LDX dst, src

< Load eXtract 抜き取り転送 >

動　作 : dst ← src

説　明 : srcの内容が、dstへ転送されます。この命令コードは、1バイトおきに有効コードが割り当てられており、16ビットデータバスモードで、8ビットデータバスメモリ上のコードをフェッチして、転送動作を行うための命令です。

詳　細 :

サイズ			ニモニック		コード
バイト	ワード	ロング			
○	×	×	LDX	(#8),#	

1	1	1	1	0	1	1	1
0	0	0	0	0	0	0	0
#8							
0	0	0	0	0	0	0	0
#							
0	0	0	0	0	0	0	0

補　足 : 第2, 第4, 第6命令コードの値が00Hでなくても、正しく命令は動作します。

フラグ :

S	Z	H	V	N	C
−	−	−	−	−	−

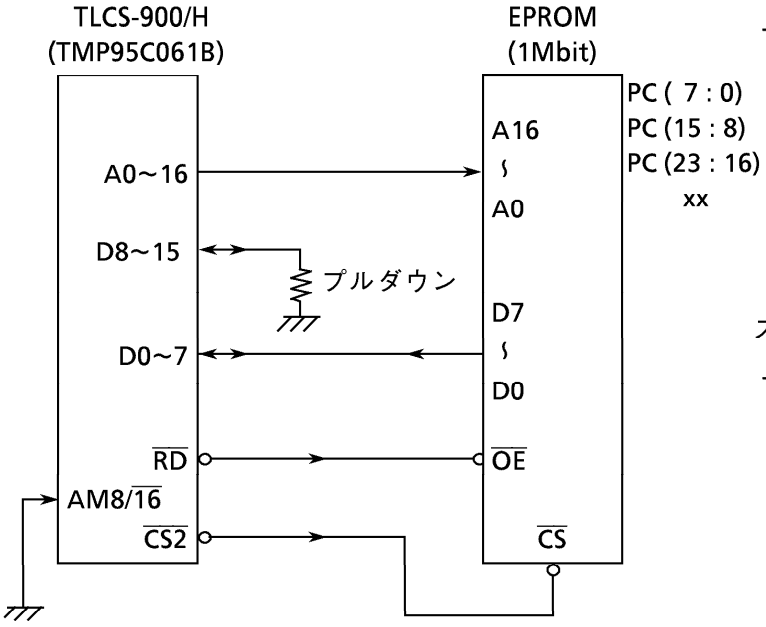
S = 変化なし。
Z = 変化なし。
H = 変化なし。
V = 変化なし。
N = 変化なし。
C = 変化なし。

実行例 ： TMP95C061Bを例に、8ビット幅のデータバスを持つEPROMでプログラムを実行させる場合を示します。

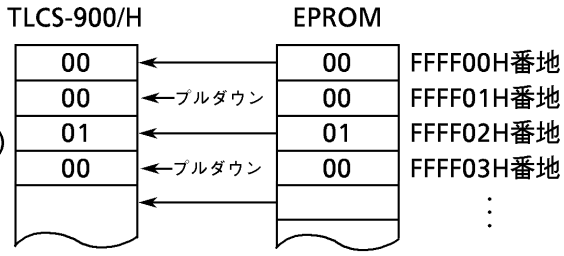
AM8/$\overline{16}$＝0のとき、TMP95C061Bはリセット後、リセットベクタFFFF00H番地から16ビット幅データバスのモードでスタートアドレスをフェッチし始めます。このため、8ビット幅のデータバスを持つ外付けメモリでプログラムをスタートする場合、上位側のデータバスD8~15端子をプルアップ/ダウンにより固定する必要があります。例えばプログラムのスタートアドレスを10000H番地に設定する場合、メモリアドレスのFFFF00H番地に10000Hを置き、データバスD8~15をプルダウンして固定します。そして、スタートアドレス番地の10000H番地に下記の命令を置きます。

```
LDX  (6AH), 17H
```

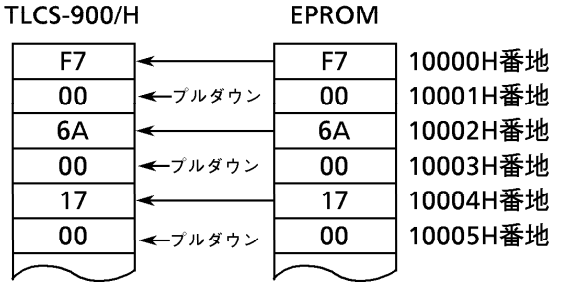
上記の命令を実行すると、内蔵のプログラマブルチップセレクト/ウェイトコントローラの6AH番地のコントロールレジスタに17Hのデータが書き込まれ、その結果、メモリアドレスの000080H~FFFFFFH番地領域は、“8ビット幅のデータバスで0WAITモード”になります。



リセット直後の動き



スタートアドレス番地からの動き



“プルダウン”は、データバスD8~15がプルダウンされていることにより00Hが入力されることを示しています。

LINK dst, num

< Link スタックフレームの生成 >

動　作 : (−XSP) ←dst, dst←XSP, XSP←XSP＋num

説　明 : dstの内容がスタック領域へ退避されます。次に、スタックポインタXSPの内容がdstへ転送されます。最後にスタックポインタXSPの内容とnumの内容(符号付き)が加算され、スタックポインタXSPへ転送されます。この命令はスタック領域にローカルな変数エリアを−numバイト分だけ確保するときに使用します。

詳　細 :

サイズ バイト	ワード	ロング	ニモニック		コード
×	×	○	LINK	r, d16	

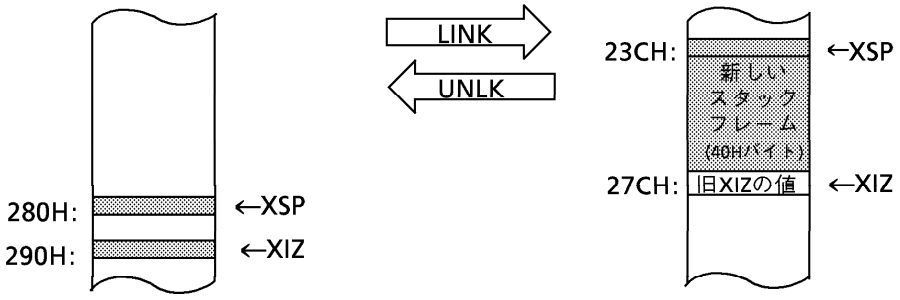
1	1	1	0	1	r		
0	0	0	0	1	1	0	0
d<7:0>							
d<15:8>							

フラグ :

S	Z	H	V	N	C
−	−	−	−	−	−

S ＝ 変化なし。
Z ＝ 変化なし。
H ＝ 変化なし。
V ＝ 変化なし。
N ＝ 変化なし。
C ＝ 変化なし。

実行例 : スタックポインタXSPが280Hで、XIZレジスタが290Hのとき、
LINK XIZ, −40H
を実行すると、メモリの27CH番地にデータ00000290H(ロングデータ)が書き込まれ、XIZレジスタは27CHに、スタックポインタXSPは23CHになります。



MDEC1 num, dst

< Modulo Decrement 1 モジュロ減少1 >

動 作 : if (dst mod num) =0 then dst ← dst+ (num−1) else dst ← dst−1.

説 明 : dstのモジュロnumが0の場合、dstにnum−1が加算されます。それ以外の場合、dstから1が減算されます。この命令は、循環構造のメモリテーブルポインタを操作するとき、使用します。

詳 細 :

サイズ バイト	ワード	ロング	ニモニック	コード
×	○	×	MDEC1 #, r	

1	1	0	1	1	r		
0	0	1	1	1	1	0	0
#<7:0>−1							
#<15:8>							

補 足 : オペランドの#は、2のn乗 (nは1~15) の値に限られます。

フラグ :

S	Z	H	V	N	C
−	−	−	−	−	−

S = 変化なし。
Z = 変化なし。
H = 変化なし。
V = 変化なし。
N = 変化なし。
C = 変化なし。

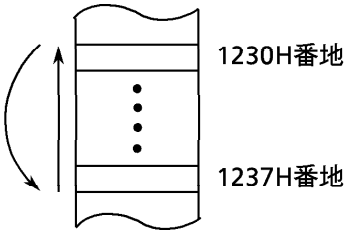
実行例 : IXレジスタを1230H~1237Hの範囲内で循環デクリメントする例です。
IXレジスタが1231Hのとき、

```
MDEC1  8, IX
```

を実行すると、IXレジスタは1230Hになり、再び

```
MDEC1  8, IX
```

を実行すると、IXレジスタのモジュロ8 (8で割った余り) が0なので、IXレジスタに8−1が加算され、IXレジスタは1237Hになります。



MDEC2 num, dst

< Modulo Decrement 2 モジュロ減少2 >

動 作 : if (dst mod num) =0 then dst ← dst+ (num−2) else dst ← dst−2.

説 明 : dstのモジュロnumが0の場合、dstにnum−2が加算されます。それ以外の場合、dstから2が減算されます。この命令は、循環構造のメモリテーブルポインタを操作するとき、使用します。

詳 細 :

サイズ バイト	ワード	ロング	ニモニック	コード
×	○	×	MDEC2 #,r	1 1 0 1 1 r / 0 0 1 1 1 1 0 1 / #<7:0>−2 / #<15:8>

補 足 : オペランドの#は、2のn乗 (nは2~15) の値に限られます。

フラグ :

S	Z	H	V	N	C
−	−	−	−	−	−

S = 変化なし。
Z = 変化なし。
H = 変化なし。
V = 変化なし。
N = 変化なし。
C = 変化なし。

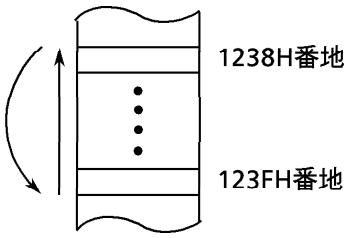
実行例 : IXレジスタを1238H~123FHの範囲内で循環デクリメントする例です。
IXレジスタが123AHのとき、

```
MDEC2  8,IX
```

を実行すると、IXレジスタは1238Hになり、再び

```
MDEC2  8,IX
```

を実行すると、IXレジスタのモジュロ8 (8で割った余り) が0なので、IXレジスタに8−2が加算され、IXレジスタは123EHになります。



MDEC4 num, dst

< Modulo Decrement 4 モジュロ減少4 >

動　作 : if (dst mod num)=0 then dst←dst+(num−4) else dst←dst−4.

説　明 : dstのモジュロnumが0の場合、dstにnum−4が加算されます。それ以外の場合、dstから4が減算されます。この命令は、循環構造のメモリテーブルポインタを操作するとき、使用します。

詳　細 :

サイズ			ニモニック	コード
バイト	ワード	ロング		
×	○	×	MDEC4 #,r	

1	1	0	1	1	r		
0	0	1	1	1	1	1	0
#<7:0>−4							
#<15:8>							

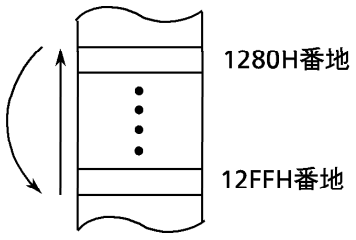
補　足 : オペランドの#は、2のn乗 (nは3~15) の値に限られます。

フラグ :

S	Z	H	V	N	C
−	−	−	−	−	−

S = 変化なし。
Z = 変化なし。
H = 変化なし。
V = 変化なし。
N = 変化なし。
C = 変化なし。

実行例 : IXレジスタを1280H~12FFHの範囲内で循環デクリメントする例です。
IXレジスタが1284Hのとき、
　MDEC4 80H,IX
を実行すると、IXレジスタは1280Hになり、再び
　MDEC4 80H,IX
を実行すると、IXレジスタのモジュロ80H (80Hで割った余り) が0なので、IXレジスタに80H−4が加算され、IXレジスタは12FCHになります。



MINC1 num, dst

< Modulo Increment 1 モジュロ増加1 >

動 作 : if (dst mod num) = (num−1) then dst ← dst− (num−1) else dst ← dst+1.

説 明 : dstのモジュロnumがnum−1の場合、dstからnum−1が減算されます。それ以外の場合、dstに1が加算されます。この命令は、循環構造のメモリテーブルポインタを操作するとき、使用します。

詳 細 :

サイズ			ニモニック	コード
バイト	ワード	ロング		
×	○	×	MINC1 #, r	

1	1	0	1	1	r		
0	0	1	1	1	0	0	0
#<7:0>−1							
#<15:8>							

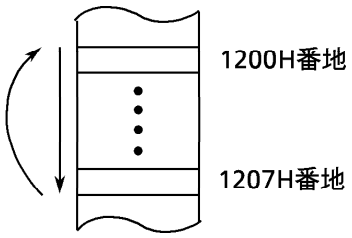
補 足 : オペランドの#は、2のn乗 (nは1~15) の値に限られます。

フラグ :

S	Z	H	V	N	C
−	−	−	−	−	−

S = 変化なし。
Z = 変化なし。
H = 変化なし。
V = 変化なし。
N = 変化なし。
C = 変化なし。

実行例 : IXレジスタを1200H~1207Hの範囲内で循環インクリメントする例です。
IXレジスタが1206Hのとき、
MINC1 8, IX
を実行すると、IXレジスタは1207Hになり、再び
MINC1 8, IX
を実行すると、IXレジスタのモジュロ8 (8で割った余り) が8−1なので、IXレジスタから8−1が減算され、IXレジスタは1200Hになります。



MINC2 num, dst

< Modulo Increment 2 モジュロ増加2 >

動 作 : if (dst mod num) = (num−2) then dst ← dst− (num−2) else dst ← dst+2.

説 明 : dstのモジュロnumがnum−2の場合、dstからnum−2が減算されます。それ以外の場合、dstに2が加算されます。この命令は、循環構造のメモリテーブルポインタを操作するとき、使用します。

詳 細 :

サイズ バイト	ワード	ロング	ニモニック	コード
×	○	×	MINC2 #, r	

1	1	0	1	1	r		
0	0	1	1	1	0	0	1
#<7:0>−2							
#<15:8>							

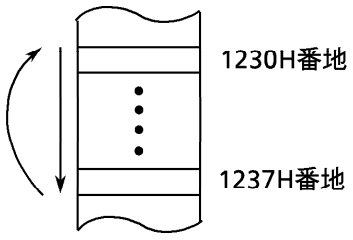
補 足 : オペランドの#は、2のn乗 (nは2~15) の値に限られます。

フラグ :

S	Z	H	V	N	C
−	−	−	−	−	−

S = 変化なし。
Z = 変化なし。
H = 変化なし。
V = 変化なし。
N = 変化なし。
C = 変化なし。

実行例 : IXレジスタを1230H~1237Hの範囲内で循環インクリメントする例です。
IXレジスタが1234Hのとき、
MINC2 8, IX
を実行すると、IXレジスタは1236Hになり、再び
MINC2 8, IX
を実行すると、IXレジスタのモジュロ8 (8で割った余り) が8−2なので、IXレジスタから8−2が減算され、IXレジスタは1230Hになります。



MINC4 num, dst

< Modulo Increment 4 モジュロ増加4 >

動 作 : if (dst mod num) = (num−4) then dst ← dst− (num−4) else dst ← dst+4.

説 明 : dstのモジュロnumがnum−4の場合、dstからnum−4が減算されます。それ以外の場合、dstに4が加算されます。この命令は、循環構造のメモリテーブルポインタを操作するとき、使用します。

詳 細 :

サイズ バイト	サイズ ワード	サイズ ロング	ニモニック	コード
×	○	×	MINC4 #, r	

1	1	0	1	1	r		
0	0	1	1	1	0	1	0
#<7:0>−4							
#<15:8>							

補 足 : オペランドの#は、2のn乗 (nは3~15) の値に限られます。

フラグ :

S	Z	H	V	N	C
−	−	−	−	−	−

S = 変化なし。
Z = 変化なし。
H = 変化なし。
V = 変化なし。
N = 変化なし。
C = 変化なし。

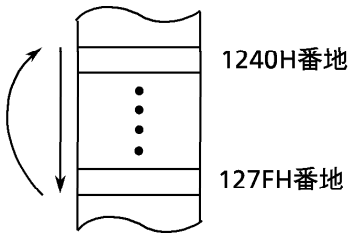
実行例 : IXレジスタを1240H~127FHの範囲内で循環インクリメントする例です。
IXレジスタが1278Hのとき、

```
MINC4  40H, IX
```

を実行すると、IXレジスタは127CHになり、再び

```
MINC4  40H, IX
```

を実行すると、IXレジスタのモジュロ40H (40Hで割った余り) が40H−4なので、IXレジスタから40H−4が減算され、IXレジスタは1240Hになります。



MIRR dst

＜ Mirror ミラー交換 ＞

動 作 : dst＜MSB:LSB＞←dst＜LSB : MSB＞

説 明 : dstの内容が、ビットパターンのイメージでミラー交換されます。

詳 細 :

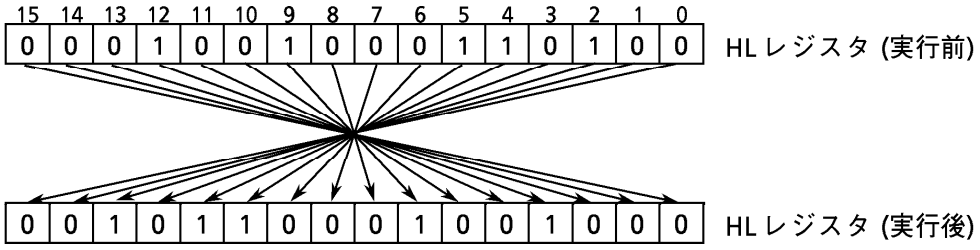
サイズ バイト	ワード	ロング	ニモニック		コード
×	○	×	MIRR	r	1 1 0 1 1 r / 0 0 0 1 0 1 1 0

フラグ :

S	Z	H	V	N	C
−	−	−	−	−	−

S ＝ 変化なし。
Z ＝ 変化なし。
H ＝ 変化なし。
V ＝ 変化なし。
N ＝ 変化なし。
C ＝ 変化なし。

実行例 : HLレジスタが0001 0010 0011 0100B (2進数) のとき、
MIRR HL
を実行すると、 HLレジスタは0010 1100 0100 1000B (2進数) になります。



MUL dst, src

< Multiply 符号なし乗算 >

動 作 : dst ← dst<下位半分>×src (符号なし)

説 明 : dstの下位半分の内容とsrcの内容が符号なし乗算され、dstへ転送されます。

詳 細 :

サイズ バイト	ワード	ロング	ニモニック		コード
○	○	×	MUL	RR, r	1 1 0 z 1 r 0 1 0 0 0 R
○	○	×	MUL	rr, #	1 1 0 z 1 r 0 0 0 0 1 0 0 0 # <7:0> # <15:8>
○	○	×	MUL	RR, (mem)	1 m 0 z m m m m 0 1 0 0 0 R

補 足 : 演算サイズがバイトのときは、"dst (ワード) ← dst (バイト) × src (バイト)"、
演算サイズがワードのときは、"dst (ロング) ← dst (ワード) × src (ワード)" になります。オペランドのdstの記述は、演算結果のサイズに合わせてください。

フラグ :

S	Z	H	V	N	C
−	−	−	−	−	−

S = 変化なし。
Z = 変化なし。
H = 変化なし。
V = 変化なし。
N = 変化なし。
C = 変化なし。

実行例 : IXレジスタが1234Hで、IYレジスタが89ABHのとき、

```
MUL XIX, IY
```

を実行すると、IXレジスタの内容とIYレジスタの内容が符号なし乗算され、XIXレジスタは09C9FCBCHになります。

注意： “MUL RR, r” と “MUL RR, (mem)” 命令のRRは、下記に示す表のようになります。

演算サイズがバイト
(16ビット←8ビット×8ビット)

RR	コード R
WA	001
BC	011
DE	101
HL	111
IX	指定不可!
IY	
IZ	
SP	

演算サイズがワード
(32ビット←16ビット×16ビット)

RR	コード R
XWA	000
XBC	001
XDE	010
XHL	011
XIX	100
XIY	101
XIZ	110
XSP	111

“MUL rr, #” 命令のrrは、下記に示す表のようになります。

演算サイズがバイト
(16ビット←8ビット×8ビット)

rr	コード r
WA	001
BC	011
DE	101
HL	111
IX	C7H : F0H
IY	C7H : F4H
IZ	C7H : F8H
SP	C7H : FCH
	1バイト目 2バイト目

(注) その他のワードレジスタも
IX~SPと同様の拡張コード方式で、
すべて指定可能。

演算サイズがワード
(32ビット←16ビット×16ビット)

rr	コード r
XWA	000
XBC	001
XDE	010
XHL	011
XIX	100
XIY	101
XIZ	110
XSP	111

(注) その他のロングワードレジスタも
拡張コード方式ですべて指定可能。

MULA dst

< Multiply and Add 符号付き積和演算 >

動 作 : dst ← dst + (XDE) × (XHL) , XHL ← XHL − 2

説 明 : XDEレジスタで示されたメモリデータ (16ビット) とXHLレジスタで示されたメモリデータ (16ビット) の内容が符号付き乗算され、その結果 (32ビット) とdstの内容 (32ビット) が加算され、dst (32ビット) へ転送されます。その後、XHLレジスタの内容が−2されます。

詳 細 :

サイズ バイト	ワード	ロング	ニモニック		コード
×	○	×	MULA	rr	1 1 0 1 1 r / 0 0 0 1 1 0 0 1

補 足 : オペランドのdstの記述は、演算結果のサイズ (ロング) に合わせてください。

フラグ :

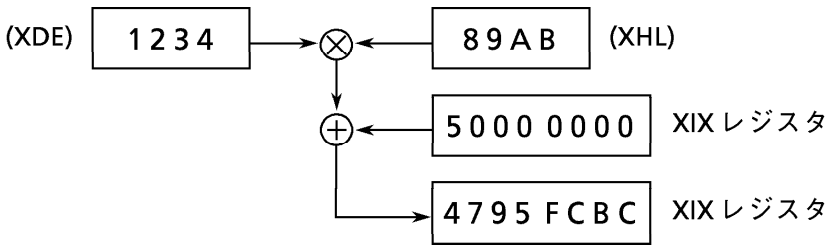
S	Z	H	V	N	C
*	*	−	*	−	−

S = 演算結果の最上位ビットの値がセットされます。
Z = 演算結果がゼロのときは“1”、それ以外のときは“0”がセットされます。
H = 変化なし。
V = 演算結果、オーバフローが発生したときは“1”、それ以外のときは“0”がセットされます。
N = 変化なし。
C = 変化なし。

実行例 : XIXレジスタが50000000Hで、XDEレジスタが100H, XHLが200H, 100H番地のメモリの内容(ワード)が1234H, 200H番地のメモリの内容 (ワード) が89ABHのとき、

MULA XIX

を実行すると、XIXレジスタは4795FCBCH、XHLレジスタは1FEHになります。



MULS dst, src

< Multiply Signed 符号付き乗算 >

動 作 : dst ← dst<下位半分>×src (符号付き)

説 明 : dstの下位半分の内容とsrcの内容が符号付き乗算され、dstへ転送されます。

詳 細 :

サイズ バイト	サイズ ワード	サイズ ロング	ニモニック		コード
○	○	×	MULS	RR, r	1 1 0 z 1 r 0 1 0 0 1 R
○	○	×	MULS	rr, #	1 1 0 z 1 r 0 0 0 0 1 0 0 1 # <7:0> # <15:8>
○	○	×	MULS	RR, (mem)	1 m 0 z m m m m 0 1 0 0 1 R

補 足 : 演算サイズがバイトのときは、"dst (ワード) ← dst (バイト) × src (バイト)"、
演算サイズがワードのときは、"dst (ロング) ← dst (ワード) × src (ワード)" になります。オペランドのdstの記述は、演算結果のサイズに合わせてください。

フラグ :

S	Z	H	V	N	C
−	−	−	−	−	−

S = 変化なし。
Z = 変化なし。
H = 変化なし。
V = 変化なし。
N = 変化なし。
C = 変化なし。

実行例 : IXレジスタが1234Hで、IYレジスタが89ABHのとき、
MULS XIX, IY
を実行すると、IXレジスタの内容とIYレジスタの内容が符号付き乗算され、XIXレジスタはF795FCBCHになります。

注意： “MULS RR, r” と “MULS RR, (mem)” 命令のRRは、下記に示す表のようになります。

演算サイズがバイト
(16ビット←8ビット×8ビット)

RR	コード R
WA	001
BC	011
DE	101
HL	111
IX	指定不可!
IY	
IZ	
SP	

演算サイズがワード
(32ビット←16ビット×16ビット)

RR	コード R
XWA	000
XBC	001
XDE	010
XHL	011
XIX	100
XIY	101
XIZ	110
XSP	111

“MULS rr, #” 命令のrrは、下記に示す表のようになります。

演算サイズがバイト
(16ビット←8ビット×8ビット)

rr	コード r
WA	001
BC	011
DE	101
HL	111
IX	C7H : F0H
IY	C7H : F4H
IZ	C7H : F8H
SP	C7H : FCH
	1バイト目 2バイト目

(注) その他のワードレジスタも
IX~SPと同様の拡張コード方式で、
すべて指定可能。

演算サイズがワード
(32ビット←16ビット×16ビット)

rr	コード r
XWA	000
XBC	001
XDE	010
XHL	011
XIX	100
XIY	101
XIZ	110
XSP	111

(注) その他のロングワードレジスタも
拡張コード方式ですべて指定可能。

NEG dst

< Negate 2の補数 >

動　作 : dst←0−dst

説　明 : ゼロからdstの内容が減算され、dstへ転送されます。

詳　細 :

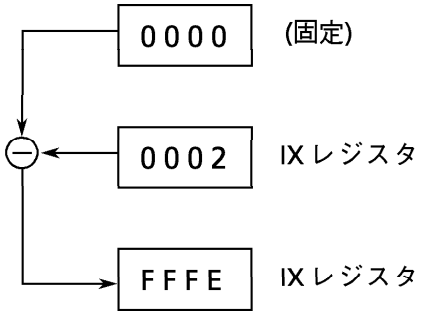
サイズ バイト	ワード	ロング	ニモニック		コード
○	○	×	NEG	r	1 1 0 z 1 r 0 0 0 0 0 1 1 1

フラグ :

S	Z	H	V	N	C
*	*	*	*	1	*

S = 演算結果の最上位ビットの値がセットされます。
Z = 演算結果がゼロのときは“1”、それ以外のときは“0”がセットされます。
H = 演算結果、ビット3からビット4へボローが発生したときは“1”、それ以外のときは“0”がセットされます。
V = 演算結果、オーバフローが発生したときは“1”、それ以外のときは“0”がセットされます。
N = “1”にセットされます。
C = 演算結果、最上位ビットからボローが発生したときは“1”、それ以外のときは“0”がセットされます。

実行例 : IXレジスタ0002Hのとき、
NEG IX
を実行すると、IXレジスタはFFFEHになります。



NOP

＜No Operation　何もしない＞

動　作：何もしない

説　明：何もせず、次の命令の実行に移ります。この命令は、オブジェクトコードが00Hです。

詳　細：

ニモニック	コード
NOP	0 0 0 0 0 0 0 0

フラグ：

S	Z	H	V	N	C
−	−	−	−	−	−

S ＝ 変化なし。
Z ＝ 変化なし。
H ＝ 変化なし。
V ＝ 変化なし。
N ＝ 変化なし。
C ＝ 変化なし。

OR dst, src

< Or 論理和 >

動 作 : dst←dst OR src

説 明 : dstの内容とsrcの内容が論理和演算され、dstへ転送されます。

(真理値表)

A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

詳 細 :

サイズ バイト	サイズ ワード	サイズ ロング	ニモニック		コード
○	○	○	OR	R, r	1 1 z z 1 r 1 1 1 0 0 R
○	○	○	OR	r, #	1 1 z z 1 r 1 1 0 0 1 1 1 0 #<7:0> #<15:8> #<23:16> #<31:24>
○	○	○	OR	R, (mem)	1 m z z m m m m 1 1 1 0 0 R
○	○	○	OR	(mem), R	1 m z z m m m m 1 1 1 0 1 R
○	○	×	OR<W>	(mem), #	1 m 0 z m m m m 0 0 1 1 1 1 1 0 #<7:0> #<15:8>

フラグ ：

S	Z	H	V	N	C
*	*	0	*	0	0

S ＝ 演算結果の最上位ビットの値がセットされます。
Z ＝ 演算結果がゼロのときは“1”、それ以外のときは“0”がセットされます。
H ＝ “0”にセットされます。
V ＝ 演算結果のパリティ(“1”の数)が偶数のときは“1”、奇数のときは“0”がセットされます。ただし、オペランド長が32ビットの場合は、不定値がセットされます。
N ＝ “0”にクリアされます。
C ＝ “0”にクリアされます。

実行例 ： HLレジスタが7350H, IXレジスタが3456Hのとき、
OR HL, IX
を実行すると、HLレジスタは7756Hになります。

```
        0111  0011  0101  0000  ←  HLレジスタ(実行前)
OR  )   0011  0100  0101  0110  ←  IXレジスタ (実行前)
        0111  0111  0101  0110  ←  HLレジスタ(実行後)
```

ORCF num, src

< Or Carry Flag キャリーフラグとの1ビット論理和 >

動　作 : CY ← CY OR src<num>

説　明 : キャリーフラグCYの内容とsrcのビットnumの内容が論理和演算され、キャリーフラグCYへ転送されます。

詳　細 :

サイズ バイト	ワード	ロング	ニモニック		コード
○	○	×	ORCF	#4, r	1 1 0 z 1 r 0 0 1 0 0 0 0 1 0 0 0 0 # 4
○	○	×	ORCF	A, r	1 1 0 z 1 r 0 0 1 0 1 0 0 1
○	×	×	ORCF	#3, (mem)	1 m 1 1 m m m m 1 0 0 0 1 #3
○	×	×	ORCF	A, (mem)	1 m 1 1 m m m m 0 0 1 0 1 0 0 1

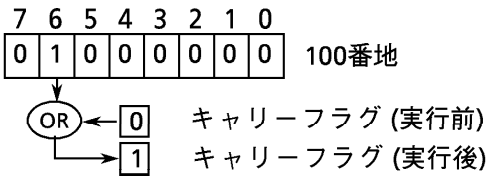
補足　: ビットnumがAレジスタで指定された場合、Aレジスタの下位4ビットの値がビットnumとして使われます。オペランドがバイトのとき、ビットnumの下位の値が8~15の場合、演算結果は不定になります。

フラグ :

S	Z	H	V	N	C
−	−	−	−	−	*

S = 変化なし。
Z = 変化なし。
H = 変化なし。
V = 変化なし。
N = 変化なし。
C = キャリーフラグCYの内容とsrcのビットnumの内容の論理和演算された値がセットされます。

実行例 : 100番地のメモリの内容が01000000B (2進数)で、キャリーフラグCYが0のとき、
ORCF 6, (100H)
を実行すると、キャリーフラグは1になります。



PAA dst

< Pointer Adjust Accumulator ポインタ偶数補正 >

動 作 : if dst <LSB> = 1 then dst ← dst + 1

説 明 : dstのLSB(最下位ビット)が1の場合、dstに1が加算されます。dstのLSBが0の場合、何もしません。
この命令は、dstの内容を偶数にするためのものです。TLCS-900では、メモリ中の16ビットデータ、または、32ビットデータをアクセスするとき、そのデータが偶数番地から配置されている方が、奇数番地からの場合に比べて、バスサイクル数が1回少なくなります。

詳 細 :

サイズ バイト	サイズ ワード	サイズ ロング	ニモニック		コード
×	○	○	PAA	r	11zz1r / 00010100

フラグ :

S	Z	H	V	N	C
−	−	−	−	−	−

S = 変化なし。
Z = 変化なし。
H = 変化なし。
V = 変化なし。
N = 変化なし。
C = 変化なし。

実行例 : XIZレジスタが00234567Hのとき、
PAA XIZ
を実行すると、XIZレジスタは+1されて00234568Hになります。

POP dst

＜Pop　スタック領域からの転送 ＞

動　作 ： dst ← (XSP＋)　　[バイト時: dst←(XSP), XSP←XSP＋1 / ワード時: dst←(XSP), XSP←XSP＋2 / ロング時: dst←(XSP), XSP←XSP＋4]

説　明 ： まず、スタックポインタXSPで示されたメモリ番地の内容が、dstへ転送されます。次に、スタックポインタXSPがオペランドサイズのバイト長だけ加算されます。

詳　細 ：

サイズ バイト	ワード	ロング	ニモニック		コード
○	×	×	POP	F	0 0 0 1 1 0 0 1
○	×	×	POP	A	0 0 0 1 0 1 0 1
×	○	○	POP	R	0 1 0 s 1 R
○	○	○	POP	r	1 1 z z 1 r 0 0 0 0 0 1 0 1
○	○	×	POP＜W＞	(mem)	1 m 1 1 m m m m 0 0 0 0 0 1 z 0

フラグ ：

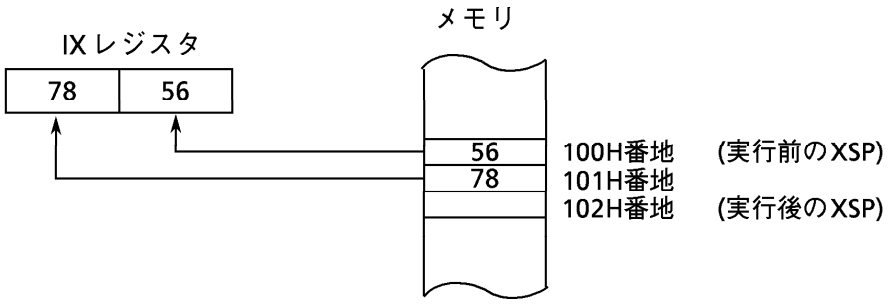
S	Z	H	V	N	C
−	−	−	−	−	−

S ＝ 変化なし。
Z ＝ 変化なし。
H ＝ 変化なし。
V ＝ 変化なし。
N ＝ 変化なし。
C ＝ 変化なし。
(注)　“POP　F”を実行すると、フラグはすべて変化します。

実行例 ： スタックポインタXSPが0100Hで、100H番地の内容が56H、101H番地の内容が78Hのとき、

POP IX

を実行すると、IXレジスタは7856Hになります。また、スタックポインタXSPは、0102Hになります。



POP SR

< Pop SR ステータスレジスタのスタック領域からの転送 >

動 作 : SR←(XSP+)

説 明 : スタックポインタXSPで示されたメモリ番地の内容が、ステータスレジスタへ転送されます。その後、スタックポインタXSPの内容が+2されます。

詳 細 :

サイズ バイト	サイズ ワード	サイズ ロング	ニモニック		コード
×	○	×	POP	SR	0 0 0 0 0 0 1 1

フラグ :

S	Z	H	V	N	C
*	*	*	*	*	*

S =
Z =
H =
V =
N =
C =
} スタックポインタXSPで示されたメモリ番地の内容がセットされます。

注 意 : 本命令の実行は「DI」状態で行ってください。この命令が実際に実行されるタイミングは、この命令がフェッチされるタイミングより、数ステート遅れます。これは、本CPUが命令キュー(4バイト)とパイプライン処理方式を採用しているためです。
また、本命令はステータスレジスタの全ビットを書き替える動作を行いますので、値を書き替えてはいけないビット(MAXビットは“1”しかライトしてはいけません)は、書き替わることがないようにしてください。

PUSH SR

＜Push SR　ステータスレジスタのスタック領域への転送＞

動　作：(－XSP)←SR

説　明：スタックポインタXSPの内容が－2されます。その後、ステータスレジスタSRの内容が、スタックポインタXSPで示されたメモリ番地へ転送されます。

詳　細：

サイズ バイト	サイズ ワード	サイズ ロング	ニモニック		コード
×	○	×	PUSH	SR	0 0 0 0 0 0 1 0

フラグ：

S	Z	H	V	N	C
－	－	－	－	－	－

S ＝ 変化なし。
Z ＝ 変化なし。
H ＝ 変化なし。
V ＝ 変化なし。
N ＝ 変化なし。
C ＝ 変化なし。

PUSH src

＜Push　スタック領域への転送＞

動　作：(−XSP) ← src　　[バイト時: XSP←XSP−1, (XSP)←src / ワード時: XSP←XSP−2, (XSP)←src / ロング時: XSP←XSP−4, (XSP)←src]

説　明：まず、スタックポインタXSPがオペランドサイズのバイト長だけ減算されます。次に、スタックポインタXSPが示すメモリ番地へ、srcの内容が転送されます。

詳　細：

サイズ バイト	ワード	ロング	ニモニック		コード
○	×	×	PUSH	F	0 0 0 1 1 0 0 0
○	×	×	PUSH	A	0 0 0 1 0 1 0 0
×	○	○	PUSH	R	0 0 1 s 1 R
○	○	○	PUSH	r	1 1 z z 1 r 0 0 0 0 0 1 0 0
○	○	×	PUSH＜W＞	#	0 0 0 0 1 0 z 1 #＜7:0＞ #＜15:8＞
○	○	×	PUSH＜W＞	(mem)	1 m 0 z m m m m 0 0 0 0 0 1 0 0

フラグ：

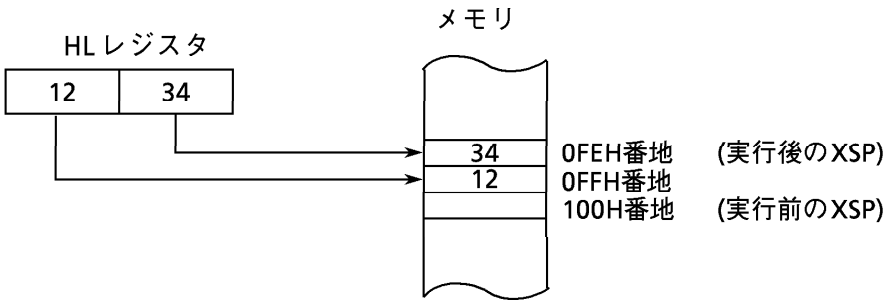
S	Z	H	V	N	C
−	−	−	−	−	−

S ＝ 変化なし。
Z ＝ 変化なし。
H ＝ 変化なし。
V ＝ 変化なし。
N ＝ 変化なし。
C ＝ 変化なし。

実行例 ： スタックポインタXSPが0100Hで、HLレジスタが1234Hのとき、

PUSH HL

を実行すると、00FEH番地は34Hに、00FFH番地は12Hになります。また、スタックポインタXSPは、00FEHになります。



RCF

< Reset Carry Flag　キャリーフラグのリセット >

動　作 : CY ← 0

説　明 : キャリーフラグCYが0にリセットされます。

詳　細 :

ニモニック	コード
RCF	0 0 0 1 0 0 0 0

フラグ :

S	Z	H	V	N	C
–	–	0	–	0	0

S = 変化なし。
Z = 変化なし。
H = “0”にリセットされます。
V = “0”にリセットされます。
N = 変化なし。
C = “0”にリセットされます。

RES num, dst

< Reset 1ビットのリセット >

動 作 : dst <num> ← 0

説 明 : dstのビットnumが“0”にリセットされます。

詳 細 :

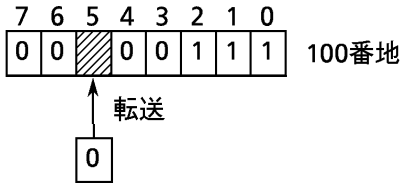
サイズ			ニモニック		コード
バイト	ワード	ロング			
○	○	×	RES	#4, r	1 1 0 z 1 r 0 0 1 1 0 0 0 0 0 0 0 0 # 4
○	×	×	RES	#3, (mem)	1 m 1 1 m m m m 1 0 1 1 0 #3

フラグ :

S	Z	H	V	N	C
–	–	–	–	–	–

S = 変化なし。
Z = 変化なし。
H = 変化なし。
V = 変化なし。
N = 変化なし。
C = 変化なし。

実行例 : 100番地のメモリの内容が00100111B (2進数)のとき、
RES 5, (100H)
を実行すると、100番地のメモリの内容は00000111B (2進数)になります。



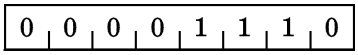
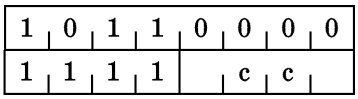
RET condition

< Return リターン >

動　作 : if ccが真 then 32ビットPC ← (XSP), XSP ← XSP+4.

説　明 : オペランドのconditionが真の場合、スタック領域からプログラムカウンタPCへリターンアドレスがPOPされます。

詳　細 :

ニモニック	コード
RET	
RET cc	

フラグ :

S	Z	H	V	N	C
−	−	−	−	−	−

S = 変化なし。
Z = 変化なし。
H = 変化なし。
V = 変化なし。
N = 変化なし。
C = 変化なし。

実行例 : スタックポインタXSPが0FCHで、0FCH番地のメモリの内容が9000H (ロングデータ) のとき、

RET

を実行すると、スタックポインタXSPは100Hになり、9000H番地にジャンプ (リターン) します。

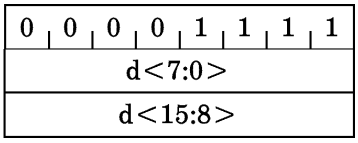
RETD num

< Return and Deallocate リターン&パラメータ領域の削除 >

動 作 : 32ビットPC ← (XSP), XSP ← XSP+4, XSP ← XSP+num

説 明 : まず、スタック領域からプログラムカウンタPCへリターンアドレスがPOPされます。次に、スタックポインタXSPにnum (符号付) の値が加算されます。

詳 細 :

ニモニック	コード
RETD d16	

フラグ :

S	Z	H	V	N	C
−	−	−	−	−	−

S = 変化なし。
Z = 変化なし。
H = 変化なし。
V = 変化なし。
N = 変化なし。
C = 変化なし。

実行例 : スタックポインタXSPが0FCHで、0FCH番地のメモリの内容が9000H (ロングデータ) のとき、

```
RETD  8
```

を実行すると、スタックポインタXSPは0FCH+4+8 → 108Hになり、9000H番地にジャンプ (リターン) します。
下記に、RETD命令の活用例を示します。この例では、サブルーチンを呼ぶ前に8バイトのパラメータをスタックにPUSHし、サブルーチン処理終了後、RETD命令で、その使用済のパラメータ領域を削除しています。

```
PUSH  WA                SAMPLE : ⋮
PUSH  BC                         ⋮
PUSH  XIX                        ⋮
CALL  SAMPLE                     RETD  8
  ⋮
  ⋮
```

RETI

＜ Return from Interrupt　割り込み処理からのリターン ＞

動　作 : SR ← (XSP), 32ビットPC ← (XSP＋2),
XSP ← XSP＋6
上記動作後、割り込みネスティングカウンタINTNESTを−1します。

説　明 : スタック領域から「ステータスレジスタSRとプログラムカウンタPC」へデータがPOPされます。
上記動作後、割り込みネスティングカウンタINTNESTを−1します。

詳　細 :

ニモニック	コード
RETI	0 0 0 0 0 1 1 1

フラグ :

S	Z	H	V	N	C
*	*	*	*	*	*

S ＝ スタック領域からPOPした値になります。
Z ＝ スタック領域からPOPした値になります。
H ＝ スタック領域からPOPした値になります。
V ＝ スタック領域からPOPした値になります。
N ＝ スタック領域からPOPした値になります。
C ＝ スタック領域からPOPした値になります。

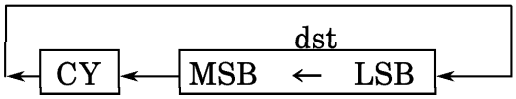
RL num, dst

< Rotate Left キャリーフラグを含む左ローテート >

動 作 : {CY & dst←CY & dstの左ローテート値} Repeat num回

説 明 : キャリーフラグCYとdstの連結した内容が左へローテートされます。これがnum回繰り返されます。

説明図:



詳 細 :

サイズ バイト	ワード	ロング	ニモニック		コード
○	○	○	RL	#4, r	1 1 z z 1 r 1 1 1 0 1 0 1 0 0 0 0 0 # 4
○	○	○	RL	A, r	1 1 z z 1 r 1 1 1 1 1 0 1 0
○	○	×	RL <W>	(mem)	1 m 0 z m m m m 0 1 1 1 1 0 1 0

補 足 : ローテート回数numがAレジスタで指定された場合、Aレジスタの下位4ビットの値がローテート回数として使われます。0を指定すると、16回ローテートされます。
dstがメモリの場合、ローテート回数は1回に制限されます。

フラグ :

S	Z	H	V	N	C
*	*	0	*	0	*

S = ローテート後のdstの最上位ビットの値がセットされます。
Z = ローテート後のdstの内容がゼロのときは“1”、それ以外のときは“0”がセットされます。
H = “0”にリセットされます。
V = ローテート後、dstのパリティ(“1”の数)が偶数のときは“1”、それ以外のときは“0”がセットされます。ただし、オペランド長が32ビットの場合は不定値がセットされます。
N = “0”にリセットされます。
C = ローテート後の値がセットされます。

実行例 ： HLレジスタが6230Hで、キャリーフラグCYが “1” のとき、

```
RL  4,HL
```

を実行すると、HLレジスタは230BH、キャリーフラグCYは “0” になります。

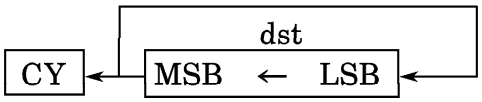
RLC num, dst

< Rotate Left without Carry キャリーフラグを含まない左ローテート >

動 作 : {CY←dst <MSB>、 dst←dstの左ローテート値} Repeat num回

説 明 : dstのMSBの内容がキャリーフラグCYへ転送され、dstの内容が左へローテートされます。これがnum回繰り返されます。

説明図:



詳 細 :

サイズ バイト	ワード	ロング	ニモニック		コード
○	○	○	RLC	#4, r	1 1 z z 1 r 1 1 1 0 1 0 0 0 0 0 0 0 # 4
○	○	○	RLC	A, r	1 1 z z 1 r 1 1 1 1 1 0 0 0
○	○	×	RLC <W> (mem)		1 m 0 z m m m m 0 1 1 1 1 0 0 0

補 足 : ローテート回数numがAレジスタで指定された場合、Aレジスタの下位4ビットの値がローテート回数として使われます。0を指定すると、16回ローテートされます。
dstがメモリの場合、ローテート回数は1回に制限されます。

フラグ :

S	Z	H	V	N	C
*	*	0	*	0	*

S = ローテート後のdstの最上位ビットの値がセットされます。
Z = ローテート後のdstの内容がゼロのときは“1”、それ以外のときは“0”がセットされます。
H = “0”にリセットされます。
V = ローテート後、dstのパリティ(“1”の数)が偶数のときは“1”、それ以外のときは“0”がセットされます。ただし、オペランド長が32ビットの場合は不定値がセットされます。
N = “0”にリセットされます。
C = 最終のローテート前のdstの最上位ビットの値がセットされます。

実行例 ： HLレジスタが1230Hのとき、

RLC 4, HL

を実行すると、HLレジスタは2301H、キャリーフラグCYは “1” になります。

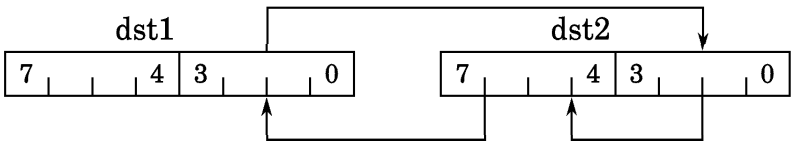
RLD dst1, dst2

< Rotate Left Digit 4ビットの左回転 >

動 作 : dst1<3:0>←dst2<7:4>, dst2<7:4>←dst2<3:0>, dst2<3:0>←dst1<3:0>

説 明 : dst1の下位4ビットと、dst2の内容が、4ビット単位で左へローテートされます。

説明図 :



詳 細 :

サイズ バイト	ワード	ロング	ニモニック		コード
○	×	×	RLD	[A,](mem)	1 m 0 0 m m m m 0 0 0 0 0 1 1 0

フラグ :

S	Z	H	V	N	C
*	*	0	*	0	−

S = ローテート後のAレジスタの最上位ビットの値がセットされます。
Z = ローテート後のAレジスタの内容がゼロのときは“1”、それ以外のときは“0”がセットされます。
H = “0”にリセットされます。
V = ローテート後、Aレジスタのパリティ(“1”の数)が偶数のときは“1”、それ以外のときは“0”がセットされます。
N = “0”にリセットされます。
C = 変化なし。

実行例 : Aレジスタが12Hで、100H番地のメモリの内容が34Hのとき、

```
RLD  A,(100H)
```

を実行すると、Aレジスタは13H, 100番地のメモリの内容は42Hになります。

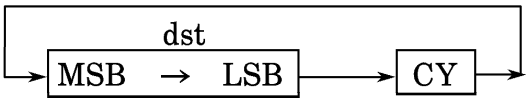
RR num, dst

< Rotate Right キャリーフラグを含む右ローテート >

動 作 : {CY & dst←CY & dstの右ローテート値} Repeat num回

説 明 : キャリーフラグCYとdstの連結した内容が右へローテートされます。これがnum回繰り返されます。

説明図:



詳 細 :

サイズ バイト	ワード	ロング	ニモニック		コード
○	○	○	RR	#4, r	1 1 z z 1 r 1 1 1 0 1 0 1 1 0 0 0 0 # 4
○	○	○	RR	A, r	1 1 z z 1 r 1 1 1 1 1 0 1 1
○	○	×	RR <W>	(mem)	1 m 0 z m m m m 0 1 1 1 1 0 1 1

補 足 : ローテート回数numがAレジスタで指定された場合、Aレジスタの下位4ビットの値がローテート回数として使われます。0を指定すると、16回ローテートされます。
dstがメモリの場合、ローテート回数は1回に制限されます。

フラグ :

S	Z	H	V	N	C
*	*	0	*	0	*

S = ローテート後のdstの最上位ビットの値がセットされます。
Z = ローテート後のdstの内容がゼロのときは“1”、それ以外のときは“0”がセットされます。
H = “0”にリセットされます。
V = ローテート後、dstのパリティ(“1”の数)が偶数のときは“1”、それ以外のときは“0”がセットされます。ただし、オペランド長が32ビットの場合は不定値がセットされます。
N = “0”にリセットされます。
C = ローテート後の値がセットされます。

実行例 ： HLレジスタが6230Hで、キャリーフラグCYが“1”のとき、

RR 4, HL

を実行すると、HLレジスタは1623H、キャリーフラグCYは “0” になります。

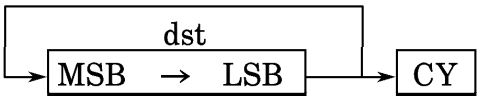
RRC num, dst

< Rotate Right without Carry キャリーフラグを含まない右ローテート >

動 作 : {CY←dst <LSB>、 dst←dstの右ローテート値} Repeat num回

説 明 : dstのLSBの内容がキャリーフラグCYへ転送され、dstの内容が右へローテートされます。これがnum回繰り返されます。

説明図:



詳 細 :

サイズ バイト	ワード	ロング	ニモニック		コード
○	○	○	RRC	#4, r	1 1 z z 1 r 1 1 1 0 1 0 0 1 0 0 0 0 # 4
○	○	○	RRC	A, r	1 1 z z 1 r 1 1 1 1 1 0 0 1
○	○	×	RRC <W> (mem)		1 m 0 z m m m m 0 1 1 1 1 0 0 1

補 足 : ローテート回数numがAレジスタで指定された場合、Aレジスタの下位4ビットの値がローテート回数として使われます。0を指定すると、16回ローテートされます。
dstがメモリの場合、ローテート回数は1回に制限されます。

フラグ :

S	Z	H	V	N	C
*	*	0	*	0	*

S = ローテート後のdstの最上位ビットの値がセットされます。
Z = ローテート後のdstの内容がゼロのときは“1”、それ以外のときは“0”がセットされます。
H = “0”にリセットされます。
V = ローテート後、dstのパリティ(“1”の数)が偶数のときは“1”、それ以外のときは“0”がセットされます。ただし、オペランド長が32ビットの場合は不定値がセットされます。
N = “0”にリセットされます。
C = 最終のローテート前のdstの最上位ビットの値がセットされます。

実行例 ： HLレジスタが1230Hのとき、

RRC 4, HL

を実行すると、HLレジスタは0123H、キャリーフラグCYは “0” になります。

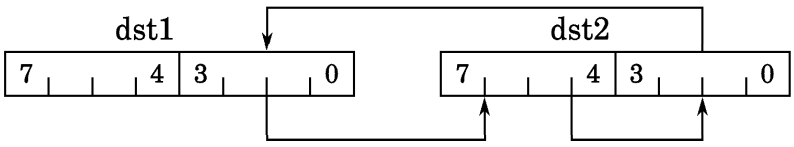
RRD dst1, dst2

< Rotate Right Digit 4ビットの右回転 >

動　作 : dst1<3:0>←dst2<3:0>, dst2<7:4>←dst1<3:0>, dst2<3:0>←dst2<7:4>

説　明 : dst1の下位4ビットと、dst2の内容が、4ビット単位で右へローテートされます。

説明図 :



詳　細 :

サイズ バイト	ワード	ロング	ニモニック		コード
○	×	×	RRD	[A,](mem)	1 m 0 0 m m m m 0 0 0 0 0 1 1 1

フラグ :

S	Z	H	V	N	C
*	*	0	*	0	−

S = ローテート後のAレジスタの最上位ビットの値がセットされます。
Z = ローテート後のAレジスタの内容がゼロのときは“1”、それ以外のときは“0”がセットされます。
H = “0”にリセットされます。
V = ローテート後、Aレジスタのパリティ(“1”の数)が偶数のときは“1”、それ以外のときは“0”がセットされます。
N = “0”にリセットされます。
C = 変化なし。

実行例 : Aレジスタが12Hで、100H番地のメモリの内容が34Hのとき、
RRD A,(100H)
を実行すると、Aレジスタは14H,100番地のメモリの内容は23Hになります。

SBC dst, src

< Subtract with Carry キャリー付き減算 >

動 作 : dst←dst−src−CY

説 明 : dstの内容からsrcの内容とキャリーフラグCYの内容が減算され、dstへ転送されます。

詳 細 :

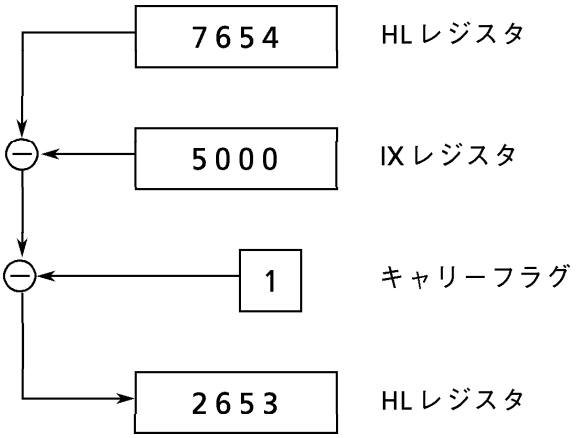
サイズ バイト	ワード	ロング	ニモニック		コード
○	○	○	SBC	R, r	11zz1r 10110R
○	○	○	SBC	r, #	11zz1r 11001011 #<7:0> #<15:8> #<23:16> #<31:24>
○	○	○	SBC	R, (mem)	1mzzmmmm 10110R
○	○	○	SBC	(mem), R	1mzzmmmm 10111R
○	○	×	SBC<W>	(mem), #	1m0zmmmm 00111011 #<7:0> #<15:8>

フラグ : S Z H V N C

S	Z	H	V	N	C
*	*	*	*	1	*

S = 演算結果の最上位ビットの値がセットされます。
Z = 演算結果がゼロのときは“1”、それ以外のときは“0”がセットされます。
H = 演算結果、ビット3からビット4へボローが発生したときは“1”、それ以外のときは“0”がセットされます。ただし、オペランド長が32ビットの場合は、不定値がセットされます。
V = 演算結果、オーバフローが発生したときは“1”、それ以外のときは“0”がセットされます。
N = “1”にセットされます。
C = 演算結果、最上位ビットからボローが発生したときは“1”、それ以外のときは“0”がセットされます。

実行例 : HLレジスタが7654H, IXレジスタが5000H, キャリーフラグCYが1のとき、
SBC HL, IX
を実行すると、HLレジスタは2653Hになります。



SCC condition, dst

< Set Condition Code コンディションコードによる値のセット >

動 作 : if ccが真 then dst←“1” else dst←“0”.

説 明 : オペランドのconditionが真の場合、“1”がdstへ転送され、偽の場合、“0”がdstへ転送されます。

詳 細 :

サイズ バイト	ワード	ロング	ニモニック		コード
○	○	×	SCC	cc, r	1 1 0 z 1 r 0 1 1 1 c c

フラグ :

S	Z	H	V	N	C
–	–	–	–	–	–

S = 変化なし。
Z = 変化なし。
H = 変化なし。
V = 変化なし。
N = 変化なし。
C = 変化なし。

実行例 : Vフラグの内容が“1”のとき、
SCC OV, HL
を実行すると、HLレジスタは0001Hになります。

SCF

< Set Carry Flag　キャリーフラグのセット >

動　作 : CY ← 1

説　明 : キャリーフラグCYが “1” にセットされます。

詳　細 :

ニモニック	コード
SCF	0 0 0 1 0 0 0 1

フラグ :

S	Z	H	V	N	C
–	–	0	–	0	1

S = 変化なし。
Z = 変化なし。
H = “0”にリセットされます。
V = 変化なし。
N = “0”にリセットされます。
C = “1”にセットされます。

SET num, dst

< Set 1ビットのセット >

動　作 : dst <num> ←1

説　明 : dstのビットnumが“1”にセットされます。

詳　細 :

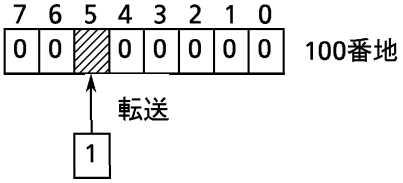
サイズ			ニモニック		コード
バイト	ワード	ロング			
○	○	×	SET	#4, r	1 1 0 z 1 r 0 0 1 1 0 0 0 1 0 0 0 0 # 4
○	×	×	SET	#3, (mem)	1 m 1 1 m m m m 1 0 1 1 1 #3

フラグ :

S	Z	H	V	N	C
–	–	–	–	–	–

S = 変化なし。
Z = 変化なし。
H = 変化なし。
V = 変化なし。
N = 変化なし。
C = 変化なし。

実行例 : 100番地のメモリの内容が00000000B (2進数)のとき、
SET 5, (100H)
を実行すると、100番地のメモリの内容は00100000B (2進数)になります。



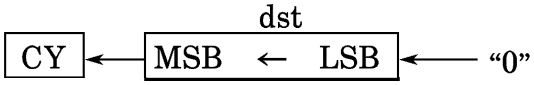
SLA num, dst

< Shift Left Arithmetic 算術左シフト >

動 作 : {CY←dst <MSB>, dst←dstの左シフト値、dst <LSB> ←0} Repeat num回

説 明 : dstのMSBの内容がキャリーフラグCYへ転送され、dstの内容が左へシフトされ、0がdstのLSBへ転送されます。これがnum回繰り返されます。

説明図:



詳 細 :

サイズ バイト	ワード	ロング	ニモニック		コード
○	○	○	SLA	#4, r	1 1 z z 1 r 1 1 1 0 1 1 0 0 0 0 0 0 # 4
○	○	○	SLA	A, r	1 1 z z 1 r 1 1 1 1 1 1 0 0
○	○	×	SLA <W> (mem)		1 m 0 z m m m m 0 1 1 1 1 1 0 0

補 足 : シフト回数numがAレジスタで指定された場合、Aレジスタの下位4ビットの値がシフト回数として使われます。0を指定すると、16回シフトされます。
dstがメモリの場合、シフト回数は1回に制限されます。

フラグ :

S	Z	H	V	N	C
*	*	0	*	0	*

S = シフト後のdstの最上位ビットの値がセットされます。
Z = シフト後のdstの内容がゼロのときは"1"、それ以外のときは"0"がセットされます。
H = "0"にリセットされます。
V = シフト後、dstのパリティ ("1"の数) が偶数のときは"1"、それ以外のときは"0"がセットされます。ただし、オペランド長が32ビットの場合は不定値がセットされます。
N = "0"にリセットされます。
C = 最終のシフト前のdstの最上位ビットの値がセットされます。

実行例 ： HLレジスタが1234Hのとき、

SLA 4,HL

を実行すると、HLレジスタは2340H、キャリーフラグCYは “1” になります。

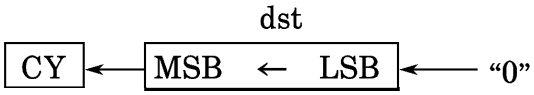
SLL num, dst

< Shift Left Logical 論理左シフト >

動 作 : {CY←dst <MSB> , dst←dstの左シフト値、dst <LSB> ←0} Repeat num回

説 明 : dstのMSBの内容がキャリーフラグCYへ転送され、dstの内容が左へシフトされ、0がdstのLSBへ転送されます。これがnum回繰り返されます。

説明図:



詳 細 :

サイズ バイト	ワード	ロング	ニモニック		コード
○	○	○	SLL	#4, r	11zz1 r / 11101110 / 0000 #4
○	○	○	SLL	A, r	11zz1 r / 11111110
○	○	×	SLL <W> (mem)		1m0zmmmm / 01111110

補 足 : シフト回数numがAレジスタで指定された場合、Aレジスタの下位4ビットの値がシフト回数として使われます。0を指定すると、16回シフトされます。
dstがメモリの場合、シフト回数は1回に制限されます。

フラグ :

S	Z	H	V	N	C
*	*	0	*	0	*

S = シフト後のdstの最上位ビットの値がセットされます。
Z = シフト後のdstの内容がゼロのときは"1"、それ以外のときは"0"がセットされます。
H = "0"にリセットされます。
V = シフト後、dstのパリティ("1"の数)が偶数のときは"1"、それ以外のときは"0"がセットされます。ただし、オペランド長が32ビットの場合は不定値がセットされます。
N = "0"にリセットされます。
C = 最終のシフト前のdstの最上位ビットの値がセットされます。

実行例 ： HLレジスタが1234Hのとき、

SLL 4, HL

を実行すると、HLレジスタは2340H、キャリーフラグCYは “1” になります。

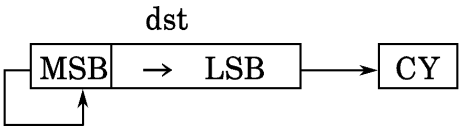
SRA num, dst

< Shift Right Arithmetic 算術右シフト >

動 作 : {CY←dst <MSB>, dst←dstの右シフト値、dst <MSB> は固定} Repeat num回

説 明 : dstのLSBの内容がキャリーフラグCYへ転送され、dstの内容が右へシフト (MSBは固定) されます。これがnum回繰り返されます。

説明図:



詳 細 :

サイズ バイト	ワード	ロング	ニモニック		コード
○	○	○	SRA	#4, r	11zz1 r / 11101101 / 0000 #4
○	○	○	SRA	A, r	11zz1 r / 11111101
○	○	×	SRA <W> (mem)		1m0zmmmm / 01111101

補 足 : シフト回数numがAレジスタで指定された場合、Aレジスタの下位4ビットの値がシフト回数として使われます。0を指定すると、16回シフトされます。
dstがメモリの場合、シフト回数は1回に制限されます。

フラグ :

S	Z	H	V	N	C
*	*	0	*	0	*

S = シフト後のdstの最上位ビットの値がセットされます。
Z = シフト後のdstの内容がゼロのときは "1"、それ以外のときは "0" がセットされます。
H = "0" にリセットされます。
V = シフト後、dstのパリティ ("1" の数) が偶数のときは "1"、それ以外のときは "0" がセットされます。ただし、オペランド長が32ビットの場合は、不定値がセットされます。
N = "0" にリセットされます。
C = 最終のシフト前のdstの最下位ビットの値がセットされます。

実行例 ： HLレジスタが8230Hのとき、

SRA 4, HL

を実行すると、HLレジスタはF823H、キャリーフラグCYは “0” になります。

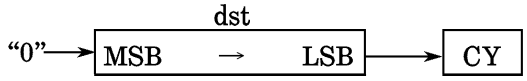
SRL num, dst

< Shift Right Logical 論理右シフト >

動 作 : {CY←dst<LSB>, dst←dstの右シフト値, dst<MSB>←0} Repeat num回

説 明 : dstのLSBの内容がキャリーフラグCYへ転送され、dstの内容が右へシフトされ、0がdstのMSBへ転送されます。これがnum回繰り返されます。

説明図 :



詳 細 :

サイズ バイト	サイズ ワード	サイズ ロング	ニモニック		コード
○	○	○	SRL	#4, r	11zz1 r / 11101111 / 0000 #4
○	○	○	SRL	A, r	11zz1 r / 11111111
○	○	×	SRL<W>	(mem)	1m0zmmmm / 01111111

補 足 : シフト回数numがAレジスタで指定された場合、Aレジスタの下位4ビットの値がシフト回数として使われます。0を指定すると、16回シフトされます。
dstがメモリの場合、シフト回数は1回に制限されます。

フラグ :

S	Z	H	V	N	C
*	*	0	*	0	*

S = シフト後のdstの最上位ビットの値がセットされます。
Z = シフト後のdstの内容がゼロのときは"1"、それ以外のときは"0"がセットされます。
H = "0"にリセットされます。
V = シフト後、dstのパリティ("1"の数)が偶数のときは"1"、それ以外のときは"0"がセットされます。ただし、オペランド長が32ビットの場合は不定値がセットされます。
N = "0"にリセットされます。
C = 最終シフト前のdstの最下位ビットの値がセットされます。

実行例 ： HLレジスタが1238Hのとき、

SRL 4, HL

を実行すると、HLレジスタは0123H、キャリーフラグCYは “1” になります。

STCF num, dst

< Store Carry Flag キャリーフラグからの1ビット転送 >

動 作 : dst<num>←CY

説 明 : キャリーフラグCYの内容が、dstのビットnumへ転送されます。

詳 細 :

サイズ バイト	サイズ ワード	サイズ ロング	ニモニック		コード
○	○	×	STCF	#4, r	1 1 0 z 1 r / 0 0 1 0 0 1 0 0 / 0 0 0 0 # 4
○	○	×	STCF	A, r	1 1 0 z 1 r / 0 0 1 0 1 1 0 0
○	×	×	STCF	#3, (mem)	1 m 1 1 m m m m / 1 0 1 0 0 #3
○	×	×	STCF	A, (mem)	1 m 1 1 m m m m / 0 0 1 0 1 1 0 0

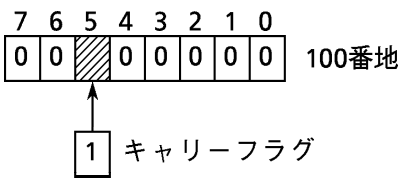
補 足 : ビットnumがAレジスタで指定された場合、Aレジスタの下位4ビットの値がビットnumとして使われます。オペランドがバイトのとき、ビットnumの下位4ビットの値が8~15の場合、オペランドの値は変化しません。

フラグ :

S	Z	H	V	N	C
−	−	−	−	−	−

S = 変化なし。
Z = 変化なし。
H = 変化なし。
V = 変化なし。
N = 変化なし。
C = 変化なし。

実行例 : 100番地のメモリの内容が00Hで、キャリーフラグCYが1のとき、
STCF 5, (100H)
を実行すると、100番地のメモリの内容は00100000B (2進数)になります。



SUB dst, src

< Subtract 減算 >

動 作 : dst←dst−src

説 明 : dstの内容からsrcの内容が減算され、dstへ転送されます。

詳 細 :

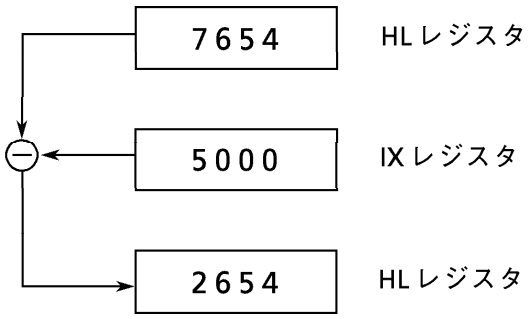
サイズ バイト	サイズ ワード	サイズ ロング	ニモニック		コード
○	○	○	SUB	R, r	11zz1r / 10100R
○	○	○	SUB	r, #	11zz1r / 11001010 / #<7:0> / #<15:8> / #<23:16> / #<31:24>
○	○	○	SUB	R, (mem)	1mzzmmmm / 10100R
○	○	○	SUB	(mem), R	1mzzmmmm / 10101R
○	○	×	SUB<W>	(mem), #	1m0zmmmm / 00111010 / #<7:0> / #<15:8>

フラグ：

S	Z	H	V	N	C
*	*	*	*	1	*

S ＝ 演算結果の最上位ビットの値がセットされます。
Z ＝ 演算結果がゼロのときは“1”、それ以外のときは“0”がセットされます。
H ＝ 演算結果、ビット3からビット4へボローが発生したときは“1”、それ以外のときは“0”がセットされます。ただし、オペランド長が32ビットの場合は、不定値がセットされます。
V ＝ 演算結果、オーバフローが発生したときは“1”、それ以外のときは“0”がセットされます。
N ＝ “1”にセットされます。
C ＝ 演算結果、最上位ビットからボローが発生したときは“1”、それ以外のときは“0”がセットされます。

実行例 ： HLレジスタが7654H, IXレジスタが5000Hのとき、
SUB HL, IX
を実行すると、HLレジスタは2654Hになります。



SWI num

< Software Interrupt ソフトウエア割り込み >

動 作 : ① XSP ← XSP－6
② (XSP) ← SR
③ (XSP＋2) ← 32ビットPC
④ PC ← (ベクタベース番地＋num×4)
補足： ベクタベース番地は、派生品ごとに定義されます。

説 明 : スタック領域へ、ステータスレジスタSRの内容と、このSWI命令の次の番地を示しているプログラムカウンタPCの内容が退避され、“ベクタベース番地＋num×4”番地の内容で示される飛び先へジャンプします。

詳 細 :

ニモニック	コード
SWI [#3]	1 1 1 1 1 #3

補足 : オペランドの値は、0~7まで指定できます。オペランドの記述を省略すると、“SWI 7”と解釈されます。

参考 : ステータスレジスタSRの構成は下記のとおりです。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SYSM	IFF2	IFF1	IFF0	MAX	RFP2	RFP1	RFP0	S	Z	“0”	H	“0”	V	N	C

フラグ :

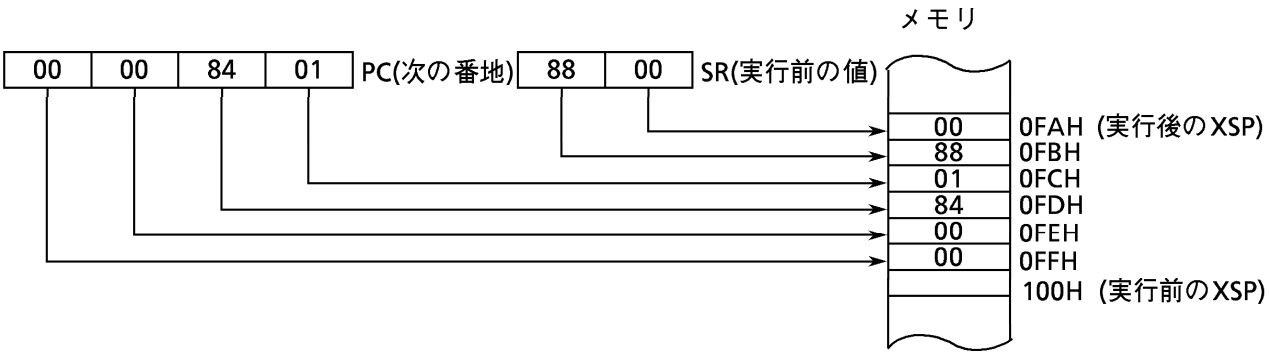
S	Z	H	V	N	C
－	－	－	－	－	－

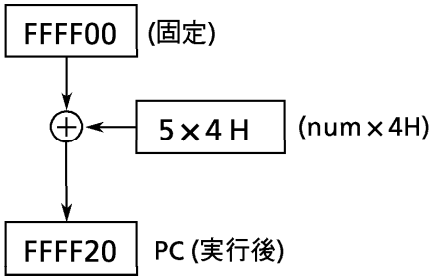
S = 変化なし。
Z = 変化なし。
H = 変化なし。
V = 変化なし。
N = 変化なし。
C = 変化なし。

実行例 : スタックポインタXSPが100H, ステータスレジスタSRが8800Hで、8400H番地のメモリにある、

SWI 5

を実行すると、00FAH番地のメモリに旧ステータスレジスタSRの内容8800Hが、00FCH番地のメモリにプログラムカウンタPCの内容00008401Hがライトされ、FFFF20H番地の内容で示されるメモリ番地へジャンプします。





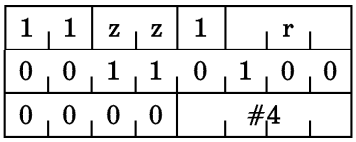
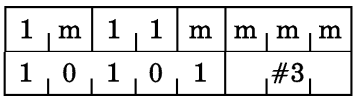
TSET num, dst

< Test and Set 1ビットのテストとセット >

動　作 : Zフラグ ←dst <num> の反転値
dst <num> ←1

説　明 : dstのビットnumの反転値が、Zフラグへ転送されます。その後、dstのビットnumが“1”にセットされます。

詳　細 :

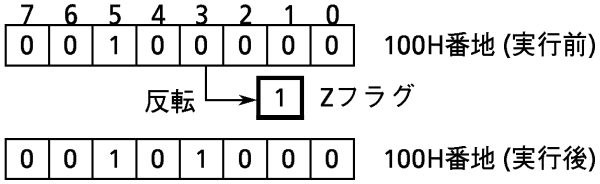
サイズ バイト	サイズ ワード	サイズ ロング	ニモニック		コード
○	○	×	TSET	#4, r	
○	×	×	TSET	#3, (mem)	

フラグ :

S	Z	H	V	N	C
×	*	1	×	0	−

S = 不定値がセットされます。
Z = src <num> の反転値がセットされます。
H = “1” にセットされます。
V = 不定値がセットされます。
N = “0” にセットされます。
C = 変化なし。

実行例 : 100番地のメモリの内容が00100000B (2進数) のとき、
TSET 3, (100H)
を実行すると、Zフラグは1になり、100H番地のメモリの内容は00101000B (2進数) になります。



UNLK dst

< Unlink スタックフレームの削除 >

動 作 : XSP ← dst, dst ← (XSP+)

説 明 : dstの内容が、スタックポインタXSPへ転送されます。その後、スタック領域からロングワードデータがdstへPOPされます。この命令は、LINK命令とペアで使用します。

詳 細 :

サイズ バイト	ワード	ロング	ニモニック		コード
×	×	○	UNLK	r	1 1 1 0 1 r 0 0 0 0 1 1 0 1

フラグ :

S	Z	H	V	N	C
–	–	–	–	–	–

S = 変化なし。
Z = 変化なし。
H = 変化なし。
V = 変化なし。
N = 変化なし。
C = 変化なし。

実行例 : LINK命令実行後、

```
UNLK XIZ
```

を実行すると、スタックポインタXSPとXIZレジスタは、LINK命令実行前と同じ値になります(詳しくは、LINK命令のページを参照してください)。

XOR dst, src

< Exclusive or 排他的論理和 >

動 作 : dst←dst XOR src

説 明 : dstの内容とsrcの内容が排他的論理和演算され、dstへ転送されます。

(真理値表)

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

詳 細 :

サイズ バイト	サイズ ワード	サイズ ロング	ニモニック		コード
○	○	○	XOR	R, r	1 1 z z 1 r 1 1 0 1 0 R
○	○	○	XOR	r, #	1 1 z z 1 r 1 1 0 0 1 1 0 1 #<7:0> #<15:8> #<23:16> #<31:24>
○	○	○	XOR	R, (mem)	1 m z z m m m m 1 1 0 1 0 R
○	○	○	XOR	(mem), R	1 m z z m m m m 1 1 0 1 1 R
○	○	×	XOR<W>	(mem), #	1 m 0 z m m m m 0 0 1 1 1 1 0 1 #<7:0> #<15:8>

フラグ : S Z H V N C

S	Z	H	V	N	C
*	*	0	*	0	0

S = 演算結果の最上位ビットの値がセットされます。
Z = 演算結果がゼロのときは“1”、それ以外のときは“0”がセットされます。
H = “0”にセットされます。
V = 演算結果のパリティ(“1”の数)が偶数のときは“1”、奇数のときは“0”がセットされます。ただし、オペランド長が32ビットの場合は、不定値がセットされます。
N = “0”にクリアされます。
C = “0”にクリアされます。

実行例 : HLレジスタが7350H, IXレジスタが3456Hのとき、
XOR HL, IX
を実行すると、HLレジスタは4706Hになります。

```
      0111  0011  0101  0000  ←  HLレジスタ(実行前)
XOR)  0011  0100  0101  0110  ←  IXレジスタ (実行前)
      0100  0111  0000  0110  ←  HLレジスタ(実行後)
```

XORCF num, src

< Enclusive Or Carry Flag キャリーフラグとの1ビット排他的論理和 >

動 作 : CY ← CY XOR src<num>

説 明 : キャリーフラグCYの内容とsrcのビットnumの内容が排他的論理和演算され、キャリーフラグCYへ転送されます。

詳 細 :

サイズ バイト	ワード	ロング	ニモニック		コード
○	○	×	XORCF	#4, r	1 1 0 z 1 r / 0 0 1 0 0 0 1 0 / 0 0 0 0 #4
○	○	×	XORCF	A, r	1 1 0 z 1 r / 0 0 1 0 1 0 1 0
○	×	×	XORCF	#3, (mem)	1 m 1 1 m m m m / 1 0 0 1 0 #3
○	×	×	XORCF	A, (mem)	1 m 1 1 m m m m / 0 0 1 0 1 0 1 0

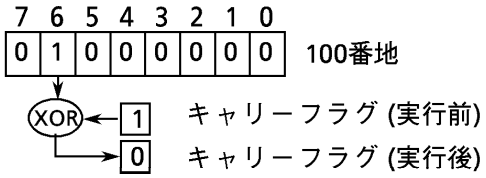
補足 : ビットnumがAレジスタで指定された場合、Aレジスタの下位4ビットの値がビットnumとして使われます。オペランドがバイトのとき、ビットnumの下位4ビットの値が8~15の場合、演算結果は不定になります。

フラグ :

S	Z	H	V	N	C
−	−	−	−	−	*

S = 変化なし。
Z = 変化なし。
H = 変化なし。
V = 変化なし。
N = 変化なし。
C = キャリーフラグCYの内容とsrcのビットnumの内容の排他的論理和演算された値がセットされます。

実行例 : 100番地のメモリの内容が01000000B (2進数)で、キャリーフラグCYが1のとき、
XORCF 6, (100H)
を実行すると、キャリーフラグは0になります。



ZCF

< Zero flag to Carry Flag Zフラグをキャリーフラグへ転送 >

動 作 : CY ←Zフラグの反転値

説 明 : Zフラグの反転値が、キャリーフラグCYへ転送されます。

詳 細 :

ニモニック	コード
ZCF	0 0 0 1 0 0 1 1

フラグ :

S	Z	H	V	N	C
−	−	×	−	0	*

S = 変化なし。
Z = 変化なし。
H = 不定値がセットされます。
V = 変化なし。
N = “0”にリセットされます。
C = Zフラグの反転値がセットされます。

実行例 : Zフラグが0のとき、
ZCF
を実行すると、キャリーフラグCYは1になります。