TOSHIBA

TLCS-870/C1 シリーズ　命令セット

株式会社 東芝 セミコンダクター社

第 1 章　概要

TLCS-870/C1 シリーズの基本機械命令は、133 種 732 命令で、下表に命令の分類を示します。TLCS-870/C1 シリーズには、1 バイト長から最長 5 バイト長の命令があります。使用頻度の高い命令は、オブジェクトコードを短くしており、メモリ効率の良いプログラムを組むことができます。

また、TLCS-870/C1 シリーズは、メモリマップド I/O 方式の採用によりシンプルな命令体系で、ニモニックは 42 種類でありながら、18 種類におよぶアドレッシングモードにより強力なメモリ操作が可能です。

基本機械命令のほかに、コーディング効率の向上を図るためアセンブラによる拡張機械命令が用意されています。

表 1-1 命令セット

転送 / 交換	8 ビットデータ転送 / 交換		7 種	49 命令
	16 ビットデータ転送 / 交換		7	43
	フラグ操作		5	5
	SP 操作		1	2
	プッシュ / ポップ		4	6
演算	8 ビット演算	比較	4	29
		増減	4	28
		算術演算	16	116
		論理演算	12	87
		十進補正	2	2
	16 ビット演算	比較	3	15
		増減	2	2
		算術演算	12	60
		論理演算	9	45
		2 の補数	1	1
	乗除算		2	2
シフト / ローテート	シフト	8 ビット (論理)	2	2
		16 ビット (算術)	2	2
	ローテート	8 ビット	2	2
ニブル処理	スワップ / ニブルローテート		3	27
ビット操作	セット / クリア / 反転 / 転送 / 交換 / 演算		18	162
分岐	ジャンプ		6	24
	コール		4	16
	リターン		3	3
その他	ソフトウエア割り込み / その他		2	2
計			133 種	732 命令

表 1-2 アドレッシングモード（１ / ２）

レジスタ間接	7 種
ダイレクト	2
レジスタ	1

表 1-2 アドレッシングモード（2 / 2）

イミディエート	1
リラティブ	2
アブソリュート	1
ベクタ	1
直接ビット	2
レジスタ間接ビット	1
計	18

1.1　記号の説明

以下の命令 / アドレッシングモードの説明では、次の記号を使用します。

記号	説明	記号	説明
A	A レジスタ	r, g	8 ビットレジスタ (表 1-3 参照)
W	W レジスタ	rr, gg	16 ビットレジスタ (表 1-4 参照)
B	B レジスタ	n	4 ビットまたは 8 ビットイミディエートデータ
C	C レジスタ	mn	16 ビットイミディエートデータ
D	D レジスタ	d	符号付き 5 ビットまたは 8 ビットディスプレースメント (–16~+15/–128~+127)
E	E レジスタ	x, y	8 ビットダイレクトアドレス (0x0000 ~ 0x00FF)
H	H レジスタ	vw, uz	16 ビットダイレクトアドレス (0x0000 ~ 0xFFFF)
L	L レジスタ	(XX)	XX で指定されるアドレスのメモリの内容
WA	レジスタペア WA	(XX + 1, XX)	XX で指定されるアドレスから連続する 2 バイトのメモリの内容
BC	レジスタペア BC	b	ビット番号 (0~7)
DE	レジスタペア DE	.b	b で指定されるビットの内容
HL	レジスタペア HL	←	転送
IX	IX レジスタ	↔	交換
IY	IY レジスタ	+	加算
PC	プログラムカウンタ	–	減算
SP	スタックポインタ	×	乗算
PSW	プログラムステータスワード	÷	除算
JF	ジャンプステータスフラグ	&	ビットごとの論理積
ZF	ゼロフラグ	|	ビットごとの論理和
CF	キャリーフラグ (1 ビットアキュムレータ)	^	ビットごとの排他的論理和
HF	ハーフキャリーフラグ	null	ノーオペレーション (何も実行せず、次のアドレスの命令に移ります。)
SF	サインフラグ	$	命令の先頭アドレス (命令実行中のプログラムカウンタの内容は、$ + 2 または $ + 3 になります。)
VF	オーバフローフラグ	(src)	ソースメモリ
$\overline{CF}$	キャリーフラグの内容の反転	(dst)	デスティネーションメモリ
IMF	割り込みマスタ許可フラグ	RBS	レジスタバンクセレクタ
NxtOp	ネクストオペレーションアドレス (次の命令の先頭アドレスの値)		

表 1-3

r, g	8 ビットレジスタ
0	A
1	W
2	C
3	B
4	E
5	D
6	L
7	H

表 1-4

rr, gg	16 ビットレジスタ
0	WA
1	BC
2	DE
3	HL
4	IX
5	IY
6	SP
7	HL

フラグのセット条件	
*	オペレーションで指定された値がセットされます。
Z	ゼロ検出情報がセットされます。 • 転送: 8 ビットのソースデータが 0x00 のとき、“1” がセットされます。0x00 以外のときは “0” がセットされます。 • 交換: 交換前の g または (src) の内容が 0x00 のとき “1” がセットされます。0x00 以外のときは “0” がセットされます。 • 演算: 演算結果が 0x00 (8 ビット演算), 0x0000 (16 ビット演算) のとき “1” がセットされます。それ以外のときは “0” がセットされます。 ただし、乗算のときは積の上位 8 ビットが、除算のときは余りが、それぞれ 0x00 のとき “1” にセットされます。0x00 以外のときは “0” がセットされます。 • シフト / ローテート: シフト / ローテート後のレジスタの内容が 0x00 のとき “1” がセットされます。0x00 以外のときは “0” がセットされます。 • その他: JF の欄に Z とある場合は、ZF にセットされる値が JF にもセットされることを示します。
C	キャリー情報がセットされます。 • 加算: 最上位ビットからキャリー (桁上げ) がセットされます。 • 減算: 最上位ビットへのボロー (桁借り) がセットされます。 • 除算: 除数が 0x00 のときまたは商が 0x100 以上のとき “1” がセットされます。それ以外のときは “0” がセットされます。 • その他: JF の欄に C とある場合は、CF にセットされる値が JF にもセットされることを示します。
$\overline{C}$	CF にセットされる値の反転値がセットされます。
H	ハーフキャリー情報がセットされます。 • 加算: ビット 3 からのキャリー (桁上げ) がセットされます。 • 減算: ビット 3 へのボロー (桁借り) がセットされます。
S	サイン情報 (データの最上位ビット) がセットされます。
V	オーバフロー情報がセットされます。
$\overline{J}$	JF にセットされる値の反転値がセットされます。
1	“1” がセットされます。
0	“0” がセットされます。
U	不定値がセットされます。
−	フラグは変化せず、命令実行前の値が保持されます。

1.2　ニモニック

TLCS-870/C1 シリーズの命令のニモニックの規則は、以下のとおりです。

ニモニックは、オペコードとオペランドから構成されています (オペランドのない命令もあります)。オペコードの次に 1 つ以上のスペースを空けてオペランドを置きます。2 つ以上のオペランドがある場合は、オペランドをカンマで区切って並べます。

ソースオペランドとデスティネーションオペランドの 2 つがある場合は、必ずデスティネーションオペランドを先に置きます。ソースオペランドが複数ある場合は、先に被演算数を置きます。

オペランドにビット指定が含まれる場合は、アドレス指定とビット指定はピリオドで区切ります。

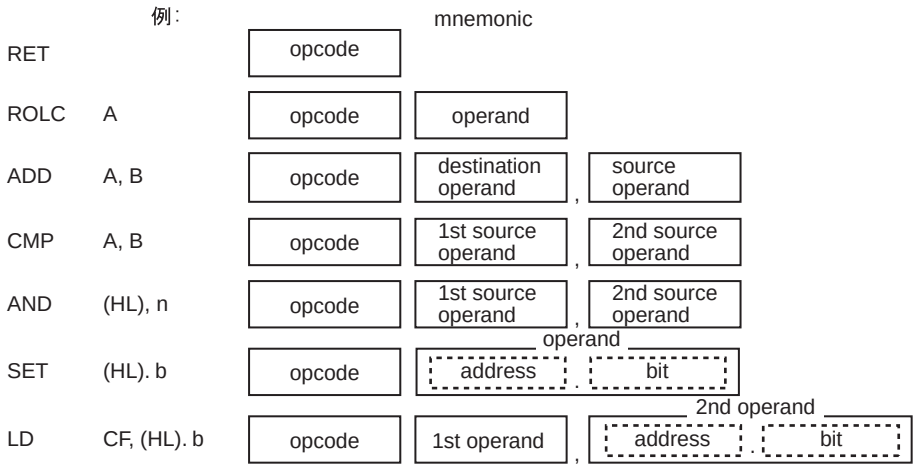


表 1-5 ニモニック一覧（１ / ２）

ニモニック	機能
ADD ADDC AND	Add Add with carry Logical AND
CALL CALLV CLR CMP CPL	Call Vector call Clear bit/byte Compare 1's complement bit
DAA DAS DEC DI* DIV	Decimal adjust for 8-bit addition Decimal adjust for 8-bit subtraction Decrement byte/word (Register) Disable maskable interrupt Divide byte quotient
EI*	Enable interrupt
INC	Increment byte/word (Register)
J* JP JR JRS	Optimized jump Absolute jump Relative jump Short relative jump
LD LDW	Load bit/byte/word (Register)/effective address Load word (Memory)
MUL	Multiply
NEG	Negate
NOP	No operation
OR	Logical OR

表 1-5 ニモニック一覧（２ / ２）

ニモニック	機能
POP PUSH	Pop up Push down
RET RETI RETN ROLC ROLD RORC RORD	Return from subroutine Return from maskable interrupt service routine Return from non-maskable interrupt service routine Rotate left through carry Rotate left digit Rotate right through carry Rotate right digit
SET SHLC SHLCA SHRC SHRCA SUB SUBB SWAP SWI	Bit test and set Logical shift left Arithmetic shift left Logical shift right Arithmetic shift right Subtract Subtract with borrow Swap nibble Software interrupt
TEST*	Bit test
XCH XOR	Exchange Logical exclusive OR

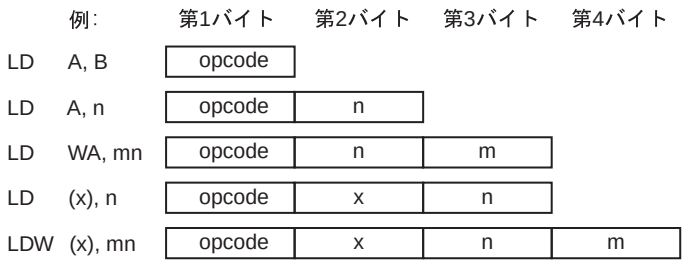
注) *: アセンブラ拡張機械命令

1.3 オブジェクトコードフォーマット

TLCS-870/C1 シリーズは、1 バイトオペコード命令と 2 バイトオペコード命令を持っています。

1.3.1 1 バイトオペコード命令のコードフォーマット

第 1 バイトにオペコードを置き、第 2 バイト以降にオペランドを置きます。オペランドが 2 バイトデータの場合、下位バイトを先に、上位バイトを後に置きます。また、オペランドがソースとデスティネーションの 2 つである場合、ソースが即値のとき (例 : LD (x), n 命令) はデスティネーションより後に置きます。



1.3.2 2 バイトオペコード命令のコードフォーマット

第 1 バイトに第 1 オペコードを置き、次に第 1 オペコードで指定されたオペランドを置き、その後に第 2 オペコードと第 2 オペコードで指定されたオペランドを置きます。

第 1 オペコードは、アドレッシングモード指定用のコードでプリフィックスコードとも呼びます。プリフィックスには、レジスタを指定するレジスタプリフィックスと、ソースまたはデスティネーションメモリを指定するソース / デスティネーションメモリプリフィックスがあります。

なお、次の 5 命令の第 1 オペコードは、指定レジスタの内容を無視します。

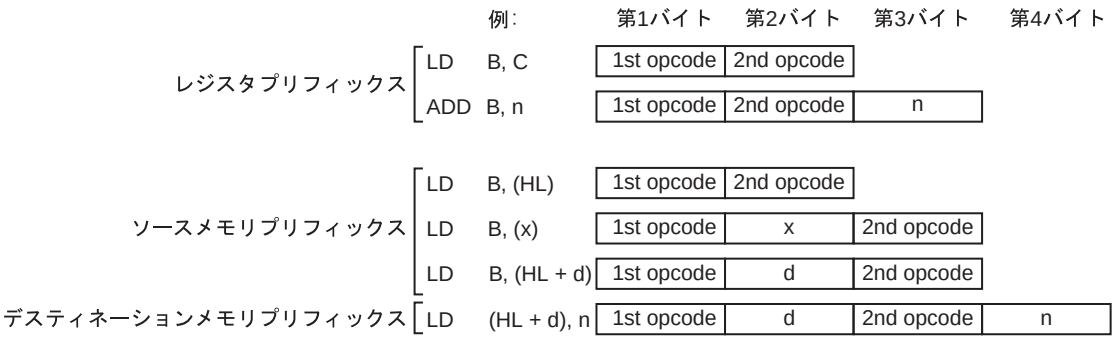
① RETN

② LD PSW, n

③ PUSH PSW

④ POP PSW

⑤ JR M/P/SLT/SGE/SLE/SGT/VS/VC, a



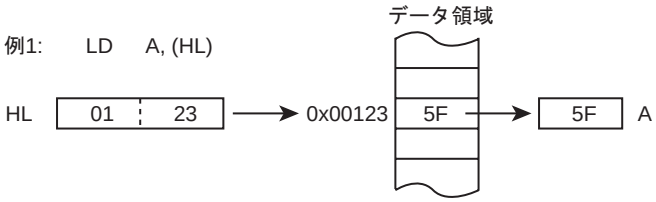
1.4　アドレッシングモード

TLCS-870/C1 シリーズには、17 種類のアドレッシングモード (アドレス指定の方法) があります。なお、命令によっては、複数のアドレッシングモードが組み合わさることもあります。

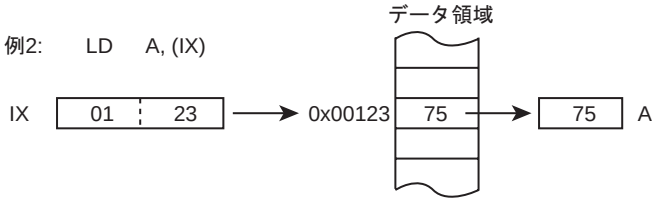
1.4.1　レジスタ間接

1.4.1.1　レジスタ間接 (HL), (DE), (IX), (IY)

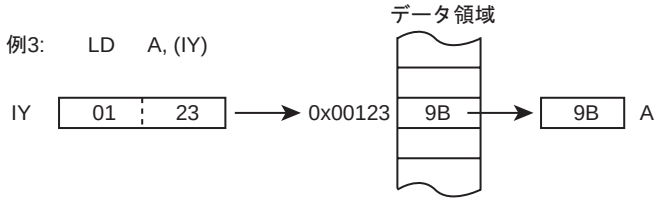
16 ビットレジスタ HL, DE, IX, IY の内容で指定されるアドレス。



HL レジスタの内容で指定されるアドレスすなわち 0x00123 番地のメモリの内容 0x5F が A レジスタにロードされます。



IX レジスタの内容で指定されるアドレスすなわち 0x00123 番地のメモリの内容 0x75 が A レジスタにロードされます。

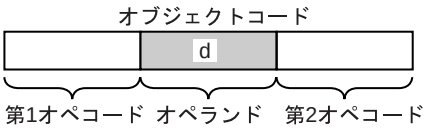


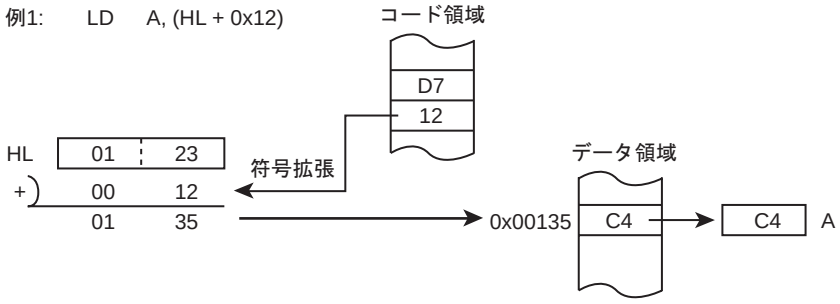
IY レジスタの内容で指定されるアドレスすなわち 0x00123 番地のメモリの内容 0x9B が A レジスタにロードされます。

1.4.1.2 8 ビットディスプレースメント オフセット付きレジスタ間接 (HL + d), (IX + d), (IY + d)

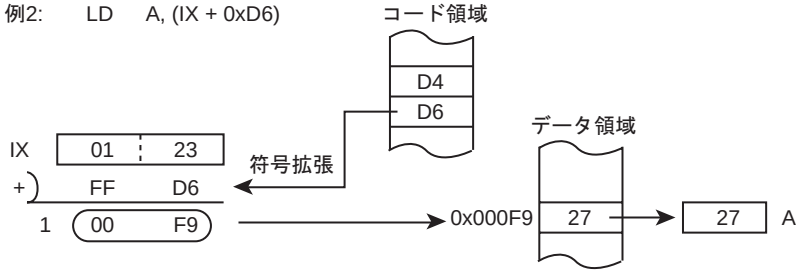
16 ビットレジスタ HL, IX, IY の内容にオブジェクトコード中の 8 ビットディスプレイスメント d を符号拡張 (下表参照) して加算した値で指定されるアドレス。なお、16 ビットレジスタ HL, IX, IY の内容は変化しません。

ディスプレースメント d	符号拡張した値
0x00 ~ 0x7F	0x0000 ~ 0x007F (0~+127)
0x80 ~ 0xFF	0xFF80 ~ 0xFFFF (–128~–1)





HL レジスタの内容 (0x0123) にディスプレースメント (0x12) を符号拡張して (0x0012) 加算した値で指定されるアドレスすなわち 0x00135 番地のメモリの内容 0xC4 が A レジスタにロードされます。

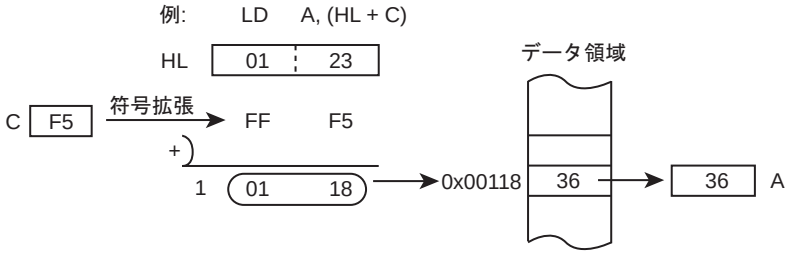


IX レジスタの内容 (0x0123) にディスプレースメント (0xD6) を符号拡張して (0xFFD6) 加算した値で指定されるアドレスすなわち 0x000F9 番地のメモリの内容 0x27 が A レジスタにロードされます。

1.4.1.3　レジスタインデックス (HL + C)

HL レジスタの内容に C レジスタの内容を符号拡張 (下表参照) して、加算した値で指定されるアドレス。なお、HL レジスタ , C レジスタの内容は変化しません。

C レジスタ	符号拡張した値
0x00 ~ 0x7F	0x0000 ~ 0x007F (0~+127)
0x80 ~ 0xFF	0xFF80 ~ 0xFFFF (–128~–1)

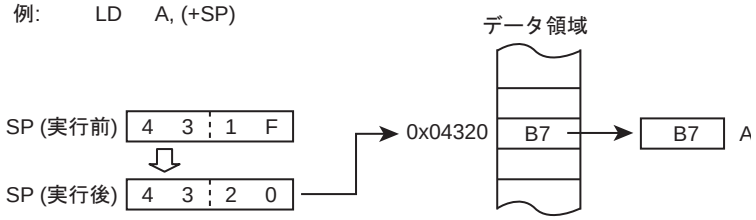


HL レジスタの内容 (0x0123) に C レジスタの内容 (0xF5) を符号拡張して (0xFFF5) 加算した値で指定されるアドレスすなわち 0x00118 番地のメモリの内容 0x36 が A レジスタにロードされます。

1.4.1.4　スタックポインタ間接オートプリインクリメント (+SP)

まず、SP の内容をインクリメントします。実効アドレスはインクリメントされた SP の内容となります。SP のインクリメントによるフラグ変化はありません。

このアドレッシングモードは、ソース (転送元) メモリのアドレス指定にのみ使用できます。

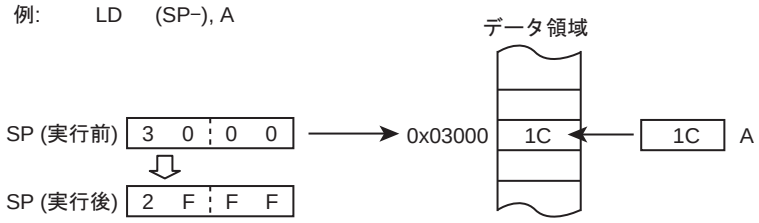


SP の内容をインクリメントし、その値で指定されるアドレスすなわち 0x04320 番地のメモリの内容 0xB7 が A レジスタにロードされます。

1.4.1.5　スタックポインタ間接オートポストデクリメント (SP–)

実効アドレスは SP の内容となります。データ処理後、SP の内容は自動的にデクリメントされます。SP のデクリメントによるフラグ変化はありません。

このアドレッシングモードは、デスティネーション (転送元) メモリのアドレス指定にのみ使用できます。

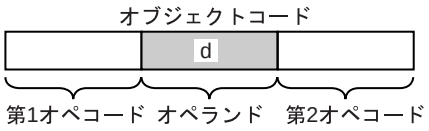


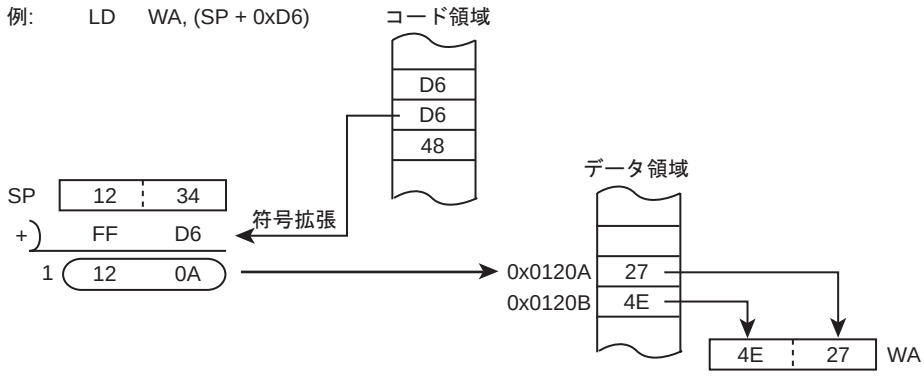
SP の内容で指定されるアドレスすなわち 0x03000 番地のメモリに A レジスタの内容 0x1C がストアされます。その後、SP はデクリメントされ、0x2FFF となります。

1.4.1.6 8 ビットディスプレースメントオフセット付きスタックポインタ間接 (SP + d)

スタックポインタ SP の内容に、オブジェクトコード中の 8 ビットディスプレースメント d を符号拡張 (下表参照) して、加算した値が実効アドレスとなります。なお、SP の内容は変化しません。

ディスプレースメント d	符号拡張した値
0x00 ~ 0x7F	0x0000 ~ 0x007F (0~+127)
0x80 ~ 0xFF	0xFF80 ~ 0xFFFF (–128~–1)



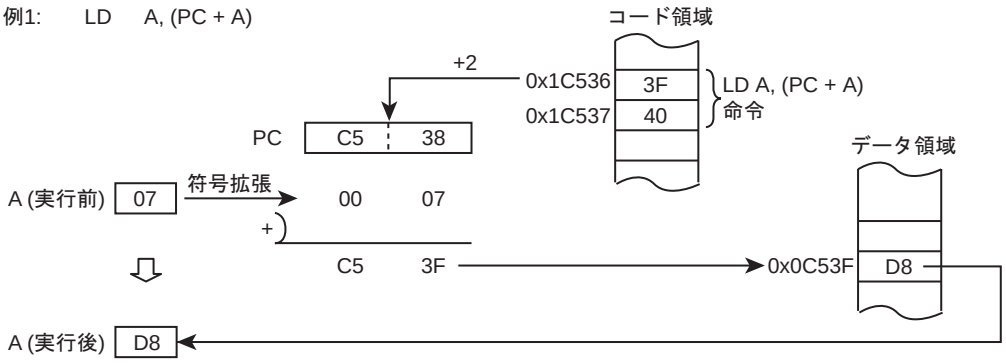


SP の内容 (0x1234) にディスプレースメント (0xD6) を符号拡張して (0xFFD6) 加算した値で指定されるアドレスすなわち 0x0120A 番地から連続する 2 バイトのメモリの内容 0x4E27 が WA レジスタにロードされます。

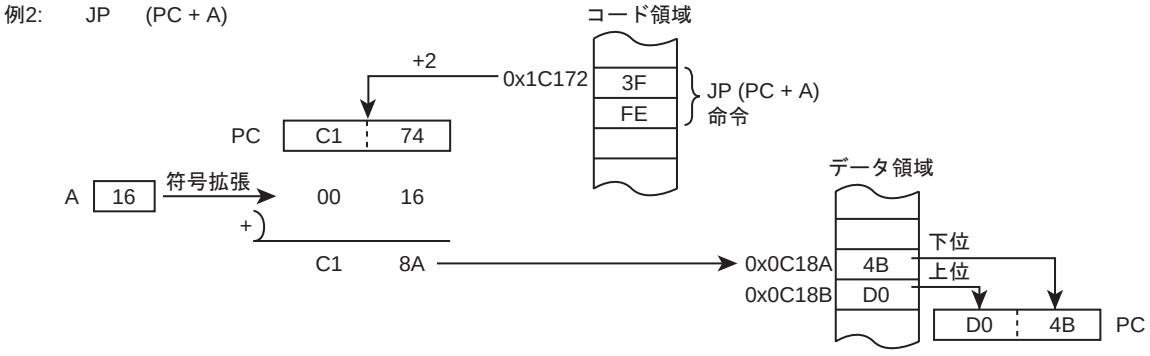
1.4.1.7 レジスタオフセットリラティブインデックス (PC + A)

プログラムカウンタの内容 (実行命令の先頭アドレス +2) に A レジスタの内容を符号拡張 (下表参照) して加算した値で指定されるアドレス。このアドレッシングモードは、ソース (転送元) アドレスの指定にのみ使用できます。このアドレッシングモードを使用することにより、BCD コード→7 セグメントコードなどのコード変換 , テーブルルックアップ , テーブルサーチや n 通りの多方向分岐処理などを容易にプログラムすることができます。

アキュムレータ	符号拡張した値
0x00 ~ 0x7F	0x0000 ~ 0x007F (0~+127)
0x80 ~ 0xFF	0xFF80 ~ 0xFFFF (–128~–1)



プログラムカウンタの内容 (0xC538) にA レジスタの内容 (0x07) を符号拡張して (0x0007) 加算した値で指定されるアドレスすなわち 0x0C53F 番地のメモリの内容 0xD8 が A レジスタにロードされます。



プログラムカウンタの内容 (0xC174) にA レジスタの内容 (0x16) を符号拡張して (0x0016) 加算した値で指定されるアドレスすなわち0x0C18A番地から連続する2バイトのメモリ内容0xD04Bがプログラムカウンタにロードされます。すなわち 0x1D04B 番地にジャンプします。

注) 870/C と 870/C1 での（PC+A) の扱いの違い

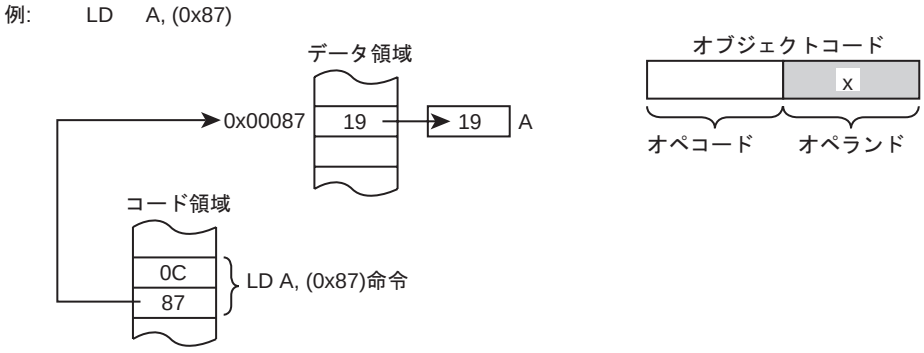
870/C1 では (PC+A) を使用してアクセスする領域はコード領域ではなく、データ領域となります。そのため、(PC+A) によるアドレッシングには互換性がありません。

870/C から 870/C1 へのプログラム移植時に注意が必要です。

1.4.2　ダイレクト

1.4.2.1　8 ビットダイレクト (x)

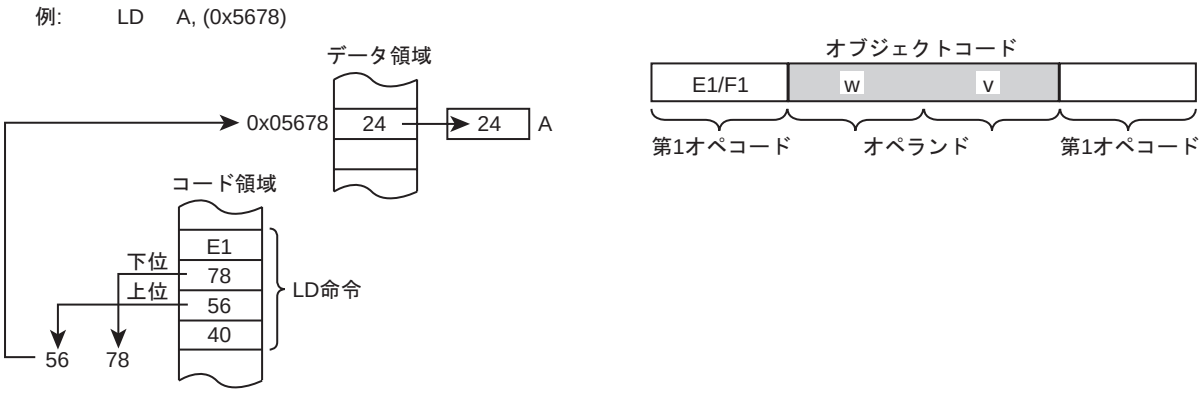
オブジェクトコード中の 8 ビット値 x で直接指定される 0x00000 ~ 0x000FF 番地のアドレス。



オブジェクトコードの x (0x87) で直接指定されるアドレスすなわち 0x00087 番地の内容 0x19 が A レジスタにロードされます。

1.4.2.2 16 ビットダイレクト (vw)

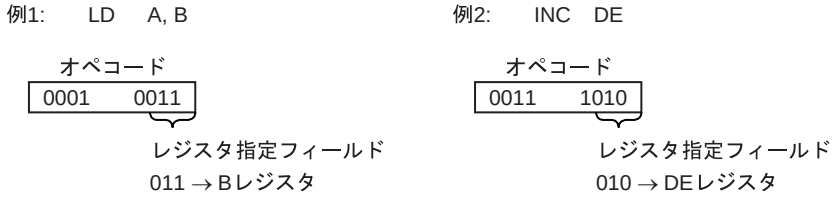
オブジェクトコード中の16ビット値vwで直接指定される0x00000 ~ 0x0FFFF番地のアドレス。



オブジェクトコードの vw (0x5678) で直接指定されるアドレスすなわち 0x05678 番地の内容 0x24 が A レジスタにロードされます。

1.4.3 レジスタ r または rr

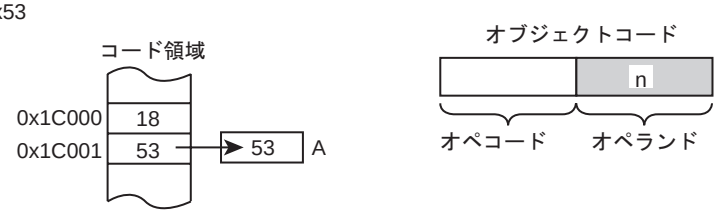
オブジェクトコード (オペコード) 中のレジスタ指定フィールドにより指定されるレジスタが操作対象となります。



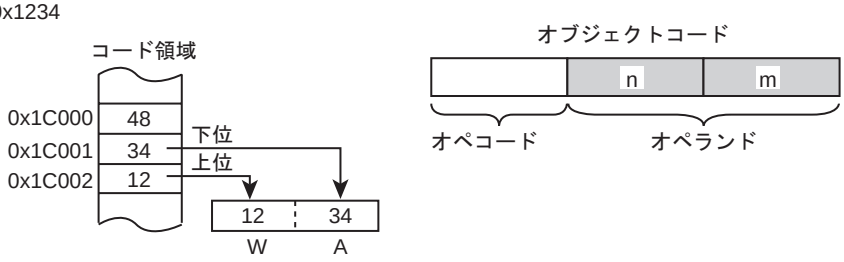
1.4.4 イミディエート n または mn

オブジェクトコード中の即値 (イミディエートデータ) が操作対象となります。なお、16 ビット即値の場合、若いアドレスの方から下位 8 ビット → 上位 8 ビットの順にメモリに格納します。

例1:　LD　A, 0x53



例2:　LD　WA, 0x1234



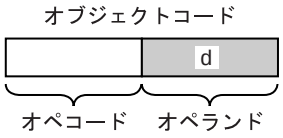
注）アセンブラソースプログラム記述上、即値をカッコで囲むことはできません。カッコで囲んだ場合はダイレクトアドレッシングモードと見なされます。

1.4.5　リラティブ (相対)

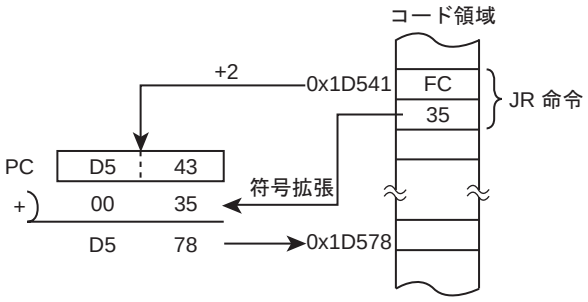
1.4.5.1　8 ビットディスプレースメントリラティブ

プログラムカウンタの内容 (実行命令の先頭アドレス +2 または +3) に、オブジェクトコード中の 8 ビットディスプレースメント値 d を符号拡張 (下表参照) して、加算した値で指定されるアドレス。このアドレッシングモードを持つ命令は、JR 命令のみです。

ディスプレースメント d	符号拡張した値
0x00 ~ 0x7F	0x0000 ~ 0x007F (0~+127)
0x80 ~ 0xFF	0xFF80 ~ 0xFFFF (–128~–1)



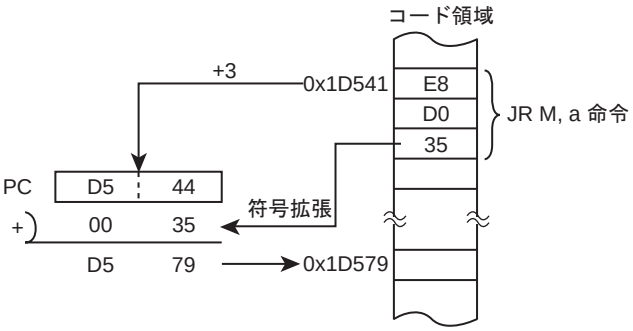
例1:　JR　$ + 2 + 35H または JR　0x0D578



プログラムカウンタの内容 (0xD543) にディスプレースメント値0x35を符号拡張して加算した値で指定されるアドレスすなわち 0x1D578 番地にジャンプします。

注) $: 実行命令の先頭アドレス

例2: JR M, $ + 3 + 35H または JR M, 0x0D579



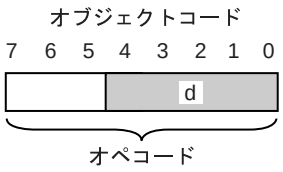
サインフラグが “1” なら、プログラムカウンタの内容 (0xD544) にディスプレースメント値0x35を符号拡張して加算した値で指定されるアドレスすなわち0x1D579番地にジャンプします。

注) $: 実行命令の先頭アドレス

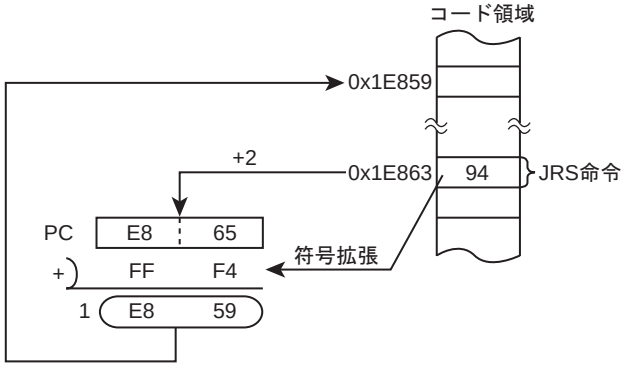
1.4.5.2 5 ビットディスプレースメントリラティブ

プログラムカウンタの内容 (実行命令の先頭アドレス +2) に、オペコード中の 5 ビットディスプレースメント値 d を符号拡張 (下表参照) して、加算した値で指定されるアドレス。このアドレッシングモードを持つ命令は、JRS 命令のみです。

ディスプレースメント d	符号拡張した値
0x00 ~0x0F	0x0000 ~ 0x000F (0~+15)
0x10 ~ 0x1F	0xFFF0 ~ 0xFFFF (–16~–1)



例: JRS T, $ + 2 + 14H または JRS T, 0x0E859

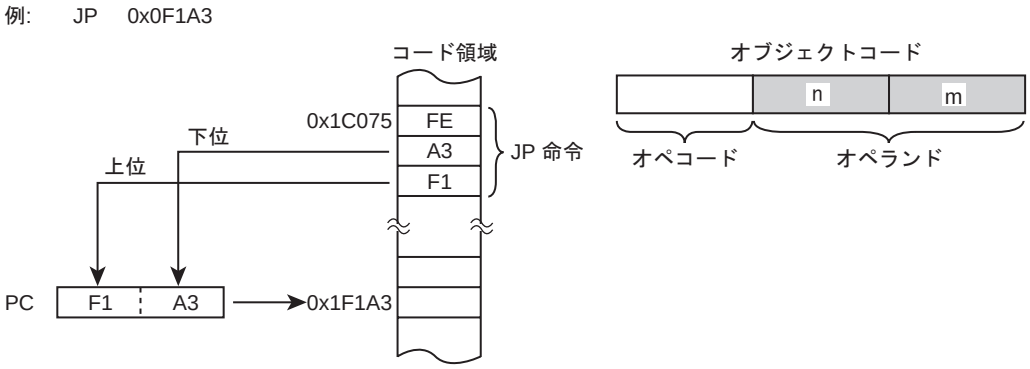


ジャンプステータスフラグが “1” なら、プログラムカウンタの内容 (0xE865) にディスプレースメント値 0x14 を符号拡張して (0xFFF4) 加算した値で指定されるアドレスすなわち 0x1E859 番地にジャンプします。

注) $: 実行命令の先頭アドレス

1.4.6　アブソリュート (絶対)

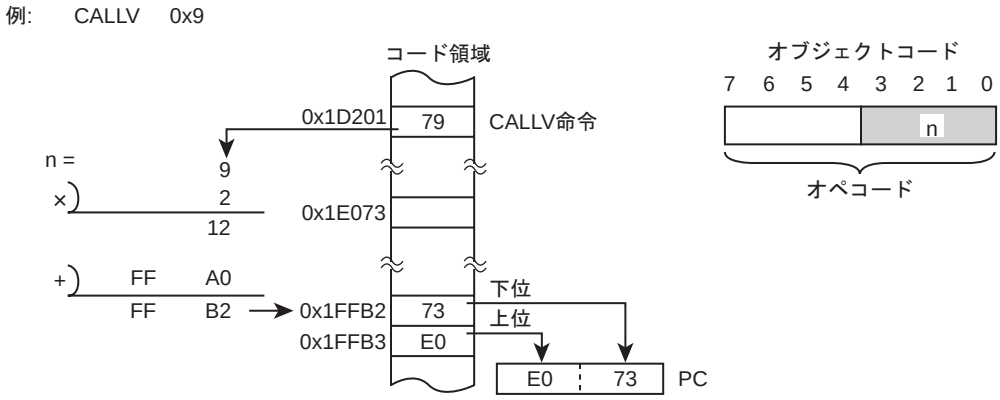
オブジェクトコード中の 16 ビット値 (下位 8 ビット → 上位 8 ビットの順に格納される) で指定されるアドレス。



オペランドで指定されるアドレスすなわち 0x1F1A3 番地にジャンプします。

1.4.7　ベクタ

オペコード中の 4 ビットデータ n を 2 倍し、ベクタコール領域の先頭アドレス値を加えた値をアドレスとするメモリから読み出した 16 ビットデータ (ベクタアドレス) で指定されるアドレス。このアドレッシングモードを持つ命令は、CALLV 命令のみです。



オペコード中の n (0x9) を 2 倍してベクタコール領域の先頭アドレス値 (0xFFA0) を加算した値 0x1FFB2 番地から連続する 2 バイトの内容 0x1E073 番地をコールします。

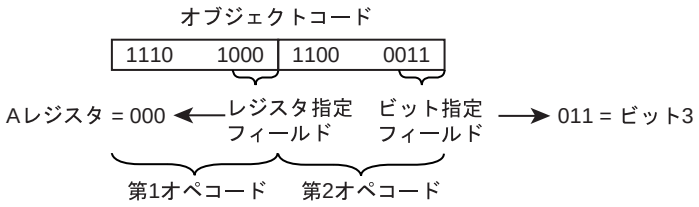
1.4.8　直接ビット

1.4.8.1　レジスタビット

オペコード中のレジスタ指定フィールド、ビット指定フィールドで指定されるレジスタのビットが操作対象となります。

例: SET A.3

Aレジスタのビット3の内容を
"1" にセットします。



1.4.8.2 メモリビット

メモリビットアドレッシングモード (HL), (DE), (IX), (IY), (HL + d), (IX + d), (IY + d), (HL + C), (+SP), (SP + d), (PC + A), (x), (vw) で指定されるアドレスの、オペコード中のビット指定フィールドで指定されるビットが操作対象となります。

例 1: SET (HL). 1

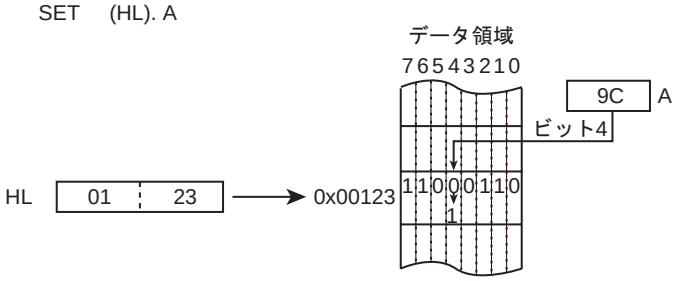
例 2: SET (HL + 0x57). 6

例 3: SET (0x00058). 3

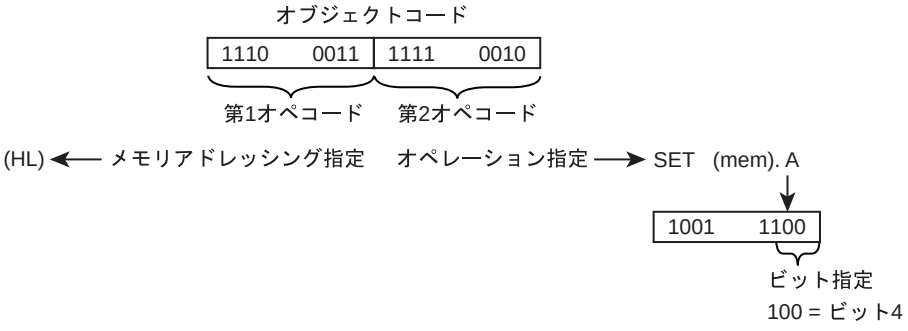
1.4.9 レジスタ間接ビット

メモリアドレッシングモード (HL), (DE), (IX), (IY), (HL + d), (IX + d), (IY + d), (HL + C), (+SP), (SP + d), (PC + A), (x), (vw) で指定されるアドレスの、A レジスタの下位 3 ビットの内容で指定されるビットが操作対象となります。

例: SET (HL). A



HL レジスタの内容 (0x0123) で指定されるアドレスの内容 (0y11000110) の A レジスタの下位 3 ビットの内容 (0y100) で指定されるビット 4 が "1" にセットされ 0y11010110 となります。



第 2 章　命令の説明

2.1　転送、交換

ニモニック	オブジェクトコード (2 進)	フラグ J Z C H S V	サイクル	オペレーション
LD A,r	0 0 0 1　0 r r r	1 Z – – – –	1	A ← r
	8 ビットレジスタ r の内容を A レジスタにロードします。ゼロフラグは、r = 0x00 のとき “1” にセットされ、r ≠ 0x00 のとき “0” にクリアされます。			
LD r,A	0 1 0 0　0 r r r	1 Z – – – –	1	r ← A
	A レジスタの内容を 8 ビットレジスタ r にロードします。ゼロフラグは、A = 0x00 のとき “1” にセットされ、A ≠ 0x00 のとき “0” にクリアされます。			
LD r,g	1 1 1 0　1 g g g　0 1 0 0　0 r r r	1 Z – – – –	2	r ← g
	8 ビットレジスタ g の内容を 8 ビットレジスタ r にロードします。ゼロフラグは、g = 0x00 のとき “1” にセットされ、g ≠ 0x00 のとき “0” にクリアされます。			
LD rr,gg	1 1 1 0　1 g g g　0 1 0 0　1 r r r	1 – – – – –	2	rr ← gg
	16 ビットレジスタ gg (WA, BC, DE または HL) の内容を 16 ビットレジスタ rr にロードします。 例 : DE = 0x1234 のとき、LD HL, DE 命令を実行すると、HL = 0x1234 となります。			
LD A,(x)	0 0 0 0　1 1 0 0　x x x x　x x x x	1 Z – – – –	3	A ← (x)
	オブジェクトコード中の x で直接指定されるアドレス (0x0000 ~ 0x00FF 番地) のメモリ内容を A レジスタにロードします。A レジスタに転送されるデータが 0x00 のとき、ゼロフラグが “1” にセットされます。			
LD A,(HL)	0 0 0 0　1 1 0 1	1 Z – – – –	2	A ← (HL)
	レジスタペア HL で指定されるアドレス (0x0000 ~ 0xFFFF 番地) のメモリ内容を A レジスタにロードします。A レジスタに転送されるデータが 0x00 のとき、ゼロフラグが “1” にセットされます。			
LD r,(x)	1 1 1 0　0 0 0 0　x x x x　x x x x　0 1 0 0　0 r r r	1 Z – – – –	4	r ← (x)
	オブジェクトコード中の x で直接指定されるアドレス (0x0000 ~ 0x00FF 番地) のメモリ内容を 8 ビットレジスタ r にロードします。レジスタ r に転送されるデータが 0x00 のとき、ゼロフラグが “1” にセットされます。			
LD r,(vw)	1 1 1 0　0 0 0 1　w w w w　w w w w　v v v v　v v v v 0 1 0 0　0 r r r	1 Z – – – –	5	r ← (vw)
	オブジェクトコード中の vw で直接指定されるアドレス (0x0000 ~ 0xFFFF 番地) のメモリ内容を 8 ビットレジスタ r にロードします。レジスタ r に転送されるデータが 0x00 のとき、ゼロフラグが “1” にセットされます。			
LD r,(DE)	1 1 1 0　0 0 1 0　0 1 0 0　0 r r r	1 Z – – – –	3	r ← (DE)
	レジスタペア DE で指定されるアドレス (0x0000 ~ 0xFFFF 番地) のメモリ内容を 8 ビットレジスタ r にロードします。レジスタ r に転送されるデータが 0x00 のとき、ゼロフラグが “1” にセットされます。			
LD r,(HL)	1 1 1 0　0 0 1 1　0 1 0 0　0 r r r	1 Z – – – –	3	r ← (HL)
	レジスタペア HL で指定されるアドレス (0x0000 ~ 0xFFFF 番地) のメモリ内容を 8 ビットレジスタ r にロードします。レジスタ r に転送されるデータが 0x00 のとき、ゼロフラグが “1” にセットされます。			
LD r,(IX)	1 1 1 0　0 1 0 0　0 1 0 0　0 r r r	1 Z – – – –	3	r ← (IX)
	インデックスレジスタ IX で指定されるアドレス (0x0000 ~ 0xFFFF 番地) のメモリ内容を 8 ビットレジスタ r にロードします。レジスタ r に転送されるデータが 0x00 のとき、ゼロフラグが “1” にセットされます。			
LD r,(IY)	1 1 1 0　0 1 0 1　0 1 0 0　0 r r r	1 Z – – – –	3	r ← (IY)
	インデックスレジスタ IY で指定されるアドレス (0x0000 ~ 0xFFFF 番地) のメモリ内容を 8 ビットレジスタ r にロードします。レジスタ r に転送されるデータが 0x00 のとき、ゼロフラグが “1” にセットされます。			
LD r,(IX+d)	1 1 0 1　0 1 0 0　d d d d　d d d d　0 1 0 0　0 r r r	1 Z – – – –	5	r ← (IX+d)
	インデックスレジスタ IX の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容を 8 ビットレジスタ r にロードします。レジスタ r に転送されるデータが 0x00 のとき、ゼロフラグが “1” にセットされます。			
LD r,(IY+d)	1 1 0 1　0 1 0 1　d d d d　d d d d　0 1 0 0　0 r r r	1 Z – – – –	5	r ← (IY+d)
	インデックスレジスタ IY の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容を 8 ビットレジスタ r にロードします。レジスタ r に転送されるデータが 0x00 のとき、ゼロフラグが “1” にセットされます。			
LD r,(SP+d)	1 1 0 1　0 1 1 0　d d d d　d d d d　0 1 0 0　0 r r r	1 Z – – – –	5	r ← (SP+d)
	スタックポインタ SP の内容にオブジェクト中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容を 8 ビットレジスタ r にロードします。レジスタ r に転送されるデータが 0x00 のとき、ゼロフラグが “1” にセットされます。			
LD r,(HL+d)	1 1 0 1　0 1 1 1　d d d d　d d d d　0 1 0 0　0 r r r	1 Z – – – –	5	r ← (HL + d)
	レジスタペア HL の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容を 8 ビットレジスタ r にロードします。レジスタ r に転送されるデータが 0x00 のとき、ゼロフラグが “1” にセットされます。			
LD r,(HL+C)	1 1 1 0　0 1 1 1　0 1 0 0　0 r r r	1 Z – – – –	5	r ← (HL+C)
	レジスタペア HL の内容に C レジスタの内容を符号拡張して加算した値で指定されるアドレスのメモリ内容を 8 ビットレジスタ r にロードします。レジスタ r に転送されるデータが 0x00 のとき、ゼロフラグが “1” にセットされます。			
LD r,(+SP)	1 1 1 0　0 1 1 0　0 1 0 0　0 r r r	1 Z – – – –	4	SP ← SP+1:r ← (SP)
	スタックポインタ SP の内容をインクリメントし、その値で指定されるアドレスのメモリ内容を 8 ビットレジスタ r にロードします。レジスタ r に転送されるデータが 0x00 のとき、ゼロフラグが “1” にセットされます。この命令は 8 ビットデータのスタックからのポップ処理に使います。			
LD r,(PC+A) 注	0 1 0 0　1 1 1 1　0 1 0 0　0 r r r	1 Z – – – –	5	r ← (PC+A)
	プログラムカウンタ PC の内容に A レジスタの内容を符号拡張して加算した値で指定されるアドレスのメモリ内容を 8 ビットレジスタ r にロードします。レジスタ r に転送されるデータが 0x00 のとき、ゼロフラグが “1” にセットされます。この命令は、コード変換処理に最適です。			
LD rr,(x)	1 1 1 0　0 0 0 0　x x x x　x x x x　0 1 0 0　1 r r r	1 – – – – –	5	rr ← (x+1, x)
	オブジェクトコード中の x で直接指定されるアドレス (0x0000 ~ 0x00FF 番地) から連続する 2 バイトのメモリ内容を 16 ビットレジスタ rr にロードします。 例 : 0x0072, 0x0073 番地がそれぞれ 0x8E, 0x59 のとき、LD WA, (0x72) 命令を実行すると、W = 0x59, A = 0x8E となります。			
LD rr,(vw)	1 1 1 0　0 0 0 1　w w w w　w w w w　v v v v　v v v v 0 1 0 0　1 r r r	1 – – – – –	6	rr ← (vw+1, vw)
	オブジェクトコード中の vw で直接指定されるアドレス (0x0000 ~ 0xFFFF 番地) から連続する 2 バイトのメモリ内容を 16 ビットレジスタ rr にロードします。vw として 0x0FFF 番地を指定した場合、レジスタの上位バイトには 0x1000 番地のメモリ内容がロードされます。			

ニモニック	オブジェクトコード (2 進)	フラグ J Z C H S V	サイクル	オペレーション
LD rr,(DE)	1 1 1 0　0 0 1 0 | 0 1 0 0　1 r r r	1 − − − − −	4	rr ← (DE+1, DE)
	レジスタペア DE で指定されるアドレスから連続する 2 バイトのメモリ内容を 16 ビットレジスタ rr にロードします。			
LD rr,(HL)	1 1 1 0　0 0 1 1 | 0 1 0 0　1 r r r	1 − − − − −	4	rr ← (HL+1, HL)
	レジスタペア HL で指定されるアドレスから連続する 2 バイトのメモリ内容を 16 ビットレジスタ rr にロードします。			
LD rr,(IX)	1 1 1 0　0 1 0 0 | 0 1 0 0　1 r r r	1 − − − − −	4	rr ← (IX+1, IX)
	インデックスレジスタ IX で指定されるアドレスから連続する 2 バイトのメモリ内容を 16 ビットレジスタ rr にロードします。			
LD rr,(IY)	1 1 1 0　0 1 0 1 | 0 1 0 0　1 r r r	1 − − − − −	4	rr ← (IY+1, IY)
	インデックスレジスタ IY で指定されるアドレスから連続する 2 バイトのメモリ内容を 16 ビットレジスタ rr にロードします。			
LD rr,(IX+d)	1 1 0 1　0 1 0 0 | d d d d　d d d d | 0 1 0 0　1 r r r	1 − − − − −	6	rr ← (IX+d+1, IX+d)
	インデックスレジスタ IX の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容を 16 ビットレジスタ rr にロードします。			
LD rr,(IY+d)	1 1 0 1　0 1 0 1 | d d d d　d d d d | 0 1 0 0　1 r r r	1 − − − − −	6	rr ← (IY+d+1, IY+d)
	インデックスレジスタ IY の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容を 16 ビットレジスタ rr にロードします。			
LD rr,(SP+d)	1 1 0 1　0 1 1 0 | d d d d　d d d d | 0 1 0 0　1 r r r	1 − − − − −	6	rr ← (SP+d+1, SP+d)
	スタックポインタ SP の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容を 16 ビットレジスタ rr にロードします。 例 :SP = 0x51E4 で 0x5216, 0x5217 番地のメモリ内容がそれぞれ 0x9F, 0xC3 のとき、LD WA, (SP + 0x32) 命令を実行すると、A = 0x9F, W = 0xC3 となります。			
LD rr,(HL+d)	1 1 0 1　0 1 1 1 | d d d d　d d d d | 0 1 0 0　1 r r r	1 − − − − −	6	rr ← (HL+d+1, HL+d)
	レジスタペア HL の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容を 16 ビットレジスタ rr にロードします。			
LD rr,(HL+C)	1 1 1 0　0 1 1 1 | 0 1 0 0　1 r r r	1 − − − − −	6	rr ← (HL+C+1, HL+C)
	レジスタペア HL の内容に C レジスタの内容を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容を 16 ビットレジスタ rr にロードします。			
LD rr,(+SP)	1 1 1 0　0 1 1 0 | 0 1 0 0　1 r r r	1 − − − − −	5	SP ← SP+1:rr ← (SP+1, SP)
	スタックポインタ SP の内容をインクリメントし、その値で指定されるアドレスから連続する 2 バイトのメモリ内容を 16 ビットレジスタ rr にロードします。			
LD rr,(PC+A) 注	0 1 0 0　1 1 1 1 | 0 1 0 0　1 r r r	1 − − − − −	6	rr ← (PC+A+1, PC+A)
	プログラムカウンタ PC の内容に A レジスタの内容を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容を 16 ビットレジスタ rr にロードします。			
LD (x),A	0 0 0 0　1 1 1 0 | x x x x　x x x x	1 − − − − −	3	(x) ← A
	オブジェクトコード中の x で直接指定されるアドレス (0x0000 ~ 0x00FF 番地) のメモリに A レジスタの内容をストアします。			
LD (HL),A	0 0 0 0　1 1 1 1	1 − − − − −	2	(HL) ← A
	レジスタペア HL で指定されるアドレス (0x0000 ~ 0xFFFF 番地) のメモリに A レジスタの内容をストアします。			
LD (x),r	1 1 1 1　0 0 0 0 | x x x x　x x x x | 0 1 1 1　1 r r r	1 − − − − −	4	(x) ← r
	オブジェクトコード中の x で直接指定されるアドレス (0x0000 ~ 0x00FF 番地) のメモリに 8 ビットレジスタ r の内容をストアします。			
LD (vw),r	1 1 1 1　0 0 0 1 | w w w w　w w w w | v v v v　v v v v 0 1 1 1　1 r r r	1 − − − − −	5	(vw) ← r
	オブジェクトコード中の vw で直接指定されるアドレス (0x0000 ~ 0xFFFF 番地) のメモリに 8 ビットレジスタ r の内容をストアします。			
LD (DE),r	1 1 1 1　0 0 1 0 | 0 1 1 1　1 r r r	1 − − − − −	3	(DE) ← r
	レジスタペア DE で指定されるアドレス (0x0000 ~ 0xFFFF 番地) のメモリに 8 ビットレジスタ r の内容をストアします。			
LD (HL),r	1 1 1 1　0 0 1 1 | 0 1 1 1　1 r r r	1 − − − − −	3	(HL) ← r
	レジスタペア HL で指定されるアドレス (0x0000 ~ 0xFFFF 番地) のメモリに 8 ビットレジスタ r の内容をストアします。			
LD (IX),r	1 1 1 1　0 1 0 0 | 0 1 1 1　1 r r r	1 − − − − −	3	(IX) ← r
	インデックスレジスタ IX で指定されるアドレス (0x0000 ~ 0xFFFF 番地) のメモリに 8 ビットレジスタ r の内容をストアします。			
LD (IY),r	1 1 1 1　0 1 0 1 | 0 1 1 1　1 r r r	1 − − − − −	3	(IY) ← r
	インデックスレジスタ IY で指定されるアドレス (0x0000 ~ 0xFFFF 番地) のメモリに 8 ビットレジスタ r の内容をストアします。			
LD (IX+d),r	0 1 0 1　0 1 0 0 | d d d d　d d d d | 0 1 1 1　1 r r r	1 − − − − −	4	(IX+d) ← r
	インデックスレジスタ IX の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリに 8 ビットレジスタ r の内容をストアします。			
LD (IY+d),r	0 1 0 1　0 1 0 1 | d d d d　d d d d | 0 1 1 1　1 r r r	1 − − − − −	4	(IY+d) ← r
	インデックスレジスタ IY の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリに 8 ビットレジスタ r の内容をストアします。			
LD (SP+d),r	0 1 0 1　0 1 1 0 | d d d d　d d d d | 0 1 1 1　1 r r r	1 − − − − −	4	(SP+d) ← r
	スタックポインタ SP の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリに 8 ビットレジスタ r の内容をストアします。			
LD (HL+d),r	0 1 0 1　0 1 1 1 | d d d d　d d d d | 0 1 1 1　1 r r r	1 − − − − −	4	(HL+d) ← r
	レジスタペア HL の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリに 8 ビットレジスタ r の内容をストアします。			
LD (HL+C),r	1 1 1 1　0 1 1 1 | 0 1 1 1　1 r r r	1 − − − − −	5	(HL+C) ← r
	レジスタペア HL の内容に C レジスタの内容を符号拡張して加算した値で指定されるアドレスのメモリに 8 ビットレジスタ r の内容をストアします。			
LD (SP−),r	1 1 1 1　0 1 1 0 | 0 1 1 1　1 r r r	1 − − − − −	4	(SP) ← r:SP ← SP−1
	スタックポインタ SP で指定されるアドレスのメモリに 8 ビットレジスタ r の内容をストアします。その後、スタックポインタ SP の内容をデクリメントします。この命令は 8 ビットレジスタのスタックへのプッシュ処理に使います。			
LD (x),rr	1 1 1 1　0 0 0 0 | x x x x　x x x x | 0 1 1 0　1 r r r	1 − − − − −	5	(x+1, x) ← rr
	オブジェクトコード中の x で直接指定されるアドレス (0x0000 ~ 0x00FF 番地) から連続する 2 バイトのメモリに 16 ビットレジスタ rr の内容を下位、上位の順にストアします。			
LD (vw),rr	1 1 1 1　0 0 0 1 | w w w w　w w w w | v v v v　v v v v 0 1 1 0　1 r r r	1 − − − − −	6	(vw+1, vw) ← rr
	オブジェクトコード中の vw で直接指定されるアドレス (0x0000 ~ 0xFFFF 番地) から連続する 2 バイトのメモリに 16 ビットレジスタ rr の内容を下位、上位の順にストアします。			
LD (DE),rr	1 1 1 1　0 0 1 0 | 0 1 1 0　1 r r r	1 − − − − −	4	(DE+1, DE) ← rr
	レジスタペア DE で指定されるアドレスから連続する 2 バイトのメモリに 16 ビットレジスタ rr の内容を下位、上位の順にストアします。			

ニモニック	オブジェクトコード (2 進)	フラグ J Z C H S V	サイクル	オペレーション
LD (HL),rr	1 1 1 1 0 0 1 1 0 1 1 0 1 r r r レジスタペア HL で指定されるアドレスから連続する 2 バイトのメモリに 16 ビットレジスタ rr の内容を下位、上位の順にストアします。	1 − − − − −	4	(HL+1, HL) ← rr
LD (IX),rr	1 1 1 1 0 1 0 0 0 1 1 0 1 r r r インデックスレジスタ IX で指定されるアドレスから連続する 2 バイトのメモリに 16 ビットレジスタ rr の内容を下位、上位の順にストアします。	1 − − − − −	4	(IX+1, IX) ← rr
LD (IY),rr	1 1 1 1 0 1 0 1 0 1 1 0 1 r r r インデックスレジスタ IY で指定されるアドレスから連続する 2 バイトのメモリに 16 ビットレジスタ rr の内容を下位、上位の順にストアします。	1 − − − − −	4	(IY+1, IY) ← rr
LD (IX+d),rr	0 1 0 1 0 1 0 0 d d d d d d d d 0 1 1 0 1 r r r インデックスレジスタ IX の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリに 16 ビットレジスタ rr の内容を下位、上位の順にストアします。	1 − − − − −	5	(IX+d+1, IX+d) ← rr
LD (IY+d),rr	0 1 0 1 0 1 0 1 d d d d d d d d 0 1 1 0 1 r r r インデックスレジスタ IY の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリに 16 ビットレジスタ rr の内容を下位、上位の順にストアします。	1 − − − − −	5	(IY+d+1, IY+d) ← rr
LD (SP+d),rr	0 1 0 1 0 1 1 0 d d d d d d d d 0 1 1 0 1 r r r スタックポインタ SP の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリに 16 ビットレジスタ rr の内容を下位、上位の順にストアします。	1 − − − − −	5	(SP+d+1, SP+d) ← rr
LD (HL+d),rr	0 1 0 1 0 1 1 1 d d d d d d d d 0 1 1 0 1 r r r レジスタペア HL の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリに 16 ビットレジスタ rr の内容を下位、上位の順にストアします。	1 − − − − −	5	(HL+d+1, HL+d) ← rr
LD (HL+C),rr	1 1 1 1 0 1 1 1 0 1 1 0 1 r r r レジスタペア HL の内容に C レジスタの内容を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリに、16 ビットレジスタ rr の内容を下位、上位の順にストアします。	1 − − − − −	6	(HL+C+1, HL+C) ← rr
LD (SP−),rr	1 1 1 1 0 1 1 0 0 1 1 0 1 r r r スタックポインタ SP で指定されるアドレスから連続する 2 バイトのメモリに 16 ビットレジスタ rr の内容をストアします。その後、スタックポインタ SP の内容をデクリメントします。	1 − − − − −	5	(SP+1, SP) ← rr:SP ← SP−1
LD r,n	0 0 0 1 1 r r r n n n n n n n n n オブジェクトコード中の即値 n を 8 ビットレジスタ r にロードします。 例 : LD A, 0x53 命令を実行すると、A = 0x53 となります。	1 − − − − −	2	r ← n
LD rr,mn	0 1 0 0 1 r r r n n n n n n n n n m m m m m m m m m オブジェクトコード中の即値 mn を 16 ビットレジスタ rr にロードします。 例 : LD WA, 0x1234 命令を実行すると、A = 0x34, W = 0x12 となります。	1 − − − − −	3	rr ← mn
LD (x),n	0 0 0 0 1 0 1 0 x x x x x x x x n n n n n n n n オブジェクトコード中の x で直接指定されるアドレス (0x0000 ~ 0x00FF 番地) のメモリにオブジェクトコード中の即値 n をストアします。	1 − − − − −	4	(x) ← n
LD (vw),n	1 1 1 1 0 0 0 1 w w w w w w w w v v v v v v v v 1 1 1 1 1 0 0 1 n n n n n n n n オブジェクトコード中の vw で直接指定されるアドレス (0x0000 ~ 0xFFFF 番地) のメモリにオブジェクトコード中の即値 n をストアします。	1 − − − − −	6	(vw) ← n
LD (DE),n	1 1 1 1 0 0 1 0 1 1 1 1 1 0 0 1 n n n n n n n n オブジェクトコード中の即値 n をレジスタペア DE で指定されるアドレスのメモリにストアします。	1 − − − − −	4	(DE) ← n
LD (HL),n	0 0 0 0 1 0 1 1 n n n n n n n n オブジェクトコード中の即値 n をレジスタペア HL で指定されるアドレスのメモリにストアします。	1 − − − − −	3	(HL) ← n
LD (IX),n	1 1 1 1 0 1 0 0 1 1 1 1 1 0 0 1 n n n n n n n n インデックスレジスタ IX で指定されるアドレスのメモリにオブジェクトコード中の即値 n をストアします。	1 − − − − −	4	(IX) ← n
LD (IY),n	1 1 1 1 0 1 0 1 1 1 1 1 1 0 0 1 n n n n n n n n インデックスレジスタ IY で指定されるアドレスのメモリにオブジェクトコード中の即値 n をストアします。	1 − − − − −	4	(IY) ← n
LD (IX+d),n	0 1 0 1 0 1 0 0 d d d d d d d d 1 1 1 1 1 0 0 1 n n n n n n n n インデックスレジスタ IX の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリにオブジェクトコード中の即値 n をストアします。	1 − − − − −	5	(IX+d) ← n
LD (IY+d),n	0 1 0 1 0 1 0 1 d d d d d d d d 1 1 1 1 1 0 0 1 n n n n n n n n インデックスレジスタ IY の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリにオブジェクトコード中の即値 n をストアします。	1 − − − − −	5	(IY+d) ← n
LD (SP+d),n	0 1 0 1 0 1 1 0 d d d d d d d d 1 1 1 1 1 0 0 1 n n n n n n n n スタックポインタ SP の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリに、オブジェクトコード中の即値 n をストアします。	1 − − − − −	5	(SP+d) ← n
LD (HL+d),n	0 1 0 1 0 1 1 1 d d d d d d d d 1 1 1 1 1 0 0 1 n n n n n n n n レジスタペア HL の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリに、オブジェクトコード中の即値 n をストアします。	1 − − − − −	5	(HL+d) ← n
LD (HL+C),n	1 1 1 1 0 1 1 1 1 1 1 1 1 0 0 1 n n n n n n n n レジスタペア HL の内容に C レジスタの内容を符号拡張して加算した値で指定されるアドレスのメモリに、オブジェクトコード中の即値 n をストアします。	1 − − − − −	6	(HL+C) ← n
LD (SP−),n	1 1 1 1 0 1 1 0 1 1 1 1 1 0 0 1 n n n n n n n n オブジェクトコード中の即値 n をスタックポインタ SP で指定されるアドレスのメモリにストアします。その後、スタックポインタ SP の内容をデクリメントします。	1 − − − − −	5	(SP) ← n:SP ← SP−1
LDW (x),mn	0 0 0 0 1 0 0 0 x x x x x x x x n n n n n n n n m m m m m m m m オブジェクトコード中の 16 ビットの即値 mn を、オブジェクトコード中の x で直接指定されるアドレス (0x0000 ~ 0x00FF 番地) から連続する 2 バイトのメモリに下位 n、上位 m の順にストアします。 例 : LDW (0x73),0x1234 を実行すると、0x0073 番地には、0x34 が、0x0074 番地には、0x12 が書き込まれます。	1 − − − − −	6	(x+1, x) ← mn
LDW (HL),mn	0 0 0 0 1 0 0 1 n n n n n n n n m m m m m m m m オブジェクトコード中の 16 ビットの即値 mn を、レジスタペア HL で指定されるアドレスから連続する 2 バイトのメモリに下位、上位の順にストアします。	1 − − − − −	5	(HL+1, HL) ← mn

ニモニック	オブジェクトコード (2 進)	フラグ J Z C H S V	サイクル	オペレーション
PUSH rr#1	0 1 0 1　0 0 r r	– – – – – –	3	(SP, SP–1) ← rr:SP ← SP–2
	スタックポインタ SP で指定されるアドレスのメモリにレジスタ rr の上位 8 ビットの内容を、前記アドレスから 1 を引いた値で指定されるアドレスのメモリにレジスタ rr の下位 8 ビットの内容を、それぞれストアします。その後、スタックポインタ SP の内容から 2 を減じます。rr としては、WA, BC, DE, HL のみ指定可能です。 例 : SP = 0x013F、WA = 0x1234 のとき、PUSH WA 命令を実行すると、0x013F, 0x013E 番地の内容は 0x12, 0x34 となります。また、SP = 0x013D になります。			
PUSH gg#2	1 1 1 0　1 g g g　1 1 0 1　1 0 0 0	– – – – – –	4	(SP, SP–1) ← gg:SP ← SP–2
	スタックポインタ SP で指定されるアドレスのメモリにレジスタ gg の上位 8 ビットの内容を、前記アドレスから 1 を引いた値で指定されるアドレスのメモリにレジスタ gg の下位 8 ビットの内容を、それぞれストアします。その後、スタックポインタ SP の内容から 2 を減じます。 例 : SP = 0x013F, IX = 0x1234 のとき、PUSH IX 命令を実行すると、0x013F, 0x013E 番地の内容は 0x12, 0x34 となります。また、SP = 0x013D になります。			
POP rr#1	1 1 0 1　0 0 r r	– – – – – –	4	SP ← SP+2:rr ← (SP, SP–1)
	スタックポインタ SP の内容に 2 を加え、その値で指定されるアドレスのメモリ内容をレジスタ rr の上位 8 ビットに、前記アドレスから 1 を引いたアドレスのメモリ内容をレジスタ rr の下位 8 ビットをそれぞれロードします。rr としては、WA, BC, DE, HL のみ指定可能です。			
POP gg#2	1 1 1 0　1 g g g　1 1 0 1　1 0 0 1	– – – – – –	5	SP ← SP+2:gg ← (SP, SP–1)
	スタックポインタ SP の内容に 2 を加え、その値で指定されるアドレスのメモリ内容をレジスタ gg の上位 8 ビットに、前記アドレスから 1 を引いたアドレスのメモリ内容をレジスタ gg の下位 8 ビットをそれぞれロードします。			
PUSH PSW	1 1 1 0　1 0 0 0　1 1 0 1　1 1 0 0	– – – – – –	3	(SP) ← PSW:SP ← SP–1
	スタックポインタ SP で指定されるアドレスのメモリにプログラムステータスワード PSW の内容をストアし、その後、スタックポインタ SP の内容をデクリメントします。 例 : SP = 0x2345, PSW = 0x62 (JF = HF = SF = VF = 0, ZF = CF = RBS = 1) のとき、PUSH PSW 命令を実行すると、0x2345 番地の内容は 0x62 となります。また、SP = 0x2344 になります。			
POP PSW	1 1 1 0　1 0 0 0　1 1 0 1　1 1 0 1	* * * * * *	4	SP ← SP+1:PSW ← (SP)
	スタックポインタ SP の内容をインクリメントし、その値で指定されるアドレスのメモリ内容をプログラムステータスワード PSW にロードします。 例 : 0x0137 番地のメモリ内容が 0x63 で、SP = 0x0136 のとき、POP PSW 命令を実行すると、SP = 0x0137, PSW = 0x62 (JF = HF = SF = VF = 0, ZF = CF = RBS = 1) となります。			
LD PSW,n	1 1 1 0　1 0 0 0　1 1 0 1　1 1 1 0　n n n n　n n n n	* * * * * *	3	PSW ← n.7-1
	プログラムステータスワード PSW に、オブジェクトコード中の即値 n をストアします。即値 n は最上位ビットから JF, ZF, CF, HF, SF, VF,RBS に対応し、最下位の 1 ビットは無視されます。 例 : LD FLAG, 0y00110100 命令を実行すると、JF = 0, ZF = 0, CF = 1, HF = 1, SF = 0, VF = 1, RBS = 0 となります。			
LD RBS,0	1 1 1 1　1 0 0 1　0 0 0 0　0 0 0 0	– – – – – –	2	RBS ← 0
	プログラムステータスワード PSW の RBS に対して、即値 0 をストアすることでレジスタバンクが BANK0 に切り替わります。なお、この命令で、フラグは変化しません。			
LD RBS,1	1 1 1 1　1 0 0 1　0 0 0 0　0 0 1 0	– – – – – –	2	RBS ← 1
	プログラムステータスワード PSW の RBS に対して、即値 1 をストアすることでレジスタバンクが BANK1 に切り替わります。なお、この命令で、フラグは変化しません。			
LD SP,SP+d	0 0 1 1　0 1 1 1　d d d d　d d d d	1 – – – – –	2	SP ← SP+d
	スタックポインタ SP の内容にオブジェクトコード中の 8 ビットデータ d を加算した値がスタックポインタ SP にロードされます。従って、この命令はスタックポインタ SP の即値加算命令になります。 例 : SP = 0x2345 のとき、LD SP, SP + 4 を実行すると、SP = 0x2349, JF = 1 となります。			
LD SP,SP–d	0 0 1 1　1 1 1 1　d d d d　d d d d	1 – – – – –	2	SP ← SP–d
	スタックポインタ SP の内容にオブジェクトコード中の 8 ビットデータ d を減算した値がスタックポインタ SP にロードされます。従って、この命令はスタックポインタ SP の即値減算命令になります。			
XCH r,g	1 1 1 0　1 g g g　0 1 1 1　0 r r r	1 Z – – – –	3	r ↔ g
	8 ビットレジスタ r の内容と 8 ビットレジスタ g の内容とを交換します。ゼロフラグは、交換前の g の内容が 0x00 のとき “1” にセットされ、0x00 以外のとき “1” にクリアされます。 例 : A = 0x3C, B = 0x5F のとき、XCH A, B 命令を実行すると、A = 0x5F, B = 0x3C, ZF = 0 となります。			
XCH rr,gg	1 1 1 0　1 g g g　0 1 1 1　1 r r r	1 – – – – –	3	rr ↔ gg
	16 ビットレジスタ rr の内容と 16 ビットレジスタ gg の内容とを交換します。ゼロフラグは変化しません。 例 : HL = 0x0123, DE = 0x9587 のとき、XCH HL, DE 命令を実行すると、HL = 0x9587, DE = 0x0123 となります。			
XCH r,(x)	1 1 1 0　0 0 0 0　x x x x　x x x x　0 1 1 1　0 r r r	1 Z – – – –	5	r ↔ (x)
	オブジェクトコード中の x で直接指定されるアドレス (0x0000 ~ 0x00FF 番地) のメモリ内容と 8 ビットレジスタ r の内容とを交換します。レジスタ r に転送されるデータが 0x00 のとき、ゼロフラグが “1” にセットされます。			
XCH r,(vw)	1 1 1 0　0 0 0 1　w w w w　w w w w　v v v v　v v v v 0 1 1 1　0 r r r	1 Z – – – –	6	r ↔ (vw)
	オブジェクトコード中の vw で直接指定されるアドレス (0x0000 ~ 0xFFFF 番地) のメモリ内容と 8 ビットレジスタ r の内容を交換します。レジスタ r に転送されるデータが 0x00 のとき、ゼロフラグが “1” にセットされます。			
XCH r,(DE)	1 1 1 0　0 0 1 0　0 1 1 1　0 r r r	1 Z – – – –	4	r ↔ (DE)
	レジスタペア DE で指定されるアドレスのメモリ内容と 8 ビットレジスタ r の内容とを交換します。レジスタ r に転送されるデータが 0x00 のとき、ゼロフラグが “1” にセットされます。			
XCH r,(HL)	1 1 1 0　0 0 1 1　0 1 1 1　0 r r r	1 Z – – – –	4	r ↔ (HL)
	レジスタペア HL で指定されるアドレスのメモリ内容と 8 ビットレジスタ r の内容とを交換します。レジスタ r に転送されるデータが 0x00 のとき、ゼロフラグが “1” にセットされます。			
XCH r,(IX)	1 1 1 0　0 1 0 0　0 1 1 1　0 r r r	1 Z – – – –	4	r ↔ (IX)
	インデックスレジスタ IX で指定されるアドレスのメモリ内容と 8 ビットレジスタ r の内容を交換します。レジスタ r に転送されるデータが 0x00 のとき、ゼロフラグが “1” にセットされます。			
XCH r,(IY)	1 1 1 0　0 1 0 1　0 1 1 1　0 r r r	1 Z – – – –	4	r ↔ (IY)
	インデックスレジスタ IY で指定されるアドレスのメモリ内容と 8 ビットレジスタ r の内容を交換します。レジスタ r に転送されるデータが 0x00 のとき、ゼロフラグが “1” にセットされます。			
XCH r,(IX+d)	1 1 0 1　0 1 0 0　d d d d　d d d d　0 1 1 1　0 r r r	1 Z – – – –	6	r ↔ (IX+d)
	インデックスレジスタ IX の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容と 8 ビットレジスタ r の内容とを交換します。レジスタ r に転送されるデータが 0x00 のとき、ゼロフラグが “1” にセットされます。			

ニモニック	オブジェクトコード (2 進)	フラグ J Z C H S V	サイクル	オペレーション
XCH r,(IY+d)	1 1 0 1 0 1 0 1 d d d d d d d d 0 1 1 1 0 r r r	1 Z – – – –	6	r ↔ (IY+d)
	インデックスレジスタ IY の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容と 8 ビットレジスタ r の内容とを交換します。レジスタ r に転送されるデータが 0x00 のとき、ゼロフラグが “1” にセットされます。			
XCH r,(SP+d)	1 1 0 1 0 1 1 0 d d d d d d d d 0 1 1 1 0 r r r	1 Z – – – –	6	r ↔ (SP+d)
	スタックポインタ SP の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容と 8 ビットレジスタ r の内容とを交換します。レジスタ r に転送されるデータが 0x00 のとき、ゼロフラグが “1” にセットされます。			
XCH r,(HL+d)	1 1 0 1 0 1 1 1 d d d d d d d d 0 1 1 1 0 r r r	1 Z – – – –	6	r ↔ (HL+d)
	レジスタペア HL の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容と 8 ビットレジスタ r の内容とを交換します。レジスタ r に転送されるデータが 0x00 のとき、ゼロフラグが “1” にセットされます。			
XCH r,(HL+C)	1 1 1 0 0 1 1 1 0 1 1 1 0 r r r	1 Z – – – –	6	r ↔ (HL+C)
	レジスタペア HL の内容に C レジスタの内容を符号拡張して加算した値で指定されるアドレスのメモリ内容と 8 ビットレジスタ r の内容とを交換します。レジスタ r に転送されるデータが 0x00 のとき、ゼロフラグが “1” にセットされます。			
XCH r, (+SP)	1 1 1 0 0 1 1 0 0 1 1 1 0 r r r	1 Z – – – –	5	SP ← SP+1:r ↔ (SP)
	スタックポインタ SP の内容をインクリメントし、その値で指定されるアドレスのメモリ内容と 8 ビットレジスタ r の内容とを交換します。レジスタ r に転送されるデータが 0x00 のとき、ゼロフラグが “1” にセットされます。			
XCH r,(PC+A) 注	0 1 0 0 1 1 1 1 0 1 1 1 0 r r r	1 Z – – – –	6	r ↔ (PC+A)
	プログラムカウンタ PC の内容に A レジスタの内容を符号拡張して加算した値で指定されるアドレスのメモリ内容と 8 ビットレジスタ r の内容とを交換します。レジスタ r に転送されるデータが 0x00 のとき、ゼロフラグが “1” にセットされます。			
XCH rr,(x)	1 1 1 0 0 0 0 0 x x x x x x x x 1 1 0 1 1 r r r	1 – – – – –	7	rr ↔ (x+1, x)
	オブジェクトコード中の x で直接指定されるアドレス (0x0000 ~ 0x00FF 番地) から連続する 2 バイトのメモリ内容と 16 ビットレジスタ rr の内容とを交換します。			
XCH rr,(vw)	1 1 1 0 0 0 0 1 w w w w w w w w v v v v v v v v 1 1 0 1 1 r r r	1 – – – – –	8	rr ↔ (vw+1, vw)
	オブジェクトコード中の vw で直接指定されるアドレス (0x0000 ~ 0xFFFF 番地) から連続する 2 バイトのメモリ内容と 16 ビットレジスタ rr の内容とを交換します。			
XCH rr,(DE)	1 1 1 0 0 0 1 0 1 1 0 1 1 r r r	1 – – – – –	6	rr ↔ (DE+1, DE)
	レジスタペア DE で指定されるアドレスから連続する 2 バイトのメモリ内容と 16 ビットレジスタ rr の内容とを交換します。			
XCH rr,(HL)	1 1 1 0 0 0 1 1 1 1 0 1 1 r r r	1 – – – – –	6	rr ↔ (HL+1, HL)
	レジスタペア HL で指定されるアドレスから連続する 2 バイトのメモリ内容と 16 ビットレジスタ rr の内容とを交換します。			
XCH rr,(IX)	1 1 1 0 0 1 0 0 1 1 0 1 1 r r r	1 – – – – –	6	rr ↔ (IX+1, IX)
	インデックスレジスタ IX で指定されるアドレスから連続する 2 バイトのメモリ内容と 16 ビットレジスタ rr の内容とを交換します。			
XCH rr,(IY)	1 1 1 0 0 1 0 1 1 1 0 1 1 r r r	1 – – – – –	6	rr ↔ (IY+1, IY)
	インデックスレジスタ IY で指定されるアドレスから連続する 2 バイトのメモリ内容と 16 ビットレジスタ rr の内容とを交換します。			
XCH rr,(IX+d)	1 1 0 1 0 1 0 0 d d d d d d d d 1 1 0 1 1 r r r	1 – – – – –	8	rr ↔ (IX+d+1, IX+d)
	インデックスレジスタ IX の内容に、オブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容と 16 ビットレジスタ rr の内容とを交換します。			
XCH rr,(IY+d)	1 1 0 1 0 1 0 1 d d d d d d d d 1 1 0 1 1 r r r	1 – – – – –	8	rr ↔ (IY+d+1, IY+d)
	インデックスレジスタ IY の内容に、オブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容と 16 ビットレジスタ rr の内容とを交換します。			
XCH rr,(SP+d)	1 1 0 1 0 1 1 0 d d d d d d d d 1 1 0 1 1 r r r	1 – – – – –	8	rr ↔ (SP+d+1, SP+d)
	スタックポインタ SP の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容と 16 ビットレジスタ rr の内容とを交換します。			
XCH rr,(HL+d)	1 1 0 1 0 1 1 1 d d d d d d d d 1 1 0 1 1 r r r	1 – – – – –	8	rr ↔ (HL+d+1, HL+d)
	レジスタペア HL の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容と 16 ビットレジスタ rr の内容とを交換します。			
XCH rr,(HL+C)	1 1 1 0 0 1 1 1 1 1 0 1 1 r r r	1 – – – – –	8	rr ↔ (HL+C+1, HL+C)
	レジスタペア HL の内容に C レジスタの内容を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容と 16 ビットレジスタ rr の内容とを交換します。			
XCH rr,(+SP)	1 1 1 0 0 1 1 0 1 1 0 1 1 r r r	1 – – – – –	7	SP ← SP+1:rr ↔ (SP+1, SP)
	スタックポインタ SP の内容をインクリメントし、その値で指定されるアドレスから連続する 2 バイトのメモリ内容と 16 ビットレジスタ rr の内容とを交換します。			
XCH rr,(PC+A) 注	0 1 0 0 1 1 1 1 1 1 0 1 1 r r r	1 – – – – –	8	rr ↔ (PC+A+1, PC+A)
	プログラムカウンタ PC の内容に A レジスタの内容を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容と 16 ビットレジスタ rr の内容とを交換します。			

#1 rr は WA, BC, DE, HL のみ

#2 gg は IX,IY のみ

注) (PC+A) を使用した命令には制限があります。詳細は、TLCS-870/C1 シリーズ 命令セット 1.4 アドレッシングモードを参照してください。

2.2 演算

ニモニック	オブジェクトコード (2 進)	フラグ J Z C H S V	サイクル	オペレーション
CMP A,n	0 1 1 0 0 1 1 1 n n n n n n n n	Z Z C H S V	2	A–n
	A レジスタの内容をオブジェクトコード中の即値 n と比較します。キャリーフラグは、A < n のとき “1” にセットされ A ≧ n のとき “0” にクリアされます。			
CMP g,n	1 1 1 0 1 g g g 0 1 1 0 0 1 1 1 n n n n n n n n	Z Z C H S V	3	g–n
	8 ビットレジスタ g の内容をオブジェクトコード中の即値 n と比較します。キャリーフラグは、g < n のとき “1” にセットされ、g ≧ n のとき “0” にクリアされます。			
CMP gg,mn	1 1 1 0 1 g g g m m m m 0 1 1 0 m m m m 1 1 1 1 n n n n n n n n	Z Z C U S V	4	gg–mn
	16 ビットレジスタ gg の内容をオブジェクトコード中の即値 mn と比較します。キャリーフラグは、gg < mn のときに “1” にセットされ、gg ≧ mn のとき “0” にクリアされます。			
CMP r,g	1 1 1 0 1 g g g 0 0 r r r 1 1 1	Z Z C H S V	2	r–g
	8 ビットレジスタ r の内容と、8 ビットレジスタ g の内容を比較します。キャリーフラグは r < g のとき “1” にセットされ、r ≧ g のとき “0” にクリアされます。			
CMP rr,gg	1 1 1 0 1 g g g 1 0 r r r 1 1 1	Z Z C U S V	3	rr–gg
	16 ビットレジスタ rr の内容と、16 ビットレジスタ gg の内容を比較します。			
CMP r,(x)	1 1 1 0 0 0 0 0 x x x x x x x x 0 0 r r r 1 1 1	Z Z C H S V	4	r–(x)
	オブジェクトコード中の x で直接指定されるアドレス (0x0000 ~ 0x00FF 番地) のメモリ内容と 8 ビットレジスタ r の内容を比較します。			
CMP r,(vw)	1 1 1 0 0 0 0 1 w w w w w w w w v v v v v v v v 0 0 r r r 1 1 1	Z Z C H S V	5	r–(vw)
	オブジェクトコード中の vw で直接指定されるアドレス (0x0000 ~ 0xFFFF 番地) のメモリ内容と 8 ビットレジスタ r の内容を比較します。			
CMP r,(DE)	1 1 1 0 0 0 1 0 0 0 r r r 1 1 1	Z Z C H S V	3	r–(DE)
	レジスタペア DE で指定されるアドレスのメモリ内容と 8 ビットレジスタ r の内容を比較します。			
CMP r,(HL)	1 1 1 0 0 0 1 1 0 0 r r r 1 1 1	Z Z C H S V	3	r–(HL)
	レジスタペア HL で指定されるアドレスのメモリ内容と 8 ビットレジスタ r の内容を比較します。			
CMP r,(IX)	1 1 1 0 0 1 0 0 0 0 r r r 1 1 1	Z Z C H S V	3	r–(IX)
	インデックスレジスタ IX で指定されるアドレスのメモリ内容と 8 ビットレジスタ r の内容を比較します。			
CMP r,(IY)	1 1 1 0 0 1 0 1 0 0 r r r 1 1 1	Z Z C H S V	3	r–(IY)
	インデックスレジスタ IY で指定されるアドレスのメモリ内容と 8 ビットレジスタ r の内容を比較します。			
CMP r,(IX+d)	1 1 0 1 0 1 0 0 d d d d d d d d 0 0 r r r 1 1 1	Z Z C H S V	5	r–(IX+d)
	インデックスレジスタ IX の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容と 8 ビットレジスタ r の内容とを比較します。			
CMP r,(IY+d)	1 1 0 1 0 1 0 1 d d d d d d d d 0 0 r r r 1 1 1	Z Z C H S V	5	r–(IY+d)
	インデックスレジスタ IY の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容と 8 ビットレジスタ r の内容とを比較します。			
CMP r,(SP+d)	1 1 0 1 0 1 1 0 d d d d d d d d 0 0 r r r 1 1 1	Z Z C H S V	5	r–(SP+d)
	スタックポインタ SP の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容と 8 ビットレジスタ r の内容とを比較します。			
CMP r,(HL+d)	1 1 0 1 0 1 1 1 d d d d d d d d 0 0 r r r 1 1 1	Z Z C H S V	5	r–(HL+d)
	レジスタペア HL の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容と 8 ビットレジスタ r の内容とを比較します。			
CMP r,(HL+C)	1 1 1 0 0 1 1 1 0 0 r r r 1 1 1	Z Z C H S V	5	r–(HL+C)
	レジスタペア HL の内容に C レジスタの内容を符号拡張して加算した値で指定されるアドレスのメモリ内容と 8 ビットレジスタ r の内容とを比較します。			
CMP r,(+SP)	1 1 1 0 0 1 1 0 0 0 r r r 1 1 1	Z Z C H S V	4	SP ← SP+1:r–(SP)
	スタックポインタ SP の内容をインクリメントし、その値で指定されるアドレスのメモリ内容と 8 ビットレジスタ r の内容を比較します。			
CMP r,(PC+A) 注	0 1 0 0 1 1 1 1 0 0 r r r 1 1 1	Z Z C H S V	5	r–(PC+A)
	プログラムカウンタ PC の内容に A レジスタの内容を符号拡張して加算した値で指定されるアドレスのメモリ内容と 8 ビットレジスタ r の内容を比較します。			
CMP (x),n	0 0 0 0 0 1 1 1 x x x x x x x x n n n n n n n n	Z Z C H S V	4	(x)–n
	オブジェクトコード中の x で直接指定されるアドレス (0x0000 ~ 0x00FF 番地) のメモリ内容をオブジェクトコード中の即値 n と比較します。			
CMP (vw),n	1 1 1 0 0 0 0 1 w w w w w w w w v v v v v v v v 0 1 1 0 0 1 1 1 n n n n n n n n	Z Z C H S V	6	(vw)–n
	オブジェクトコード中の vw で直接指定されるアドレス (0x0000 ~ 0xFFFF 番地) のメモリ内容をオブジェクトコード中の即値 n と比較します。			
CMP (DE),n	1 1 1 0 0 0 1 0 0 1 1 0 0 1 1 1 n n n n n n n n	Z Z C H S V	4	(DE)–n
	レジスタペア DE で指定されるアドレスのメモリ内容をオブジェクトコード中の即値 n と比較します。			
CMP (HL),n	1 1 1 0 0 0 1 1 0 1 1 0 0 1 1 1 n n n n n n n n	Z Z C H S V	4	(HL)–n
	レジスタペア HL で指定されるアドレスのメモリ内容をオブジェクトコード中の即値 n と比較します。			
CMP (IX),n	1 1 1 0 0 1 0 0 0 1 1 0 0 1 1 1 n n n n n n n n	Z Z C H S V	4	(IX)–n
	インデックスレジスタ IX で指定されるアドレスのメモリ内容をオブジェクトコード中の即値 n と比較します。			
CMP (IY),n	1 1 1 0 0 1 0 1 0 1 1 0 0 1 1 1 n n n n n n n n	Z Z C H S V	4	(IY)–n
	インデックスレジスタ IY で指定されるアドレスのメモリ内容をオブジェクトコード中の即値 n と比較します。			
CMP (IX+d),n	1 1 0 1 0 1 0 0 d d d d d d d d 0 1 1 0 0 1 1 1 n n n n n n n n	Z Z C H S V	6	(IX+d)–n
	インデックスレジスタ IX の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容をオブジェクトコード中の即値 n と比較します。			

ニモニック	オブジェクトコード (2 進)			フラグ J Z C H S V	サイクル	オペレーション
CMP (IY+d),n	1 1 0 1 0 1 0 1 n n n n n n n n	d d d d d d d d	0 1 1 0 0 1 1 1	Z Z C H S V	6	(IY+d)–n
	インデックスレジスタ IY の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容をオブジェクトコード中の即値 n と比較します。					
CMP (SP+d),n	1 1 0 1 0 1 1 0 n n n n n n n n	d d d d d d d d	0 1 1 0 0 1 1 1	Z Z C H S V	6	(SP+d)–n
	スタックポインタ SP の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容を、オブジェクトコード中の即値 n と比較します。					
CMP (HL+d),n	1 1 0 1 0 1 1 1 n n n n n n n n	d d d d d d d d	0 1 1 0 0 1 1 1	Z Z C H S V	6	(HL+d)–n
	レジスタペア HL の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で設定されるアドレスのメモリ内容をオブジェクトコード中の即値 n と比較します。					
CMP (HL+C),n	1 1 1 0 0 1 1 1	0 1 1 0 0 1 1 1	n n n n n n n n	Z Z C H S V	6	(HL+C)–n
	レジスタペア HL の内容に C レジスタの内容を符号拡張して加算した値で指定されるアドレスのメモリ内容をオブジェクトコード中の即値 n と比較します。					
CMP (+SP),n	1 1 1 0 0 1 1 0	0 1 1 0 0 1 1 1	n n n n n n n n	Z Z C H S V	5	SP ← SP+1:(SP)–n
	スタックポインタ SP の内容をインクリメントし、その値で指定されるアドレスのメモリ内容をオブジェクトコード中の即値 n と比較します。					
CMP (PC+A),n [注]	0 1 0 0 1 1 1 1	0 1 1 0 0 1 1 1	n n n n n n n n	Z Z C H S V	6	(PC+A)–n
	プログラムカウンタ PC の内容に A レジスタの内容を符号拡張して加算した値で指定されるアドレスのメモリ内容をオブジェクトコード中の即値 n と比較します。					
CMP rr,(x)	1 1 1 0 0 0 0 0	x x x x x x x x	1 0 r r r 1 1 1	Z Z C U S V	5	rr–(x)
	オブジェクトコード中の x で直接指定されるアドレス (0x0000 ~ 0x00FF 番地) から連続する 2 バイトのメモリ内容を 16 ビットレジスタ rr の内容と比較します。					
CMP rr,(vw)	1 1 1 0 0 0 0 1 1 0 r r r 1 1 1	w w w w w w w w	v v v v v v v v	Z Z C U S V	6	rr–(vw)
	オブジェクトコード中の vw で直接指定されるアドレス (0x0000 ~ 0xFFFF 番地) から連続する 2 バイトのメモリ内容を 16 ビットレジスタ rr の内容と比較します。					
CMP rr,(DE)	1 1 1 0 0 0 1 0	1 0 r r r 1 1 1		Z Z C U S V	4	rr–(DE)
	レジスタペア DE で指定されるアドレスから連続する 2 バイトのメモリ内容を 16 ビットレジスタ rr の内容と比較します。					
CMP rr,(HL)	1 1 1 0 0 0 1 1	1 0 r r r 1 1 1		Z Z C U S V	4	rr–(HL)
	レジスタペア HL で指定されるアドレスから連続する 2 バイトのメモリ内容を 16 ビットレジスタ rr の内容と比較します。					
CMP rr,(IX)	1 1 1 0 0 1 0 0	1 0 r r r 1 1 1		Z Z C U S V	4	rr–(IX)
	インデックスレジスタ IX で指定されるアドレスから連続する 2 バイトのメモリ内容を 16 ビットレジスタ rr の内容と比較します。					
CMP rr,(IY)	1 1 1 0 0 1 0 1	1 0 r r r 1 1 1		Z Z C U S V	4	rr–(IY)
	インデックスレジスタ IY で指定されるアドレスから連続する 2 バイトのメモリ内容を 16 ビットレジスタ rr の内容と比較します。					
CMP rr,(IX+d)	1 1 0 1 0 1 0 0	d d d d d d d d	1 0 r r r 1 1 1	Z Z C U S V	6	rr–(IX+d)
	インデックスレジスタ IX にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容を 16 ビットレジスタ rr の内容と比較します。					
CMP rr,(IY+d)	1 1 0 1 0 1 0 1	d d d d d d d d	1 0 r r r 1 1 1	Z Z C U S V	6	rr–(IY+d)
	インデックスレジスタ IY にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容を 16 ビットレジスタ rr の内容と比較します。					
CMP rr,(SP+d)	1 1 0 1 0 1 1 0	d d d d d d d d	1 0 r r r 1 1 1	Z Z C U S V	6	rr–(SP+d)
	スタックポインタ SP の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容を 16 ビットレジスタ rr の内容と比較します。					
CMP rr,(HL+d)	1 1 0 1 0 1 1 1	d d d d d d d d	1 0 r r r 1 1 1	Z Z C U S V	6	rr–(HL+d)
	レジスタペア HL の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容を 16 ビットレジスタ rr の内容と比較します。					
CMP rr,(HL+C)	1 1 1 0 0 1 1 1	1 0 r r r 1 1 1		Z Z C U S V	6	rr–(HL+C)
	レジスタペア HL の内容に C レジスタの内容を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容を 16 ビットレジスタ rr の内容と比較します。					
CMP rr,(+SP)	1 1 1 0 0 1 1 0	1 0 r r r 1 1 1		Z Z C U S V	5	SP ← SP+1:rr–(SP)
	スタックポインタ SP の内容をインクリメントし、その値で指定されるアドレスから連続する 2 バイトのメモリ内容を 16 ビットレジスタ rr の内容と比較します。					
CMP rr,(PC+A) [注]	0 1 0 0 1 1 1 1	1 0 r r r 1 1 1		Z Z C U S V	6	rr–(PC+A)
	プログラムカウンタ PC の内容に A レジスタの内容を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容を 16 ビットレジスタ rr の内容と比較します。					
ADD A,n	0 1 1 0 0 0 0 1	n n n n n n n n		C Z C H S V	2	A ← A+n
	A レジスタの内容に、オブジェクトコード中の即値 n を加算し、結果を A レジスタに入れます。					
ADD g,n	1 1 1 0 1 g g g	0 1 1 0 0 0 0 1	n n n n n n n n	C Z C H S V	3	g ← g+n
	8 ビットレジスタ g の内容に、オブジェクトコード中の即値 n を加算し、結果をレジスタ g に入れます。					
ADD gg,mn	1 1 1 0 1 g g g m m m m m m m m	0 1 1 0 1 0 0 1	n n n n n n n n	C Z C U S V	4	gg ← gg+mn
	16 ビットレジスタ gg の内容に、オブジェクトコード中の即値 mn を加算し、結果をレジスタ gg に入れます。					
ADD r,g	1 1 1 0 1 g g g	0 0 r r r 0 0 1		C Z C H S V	2	r ← r+g
	8 ビットレジスタ r の内容に、8 ビットレジスタ g の内容を加算し、結果をレジスタ r に入れます。					
ADD rr,gg	1 1 1 0 1 g g g	1 0 r r r 0 0 1		C Z C U S V	3	rr ← rr+gg
	16 ビットレジスタ rr の内容に、16 ビットレジスタ gg の内容を加算し、結果をレジスタ rr に入れます。					
ADD r,(x)	1 1 1 0 0 0 0 0	x x x x x x x x	0 0 r r r 0 0 1	C Z C H S V	4	r ← r+(x)
	8 ビットレジスタ r の内容に、オブジェクトコード中の x で直接指定されるアドレス (0x0000 ~ 0x00FF 番地) のメモリ内容を加算し、結果をレジスタ r に入れます。					
ADD r,(vw)	1 1 1 0 0 0 0 1 0 0 r r r 0 0 1	w w w w w w w w	v v v v v v v v	C Z C H S V	5	r ← r+(vw)
	8 ビットレジスタ r の内容に、オブジェクトコード中の vw で直接指定されるアドレス (0x0000 ~ 0xFFFF 番地) のメモリ内容を加算し、結果をレジスタ r に入れます。					

ニモニック	オブジェクトコード (2 進)	フラグ J Z C H S V	サイクル	オペレーション
ADD r,(DE)	1 1 1 0 0 0 1 0 0 0 r r r 0 0 1	C Z C H S V	3	r ← r+(DE)
	8 ビットレジスタ r の内容に、レジスタペア DE で指定されるアドレスのメモリ内容を加算し、結果をレジスタ r に入れます。			
ADD r,(HL)	1 1 1 0 0 0 1 1 0 0 r r r 0 0 1	C Z C H S V	3	r ← r+(HL)
	8 ビットレジスタ r の内容に、レジスタペア HL で指定されるアドレスのメモリ内容を加算し、結果をレジスタ r に入れます。			
ADD r,(IX)	1 1 1 0 0 1 0 0 0 0 r r r 0 0 1	C Z C H S V	3	r ← r+(IX)
	8 ビットレジスタ r の内容に、インデックスレジスタ IX で指定されるアドレスのメモリ内容を加算し、結果をレジスタ r に入れます。			
ADD r,(IY)	1 1 1 0 0 1 0 1 0 0 r r r 0 0 1	C Z C H S V	3	r ← r+(IY)
	8 ビットレジスタ r の内容に、インデックスレジスタ IY で指定されるアドレスのメモリ内容を加算し、結果をレジスタ r に入れます。			
ADD r,(IX+d)	1 1 0 1 0 1 0 0 d d d d d d d d 0 0 r r r 0 0 1	C Z C H S V	5	r ← r+(IX+d)
	8 ビットレジスタ r の内容に、インデックスレジスタ IX にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容を加算し、結果をレジスタ r に入れます。			
ADD r,(IY+d)	1 1 0 1 0 1 0 1 d d d d d d d d 0 0 r r r 0 0 1	C Z C H S V	5	r ← r+(IY+d)
	8 ビットレジスタ r の内容に、インデックスレジスタ IY にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容を加算し、結果をレジスタ r に入れます。			
ADD r,(SP+d)	1 1 0 1 0 1 1 0 d d d d d d d d 0 0 r r r 0 0 1	C Z C H S V	5	r ← r+(SP+d)
	8 ビットレジスタ r の内容に、スタックポインタ SP の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容に 8 ビットレジスタ r の内容を加算し、結果をレジスタ r に入れます。			
ADD r,(HL+d)	1 1 0 1 0 1 1 1 d d d d d d d d 0 0 r r r 0 0 1	C Z C H S V	5	r ← r+(HL+d)
	8 ビットレジスタ r の内容に、レジスタペア HL の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容を加算し、結果をレジスタ r に入れます。			
ADD r,(HL+C)	1 1 1 0 0 1 1 1 0 0 r r r 0 0 1	C Z C H S V	5	r ← r+(HL+C)
	8 ビットレジスタ r の内容に、レジスタペア HL の内容に C レジスタの内容を符号拡張して加算した値で指定されるアドレスのメモリ内容を加算し、結果をレジスタ r に入れます。			
ADD r,(+SP)	1 1 1 0 0 1 1 0 0 0 r r r 0 0 1	C Z C H S V	4	SP ← SP+1:r ← r+(SP)
	8 ビットレジスタ r の内容に、スタックポインタ SP の内容をインクリメントし、その値で指定されるアドレスのメモリ内容を加算し、結果をレジスタ r に入れます。			
ADD r,(PC+A) 注	0 1 0 0 1 1 1 1 0 0 r r r 0 0 1	C Z C H S V	5	r ← r+(PC+A)
	8 ビットレジスタ r の内容に、プログラムカウンタ PC の内容に A レジスタの内容を符号拡張して加算した値で指定されるアドレスのメモリ内容を加算し、結果をレジスタ r に入れます。			
ADD (x),n	1 1 1 0 0 0 0 0 x x x x x x x x 0 1 1 0 0 0 0 1 n n n n n n n n	C Z C H S V	6	(x) ← (x)+n
	オブジェクトコード中の x で直接指定されるアドレス (0x0000 ~ 0x00FF 番地) のメモリ内容に、オブジェクトコード中の即値 n を加算し、結果を前記のアドレスに入れます。			
ADD (vw),n	1 1 1 0 0 0 0 1 w w w w w w w w v v v v v v v v 0 1 1 0 0 0 0 1 n n n n n n n n	C Z C H S V	7	(vw) ← (vw)+n
	オブジェクトコード中の vw で直接指定されるアドレス (0x0000 ~ 0xFFFF 番地) のメモリ内容に、オブジェクトコード中の即値 n を加算し、結果を前記のアドレスに入れます。			
ADD (DE),n	1 1 1 0 0 0 1 0 0 1 1 0 0 0 0 1 n n n n n n n n	C Z C H S V	5	(DE) ← (DE)+n
	レジスタペア DE で指定されるアドレスのメモリ内容に、オブジェクトコード中の即値 n を加算し、結果を前記のアドレスに入れます。			
ADD (HL),n	1 1 1 0 0 0 1 1 0 1 1 0 0 0 0 1 n n n n n n n n	C Z C H S V	5	(HL) ← (HL)+n
	レジスタペア HL で指定されるアドレスのメモリ内容に、オブジェクトコード中の即値 n を加算し、結果を前記のアドレスに入れます。			
ADD (IX),n	1 1 1 0 0 1 0 0 0 1 1 0 0 0 0 1 n n n n n n n n	C Z C H S V	5	(IX) ← (IX)+n
	インデックスレジスタ IX で指定されるアドレスのメモリ内容に、オブジェクトコード中の即値 n を加算し、結果を前記のアドレスに入れます。			
ADD (IY),n	1 1 1 0 0 1 0 1 0 1 1 0 0 0 0 1 n n n n n n n n	C Z C H S V	5	(IY) ← (IY)+n
	インデックスレジスタ IY で指定されるアドレスのメモリ内容に、オブジェクトコード中の即値 n を加算し、結果を前記のアドレスに入れます。			
ADD (IX+d),n	1 1 0 1 0 1 0 0 d d d d d d d d 0 1 1 0 0 0 0 1 n n n n n n n n	C Z C H S V	7	(IX+d) ← (IX+d)+n
	インデックスレジスタ IX の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容にオブジェクトコード中の即値 n を加算し、結果を前記のアドレスに入れます。			
ADD (IY+d),n	1 1 0 1 0 1 0 1 d d d d d d d d 0 1 1 0 0 0 0 1 n n n n n n n n	C Z C H S V	7	(IY+d) ← (IY+d)+n
	インデックスレジスタ IY の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容にオブジェクトコード中の即値 n を加算し、結果を前記のアドレスに入れます。			
ADD (SP+d),n	1 1 0 1 0 1 1 0 d d d d d d d d 0 1 1 0 0 0 0 1 n n n n n n n n	C Z C H S V	7	(SP+d) ← (SP+d)+n
	スタックポインタ SP の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容に、オブジェクトコード中の即値 n を加算し、結果を前記のアドレスに入れます。			
ADD (HL+d),n	1 1 0 1 0 1 1 1 d d d d d d d d 0 1 1 0 0 0 0 1 n n n n n n n n	C Z C H S V	7	(HL+d) ← (HL+d)+n
	レジスタペア HL の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容に、オブジェクトコード中の即値 n を加算し、結果を前記のアドレスに入れます。			
ADD (HL+C),n	1 1 1 0 0 1 1 1 0 1 1 0 0 0 0 1 n n n n n n n n	C Z C H S V	7	(HL+C) ← (HL+C)+n
	レジスタペア HL の内容に C レジスタの内容を符号拡張して加算した値で指定されるアドレスのメモリ内容に、オブジェクトコード中の即値 n を加算し、結果を前記のアドレスに入れます。			
ADD (+SP),n	1 1 1 0 0 1 1 0 0 1 1 0 0 0 0 1 n n n n n n n n	C Z C H S V	6	SP ← SP+1:(SP) ← (SP)+n
	スタックポインタ SP の内容をインクリメントし、その値で指定されるアドレスのメモリ内容にオブジェクトコード中の即値 n を加算し、結果を前記のアドレスに入れます。			
ADD (PC+A),n 注	0 1 0 0 1 1 1 1 0 1 1 0 0 0 0 1 n n n n n n n n	C Z C H S V	7	(PC+A) ← (PC+A)+n
	プログラムカウンタ PC の内容に A レジスタの内容を符号拡張して加算した値で指定されるアドレスのメモリ内容に、オブジェクトコード中の即値 n を加算し、結果を前記のアドレスに入れます。			
ADD rr,(x)	1 1 1 0 0 0 0 0 x x x x x x x x 1 0 r r r 0 0 1	C Z C U S V	5	rr ← rr+(x+1, x)
	16 ビットレジスタ rr の内容に、オブジェクトコード中の x で直接指定されるアドレス (0x0000 ~ 0x00FF 番地) から連続する 2 バイトのメモリ内容を加算し、結果をレジスタ rr に入れます。			

ニモニック	オブジェクトコード (2 進)	フラグ J Z C H S V	サイクル	オペレーション
ADD rr,(vw)	1 1 1 0 0 0 0 1 w w w w w w w w v v v v v v v v 1 0 r r r 0 0 1	C Z C U S V	6	rr ← rr+(vw+1, vw)
	16 ビットレジスタ rr の内容に、オブジェクトコード中の vw で直接指定されるアドレス (0x0000 ~ 0xFFFF 番地) から連続する 2 バイトのメモリ内容を加算し、結果を前記のアドレスに入れます。			
ADD rr,(DE)	1 1 1 0 0 0 1 0 1 0 r r r 0 0 1	C Z C U S V	4	rr ← rr+(DE+1, DE)
	16 ビットレジスタ rr の内容に、レジスタペア DE で指定されるアドレスから連続する 2 バイトのメモリ内容を加算し、結果をレジスタ rr に入れます。			
ADD rr,(HL)	1 1 1 0 0 0 1 1 1 0 r r r 0 0 1	C Z C U S V	4	rr ← rr+(HL+1, HL)
	16 ビットレジスタ rr の内容に、レジスタペア HL で指定されるアドレスから連続する 2 バイトのメモリ内容を加算し、結果をレジスタ rr に入れます。			
ADD rr,(IX)	1 1 1 0 0 1 0 0 1 0 r r r 0 0 1	C Z C U S V	4	rr ← rr+(IX+1, IX)
	16 ビットレジスタ rr の内容に、インデックスレジスタ IX で指定されるアドレスから連続する 2 バイトのメモリ内容を加算し、結果をレジスタ rr に入れます。			
ADD rr,(IY)	1 1 1 0 0 1 0 1 1 0 r r r 0 0 1	C Z C U S V	4	rr ← rr+(IY+1, IY)
	16 ビットレジスタ rr の内容に、インデックスレジスタ IY で指定されるアドレスから連続する 2 バイトのメモリ内容を加算し、結果をレジスタ rr に入れます。			
ADD rr,(IX+d)	1 1 0 1 0 1 0 0 d d d d d d d d 1 0 r r r 0 0 1	C Z C U S V	6	rr ← rr+(IX+d+1, IX+d)
	16 ビットレジスタ rr の内容に、インデックスレジスタ IX にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容を加算し、結果をレジスタ rr に入れます。			
ADD rr,(IY+d)	1 1 0 1 0 1 0 1 d d d d d d d d 1 0 r r r 0 0 1	C Z C U S V	6	rr ← rr+(IY+d+1, IY+d)
	16 ビットレジスタ rr の内容に、インデックスレジスタ IY にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容を加算し、結果をレジスタ rr に入れます。			
ADD rr,(SP+d)	1 1 0 1 0 1 1 0 d d d d d d d d 1 0 r r r 0 0 1	C Z C U S V	6	rr ← rr+(SP+d+1, SP+d)
	16 ビットレジスタ rr の内容に、スタックポインタ SP の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容を加算し、結果をレジスタ rr に入れます。			
ADD rr,(HL+d)	1 1 0 1 0 1 1 1 d d d d d d d d 1 0 r r r 0 0 1	C Z C U S V	6	rr ← rr+(HL+d+1, HL+d)
	16 ビットレジスタ rr の内容に、レジスタペア HL の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容を加算し、結果をレジスタ rr に入れます。			
ADD rr,(HL+C)	1 1 1 0 0 1 1 1 1 0 r r r 0 0 1	C Z C U S V	6	rr ← rr+(HL+C+1, HL+C)
	16 ビットレジスタ rr の内容に、レジスタペア HL の内容に C レジスタの内容を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容を加算し、結果をレジスタ rr に入れます。			
ADD rr,(+SP)	1 1 1 0 0 1 1 0 1 0 r r r 0 0 1	C Z C U S V	5	SP ← SP+1:rr ← rr+(SP+1, SP)
	16 ビットレジスタ rr の内容に、スタックポインタ SP の内容をインクリメントしその値で指定されるアドレスから連続する 2 バイトのメモリ内容を加算し、結果をレジスタ rr に入れます。			
ADD rr,(PC+A) 注	0 1 0 0 1 1 1 1 1 0 r r r 0 0 1	C Z C U S V	6	rr ← rr+(PC+A+1, PC+A)
	16 ビットレジスタ rr の内容に、プログラムカウンタ PC の内容に A レジスタの内容を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容を加算し、結果をレジスタ rr に入れます。			
ADDC A,n	0 1 1 0 0 0 0 0 n n n n n n n n	C Z C H S V	2	A ← A+n+CF
	A レジスタの内容に、オブジェクトコード中の即値 n およびキャリーフラグの内容を加算し、結果を A レジスタに入れます。			
ADDC g,n	1 1 1 0 1 g g g 0 1 1 0 0 0 0 0 n n n n n n n n	C Z C H S V	3	g ← g+n+CF
	8 ビットレジスタ g の内容に、オブジェクトコード中の即値 n およびキャリーフラグの内容を加算し、結果をレジスタ g に入れます。			
ADDC gg,mn	1 1 1 0 1 g g g 0 1 1 0 1 0 0 0 n n n n n n n n m m m m m m m m	C Z C U S V	4	gg ← gg+mn+CF
	16 ビットレジスタ gg の内容に、オブジェクトコード中の即値 mn およびキャリーフラグの内容を加算し、結果をレジスタ gg に入れます。			
ADDC r,g	1 1 1 0 1 g g g 0 0 r r r 0 0 0	C Z C H S V	2	r ← r+g+CF
	8 ビットレジスタ r の内容に、8 ビットレジスタ g の内容およびキャリーフラグの内容を加算し、結果をレジスタ r に入れます。			
ADDC rr,gg	1 1 1 0 1 g g g 1 0 r r r 0 0 0	C Z C U S V	3	rr ← rr+gg+CF
	16 ビットレジスタ rr の内容に、16 ビットレジスタ gg の内容およびキャリーフラグの内容を加算し、結果をレジスタ rr に入れます。			
ADDC r,(x)	1 1 1 0 0 0 0 0 x x x x x x x x 0 0 r r r 0 0 0	C Z C H S V	4	r ← r+(x)+CF
	8 ビットレジスタ r の内容に、オブジェクトコード中の x で直接指定されるアドレス (0x0000 ~ 0x00FF 番地) のメモリ内容およびキャリーフラグの内容を加算し、結果をレジスタ r に入れます。			
ADDC r, (vw)	1 1 1 0 0 0 0 1 w w w w w w w w v v v v v v v v 0 0 r r r 0 0 0	C Z C H S V	5	r ← r+(vw)+CF
	8 ビットレジスタ r の内容に、オブジェクトコード中の vw で直接指定されるアドレス (0x0000 ~ 0xFFFF 番地) のメモリ内容およびキャリーフラグの内容を加算し、結果をレジスタ r に入れます。			
ADDC r,(DE)	1 1 1 0 0 0 1 0 0 0 r r r 0 0 0	C Z C H S V	3	r ← r+(DE)+CF
	8 ビットレジスタ r の内容に、レジスタペア DE で指定されるアドレスのメモリ内容およびキャリーフラグの内容を加算し、結果をレジスタ r に入れます。			
ADDC r,(HL)	1 1 1 0 0 0 1 1 0 0 r r r 0 0 0	C Z C H S V	3	r ← r+(HL)+CF
	8 ビットレジスタ r の内容に、レジスタペア HL で指定されるアドレスのメモリ内容およびキャリーフラグの内容を加算し、結果をレジスタ r に入れます。			
ADDC r,(IX)	1 1 1 0 0 1 0 0 0 0 r r r 0 0 0	C Z C H S V	3	r ← r+(IX)+CF
	8 ビットレジスタ r の内容に、インデックスレジスタ IX で指定されるアドレスのメモリ内容およびキャリーフラグの内容を加算し、結果をレジスタ r に入れます。			
ADDC r,(IY)	1 1 1 0 0 1 0 1 0 0 r r r 0 0 0	C Z C H S V	3	r ← r+(IY)+CF
	8 ビットレジスタ r の内容に、インデックスレジスタ IY で指定されるアドレスのメモリ内容およびキャリーフラグの内容を加算し、結果をレジスタ r に入れます。			
ADDC r,(IX+d)	1 1 0 1 0 1 0 0 d d d d d d d d 0 0 r r r 0 0 0	C Z C H S V	5	r ← r+(IX+d)+CF
	8 ビットレジスタ r の内容に、インデックスレジスタ IX の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容およびキャリーフラグの内容を加算し、結果をレジスタ r に入れます。			
ADDC r,(IY+d)	1 1 0 1 0 1 0 1 d d d d d d d d 0 0 r r r 0 0 0	C Z C H S V	5	r ← r+(IY+d)+CF
	8 ビットレジスタ r の内容に、インデックスレジスタ IY の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容およびキャリーフラグの内容を加算し、結果をレジスタ r に入れます。			
ADDC r,(SP+d)	1 1 0 1 0 1 1 0 d d d d d d d d 0 0 r r r 0 0 0	C Z C H S V	5	r ← r+(SP+d)+CF
	8 ビットレジスタ r の内容に、スタックポインタ SP の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容およびキャリーフラグの内容を加算し、結果をレジスタ r に入れます。			

ニモニック	オブジェクトコード (2 進)	フラグ J Z C H S V	サイクル	オペレーション
ADDC r,(HL+d)	1 1 0 1 0 1 1 1 d d d d d d d d 0 0 r r r 0 0 0	C Z C H S V	5	r ← r+(HL+d)+CF
	8 ビットレジスタ r の内容に、レジスタペア HL の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容およびキャリーフラグの内容を加算し、結果をレジスタ r に入れます。			
ADDC r,(HL+C)	1 1 1 0 0 1 1 1 0 0 r r r 0 0 0	C Z C H S V	5	r ← r+(HL+C)+CF
	8 ビットレジスタ r の内容に、レジスタペア HL の内容に C レジスタの内容を符号拡張して加算した値で指定されるアドレスのメモリ内容およびキャリーフラグの内容を加算し、結果をレジスタ r に入れます。			
ADDC r,(+SP)	1 1 1 0 0 1 1 0 0 0 r r r 0 0 0	C Z C H S V	4	SP ← SP+1:r ← r+(SP)+CF
	8 ビットレジスタ r の内容に、スタックポインタ SP の内容をインクリメントし、その値で指定されるアドレスのメモリ内容およびキャリーフラグの内容を加算し、結果をレジスタ r に入れます。			
ADDC r,(PC+A) 注	0 1 0 0 1 1 1 1 0 0 r r r 0 0 0	C Z C H S V	5	r ← r+(PC+A)+CF
	8 ビットレジスタ r の内容に、プログラムカウンタ PC の内容に A レジスタの内容を符号拡張して加算した値で指定されるアドレスのメモリ内容およびキャリーフラグの内容を加算し、結果をレジスタ r に入れます。			
ADDC (x),n	1 1 1 0 0 0 0 0 x x x x x x x x 0 1 1 0 0 0 0 0 n n n n n n n n	C Z C H S V	6	(x) ← (x)+n+CF
	オブジェクトコード中の x で直接指定されるアドレス (0x0000 ~ 0x00FF 番地) のメモリ内容に、オブジェクトコード中の即値 n およびキャリーフラグの内容を加算し、結果を前記のアドレスに入れます。			
ADDC (vw),n	1 1 1 0 0 0 0 1 w w w w w w w w v v v v v v v v 0 1 1 0 0 0 0 0 n n n n n n n n	C Z C H S V	7	(vw) ← (vw)+n+CF
	オブジェクトコード中の vw で直接指定されるアドレス (0x0000 ~ 0xFFFF 番地) のメモリ内容に、オブジェクトコード中の即値 n およびキャリーフラグの内容を加算し、結果を前記のアドレスに入れます。			
ADDC (DE),n	1 1 1 0 0 0 1 0 0 1 1 0 0 0 0 0 n n n n n n n n	C Z C H S V	5	(DE) ← (DE)+n+CF
	レジスタペア DE で指定されるアドレスのメモリ内容に、オブジェクトコード中の即値 n およびキャリーフラグの内容を加算し、結果を前記のアドレスに入れます。			
ADDC (HL),n	1 1 1 0 0 0 1 1 0 1 1 0 0 0 0 0 n n n n n n n n	C Z C H S V	5	(HL) ← (HL)+n +CF
	レジスタペア HL で指定されるアドレスのメモリ内容に、オブジェクトコード中の即値 n およびキャリーフラグの内容を加算し、結果を前記のアドレスに入れます。			
ADDC (IX),n	1 1 1 0 0 1 0 0 0 1 1 0 0 0 0 0 n n n n n n n n	C Z C H S V	5	(IX) ← (IX)+n+CF
	インデックスレジスタ IX で指定されるアドレスのメモリ内容に、オブジェクトコード中の即値 n およびキャリーフラグの内容を加算し、結果を前記のアドレスに入れます。			
ADDC (IY),n	1 1 1 0 0 1 0 1 0 1 1 0 0 0 0 0 n n n n n n n n	C Z C H S V	5	(IY) ← (IY)+n+CF
	インデックスレジスタ IY で指定されるアドレスのメモリ内容に、オブジェクトコード中の即値 n およびキャリーフラグの内容を加算し、結果を前記のアドレスに入れます。			
ADDC (IX+d),n	1 1 0 1 0 1 0 0 d d d d d d d d 0 1 1 0 0 0 0 0 n n n n n n n n	C Z C H S V	7	(IX+d) ← (IX+d)+n+CF
	インデックスレジスタ IX の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容に、オブジェクトコード中の即値 n およびキャリーフラグの内容を加算し、結果を前記のアドレスに入れます。			
ADDC (IY+d),n	1 1 0 1 0 1 0 1 d d d d d d d d 0 1 1 0 0 0 0 0 n n n n n n n n	C Z C H S V	7	(IY+d) ← (IY+d)+n+CF
	インデックスレジスタ IY の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容に、オブジェクトコード中の即値 n およびキャリーフラグの内容を加算し、結果を前記のアドレスに入れます。			
ADDC (SP+d),n	1 1 0 1 0 1 1 0 d d d d d d d d 0 1 1 0 0 0 0 0 n n n n n n n n	C Z C H S V	7	(SP+d) ← (SP+d)+n+CF
	スタックポインタ SP の内容に、オブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容に、オブジェクトコード中の即値 n およびキャリーフラグの内容を加算し、結果を前記のアドレスに入れます。			
ADDC (HL+d),n	1 1 0 1 0 1 1 1 d d d d d d d d 0 1 1 0 0 0 0 0 n n n n n n n n	C Z C H S V	7	(HL+d) ← (HL+d)+n+CF
	レジスタペア HL の内容に、オブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容に、オブジェクトコード中の即値 n およびキャリーフラグの内容を加算し、結果を前記のアドレスに入れます。			
ADDC (HL+C),n	1 1 1 0 0 1 1 1 0 1 1 0 0 0 0 0 n n n n n n n n	C Z C H S V	7	(HL+C) ← (HL+C)+n+CF
	レジスタペア HL の内容に、C レジスタの内容を符号拡張して加算した値で指定されるアドレスのメモリ内容に、オブジェクトコード中の即値 n およびキャリーフラグの内容を加算し、結果を前記のアドレスに入れます。			
ADDC (+SP),n	1 1 1 0 0 1 1 0 0 1 1 0 0 0 0 0 n n n n n n n n	C Z C H S V	6	SP ← SP+1:(SP) ← (SP)+n+CF
	スタックポインタ SP の内容をインクリメントし、その値で指定されるアドレスのメモリ内容に、オブジェクトコード中の即値 n およびキャリーフラグの内容を加算し、結果を前記のアドレスに入れます。			
ADDC (PC+A),n 注	0 1 0 0 1 1 1 1 0 1 1 0 0 0 0 0 n n n n n n n n	C Z C H S V	7	(PC+A) ← (PC+A)+n+CF
	プログラムカウンタ PC の内容に A レジスタの内容を符号拡張して加算した値で指定されるアドレスのメモリ内容に、オブジェクトコード中の即値 n およびキャリーフラグの内容を加算し、結果を前記のアドレスに入れます。			
ADDC rr,(x)	1 1 1 0 0 0 0 0 x x x x x x x x 1 0 r r r 0 0 0	C Z C U S V	5	rr ← rr+(x+1, x)+CF
	16 ビットレジスタ rr の内容に、オブジェクトコード中の x で直接指定されるアドレス (0x0000 ~ 0x00FF 番地) から連続する 2 バイトのメモリ内容およびキャリーフラグの内容を加算し、結果をレジスタ rr に入れます。			
ADDC rr,(vw)	1 1 1 0 0 0 0 1 w w w w w w w w v v v v v v v v 1 0 r r r 0 0 0	C Z C U S V	6	rr ← rr+(vw+1, vw)+CF
	16 ビットレジスタ rr の内容に、オブジェクトコード中の vw で直接指定されるアドレス (0x0000 ~ 0xFFFF 番地) から連続する 2 バイトのメモリ内容およびキャリーフラグの内容を加算し、結果をレジスタ rr に入れます。			
ADDC rr,(DE)	1 1 1 0 0 0 1 0 1 0 r r r 0 0 0	C Z C U S V	4	rr ← rr+(DE+1, DE)+CF
	16 ビットレジスタ rr の内容に、レジスタペア DE で指定されるアドレスから連続する 2 バイトのメモリ内容およびキャリーフラグの内容を加算し、結果をレジスタ rr に入れます。			
ADDC rr,(HL)	1 1 1 0 0 0 1 1 1 0 r r r 0 0 0	C Z C U S V	4	rr ← rr+(HL+1, HL)+CF
	16 ビットレジスタ rr の内容に、レジスタペア HL で指定されるアドレスから連続する 2 バイトのメモリ内容およびキャリーフラグの内容を加算し、結果をレジスタ rr に入れます。			
ADDC rr,(IX)	1 1 1 0 0 1 0 0 1 0 r r r 0 0 0	C Z C U S V	4	rr ← rr+(IX+1, IX)+CF
	16 ビットレジスタ rr の内容に、インデックスレジスタ IX で指定されるアドレスから連続する 2 バイトのメモリ内容およびキャリーフラグの内容を加算し、結果をレジスタ rr に入れます。			
ADDC rr,(IY)	1 1 1 0 0 1 0 1 1 0 r r r 0 0 0	C Z C U S V	4	rr ← rr+(IY+1,IY)+CF
	16 ビットレジスタ rr の内容に、インデックスレジスタ IY で指定されるアドレスから連続する 2 バイトのメモリ内容およびキャリーフラグの内容を加算し、結果をレジスタ rr に入れます。			

ニモニック	オブジェクトコード (2 進)	フラグ J Z C H S V	サイクル	オペレーション
ADDC rr,(IX+d)	1 1 0 1 0 1 0 0 d d d d d d d d 1 0 r r r 0 0 0	C Z C U S V	6	rr ← rr+(IX+d+1, IX+d)+CF
	16 ビットレジスタ rr の内容に、インデックスレジスタ IX の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容およびキャリーフラグの内容を加算し、結果をレジスタ rr に入れます。			
ADDC rr,(IY+d)	1 1 0 1 0 1 0 1 d d d d d d d d 1 0 r r r 0 0 0	C Z C U S V	6	rr ← rr+(IY+d+1, IY+d)+CF
	16 ビットレジスタ rr の内容に、インデックスレジスタ IY の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容およびキャリーフラグの内容を加算し、結果をレジスタ rr に入れます。			
ADDC rr,(SP+d)	1 1 0 1 0 1 1 0 d d d d d d d d 1 0 r r r 0 0 0	C Z C U S V	6	rr ← rr+(SP+d+1, SP+d)+CF
	16 ビットレジスタ rr の内容に、スタックポインタ SP の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容およびキャリーフラグの内容を加算し、結果をレジスタ rr に入れます。			
ADDC rr,(HL+d)	1 1 0 1 0 1 1 1 d d d d d d d d 1 0 r r r 0 0 0	C Z C U S V	6	rr ← rr+(HL+d+1, HL+d)+CF
	16 ビットレジスタ rr の内容に、レジスタペア HL の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容およびキャリーフラグの内容を加算し、結果をレジスタ rr に入れます。			
ADDC rr,(HL+C)	1 1 1 0 0 1 1 1 1 0 r r r 0 0 0	C Z C U S V	6	rr ← rr+(HL+C+1, HL+C)+CF
	16 ビットレジスタ rr の内容に、レジスタペア HL の内容に C レジスタの内容を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容およびキャリーフラグの内容を加算し、結果をレジスタ rr に入れます。			
ADDC rr,(+SP)	1 1 1 0 0 1 1 0 1 0 r r r 0 0 0	C Z C U S V	5	SP ← SP+1:rr ← rr+(SP+1, SP)+CF
	16 ビットレジスタ rr の内容に、スタックポインタ SP の内容をインクリメントし、その値で指定されるアドレスから連続する 2 バイトのメモリ内容およびキャリーフラグの内容を加算し、結果をレジスタ rr に入れます。			
ADDC rr,(PC+A) 注	0 1 0 0 1 1 1 1 1 0 r r r 0 0 0	C Z C U S V	6	rr ← rr+(PC+A+1, PC+A)+CF
	16 ビットレジスタ rr の内容に、プログラムカウンタ PC の内容に A レジスタの内容を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容およびキャリーフラグの内容を加算し、結果をレジスタ rr に入れます。			
SUB A,n	0 1 1 0 0 0 1 1 n n n n n n n n	C Z C H S V	2	A ← A–n
	A レジスタの内容から、オブジェクトコード中の即値 n を引き、結果を A レジスタに入れます。			
SUB g,n	1 1 1 0 1 g g g 0 1 1 0 0 0 1 1 n n n n n n n n	C Z C H S V	3	g ← g–n
	8 ビットレジスタ g の内容から、オブジェクトコード中の即値 n を引き、結果をレジスタ g に入れます。			
SUB gg,mn	1 1 1 0 1 g g g 0 1 1 0 1 0 1 1 n n n n n n n n m m m m m m m m	C Z C U S V	4	gg ← gg–mn
	16 ビットレジスタ gg の内容から、オブジェクトコード中の即値 mn を引き、結果をレジスタ gg に入れます。			
SUB r,g	1 1 1 0 1 g g g 0 0 r r r 0 1 1	C Z C H S V	2	r ← r–g
	8 ビットレジスタ r の内容から、8 ビットレジスタ g の内容を引き、結果をレジスタ r に入れます。			
SUB rr,gg	1 1 1 0 1 g g g 1 0 r r r 0 1 1	C Z C U S V	3	rr ← rr–gg
	16 ビットレジスタ rr の内容から、16 ビットレジスタ gg の内容を引き、結果をレジスタ rr に入れます。			
SUB r,(x)	1 1 1 0 0 0 0 0 x x x x x x x x 0 0 r r r 0 1 1	C Z C H S V	4	r ← r–(x)
	8 ビットレジスタ r の内容から、オブジェクトコード中の x で直接指定されるアドレス (0x0000 ~ 0x00FF 番地) のメモリ内容を引き、結果をレジスタ r に入れます。			
SUB r,(vw)	1 1 1 0 0 0 0 1 w w w w w w w w v v v v v v v v 0 0 r r r 0 1 1	C Z C H S V	5	r ← r–(vw)
	8 ビットレジスタ r の内容から、オブジェクトコード中の vw で直接指定されるアドレス (0x0000 ~ 0xFFFF 番地) のメモリ内容を引き、結果をレジスタ r に入れます。			
SUB r,(DE)	1 1 1 0 0 0 1 0 0 0 r r r 0 1 1	C Z C H S V	3	r ← r–(DE)
	8 ビットレジスタ r の内容から、レジスタペア DE で指定されるアドレスのメモリ内容を引き、結果をレジスタ r に入れます。			
SUB r,(HL)	1 1 1 0 0 0 1 1 0 0 r r r 0 1 1	C Z C H S V	3	r ← r–(HL)
	8 ビットレジスタ r の内容から、レジスタペア HL で指定されるアドレスのメモリ内容を引き、結果をレジスタ r に入れます。			
SUB r,(IX)	1 1 1 0 0 1 0 0 0 0 r r r 0 1 1	C Z C H S V	3	r ← r–(IX)
	8 ビットレジスタ r の内容から、インデックスレジスタ IX で指定されるアドレスのメモリ内容を引き、結果をレジスタ r に入れます。			
SUB r,(IY)	1 1 1 0 0 1 0 1 0 0 r r r 0 1 1	C Z C H S V	3	r ← r–(IY)
	8 ビットレジスタ r の内容から、インデックスレジスタ IY で指定されるアドレスのメモリ内容を引き、結果をレジスタ r に入れます。			
SUB r,(IX+d)	1 1 0 1 0 1 0 0 d d d d d d d d 0 0 r r r 0 1 1	C Z C H S V	5	r ← r–(IX+d)
	8 ビットレジスタ r の内容から、インデックスレジスタ IX の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容を引き、結果をレジスタ r に入れます。			
SUB r,(IY+d)	1 1 0 1 0 1 0 1 d d d d d d d d 0 0 r r r 0 1 1	C Z C H S V	5	r ← r–(IY+d)
	8 ビットレジスタ r の内容から、インデックスレジスタ IY の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容を引き、結果をレジスタ r に入れます。			
SUB r,(SP+d)	1 1 0 1 0 1 1 0 d d d d d d d d 0 0 r r r 0 1 1	C Z C H S V	5	r ← r–(SP+d)
	8 ビットレジスタ r の内容から、スタックポインタ SP の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容を引き、結果をレジスタ r に入れます。			
SUB r,(HL+d)	1 1 0 1 0 1 1 1 d d d d d d d d 0 0 r r r 0 1 1	C Z C H S V	5	r ← r–(HL+d)
	8 ビットレジスタ r の内容から、レジスタペア HL の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容を引き、結果をレジスタ r に入れます。			
SUB r,(HL+C)	1 1 1 0 0 1 1 1 0 0 r r r 0 1 1	C Z C H S V	5	r ← r–(HL)+C
	8 ビットレジスタ r の内容から、レジスタペア HL の内容に C レジスタの内容を符号拡張して加算した値で指定されるアドレスのメモリ内容を引き、結果をレジスタ r に入れます。			
SUB r,(+SP)	1 1 1 0 0 1 1 0 0 0 r r r 0 1 1	C Z C H S V	4	SP ← SP+1:r ← r–(SP)
	8 ビットレジスタ r の内容から、スタックポインタ SP の内容をインクリメントしその値で指定されるアドレスのメモリ内容を引き、結果をレジスタ r に入れます。			
SUB r,(PC+A) 注	0 1 0 0 1 1 1 1 0 0 r r r 0 1 1	C Z C H S V	5	r ← r–(PC+A)
	8 ビットレジスタ r の内容から、プログラムカウンタ PC の内容に A レジスタの内容を符号拡張して加算した値で指定されるアドレスのメモリ内容を引き、結果をレジスタ r に入れます。			
SUB (x),n	1 1 1 0 0 0 0 0 x x x x x x x x 0 1 1 0 0 0 1 1 n n n n n n n n	C Z C H S V	6	(x) ← (x)–n
	オブジェクトコード中の x で直接指定されるアドレス (0x0000 ~ 0x00FF 番地) のメモリ内容から、オブジェクトコード中の即値 n を引き、結果を前記のアドレスに入れます。			
SUB (vw),n	1 1 1 0 0 0 0 1 w w w w w w w w v v v v v v v v 0 1 1 0 0 0 1 1 n n n n n n n n	C Z C H S V	7	(vw) ← (vw)–n
	オブジェクトコード中の vw で直接指定されるアドレス (0x0000 ~ 0xFFFF 番地) のメモリ内容から、オブジェクトコード中の即値 n を引き、結果を前記のアドレスに入れます。			

ニモニック	オブジェクトコード (2 進)			フラグ J Z C H S V	サイクル	オペレーション
SUB (DE),n	1 1 1 0 0 0 1 0	0 1 1 0 0 0 1 1	n n n n n n n n	C Z C H S V	5	(DE) ← (DE)–n
	レジスタペア DE で指定されるアドレスのメモリ内容から、オブジェクトコード中の即値 n を引き、結果を前記のアドレスに入れます。					
SUB (HL),n	1 1 1 0 0 0 1 1	0 1 1 0 0 0 1 1	n n n n n n n n	C Z C H S V	5	(HL) ← (HL)–n
	レジスタペア HL で指定されるアドレスのメモリ内容から、オブジェクトコード中の即値 n を引き、結果を前記のアドレスに入れます。					
SUB (IX),n	1 1 1 0 0 1 0 0	0 1 1 0 0 0 1 1	n n n n n n n n	C Z C H S V	5	(IX) ← (IX)–n
	インデックスレジスタ IX で指定されるアドレスのメモリ内容から、オブジェクトコード中の即値 n を引き、結果を前記のアドレスに入れます。					
SUB (IY),n	1 1 1 0 0 1 0 1	0 1 1 0 0 0 1 1	n n n n n n n n	C Z C H S V	5	(IY) ← (IY)–n
	インデックスレジスタ IY で指定されるアドレスのメモリ内容から、オブジェクトコード中の即値 n を引き、結果を前記のアドレスに入れます。					
SUB (IX+d),n	1 1 0 1 0 1 0 0 n n n n n n n n	d d d d d d d d	0 1 1 0 0 0 1 1	C Z C H S V	7	(IX+d) ← (IX+d)–n
	インデックスレジスタ IX の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容から、オブジェクトコード中の即値 n を引き、結果を前記のアドレスに入れます。					
SUB (IY+d),n	1 1 0 1 0 1 0 1 n n n n n n n n	d d d d d d d d	0 1 1 0 0 0 1 1	C Z C H S V	7	(IY+d) ← (IY+d)–n
	インデックスレジスタ IY の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容から、オブジェクトコード中の即値 n を引き、結果を前記のアドレスに入れます。					
SUB (SP+d),n	1 1 0 1 0 1 1 0 n n n n n n n n	d d d d d d d d	0 1 1 0 0 0 1 1	C Z C H S V	7	(SP+d) ← (SP+d)–n
	スタックポインタ SP の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容から、オブジェクトコード中の即値 n を引き、結果を前記のアドレスに入れます。					
SUB (HL+d),n	1 1 0 1 0 1 1 1 n n n n n n n n	d d d d d d d d	0 1 1 0 0 0 1 1	C Z C H S V	7	(HL+d) ← (HL+d)–n
	レジスタペア HL の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容から、オブジェクトコード中の即値 n を引き、結果を前記のアドレスに入れます。					
SUB (HL+C),n	1 1 1 0 0 1 1 1	0 1 1 0 0 0 1 1	n n n n n n n n	C Z C H S V	7	(HL+C) ← (HL+C)–n
	レジスタペア HL の内容に C レジスタの内容を符号拡張して加算した値で指定されるアドレスのメモリ内容から、オブジェクトコード中の即値 n を引き、結果を前記のアドレスに入れます。					
SUB (+SP),n	1 1 1 0 0 1 1 0	0 1 1 0 0 0 1 1	n n n n n n n n	C Z C H S V	6	SP ← SP+1:(SP) ← (SP)–n
	スタックポインタ SP の内容をインクリメントし、その値で指定されるアドレスのメモリ内容からオブジェクトコード中の即値 n を引き、結果を前記のアドレスに入れます。					
SUB (PC+A),n 注	0 1 0 0 1 1 1 1	0 1 1 0 0 0 1 1	n n n n n n n n	C Z C H S V	7	(PC+A) ← (PC+A)–n
	プログラムカウンタ PC の内容に A レジスタの内容を符号拡張して加算した値で指定されるアドレスのメモリ内容から、オブジェクトコード中の即値 n を引き、結果を前記のアドレスに入れます。					
SUB rr,(x)	1 1 1 0 0 0 0 0	x x x x x x x x	1 0 r r r 0 1 1	C Z C U S V	5	rr ← rr–(x+1, x)
	16 ビットレジスタ rr の内容から、オブジェクトコード中の x で直接指定されるアドレス (0x0000 ~ 0x00FF 番地) から連続する 2 バイトのメモリ内容を引き、結果をレジスタ rr に入れます。					
SUB rr,(vw)	1 1 1 0 0 0 0 1 1 0 r r r 0 1 1	w w w w w w w w	v v v v v v v v	C Z C U S V	6	rr ← rr–(vw+1, vw)
	16 ビットレジスタ rr の内容から、オブジェクトコード中の vw で直接指定されるアドレス (0x0000 ~ 0xFFFF 番地) から連続する 2 バイトのメモリ内容を引き、結果をレジスタ rr に入れます。					
SUB rr,(DE)	1 1 1 0 0 0 1 0	1 0 r r r 0 1 1		C Z C U S V	4	rr ← rr–(DE+1, DE)
	16 ビットレジスタ rr の内容から、レジスタペア DE で指定されるアドレスから連続する 2 バイトのメモリ内容を引き、結果をレジスタ rr に入れます。					
SUB rr,(HL)	1 1 1 0 0 0 1 1	1 0 r r r 0 1 1		C Z C U S V	4	rr ← rr–(HL+1, HL)
	16 ビットレジスタ rr の内容から、レジスタペア HL で指定されるアドレスから連続する 2 バイトのメモリ内容を引き、結果をレジスタ rr に入れます。					
SUB rr,(IX)	1 1 1 0 0 1 0 0	1 0 r r r 0 1 1		C Z C U S V	4	rr ← rr–(IX+1, IX)
	16 ビットレジスタ rr の内容から、インデックスレジスタ IX で指定されるアドレスから連続する 2 バイトのメモリ内容を引き、結果をレジスタ rr に入れます。					
SUB rr,(IY)	1 1 1 0 0 1 0 1	1 0 r r r 0 1 1		C Z C U S V	4	rr ← rr–(IY+1, IY)
	16 ビットレジスタ rr の内容から、インデックスレジスタ IY で指定されるアドレスから連続する 2 バイトのメモリ内容を引き、結果をレジスタ rr に入れます。					
SUB rr,(IX+d)	1 1 0 1 0 1 0 0	d d d d d d d d	1 0 r r r 0 1 1	C Z C U S V	6	rr ← rr–(IX+d+1, IX+d)
	16 ビットレジスタ rr の内容から、インデックスレジスタ IX の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容を引き、結果をレジスタ rr に入れます。					
SUB rr,(IY+d)	1 1 0 1 0 1 0 1	d d d d d d d d	1 0 r r r 0 1 1	C Z C U S V	6	rr ← rr–(IY+d+1, IY+d)
	16 ビットレジスタ rr の内容から、インデックスレジスタ IY の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容を引き、結果をレジスタ rr に入れます。					
SUB rr,(SP+d)	1 1 0 1 0 1 1 0	d d d d d d d d	1 0 r r r 0 1 1	C Z C U S V	6	rr ← rr–(SP+d+1, SP+d)
	16 ビットレジスタ rr の内容から、スタックポインタ SP の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容を引き、結果をレジスタ rr に入れます。					
SUB rr,(HL+d)	1 1 0 1 0 1 1 1	d d d d d d d d	1 0 r r r 0 1 1	C Z C U S V	6	rr ← rr–(HL+d+1, HL+d)
	16 ビットレジスタ rr の内容から、レジスタペア HL の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容を引き、結果をレジスタ rr に入れます。					
SUB rr,(HL+C)	1 1 1 0 0 1 1 1	1 0 r r r 0 1 1		C Z C U S V	6	rr ← rr–(HL+C+1, HL+C)
	16 ビットレジスタ rr の内容から、レジスタペア HL の内容に C レジスタの内容を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容を引き、結果をレジスタ rr に入れます。					
SUB rr,(+SP)	1 1 1 0 0 1 1 0	1 0 r r r 0 1 1		C Z C U S V	5	SP ← SP+1:rr ← rr–(SP+1, SP)
	16 ビットレジスタ rr の内容から、スタックポインタ SP の内容をインクリメントしその値で指定されるアドレスから連続する 2 バイトのメモリ内容を引き、結果をレジスタ rr に入れます。					
SUB rr,(PC+A) 注	0 1 0 0 1 1 1 1	1 0 r r r 0 1 1		C Z C U S V	6	rr ← rr–(PC+A+1, PC+A)
	16 ビットレジスタ rr の内容から、プログラムカウンタ PC の内容に A レジスタの内容を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容を引き、結果をレジスタ rr に入れます。					
SUBB A,n	0 1 1 0 0 0 1 0	n n n n n n n n		C Z C H S V	2	A ← A–n–CF
	A レジスタの内容から、オブジェクトコード中の即値 n およびキャリーフラグの内容を引き、結果を A レジスタに入れます。					

ニモニック	オブジェクトコード (2 進)	フラグ J Z C H S V	サイクル	オペレーション
SUBB g,n	1 1 1 0 1 g g g 0 1 1 0 0 0 1 0 n n n n n n n n	C Z C H S V	3	g ← g–n–CF
	8 ビットレジスタ g の内容から、オブジェクトコード中の即値 n およびキャリーフラグの内容を引き、結果をレジスタ g に入れます。			
SUBB gg,mn	1 1 1 0 1 g g g 0 1 1 0 1 0 1 0 n n n n n n n n m m m m m m m m	C Z C U S V	4	gg ← gg–mn–CF
	16 ビットレジスタ gg の内容から、オブジェクトコード中の即値 mn およびキャリーフラグの内容を引き、結果をレジスタ gg に入れます。			
SUBB r,g	1 1 1 0 1 g g g 0 0 r r r 0 1 0	C Z C H S V	2	r ← r–g–CF
	8 ビットレジスタ r の内容から、8 ビットレジスタ g の内容およびキャリーフラグの内容を引き、結果をレジスタ r に入れます。			
SUBB rr,gg	1 1 1 0 1 g g g 1 0 r r r 0 1 0	C Z C U S V	3	rr ← rr–gg–CF
	16 ビットレジスタ rr の内容から、16 ビットレジスタ gg の内容およびキャリーフラグの内容を引き、結果をレジスタ rr に入れます。			
SUBB r,(x)	1 1 1 0 0 0 0 0 x x x x x x x x 0 0 r r r 0 1 0	C Z C H S V	4	r ← r–(x)–CF
	8 ビットレジスタ r の内容から、オブジェクトコード中の x で直接指定されるアドレス (0x0000 ~ 0x00FF 番地) のメモリ内容およびキャリーフラグの内容を引き、結果をレジスタ r に入れます。			
SUBB r,(vw)	1 1 1 0 0 0 0 1 w w w w w w w w v v v v v v v v 0 0 r r r 0 1 0	C Z C H S V	5	r ← r–(vw)–CF
	8 ビットレジスタ r の内容から、オブジェクトコード中の vw で直接指定されるアドレス (0x0000 ~ 0xFFFF 番地) のメモリ内容およびキャリーフラグの内容を引き、結果をレジスタ r に入れます。			
SUBB r,(DE)	1 1 1 0 0 0 1 0 0 0 r r r 0 1 0	C Z C H S V	3	r ← r–(DE)–CF
	8 ビットレジスタ r の内容から、レジスタペア DE で指定されるアドレスのメモリ内容およびキャリーフラグの内容を引き、結果をレジスタ r に入れます。			
SUBB r,(HL)	1 1 1 0 0 0 1 1 0 0 r r r 0 1 0	C Z C H S V	3	r ← r–(HL)–CF
	8 ビットレジスタ r の内容から、レジスタペア HL で指定されるアドレスのメモリ内容およびキャリーフラグの内容を引き、結果をレジスタ r に入れます。			
SUBB r,(IX)	1 1 1 0 0 1 0 0 0 0 r r r 0 1 0	C Z C H S V	3	r ← r–(IX)–CF
	8 ビットレジスタ r の内容から、インデックスレジスタ IX で指定されるアドレスのメモリ内容およびキャリーフラグの内容を引き、結果をレジスタ r に入れます。			
SUBB r,(IY)	1 1 1 0 0 1 0 1 0 0 r r r 0 1 0	C Z C H S V	3	r ← r–(IY)–CF
	8 ビットレジスタ r の内容から、インデックスレジスタ IY で指定されるアドレスのメモリ内容およびキャリーフラグの内容を引き、結果をレジスタ r に入れます。			
SUBB r,(IX+d)	1 1 0 1 0 1 0 0 d d d d d d d d 0 0 r r r 0 1 0	C Z C H S V	5	r ← r–(IX+d)–CF
	8 ビットレジスタ r の内容から、インデックスレジスタ IX の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容およびキャリーフラグの内容を引き、結果をレジスタ r に入れます。			
SUBB r,(IY+d)	1 1 0 1 0 1 0 1 d d d d d d d d 0 0 r r r 0 1 0	C Z C H S V	5	r ← r–(IX+d)–CF
	8 ビットレジスタ r の内容から、インデックスレジスタ IY の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容およびキャリーフラグの内容を引き、結果をレジスタ r に入れます。			
SUBB r,(SP+d)	1 1 0 1 0 1 1 0 d d d d d d d d 0 0 r r r 0 1 0	C Z C H S V	5	r ← r–(SP+d)–CF
	8 ビットレジスタ r の内容から、スタックポインタ SP の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容およびキャリーフラグの内容を引き、結果をレジスタ r に入れます。			
SUBB r,(HL+d)	1 1 0 1 0 1 1 1 d d d d d d d d 0 0 r r r 0 1 0	C Z C H S V	5	r ← r–(HL+d)–CF
	8 ビットレジスタ r の内容から、レジスタペア HL の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容およびキャリーフラグの内容を引き、結果をレジスタ r に入れます。			
SUBB r,(HL+C)	1 1 1 0 0 1 1 1 0 0 r r r 0 1 0	C Z C H S V	5	r ← r–(HL+C)–CF
	8 ビットレジスタ r の内容から、レジスタペア HL の内容に C レジスタの内容を符号拡張して加算した値で指定されるアドレスのメモリ内容およびキャリーフラグの内容を引き、結果をレジスタ r に入れます。			
SUBB r,(+SP)	1 1 1 0 0 1 1 0 0 0 r r r 0 1 0	C Z C H S V	4	SP ← SP+1:r ← r–(SP)–CF
	8 ビットレジスタ r の内容から、スタックポインタ SP の内容をインクリメントし、その値で指定されるアドレスのメモリ内容およびキャリーフラグの内容を引き、結果をレジスタ r に入れます。			
SUBB r,(PC+A) 注	0 1 0 0 1 1 1 1 0 0 r r r 0 1 0	C Z C H S V	5	r ← r–(PC+A)–CF
	8 ビットレジスタ r の内容から、プログラムカウンタ PC の内容に A レジスタの内容を符号拡張して加算した値で指定されるアドレスのメモリ内容およびキャリーフラグの内容を引き、結果をレジスタ r に入れます。			
SUBB (x),n	1 1 1 0 0 0 0 0 x x x x x x x x 0 1 1 0 0 0 1 0 n n n n n n n n	C Z C H S V	6	(x) ← (x)–n–CF
	オブジェクトコード中の x で直接指定されるアドレス (0x0000 ~ 0x00FF 番地) のメモリ内容から、オブジェクトコード中の即値 n およびキャリーフラグの内容を引き、結果を前記のアドレスに入れます。			
SUBB (vw),n	1 1 1 0 0 0 0 1 w w w w w w w w v v v v v v v v 0 1 1 0 0 0 1 0 n n n n n n n n	C Z C H S V	7	(vw) ← (vw)–n–CF
	オブジェクトコード中の vw で直接指定されるアドレス (0x0000 ~ 0xFFFF 番地) のメモリ内容から、オブジェクトコード中の即値 n およびキャリーフラグの内容を引き、結果を前記のアドレスに入れます。			
SUBB (DE),n	1 1 1 0 0 0 1 0 0 1 1 0 0 0 1 0 n n n n n n n n	C Z C H S V	5	(DE) ← (DE)–n–CF
	レジスタペア DE で指定されるアドレスのメモリ内容から、オブジェクトコード中の即値 n およびキャリーフラグの内容を引き、結果を前記のアドレスに入れます。			
SUBB (HL),n	1 1 1 0 0 0 1 1 0 1 1 0 0 0 1 0 n n n n n n n n	C Z C H S V	5	(HL) ← (HL)–n–CF
	レジスタペア HL で指定されるアドレスのメモリ内容から、オブジェクトコード中の即値 n およびキャリーフラグの内容を引き、結果を前記のアドレスに入れます。			
SUBB (IX),n	1 1 1 0 0 1 0 0 0 1 1 0 0 0 1 0 n n n n n n n n	C Z C H S V	5	(IX) ← (IX)–n–CF
	インデックスレジスタ IX で指定されるアドレスのメモリ内容から、オブジェクトコード中の即値 n およびキャリーフラグの内容を引き、結果を前記のアドレスに入れます。			
SUBB (IY),n	1 1 1 0 0 1 0 1 0 1 1 0 0 0 1 0 n n n n n n n n	C Z C H S V	5	(IY) ← (IY)–n–CF
	インデックスレジスタ IY で指定されるアドレスのメモリ内容から、オブジェクトコード中の即値 n およびキャリーフラグの内容を引き、結果を前記のアドレスに入れます。			
SUBB (IX+d),n	1 1 0 1 0 1 0 0 d d d d d d d d 0 1 1 0 0 0 1 0 n n n n n n n n	C Z C H S V	7	(IX+d) ← (IX+d)–n–CF
	インデックスレジスタ IX の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容から、オブジェクトコード中の即値 n およびキャリーフラグの内容を引き、結果を前記のアドレスに入れます。			
SUBB (IY+d),n	1 1 0 1 0 1 0 1 d d d d d d d d 0 1 1 0 0 0 1 0 n n n n n n n n	C Z C H S V	7	(IY+d) ← (IY+d)–n–CF
	インデックスレジスタ IY の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容から、オブジェクトコード中の即値 n およびキャリーフラグの内容を引き、結果を前記のアドレスに入れます。			

ニモニック	オブジェクトコード (2 進)			フラグ J Z C H S V	サイクル	オペレーション
SUBB (SP+d),n	1 1 0 1 0 1 1 0 n n n n	d d d d d d d d	0 1 1 0 0 0 1 0	C Z C H S V	7	(SP+d) ← (SP+d)–n–CF
	スタックポインタ SP の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容から、オブジェクトコード中の即値 n およびキャリーフラグの内容を引き、結果を前記のアドレスに入れます。					
SUBB (HL+d),n	1 1 0 1 0 1 1 1 n n n n	d d d d d d d d	0 1 1 0 0 0 1 0	C Z C H S V	7	(HL+d) ← (HL+d)–n–CF
	レジスタペア HL の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容から、オブジェクトコード中の即値 n およびキャリーフラグの内容を引き、結果を前記のアドレスに入れます。					
SUBB (HL+C),n	1 1 1 0 0 1 1 1	0 1 1 0 0 0 1 0	n n n n n n n n	C Z C H S V	7	(HL+C) ← (HL+C)–n–CF
	レジスタペア HL の内容に C レジスタの内容を符号拡張して加算した値で指定されるアドレスのメモリ内容から、オブジェクトコード中の即値 n およびキャリーフラグの内容を引き、結果を前記のアドレスに入れます。					
SUBB (+SP),n	1 1 1 0 0 1 1 0	0 1 1 0 0 0 1 0	n n n n n n n n	C Z C H S V	6	SP ← SP+1:(SP) ← (SP)–n–CF
	スタックポインタ SP の内容をインクリメントし、その値で指定されたアドレスのメモリ内容からオブジェクトコード中の即値 n およびキャリーフラグの内容を引き、結果を前記のアドレスに入れます。					
SUBB (PC+A),n 注	0 1 0 0 1 1 1 1	0 1 1 0 0 0 1 0	n n n n n n n n	C Z C H S V	7	(PC+A) ← (PC+A)–n–CF
	プログラムカウンタ PC の内容に A レジスタの内容を符号拡張して加算した値で指定されるアドレスのメモリ内容から、オブジェクトコード中の即値 n およびキャリーフラグの内容を引き、結果を前記のアドレスに入れます。					
SUBB rr,(x)	1 1 1 0 0 0 0 0	x x x x x x x x	1 0 r r r 0 1 0	C Z C U S V	5	rr ← rr–(x+1, x)–CF
	16 ビットレジスタ rr の内容から、オブジェクトコード中の x で直接指定されるアドレス (0x0000 ~ 0x00FF 番地) から連続する 2 バイトのメモリ内容およびキャリーフラグの内容を引き、結果をレジスタ rr に入れます。					
SUBB rr,(vw)	1 1 1 0 0 0 0 1 1 0 r r r 0 1 0	w w w w w w w w	v v v v v v v v	C Z C U S V	6	rr ← rr–(vw+1, vw)–CF
	16 ビットレジスタ rr の内容から、オブジェクトコード中の vw で直接指定されるアドレス (0x0000 ~ 0xFFFF 番地) から連続する 2 バイトのメモリ内容およびキャリーフラグの内容を引き、結果をレジスタ rr に入れます。					
SUBB rr,(DE)	1 1 1 0 0 0 1 0	1 0 r r r 0 1 0		C Z C U S V	4	rr ← rr–(DE+1, DE)–CF
	16 ビットレジスタ rr の内容から、レジスタペア DE で指定されるアドレスから連続する 2 バイトのメモリ内容およびキャリーフラグの内容を引き、結果をレジスタ rr に入れます。					
SUBB rr,(HL)	1 1 1 0 0 0 1 1	1 0 r r r 0 1 0		C Z C U S V	4	rr ← rr–(HL+1, HL)–CF
	16 ビットレジスタ rr の内容から、レジスタペア HL で指定されるアドレスから連続する 2 バイトのメモリ内容およびキャリーフラグの内容を引き、結果をレジスタ rr に入れます。					
SUBB rr,(IX)	1 1 1 0 0 1 0 0	1 0 r r r 0 1 0		C Z C U S V	4	rr ← rr–(IX+1, IX)–CF
	16 ビットレジスタ rr の内容から、インデックスレジスタ IX で指定されるアドレスから連続する 2 バイトのメモリ内容およびキャリーフラグの内容を引き、結果をレジスタ rr に入れます。					
SUBB rr,(IY)	1 1 1 0 0 1 0 1	1 0 r r r 0 1 0		C Z C U S V	4	rr ← rr–(IY+1, IY)–CF
	16 ビットレジスタ rr の内容から、インデックスレジスタ IY で指定されるアドレスから連続する 2 バイトのメモリ内容およびキャリーフラグの内容を引き、結果をレジスタ rr に入れます。					
SUBB rr,(IX+d)	1 1 0 1 0 1 0 0	d d d d d d d d	1 0 r r r 0 1 0	C Z C U S V	6	rr ← rr–(IX+d+1, IX+d)–CF
	16 ビットレジスタ rr の内容から、インデックスレジスタ IX の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容およびキャリーフラグの内容を引き、結果をレジスタ rr に入れます。					
SUBB rr,(IY+d)	1 1 0 1 0 1 0 1	d d d d d d d d	1 0 r r r 0 1 0	C Z C U S V	6	rr ← rr–(IY+d+1, IY+d)–CF
	16 ビットレジスタ rr の内容から、インデックスレジスタ IY の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容およびキャリーフラグの内容を引き、結果をレジスタ rr に入れます。					
SUBB rr,(SP+d)	1 1 0 1 0 1 1 0	d d d d d d d d	1 0 r r r 0 1 0	C Z C U S V	6	rr ← rr–(SP+d+1, SP+d)–CF
	16 ビットレジスタ rr の内容から、スタックポインタ SP の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容およびキャリーフラグの内容を引き、結果をレジスタ rr に入れます。					
SUBB rr,(HL+d)	1 1 0 1 0 1 1 1	d d d d d d d d	1 0 r r r 0 1 0	C Z C U S V	6	rr ← rr–(HL+d+1, HL+d)–CF
	16 ビットレジスタ rr の内容から、レジスタペア HL の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容およびキャリーフラグの内容を引き、結果をレジスタ rr に入れます。					
SUBB rr,(HL+C)	1 1 1 0 0 1 1 1	1 0 r r r 0 1 0		C Z C U S V	6	rr ← rr–(HL+C+1, HL+C)–CF
	16 ビットレジスタ rr の内容から、レジスタペア HL の内容に C レジスタの内容を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容およびキャリーフラグの内容を引き、結果をレジスタ rr に入れます。					
SUBB rr,(+SP)	1 1 1 0 0 1 1 0	1 0 r r r 0 1 0		C Z C U S V	5	SP ← SP+1:rr ← rr–(SP+1, SP)–CF
	16 ビットレジスタ rr の内容から、スタックポインタ SP の内容をインクリメントしその値で指定されるアドレスから連続する 2 バイトのメモリ内容およびキャリーフラグの内容を引き、結果をレジスタ rr に入れます。					
SUBB rr,(PC+A) 注	0 1 0 0 1 1 1 1	1 0 r r r 0 1 0		C Z C U S V	6	rr ← rr–(PC+A+1, PC+A)–CF
	16 ビットレジスタ rr の内容から、プログラムカウンタ PC の内容に A レジスタの内容を符号拡張して加算した値で指定されるアドレスのメモリ内容およびキャリーフラグの内容を引き、結果をレジスタ rr に入れます。					
AND A,n	0 1 1 0 0 1 0 0	n n n n n n n n		Z Z – – – –	2	A ← A&n
	A レジスタの内容と、オブジェクトコード中の即値 n とでビットごとの論理積を取り、結果を A レジスタに入れます。					
AND g,n	1 1 1 0 1 g g g	0 1 1 0 0 1 0 0	n n n n n n n n	Z Z – – – –	3	g ← g&n
	8 ビットレジスタ g の内容と、オブジェクトコード中の即値 n とでビットごとの論理積を取り、結果をレジスタ g に入れます。					
AND gg,mn	1 1 1 0 1 g g g m m m m	0 1 1 0 1 1 0 0	n n n n m m m m	Z Z – – – –	4	gg ← gg&mn
	16 ビットレジスタ gg の内容と、オブジェクトコード中の即値 mn とでビットごとの論理積を取り、結果をレジスタ gg に入れます。					
AND r,g	1 1 1 0 1 g g g	0 0 r r r 1 0 0		Z Z – – – –	2	r ← r&g
	8 ビットレジスタ r の内容と、8 ビットレジスタ g の内容とでビットごとの論理積を取り、結果をレジスタ r に入れます。					
AND rr,gg	1 1 1 0 1 g g g	1 0 r r r 1 0 0		Z Z – – – –	3	rr ← rr&gg
	16 ビットレジスタ rr の内容と、16 ビットレジスタ gg の内容とでビットごとの論理積を取り、結果をレジスタ rr に入れます。					
AND r,(x)	1 1 1 0 0 0 0 0	x x x x x x x x	0 0 r r r 1 0 0	Z Z – – – –	4	r ← r&(x)
	8 ビットレジスタ r の内容と、オブジェクトコード中の即値 x で直接指定されるアドレス (0x0000 ~ 0x00FF 番地) のメモリ内容とでビットごとの論理積を取り、結果をレジスタ r に入れます。					
AND r,(vw)	1 1 1 0 0 0 0 1 0 0 r r r 1 0 0	w w w w w w w w	v v v v v v v v	Z Z – – – –	5	r ← r&(vw)
	8 ビットレジスタ r の内容と、オブジェクトコード中の即値 vw で直接指定されるアドレス (0x0000 ~ 0xFFFF 番地) のメモリ内容とでビットごとの論理積を取り、結果をレジスタ r に入れます。					

ニモニック	オブジェクトコード (2 進)	フラグ J Z C H S V	サイクル	オペレーション
AND r,(DE)	1 1 1 0 0 0 1 0 0 0 r r r 1 0 0	Z Z – – – –	3	r ← r&(DE)
	8 ビットレジスタ r の内容と、レジスタペア DE で指定されるアドレスのメモリ内容とでビットごとの論理積を取り、結果をレジスタ r に入れます。			
AND r,(HL)	1 1 1 0 0 0 1 1 0 0 r r r 1 0 0	Z Z – – – –	3	r ← r&(HL)
	8 ビットレジスタ r の内容と、レジスタペア HL で指定されるアドレスのメモリ内容とでビットごとの論理積を取り、結果をレジスタ r に入れます。			
AND r,(IX)	1 1 1 0 0 1 0 0 0 0 r r r 1 0 0	Z Z – – – –	3	r ← r&(IX)
	8 ビットレジスタ r の内容と、インデックスレジスタ IX で指定されるアドレスのメモリ内容とでビットごとの論理積を取り、結果をレジスタ r に入れます。			
AND r,(IY)	1 1 1 0 0 1 0 1 0 0 r r r 1 0 0	Z Z – – – –	3	r ← r&(IY)
	8 ビットレジスタ r の内容と、インデックスレジスタ IY で指定されるアドレスのメモリ内容とでビットごとの論理積を取り、結果をレジスタ r に入れます。			
AND r,(IX+d)	1 1 0 1 0 1 0 0 d d d d d d d d 0 0 r r r 1 0 0	Z Z – – – –	5	r ← r&(IX+d)
	8 ビットレジスタ r の内容と、インデックスレジスタ IX の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容とでビットごとの論理積を取り、結果をレジスタ r に入れます。			
AND r,(IY+d)	1 1 0 1 0 1 0 1 d d d d d d d d 0 0 r r r 1 0 0	Z Z – – – –	5	r ← r&(IY+d)
	8 ビットレジスタ r の内容と、インデックスレジスタ IY の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容とでビットごとの論理積を取り、結果をレジスタ r に入れます。			
AND r,(SP+d)	1 1 0 1 0 1 1 0 d d d d d d d d 0 0 r r r 1 0 0	Z Z – – – –	5	r ← r&(SP+d)
	8 ビットレジスタ r の内容と、スタックポインタ SP の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容とでビットごとの論理積を取り、結果をレジスタ r に入れます。			
AND r,(HL+d)	1 1 0 1 0 1 1 1 d d d d d d d d 0 0 r r r 1 0 0	Z Z – – – –	5	r ← r&(HL+d)
	8 ビットレジスタ r の内容と、レジスタペア HL の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容とでビットごとの論理積を取り、結果をレジスタ r に入れます。			
AND r,(HL+C)	1 1 1 0 0 1 1 1 0 0 r r r 1 0 0	Z Z – – – –	5	r ← r&(HL+C)
	8 ビットレジスタ r の内容と、レジスタペア HL の内容に C レジスタの内容を符号拡張して加算した値で指定されるアドレスのメモリ内容とでビットごとの論理積を取り、結果をレジスタ r に入れます。			
AND r,(+SP)	1 1 1 0 0 1 1 0 0 0 r r r 1 0 0	Z Z – – – –	4	SP ← SP+1:r ← r&(SP)
	8 ビットレジスタ r の内容と、スタックポインタ SP の内容をインクリメントし、その値で指定されるアドレスのメモリ内容とでビットごとの論理積を取り、結果をレジスタ r に入れます。			
AND r,(PC+A) 注	0 1 0 0 1 1 1 1 0 0 r r r 1 0 0	Z Z – – – –	5	r ← r&(PC+A)
	8 ビットレジスタ r の内容と、プログラムカウンタ PC の内容に A レジスタの内容を符号拡張して加算した値で指定されるアドレスのメモリ内容とでビットごとの論理積を取り、結果をレジスタ r に入れます。			
AND (X),n	1 1 1 0 0 0 0 0 x x x x x x x x 0 1 1 0 0 1 0 0 n n n n n n n n	Z Z – – – –	6	(x) ← (x)&n
	オブジェクトコード中の x で直接指定されるアドレス (0x0000 ~ 0x00FF 番地) のメモリ内容と、オブジェクトコード中の即値 n とでビットごとの論理積を取り、結果を前記のアドレスに入れます。			
AND (vw),n	1 1 1 0 0 0 0 1 w w w w w w w w v v v v v v v v 0 1 1 0 0 1 0 0 n n n n n n n n	Z Z – – – –	7	(vw) ← (vw)&n
	オブジェクトコード中の vw で直接指定されるアドレス (0x0000 ~ 0xFFFF 番地) のメモリ内容と、オブジェクトコード中の即値 n とでビットごとの論理積を取り、結果を前記のアドレスに入れます。			
AND (DE),n	1 1 1 0 0 0 1 0 0 1 1 0 0 1 0 0 n n n n n n n n	Z Z – – – –	5	(DE) ← (DE)&n
	レジスタペア DE で指定されるアドレスのメモリ内容と、オブジェクトコード中の即値 n とでビットごとの論理積を取り、結果を前記のアドレスに入れます。			
AND (HL),n	1 1 1 0 0 0 1 1 0 1 1 0 0 1 0 0 n n n n n n n n	Z Z – – – –	5	(HL) ← (HL)&n
	レジスタペア HL で指定されるアドレスのメモリ内容と、オブジェクトコード中の即値 n とでビットごとの論理積を取り、結果を前記のアドレスに入れます。			
AND (IX),n	1 1 1 0 0 1 0 0 0 1 1 0 0 1 0 0 n n n n n n n n	Z Z – – – –	5	(IX) ← (IX)&n
	インデックスレジスタ IX で指定されるアドレスのメモリ内容と、オブジェクトコード中の即値 n とでビットごとの論理積を取り、結果を前記のアドレスに入れます。			
AND (IY),n	1 1 1 0 0 1 0 1 0 1 1 0 0 1 0 0 n n n n n n n n	Z Z – – – –	5	(IY) ← (IY)&n
	インデックスレジスタ IY で指定されるアドレスのメモリ内容と、オブジェクトコード中の即値 n とでビットごとの論理積を取り、結果を前記のアドレスに入れます。			
AND (IX+d),n	1 1 0 1 0 1 0 0 d d d d d d d d 0 1 1 0 0 1 0 0 n n n n n n n n	Z Z – – – –	7	(IX+d) ← (IX+d)&n
	インデックスレジスタ IX の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容と、オブジェクトコード中の即値 n とでビットごとの論理積を取り、結果を前記のアドレスに入れます。			
AND (IY+d),n	1 1 0 1 0 1 0 1 d d d d d d d d 0 1 1 0 0 1 0 0 n n n n n n n n	Z Z – – – –	7	(IY+d) ← (IY+d)&n
	インデックスレジスタ IY の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容と、オブジェクトコード中の即値 n とでビットごとの論理積を取り、結果を前記のアドレスに入れます。			
AND (SP+d),n	1 1 0 1 0 1 1 0 d d d d d d d d 0 1 1 0 0 1 0 0 n n n n n n n n	Z Z – – – –	7	(SP+d) ← (SP+d)&n
	スタックポインタ SP の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容と、オブジェクトコード中の即値 n とでビットごとの論理積を取り、結果を前記のアドレスに入れます。			
AND (HL+d),n	1 1 0 1 0 1 1 1 d d d d d d d d 0 1 1 0 0 1 0 0 n n n n n n n n	Z Z – – – –	7	(HL+d) ← (HL+d)&n
	レジスタペア HL の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容と、オブジェクトコード中の即値 n とでビットごとの論理積を取り、結果を前記のアドレスに入れます。			
AND (HL+C),n	1 1 1 0 0 1 1 1 0 1 1 0 0 1 0 0 n n n n n n n n	Z Z – – – –	7	(HL+C) ← (HL+C)&n
	レジスタペア HL の内容に C レジスタの内容を符号拡張して加算した値で指定されるアドレスのメモリ内容と、オブジェクトコード中の即値 n とでビットごとの論理積を取り、結果を前記のアドレスに入れます。			
AND (+SP),n	1 1 1 0 0 1 1 0 0 1 1 0 0 1 0 0 n n n n n n n n	Z Z – – – –	6	SP ← SP+1:(SP) ← (SP)&n
	スタックポインタ SP の内容をインクリメントし、その値で指定されるアドレスのメモリ内容と、オブジェクトコード中の即値 n とでビットごとの論理積を取り、結果を前記のアドレスに入れます。			
AND (PC+A),n 注	0 1 0 0 1 1 1 1 0 1 1 0 0 1 0 0 n n n n n n n n	Z Z – – – –	7	(PC+A) ← (PC+A)&n
	プログラムカウンタ PC の内容に A レジスタの内容を符号拡張して加算した値で指定されるアドレスのメモリ内容と、オブジェクトコード中の即値 n とでビットごとの論理積を取り、結果を前記のアドレスに入れます。			

ニモニック	オブジェクトコード (2 進)	フラグ J Z C H S V	サイクル	オペレーション
AND rr,(x)	1 1 1 0 0 0 0 0 | x x x x x x x x | 1 0 r r r 1 0 0	Z Z – – – –	5	rr ← rr&(x)
	16 ビットレジスタ rr の内容と、オブジェクトコード中の x で直接指定されるアドレス (0x0000 ~ 0x00FF 番地) から連続する 2 バイトのメモリ内容とでビットごとの論理積を取り、結果をレジスタ rr に入れます。			
AND rr,(vw)	1 1 1 0 0 0 0 1 | w w w w w w w w | v v v v v v v v | 1 0 r r r 1 0 0	Z Z – – – –	6	rr ← rr&(vw)
	16 ビットレジスタ rr の内容と、オブジェクトコード中の vw で直接指定されるアドレス (0x0000 ~ 0xFFFF 番地) から連続する 2 バイトのメモリ内容とでビットごとの論理積を取り、結果をレジスタ rr に入れます。			
AND rr,(DE)	1 1 1 0 0 0 1 0 | 1 0 r r r 1 0 0	Z Z – – – –	4	rr ← rr&(DE)
	16 ビットレジスタ rr の内容と、レジスタペア DE で指定されるアドレスから連続する 2 バイトのメモリ内容とでビットごとの論理積を取り、結果をレジスタ rr に入れます。			
AND rr,(HL)	1 1 1 0 0 0 1 1 | 1 0 r r r 1 0 0	Z Z – – – –	4	rr ← rr&(HL)
	16 ビットレジスタ rr の内容と、レジスタペア HL で指定されるアドレスから連続する 2 バイトのメモリ内容とでビットごとの論理積を取り、結果をレジスタ rr に入れます。			
AND rr,(IX)	1 1 1 0 0 1 0 0 | 1 0 r r r 1 0 0	Z Z – – – –	4	rr ← rr&(IX)
	16 ビットレジスタ rr の内容と、インデックスレジスタ IX で指定されるアドレスから連続する 2 バイトのメモリ内容とでビットごとの論理積を取り、結果をレジスタ rr に入れます。			
AND rr,(IY)	1 1 1 0 0 1 0 1 | 1 0 r r r 1 0 0	Z Z – – – –	4	rr ← rr&(IY)
	16 ビットレジスタ rr の内容と、インデックスレジスタ IY で指定されるアドレスから連続する 2 バイトのメモリ内容とでビットごとの論理積を取り、結果をレジスタ rr に入れます。			
AND rr,(IX+d)	1 1 0 1 0 1 0 0 | d d d d d d d d | 1 0 r r r 1 0 0	Z Z – – – –	6	rr ← rr&(IX+d)
	16 ビットレジスタ rr の内容と、インデックスレジスタ IX の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容とでビットごとの論理積を取り、結果をレジスタ rr に入れます。			
AND rr,(IY+d)	1 1 0 1 0 1 0 1 | d d d d d d d d | 1 0 r r r 1 0 0	Z Z – – – –	6	rr ← rr&(IY+d)
	16 ビットレジスタ rr の内容と、インデックスレジスタ IY の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容とでビットごとの論理積を取り、結果をレジスタ rr に入れます。			
AND rr,(SP+d)	1 1 0 1 0 1 1 0 | d d d d d d d d | 1 0 r r r 1 0 0	Z Z – – – –	6	rr ← rr&(SP+d)
	16 ビットレジスタ rr の内容と、スタックポインタ SP の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容とでビットごとの論理積を取り、結果をレジスタ rr に入れます。			
AND rr,(HL+d)	1 1 0 1 0 1 1 1 | d d d d d d d d | 1 0 r r r 1 0 0	Z Z – – – –	6	rr ← rr&(HL+d)
	16 ビットレジスタ rr の内容と、レジスタペア HL の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容とでビットごとの論理積を取り、結果をレジスタ rr に入れます。			
AND rr,(HL+C)	1 1 1 0 0 1 1 1 | 1 0 r r r 1 0 0	Z Z – – – –	6	rr ← rr&(HL+C)
	16 ビットレジスタ rr の内容と、レジスタペア HL の内容に C レジスタの内容を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容とでビットごとの論理積を取り、結果をレジスタ rr に入れます。			
AND rr,(+SP)	1 1 1 0 0 1 1 0 | 1 0 r r r 1 0 0	Z Z – – – –	5	SP ← SP+1:rr ← rr&(SP)
	16 ビットレジスタ rr の内容と、スタックポインタ SP の内容をインクリメントし、その値で指定されるアドレスから連続する 2 バイトのメモリ内容とでビットごとの論理積を取り、結果をレジスタ rr に入れます。			
AND rr,(PC+A) 注	0 1 0 0 1 1 1 1 | 1 0 r r r 1 0 0	Z Z – – – –	6	rr ← rr&(PC+A)
	16 ビットレジスタ rr の内容と、プログラムカウンタ PC の内容に A レジスタの内容を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容とでビットごとの論理積を取り、結果をレジスタ rr に入れます。			
OR A,n	0 1 1 0 0 1 1 0 | n n n n n n n n	Z Z – – – –	2	A ← A | n
	A レジスタの内容と、オブジェクトコード中の即値 n とでビットごとの論理和を取り、結果を A レジスタに入れます。			
OR g,n	1 1 1 0 1 g g g | 0 1 1 0 0 1 1 0 | n n n n n n n n	Z Z – – – –	3	g ← g | n
	8 ビットレジスタ g の内容と、オブジェクトコード中の即値 n とでビットごとの論理和を取り、結果をレジスタ g に入れます。			
OR gg,mn	1 1 1 0 1 g g g | 0 1 1 0 1 1 1 0 | n n n n n n n n | m m m m m m m m	Z Z – – – –	4	gg ← gg | mn
	16 ビットレジスタ gg の内容と、オブジェクトコード中の即値 mn とでビットごとの論理和を取り、結果をレジスタ gg に入れます。			
OR r,g	1 1 1 0 1 g g g | 0 0 r r r 1 1 0	Z Z – – – –	2	r ← r | g
	8 ビットレジスタ r の内容と、8 ビットレジスタ g の内容とでビットごとの論理和を取り、結果をレジスタ r に入れます。			
OR rr,gg	1 1 1 0 1 g g g | 1 0 r r r 1 1 0	Z Z – – – –	3	rr ← rr | gg
	16 ビットレジスタ rr の内容と、16 ビットレジスタ gg の内容とでビットごとの論理和を取り、結果をレジスタ rr に入れます。			
OR r,(x)	1 1 1 0 0 0 0 0 | x x x x x x x x | 0 0 r r r 1 1 0	Z Z – – – –	4	r ← r | (x)
	8 ビットレジスタ r の内容と、オブジェクトコード中の x で直接指定されるアドレス (0x0000 ~ 0x00FF 番地) のメモリ内容とでビットごとの論理和を取り、結果をレジスタ r に入れます。			
OR r,(vw)	1 1 1 0 0 0 0 1 | w w w w w w w w | v v v v v v v v | 0 0 r r r 1 1 0	Z Z – – – –	5	r ← r | (vw)
	8 ビットレジスタ r の内容と、オブジェクトコード中の vw で直接指定されるアドレス (0x0000 ~ 0xFFFF 番地) のメモリ内容とでビットごとの論理和を取り、結果をレジスタ r に入れます。			
OR r,(DE)	1 1 1 0 0 0 1 0 | 0 0 r r r 1 1 0	Z Z – – – –	3	r ← r | (DE)
	8 ビットレジスタ r の内容と、レジスタペア DE で指定されるアドレスのメモリ内容とでビットごとの論理和を取り、結果をレジスタ r に入れます。			
OR r,(HL)	1 1 1 0 0 0 1 1 | 0 0 r r r 1 1 0	Z Z – – – –	3	r ← r | (HL)
	8 ビットレジスタ r の内容と、レジスタペア HL で指定されるアドレスのメモリ内容とでビットごとの論理和を取り、結果をレジスタ r に入れます。			
OR r,(IX)	1 1 1 0 0 1 0 0 | 0 0 r r r 1 1 0	Z Z – – – –	3	r ← r | (IX)
	8 ビットレジスタ r の内容と、インデックスレジスタ IX で指定されるアドレスのメモリ内容とでビットごとの論理和を取り、結果をレジスタ r に入れます。			
OR r,(IY)	1 1 1 0 0 1 0 1 | 0 0 r r r 1 1 0	Z Z – – – –	3	r ← r | (IY)
	8 ビットレジスタ r の内容と、インデックスレジスタ IY で指定されるアドレスのメモリ内容とでビットごとの論理和を取り、結果をレジスタ r に入れます。			
OR r,(IX+d)	1 1 0 1 0 1 0 0 | d d d d d d d d | 0 0 r r r 1 1 0	Z Z – – – –	5	r ← r | (IX+d)
	8 ビットレジスタ r の内容と、インデックスレジスタ IX の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容とでビットごとの論理和を取り、結果をレジスタ r に入れます。			
OR r,(IY+d)	1 1 0 1 0 1 0 1 | d d d d d d d d | 0 0 r r r 1 1 0	Z Z – – – –	5	r ← r | (IY+d)
	8 ビットレジスタ r の内容と、インデックスレジスタ IY の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容とでビットごとの論理和を取り、結果をレジスタ r に入れます。			

ニモニック	オブジェクトコード (2 進)	フラグ J Z C H S V	サイクル	オペレーション
OR r,(SP+d)	1 1 0 1 0 1 1 0 d d d d d d d d 0 0 r r r 1 1 0	Z Z – – – –	5	r ← r | (SP+d)
	8 ビットレジスタ r の内容と、スタックポインタ SP の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容とでビットごとの論理和を取り、結果をレジスタ r に入れます。			
OR r,(HL+d)	1 1 0 1 0 1 1 1 d d d d d d d d 0 0 r r r 1 1 0	Z Z – – – –	5	r ← r | (HL+d)
	8 ビットレジスタ r の内容と、レジスタペア HL の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容とでビットごとの論理和を取り、結果をレジスタ r に入れます。			
OR r,(HL+C)	1 1 1 0 0 1 1 1 0 0 r r r 1 1 0	Z Z – – – –	5	r ← r | (HL+C)
	8 ビットレジスタ r の内容と、レジスタペア HL の内容に C レジスタの内容を符号拡張して加算した値で指定されるアドレスのメモリ内容とでビットごとの論理和を取り、結果をレジスタ r に入れます。			
OR r,(+SP)	1 1 1 0 0 1 1 0 0 0 r r r 1 1 0	Z Z – – – –	4	SP ← SP+1:r ← r | (SP)
	8 ビットレジスタ r の内容と、スタックポインタ SP の内容をインクリメントし、その値で指定されるアドレスのメモリ内容とでビットごとの論理和を取り、結果をレジスタ r に入れます。			
OR r,(PC+A) 注	0 1 0 0 1 1 1 1 0 0 r r r 1 1 0	Z Z – – – –	5	r ← r | (PC+A)
	8 ビットレジスタ r の内容と、プログラムカウンタ PC の内容に A レジスタの内容を符号拡張して加算した値で指定されるアドレスのメモリ内容とでビットごとの論理和を取り、結果をレジスタ r に入れます。			
OR (x),n	1 1 1 0 0 0 0 0 x x x x x x x x 0 1 1 0 0 1 1 0 n n n n n n n n	Z Z – – – –	6	(x) ← (x) | n
	オブジェクトコード中の x で直接指定されるアドレス (0x0000 ~ 0x00FF 番地) のメモリ内容と、オブジェクトコード中の即値 n とでビットごとの論理和を取り、結果を前記のアドレスに入れます。			
OR (vw),n	1 1 1 0 0 0 0 1 w w w w w w w w v v v v v v v v 0 1 1 0 0 1 1 0 n n n n n n n n	Z Z – – – –	7	(vw) ← (vw) | n
	オブジェクトコード中の vw で直接指定されるアドレス (0x0000 ~ 0xFFFF 番地) のメモリ内容と、オブジェクトコード中の即値 n とでビットごとの論理和を取り、結果を前記のアドレスに入れます。			
OR (DE),n	1 1 1 0 0 0 1 0 0 1 1 0 0 1 1 0 n n n n n n n n	Z Z – – – –	5	(DE) ← (DE) | n
	レジスタペア DE で指定されるアドレスのメモリ内容と、オブジェクトコード中の即値 n とでビットごとの論理和を取り、結果を前記のアドレスに入れます。			
OR (HL),n	1 1 1 0 0 0 1 1 0 1 1 0 0 1 1 0 n n n n n n n n	Z Z – – – –	5	(HL) ← (HL) | n
	レジスタペア HL で指定されるアドレスのメモリ内容と、オブジェクトコード中の即値 n とでビットごとの論理和を取り、結果を前記のアドレスに入れます。			
OR (IX),n	1 1 1 0 0 1 0 0 0 1 1 0 0 1 1 0 n n n n n n n n	Z Z – – – –	5	(IX) ← (IX) | n
	インデックスレジスタ IX で指定されるアドレスのメモリ内容と、オブジェクトコード中の即値 n とでビットごとの論理和を取り、結果を前記のアドレスに入れます。			
OR (IY),n	1 1 1 0 0 1 0 1 0 1 1 0 0 1 1 0 n n n n n n n n	Z Z – – – –	5	(IY) ← (IY) | n
	インデックスレジスタ IY で指定されるアドレスのメモリ内容と、オブジェクトコード中の即値 n とでビットごとの論理和を取り、結果を前記のアドレスに入れます。			
OR (IX+d),n	1 1 0 1 0 1 0 0 d d d d d d d d 0 1 1 0 0 1 1 0 n n n n n n n n	Z Z – – – –	7	(IX+d) ← (IX+d) | n
	インデックスレジスタ IX の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容と、オブジェクトコード中の即値 n とでビットごとの論理和を取り、結果を前記のアドレスに入れます。			
OR (IY+d),n	1 1 0 1 0 1 0 1 d d d d d d d d 0 1 1 0 0 1 1 0 n n n n n n n n	Z Z – – – –	7	(IY+d) ← (IY+d) | n
	インデックスレジスタ IY の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容と、オブジェクトコード中の即値 n とでビットごとの論理和を取り、結果を前記のアドレスに入れます。			
OR (SP+d),n	1 1 0 1 0 1 1 0 d d d d d d d d 0 1 1 0 0 1 1 0 n n n n n n n n	Z Z – – – –	7	(SP+d) ← (SP+d) | n
	スタックポインタ SP の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容と、オブジェクトコード中の即値 n とでビットごとの論理和を取り、結果を前記のアドレスに入れます。			
OR (HL+d),n	1 1 0 1 0 1 1 1 d d d d d d d d 0 1 1 0 0 1 1 0 n n n n n n n n	Z Z – – – –	7	(HL+d) ← (HL+d) | n
	レジスタペア HL の内容に、オブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容と、オブジェクトコード中の即値 n とでビットごとの論理和を取り、結果を前記のアドレスに入れます。			
OR (HL+C),n	1 1 1 0 0 1 1 1 0 1 1 0 0 1 1 0 n n n n n n n n	Z Z – – – –	7	(HL+C) ← (HL+C) | n
	レジスタペア HL の内容に C レジスタの内容を符号拡張して加算した値で指定されるアドレスのメモリ内容と、オブジェクトコード中の即値 n とでビットごとの論理和を取り、結果を前記のアドレスに入れます。			
OR (+SP),n	1 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 n n n n n n n n	Z Z – – – –	6	SP ← SP+1:(SP) ← (SP) | n
	スタックポインタ SP の内容をインクリメントし、その値で指定されるアドレスのメモリ内容と、オブジェクトコード中の即値 n とでビットごとの論理和を取り、結果を前記のアドレスに入れます。			
OR (PC+A),n 注	0 1 0 0 1 1 1 1 0 1 1 0 0 1 1 0 n n n n n n n n	Z Z – – – –	7	(PC+A) ← (PC+A) | n
	プログラムカウンタ PC の内容に A レジスタの内容を符号拡張して加算した値で指定されるアドレスのメモリ内容と、オブジェクトコード中の即値 n とでビットごとの論理和を取り、結果を前記のアドレスに入れます。			
OR rr,(x)	1 1 1 0 0 0 0 0 x x x x x x x x 1 0 r r r 1 1 0	Z Z – – – –	5	rr ← rr | (x)
	16 ビットレジスタ rr の内容と、オブジェクトコード中の x で直接指定されるアドレス (0x0000 ~ 0x00FF 番地) から連続する 2 バイトのメモリ内容とでビットごとの論理和を取り、結果をレジスタ rr に入れます。			
OR rr,(vw)	1 1 1 0 0 0 0 1 w w w w w w w w v v v v v v v v 1 0 r r r 1 1 0	Z Z – – – –	6	rr ← rr | (vw)
	16 ビットレジスタ rr の内容と、オブジェクトコード中の vw で直接指定されるアドレス (0x0000 ~ 0xFFFF 番地) から連続する 2 バイトのメモリ内容とでビットごとの論理和を取り、結果をレジスタ rr に入れます。			
OR rr,(DE)	1 1 1 0 0 0 1 0 1 0 r r r 1 1 0	Z Z – – – –	4	rr ← rr | (DE)
	16 ビットレジスタ rr の内容と、レジスタペア DE で指定されるアドレスから連続する 2 バイトのメモリ内容とでビットごとの論理和を取り、結果をレジスタ rr に入れます。			
OR rr,(HL)	1 1 1 0 0 0 1 1 1 0 r r r 1 1 0	Z Z – – – –	4	rr ← rr | (HL)
	16 ビットレジスタ rr の内容と、レジスタペア HL で指定されるアドレスから連続する 2 バイトのメモリ内容とでビットごとの論理和を取り、結果をレジスタ rr に入れます。			
OR rr,(IX)	1 1 1 0 0 1 0 0 1 0 r r r 1 1 0	Z Z – – – –	4	rr ← rr | (IX)
	16 ビットレジスタ rr の内容と、インデックスレジスタ IX で指定されるアドレスから連続する 2 バイトのメモリ内容とでビットごとの論理和を取り、結果をレジスタ rr に入れます。			

ニモニック	オブジェクトコード (2 進)	フラグ J Z C H S V	サイクル	オペレーション
OR rr,(IY)	1 1 1 0 0 1 0 1 1 0 r r r 1 1 0	Z Z – – – –	4	rr ← rr (IY)
	16 ビットレジスタ rr の内容と、インデックスレジスタ IY で指定されるアドレスから連続する 2 バイトのメモリ内容とでビットごとの論理和を取り、結果をレジスタ rr に入れます。			
OR rr,(IX+d)	1 1 0 1 0 1 0 0 d d d d d d d d 1 0 r r r 1 1 0	Z Z – – – –	6	rr ← rr (IX+d)
	16 ビットレジスタ rr の内容と、インデックスレジスタ IX の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容とでビットごとの論理和を取り、結果をレジスタ rr に入れます。			
OR rr,(IY+d)	1 1 0 1 0 1 0 1 d d d d d d d d 1 0 r r r 1 1 0	Z Z – – – –	6	rr ← rr (IY+d)
	16 ビットレジスタ rr の内容と、インデックスレジスタ IY の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容とでビットごとの論理和を取り、結果をレジスタ rr に入れます。			
OR rr,(SP+d)	1 1 0 1 0 1 1 0 d d d d d d d d 1 0 r r r 1 1 0	Z Z – – – –	6	rr ← rr (SP+d)
	16 ビットレジスタ rr の内容と、スタックポインタ SP の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容とでビットごとの論理和を取り、結果をレジスタ rr に入れます。			
OR rr,(HL+d)	1 1 0 1 0 1 1 1 d d d d d d d d 1 0 r r r 1 1 0	Z Z – – – –	6	rr ← rr (HL+d)
	16 ビットレジスタ rr の内容と、レジスタペア HL の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容とでビットごとの論理和を取り、結果をレジスタ rr に入れます。			
OR rr,(HL+C)	1 1 1 0 0 1 1 1 1 0 r r r 1 1 0	Z Z – – – –	6	rr ← rr (HL+C)
	16 ビットレジスタ rr の内容と、レジスタペア HL の内容に C レジスタの内容を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容とでビットごとの論理和を取り、結果をレジスタ rr に入れます。			
OR rr,(+SP)	1 1 1 0 0 1 1 0 1 0 r r r 1 1 0	Z Z – – – –	5	SP ← SP+1:rr ← rr (SP)
	16 ビットレジスタ rr の内容と、スタックポインタ SP の内容をインクリメントし、その値で指定されるアドレスから連続する 2 バイトのメモリ内容とでビットごとの論理和を取り、結果をレジスタ rr に入れます。			
OR rr,(PC+A) 注	0 1 0 0 1 1 1 1 1 0 r r r 1 1 0	Z Z – – – –	6	rr ← rr (PC+A)
	16 ビットレジスタ rr の内容と、プログラムカウンタ PC の内容に A レジスタの内容を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容とでビットごとの論理和を取り、結果をレジスタ rr に入れます。			
XOR A,n	0 1 1 0 0 1 0 1 n n n n n n n n	Z Z – – – –	2	A ←A^n
	A レジスタの内容と、オブジェクトコード中の即値 n とでビットごとの排他的論理和を取り、結果を A レジスタに入れます。n = 0xFF のときは、1 の補数 (データ反転) 命令になります。 例 : A = 0x69 のとき、XOR A, 0xFF 命令を実行すると、A = 0x96, ZF = 0 となります。			
XOR g,n	1 1 1 0 1 g g g 0 1 1 0 0 1 0 1 n n n n n n n n	Z Z – – – –	3	g ←g^n
	8 ビットレジスタ g の内容と、オブジェクトコード中の即値 n とでビットごとの排他的論理和を取り、結果をレジスタ g に入れます。n = 0xFF のときは、1 の補数 (データ反転) 命令になります。			
XOR gg,mn	1 1 1 0 1 g g g 0 1 1 0 1 1 0 1 n n n n n n n n m m m m m m m m	Z Z – – – –	4	gg ← gg^mn
	16 ビットレジスタ gg の内容と、オブジェクトコード中の即値 mn とでビットごとの排他的論理和を取り、結果をレジスタ gg に入れます。mn = 0xFFFF のときは、1 の補数 (データ反転) 命令になります。			
XOR r,g	1 1 1 0 1 g g g 0 0 r r r 1 0 1	Z Z – – – –	2	r ←r^g
	8 ビットレジスタ r の内容と、8 ビットレジスタ g の内容とでビットごとの排他的論理和を取り、結果をレジスタ r に入れます。			
XOR rr,gg	1 1 1 0 1 g g g 1 0 r r r 1 0 1	Z Z – – – –	3	rr ←rr^gg
	16 ビットレジスタ rr の内容と、16 ビットレジスタ gg の内容とでビットごとの排他的論理和を取り、結果をレジスタ rr に入れます。			
XOR r,(x)	1 1 1 0 0 0 0 0 x x x x x x x x 0 0 r r r 1 0 1	Z Z – – – –	4	r ←r^(x)
	8 ビットレジスタ r の内容と、オブジェクトコード中の x で直接指定されるアドレス (0x0000 ~ 0x00FF 番地) のメモリ内容とでビットごとの排他的論理和を取り、結果をレジスタ r に入れます。			
XOR r,(vw)	1 1 1 0 0 0 0 1 w w w w w w w w v v v v v v v v 0 0 r r r 1 0 1	Z Z – – – –	5	r ← r^(vw)
	8 ビットレジスタ r の内容と、オブジェクトコード中の vw で直接指定されるアドレス (0x0000 ~ 0xFFFF 番地) のメモリ内容とでビットごとの排他的論理和を取り、結果をレジスタ r に入れます。			
XOR r,(DE)	1 1 1 0 0 0 1 0 0 0 r r r 1 0 1	Z Z – – – –	3	r ←r^(DE)
	8 ビットレジスタ r の内容と、レジスタペア DE で指定されるアドレスのメモリ内容とでビットごとの排他的論理和を取り、結果をレジスタ r に入れます。			
XOR r,(HL)	1 1 1 0 0 0 1 1 0 0 r r r 1 0 1	Z Z – – – –	3	r ←r^(HL)
	8 ビットレジスタ r の内容と、レジスタペア HL で指定されるアドレスのメモリ内容とでビットごとの排他的論理和を取り、結果をレジスタ r に入れます。			
XOR r,(IX)	1 1 1 0 0 1 0 0 0 0 r r r 1 0 1	Z Z – – – –	3	r ←r^(IX)
	8 ビットレジスタ r の内容と、インデックスレジスタ IX で指定されるアドレスのメモリ内容とでビットごとの排他的論理和を取り、結果をレジスタ r に入れます。			
XOR r,(IY)	1 1 1 0 0 1 0 1 0 0 r r r 1 0 1	Z Z – – – –	3	r ←r^(IY)
	8 ビットレジスタ r の内容と、インデックスレジスタ IY で指定されるアドレスのメモリ内容とでビットごとの排他的論理和を取り、結果をレジスタ r に入れます。			
XOR r,(IX+d)	1 1 0 1 0 1 0 0 d d d d d d d d 0 0 r r r 1 0 1	Z Z – – – –	5	r ←r^(IX+d)
	8 ビットレジスタ r の内容と、インデックスレジスタ IX の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容とでビットごとの排他的論理和を取り、結果をレジスタ r に入れます。			
XOR r,(IY+d)	1 1 0 1 0 1 0 1 d d d d d d d d 0 0 r r r 1 0 1	Z Z – – – –	5	r ←r^(IY+d)
	8 ビットレジスタ r の内容と、インデックスレジスタ IY の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容とでビットごとの排他的論理和を取り、結果をレジスタ r に入れます。			
XOR r,(SP+d)	1 1 0 1 0 1 1 0 d d d d d d d d 0 0 r r r 1 0 1	Z Z – – – –	5	r ←r^(SP+d)
	8 ビットレジスタ r の内容と、スタックポインタ SP の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容とでビットごとの排他的論理和を取り、結果をレジスタ r に入れます。			
XOR r,(HL+d)	1 1 0 1 0 1 1 1 d d d d d d d d 0 0 r r r 1 0 1	Z Z – – – –	5	r ←r^(HL+d)
	8 ビットレジスタ r の内容と、レジスタペア HL の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容とでビットごとの排他的論理和を取り、結果をレジスタ r に入れます。			
XOR r,(HL+C)	1 1 1 0 0 1 1 1 0 0 r r r 1 0 1	Z Z – – – –	5	r ←r^(HL+C)
	8 ビットレジスタ r の内容と、レジスタペア HL の内容に C レジスタの内容を符号拡張して加算した値で指定されるアドレスのメモリ内容とでビットごとの排他的論理和を取り、結果をレジスタ r に入れます。			
XOR r,(+SP)	1 1 1 0 0 1 1 0 0 0 r r r 1 0 1	Z Z – – – –	4	SP ← SP+1:r ← r^(SP)
	8 ビットレジスタ r の内容と、スタックポインタ SP の内容をインクリメントし、その値で指定されるアドレスのメモリ内容とでビットごとの排他的論理和を取り、結果をレジスタ r に入れます。			

ニモニック	オブジェクトコード (2 進)	フラグ J Z C H S V	サイクル	オペレーション
XOR r,(PC+A) 注	0 1 0 0 1 1 1 1 0 0 r r r 1 0 1	Z Z – – – –	5	r ←r^(PC+A)
	8 ビットレジスタ r の内容と、プログラムカウンタ PC の内容に A レジスタの内容を符号拡張して加算した値で指定されるアドレスのメモリ内容とでビットごとの排他的論理和を取り、結果をレジスタ r に入れます。			
XOR (x),n	1 1 1 0 0 0 0 0 x x x x x x x x 0 1 1 0 0 1 0 1 n n n n n n n n	Z Z – – – –	6	(x) ← (x)^n
	オブジェクトコード中の x で直接指定されるアドレス (0x0000 ~ 0x00FF 番地) のメモリ内容と、オブジェクトコード中の即値 n とでビットごとの排他的論理和を取り、結果を前記のアドレスに入れます。			
XOR (vw),n	1 1 1 0 0 0 0 1 w w w w w w w w v v v v v v v v 0 1 1 0 0 1 0 1 n n n n n n n n	Z Z – – – –	7	(vw) ← (vw)^n
	オブジェクトコード中の vw で直接指定されるアドレス (0x0000 ~ 0xFFFF 番地) のメモリ内容と、オブジェクトコード中の即値 n とでビットごとの排他的論理和を取り、結果を前記のアドレスに入れます。			
XOR (DE),n	1 1 1 0 0 0 1 0 0 1 1 0 0 1 0 1 n n n n n n n n	Z Z – – – –	5	(DE) ← (DE)^n
	レジスタペア DE で指定されるアドレスのメモリ内容と、オブジェクトコード中の即値 n とでビットごとの排他的論理和を取り、結果を前記のアドレスに入れます。			
XOR (HL),n	1 1 1 0 0 0 1 1 0 1 1 0 0 1 0 1 n n n n n n n n	Z Z – – – –	5	(HL) ← (HL)^n
	レジスタペア HL で指定されるアドレスのメモリ内容と、オブジェクトコード中の即値 n とでビットごとの排他的論理和を取り、結果を前記のアドレスに入れます。			
XOR (IX),n	1 1 1 0 0 1 0 0 0 1 1 0 0 1 0 1 n n n n n n n n	Z Z – – – –	5	(IX) ← (IX)^n
	インデックスレジスタ IX で指定されるアドレスのメモリ内容と、オブジェクトコード中の即値 n とでビットごとの排他的論理和を取り、結果を前記のアドレスに入れます。			
XOR (IY),n	1 1 1 0 0 1 0 1 0 1 1 0 0 1 0 1 n n n n n n n n	Z Z – – – –	5	(IY) ← (IY)^n
	インデックスレジスタ IY で指定されるアドレスのメモリ内容と、オブジェクトコード中の即値 n とでビットごとの排他的論理和を取り、結果を前記のアドレスに入れます。			
XOR (IX+d),n	1 1 0 1 0 1 0 0 d d d d d d d d 0 1 1 0 0 1 0 1 n n n n n n n n	Z Z – – – –	7	(IX+d) ← (IX+d)^n
	インデックスレジスタ IX の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容と、オブジェクトコード中の即値 n とでビットごとの排他的論理和を取り、結果を前記のアドレスに入れます。			
XOR (IY+d),n	1 1 0 1 0 1 0 1 d d d d d d d d 0 1 1 0 0 1 0 1 n n n n n n n n	Z Z – – – –	7	(IY+d) ← (IY+d)^n
	インデックスレジスタ IY の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容と、オブジェクトコード中の即値 n とでビットごとの排他的論理和を取り、結果を前記のアドレスに入れます。			
XOR (SP+d),n	1 1 0 1 0 1 1 0 d d d d d d d d 0 1 1 0 0 1 0 1 n n n n n n n n	Z Z – – – –	7	(SP+d) ← (SP+d)^n
	スタックポインタ SP の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容と、オブジェクトコード中の即値 n とでビットごとの排他的論理和を取り、結果を前記のアドレスに入れます。			
XOR (HL+d),n	1 1 0 1 0 1 1 1 d d d d d d d d 0 1 1 0 0 1 0 1 n n n n n n n n	Z Z – – – –	7	(HL+d) ← (HL+d)^n
	レジスタペア HL の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容と、オブジェクトコード中の即値 n とでビットごとの排他的論理和を取り、結果を前記のアドレスに入れます。			
XOR (HL+C),n	1 1 1 0 0 1 1 1 0 1 1 0 0 1 0 1 n n n n n n n n	Z Z – – – –	7	(HL+C) ← (HL+C)^n
	レジスタペア HL の内容に C レジスタの内容を符号拡張して加算した値で指定されるアドレスのメモリ内容と、オブジェクトコード中の即値 n とでビットごとの排他的論理和を取り、結果を前記のアドレスに入れます。			
XOR (+SP),n	1 1 1 0 0 1 1 0 0 1 1 0 0 1 0 1 n n n n n n n n	Z Z – – – –	6	SP ← SP+1:(SP) ← (SP)^n
	スタックポインタ SP の内容をインクリメントし、その値で指定されるアドレスのメモリ内容と、オブジェクトコード中の即値 n とでビットごとの排他的論理和を取り、結果を前記のアドレスに入れます。			
XOR (PC+A),n 注	0 1 0 0 1 1 1 1 0 1 1 0 0 1 0 1 n n n n n n n n	Z Z – – – –	7	(PC+A) ← (PC+A)^n
	プログラムカウンタ PC の内容に A レジスタの内容を符号拡張して加算した値で指定されるアドレスのメモリ内容と、オブジェクトコード中の即値 n とでビットごとの排他的論理和を取り、結果を前記のアドレスに入れます。			
XOR rr,(x)	1 1 1 0 0 0 0 0 x x x x x x x x 1 0 r r r 1 0 1	Z Z – – – –	5	rr ← rr^(x)
	16 ビットレジスタ rr の内容と、オブジェクトコード中の x で直接指定されるアドレス (0x0000 ~ 0x00FF 番地) から連続する 2 バイトのメモリ内容とでビットごとの排他的論理和を取り、結果をレジスタ rr に入れます。			
XOR rr,(vw)	1 1 1 0 0 0 0 1 w w w w w w w w v v v v v v v v 1 0 r r r 1 0 1	Z Z – – – –	6	rr ← rr^(vw)
	16 ビットレジスタ rr の内容と、オブジェクトコード中の vw で直接指定されるアドレス (0x0000 ~ 0xFFFF 番地) から連続する 2 バイトのメモリ内容とでビットごとの排他的論理和を取り、結果をレジスタ rr に入れます。			
XOR rr,(DE)	1 1 1 0 0 0 1 0 1 0 r r r 1 0 1	Z Z – – – –	4	rr ← rr^(DE)
	16 ビットレジスタ rr の内容と、レジスタペア DE で指定されるアドレスから連続する 2 バイトのメモリ内容とでビットごとの排他的論理和を取り、結果をレジスタ rr に入れます。			
XOR rr,(HL)	1 1 1 0 0 0 1 1 1 0 r r r 1 0 1	Z Z – – – –	4	rr ← rr^(HL)
	16 ビットレジスタ rr の内容と、レジスタペア HL で指定されるアドレスから連続する 2 バイトのメモリ内容とでビットごとの排他的論理和を取り、結果をレジスタ rr にいれます。			
XOR rr,(IX)	1 1 1 0 0 1 0 0 1 0 r r r 1 0 1	Z Z – – – –	4	rr ← rr^(IX)
	16 ビットレジスタ rr の内容と、インデックスレジスタ IX で指定されるアドレスから連続する 2 バイトのメモリ内容とでビットごとの排他的論理和を取り、結果をレジスタ rr に入れます。			
XOR rr,(IY)	1 1 1 0 0 1 0 1 1 0 r r r 1 0 1	Z Z – – – –	4	rr ← rr^(IY)
	16 ビットレジスタ rr の内容と、インデックスレジスタ IY で指定されるアドレスから連続する 2 バイトのメモリ内容とでビットごとの排他的論理和を取り、結果をレジスタ rr に入れます。			
XOR rr,(IX+d)	1 1 0 1 0 1 0 0 d d d d d d d d 1 0 r r r 1 0 1	Z Z – – – –	6	rr ← rr^(IX+d)
	16 ビットレジスタ rr の内容と、インデックスレジスタ IX の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容とでビットごとの排他的論理和を取り、結果をレジスタ rr に入れます。			
XOR rr,(IY+d)	1 1 0 1 0 1 0 1 d d d d d d d d 1 0 r r r 1 0 1	Z Z – – – –	6	rr ← rr^(IY+d)
	16 ビットレジスタ rr の内容と、インデックスレジスタ IY の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容とでビットごとの排他的論理和を取り、結果をレジスタ rr に入れます。			
XOR rr,(SP+d)	1 1 0 1 0 1 1 0 d d d d d d d d 1 0 r r r 1 0 1	Z Z – – – –	6	rr ← rr^(SP+d)
	16 ビットレジスタ rr の内容と、スタックポインタ SP の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容とでビットごとの排他的論理和を取り、結果をレジスタ rr に入れます。			

ニモニック	オブジェクトコード (2 進)	フラグ J Z C H S V	サイクル	オペレーション
XOR rr,(HL+d)	1 1 0 1 0 1 1 1 d d d d d d d d 1 0 r r r 1 0 1	Z Z – – – –	6	rr ← rr^(HL+d)
	16 ビットレジスタ rr の内容と、レジスタペア HL にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容とでビットごとの排他的論理和を取り、結果をレジスタ rr に入れます。			
XOR rr,(HL+C)	1 1 1 0 0 1 1 1 1 0 r r r 1 0 1	Z Z – – – –	6	rr ← rr^(HL+C)
	16 ビットレジスタ rr の内容と、レジスタペア HL の内容に C レジスタの内容を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容とでビットごとの排他的論理和を取り、結果をレジスタ rr に入れます。			
XOR rr,(+SP)	1 1 1 0 0 1 1 0 1 0 r r r 1 0 1	Z Z – – – –	5	SP ← SP+1:rr ← rr^(SP)
	16 ビットレジスタ rr の内容と、スタックポインタ SP の内容をインクリメントし、その値で指定されるアドレスから連続する 2 バイトのメモリ内容とでビットごとの排他的論理和を取り、結果をレジスタに入れます。			
XOR rr,(PC+A) 注	0 1 0 0 1 1 1 1 1 0 r r r 1 0 1	Z Z – – – –	6	rr ← rr^(PC+A)
	16 ビットレジスタ rr の内容と、プログラムカウンタ PC の内容に A レジスタの内容を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容とでビットごとの排他的論理和を取り、結果をレジスタ rr に入れます。			
INC r	0 0 1 0 0 r r r	C Z – – – –	1	r ← r+1
	8 ビットレジスタ r の内容をインクリメントします。オーバフローするとジャンプ ステータス フラグ , ゼロフラグがともに “1” にセットされます。キャリーフラグは変化しません。 例 : L = 0xFF のとき、INC L 命令を実行すると、L = 0x00, ZF = 1, JF = 1 となります。			
INC rr	0 0 1 1 0 r r r	C Z – – – –	2	rr ← rr+1
	16 ビットレジスタ rr の内容をインクリメントします。オーバフローするとジャンプ ステータス フラグ , ゼロフラグがともに “1” にセットされます。 例 : HL = 0x1234 のとき、INC HL 命令を実行すると、HL = 0x1235, ZF = 0, JF = 0 となります。			
INC (x)	1 1 1 0 0 0 0 0 x x x x x x x x 1 1 1 1 0 0 0 0	C Z – – – –	5	(x) ← (x)+1
	オブジェクトコード中の x で直接指定されるアドレス (0x0000 ~ 0x00FF 番地) のメモリ内容をインクリメントします。オーバフローするとジャンプ ステータス フラグ , ゼロフラグがともに “1” にセットされます。			
INC (vw)	1 1 1 0 0 0 0 1 w w w w w w w w v v v v v v v v 1 1 1 1 0 0 0 0	C Z – – – –	6	(vw) ← (vw)+1
	オブジェクトコード中の vw で直接指定されるアドレス (0x0000 ~ 0xFFFF 番地) のメモリ内容をインクリメントします。オーバフローするとジャンプ ステータス フラグ , ゼロフラグがともに “1” にセットされます。			
INC (DE)	1 1 1 0 0 0 1 0 1 1 1 1 0 0 0 0	C Z – – – –	4	(DE) ← (DE)+1
	レジスタペア DE で指定されるアドレスのメモリ内容をインクリメントします。オーバフローするとジャンプ ステータス フラグ , ゼロフラグがともに “1” にセットされます。			
INC (HL)	1 1 1 0 0 0 1 1 1 1 1 1 0 0 0 0	C Z – – – –	4	(HL) ← (HL)+1
	レジスタペア HL で指定されるアドレスのメモリ内容をインクリメントします。オーバフローするとジャンプ ステータス フラグ , ゼロフラグがともに “1” にセットされます。			
INC (IX)	1 1 1 0 0 1 0 0 1 1 1 1 0 0 0 0	C Z – – – –	4	(IX) ← (IX)+1
	インデックスレジスタ IX で指定されるアドレスのメモリ内容をインクリメントします。オーバフローするとジャンプ ステータス フラグ , ゼロフラグがともに “1” にセットされます。			
INC (IY)	1 1 1 0 0 1 0 1 1 1 1 1 0 0 0 0	C Z – – – –	4	(IY) ← (IY)+1
	インデックスレジスタ IY で指定されるアドレスのメモリ内容をインクリメントします。オーバフローするとジャンプ ステータス フラグ , ゼロフラグがともに “1” にセットされます。			
INC (IX+d)	1 1 0 1 0 1 0 0 d d d d d d d d 1 1 1 1 0 0 0 0	C Z – – – –	6	(IX+d) ← (IX+d)+1
	インデックスレジスタ IX の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容をインクリメントします。オーバフローするとジャンプ ステータス フラグ , ゼロフラグがともに “1” にセットされます。			
INC (IY+d)	1 1 0 1 0 1 0 1 d d d d d d d d 1 1 1 1 0 0 0 0	C Z – – – –	6	(IY+d) ← (IY+d)+1
	インデックスレジスタ IY の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容をインクリメントします。オーバフローするとジャンプ ステータス フラグ , ゼロフラグがともに “1” にセットされます。			
INC (SP+d)	1 1 0 1 0 1 1 0 d d d d d d d d 1 1 1 1 0 0 0 0	C Z – – – –	6	(SP+d) ← (SP+d)+1
	スタックポインタ SP の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容をインクリメントします。オーバフローするとジャンプ ステータス フラグ , ゼロフラグがともに “1” にセットされます。			
INC (HL+d)	1 1 0 1 0 1 1 1 d d d d d d d d 1 1 1 1 0 0 0 0	C Z – – – –	6	(HL+d) ← (HL+d)+1
	レジスタペア HL の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容をインクリメントします。オーバフローするとジャンプ ステータス フラグ , ゼロフラグがともに “1” にセットされます。			
INC (HL+C)	1 1 1 0 0 1 1 1 1 1 1 1 0 0 0 0	C Z – – – –	6	(HL+C) ← (HL+C)+1
	レジスタペア HL の内容に C レジスタの内容を符号拡張して加算した値で指定されるアドレスのメモリ内容をインクリメントします。オーバフローするとジャンプ ステータス フラグ , ゼロフラグがともに “1” にセットされます。			
INC (+SP)	1 1 1 0 0 1 1 0 1 1 1 1 0 0 0 0	C Z – – – –	5	SP ← SP+1:(SP) ← (SP)+1
	スタックポインタ SP の内容をインクリメントし、その値で指定されるアドレスのメモリ内容をインクリメントします。メモリの内容がオーバフローするとジャンプ ステータス フラグ , ゼロフラグがともに “1” にセットされます。			
INC (PC+A) 注	0 1 0 0 1 1 1 1 1 1 1 1 0 0 0 0	C Z – – – –	6	(PC+A) ← (PC+A)+1
	プログラムカウンタ PC の内容に A レジスタの内容を符号拡張して加算した値で指定されるアドレスのメモリ内容をインクリメントします。オーバフローするとジャンプ ステータス フラグ , ゼロフラグがともに “1” にセットされます。			
DEC r	0 0 1 0 1 r r r	C Z – – – –	1	r ← r–1
	8 ビットレジスタ r の内容をデクリメントします。アンダーフローする (結果が 0xFF となる) とジャンプ ステータス フラグが “1” にセットされます。キャリーフラグは変化しません。 例 : L = 0x00 のとき、DEC L 命令を実行すると、L = 0xFF, ZF = 0, JF = 1 となります。			
DEC rr	0 0 1 1 1 r r r	C Z – – – –	2	rr ← rr–1
	16 ビットレジスタ rr の内容をデクリメントします。アンダーフローする (結果が 0xFFFF となる) とジャンプ ステータス フラグが “1” にセットされます。 例 : HL = 0x8765 のとき、DEC HL 命令を実行すると、HL = 0x8764, ZF = 0, JF = 0 となります。			
DEC (x)	1 1 1 0 0 0 0 0 x x x x x x x x 1 1 1 1 1 0 0 0	C Z – – – –	5	(x) ← (x)–1
	オブジェクトコード中の x で直接指定されるアドレス (0x0000 ~ 0x00FF 番地) のメモリ内容をデクリメントします。アンダーフローするとジャンプ ステータス フラグが “1” にセットされます。			
DEC (vw)	1 1 1 0 0 0 0 1 w w w w w w w w v v v v v v v v 1 1 1 1 1 0 0 0	C Z – – – –	6	(vw) ← (vw)–1
	オブジェクトコード中の vw で直接指定されるアドレス (0x0000 ~ 0xFFFF 番地) のメモリ内容をデクリメントします。アンダーフローするとジャンプ ステータス フラグが “1” にセットされます。			

ニモニック	オブジェクトコード (2 進)	フラグ J Z C H S V	サイクル	オペレーション
DEC (DE)	1 1 1 0 0 0 1 0 | 1 1 1 1 1 0 0 0	C Z – – – –	4	(DE) ← (DE)–1
	レジスタペア DE で指定されるアドレスのメモリ内容をデクリメントします。アンダーフローするとジャンプ ステータス フラグが “1” にセットされます。			
DEC (HL)	1 1 1 0 0 0 1 1 | 1 1 1 1 1 0 0 0	C Z – – – –	4	(HL) ← (HL)–1
	レジスタペア HL で指定されるアドレスのメモリ内容をデクリメントします。アンダーフローするとジャンプ ステータス フラグが “1” にセットされます。			
DEC (IX)	1 1 1 0 0 1 0 0 | 1 1 1 1 1 0 0 0	C Z – – – –	4	(IX) ← (IX)–1
	インデックスレジスタ IX で指定されるアドレスのメモリ内容をデクリメントします。アンダーフローするとジャンプ ステータス フラグが “1” にセットされます。			
DEC (IY)	1 1 1 0 0 1 0 1 | 1 1 1 1 1 0 0 0	C Z – – – –	4	(IY) ← (IY)–1
	インデックスレジスタ IY で指定されるアドレスのメモリ内容をデクリメントします。アンダーフローするとジャンプ ステータス フラグが “1” にセットされます。			
DEC (IX+d)	1 1 0 1 0 1 0 0 | d d d d d d d d | 1 1 1 1 1 0 0 0	C Z – – – –	6	(IX+d) ← (IX+d)–1
	インデックスレジスタ IX の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容をデクリメントします。アンダーフローするとジャンプ ステータス フラグが “1” にセットされます。			
DEC (IY+d)	1 1 0 1 0 1 0 1 | d d d d d d d d | 1 1 1 1 1 0 0 0	C Z – – – –	6	(IY+d) ← (IY+d)–1
	インデックスレジスタ IY の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容をデクリメントします。アンダーフローするとジャンプ ステータス フラグが “1” にセットされます。			
DEC (SP+d)	1 1 0 1 0 1 1 0 | d d d d d d d d | 1 1 1 1 1 0 0 0	C Z – – – –	6	(SP+d) ← (SP+d)–1
	スタックポインタ SP の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容をデクリメントします。アンダーフローするとジャンプ ステータス フラグが “1” にセットされます。			
DEC (HL+d)	1 1 0 1 0 1 1 1 | d d d d d d d d | 1 1 1 1 1 0 0 0	C Z – – – –	6	(HL+d) ← (HL+d)–1
	レジスタペア HL の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスのメモリ内容をデクリメントします。アンダーフローするとジャンプ ステータス フラグが “1” にセットされます。			
DEC (HL+C)	1 1 1 0 0 1 1 1 | 1 1 1 1 1 0 0 0	C Z – – – –	6	(HL+C) ← (HL+C)–1
	レジスタペア HL の内容に C レジスタの内容を符号拡張して加算した値で指定されるアドレスのメモリ内容をデクリメントします。アンダーフローするとジャンプ ステータス フラグが “1” にセットされます。			
DEC (+SP)	1 1 1 0 0 1 1 0 | 1 1 1 1 1 0 0 0	C Z – – – –	5	SP ← SP+1:(SP) ← (SP)–1
	スタックポインタ SP の内容をデクリメントし、その値で指定されるアドレスの内容をデクリメントします。アンダーフローするとジャンプ ステータス フラグが “1” にセットされます。			
DEC (PC+A) 注	0 1 0 0 1 1 1 1 | 1 1 1 1 1 0 0 0	C Z – – – –	6	(PC+A) ← (PC+A)–1
	プログラムカウンタ PC の内容に A レジスタの内容を符号拡張して加算した値で指定されるアドレスのメモリ内容をデクリメントします。アンダーフローするとジャンプ ステータス フラグが “1” にセットされます。			
DAA g	1 1 1 0 1 g g g | 1 1 0 1 1 0 1 0	C Z C H – –	2	Decimal adjustment against g (after addition)

8 ビットのパックド BCD データの加算の際、加算命令 (ADD/ADDC) 実行後のレジスタ g の内容を十進補正します。
多桁の BCD データの加算を行う場合は、8 ビット単位で下位桁から加算、補正を繰り返します。

補正前のキャリーフラグ	補正前のレジスタgの上位4ビット	補正前のハーフキャリーフラグ	補正前のレジスタgの下位4ビット	補正のために加算される値	補正後のキャリーフラグ
0	0~9	0	0~9	00	0
0	0~8	0	A~F	06	0
0	0~9	1	0~3	06	0
0	A~F	0	0~9	60	1
0	9~F	0	A~F	66	1
0	A~F	1	0~3	66	1
1	0~2	0	0~9	60	1
1	0~2	0	A~F	66	1
1	0~3	1	0~3	66	1

例 : A = 0x26, B = 0x57 のとき、ADD A, B 命令を実行すると、A = 0x7D, CF = 0, HF = 0 となり、この状態で、DAA A 命令を実行すると、A = 0x83, CF = 0 となります。

DAS g	1 1 1 0 1 g g g | 1 1 0 1 1 0 1 1	C Z C H – –	2	Decimal adjustment against g (after subtraction)

8 ビットのパックド BCD データの減算の際、減算命令 (SUB/SUBB) 実行後のレジスタ g の内容を十進補正します。
多桁の BCD データの加算を行う場合は、8 ビット単位で下位桁から加算、補正を繰り返します。

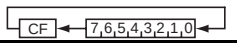
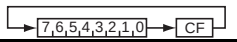
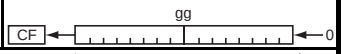
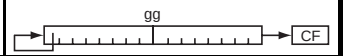
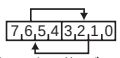
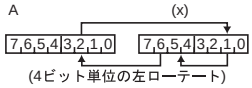
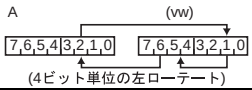
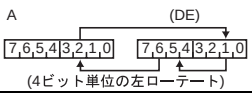
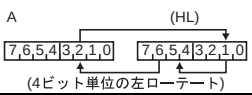
補正前のキャリーフラグ	補正前のレジスタgの上位4ビット	補正前のハーフキャリーフラグ	補正前のレジスタgの下位4ビット	補正のために加算される値	補正後のキャリーフラグ
0	0~9	0	0~9	00	0
0	0~8	1	6~F	FA	0
1	7~F	0	0~9	A0	1
1	6~F	1	6~F	9A	1

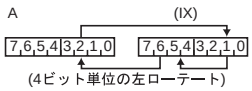
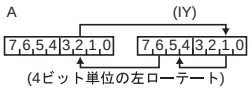
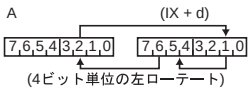
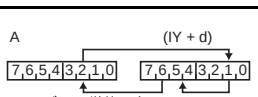
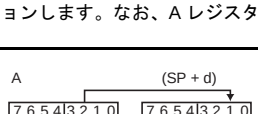
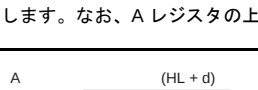
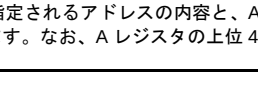
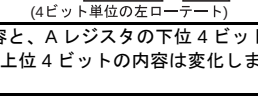
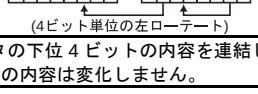
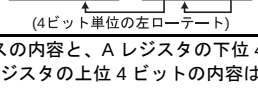
例 : A = 0x87, B = 0x39 のとき、SUB A, B 命令を実行すると、A = 0x4E, CF = 0, HF = 1 となり、この状態で、DAS A 命令を実行すると、A = 0x48, CF = 0 となります。

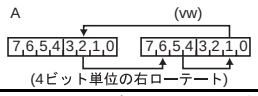
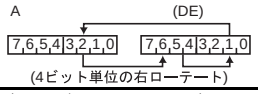
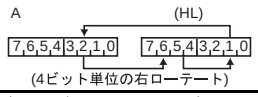
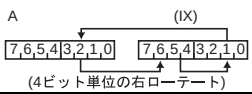
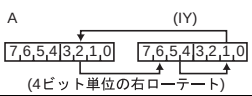
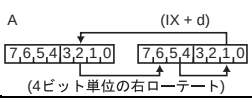
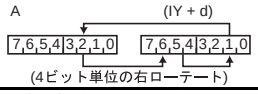
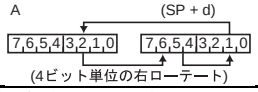
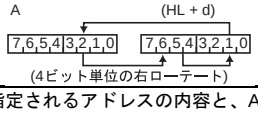
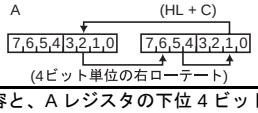
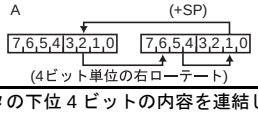
ニモニック	オブジェクトコード (2 進)	フラグ J Z C H S V	サイクル	オペレーション
MUL W,A	1 1 1 0 1 0 0 0 | 1 1 1 1 0 0 1 0	Z Z – – – –	13	WA ← W×A
	W レジスタの内容 (符号なし整数) に A レジスタの内容 (符号なし整数) を掛け、結果をレジスタペア WA に入れます。ゼロフラグは、結果の上位 8 ビット (W レジスタに格納される値) が 0x00 のとき “1” にセットされ、0x00 以外のとき “0” にクリアされます。 例 1: W = 0x87, A = 0xF2 のとき、この命令を実行すると、WA = 0x7F9E, ZF = 0 となります。 例 2: W = 0x16, A = 0x05 のとき、この命令を実行すると、WA = 0x006E, ZF = 1 となります。			
MUL B,C	1 1 1 0 1 0 0 1 | 1 1 1 1 0 0 1 0	Z Z – – – –	13	BC ← B×C
	B レジスタの内容 (符号なし整数) に C レジスタの内容 (符号なし整数) を掛け、結果をレジスタペア BC に入れます。結果の上位 8 ビット (B レジスタに格納される値) が 0x00 のとき、ゼロフラグが “1” にセットされます。			
MUL D,E	1 1 1 0 1 0 1 0 | 1 1 1 1 0 0 1 0	Z Z – – – –	13	DE ← D×E
	D レジスタの内容 (符号なし整数) に E レジスタの内容 (符号なし整数) を掛け、結果をレジスタペア DE に入れます。結果の上位 8 ビット (D レジスタに格納される値) が 0x00 のとき、ゼロフラグが “1” にセットされます。			
MUL H,L	1 1 1 0 1 0 1 1 | 1 1 1 1 0 0 1 0	Z Z – – – –	13	HL ← H×L
	H レジスタの内容 (符号なし整数) に L レジスタの内容 (符号なし整数) を掛け、結果をレジスタペア HL に入れます。結果の上位 8 ビット (H レジスタに格納される値) が 0x00 のとき、ゼロフラグが “1” にセットされます。			
DIV WA,C	1 1 1 0 1 0 0 0 | 1 1 1 1 0 0 1 1	Z Z C – – –	13	A ← WA÷C, W ← 余り
	レジスタペア WA の内容 (符号なし整数) を C レジスタの内容 (符号なし整数) で割り、商を A レジスタに、余りを W レジスタにそれぞれ入れます。除数 (C レジスタの内容) が 0x00 のとき、または商が 0x100 以上のときキャリーフラグは “1” にセットされ (この場合、A レジスタおよび W レジスタの内容は不定になります) 、それ以外のときキャリーフラグは “0” にクリアされます。また、ゼロフラグは、余りが 0x00 のとき “1” にセットされ、0x00 以外のとき “0” にクリアされます。なお、C レジスタの内容は変化しません。 例 1: WA = 0x1234, C = 0x56 のとき、この命令を実行すると、 A = 0x36, W = 0x10, CF = 0, ZF = 0 となります。 例 2: WA = 0x4830, C = 0x9A のとき、この命令を実行すると、 A = 0x78, W = 0x00, CF = 0, ZF = 1 となります。 例 3: WA = 0x3210, C = 0x27 のとき、この命令を実行すると、 A = 0x48, W = 0x18, CF = 1, ZF = 0 となります。			
DIV DE,C	1 1 1 0 1 0 1 0 | 1 1 1 1 0 0 1 1	Z Z C – – –	13	E ← DE÷C, D ← 余り
	レジスタペア DE の内容 (符号なし整数) を C レジスタの内容 (符号なし整数) で割り、商の下位 8 ビットを E レジスタに、余りを D レジスタにそれぞれ入れます。除数 (C レジスタの内容) が 0x00 のとき、または商が 0x100 以上のとき、キャリーフラグが “1” にセットされます (この場合、レジスタペア DE の内容は不定になります) 。また、余りが 0x00 のとき、ゼロフラグが “1” にセットされます。なお、C レジスタの内容は変化しません。			
DIV HL,C	1 1 1 0 1 0 1 1 | 1 1 1 1 0 0 1 1	Z Z C – – –	13	L ← HL÷C, H ← 余り
	レジスタペア HL の内容 (符号なし整数) を C レジスタの内容 (符号なし整数) で割り、商の下位 8 ビットを L レジスタに、余りを H レジスタにそれぞれ入れます。除数 (C レジスタの内容) が 0x00 のとき、または商が 0x100 以上のとき、キャリーフラグが “1” にセットされます (この場合、レジスタペア HL の内容は不定になります) 。また、余りが 0x00 のとき、ゼロフラグが “1” にセットされます。なお、C レジスタの内容は変化しません。			
NEG CS,gg	1 1 1 0 1 g g g | 1 1 1 1 1 0 1 0	1 – – – – –	3	if CF=1 then gg ← 0–gg else null
	キャリーフラグが “1” のとき、16 ビットレジスタ gg の 2 の補数を取りレジスタ gg に格納し、ジャンプステータスフラグを “1” にセットします。 キャリーフラグが “0” のときは、ジャンプステータスフラグを “1” にセットし、次命令の実行に移ります (この場合、レジスタ gg の内容は変化しません)。 例 1: HL = 0x5678, CF = 1 のとき、NEG CS, HL 命令を実行すると、HL = 0xA988, CF = 1 となります。 例 2: DE = 0x89AB, CF = 0 のとき、NEG CS, DE 命令を実行すると、DE = 0x89AB, CF = 0 となります。			

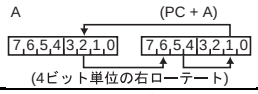
注) (PC+A) を使用した命令には制限があります。詳細は、TLCS-870/C1 シリーズ 命令セット 1.4 アドレッシングモードを参照してください。

2.3 シフト、ローテート、ニブル処理

ニモニック	オブジェクトコード (2 進)	フラグ J Z C H S V	サイクル	オペレーション
SHLC g	1 1 1 0 1 g g g | 1 1 1 1 0 1 0 0	C Z * – – –	2	
	8 ビットレジスタ g の内容を左へ 1 ビットずつ論理シフトします (レジスタ g の最下位ビットには “0” が入ります。キャリーフラグには、レジスタ g の最上位ビットの内容が入ります) 。シフト後のレジスタ g の内容が 0x00 のとき、ゼロフラグが “1” にセットされます。 例 : A = 0y00111011, CF = 1 のとき、この命令を実行すると、A = 0y01110110, CF = 0, ZF = 0 となります。			
SHRC g	1 1 1 0 1 g g g | 1 1 1 1 0 1 0 1	C Z * – – –	2	
	8 ビットレジスタ g の内容を右へ 1 ビットずつ論理シフトします (レジスタ g の最上位ビットには “0” が入ります。キャリーフラグには、レジスタ g の最下位ビットの内容が入ります) 。シフト後のレジスタ g の内容が 0x00 のとき、ゼロフラグが “1” にセットされます。 例 : A = 0y01011101, CF = 0 のとき、この命令を実行すると、A = 0y00101110, CF = 1, ZF = 0 となります。			
ROLC g	1 1 1 0 1 g g g | 1 1 1 1 0 1 1 0	C Z * – – –	2	
	8 ビットレジスタ g とキャリーフラグの内容を左へ 1 ビットずつローテーションします。ローテーション後のレジスタ g の内容が 0x00 のとき、ゼロフラグが “1” にセットされます。 例 1: A = 0y10010110, CF = 0 のとき、この命令を実行すると、A = 0y00101100, CF = 1, JF = 1, ZF = 0 となります。 例 2: A = 0y10000000, CF = 0 のとき、この命令を実行すると、A = 0y00000000, CF = 1, JF = 1, ZF = 1 となります。			
RORC g	1 1 1 0 1 g g g | 1 1 1 1 0 1 1 1	C Z * – – –	2	
	8 ビットレジスタ g とキャリーフラグの内容を右へ 1 ビットずつローテーションします。ローテーション後のレジスタ g の内容が 0x00 のとき、ゼロフラグが “1” にセットされます。 例 : A = 0y01101101, CF = 1 のとき、この命令を実行すると、A = 0y10110110, CF = 1, ZF = 0 となります。			
SHLCA gg	1 1 1 0 1 g g g | 1 1 1 1 0 0 0 0	C Z * – S V	3	
	16 ビットレジスタ gg の内容を左へ 1 ビットずつ算術シフトします (レジスタの最下位ビットには “0” が入ります。キャリーフラグには、レジスタ gg の最上位ビットの内容が入ります)。シフト後のレジスタ gg の内容が 0x0000 のとき、ゼロフラグが “1” にセットされます。レジスタ gg の最上位ビットの内容がシフトで変化するとオーバフローフラグが “1” にセットされます。 例 : HL = 0x3456, CF = 1 のとき、SHLCA HL 命令を実行すると、HL = 0x68AC, CF = 0, JF = 0, ZF = 0, SF = 0, VF = 0 となります。			
SHRCA gg	1 1 1 0 1 g g g | 1 1 1 1 0 0 0 1	C Z * – S 0	3	
	16 ビットレジスタ gg の内容を右へ 1 ビットずつ算術シフトします (レジスタの最上位ビットには “0” は変化しません。キャリーフラグには、レジスタ gg の最下位ビットの内容が入ります)。シフト後のレジスタ gg の内容が 0x0000 のとき、ゼロフラグが “1” にセットされます。 例 : DE = 0x89AB, CF = 0 のとき、SHRCA DE 命令を実行すると、DE = 0xC4D5, CF = 1, JF = 0, ZF = 0, SF = 1, VF = 0 となります。			
SWAP g	1 1 1 0 1 g g g | 1 1 1 1 1 1 1 1	1 – – – – –	7	
	8 ビットレジスタ g の上位 4 ビットと下位 4 ビットの内容を交換します。 例 : A = 0x25 のとき、この命令を実行すると、A = 0x52 となります。			
ROLD A,(x)	1 1 1 0 0 0 0 0 | x x x x x x x x | 1 1 1 1 0 1 1 0	1 – – – – –	9	
	オブジェクトコード中の x で直接指定されるアドレス (0x0000 ~ 0x00FF 番地) の内容と、A レジスタの下位 4 ビットの内容を連結した 12 ビットデータを左へ 4 ビット単位にローテーションします。なお、A レジスタの上位 4 ビットの内容は変化しません。 例 : A レジスタが 0x12, 0x0087 番地の内容が 0x56 のとき、ROLD A, (0x87) 命令を実行すると、 A レジスタが 0x15, 0x0087 番地の内容が 0x62 となります。			
ROLD A,(vw)	1 1 1 0 0 0 0 1 | w w w w w w w w | v v v v v v v v | 1 1 1 1 0 1 1 0	1 – – – – –	10	
	オブジェクトコード中の vw で直接指定されるアドレスの (0x0000 ~ 0xFFFF 番地) 内容と、A レジスタの下位 4 ビットの内容を連結した 12 ビットデータを左へ 4 ビット単位にローテーションします。なお、A レジスタの上位 4 ビットの内容は変化しません。			
ROLD A,(DE)	1 1 1 0 0 0 1 0 | 1 1 1 1 0 1 1 0	1 – – – – –	8	
	レジスタペア DE で指定されるアドレスの内容と、A レジスタの下位 4 ビットの内容を連結した 12 ビットデータを左へ 4 ビット単位にローテーションします。なお、A レジスタの上位 4 ビットの内容は変化しません。			
ROLD A,(HL)	1 1 1 0 0 0 1 1 | 1 1 1 1 0 1 1 0	1 – – – – –	8	
	レジスタペア HL で指定されるアドレスの内容と、A レジスタの下位 4 ビットの内容を連結した 12 ビットデータを左へ 4 ビット単位にローテーションします。なお、A レジスタの上位 4 ビットの内容は変化しません。			

ニモニック	オブジェクトコード (2 進)			フラグ J Z C H S V	サイクル	オペレーション
ROLD A,(IX)	1 1 1 0 0 1 0 0	1 1 1 1 0 1 1 0		1 − − − − −	8	
	インデックスレジスタ IX で指定されるアドレスの内容と、A レジスタの下位 4 ビットの内容を連結した 12 ビットデータを左へ 4 ビット単位にローテーションします。なお、A レジスタの上位 4 ビットの内容は変化しません。					
ROLD A,(IY)	1 1 1 0 0 1 0 1	1 1 1 1 0 1 1 0		1 − − − − −	8	
	インデックスレジスタ IY で指定されるアドレスの内容と、A レジスタの下位 4 ビットの内容を連結した 12 ビットデータを左へ 4 ビット単位にローテーションします。なお、A レジスタの上位 4 ビットの内容は変化しません。					
ROLD A,(IX+d)	1 1 0 1 0 1 0 0	d d d d d d d d	1 1 1 1 0 1 1 0	1 − − − − −	10	
	インデックスレジスタ IX の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスの内容と、A レジスタの下位 4 ビットの内容を連結した 12 ビットデータを左へ 4 ビット単位にローテーションします。なお、A レジスタの上位 4 ビットの内容は変化しません。					
ROLD A,(IY+d)	1 1 0 1 0 1 0 1	d d d d d d d d	1 1 1 1 0 1 1 0	1 − − − − −	10	
	インデックスレジスタ IY の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスの内容と、A レジスタの下位 4 ビットの内容を連結した 12 ビットデータを左へ 4 ビット単位にローテーションします。なお、A レジスタの上位 4 ビットの内容は変化しません。					
ROLD A,(SP+d)	1 1 0 1 0 1 1 0	d d d d d d d d	1 1 1 1 0 1 1 0	1 − − − − −	10	
	スタックポインタ SP の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスの内容と、A レジスタの下位 4 ビットの内容を連結した 12 ビットデータを左へ 4 ビット単位にローテーションします。なお、A レジスタの上位 4 ビットの内容は変化しません。					
ROLD A,(HL+d)	1 1 0 1 0 1 1 1	d d d d d d d d	1 1 1 1 0 1 1 0	1 − − − − −	10	
	レジスタペア HL の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスの内容と、A レジスタの下位 4 ビットの内容を連結した 12 ビットデータを左へ 4 ビット単位にローテーションします。なお、A レジスタの上位 4 ビットの内容は変化しません。					
ROLD A,(HL+C)	1 1 1 0 0 1 1 1	1 1 1 1 0 1 1 0		1 − − − − −	10	
	レジスタペア HL の内容に C レジスタの内容を符号拡張して加算した値で指定されるアドレスの内容と、A レジスタの下位 4 ビットの内容を連結した 12 ビットデータを左へ 4 ビット単位にローテーションします。なお、A レジスタの上位 4 ビットの内容は変化しません。					
ROLD A,(+SP)	1 1 1 0 0 1 1 0	1 1 1 1 0 1 1 0		1 − − − − −	9	
	スタックポインタ SP をインクリメントし、その値で指定されるアドレスのメモリ内容と A レジスタの下位 4 ビットの内容を連結した 12 ビットデータを左へ 4 ビット単位にローテーションします。なお、A レジスタの上位 4 ビットの内容は変化しません。					
ROLD A,(PC+A) 注	0 1 0 0 1 1 1 1	1 1 1 1 0 1 1 0		1 − − − − −	10	
	プログラムカウンタ PC の内容に A レジスタの内容を符号拡張して加算した値で指定されるアドレスの内容と、A レジスタの下位 4 ビットの内容を連結した 12 ビットデータを左へ 4 ビット単位にローテーションします。なお、A レジスタの上位 4 ビットの内容は変化しません。					
RORD A,(x)	1 1 1 0 0 0 0 0	x x x x x x x x	1 1 1 1 0 1 1 1	1 − − − − −	9	
	オブジェクトコード中の x で直接指定されるアドレス (0x0000 ~ 0x00FF 番地) の内容と、A レジスタの下位 4 ビットの内容を連結した 12 ビットデータを右へ 4 ビット単位にローテーションします。なお、A レジスタの上位 4 ビットの内容は変化しません。 例 :A レジスタが 0x12, 0x0087 番地の内容が 0x56 のとき、RORD A, (0x87) 命令を実行すると、 A レジスタが 0x16, 0x0087 番地の内容が 0x25 となります。					

ニモニック	オブジェクトコード (2 進)				フラグ J Z C H S V	サイクル	オペレーション
RORD A,(vw)	1 1 1 0 0 0 0 1 1 1 1 1 0 1 1 1	w w w w w w w w	v v v v v v v v		1 − − − − −	10	
オブジェクトコード中の vw で直接指定されるアドレス (0x0000 ~ 0xFFFF 番地) の内容と、A レジスタの下位 4 ビットの内容を連結した 12 ビットデータを右へ 4 ビット単位にローテーションします。なお、A レジスタの上位 4 ビットの内容は変化しません。							
RORD A,(DE)	1 1 1 0 0 0 1 0	1 1 1 1 0 1 1 1			1 − − − − −	8	
レジスタペア DE で指定されるアドレスの内容と、A レジスタの下位 4 ビットの内容を連結した 12 ビットデータを右へ 4 ビット単位にローテーションします。なお、A レジスタの上位 4 ビットの内容は変化しません。							
RORD A,(HL)	1 1 1 0 0 0 1 1	1 1 1 1 0 1 1 1			1 − − − − −	8	
レジスタペア HL で指定されるアドレスの内容と、A レジスタの下位 4 ビットの内容を連結した 12 ビットデータを右へ 4 ビット単位にローテーションします。なお、A レジスタの上位 4 ビットの内容は変化しません。							
RORD A,(IX)	1 1 1 0 0 1 0 0	1 1 1 1 0 1 1 1			1 − − − − −	8	
インデックスレジスタ IX で指定されるアドレスの内容と、A レジスタの下位 4 ビットの内容を連結した 12 ビットデータを右へ 4 ビット単位にローテーションします。なお、A レジスタの上位 4 ビットの内容は変化しません。							
RORD A,(IY)	1 1 1 0 0 1 0 1	1 1 1 1 0 1 1 1			1 − − − − −	8	
インデックスレジスタ IY で指定されるアドレスの内容と、A レジスタの下位 4 ビットの内容を連結した 12 ビットデータを右へ 4 ビット単位にローテーションします。なお、A レジスタの上位 4 ビットの内容は変化しません。							
RORD A,(IX+d)	1 1 0 1 0 1 0 0	d d d d d d d d	1 1 1 1 0 1 1 1		1 − − − − −	10	
インデックスレジスタ IX の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスの内容と、A レジスタの下位 4 ビットの内容を連結した 12 ビットデータを右へ 4 ビット単位にローテーションします。なお、A レジスタの上位 4 ビットの内容は変化しません。							
RORD A,(IY+d)	1 1 0 1 0 1 0 1	d d d d d d d d	1 1 1 1 0 1 1 1		1 − − − − −	10	
インデックスレジスタ IY の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスの内容と、A レジスタの下位 4 ビットの内容を連結した 12 ビットデータを右へ 4 ビット単位にローテーションします。なお、A レジスタの上位 4 ビットの内容は変化しません。							
RORD A,(SP+d)	1 1 0 1 0 1 1 0	d d d d d d d d	1 1 1 1 0 1 1 1		1 − − − − −	10	
スタックポインタ SP の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスの内容と、A レジスタの下位 4 ビットの内容を連結した 12 ビットデータを右へ 4 ビット単位にローテーションします。なお、A レジスタの上位 4 ビットの内容は変化しません。							
RORD A,(HL+d)	1 1 0 1 0 1 1 1	d d d d d d d d	1 1 1 1 0 1 1 1		1 − − − − −	10	
レジスタペア HL の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスの内容と、A レジスタの下位 4 ビットの内容を連結した 12 ビットデータを右へ 4 ビット単位にローテーションします。なお、A レジスタの上位 4 ビットの内容は変化しません。							
RORD A,(HL+C)	1 1 1 0 0 1 1 1	1 1 1 1 0 1 1 1			1 − − − − −	10	
レジスタペア HL の内容に C レジスタの内容を符号拡張して加算した値で指定されるアドレスの内容と、A レジスタの下位 4 ビットの内容を連結した 12 ビットデータを右へ 4 ビット単位にローテーションします。なお、A レジスタの上位 4 ビットの内容は変化しません。							
RORD A,(+SP)	1 1 1 0 0 1 1 0	1 1 1 1 0 1 1 1			1 − − − − −	9	
スタックポイントをインクリメントし、その値で指定されるアドレスのメモリの内容と、A レジスタの下位 4 ビットの内容を連結した 12 ビットデータを右へ 4 ビット単位にローテーションします。なお、A レジスタの上位 4 ビットの内容は変化しません。							

ニモニック	オブジェクトコード (2 進)	フラグ J Z C H S V	サイクル	オペレーション
RORD A,(PC+A) 注	0 1 0 0　1 1 1 1 | 1 1 1 1　0 1 1 1	1 − − − − −	10	
	プログラムカウンタ PC の内容に A レジスタの内容を符号拡張して加算した値で指定されるアドレスの内容と、A レジスタの下位 4 ビットの内容を連結した 12 ビットデータを右へ 4 ビット単位にローテーションします。なお、A レジスタの上位 4 ビットの内容は変化しません。			

注) (PC+A) を使用した命令には制限があります。詳細は、TLCS-870/C1 シリーズ 命令セット 1.4 アドレッシングモードを参照してください。

2.4 ビット操作、フラグ操作

ニモニック	オブジェクトコード (2 進)	フラグ J Z C H S V	サイクル	オペレーション
SET g.b	1 1 1 0 1 g g g | 1 1 0 0 0 b b b	Z * – – – –	3	ZF ← $\overline{\text{g.b}}$:g.b ← 1
	8 ビットレジスタ g の、オブジェクトコード中の b で指定されるビットの内容の反転値をゼロフラグに入れた後、そのビットの内容を “1” にセットします。 例 : A = 0x3C のとき、SET A.7 命令を実行すると、ZF = 1, A = 0xBC となります。			
SET (x).b	1 1 0 0 0 b b b | x x x x x x x x	Z * – – – –	4	ZF ← $\overline{\text{(x).b}}$:(x).b ← 1
	オブジェクトコード中の x で直接指定されるアドレス (0x0000 ~ 0x00FF 番地) の、オブジェクトコード中の b で指定されるビットの内容の反転値をゼロフラグに入れた後、そのビットの内容を “1” にセットします。			
SET (vw).b	1 1 1 0 0 0 0 1 | w w w w w w w w | v v v v v v v v 1 1 0 0 0 b b b	Z * – – – –	6	ZF ← $\overline{\text{(vw).b}}$:(vw).b ← 1
	オブジェクトコード中の vw で直接指定されるアドレス (0x0000 ~ 0xFFFF 番地) の、オブジェクトコード中の b で指定されるビットの内容の反転値をゼロフラグに入れた後、そのビットの内容を “1” にセットします。			
SET (DE).b	1 1 1 0 0 0 1 0 | 1 1 0 0 0 b b b	Z * – – – –	4	ZF ← $\overline{\text{(DE).b}}$:(DE).b ← 1
	レジスタペア DE で指定されるアドレスの、オブジェクトコード中の b で指定されるビットの内容の反転値をゼロフラグに入れた後、そのビットの内容を “1” にセットします。			
SET (HL).b	1 1 1 0 0 0 1 1 | 1 1 0 0 0 b b b	Z * – – – –	4	ZF ← $\overline{\text{(HL).b}}$:(HL).b ← 1
	レジスタペア HL で指定されるアドレスの、オブジェクトコード中の b で指定されるビットの内容の反転値をゼロフラグに入れた後、そのビットの内容を “1” にセットします。			
SET (IX).b	1 1 1 0 0 1 0 0 | 1 1 0 0 0 b b b	Z * – – – –	4	ZF ← $\overline{\text{(IX).b}}$:(IX).b ← 1
	インデックスレジスタ IX で指定されるアドレスの、オブジェクトコード中の b で指定されるビットの内容の反転値をゼロフラグに入れた後、そのビットの内容を “1” にセットします。			
SET (IY).b	1 1 1 0 0 1 0 1 | 1 1 0 0 0 b b b	Z * – – – –	4	ZF ← $\overline{\text{(IY).b}}$:(IY).b ← 1
	インデックスレジスタ IY で指定されるアドレスの、オブジェクトコード中の b で指定されるビットの内容の反転値をゼロフラグに入れた後、そのビットの内容を “1” にセットします。			
SET (IX+d).b	1 1 0 1 0 1 0 0 | d d d d d d d d | 1 1 0 0 0 b b b	Z * – – – –	6	ZF ← $\overline{\text{(IX+d).b}}$:(IX+d).b ← 1
	インデックスレジスタ IX の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスの、オブジェクトコード中の b で指定されるビットの内容の反転値をゼロフラグに入れた後、そのビットの内容を “1” にセットします。			
SET (IY+d).b	1 1 0 1 0 1 0 1 | d d d d d d d d | 1 1 0 0 0 b b b	Z * – – – –	6	ZF ← $\overline{\text{(IY+d).b}}$:(IY+d).b ← 1
	インデックスレジスタ IY の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスの、オブジェクトコード中の b で指定されるビットの内容の反転値をゼロフラグに入れた後、そのビットの内容を “1” にセットします。			
SET (SP+d).b	1 1 0 1 0 1 1 0 | d d d d d d d d | 1 1 0 0 0 b b b	Z * – – – –	6	ZF ← $\overline{\text{(SP+d).b}}$:(SP+d).b ← 1
	スタックポインタの内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスの、オブジェクトコード中の b で指定されるビットの内容の反転値をゼロフラグに入れた後、そのビットの内容を “1” にセットします。			
SET (HL+d).b	1 1 0 1 0 1 1 1 | d d d d d d d d | 1 1 0 0 0 b b b	Z * – – – –	6	ZF ← $\overline{\text{(HL+d).b}}$:(HL+d).b ← 1
	レジスタペア HL の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスの、オブジェクトコード中の b で指定されるビットの内容の反転値をゼロフラグに入れた後、そのビットの内容を “1” にセットします。			
SET (HL+C).b	1 1 1 0 0 1 1 1 | 1 1 0 0 0 b b b	Z * – – – –	6	ZF ← $\overline{\text{(HL+C).b}}$:(HL+C).b ← 1
	レジスタペア HL の内容に C レジスタの内容を符号拡張して加算した値で指定されるアドレスの、オブジェクトコード中の b で指定されるビットの内容の反転値をゼロフラグに入れた後、そのビットの内容を “1” にセットします。			
SET (+SP).b	1 1 1 0 0 1 1 0 | 1 1 0 0 0 b b b	Z * – – – –	5	SP ← SP+1:ZF ← $\overline{\text{(SP).b}}$:(SP).b ← 1
	スタックポインタ SP の内容をインクリメントし、その値で指定されるアドレスの、オブジェクトコード中の b で指定されるビットの内容の反転値をゼロフラグに入れた後、そのビットの内容を “1” にセットします。			
SET (PC+A).b 注	0 1 0 0 1 1 1 1 | 1 1 0 0 0 b b b	Z * – – – –	6	ZF ← $\overline{\text{(PC+A).b}}$:(PC+A).b ← 1
	プログラムカウンタ PC の内容に A レジスタの内容を符号拡張して加算した値で指定されるアドレスの、オブジェクトコード中の b で指定されるビットの内容の反転値をゼロフラグに入れた後、そのビットの内容を “1” にセットします。			
SET (x).A	1 1 1 0 0 0 0 0 | x x x x x x x x | 1 1 1 1 0 0 1 0	Z * – – – –	5	ZF ← $\overline{\text{(x).A}}$:(x).A ← 1
	オブジェクトコード中の x で直接指定されるアドレス (0x0000 ~ 0x00FF 番地) の、A レジスタの下位 3 ビットで指定されるビットの内容の反転値をゼロフラグに入れた後、そのビットの内容を “1” にセットします。			
SET (vw).A	1 1 1 0 0 0 0 1 | w w w w w w w w | v v v v v v v v 1 1 1 1 0 0 1 0	Z * – – – –	6	ZF ← $\overline{\text{(vw).A}}$:(vw).A ← 1
	オブジェクトコード中の vw で直接指定されるアドレス (0x0000 ~ 0xFFFF 番地) の、A レジスタの下位 3 ビットで指定されるビットの内容の反転値をゼロフラグに入れた後、そのビットの内容を “1” にセットします。			
SET (DE).A	1 1 1 0 0 0 1 0 | 1 1 1 1 0 0 1 0	Z * – – – –	4	ZF ← $\overline{\text{(DE).A}}$:(DE).A ← 1
	レジスタペア DE で指定されるアドレスの、A レジスタの下位 3 ビットで指定されるビットの反転値をゼロフラグに入れた後、そのビットの内容を “1” にセットします。			
SET (HL).A	1 1 1 0 0 0 1 1 | 1 1 1 1 0 0 1 0	Z * – – – –	4	ZF ← $\overline{\text{(HL).A}}$:(HL).A ← 1
	レジスタペア HL で指定されるアドレスの、A レジスタの下位 3 ビットで指定されるビットの反転値をゼロフラグに入れた後、そのビットの内容を “1” にセットします。			
SET (IX).A	1 1 1 0 0 1 0 0 | 1 1 1 1 0 0 1 0	Z * – – – –	4	ZF ← $\overline{\text{(IX).A}}$:(IX).A ← 1
	インデックスレジスタ IX で指定されるアドレスの、A レジスタの下位 3 ビットで指定されるビットの反転値をゼロフラグに入れた後、そのビットの内容を “1” にセットします。			
SET (IY).A	1 1 1 0 0 1 0 1 | 1 1 1 1 0 0 1 0	Z * – – – –	4	ZF ← $\overline{\text{(IY).A}}$:(IY).A ← 1
	インデックスレジスタ IY で指定されるアドレスの、A レジスタの下位 3 ビットで指定されるビットの反転値をゼロフラグに入れた後、そのビットの内容を “1” にセットします。			
SET (IX+d).A	1 1 0 1 0 1 0 0 | d d d d d d d d | 1 1 1 1 0 0 1 0	Z * – – – –	6	ZF ← $\overline{\text{(IX+d).A}}$:(IX+d).A ← 1
	インデックスレジスタ IX レジスタペアの内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスの、A レジスタの下位 3 ビットで指定されるビットの内容の反転値をゼロフラグに入れた後、そのビットの内容を “1” にセットします。			

ニモニック	オブジェクトコード (2 進)	フラグ J Z C H S V	サイクル	オペレーション
SET (IY+d).A	1 1 0 1 0 1 0 1 d d d d d d d d 1 1 1 1 0 0 1 0	Z * – – – –	6	ZF ← $\overline{(IY+d).A}$:(IY+d).A ← 1
	インデックスレジスタ IY レジスタペアの内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスの、A レジスタの下位 3 ビットで指定されるビットの内容の反転値をゼロフラグに入れた後、そのビットの内容を “1” にセットします。			
SET (SP+d).A	1 1 0 1 0 1 1 0 d d d d d d d d 1 1 1 1 0 0 1 0	Z * – – – –	6	ZF ← $\overline{(SP+d).A}$:(SP+d).A ← 1
	スタックポインタ SP の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスの、A レジスタの下位 3 ビットで指定されるビットの内容の反転値をゼロフラグに入れた後、そのビットの内容を “1” にセットします。			
SET (HL+d).A	1 1 0 1 0 1 1 1 d d d d d d d d 1 1 1 1 0 0 1 0	Z * – – – –	6	ZF ← $\overline{(HL+d).A}$:(HL+d).A ← 1
	レジスタペア HL の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスの、A レジスタの下位 3 ビットで指定されるビットの内容の反転値をゼロフラグに入れた後、そのビットの内容を “1” にセットします。			
SET (HL+C).A	1 1 1 0 0 1 1 1 1 1 1 1 0 0 1 0	Z * – – – –	6	ZF ← $\overline{(HL+C).A}$:(HL+C).A ← 1
	レジスタペア HL の内容に C レジスタの内容を符号拡張して加算した値で指定されるアドレスの、A レジスタの下位 3 ビットで指定されるビットの内容の反転値をゼロフラグに入れた後、そのビットの内容を “1” にセットします。			
SET (+SP).A	1 1 1 0 0 1 1 0 1 1 1 1 0 0 1 0	Z * – – – –	5	SP ← SP+1:ZF ← $\overline{(SP).A}$:(SP).A ← 1
	スタックポインタ SP の内容をインクリメントし、その値で指定されるアドレスの A レジスタの下位 3 ビットで指定されるビット内容の反転値をゼロフラグに入れた後、そのビットの内容を “1” にセットします。			
SET (PC+A).A 注	0 1 0 0 1 1 1 1 1 1 1 1 0 0 1 0	Z * – – – –	6	ZF ← $\overline{(PC+A).A}$:(PC+A).A ← 1
	プログラムカウンタ PC の内容に A レジスタの内容を符号拡張して加算した値で指定されるアドレスの、A レジスタの下位 3 ビットで指定されるビットの内容の反転値をゼロフラグに入れた後、そのビットの内容を “1” にセットします。			
CLR g.b	1 1 1 0 1 g g g 1 1 0 0 1 b b b	Z * – – – –	3	ZF ← $\overline{g.b}$:g.b ← 0
	8 ビットレジスタ g の、オブジェクトコード中の b で指定されるビットの内容の反転値をゼロフラグに入れた後、そのビットの内容を “0” にクリアします。 例 : A = 0x3C のとき、CLR A.2 命令を実行すると、ZF = 0, A = 0x38 となります。			
CLR (x).b	1 1 0 0 1 b b b x x x x x x x x	Z * – – – –	4	ZF ← $\overline{(x).b}$:(x).b ← 0
	オブジェクトコード中の x で直接指定されるアドレス (0x0000 ~ 0x00FF 番地) の、オブジェクトコード中の b で指定されるビットの内容の反転値をゼロフラグに入れた後、そのビットの内容を “0” にクリアします。			
CLR (vw).b	1 1 1 0 0 0 0 1 w w w w w w w w v v v v v v v v 1 1 0 0 1 b b b	Z * – – – –	6	ZF ← $\overline{(vw).b}$:(vw).b ← 0
	オブジェクトコード中の vw で直接指定されるアドレス (0x0000 ~ 0xFFFF 番地) の、オブジェクトコード中の b で指定されるビットの内容の反転値をゼロフラグに入れた後、そのビットの内容を “0” にクリアします。			
CLR (DE).b	1 1 1 0 0 0 1 0 1 1 0 0 1 b b b	Z * – – – –	4	ZF ← $\overline{(DE).b}$:(DE).b ← 0
	レジスタペア DE で指定されるアドレスの、オブジェクトコード中の b で指定されるビットの内容の反転値をゼロフラグに入れた後、そのビットの内容を “0” にクリアします。			
CLR (HL).b	1 1 1 0 0 0 1 1 1 1 0 0 1 b b b	Z * – – – –	4	ZF ← $\overline{(HL).b}$:(HL).b ← 0
	レジスタペア HL で指定されるアドレスの、オブジェクトコード中の b で指定されるビットの内容の反転値をゼロフラグに入れた後、そのビットの内容を “0” にクリアします。			
CLR (IX).b	1 1 1 0 0 1 0 0 1 1 0 0 1 b b b	Z * – – – –	4	ZF ← $\overline{(IX).b}$:(IX).b ← 0
	インデックスレジスタ IX で指定されるアドレスの、オブジェクトコード中の b で指定されるビットの内容の反転値をゼロフラグに入れた後、そのビットの内容を “0” にクリアします。			
CLR (IY).b	1 1 1 0 0 1 0 1 1 1 0 0 1 b b b	Z * – – – –	4	ZF ← $\overline{(IY).b}$:(IY).b ← 0
	インデックスレジスタ IY で指定されるアドレスの、オブジェクトコード中の b で指定されるビットの内容の反転値をゼロフラグに入れた後、そのビットの内容を “0” にクリアします。			
CLR (IX+d).b	1 1 0 1 0 1 0 0 d d d d d d d d 1 1 0 0 1 b b b	Z * – – – –	6	ZF ← $\overline{(IX+d).b}$:(IX+d).b ← 0
	インデックスレジスタ IX の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスの、オブジェクトコード中の b で指定されるビットの内容の反転値をゼロフラグに入れた後、そのビットの内容を “0” にクリアします。			
CLR (IY+d).b	1 1 0 1 0 1 0 1 d d d d d d d d 1 1 0 0 1 b b b	Z * – – – –	6	ZF ← $\overline{(IY+d).b}$:(IY+d).b ← 0
	インデックスレジスタ IY の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスの、オブジェクトコード中の b で指定されるビットの内容の反転値をゼロフラグに入れた後、そのビットの内容を “0” にクリアします。			
CLR (SP+d).b	1 1 0 1 0 1 1 0 d d d d d d d d 1 1 0 0 1 b b b	Z * – – – –	6	ZF ← $\overline{(SP+d).b}$:(SP+d).b ← 0
	スタックポインタ SP の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスの、オブジェクトコード中の b で指定されるビットの内容の反転値をゼロフラグに入れた後、そのビットの内容を “0” にクリアします。			
CLR (HL+d).b	1 1 0 1 0 1 1 1 d d d d d d d d 1 1 0 0 1 b b b	Z * – – – –	6	ZF ← $\overline{(HL+d).b}$:(HL+d).b ← 0
	レジスタペア HL の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスの、オブジェクトコード中の b で指定されるビットの内容の反転値をゼロフラグに入れた後、そのビットの内容を “0” にクリアします。			
CLR (HL+C).b	1 1 1 0 0 1 1 1 1 1 0 0 1 b b b	Z * – – – –	6	ZF ← $\overline{(HL+C).b}$:(HL+C).b ← 0
	レジスタペア HL の内容に C レジスタの内容を符号拡張して加算した値で指定されるアドレスの、オブジェクトコード中の b で指定されるビットの内容の反転値をゼロフラグに入れた後、そのビットの内容を “0” にクリアします。			
CLR (+SP).b	1 1 1 0 0 1 1 0 1 1 0 0 1 b b b	Z * – – – –	5	SP ← SP+1:ZF ← $\overline{(SP).b}$:(SP).b ← 0
	スタックポインタ SP の内容をインクリメントし、その値で指定されるアドレスの、オブジェクトコード中の b で指定されるビットの内容の反転値をゼロフラグに入れた後、そのビットの内容を “0” にクリアします。			
CLR (PC+A).b 注	0 1 0 0 1 1 1 1 1 1 0 0 1 b b b	Z * – – – –	6	ZF ← $\overline{(PC+A).b}$:(PC+A).b ← 0
	プログラムカウンタ PC の内容に、A レジスタの内容を符号拡張して加算した値で指定されるアドレスの、オブジェクトコード中の b で指定されるビットの内容の反転値をゼロフラグに入れた後、そのビットの内容を “0” にクリアします。			
CLR (x).A	1 1 1 0 0 0 0 0 x x x x x x x x 1 1 1 1 1 0 1 0	Z * – – – –	5	ZF ← $\overline{(x).A}$:(x).A ← 0
	オブジェクトコード中の x で直接指定されるアドレス (0x0000 ~ 0x00FF 番地) の、A レジスタの下位 3 ビットで指定されるビットの内容の反転値をゼロフラグに入れた後、そのビットの内容を “0” にクリアします。			
CLR (vw).A	1 1 1 0 0 0 0 1 w w w w w w w w v v v v v v v v 1 1 1 1 1 0 1 0	Z * – – – –	6	ZF ← $\overline{(vw).A}$:(vw).A ← 0
	オブジェクトコード中の vw で直接指定されるアドレス (0x0000 ~ 0x00FF 番地) の、A レジスタの下位 3 ビットで指定されるビットの内容の反転値をゼロフラグに入れた後、そのビットの内容を “0” にクリアします。			

ニモニック	オブジェクトコード (2 進)					フラグ J Z C H S V	サイクル	オペレーション
CLR (DE).A	1 1 1 0 0 0 1 0	1 1 1 1 1 0 1 0				Z * – – – –	4	ZF ← (DE).A:(DE).A ← 0
	レジスタペア DE で指定されるアドレスの、A レジスタの下位 3 ビットで指定されるビットの内容の反転値をゼロフラグに入れた後、そのビットの内容を “0” にクリアします。							
CLR (HL).A	1 1 1 0 0 0 1 1	1 1 1 1 1 0 1 0				Z * – – – –	4	ZF ← (HL).A:(HL).A ← 0
	レジスタペア HL で指定されるアドレスの、A レジスタの下位 3 ビットで指定されるビットの内容の反転値をゼロフラグに入れた後、そのビットの内容を “0” にクリアします。							
CLR (IX).A	1 1 1 0 0 1 0 0	1 1 1 1 1 0 1 0				Z * – – – –	4	ZF ← (IX).A:(IX).A ← 0
	インデックスレジスタ IX で指定されるアドレスの、A レジスタの下位 3 ビットで指定されるビットの内容の反転値をゼロフラグに入れた後、そのビットの内容を “0” にクリアします。							
CLR (IY).A	1 1 1 0 0 1 0 1	1 1 1 1 1 0 1 0				Z * – – – –	4	ZF ← (IY).A:(IY).A ← 0
	インデックスレジスタ IY で指定されるアドレスの、A レジスタの下位 3 ビットで指定されるビットの内容の反転値をゼロフラグに入れた後、そのビットの内容を “0” にクリアします。							
CLR (IX+d).A	1 1 0 1 0 1 0 0	d d d d d d d d	1 1 1 1 1 0 1 0			Z * – – – –	6	ZF ← (IX+d).A:(IX+d).A ← 0
	インデックスレジスタ IX の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスの、A レジスタの下位 3 ビットで指定されるビットの内容の反転値をゼロフラグに入れた後、そのビットの内容を “0” にクリアします。							
CLR (IY+d).A	1 1 0 1 0 1 0 1	d d d d d d d d	1 1 1 1 1 0 1 0			Z * – – – –	6	ZF ← (IY+d).A:(IY+d).A ← 0
	インデックスレジスタ IY の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスの、A レジスタの下位 3 ビットで指定されるビットの内容の反転値をゼロフラグに入れた後、そのビットの内容を “0” にクリアします。							
CLR (SP+d).A	1 1 0 1 0 1 1 0	d d d d d d d d	1 1 1 1 1 0 1 0			Z * – – – –	6	ZF ← (SP+d).A:(SP+d).A ← 0
	スタックポインタ SP の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスの、A レジスタの下位 3 ビットで指定されるビットの内容の反転値をゼロフラグに入れた後、そのビットの内容を “0” にクリアします。							
CLR (HL+d).A	1 1 0 1 0 1 1 1	d d d d d d d d	1 1 1 1 1 0 1 0			Z * – – – –	6	ZF ← (HL+d).A:(HL+d).A ← 0
	レジスタペア HL の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスの、A レジスタの下位 3 ビットで指定されるビットの内容の反転値をゼロフラグに入れた後、そのビットの内容を “0” にクリアします。							
CLR (HL+C).A	1 1 1 0 0 1 1 1	1 1 1 1 1 0 1 0				Z * – – – –	6	ZF ← (HL+C).A:(HL+C).A ← 0
	レジスタペア HL の内容に C レジスタの内容を符号拡張して加算した値で指定されるアドレスの、A レジスタの下位 3 ビットで指定されるビットの内容の反転値をゼロフラグに入れた後、そのビットの内容を “0” にクリアします。							
CLR (+SP).A	1 1 1 0 0 1 1 0	1 1 1 1 1 0 1 0				Z * – – – –	5	SP ← SP+1:ZF ← (SP).A:(SP).A ← 0
	SP レジスタペアの内容をインクリメントし、その値で指定されるアドレスの、A レジスタの下位 3 ビットで指定されるビットの内容の反転値をゼロフラグに入れた後、そのビットの内容を “0” にクリアします。							
CLR (PC+A).A 注	0 1 0 0 1 1 1 1	1 1 1 1 1 0 1 0				Z * – – – –	6	ZF ← (PC+A).A:(PC+A).A ← 0
	プログラムカウンタ PC の内容に、A レジスタの内容を符号拡張して加算した値で指定されるアドレスの、A レジスタの下位 3 ビットで指定されるビットの内容の反転値をゼロフラグに入れた後、そのビットの内容を “0” にクリアします。							
LD CF,g.b	1 1 1 0 1 g g g	0 1 0 1 1 b b b				C – * – – –	2	CF ← g.b
	8 ビットレジスタ g の、オブジェクトコード中の b で指定されるビットの内容をキャリーフラグに入れます。 例 : A = 0y01101101 のとき、LD CF, A.4 命令を実行すると、CF = 0, JF = 1 となります。							
LD CF,(x).b	0 1 0 1 1 b b b	x x x x x x x x				C – * – – –	3	CF ← (x).b
	オブジェクトコード中の x で直接 指定されるアドレス (0x0000 ~ 0x0FF 番地) の、オブジェクトコード中の b で指定されるビットの内容をキャリーフラグにロードします。							
LD CF,(vw).b	1 1 1 0 0 0 0 1 0 1 0 1 1 b b b	w w w w w w w w	v v v v v v v v			C – * – – –	5	CF ← (vw).b
	オブジェクトコード中の vw で直接指定されるアドレス (0x0000 ~ 0xFFFF 番地) の、オブジェクトコード中の b で指定されるビットの内容をキャリーフラグにロードします。							
LD CF,(DE).b	1 1 1 0 0 0 1 0	0 1 0 1 1 b b b				C – * – – –	3	CF ← (DE).b
	レジスタペア DE で指定されるアドレスの、オブジェクトコード中の b で指定されるビットの内容をキャリーフラグにロードします。							
LD CF,(HL).b	1 1 1 0 0 0 1 1	0 1 0 1 1 b b b				C – * – – –	3	CF ← (HL).b
	レジスタペア HL で指定されるアドレスの、オブジェクトコード中の b で指定されるビットの内容をキャリーフラグにロードします。							
LD CF,(IX).b	1 1 1 0 0 1 0 0	0 1 0 1 1 b b b				C – * – – –	3	CF ← (IX).b
	インデックスレジスタ IX で指定されるアドレスの、オブジェクトコード中の b で指定されるビットの内容をキャリーフラグにロードします。							
LD CF,(IY).b	1 1 1 0 0 1 0 1	0 1 0 1 1 b b b				C – * – – –	3	CF ← (IY).b
	インデックスレジスタ IY で指定されるアドレスの、オブジェクトコード中の b で指定されるビットの内容をキャリーフラグにロードします。							
LD CF,(IX+d).b	1 1 0 1 0 1 0 0	d d d d d d d d	0 1 0 1 1 b b b			C – * – – –	5	CF ← (IX+d).b
	インデックスレジスタ IX の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスの、オブジェクトコード中の b で指定されるビットの内容をキャリーフラグにロードします。							
LD CF,(IY+d).b	1 1 0 1 0 1 0 1	d d d d d d d d	0 1 0 1 1 b b b			C – * – – –	5	CF ← (IY+d).b
	インデックスレジスタ IY の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスの、オブジェクトコード中の b で指定されるビットの内容をキャリーフラグにロードします。							
LD CF,(SP+d).b	1 1 0 1 0 1 1 0	d d d d d d d d	0 1 0 1 1 b b b			C – * – – –	5	CF ← (SP+d).b
	スタックポインタ SP の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスの、オブジェクトコード中の b で指定されるビットの内容をキャリーフラグにロードします。							
LD CF,(HL+d).b	1 1 0 1 0 1 1 1	d d d d d d d d	0 1 0 1 1 b b b			C – * – – –	5	CF ← (HL+d).b
	レジスタペア HL の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスの、オブジェクトコード中の b で指定されるビットの内容をキャリーフラグにロードします。							
LD CF,(HL+C).b	1 1 1 0 0 1 1 1	0 1 0 1 1 b b b				C – * – – –	5	CF ← (HL+C).b
	レジスタペア HL の内容に C レジスタの内容を符号拡張して加算した値で指定されるアドレスの、オブジェクトコード中の b で指定されるビットの内容をキャリーフラグにロードします。							
LD CF,(+SP).b	1 1 1 0 0 1 1 0	0 1 0 1 1 b b b				C – * – – –	4	SP ← SP+1:CF ← (SP).b
	スタックポインタ SP の内容をインクリメントし、その値で指定されるアドレスの、オブジェクトコード中の b で指定されるビットの内容をキャリーフラグにロードします。							

ニモニック	オブジェクトコード (2 進)	フラグ J Z C H S V	サイクル	オペレーション
LD CF,(PC+A).b 注	0 1 0 0 1 1 1 1 | 0 1 0 1 1 b b b	$\overline{C}$ – * – – –	5	CF ← (PC+A).b
	プログラムカウンタ PC の内容に A レジスタの内容を符号拡張して加算した値で指定されるアドレスの、オブジェクトコード中の b で指定されるビットの内容をキャリーフラグにロードします。			
LD CF,(x).A	1 1 1 0 0 0 0 0 | x x x x x x x x | 1 1 1 1 1 1 0 0	$\overline{C}$ – * – – –	4	CF ← (x).A
	オブジェクトコード中の x で直接 指定されるアドレス (0x0000 ~ 0x00FF 番地) の、A レジスタの下位 3 ビットで指定されるビットの内容をキャリーフラグにロードします。			
LD CF,(vw).A	1 1 1 0 0 0 0 1 | w w w w w w w w | v v v v v v v v 1 1 1 1 1 1 0 0	$\overline{C}$ – * – – –	5	CF ← (vw).A
	オブジェクトコード中の vw で直接指定されるアドレス (0x0000 ~ 0xFFFF 番地) の、A レジスタの下位 3 ビットで指定されるビットの内容をキャリーフラグにロードします。			
LD CF,(DE).A	1 1 1 0 0 0 1 0 | 1 1 1 1 1 1 0 0	$\overline{C}$ – * – – –	3	CF ← (DE).A
	レジスタペア DE で指定されるアドレスの、A レジスタの下位 3 ビットで指定されるビットの内容をキャリーフラグにロードします。			
LD CF,(HL).A	1 1 1 0 0 0 1 1 | 1 1 1 1 1 1 0 0	$\overline{C}$ – * – – –	3	CF ← (HL).A
	レジスタペア HL で指定されるアドレスの、A レジスタの下位 3 ビットで指定されるビットの内容をキャリーフラグにロードします。			
LD CF,(IX).A	1 1 1 0 0 1 0 0 | 1 1 1 1 1 1 0 0	$\overline{C}$ – * – – –	3	CF ← (IX).A
	インデックスレジスタ IX で指定されるアドレスの、A レジスタの下位 3 ビットで指定されるビットの内容をキャリーフラグにロードします。			
LD CF,(IY).A	1 1 1 0 0 1 0 1 | 1 1 1 1 1 1 0 0	$\overline{C}$ – * – – –	3	CF ← (IY).A
	インデックスレジスタ IY で指定されるアドレスの、A レジスタの下位 3 ビットで指定されるビットの内容をキャリーフラグにロードします。			
LD CF,(IX+d).A	1 1 0 1 0 1 0 0 | d d d d d d d d | 1 1 1 1 1 1 0 0	$\overline{C}$ – * – – –	5	CF ← (IX+d).A
	インデックスレジスタ IX の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスの、A レジスタの下位 3 ビットで指定されるビットの内容をキャリーフラグにロードします。			
LD CF,(IY+d).A	1 1 0 1 0 1 0 1 | d d d d d d d d | 1 1 1 1 1 1 0 0	$\overline{C}$ – * – – –	5	CF ← (IY+d).A
	インデックスレジスタ IY の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスの、A レジスタの下位 3 ビットで指定されるビットの内容をキャリーフラグにロードします。			
LD CF,(SP+d).A	1 1 0 1 0 1 1 0 | d d d d d d d d | 1 1 1 1 1 1 0 0	$\overline{C}$ – * – – –	5	CF ← (SP+d).A
	スタックポインタ SP の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスの、A レジスタの下位 3 ビットで指定されるビットの内容をキャリーフラグにロードします。			
LD CF,(HL+d).A	1 1 0 1 0 1 1 1 | d d d d d d d d | 1 1 1 1 1 1 0 0	$\overline{C}$ – * – – –	5	CF ← (HL+d).A
	レジスタペア HL の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスの、A レジスタの下位 3 ビットで指定されるビットの内容をキャリーフラグにロードします。			
LD CF,(HL+C).A	1 1 1 0 0 1 1 1 | 1 1 1 1 1 1 0 0	$\overline{C}$ – * – – –	5	CF ← (HL+C).A
	レジスタペア HL の内容に C レジスタの内容を符号拡張して加算した値で指定されるアドレスの、A レジスタの下位 3 ビットで指定されるビットの内容をキャリーフラグにロードします。			
LD CF,(+SP).A	1 1 1 0 0 1 1 0 | 1 1 1 1 1 1 0 0	$\overline{C}$ – * – – –	4	SP ← SP+1:CF ← (SP).A
	スタックポインタ SP の内容をインクリメントし、その値で指定されるアドレスの、A レジスタの下位 3 ビットで指定されるビットの内容をキャリーフラグにロードします。			
LD CF,(PC+A).A 注	0 1 0 0 1 1 1 1 | 1 1 1 1 1 1 0 0	$\overline{C}$ – * – – –	5	CF ← (PC+A).A
	プログラムカウンタ PC の内容に A レジスタの内容を符号拡張して加算した値で指定されるアドレスの、A レジスタの下位 3 ビットで指定されるビットの内容をキャリーフラグにロードします。			
TEST g.b#1	1 1 1 0 1 g g g | 0 1 0 1 1 b b b	* – $\overline{J}$ – – –	2	JF ← $\overline{g.b}$
	8 ビットレジスタ g の、オブジェクトコード中の b で指定されるビットの内容の反転値をジャンプ ステータス フラグに入れます。 例 : A = 0y01011100 のとき、TEST A. 5 命令を実行すると、JF = 1, CF = 0 となります。			
TEST (x).b#1	0 1 0 1 1 b b b | x x x x x x x x |	* – $\overline{J}$ – – –	3	JF ← $\overline{(x).b}$
	オブジェクトコード中の x で直接指定されるアドレス (0x0000 ~ 0x00FF 番地) の、オブジェクトコード中の b で指定されるビットの内容の反転値をジャンプ ステータス フラグに入れます。			
TEST (vw).b#1	1 1 1 0 0 0 0 1 | w w w w w w w w | v v v v v v v v 0 1 0 1 1 b b b	* – $\overline{J}$ – – –	5	JF ← $\overline{(vw).b}$
	オブジェクトコード中の vw で直接指定されるアドレス (0x0000 ~ 0xFFFF 番地) の内容の、オブジェクトコード中の b で指定されるビットの内容の反転値をジャンプ ステータス フラグに入れます。			
TEST (DE).b#1	1 1 1 0 0 0 1 0 | 0 1 0 1 1 b b b	* – $\overline{J}$ – – –	3	JF ← $\overline{(DE).b}$
	レジスタペア DE で指定されるアドレスの内容の、オブジェクトコード中の b で指定されるビットの内容の反転値をジャンプ ステータス フラグに入れます。			
TEST (HL).b#1	1 1 1 0 0 0 1 1 | 0 1 0 1 1 b b b	* – $\overline{J}$ – – –	3	JF ← $\overline{(HL).b}$
	レジスタペア HL で指定されるアドレスの内容の、オブジェクトコード中の b で指定されるビットの内容の反転値をジャンプ ステータス フラグに入れます。			
TEST (IX).b#1	1 1 1 0 0 1 0 0 | 0 1 0 1 1 b b b	* – $\overline{J}$ – – –	3	JF ← $\overline{(IX).b}$
	インデックスレジスタ IX で指定されるアドレスの内容の、オブジェクトコード中の b で指定されるビットの内容の反転値をジャンプ ステータス フラグに入れます。			
TEST (IY).b#1	1 1 1 0 0 1 0 1 | 0 1 0 1 1 b b b	* – $\overline{J}$ – – –	3	JF ← $\overline{(IY).b}$
	インデックスレジスタ IY で指定されるアドレスの内容の、オブジェクトコード中の b で指定されるビットの内容の反転値をジャンプ ステータス フラグに入れます。			
TEST (IX+d).b#1	1 1 0 1 0 1 0 0 | d d d d d d d d | 0 1 0 1 1 b b b	* – $\overline{J}$ – – –	5	JF ← $\overline{(IX+d).b}$
	インデックスレジスタ IX の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスの、オブジェクトコード中の b で指定されるビットの内容の反転値をジャンプ ステータス フラグに入れます。			
TEST (IY+d).b#1	1 1 0 1 0 1 0 1 | d d d d d d d d | 0 1 0 1 1 b b b	* – $\overline{J}$ – – –	5	JF ← $\overline{(IY+d).b}$
	インデックスレジスタ IY の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスの、オブジェクトコード中の b で指定されるビットの内容の反転値をジャンプ ステータス フラグに入れます。			

ニモニック	オブジェクトコード (2 進)	フラグ J Z C H S V	サイクル	オペレーション
TEST (SP+d).b#1	1 1 0 1 0 1 1 0 d d d d d d d d 0 1 0 1 1 b b b	* – J – – –	5	JF ← (SP+d).b
	スタックポインタ SP の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスの、オブジェクトコード中の b で指定されるビットの内容の反転値をジャンプ ステータス フラグに入れます。			
TEST (HL+d).b#1	1 1 0 1 0 1 1 1 d d d d d d d d 0 1 0 1 1 b b b	* – J – – –	5	JF ← (HL+d).b
	レジスタペア HL の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスの、オブジェクトコード中の b で指定されるビットの内容の反転値をジャンプ ステータス フラグに入れます。			
TEST (HL+C).b#1	1 1 1 0 0 1 1 1 0 1 0 1 1 b b b	* – J – – –	5	JF ← (HL+C).b
	レジスタペア HL の内容に C レジスタの内容を符号拡張して加算した値で指定されるアドレスの、オブジェクトコード中の b で指定されるビットの反転値をジャンプ ステータス フラグに入れます。			
TEST (+SP).b#1	1 1 1 0 0 1 1 0 0 1 0 1 1 b b b	* – J – – –	4	SP ← SP+1:JF ← (SP).b
	スタックポインタ SP の内容をインクリメントし、その値で指定されるアドレスの、オブジェクトコード中の b で指定されるビットの内容の反転値をジャンプ ステータス フラグに入れます。			
TEST (PC+A).b#1 注	0 1 0 0 1 1 1 1 0 1 0 1 1 b b b	* – J – – –	5	JF ← (PC+A).b
	プログラムカウンタ PC の内容に A レジスタの内容を符号拡張して加算した値で指定されるアドレスの、オブジェクトコード中の b で指定されるビットの内容の反転値をジャンプ ステータス フラグに入れます。			
TEST (x).A#1	1 1 1 0 0 0 0 0 x x x x x x x x 1 1 1 1 1 1 0 0	* – J – – –	4	JF ← (x).A
	オブジェクトコード中の x で直接指定されるアドレス (0x0000 ~ 0x00FF 番地) の、A レジスタの下位 3 ビットで指定されるビットの内容の反転値をジャンプ ステータス フラグに入れます。			
TEST (vw).A#1	1 1 1 0 0 0 0 1 w w w w w w w w v v v v v v v v 1 1 1 1 1 1 0 0	* – J – – –	5	JF ← (vw).A
	オブジェクトコード中の vw で直接指定されるアドレス (0x0000 ~ 0xFFFF 番地) の内容の、A レジスタの下位 3 ビットで指定されるビットの内容の反転値をジャンプ ステータス フラグに入れます。			
TEST (DE).A#1	1 1 1 0 0 0 1 0 1 1 1 1 1 1 0 0	* – J – – –	3	JF ← (DE).A
	レジスタペア DE で指定されるアドレスの内容の、A レジスタの下位 3 ビットで指定されるビットの内容の反転値をジャンプ ステータス フラグに入れます。			
TEST (HL).A#1	1 1 1 0 0 0 1 1 1 1 1 1 1 1 0 0	* – J – – –	3	JF ← (HL).A
	レジスタペア HL で指定されるアドレスの内容の、A レジスタの下位 3 ビットで指定されるビットの内容の反転値をジャンプ ステータス フラグに入れます。			
TEST (IX).A#1	1 1 1 0 0 1 0 0 1 1 1 1 1 1 0 0	* – J – – –	3	JF ← (IX).A
	インデックスレジスタ IX で指定されるアドレスの内容の、A レジスタの下位 3 ビットで指定されるビットの内容の反転値をジャンプ ステータス フラグに入れます。			
TEST (IY).A#1	1 1 1 0 0 1 0 1 1 1 1 1 1 1 0 0	* – J – – –	3	JF ← (IY).A
	インデックスレジスタ IY で指定されるアドレスの内容の、A レジスタの下位 3 ビットで指定されるビットの内容の反転値をジャンプ ステータス フラグに入れます。			
TEST (IX+d).A#1	1 1 0 1 0 1 0 0 d d d d d d d d 1 1 1 1 1 1 0 0	* – J – – –	5	JF ← (IX+d).A
	インデックスレジスタ IX の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスの、A レジスタの下位 3 ビットで指定されるビットの内容の反転値をジャンプ ステータス フラグに入れます。			
TEST (IY+d).A#1	1 1 0 1 0 1 0 1 d d d d d d d d 1 1 1 1 1 1 0 0	* – J – – –	5	JF ← (IY+d).A
	インデックスレジスタ IY の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスの、A レジスタの下位 3 ビットで指定されるビットの内容の反転値をジャンプ ステータス フラグに入れます。			
TEST (SP+d).A#1	1 1 0 1 0 1 1 0 d d d d d d d d 1 1 1 1 1 1 0 0	* – J – – –	5	JF ← (SP+d).A
	スタックポインタ SP の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスの、A レジスタの下位 3 ビットで指定されるビットの内容の反転値をジャンプ ステータス フラグに入れます。			
TEST (HL+d).A#1	1 1 0 1 0 1 1 1 d d d d d d d d 1 1 1 1 1 1 0 0	* – J – – –	5	JF ← (HL+d).A
	レジスタペア HL の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスの、A レジスタの下位 3 ビットで指定されるビットの内容の反転値をジャンプ ステータス フラグに入れます。			
TEST (HL+C).A#1	1 1 1 0 0 1 1 1 1 1 1 1 1 1 0 0	* – J – – –	5	JF ← (HL+C).A
	レジスタペア HL の内容に C レジスタの内容を符号拡張して加算した値で指定されるアドレスの、A レジスタの下位 3 ビットで指定されるビットの反転値をジャンプ ステータス フラグに入れます。			
TEST (+SP).A#1	1 1 1 0 0 1 1 0 1 1 1 1 1 1 0 0	* – J – – –	4	SP ← SP+1:JF ← (SP).A
	スタックポインタ SP の内容をインクリメントし、その値で指定されるアドレスの、オブジェクトコード中の b で指定されるビットの内容の反転値をジャンプ ステータス フラグにいれます。			
TEST (PC+A).A#1 注	0 1 0 0 1 1 1 1 1 1 1 1 1 1 0 0	* – J – – –	5	JF ← (PC+A).A
	プログラムカウンタ PC の内容に A レジスタの内容を符号拡張して加算した値で指定されるアドレスの、A レジスタの下位 3 ビットで指定されるビットの内容の反転値をジャンプ ステータス フラグに入れます。			
LD g.b,CF	1 1 1 0 1 g g g 1 1 1 0 1 b b b	1 – – – – –	2	g.b ← CF
	8 ビットレジスタ g の、オブジェクトコード中の b で指定されるビットにキャリーフラグの内容を入れます。 例 1: A = 0x15, CF = 1 のとき、LD A.5, CF 命令を実行すると、A = 0x35, CF = 1 となります。 例 2: B = 0x7E, CF = 0 のとき、LD B.2, CF 命令を実行すると、B = 0x7A, CF = 0 となります。			
LD (x).b,CF	1 1 1 0 0 0 0 0 x x x x x x x x 1 1 1 0 1 b b b	1 – – – – –	5	(x).b ← CF
	オブジェクトコード中の x で指定されるアドレス (0x0000 ~ 0x00FF 番地) の、オブジェクトコード中の b で指定されるビットにキャリーフラグの内容をストアします。			
LD (vw).b,CF	1 1 1 0 0 0 0 1 w w w w w w w w v v v v v v v v 1 1 1 0 1 b b b	1 – – – – –	6	(vw).b ← CF
	オブジェクトコード中の vw で直接指定されるアドレス (0x0000 ~ 0xFFFF 番地) の、オブジェクトコード中の b で指定されるビットにキャリーフラグの内容をストアします。			
LD (DE).b,CF	1 1 1 0 0 0 1 0 1 1 1 0 1 b b b	1 – – – – –	4	(DE).b ← CF
	レジスタペア DE で指定されるアドレスの、オブジェクトコード中の b で指定されるビットにキャリーフラグの内容をストアします。			
LD (HL).b,CF	1 1 1 0 0 0 1 1 1 1 1 0 1 b b b	1 – – – – –	4	(HL).b ← CF
	レジスタペア HL で指定されるアドレスの、オブジェクトコード中の b で指定されるビットにキャリーフラグの内容をストアします。			

ニモニック	オブジェクトコード (2 進)	フラグ J Z C H S V	サイクル	オペレーション
LD (IX).b,CF	1 1 1 0 0 1 0 0 | 1 1 1 0 1 b b b	1 – – – – –	4	(IX).b ← CF
	インデックスレジスタ IX で指定されるアドレスの、オブジェクトコード中の b で指定されるビットにキャリーフラグの内容をストアします。			
LD (IY).b,CF	1 1 1 0 0 1 0 1 | 1 1 1 0 1 b b b	1 – – – – –	4	(IY).b ← CF
	インデックスレジスタ IY で指定されるアドレスの、オブジェクトコード中の b で指定されるビットにキャリーフラグの内容をストアします。			
LD (IX+d).b,CF	1 1 0 1 0 1 0 0 | d d d d d d d d | 1 1 1 0 1 b b b	1 – – – – –	6	(IX+d).b ← CF
	インデックスレジスタ IX の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスの、オブジェクトコード中の b で指定されるビットにキャリーフラグの内容をストアします。			
LD (IY+d).b,CF	1 1 0 1 0 1 0 1 | d d d d d d d d | 1 1 1 0 1 b b b	1 – – – – –	6	(IY+d).b ← CF
	インデックスレジスタ IY の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスの、オブジェクトコード中の b で指定されるビットにキャリーフラグの内容をストアします。			
LD (SP+d).b,CF	1 1 0 1 0 1 1 0 | d d d d d d d d | 1 1 1 0 1 b b b	1 – – – – –	6	(SP+d).b ← CF
	スタックポインタ SP の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスの、オブジェクトコード中の b で指定されるビットにキャリーフラグの内容をストアします。			
LD (HL+d).b,CF	1 1 0 1 0 1 1 1 | d d d d d d d d | 1 1 1 0 1 b b b	1 – – – – –	6	(HL+d).b ← CF
	レジスタペア HL の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスの、オブジェクトコード中の b で指定されるビットにキャリーフラグの内容をストアします。			
LD (HL+C).b,CF	1 1 1 0 0 1 1 1 | 1 1 1 0 1 b b b	1 – – – – –	6	(HL+C).b ← CF
	レジスタペア HL の内容に C レジスタの内容を符号拡張して加算した値で指定されるアドレスの、オブジェクトコード中の b で指定されるビットにキャリーフラグの内容をストアします。			
LD (+SP).b,CF	1 1 1 0 0 1 1 0 | 1 1 1 0 1 b b b	1 – – – – –	5	SP ← SP+1:(SP).b ← CF
	スタックポインタ SP の内容をインクリメントし、その値で指定されるアドレスの、オブジェクトコードの b で指定されるビットにキャリーフラグの内容をストアします。			
LD (PC+A).b,CF 注	0 1 0 0 1 1 1 1 | 1 1 1 0 1 b b b	1 – – – – –	6	(PC+A).b ← CF
	プログラムカウンタ PC の内容に A レジスタの内容を符号拡張して加算した値で指定されるアドレスの、オブジェクトコード中の b で指定されるビットにキャリーフラグの内容をストアします。			
LD (x).A,CF	1 1 1 0 0 0 0 0 | x x x x x x x x | 1 1 1 1 0 0 1 1	1 – – – – –	5	(x).A ← CF
	オブジェクトコード中の x で指定されるアドレス (0x0000 ~ 0x00FF 番地) の、A レジスタの下位 3 ビットで指定されるビットにキャリーフラグの内容をストアします。			
LD (vw).A,CF	1 1 1 0 0 0 0 1 | w w w w w w w w | v v v v v v v v 1 1 1 1 0 0 1 1	1 – – – – –	6	(vw).A ← CF
	オブジェクトコード中の vw で直接指定されるアドレスの、A レジスタの下位 3 ビットで指定されるビットにキャリーフラグの内容をストアします。			
LD (DE).A,CF	1 1 1 0 0 0 1 0 | 1 1 1 1 0 0 1 1	1 – – – – –	4	(DE).A ← CF
	レジスタペア DE で指定されるアドレスの、A レジスタの下位 3 ビットで指定されるビットにキャリーフラグの内容をストアします。			
LD (HL).A,CF	1 1 1 0 0 0 1 1 | 1 1 1 1 0 0 1 1	1 – – – – –	4	(HL).A ← CF
	レジスタペア HL で指定されるアドレスの、A レジスタの下位 3 ビットで指定されるビットにキャリーフラグの内容をストアします。			
LD (IX).A,CF	1 1 1 0 0 1 0 0 | 1 1 1 1 0 0 1 1	1 – – – – –	4	(IX).A ← CF
	インデックスレジスタ IX で指定されるアドレスの、A レジスタの下位 3 ビットで指定されるビットにキャリーフラグの内容をストアします。			
LD (IY).A,CF	1 1 1 0 0 1 0 1 | 1 1 1 1 0 0 1 1	1 – – – – –	4	(IY).A ← CF
	インデックスレジスタ IY で指定されるアドレスの、A レジスタの下位 3 ビットで指定されるビットにキャリーフラグの内容をストアします。			
LD (IX+d).A,CF	1 1 0 1 0 1 0 0 | d d d d d d d d | 1 1 1 1 0 0 1 1	1 – – – – –	6	(IX+d).A ← CF
	インデックスレジスタ IX の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスの、A レジスタの下位 3 ビットで指定されるビットにキャリーフラグの内容をストアします。			
LD (IY+d).A,CF	1 1 0 1 0 1 0 1 | d d d d d d d d | 1 1 1 1 0 0 1 1	1 – – – – –	6	(IY+d).A ← CF
	インデックスレジスタ IY の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスの、A レジスタの下位 3 ビットで指定されるビットにキャリーフラグの内容をストアします。			
LD (SP+d).A,CF	1 1 0 1 0 1 1 0 | d d d d d d d d | 1 1 1 1 0 0 1 1	1 – – – – –	6	(SP+d).A ← CF
	スタックポインタ SP の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスの、A レジスタの下位 3 ビットで指定されるビットにキャリーフラグの内容をストアします。			
LD (HL+d).A,CF	1 1 0 1 0 1 1 1 | d d d d d d d d | 1 1 1 1 0 0 1 1	1 – – – – –	6	(HL+d).A ← CF
	レジスタペア HL の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスの、A レジスタの下位 3 ビットで指定されるビットにキャリーフラグの内容をストアします。			
LD (HL+C).A,CF	1 1 1 0 0 1 1 1 | 1 1 1 1 0 0 1 1	1 – – – – –	6	(HL+C).A ← CF
	レジスタペア HL の内容に C レジスタの内容を符号拡張して加算した値で指定されるアドレスの、A レジスタの下位 3 ビットで指定されるビットにキャリーフラグの内容をストアします。			
LD (+SP).A,CF	1 1 1 0 0 1 1 0 | 1 1 1 1 0 0 1 1	1 – – – – –	5	SP ← SP+1:(SP).A ← CF
	スタックポインタ SP の内容をインクリメントし、その値で指定されるアドレスの、A レジスタの下位 3 ビットで指定されるビットにキャリーフラグの内容をストアします。			
LD (PC+A).A,CF 注	0 1 0 0 1 1 1 1 | 1 1 1 1 0 0 1 1	1 – – – – –	6	(PC+A).A ← CF
	プログラムカウンタ PC の内容に A レジスタの内容を符号拡張して加算した値で指定されるアドレスの、A レジスタの下位 3 ビットで指定されるビットにキャリーフラグの内容をストアします。			
CPL g.b	1 1 1 0 1 g g g | 1 1 1 0 0 b b b	Z * – – – –	3	ZF ← $\overline{g.b}$:g.b ← $\overline{g.b}$
	8 ビットレジスタ g の、オブジェクトコード中の b で指定されるビットの内容の反転値をゼロフラグに入れた後、そのビットの内容を反転します。 例 : A = 0x3C のとき、CPL A.3 命令を実行すると、ZF = 0, A = 0x34 となります。			
CPL (x).b	1 1 1 0 0 0 0 0 | x x x x x x x x | 1 1 1 0 0 b b b	Z * – – – –	5	ZF ← $\overline{(x).b}$:(x).b ← $\overline{(x).b}$
	オブジェクトコード中の x で直接指定されるアドレス (0x0000 ~ 0x00FF 番地) の、A レジスタの下位 3 ビットで指定されるビットの内容の反転値をゼロフラグに入れた後、そのビットの内容を反転します。			
CPL (vw).b	1 1 1 0 0 0 0 1 | w w w w w w w w | v v v v v v v v 1 1 1 0 0 b b b	Z * – – – –	6	ZF ← $\overline{(vw).b}$:(vw).b ← $\overline{(vw).b}$
	オブジェクトコード中の vw で直接指定されるアドレス (0x0000 ~ 0xFFFF 番地) の、オブジェクトコード中の b で指定されるビットの内容の反転値をゼロフラグに入れた後、そのビットの内容を反転します。			

ニモニック	オブジェクトコード (2 進)	フラグ J Z C H S V	サイクル	オペレーション
CPL (DE).b	1 1 1 0 0 0 1 0 | 1 1 1 0 0 b b b	Z * – – – –	4	ZF ← $\overline{(DE).b}$:(DE).b ← $\overline{(DE).b}$
	レジスタペア DE で指定されるアドレスの、オブジェクトコード中の b で指定されるビットの内容の反転値をゼロフラグに入れた後、そのビットの内容を反転します。			
CPL (HL).b	1 1 1 0 0 0 1 1 | 1 1 1 0 0 b b b	Z * – – – –	4	ZF ← $\overline{(HL).b}$:(HL).b ← $\overline{(HL).b}$
	レジスタペア HL で指定されるアドレスの、オブジェクトコード中の b で指定されるビットの内容の反転値をゼロフラグに入れた後、そのビットの内容を反転します。			
CPL (IX).b	1 1 1 0 0 1 0 0 | 1 1 1 0 0 b b b	Z * – – – –	4	ZF ← $\overline{(IX).b}$:(IX).b ← $\overline{(IX).b}$
	インデックスレジスタ IX で指定されるアドレスの、オブジェクトコード中の b で指定されるビットの内容の反転値をゼロフラグに入れた後、そのビットの内容を反転します。			
CPL (IY).b	1 1 1 0 0 1 0 1 | 1 1 1 0 0 b b b	Z * – – – –	4	ZF ← $\overline{(IY).b}$:(IY).b ← $\overline{(IY).b}$
	インデックスレジスタ IY で指定されるアドレスの、オブジェクトコード中の b で指定されるビットの内容の反転値をゼロフラグに入れた後、そのビットの内容を反転します。			
CPL (IX+d).b	1 1 0 1 0 1 0 0 | d d d d d d d d | 1 1 1 0 0 b b b	Z * – – – –	6	ZF ← $\overline{(IX+d).b}$:(IX+d).b ← $\overline{(IX+d).b}$
	インデックスレジスタ IX の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスの、オブジェクトコード中の b で指定されるビットの内容を反転します。			
CPL (IY+d).b	1 1 0 1 0 1 0 1 | d d d d d d d d | 1 1 1 0 0 b b b	Z * – – – –	6	ZF ← $\overline{(IY+d).b}$:(IY+d).b ← $\overline{(IY+d).b}$
	インデックスレジスタ IY の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスの、オブジェクトコード中の b で指定されるビットの内容を反転します。			
CPL (SP+d).b	1 1 0 1 0 1 1 0 | d d d d d d d d | 1 1 1 0 0 b b b	Z * – – – –	6	ZF ← $\overline{(SP+d).b}$:(SP+d).b ← $\overline{(SP+d).b}$
	スタックポインタ SP の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスの、オブジェクトコード中の b で指定されるビットの内容を反転します。			
CPL (HL+d).b	1 1 0 1 0 1 1 1 | d d d d d d d d | 1 1 1 0 0 b b b	Z * – – – –	6	ZF ← $\overline{(HL+d).b}$:(HL+d).b ← $\overline{(HL+d).b}$
	レジスタペア HL の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスの、オブジェクトコード中の b で指定されるビットの内容を反転します。			
CPL (HL+C).b	1 1 1 0 0 1 1 1 | 1 1 1 0 0 b b b	Z * – – – –	6	ZF ← $\overline{(HL+C).b}$:(HL+C).b ← $\overline{(HL+C).b}$
	レジスタペア HL の内容に C レジスタの内容を符号拡張して加算した値で指定されるアドレスの、オブジェクトコード中の b で指定されるビットの内容を反転します。			
CPL (+SP).b	1 1 1 0 0 1 1 0 | 1 1 1 0 0 b b b	Z * – – – –	5	SP ← SP+1:ZF ← $\overline{(SP).b}$:(SP).b ← $\overline{(SP).b}$
	スタックポインタ SP の内容をインクリメントし、その値で指定されるアドレスの、オブジェクトコード中の b で指定されるビットの内容を反転します。			
CPL (PC+A).b 注	0 1 0 0 1 1 1 1 | 1 1 1 0 0 b b b	Z * – – – –	6	ZF ← $\overline{(PC+A).b}$:(PC+A).b ← $\overline{(PC+A).b}$
	プログラムカウンタ PC の内容に A レジスタの内容を符号拡張して加算した値で指定されるアドレスの、オブジェクトコード中の b で指定されるビットの内容を反転します。			
CPL (x).A	1 1 1 0 0 0 0 0 | x x x x x x x x | 1 1 1 1 1 0 1 1	Z * – – – –	5	ZF ← $\overline{(x).A}$:(x).A ← $\overline{(x).A}$
	オブジェクトコード中の x で直接指定されるアドレス (0x0000 ~ 0x00FF 番地) の、A レジスタの下位 3 ビットで指定されるビットの内容の反転値をゼロフラグに入れた後、そのビットの内容を反転します。			
CPL (vw).A	1 1 1 0 0 0 0 1 | w w w w w w w w | v v v v v v v v | 1 1 1 1 1 0 1 1	Z * – – – –	6	ZF ← $\overline{(vw).A}$:(vw).A ← $\overline{(vw).A}$
	オブジェクトコード中の vw で直接指定されるアドレス (0x0000 ~ 0xFFFF 番地) の、A レジスタの下位 3 ビットで指定されるビットの内容の反転値をゼロフラグに入れた後、そのビットの内容を反転します。			
CPL (DE).A	1 1 1 0 0 0 1 0 | 1 1 1 1 1 0 1 1	Z * – – – –	4	ZF ← $\overline{(DE).A}$:(DE).A ← $\overline{(DE).A}$
	レジスタペア DE で指定されるアドレスの、A レジスタの下位 3 ビットで指定されるビットの内容の反転値をゼロフラグに入れた後、そのビットの内容を反転します。			
CPL (HL).A	1 1 1 0 0 0 1 1 | 1 1 1 1 1 0 1 1	Z * – – – –	4	ZF ← $\overline{(HL).A}$:(HL).A ← $\overline{(HL).A}$
	レジスタペア HL で指定されるアドレスの、A レジスタの下位 3 ビットで指定されるビットの内容の反転値をゼロフラグに入れた後、そのビットの内容を反転します。			
CPL (IX).A	1 1 1 0 0 1 0 0 | 1 1 1 1 1 0 1 1	Z * – – – –	4	ZF ← $\overline{(IX).A}$:(IX).A ← $\overline{(IX).A}$
	インデックスレジスタ IX で指定されるアドレスの、A レジスタの下位 3 ビットで指定されるビットの内容の反転値をゼロフラグに入れた後、そのビットの内容を反転します。			
CPL (IY).A	1 1 1 0 0 1 0 1 | 1 1 1 1 1 0 1 1	Z * – – – –	4	ZF ← $\overline{(IY).A}$:(IY).A ← $\overline{(IY).A}$
	インデックスレジスタ IY で指定されるアドレスの、A レジスタの下位 3 ビットで指定されるビットの内容の反転値をゼロフラグに入れた後、そのビットの内容を反転します。			
CPL (IX+d).A	1 1 0 1 0 1 0 0 | d d d d d d d d | 1 1 1 1 1 0 1 1	Z * – – – –	6	ZF ← $\overline{(IX+d).A}$:(IX+d).A ← $\overline{(IX+d).A}$
	インデックスレジスタ IX の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスの、A レジスタの下位 3 ビットで指定されるビットの内容を反転します。			
CPL (IY+d).A	1 1 0 1 0 1 0 1 | d d d d d d d d | 1 1 1 1 1 0 1 1	Z * – – – –	6	ZF ← $\overline{(IY+d).A}$:(IY+d).A ← $\overline{(IY+d).A}$
	インデックスレジスタ IY の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスの、A レジスタの下位 3 ビットで指定されるビットの内容を反転します。			
CPL (SP+d).A	1 1 0 1 0 1 1 0 | d d d d d d d d | 1 1 1 1 1 0 1 1	Z * – – – –	6	ZF ← $\overline{(SP+d).A}$:(SP+d).A ← $\overline{(SP+d).A}$
	スタックポインタ SP の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスの、A レジスタの下位 3 ビットで指定されるビットの内容を反転します。			
CPL (HL+d).A	1 1 0 1 0 1 1 1 | d d d d d d d d | 1 1 1 1 1 0 1 1	Z * – – – –	6	ZF ← $\overline{(HL+d).A}$:(HL+d).A ← $\overline{(HL+d).A}$
	レジスタペア HL の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスの、A レジスタの下位 3 ビットで指定されるビットの内容を反転します。			
CPL (HL+C).A	1 1 1 0 0 1 1 1 | 1 1 1 1 1 0 1 1	Z * – – – –	6	ZF ← $\overline{(HL+C).A}$:(HL+C).A ← $\overline{(HL+C).A}$
	レジスタペア HL の内容に C レジスタの内容を符号拡張して加算した値で指定されるアドレスの、A レジスタの下位 3 ビットで指定されるビットの内容を反転します。			
CPL (+SP).A	1 1 1 0 0 1 1 0 | 1 1 1 1 1 0 1 1	Z * – – – –	5	SP ← SP+1:ZF ← $\overline{(SP).A}$:(SP).A ← $\overline{(SP).A}$
	スタックポインタ SP の内容をインクリメントし、その値で指定されるアドレスの、A レジスタの下位 3 ビットで指定されるビットの内容を反転します。			
CPL (PC+A).A 注	0 1 0 0 1 1 1 1 | 1 1 1 1 1 0 1 1	Z * – – – –	6	ZF ← $\overline{(PC+A).A}$:(PC+A).A ← $\overline{(PC+A).A}$
	プログラムカウンタ PC の内容に A レジスタの内容を符号拡張して加算した値で指定されるアドレスの、A レジスタの下位 3 ビットで指定されるビットの内容を反転します。			

ニモニック	オブジェクトコード (2 進)				フラグ J Z C H S V	サイクル	オペレーション
XOR CF,g.b	1 1 1 0 1 g g g	0 1 0 1 0 b b b			$\overline{C}$ – * – – –	2	CF ← CF ^ g.b
	8 ビットレジスタ g の、オブジェクトコード中の b で指定されるビットの内容とキャリーフラグの内容とで排他的論理和をとり、結果をキャリーフラグに入れます。 例 : A = 0x5B, CF = 1 のとき、XOR CF, A.3 命令を実行すると、A = 0x5B, CF = 0, JF = 1 となります。						
XOR CF,(x).b	1 1 1 0 0 0 0 0	x x x x x x x x	0 1 0 1 0 b b b		$\overline{C}$ – * – – –	4	CF ← CF ^ (x).b
	オブジェクトコード中の x で直接指定されるアドレス (0x0000 ~ 0x00FF 番地) の、オブジェクトコード中の b で指定されるビットの内容とキャリーフラグの内容とで排他的論理和を取り、結果をキャリーフラグに入れます。						
XOR CF,(vw).b	1 1 1 0 0 0 0 1 0 1 0 1 0 b b b	w w w w w w w w	v v v v v v v v		$\overline{C}$ – * – – –	5	CF ← CF ^ (vw).b
	オブジェクトコード中の vw で直接指定されるアドレス (0x0000 ~ 0xFFFF 番地) の、オブジェクトコード中の b で指定されるビットの内容とキャリーフラグの内容とで排他的論理和を取り、結果をキャリーフラグに入れます。						
XOR CF,(DE).b	1 1 1 0 0 0 1 0	0 1 0 1 0 b b b			$\overline{C}$ – * – – –	3	CF ← CF ^ (DE).b
	レジスタペア DE で指定されるアドレスの、オブジェクトコード中の b で指定されるビットの内容とキャリーフラグの内容とで排他的論理和を取り、結果をキャリーフラグに入れます。						
XOR CF,(HL).b	1 1 1 0 0 0 1 1	0 1 0 1 0 b b b			$\overline{C}$ – * – – –	3	CF ← CF ^ (HL).b
	レジスタペア HL で指定されるアドレスの、オブジェクトコード中の b で指定されるビットの内容とキャリーフラグの内容とで排他的論理和を取り、結果をキャリーフラグに入れます。						
XOR CF,(IX).b	1 1 1 0 0 1 0 0	0 1 0 1 0 b b b			$\overline{C}$ – * – – –	3	CF ← CF ^ (IX).b
	インデックスレジスタ IX で指定されるアドレスの、オブジェクトコード中の b で指定されるビットの内容とキャリーフラグの内容とで排他的論理和を取り、結果をキャリーフラグに入れます。						
XOR CF,(IY).b	1 1 1 0 0 1 0 1	0 1 0 1 0 b b b			$\overline{C}$ – * – – –	3	CF ← CF ^ (IY).b
	インデックスレジスタ IY で指定されるアドレスの、オブジェクトコード中の b で指定されるビットの内容とキャリーフラグの内容とで排他的論理和を取り、結果をキャリーフラグに入れます。						
XOR CF,(IX+d).b	1 1 0 1 0 1 0 0	d d d d d d d d	0 1 0 1 0 b b b		$\overline{C}$ – * – – –	5	CF ← CF ^ (IX+d).b
	インデックスレジスタ IX の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスの、オブジェクトコード中の b で指定されるビットの内容とキャリーフラグの内容とで排他的論理和を取り、結果をキャリーフラグに入れます。						
XOR CF,(IY+d).b	1 1 0 1 0 1 0 1	d d d d d d d d	0 1 0 1 0 b b b		$\overline{C}$ – * – – –	5	CF ← CF ^ (IY+d).b
	インデックスレジスタ IY の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスの、オブジェクトコード中の b で指定されるビットの内容とキャリーフラグの内容とで排他的論理和を取り、結果をキャリーフラグに入れます。						
XOR CF,(SP+d).b	1 1 0 1 0 1 1 0	d d d d d d d d	0 1 0 1 0 b b b		$\overline{C}$ – * – – –	5	CF ← CF ^ (SP+d).b
	スタックポインタ SP の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスの、オブジェクトコード中の b で指定されるビットの内容とキャリーフラグの内容とで排他的論理和を取り、結果をキャリーフラグに入れます。						
XOR CF,(HL+d).b	1 1 0 1 0 1 1 1	d d d d d d d d	0 1 0 1 0 b b b		$\overline{C}$ – * – – –	5	CF ← CF ^ (HL+d).b
	レジスタペア HL の内容にオブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスの、オブジェクトコード中の b で指定されるビットの内容とキャリーフラグの内容とで排他的論理和を取り、結果をキャリーフラグに入れます。						
XOR CF,(HL+C).b	1 1 1 0 0 1 1 1	0 1 0 1 0 b b b			$\overline{C}$ – * – – –	5	CF ← CF ^ (HL+C).b
	レジスタペア HL の内容に C レジスタの内容を符号拡張して加算した値で指定されるアドレスの、オブジェクトコード中の b で指定されるビットの内容とキャリーフラグの内容とで排他的論理和を取り、結果をキャリーフラグに入れます。						
XOR CF,(+SP).b	1 1 1 0 0 1 1 0	0 1 0 1 0 b b b			$\overline{C}$ – * – – –	4	SP ← SP+1:CF ← CF ^ (SP).b
	スタックポインタ SP の内容をインクリメントし、その値で指定されるアドレスの、オブジェクトコード中の b で指定されるビットの内容とキャリーフラグの内容とで排他的論理和を取り、結果をキャリーフラグに入れます。						
XOR CF,(PC+A).b 注	0 1 0 0 1 1 1 1	0 1 0 1 0 b b b			$\overline{C}$ – * – – –	5	CF ← CF ^ (PC+A).b
	プログラムカウンタ PC の内容に A レジスタの内容を符号拡張して加算した値で指定されるアドレスの、オブジェクトコード中の b で指定されるビットの内容とキャリーフラグの内容とで排他的論理和を取り、結果をキャリーフラグに入れます。						
SET CF	0 0 0 0 0 1 0 1				0 – 1 – – –	1	CF ← 1
	キャリーフラグを “1” にセットします。						
CLR CF	0 0 0 0 0 1 0 0				1 – 0 – – –	1	CF ← 0
	キャリーフラグを “0” にクリアします。						
CPL CF	0 0 0 0 0 1 1 0				* – * – – –	1	JF ← CF:CF ← $\overline{CF}$
	キャリーフラグの内容をジャンプステータスフラグに入れた後、キャリーフラグの内容を反転します。 例 : CF = 0 で、この命令を実行すると、JF = 0, CF = 1 となります。						
DI#1	1 1 0 0 1 0 0 0	0 0 1 1 1 0 1 0			Z * – – – –	4	ZF ← $\overline{IMF}$:IMF ← 0
	割り込みマスタ許可フラグの内容の反転値をゼロフラグに入れた後、割り込みマスタ許可フラグを “0” にクリアします (マスカブル割り込みの受け付けを禁止します) 。						
EI#1	1 1 0 0 0 0 0 0	0 0 1 1 1 0 1 0			Z * – – – –	4	ZF ← $\overline{IMF}$:IMF ← 1
	割り込みマスタ許可フラグの内容の反転値をゼロフラグに入れた後、割り込みマスタ許可フラグを “1” にセットします (マスカブル割り込みの受け付けを許可します) 。						

#1 アセンブラ用拡張命令

注) (PC+A) を使用した命令には制限があります。詳細は、TLCS-870/C1 シリーズ 命令セット 1.4 アドレッシングモードを参照してください。

2.5 ジャンプ

ニモニック	オブジェクトコード (2 進)	フラグ J Z C H S V	サイクル	オペレーション
JRS T,$+2+d	1 0 0 d d d d	1 – – – – –	4/2(t/f)	if JF=1 then PC ← PC+d else null
	ジャンプステータスフラグが “1” のとき、プログラムカウンタ PC の内容 (JRS 命令の置かれているアドレスの 2 番地先を示しています) に、オブジェクトコード中の 5 ビットディスプレースメント d を符号拡張して (–16~+15) 加算した値で指定されるアドレスにジャンプします (4 サイクル命令)。 ジャンプ ステータス フラグが “0” のときは、ジャンプ ステータス フラグを “1” にセットし、次命令の実行に移ります (2 サイクル命令)。 例 : ジャンプ ステータス フラグが “1” のとき、0xC134 番地に置かれた JRS T, $+9 命令を実行すると、0xC13D 番地にジャンプします。			
JRS F,$+2+d	1 0 1 d d d d	1 – – – – –	4/2(t/f)	if JF=0 then PC ← PC+d else null
	ジャンプステータスフラグが “0” のとき、プログラムカウンタ PC の内容に、オブジェクトコード中の 5 ビットディスプレースメント d を符号拡張して (–16~+15) 加算した値で指定されるアドレスにジャンプします (4 サイクル)。 ジャンプステータスフラグが “1” のときは、何もせず次命令の実行に移ります (2 サイクル)。			
JR T,$+2+d	1 1 0 1 1 1 1 0 d d d d d d d d	1 – – – – –	4/2(t/f)	if JF=1 then PC ← PC+d else null
	ジャンプステータスフラグが “1” のとき、プログラムカウンタ PC の内容 (JR 命令の置かれているアドレスの 2 番地先を示しています) に、オブジェクトコード中の 8 ビットディスプレースメント d を符号拡張して (–128~+127) 加算した値で指定されるアドレスにジャンプします (4 サイクル)。 ジャンプステータスフラグが “0” のときは、ジャンプステータスフラグを “1” にセットし、次命令の実行に移ります (2 サイクル)。 例 : ジャンプステータスフラグが “1” のとき、0xC134 番地に置かれた JR T, $+0xF6 命令を実行すると、0xC12A 番地にジャンプします。			
JR F,$+2+d	1 1 0 1 1 1 1 1 d d d d d d d d	1 – – – – –	4/2(t/f)	if JF=0 then PC ← PC+d else null
	ジャンプ ステータス フラグが “0” のとき、プログラムカウンタ PC の内容に、オブジェクトコード中の 8 ビットディスプレースメント d を符号拡張して (–128~+127) 加算した値で指定されるアドレスにジャンプします (4 サイクル) 。 ジャンプ ステータス フラグが “1” のときは、何もせず次命令の実行に移ります (2 サイクル)。			
JR EQ,$+2+d または JR Z,$+2+d	1 1 0 1 1 0 0 0 d d d d d d d d	1 – – – – –	4/2(t/f)	if ZF=1 then PC ← PC+d else null
	ゼロフラグが “1” のとき、プログラムカウンタ PC の内容に、オブジェクトコード中の 8 ビットディスプレースメント d を符号拡張して (–128~+127) 加算した値で指定されるアドレスにジャンプします (4 サイクル)。 ゼロフラグが “0” のときは、ジャンプステータスフラグを “1” にセットし、次命令の実行に移ります (2 サイクル)。			
JR NE,$+2+d または JR NZ,$+2+d	1 1 0 1 1 0 0 1 d d d d d d d d	1 – – – – –	4/2(t/f)	if ZF=0 then PC ← PC+d else null
	ゼロフラグが “0” のとき、プログラムカウンタ PC の内容に、オブジェクトコード中の 8 ビットディスプレースメント d を符号拡張して (–128~+127) 加算した値で指定されるアドレスにジャンプします (4 サイクル)。 ゼロフラグが “1” のときは、ジャンプステータスフラグを “1” にセットし、次命令の実行に移ります (2 サイクル)。			
JR CS,$+2+d または JR LT,$+2+d	1 1 0 1 1 0 1 0 d d d d d d d d	1 – – – – –	4/2(t/f)	if CF=1 then PC ← PC+d else null
	キャリーフラグが “1” のとき、プログラムカウンタ PC の内容に、オブジェクトコード中の 8 ビットディスプレースメント d を符号拡張して (–128~+127) 加算した値で指定されるアドレスにジャンプします (4 サイクル)。 キャリーフラグが “0” のときは、ジャンプステータスフラグを “1” にセットし、次命令の実行に移ります (2 サイクル)。			
JR CC,$+2+d または JR GE,$+2+d	1 1 0 1 1 0 1 1 d d d d d d d d	1 – – – – –	4/2(t/f)	if CF=0 then PC ← PC+d else null
	キャリーフラグが “0” のとき、プログラムカウンタ PC の内容に、オブジェクトコード中の 8 ビットディスプレースメント d を符号拡張して (–128~+127) 加算した値で指定されるアドレスにジャンプします (4 サイクル)。 キャリーフラグが “1” のときは、ジャンプ ステータス フラグを “1” にセットし、次命令の実行に移ります (2 サイクル)。			
JR LE,$+2+d	1 1 0 1 1 1 0 0 d d d d d d d d	1 – – – – –	4/2(t/f)	if (CF ∣ ZF)=1 then PC ← PC+d else null
	キャリーフラグまたはゼロフラグが “1” のとき、プログラムカウンタ PC の内容にオブジェクトコード中の 8 ビットディスプレースメント d を符号拡張して (–128~+127) 加算した値で指定されるアドレスにジャンプします (4 サイクル)。 キャリーフラグとゼロフラグがともに “0” のときは、ジャンプ ステータス フラグを “1” にセットし、次命令の実行に移ります (2 サイクル)。			
JR GT,$+2+d	1 1 0 1 1 1 0 1 d d d d d d d d	1 – – – – –	4/2(t/f)	if (CF ∣ ZF)=0 then PC ← PC+d else null
	キャリーフラグとゼロフラグがともに “0” のとき、プログラムカウンタ PC の内容にオブジェクトコード中の 8 ビットディスプレースメント d を符号拡張して (–128~+127) 加算した値で指定されるアドレスにジャンプします (4 サイクル)。 キャリーフラグまたはゼロフラグが “1” のときは、ジャンプステータスフラグを “1” にセットし、次命令の実行に移ります (2 サイクル)。			
JR M,$+3+d	1 1 1 0 1 0 0 0 1 1 0 1 0 0 0 0 d d d d d d d d	1 – – – – –	5/3(t/f)	if SF=1 then PC ← PC+d else null
	サインフラグが “1” のとき、プログラムカウンタ PC の内容 (JR 命令の置かれているアドレスの 3 番地先を示しています) に、オブジェクトコード中の 8 ビットディスプレースメント d を符号拡張して (–128~+127) 加算した値で指定されるアドレスにジャンプします (5 サイクル)。 サインフラグが “0” のときは、ジャンプステータスフラグを “1” にセットし、次命令の実行に移ります (3 サイクル)。			
JR P,$+3+d	1 1 1 0 1 0 0 0 1 1 0 1 0 0 0 1 d d d d d d d d	1 – – – – –	5/3(t/f)	if SF=0 then PC ← PC+d else null
	サインフラグが “0” のとき、プログラムカウンタ PC の内容に、オブジェクトコード中の 8 ビットディスプレースメント d を符号拡張して (–128~+127) 加算した値で指定されるアドレスにジャンプします (5 サイクル)。 サインフラグが “1” のときは、ジャンプステータスフラグを “1” にセットし、次命令の実行に移ります (3 サイクル)。			
JR SLT,$+3+d	1 1 1 0 1 0 0 0 1 1 0 1 0 0 1 0 d d d d d d d d	1 – – – – –	5/3(t/f)	if (SF ^ VF)=1 then PC ← PC+d else null
	サインとオーバフローフラグの排他的論理和の結果が “1” のとき、プログラムカウンタ PC の内容に、オブジェクトコード中の 8 ビットディスプレースメント d を符号拡張して (–128~+127) 加算した値で指定されるアドレスにジャンプします (5 サイクル)。 サインとオーバフローフラグの排他的論理和の結果が “0” のときは、ジャンプステータスフラグを “1” にセットし、次命令の実行に移ります (3 サイクル)。			
JR SGE,$+3+d	1 1 1 0 1 0 0 0 1 1 0 1 0 0 1 1 d d d d d d d d	1 – – – – –	5/3(t/f)	if (SF ^ VF)=0 then PC ← PC+d else null
	サインとオーバフローフラグの排他的論理和の結果が “0” のとき、プログラムカウンタ PC の内容に、オブジェクトコード中の 8 ビットディスプレースメント d を符号拡張して (–128~+127) 加算した値で指定されるアドレスにジャンプします (5 サイクル)。 サインとオーバフローフラグの排他的論理和の結果が “1” のときは、ジャンプステータスフラグを “1” にセットし、次命令の実行に移ります (3 サイクル)。			
JR SLE,$+3+d	1 1 1 0 1 0 0 0 1 1 0 1 0 1 0 0 d d d d d d d d	1 – – – – –	5/3(t/f)	if ZF ∣ (SF ^ VF)=1 then PC ← PC+d else null
	サインとオーバフローフラグの排他的論理和の結果またはゼロフラグの内容が “1” のとき、プログラムカウンタ PC の内容に、オブジェクトコード中の 8 ビットディスプレースメント d を符号拡張して (–128~+127) 加算した値で指定されるアドレスにジャンプします (5 サイクル)。 サインとオーバフローフラグの排他的論理和の結果およびゼロフラグの内容がともに “0” のときは、ジャンプステータスフラグを “1” にセットし、次命令の実行に移ります (3 サイクル)。			
JR SGT,$+3+d	1 1 1 0 1 0 0 0 1 1 0 1 0 1 0 1 d d d d d d d d	1 – – – – –	5/3(t/f)	if ZF ∣ (SF ^ VF)=0 then PC ← PC+d else null
	サインとオーバフローフラグの排他的論理和の結果およびゼロフラグの内容がともに “0” のとき、プログラムカウンタ PC の内容に、オブジェクトコード中の 8 ビットディスプレースメント d を符号拡張して (–128~+127) 加算した値で指定されるアドレスにジャンプします (5 サイクル)。 サインとオーバフローフラグの排他的論理和の結果またはゼロフラグの内容が “1” のときは、ジャンプステータスフラグを “1” にセットし、次命令の実行に移ります (3 サイクル)。			

ニモニック	オブジェクトコード (2 進)	フラグ J Z C H S V	サイクル	オペレーション
JR VS,$+3+d	1 1 1 0 1 0 0 0 | 1 1 0 1 0 1 1 0 | d d d d d d d d	1 – – – – –	5/3(t/f)	if VF=1 then PC ← PC+d else null
	オーバフローフラグが “1” のとき、プログラムカウンタ PC の内容に、オブジェクトコード中の 8 ビットディスプレースメント d を符号拡張して (–128~+127) 加算した値で指定されるアドレスにジャンプします (5 サイクル)。 オーバフローフラグが “1” のときは、ジャンプステータスフラグを “1” にセットし、次命令の実行に移ります (3 サイクル)。			
JR VC,$+3+d	1 1 1 0 1 0 0 0 | 1 1 0 1 0 1 1 1 | d d d d d d d d	1 – – – – –	5/3(t/f)	if VF=0 then PC ← PC+d else null
	オーバフローフラグが “0” のとき、プログラムカウンタ PC の内容に、オブジェクトコード中の 8 ビットディスプレースメント d を符号拡張して (–128~+127) 加算した値で指定されるアドレスにジャンプします (5 サイクル)。 オーバフローフラグが “1” のときは、ジャンプステータスフラグを “1” にセットし、次命令の実行に移ります (3 サイクル)。			
JR $+2+d	1 1 1 1 1 1 0 0 | d d d d d d d d	1 – – – – –	4	PC ← PC+d
	プログラムカウンタ PC の内容 (JR 命令の置かれているアドレスの 2 番地先を示しています) に、オブジェクトコード中の 8 ビットディスプレースメント d を符号拡張して (–128~+127) 加算した値で指定されるアドレスに無条件にジャンプします。 例 : 0xD5A7 番地に置かれた JR $+0x73 命令を実行すると、0xD61C 番地にジャンプします。			
JP mn	1 1 1 1 1 1 1 0 | n n n n n n n n | m m m m m m m m	– – – – – –	4	PC ← mn
	オブジェクトコード中の 16 ビットデータ mn で直接指定されるアドレスに無条件にジャンプします。			
JP gg	1 1 1 0 1 g g g | 1 1 1 1 1 1 1 0	– – – – – –	3	PC ← gg
	16 ビットレジスタ gg で指定されるアドレスに無条件にジャンプします。 例 : HL = 0xE325 のとき、JP HL 命令を実行すると、0xE325 番地にジャンプします。			
JP (x)	1 1 1 0 0 0 0 0 | x x x x x x x x | 1 1 1 1 1 1 1 0	– – – – – –	6	PC ← (x+1,x)
	オブジェクトコード中の x で直接指定されるアドレス (0x0000 ~ 0x00FF 番地) から連続する 2 バイトのメモリに格納されているアドレスに無条件にジャンプします。 例 : 0x0085, 0x0086 番地の内容がそれぞれ 0x27, 0xC3 のとき、JP (0x85) 命令を実行すると、0xC327 番地にジャンプします。			
JP (vw)	1 1 1 0 0 0 0 1 | w w w w w w w w | v v v v v v v v 1 1 1 1 1 1 1 0	– – – – – –	7	PC ← (vw+1,vw)
	オブジェクトコード中の vw で直接指定されるアドレス (0x0000 ~ 0xFFFF 番地) から連続する 2 バイトのメモリに格納されているアドレスに無条件にジャンプします。			
JP (DE)	1 1 1 0 0 0 1 0 | 1 1 1 1 1 1 1 0	– – – – – –	5	PC ← (DE+1,DE)
	レジスタペア DE で指定されるアドレスから連続する 2 バイトのメモリに格納されているアドレスに無条件にジャンプします。 例 : DE = 0x0125 で、0x0125, 0x0126 番地の内容がそれぞれ 0x87, 0xE5 のとき、JP (DE) 命令を実行すると、0xE587 番地にジャンプします。			
JP (HL)	1 1 1 0 0 0 1 1 | 1 1 1 1 1 1 1 0	– – – – – –	5	PC ← (HL+1,HL)
	レジスタペア HL で指定されるアドレスから連続する 2 バイトのメモリに格納されているアドレスに無条件にジャンプします。 例 : HL= 0x0125 で、0x0125, 0x0126 番地の内容がそれぞれ 0x87, 0xE5 のとき、JP (HL) 命令を実行すると、0xE587 番地にジャンプします。			
JP (IX)	1 1 1 0 0 1 0 0 | 1 1 1 1 1 1 1 0	– – – – – –	5	PC ← (IX+1,IX)
	インデックスレジスタ IX で指定されるアドレスから連続する 2 バイトのメモリに格納されているアドレスに無条件にジャンプします。 例 : IX = 0x0125 で、0x0125, 0x0126 番地の内容がそれぞれ 0x87, 0xE5 のとき、JP (IX) 命令を実行すると、0xE587 番地にジャンプします。			
JP (IY)	1 1 1 0 0 1 0 1 | 1 1 1 1 1 1 1 0	– – – – – –	5	PC ← (IY+1,IY)
	インデックスレジスタ IY で指定されるアドレスから連続する 2 バイトのメモリに格納されているアドレスに無条件にジャンプします。 例 : IY = 0x0125 で、0x0125, 0x0126 番地の内容がそれぞれ 0x87, 0xE5 のとき、JP (IY) 命令を実行すると、0xE587 番地にジャンプします。			
JP (IX+d)	1 1 0 1 0 1 0 0 | d d d d d d d d | 1 1 1 1 1 1 1 0	– – – – – –	7	PC ← (IX+d+1,IX+d)
	インデックスレジスタ IX の内容に、オブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリに格納されているアドレスに無条件にジャンプします。			
JP (IY+d)	1 1 0 1 0 1 0 1 | d d d d d d d d | 1 1 1 1 1 1 1 0	– – – – – –	7	PC ← (IY+d+1,IY+d)
	インデックスレジスタ IY の内容に、オブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリに格納されているアドレスに無条件にジャンプします。			
JP (SP+d)	1 1 0 1 0 1 1 0 | d d d d d d d d | 1 1 1 1 1 1 1 0	– – – – – –	7	PC ← (SP+d+1,SP+d)
	スタックポインタ SP の内容に、オブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリに格納されているアドレスに無条件にジャンプします。			
JP (HL+d)	1 1 0 1 0 1 1 1 | d d d d d d d d | 1 1 1 1 1 1 1 0	– – – – – –	7	PC ← (HL+d+1,HL+d)
	レジスタペア HL の内容に、オブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリに格納されているアドレスに無条件にジャンプします。			
JP (HL+C)	1 1 1 0 0 1 1 1 | 1 1 1 1 1 1 1 0	– – – – – –	7	PC ← (HL+C+1,HL+C)
	レジスタペア HL の内容に、C レジスタの内容を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリに格納されているアドレスに無条件にジャンプします。			
JP (+SP)	1 1 1 0 0 1 1 0 | 1 1 1 1 1 1 1 0	– – – – – –	6	SP ← SP+1:PC ← (SP+1,SP)
	スタックポインタ SP をインクリメントし、その値で指定されるアドレスから連続する 2 バイトのメモリに格納されているアドレスに、無条件にジャンプします。			
JP (PC+A) 注	0 1 0 0 1 1 1 1 | 1 1 1 1 1 1 1 0	– – – – – –	7	PC ← (PC+A+1,PC+A)
	プログラムカウンタ PC の内容 (JP 命令の置かれているアドレスの 2 番地先を示しています) に、A レジスタの内容を符号拡張 して加算した値で指定されるアドレスから連続する 2 バイトのプログラムメモリに格納されているアドレスに無条件にジャンプします。この命令は、多方向分岐処理に適しています。			

注) (PC+A) を使用した命令には制限があります。詳細は、TLCS-870/C1 シリーズ 命令セット 1.4 アドレッシングモードを参照してください。

2.6 サブルーチンコール、リターン、ソフトウエア割り込み、ノーオペレーション

ニモニック	オブジェクトコード (2 進)				フラグ J Z C H S V	サイクル	オペレーション
CALLV n	0 1 1 1 n n n n				– – – – – –	7	(SP,SP-1) ← NxtOp:SP ← SP-2: PC ← (n × 2+ ベクタコール領域の先頭アドレス値 +1,n × 2+ ベクタコール領域の先頭アドレス値)
	ベクタ方式のサブルーチンコールを行います。戻り番地 NxtOp(CALLV 命令の次の命令の先頭アドレス) を上位バイト、下位バイトの順にスタックポインタ SP で指定されるアドレスのデータメモリにセーブし、スタックポインタ SP の内容を 2 回デクリメントします。その後、オブジェクトコード中の 4 ビット即値 n (0~15) を 2 倍した値にベクタコール領域の先頭アドレス値を加えた値で指定されるアドレスから連続する 2 バイトのプログラムメモリの内容 (ベクタ) をプログラムカウンタ PC に入れます。この命令は、プログラムの圧縮に便利です。 例 : SP = 0x0123 で、4 × 2+ ベクタコール領域の先頭アドレス値 +1,4 × 2+ ベクタコール領域の先頭アドレス値の内容がそれぞれ 0x45, 0xE6 のとき、0xD1A4 番地に置かれた CALLV 4H 命令を実行すると、0x0123, 0x0122 番地にそれぞれ 0xD1, 0xA5 が書き込まれ、SP = 0x0121 となり、0xE645 番地をコールします。						
CALL mn	1 1 1 1 1 1 0 1	n n n n n n n n	m m m m m m m m		– – – – – –	6	(SP,SP-1) ← NxtOp:SP ← SP-2: PC ← mn
	アブソリュート方式のサブルーチンコールを行います。戻り番地 NxtOp(CALL 命令の次の命令の先頭アドレス) を上位バイト、下位バイトの順にスタックポインタ SP で指定されるアドレスのデータメモリにセーブし、スタックポインタ SP の内容を 2 回デクリメントします。その後、オブジェクトコード中の 16 ビット絶対アドレス mn をプログラムカウンタ PC に入れます。 例 : SP = 0x0138 のとき、0xE273 番地に置かれた CALL 0xCE05 命令を実行すると、0x0138, 0x0137 番地にそれぞれ 0xE2, 0x76 が書き込まれ、SP = 0x0136 となり、0xCE05 番地をコールします。						
CALL gg	1 1 1 0 1 g g g	1 1 1 1 1 1 0 1			– – – – – –	6	(SP,SP-1) ← NxtOp:SP ← SP-2: PC ← gg
	戻り番地 NxtOp(CALL 命令の次の命令の先頭アドレス) を上位バイト、下位バイトの順にスタックポインタ SP で指定されるアドレスのデータメモリにセーブし、スタックポインタ SP の内容を 2 回デクリメントします。その後、16 ビットレジスタ gg の内容をプログラムカウンタ PC に入れます。 例 : SP = 0x0126, HL = 0xC8A7 のとき、0xD491 番地に置かれた CALL HL 命令を実行すると、0x0126, 0x0125 番地にそれぞれ 0xD4, 0x93 が書き込まれ、SP = 0x0124 となり、0xC8A7 番地をコールします。						
CALL (x)	1 1 1 0 0 0 0 0	x x x x x x x x	1 1 1 1 1 1 0 1		– – – – – –	8	(SP,SP-1) ← NxtOp:SP ← SP-2: PC ← (x+1,x)
	戻り番地 NxtOp(CALL 命令の次の命令の先頭アドレス) を上位バイト、下位バイトの順にスタックポインタ SP で指定されるアドレスのデータメモリにセーブし、スタックポインタ SP の内容を 2 回デクリメントします。その後、オブジェクトコード中の x で直接指定されるアドレス (0x0000 ~ 0x00FF 番地) から連続する 2 バイトのメモリ内容 (サブルーチンのエントリーアドレス) をプログラムカウンタ PC に入れます。						
CALL (vw)	1 1 1 0 0 0 0 1 1 1 1 1 1 1 0 1	w w w w w w w w	v v v v v v v v		– – – – – –	9	(SP,SP-1) ← NxtOp:SP ← SP-2: PC ← (vw+1,vw)
	戻り番地 NxtOp(CALL 命令の次の命令の先頭アドレス) を上位バイト、下位バイトの順にスタックポインタ SP で指定されるアドレスのデータメモリにセーブし、スタックポインタ SP の内容を 2 回デクリメントします。その後、オブジェクトコード中の vw で直接指定されるアドレス (0x0000 ~ 0xFFFF 番地) から連続する 2 バイトのメモリ内容をプログラムカウンタ PC に入れます。						
CALL (DE)	1 1 1 0 0 0 1 0	1 1 1 1 1 1 0 1			– – – – – –	7	(SP,SP-1) ← NxtOp:SP ← SP-2: PC ← (DE+1,DE)
	戻り番地 NxtOp(CALL 命令の次の命令の先頭アドレス) を上位バイト、下位バイトの順にスタックポインタ SP で指定されるアドレスのデータメモリにセーブし、スタックポインタ SP の内容を 2 回デクリメントします。その後、レジスタペア DE の内容で指定されるアドレスから連続する 2 バイトのメモリ内容をプログラムカウンタ PC に入れます。						
CALL (HL)	1 1 1 0 0 0 1 1	1 1 1 1 1 1 0 1			– – – – – –	7	(SP,SP-1) ← NxtOp:SP ← SP-2: PC ← (HL+1,HL)
	戻り番地 NxtOp(CALL 命令の次の命令の先頭アドレス) を上位バイト、下位バイトの順にスタックポインタ SP で指定されるアドレスのデータメモリにセーブし、スタックポインタ SP の内容を 2 回デクリメントします。その後、レジスタペア HL の内容で指定されるアドレスから連続する 2 バイトのメモリ内容をプログラムカウンタ PC に入れます。						
CALL (IX)	1 1 1 0 0 1 0 0	1 1 1 1 1 1 0 1			– – – – – –	7	(SP,SP-1) ← NxtOp:SP ← SP-2: PC ← (IX+1,IX)
	戻り番地 NxtOp(CALL 命令の次の命令の先頭アドレス) を上位バイト、下位バイトの順にスタックポインタ SP で指定されるアドレスのデータメモリにセーブし、スタックポインタ SP の内容を 2 回デクリメントします。その後、インデックスレジスタ IX の内容で指定されるアドレスから連続する 2 バイトのメモリ内容をプログラムカウンタ PC に入れます。						
CALL (IY)	1 1 1 0 0 1 0 1	1 1 1 1 1 1 0 1			– – – – – –	7	(SP,SP-1) ← NxtOp:SP ← SP-2: PC ← (IY+1,IY)
	戻り番地 NxtOp(CALL 命令の次の命令の先頭アドレス) を上位バイト、下位バイトの順にスタックポインタ SP で指定されるアドレスのデータメモリにセーブし、スタックポインタ SP の内容を 2 回デクリメントします。その後、インデックスレジスタ IY の内容で指定されるアドレスから連続する 2 バイトのメモリ内容をプログラムカウンタ PC に入れます。						
CALL (IX+d)	1 1 0 1 0 1 0 0	d d d d d d d d	1 1 1 1 1 1 0 1		– – – – – –	9	(SP,SP-1) ← NxtOp:SP ← SP-2: PC ← (IX+d+1,IX+d)
	戻り番地 NxtOp(CALL 命令の次の命令の先頭アドレス) を上位バイト、下位バイトの順にスタックポインタ SP で指定されるアドレスのデータメモリにセーブし、スタックポインタ SP の内容を 2 回デクリメントします。その後、インデックスレジスタ IX の内容に、オブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容をプログラムカウンタ PC に入れます。						
CALL (IY+d)	1 1 0 1 0 1 0 1	d d d d d d d d	1 1 1 1 1 1 0 1		– – – – – –	9	(SP,SP-1) ← NxtOp:SP ← SP-2: PC ← (IY+d+1,IY+d)
	戻り番地 NxtOp(CALL 命令の次の命令の先頭アドレス) を上位バイト、下位バイトの順にスタックポインタ SP で指定されるアドレスのデータメモリにセーブし、スタックポインタ SP の内容を 2 回デクリメントします。その後、インデックスレジスタ IY の内容に、オブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容をプログラムカウンタ PC に入れます。						

ニモニック	オブジェクトコード (2 進)				フラグ J Z C H S V	サイクル	オペレーション
CALL (SP+d)	1 1 0 1 0 1 1 0	d d d d d d d d	1 1 1 1 1 1 0 1		– – – – – –	9	(SP,SP-1) ← NxtOp:SP ← SP-2: PC ← (SP+d+1,SP+d)
	戻り番地 NxtOp(CALL 命令の次の命令の先頭アドレス) を上位バイト、下位バイトの順にスタックポインタ SP で指定されるアドレスのデータメモリにセーブし、スタックポインタ SP の内容を 2 回デクリメントします。その後、スタックポインタ SP の内容に、オブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容をプログラムカウンタ PC に入れます。						
CALL (HL+d)	1 1 0 1 0 1 1 1	d d d d d d d d	1 1 1 1 1 1 0 1		– – – – – –	9	(SP,SP-1) ← NxtOp:SP ← SP-2:
	戻り番地 NxtOp(CALL 命令の次の命令の先頭アドレス) を上位バイト、下位バイトの順にスタックポインタ SP で指定されるアドレスのデータメモリにセーブし、スタックポインタ SP の内容を 2 回デクリメントします。その後、レジスタペア HL の内容に、オブジェクトコード中の 8 ビットデータ d を符号拡張して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容をプログラムカウンタ PC に入れます。						
CALL (+SP)	1 1 1 0 0 1 1 0	1 1 1 1 1 1 0 1			– – – – – –	8	SP ← SP+1:(SP-1) ← $NxtOp_L$: PC_L ← (SP):(SP) ← $NxtOp_H$: PC_H ← (SP+1):SP ← SP-2
	スタックポインタ SP の内容をインクリメントします。次に、戻り番地 NxtOp(CALL 命令の次の命令の先頭アドレス) の下位バイトをスタックポインタ SP の内容 -1 で指定されるアドレスのメモリにセーブし、スタックポインタ SP の内容で指定されるアドレスのメモリの内容をプログラムカウンタ PC の下位バイトに入れます。さらに、戻り番地 NxtOP(CALL 命令の次の命令の先頭アドレス) の上位バイトをスタックポインタ SP の内容で指定されるアドレスのメモリにセーブし、スタックポインタ SP+1 の内容で指定されるアドレスのメモリの内容をプログラムカウンタ PC の上位バイトに入れます。その後、スタックポインタ SP の内容を 2 回デクリメントします。						
CALL (HL+C)	1 1 1 0 0 1 1 1	1 1 1 1 1 1 0 1			– – – – – –	9	(SP,SP-1) ← NxtOp:SP ← SP-2: PC ← (HL+C+1,HL+C)
	戻り番地 NxtOp(CALL 命令の次の命令の先頭アドレス) を上位バイト、下位バイトの順にスタックポインタ SP で指定されるアドレスのデータメモリにセーブし、スタックポインタ SP の内容を 2 回デクリメントします。その後、レジスタペア HL の内容に C レジスタの内容を符号拡張 (最上位ビットが符号ビットです) して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容をプログラムカウンタ PC に入れます。						
CALL (PC+A) 注	0 1 0 0 1 1 1 1	1 1 1 1 1 1 0 1			– – – – – –	9	(SP,SP-1) ← NxtOp:SP ← SP-2:
	戻り番地 NxtOp(CALL 命令の次の命令の先頭アドレス) を上位バイト、下位バイトの順にスタックポインタ SP で指定されるアドレスのデータメモリにセーブし、スタックポインタ SP の内容を 2 回デクリメントします。その後、プログラムカウンタ PC の内容に A レジスタの内容を符号拡張 (最上位ビットが符号ビットです) して加算した値で指定されるアドレスから連続する 2 バイトのメモリ内容をプログラムカウンタ PC に入れます。						
RET	1 1 1 1 1 0 1 0				– – – – – –	6	SP ← SP+2:PC ← (SP,SP-1)
	サブルーチンからのリターンを行います。すなわち、スタックポインタ SP の内容を 2 回インクリメントし、その値で指定されるアドレスのデータメモリ内容 (戻り番地) を下位バイト、上位バイトの順にプログラムカウンタ PC にロードします。 例 : SP = 0x0123 で、0x0124, 0x0125 番地の内容がそれぞれ 0x3E, 0xC8 のとき、この命令を実行すると、SP = 0x0125 となり 0xC83E 番地にジャンプします。						
RETI	1 1 1 1 1 0 1 1				* * * * * *	6	SP ← SP+2:PC ← (SP,SP-1): SP ← SP+1:PSW ← (SP): IMF ← (SP).0
	マスカブル割り込みサービスルーチンからのリターンを行います。すなわち、スタックポインタ SP の内容を 2 回インクリメントし、その値で指定されるアドレスのデータメモリ内容 (戻り番地) をプログラムカウンタ PC に入れます。その後、さらにスタックポインタ SP の内容をインクリメントし、その値で指定されるアドレスのメモリ内容をプログラム ステータス ワード PSW および割り込みマスタ許可フラグ IMF に入れます (ビット 7~1 が PSW に、ビット 0 が IMF に入ります)。						
RETN	1 1 1 0 1 0 0 0	1 1 1 1 1 0 1 1			* * * * * *	7	SP ← SP+2:PC ← (SP,SP-1): SP ← SP+1:PSW ← (SP): IMF ← (SP).0
	ノンマスカブル割り込みサービスルーチンからのリターンを行います。すなわち、スタックポインタ SP の内容を 2 回インクリメントし、その値で指定されるアドレスのデータメモリ内容 (戻り番地) をプログラムカウンタ PC に入れます。その後、さらにスタックポインタ SP の内容をインクリメントし、その値で指定されるアドレスの内容をプログラム ステータス ワード PSW および割り込みマスタ許可フラグ IMF に入れます (ビット 7~1 が PSW に、ビット 0 が IMF に入ります)。						
SWI	1 1 1 1 1 1 1 1				– – – – – –	9	(SP) ← PSW:(SP).0 ← IMF: (SP-1,SP-2) ← NxtOp: PC ← (SWI ベクタアドレス +1,SWI ベクタアドレス)
	ソフトウエア割り込みを行います。すなわち、プログラムステータスワード PSW および割り込みマスタ許可フラグ IMF の内容をスタックポインタ SP の内容で指定されるアドレスのデータメモリにセーブし (PSW の内容がメモリのビット 7~1 に、IMF の内容がメモリのビット 0 にセーブされます)、スタックポインタ SP の内容をデクリメントします。次に 戻り番地 NxtOp(SWI 命令の次の命令の先頭アドレス) を上位バイト、下位バイトの順にスタックポインタ SP の内容で指定されるアドレスのデータメモリにセーブし、スタックポインタ SP の内容をさらに 2 回デクリメントします。割り込みマスタ許可フラグ IMF は “0” にクリアされます。その後、SWI ベクタアドレスと SWI ベクタアドレス +1 のプログラムメモリの内容 (ベクタ) をプログラムカウンタ PC に入れます。 例 : SP = 0x0128, PSW = 0x46 で、SWI ベクタアドレスと SWI ベクタアドレス +1 の内容がそれぞれ 0x03, 0xE7 のとき、0xCA74 番地に置かれた SWI 命令を実行すると、0x0128, 0x0127, 0x0126 番地にそれぞれ、0x46, 0x75, 0xCA が書き込まれ、SP = 0x0125, IMF = 0 となり、0xE703 番地にジャンプします。						
NOP	0 0 0 0 0 0 0 0				– – – – – –	1	no Operation
	ノーオペレーション (何も実行せず、次の命令の実行に移ります)。						

注) (PC+A) を使用した命令には制限があります。詳細は、TLCS-870/C1 シリーズ 命令セット 1.4 アドレッシングモードを参照してください。

第 3 章　TLCS-870/C1 命令一覧

3.1　転送、交換

ニモニック	オブジェクトコード (2 進)					フラグ J Z C H S V	サイクル	オペレーション
LD A,r	0 0 0 1 0 r r r					1 Z – – – –	1	A ← r
LD r,A	0 1 0 0 0 r r r					1 Z – – – –	1	r ← A
LD r,g	1 1 1 0 1 g g g	0 1 0 0 0 r r r				1 Z – – – –	2	r ← g
LD rr,gg	1 1 1 0 1 g g g	0 1 0 0 1 r r r				1 – – – – –	2	rr ← gg
LD A,(x)	0 0 0 0 1 1 0 0	x x x x x x x x				1 Z – – – –	3	A ← (x)
LD A,(HL)	0 0 0 0 1 1 0 1					1 Z – – – –	2	A ← (HL)
LD r,(x)	1 1 1 0 0 0 0 0	x x x x x x x x	0 1 0 0 0 r r r			1 Z – – – –	4	r ← (x)
LD r,(vw)	1 1 1 0 0 0 0 1	w w w w w w w w	v v v v v v v v	0 1 0 0 0 r r r		1 Z – – – –	5	r ← (vw)
LD r,(DE)	1 1 1 0 0 0 1 0	0 1 0 0 0 r r r				1 Z – – – –	3	r ← (DE)
LD r,(HL)	1 1 1 0 0 0 1 1	0 1 0 0 0 r r r				1 Z – – – –	3	r ← (HL)
LD r,(IX)	1 1 1 0 0 1 0 0	0 1 0 0 0 r r r				1 Z – – – –	3	r ← (IX)
LD r,(IY)	1 1 1 0 0 1 0 1	0 1 0 0 0 r r r				1 Z – – – –	3	r ← (IY)
LD r,(IX+d)	1 1 0 1 0 1 0 0	d d d d d d d d	0 1 0 0 0 r r r			1 Z – – – –	5	r ← (IX+d)
LD r,(IY+d)	1 1 0 1 0 1 0 1	d d d d d d d d	0 1 0 0 0 r r r			1 Z – – – –	5	r ← (IY+d)
LD r,(SP+d)	1 1 0 1 0 1 1 0	d d d d d d d d	0 1 0 0 0 r r r			1 Z – – – –	5	r ← (SP+d)
LD r,(HL+d)	1 1 0 1 0 1 1 1	d d d d d d d d	0 1 0 0 0 r r r			1 Z – – – –	5	r ← (HL+d)
LD r,(HL+C)	1 1 1 0 0 1 1 1	0 1 0 0 0 r r r				1 Z – – – –	5	r ← (HL+C)
LD r,(+SP)	1 1 1 0 0 1 1 0	0 1 0 0 0 r r r				1 Z – – – –	4	SP ← SP+1:r ← (SP)
LD r,(PC+A) 注	0 1 0 0 1 1 1 1	0 1 0 0 0 r r r				1 Z – – – –	5	r ← (PC+A)
LD rr,(x)	1 1 1 0 0 0 0 0	x x x x x x x x	0 1 0 0 1 r r r			1 – – – – –	5	rr ← (x+1,x)
LD rr,(vw)	1 1 1 0 0 0 0 1	w w w w w w w w	v v v v v v v v	0 1 0 0 1 r r r		1 – – – – –	6	rr ← (vw+1,vw)
LD rr,(DE)	1 1 1 0 0 0 1 0	0 1 0 0 1 r r r				1 – – – – –	4	rr ← (DE+1,DE)
LD rr,(HL)	1 1 1 0 0 0 1 1	0 1 0 0 1 r r r				1 – – – – –	4	rr ← (HL+1,HL)
LD rr,(IX)	1 1 1 0 0 1 0 0	0 1 0 0 1 r r r				1 – – – – –	4	rr ← (IX+1,IX)
LD rr,(IY)	1 1 1 0 0 1 0 1	0 1 0 0 1 r r r				1 – – – – –	4	rr ← (IY+1,IY)
LD rr,(IX+d)	1 1 0 1 0 1 0 0	d d d d d d d d	0 1 0 0 1 r r r			1 – – – – –	6	rr ← (IX+d+1,IX+d)
LD rr,(IY+d)	1 1 0 1 0 1 0 1	d d d d d d d d	0 1 0 0 1 r r r			1 – – – – –	6	rr ← (IY+d+1,IY+d)
LD rr,(SP+d)	1 1 0 1 0 1 1 0	d d d d d d d d	0 1 0 0 1 r r r			1 – – – – –	6	rr ← (SP+d+1,SP+d)
LD rr,(HL+d)	1 1 0 1 0 1 1 1	d d d d d d d d	0 1 0 0 1 r r r			1 – – – – –	6	rr ← (HL+d+1,HL+d)
LD rr,(HL+C)	1 1 1 0 0 1 1 1	0 1 0 0 1 r r r				1 – – – – –	6	rr ← (HL+C+1,HL+C)
LD rr,(+SP)	1 1 1 0 0 1 1 0	0 1 0 0 1 r r r				1 – – – – –	5	SP ← SP+1:rr ← (SP+1,SP)
LD rr,(PC+A) 注	0 1 0 0 1 1 1 1	0 1 0 0 1 r r r				1 – – – – –	6	rr ← (PC+A+1,PC+A)
LD (x),A	0 0 0 0 1 1 1 0	x x x x x x x x				1 – – – – –	3	(x) ← A
LD (HL),A	0 0 0 0 1 1 1 1					1 – – – – –	2	(HL) ← A
LD (x),r	1 1 1 1 0 0 0 0	x x x x x x x x	0 1 1 1 1 r r r			1 – – – – –	4	(x) ← r
LD (vw),r	1 1 1 1 0 0 0 1	w w w w w w w w	v v v v v v v v	0 1 1 1 1 r r r		1 – – – – –	5	(vw) ← r
LD (DE),r	1 1 1 1 0 0 1 0	0 1 1 1 1 r r r				1 – – – – –	3	(DE) ← r
LD (HL),r	1 1 1 1 0 0 1 1	0 1 1 1 1 r r r				1 – – – – –	3	(HL) ← r
LD (IX),r	1 1 1 1 0 1 0 0	0 1 1 1 1 r r r				1 – – – – –	3	(IX) ← r
LD (IY),r	1 1 1 1 0 1 0 1	0 1 1 1 1 r r r				1 – – – – –	3	(IY) ← r
LD (IX+d),r	0 1 0 1 0 1 0 0	d d d d d d d d	0 1 1 1 1 r r r			1 – – – – –	4	(IX+d) ← r
LD (IY+d),r	0 1 0 1 0 1 0 1	d d d d d d d d	0 1 1 1 1 r r r			1 – – – – –	4	(IY+d) ← r
LD (SP+d),r	0 1 0 1 0 1 1 0	d d d d d d d d	0 1 1 1 1 r r r			1 – – – – –	4	(SP+d) ← r
LD (HL+d),r	0 1 0 1 0 1 1 1	d d d d d d d d	0 1 1 1 1 r r r			1 – – – – –	4	(HL+d) ← r
LD (HL+C),r	1 1 1 1 0 1 1 1	0 1 1 1 1 r r r				1 – – – – –	5	(HL+C) ← r
LD (SP–),r	1 1 1 1 0 1 1 0	0 1 1 1 1 r r r				1 – – – – –	4	(SP) ← r:SP ← SP–1
LD (x),rr	1 1 1 1 0 0 0 0	x x x x x x x x	0 1 1 0 1 r r r			1 – – – – –	5	(x+1,x) ← rr
LD (vw),rr	1 1 1 1 0 0 0 1	w w w w w w w w	v v v v v v v v	0 1 1 0 1 r r r		1 – – – – –	6	(vw+1,vw) ← rr
LD (DE),rr	1 1 1 1 0 0 1 0	0 1 1 0 1 r r r				1 – – – – –	4	(DE+1,DE) ← rr
LD (HL),rr	1 1 1 1 0 0 1 1	0 1 1 0 1 r r r				1 – – – – –	4	(HL+1,HL) ← rr
LD (IX),rr	1 1 1 1 0 1 0 0	0 1 1 0 1 r r r				1 – – – – –	4	(IX+1,IX) ← rr
LD (IY),rr	1 1 1 1 0 1 0 1	0 1 1 0 1 r r r				1 – – – – –	4	(IY+1,IY) ← rr
LD (IX+d),rr	0 1 0 1 0 1 0 0	d d d d d d d d	0 1 1 0 1 r r r			1 – – – – –	5	(IX+d+1,IX+d) ← rr
LD (IY+d),rr	0 1 0 1 0 1 0 1	d d d d d d d d	0 1 1 0 1 r r r			1 – – – – –	5	(IY+d+1,IY+d) ← rr
LD (SP+d),rr	0 1 0 1 0 1 1 0	d d d d d d d d	0 1 1 0 1 r r r			1 – – – – –	5	(SP+d+1,SP+d) ← rr
LD (HL+d),rr	0 1 0 1 0 1 1 1	d d d d d d d d	0 1 1 0 1 r r r			1 – – – – –	5	(HL+d+1,HL+d) ← rr
LD (HL+C),rr	1 1 1 1 0 1 1 1	0 1 1 0 1 r r r				1 – – – – –	6	(HL+C+1,HL+C) ← rr
LD (SP–),rr	1 1 1 1 0 1 1 0	0 1 1 0 1 r r r				1 – – – – –	5	(SP+1,SP) ← rr:SP ← SP–1

ニモニック	オブジェクトコード (2 進)	フラグ J Z C H S V	サイクル	オペレーション
LD r,n	0 0 0 1 1 r r r n n n n n n n n	1 – – – – –	2	r ← n
LD rr,mn	0 1 0 0 1 r r r n n n n n n n n m m m m m m m m	1 – – – – –	3	rr ← mn
LD (x),n	0 0 0 0 1 0 1 0 x x x x x x x x n n n n n n n n	1 – – – – –	4	(x) ← n
LD (vw),n	1 1 1 1 0 0 0 1 w w w w w w w w v v v v v v v v	1 – – – – –	6	(vw) ← n
	1 1 1 1 1 0 0 1 n n n n n n n n			
LD (DE),n	1 1 1 1 0 0 1 0 1 1 1 1 1 0 0 1 n n n n n n n n	1 – – – – –	4	(DE) ← n
LD (HL),n	0 0 0 0 1 0 1 1 n n n n n n n n	1 – – – – –	3	(HL) ← n
LD (IX),n	1 1 1 1 0 1 0 0 1 1 1 1 1 0 0 1 n n n n n n n n	1 – – – – –	4	(IX) ← n
LD (IY),n	1 1 1 1 0 1 0 1 1 1 1 1 1 0 0 1 n n n n n n n n	1 – – – – –	4	(IY) ← n
LD (IX+d),n	0 1 0 1 0 1 0 0 d d d d d d d d 1 1 1 1 1 0 0 1	1 – – – – –	5	(IX+d) ← n
	n n n n n n n n			
LD (IY+d),n	0 1 0 1 0 1 0 1 d d d d d d d d 1 1 1 1 1 0 0 1	1 – – – – –	5	(IY+d) ← n
	n n n n n n n n			
LD (SP+d),n	0 1 0 1 0 1 1 0 d d d d d d d d 1 1 1 1 1 0 0 1	1 – – – – –	5	(SP+d) ← n
	n n n n n n n n			
LD (HL+d),n	0 1 0 1 0 1 1 1 d d d d d d d d 1 1 1 1 1 0 0 1	1 – – – – –	5	(HL+d) ← n
	n n n n n n n n			
LD (HL+C),n	1 1 1 1 0 1 1 1 1 1 1 1 1 0 0 1 n n n n n n n n	1 – – – – –	6	(HL+C) ← n
LD (SP–),n	1 1 1 1 0 1 1 0 1 1 1 1 1 0 0 1 n n n n n n n n	1 – – – – –	5	(SP) ← n:SP ← SP–1
LDW (x),mn	0 0 0 0 1 0 0 0 x x x x x x x x n n n n n n n n	1 – – – – –	6	(x+1,x) ← mn
	m m m m m m m m			
LDW (HL),mn	0 0 0 0 1 0 0 1 n n n n n n n n m m m m m m m m	1 – – – – –	5	(HL+1,HL) ← mn
PUSH rr #1	0 1 0 1 0 0 r r	– – – – – –	3	(SP, SP–1) ← rr:SP ← SP–2
PUSH gg #2	1 1 1 0 1 g g g 1 1 0 1 1 0 0 0	– – – – – –	4	(SP, SP–1) ← gg:SP ← SP–2
POP rr #1	1 1 0 1 0 0 r r	– – – – – –	4	SP ← SP+2:rr ← (SP, SP–1)
POP gg #2	1 1 1 0 1 g g g 1 1 0 1 1 0 0 1	– – – – – –	5	SP ← SP+2:gg ← (SP, SP–1)
PUSH PSW	1 1 1 0 1 0 0 0 1 1 0 1 1 1 0 0	– – – – – –	3	(SP) ← PSW:SP ← SP–1
POP PSW	1 1 1 0 1 0 0 0 1 1 0 1 1 1 0 1	* * * * * *	4	SP ← SP+1:PSW ← (SP)
LD PSW, n	1 1 1 0 1 0 0 0 1 1 0 1 1 1 1 0 n n n n n n n n	* * * * * *	3	PSW ← n
LD RBS,0	1 1 1 1 1 0 0 1 0 0 0 0 0 0 0 0	– – – – – –	2	RBS ← 0
LD RBS,1	1 1 1 1 1 0 0 1 0 0 0 0 0 0 1 0	– – – – – –	2	RBS ← 1
LD SP,SP+d	0 0 1 1 0 1 1 1 d d d d d d d d	1 – – – – –	2	SP ← SP+d
LD SP,SP–d	0 0 1 1 1 1 1 1 d d d d d d d d	1 – – – – –	2	SP ← SP–d
XCH r,g	1 1 1 0 1 g g g 0 1 1 1 0 r r r	1 Z – – – –	3	r ↔ g
XCH rr,gg	1 1 1 0 1 g g g 0 1 1 1 1 r r r	1 – – – – –	3	rr ↔ gg
XCH r,(x)	1 1 1 0 0 0 0 0 x x x x x x x x 0 1 1 1 0 r r r	1 Z – – – –	5	r ↔ (x)
XCH r,(vw)	1 1 1 0 0 0 0 1 w w w w w w w w v v v v v v v v	1 Z – – – –	6	r ↔ (vw)
	0 1 1 1 0 r r r			
XCH r,(DE)	1 1 1 0 0 0 1 0 0 1 1 1 0 r r r	1 Z – – – –	4	r ↔ (DE)
XCH r,(HL)	1 1 1 0 0 0 1 1 0 1 1 1 0 r r r	1 Z – – – –	4	r ↔ (HL)
XCH r,(IX)	1 1 1 0 0 1 0 0 0 1 1 1 0 r r r	1 Z – – – –	4	r ↔ (IX)
XCH r,(IY)	1 1 1 0 0 1 0 1 0 1 1 1 0 r r r	1 Z – – – –	4	r ↔ (IY)
XCH r,(IX+d)	1 1 0 1 0 1 0 0 d d d d d d d d 0 1 1 1 0 r r r	1 Z – – – –	6	r ↔ (IX+d)
XCH r,(IY+d)	1 1 0 1 0 1 0 1 d d d d d d d d 0 1 1 1 0 r r r	1 Z – – – –	6	r ↔ (IY+d)
XCH r,(SP+d)	1 1 0 1 0 1 1 0 d d d d d d d d 0 1 1 1 0 r r r	1 Z – – – –	6	r ↔ (SP+d)
XCH r,(HL+d)	1 1 0 1 0 1 1 1 d d d d d d d d 0 1 1 1 0 r r r	1 Z – – – –	6	r ↔ (HL+d)
XCH r,(HL+C)	1 1 1 0 0 1 1 1 0 1 1 1 0 r r r	1 Z – – – –	6	r ↔ (HL+C)
XCH r,(+SP)	1 1 1 0 0 1 1 0 0 1 1 1 0 r r r	1 Z – – – –	5	SP ← SP+1:r ↔ (SP)
XCH r,(PC+A) 注	0 1 0 0 1 1 1 1 0 1 1 1 0 r r r	1 Z – – – –	6	r ↔ (PC+A)
XCH rr,(x)	1 1 1 0 0 0 0 0 x x x x x x x x 1 1 0 1 1 r r r	1 – – – – –	7	rr ↔ (x+1,x)
XCH rr,(vw)	1 1 1 0 0 0 0 1 w w w w w w w w v v v v v v v v	1 – – – – –	8	rr ↔ (vw+1,vw)
	1 1 0 1 1 r r r			
XCH rr,(DE)	1 1 1 0 0 0 1 0 1 1 0 1 1 r r r	1 – – – – –	6	rr ↔ (DE+1,DE)
XCH rr,(HL)	1 1 1 0 0 0 1 1 1 1 0 1 1 r r r	1 – – – – –	6	rr ↔ (HL+1,HL)
XCH rr,(IX)	1 1 1 0 0 1 0 0 1 1 0 1 1 r r r	1 – – – – –	6	rr ↔ (IX+1,IX)
XCH rr,(IY)	1 1 1 0 0 1 0 1 1 1 0 1 1 r r r	1 – – – – –	6	rr ↔ (IY+1,IY)
XCH rr,(IX+d)	1 1 0 1 0 1 0 0 d d d d d d d d 1 1 0 1 1 r r r	1 – – – – –	8	rr ↔ (IX+d+1, IX+d)
XCH rr,(IY+d)	1 1 0 1 0 1 0 1 d d d d d d d d 1 1 0 1 1 r r r	1 – – – – –	8	rr ↔ (IY+d+1, IY+d)
XCH rr,(SP+d)	1 1 0 1 0 1 1 0 d d d d d d d d 1 1 0 1 1 r r r	1 – – – – –	8	rr ↔ (SP+d+1,SP+d)
XCH rr,(HL+d)	1 1 0 1 0 1 1 1 d d d d d d d d 1 1 0 1 1 r r r	1 – – – – –	8	rr ↔ (HL+d+1,HL+d)
XCH rr,(HL+C)	1 1 1 0 0 1 1 1 1 1 0 1 1 r r r	1 – – – – –	8	rr ↔ (HL+C+1,HL+C)
XCH rr,(+SP)	1 1 1 0 0 1 1 0 1 1 0 1 1 r r r	1 – – – – –	7	SP ← SP+1:rr ↔ (SP+1,SP)
XCH rr,(PC+A) 注	0 1 0 0 1 1 1 1 1 1 0 1 1 r r r	1 – – – – –	8	rr ↔ (PC+A+1,PC+A)

#1　rr は WA, BC, DE, HL のみ
#2　gg は IX,IY のみ

注)　(PC+A) を使用した命令には制限があります。詳細は、TLCS-870/C1 シリーズ 命令セット 1.4 アドレッシングモードを参照してください。

3.2 演算

ニモニック	オブジェクトコード (2 進)	フラグ J Z C H S V	サイクル	オペレーション
CMP A,n	0 1 1 0 0 1 1 1 n n n n n n n n	Z Z C H S V	2	A–n
CMP g,n	1 1 1 0 1 g g g 0 1 1 0 0 1 1 1 n n n n n n n n	Z Z C H S V	3	g–n
CMP gg,mn	1 1 1 0 1 g g g 0 1 1 0 1 1 1 1 n n n n n n n n m m m m m m m m	Z Z C U S V	4	gg–mn
CMP r,g	1 1 1 0 1 g g g 0 0 r r r 1 1 1	Z Z C H S V	2	r–g
CMP rr,gg	1 1 1 0 1 g g g 1 0 r r r 1 1 1	Z Z C U S V	3	rr–gg
CMP r,(x)	1 1 1 0 0 0 0 0 x x x x x x x x 0 0 r r r 1 1 1	Z Z C H S V	4	r–(x)
CMP r,(vw)	1 1 1 0 0 0 0 1 w w w w w w w w v v v v v v v v 0 0 r r r 1 1 1	Z Z C H S V	5	r–(vw)
CMP r,(DE)	1 1 1 0 0 0 1 0 0 0 r r r 1 1 1	Z Z C H S V	3	r–(DE)
CMP r,(HL)	1 1 1 0 0 0 1 1 0 0 r r r 1 1 1	Z Z C H S V	3	r–(HL)
CMP r,(IX)	1 1 1 0 0 1 0 0 0 0 r r r 1 1 1	Z Z C H S V	3	r–(IX)
CMP r,(IY)	1 1 1 0 0 1 0 1 0 0 r r r 1 1 1	Z Z C H S V	3	r–(IY)
CMP r,(IX+d)	1 1 0 1 0 1 0 0 d d d d d d d d 0 0 r r r 1 1 1	Z Z C H S V	5	r–(IX+d)
CMP r,(IY+d)	1 1 0 1 0 1 0 1 d d d d d d d d 0 0 r r r 1 1 1	Z Z C H S V	5	r–(IY+d)
CMP r,(SP+d)	1 1 0 1 0 1 1 0 d d d d d d d d 0 0 r r r 1 1 1	Z Z C H S V	5	r–(SP+d)
CMP r,(HL+d)	1 1 0 1 0 1 1 1 d d d d d d d d 0 0 r r r 1 1 1	Z Z C H S V	5	r–(HL+d)
CMP r,(HL+C)	1 1 1 0 0 1 1 1 0 0 r r r 1 1 1	Z Z C H S V	5	r–(HL+C)
CMP r,(+SP)	1 1 1 0 0 1 1 0 0 0 r r r 1 1 1	Z Z C H S V	4	SP ← SP+1:r–(SP)
CMP r,(PC+A) 注	0 1 0 0 1 1 1 1 0 0 r r r 1 1 1	Z Z C H S V	5	r–(PC+A)
CMP (x),n	0 0 0 0 0 1 1 1 x x x x x x x x n n n n n n n n	Z Z C H S V	4	(x)–n
CMP (vw),n	1 1 1 0 0 0 0 1 w w w w w w w w v v v v v v v v 0 1 1 0 0 1 1 1 n n n n n n n n	Z Z C H S V	6	(vw)–n
CMP (DE),n	1 1 1 0 0 0 1 0 0 1 1 0 0 1 1 1 n n n n n n n n	Z Z C H S V	4	(DE)–n
CMP (HL),n	1 1 1 0 0 0 1 1 0 1 1 0 0 1 1 1 n n n n n n n n	Z Z C H S V	4	(HL)–n
CMP (IX),n	1 1 1 0 0 1 0 0 0 1 1 0 0 1 1 1 n n n n n n n n	Z Z C H S V	4	(IX)–n
CMP (IY),n	1 1 1 0 0 1 0 1 0 1 1 0 0 1 1 1 n n n n n n n n	Z Z C H S V	4	(IY)–n
CMP (IX+d),n	1 1 0 1 0 1 0 0 d d d d d d d d 0 1 1 0 0 1 1 1 n n n n n n n n	Z Z C H S V	6	(IX+d)–n
CMP (IY+d),n	1 1 0 1 0 1 0 1 d d d d d d d d 0 1 1 0 0 1 1 1 n n n n n n n n	Z Z C H S V	6	(IY+d)–n
CMP (SP+d),n	1 1 0 1 0 1 1 0 d d d d d d d d 0 1 1 0 0 1 1 1 n n n n n n n n	Z Z C H S V	6	(SP+d)–n
CMP (HL+d),n	1 1 0 1 0 1 1 1 d d d d d d d d 0 1 1 0 0 1 1 1 n n n n n n n n	Z Z C H S V	6	(HL+d)–n
CMP (HL+C),n	1 1 1 0 0 1 1 1 0 1 1 0 0 1 1 1 n n n n n n n n	Z Z C H S V	6	(HL+C)–n
CMP (+SP),n	1 1 1 0 0 1 1 0 0 1 1 0 0 1 1 1 n n n n n n n n	Z Z C H S V	5	SP ← SP+1:(SP)–n
CMP (PC+A),n 注	0 1 0 0 1 1 1 1 0 1 1 0 0 1 1 1 n n n n n n n n	Z Z C H S V	6	(PC+A)–n
CMP rr,(x)	1 1 1 0 0 0 0 0 x x x x x x x x 1 0 r r r 1 1 1	Z Z C U S V	5	rr–(x)
CMP rr,(vw)	1 1 1 0 0 0 0 1 w w w w w w w w v v v v v v v v 1 0 r r r 1 1 1	Z Z C U S V	6	rr–(vw)
CMP rr,(DE)	1 1 1 0 0 0 1 0 1 0 r r r 1 1 1	Z Z C U S V	4	rr–(DE)
CMP rr,(HL)	1 1 1 0 0 0 1 1 1 0 r r r 1 1 1	Z Z C U S V	4	rr–(HL)
CMP rr,(IX)	1 1 1 0 0 1 0 0 1 0 r r r 1 1 1	Z Z C U S V	4	rr–(IX)
CMP rr,(IY)	1 1 1 0 0 1 0 1 1 0 r r r 1 1 1	Z Z C U S V	4	rr–(IY)
CMP rr,(IX+d)	1 1 0 1 0 1 0 0 d d d d d d d d 1 0 r r r 1 1 1	Z Z C U S V	6	rr–(IX+d)
CMP rr,(IY+d)	1 1 0 1 0 1 0 1 d d d d d d d d 1 0 r r r 1 1 1	Z Z C U S V	6	rr–(IY+d)
CMP rr,(SP+d)	1 1 0 1 0 1 1 0 d d d d d d d d 1 0 r r r 1 1 1	Z Z C U S V	6	rr–(SP+d)
CMP rr,(HL+d)	1 1 0 1 0 1 1 1 d d d d d d d d 1 0 r r r 1 1 1	Z Z C U S V	6	rr–(HL+d)
CMP rr,(HL+C)	1 1 1 0 0 1 1 1 1 0 r r r 1 1 1	Z Z C U S V	6	rr–(HL+C)
CMP rr,(+SP)	1 1 1 0 0 1 1 0 1 0 r r r 1 1 1	Z Z C U S V	5	SP ← SP+1:rr–(SP)
CMP rr,(PC+A) 注	0 1 0 0 1 1 1 1 1 0 r r r 1 1 1	Z Z C U S V	6	rr–(PC+A)
ADD A,n	0 1 1 0 0 0 0 1 n n n n n n n n	C Z C H S V	2	A ← A+n
ADD g,n	1 1 1 0 1 g g g 0 1 1 0 0 0 0 1 n n n n n n n n	C Z C H S V	3	g ← g+n
ADD gg,mn	1 1 1 0 1 g g g 0 1 1 0 1 0 0 1 n n n n n n n n m m m m m m m m	C Z C U S V	4	gg ← gg+mn
ADD r,g	1 1 1 0 1 g g g 0 0 r r r 0 0 1	C Z C H S V	2	r ← r+g
ADD rr,gg	1 1 1 0 1 g g g 1 0 r r r 0 0 1	C Z C U S V	3	rr ← rr+gg
ADD r,(x)	1 1 1 0 0 0 0 0 x x x x x x x x 0 0 r r r 0 0 1	C Z C H S V	4	r ← r+(x)
ADD r,(vw)	1 1 1 0 0 0 0 1 w w w w w w w w v v v v v v v v 0 0 r r r 0 0 1	C Z C H S V	5	r ← r+(vw)
ADD r,(DE)	1 1 1 0 0 0 1 0 0 0 r r r 0 0 1	C Z C H S V	3	r ← r+(DE)
ADD r,(HL)	1 1 1 0 0 0 1 1 0 0 r r r 0 0 1	C Z C H S V	3	r ← r+(HL)
ADD r,(IX)	1 1 1 0 0 1 0 0 0 0 r r r 0 0 1	C Z C H S V	3	r ← r+(IX)
ADD r,(IY)	1 1 1 0 0 1 0 1 0 0 r r r 0 0 1	C Z C H S V	3	r ← r+(IY)
ADD r,(IX+d)	1 1 0 1 0 1 0 0 d d d d d d d d 0 0 r r r 0 0 1	C Z C H S V	5	r ← r+(IX+d)
ADD r,(IY+d)	1 1 0 1 0 1 0 1 d d d d d d d d 0 0 r r r 0 0 1	C Z C H S V	5	r ← r+(IY+d)
ADD r,(SP+d)	1 1 0 1 0 1 1 0 d d d d d d d d 0 0 r r r 0 0 1	C Z C H S V	5	r ← r+(SP+d)

ニモニック	オブジェクトコード (2 進)	フラグ J Z C H S V	サイクル	オペレーション
ADD r,(HL+d)	1 1 0 1 0 1 1 1 d d d d d d d d 0 0 r r r 0 0 1	C Z C H S V	5	r ← r+(HL+d)
ADD r,(HL+C)	1 1 1 0 0 1 1 1 0 0 r r r 0 0 1	C Z C H S V	5	r ← r+(HL+C)
ADD r,(+SP)	1 1 1 0 0 1 1 0 0 0 r r r 0 0 1	C Z C H S V	4	SP ← SP+1:r ← r+(SP)
ADD r,(PC+A) 注	0 1 0 0 1 1 1 1 0 0 r r r 0 0 1	C Z C H S V	5	r ← r+(PC+A)
ADD (x),n	1 1 1 0 0 0 0 0 x x x x x x x x 0 1 1 0 0 0 0 1 n n n n n n n n	C Z C H S V	6	(x) ← (x)+n
ADD (vw),n	1 1 1 0 0 0 0 1 w w w w w w w w v v v v v v v v 0 1 1 0 0 0 0 1 n n n n n n n n	C Z C H S V	7	(vw) ← (vw)+n
ADD (DE),n	1 1 1 0 0 0 1 0 0 1 1 0 0 0 0 1 n n n n n n n n	C Z C H S V	5	(DE) ← (DE)+n
ADD (HL),n	1 1 1 0 0 0 1 1 0 1 1 0 0 0 0 1 n n n n n n n n	C Z C H S V	5	(HL) ← (HL)+n
ADD (IX),n	1 1 1 0 0 1 0 0 0 1 1 0 0 0 0 1 n n n n n n n n	C Z C H S V	5	(IX) ← (IX)+n
ADD (IY),n	1 1 1 0 0 1 0 1 0 1 1 0 0 0 0 1 n n n n n n n n	C Z C H S V	5	(IY) ← (IY)+n
ADD (IX+d),n	1 1 0 1 0 1 0 0 d d d d d d d d 0 1 1 0 0 0 0 1 n n n n n n n n	C Z C H S V	7	(IX+d) ← (IX+d)+n
ADD (IY+d),n	1 1 0 1 0 1 0 1 d d d d d d d d 0 1 1 0 0 0 0 1 n n n n n n n n	C Z C H S V	7	(IY+d) ← (IY+d)+n
ADD (SP+d),n	1 1 0 1 0 1 1 0 d d d d d d d d 0 1 1 0 0 0 0 1 n n n n n n n n	C Z C H S V	7	(SP+d) ← (SP+d)+n
ADD (HL+d),n	1 1 0 1 0 1 1 1 d d d d d d d d 0 1 1 0 0 0 0 1 n n n n n n n n	C Z C H S V	7	(HL+d) ← (HL+d)+n
ADD (HL+C),n	1 1 1 0 0 1 1 1 0 1 1 0 0 0 0 1 n n n n n n n n	C Z C H S V	7	(HL+C) ← (HL+C)+n
ADD (+SP),n	1 1 1 0 0 1 1 0 0 1 1 0 0 0 0 1 n n n n n n n n	C Z C H S V	6	SP ← SP+1:(SP) ← (SP)+n
ADD (PC+A),n 注	0 1 0 0 1 1 1 1 0 1 1 0 0 0 0 1 n n n n n n n n	C Z C H S V	7	(PC+A) ← (PC+A)+n
ADD rr,(x)	1 1 1 0 0 0 0 0 x x x x x x x x 1 0 r r r 0 0 1	C Z C U S V	5	rr ← rr+(x+1, x)
ADD rr,(vw)	1 1 1 0 0 0 0 1 w w w w w w w w v v v v v v v v 1 0 r r r 0 0 1	C Z C U S V	6	rr ← rr+(vw+1, vw)
ADD rr,(DE)	1 1 1 0 0 0 1 0 1 0 r r r 0 0 1	C Z C U S V	4	rr ← rr+(DE+1, DE)
ADD rr,(HL)	1 1 1 0 0 0 1 1 1 0 r r r 0 0 1	C Z C U S V	4	rr ← rr+(HL+1, HL)
ADD rr,(IX)	1 1 1 0 0 1 0 0 1 0 r r r 0 0 1	C Z C U S V	4	rr ← rr+(IX+1, IX)
ADD rr,(IY)	1 1 1 0 0 1 0 1 1 0 r r r 0 0 1	C Z C U S V	4	rr ← rr+(IY+1, IY)
ADD rr,(IX+d)	1 1 0 1 0 1 0 0 d d d d d d d d 1 0 r r r 0 0 1	C Z C U S V	6	rr ← rr+(IX+d+1, IX+d)
ADD rr,(IY+d)	1 1 0 1 0 1 0 1 d d d d d d d d 1 0 r r r 0 0 1	C Z C U S V	6	rr ← rr+(IY+d+1, IY+d)
ADD rr,(SP+d)	1 1 0 1 0 1 1 0 d d d d d d d d 1 0 r r r 0 0 1	C Z C U S V	6	rr ← rr+(SP+d+1, SP+d)
ADD rr,(HL+d)	1 1 0 1 0 1 1 1 d d d d d d d d 1 0 r r r 0 0 1	C Z C U S V	6	rr ← rr+(HL+d+1, HL+d)
ADD rr,(HL+C)	1 1 1 0 0 1 1 1 1 0 r r r 0 0 1	C Z C U S V	6	rr ← rr+(HL+C+1, HL+C)
ADD rr,(+SP)	1 1 1 0 0 1 1 0 1 0 r r r 0 0 1	C Z C U S V	5	SP ← SP+1:rr ← rr+(SP+1, SP)
ADD rr,(PC+A) 注	0 1 0 0 1 1 1 1 1 0 r r r 0 0 1	C Z C U S V	6	rr ← rr+(PC+A+1, PC+A)
ADDC A,n	0 1 1 0 0 0 0 0 n n n n n n n n	C Z C H S V	2	A ← A+n+CF
ADDC g,n	1 1 1 0 1 g g g 0 1 1 0 0 0 0 0 n n n n n n n n	C Z C H S V	3	g ← g+n+CF
ADDC gg,mn	1 1 1 0 1 g g g 0 1 1 0 1 0 0 0 n n n n n n n n m m m m m m m m	C Z C U S V	4	gg ← gg+mn+CF
ADDC r,g	1 1 1 0 1 g g g 0 0 r r r 0 0 0	C Z C H S V	2	r ← r+g+CF
ADDC rr,gg	1 1 1 0 1 g g g 1 0 r r r 0 0 0	C Z C U S V	3	rr ← rr+gg+CF
ADDC r,(x)	1 1 1 0 0 0 0 0 x x x x x x x x 0 0 r r r 0 0 0	C Z C H S V	4	r ← r+(x)+CF
ADDC r,(vw)	1 1 1 0 0 0 0 1 w w w w w w w w v v v v v v v v 0 0 r r r 0 0 0	C Z C H S V	5	r ← r+(vw)+CF
ADDC r,(DE)	1 1 1 0 0 0 1 0 0 0 r r r 0 0 0	C Z C H S V	3	r ← r+(DE)+CF
ADDC r,(HL)	1 1 1 0 0 0 1 1 0 0 r r r 0 0 0	C Z C H S V	3	r ← r+(HL)+CF
ADDC r,(IX)	1 1 1 0 0 1 0 0 0 0 r r r 0 0 0	C Z C H S V	3	r ← r+(IX)+CF
ADDC r,(IY)	1 1 1 0 0 1 0 1 0 0 r r r 0 0 0	C Z C H S V	3	r ← r+(IY)+CF
ADDC r,(IX+d)	1 1 0 1 0 1 0 0 d d d d d d d d 0 0 r r r 0 0 0	C Z C H S V	5	r ← r+(IX+d)+CF
ADDC r,(IY+d)	1 1 0 1 0 1 0 1 d d d d d d d d 0 0 r r r 0 0 0	C Z C H S V	5	r ← r+(IY+d)+CF
ADDC r,(SP+d)	1 1 0 1 0 1 1 0 d d d d d d d d 0 0 r r r 0 0 0	C Z C H S V	5	r ← r+(SP+d)+CF
ADDC r,(HL+d)	1 1 0 1 0 1 1 1 d d d d d d d d 0 0 r r r 0 0 0	C Z C H S V	5	r ← r+(HL+d)+CF
ADDC r,(HL+C)	1 1 1 0 0 1 1 1 0 0 r r r 0 0 0	C Z C H S V	5	r ← r+(HL+C)+CF
ADDC r,(+SP)	1 1 1 0 0 1 1 0 0 0 r r r 0 0 0	C Z C H S V	4	SP ← SP+1:r ← r+(SP)+CF
ADDC r,(PC+A) 注	0 1 0 0 1 1 1 1 0 0 r r r 0 0 0	C Z C H S V	5	r ← r+(PC+A)+CF
ADDC (x),n	1 1 1 0 0 0 0 0 x x x x x x x x 0 1 1 0 0 0 0 0 n n n n n n n n	C Z C H S V	6	(x) ← (x)+n+CF
ADDC (vw),n	1 1 1 0 0 0 0 1 w w w w w w w w v v v v v v v v 0 1 1 0 0 0 0 0 n n n n n n n n	C Z C H S V	7	(vw) ← (vw)+n+CF
ADDC (DE),n	1 1 1 0 0 0 1 0 0 1 1 0 0 0 0 0 n n n n n n n n	C Z C H S V	5	(DE) ← (DE)+n+CF
ADDC (HL),n	1 1 1 0 0 0 1 1 0 1 1 0 0 0 0 0 n n n n n n n n	C Z C H S V	5	(HL) ← (HL)+n+CF
ADDC (IX),n	1 1 1 0 0 1 0 0 0 1 1 0 0 0 0 0 n n n n n n n n	C Z C H S V	5	(IX) ← (IX)+n+CF
ADDC (IY),n	1 1 1 0 0 1 0 1 0 1 1 0 0 0 0 0 n n n n n n n n	C Z C H S V	5	(IY) ← (IY)+n+CF
ADDC (IX+d),n	1 1 0 1 0 1 0 0 d d d d d d d d 0 1 1 0 0 0 0 0 n n n n n n n n	C Z C H S V	7	(IX+d) ← (IX+d)+n+CF
ADDC (IY+d),n	1 1 0 1 0 1 0 1 d d d d d d d d 0 1 1 0 0 0 0 0 n n n n n n n n	C Z C H S V	7	(IY+d) ← (IY+d)+n+CF
ADDC (SP+d),n	1 1 0 1 0 1 1 0 d d d d d d d d 0 1 1 0 0 0 0 0 n n n n n n n n	C Z C H S V	7	(SP+d) ← (SP+d)+n+CF
ADDC (HL+d),n	1 1 0 1 0 1 1 1 d d d d d d d d 0 1 1 0 0 0 0 0	C Z C H S V	7	(HL+d) ← (HL+d)+n+CF

ニモニック	オブジェクトコード (2 進)																フラグ J Z C H S V	サイクル	オペレーション
	n n n n	n n n n																	
ADDC (HL+C),n	1 1 1 0	0 1 1 1	0 1 1 0	0 0 0 0	n n n n	n n n n											C Z C H S V	7	(HL+C) ← (HL+C)+n+CF
ADDC (+SP),n	1 1 1 0	0 1 1 0	0 1 1 0	0 0 0 0	n n n n	n n n n											C Z C H S V	6	SP ← SP+1:(SP) ← (SP)+n+CF
ADDC (PC+A),n 注	0 1 0 0	1 1 1 1	0 1 1 0	0 0 0 0	n n n n	n n n n											C Z C H S V	7	(PC+A) ← (PC+A)+n+CF
ADDC rr,(x)	1 1 1 0	0 0 0 0	x x x x	x x x x	1 0 r r	r 0 0 0											C Z C U S V	5	rr ← rr+(x+1, x)+CF
ADDC rr,(vw)	1 1 1 0	0 0 0 1	w w w w	w w w w	v v v v	v v v v											C Z C U S V	6	rr ← rr+(vw+1, vw)+CF
	1 0 r r	r 0 0 0																	
ADDC rr,(DE)	1 1 1 0	0 0 1 0	1 0 r r	r 0 0 0													C Z C U S V	4	rr ← rr+(DE+1, DE)+CF
ADDC rr,(HL)	1 1 1 0	0 0 1 1	1 0 r r	r 0 0 0													C Z C U S V	4	rr ← rr+(HL+1, HL)+CF
ADDC rr,(IX)	1 1 1 0	0 1 0 0	1 0 r r	r 0 0 0													C Z C U S V	4	rr ← rr+(IX+1, IX)+CF
ADDC rr,(IY)	1 1 1 0	0 1 0 1	1 0 r r	r 0 0 0													C Z C U S V	4	rr ← rr+(IY+1, IY)+CF
ADDC rr,(IX+d)	1 1 0 1	0 1 0 0	d d d d	d d d d	1 0 r r	r 0 0 0											C Z C U S V	6	rr ← rr+(IX+d+1, IX+d)+CF
ADDC rr,(IY+d)	1 1 0 1	0 1 0 1	d d d d	d d d d	1 0 r r	r 0 0 0											C Z C U S V	6	rr ← rr+(IY+d+1, IY+d)+CF
ADDC rr,(SP+d)	1 1 0 1	0 1 1 0	d d d d	d d d d	1 0 r r	r 0 0 0											C Z C U S V	6	rr ← rr+(SP+d+1, SP+d)+CF
ADDC rr,(HL+d)	1 1 0 1	0 1 1 1	d d d d	d d d d	1 0 r r	r 0 0 0											C Z C U S V	6	rr ← rr+(HL+d+1, HL+d)+CF
ADDC rr,(HL+C)	1 1 1 0	0 1 1 1	1 0 r r	r 0 0 0													C Z C U S V	6	rr ← rr+(HL+C+1, HL+C)+CF
ADDC rr,(+SP)	1 1 1 0	0 1 1 0	1 0 r r	r 0 0 0													C Z C U S V	5	SP ← SP+1:rr ← rr+(SP+1, SP)+CF
ADDC rr,(PC+A) 注	0 1 0 0	1 1 1 1	1 0 r r	r 0 0 0													C Z C U S V	6	rr ← rr+(PC+A+1, PC+A)+CF
SUB A,n	0 1 1 0	0 0 1 1	n n n n	n n n n													C Z C H S V	2	A ← A-n
SUB g,n	1 1 1 0	1 g g g	0 1 1 0	0 0 1 1	n n n n	n n n n											C Z C H S V	3	g ← g-n
SUB gg,mn	1 1 1 0	1 g g g	0 1 1 0	1 0 1 1	n n n n	n n n n											C Z C U S V	4	gg ← gg-mn
	m m m m	m m m m																	
SUB r,g	1 1 1 0	1 g g g	0 0 r r	r 0 1 1													C Z C H S V	2	r ← r-g
SUB rr,gg	1 1 1 0	1 g g g	1 0 r r	r 0 1 1													C Z C U S V	3	rr ← rr-gg
SUB r,(x)	1 1 1 0	0 0 0 0	x x x x	x x x x	0 0 r r	r 0 1 1											C Z C H S V	4	r ← r-(x)
SUB r,(vw)	1 1 1 0	0 0 0 1	w w w w	w w w w	v v v v	v v v v											C Z C H S V	5	r ← r-(vw)
	0 0 r r	r 0 1 1																	
SUB r,(DE)	1 1 1 0	0 0 1 0	0 0 r r	r 0 1 1													C Z C H S V	3	r ← r-(DE)
SUB r,(HL)	1 1 1 0	0 0 1 1	0 0 r r	r 0 1 1													C Z C H S V	3	r ← r-(HL)
SUB r,(IX)	1 1 1 0	0 1 0 0	0 0 r r	r 0 1 1													C Z C H S V	3	r ← r-(IX)
SUB r,(IY)	1 1 1 0	0 1 0 1	0 0 r r	r 0 1 1													C Z C H S V	3	r ← r-(IY)
SUB r,(IX+d)	1 1 0 1	0 1 0 0	d d d d	d d d d	0 0 r r	r 0 1 1											C Z C H S V	5	r ← r-(IX+d)
SUB r,(IY+d)	1 1 0 1	0 1 0 1	d d d d	d d d d	0 0 r r	r 0 1 1											C Z C H S V	5	r ← r-(IY+d)
SUB r,(SP+d)	1 1 0 1	0 1 1 0	d d d d	d d d d	0 0 r r	r 0 1 1											C Z C H S V	5	r ← r-(SP+d)
SUB r,(HL+d)	1 1 0 1	0 1 1 1	d d d d	d d d d	0 0 r r	r 0 1 1											C Z C H S V	5	r ← r-(HL+d)
SUB r,(HL+C)	1 1 1 0	0 1 1 1	0 0 r r	r 0 1 1													C Z C H S V	5	r ← r-(HL+C)
SUB r,(+SP)	1 1 1 0	0 1 1 0	0 0 r r	r 0 1 1													C Z C H S V	4	SP ← SP+1:r ← r-(SP)
SUB r,(PC+A) 注	0 1 0 0	1 1 1 1	0 0 r r	r 0 1 1													C Z C H S V	5	r ← r-(PC+A)
SUB (x),n	1 1 1 0	0 0 0 0	x x x x	x x x x	0 1 1 0	0 0 1 1											C Z C H S V	6	(x) ← (x)-n
	n n n n	n n n n																	
SUB (vw),n	1 1 1 0	0 0 0 1	w w w w	w w w w	v v v v	v v v v											C Z C H S V	7	(vw) ← (vw)-n
	0 1 1 0	0 0 1 1	n n n n	n n n n															
SUB (DE),n	1 1 1 0	0 0 1 0	0 1 1 0	0 0 1 1	n n n n	n n n n											C Z C H S V	5	(DE) ← (DE)-n
SUB (HL),n	1 1 1 0	0 0 1 1	0 1 1 0	0 0 1 1	n n n n	n n n n											C Z C H S V	5	(HL) ← (HL)-n
SUB (IX),n	1 1 1 0	0 1 0 0	0 1 1 0	0 0 1 1	n n n n	n n n n											C Z C H S V	5	(IX) ← (IX)-n
SUB (IY),n	1 1 1 0	0 1 0 1	0 1 1 0	0 0 1 1	n n n n	n n n n											C Z C H S V	5	(IY) ← (IY)-n
SUB (IX+d),n	1 1 0 1	0 1 0 0	d d d d	d d d d	0 1 1 0	0 0 1 1											C Z C H S V	7	(IX+d) ← (IX+d)-n
	n n n n	n n n n																	
SUB (IY+d),n	1 1 0 1	0 1 0 1	d d d d	d d d d	0 1 1 0	0 0 1 1											C Z C H S V	7	(IY+d) ← (IY+d)-n
	n n n n	n n n n																	
SUB (SP+d),n	1 1 0 1	0 1 1 0	d d d d	d d d d	0 1 1 0	0 0 1 1											C Z C H S V	7	(SP+d) ← (SP+d)-n
	n n n n	n n n n																	
SUB (HL+d),n	1 1 0 1	0 1 1 1	d d d d	d d d d	0 1 1 0	0 0 1 1											C Z C H S V	7	(HL+d) ← (HL+d)-n
	n n n n	n n n n																	
SUB (HL+C),n	1 1 1 0	0 1 1 1	0 1 1 0	0 0 1 1	n n n n	n n n n											C Z C H S V	7	(HL+C) ← (HL+C)-n
SUB (+SP),n	1 1 1 0	0 1 1 0	0 1 1 0	0 0 1 1	n n n n	n n n n											C Z C H S V	6	SP ← SP+1:(SP) ← (SP)-n
SUB (PC+A),n 注	0 1 0 0	1 1 1 1	0 1 1 0	0 0 1 1	n n n n	n n n n											C Z C H S V	7	(PC+A) ← (PC+A)-n
SUB rr,(x)	1 1 1 0	0 0 0 0	x x x x	x x x x	1 0 r r	r 0 1 1											C Z C U S V	5	rr ← rr-(x+1, x)
SUB rr,(vw)	1 1 1 0	0 0 0 1	w w w w	w w w w	v v v v	v v v v											C Z C U S V	6	rr ← rr-(vw+1, vw)
	1 0 r r	r 0 1 1																	
SUB rr,(DE)	1 1 1 0	0 0 1 0	1 0 r r	r 0 1 1													C Z C U S V	4	rr ← rr-(DE+1, DE)
SUB rr,(HL)	1 1 1 0	0 0 1 1	1 0 r r	r 0 1 1													C Z C U S V	4	rr ← rr-(HL+1, HL)
SUB rr,(IX)	1 1 1 0	0 1 0 0	1 0 r r	r 0 1 1													C Z C U S V	4	rr ← rr-(IX+1, IX)
SUB rr,(IY)	1 1 1 0	0 1 0 1	1 0 r r	r 0 1 1													C Z C U S V	4	rr ← rr-(IY+1, IY)
SUB rr,(IX+d)	1 1 0 1	0 1 0 0	d d d d	d d d d	1 0 r r	r 0 1 1											C Z C U S V	6	rr ← rr-(IX+d+1, IX+d)
SUB rr,(IY+d)	1 1 0 1	0 1 0 1	d d d d	d d d d	1 0 r r	r 0 1 1											C Z C U S V	6	rr ← rr-(IY+d+1, IY+d)
SUB rr,(SP+d)	1 1 0 1	0 1 1 0	d d d d	d d d d	1 0 r r	r 0 1 1											C Z C U S V	6	rr ← rr-(SP+d+1, SP+d)
SUB rr,(HL+d)	1 1 0 1	0 1 1 1	d d d d	d d d d	1 0 r r	r 0 1 1											C Z C U S V	6	rr ← rr-(HL+d+1, HL+d)
SUB rr,(HL+C)	1 1 1 0	0 1 1 1	1 0 r r	r 0 1 1													C Z C U S V	6	rr ← rr-(HL+C+1, HL+C)
SUB rr,(+SP)	1 1 1 0	0 1 1 0	1 0 r r	r 0 1 1													C Z C U S V	5	SP ← SP+1:rr ← rr-(SP+1, SP)
SUB rr,(PC+A) 注	0 1 0 0	1 1 1 1	1 0 r r	r 0 1 1													C Z C U S V	6	rr ← rr-(PC+A+1, PC+A)
SUBB A,n	0 1 1 0	0 0 1 0	n n n n	n n n n													C Z C H S V	2	A ← A-n-CF

ニモニック	オブジェクトコード (2 進)				フラグ J Z C H S V	サイクル	オペレーション
SUBB g,n	1 1 1 0 1 g g g	0 1 1 0 0 0 1 0	n n n n n n n n		C Z C H S V	3	g ← g-n-CF
SUBB gg,mn	1 1 1 0 1 g g g m m m m m m m m	0 1 1 0 1 0 1 0	n n n n n n n n		C Z C U S V	4	gg ← gg-mn-CF
SUBB r,g	1 1 1 0 1 g g g	0 0 r r r 0 1 0			C Z C H S V	2	r ← r-g-CF
SUBB rr,gg	1 1 1 0 1 g g g	1 0 r r r 0 1 0			C Z C U S V	3	rr ← rr-gg-CF
SUBB r,(x)	1 1 1 0 0 0 0 0	x x x x x x x x	0 0 r r r 0 1 0		C Z C H S V	4	r ← r-(x)-CF
SUBB r,(vw)	1 1 1 0 0 0 0 1 0 0 r r r 0 1 0	w w w w w w w w	v v v v v v v v		C Z C H S V	5	r ← r-(vw)-CF
SUBB r,(DE)	1 1 1 0 0 0 1 0	0 0 r r r 0 1 0			C Z C H S V	3	r ← r-(DE)-CF
SUBB r,(HL)	1 1 1 0 0 0 1 1	0 0 r r r 0 1 0			C Z C H S V	3	r ← r-(HL)-CF
SUBB r,(IX)	1 1 1 0 0 1 0 0	0 0 r r r 0 1 0			C Z C H S V	3	r ← r-(IX)-CF
SUBB r,(IY)	1 1 1 0 0 1 0 1	0 0 r r r 0 1 0			C Z C H S V	3	r ← r-(IY)-CF
SUBB r,(IX+d)	1 1 0 1 0 1 0 0	d d d d d d d d	0 0 r r r 0 1 0		C Z C H S V	5	r ← r-(IX+d)-CF
SUBB r,(IY+d)	1 1 0 1 0 1 0 1	d d d d d d d d	0 0 r r r 0 1 0		C Z C H S V	5	r ← r-(IY+d)-CF
SUBB r,(SP+d)	1 1 0 1 0 1 1 0	d d d d d d d d	0 0 r r r 0 1 0		C Z C H S V	5	r ← r-(SP+d)-CF
SUBB r,(HL+d)	1 1 0 1 0 1 1 1	d d d d d d d d	0 0 r r r 0 1 0		C Z C H S V	5	r ← r-(HL+d)-CF
SUBB r,(HL+C)	1 1 1 0 0 1 1 1	0 0 r r r 0 1 0			C Z C H S V	5	r ← r-(HL+C)-CF
SUBB r,(+SP)	1 1 1 0 0 1 1 0	0 0 r r r 0 1 0			C Z C H S V	4	SP ← SP+1:r ← r-(SP)-CF
SUBB r,(PC+A) 注	0 1 0 0 1 1 1 1	0 0 r r r 0 1 0			C Z C H S V	5	r ← r-(PC+A)-CF
SUBB (x),n	1 1 1 0 0 0 0 0 n n n n n n n n	x x x x x x x x	0 1 1 0	0 0 1 0	C Z C H S V	6	(x) ← (x)-n-CF
SUBB (vw),n	1 1 1 0 0 0 0 1 0 1 1 0 0 0 1 0 n n n n n n n n	w w w w w w w w	v v v v	v v v v	C Z C H S V	7	(vw) ← (vw)-n-CF
SUBB (DE),n	1 1 1 0 0 0 1 0 n n n n n n n n	0 1 1 0 0 0 1 0			C Z C H S V	5	(DE) ← (DE)-n-CF
SUBB (HL),n	1 1 1 0 0 0 1 1 n n n n n n n n	0 1 1 0 0 0 1 0			C Z C H S V	5	(HL) ← (HL)-n-CF
SUBB (IX),n	1 1 1 0 0 1 0 0 n n n n n n n n	0 1 1 0 0 0 1 0			C Z C H S V	5	(IX) ← (IX)-n-CF
SUBB (IY),n	1 1 1 0 0 1 0 1 n n n n n n n n	0 1 1 0 0 0 1 0			C Z C H S V	5	(IY) ← (IY)-n-CF
SUBB (IX+d),n	1 1 0 1 0 1 0 0 n n n n n n n n	d d d d d d d d	0 1 1 0	0 0 1 0	C Z C H S V	7	(IX+d) ← (IX+d)-n-CF
SUBB (IY+d),n	1 1 0 1 0 1 0 1 n n n n n n n n	d d d d d d d d	0 1 1 0	0 0 1 0	C Z C H S V	7	(IY+d) ← (IY+d)-n-CF
SUBB (SP+d),n	1 1 0 1 0 1 1 0 n n n n n n n n	d d d d d d d d	0 1 1 0	0 0 1 0	C Z C H S V	7	(SP+d) ← (SP+d)-n-CF
SUBB (HL+d),n	1 1 0 1 0 1 1 1 n n n n n n n n	d d d d d d d d	0 1 1 0	0 0 1 0	C Z C H S V	7	(HL+d) ← (HL+d)-n-CF
SUBB (HL+C),n	1 1 1 0 0 1 1 1 n n n n n n n n	0 1 1 0	0 0 1 0	n n n n	C Z C H S V	7	(HL+C) ← (HL+C)-n-CF
SUBB (+SP),n	1 1 1 0 0 1 1 0 n n n n n n n n	0 1 1 0	0 0 1 0	n n n n	C Z C H S V	6	SP ← SP+1:(SP) ← (SP)-n-CF
SUBB (PC+A),n 注	0 1 0 0 1 1 1 1 n n n n n n n n	0 1 1 0	0 0 1 0	n n n n	C Z C H S V	7	(PC+A) ← (PC+A)-n-CF
SUBB rr,(x)	1 1 1 0 0 0 0 0	x x x x x x x x	1 0 r r r 0 1 0		C Z C U S V	5	rr ← rr-(x+1, x)-CF
SUBB rr,(vw)	1 1 1 0 0 0 0 1 1 0 r r r 0 1 0	w w w w w w w w	v v v v	v v v v	C Z C U S V	6	rr ← rr-(vw+1,vw)-CF
SUBB rr,(DE)	1 1 1 0 0 0 1 0	1 0 r r r 0 1 0			C Z C U S V	4	rr ← rr-(DE+1, DE)-CF
SUBB rr,(HL)	1 1 1 0 0 0 1 1	1 0 r r r 0 1 0			C Z C U S V	4	rr ← rr-(HL+1, HL)-CF
SUBB rr,(IX)	1 1 1 0 0 1 0 0	1 0 r r r 0 1 0			C Z C U S V	4	rr ← rr-(IX+1, IX)-CF
SUBB rr,(IY)	1 1 1 0 0 1 0 1	1 0 r r r 0 1 0			C Z C U S V	4	rr ← rr-(IY+1, IY+d)-CF
SUBB rr,(IX+d)	1 1 0 1 0 1 0 0	d d d d d d d d	1 0 r r r 0 1 0		C Z C U S V	6	rr ← rr-(IX+d+1, IX+d)-CF
SUBB rr,(IY+d)	1 1 0 1 0 1 0 1	d d d d d d d d	1 0 r r r 0 1 0		C Z C U S V	6	rr ← rr-(IY+d+1, IY+d)-CF
SUBB rr,(SP+d)	1 1 0 1 0 1 1 0	d d d d d d d d	1 0 r r r 0 1 0		C Z C U S V	6	rr ← rr-(SP+d+1, SP+d)-CF
SUBB rr,(HL+d)	1 1 0 1 0 1 1 1	d d d d d d d d	1 0 r r r 0 1 0		C Z C U S V	6	rr ← rr-(HL+d+1, HL+d)-CF
SUBB rr,(HL+C)	1 1 1 0 0 1 1 1	1 0 r r r 0 1 0			C Z C U S V	6	rr ← rr-(HL+C+1, HL+C)-CF
SUBB rr,(+SP)	1 1 1 0 0 1 1 0	1 0 r r r 0 1 0			C Z C U S V	5	SP ← SP+1:rr ← rr-(SP+1, SP)-CF
SUBB rr,(PC+A) 注	0 1 0 0 1 1 1 1	1 0 r r r 0 1 0			C Z C U S V	6	rr ← rr-(PC+A+1, PC+A)-CF
AND A,n	0 1 1 0 0 1 0 0	n n n n n n n n			Z Z – – – –	2	A ← A&n
AND g,n	1 1 1 0 1 g g g	0 1 1 0 0 1 0 0	n n n n n n n n		Z Z – – – –	3	g ← g&n
AND gg,mn	1 1 1 0 1 g g g m m m m m m m m	0 1 1 0 1 1 0 0	n n n n n n n n		Z Z – – – –	4	gg ← gg&mn
AND r,g	1 1 1 0 1 g g g	0 0 r r r 1 0 0			Z Z – – – –	2	r ← r&g
AND rr,gg	1 1 1 0 1 g g g	1 0 r r r 1 0 0			Z Z – – – –	3	rr ← rr&gg
AND r,(x)	1 1 1 0 0 0 0 0	x x x x x x x x	0 0 r r r 1 0 0		Z Z – – – –	4	r ← r&(x)
AND r,(vw)	1 1 1 0 0 0 0 1 0 0 r r r 1 0 0	w w w w w w w w	v v v v	v v v v	Z Z – – – –	5	r ← r&(vw)
AND r,(DE)	1 1 1 0 0 0 1 0	0 0 r r r 1 0 0			Z Z – – – –	3	r ← r&(DE)
AND r,(HL)	1 1 1 0 0 0 1 1	0 0 r r r 1 0 0			Z Z – – – –	3	r ← r&(HL)
AND r,(IX)	1 1 1 0 0 1 0 0	0 0 r r r 1 0 0			Z Z – – – –	3	r ← r&(IX)
AND r,(IY)	1 1 1 0 0 1 0 1	0 0 r r r 1 0 0			Z Z – – – –	3	r ← r&(IY)
AND r,(IX+d)	1 1 0 1 0 1 0 0	d d d d d d d d	0 0 r r r 1 0 0		Z Z – – – –	5	r ← r&(IX+d)
AND r,(IY+d)	1 1 0 1 0 1 0 1	d d d d d d d d	0 0 r r r 1 0 0		Z Z – – – –	5	r ← r&(IY+d)
AND r,(SP+d)	1 1 0 1 0 1 1 0	d d d d d d d d	0 0 r r r 1 0 0		Z Z – – – –	5	r ← r&(SP+d)
AND r,(HL+d)	1 1 0 1 0 1 1 1	d d d d d d d d	0 0 r r r 1 0 0		Z Z – – – –	5	r ← r&(HL+d)
AND r,(HL+C)	1 1 1 0 0 1 1 1	0 0 r r r 1 0 0			Z Z – – – –	5	r ← r&(HL+C)
AND r,(+SP)	1 1 1 0 0 1 1 0	0 0 r r r 1 0 0			Z Z – – – –	4	SP ← SP+1:r ← r&(SP)
AND r,(PC+A) 注	0 1 0 0 1 1 1 1	0 0 r r r 1 0 0			Z Z – – – –	5	r ← r&(PC+A)

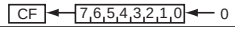
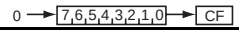
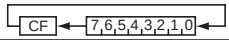
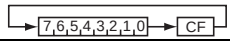
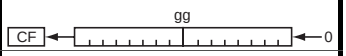
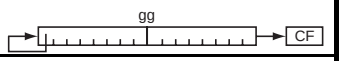
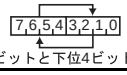
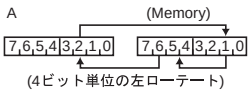
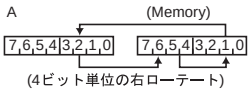
ニモニック	オブジェクトコード (2 進)				フラグ J Z C H S V	サイクル	オペレーション
AND (x),n	1 1 1 0 0 0 0 0	x x x x x x x x	0 1 1 0 0 1 0 0	n n n n n n n n	Z Z – – – –	6	(x) ← (x)&n
AND (vw),n	1 1 1 0 0 0 0 1	w w w w w w w w	v v v v v v v v	n n n n n n n n	Z Z – – – –	7	(vw) ← (vw)&n
AND (DE),n	1 1 1 0 0 0 1 0	0 1 1 0 0 1 0 0	n n n n n n n n		Z Z – – – –	5	(DE) ← (DE)&n
AND (HL),n	1 1 1 0 0 0 1 1	0 1 1 0 0 1 0 0	n n n n n n n n		Z Z – – – –	5	(HL) ← (HL)&n
AND (IX),n	1 1 1 0 0 1 0 0	0 1 1 0 0 1 0 0	n n n n n n n n		Z Z – – – –	5	(IX) ← (IX)&n
AND (IY),n	1 1 1 0 0 1 0 1	0 1 1 0 0 1 0 0	n n n n n n n n		Z Z – – – –	5	(IY) ← (IY)&n
AND (IX+d),n	1 1 0 1 0 1 0 0	d d d d d d d d	0 1 1 0 0 1 0 0	n n n n n n n n	Z Z – – – –	7	(IX+d) ← (IX+d)&n
AND (IY+d),n	1 1 0 1 0 1 0 1	d d d d d d d d	0 1 1 0 0 1 0 0	n n n n n n n n	Z Z – – – –	7	(IY+d) ← (IY+d)&n
AND (SP+d),n	1 1 0 1 0 1 1 0	d d d d d d d d	0 1 1 0 0 1 0 0	n n n n n n n n	Z Z – – – –	7	(SP+d) ← (SP+d)&n
AND (HL+d),n	1 1 0 1 0 1 1 1	d d d d d d d d	0 1 1 0 0 1 0 0	n n n n n n n n	Z Z – – – –	7	(HL+d) ← (HL+d)&n
AND (HL+C),n	1 1 1 0 0 1 1 1	0 1 1 0 0 1 0 0	n n n n n n n n		Z Z – – – –	7	(HL+C) ← (HL+C)&n
AND (+SP),n	1 1 1 0 0 1 1 0	0 1 1 0 0 1 0 0	n n n n n n n n		Z Z – – – –	6	SP ← SP+1:(SP) ← (SP)&n
AND (PC+A),n 注	0 1 0 0 1 1 1 1	0 1 1 0 0 1 0 0	n n n n n n n n		Z Z – – – –	7	(PC+A) ← (PC+A)&n
AND rr,(x)	1 1 1 0 0 0 0 0	x x x x x x x x	1 0 r r r 1 0 0		Z Z – – – –	5	rr ← rr&(x+1,x)
AND rr,(vw)	1 1 1 0 0 0 0 1	w w w w w w w w	v v v v v v v v	1 0 r r r 1 0 0	Z Z – – – –	6	rr ← rr&(vw+1,vw)
AND rr,(DE)	1 1 1 0 0 0 1 0	1 0 r r r 1 0 0			Z Z – – – –	4	rr ← rr&(DE+1,DE)
AND rr,(HL)	1 1 1 0 0 0 1 1	1 0 r r r 1 0 0			Z Z – – – –	4	rr ← rr&(HL+1,HL)
AND rr,(IX)	1 1 1 0 0 1 0 0	1 0 r r r 1 0 0			Z Z – – – –	4	rr ← rr&(IX+1,IX)
AND rr,(IY)	1 1 1 0 0 1 0 1	1 0 r r r 1 0 0			Z Z – – – –	4	rr ← rr&(IY+1,IY)
AND rr,(IX+d)	1 1 0 1 0 1 0 0	d d d d d d d d	1 0 r r r 1 0 0		Z Z – – – –	6	rr ← rr&(IX+d+1,IX+d)
AND rr,(IY+d)	1 1 0 1 0 1 0 1	d d d d d d d d	1 0 r r r 1 0 0		Z Z – – – –	6	rr ← rr&(IY+d+1,IY+d)
AND rr,(SP+d)	1 1 0 1 0 1 1 0	d d d d d d d d	1 0 r r r 1 0 0		Z Z – – – –	6	rr ← rr&(SP+d+1,SP+d)
AND rr,(HL+d)	1 1 0 1 0 1 1 1	d d d d d d d d	1 0 r r r 1 0 0		Z Z – – – –	6	rr ← rr&(HL+d+1,HL+d)
AND rr,(HL+C)	1 1 1 0 0 1 1 1	1 0 r r r 1 0 0			Z Z – – – –	6	rr ← rr&(HL+C+1,HL+C)
AND rr,(+SP)	1 1 1 0 0 1 1 0	1 0 r r r 1 0 0			Z Z – – – –	5	SP ← SP+1:rr ← rr&(SP+1,SP)
AND rr,(PC+A) 注	0 1 0 0 1 1 1 1	1 0 r r r 1 0 0			Z Z – – – –	6	rr ← rr&(PC+A+1,PC+A)
OR A,n	0 1 1 0 0 1 1 0	n n n n n n n n			Z Z – – – –	2	A ← A | n
OR g,n	1 1 1 0 1 g g g	0 1 1 0 0 1 1 0	n n n n n n n n		Z Z – – – –	3	g ← g | n
OR gg,mn	1 1 1 0 1 g g g	0 1 1 0 1 1 1 0	n n n n n n n n	m m m m m m m m	Z Z – – – –	4	gg ← gg | mn
OR r,g	1 1 1 0 1 g g g	0 0 r r r 1 1 0			Z Z – – – –	2	r ← r | g
OR rr,gg	1 1 1 0 1 g g g	1 0 r r r 1 1 0			Z Z – – – –	3	rr ← rr | gg
OR r,(x)	1 1 1 0 0 0 0 0	x x x x x x x x	0 0 r r r 1 1 0		Z Z – – – –	4	r ← r | (x)
OR r,(vw)	1 1 1 0 0 0 0 1	w w w w w w w w	v v v v v v v v	0 0 r r r 1 1 0	Z Z – – – –	5	r ← r | (vw)
OR r,(DE)	1 1 1 0 0 0 1 0	0 0 r r r 1 1 0			Z Z – – – –	3	r ← r | (DE)
OR r,(HL)	1 1 1 0 0 0 1 1	0 0 r r r 1 1 0			Z Z – – – –	3	r ← r | (HL)
OR r,(IX)	1 1 1 0 0 1 0 0	0 0 r r r 1 1 0			Z Z – – – –	3	r ← r | (IX)
OR r,(IY)	1 1 1 0 0 1 0 1	0 0 r r r 1 1 0			Z Z – – – –	3	r ← r | (IY)
OR r,(IX+d)	1 1 0 1 0 1 0 0	d d d d d d d d	0 0 r r r 1 1 0		Z Z – – – –	5	r ← r | (IX+d)
OR r,(IY+d)	1 1 0 1 0 1 0 1	d d d d d d d d	0 0 r r r 1 1 0		Z Z – – – –	5	r ← r | (IY+d)
OR r,(SP+d)	1 1 0 1 0 1 1 0	d d d d d d d d	0 0 r r r 1 1 0		Z Z – – – –	5	r ← r | (SP+d)
OR r,(HL+d)	1 1 0 1 0 1 1 1	d d d d d d d d	0 0 r r r 1 1 0		Z Z – – – –	5	r ← r | (HL+d)
OR r,(HL+C)	1 1 1 0 0 1 1 1	0 0 r r r 1 1 0			Z Z – – – –	5	r ← r | (HL+C)
OR r,(+SP)	1 1 1 0 0 1 1 0	0 0 r r r 1 1 0			Z Z – – – –	4	SP ← SP+1:r ← r | (SP)
OR r,(PC+A) 注	0 1 0 0 1 1 1 1	0 0 r r r 1 1 0			Z Z – – – –	5	r ← r | (PC+A)
OR (x),n	1 1 1 0 0 0 0 0	x x x x x x x x	0 1 1 0 0 1 1 0	n n n n n n n n	Z Z – – – –	6	(x) ← (x) | n
OR (vw),n	1 1 1 0 0 0 0 1	w w w w w w w w	v v v v v v v v	n n n n n n n n	Z Z – – – –	7	(vw) ← (vw) | n
OR (DE),n	1 1 1 0 0 0 1 0	0 1 1 0 0 1 1 0	n n n n n n n n		Z Z – – – –	5	(DE) ← (DE) | n
OR (HL),n	1 1 1 0 0 0 1 1	0 1 1 0 0 1 1 0	n n n n n n n n		Z Z – – – –	5	(HL) ← (HL) | n
OR (IX),n	1 1 1 0 0 1 0 0	0 1 1 0 0 1 1 0	n n n n n n n n		Z Z – – – –	5	(IX) ← (IX) | n
OR (IY),n	1 1 1 0 0 1 0 1	0 1 1 0 0 1 1 0	n n n n n n n n		Z Z – – – –	5	(IY) ← (IY) | n
OR (IX+d),n	1 1 0 1 0 1 0 0	d d d d d d d d	0 1 1 0 0 1 1 0	n n n n n n n n	Z Z – – – –	7	(IX+d) ← (IX+d) | n
OR (IY+d),n	1 1 0 1 0 1 0 1	d d d d d d d d	0 1 1 0 0 1 1 0	n n n n n n n n	Z Z – – – –	7	(IY+d) ← (IY+d) | n
OR (SP+d),n	1 1 0 1 0 1 1 0	d d d d d d d d	0 1 1 0 0 1 1 0	n n n n n n n n	Z Z – – – –	7	(SP+d) ← (SP+d) | n
OR (HL+d),n	1 1 0 1 0 1 1 1	d d d d d d d d	0 1 1 0 0 1 1 0	n n n n n n n n	Z Z – – – –	7	(HL+d) ← (HL+d) | n
OR (HL+C),n	1 1 1 0 0 1 1 1	0 1 1 0 0 1 1 0	n n n n n n n n		Z Z – – – –	7	(HL+C) ← (HL+C) | n
OR (+SP),n	1 1 1 0 0 1 1 0	0 1 1 0 0 1 1 0	n n n n n n n n		Z Z – – – –	6	SP ← SP+1:(SP) ← (SP) | n

ニモニック	オブジェクトコード (2 進)	フラグ J Z C H S V	サイクル	オペレーション
OR (PC+A),n 注	0 1 0 0 1 1 1 1 0 1 1 0 0 1 1 0 n n n n n n n n	Z Z – – – –	7	(PC+A) ← (PC+A)|n
OR rr,(x)	1 1 1 0 0 0 0 0 x x x x x x x x 1 0 r r r 1 1 0	Z Z – – – –	5	rr ← rr|(x+1,x)
OR rr,(vw)	1 1 1 0 0 0 0 1 w w w w w w w w v v v v v v v v 1 0 r r r 1 1 0	Z Z – – – –	6	rr ← rr|(vw+1,vw)
OR rr,(DE)	1 1 1 0 0 0 1 0 1 0 r r r 1 1 0	Z Z – – – –	4	rr ← rr|(DE+1,DE)
OR rr,(HL)	1 1 1 0 0 0 1 1 1 0 r r r 1 1 0	Z Z – – – –	4	rr ← rr|(HL+1,HL)
OR rr,(IX)	1 1 1 0 0 1 0 0 1 0 r r r 1 1 0	Z Z – – – –	4	rr ← rr|(IX+1,IX)
OR rr,(IY)	1 1 1 0 0 1 0 1 1 0 r r r 1 1 0	Z Z – – – –	4	rr ← rr|(IY+1,IY)
OR rr,(IX+d)	1 1 0 1 0 1 0 0 d d d d d d d d 1 0 r r r 1 1 0	Z Z – – – –	6	rr ← rr|(IX+d+1,IX+d)
OR rr,(IY+d)	1 1 0 1 0 1 0 1 d d d d d d d d 1 0 r r r 1 1 0	Z Z – – – –	6	rr ← rr|(IY+d+1,IY+d)
OR rr,(SP+d)	1 1 0 1 0 1 1 0 d d d d d d d d 1 0 r r r 1 1 0	Z Z – – – –	6	rr ← rr|(SP+d+1,SP+d)
OR rr,(HL+d)	1 1 0 1 0 1 1 1 d d d d d d d d 1 0 r r r 1 1 0	Z Z – – – –	6	rr ← rr|(HL+d+1,HL+d)
OR rr,(HL+C)	1 1 1 0 0 1 1 1 1 0 r r r 1 1 0	Z Z – – – –	6	rr ← rr|(HL+C+1,HL+C)
OR rr,(+SP)	1 1 1 0 0 1 1 0 1 0 r r r 1 1 0	Z Z – – – –	5	SP ← SP+1:rr ← rr|(SP+1,SP)
OR rr,(PC+A) 注	0 1 0 0 1 1 1 1 1 0 r r r 1 1 0	Z Z – – – –	6	rr ← rr|(PC+A+1,PC+A)
XOR A,n	0 1 1 0 0 1 0 1 n n n n n n n n	Z Z – – – –	2	A ← A ^ n
XOR g,n	1 1 1 0 1 g g g 0 1 1 0 0 1 0 1 n n n n n n n n	Z Z – – – –	3	g ← g ^ n
XOR gg,mn	1 1 1 0 1 g g g 0 1 1 0 1 1 0 1 n n n n n n n n m m m m m m m m	Z Z – – – –	4	gg ← gg ^ mn
XOR r,g	1 1 1 0 1 g g g 0 0 r r r 1 0 1	Z Z – – – –	2	r ← r ^ g
XOR rr,gg	1 1 1 0 1 g g g 1 0 r r r 1 0 1	Z Z – – – –	3	rr ← rr ^ gg
XOR r,(x)	1 1 1 0 0 0 0 0 x x x x x x x x 0 0 r r r 1 0 1	Z Z – – – –	4	r ← r ^ (x)
XOR r,(vw)	1 1 1 0 0 0 0 1 w w w w w w w w v v v v v v v v 0 0 r r r 1 0 1	Z Z – – – –	5	r ← r ^ (vw)
XOR r,(DE)	1 1 1 0 0 0 1 0 0 0 r r r 1 0 1	Z Z – – – –	3	r ← r ^ (DE)
XOR r,(HL)	1 1 1 0 0 0 1 1 0 0 r r r 1 0 1	Z Z – – – –	3	r ← r ^(HL)
XOR r,(IX)	1 1 1 0 0 1 0 0 0 0 r r r 1 0 1	Z Z – – – –	3	r ← r ^ (IX)
XOR r,(IY)	1 1 1 0 0 1 0 1 0 0 r r r 1 0 1	Z Z – – – –	3	r ← r ^ (IY)
XOR r,(IX+d)	1 1 0 1 0 1 0 0 d d d d d d d d 0 0 r r r 1 0 1	Z Z – – – –	5	r ← r ^ (IX+d)
XOR r,(IY+d)	1 1 0 1 0 1 0 1 d d d d d d d d 0 0 r r r 1 0 1	Z Z – – – –	5	r ← r ^ (IY+d)
XOR r,(SP+d)	1 1 0 1 0 1 1 0 d d d d d d d d 0 0 r r r 1 0 1	Z Z – – – –	5	r ← r ^ (SP+d)
XOR r,(HL+d)	1 1 0 1 0 1 1 1 d d d d d d d d 0 0 r r r 1 0 1	Z Z – – – –	5	r ← r ^ (HL+d)
XOR r,(HL+C)	1 1 1 0 0 1 1 1 0 0 r r r 1 0 1	Z Z – – – –	5	r ← r ^ (HL+C)
XOR r,(+SP)	1 1 1 0 0 1 1 0 0 0 r r r 1 0 1	Z Z – – – –	4	SP ← SP+1:r ← r ^ (SP)
XOR r,(PC+A) 注	0 1 0 0 1 1 1 1 0 0 r r r 1 0 1	Z Z – – – –	5	r ← r ^ (PC+A)
XOR (x),n	1 1 1 0 0 0 0 0 x x x x x x x x 0 1 1 0 0 1 0 1 n n n n n n n n	Z Z – – – –	6	(x) ← (x) ^ n
XOR (vw),n	1 1 1 0 0 0 0 1 w w w w w w w w v v v v v v v v 0 1 1 0 0 1 0 1 n n n n n n n n	Z Z – – – –	7	(vw) ← (vw) ^ n
XOR (DE),n	1 1 1 0 0 0 1 0 0 1 1 0 0 1 0 1 n n n n n n n n	Z Z – – – –	5	(DE) ← (DE) ^ n
XOR (HL),n	1 1 1 0 0 0 1 1 0 1 1 0 0 1 0 1 n n n n n n n n	Z Z – – – –	5	(HL) ← (HL) ^ n
XOR (IX),n	1 1 1 0 0 1 0 0 0 1 1 0 0 1 0 1 n n n n n n n n	Z Z – – – –	5	(IX) ← (IX) ^ n
XOR (IY),n	1 1 1 0 0 1 0 1 0 1 1 0 0 1 0 1 n n n n n n n n	Z Z – – – –	5	(IY) ← (IY) ^ n
XOR (IX+d),n	1 1 0 1 0 1 0 0 d d d d d d d d 0 1 1 0 0 1 0 1 n n n n n n n n	Z Z – – – –	7	(IX+d) ← (IX+d) ^ n
XOR (IY+d),n	1 1 0 1 0 1 0 1 d d d d d d d d 0 1 1 0 0 1 0 1 n n n n n n n n	Z Z – – – –	7	(IY+d) ← (IY+d) ^ n
XOR (SP+d),n	1 1 0 1 0 1 1 0 d d d d d d d d 0 1 1 0 0 1 0 1 n n n n n n n n	Z Z – – – –	7	(SP+d) ← (SP+d) ^n
XOR (HL+d),n	1 1 0 1 0 1 1 1 d d d d d d d d 0 1 1 0 0 1 0 1 n n n n n n n n	Z Z – – – –	7	(HL+d) ← (HL+d) ^ n
XOR (HL+C),n	1 1 1 0 0 1 1 1 0 1 1 0 0 1 0 1 n n n n n n n n	Z Z – – – –	7	(HL+C) ← (HL+C) ^n
XOR (+SP),n	1 1 1 0 0 1 1 0 0 1 1 0 0 1 0 1 n n n n n n n n	Z Z – – – –	6	SP ← SP+1:(SP) ← (SP) ^n
XOR (PC+A),n 注	0 1 0 0 1 1 1 1 0 1 1 0 0 1 0 1 n n n n n n n n	Z Z – – – –	7	(PC+A) ← (PC+A) ^ n
XOR rr,(x)	1 1 1 0 0 0 0 0 x x x x x x x x 1 0 r r r 1 0 1	Z Z – – – –	5	rr ← rr ^ (x+1,x)
XOR rr,(vw)	1 1 1 0 0 0 0 1 w w w w w w w w v v v v v v v v 1 0 r r r 1 0 1	Z Z – – – –	6	rr ← rr ^ (vw+1,vw)
XOR rr,(DE)	1 1 1 0 0 0 1 0 1 0 r r r 1 0 1	Z Z – – – –	4	rr ← rr ^ (DE+1,DE)
XOR rr,(HL)	1 1 1 0 0 0 1 1 1 0 r r r 1 0 1	Z Z – – – –	4	rr ← rr ^ (HL+1,HL)
XOR rr,(IX)	1 1 1 0 0 1 0 0 1 0 r r r 1 0 1	Z Z – – – –	4	rr ← rr ^(IX+1,IX)
XOR rr,(IY)	1 1 1 0 0 1 0 1 1 0 r r r 1 0 1	Z Z – – – –	4	rr ← rr ^ (IY+1,IY)
XOR rr,(IX+d)	1 1 0 1 0 1 0 0 d d d d d d d d 1 0 r r r 1 0 1	Z Z – – – –	6	rr ← rr ^ (IX+d+1,IX+d)
XOR rr,(IY+d)	1 1 0 1 0 1 0 1 d d d d d d d d 1 0 r r r 1 0 1	Z Z – – – –	6	rr ← rr ^ (IY+d+1,IY+d)
XOR rr,(SP+d)	1 1 0 1 0 1 1 0 d d d d d d d d 1 0 r r r 1 0 1	Z Z – – – –	6	rr ← rr ^ (SP+d+1,SP+d)
XOR rr,(HL+d)	1 1 0 1 0 1 1 1 d d d d d d d d 1 0 r r r 1 0 1	Z Z – – – –	6	rr ← rr ^ (HL+d+1,HL+d)
XOR rr,(HL+C)	1 1 1 0 0 1 1 1 1 0 r r r 1 0 1	Z Z – – – –	6	rr ← rr ^ (HL+C+1,HL+C)
XOR rr,(+SP)	1 1 1 0 0 1 1 0 1 0 r r r 1 0 1	Z Z – – – –	5	SP ← SP+1:rr ← rr ^ (SP+1,SP)
XOR rr,(PC+A) 注	0 1 0 0 1 1 1 1 1 0 r r r 1 0 1	Z Z – – – –	6	rr ← rr ^ (PC+A+1,PC+A)
INC r	0 0 1 0 0 r r r	C Z – – – –	1	r ← r+1
INC rr	0 0 1 1 0 r r r	C Z – – – –	2	rr ← rr+1
INC (x)	1 1 1 0 0 0 0 0 x x x x x x x x 1 1 1 1 0 0 0 0	C Z – – – –	5	(x) ← (x)+1

ニモニック	オブジェクトコード (2 進)	フラグ J Z C H S V	サイクル	オペレーション
INC (vw)	1 1 1 0 0 0 0 1 w w w w w w w w v v v v v v v v 1 1 1 1 0 0 0 0	C Z – – – –	6	(vw) ← (vw)+1
INC (DE)	1 1 1 0 0 0 1 0 1 1 1 1 0 0 0 0	C Z – – – –	4	(DE) ← (DE)+1
INC (HL)	1 1 1 0 0 0 1 1 1 1 1 1 0 0 0 0	C Z – – – –	4	(HL) ← (HL)+1
INC (IX)	1 1 1 0 0 1 0 0 1 1 1 1 0 0 0 0	C Z – – – –	4	(IX) ← (IX)+1
INC (IY)	1 1 1 0 0 1 0 1 1 1 1 1 0 0 0 0	C Z – – – –	4	(IY) ← (IY)+1
INC (IX+d)	1 1 0 1 0 1 0 0 d d d d d d d d 1 1 1 1 0 0 0 0	C Z – – – –	6	(IX+d) ← (IX+d)+1
INC (IY+d)	1 1 0 1 0 1 0 1 d d d d d d d d 1 1 1 1 0 0 0 0	C Z – – – –	6	(IY+d) ← (IY+d)+1
INC (SP+d)	1 1 0 1 0 1 1 0 d d d d d d d d 1 1 1 1 0 0 0 0	C Z – – – –	6	(SP+d) ← (SP+d)+1
INC (HL+d)	1 1 0 1 0 1 1 1 d d d d d d d d 1 1 1 1 0 0 0 0	C Z – – – –	6	(HL+d) ← (HL+d)+1
INC (HL+C)	1 1 1 0 0 1 1 1 1 1 1 1 0 0 0 0	C Z – – – –	6	(HL+C) ← (HL+C)+1
INC (+SP)	1 1 1 0 0 1 1 0 1 1 1 1 0 0 0 0	C Z – – – –	5	SP ← SP+1:(SP) ← (SP)+1
INC (PC+A) 注	0 1 0 0 1 1 1 1 1 1 1 1 0 0 0 0	C Z – – – –	6	(PC+A) ← (PC+A)+1
DEC r	0 0 1 0 1 r r r	C Z – – – –	1	r ← r–1
DEC rr	0 0 1 1 1 r r r	C Z – – – –	2	rr ← rr–1
DEC (x)	1 1 1 0 0 0 0 0 x x x x x x x x 1 1 1 1 1 0 0 0	C Z – – – –	5	(x) ← (x)–1
DEC (vw)	1 1 1 0 0 0 0 1 w w w w w w w w v v v v v v v v 1 1 1 1 1 0 0 0	C Z – – – –	6	(vw) ← (vw)–1
DEC (DE)	1 1 1 0 0 0 1 0 1 1 1 1 1 0 0 0	C Z – – – –	4	(DE) ← (DE)–1
DEC (HL)	1 1 1 0 0 0 1 1 1 1 1 1 1 0 0 0	C Z – – – –	4	(HL) ← (HL)–1
DEC (IX)	1 1 1 0 0 1 0 0 1 1 1 1 1 0 0 0	C Z – – – –	4	(IX) ← (IX)–1
DEC (IY)	1 1 1 0 0 1 0 1 1 1 1 1 1 0 0 0	C Z – – – –	4	(IY) ← (IY)–1
DEC (IX+d)	1 1 0 1 0 1 0 0 d d d d d d d d 1 1 1 1 1 0 0 0	C Z – – – –	6	(IX+d) ← (IX+d)–1
DEC (IY+d)	1 1 0 1 0 1 0 1 d d d d d d d d 1 1 1 1 1 0 0 0	C Z – – – –	6	(IY+d) ← (IY+d)–1
DEC (SP+d)	1 1 0 1 0 1 1 0 d d d d d d d d 1 1 1 1 1 0 0 0	C Z – – – –	6	(SP+d) ← (SP+d)–1
DEC (HL+d)	1 1 0 1 0 1 1 1 d d d d d d d d 1 1 1 1 1 0 0 0	C Z – – – –	6	(HL+d) ← (HL+d)–1
DEC (HL+C)	1 1 1 0 0 1 1 1 1 1 1 1 1 0 0 0	C Z – – – –	6	(HL+C) ← (HL+C)–1
DEC (+SP)	1 1 1 0 0 1 1 0 1 1 1 1 1 0 0 0	C Z – – – –	5	SP ← SP+1:(SP) ← (SP)–1
DEC (PC+A) 注	0 1 0 0 1 1 1 1 1 1 1 1 1 0 0 0	C Z – – – –	6	(PC+A) ← (PC+A)–1
DAA g	1 1 1 0 1 g g g 1 1 0 1 1 0 1 0	C Z C H – –	2	g を十進補正 (加算後)
DAS g	1 1 1 0 1 g g g 1 1 0 1 1 0 1 1	C Z C H – –	2	g を十進補正 (減算後)
MUL W,A	1 1 1 0 1 0 0 0 1 1 1 1 0 0 1 0	Z Z – – – –	13	WA ← W × A
MUL B,C	1 1 1 0 1 0 0 1 1 1 1 1 0 0 1 0	Z Z – – – –	13	BC ← B × C
MUL D,E	1 1 1 0 1 0 1 0 1 1 1 1 0 0 1 0	Z Z – – – –	13	DE ← D × E
MUL H,L	1 1 1 0 1 0 1 1 1 1 1 1 0 0 1 0	Z Z – – – –	13	HL ← H × L
DIV WA,C	1 1 1 0 1 0 0 0 1 1 1 1 0 0 1 1	Z Z C – – –	13	A ← WA ÷ C,W ← 余り
DIV DE,C	1 1 1 0 1 0 1 0 1 1 1 1 0 0 1 1	Z Z C – – –	13	E ← DE ÷ C,D ← 余り
DIV HL,C	1 1 1 0 1 0 1 1 1 1 1 1 0 0 1 1	Z Z C – – –	13	L ← HL ÷ C,H ← 余り
NEG CS,gg	1 1 1 0 1 g g g 1 1 1 1 1 0 1 0	1 – – – – –	3	if CF = 1 then gg ← 0-gg else null

注) (PC+A) を使用した命令には制限があります。詳細は、TLCS-870/C1 シリーズ 命令セット 1.4 アドレッシングモードを参照してください。

3.3 シフト、ローテート、ニブル処理

ニモニック	オブジェクトコード (2 進)				フラグ J Z C H S V	サイクル	オペレーション
SHLC g	1 1 1 0 1 g g g	1 1 1 1 0 1 0 0			C Z * − − −	2	
SHRC g	1 1 1 0 1 g g g	1 1 1 1 0 1 0 1			C Z * − − −	2	
ROLC g	1 1 1 0 1 g g g	1 1 1 1 0 1 1 0			C Z * − − −	2	
RORC g	1 1 1 0 1 g g g	1 1 1 1 0 1 1 1			C Z * − − −	2	
SHLCA gg	1 1 1 0 1 g g g	1 1 1 1 0 0 0 0			C Z * − S V	3	
SHRCA gg	1 1 1 0 1 g g g	1 1 1 1 0 0 0 1			C Z * − S 0	3	
SWAP g	1 1 1 0 1 g g g	1 1 1 1 1 1 1 1			1 − − − − −	7	
ROLD A,(x)	1 1 1 0 0 0 0 0	x x x x	x x x x	1 1 1 1 0 1 1 0	1 − − − − −	9	
ROLD A,(vw)	1 1 1 0 0 0 0 1	w w w w	w w w w	v v v v v v v v	1 − − − − −	10	
	1 1 1 1 0 1 1 0						
ROLD A,(DE)	1 1 1 0 0 0 1 0	1 1 1 1	0 1 1 0		1 − − − − −	8	
ROLD A,(HL)	1 1 1 0 0 0 1 1	1 1 1 1	0 1 1 0		1 − − − − −	8	
ROLD A,(IX)	1 1 1 0 0 1 0 0	1 1 1 1	0 1 1 0		1 − − − − −	8	
ROLD A,(IY)	1 1 1 0 0 1 0 1	1 1 1 1	0 1 1 0		1 − − − − −	8	
ROLD A,(IX+d)	1 1 0 1 0 1 0 0	d d d d	d d d d	1 1 1 1 0 1 1 0	1 − − − − −	10	
ROLD A,(IY+d)	1 1 0 1 0 1 0 1	d d d d	d d d d	1 1 1 1 0 1 1 0	1 − − − − −	10	
ROLD A,(SP+d)	1 1 0 1 0 1 1 0	d d d d	d d d d	1 1 1 1 0 1 1 0	1 − − − − −	10	
ROLD A,(HL+d)	1 1 0 1 0 1 1 1	d d d d	d d d d	1 1 1 1 0 1 1 0	1 − − − − −	10	
ROLD A,(HL+C)	1 1 1 0 0 1 1 1	1 1 1 1	0 1 1 0		1 − − − − −	10	
ROLD A,(+SP)	1 1 1 0 0 1 1 0	1 1 1 1	0 1 1 0		1 − − − − −	9	
ROLD A,(PC+A) 注	0 1 0 0 1 1 1 1	1 1 1 1	0 1 1 0		1 − − − − −	10	
RORD A,(x)	1 1 1 0 0 0 0 0	x x x x	x x x x	1 1 1 1 0 1 1 1	1 − − − − −	9	
RORD A,(vw)	1 1 1 0 0 0 0 1	w w w w	w w w w	v v v v v v v v	1 − − − − −	10	
	1 1 1 1 0 1 1 1						
RORD A,(DE)	1 1 1 0 0 0 1 0	1 1 1 1	0 1 1 1		1 − − − − −	8	
RORD A,(HL)	1 1 1 0 0 0 1 1	1 1 1 1	0 1 1 1		1 − − − − −	8	
RORD A,(IX)	1 1 1 0 0 1 0 0	1 1 1 1	0 1 1 1		1 − − − − −	8	
RORD A,(IY)	1 1 1 0 0 1 0 1	1 1 1 1	0 1 1 1		1 − − − − −	8	
RORD A,(IX+d)	1 1 0 1 0 1 0 0	d d d d	d d d d	1 1 1 1 0 1 1 1	1 − − − − −	10	
RORD A,(IY+d)	1 1 0 1 0 1 0 1	d d d d	d d d d	1 1 1 1 0 1 1 1	1 − − − − −	10	
RORD A,(SP+d)	1 1 0 1 0 1 1 0	d d d d	d d d d	1 1 1 1 0 1 1 1	1 − − − − −	10	
RORD A,(HL+d)	1 1 0 1 0 1 1 1	d d d d	d d d d	1 1 1 1 0 1 1 1	1 − − − − −	10	
RORD A,(HL+C)	1 1 1 0 0 1 1 1	1 1 1 1	0 1 1 1		1 − − − − −	10	
RORD A,(+SP)	1 1 1 0 0 1 1 0	1 1 1 1	0 1 1 1		1 − − − − −	9	
RORD A,(PC+A) 注	0 1 0 0 1 1 1 1	1 1 1 1	0 1 1 1		1 − − − − −	10	

注) (PC+A) を使用した命令には制限があります。詳細は、TLCS-870/C1 シリーズ 命令セット 1.4 アドレッシングモードを参照してください。

3.4 ビット操作、フラグ操作

ニモニック	オブジェクトコード (2 進)	フラグ J Z C H S V	サイクル	オペレーション
SET g.b	1 1 1 0 1 g g g 1 1 0 0 0 b b b	Z * – – – –	3	ZF ← $\overline{\text{g.b}}$:g.b ← 1
SET (x).b	1 1 0 0 0 b b b x x x x x x x x	Z * – – – –	4	ZF ← $\overline{\text{(x).b}}$:(x).b ← 1
SET (vw).b	1 1 1 0 0 0 0 1 w w w w w w w w v v v v v v v v	Z * – – – –	6	ZF ← $\overline{\text{(vw).b}}$:(vw).b ← 1
	1 1 0 0 0 b b b			
SET (DE).b	1 1 1 0 0 0 1 0 1 1 0 0 0 b b b	Z * – – – –	4	ZF ← $\overline{\text{(DE).b}}$:(DE).b ← 1
SET (HL).b	1 1 1 0 0 0 1 1 1 1 0 0 0 b b b	Z * – – – –	4	ZF ← $\overline{\text{(HL).b}}$:(HL).b ← 1
SET (IX).b	1 1 1 0 0 1 0 0 1 1 0 0 0 b b b	Z * – – – –	4	ZF ← $\overline{\text{(IX).b}}$:(IX).b ← 1
SET (IY).b	1 1 1 0 0 1 0 1 1 1 0 0 0 b b b	Z * – – – –	4	ZF ← $\overline{\text{(IY).b}}$:(IY).b ← 1
SET (IX+d).b	1 1 0 1 0 1 0 0 d d d d d d d d 1 1 0 0 0 b b b	Z * – – – –	6	ZF ← $\overline{\text{(IX+d).b}}$:(IX+d).b ← 1
SET (IY+d).b	1 1 0 1 0 1 0 1 d d d d d d d d 1 1 0 0 0 b b b	Z * – – – –	6	ZF ← $\overline{\text{(IY+d).b}}$:(IY+d).b ← 1
SET (SP+d).b	1 1 0 1 0 1 1 0 d d d d d d d d 1 1 0 0 0 b b b	Z * – – – –	6	ZF ← $\overline{\text{(SP+d).b}}$:(SP+d).b ← 1
SET (HL+d).b	1 1 0 1 0 1 1 1 d d d d d d d d 1 1 0 0 0 b b b	Z * – – – –	6	ZF ← $\overline{\text{(HL+d).b}}$:(HL+d).b ← 1
SET (HL+C).b	1 1 1 0 0 1 1 1 1 1 0 0 0 b b b	Z * – – – –	6	ZF ← $\overline{\text{(HL+C).b}}$:(HL+C).b ← 1
SET (+SP).b	1 1 1 0 0 1 1 0 1 1 0 0 0 b b b	Z * – – – –	5	SP ← SP+1:ZF ← $\overline{\text{(SP).b}}$:(SP).b ← 1
SET (PC+A).b 注	0 1 0 0 1 1 1 1 1 1 0 0 0 b b b	Z * – – – –	6	ZF ← $\overline{\text{(PC+A).b}}$:(PC+A).b ← 1
SET (x).A	1 1 1 0 0 0 0 0 x x x x x x x x 1 1 1 1 0 0 1 0	Z * – – – –	5	ZF ← $\overline{\text{(x).A}}$:(x).A ← 1
SET (vw).A	1 1 1 0 0 0 0 1 w w w w w w w w v v v v v v v v	Z * – – – –	6	ZF ← $\overline{\text{(vw).A}}$:(vw).A ← 1
	1 1 1 1 0 0 1 0			
SET (DE).A	1 1 1 0 0 0 1 0 1 1 1 1 0 0 1 0	Z * – – – –	4	ZF ← $\overline{\text{(DE).A}}$:(DE).A ← 1
SET (HL).A	1 1 1 0 0 0 1 1 1 1 1 1 0 0 1 0	Z * – – – –	4	ZF ← $\overline{\text{(HL).A}}$:(HL).A ← 1
SET (IX).A	1 1 1 0 0 1 0 0 1 1 1 1 0 0 1 0	Z * – – – –	4	ZF ← $\overline{\text{(IX).A}}$:(IX).A ← 1
SET (IY).A	1 1 1 0 0 1 0 1 1 1 1 1 0 0 1 0	Z * – – – –	4	ZF ← $\overline{\text{(IY).A}}$:(IY).A ← 1
SET (IX+d).A	1 1 0 1 0 1 0 0 d d d d d d d d 1 1 1 1 0 0 1 0	Z * – – – –	6	ZF ← $\overline{\text{(IX+d).A}}$:(IX+d).A ← 1
SET (IY+d).A	1 1 0 1 0 1 0 1 d d d d d d d d 1 1 1 1 0 0 1 0	Z * – – – –	6	ZF ← $\overline{\text{(IY+d).A}}$:(IY+d).A ← 1
SET (SP+d).A	1 1 0 1 0 1 1 0 d d d d d d d d 1 1 1 1 0 0 1 0	Z * – – – –	6	ZF ← $\overline{\text{(SP+d).A}}$:(SP+d).A ← 1
SET (HL+d).A	1 1 0 1 0 1 1 1 d d d d d d d d 1 1 1 1 0 0 1 0	Z * – – – –	6	ZF ← $\overline{\text{(HL+d).A}}$:(HL+d).A ← 1
SET (HL+C).A	1 1 1 0 0 1 1 1 1 1 1 1 0 0 1 0	Z * – – – –	6	ZF ← $\overline{\text{(HL+C).A}}$:(HL+C).A ← 1
SET (+SP).A	1 1 1 0 0 1 1 0 1 1 1 1 0 0 1 0	Z * – – – –	5	SP ← SP+1:ZF ← $\overline{\text{(SP).A}}$:(SP).A ← 1
SET (PC+A).A 注	0 1 0 0 1 1 1 1 1 1 1 1 0 0 1 0	Z * – – – –	6	ZF ← $\overline{\text{(PC+A).A}}$:(PC+A).A ← 1
CLR g.b	1 1 1 0 1 g g g 1 1 0 0 1 b b b	Z * – – – –	3	ZF ← $\overline{\text{g.b}}$:g.b ← 0
CLR (x).b	1 1 0 0 1 b b b x x x x x x x x	Z * – – – –	4	ZF ← $\overline{\text{(x).b}}$:(x).b ← 0
CLR (vw).b	1 1 1 0 0 0 0 1 w w w w w w w w v v v v v v v v	Z * – – – –	6	ZF ← $\overline{\text{(vw).b}}$:(vw).b ← 0
	1 1 0 0 1 b b b			
CLR (DE).b	1 1 1 0 0 0 1 0 1 1 0 0 1 b b b	Z * – – – –	4	ZF ← $\overline{\text{(DE).b}}$:(DE).b ← 0
CLR (HL).b	1 1 1 0 0 0 1 1 1 1 0 0 1 b b b	Z * – – – –	4	ZF ← $\overline{\text{(HL).b}}$:(HL).b ← 0
CLR (IX).b	1 1 1 0 0 1 0 0 1 1 0 0 1 b b b	Z * – – – –	4	ZF ← $\overline{\text{(IX).b}}$:(IX).b ← 0
CLR (IY).b	1 1 1 0 0 1 0 1 1 1 0 0 1 b b b	Z * – – – –	4	ZF ← $\overline{\text{(IY).b}}$:(IY).b ← 0
CLR (IX+d).b	1 1 0 1 0 1 0 0 d d d d d d d d 1 1 0 0 1 b b b	Z * – – – –	6	ZF ← $\overline{\text{(IX+d).b}}$:(IX+d).b ← 0
CLR (IY+d).b	1 1 0 1 0 1 0 1 d d d d d d d d 1 1 0 0 1 b b b	Z * – – – –	6	ZF ← $\overline{\text{(IY+d).b}}$:(IY+d).b ← 0
CLR (SP+d).b	1 1 0 1 0 1 1 0 d d d d d d d d 1 1 0 0 1 b b b	Z * – – – –	6	ZF ← $\overline{\text{(SP+d).b}}$:(SP+d).b ← 0
CLR (HL+d).b	1 1 0 1 0 1 1 1 d d d d d d d d 1 1 0 0 1 b b b	Z * – – – –	6	ZF ← $\overline{\text{(HL+d).b}}$:(HL+d).b ← 0
CLR (HL+C).b	1 1 1 0 0 1 1 1 1 1 0 0 1 b b b	Z * – – – –	6	ZF ← $\overline{\text{(HL+C).b}}$:(HL+C).b ← 0
CLR (+SP).b	1 1 1 0 0 1 1 0 1 1 0 0 1 b b b	Z * – – – –	5	SP ← SP+1:ZF ← $\overline{\text{(SP).b}}$:(SP).b ← 0
CLR (PC+A).b 注	0 1 0 0 1 1 1 1 1 1 0 0 1 b b b	Z * – – – –	6	ZF ← $\overline{\text{(PC+A).b}}$:(PC+A).b ← 0
CLR (x).A	1 1 1 0 0 0 0 0 x x x x x x x x 1 1 1 1 1 0 1 0	Z * – – – –	5	ZF ← $\overline{\text{(x).A}}$:(x).A ← 0
CLR (vw).A	1 1 1 0 0 0 0 1 w w w w w w w w v v v v v v v v	Z * – – – –	6	ZF ← $\overline{\text{(vw).A}}$:(vw).A ← 0
	1 1 1 1 1 0 1 0			
CLR (DE).A	1 1 1 0 0 0 1 0 1 1 1 1 1 0 1 0	Z * – – – –	4	ZF ← $\overline{\text{(DE).A}}$:(DE).A ← 0
CLR (HL).A	1 1 1 0 0 0 1 1 1 1 1 1 1 0 1 0	Z * – – – –	4	ZF ← $\overline{\text{(HL).A}}$:(HL).A ← 0
CLR (IX).A	1 1 1 0 0 1 0 0 1 1 1 1 1 0 1 0	Z * – – – –	4	ZF ← $\overline{\text{(IX).A}}$:(IX).A ← 0
CLR (IY).A	1 1 1 0 0 1 0 1 1 1 1 1 1 0 1 0	Z * – – – –	4	ZF ← $\overline{\text{(IY).A}}$:(IY).A ← 0
CLR (IX+d).A	1 1 0 1 0 1 0 0 d d d d d d d d 1 1 1 1 1 0 1 0	Z * – – – –	6	ZF ← $\overline{\text{(IX+d).A}}$:(IX+d).A ← 0
CLR (IY+d).A	1 1 0 1 0 1 0 1 d d d d d d d d 1 1 1 1 1 0 1 0	Z * – – – –	6	ZF ← $\overline{\text{(IY+d).A}}$:(IY+d).A ← 0
CLR (SP+d).A	1 1 0 1 0 1 1 0 d d d d d d d d 1 1 1 1 1 0 1 0	Z * – – – –	6	ZF ← $\overline{\text{(SP+d).A}}$:(SP+d).A ← 0
CLR (HL+d).A	1 1 0 1 0 1 1 1 d d d d d d d d 1 1 1 1 1 0 1 0	Z * – – – –	6	ZF ← $\overline{\text{(HL+d).A}}$:(HL+d).A ← 0
CLR (HL+C).A	1 1 1 0 0 1 1 1 1 1 1 1 0 0 1 0	Z * – – – –	6	ZF ← $\overline{\text{(HL+C).A}}$:(HL+C).A ← 0
CLR (+SP).A	1 1 1 0 0 1 1 0 1 1 1 1 0 0 1 0	Z * – – – –	5	SP ← SP+1:ZF ← $\overline{\text{(SP).A}}$:(SP).A ← 0
CLR (PC+A).A 注	0 1 0 0 1 1 1 1 1 1 1 1 0 0 1 0	Z * – – – –	6	ZF ← $\overline{\text{(PC+A).A}}$:(PC+A).A ← 0
LD CF,g.b	1 1 1 0 1 g g g 0 1 0 1 1 b b b	C – * – – –	2	CF ← g.b
LD CF,(x).b	0 1 0 1 1 b b b x x x x x x x x	C – * – – –	3	CF ← (x).b
LD CF,(vw).b	1 1 1 0 0 0 0 1 w w w w w w w w v v v v v v v v	C – * – – –	5	CF ← (vw).b
	0 1 0 1 1 b b b			
LD CF,(DE).b	1 1 1 0 0 0 1 0 0 1 0 1 1 b b b	C – * – – –	3	CF ← (DE).b
LD CF,(HL).b	1 1 1 0 0 0 1 1 0 1 0 1 1 b b b	C – * – – –	3	CF ← (HL).b
LD CF,(IX).b	1 1 1 0 0 1 0 0 0 1 0 1 1 b b b	C – * – – –	3	CF ← (IX).b
LD CF,(IY).b	1 1 1 0 0 1 0 1 0 1 0 1 1 b b b	C – * – – –	3	CF ← (IY).b
LD CF,(IX+d).b	1 1 0 1 0 1 0 0 d d d d d d d d 0 1 0 1 1 b b b	C – * – – –	5	CF ← (IX+d).b

ニモニック	オブジェクトコード (2 進)	フラグ J Z C H S V	サイクル	オペレーション
LD CF,(IY+d).b	1 1 0 1 0 1 0 1 d d d d d d d d 0 1 0 1 1 b b b	C – * – – –	5	CF ← (IY+d).b
LD CF,(SP+d).b	1 1 0 1 0 1 1 0 d d d d d d d d 0 1 0 1 1 b b b	C – * – – –	5	CF ← (SP+d).b
LD CF,(HL+d).b	1 1 0 1 0 1 1 1 d d d d d d d d 0 1 0 1 1 b b b	C – * – – –	5	CF ← (HL+d).b
LD CF,(HL+C).b	1 1 1 0 0 1 1 1 0 1 0 1 1 b b b	C – * – – –	5	CF ← (HL+C).b
LD CF,(+SP).b	1 1 1 0 0 1 1 0 0 1 0 1 1 b b b	C – * – – –	4	SP ← SP+1:CF ← (SP).b
LD CF,(PC+A).b 注	0 1 0 0 1 1 1 1 0 1 0 1 1 b b b	C – * – – –	5	CF ← (PC+A).b
LD CF,(x).A	1 1 1 0 0 0 0 0 x x x x x x x x 1 1 1 1 1 1 0 0	C – * – – –	4	CF ← (x).A
LD CF,(vw).A	1 1 1 0 0 0 0 1 w w w w w w w w v v v v v v v v	C – * – – –	5	CF ← (vw).A
	1 1 1 1 1 1 0 0			
LD CF,(DE).A	1 1 1 0 0 0 1 0 1 1 1 1 1 1 0 0	C – * – – –	3	CF ← (DE).A
LD CF,(HL).A	1 1 1 0 0 0 1 1 1 1 1 1 1 1 0 0	C – * – – –	3	CF ← (HL).A
LD CF,(IX).A	1 1 1 0 0 1 0 0 1 1 1 1 1 1 0 0	C – * – – –	3	CF ← (IX).A
LD CF,(IY).A	1 1 1 0 0 1 0 1 1 1 1 1 1 1 0 0	C – * – – –	3	CF ← (IY).A
LD CF,(IX+d).A	1 1 0 1 0 1 0 0 d d d d d d d d 1 1 1 1 1 1 0 0	C – * – – –	5	CF ← (IX+d).A
LD CF,(IY+d).A	1 1 0 1 0 1 0 1 d d d d d d d d 1 1 1 1 1 1 0 0	C – * – – –	5	CF ← (IY+d).A
LD CF,(SP+d).A	1 1 0 1 0 1 1 0 d d d d d d d d 1 1 1 1 1 1 0 0	C – * – – –	5	CF ← (SP+d).A
LD CF,(HL+d).A	1 1 0 1 0 1 1 1 d d d d d d d d 1 1 1 1 1 1 0 0	C – * – – –	5	CF ← (HL+d).A
LD CF,(HL+C).A	1 1 1 0 0 1 1 1 1 1 1 1 1 1 0 0	C – * – – –	5	CF ← (HL+C).A
LD CF,(+SP).A	1 1 1 0 0 1 1 0 1 1 1 1 1 1 0 0	C – * – – –	4	SP ← SP+1:CF ← (SP).A
LD CF,(PC+A).A 注	0 1 0 0 1 1 1 1 1 1 1 1 1 1 0 0	C – * – – –	5	CF ← (PC+A).A
TEST g.b#1	1 1 1 0 1 g g g 0 1 0 1 1 b b b	* – J – – –	2	JF ← $\overline{g.b}$
TEST (x).b#1	0 1 0 1 1 b b b x x x x x x x x	* – J – – –	3	JF ← $\overline{(x).b}$
TEST (vw).b#1	1 1 1 0 0 0 0 1 w w w w w w w w v v v v v v v v	* – J – – –	5	JF ← $\overline{(vw).b}$
	0 1 0 1 1 b b b			
TEST (DE).b#1	1 1 1 0 0 0 1 0 0 1 0 1 1 b b b	* – J – – –	3	JF ← $\overline{(DE).b}$
TEST (HL).b#1	1 1 1 0 0 0 1 1 0 1 0 1 1 b b b	* – J – – –	3	JF ← $\overline{(HL).b}$
TEST (IX).b#1	1 1 1 0 0 1 0 0 0 1 0 1 1 b b b	* – J – – –	3	JF ← $\overline{(IX).b}$
TEST (IY).b#1	1 1 1 0 0 1 0 1 0 1 0 1 1 b b b	* – J – – –	3	JF ← $\overline{(IY).b}$
TEST (IX+d).b#1	1 1 0 1 0 1 0 0 d d d d d d d d 0 1 0 1 1 b b b	* – J – – –	5	JF ← $\overline{(IX+d).b}$
TEST (IY+d).b#1	1 1 0 1 0 1 0 1 d d d d d d d d 0 1 0 1 1 b b b	* – J – – –	5	JF ← $\overline{(IY+d).b}$
TEST (SP+d).b#1	1 1 0 1 0 1 1 0 d d d d d d d d 0 1 0 1 1 b b b	* – J – – –	5	JF ← $\overline{(SP+d).b}$
TEST (HL+d).b#1	1 1 0 1 0 1 1 1 d d d d d d d d 0 1 0 1 1 b b b	* – J – – –	5	JF ← $\overline{(HL+d).b}$
TEST (HL+C).b#1	1 1 1 0 0 1 1 1 0 1 0 1 1 b b b	* – J – – –	5	JF ← $\overline{(HL+C).b}$
TEST (+SP).b#1	1 1 1 0 0 1 1 0 0 1 0 1 1 b b b	* – J – – –	4	SP ← SP+1:JF ← $\overline{(SP).b}$
TEST (PC+A).b#1 注	0 1 0 0 1 1 1 1 0 1 0 1 1 b b b	* – J – – –	5	JF ← $\overline{(PC+A).b}$
TEST (x).A#1	1 1 1 0 0 0 0 0 x x x x x x x x 1 1 1 1 1 1 0 0	* – J – – –	4	JF ← $\overline{(x).A}$
TEST (vw).A#1	1 1 1 0 0 0 0 1 w w w w w w w w v v v v v v v v	* – J – – –	5	JF ← $\overline{(vw).A}$
	1 1 1 1 1 1 0 0			
TEST (DE).A#1	1 1 1 0 0 0 1 0 1 1 1 1 1 1 0 0	* – J – – –	3	JF ← $\overline{(DE).A}$
TEST (HL).A#1	1 1 1 0 0 0 1 1 1 1 1 1 1 1 0 0	* – J – – –	3	JF ← $\overline{(HL).A}$
TEST (IX).A#1	1 1 1 0 0 1 0 0 1 1 1 1 1 1 0 0	* – J – – –	3	JF ← $\overline{(IX).A}$
TEST (IY).A#1	1 1 1 0 0 1 0 1 1 1 1 1 1 1 0 0	* – J – – –	3	JF ← $\overline{(IY).A}$
TEST (IX+d).A#1	1 1 0 1 0 1 0 0 d d d d d d d d 1 1 1 1 1 1 0 0	* – J – – –	5	JF ← $\overline{(IX+d).A}$
TEST (IY+d).A#1	1 1 0 1 0 1 0 1 d d d d d d d d 1 1 1 1 1 1 0 0	* – J – – –	5	JF ← $\overline{(IY+d).A}$
TEST (SP+d).A#1	1 1 0 1 0 1 1 0 d d d d d d d d 1 1 1 1 1 1 0 0	* – J – – –	5	JF ← $\overline{(SP+d).A}$
TEST (HL+d).A#1	1 1 0 1 0 1 1 1 d d d d d d d d 1 1 1 1 1 1 0 0	* – J – – –	5	JF ← $\overline{(HL+d).A}$
TEST (HL+C).A#1	1 1 1 0 0 1 1 1 1 1 1 1 1 1 0 0	* – J – – –	5	JF ← $\overline{(HL+C).A}$
TEST (+SP).A#1	1 1 1 0 0 1 1 0 1 1 1 1 1 1 0 0	* – J – – –	4	SP ← SP+1:JF ← $\overline{(SP).A}$
TEST (PC+A).A#1 注	0 1 0 0 1 1 1 1 1 1 1 1 1 1 0 0	* – J – – –	5	JF ← $\overline{(PC+A).A}$
LD g.b,CF	1 1 1 0 1 g g g 1 1 1 0 1 b b b	1 – – – – –	2	g.b ← CF
LD (x).b,CF	1 1 1 0 0 0 0 0 x x x x x x x x 1 1 1 0 1 b b b	1 – – – – –	5	(x).b ← CF
LD (vw).b,CF	1 1 1 0 0 0 0 1 w w w w w w w w v v v v v v v v	1 – – – – –	6	(vw).b ← CF
	1 1 1 0 1 b b b			
LD (DE).b,CF	1 1 1 0 0 0 1 0 1 1 1 0 1 b b b	1 – – – – –	4	(DE).b ← CF
LD (HL).b,CF	1 1 1 0 0 0 1 1 1 1 1 0 1 b b b	1 – – – – –	4	(HL).b ← CF
LD (IX).b,CF	1 1 1 0 0 1 0 0 1 1 1 0 1 b b b	1 – – – – –	4	(IX).b ← CF
LD (IY).b,CF	1 1 1 0 0 1 0 1 1 1 1 0 1 b b b	1 – – – – –	4	(IY).b ← CF
LD (IX+d).b,CF	1 1 0 1 0 1 0 0 d d d d d d d d 1 1 1 0 1 b b b	1 – – – – –	6	(IX+d).b ← CF
LD (IY+d).b,CF	1 1 0 1 0 1 0 1 d d d d d d d d 1 1 1 0 1 b b b	1 – – – – –	6	(IY+d).b ← CF
LD (SP+d).b,CF	1 1 0 1 0 1 1 0 d d d d d d d d 1 1 1 0 1 b b b	1 – – – – –	6	(SP+d).b ← CF
LD (HL+d).b,CF	1 1 0 1 0 1 1 1 d d d d d d d d 1 1 1 0 1 b b b	1 – – – – –	6	(HL+d).b ← CF
LD (HL+C).b,CF	1 1 1 0 0 1 1 1 1 1 1 0 1 b b b	1 – – – – –	6	(HL+C).b ← CF
LD (+SP).b,CF	1 1 1 0 0 1 1 0 1 1 1 0 1 b b b	1 – – – – –	5	SP ← SP+1:(SP).b ← CF
LD (PC+A).b,CF 注	0 1 0 0 1 1 1 1 1 1 1 0 1 b b b	1 – – – – –	6	(PC+A).b ← CF
LD (x).A,CF	1 1 1 0 0 0 0 0 x x x x x x x x 1 1 1 1 0 0 1 1	1 – – – – –	5	(x).A ← CF
LD (vw).A,CF	1 1 1 0 0 0 0 1 w w w w w w w w v v v v v v v v	1 – – – – –	6	(vw).A ← CF
	1 1 1 1 0 0 1 1			
LD (DE).A,CF	1 1 1 0 0 0 1 0 1 1 1 1 0 0 1 1	1 – – – – –	4	(DE).A ← CF

ニモニック	オブジェクトコード (2 進)	フラグ J Z C H S V	サイクル	オペレーション
LD (HL).A,CF	1 1 1 0 0 0 1 1 1 1 1 1 0 0 1 1	1 – – – – –	4	(HL).A ← CF
LD (IX).A,CF	1 1 1 0 0 1 0 0 1 1 1 1 0 0 1 1	1 – – – – –	4	(IX).A ← CF
LD (IY).A,CF	1 1 1 0 0 1 0 1 1 1 1 1 0 0 1 1	1 – – – – –	4	(IY).A ← CF
LD (IX+d).A,CF	1 1 0 1 0 1 0 0 d d d d d d d d 1 1 1 1 0 0 1 1	1 – – – – –	6	(IX+d).A ← CF
LD (IY+d).A,CF	1 1 0 1 0 1 0 1 d d d d d d d d 1 1 1 1 0 0 1 1	1 – – – – –	6	(IY+d).A ← CF
LD (SP+d).A,CF	1 1 0 1 0 1 1 0 d d d d d d d d 1 1 1 1 0 0 1 1	1 – – – – –	6	(SP+d).A ← CF
LD (HL+d).A,CF	1 1 0 1 0 1 1 1 d d d d d d d d 1 1 1 1 0 0 1 1	1 – – – – –	6	(HL+d).A ← CF
LD (HL+C).A,CF	1 1 1 0 0 1 1 1 1 1 1 1 0 0 1 1	1 – – – – –	6	(HL+C).A ← CF
LD (+SP).A,CF	1 1 1 0 0 1 1 0 1 1 1 1 0 0 1 1	1 – – – – –	5	SP ← SP+1:(SP).A ← CF
LD (PC+A).A,CF 注	0 1 0 0 1 1 1 1 1 1 1 1 0 0 1 1	1 – – – – –	6	(PC+A).A ← CF
CPL g.b	1 1 1 0 1 g g g 1 1 1 0 0 b b b	Z * – – – –	3	ZF ← $\overline{g.b}$:g.b ← $\overline{g.b}$
CPL (x).b	1 1 1 0 0 0 0 0 x x x x x x x x 1 1 1 0 0 b b b	Z * – – – –	5	ZF ← $\overline{(x).b}$:(x).b ← $\overline{(x).b}$
CPL (vw).b	1 1 1 0 0 0 0 1 w w w w w w w w v v v v v v v v 1 1 1 0 0 b b b	Z * – – – –	6	ZF ← $\overline{(vw).b}$:(vw).b ← $\overline{(vw).b}$
CPL (DE).b	1 1 1 0 0 0 1 0 1 1 1 0 0 b b b	Z * – – – –	4	ZF ← $\overline{(DE).b}$:(DE).b ← $\overline{(DE).b}$
CPL (HL).b	1 1 1 0 0 0 1 1 1 1 1 0 0 b b b	Z * – – – –	4	ZF ← $\overline{(HL).b}$:(HL).b ← $\overline{(HL).b}$
CPL (IX).b	1 1 1 0 0 1 0 0 1 1 1 0 0 b b b	Z * – – – –	4	ZF ← $\overline{(IX).b}$:(IX).b ← $\overline{(IX).b}$
CPL (IY).b	1 1 1 0 0 1 0 1 1 1 1 0 0 b b b	Z * – – – –	4	ZF ← $\overline{(IY).b}$:(IY).b ← $\overline{(IY).b}$
CPL (IX+d).b	1 1 0 1 0 1 0 0 d d d d d d d d 1 1 1 0 0 b b b	Z * – – – –	6	ZF ← $\overline{(IX+d).b}$:(IX+d).b ← $\overline{(IX+d).b}$
CPL (IY+d).b	1 1 0 1 0 1 0 1 d d d d d d d d 1 1 1 0 0 b b b	Z * – – – –	6	ZF ← $\overline{(IY+d).b}$:(IY+d).b ← $\overline{(IY+d).b}$
CPL (SP+d).b	1 1 0 1 0 1 1 0 d d d d d d d d 1 1 1 0 0 b b b	Z * – – – –	6	ZF ← $\overline{(SP+d).b}$:(SP+d).b ← $\overline{(SP+d).b}$
CPL (HL+d).b	1 1 0 1 0 1 1 1 d d d d d d d d 1 1 1 0 0 b b b	Z * – – – –	6	ZF ← $\overline{(HL+d).b}$:(HL+d).b ← $\overline{(HL+d).b}$
CPL (HL+C).b	1 1 1 0 0 1 1 1 1 1 1 0 0 b b b	Z * – – – –	6	ZF ← $\overline{(HL+C).b}$:(HL+C).b ← $\overline{(HL+C).b}$
CPL (+SP).b	1 1 1 0 0 1 1 0 1 1 1 0 0 b b b	Z * – – – –	5	SP ← SP+1:ZF ← $\overline{(SP).b}$: (SP).b ← $\overline{(SP).b}$
CPL (PC+A).b 注	0 1 0 0 1 1 1 1 1 1 1 0 0 b b b	Z * – – – –	6	ZF ← $\overline{(PC+A).b}$:(PC+A).b ← $\overline{(PC+A).b}$
CPL (x).A	1 1 1 0 0 0 0 0 x x x x x x x x 1 1 1 1 1 0 1 1	Z * – – – –	5	ZF ← $\overline{(x).A}$:(x).A ← $\overline{(x).A}$
CPL (vw).A	1 1 1 0 0 0 0 1 w w w w w w w w v v v v v v v v 1 1 1 1 1 0 1 1	Z * – – – –	6	ZF ← $\overline{(vw).A}$:(vw).A ← $\overline{(vw).A}$
CPL (DE).A	1 1 1 0 0 0 1 0 1 1 1 1 1 0 1 1	Z * – – – –	4	ZF ← $\overline{(DE).A}$:(DE).A ← $\overline{(DE).A}$
CPL (HL).A	1 1 1 0 0 0 1 1 1 1 1 1 1 0 1 1	Z * – – – –	4	ZF ← $\overline{(HL).A}$:(HL).A ← $\overline{(HL).A}$
CPL (IX).A	1 1 1 0 0 1 0 0 1 1 1 1 1 0 1 1	Z * – – – –	4	ZF ← $\overline{(IX).A}$:(IX).A ← $\overline{(IX).A}$
CPL (IY).A	1 1 1 0 0 1 0 1 1 1 1 1 1 0 1 1	Z * – – – –	4	ZF ← $\overline{(IY).A}$:(IY).A ← $\overline{(IY).A}$
CPL (IX+d).A	1 1 0 1 0 1 0 0 d d d d d d d d 1 1 1 1 1 0 1 1	Z * – – – –	6	ZF ← $\overline{(IX+d).A}$:(IX+d).A ← $\overline{(IX+d).A}$
CPL (IY+d).A	1 1 0 1 0 1 0 1 d d d d d d d d 1 1 1 1 1 0 1 1	Z * – – – –	6	ZF ← $\overline{(IY+d).A}$:(IY+d).A ← $\overline{(IY+d).A}$
CPL (SP+d).A	1 1 0 1 0 1 1 0 d d d d d d d d 1 1 1 1 1 0 1 1	Z * – – – –	6	ZF ← $\overline{(SP+d).A}$:(SP+d).A ← $\overline{(SP+d).A}$
CPL (HL+d).A	1 1 0 1 0 1 1 1 d d d d d d d d 1 1 1 1 1 0 1 1	Z * – – – –	6	ZF ← $\overline{(HL+d).A}$:(HL+d).A ← $\overline{(HL+d).A}$
CPL (HL+C).A	1 1 1 0 0 1 1 1 1 1 1 1 1 0 1 1	Z * – – – –	6	ZF ← $\overline{(HL+C).A}$:(HL+C).A ← $\overline{(HL+C).A}$
CPL (+SP).A	1 1 1 0 0 1 1 0 1 1 1 1 1 0 1 1	Z * – – – –	5	SP ← SP+1:ZF ← $\overline{(SP).A}$: (SP).A ← $\overline{(SP).A}$
CPL (PC+A).A 注	0 1 0 0 1 1 1 1 1 1 1 1 1 0 1 1	Z * – – – –	6	ZF ← $\overline{(PC+A).A}$:(PC+A).A ← $\overline{(PC+A).A}$
XOR CF,g.b	1 1 1 0 1 g g g 0 1 0 1 0 b b b	$\overline{C}$ – * – – –	2	CF ← CF ^ g.b
XOR CF,(x).b	1 1 1 0 0 0 0 0 x x x x x x x x 0 1 0 1 0 b b b	$\overline{C}$ – * – – –	4	CF ← CF ^ (x).b
XOR CF,(vw).b	1 1 1 0 0 0 0 1 w w w w w w w w v v v v v v v v 0 1 0 1 0 b b b	$\overline{C}$ – * – – –	5	CF ← CF ^ (vw).b
XOR CF,(DE).b	1 1 1 0 0 0 1 0 0 1 0 1 0 b b b	$\overline{C}$ – * – – –	3	CF ← CF ^ (DE).b
XOR CF,(HL).b	1 1 1 0 0 0 1 1 0 1 0 1 0 b b b	$\overline{C}$ – * – – –	3	CF ← CF ^ (HL).b
XOR CF,(IX).b	1 1 1 0 0 1 0 0 0 1 0 1 0 b b b	$\overline{C}$ – * – – –	3	CF ← CF ^ (IX).b
XOR CF,(IY).b	1 1 1 0 0 1 0 1 0 1 0 1 0 b b b	$\overline{C}$ – * – – –	3	CF ← CF ^ (IY).b
XOR CF,(IX+d).b	1 1 0 1 0 1 0 0 d d d d d d d d 0 1 0 1 0 b b b	$\overline{C}$ – * – – –	5	CF ← CF ^ (IX+d).b
XOR CF,(IY+d).b	1 1 0 1 0 1 0 1 d d d d d d d d 0 1 0 1 0 b b b	$\overline{C}$ – * – – –	5	CF ← CF ^ (IY+d).b
XOR CF,(SP+d).b	1 1 0 1 0 1 1 0 d d d d d d d d 0 1 0 1 0 b b b	$\overline{C}$ – * – – –	5	CF ← CF ^ (SP+d).b
XOR CF,(HL+d).b	1 1 0 1 0 1 1 1 d d d d d d d d 0 1 0 1 0 b b b	$\overline{C}$ – * – – –	5	CF ← CF ^ (HL+d).b
XOR CF,(HL+C).b	1 1 1 0 0 1 1 1 0 1 0 1 0 b b b	$\overline{C}$ – * – – –	5	CF ← CF ^ (HL+C).b
XOR CF,(+SP).b	1 1 1 0 0 1 1 0 0 1 0 1 0 b b b	$\overline{C}$ – * – – –	4	SP ← SP+1:CF ← CF ^ (SP).b
XOR CF,(PC+A).b 注	0 1 0 0 1 1 1 1 0 1 0 1 0 b b b	$\overline{C}$ – * – – –	5	CF ← CF ^ (PC+A).b
SET CF	0 0 0 0 0 1 0 1	0 – 1 – – –	1	CF ← 1
CLR CF	0 0 0 0 0 1 0 0	1 – 0 – – –	1	CF ← 0
CPL CF	0 0 0 0 0 1 1 0	* – * – – –	1	JF ← CF:CF ← $\overline{CF}$
DI #1	1 1 0 0 1 0 0 0 0 0 1 1 1 0 1 0	Z * – – – –	4	ZF ← $\overline{IMF}$:IMF ← 0
EI #1	1 1 0 0 0 0 0 0 0 0 1 1 1 0 1 0	Z * – – – –	4	ZF ← $\overline{IMF}$:IMF ← 1

#1　アセンブラ用拡張命令

注) (PC+A) を使用した命令には制限があります。詳細は、TLCS-870/C1 シリーズ 命令セット 1.4 アドレッシングモードを参照してください。

3.5 ジャンプ

ニモニック	オブジェクトコード (2 進)	フラグ J Z C H S V	サイクル	オペレーション
JRS T,$+2+d	1 0 0 d d d d d	1 – – – – –	4/2(t/f)	if JF = 1 then PC ← PC+d else null
JRS F,$+2+d	1 0 1 d d d d d	1 – – – – –	4/2(t/f)	if JF = 0 then PC ← PC+d else null
JR T,$+2+d	1 1 0 1 1 1 1 0 d d d d d d d d	1 – – – – –	4/2(t/f)	if JF = 1 then PC ← PC+d else null
JR F,$+2+d	1 1 0 1 1 1 1 1 d d d d d d d d	1 – – – – –	4/2(t/f)	if JF = 0 then PC ← PC+d else null
JR EQ,$+2+d	1 1 0 1 1 0 0 0 d d d d d d d d	1 – – – – –	4/2(t/f)	if ZF = 1 then PC ← PC+d else null
JR Z,$+2+d				
JR NE,$+2+d	1 1 0 1 1 0 0 1 d d d d d d d d	1 – – – – –	4/2(t/f)	if ZF = 0 then PC ← PC+d else null
JR NZ,$+2+d				
JR CS,$+2+d	1 1 0 1 1 0 1 0 d d d d d d d d	1 – – – – –	4/2(t/f)	if CF = 1 then PC ← PC+d else null
JR LT,$+2+d				
JR CC,$+2+d	1 1 0 1 1 0 1 1 d d d d d d d d	1 – – – – –	4/2(t/f)	if CF = 0 then PC ← PC+d else null
JR GE,$+2+d				
JR LE,$+2+d	1 1 0 1 1 1 0 0 d d d d d d d d	1 – – – – –	4/2(t/f)	if (CF|ZF) = 1 then PC ← PC+d else null
JR GT,$+2+d	1 1 0 1 1 1 0 1 d d d d d d d d	1 – – – – –	4/2(t/f)	if (CF|ZF) = 0 then PC ← PC+d else null
JR M,$+3+d	1 1 1 0 1 0 0 0 1 1 0 1 0 0 0 0 d d d d d d d d	1 – – – – –	5/3(t/f)	if SF = 1 then PC ← PC+d else null
JR P,$+3+d	1 1 1 0 1 0 0 0 1 1 0 1 0 0 0 1 d d d d d d d d	1 – – – – –	5/3(t/f)	if SF = 0 then PC ← PC+d else null
JR SLT,$+3+d	1 1 1 0 1 0 0 0 1 1 0 1 0 0 1 0 d d d d d d d d	1 – – – – –	5/3(t/f)	if (SF ^ VF) = 1 then PC ← PC+d else null
JR SGE,$+3+d	1 1 1 0 1 0 0 0 1 1 0 1 0 0 1 1 d d d d d d d d	1 – – – – –	5/3(t/f)	if (SF ^ VF) = 0 then PC ← PC+d else null
JR SLE,$+3+d	1 1 1 0 1 0 0 0 1 1 0 1 0 1 0 0 d d d d d d d d	1 – – – – –	5/3(t/f)	if ZF| (SF ^ VF) = 1 then PC ← PC+d else null
JR SGT,$+3+d	1 1 1 0 1 0 0 0 1 1 0 1 0 1 0 1 d d d d d d d d	1 – – – – –	5/3(t/f)	if ZF| (SF ^ VF) = 0 then PC ← PC+d else null
JR VS,$+3+d	1 1 1 0 1 0 0 0 1 1 0 1 0 1 1 0 d d d d d d d d	1 – – – – –	5/3(t/f)	if VF = 1 then PC ← PC+d else null
JR VC,$+3+d	1 1 1 0 1 0 0 0 1 1 0 1 0 1 1 1 d d d d d d d d	1 – – – – –	5/3(t/f)	if VF = 0 then PC ← PC+d else null
JR $+2+d	1 1 1 1 1 1 0 0 d d d d d d d d	1 – – – – –	4	PC ← PC+d
JP mn	1 1 1 1 1 1 1 0 n n n n n n n n m m m m m m m m	– – – – – –	4	PC ← mn
JP gg	1 1 1 0 1 g g g 1 1 1 1 1 1 1 0	– – – – – –	3	PC ← gg
JP (x)	1 1 1 0 0 0 0 0 x x x x x x x x 1 1 1 1 1 1 1 0	– – – – – –	6	PC ← (x+1,x)
JP (vw)	1 1 1 0 0 0 0 1 w w w w w w w w v v v v v v v v 1 1 1 1 1 1 1 0	– – – – – –	7	PC ← (vw+1,vw)
JP (DE)	1 1 1 0 0 0 1 0 1 1 1 1 1 1 1 0	– – – – – –	5	PC ← (DE+1,DE)
JP (HL)	1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 0	– – – – – –	5	PC ← (HL+1,HL)
JP (IX)	1 1 1 0 0 1 0 0 1 1 1 1 1 1 1 0	– – – – – –	5	PC ← (IX+1,IX)
JP (IY)	1 1 1 0 0 1 0 1 1 1 1 1 1 1 1 0	– – – – – –	5	PC ← (IY+1,IY)
JP (IX+d)	1 1 0 1 0 1 0 0 d d d d d d d d 1 1 1 1 1 1 1 0	– – – – – –	7	PC ← (IX+d+1,IX+d)
JP (IY+d)	1 1 0 1 0 1 0 1 d d d d d d d d 1 1 1 1 1 1 1 0	– – – – – –	7	PC ← (IY+d+1,IY+d)
JP (SP+d)	1 1 0 1 0 1 1 0 d d d d d d d d 1 1 1 1 1 1 1 0	– – – – – –	7	PC ← (SP+d+1,SP+d)
JP (HL+d)	1 1 0 1 0 1 1 1 d d d d d d d d 1 1 1 1 1 1 1 0	– – – – – –	7	PC ← (HL+d+1,HL+d)
JP (HL+C)	1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 0	– – – – – –	7	PC ← (HL+C+1,HL+C)
JP (+SP)	1 1 1 0 0 1 1 0 1 1 1 1 1 1 1 0	– – – – – –	6	SP ← SP+1:PC ← (SP+1,SP)
JP (PC+A) 注	0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 0	– – – – – –	7	PC ← (PC+A+1,PC+A)

注) (PC+A) を使用した命令には制限があります。詳細は、TLCS-870/C1 シリーズ 命令セット 1.4 アドレッシングモードを参照してください。

3.6 サブルーチンコール、リターン、ソフトウエア割り込み、ノーオペレーション

ニモニック	オブジェクトコード (2 進)				フラグ J Z C H S V	サイクル	オペレーション
CALLV n	0 1 1 1 n n n n				– – – – – –	7	(SP,SP-1) ← NxtOp: SP ← SP-2: PC ← (n × 2+ ベクタコール領域の先頭アドレス値 +1,n × 2+ ベクタコール領域の先頭アドレス値)
CALL mn	1 1 1 1 1 1 0 1	n n n n n n n n	m m m m m m m m		– – – – – –	6	(SP,SP-1) ← NxtOp:SP ← SP-2: PC ← mn
CALL gg	1 1 1 0 1 g g g	1 1 1 1 1 1 0 1			– – – – – –	6	(SP,SP-1) ← NxtOp:SP ← SP-2: PC ← gg
CALL (x)	1 1 1 0 0 0 0 0	x x x x x x x x	1 1 1 1 1 1 0 1		– – – – – –	8	(SP,SP-1) ← NxtOp:SP ← SP-2: PC ← (x+1,x)
CALL (vw)	1 1 1 0 0 0 0 1 1 1 1 1 1 1 0 1	w w w w w w w w	v v v v v v v v		– – – – – –	9	(SP,SP-1) ← NxtOp:SP ← SP-2: PC ← (vw+1,vw)
CALL (DE)	1 1 1 0 0 0 1 0	1 1 1 1 1 1 0 1			– – – – – –	7	(SP,SP-1) ← NxtOp:SP ← SP-2: PC ← (DE+1,DE)
CALL (HL)	1 1 1 0 0 0 1 1	1 1 1 1 1 1 0 1			– – – – – –	7	(SP,SP-1) ← NxtOp:SP ← SP-2: PC ← (HL+1,HL)
CALL (IX)	1 1 1 0 0 1 0 0	1 1 1 1 1 1 0 1			– – – – – –	7	(SP,SP-1) ← NxtOp:SP ← SP-2: PC ← (IX+1,IX)
CALL (IY)	1 1 1 0 0 1 0 1	1 1 1 1 1 1 0 1			– – – – – –	7	(SP,SP-1) ← NxtOp:SP ← SP-2: PC ← (IY+1,IY)
CALL (IX+d)	1 1 0 1 0 1 0 0	d d d d d d d d	1 1 1 1 1 1 0 1		– – – – – –	9	(SP,SP-1) ← NxtOp:SP ← SP-2: PC ← (IX+d+1,IX+d)
CALL (IY+d)	1 1 0 1 0 1 0 1	d d d d d d d d	1 1 1 1 1 1 0 1		– – – – – –	9	(SP,SP-1) ← NxtOp:SP ← SP-2: PC ← (IY+d+1,IY+d)
CALL (SP+d)	1 1 0 1 0 1 1 0	d d d d d d d d	1 1 1 1 1 1 0 1		– – – – – –	9	(SP,SP-1) ← NxtOp:SP ← SP-2: PC ← (SP+d+1,SP+d)
CALL (HL+d)	1 1 0 1 0 1 1 1	d d d d d d d d	1 1 1 1 1 1 0 1		– – – – – –	9	(SP,SP-1) ← NxtOp:SP ← SP-2: PC ← (HL+d+1,HL+d)
CALL (+SP)	1 1 1 0 0 1 1 0	1 1 1 1 1 1 0 1			– – – – – –	8	SP ← SP+1: (SP-1) ← $NxtOP_L$: PC_L ← (SP): (SP) ← $NxtOp_H$: PC_H ← (SP+1): SP ← SP-2
CALL (HL+C)	1 1 1 0 0 1 1 1	1 1 1 1 1 1 0 1			– – – – – –	9	(SP,SP-1) ← NxtOp:SP ← SP-2: PC ← (HL+C+1,HL+C)
CALL (PC+A) 注	0 1 0 0 1 1 1 1	1 1 1 1 1 1 0 1			– – – – – –	9	(SP,SP-1) ← NxtOp:SP ← SP-2: PC ← (PC+A+1,PC+A)
RET	1 1 1 1 1 0 1 0				– – – – – –	6	SP ← SP+2:PC ← (SP,SP-1)
RETI	1 1 1 1 1 0 1 1				* * * * * *	6	SP ← SP+2:PC ← (SP,SP-1): SP ← SP+1: PSW ← (SP): IMF ← (SP).0
RETN	1 1 1 0 1 0 0 0	1 1 1 1 1 0 1 1			* * * * * *	7	SP ← SP+2:PC ← (SP,SP-1): SP ← SP+1: PSW ← (SP): IMF ← (SP).0
SWI	1 1 1 1 1 1 1 1				– – – – – –	9	(SP) ← PSW:(SP).0 ← IMF: (SP-1,SP-2) ← NxtOp: PC ← (SWI ベクタアドレス +1,SWI ベクタアドレス)
NOP	0 0 0 0 0 0 0 0				– – – – – –	1	ノーオペレーション

注) (PC+A) を使用した命令には制限があります。詳細は、TLCS-870/C1 シリーズ 命令セット 1.4 アドレッシングモードを参照してください。

第 4 章　TLCS-870/C1 命令コードマップ

4.1　第 1 オペコード

下位 上位	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NOP				CLR CF	SET CF	CPL CF	CMP (x), n	LDW (mem), mn (x)	(HL)	LD (mem), n (x)	(HL)	LD A, (mem) (x)	(HL)	LD (mem), A (x)	(HL)
1	LD A,r								LD r,n							
	A	W	C	B	E	D	L	H	A	W	C	B	E	D	L	H
2	INC r								DEC r							
	A	W	C	B	E	D	L	H	A	W	C	B	E	D	L	H
3	INC rr							LD SP,SP+d	DEC rr							LD SP,SP-d
	WA	BC	DE	HL	IX	IY	SP		WA	BC	DE	HL	IX	IY	SP	
4	LD r,A								LD rr,mn							(src)
	A	W	C	B	E	D	L	H	WA	BC	DE	HL	IX	IY	SP	(PC+A)
5	PUSH rr (注 4)				(dst)				LD CF,(x).b							
	WA	BC	DE	HL	(IX+d)	(IY+d)	(SP+d)	(HL+d)	0	1	2	3	4	5	6	7
6	ADDC A, n	ADD A, n	SUBB A, n	SUB A, n	AND A, n	XOR A, n	OR A, n	CMP A, n								
7	CALLV n															
8	JRS T,a															
9																
A	JRS F,a															
B																
C	SET (x).b								CLR (x).b							
	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
D	POP rr (注 4)				(src)				JR cc, a							
	WA	BC	DE	HL	(IX+d)	(IY+d)	(SP+d)	(HL+d)	EQ/Z	NE/NZ	LT/CS	GE/CC	LE	GT	T	F
E	source memory prefix: (src)								register prefix: g/gg							
	(x)	(vw)	(DE)	(HL)	(IX)	(IY)	(+SP)	(HL+C)	A/WA	W/BC	C/DE	B/HL	E/IX	D/IY	L/SP	H/HL
F	destination memory prefix: (dst)									LD RBS,0 or 1	RET	RETI	JR a	CALL mn	JP mn	SWI
	(x)	(vw)	(DE)	(HL)	(IX)	(IY)	(SP-)	(HL+C)								

注 1)　網掛け位置のコードは未定義です。

注 2)　register prefix 命令 (0xE8 ~ 0xEF) は、次のページの 1.7 (2) が第 2 オペコードです。

注 3)　source/destination prefix 命令 (0xE0 ~ 0xE7 / 0xF0 ~ 0xF7 / 0x54~0x57 / 0xD4 ~ 0xD7 / 0x4F) は、1.7 (3) が第 2 オペコードです。

注 4)　PUSH rr/POP rr 命令に使用できるレジスタ (rr) は、WA, BC, DE, HL のみです。

4.2　第 2 オペコード (レジスタプリフィックス)

下位 上位	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	ADDC A, g	ADD A, g	SUBB A, g	SUB A, g	AND A, g	XOR A, g	OR A, g	CMP A, g	ADDC W, g	ADD W, g	SUBB W, g	SUB W, g	AND W, g	XOR W, g	OR W, g	CMP W, g
1	C, g	C, g	C, g	C, g	C, g	C, g	C, g	C, g	B, g	B, g	B, g	B, g	B, g	B, g	B, g	B, g
2	E, g	E, g	E, g	E, g	E, g	E, g	E, g	E, g	D, g	D, g	D, g	D, g	D, g	D, g	D, g	D, g
3	L, g	L, g	L, g	L, g	L, g	L, g	L, g	L, g	H, g	H, g	H, g	H, g	H, g	H, g	H, g	H, g
4	LD r, g A	W	C	B	E	D	L	H	LD rr, gg WA	BC	DE	HL	IX	IY	SP	
5	XOR CF, g.b 0	1	2	3	4	5	6	7	LD CF, g.b 0	1	2	3	4	5	6	7
6	ADDC g, n	ADD g, n	SUBB g, n	SUB g, n	AND g, n	XOR g, n	OR g, n	CMP g, n	ADDC gg, mn	ADD gg, mn	SUBB gg, mn	SUB gg, mn	AND gg, mn	XOR gg, mn	OR gg, mn	CMP gg, mn
7	XCH r, g A	W	C	B	E	D	L	H	XCH rr, gg WA	BC	DE	HL	IX	IY	SP	
8	ADDC WA, gg	ADD WA, gg	SUBB WA, gg	SUB WA, gg	AND WA, gg	XOR WA, gg	OR WA, gg	CMP WA, gg	ADDC BC, gg	ADD BC, gg	SUBB BC, gg	SUB BC, gg	AND BC, gg	XOR BC, gg	OR BC, gg	CMP BC, gg
9	DE, gg	DE, gg	DE, gg	DE, gg	DE, gg	DE, gg	DE, gg	DE, gg	HL, gg	HL, gg	HL, gg	HL, gg	HL, gg	HL, gg	HL, gg	HL, gg
A	IX, gg	IX, gg	IX, gg	IX, gg	IX, gg	IX, gg	IX, gg	IX, gg	IY, gg	IY, gg	IY, gg	IY, gg	IY, gg	IY, gg	IY, gg	IY, gg
B	SP, gg	SP, gg	SP, gg	SP, gg	SP, gg	SP, gg	SP, gg	SP, gg	HL, gg	HL, gg	HL, gg	HL, gg	HL, gg	HL, gg	HL, gg	HL, gg
C	SET g.b 0	1	2	3	4	5	6	7	CLR g.b 0	1	2	3	4	5	6	7
D	JR cc, a M	P	SLT	SGE	SLE	SGT	VS	VC	PUSH gg	POP gg	DAA g	DAS g	PUSH PSW	POP PSW	LD PSW, n	
E	CPL g.b 0	1	2	3	4	5	6	7	LD g.b, CF 0	1	2	3	4	5	6	7
F	SHLCA gg	SHRCA gg	MUL ggH, ggL	DIV gg, C	SHLC g	SHRC g	ROLC g	RORC g			NEG CS, gg	RETN		CALL gg	JP gg	SWAP g

注 1)　網掛け位置のコードは未定義です。

4.3 第 2 オペコード (メモリプリフィックス)

下位 上位	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	ADDC A, (src)	ADD A, (src)	SUBB A, (src)	SUB A, (src)	AND A, (src)	XOR A, (src)	OR A, (src)	CMP A, (src)	ADDC W, (src)	ADD W, (src)	SUBB W, (src)	SUB W, (src)	AND W, (src)	XOR W, (src)	OR W, (src)	CMP W, (src)
1	C, (src)	C, (src)	C, (src)	C, (src)	C, (src)	C, (src)	C, (src)	C, (src)	B, (src)	B, (src)	B, (src)	B, (src)	B, (src)	B, (src)	B, (src)	B, (src)
2	E, (src)	E, (src)	E, (src)	E, (src)	E, (src)	E, (src)	E, (src)	E, (src)	D, (src)	D, (src)	D, (src)	D, (src)	D, (src)	D, (src)	D, (src)	D, (src)
3	L, (src)	L, (src)	L, (src)	L, (src)	L, (src)	L, (src)	L, (src)	L, (src)	H, (src)	H, (src)	H, (src)	H, (src)	H, (src)	H, (src)	H, (src)	H, (src)
4	LD r, (src) A	W	C	B	E	D	L	H	LD rr, (src) WA	BC	DE	HL	IX	IY	SP	
5	XOR CF, (src).b 0	1	2	3	4	5	6	7	LD CF, (src).b 0	1	2	3	4	5	6	7
6	ADDC (src), n	ADD (src), n	SUBB (src), n	SUB (src), n	AND (src), n	XOR (src), n	OR (src), n	CMP (src), n	LD (dst), rr WA	BC	DE	HL	IX	IY	SP	
7	XCH r, (src) A	W	C	B	E	D	L	H	LD (dst), r A	W	C	B	E	D	L	H
8	ADDC WA, (src)	ADD WA, (src)	SUBB WA, (src)	SUB WA, (src)	AND WA, (src)	XOR WA, (src)	OR WA, (src)	CMP WA, (src)	ADDC BC, (src)	ADD BC, (src)	SUBB BC, (src)	SUB BC, (src)	AND BC, (src)	XOR BC, (src)	OR BC, (src)	CMP BC, (src)
9	DE, (src)	DE, (src)	DE, (src)	DE, (src)	DE, (src)	DE, (src)	DE, (src)	DE, (src)	HL, (src)	HL, (src)	HL, (src)	HL, (src)	HL, (src)	HL, (src)	HL, (src)	HL, (src)
A	IX, (src)	IX, (src)	IX, (src)	IX, (src)	IX, (src)	IX, (src)	IX, (src)	IX, (src)	IY, (src)	IY, (src)	IY, (src)	IY, (src)	IY, (src)	IY, (src)	IY, (src)	IY, (src)
B	SP, (src)	SP, (src)	SP, (src)	SP, (src)	SP, (src)	SP, (src)	SP, (src)	SP, (src)	HL, (src)	HL, (src)	HL, (src)	HL, (src)	HL, (src)	HL, (src)	HL, (src)	HL, (src)
C	SET (src).b 0	1	2	3	4	5	6	7	CLR (src).b 0	1	2	3	4	5	6	7
D									XCH rr, (src) WA	BC	DE	HL	IX	IY	SP	HL
E	CPL (src).b 0	1	2	3	4	5	6	7	LD (src).b, CF 0	1	2	3	4	5	6	7
F	INC (src)		SET (src). A	LD (src).A, CF			ROLD A,(src)	RORD A,(src)	DEC (src)	LD (dst), n	CLR (src).A	CPL (src).A	LD CF, (src).A	CALL (src)	JP (src)	

注 1) 網掛け位置のコードは未定義です。

注) レジスタプリフィックスの第 1 オペコードは、以下のとおりです。

g	第 1 オペコード
A	E8
W	E9
C	EA
B	EB
E	EC
D	ED
L	EE
H	EF

gg	第 1 オペコード
WA	E8
BC	E9
DE	EA
HL	EB
IX	EC
IY	ED
SP	EE
HL	EF

注) メモリプリフィックスの第 1 オペコードは、以下のとおりです。

(src)	第 1 オペコード
(x)	E0
(vw)	E1
(DE)	E2
(HL)	E3
(IX)	E4
(IY)	E5
(+SP)	E6
(HL+C)	E7
(IX+d)	D4
(IY+d)	D5
(SP+d)	D6
(HL+d)	D7
(PC+A)	4F

(dst)	第 1 オペコード
(x)	F0
(vw)	F1
(DE)	F2
(HL)	F3
(IX)	F4
(IY)	F5
(SP-)	F6
(HL+C)	F7
(IX+d)	54
(IY+d)	55
(SP+d)	56
(HL+d)	57

第 5 章　修正履歴

5.1　修正履歴

Rev	修正内容
RA001	16 進数の表記を H から 0x に、2 進数の表記を B から 0y に修正しました。
	LD RBS,n の記述を LD RBS,0、LD RBS,1 記述に変更しました。