接

例

設定値		結果					
YP	pp	YP	pp	SC			
				N	V	C	Z
5AH	42H	42H	42H	—	—	—	—

注 意 MODEL0/1では本命令が使用できません。

LD [BR:ll], #nn ////// Load immediate data nn into location [BR:ll] ////////////////////////// 4 cycle ///

機 能 [BR:*ll*] ← nn

8ビット即値データnnをBRレジスタの内容を上位バイト、8ビット絶対アドレス*ll*を下位バイトとしてアドレス指定されるデータメモリにロードします。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

コード

MSB							LSB	
1	1	0	1	1	1	0	1	DDH
l	*l*							*ll*
n	n							nn

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

モード Src: 即値データ
Dst: 8ビット絶対

例

設定値		結果					
[BR:*ll*]	nn	[BR:*ll*]	nn	SC			
				N	V	C	Z
5AH	42H	42H	42H	—	—	—	—

LD [HL], #nn ////////////// Load immediate data nn into location [HL] ////////////////////////// 3 cycle ///

機 能 [HL] ← nn

8ビット即値データnnをHLレジスタでアドレス指定されるデータメモリにロードします。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

コード

MSB							LSB	
1	0	1	1	0	1	0	1	B5H
n	n							nn

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

モード Src: 即値データ
Dst: レジスタ間接

例

設定値		結果					
[HL]	nn	[HL]	nn	SC			
				N	V	C	Z
5AH	42H	42H	42H	—	—	—	—

LD [ir], #nn /////////////// Load immediate data nn into location [ir reg.] ////////////////////// 3 cycle ///

機 能 [ir] ← nn

8ビット即値データnnをirレジスタ(IX/IY)でアドレス指定されるデータメモリにロードします。
XPレジスタ(IX指定時)、YPレジスタ(IY指定時)の内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: 即値データ
Dst: レジスタ間接

例

設定値		結果					
[ir]	nn	[ir]	nn	SC			
				N	V	C	Z
5AH	42H	42H	42H	—	—	—	—

コード

MSB							LSB	
1	0	1	1	0	1	1	ir	B6H、B7H ※
n	n							nn

※

ir		ニーモニック	コード
IX	0	LD [IX], #nn	B6H
IY	1	LD [IY], #nn	B7H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

LD r, [BR:ll] /////////////// Load location [BR:ll] into r reg. // 3 cycle ///

機 能 r ← [BR:*ll*]

BRレジスタの内容を上位バイト、8ビット絶対アドレス*ll*を下位バイトとしてアドレス指定されるデータメモリの内容をrレジスタ(A/B/L/H)にロードします。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: 8ビット絶対
Dst: レジスタ直接

例

設定値		結果					
r	[BR:*ll*]	r	[BR:*ll*]	SC			
				N	V	C	Z
5AH	42H	42H	42H	—	—	—	—

コード

MSB							LSB	
0	1	0	r		1	0	0	44H、4CH、54H、5CH ※
l	*l*							*ll*

※

r		ニーモニック	コード
A	00	LD A, [BR:*ll*]	44H
B	01	LD B, [BR:*ll*]	4CH
L	10	LD L, [BR:*ll*]	54H
H	11	LD H, [BR:*ll*]	5CH

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

LD [HL], [BR:ll] ///// *Load location [BR:ll] into location [HL]* ////////////////////////////// *4 cycle ///*

機 能 **[HL] ← [BR:*ll*]**

BRレジスタの内容を上位バイト、8ビット絶対アドレス*ll*を下位バイトとしてアドレス指定されるデータメモリの内容をHLレジスタでアドレス指定されるデータメモリにロードします。
EPレジスタの内容がデータメモリ双方のページアドレスになります(MODEL2/3)。

モード **Src: 8ビット絶対**
Dst: レジスタ間接

例

設定値		結果					
[HL]	[BR:*ll*]	[HL]	[BR:*ll*]	SC			
				N	V	C	Z
5AH	42H	42H	42H	—	—	—	—

コード

MSB							LSB	
0	1	1	0	1	1	0	0	6CH
l	*l*							*ll*

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

LD [ir], [BR:ll] ///////// *Load location [BR:ll] into location [ir reg.]* ////////////////////////// *4 cycle ///*

機 能 **[ir] ← [BR:*ll*]**

BRレジスタの内容を上位バイト、8ビット絶対アドレス*ll*を下位バイトとしてアドレス指定されるデータメモリの内容をirレジスタ(IX/IY)でアドレス指定されるデータメモリにロードします。
EPレジスタの内容がデータメモリ[BR:*ll*]のページアドレス、XPレジスタ(IX指定時)、YPレジスタ(IY指定時)の内容がデータメモリ[ir]のページアドレスになります(MODEL2/3)。

モード **Src: 8ビット絶対**
Dst: レジスタ間接

例

設定値		結果					
[ir]	[BR:*ll*]	[ir]	[BR:*ll*]	SC			
				N	V	C	Z
5AH	42H	42H	42H	—	—	—	—

コード

MSB							LSB	
0	1	1	ir	0	1	0	0	64H、74H ※
l	*l*							*ll*

※

ir		ニーモニック	コード
IX	0	LD [IX], [BR:*ll*]	64H
IY	1	LD [IY], [BR:*ll*]	74H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

LD r, [hhll] ////////////////// *Load location [hhll] into r reg.* // *5 cycle* ///

機 能 r ← [hh*ll*]

16ビット絶対アドレスhh*ll*でアドレス指定されたデータメモリの内容をrレジスタ(A/B/L/H)にロードします。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: 16ビット絶対
Dst: レジスタ直接

例

設定値		結果					
r	[hh*ll*]	r	[hh*ll*]	SC			
				N	V	C	Z
5AH	42H	42H	42H	—	—	—	—

コード MSB LSB

1	1	0	0	1	1	1	0	CEH

1	1	0	1	0	0	r		D0H～D3H ※

								ll

								hh

※

r		ニーモニック	コード
A	00	LD A, [hh*ll*]	D0H
B	01	LD B, [hh*ll*]	D1H
L	10	LD L, [hh*ll*]	D2H
H	11	LD H, [hh*ll*]	D3H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

LD r, [HL] ////////////////////// *Load location [HL] into r reg.* // *2 cycle* ///

機 能 r ← [HL]

HLレジスタでアドレス指定されたデータメモリの内容をrレジスタ(A/B/L/H)にロードします。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: レジスタ間接
Dst: レジスタ直接

例

設定値		結果					
r	[HL]	r	[HL]	SC			
				N	V	C	Z
5AH	42H	42H	42H	—	—	—	—

コード MSB LSB

0	1	0	r		1	0	1	45H、4DH、55H、5DH ※

※

r		ニーモニック	コード
A	00	LD A, [HL]	45H
B	01	LD B, [HL]	4DH
L	10	LD L, [HL]	55H
H	11	LD H, [HL]	5DH

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

LD [BR:ll], [HL] ///// *Load location [HL] into location [BR:ll]* ////////////////////////////// *4 cycle* ///

機 能 [BR:*ll*] ← [HL]

HLレジスタでアドレス指定されたデータメモリの内容をBRレジスタの内容を上位バイト、8ビット絶対アドレス*ll*を下位バイトとしてアドレス指定されるデータメモリにロードします。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: レジスタ間接
Dst: 8ビット絶対

例

設定値		結果					
[BR:*ll*]	[HL]	[BR:*ll*]	[HL]	SC			
				N	V	C	Z
5AH	42H	42H	42H	—	—	—	—

コード

MSB							LSB	
0	1	1	1	1	1	0	1	7DH
			l	*l*				*ll*

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

LD [HL], [HL] ////////// *Load location [HL] into location [HL]* ////////////////////////////////// *3 cycle* ///

機 能 [HL] ← [HL]

HLレジスタでアドレス指定されたデータメモリの内容を同じアドレスにロードします。結果的には3サイクルを消費するのみとなります。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: レジスタ間接
Dst: レジスタ間接

例

設定値	結果				
[HL]	[HL]	SC			
		N	V	C	Z
42H	42H	—	—	—	—

コード

MSB							LSB	
0	1	1	0	1	1	0	1	6DH

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

LD [ir], [HL] ////////////// *Load location [HL] into location [ir reg.]* ////////////////////////////// *3 cycle* ///

機 能 [ir] ← [HL]

HLレジスタでアドレス指定されたデータメモリの内容をirレジスタ(IX/IY)でアドレス指定されるデータメモリにロードします。
EPレジスタの内容がデータメモリ[HL]のページアドレス、XPレジスタ(IX指定時)、YPレジスタ(IY指定時)の内容がデータメモリ[ir]のページアドレスになります(MODEL2/3)。

モード Src: レジスタ間接
Dst: レジスタ間接

例

設定値		結果					
[ir]	[HL]	[ir]	[HL]	SC			
				N	V	C	Z
5AH	42H	42H	42H	—	—	—	—

コード

MSB							LSB	
0	1	1	ir	0	1	0	1	65H、75H ※

※

ir		ニーモニック	コード
IX	0	LD [IX], [HL]	65H
IY	1	LD [IY], [HL]	75H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

LD r, [IX] ////////////////////// Load location [IX] into r reg. // 2 cycle ///

機 能 r ← [IX]

IXレジスタでアドレス指定されたデータメモリの内容をrレジスタ(A/B/L/H)にロードします。
XPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: レジスタ間接
Dst: レジスタ直接

例

設定値		結果					
r	[IX]	r	[IX]	SC			
				N	V	C	Z
5AH	42H	42H	42H	—	—	—	—

コード

MSB						LSB	
0	1	0	r	1	1	0	46H、4EH、56H、5EH ※

※

r		ニーモニック	コード
A	00	LD A, [IX]	46H
B	01	LD B, [IX]	4EH
L	10	LD L, [IX]	56H
H	11	LD H, [IX]	5EH

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

LD [BR:ll], [IX] /////// Load location [IX] into location [BR:ll] ////////////////////////////////// 4 cycle ///

機 能 [BR:*ll*] ← [IX]

IXレジスタでアドレス指定されたデータメモリの内容をBRレジスタの内容を上位バイト、8ビット絶対アドレス*ll*を下位バイトとしてアドレス指定されるデータメモリにロードします。
EPレジスタの内容がデータメモリ[BR:*ll*]のページアドレス、XPレジスタ内容がデータメモリ[IX]のページアドレスになります(MODEL2/3)。

モード Src: レジスタ間接
Dst: 8ビット絶対

例

設定値		結果					
[BR:*ll*]	[IX]	[BR:*ll*]	[IX]	SC			
				N	V	C	Z
5AH	42H	42H	42H	—	—	—	—

コード

MSB						LSB	
0	1	1	1	1	1	1	0	7EH
			l	*l*				*ll*

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

LD [HL], [IX] ////////////// Load location [IX] into location [HL] ////////////////////////////////////// 3 cycle ///

機 能 [HL] ← [IX]

IXレジスタでアドレス指定されたデータメモリの内容をHLレジスタでアドレス指定されるデータメモリにロードします。
EPレジスタの内容がデータメモリ[HL]のページアドレス、XPレジスタ内容がデータメモリ[IX]のページアドレスになります(MODEL2/3)。

モード Src: レジスタ間接
Dst: レジスタ間接

例

設定値		結果					
[HL]	[IX]	[HL]	[IX]	SC			
				N	V	C	Z
5AH	42H	42H	42H	—	—	—	—

コード

MSB						LSB	
0	1	1	0	1	1	1	0	6EH

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

LD [ir], [IX] //////////////// *Load location [IX] into location [ir reg.]* //////////////////////////////// *3 cycle* ///

機 能 [ir] ← [IX]

IXレジスタでアドレス指定されたデータメモリの内容をirレジスタ(IX/IY)でアドレス指定されるデータメモリにロードします。
XPレジスタ(IX指定時)、YPレジスタ(IY指定時)の内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: レジスタ間接
Dst: レジスタ間接

例

設定値		結果					
[IY]	[IX]	[IY]	[IX]	SC			
				N	V	C	Z
5AH	42H	42H	42H	—	—	—	—

コード

MSB							LSB	
0	1	1	ir	0	1	1	0	66H、76H ※

※

ir		ニーモニック	コード
IX	0	LD [IX], [IX]	66H
IY	1	LD [IY], [IX]	76H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

LD r, [IY] ////////////////////// *Load location [IY] into r reg.* // *2 cycle* ///

機 能 r ← [IY]

IYレジスタでアドレス指定されたデータメモリの内容をrレジスタ(A/B/L/H)にロードします。
YPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: レジスタ間接
Dst: レジスタ直接

例

設定値		結果					
r	[IY]	r	[IY]	SC			
				N	V	C	Z
5AH	42H	42H	42H	—	—	—	—

コード

MSB							LSB	
0	1	0	r	1	1	1		47H、4FH、57H、5FH ※

※

r		ニーモニック	コード
A	00	LD A, [IY]	47H
B	01	LD B, [IY]	4FH
L	10	LD L, [IY]	57H
H	11	LD H, [IY]	5FH

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

LD [BR:ll], [IY] /////// Load location [IY] into location [BR:ll] ////////////////////////////////// 4 cycle ///

機 能 [BR:*ll*] ← [IY]

IYレジスタでアドレス指定されたデータメモリの内容をBRレジスタの内容を上位バイト、8ビット絶対アドレス*ll*を下位バイトとしてアドレス指定されるデータメモリにロードします。
EPレジスタの内容がデータメモリ[BR:*ll*]のページアドレス、YPレジスタ内容がデータメモリ[IY]のページアドレスになります(MODEL2/3)。

モード Src: レジスタ間接
Dst: 8ビット絶対

例

設定値		結果					
[BR:*ll*]	[IY]	[BR:*ll*]	[IY]	SC			
				N	V	C	Z
5AH	42H	42H	42H	—	—	—	—

コード

MSB							LSB	
0	1	1	1	1	1	1	1	7FH
			l	*l*				*ll*

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

LD [HL], [IY] //////////// Load location [IY] into location [HL] ////////////////////////////////// 3 cycle ///

機 能 [HL] ← [IY]

IYレジスタでアドレス指定されたデータメモリの内容をHLレジスタでアドレス指定されるデータメモリにロードします。
EPレジスタの内容がデータメモリ[HL]のページアドレス、YPレジスタ内容がデータメモリ[IY]のページアドレスになります(MODEL2/3)。

モード Src: レジスタ間接
Dst: レジスタ間接

例

設定値		結果					
[HL]	[IY]	[HL]	[IY]	SC			
				N	V	C	Z
5AH	42H	42H	42H	—	—	—	—

コード

MSB							LSB	
0	1	1	0	1	1	1	1	6FH

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

LD [ir], [IY] ////////////////// Load location [IY] into location [ir reg.] ////////////////////////////////// 3 cycle ///

機 能 [ir] ← [IY]

IYレジスタでアドレス指定されたデータメモリの内容をirレジスタ(IX/IY)でアドレス指定されるデータメモリにロードします。
XPレジスタ(IX指定時)、YPレジスタ(IY指定時)の内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: レジスタ間接
Dst: レジスタ間接

例

設定値		結果					
[IX]	[IY]	[IX]	[IY]	SC			
				N	V	C	Z
5AH	42H	42H	42H	—	—	—	—

コード

MSB							LSB	
0	1	1	ir	0	1	1	1	67H、77H ※

※

ir		ニーモニック	コード
IX	0	LD [IX], [IY]	67H
IY	1	LD [IY], [IY]	77H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

LD r, [IX+dd] ////////////// *Load location [IX + dd] into r reg.* ////////////////////////////////////// *4 cycle* ///

機 能 r ← [IX+dd]

IXレジスタの内容とディスプレースメントddの和でアドレス指定されたデータメモリの内容をrレジスタ(A/B/L/H)にロードします。
ddは符号付きデータとして扱われ、範囲は-128～127です。
XPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: ディスプレースメント付きレジスタ間接
Dst: レジスタ直接

例

設定値		結果					
r	[IX+dd]	r	[IX+dd]	SC			
				N	V	C	Z
5AH	42H	42H	42H	—	—	—	—

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	1	0	r	0	0	0		40H、48H、50H、58H ※
			d	d				dd

※

r		ニーモニック	コード
A	00	LD A, [IX+dd]	40H
B	01	LD B, [IX+dd]	48H
L	10	LD L, [IX+dd]	50H
H	11	LD H, [IX+dd]	58H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

LD [HL], [IX+dd] /// *Load location [IX + dd] into location [HL]* ////////////////////////// *5 cycle* ///

機 能 [HL] ← [IX+dd]

IXレジスタの内容とディスプレースメントddの和でアドレス指定されたデータメモリの内容をHLレジスタでアドレス指定されるデータメモリにロードします。
ddは符号付きデータとして扱われ、範囲は-128～127です。
EPレジスタの内容がデータメモリ[HL]のページアドレス、XPレジスタ内容がデータメモリ[IX+dd]のページアドレスになります(MODEL2/3)。

モード Src: ディスプレースメント付きレジスタ間接
Dst: レジスタ間接

例

設定値		結果					
[HL]	[IX+dd]	[HL]	[IX+dd]	SC			
				N	V	C	Z
5AH	42H	42H	42H	—	—	—	—

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	1	1	0	0	0	0	0	60H
			d	d				dd

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

LD [ir], [IX+dd] ////// Load location [IX + dd] into location [ir reg.] //////////////////////// 5 cycle ///

機 能 [ir] ← [IX+dd]

IXレジスタの内容とディスプレースメントddの和でアドレス指定されたデータメモリの内容をirレジスタ(IX/IY)でアドレス指定されるデータメモリにロードします。
ddは符号付きデータとして扱われ、範囲は-128～127です。
XPレジスタ(IX指定時)、YPレジスタ(IY指定時)の内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: ディスプレースメント付きレジスタ間接
Dst: レジスタ間接

例

設定値		結果					
[ir]	[IX+dd]	[ir]	[IX+dd]	SC			
				N	V	C	Z
5AH	42H	42H	42H	—	—	—	—

コード

MSB LSB

1	1	0	0	1	1	1	0	CEH

0	1	1	ir	1	0	0	0	68H、78H ※

d	d	dd

※

ir		ニーモニック	コード
IX	0	LD [IX], [IX+dd]	68H
IY	1	LD [IY], [IX+dd]	78H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

LD r, [IY+dd] ////////////// Load location [IY + dd] into r reg. // 4 cycle ///

機 能 r ← [IY+dd]

IYレジスタの内容とディスプレースメントddの和でアドレス指定されたデータメモリの内容をrレジスタ(A/B/L/H)にロードします。
ddは符号付きデータとして扱われ、範囲は-128～127です。
YPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: ディスプレースメント付きレジスタ間接
Dst: レジスタ直接

例

設定値		結果					
r	[IY+dd]	r	[IY+dd]	SC			
				N	V	C	Z
5AH	42H	42H	42H	—	—	—	—

コード

MSB LSB

1	1	0	0	1	1	1	0	CEH

0	1	0	r	0	0	1	41H、49H、51H、59H ※

d	d	dd

※

r		ニーモニック	コード
A	00	LD A, [IY+dd]	41H
B	01	LD B, [IY+dd]	49H
L	10	LD L, [IY+dd]	51H
H	11	LD H, [IY+dd]	59H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

LD [HL], [IY+dd] /// Load location [IY + dd] into location [HL] ////////////////////////// 5 cycle ///

機 能 [HL] ← [IY+dd]

IYレジスタの内容とディスプレースメントddの和でアドレス指定されたデータメモリの内容をHLレジスタでアドレス指定されるデータメモリにロードします。
ddは符号付きデータとして扱われ、範囲は-128～127です。
EPレジスタの内容がデータメモリ[HL]のページアドレス、YPレジスタ内容がデータメモリ[IY+dd]のページアドレスになります(MODEL2/3)。

モード Src: ディスプレースメント付きレジスタ間接
Dst: レジスタ間接

例

設定値		結果					
[HL]	[IY+dd]	[HL]	[IY+dd]	SC			
				N	V	C	Z
5AH	42H	42H	42H	—	—	—	—

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	1	1	0	0	0	0	1	61H
			d	d				dd

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

LD [ir], [IY+dd] /////// Load location [IY + dd] into location [ir reg.] ////////////////////// 5 cycle ///

機 能 [ir] ← [IY+dd]

IYレジスタの内容とディスプレースメントddの和でアドレス指定されたデータメモリの内容をirレジスタ(IX/IY)でアドレス指定されるデータメモリにロードします。
ddは符号付きデータとして扱われ、範囲は-128～127です。
XPレジスタ(IX指定時)、YPレジスタ(IY指定時)の内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: ディスプレースメント付きレジスタ間接
Dst: レジスタ間接

例

設定値		結果					
[ir]	[IY+dd]	[ir]	[IY+dd]	SC			
				N	V	C	Z
5AH	42H	42H	42H	—	—	—	—

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	1	1	ir	1	0	0	1	69H、79H ※
			d	d				dd

※

ir		ニーモニック	コード
IX	0	LD [IX], [IY+dd]	69H
IY	1	LD [IY], [IY+dd]	79H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

LD r, [IX+L] *//////////////// Load location [IX + L] into r reg. // 4 cycle ///*

機 能 r ← [IX+L]

IXレジスタの内容とLレジスタの内容の和でアドレス指定されたデータメモリの内容をrレジスタ(A/B/L/H)にロードします。
Lレジスタの内容は符号付きデータとして扱われ、範囲は-128～127です。
XPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: インデックスレジスタ付きレジスタ間接
Dst: レジスタ直接

例

設定値		結果					
r	[IX+L]	r	[IX+L]	SC			
				N	V	C	Z
5AH	42H	42H	42H	—	—	—	—

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	1	0	r		0	1	0	42H、4AH、52H、5AH ※

※

r		ニーモニック	コード
A	00	LD A, [IX+L]	42H
B	01	LD B, [IX+L]	4AH
L	10	LD L, [IX+L]	52H
H	11	LD H, [IX+L]	5AH

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

LD [HL], [IX+L] *////// Load location [IX + L] into location [HL] ////////////////////////////// 5 cycle ///*

機 能 [HL] ← [IX+L]

IXレジスタの内容とLレジスタの内容の和でアドレス指定されたデータメモリの内容をHLレジスタでアドレス指定されるデータメモリにロードします。
Lレジスタの内容は符号付きデータとして扱われ、範囲は-128～127です。
EPレジスタの内容がデータメモリ[HL]のページアドレス、XPレジスタ内容がデータメモリ[IX+L]のページアドレスになります(MODEL2/3)。

モード Src: インデックスレジスタ付きレジスタ間接
Dst: レジスタ間接

例

設定値		結果					
[HL]	[IX+L]	[HL]	[IX+L]	SC			
				N	V	C	Z
5AH	42H	42H	42H	—	—	—	—

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	1	1	0	0	0	1	0	62H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

LD [ir], [IX+L] ////////// *Load location [IX + L] into location [ir reg.]* ////////////////////// *5 cycle* ///

機 能 [ir] ← [IX+L]

IXレジスタの内容とLレジスタの内容の和でアドレス指定されたデータメモリの内容をirレジスタ(IX/IY)でアドレス指定されるデータメモリにロードします。
Lレジスタの内容は符号付きデータとして扱われ、範囲は-128～127です。
XPレジスタ(IX指定時)、YPレジスタ(IY指定時)の内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: インデックスレジスタ付きレジスタ間接
Dst: レジスタ間接

例

設定値		結果					
[ir]	[IX+L]	[ir]	[IX+L]	SC			
				N	V	C	Z
5AH	42H	42H	42H	—	—	—	—

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	1	1	ir	1	0	1	0	6AH、7AH ※

※

r		ニーモニック	コード
IX	0	LD [IX], [IX+L]	6AH
IY	1	LD [IY], [IX+L]	7AH

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

LD r, [IY+L] //////////////// *Load location [IY + L] into r reg.* // *4 cycle* ///

機 能 r ← [IY+L]

IYレジスタの内容とLレジスタの内容の和でアドレス指定されたデータメモリの内容をrレジスタ(A/B/L/H)にロードします。
Lレジスタの内容は符号付きデータとして扱われ、範囲は-128～127です。
YPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: インデックスレジスタ付きレジスタ間接
Dst: レジスタ直接

例

設定値		結果					
r	[IY+L]	r	[IY+L]	SC			
				N	V	C	Z
5AH	42H	42H	42H	—	—	—	—

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	1	0	r		0	1	1	43H、4BH、53H、5BH ※

※

r		ニーモニック	コード
A	00	LD A, [IY+L]	43H
B	01	LD B, [IY+L]	4BH
L	10	LD L, [IY+L]	53H
H	11	LD H, [IY+L]	5BH

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

LD [HL], [IY+L] *////// Load location [IY + L] into location [HL] ////////////////////////////// 5 cycle ///*

機 能 [HL] ← [IY+L]

IYレジスタの内容とLレジスタの内容の和でアドレス指定されたデータメモリの内容をHLレジスタでアドレス指定されるデータメモリにロードします。
Lレジスタの内容は符号付きデータとして扱われ、範囲は-128～127です。
EPレジスタの内容がデータメモリ[HL]のページアドレス、YPレジスタ内容がデータメモリ[IY+L]のページアドレスになります(MODEL2/3)。

モード Src: インデックスレジスタ付きレジスタ間接
Dst: レジスタ間接

例

設定値		結果					
[HL]	[IY+L]	[HL]	[IY+L]	SC			
				N	V	C	Z
5AH	42H	42H	42H	—	—	—	—

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	1	1	0	0	0	1	1	63H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

LD [ir], [IY+L] *////////// Load location [IY + L] into location [ir reg.] ////////////////////////// 5 cycle ///*

機 能 [ir] ← [IY+L]

IYレジスタの内容とLレジスタの内容の和でアドレス指定されたデータメモリの内容をirレジスタ(IX/IY)でアドレス指定されるデータメモリにロードします。
Lレジスタの内容は符号付きデータとして扱われ、範囲は-128～127です。
XPレジスタ(IX指定時)、YPレジスタ(IY指定時)の内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: インデックスレジスタ付きレジスタ間接
Dst: レジスタ間接

例

設定値		結果					
[ir]	[IY+L]	[ir]	[IY+L]	SC			
				N	V	C	Z
5AH	42H	42H	42H	—	—	—	—

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	1	1	ir	1	0	1	1	6BH、7BH ※

※

ir		ニーモニック	コード
IX	0	LD [IX], [IY+L]	6BH
IY	1	LD [IY], [IY+L]	7BH

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

LD rp, rp' ////////////////// Load rp' reg. into rp reg. // 2 cycle ///

機 能 rp ← rp'

rp'レジスタ(BA/HL/IX/IY)の内容をrpレジスタ(BA/HL/IX/IY)にロードします。

モード Src: レジスタ直接
Dst: レジスタ直接

例

設定値		結果					
rp	rp'	rp	rp'	SC			
				N	V	C	Z
3521H	E964H	E964H	E964H	—	—	—	—

(rp≠rp')

コード

MSB							LSB	
1	1	0	0	1	1	1	1	CFH

1	1	1	0	rp	rp'	※

※

rp \ rp'	BA (0,0)	HL (0,1)	IX (1,0)	IY (1,1)
BA(0,0)	E0H	E1H	E2H	E3H
HL(0,1)	E4H	E5H	E6H	E7H
IX(1,0)	E8H	E9H	EAH	EBH
IY(1,1)	ECH	EDH	EEH	EFH

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

LD BA, PC ////////////////// Load PC into BA reg. // 2 cycle ///

機 能 BA ← PC + 2

プログラムカウンタ(PC)の内容をBAレジスタにロードします。ロード値は本命令の先頭アドレス+2となります。

モード Src: レジスタ直接
Dst: レジスタ直接

例

設定値		結果					
BA	PC	BA	PC	SC			
				N	V	C	Z
3521H	E964H	E964H	E964H	—	—	—	—

コード

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
1	1	1	1	1	0	0	1	F9H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

LD BA, SP ////////////////// Load SP into BA reg. // 2 cycle ///

機 能 BA ← SP

スタックポインタ(SP)の内容をBAレジスタにロードします。

モード Src: レジスタ直接
Dst: レジスタ直接

例

設定値		結果					
BA	SP	BA	SP	SC			
				N	V	C	Z
3521H	E964H	E964H	E964H	—	—	—	—

コード

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
1	1	1	1	1	0	0	0	F8H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

LD HL, PC ////////////////// Load PC into HL reg. // 2 cycle ///

機 能 HL ← PC + 2

プログラムカウンタ(PC)の内容をHLレジスタにロードします。ロード値は本命令の先頭アドレス+2となります。

モード Src: レジスタ直接
Dst: レジスタ直接

例

設定値		結果					
HL	PC	HL	PC	SC			
				N	V	C	Z
3521H	E964H	E964H	E964H	—	—	—	—

コード

MSB LSB

1	1	0	0	1	1	1	1	CFH

1	1	1	1	0	1	0	1	F5H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

LD HL, SP ////////////////// Load SP into HL reg. // 2 cycle ///

機 能 HL ← SP

スタックポインタ(SP)の内容をHLレジスタにロードします。

モード Src: レジスタ直接
Dst: レジスタ直接

例

設定値		結果					
HL	SP	HL	SP	SC			
				N	V	C	Z
3521H	E964H	E964H	E964H	—	—	—	—

コード

MSB LSB

1	1	0	0	1	1	1	1	CFH

1	1	1	1	0	1	0	0	F4H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

LD IX, SP ////////////////////// Load SP into IX reg. // 2 cycle ///

機 能 IX ← SP

スタックポインタ(SP)の内容をIXレジスタにロードします。

モード Src: レジスタ直接
Dst: レジスタ直接

例

設定値		結果					
IX	SP	IX	SP	SC			
				N	V	C	Z
3521H	E964H	E964H	E964H	—	—	—	—

コード

MSB LSB

1	1	0	0	1	1	1	1	CFH

1	1	1	1	1	0	1	0	FAH

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

LD IY, SP ////////////////////// Load SP into IY reg. // 2 cycle ///

機 能 IY ← SP

スタックポインタ(SP)の内容をIYレジスタにロードします。

モード Src: レジスタ直接
Dst: レジスタ直接

例

設定値		結果					
IY	SP	IY	SP	SC			
				N	V	C	Z
3521H	E964H	E964H	E964H	—	—	—	—

コード

MSB LSB

1	1	0	0	1	1	1	1	CFH

1	1	1	1	1	1	1	0	FEH

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

LD SP, rp *Load rp reg. into SP* 2 cycle

機 能 SP ← rp

rpレジスタ(BA/HL/IX/IY)の内容をスタックポインタ(SP)にロードします。

モード Src: レジスタ直接
Dst: レジスタ直接

例

設定値		結果					
SP	rp	SP	rp	SC			
				N	V	C	Z
3521H	E964H	E964H	E964H	—	—	—	—

コード

MSB LSB

1	1	0	0	1	1	1	1	CFH

1	1	1	1	0	0	rp	F0H～F3H ※

※

rp		ニーモニック	コード
BA	00	LD SP, BA	F0H
HL	01	LD SP, HL	F1H
IX	10	LD SP, IX	F2H
IY	11	LD SP, IY	F3H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

LD [hhll], rp *Load rp reg. into location [hhll]* 5 cycle

機 能 [hh*ll*] ← rp(L), [hh*ll*+1] ← rp(H)

rpレジスタ(BA/HL/IX/IY)の下位バイトをデータメモリのアドレスhh*ll*に、上位バイトをその次のアドレスhh*ll*+1にストアします。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: レジスタ直接
Dst: 16ビット絶対

例

設定値	結果						
rp	[hh*ll*]	[hh*ll*+1]	rp	SC			
				N	V	C	Z
E964H	64H	E9H	E964H	—	—	—	—

コード

MSB LSB

1	0	1	1	1	1	rp	BCH～BFH ※

l	*l*	*ll*

h	h	hh

※

rp		ニーモニック	コード
BA	00	LD [hh*ll*], BA	BCH
HL	01	LD [hh*ll*], HL	BDH
IX	10	LD [hh*ll*], IX	BEH
IY	11	LD [hh*ll*], IY	BFH

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

LD [hhll], SP *Load SP into location [hhll]* *6 cycle*

機 能 **[hh*ll*] ← SP(L), [hh*ll*+1] ← SP(H)**

スタックポインタ(SP)の下位バイトをデータメモリのアドレスhh*ll*に、上位バイトをその次のアドレスhh*ll*+1にストアします。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード **Src: レジスタ直接**
Dst: 16ビット絶対

例

設定値		結果					
SP	[hh*ll*]	[hh*ll*+1]	SP	SC			
				N	V	C	Z
E964H	64H	E9H	E964H	—	—	—	—

コード

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
0	1	1	1	1	1	0	0	7CH
l	*l*	*l*	*l*	*l*	*l*	*l*	*l*	*ll*
h	h	h	h	h	h	h	h	hh

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

LD [HL], rp *Load rp reg. into location [HL]* *5 cycle*

機 能 **[HL] ← rp(L), [HL+1] ← rp(H)**

rpレジスタ(BA/HL/IX/IY)の下位バイトをHLレジスタの内容で指定されたデータメモリのアドレスに、上位バイトをその次のアドレスにストアします。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード **Src: レジスタ直接**
Dst: レジスタ間接

例

設定値		結果					
rp	[HL]	[HL+1]	rp	SC			
				N	V	C	Z
E964H	64H	E9H	E964H	—	—	—	—

コード

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
1	1	0	0	0	1	rp		C4H～C7H ※

※

rp		ニーモニック	コード
BA	00	LD [HL], BA	C4H
HL	01	LD [HL], HL	C5H
IX	10	LD [HL], IX	C6H
IY	11	LD [HL], IY	C7H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

LD [IX], rp //////////////////// *Load rp reg. into location [IX]* // *5 cycle* ///

機 能 [IX] ← rp(L), [IX+1] ← rp(H)

rpレジスタ(BA/HL/IX/IY)の下位バイトをIXレジスタの内容で指定されたデータメモリのアドレスに、上位バイトをその次のアドレスにストアします。XPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: レジスタ直接
Dst: レジスタ間接

例

設定値	結果						
rp	[IX]	[IX+1]	rp	SC			
				N	V	C	Z
E964H	64H	E9H	E964H	—	—	—	—

コード

MSB LSB

1	1	0	0	1	1	1	1	CFH

1	1	0	1	0	1	rp	D4H～D7H ※

※

rp		ニーモニック	コード
BA	00	LD [IX], BA	D4H
HL	01	LD [IX], HL	D5H
IX	10	LD [IX], IX	D6H
IY	11	LD [IX], IY	D7H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

LD [IY], rp //////////////////// *Load rp reg. into location [IY]* // *5 cycle* ///

機 能 [IY] ← rp(L), [IY+1] ← rp(H)

rpレジスタ(BA/HL/IX/IY)の下位バイトをIYレジスタの内容で指定されたデータメモリのアドレスに、上位バイトをその次のアドレスにストアします。YPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: レジスタ直接
Dst: レジスタ間接

例

設定値	結果						
rp	[IY]	[IY+1]	rp	SC			
				N	V	C	Z
E964H	64H	E9H	E964H	—	—	—	—

コード

MSB LSB

1	1	0	0	1	1	1	1	CFH

1	1	0	1	1	1	rp	DCH～DFH ※

※

rp		ニーモニック	コード
BA	00	LD [IY], BA	DCH
HL	01	LD [IY], HL	DDH
IX	10	LD [IY], IX	DEH
IY	11	LD [IY], IY	DFH

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

LD [SP+dd], rp ////// *Load rp reg. into location [SP + dd]* ////////////////////////////////////// *6 cycle* ///

機 能 **[SP+dd] ← rp(L), [SP+dd+1] ← rp(H)**

rpレジスタ(BA/HL/IX/IY)の下位バイトをSPの内容とディスプレースメントddの和で指定されたデータメモリのアドレスに、上位バイトをその次のアドレスにストアします。
ddは符号付きデータとして扱われ、範囲は-128～127です。

モード **Src: レジスタ直接**
Dst: ディスプレースメント付きレジスタ間接

例

設定値		結果					
rp	[SP+dd]	[SP+dd+1]	rp	SC			
				N	V	C	Z
E964H	64H	E9H	E964H	—	—	—	—

コード

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
0	1	1	1	0	1	rp		74H～77H ※
d	d							dd

※

rp		ニーモニック	コード
BA	00	LD [SP+dd], BA	74H
HL	01	LD [SP+dd], HL	75H
IX	10	LD [SP+dd], IX	76H
IY	11	LD [SP+dd], IY	77H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

LD rp, #mmnn /////// *Load immediate data mmnn into rp reg.* ////////////////////////////////// *3 cycle* ///

機 能 **rp ← mmnn**

16ビット即値データmmnnをrpレジスタ(BA/HL/IX/IY)にロードします。

モード **Src: 即値データ**
Dst: レジスタ直接

例

設定値		結果					
rp	mmnn	rp	mmnn	SC			
				N	V	C	Z
3521H	E964H	E964H	E964H	—	—	—	—

コード

MSB							LSB	
1	1	0	0	0	1	rp		C4H～C7H ※
n	n							nn
m	m							mm

※

rp		ニーモニック	コード
BA	00	LD BA, #mmnn	C4H
HL	01	LD HL, #mmnn	C5H
IX	10	LD IX, #mmnn	C6H
IY	11	LD IY, #mmnn	C7H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

LD SP, #mmnn ////// Load immediate data mmnn into SP //////////////////////////////////// 4 cycle ///

機 能 SP ← mmnn

16ビット即値データmmnnをスタックポインタ(SP)にロードします。

モード Src: 即値データ
Dst: レジスタ直接

例

設定値		結果					
SP	mmnn	SP	mmnn	SC			
				N	V	C	Z
3521H	E964H	E964H	E964H	—	—	—	—

コード

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
0	1	1	0	1	1	1	0	6EH
n	n	n	n	n	n	n	n	nn
m	m	m	m	m	m	m	m	mm

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

LD rp, [hhll] ////////////// Load location [hhll] into rp reg. // 5 cycle ///

機 能 rp(L) ← [hh*ll*], rp(H) ← [hh*ll*+1]

16ビット絶対アドレスhh*ll*でアドレス指定されたデータメモリの内容を下位バイト、その次のアドレスの内容を上位バイトとしてrpレジスタ(BA/HL/IX/IY)にロードします。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: 16ビット絶対
Dst: レジスタ直接

例

設定値			結果					
rp	[hh*ll*]	[hh*ll*+1]	rp	SC				
				N	V	C	Z	
3521H	64H	E9H	E964H	—	—	—	—	

コード

MSB							LSB	
1	0	1	1	1	0	rp		B8H〜BBH ※
l	*l*	*l*	*l*	*l*	*l*	*l*	*l*	*ll*
h	h	h	h	h	h	h	h	hh

※

rp		ニーモニック	コード
BA	00	LD BA, [hh*ll*]	B8H
HL	01	LD HL, [hh*ll*]	B9H
IX	10	LD IX, [hh*ll*]	BAH
IY	11	LD IY, [hh*ll*]	BBH

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

LD SP, [hhll] /////////////// Load location [hhll] into SP // 6 cycle ///

機 能 SP(L) ← [hh*ll*], SP(H) ← [hh*ll*+1]

16ビット絶対アドレスhh*ll*でアドレス指定されたデータメモリの内容を下位バイト、その次のアドレスの内容を上位バイトとしてスタックポインタ(SP)にロードします。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: 16ビット絶対
Dst: レジスタ直接

例

設定値			結果				
SP	[hh*ll*]	[hh*ll*+1]	SP	SC			
				N	V	C	Z
3521H	64H	E9H	E964H	—	—	—	—

コード

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
0	1	1	1	1	0	0	0	78H
				l	*l*			*ll*
				h	h			hh

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

LD rp, [HL] ///////////////// Load location [HL] into rp reg. // 5 cycle ///

機 能 rp(L) ← [HL], rp(H) ← [HL+1]

HLレジスタでアドレス指定されたデータメモリの内容を下位バイト、その次のアドレスの内容を上位バイトとしてrpレジスタ(BA/HL/IX/IY)にロードします。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: レジスタ間接
Dst: レジスタ直接

例

設定値			結果				
rp	[HL]	[HL+1]	rp	SC			
				N	V	C	Z
3521H	64H	E9H	E964H	—	—	—	—

コード

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
1	1	0	0	0	0	rp		C0H～C3H ※

※

rp		ニーモニック	コード
BA	00	LD BA, [HL]	C0H
HL	01	LD HL, [HL]	C1H
IX	10	LD IX, [HL]	C2H
IY	11	LD IY, [HL]	C3H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

LD rp, [IX] //////////////////// *Load location [IX] into rp reg.* // *5 cycle* ///

機 能 rp(L) ← [IX], rp(H) ← [IX+1]

IXレジスタでアドレス指定されたデータメモリの内容を下位バイト、その次のアドレスの内容を上位バイトとしてrpレジスタ(BA/HL/IX/IY)にロードします。
XPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: レジスタ間接
Dst: レジスタ直接

例

設定値			結果				
rp	[IX]	[IX+1]	rp	SC			
				N	V	C	Z
3521H	64H	E9H	E964H	—	—	—	—

コード

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
1	1	0	1	0	0	rp		D0H～D3H ※

※

rp		ニーモニック	コード
BA	00	LD BA, [IX]	D0H
HL	01	LD HL, [IX]	D1H
IX	10	LD IX, [IX]	D2H
IY	11	LD IY, [IX]	D3H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

LD rp, [IY] //////////////////// *Load location [IY] into rp reg.* // *5 cycle* ///

機 能 rp(L) ← [IY], rp(H) ← [IY+1]

IYレジスタでアドレス指定されたデータメモリの内容を下位バイト、その次のアドレスの内容を上位バイトとしてrpレジスタ(BA/HL/IX/IY)にロードします。
YPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: レジスタ間接
Dst: レジスタ直接

例

設定値			結果				
rp	[IY]	[IY+1]	rp	SC			
				N	V	C	Z
3521H	64H	E9H	E964H	—	—	—	—

コード

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
1	1	0	1	1	0	rp		D8H～DBH ※

※

rp		ニーモニック	コード
BA	00	LD BA, [IY]	D8H
HL	01	LD HL, [IY]	D9H
IX	10	LD IX, [IY]	DAH
IY	11	LD IY, [IY]	DBH

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

LD rp, [SP+dd] ////// *Load location [SP + dd] into rp reg.* ////////////////////////////////////// *6 cycle* ///

機 能 rp(L) ← [SP+dd], rp(H) ← [SP+dd+1]

スタックポインタ(SP)の内容とディスプレースメントddの和でアドレス指定されたデータメモリの内容を下位バイト、その次のアドレスの内容を上位バイトとしてrpレジスタ(BA/HL/IX/IY)にロードします。

ddは符号付きデータとして扱われ、範囲は-128～127です。

モード Src: ディスプレースメント付きレジスタ間接
Dst: レジスタ直接

例

設定値			結果				
rp	[SP+dd]	[SP+dd+1]	rp	SC			
				N	V	C	Z
3521H	64H	E9H	E964H	—	—	—	—

コード

MSB LSB

1	1	0	0	1	1	1	1	CFH

0	1	1	1	0	0	rp		70H～73H ※

						d	d	dd

※

rp		ニーモニック	コード
BA	00	LD BA, [SP+dd]	70H
HL	01	LD HL, [SP+dd]	71H
IX	10	LD IX, [SP+dd]	72H
IY	11	LD IY, [SP+dd]	73H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

MLT ////////////////////////////////////// *Multiply* // *12 cycle* ///

機 能 HL ← L * A

Lレジスタの内容とAレジスタの内容の乗算を行い、結果をHLレジスタにストアします。

モード インプライド (レジスタ直接)

例

設定値		結果				
L	A	HL	SC			
			N	V	C	Z
00H	64H	0000H	0	0	0	1
64H	58H	2260H	0	0	0	0
A5H	93H	5EBFH	0	0	0	0
C8H	A5H	80E8H	1	0	0	0

コード

MSB LSB

1	1	0	0	1	1	1	0	CEH

1	1	0	1	1	0	0	0	D8H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	0	0	↕

注 意 MODEL0/2では本命令が使用できません。

NEG r ////////////////////////// Negate r reg. // 3 cycle ///

機 能 r ← 0 - r

rレジスタ(A/B)の内容を0から減算(2の補数を作成)し、結果をrレジスタにストアします。

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	1	0	0	1	0	r	A4H、A5H ※

※

r		ニーモニック	コード
A	0	NEG A	A4H
B	1	NEG B	A5H

フラグ

I1	I0	U	D	N	V	C	Z
−	−	★	★	↕	↕	↕	↕

モード レジスタ直接

例

設定値		結果			
r	r	SC			
		N	V	C	Z
D=0, U=0の場合					
57H	A9H	1	0	1	0
00H	00H	0	0	0	1
2BH	D5H	1	0	1	0
80H	80H	1	1	1	0
D=1, U=0の場合					
57	43	0	0	1	0
D=1, U=1の場合					
57	03	0	0	1	0

NEG [BR:ll] ////////////// Negate location [BR:ll] // 5 cycle ///

機 能 [BR:*ll*] ← 0 - [BR:*ll*]

BRレジスタの内容を上位バイト、8ビット絶対アドレス*ll*を下位バイトとしてアドレス指定されるデータメモリの内容を0から減算(2の補数を作成)し、結果をそのアドレスにストアします。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	1	0	0	1	1	0	A6H
				l	*l*			*ll*

フラグ

I1	I0	U	D	N	V	C	Z
−	−	★	★	↕	↕	↕	↕

モード 8ビット絶対

例

設定値		結果			
[BR:*ll*]	[BR:*ll*]	SC			
		N	V	C	Z
D=0, U=0の場合					
57H	A9H	1	0	1	0
00H	00H	0	0	0	1
2BH	D5H	1	0	1	0
80H	80H	1	1	1	0
D=1, U=0の場合					
57	43	0	0	1	0
D=1, U=1の場合					
57	03	0	0	1	0

NEG [HL] ////////////////////// Negate location [HL] // 4 cycle ///

機 能 [HL] ← 0 - [HL]

HLレジスタでアドレス指定されたデータメモリの内容を0から減算(2の補数を作成)し、結果をそのアドレスにストアします。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	1	0	0	1	1	1	A7H

フラグ

I1	I0	U	D	N	V	C	Z
−	−	★	★	↕	↕	↕	↕

モード レジスタ間接

例

設定値		結果			
[HL]	[HL]	SC			
		N	V	C	Z
D=0, U=0の場合					
57H	A9H	1	0	1	0
00H	00H	0	0	0	1
2BH	D5H	1	0	1	0
80H	80H	1	1	1	0
D=1, U=0の場合					
57	43	0	0	1	0
D=1, U=1の場合					
57	03	0	0	1	0

NOP ////////////////////////////////// *No Operation* // *2 cycle* ///

機 能 **No Operation**

他に影響を与える動作は行わずに2サイクルを費やします。プログラムカウンタ(PC)はインクリメント(+1)されます。

コード

MSB							LSB	
1	1	1	1	1	1	1	1	FFH

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

OR A, r ////////////////////////// *Logical OR of r reg. and A reg.* // *2 cycle* ///

機 能 **A ← A ∨ r**

rレジスタ(A/B)の内容とAレジスタの内容との論理和をとり、結果をAレジスタにストアします。

モード **Src: レジスタ直接**
Dst: レジスタ直接

コード

MSB							LSB	
0	0	1	0	1	0	0	r	28H、29H ※

※

r		ニーモニック	コード
A	0	OR A, A	28H
B	1	OR A, B	29H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	—	↕

例

設定値		結果				
A	B	A	SC			
			N	V	C	Z
32H	6CH	7EH	0	—	—	0
86H	41H	C7H	1	—	—	0

OR A, #nn ////////////////// *Logical OR of immediate data nn and A reg.* ////////////////////////// *2 cycle* ///

機 能 **A ← A ∨ nn**

8ビット即値データnnとAレジスタの内容との論理和をとり、結果をAレジスタにストアします。

モード **Src: 即値データ**
Dst: レジスタ直接

コード

MSB							LSB	
0	0	1	0	1	0	1	0	2AH
				n	n			nn

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	—	↕

例

設定値		結果				
A	nn	A	SC			
			N	V	C	Z
32H	6CH	7EH	0	—	—	0
86H	41H	C7H	1	—	—	0

OR A, [BR:ll] ////////////// Logical OR of location [BR:ll] and A reg. ////////////////////////////// 3 cycle ///

機 能 A ← A ∨ [BR:*ll*]

BRレジスタの内容を上位バイト、8ビット絶対アドレス*ll*を下位バイトとしてアドレス指定されるデータメモリの内容とAレジスタの内容との論理和をとり、結果をAレジスタにストアします。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: 8ビット絶対
Dst: レジスタ直接

例

設定値		結果				
A	[BR:*ll*]	A	SC			
			N	V	C	Z
32H	6CH	7EH	0	—	—	0
86H	41H	C7H	1	—	—	0

コード

MSB LSB

0	0	1	0	1	1	0	0	2CH
				l	*l*			*ll*

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	—	↕

OR A, [hhll] ////////////// Logical OR of location [hhll] and A reg. ////////////////////////////// 4 cycle ///

機 能 A ← A ∨ [hh*ll*]

16ビット絶対アドレスhh*ll*でアドレス指定されたデータメモリの内容とAレジスタの内容との論理和をとり、結果をAレジスタにストアします。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: 16ビット絶対
Dst: レジスタ直接

例

設定値		結果				
A	[hh*ll*]	A	SC			
			N	V	C	Z
32H	6CH	7EH	0	—	—	0
86H	41H	C7H	1	—	—	0

コード

MSB LSB

0	0	1	0	1	1	0	1	2DH
				l	*l*			*ll*
				h	h			hh

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	—	↕

OR A, [HL] ////////////////// Logical OR of location [HL] and A reg. ////////////////////////////// 2 cycle ///

機 能 A ← A ∨ [HL]

HLレジスタでアドレス指定されたデータメモリの内容とAレジスタの内容との論理和をとり、結果をAレジスタにストアします。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: レジスタ間接
Dst: レジスタ直接

例

設定値		結果				
A	[HL]	A	SC			
			N	V	C	Z
32H	6CH	7EH	0	—	—	0
86H	41H	C7H	1	—	—	0

コード

MSB LSB

0	0	1	0	1	0	1	1	2BH

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	—	↕

OR A, [ir] *Logical OR of location [ir reg.] and A reg. 2 cycle*

機 能 A ← A ∨ [ir]

irレジスタ(IX/IY)でアドレス指定されたデータメモリの内容とAレジスタの内容との論理和をとり、結果をAレジスタにストアします。
XPレジスタ(IX指定時)、YPレジスタ(IY指定時)の内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: レジスタ間接
Dst: レジスタ直接

例

設定値		結果				
A	[ir]	A	SC			
			N	V	C	Z
32H	6CH	7EH	0	—	—	0
86H	41H	C7H	1	—	—	0

コード

MSB							LSB	
0	0	1	0	1	1	1	ir	2EH、2FH ※

※

ir		ニーモニック	コード
IX	0	OR A, [IX]	2EH
IY	1	OR A, [IY]	2FH

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	—	↕

OR A, [ir+dd] *Logical OR of location [ir reg. + dd] and A reg. 4 cycle*

機 能 A ← A ∨ [ir+dd]

irレジスタ(IX/IY)の内容とディスプレースメントddの和でアドレス指定されたデータメモリの内容とAレジスタの内容との論理和をとり、結果をAレジスタにストアします。
ddは符号付きデータとして扱われ、範囲は-128～127です。
XPレジスタ(IX指定時)、YPレジスタ(IY指定時)の内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: ディスプレースメント付きレジスタ間接
Dst: レジスタ直接

例

設定値		結果				
A	[ir+dd]	A	SC			
			N	V	C	Z
32H	6CH	7EH	0	—	—	0
86H	41H	C7H	1	—	—	0

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	1	0	1	0	0	ir	28H、29H ※
d	d	d	d	d	d	d	d	dd

※

ir		ニーモニック	コード
IX	0	OR A, [IX+dd]	28H
IY	1	OR A, [IY+dd]	29H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	—	↕

OR A, [ir+L] /////////////// Logical OR of location [ir reg. + L] and A reg. ////////////////////// 4 cycle ///

機 能 A ← A ∨ [ir+L]

irレジスタ(IX/IY)の内容とLレジスタの内容の和でアドレス指定されたデータメモリの内容とAレジスタの内容との論理和をとり、結果をAレジスタにストアします。
Lレジスタの内容は符号付きデータとして扱われ、範囲は-128～127です。
XPレジスタ(IX指定時)、YPレジスタ(IY指定時)の内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: インデックスレジスタ付きレジスタ間接
Dst: レジスタ直接

例

設定値		結果				
A	[ir+L]	A	SC			
			N	V	C	Z
32H	6CH	7EH	0	—	—	0
86H	41H	C7H	1	—	—	0

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	1	0	1	0	1	ir	2AH、2BH ※

※

ir		ニーモニック	コード
IX	0	OR A, [IX+L]	2AH
IY	1	OR A, [IY+L]	2BH

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	—	↕

OR B, #nn ////////////////// Logical OR of immediate data nn and B reg. /////////////////////////// 3 cycle ///

機 能 B ← B ∨ nn

8ビット即値データnnとBレジスタの内容との論理和をとり、結果をBレジスタにストアします。

モード Src: 即値データ
Dst: レジスタ直接

例

設定値		結果				
B	nn	B	SC			
			N	V	C	Z
32H	6CH	7EH	0	—	—	0
86H	41H	C7H	1	—	—	0

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	1	1	0	1	0	0	B4H
			n	n				nn

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	—	↕

OR L, #nn ////////////////// Logical OR of immediate data nn and L reg. /////////////////////////// 3 cycle ///

機 能 L ← L ∨ nn

8ビット即値データnnとLレジスタの内容との論理和をとり、結果をLレジスタにストアします。

モード Src: 即値データ
Dst: レジスタ直接

例

設定値		結果				
L	nn	L	SC			
			N	V	C	Z
32H	6CH	7EH	0	—	—	0
86H	41H	C7H	1	—	—	0

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	1	1	0	1	0	1	B5H
			n	n				nn

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	—	↕

OR H, #nn *////////////////// Logical OR of immediate data nn and H reg. ////////////////////////// 3 cycle ///*

機 能 H ← H ∨ nn

8ビット即値データnnとHレジスタの内容との論理和をとり、結果をHレジスタにストアします。

モード Src: 即値データ
Dst: レジスタ直接

例

設定値		結果				
H	nn	H	SC			
			N	V	C	Z
32H	6CH	7EH	0	—	—	0
86H	41H	C7H	1	—	—	0

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	1	1	0	1	1	0	B6H
n	n							nn

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	—	↕

OR SC, #nn */////////////// Logical OR of immediate data nn and SC ////////////////////////////////// 3 cycle ///*

機 能 SC ← SC ∨ nn

8ビット即値データnnとシステムコンディションフラグ(SC)の内容との論理和をとり、結果をシステムコンディションフラグ(SC)にセットします。

モード Src: 即値データ
Dst: レジスタ直接

例

設定値		結果							
SC	nn	SC	SC						
			I1	I0	U	D	N	V	C	Z
32H	6CH	7EH	0	1	1	1	1	1	1	0
86H	41H	C7H	1	1	0	0	0	1	1	1

コード

MSB							LSB	
1	0	0	1	1	1	0	1	9DH
n	n							nn

フラグ

I1	I0	U	D	N	V	C	Z
↑	↑	↑	↑	↑	↑	↑	↑

OR [BR:ll], #nn *///// Logical OR of immediate data nn and location [BR:ll] /////////// 5 cycle ///*

機 能 [BR:*ll*] ← [BR:*ll*] ∨ nn

8ビット即値データnnとBRレジスタの内容を上位バイト、8ビット絶対アドレス*ll*を下位バイトとしてアドレス指定されるデータメモリの内容との論理和をとり、結果をそのアドレスにストアします。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: 即値データ
Dst: 8ビット絶対

例

設定値		結果				
[BR:*ll*]	nn	[BR:*ll*]	SC			
			N	V	C	Z
32H	6CH	7EH	0	—	—	0
86H	41H	C7H	1	—	—	0

コード

MSB							LSB	
1	1	0	1	1	0	0	1	D9H
l	*l*							*ll*
n	n							nn

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	—	↕

OR [HL], A ////////////////// Logical OR of A reg. and location [HL] ////////////////////////////////// 4 cycle ///

機 能 [HL] ← [HL] ∨ A

Aレジスタの内容とHLレジスタでアドレス指定されたデータメモリの内容との論理和をとり、結果をそのアドレスにストアします。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: レジスタ直接
Dst: レジスタ間接

例

設定値		結果				
[HL]	A	[HL]	SC			
			N	V	C	Z
32H	6CH	7EH	0	—	—	0
86H	41H	C7H	1	—	—	0

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	1	0	1	1	0	0	2CH

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	—	↕

OR [HL], #nn //////////// Logical OR of immediate data nn and location [HL] ////////////// 5 cycle ///

機 能 [HL] ← [HL] ∨ nn

8ビット即値データnnとHLレジスタでアドレス指定されたデータメモリの内容との論理和をとり、結果をそのアドレスにストアします。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: 即値データ
Dst: レジスタ間接

例

設定値		結果				
[HL]	nn	[HL]	SC			
			N	V	C	Z
32H	6CH	7EH	0	—	—	0
86H	41H	C7H	1	—	—	0

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	1	0	1	1	0	1	2DH
n	n							nn

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	—	↕

OR [HL], [ir] ////////////// Logical OR of location [ir reg.] and location [HL] ////////////////// 5 cycle ///

機 能 [HL] ← [HL] ∨ [ir]

irレジスタ(IX/IY)とHLレジスタでそれぞれアドレス指定されたデータメモリの内容の論理和をとり、結果をデータメモリ[HL]にストアします。
EPレジスタの内容がデータメモリ[HL]のページアドレス、XPレジスタ(IX指定時)、YPレジスタ(IY指定時)の内容がデータメモリ[ir]のページアドレスになります(MODEL2/3)。

モード Src: レジスタ間接
Dst: レジスタ間接

例

設定値		結果				
[HL]	[ir]	[HL]	SC			
			N	V	C	Z
32H	6CH	7EH	0	—	—	0
86H	41H	C7H	1	—	—	0

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	1	0	1	1	1	ir	2EH、2FH ※

※

ir		ニーモニック	コード
IX	0	OR [HL], [IX]	2EH
IY	1	OR [HL], [IY]	2FH

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	—	↕

PACK ////////////////////////////// *Pack BA reg. to A reg.* // *2 cycle* ///

機 能 B A → A

*	m	*	n	→	m	n

BAレジスタの内容をパックし、Aレジスタにストアします。Aレジスタの上位4ビットがBレジスタの下位4ビットの内容に置き換えられます。

モード インプライド(レジスタ直接)

例

設定値	結果					
BA	A	B	SC			
			N	V	C	Z
38C4H	84H	38H	—	—	—	—

コード

MSB							LSB	
1	1	0	1	1	1	1	0	DEH

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

POP r ////////////////////////////// *Pop top of stack into r reg.* // *3 cycle* ///

機 能 r ← [SP], SP ← SP + 1

スタックポインタ(SP)が示すアドレスの内容をrレジスタ(A/B/L/H)にロードし、SPをインクリメント(+1)します。

モード レジスタ直接

例 POP Aを実行。

	A	SP
実行前	82H	FFFFH
実行後	5AH	0000H

Stack: 5AH — 00FFFFH

コード

MSB							LSB	
1	1	0	0	1	1	1	1	CFH

1	0	1	1	0	1	r	B4H〜B7H ※

※

r		ニーモニック	コード
A	00	POP A	B4H
B	01	POP B	B5H
L	10	POP L	B6H
H	11	POP H	B7H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

POP rp ////////////////////////////// *Pop top of stack into rp reg.* // *3 cycle* ///

機 能 $rp_{(L)}$ ← [SP], $rp_{(H)}$ ← [SP+1], SP ← SP + 2

スタックポインタ(SP)が示すアドレスからの2バイトの内容をrpレジスタ(BA/HL/IX/IY)の下位バイト、上位バイトの順にロードし、SPに2を加算します。

モード レジスタ直接

例 POP BAを実行。

	B	A	SP
実行前	67H	05H	FFFEH
実行後	23H	5AH	0000H

Stack: 5AH — 00FFFEH; 23H — 00FFFFH

コード

MSB							LSB	
1	0	1	0	1	0	rp		A8H〜ABH ※

※

rp		ニーモニック	コード
BA	00	POP BA	A8H
HL	01	POP HL	A9H
IX	10	POP IX	AAH
IY	11	POP IY	ABH

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

POP BR ////////////////////// Pop top of stack into BR reg. // 2 cycle ///

機 能 BR ← [SP], SP ← SP + 1

スタックポインタ(SP)が示すアドレスの内容をBRレジスタにロードし、SPをインクリメント(+1)します。

モード レジスタ直接

例

	BR	SP
実行前	82H	FFFFH
実行後	5AH	0000H

Stack: 5AH (00FFFFH)

コード

MSB							LSB	
1	0	1	0	1	1	0	0	ACH

フラグ

I1	I0	U	D	N	V	C	Z
−	−	−	−	−	−	−	−

POP EP ////////////////////// Pop top of stack into EP reg. // 2 cycle ///

機 能 EP ← [SP], SP ← SP + 1

スタックポインタ(SP)が示すアドレスの内容をEPレジスタにロードし、SPをインクリメント(+1)します。

モード レジスタ直接

例

	EP	SP
実行前	82H	FFFFH
実行後	5AH	0000H

Stack: 5AH (00FFFFH)

コード

MSB							LSB	
1	0	1	0	1	1	0	1	ADH

フラグ

I1	I0	U	D	N	V	C	Z
−	−	−	−	−	−	−	−

注 意 MODEL0/1では本命令が使用できません。

POP IP ////////////////////// Pop top of stack into IP reg. // 3 cycle ///

機 能 YP ← [SP], XP ← [SP+1], SP ← SP + 2

スタックポインタ(SP)が示すアドレスからの2バイトの内容をYPレジスタ、XPレジスタの順にロードし、SPに2を加算します。

モード レジスタ直接

例 POP IPを実行。

	XP	YP	SP
実行前	67H	05H	FFFEH
実行後	23H	5AH	0000H

Stack: 5AH (00FFFEH), 23H (00FFFFH)

コード

MSB							LSB	
1	0	1	0	1	1	1	0	AEH

フラグ

I1	I0	U	D	N	V	C	Z
−	−	−	−	−	−	−	−

注 意 MODEL0/1では本命令が使用できません。

POP SC ////////////////////// Pop top of stack into SC // 2 cycle ///

機 能 SC ← [SP], SP ← SP + 1

スタックポインタ(SP)が示すアドレスの内容をシステムコンディションフラグ(SC)にセットし、SPをインクリメント(+1)します。

モード レジスタ直接

例

	SC	SP
実行前	82H	FFFFH
実行後	5AH	0000H

Stack: 5AH (00FFFFH)

コード

MSB							LSB	
1	0	1	0	1	1	1	1	AFH

フラグ

I1	I0	U	D	N	V	C	Z
↕	↕	↕	↕	↕	↕	↕	↕

POP ALE /////////////////// Pop all registers including expand registers ////////////////////////// 14 cycle ///

機 能 **POP IP, EP, BR, IY, IX, HL, BA**

スタックポインタ(SP)が示すアドレスからの内容をIP(YP/XP)、EP、BR、IY、IX、HL、BAレジスタの順にポップします。"PUSH ALE"命令で一括して退避させたレジスタの内容を一度に復帰できます。SPにはポップしたバイト数分の12が加算されます。MODEL2/3で使用するレジスタを復帰します。

モード **インプライド(レジスタ直接)**

例 SP=FFF4Hの場合、POP) ALEを実行すると、下図のようにスタックの内容が各レジスタに戻されSP=0000Hとなります。

00FFFAH	IX(L)	00FFF4H	YP
00FFFBH	IX(H)	00FFF5H	XP
00FFFCH	L	00FFF6H	EP
00FFFDH	H	00FFF7H	BR
00FFFEH	A	00FFF8H	IY(L)
00FFFFH	B	00FFF9H	IY(H)

コード

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
1	0	1	1	1	1	0	1	BDH

フラグ

I1	I0	U	D	N	V	C	Z
↕	↕	↕	↕	↕	↕	↕	↕

注 意 **MODEL0/1では本命令が使用できません。**

POP ALL /////////////////// Pop all registers // 11 cycle ///

機 能 **POP BR, IY, IX, HL, BA**

スタックポインタ(SP)が示すアドレスからの内容をBR、IY、IX、HL、BAレジスタの順にポップします。"PUSH ALL"命令で一括して退避させたレジスタの内容を一度に復帰できます。SPにはポップしたバイト数分の9が加算されます。MODEL0/1で使用するレジスタを復帰します。

モード **インプライド(レジスタ直接)**

例 SP=FFF7Hの場合、POP) ALLを実行すると、下図のようにスタックの内容が各レジスタに戻されSP=0000Hとなります。

00FFFCH	L	00FFF7H	BR
00FFFDH	H	00FFF8H	IY(L)
00FFFEH	A	00FFF9H	IY(H)
00FFFFH	B	00FFFAH	IX(L)
		00FFFBH	IX(H)

コード

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
1	0	1	1	1	1	0	0	BCH

フラグ

I1	I0	U	D	N	V	C	Z
↕	↕	↕	↕	↕	↕	↕	↕

PUSH r /////////////////////////// Push r reg. onto stack // 3 cycle ///

機 能 **[SP-1] ← r, SP ← SP - 1**

rレジスタ(A/B/L/H)の内容をスタックポインタ(SP)から1を減算した内容で示されるアドレスにストアします。SPはデクリメント(-1)されます。

モード **レジスタ直接**

例 PUSH) Aを実行。

	A	SP
実行前	5AH	0000H
実行後	5AH	FFFFH

Stack: 5AH (A) 00FFFFH

コード

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
1	0	1	1	0	0	r		B0H～B3H ※

※

r		ニーモニック	コード
A	00	PUSH A	B0H
B	01	PUSH B	B1H
L	10	PUSH L	B2H
H	11	PUSH H	B3H

フラグ

I1	I0	U	D	N	V	C	Z
−	−	−	−	−	−	−	−

PUSH rp ////////////////////// Push rp reg. onto stack // 4 cycle ///

機 能 [SP-1] ← rp(H), [SP-2] ← rp(L), SP ← SP - 2

rpレジスタ(BA/HL/IX/IY)の内容を上位バイト、下位バイトの順にスタックポインタ(SP)から1を減算した内容で示されるアドレスと、さらにひとつ前のアドレスにストアします。
SPからは2が減算されます。

モード レジスタ直接

例 PUSH BAを実行。

	BA	SP	Stack	
実行前	235AH	0000H		
			5AH (A)	00FFFEH
実行後	235AH	FFFEH	23H (B)	00FFFFH

コード

MSB						LSB	
1	0	1	0	0	0	rp	A0H～A3H ※

※

rp		ニーモニック	コード
BA	00	PUSH BA	A0H
HL	01	PUSH HL	A1H
IX	10	PUSH IX	A2H
IY	11	PUSH IY	A3H

フラグ

I1	I0	U	D	N	V	C	Z
−	−	−	−	−	−	−	−

PUSH BR ////////////////////// Push BR reg. onto stack // 3 cycle ///

機 能 [SP-1] ← BR, SP ← SP - 1

BRレジスタの内容をスタックポインタ(SP)から1を減算した内容で示されるアドレスにストアします。SPはデクリメント(-1)されます。

モード レジスタ直接

例

	BR	SP	Stack	
実行前	5AH	0000H		
実行後	5AH	FFFFH	5AH (BR)	00FFFFH

コード

MSB							LSB	
1	0	1	0	0	1	0	0	A4H

フラグ

I1	I0	U	D	N	V	C	Z
−	−	−	−	−	−	−	−

PUSH EP ////////////////////// Push EP reg. onto stack // 3 cycle ///

機 能 [SP-1] ← EP, SP ← SP - 1

EPレジスタの内容をスタックポインタ(SP)から1を減算した内容で示されるアドレスにストアします。SPはデクリメント(-1)されます。

モード レジスタ直接

例

	EP	SP	Stack	
実行前	5AH	0000H		
実行後	5AH	FFFFH	5AH (EP)	00FFFFH

コード

MSB							LSB	
1	0	1	0	0	1	0	1	A5H

フラグ

I1	I0	U	D	N	V	C	Z
−	−	−	−	−	−	−	−

注 意 MODEL0/1では本命令が使用できません。

PUSH IP ////////////////////// *Push IP reg. onto stack* // *4 cycle* ///

機 能 **[SP-1] ← XP, [SP-2] ← YP, SP ← SP - 2**

YPレジスタ、XPレジスタの内容をそれぞれスタックポインタ(SP)から1を減算した内容で示されるアドレスと、さらにひとつ前のアドレスにストアします。SPからは2が減算されます。

コード

MSB							LSB	
1	0	1	0	0	1	1	0	A6H

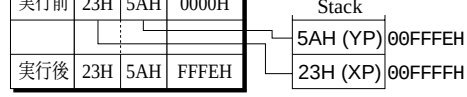
フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

モード **レジスタ直接**

例 PUSH IPを実行。

	XP	YP	SP
実行前	23H	5AH	0000H
実行後	23H	5AH	FFFEH



注 意 **MODEL0/1では本命令が使用できません。**

PUSH SC ////////////////////// *Push SC onto stack* // *3 cycle* ///

機 能 **[SP-1] ← SC, SP ← SP - 1**

システムコンディションフラグ(SC)の内容をスタックポインタ(SP)から1を減算した内容で示されるアドレスにストアします。SPはデクリメント(-1)されます。

コード

MSB							LSB	
1	0	1	0	0	1	1	1	A7H

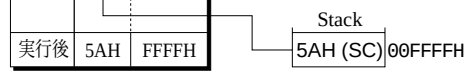
フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

モード **レジスタ直接**

例

	SC	SP
実行前	5AH	0000H
実行後	5AH	FFFFH



PUSH ALE ////////////////// *Push all registers including expand registers* ////////////////////// *15 cycle* ///

機 能 **PUSH BA, HL, IX, IY, BR, EP, IP**

BA、HL、IX、IY、BR、EP、IP(XP/YP)の順にレジスタの内容をスタックポインタ(SP)が示すアドレスから下位のアドレスに向けてプッシュします。SPはプッシュしたバイト数分の12が減算されます。
MODEL2/3で使用するレジスタを退避します。

コード

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
1	0	1	1	1	0	0	1	B9H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

モード **インプライド(レジスタ直接)**

例 SP=0000Hの場合、PUSH ALEを実行すると、下図のようにレジスタがスタックされSP=FFF4Hとなります。

00FFFAH	IX(L)	00FFF4H	YP
00FFFBH	IX(H)	00FFF5H	XP
00FFFCH	L	00FFF6H	EP
00FFFDH	H	00FFF7H	BR
00FFFEH	A	00FFF8H	IY(L)
00FFFFH	B	00FFF9H	IY(H)

注 意 **MODEL0/1では本命令が使用できません。**

PUSH ALL ////////////////// *Push all registers* // *12 cycle* ///

機 能 **PUSH BA, HL, IX, IY, BR**

BA、HL、IX、IY、BRの順にレジスタの内容をスタックポインタ(SP)が示すアドレスから下位のアドレスに向けてプッシュします。SPはプッシュしたバイト数分の9が減算されます。
MODEL0/1で使用するレジスタを退避します。

コード

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
1	0	1	1	1	0	0	0	B8H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

モード **インプライド(レジスタ直接)**

例 SP=0000Hの場合、PUSH ALLを実行すると、下図のようにレジスタがスタックされSP=FFF7Hとなります。

00FFFCH	L	00FFF7H	BR
00FFFDH	H	00FFF8H	IY(L)
00FFFEH	A	00FFF9H	IY(H)
00FFFFH	B	00FFFAH	IX(L)
		00FFFBH	IX(H)

RET ////////////////////////////////// Return from subroutine ////////////////////////////////// 3(MIN)/4(MAX) cycle ///

機 能 **■MODEL0/1, MODEL2/3(ミニマムモード)**

$PC_{(L)} \leftarrow [SP]$, $PC_{(H)} \leftarrow [SP+1]$, $SP \leftarrow SP+2$

■MODEL2/3(マキシマムモード)

$PC_{(L)} \leftarrow [SP]$, $PC_{(H)} \leftarrow [SP+1]$, $CB \leftarrow [SP+2]$, $NB \leftarrow CB$, $SP \leftarrow SP+3$

スタックポインタ(SP)で示されるアドレスからの2バイトをプログラムカウンタ(PC)の下位バイト、上位バイトの順にロードして、サブルーチンからリターンします。

MODEL2/3のマキシマムモードでは、続く1バイトをバンクアドレスとしてCBにロードし、呼び出し元のバンクに戻します。同時に、そのバンクアドレスをNBにも再設定します。

SPは復帰させたバイト数分(ミニマムモード: 2バイト、マキシマムモード: 3バイト)だけ加算されます。

コード

MSB							LSB	
1	1	1	1	1	0	0	0	F8H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

例 MODEL2/3マキシマムモードにおいて物理アドレス013100H番地にあるRET命令を実行。

	NB	CB	PC(論理アドレス)	SP
実行前	02H	02H	B100H	FFFDH
実行後	01H	01H	9003H	0000H

Stack	
03H($PC_{(L)}$)	00FFFDH
90H($PC_{(H)}$)	00FFFEH
01H(CB)	00FFFFH

上記の例では物理アドレス009003Hへ復帰します。
MODEL2/3ミニマムモードでは、CBがスタックされていませんのでCB, NBは変化しません。
MODEL0/1ではCB, NBはありません。

注 意 **例外処理ルーチンからのリターンには次の"RETE"命令を使用してください。**

RETE ////////////////////////////////// Return from exception processing routine ////////////// 4(MIN)/5(MAX) cycle ///

機 能 **■MODEL0/1, MODEL2/3(ミニマムモード)**

SC ← [SP], $PC_{(L)} \leftarrow [SP+1]$, $PC_{(H)} \leftarrow [SP+2]$, $SP \leftarrow SP+3$

■MODEL2/3(マキシマムモード)

SC ← [SP], $PC_{(L)} \leftarrow [SP+1]$, $PC_{(H)} \leftarrow [SP+2]$, $CB \leftarrow [SP+3]$, $NB \leftarrow CB$, $SP \leftarrow SP+4$

スタックポインタ(SP)で示されるアドレスからの3バイトをシステムコンディションフラグ(SC)、プログラムカウンタ(PC)の下位バイト、上位バイトの順にロードして、例外処理ルーチンからリターンします。

MODEL2/3のマキシマムモードでは、続く1バイトをバンクアドレスとしてCBにロードし、割り込み発生時のバンクに戻します。同時に、そのバンクアドレスをNBにも再設定します。

SPは復帰させたバイト数分(ミニマムモード: 3バイト、マキシマムモード: 4バイト)だけ加算されます。

コード

MSB							LSB	
1	1	1	1	1	0	0	1	F9H

フラグ

I1	I0	U	D	N	V	C	Z
↕	↕	↕	↕	↕	↕	↕	↕

例 MODEL2/3マキシマムモードにおいて物理アドレス013100H番地にあるRETE命令を実行。

	NB	CB	PC(論理アドレス)	SP	SC
実行前	02H	02H	B100H	FFFCH	91H
実行後	01H	01H	9003H	0000H	54H

Stack	
54H(SC)	00FFFCH
03H($PC_{(L)}$)	00FFFDH
90H($PC_{(H)}$)	00FFFEH
01H(CB)	00FFFFH

上記の例では物理アドレス009003Hへ復帰します。
MODEL2/3ミニマムモードでは、CBがスタックされていませんのでCB, NBは変化しません。
MODEL0/1ではCB, NBはありません。

RETS ////////////////////////////////// Return from subroutine then skip 2 byte ////////////////// 5(MIN)/6(MAX) cycle ///

機 能 ■MODEL0/1, MODEL2/3(ミニマムモード)

PC(L) ← [SP], PC(H) ← [SP+1], SP ← SP + 2, PC ← PC + 2

■MODEL2/3(マキシマムモード)

PC(L) ← [SP], PC(H) ← [SP+1], CB ← [SP+2], NB ← CB, SP ← SP + 3, PC ← PC + 2

"RET"命令を実行後、リターンしたアドレスの2バイト命令をスキップします。

コード

MSB							LSB	
1	1	1	1	1	0	1	0	FAH

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

例 MODEL2/3マキシマムモードにおいて物理アドレス013100H番地にあるRET命令を実行。

	NB	CB	PC(論理アドレス)	SP
実行前	02H	02H	B100H	FFFDH
実行後	01H	01H	9005H	0000H

+2

Stack	
03H(PC(L))	00FFFDH
90H(PC(H))	00FFFEH
01H(CB)	00FFFFH

上記の例では物理アドレス009005Hへ復帰します。
MODEL2/3ミニマムモードでは、CBがスタックされていませんのでCB, NBは変化しません。
MODEL0/1ではCB, NBはありません。

RL r ////////////////////////////////// Rotate left r reg. with carry // 3 cycle ///

機 能 C ← 7 6 5 4 3 2 1 0 ← r

rレジスタ(A/B)の内容をキャリー(C)を含めて1ビット分左に回転させます。キャリー(C)の内容がレジスタのビット0に、レジスタのビット7はキャリー(C)に移動します。

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	0	1	0	0	0	r	90H、91H ※

※

r		ニーモニック	コード
A	0	RL A	90H
B	1	RL B	91H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	↕	↕

モード レジスタ直接

例

設定値		結果				
r	C	r	SC			
			N	V	C	Z
83H	0	06H	0	—	1	0
4CH	0	98H	1	—	0	0
A2H	1	45H	0	—	1	0

RL [BR:ll] //////////////////// Rotate left location [BR:ll] with carry ////////////////////////////////////// 5 cycle ///

機 能 C ← 7 6 5 4 3 2 1 0 ← [BR:*ll*]

BRレジスタの内容を上位バイト、8ビット絶対アドレス*ll*を下位バイトとしてアドレス指定されたデータメモリの内容をキャリー(C)を含めて1ビット分左に回転させます。キャリー(C)の内容がデータのビット0に、データのビット7はキャリー(C)に移動します。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード 8ビット絶対

例

設定値		結果				
[BR:*ll*]	C	[BR:*ll*]	SC			
			N	V	C	Z
83H	0	06H	0	—	1	0
4CH	0	98H	1	—	0	0
A2H	1	45H	0	—	1	0

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	0	1	0	0	1	0	92H
				l	*l*			*ll*

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	↕	↕

RL [HL] //////////////////////// Rotate left location [HL] with carry ////////////////////////////////////// 4 cycle ///

機 能 C ← 7 6 5 4 3 2 1 0 ← [HL]

HLレジスタで指定されたデータメモリの内容をキャリー(C)を含めて1ビット分左に回転させます。キャリー(C)の内容がデータのビット0に、データのビット7はキャリー(C)に移動します。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード レジスタ間接

例

設定値		結果				
[HL]	C	[HL]	SC			
			N	V	C	Z
83H	0	06H	0	—	1	0
4CH	0	98H	1	—	0	0
A2H	1	45H	0	—	1	0

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	0	1	0	0	1	1	93H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	↕	↕

RLC r ////////////////////////////// Rotate left r reg. circular // 3 cycle ///

機 能 C ← 7 6 5 4 3 2 1 0 ← r

rレジスタ(A/B)の内容を1ビット分左に回転させます。レジスタのビット7がビット0およびキャリー(C)に移動します。

モード レジスタ直接

例

設定値		結果				
r	C	r	SC			
			N	V	C	Z
E3H	0	C7H	1	—	1	0
3BH	1	76H	0	—	0	0

コード

MSB LSB

1	1	0	0	1	1	1	0	CEH

1	0	0	1	0	1	0	r	94H、95H ※

※

r		ニーモニック	コード
A	0	RLC A	94H
B	1	RLC B	95H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	↕	↕

RLC [BR:ll] ////////////// Rotate left location [BR:ll] circular ////////////////////////////////////// 5 cycle ///

機 能 C ← 7 6 5 4 3 2 1 0 ← [BR:*ll*]

BRレジスタの内容を上位バイト、8ビット絶対アドレス*ll*を下位バイトとしてアドレス指定されたデータメモリの内容を1ビット分左に回転させます。データのビット7がビット0およびキャリー(C)に移動します。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード 8ビット絶対

例

設定値		結果				
[BR:*ll*]	C	[BR:*ll*]	SC			
			N	V	C	Z
E3H	0	C7H	1	—	1	0
3BH	1	76H	0	—	0	0

コード

MSB LSB

1	1	0	0	1	1	1	0	CEH

1	0	0	1	0	1	1	0	96H

				l	*l*			*ll*

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	↕	↕

RLC [HL] ////////////////////// Rotate left location [HL] circular // 4 cycle ///

機 能 C ← 7 6 5 4 3 2 1 0 ← [HL]

HLレジスタで指定されたデータメモリの内容を1ビット分左に回転させます。データのビット7がビット0およびキャリー(C)に移動します。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード レジスタ間接

例

設定値		結果				
[HL]	C	[HL]	SC			
			N	V	C	Z
E3H	0	C7H	1	—	1	0
3BH	1	76H	0	—	0	0

コード

MSB LSB

1	1	0	0	1	1	1	0	CEH

1	0	0	1	0	1	1	1	97H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	↕	↕

RR r ////////////////////////////////// *Rotate right r reg. with carry* // *3 cycle* ///

機 能 ▶7 6 5 4 3 2 1 0▶C r

rレジスタ(A/B)の内容をキャリー(C)を含めて1ビット分右に回転させます。キャリー(C)の内容がレジスタのビット7に、レジスタのビット0はキャリー(C)に移動します。

モード レジスタ直接

例

設定値		結果				
r	C	r	SC			
			N	V	C	Z
7EH	0	3FH	0	—	0	0
51H	0	28H	0	—	1	0
D4H	1	EAH	1	—	0	0

コード

MSB LSB

1	1	0	0	1	1	1	0	CEH

1	0	0	1	1	0	0	r	98H、99H ※

※

r		ニーモニック	コード
A	0	RR A	98H
B	1	RR B	99H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	↕	↕

RR [BR:ll] ////////////////// *Rotate right location [BR:ll] with carry* ////////////////////////////////// *5 cycle* ///

機 能 ▶7 6 5 4 3 2 1 0▶C [BR:*ll*]

BRレジスタの内容を上位バイト、8ビット絶対アドレス*ll*を下位バイトとしてアドレス指定されたデータメモリの内容をキャリー(C)を含めて1ビット分右に回転させます。キャリー(C)の内容がデータのビット7に、データのビット0はキャリー(C)に移動します。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード 8ビット絶対

例

設定値		結果				
[BR:*ll*]	C	[BR:*ll*]	SC			
			N	V	C	Z
7EH	0	3FH	0	—	0	0
51H	0	28H	0	—	1	0
D4H	1	EAH	1	—	0	0

コード

MSB LSB

1	1	0	0	1	1	1	0	CEH

1	0	0	1	1	0	1	0	9AH

			l	*l*				*ll*

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	↕	↕

RR [HL] ////////////////////// *Rotate right location [HL] with carry* ////////////////////////////////// *4 cycle* ///

機 能 ↳7|6|5|4|3|2|1|0→C┘ [HL]

HLレジスタで指定されたデータメモリの内容をキャリー(C)を含めて1ビット分右に回転させます。キャリー(C)の内容がデータのビット7に、データのビット0はキャリー(C)に移動します。EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード レジスタ間接

例

設定値		結果				
[HL]	C	[HL]	SC			
			N	V	C	Z
7EH	0	3FH	0	—	0	0
51H	0	28H	0	—	1	0
D4H	1	EAH	1	—	0	0

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	0	1	1	0	1	1	9BH

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	↕	↕

RRC r ////////////////////////////// *Rotate right r reg. circular* // *3 cycle* ///

機 能 ↳7|6|5|4|3|2|1|0→C r

rレジスタ(A/B)の内容を1ビット分右に回転させます。レジスタのビット0がビット7およびキャリー(C)に移動します。

モード レジスタ直接

例

設定値		結果				
r	C	r	SC			
			N	V	C	Z
C6H	1	63H	0	—	0	0
D7H	0	EBH	1	—	1	0

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	0	1	1	1	0	r	9CH、9DH ※

※

r		ニーモニック	コード
A	0	RRC A	9CH
B	1	RRC B	9DH

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	↕	↕

RRC [BR:ll] //////////////// Rotate right location [BR:ll] circular ////////////////////////////////////// 5 cycle ///

機 能 ▶7 6 5 4 3 2 1 0▶C [BR:*ll*]

BRレジスタの内容を上位バイト、8ビット絶対アドレス*ll*を下位バイトとしてアドレス指定されたデータメモリの内容を1ビット分右に回転させます。データのビット0がビット7およびキャリー(C)に移動します。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード 8ビット絶対

例

設定値		結果				
[BR:*ll*]	C	[BR:*ll*]	SC			
			N	V	C	Z
C6H	1	63H	0	—	0	0
D7H	0	EBH	1	—	1	0

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	0	1	1	1	1	0	9EH
				l	*l*			*ll*

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	↕	↕

RRC [HL] //////////////////// Rotate right location [HL] circuler // 4 cycle ///

機 能 ▶7 6 5 4 3 2 1 0▶C [HL]

HLレジスタで指定されたデータメモリの内容を1ビット分右に回転させます。データのビット0がビット7およびキャリー(C)に移動します。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード レジスタ間接

例

設定値		結果				
[HL]	C	[HL]	SC			
			N	V	C	Z
C6H	1	63H	0	—	0	0
D7H	0	EBH	1	—	1	0

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	0	1	1	1	1	1	9FH

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	↕	↕

SBC A, r ////////////////////// Subtract with carry r reg. from A reg. ////////////////////////////////////// 2 cycle ///

機 能 A ← A - r - C

rレジスタ(A/B)の内容とキャリー(C)をAレジスタから減算します。

コード

MSB							LSB	
0	0	0	1	1	0	0	r	18H、19H ※

※

r		ニーモニック	コード
A	0	SBC A, A	18H
B	1	SBC A, B	19H

フラグ

I1	I0	U	D	N	V	C	Z
−	−	★	★	↕	↕	↕	↕

モード Src: レジスタ直接
Dst: レジスタ直接

例

設定値			結果				
A	B	C	A	SC			
				N	V	C	Z
D=0, U=0の場合							
A8H	42H	1	65H	0	1	0	0
36H	5AH	1	DBH	1	0	1	0
D=1, U=0の場合							
88	39	1	48	0	0	0	0
D=1, U=1の場合							
88	39	1	08	0	0	1	0

SBC A, #nn ///////////////// Subtract with carry immediate data nn from A reg. ///////////////// 2 cycle ///

機 能 A ← A - nn - C

8ビット即値データnnとキャリー(C)をAレジスタから減算します。

コード

MSB							LSB	
0	0	0	1	1	0	1	0	1AH
n	n							nn

フラグ

I1	I0	U	D	N	V	C	Z
−	−	★	★	↕	↕	↕	↕

モード Src: 即値データ
Dst: レジスタ直接

例

設定値			結果				
A	nn	C	A	SC			
				N	V	C	Z
D=0, U=0の場合							
A8H	42H	1	65H	0	1	0	0
36H	5AH	1	DBH	1	0	1	0
D=1, U=0の場合							
88	39	1	48	0	0	0	0
D=1, U=1の場合							
88	39	1	08	0	0	1	0

SBC A, [BR:ll] //////// Subtract with carry location [BR:ll] from A reg. ////////////////////// 3 cycle ///

機 能 A ← A - [BR:*ll*] - C

BRレジスタの内容を上位バイト、8ビット絶対アドレス*ll*を下位バイトとしてアドレス指定されたデータメモリの内容とキャリー(C)をAレジスタから減算します。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

コード

MSB							LSB	
0	0	0	1	1	1	0	0	1CH
l	*l*							*ll*

フラグ

I1	I0	U	D	N	V	C	Z
−	−	★	★	↕	↕	↕	↕

モード Src: 8ビット絶対
Dst: レジスタ直接

例

設定値			結果				
A	[BR:*ll*]	C	A	SC			
				N	V	C	Z
D=0, U=0の場合							
A8H	42H	1	65H	0	1	0	0
36H	5AH	1	DBH	1	0	1	0
D=1, U=0の場合							
88	39	1	48	0	0	0	0
D=1, U=1の場合							
88	39	1	08	0	0	1	0

SBC A, [hhll] ////////////// Subtract with carry location [hhll] from A reg. ////////////////////// 4 cycle ///

機 能 A ← A - [hh*ll*] - C

16ビット絶対アドレスhh*ll*でアドレス指定されたデータメモリの内容とキャリー(C)をAレジスタから減算します。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

コード

MSB							LSB	
0	0	0	1	1	1	0	1	1DH
l	*l*							*ll*
h	h							hh

フラグ

I1	I0	U	D	N	V	C	Z
−	−	★	★	↕	↕	↕	↕

モード Src: 16ビット絶対
Dst: レジスタ直接

例

設定値			結果				
A	[hh*ll*]	C	A	SC			
				N	V	C	Z
D=0, U=0の場合							
A8H	42H	1	65H	0	1	0	0
36H	5AH	1	DBH	1	0	1	0
D=1, U=0の場合							
88	39	1	48	0	0	0	0
D=1, U=1の場合							
88	39	1	08	0	0	1	0

SBC A, [HL] ////////////// Subtract with carry location [HL] from A reg. ////////////////////// 2 cycle ///

機 能 A ← A - [HL] - C

HLレジスタでアドレス指定されたデータメモリの内容とキャリー(C)をAレジスタから減算します。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

コード

MSB							LSB	
0	0	0	1	1	0	1	1	1BH

フラグ

I1	I0	U	D	N	V	C	Z
−	−	★	★	↕	↕	↕	↕

モード Src: レジスタ間接
Dst: レジスタ直接

例

設定値			結果				
A	[HL]	C	A	SC			
				N	V	C	Z
D=0, U=0の場合							
A8H	42H	1	65H	0	1	0	0
36H	5AH	1	DBH	1	0	1	0
D=1, U=0の場合							
88	39	1	48	0	0	0	0
D=1, U=1の場合							
88	39	1	08	0	0	1	0

SBC A, [ir] ////////////////// Subtract with carry location [ir reg.] from A reg. ////////////////////// 2 cycle ///

機 能 A ← A - [ir] - C

irレジスタ(IX/IY)でアドレス指定されたデータメモリの内容とキャリー(C)をAレジスタから減算します。
XPレジスタ(IX指定時)、YPレジスタ(IY指定時)の内容がデータメモリのページアドレスになります(MODEL2/3)。

コード

MSB							LSB	
0	0	0	1	1	1	1	ir	1EH、1FH ※

※

ir		ニーモニック	コード
IX	0	SBC A, [IX]	1EH
IY	1	SBC A, [IY]	1FH

フラグ

I1	I0	U	D	N	V	C	Z
−	−	★	★	↕	↕	↕	↕

モード Src: レジスタ間接
Dst: レジスタ直接

例

設定値			結果				
A	[ir]	C	A	SC			
				N	V	C	Z
D=0, U=0の場合							
A8H	42H	1	65H	0	1	0	0
36H	5AH	1	DBH	1	0	1	0
D=1, U=0の場合							
88	39	1	48	0	0	0	0
D=1, U=1の場合							
88	39	1	08	0	0	1	0

SBC A, [ir+dd] /////// Subtract with carry location [ir reg. + dd] from A reg. ////////// 4 cycle ///

機 能 A ← A - [ir+dd] - C

irレジスタ(IX/IY)の内容とディスプレースメントddの和でアドレス指定されたデータメモリの内容とキャリー(C)をAレジスタから減算します。
ddは符号付きデータとして扱われ、範囲は-128～127です。
XPレジスタ(IX指定時)、YPレジスタ(IY指定時)の内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: ディスプレースメント付きレジスタ間接
Dst: レジスタ直接

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	0	1	1	0	0	ir	18H、19H ※
d	d	d	d	d	d	d	d	dd

※

ir		ニーモニック	コード
IX	0	SBC A, [IX+dd]	18H
IY	1	SBC A, [IY+dd]	19H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	★	★	↕	↕	↕	↕

例

設定値			結果				
A	[ir+dd]	C	A	SC			
				N	V	C	Z
D=0, U=0の場合							
A8H	42H	1	65H	0	1	0	0
36H	5AH	1	DBH	1	0	1	0
D=1, U=0の場合							
88	39	1	48	0	0	0	0
D=1, U=1の場合							
88	39	1	08	0	0	1	0

SBC A, [ir+L] //////////// Subtract with carry location [ir reg. + L] from A reg. ////////// 4 cycle ///

機 能 A ← A - [ir+L] - C

irレジスタ(IX/IY)の内容とLレジスタの内容の和でアドレス指定されたデータメモリの内容とキャリー(C)をAレジスタから減算します。
Lレジスタの内容は符号付きデータとして扱われ、範囲は-128～127です。
XPレジスタ(IX指定時)、YPレジスタ(IY指定時)の内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: インデックスレジスタ付きレジスタ間接
Dst: レジスタ直接

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	0	1	1	0	1	ir	1AH、1BH ※

※

ir		ニーモニック	コード
IX	0	SBC A, [IX+L]	1AH
IY	1	SBC A, [IY+L]	1BH

フラグ

I1	I0	U	D	N	V	C	Z
—	—	★	★	↕	↕	↕	↕

例

設定値			結果				
A	[ir+L]	C	A	SC			
				N	V	C	Z
D=0, U=0の場合							
A8H	42H	1	65H	0	1	0	0
36H	5AH	1	DBH	1	0	1	0
D=1, U=0の場合							
88	39	1	48	0	0	0	0
D=1, U=1の場合							
88	39	1	08	0	0	1	0

SBC [HL], A *//////////////// Subtract with carry A reg. from location [HL] ////////////////////// 4 cycle ///*

機 能 **[HL] ← [HL] - A - C**

Aレジスタの内容とキャリー(C)をHLレジスタでアドレス指定されたデータメモリから減算します。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード **Src: レジスタ直接**
Dst: レジスタ間接

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	0	1	1	1	0	0	1CH

フラグ

I1	I0	U	D	N	V	C	Z
−	−	★	★	↕	↕	↕	↕

例

設定値			結果				
[HL]	A	C	[HL]	SC			
				N	V	C	Z
D=0, U=0の場合							
A8H	42H	1	65H	0	1	0	0
36H	5AH	1	DBH	1	0	1	0
D=1, U=0の場合							
88	39	1	48	0	0	0	0
D=1, U=1の場合							
88	39	1	08	0	0	1	0

SBC [HL], #nn *//////// Subtract with carry immediate data nn from location [HL] // 5 cycle ///*

機 能 **[HL] ← [HL] - nn - C**

8ビット即値データnnとキャリー(C)をHLレジスタでアドレス指定されたデータメモリから減算します。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード **Src: 即値データ**
Dst: レジスタ間接

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	0	1	1	1	0	1	1DH
n	n							nn

フラグ

I1	I0	U	D	N	V	C	Z
−	−	★	★	↕	↕	↕	↕

例

設定値			結果				
[HL]	nn	C	[HL]	SC			
				N	V	C	Z
D=0, U=0の場合							
A8H	42H	1	65H	0	1	0	0
36H	5AH	1	DBH	1	0	1	0
D=1, U=0の場合							
88	39	1	48	0	0	0	0
D=1, U=1の場合							
88	39	1	08	0	0	1	0

SBC [HL], [ir] *////////// Subtract with carry location [ir reg.] from location [HL] ///// 5 cycle ///*

機 能 **[HL] ← [HL] - [ir] - C**

irレジスタ(IX/IY)でアドレス指定されたデータメモリの内容とキャリー(C)をHLレジスタでアドレス指定されるデータメモリから減算します。
EPレジスタの内容がデータメモリ[HL]のページアドレス、XPレジスタ(IX指定時)、YPレジスタ(IY指定時)の内容がデータメモリ[ir]のページアドレスになります(MODEL2/3)。

モード **Src: レジスタ間接**
Dst: レジスタ間接

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	0	1	1	1	1	ir	1EH、1FH ※

※

ir		ニーモニック	コード
IX	0	SBC [HL], [IX]	1EH
IY	1	SBC [HL], [IY]	1FH

フラグ

I1	I0	U	D	N	V	C	Z
−	−	★	★	↕	↕	↕	↕

例

設定値			結果				
[HL]	[ir]	C	[HL]	SC			
				N	V	C	Z
D=0, U=0の場合							
A8H	42H	1	65H	0	1	0	0
36H	5AH	1	DBH	1	0	1	0
D=1, U=0の場合							
88	39	1	48	0	0	0	0
D=1, U=1の場合							
88	39	1	08	0	0	1	0

SBC BA, rp *Subtract with carry rp reg. from BA reg. 4 cycle*

機 能 BA ← BA - rp - C

rpレジスタ(BA/HL/IX/IY)の内容とキャリー(C)をBAレジスタから減算します。

モード Src: レジスタ直接
Dst: レジスタ直接

例

設定値			結果				
BA	rp	C	BA	SC			
				N	V	C	Z
63C2H	2125H	1	429CH	0	1	0	0
205CH	7120H	1	AF3BH	1	0	1	0

(rp≠BA)

コード

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
0	0	0	0	1	1	rp		0CH～0FH ※

※

rp		ニーモニック	コード
BA	00	SBC BA, BA	0CH
HL	01	SBC BA, HL	0DH
IX	10	SBC BA, IX	0EH
IY	11	SBC BA, IY	0FH

フラグ

I1	I0	U	D	N	V	C	Z
−	−	−	−	↕	↕	↕	↕

SBC BA, #mmnn *Subtract with carry immediate data mmnn from BA reg. 6 cycle*

機 能 BA ← BA - mmnn - C

16ビット即値データmmnnとキャリー(C)をBAレジスタから減算します。

モード Src: 即値データ
Dst: レジスタ直接

例

設定値			結果				
BA	mmnn	C	BA	SC			
				N	V	C	Z
63C2H	2125H	1	429CH	0	1	0	0
205CH	7120H	1	AF3BH	1	0	1	0

コード

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
0	1	1	0	0	0	1	0	62H
n	n							nn
m	m							mm

フラグ

I1	I0	U	D	N	V	C	Z
−	−	−	−	↕	↕	↕	↕

SBC HL, rp *Subtract with carry rp reg. from HL reg. 4 cycle*

機 能 HL ← HL - rp - C

rpレジスタ(BA/HL/IX/IY)の内容とキャリー(C)をHLレジスタから減算します。

モード Src: レジスタ直接
Dst: レジスタ直接

例

設定値			結果				
HL	rp	C	HL	SC			
				N	V	C	Z
63C2H	2125H	1	429CH	0	1	0	0
205CH	7120H	1	AF3BH	1	0	1	0

(rp≠HL)

コード

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
0	0	1	0	1	1	rp		2CH～2FH ※

※

rp		ニーモニック	コード
BA	00	SBC HL, BA	2CH
HL	01	SBC HL, HL	2DH
IX	10	SBC HL, IX	2EH
IY	11	SBC HL, IY	2FH

フラグ

I1	I0	U	D	N	V	C	Z
−	−	−	−	↕	↕	↕	↕

SBC HL, #mmnn /// Subtract with carry immediate data mmnn from HL reg. ////// 4 cycle ///

機 能 HL ← HL - mmnn - C

16ビット即値データmmnnとキャリー(C)をHLレジスタから減算します。

モード Src: 即値データ
Dst: レジスタ直接

例

設定値			結果				
HL	mmnn	C	HL	SC			
				N	V	C	Z
63C2H	2125H	1	429CH	0	1	0	0
205CH	7120H	1	AF3BH	1	0	1	0

コード

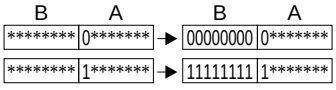
MSB							LSB	
1	1	0	0	1	1	1	1	CFH
0	1	1	0	0	0	1	1	63H
n	n							nn
m	m							mm

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	↕	↕	↕

SEP ////////////////////////////////////// *Sign expand A reg. to BA reg.* // *3 cycle ///*

機 能



Aレジスタにストアされている8ビットデータの符号ビット(ビット7)をBレジスタに拡張し、BAレジスタとして扱う16ビットデータにします。
Aレジスタの値が正(ビット7が'0')の場合にBレジスタは00Hに、負(ビット7が'1')の場合はFFHになります。

モード インプライド(レジスタ直接)

例

設定値		結果					
B	A	BA	SC				
			N	V	C	Z	
5CH	76H	0076H	—	—	—	—	
42H	A5H	FFA5H	—	—	—	—	

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	1	0	1	0	0	0	A8H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

SLA r ////////////////////////////// Shift r reg. left arithmetic // 3 cycle ///

機 能 C←7 6 5 4 3 2 1 0←0 r

rレジスタ(A/B)の内容を1ビット左にシフトします。レジスタのビット7はキャリー(C)に移動し、レジスタのビット0には'0'が入ります。
"SLL"命令と同様の結果が得られますが、"SLA"命令は算術シフトのため、オーバーフロー(V)フラグも変化します。

モード レジスタ直接

例

設定値	結果				
r	r	SC			
		N	V	C	Z
00111100	01111000	0	0	0	0
10010000	00100000	0	1	1	0

コード

MSB LSB

1	1	0	0	1	1	1	0	CEH

1	0	0	0	0	0	0	r	80H、81H ※

※

r		ニーモニック	コード
A	0	SLA A	80H
B	1	SLA B	81H

フラグ

I1	I0	U	D	N	V	C	Z
−	−	−	−	↕	↕	↕	↕

SLA [BR:ll] ////////////////// Shift location [BR:ll] left arithmetic // 5 cycle ///

機 能 C←7 6 5 4 3 2 1 0←0 [BR:*ll*]

BRレジスタの内容を上位バイト、8ビット絶対アドレス*ll*を下位バイトとしてアドレス指定されたデータメモリの内容を1ビット左にシフトします。データのビット7はキャリー(C)に移動し、データのビット0は'0'になります。
"SLL"命令と同様の結果が得られますが、"SLA"命令は算術シフトのため、オーバーフロー(V)フラグも変化します。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード 8ビット絶対

例

設定値	結果				
[BR:*ll*]	[BR:*ll*]	SC			
		N	V	C	Z
00111100	01111000	0	0	0	0
10010000	00100000	0	1	1	0

コード

MSB LSB

1	1	0	0	1	1	1	0	CEH

1	0	0	0	0	0	1	0	82H

				l	*l*			*ll*

フラグ

I1	I0	U	D	N	V	C	Z
−	−	−	−	↕	↕	↕	↕

SLA [HL] ////////////////////// Shift location [HL] left arithmetic // 4 cycle ///

機 能 C←7 6 5 4 3 2 1 0←0 [HL]

HLレジスタで指定されたデータメモリの内容を1ビット左にシフトします。データのビット7はキャリー(C)に移動し、データのビット0は'0'になります。
"SLL"命令と同様の結果が得られますが、"SLA"命令は算術シフトのため、オーバーフロー(V)フラグも変化します。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード レジスタ間接

例

設定値	結果				
[HL]	[HL]	SC			
		N	V	C	Z
00111100	01111000	0	0	0	0
10010000	00100000	0	1	1	0

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	0	0	0	0	1	1	83H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	↕	↕	↕

SLL r ////////////////////////////////// Shift r reg. left logical // 3 cycle ///

機 能 C←7 6 5 4 3 2 1 0←0 r

rレジスタ(A/B)の内容を1ビット左にシフトします。レジスタのビット7はキャリー(C)に移動し、レジスタのビット0には'0'が入ります。
"SLA"命令と同様の結果が得られますが、"SLL"命令は論理シフトのため、オーバーフロー(V)フラグは変化しません。

モード レジスタ直接

例

設定値	結果				
r	r	SC			
		N	V	C	Z
00111100	01111000	1	—	0	0
10010000	00100000	0	—	1	0

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	0	0	0	1	0	r	84H、85H ※

※

r		ニーモニック	コード
A	0	SLL A	84H
B	1	SLL B	85H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	↕	↕

SLL [BR:ll] ////////////////// Shift location [BR:ll] left logical // 5 cycle ///

機 能 C←7 6 5 4 3 2 1 0←0 [BR:*ll*]

BRレジスタの内容を上位バイト、8ビット絶対アドレス*ll*を下位バイトとしてアドレス指定されたデータメモリの内容を1ビット左にシフトします。データのビット7はキャリー(C)に移動し、データのビット0は'0'になります。
"SLA"命令と同様の結果が得られますが、"SLL"命令は論理シフトのため、オーバーフロー(V)フラグは変化しません。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード 8ビット絶対

例

設定値	結果				
[BR:*ll*]	[BR:*ll*]	SC			
		N	V	C	Z
00111100	01111000	1	—	0	0
10010000	00100000	0	—	1	0

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	0	0	0	1	1	0	86H
				l	*l*			*ll*

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	↕	↕

SLL [HL] ////////////////////// Shift location [HL] left logical // 4 cycle ///

機 能 C←7 6 5 4 3 2 1 0←0 [HL]

HLレジスタで指定されたデータメモリの内容を1ビット左にシフトします。データのビット7はキャリー(C)に移動し、データのビット0は'0'になります。
"SLA"命令と同様の結果が得られますが、"SLL"命令は論理シフトのため、オーバーフロー(V)フラグは変化しません。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード レジスタ間接

例

設定値	結果				
[HL]	[HL]	SC			
		N	V	C	Z
00111100	01111000	1	—	0	0
10010000	00100000	0	—	1	0

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	0	0	0	1	1	1	87H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	↕	↕

SLP ////////////////////////////////////// Set CPU to SLEEP mode // 3 cycle ///

機 能 **SLEEP**

CPUをSLEEP状態にします。
SLEEP状態ではCPUと発振回路を含む周辺回路が動作を停止し、消費電力を大幅に低減できます。
SLEEP状態からはMCU外部の割り込みにより通常のプログラム実行状態に戻ります。

☞ "3.7.2 SLEEP状態"

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	1	0	1	1	1	1	AFH

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

SRA r ////////////////////////////////// Shift r reg. right arithmetic // 3 cycle ///

機 能 ▶ 7 6 5 4 3 2 1 0 ▶ C r

rレジスタ(A/B)の内容を1ビット右にシフトします。レジスタのビット0はキャリー(C)に移動し、レジスタのビット7は変化しません。
オーバーフロー(V)フラグは'0'にリセットされます。

モード **レジスタ直接**

例

設定値	結果				
r	r	SC			
		N	V	C	Z
01000100	00100010	0	0	0	0
10111001	11011100	1	0	1	0

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	0	0	1	0	0	r	88H、89H ※

※

r		ニーモニック	コード
A	0	SRA A	88H
B	1	SRA B	89H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	0	↕	↕

SRA [BR:ll] ////////////// Shift location [BR:ll] right arithmetic ////////////////////////////////// 5 cycle ///

機 能 7 6 5 4 3 2 1 0 → C [BR:*ll*]

BRレジスタの内容を上位バイト、8ビット絶対アドレス*ll*を下位バイトとしてアドレス指定されたデータメモリの内容を1ビット右にシフトします。データのビット0はキャリー(C)に移動し、データのビット7は変化しません。
オーバーフロー(V)フラグは'0'にリセットされます。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード 8ビット絶対

例

設定値	結果				
[BR:*ll*]	[BR:*ll*]	SC			
		N	V	C	Z
01000100	00100010	0	0	0	0
10111001	11011100	1	0	1	0

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	0	0	1	0	1	0	8AH
				l	*l*			*ll*

フラグ

I1	I0	U	D	N	V	C	Z
−	−	−	−	↕	0	↕	↕

SRA [HL] ////////////////////// Shift location [HL] right arithmetic ////////////////////////////////// 4 cycle ///

機 能 7 6 5 4 3 2 1 0 → C [HL]

HLレジスタで指定されたデータメモリの内容を1ビット右にシフトします。データのビット0はキャリー(C)に移動し、データのビット7は変化しません。
オーバーフロー(V)フラグは'0'にリセットされます。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード レジスタ間接

例

設定値	結果				
[HL]	[HL]	SC			
		N	V	C	Z
01000100	00100010	0	0	0	0
10111001	11011100	1	0	1	0

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	0	0	1	0	1	1	8BH

フラグ

I1	I0	U	D	N	V	C	Z
−	−	−	−	↕	0	↕	↕

SRL r ////////////////////////////// Shift r reg. right logical // 3 cycle ///

機 能 0→7 6 5 4 3 2 1 0→C r

rレジスタ(A/B)の内容を1ビット右にシフトします。レジスタのビット0はキャリー(C)に移動し、レジスタのビット7には'0'が入ります。

モード レジスタ直接

例

設定値	結果				
r	r	SC			
		N	V	C	Z
01000100	00100010	0	—	0	0
01101101	00110110	0	—	1	0

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	0	0	1	1	0	r	8CH、8DH ※

※

r		ニーモニック	コード
A	0	SRL A	8CH
B	1	SRL B	8DH

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	0	—	↕	↕

SRL [BR:ll] ////////////////// Shift location [BR:ll] right logical // 5 cycle ///

機 能 0→7 6 5 4 3 2 1 0→C [BR:*ll*]

BRレジスタの内容を上位バイト、8ビット絶対アドレス*ll*を下位バイトとしてアドレス指定されたデータメモリの内容を1ビット右にシフトします。データのビット0はキャリー(C)に移動し、データのビット7は'0'になります。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード 8ビット絶対

例

設定値	結果				
[BR:*ll*]	[BR:*ll*]	SC			
		N	V	C	Z
01000100	00100010	0	—	0	0
01101101	00110110	0	—	1	0

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	0	0	1	1	1	0	8EH
				l	*l*			*ll*

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	0	—	↕	↕

SRL [HL] ////////////////////// Shift location [HL] right logical // 4 cycle ///

機 能 0→7 6 5 4 3 2 1 0→C [HL]

HLレジスタで指定されたデータメモリの内容を1ビット右にシフトします。データのビット0はキャリー(C)に移動し、データのビット7は'0'になります。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード レジスタ間接

例

設定値	結果				
[HL]	[HL]	SC			
		N	V	C	Z
01000100	00100010	0	—	0	0
01101101	00110110	0	—	1	0

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	0	0	1	1	1	1	8FH

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	0	—	↕	↕

SUB A, r ////////////////////// Subtract r reg. from A reg. // 2 cycle ///

機 能 A ← A - r

rレジスタ(A/B)の内容をAレジスタから減算します。

モード Src: レジスタ直接
Dst: レジスタ直接

コード

MSB							LSB	
0	0	0	1	0	0	0	r	10H、11H ※

※

r		ニーモニック	コード
A	0	SUB A, A	10H
B	1	SUB A, B	11H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	★	★	↕	↕	↕	↕

例

設定値		結果				
A	B	A	SC			
			N	V	C	Z
D=0, U=0の場合						
A8H	42H	66H	0	1	0	0
36H	5AH	DCH	1	0	1	0
D=1, U=0の場合						
88	39	49	0	0	0	0
D=1, U=1の場合						
88	39	09	0	0	1	0

SUB A, #nn ////////////////// Subtract immediate data nn from A reg. ////////////////////////////////// 2 cycle ///

機 能 A ← A - nn

8ビット即値データnnをAレジスタから減算します。

モード Src: 即値データ
Dst: レジスタ直接

コード

MSB							LSB	
0	0	0	1	0	0	1	0	12H
			n	n				nn

フラグ

I1	I0	U	D	N	V	C	Z
—	—	★	★	↕	↕	↕	↕

例

設定値		結果				
A	nn	A	SC			
			N	V	C	Z
D=0, U=0の場合						
A8H	42H	66H	0	1	0	0
36H	5AH	DCH	1	0	1	0
D=1, U=0の場合						
88	39	49	0	0	0	0
D=1, U=1の場合						
88	39	09	0	0	1	0

SUB A, [BR:ll] //////// Subtract location [BR:ll] from A reg. ////////////////////////////////////// 3 cycle ///

機 能 A ← A - [BR:*ll*]

BRレジスタの内容を上位バイト、8ビット絶対アドレス*ll*を下位バイトとしてアドレス指定されるデータメモリの内容をAレジスタから減算します。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: 8ビット絶対
Dst: レジスタ直接

例

設定値		結果				
A	[BR:*ll*]	A	SC			
			N	V	C	Z
D=0, U=0の場合						
A8H	42H	66H	0	1	0	0
36H	5AH	DCH	1	0	1	0
D=1, U=0の場合						
88	39	49	0	0	0	0
D=1, U=1の場合						
88	39	09	0	0	1	0

コード

MSB							LSB	
0	0	0	1	0	1	0	0	14H
			l	*l*				*ll*

フラグ

I1	I0	U	D	N	V	C	Z
–	–	★	★	↕	↕	↕	↕

SUB A, [hhll] ////////////// Subtract location [hhll] from A reg. ////////////////////////////////////// 4 cycle ///

機 能 A ← A - [hh*ll*]

16ビット絶対アドレスhh*ll*でアドレス指定されたデータメモリの内容をAレジスタから減算します。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: 16ビット絶対
Dst: レジスタ直接

例

設定値		結果				
A	[hh*ll*]	A	SC			
			N	V	C	Z
D=0, U=0の場合						
A8H	42H	66H	0	1	0	0
36H	5AH	DCH	1	0	1	0
D=1, U=0の場合						
88	39	49	0	0	0	0
D=1, U=1の場合						
88	39	09	0	0	1	0

コード

MSB							LSB	
0	0	0	1	0	1	0	1	15H
			l	*l*				*ll*
			h	h				hh

フラグ

I1	I0	U	D	N	V	C	Z
–	–	★	★	↕	↕	↕	↕

SUB A, [HL] /////////////// Subtract location [HL] from A reg. ////////////////////////////////////// 2 cycle ///

機 能 A ← A - [HL]

HLレジスタでアドレス指定されたデータメモリの内容をAレジスタから減算します。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: レジスタ間接
Dst: レジスタ直接

例

設定値		結果				
A	[HL]	A	SC			
			N	V	C	Z
D=0, U=0の場合						
A8H	42H	66H	0	1	0	0
36H	5AH	DCH	1	0	1	0
D=1, U=0の場合						
88	39	49	0	0	0	0
D=1, U=1の場合						
88	39	09	0	0	1	0

コード

MSB							LSB	
0	0	0	1	0	0	1	1	13H

フラグ

I1	I0	U	D	N	V	C	Z
–	–	★	★	↕	↕	↕	↕

SUB A, [ir] //////////////////// Subtract location [ir reg.] from A reg. ////////////////////////////////////// 2 cycle ///

機 能 A ← A - [ir]

irレジスタ(IX/IY)でアドレス指定されたデータメモリの内容をAレジスタから減算します。
XPレジスタ(IX指定時)、YPレジスタ(IY指定時)の内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: レジスタ間接
Dst: レジスタ直接

例

設定値		結果				
A	[ir]	A	SC			
			N	V	C	Z
D=0, U=0の場合						
A8H	42H	66H	0	1	0	0
36H	5AH	DCH	1	0	1	0
D=1, U=0の場合						
88	39	49	0	0	0	0
D=1, U=1の場合						
88	39	09	0	0	1	0

コード

MSB							LSB	
0	0	0	1	0	1	1	ir	16H、17H ※

※

ir		ニーモニック	コード
IX	0	SUB A, [IX]	16H
IY	1	SUB A, [IY]	17H

フラグ

I1	I0	U	D	N	V	C	Z
−	−	★	★	↕	↕	↕	↕

SUB A, [ir+dd] /////// Subtract location [ir reg. + dd] from A reg. ////////////////////////// 4 cycle ///

機 能 A ← A - [ir+dd]

irレジスタ(IX/IY)の内容とディスプレースメントddの和でアドレス指定されたデータメモリの内容をAレジスタから減算します。
ddは符号付きデータとして扱われ、範囲は-128～127です。
XPレジスタ(IX指定時)、YPレジスタ(IY指定時)の内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: ディスプレースメント付きレジスタ間接
Dst: レジスタ直接

例

設定値		結果				
A	[ir+dd]	A	SC			
			N	V	C	Z
D=0, U=0の場合						
A8H	42H	66H	0	1	0	0
36H	5AH	DCH	1	0	1	0
D=1, U=0の場合						
88	39	49	0	0	0	0
D=1, U=1の場合						
88	39	09	0	0	1	0

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	0	1	0	0	0	ir	10H、11H ※
d	d	d	d	d	d	d	d	dd

※

ir		ニーモニック	コード
IX	0	SUB A, [IX+dd]	10H
IY	1	SUB A, [IY+dd]	11H

フラグ

I1	I0	U	D	N	V	C	Z
−	−	★	★	↕	↕	↕	↕

SUB A, [ir+L] *|||||||||||| Subtract location [ir reg. + L] from A reg. |||||||||||||||||||||||||| 4 cycle ///*

機 能 **A ← A - [ir+L]**

irレジスタ(IX/IY)の内容とLレジスタの内容の和でアドレス指定されたデータメモリの内容をAレジスタから減算します。
Lレジスタの内容は符号付きデータとして扱われ、範囲は-128～127です。
XPレジスタ(IX指定時)、YPレジスタ(IY指定時)の内容がデータメモリのページアドレスになります(MODEL2/3)。

モード **Src: インデックスレジスタ付きレジスタ間接**
Dst: レジスタ直接

例

設定値		結果				
A	[ir+L]	A	SC			
			N	V	C	Z
D=0, U=0の場合						
A8H	42H	66H	0	1	0	0
36H	5AH	DCH	1	0	1	0
D=1, U=0の場合						
88	39	49	0	0	0	0
D=1, U=1の場合						
88	39	09	0	0	1	0

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	0	1	0	0	1	ir	12H、13H ※

※

ir		ニーモニック	コード
IX	0	SUB A, [IX+L]	12H
IY	1	SUB A, [IY+L]	13H

フラグ

I1	I0	U	D	N	V	C	Z
−	−	★	★	↕	↕	↕	↕

SUB [HL], A *|||||||||||||| Subtract A reg. from location [HL] |||||||||||||||||||||||||||||||||||||||| 4 cycle ///*

機 能 **[HL] ← [HL] - A**

Aレジスタの内容をHLレジスタでアドレス指定されたデータメモリから減算します。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード **Src: レジスタ直接**
Dst: レジスタ間接

例

設定値		結果				
[HL]	A	[HL]	SC			
			N	V	C	Z
D=0, U=0の場合						
A8H	42H	66H	0	1	0	0
36H	5AH	DCH	1	0	1	0
D=1, U=0の場合						
88	39	49	0	0	0	0
D=1, U=1の場合						
88	39	09	0	0	1	0

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	0	1	0	1	0	0	14H

フラグ

I1	I0	U	D	N	V	C	Z
−	−	★	★	↕	↕	↕	↕

SUB [HL], #nn *|||||||| Subtract immediate data nn from location [HL] |||||||||||||||||||| 5 cycle ///*

機 能 **[HL] ← [HL] - nn**

8ビット即値データnnをHLレジスタでアドレス指定されたデータメモリから減算します。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード **Src: 即値データ**
Dst: レジスタ間接

例

設定値		結果				
[HL]	nn	[HL]	SC			
			N	V	C	Z
D=0, U=0の場合						
A8H	42H	66H	0	1	0	0
36H	5AH	DCH	1	0	1	0
D=1, U=0の場合						
88	39	49	0	0	0	0
D=1, U=1の場合						
88	39	09	0	0	1	0

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	0	1	0	1	0	1	15H
			n	n				nn

フラグ

I1	I0	U	D	N	V	C	Z
−	−	★	★	↕	↕	↕	↕

SUB [HL], [ir] ////////// Subtract location [ir reg.] from location [HL] ////////////////////// 5 cycle ///

機 能 [HL] ← [HL] - [ir]

irレジスタ(IX/IY)でアドレス指定されたデータメモリの内容をHLレジスタでアドレス指定されるデータメモリから減算します。
EPレジスタの内容がデータメモリ[HL]のページアドレス、XPレジスタ(IX指定時)、YPレジスタ(IY指定時)の内容がデータメモリ[ir]のページアドレスになります(MODEL2/3)。

モード Src: レジスタ間接
Dst: レジスタ間接

例

設定値		結果				
[HL]	[ir]	[HL]	SC			
			N	V	C	Z
D=0, U=0の場合						
A8H	42H	66H	0	1	0	0
36H	5AH	DCH	1	0	1	0
D=1, U=0の場合						
88	39	49	0	0	0	0
D=1, U=1の場合						
88	39	09	0	0	1	0

コード MSB LSB

1	1	0	0	1	1	1	0	CEH

0	0	0	1	0	1	1	ir	16H、17H ※

※

ir		ニーモニック	コード
IX	0	SUB [HL], [IX]	16H
IY	1	SUB [HL], [IY]	17H

フラグ

I1	I0	U	D	N	V	C	Z
−	−	★	★	↕	↕	↕	↕

SUB BA, rp /////////////// Subtract rp reg. from BA reg. // 4 cycle ///

機 能 BA ← BA - rp

rpレジスタ(BA/HL/IX/IY)の内容をBAレジスタから減算します。

モード Src: レジスタ直接
Dst: レジスタ直接

例

設定値		結果				
BA	rp	BA	SC			
			N	V	C	Z
63C2H	2125H	429DH	0	0	0	0
C261H	5A32H	682FH	0	1	0	0
205CH	7120H	AF3CH	1	0	1	0

(rp≠BA)

コード MSB LSB

1	1	0	0	1	1	1	1	CFH

0	0	0	0	1	0	rp	08H〜0BH ※

※

rp		ニーモニック	コード
BA	00	SUB BA, BA	08H
HL	01	SUB BA, HL	09H
IX	10	SUB BA, IX	0AH
IY	11	SUB BA, IY	0BH

フラグ

I1	I0	U	D	N	V	C	Z
−	−	−	−	↕	↕	↕	↕

SUB BA, #mmnn /// Subtract immediate data mmnn from BA reg. ////////////////////// 3 cycle ///

機 能 BA ← BA - mmnn

16ビット即値データmmnnをBAレジスタから減算します。

モード Src: 即値データ
Dst: レジスタ直接

例

設定値		結果				
BA	mmnn	BA	SC			
			N	V	C	Z
63C2H	2125H	429DH	0	0	0	0
C261H	5A32H	682FH	0	1	0	0
205CH	7120H	AF3CH	1	0	1	0

コード MSB LSB

1	1	0	1	0	0	0	0	D0H

n	n	nn

m	m	mm

フラグ

I1	I0	U	D	N	V	C	Z
−	−	−	−	↕	↕	↕	↕

SUB HL, rp ////////////////// Subtract rp reg. from HL reg. // 4 cycle ///

機 能 HL ← HL - rp

rpレジスタ(BA/HL/IX/IY)の内容をHLレジスタから減算します。

モード Src: レジスタ直接
Dst: レジスタ直接

例

設定値		結果				
HL	rp	HL	SC			
			N	V	C	Z
63C2H	2125H	429DH	0	0	0	0
C261H	5A32H	682FH	0	1	0	0
205CH	7120H	AF3CH	1	0	1	0

(rp≠HL)

コード

MSB LSB

1	1	0	0	1	1	1	1	CFH

0	0	1	0	1	0	rp		28H～2BH ※

※

rp		ニーモニック	コード
BA	00	SUB HL, BA	28H
HL	01	SUB HL, HL	29H
IX	10	SUB HL, IX	2AH
IY	11	SUB HL, IY	2BH

フラグ

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	↕	↕	↕

SUB HL, #mmnn /// Subtract immediate data mmnn from HL reg. ////////////////////////// 3 cycle ///

機 能 HL ← HL - mmnn

16ビット即値データmmnnをHLレジスタから減算します。

モード Src: 即値データ
Dst: レジスタ直接

例

設定値		結果				
HL	mmnn	HL	SC			
			N	V	C	Z
63C2H	2125H	429DH	0	0	0	0
C261H	5A32H	682FH	0	1	0	0
205CH	7120H	AF3CH	1	0	1	0

コード

MSB LSB

1	1	0	1	0	0	0	1	D1H

n	n	nn

m	m	mm

フラグ

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	↕	↕	↕

SUB IX, rp ////////////////// Subtract rp reg. from IX reg. // 4 cycle ///

機 能 IX ← IX - rp

rpレジスタ(BA/HL)の内容をIXレジスタから減算します。

モード Src: レジスタ直接
Dst: レジスタ直接

例

設定値		結果				
IX	rp	IX	SC			
			N	V	C	Z
63C2H	2125H	429DH	0	0	0	0
C261H	5A32H	682FH	0	1	0	0
205CH	7120H	AF3CH	1	0	1	0

コード

MSB LSB

1	1	0	0	1	1	1	1	CFH

0	1	0	0	1	0	0	rp	48H、49H ※

※

rp		ニーモニック	コード
BA	0	SUB IX, BA	48H
HL	1	SUB IX, HL	49H

フラグ

I1	I0	U	D	N	V	C	Z
–	–	–	–	↕	↕	↕	↕

SUB IX, #mmnn ///// Subtract immediate data mmnn from IX reg. /////////////////////////// 3 cycle ///

機 能 IX ← IX - mmnn

16ビット即値データmmnnをIXレジスタから減算します。

モード Src: 即値データ
Dst: レジスタ直接

コード

MSB							LSB	
1	1	0	1	0	0	1	0	D2H
n	n							nn
m	m							mm

フラグ

I1	I0	U	D	N	V	C	Z
−	−	−	−	↕	↕	↕	↕

例

設定値		結果				
IX	mmnn	IX	SC			
			N	V	C	Z
63C2H	2125H	429DH	0	0	0	0
C261H	5A32H	682FH	0	1	0	0
205CH	7120H	AF3CH	1	0	1	0

SUB IY, rp ////////////////// Subtract rp reg. from IY reg. // 4 cycle ///

機 能 IY ← IY - rp

rpレジスタ(BA/HL)の内容をIYレジスタから減算します。

モード Src: レジスタ直接
Dst: レジスタ直接

コード

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
0	1	0	0	1	0	1	rp	4AH、4BH ※

※

rp		ニーモニック	コード
BA	0	SUB IY, BA	4AH
HL	1	SUB IY, HL	4BH

フラグ

I1	I0	U	D	N	V	C	Z
−	−	−	−	↕	↕	↕	↕

例

設定値		結果				
IY	rp	IY	SC			
			N	V	C	Z
63C2H	2125H	429DH	0	0	0	0
C261H	5A32H	682FH	0	1	0	0
205CH	7120H	AF3CH	1	0	1	0

SUB IY, #mmnn ////// Subtract immediate data mmnn from IY reg. /////////////////////////// 3 cycle ///

機 能 IY ← IY + mmnn

16ビット即値データmmnnをIYレジスタから減算します。

モード Src: 即値データ
Dst: レジスタ直接

コード

MSB							LSB	
1	1	0	1	0	0	1	1	D3H
n	n							nn
m	m							mm

フラグ

I1	I0	U	D	N	V	C	Z
−	−	−	−	↕	↕	↕	↕

例

設定値		結果				
IY	mmnn	IY	SC			
			N	V	C	Z
63C2H	2125H	429DH	0	0	0	0
C261H	5A32H	682FH	0	1	0	0
205CH	7120H	AF3CH	1	0	1	0

SUB SP, rp *////////////////// Subtract rp reg. from SP // 4 cycle ///*

機 能 SP ← SP - rp

rpレジスタ(BA/HL)の内容をスタックポインタ(SP)から減算します。

モード Src: レジスタ直接
Dst: レジスタ直接

コード

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
0	1	0	0	1	1	0	rp	4CH、4DH ※

※

rp		ニーモニック	コード
BA	0	SUB SP, BA	4CH
HL	1	SUB SP, HL	4DH

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	↕	↕	↕

例

設定値		結果				
SP	rp	SP	SC			
			N	V	C	Z
63C2H	2125H	429DH	0	0	0	0
C261H	5A32H	682FH	0	1	0	0
205CH	7120H	AF3CH	1	0	1	0

SUB SP, #mmnn *//// Subtract immediate data mmnn from SP ////////////////////////////////// 4 cycle ///*

機 能 SP ← SP - mmnn

16ビット即値データmmnnをスタックポインタ(SP)から減算します。

モード Src: 即値データ
Dst: レジスタ直接

コード

MSB							LSB	
1	1	0	0	1	1	1	1	CFH
0	1	1	0	1	0	1	0	6AH
				n	n			nn
				m	m			mm

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	↕	↕	↕

例

設定値		結果				
SP	mmnn	SP	SC			
			N	V	C	Z
63C2H	2125H	429DH	0	0	0	0
C261H	5A32H	682FH	0	1	0	0
205CH	7120H	AF3CH	1	0	1	0

SWAP A *////////////////////////// Swap high-order and low-order of A reg. ////////////////////////////////// 2 cycle ///*

機 能 A(H) ↔ A(L)

Aレジスタの上位4ビットと下位4ビットの内容を入れ替えます。

モード レジスタ直接

コード

MSB							LSB	
1	1	1	1	0	1	1	0	F6H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

例

設定値	結果				
A	A	SC			
		N	V	C	Z
4CH	C4H	—	—	—	—
62H	26H	—	—	—	—

SWAP [HL] //////////////// Swap high-order and low-order of location [HL] ////////////////// 3 cycle ///

機 能 [HL](H) ↔ [HL](L)

HLレジスタで指定されたデータメモリの上位4ビットと下位4ビットの内容を入れ替えます。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード レジスタ間接

例

設定値	結果				
[HL]	[HL]	SC			
		N	V	C	Z
4CH	C4H	—	—	—	—
62H	26H	—	—	—	—

コード

MSB							LSB	
1	1	1	1	0	1	1	1	F7H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

UPCK ////////////////////////////// Unpack A reg. to BA reg. // 2 cycle ///

機 能

A		B	A
m	n	0	m	0	n

Aレジスタの内容をアンパックし、BAレジスタにストアします。Bレジスタの下位4ビットがAレジスタの上位4ビットの内容に置き換えられ、Aレジスタ、Bレジスタとも上位4ビットは'0'になります。

モード インプライド(レジスタ直接)

例

設定値	結果				
A	BA	SC			
		N	V	C	Z
84H	0804H	—	—	—	—

コード

MSB							LSB	
1	1	0	1	1	1	1	1	DFH

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	—	—	—	—

XOR A, r ////////////////////// Exclusive OR r reg. and A reg. // 2 cycle ///

機 能 A ← A ∀ r

rレジスタ(A/B)の内容とAレジスタの内容との排他的論理和をとり、結果をAレジスタにストアします。

モード Src: レジスタ直接
Dst: レジスタ直接

例

設定値		結果				
A	B	A	SC			
			N	V	C	Z
2CH	41H	6DH	0	—	—	0
7AH	B6H	CCH	1	—	—	0

コード

MSB							LSB	
0	0	1	1	1	0	0	r	38H、39H ※

※

r		ニーモニック	コード
A	0	XOR A, A	38H
B	1	XOR A, B	39H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	—	↕

XOR A, #nn /////////////// Exclusive OR immediate data nn and A reg. ////////////////////////// 2 cycle ///

機 能 A ← A ∀ nn

8ビット即値データnnとAレジスタの内容との排他的論理和をとり、結果をAレジスタにストアします。

コード

MSB							LSB	
0	0	1	1	1	0	1	0	3AH
				n	n			nn

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	—	↕

モード Src: 即値データ
Dst: レジスタ直接

例

設定値		結果				
A	nn	A	SC			
			N	V	C	Z
2CH	41H	6DH	0	—	—	0
7AH	B6H	CCH	1	—	—	0

XOR A, [BR:ll] /////// Exclusive OR location [BR:ll] and A reg. ////////////////////////////// 3 cycle ///

機 能 A ← A ∀ [BR:*ll*]

BRレジスタの内容を上位バイト、8ビット絶対アドレス*ll*を下位バイトとしてアドレス指定されるデータメモリの内容とAレジスタの内容との排他的論理和をとり、結果をAレジスタにストアします。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

コード

MSB							LSB	
0	0	1	1	1	1	0	0	3CH
				l	*l*			*ll*

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	—	↕

モード Src: 8ビット絶対
Dst: レジスタ直接

例

設定値		結果				
A	[BR:*ll*]	A	SC			
			N	V	C	Z
2CH	41H	6DH	0	—	—	0
7AH	B6H	CCH	1	—	—	0

XOR A, [hhll] /////////// Exclusive OR location [hhll] and A reg. ////////////////////////////// 4 cycle ///

機 能 A ← A ∀ [hh*ll*]

16ビット絶対アドレスhh*ll*でアドレス指定されたデータメモリの内容とAレジスタの内容との排他的論理和をとり、結果をAレジスタにストアします。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

コード

MSB							LSB	
0	0	1	1	1	1	0	1	3DH
				l	*l*			*ll*
				h	h			hh

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	—	↕

モード Src: 16ビット絶対
Dst: レジスタ直接

例

設定値		結果				
A	[hh*ll*]	A	SC			
			N	V	C	Z
2CH	41H	6DH	0	—	—	0
7AH	B6H	CCH	1	—	—	0

XOR A, [HL] /////////////// Exclusive OR location [HL] and A reg. ////////////////////////////////// 2 cycle ///

機 能 A ← A ∀ [HL]

HLレジスタでアドレス指定されたデータメモリの内容とAレジスタの内容との排他的論理和をとり、結果をAレジスタにストアします。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: レジスタ間接
Dst: レジスタ直接

例

設定値		結果				
A	[HL]	A	SC			
			N	V	C	Z
2CH	41H	6DH	0	—	—	0
7AH	B6H	CCH	1	—	—	0

コード

MSB							LSB	
0	0	1	1	1	0	1	1	3BH

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	—	↕

XOR A, [ir] ///////////////// Exclusive OR location [ir reg.] and A reg. ////////////////////////////// 2 cycle ///

機 能 A ← A ∀ [ir]

irレジスタ(IX/IY)でアドレス指定されたデータメモリの内容とAレジスタの内容との排他的論理和をとり、結果をAレジスタにストアします。
XPレジスタ(IX指定時)、YPレジスタ(IY指定時)の内容がページアドレスになります(MODEL2/3)。

モード Src: レジスタ間接
Dst: レジスタ直接

例

設定値		結果				
A	[ir]	A	SC			
			N	V	C	Z
2CH	41H	6DH	0	—	—	0
7AH	B6H	CCH	1	—	—	0

コード

MSB							LSB	
0	0	1	1	1	1	1	ir	3EH、3FH ※

※

ir		ニーモニック	コード
IX	0	XOR A, [IX]	3EH
IY	1	XOR A, [IY]	3FH

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	—	↕

XOR A, [ir+dd] /////// Exclusive OR location [ir reg. + dd] and A reg. ////////////////////// 4 cycle ///

機 能 A ← A ∀ [ir+dd]

irレジスタ(IX/IY)の内容とディスプレースメントddの和でアドレス指定されたデータメモリの内容とAレジスタの内容との排他的論理和をとり、結果をAレジスタにストアします。ddは符号付きデータとして扱われ、範囲は-128～127です。
XPレジスタ(IX指定時)、YPレジスタ(IY指定時)の内容がページアドレスになります(MODEL2/3)。

モード Src: ディスプレースメント付きレジスタ間接
Dst: レジスタ直接

例

設定値		結果				
A	[ir+dd]	A	SC			
			N	V	C	Z
2CH	41H	6DH	0	—	—	0
7AH	B6H	CCH	1	—	—	0

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	1	1	1	0	0	ir	38H、39H ※
d	d							dd

※

ir		ニーモニック	コード
IX	0	XOR A, [IX+dd]	38H
IY	1	XOR A, [IY+dd]	39H

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	—	↕

XOR A, [ir+L] ////////// Exclusive OR location [ir reg. + L] and A reg. ////////////////////// 4 cycle ///

機 能 A ← A ∀ [ir+L]

irレジスタ(IX/IY)の内容とLレジスタの内容の和でアドレス指定されたデータメモリの内容とAレジスタの内容との排他的論理和をとり、結果をAレジスタにストアします。
Lレジスタの内容は符号付きデータとして扱われ、範囲は-128～127です。
XPレジスタ(IX指定時)、YPレジスタ(IY指定時)の内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: インデックスレジスタ付きレジスタ間接
Dst: レジスタ直接

例

設定値		結果				
A	[ir+L]	A	SC			
			N	V	C	Z
2CH	41H	6DH	0	—	—	0
7AH	B6H	CCH	1	—	—	0

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	1	1	1	0	1	ir	3AH、3BH ※

※

ir		ニーモニック	コード
IX	0	XOR A, [IX+L]	3AH
IY	1	XOR A, [IY+L]	3BH

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	—	↕

XOR B, #nn ////////////// Exclusive OR immediate data nn and B reg. ////////////////////////// 3 cycle ///

機 能 B ← B ∀ nn

8ビット即値データnnとBレジスタの内容との排他的論理和をとり、結果をBレジスタにストアします。

モード Src: 即値データ
Dst: レジスタ直接

例

設定値		結果				
B	nn	B	SC			
			N	V	C	Z
2CH	41H	6DH	0	—	—	0
7AH	B6H	CCH	1	—	—	0

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	1	1	1	0	0	0	B8H
n	n							nn

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	—	↕

XOR L, #nn ////////////// Exclusive OR immediate data nn and L reg. ////////////////////////// 3 cycle ///

機 能 L ← L ∀ nn

8ビット即値データnnとLレジスタの内容との排他的論理和をとり、結果をLレジスタにストアします。

モード Src: 即値データ
Dst: レジスタ直接

例

設定値		結果				
L	nn	L	SC			
			N	V	C	Z
2CH	41H	6DH	0	—	—	0
7AH	B6H	CCH	1	—	—	0

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	1	1	1	0	0	1	B9H
n	n							nn

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	—	↕

XOR H, #nn /////////////// *Exclusive OR immediate data nn and H reg.* ////////////////////////// 3 cycle ///

機 能 H ← H ∀ nn

8ビット即値データnnとHレジスタの内容との排他的論理和をとり、結果をHレジスタにストアします。

モード Src: 即値データ
Dst: レジスタ直接

例

設定値		結果				
H	nn	H	SC			
			N	V	C	Z
2CH	41H	6DH	0	—	—	0
7AH	B6H	CCH	1	—	—	0

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
1	0	1	1	1	0	1	0	BAH
			n	n				nn

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	—	↕

XOR SC, #nn ////////// *Exclusive OR immediate data nn and SC* ////////////////////////////////// 3 cycle ///

機 能 SC ← SC ∀ nn

8ビット即値データnnとシステムコンディションフラグ(SC)の内容との排他的論理和をとり、結果をシステムコンディションフラグ(SC)にセットします。

モード Src: 即値データ
Dst: レジスタ直接

例

設定値		結果							
SC	nn	SC	SC						
			I1	I0	U	D	N	V	C	Z
2CH	41H	6DH	0	1	1	0	1	1	0	1
7AH	B6H	CCH	1	1	0	0	1	1	0	0

コード

MSB							LSB	
1	0	0	1	1	1	1	0	9EH
			n	n				nn

フラグ

I1	I0	U	D	N	V	C	Z
↕	↕	↕	↕	↕	↕	↕	↕

XOR [BR:ll], #nn // *Exclusive OR immediate data nn and location [BR:ll]* ////////// 5 cycle ///

機 能 [BR:*ll*] ← [BR:*ll*] ∀ nn

8ビット即値データnnとBRレジスタの内容を上位バイト、8ビット絶対アドレス*ll*を下位バイトとしてアドレス指定されるデータメモリの内容との排他的論理和をとり、結果をそのアドレスにストアします。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: 即値データ
Dst: 8ビット絶対

例

設定値		結果				
[BR:*ll*]	nn	[BR:*ll*]	SC			
			N	V	C	Z
2CH	41H	6DH	0	—	—	0
7AH	B6H	CCH	1	—	—	0

コード

MSB							LSB	
1	1	0	1	1	0	1	0	DAH
			l	*l*				*ll*
			n	n				nn

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	—	↕

XOR [HL], A /////////////// Exclusive OR A reg. and location [HL] ////////////////////////////////// 4 cycle ///

機 能 [HL] ← [HL] ∀ A

Aレジスタの内容とHLレジスタでアドレス指定されたデータメモリの内容との排他的論理和をとり、結果をそのアドレスにストアします。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: レジスタ直接
Dst: レジスタ間接

例

設定値		結果				
[HL]	A	[HL]	SC			
			N	V	C	Z
2CH	41H	6DH	0	—	—	0
7AH	B6H	CCH	1	—	—	0

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	1	1	1	1	0	0	3CH

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	—	↕

XOR [HL], #nn /////// Exclusive OR immediate data nn and location [HL] ////////////// 5 cycle ///

機 能 [HL] ← [HL] ∀ nn

8ビット即値データnnとHLレジスタでアドレス指定されたデータメモリの内容との排他的論理和をとり、結果をそのアドレスにストアします。
EPレジスタの内容がデータメモリのページアドレスになります(MODEL2/3)。

モード Src: 即値データ
Dst: レジスタ間接

例

設定値		結果				
[HL]	nn	[HL]	SC			
			N	V	C	Z
2CH	41H	6DH	0	—	—	0
7AH	B6H	CCH	1	—	—	0

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	1	1	1	1	0	1	3DH
n	n							nn

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	—	↕

XOR [HL], [ir] ///////// Exclusive OR location [ir reg.] and location [HL] ////////////////// 5 cycle ///

機 能 [HL] ← [HL] ∀ [ir]

irレジスタ(IX/IY)とHLレジスタでそれぞれアドレス指定されたデータメモリの内容の排他的論理和をとり、結果をデータメモリ[HL]にストアします。
EPレジスタの内容がデータメモリ[HL]のページアドレス、XPレジスタ(IX指定時)、YPレジスタ(IY指定時)の内容がデータメモリ[ir]のページアドレスになります(MODEL2/3)。

モード Src: レジスタ間接
Dst: レジスタ間接

例

設定値		結果				
[HL]	[ir]	[HL]	SC			
			N	V	C	Z
2CH	41H	6DH	0	—	—	0
7AH	B6H	CCH	1	—	—	0

コード

MSB							LSB	
1	1	0	0	1	1	1	0	CEH
0	0	1	1	1	1	1	ir	3EH、3FH ※

※

ir		ニーモニック	コード
IX	0	XOR [HL], [IX]	3EH
IY	1	XOR [HL], [IY]	3FH

フラグ

I1	I0	U	D	N	V	C	Z
—	—	—	—	↕	—	—	↕

APPENDIX A オペコードマップ

第1オペコード

L＼H	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	ADD A,A	SUB A,A	AND A,A	CP A,A	LD A,A	LD L,A	LD [IX],A	LD [IY],A	INC A	INC BA	PUSH BA	LD A,#nn	ADD BA,#mmnn	SUB BA,#mmnn	CARS C,rr	CARS rr
1	ADD A,B	SUB A,B	AND A,B	CP A,B	LD A,B	LD L,B	LD [IX],B	LD [IY],B	INC B	INC HL	PUSH HL	LD B,#nn	ADD HL,#mmnn	SUB HL,#mmnn	CARS NC,rr	JRS rr
2	ADD A,#nn	SUB A,#nn	AND A,#nn	CP A,#nn	LD A,L	LD L,L	LD [IX],L	LD [IY],L	INC L	INC IX	PUSH IX	LD L,#nn	ADD IX,#mmnn	SUB IX,#mmnn	CARS Z,rr	CARL qqrr
3	ADD A,[HL]	SUB A,[HL]	AND A,[HL]	CP A,[HL]	LD A,H	LD L,H	LD [IX],H	LD [IY],H	INC H	INC IY	PUSH IY	LD H,#nn	ADD IY,#mmnn	SUB IY,#mmnn	CARS NZ,rr	JRL qqrr
4	ADD A,[BR:*ll*]	SUB A,[BR:*ll*]	AND A,[BR:*ll*]	CP A,[BR:*ll*]	LD A,[BR:*ll*]	LD L,[BR:*ll*]	LD [IX],[BR:*ll*]	LD [IY],[BR:*ll*]	INC BR	BIT A,B	PUSH BR	LD BR,#hh	LD BA,#mmnn	CP BA,#mmnn	JRS C,rr	JP HL
5	ADD A,[hh*ll*]	SUB A,[hh*ll*]	AND A,[hh*ll*]	CP A,[hh*ll*]	LD A,[HL]	LD L,[HL]	LD [IX],[HL]	LD [IY],[HL]	INC [BR:*ll*]	BIT [HL],#nn	PUSH EP	LD [HL],#nn	LD HL,#mmnn	CP HL,#mmnn	JRS NC,rr	DJR NZ,rr
6	ADD A,[IX]	SUB A,[IX]	AND A,[IX]	CP A,[IX]	LD A,[IX]	LD L,[IX]	LD [IX],[IX]	LD [IY],[IX]	INC [HL]	BIT A,#nn	PUSH IP	LD [IX],#nn	LD IX,#mmnn	CP IX,#mmnn	JRS Z,rr	SWAP A
7	ADD A,[IY]	SUB A,[IY]	AND A,[IY]	CP A,[IY]	LD A,[IY]	LD L,[IY]	LD [IX],[IY]	LD [IY],[IY]	INC SP	BIT B,#nn	PUSH SC	LD [IY],#nn	LD IY,#mmnn	CP IY,#mmnn	JRS NZ,rr	SWAP [HL]
8	ADC A,A	SBC A,A	OR A,A	XOR A,A	LD B,A	LD H,A	LD [HL],A	LD [BR:*ll*],A	DEC A	DEC BA	POP BA	LD BA,[hh*ll*]	EX BA,HL	AND [BR:*ll*],#nn	CARL C,qqrr	RET
9	ADC A,B	SBC A,B	OR A,B	XOR A,B	LD B,B	LD H,B	LD [HL],B	LD [BR:*ll*],B	DEC B	DEC HL	POP HL	LD HL,[hh*ll*]	EX BA,IX	OR [BR:*ll*],#nn	CARL NC,qqrr	RETE
A	ADC A,#nn	SBC A,#nn	OR A,#nn	XOR A,#nn	LD B,L	LD H,L	LD [HL],L	LD [BR:*ll*],L	DEC L	DEC IX	POP IX	LD IX,[hh*ll*]	EX BA,IY	XOR [BR:*ll*],#nn	CARL Z,qqrr	RETS
B	ADC A,[HL]	SBC A,[HL]	OR A,[HL]	XOR A,[HL]	LD B,H	LD H,H	LD [HL],H	LD [BR:*ll*],H	DEC H	DEC IY	POP IY	LD IY,[hh*ll*]	EX BA,SP	CP [BR:*ll*],#nn	CARL NZ,qqrr	CALL [hh*ll*]
C	ADC A,[BR:*ll*]	SBC A,[BR:*ll*]	OR A,[BR:*ll*]	XOR A,[BR:*ll*]	LD B,[BR:*ll*]	LD H,[BR:*ll*]	LD [HL],[BR:*ll*]	**未定義 コード**	DEC BR	AND SC,#nn	POP BR	LD [hh*ll*],BA	EX A,B	BIT [BR:*ll*],#nn	JRL C,qqrr	INT [kk]
D	ADC A,[hh*ll*]	SBC A,[hh*ll*]	OR A,[hh*ll*]	XOR A,[hh*ll*]	LD B,[HL]	LD H,[HL]	LD [HL],[HL]	LD [BR:*ll*],[HL]	DEC [BR:*ll*]	OR SC,#nn	POP EP	LD [hh*ll*],HL	EX A,[HL]	LD [BR:*ll*],#nn	JRL NC,qqrr	JP [kk]
E	ADC A,[IX]	SBC A,[IX]	OR A,[IX]	XOR A,[IX]	LD B,[IX]	LD H,[IX]	LD [HL],[IX]	LD [BR:*ll*],[IX]	DEC [HL]	XOR SC,#nn	POP IP	LD [hh*ll*],IX	**拡張 コード**	PACK	JRL Z,qqrr	**未定義 コード**
F	ADC A,[IY]	SBC A,[IY]	OR A,[IY]	XOR A,[IY]	LD B,[IY]	LD H,[IY]	LD [HL],[IY]	LD [BR:*ll*],[IY]	DEC SP	LD SC,#nn	POP SC	LD [hh*ll*],IY	**拡張 コード**	UPCK	JRL NZ,qqrr	NOP

オペコードマップ(2/3)

第2オペコード(第1オペコード = CE)

L \ H	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	ADD A,[IX+dd]	SUB A,[IX+dd]	AND A,[IX+dd]	CP A,[IX+dd]	LD A,[IX+dd]	LD L,[IX+dd]	LD [HL],[IX+dd]		SLA A	RL A	CPL A	AND B,#nn	LD A,BR	LD A,[hh*ll*]	JRS LT,rr	CARS LT,rr
1	ADD A,[IY+dd]	SUB A,[IY+dd]	AND A,[IY+dd]	CP A,[IY+dd]	LD A,[IY+dd]	LD L,[IY+dd]	LD [HL],[IY+dd]		SLA B	RL B	CPL B	AND L,#nn	LD A,SC	LD B,[hh*ll*]	JRS LE,rr	CARS LE,rr
2	ADD A,[IX+L]	SUB A,[IX+L]	AND A,[IX+L]	CP A,[IX+L]	LD A,[IX+L]	LD L,[IX+L]	LD [HL],[IX+L]		SLA [BR:*ll*]	RL [BR:*ll*]	CPL [BR:*ll*]	AND H,#nn	LD BR,A	LD L,[hh*ll*]	JRS GT,rr	CARS GT,rr
3	ADD A,[IY+L]	SUB A,[IY+L]	AND A,[IY+L]	CP A,[IY+L]	LD A,[IY+L]	LD L,[IY+L]	LD [HL],[IY+L]		SLA [HL]	RL [HL]	CPL [HL]	**未定義コード**	LD SC,A	LD H,[hh*ll*]	JRS GE,rr	CARS GE,rr
4	ADD [HL],A	SUB [HL],A	AND [HL],A	CP [HL],A	LD [IX+dd],A	LD [IX+dd],L			SLL A	RLC A	NEG A	OR B,#nn	LD NB,#bb	LD [hh*ll*],A	JRS V,rr	CARS V,rr
5	ADD [HL],#nn	SUB [HL],#nn	AND [HL],#nn	CP [HL],#nn	LD [IY+dd],A	LD [IY+dd],L			SLL B	RLC B	NEG B	OR L,#nn	LD EP,#pp	LD [hh*ll*],B	JRS NV,rr	CARS NV,rr
6	ADD [HL],[IX]	SUB [HL],[IX]	AND [HL],[IX]	CP [HL],[IX]	LD [IX+L],A	LD [IX+L],L			SLL [BR:*ll*]	RLC [BR:*ll*]	NEG [BR:*ll*]	OR H,#nn	LD XP,#pp	LD [hh*ll*],L	JRS P,rr	CARS P,rr
7	ADD [HL],[IY]	SUB [HL],[IY]	AND [HL],[IY]	CP [HL],[IY]	LD [IY+L],A	LD [IY+L],L			SLL [HL]	RLC [HL]	NEG [HL]	**未定義コード**	LD YP,#pp	LD [hh*ll*],H	JRS M,rr	CARS M,rr
8	ADC A,[IX+dd]	SBC A,[IX+dd]	OR A,[IX+dd]	XOR A,[IX+dd]	LD B,[IX+dd]	LD H,[IX+dd]	LD [IX],[IX+dd]	LD [IY],[IX+dd]	SRA A	RR A	SEP	XOR B,#nn	LD A,NB	MLT	JRS F0,rr	CARS F0,rr
9	ADC A,[IY+dd]	SBC A,[IY+dd]	OR A,[IY+dd]	XOR A,[IY+dd]	LD B,[IY+dd]	LD H,[IY+dd]	LD [IX],[IY+dd]	LD [IY],[IY+dd]	SRA B	RR B		XOR L,#nn	LD A,EP	DIV	JRS F1,rr	CARS F1,rr
A	ADC A,[IX+L]	SBC A,[IX+L]	OR A,[IX+L]	XOR A,[IX+L]	LD B,[IX+L]	LD H,[IX+L]	LD [IX],[IX+L]	LD [IY],[IX+L]	SRA [BR:*ll*]	RR [BR:*ll*]		XOR H,#nn	LD A,XP		JRS F2,rr	CARS F2,rr
B	ADC A,[IY+L]	SBC A,[IY+L]	OR A,[IY+L]	XOR A,[IY+L]	LD B,[IY+L]	LD H,[IY+L]	LD [IX],[IY+L]	LD [IY],[IY+L]	SRA [HL]	RR [HL]	**未定義コード領域**		LD A,YP		JRS F3,rr	CARS F3,rr
C	ADC [HL],A	SBC [HL],A	OR [HL],A	XOR [HL],A	LD [IX+dd],B	LD [IX+dd],H			SRL A	RRC A		CP B,#nn	LD NB,A	**未定義コード領域**	JRS NF0,rr	CARS NF0,rr
D	ADC [HL],#nn	SBC [HL],#nn	OR [HL],#nn	XOR [HL],#nn	LD [IY+dd],B	LD [IY+dd],H			SRL B	RRC B		CP L,#nn	LD EP,A		JRS NF1,rr	CARS NF1,rr
E	ADC [HL],[IX]	SBC [HL],[IX]	OR [HL],[IX]	XOR [HL],[IX]	LD [IX+L],B	LD [IX+L],H			SRL [BR:*ll*]	RRC [BR:*ll*]	HALT	CP H,#nn	LD XP,A		JRS NF2,rr	CARS NF2,rr
F	ADC [HL],[IY]	SBC [HL],[IY]	OR [HL],[IY]	XOR [HL],[IY]	LD [IY+L],B	LD [IY+L],H			SRL [HL]	RRC [HL]	SLP	CP BR,#hh	LD YP,A		JRS NF3,rr	CARS NF3,rr

第2オペコード(第1オペコード = CF)

L＼H	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	ADD BA,BA		ADD HL,BA		ADD IX,BA		ADC BA,#mmnn	LD BA,[SP+dd]				PUSH A	LD BA,[HL]	LD BA,[IX]	LD BA,BA	LD SP,BA
1	ADD BA,HL		ADD HL,HL		ADD IX,HL		ADC HL,#mmnn	LD HL,[SP+dd]				PUSH B	LD HL,[HL]	LD HL,[IX]	LD BA,HL	LD SP,HL
2	ADD BA,IX		ADD HL,IX		ADD IY,BA		SBC BA,#mmnn	LD IX,[SP+dd]				PUSH L	LD IX,[HL]	LD IX,[IX]	LD BA,IX	LD SP,IX
3	ADD BA,IY	未定義コード領域	ADD HL,IY		ADD IY,HL		SBC HL,#mmnn	LD IY,[SP+dd]				PUSH H	LD IY,[HL]	LD IY,[IX]	LD BA,IY	LD SP,IY
4	ADC BA,BA		ADC HL,BA		ADD SP,BA			LD [SP+dd],BA				POP A	LD [HL],BA	LD [IX],BA	LD HL,BA	LD HL,SP
5	ADC BA,HL		ADC HL,HL		ADD SP,HL			LD [SP+dd],HL				POP B	LD [HL],HL	LD [IX],HL	LD HL,HL	LD HL,PC
6	ADC BA,IX		ADC HL,IX	未定義コード領域				LD [SP+dd],IX				POP L	LD [HL],IX	LD [IX],IX	LD HL,IX	未定義コード領域
7	ADC BA,IY		ADC HL,IY					LD [SP+dd],IY		未定義コード領域		POP H	LD [HL],IY	LD [IX],IY	LD HL,IY	
8	SUB BA,BA	CP BA,BA	SUB HL,BA	CP HL,BA	SUB IX,BA		ADD SP,#mmnn	LD SP,[hh*ll*]				PUSH ALL		LD BA,[IY]	LD IX,BA	LD BA,SP
9	SUB BA,HL	CP BA,HL	SUB HL,HL	CP HL,HL	SUB IX,HL							PUSH ALE		LD HL,[IY]	LD IX,HL	LD BA,PC
A	SUB BA,IX	CP BA,IX	SUB HL,IX	CP HL,IX	SUB IY,BA		SUB SP,#mmnn							LD IX,[IY]	LD IX,IX	LD IX,SP
B	SUB BA,IY	CP BA,IY	SUB HL,IY	CP HL,IY	SUB IY,HL									LD IY,[IY]	LD IX,IY	未定義コード領域
C	SBC BA,BA		SBC HL,BA		SUB SP,BA	CP SP,BA	CP SP,#mmnn	LD [hh*ll*],SP				POP ALL		LD [IY],BA	LD IY,BA	
D	SBC BA,HL	未定義コード領域	SBC HL,HL		SUB SP,HL	CP SP,HL						POP ALE		LD [IY],HL	LD IY,HL	
E	SBC BA,IX		SBC HL,IX				LD SP,#mmnn							LD [IY],IX	LD IY,IX	LD IY,SP
F	SBC BA,IY		SBC HL,IY											LD [IY],IY	LD IY,IY	未定義コード

APPENDIX　B　アドレッシングモード別命令一覧表

アドレッシングモード別命令一覧表(1/12)

8ビット算術論理演算命令

ニーモニック	即値データ			レジスタ直接			レジスタ間接			ディスプレースメント付きレジスタ間接			インデックスレジスタ付きレジスタ間接			8ビット絶対			16ビット絶対		
	オペランド	バイト	サイクル	オペランド	バイト	サイクル	オペランド	バイト	サイクル	オペランド	バイト	サイクル	オペランド	バイト	サイクル	オペランド	バイト	サイクル	オペランド	バイト	サイクル
ADD	A,#nn	2	2	A,A	1	2	A,[HL]	1	2	A,[IX+dd]	3	4	A,[IX+L]	2	4	A,[BR:/]	2	3	A,[hh/]	3	4
				A,B	1	2	A,[IX]	1	2	A,[IY+dd]	3	4	A,[IY+L]	2	4						
							A,[IY]	1	2												
	[HL],#nn	3	5	[HL],A	2	4	[HL],[IX]	2	5												
							[HL],[IY]	2	5												
ADC	A,#nn	2	2	A,A	1	2	A,[HL]	1	2	A,[IX+dd]	3	4	A,[IX+L]	2	4	A,[BR:/]	2	3	A,[hh/]	3	4
				A,B	1	2	A,[IX]	1	2	A,[IY+dd]	3	4	A,[IY+L]	2	4						
							A,[IY]	1	2												
	[HL],#nn	3	5	[HL],A	2	4	[HL],[IX]	2	5												
							[HL],[IY]	2	5												
SUB	A,#nn	2	2	A,A	1	2	A,[HL]	1	2	A,[IX+dd]	3	4	A,[IX+L]	2	4	A,[BR:/]	2	3	A,[hh/]	3	4
				A,B	1	2	A,[IX]	1	2	A,[IY+dd]	3	4	A,[IY+L]	2	4						
							A,[IY]	1	2												
	[HL],#nn	3	5	[HL],A	2	4	[HL],[IX]	2	5												
							[HL],[IY]	2	5												
SBC	A,#nn	2	2	A,A	1	2	A,[HL]	1	2	A,[IX+dd]	3	4	A,[IX+L]	2	4	A,[BR:/]	2	3	A,[hh/]	3	4
				A,B	1	2	A,[IX]	1	2	A,[IY+dd]	3	4	A,[IY+L]	2	4						
							A,[IY]	1	2												
	[HL],#nn	3	5	[HL],A	2	4	[HL],[IX]	2	5												
							[HL],[IY]	2	5												
AND	A,#nn	2	2	A,A	1	2	A,[HL]	1	2	A,[IX+dd]	3	4	A,[IX+L]	2	4	A,[BR:/]	2	3	A,[hh/]	3	4
	B,#nn	3	2	A,B	1	2	A,[IX]	1	2	A,[IY+dd]	3	4	A,[IY+L]	2	4						
	L,#nn	3	2				A,[IY]	1	2												
	H,#nn	3	2																		
	SC,#nn	2	2																		
	[BR:/],#nn	3	5	[HL],A	2	4	[HL],[IX]	2	5												
	[HL],#nn	3	5				[HL],[IY]	2	5												
OR	A,#nn	2	2	A,A	1	2	A,[HL]	1	2	A,[IX+dd]	3	4	A,[IX+L]	2	4	A,[BR:/]	2	3	A,[hh/]	3	4
	B,#nn	3	3	A,B	1	2	A,[IX]	1	2	A,[IY+dd]	3	4	A,[IY+L]	2	4						
	L,#nn	3	3				A,[IY]	1	2												
	H,#nn	3	3																		
	SC,#nn	2	3																		
	[BR:/],#nn	3	5	[HL],A	2	4	[HL],[IX]	2	5												
	[HL],#nn	3	5				[HL],[IY]	2	5												

8ビット算術論理演算命令

ニーモニック	即値データ			レジスタ直接			レジスタ間接			ディスプレースメント付きレジスタ間接			インデックスレジスタ付きレジスタ間接			8ビット絶対			16ビット絶対		
	オペランド	バイト	サイクル	オペランド	バイト	サイクル	オペランド	バイト	サイクル	オペランド	バイト	サイクル	オペランド	バイト	サイクル	オペランド	バイト	サイクル	オペランド	バイト	サイクル
XOR	A,#nn	2	2	A,A	1	2	A,[HL]	1	2	A,[IX+dd]	3	4	A,[IX+L]	2	4	A,[BR:*ll*]	2	3	A,[hh/*ll*]	3	4
	B,#nn	3	3	A,B	1	2	A,[IX]	1	2	A,[IY+dd]	3	4	A,[IY+L]	2	4						
	L,#nn	3	3				A,[IY]	1	2												
	H,#nn	3	3																		
	SC,#nn	2	3																		
	[BR:*ll*],#nn	3	5	[HL],A	2	4	[HL],[IX]	2	5												
	[HL],#nn	3	5				[HL],[IY]	2	5												
CP	A,#nn	2	2	A,A	1	2	A,[HL]	1	2	A,[IX+dd]	3	4	A,[IX+L]	2	4	A,[BR:*ll*]	2	3	A,[hh/*ll*]	3	4
	B,#nn	3	3	A,B	1	2	A,[IX]	1	2	A,[IY+dd]	3	4	A,[IY+L]	2	4						
	L,#nn	3	3				A,[IY]	1	2												
	H,#nn	3	3																		
	BR,#hh	3	3																		
	[BR:*ll*],#nn	3	4	[HL],A	2	3	[HL],[IX]	2	4												
	[HL],#nn	3	4				[HL],[IY]	2	4												
BIT	A,#nn	2	2	A,B	1	2															
	B,#nn	2	2																		
	[BR:*ll*],#nn	3	4																		
	[HL],#nn	2	3																		
INC				A	1	2	[HL]	1	3							[BR:*ll*]	2	4			
				B	1	2															
				L	1	2															
				H	1	2															
				BR	1	2															
DEC				A	1	2	[HL]	1	3							[BR:*ll*]	2	4			
				B	1	2															
				L	1	2															
				H	1	2															
				BR	1	2															
CPL				A	2	3	[HL]	2	4							[BR:*ll*]	3	5			
				B	2	3															
NEG				A	2	3	[HL]	2	4							[BR:*ll*]	3	5			
				B	2	3															

8ビット転送命令

	即値データ			レジスタ直接			レジスタ間接			ディスプレースメント付きレジスタ間接			インデックスレジスタ付きレジスタ間接			8ビット絶対			16ビット絶対		
ニーモニック	オペランド	バイト	サイクル	オペランド	バイト	サイクル	オペランド	バイト	サイクル	オペランド	バイト	サイクル	オペランド	バイト	サイクル	オペランド	バイト	サイクル	オペランド	バイト	サイクル
LD	A,#nn	2	2	A,A	1	1	A,[HL]	1	2	A,[IX+dd]	3	4	A,[IX+L]	2	4	A,[BR:*ll*]	2	3	A,[hh*ll*]	4	5
				A,B	1	1	A,[IX]	1	2	A,[IY+dd]	3	4	A,[IY+L]	2	4						
				A,L	1	1	A,[IY]	1	2												
				A,H	1	1															
				A,BR	2	2															
				A,SC	2	2															
				A,NB	2	2															
				A,EP	2	2															
				A,XP	2	2															
				A,YP	2	2															
	B,#nn	2	2	B,A	1	1	B,[HL]	1	2	B,[IX+dd]	3	4	B,[IX+L]	2	4	B,[BR:*ll*]	2	3	B,[hh*ll*]	4	5
				B,B	1	1	B,[IX]	1	2	B,[IY+dd]	3	4	B,[IY+L]	2	4						
				B,L	1	1	B,[IY]	1	2												
				B,H	1	1															
	L,#nn	2	2	L,A	1	1	L,[HL]	1	2	L,[IX+dd]	3	4	L,[IX+L]	2	4	L,[BR:*ll*]	2	3	L,[hh*ll*]	4	5
				L,B	1	1	L,[IX]	1	2	L,[IY+dd]	3	4	L,[IY+L]	2	4						
				L,L	1	1	L,[IY]	1	2												
				L,H	1	1															
	H,#nn	2	2	H,A	1	1	H,[HL]	1	2	H,[IX+dd]	3	4	H,[IX+L]	2	4	H,[BR:*ll*]	2	3	H,[hh*ll*]	4	5
				H,B	1	1	H,[IX]	1	2	H,[IY+dd]	3	4	H,[IY+L]	2	4						
				H,L	1	1	H,[IY]	1	2												
				H,H	1	1															
	BR,#hh	2	2	BR,A	2	2															
	SC,#nn	2	3	SC,A	2	3															
	[BR:*ll*],#nn	3	4	[BR:*ll*],A	2	3	[BR:*ll*],[HL]	2	4												
				[BR:*ll*],B	2	3	[BR:*ll*],[IX]	2	4												
				[BR:*ll*],L	2	3	[BR:*ll*],[IY]	2	4												
				[BR:*ll*],H	2	3															
				[hh*ll*],A	4	5															
				[hh*ll*],B	4	5															
				[hh*ll*],L	4	5															
				[hh*ll*],H	4	5															

8ビット転送命令

	即値データ			レジスタ直接			レジスタ間接			ディスプレースメント付きレジスタ間接			インデックスレジスタ付きレジスタ間接			8ビット絶対			16ビット絶対		
ニーモニック	オペランド	バイト	サイクル	オペランド	バイト	サイクル	オペランド	バイト	サイクル	オペランド	バイト	サイクル	オペランド	バイト	サイクル	オペランド	バイト	サイクル	オペランド	バイト	サイクル
LD	[HL],#nn	2	3	[HL],A	1	2	[HL],[HL]	1	3	[HL],[IX+dd]	3	5	[HL],[IX+L]	2	5	[HL],[BR:*ll*]	2	4			
				[HL],B	1	2	[HL],[IX]	1	3	[HL],[IY+dd]	3	5	[HL],[IY+L]	2	5						
				[HL],L	1	2	[HL],[IY]	1	3												
				[HL],H	1	2															
	[IX],#nn	2	3	[IX],A	1	2	[IX],{HL]	1	3	[IX],[IX+dd]	3	5	[IX],[IX+L]	2	5	[IX],[BR:*ll*]	2	4			
				[IX],B	1	2	[IX],[IX]	1	3	[IX],[IY+dd]	3	5	[IX],[IY+L]	2	5						
				[IX],L	1	2	[IX],[IY]	1	3												
				[IX],H	1	2															
	[IY],#nn	2	3	[IY],A	1	2	[IY],[HL]	1	3	[IY],[IX+dd]	3	5	[IY],[IX+L]	2	5	[IY],[BR:*ll*]	2	4			
				[IY],B	1	2	[IY],[IX]	1	3	[IY],[IY+dd]	3	5	[IY],[IY+L]	2	5						
				[IY],L	1	2	[IY],[IY]	1	3												
				[IY],H	1	2															
				[IX+dd],A	3	4															
				[IX+dd],B	3	4															
				[IX+dd],L	3	4															
				[IX+dd],H	3	4															
				[IY+dd],A	3	4															
				[IY+dd],B	3	4															
				[IY+dd],L	3	4															
				[IY+dd],H	3	4															
				[IX+L],A	2	4															
				[IX+L],B	2	4															
				[IX+L],L	2	4															
				[IX+L],H	2	4															
				[IY+L],A	2	4															
				[IY+L],B	2	4															
				[IY+L],L	2	4															
				[IY+L],H	2	4															
	NB,#bb	3	4	NB,A	2	3															
	EP,#pp	3	3	EP,A	2	2															
	XP,#pp	3	3	XP,A	2	2															
	YP,#pp	3	3	YP,A	2	2															
EX				A,B	1	2	A,[HL]	1	3												
SWAP				A	1	2	[HL]	1	3												

アドレッシングモード別命令一覧表(5/12)

ローテート/シフト命令

ニーモニック	即値データ			レジスタ直接			レジスタ間接			ディスプレースメント付きレジスタ間接			インデックスレジスタ付きレジスタ間接			8ビット絶対			16ビット絶対		
	オペランド	バイト	サイクル	オペランド	バイト	サイクル	オペランド	バイト	サイクル	オペランド	バイト	サイクル	オペランド	バイト	サイクル	オペランド	バイト	サイクル	オペランド	バイト	サイクル
RL				A	2	3	[HL]	2	4							[BR:*II*]	3	5			
				B	2	3															
RLC				A	2	3	[HL]	2	4							[BR:*II*]	3	5			
				B	2	3															
RR				A	2	3	[HL]	2	4							[BR:*II*]	3	5			
				B	2	3															
RRC				A	2	3	[HL]	2	4							[BR:*II*]	3	5			
				B	2	3															
SLA				A	2	3	[HL]	2	4							[BR:*II*]	3	5			
				B	2	3															
SLL				A	2	3	[HL]	2	4							[BR:*II*]	3	5			
				B	2	3															
SRA				A	2	3	[HL]	2	4							[BR:*II*]	3	5			
				B	2	3															
SRL				A	2	3	[HL]	2	4							[BR:*II*]	3	5			
				B	2	3															

16ビット算術演算命令

	即値データ			レジスタ直接			レジスタ間接			ディスプレースメント付きレジスタ間接			インデックスレジスタ付きレジスタ間接			8ビット絶対			16ビット絶対		
ニーモニック	オペランド	バイト	サイクル	オペランド	バイト	サイクル	オペランド	バイト	サイクル	オペランド	バイト	サイクル	オペランド	バイト	サイクル	オペランド	バイト	サイクル	オペランド	バイト	サイクル
ADD	BA,#mmnn	3	3	BA,BA	2	4															
				BA,HL	2	4															
				BA,IX	2	4															
				BA,IY	2	4															
	HL,#mmnn	3	3	HL,BA	2	4															
				HL,HL	2	4															
				HL,IX	2	4															
				HL,IY	2	4															
	IX,#mmnn	3	3	IX,BA	2	4															
				IX,HL	2	4															
	IY,#mmnn	3	3	IY,BA	2	4															
				IY,HL	2	4															
	SP,#mmnn	4	4	SP,BA	2	4															
				SP,HL	2	4															
ADC	BA,#mmnn	4	4	BA,BA	2	4															
				BA,HL	2	4															
				BA,IX	2	4															
				BA,IY	2	4															
	HL,#mmnn	4	4	HL,BA	2	4															
				HL,HL	2	4															
				HL,IX	2	4															
				HL,IY	2	4															
SUB	BA,#mmnn	3	3	BA,BA	2	4															
				BA,HL	2	4															
				BA,IX	2	4															
				BA,IY	2	4															
	HL,#mmnn	3	3	HL,BA	2	4															
				HL,HL	2	4															
				HL,IX	2	4															
				HL,IY	2	4															
	IX,#mmnn	3	3	IX,BA	2	4															
				IX,HL	2	4															
	IY,#mmnn	3	3	IY,BA	2	4															
				IY,HL	2	4															
	SP,#mmnn	4	4	SP,BA	2	4															
				SP,HL	2	4															

16ビット算術演算命令

	即値データ			レジスタ直接			レジスタ間接			ディスプレースメント付きレジスタ間接			インデックスレジスタ付きレジスタ間接			8ビット絶対			16ビット絶対		
ニーモニック	オペランド	バイト	サイクル	オペランド	バイト	サイクル	オペランド	バイト	サイクル	オペランド	バイト	サイクル	オペランド	バイト	サイクル	オペランド	バイト	サイクル	オペランド	バイト	サイクル
SBC	BA,#mmnn	4	4	BA,BA	2	4															
				BA,HL	2	4															
				BA,IX	2	4															
				BA,IY	2	4															
	HL,#mmnn	4	4	HL,BA	2	4															
				HL,HL	2	4															
				HL,IX	2	4															
				HL,IY	2	4															
CP	BA,#mmnn	3	3	BA,BA	2	4															
				BA,HL	2	4															
				BA,IX	2	4															
				BA,IY	2	4															
	HL,#mmnn	3	3	HL,BA	2	4															
				HL,HL	2	4															
				HL,IX	2	4															
				HL,IY	2	4															
	IX,#mmnn	3	3																		
	IY,#mmnn	3	3																		
	SP,#mmnn	4	4	SP,BA	2	4															
				SP,HL	2	4															
INC				BA	1	2															
				HL	1	2															
				IX	1	2															
				IY	1	2															
				SP	1	2															
DEC				BA	1	2															
				HL	1	2															
				IX	1	2															
				IY	1	2															
				SP	1	2															

16ビット転送命令

	即値データ			レジスタ直接			レジスタ間接			ディスプレースメント付きレジスタ間接			インデックスレジスタ付きレジスタ間接			8ビット絶対			16ビット絶対		
ニーモニック	オペランド	バイト	サイクル	オペランド	バイト	サイクル	オペランド	バイト	サイクル	オペランド	バイト	サイクル	オペランド	バイト	サイクル	オペランド	バイト	サイクル	オペランド	バイト	サイクル
LD	BA,#mmnn	3	3	BA,BA	2	2	BA,[HL]	2	5	BA,[SP+dd]	3	6							BA,[hh*ll*]	3	5
				BA,HL	2	2	BA,[IX]	2	5												
				BA,IX	2	2	BA,[IY]	2	5												
				BA,IY	2	2															
				BA,SP	2	2															
				BA,PC	2	2															
	HL,#mmnn	3	3	HL,BA	2	2	HL,[HL]	2	5	HL,[SP+dd]	3	6							HL,[hh*ll*]	3	5
				HL,HL	2	2	HL,[IX]	2	5												
				HL,IX	2	2	HL,[IY]	2	5												
				HL,IY	2	2															
				HL,SP	2	2															
				HL,PC	2	2															
	IX,#mmnn	3	3	IX,BA	2	2	IX,[HL]	2	5	IX,[SP+dd]	3	6							IX,[hh*ll*]	3	5
				IX,HL	2	2	IX,[IX]	2	5												
				IX,IX	2	2	IX,[IY]	2	5												
				IX,IY	2	2															
				IX,SP	2	2															
	IY,#mmnn	3	3	IY,BA	2	2	IY,[HL]	2	5	IY,[SP+dd]	3	6							IY,[hh*ll*]	3	5
				IY,HL	2	2	IY,[IX]	2	5												
				IY,IX	2	2	IY,[IY]	2	5												
				IY,IY	2	2															
				IY,SP	2	2															
	SP,#mmnn	4	4	SP,BA	2	2													SP,[hh*ll*]	4	5
				SP,HL	2	2															
				SP,IX	2	2															
				SP,IY	2	2															
				[hh*ll*],BA	3	5															
				[hh*ll*],HL	3	5															
				[hh*ll*],IX	3	5															
				[hh*ll*],IY	3	5															
				[hh*ll*],SP	4	6															

アドレッシングモード別命令一覧表(9/12)

16ビット転送命令

ニーモニック	即値データ			レジスタ直接			レジスタ間接			ディスプレースメント付きレジスタ間接			インデックスレジスタ付きレジスタ間接			8ビット絶対			16ビット絶対		
	オペランド	バイト	サイクル	オペランド	バイト	サイクル	オペランド	バイト	サイクル	オペランド	バイト	サイクル	オペランド	バイト	サイクル	オペランド	バイト	サイクル	オペランド	バイト	サイクル
LD				[HL],BA	2	5															
				[HL],HL	2	5															
				[HL],IX	2	5															
				[HL],IY	2	5															
				[IX],BA	2	5															
				[IX],HL	2	5															
				[IX],IX	2	5															
				[IX],IY	2	5															
				[IY],BA	2	5															
				[IY],HL	2	5															
				[IY],IX	2	5															
				[IY],IY	2	5															
				[SP+dd],BA	3	6															
				[SP+dd],HL	3	6															
				[SP+dd],IX	3	6															
				[SP+dd],IY	3	6															
EX				BA,HL	1	3															
				BA,IX	1	3															
				BA,IY	1	3															
				BA,SP	1	3															

分岐命令

	8ビット間接			16ビット間接			符号付き８ビット PC相対			符号付き16ビット PC相対			レジスタ直接			その他		
ニーモニック	オペランド	バイト	サイクル	オペランド	バイト	サイクル	オペランド	バイト	サイクル	オペランド	バイト	サイクル	オペランド	バイト	サイクル	オペランド	バイト	サイクル
JRS							rr	2	2									
							C,rr	2	2									
							NC,rr	2	2									
							Z,rr	2	2									
							NZ,rr	2	2									
							LT,rr	3	3									
							LE,rr	3	3									
							GT,rr	3	3									
							GE,rr	3	3									
							V,rr	3	3									
							NV,rr	3	3									
							P,rr	3	3									
							M,rr	3	3									
							F0,rr	3	3									
							F1,rr	3	3									
							F2,rr	3	3									
							F3,rr	3	3									
							NF0,rr	3	3									
							NF1,rr	3	3									
							NF2,rr	3	3									
							NF3,rr	3	3									
JRL										qqrr	3	3						
										C,qqrr	3	3						
										NC,qqrr	3	3						
										Z,qqrr	3	3						
										NZ,qqrr	3	3						
JP	[kk]	2	4										HL	1	2			
DJR							NZ,rr	2	4									

分岐命令

サイクル (Min.: ミニマムモード、Max.: マキシマムモード、スキップ: 分岐なし)

	8ビット間接				16ビット間接				符号付き8ビットPC相対					符号付き16ビットPC相対					その他			
			サイクル				サイクル				サイクル					サイクル					サイクル		
ニーモニック	オペランド	バイト	Min.	Max.	オペランド	バイト	Min.	Max.	オペランド	バイト	Min.	Max.	スキップ	オペランド	バイト	Min.	Max.	スキップ	オペランド	バイト	Min.	Max.
CARS									rr	2	4	5	–									
									C,rr	2	4	5	2									
									NC,rr	2	4	5	2									
									Z,rr	2	4	5	2									
									NZ,rr	2	4	5	2									
									LT,rr	3	5	6	3									
									LE,rr	3	5	6	3									
									GT,rr	3	5	6	3									
									GE,rr	3	5	6	3									
									V,rr	3	5	6	3									
									NV,rr	3	5	6	3									
									P,rr	3	5	6	3									
									M,rr	3	5	6	3									
									F0,rr	3	5	6	3									
									F1,rr	3	5	6	3									
									F2,rr	3	5	6	3									
									F3,rr	3	5	6	3									
									NF0,rr	3	5	6	3									
									NF1,rr	3	5	6	3									
									NF2,rr	3	5	6	3									
									NF3,rr	3	5	6	3									
CARL														qqrr	3	5	6	–				
														C,qqrr	3	5	6	3				
														NC,qqrr	3	5	6	3				
														Z,qqrr	3	5	6	3				
														NZ,qqrr	3	5	6	3				
CALL					[hh*ll*]	3	7	8														
RET																				2	3	4
RETE																				1	4	5
RETS																				1	5	6
INT	[kk]	2	7	8																		

スタック制御命令

ニーモニック	レジスタ直接			インプライド		
	オペランド	バイト	サイクル	オペランド	バイト	サイクル
POP	A	2	3	ALL	2	11
	B	2	3			
	L	2	3			
	H	2	3			
	BR	1	2	ALE	2	14
	SC	1	2			
	EP	1	2			
	IP	1	3			
	BA	1	3			
	HL	1	3			
	IX	1	3			
	IY	1	3			
PUSH	A	2	3	ALL	2	12
	B	2	3			
	L	2	3			
	H	2	3			
	BR	1	3	ALE	2	15
	SC	1	3			
	EP	1	3			
	IP	1	4			
	BA	1	4			
	HL	1	4			
	IX	1	4			
	IY	1	4			

乗除算命令

ニーモニック	インプライド		
	オペランド	バイト	サイクル
MLT		2	12
DIV		2	13

演算補助命令

ニーモニック	インプライド		
	オペランド	バイト	サイクル
PACK		1	2
UPCK		1	2
SEP		2	3

システム制御命令

ニーモニック			
	オペランド	バイト	サイクル
NOP		1	2
HALT		2	3
SLP		2	3

APPENDIX C 命令索引

(アルファベット順)

CARL PC相対ロングコール

CARS PC相対ショートコール

CP 8ビット比較

CP 16ビット比較

CPL ビット反転(1の補数)

DEC 8ビットデクリメント(-1)

DEC 16ビットデクリメント(-1)

DIV 除算

DJR PC相対ショートジャンプ(Bレジスタデクリメント付き)

EX 8ビットデータ交換

EX 16ビットデータ交換

LD　8ビットロード

LD　8ビットロード

LD　16ビットロード

セイコーエプソン株式会社
電子デバイス営業本部

ED営業推進部　〒191-8501　東京都日野市日野421-8
IC営業技術G　TEL（042）587-5816（直通）　FAX（042）587-5624
〈東日本〉
ED東京営業部　〒191-8501　東京都日野市日野421-8
東京IC営業G　TEL（042）587-5313（直通）　FAX（042）587-5116
〈西日本〉
ED大阪営業部　〒541-0059　大阪市中央区博労町3-5-1　エプソン大阪ビル15F
TEL（06）6120-6000（代表）　FAX（06）6120-6100
〈東海・北陸〉
ED名古屋営業部　〒461-0005　名古屋市東区東桜1-10-24　栄大野ビル4F
TEL（052）953-8031（代表）　FAX（052）953-8041
〈長野〉
ED長野営業部　〒392-8502　長野県諏訪市大和3-3-5
TEL（0266）58-8171（直通）　FAX（0266）58-9917
〈東北〉
ED仙台営業所　〒980-0013　宮城県仙台市青葉区花京院1-1-20　花京院スクエア19F
TEL（022）263-7975（代表）　FAX（022）263-7990

インターネットによる電子デバイスのご紹介　http://www.epson.co.jp/device/

2001年2月改訂
1993年2月作成 ⓂB