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	1 1 z z 1 r 1 0 1 1 0 R
○	○	○	SBC	r, #	1 1 z z 1 r 1 1 0 0 1 0 1 1 #<7:0> #<15:8> #<23:16> #<31:24>
○	○	○	SBC	R, (mem)	1 m z z m m m m 1 0 1 1 0 R
○	○	○	SBC	(mem), R	1 m z z m m m m 1 0 1 1 1 R
○	○	×	SBC<W>	(mem), #	1 m 0 z m m m m 0 0 1 1 1 0 1 1 #<7:0> #<15:8>

フラグ：

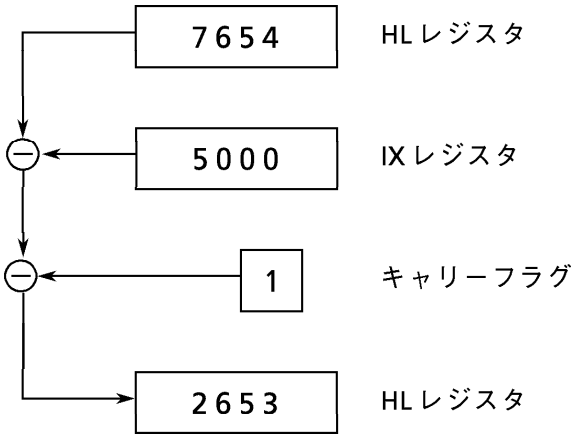
S	Z	H	V	N	C
*	*	*	*	1	*

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, 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 :

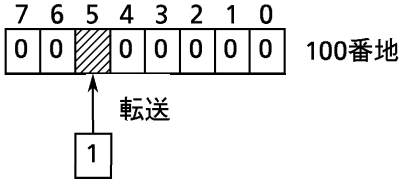
サイズ			ニモニック		コード
バイト	ワード	ロング			
○	○	×	SET	#4, r	1 1 0 z 1 r 0 0 1 1 0 0 0 1 0 0 0 0 # 4
○	×	×	SET	#3, (mem)	1 m 1 1 m m m m 1 0 1 1 1 #3

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。



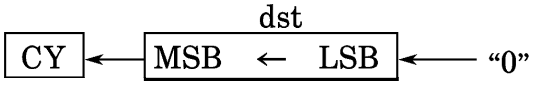
SLA num, dst

< 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。

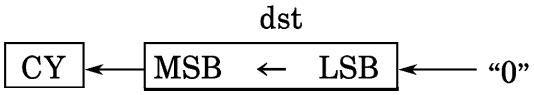
SLL num, dst

< 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。

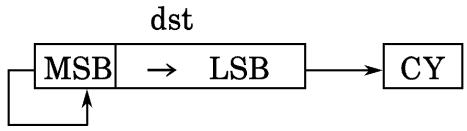
SRA num, dst

< 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。

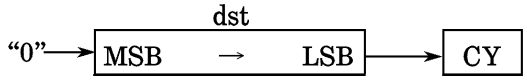
SRL num, dst

< 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	ワード	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

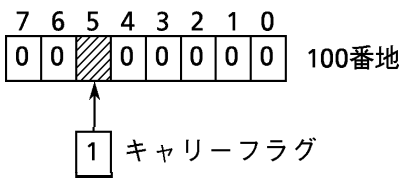
補 足 : ビットnumがAレジスタで指定された場合、Aレジスタの下位4ビットの値がビットnumとして使われます。オペランドがバイトのとき、ビットnumの下位4ビットの値が8~15の場合、オペランドの値は変化しません。

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 :

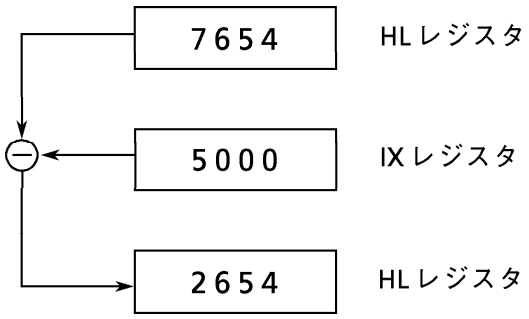
サイズ バイト	サイズ ワード	サイズ ロング	ニモニック		コード
○	○	○	SUB	R, r	11zz1r / 10100R
○	○	○	SUB	r, #	11zz1r / 11001010 / #<7:0> / #<15:8> / #<23:16> / #<31:24>
○	○	○	SUB	R, (mem)	1mzzmmmm / 10100R
○	○	○	SUB	(mem), R	1mzzmmmm / 10101R
○	○	×	SUB<W>	(mem), #	1m0zmmmm / 00110110 / #<7:0> / #<15:8>

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−4 …ミニマムモードの場合
または
XSP←XSP−6 …マキシマムモードの場合
② (XSP)←SR
③ (XSP+2)←16ビットPC …ミニマムモードの場合
または
(XSP+2)←32ビットPC …マキシマムモードの場合
④ PC←(ベクタベース番地+num×4)
⑤ INTNEST ←INTNEST+1
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。
SWI0~7は、レベル7の割り込みですが、本命令実行により、割り込みレベルマスクレジスタIFF2~0は変化しません。従って、例えばIFF2~0=1でSWI割り込み処理を実行中、レベル1の割り込みが要求されると、その割り込みを受付ネスティング状態となります。

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 ：

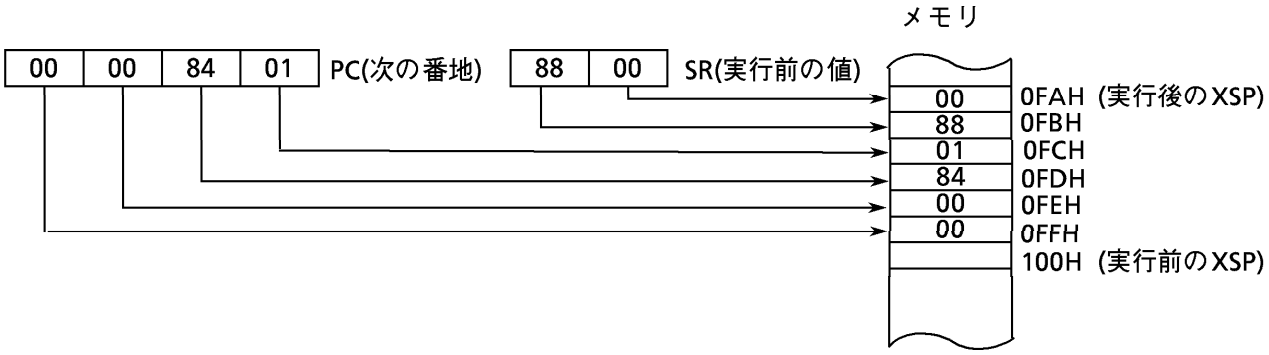
S	Z	H	V	N	C
–	–	–	–	–	–

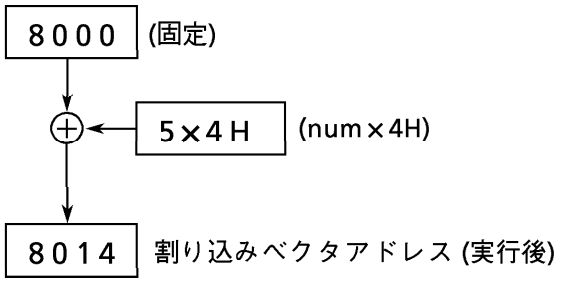
S ＝ 変化なし。
Z ＝ 変化なし。
H ＝ 変化なし。
V ＝ 変化なし。
N ＝ 変化なし。
C ＝ 変化なし。

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、8014H番地 (TMP93CS40の場合) に定義されている割り込みベクタをリードし、その番地へジャンプします。





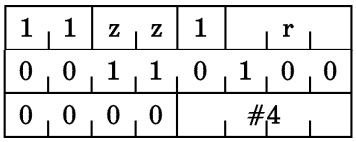
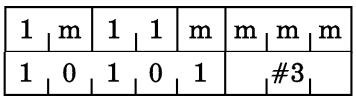
TSET num, dst

< 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 :

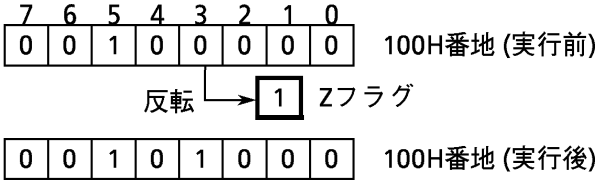
サイズ バイト	サイズ ワード	サイズ ロング	ニモニック		コード
○	○	×	TSET	#4, r	
○	×	×	TSET	#3, (mem)	

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			ニモニック		コード
バイト	ワード	ロング			
×	×	○	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	サイズ ワード	サイズ 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

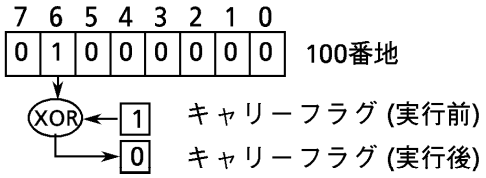
補足 : ビットnumがAレジスタで指定された場合、Aレジスタの下位4ビットの値がビットnumとして使われます。オペランドがバイトのとき、ビットnumの下位4ビットの値が8~15の場合、演算結果は不定になります。

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 :

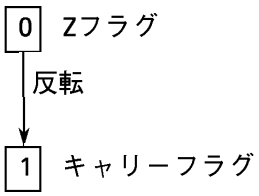
ニモニック	コード
ZCF	0 0 0 1 0 0 1 1

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。



付録 B. 命令一覧表

■ 本文中の記号の説明

1. サイズ欄

B	オペランドサイズがバイト (8ビット)
W	オペランドサイズがワード (16ビット)
L	オペランドサイズがロング (32ビット)

2. ニモニック欄

R	8/16/32ビットのカレントバンクレジスタを含む8本の汎用レジスタ 8ビットの場合: W, A, B, C, D, E, H, L 16ビットの場合: WA, BC, DE, HL, IX, IY, IZ, SP 32ビットの場合: XWA, XBC, XDE, XHL, XIX, XIY, XIZ, XSP
r	8/16/32ビットの汎用レジスタ
cr	8/16/32ビットのCPUの全コントロールレジスタ DMAS0~3, DMAD0~3, DMAC0~3, DMAM0~3, INTNEST
A	Aレジスタ(8ビット)
F	フラグレジスタ(8ビット)
F'	裏フラグレジスタ(8ビット)
SR	ステータスレジスタ(16ビット)
PC	プログラムカウンタ(ミニマムモード＝16ビット,マキシマムモード＝32ビット)
(mem)	8/16/32ビットのメモリ内のデータ
mem	実効アドレス値
<W>	オペランドサイズがワードのとき "W" の記述が必要であることを示します。
[ ]	カッコ内のオペランドの記述が、省略できることを示します。
#	8/16/32ビットの即値データ
#3	3 ビットの即値データ; 0~7 or1~8 ……短縮コード用
#4	4 ビットの即値データ; 0~15or1~16
d8	8 ビットのディスプレースメント; −80H~＋7FH
d16	16ビットのディスプレースメント; −8000H~＋7FFFH
cc	コンディションコード
(#8)	ダイレクトアドレッシング ; (00H) ~ (0FFH) …256バイト空間
(#16)	64KBエリアアドレッシング ; (0000H) ~ (0FFFFH)
$	その命令が置かれている先頭アドレス

3. コード欄

Z	オペランドサイズを表す指定コードです。 バイト (8ビット) =0 ワード (16ビット) =2 ロング (32ビット) =4
ZZ	オペランドサイズを表す指定コードです。 バイト (8ビット) =00H ワード (16ビット) =10H ロング (32ビット) =20H

4. フラグ欄 (SZHVNC)

−	フラグは変化しません。。
*	命令の実行によって、フラグは変化します。
0	フラグは “0” にクリアされます。
1	フラグは “1” にセットされます。
P	命令の実行によって、フラグは変化します。(パリティフラグとして働きます)
V	命令の実行によって、フラグは変化します。(オーバフローフラグとして働きます)
X	フラグには、不定値がセットされます。

5. 命令長欄

命令長を、バイト単位で示しています。

+#	イミディエートデータ長を加えてください。
+M	アドレッシングコード長を加えてください。
+#M	イミディエートデータ長+アドレッシングコード長を加えてください。

6. ステート欄

命令の実行処理時間を、ステート単位で、8ビット,16ビット,32ビット処理の場合を順に示しています。

1ステート = 100 ns @ 20 MHz発振時

■ 900/L命令一覧表 (1/10)

(1) 転送

命令群	サイズ	ニモニック	コード (16進)	機 能	SZHVNC	命令長	ステート
LD	BWL	LD R,r	C8+zz+r :88+R	R ← r	------	2	4. 4. 4
	BWL	LD r,R	C8+zz+r :98+R	r ← R	------	2	4. 4. 4
	BWL	LD r,#3	C8+zz+r :A8+#3	r ← #3	------	2	4. 4. 4
	BWL	LD R,#	20+zz+R :#	R ← #	------	1+#	2. 3. 5
	BWL	LD r,#	C8+zz+r :03:#	r ← #	------	2+#	4. 4. 6
	BWL	LD R,(mem)	80+zz+mem:20+R	R ← (mem)	------	2+M	4. 4. 6
	BWL	LD (mem),R	B0+mem :40+zz+R	(mem) ← R	------	2+M	4. 4. 6
	BW-	LD<W> (#8) ,#	08+z :#8:#	(#8) ← #	------	2+#	5. 6. -
	BW-	LD<W> (mem),#	B0+mem :00+z:#	(mem) ← #	------	2+M#	5. 6. -
	BW-	LD<W> (#16),(mem)	80+zz+mem:19:#16	(#16) ← (mem)	------	4+M	8. 8. -
	BW-	LD<W> (mem),(#16)	B0+mem :14+z:#16	(mem) ← (#16)	------	4+M	8. 8. -
PUSH	B--	PUSH F	18	(−XSP) ← F	------	1	3. -. -
	B--	PUSH A	14	(−XSP) ← A	------	1	3. -. -
	-WL	PUSH R	18+zz+R	(−XSP) ← R	------	1	-. 3. 5
	BWL	PUSH r	C8+zz+r :04	(−XSP) ← r	------	2	5. 5. 7
	BW-	PUSH<W> #	09+z :#	(−XSP) ← #	------	1+#	4. 5. -
	BW-	PUSH<W> (mem)	80+zz+mem:04	(−XSP) ← (mem)	------	2+M	7. 7. -
POP	B--	POP F	19	F ← (XSP+)	******	1	4. -. -
	B--	POP A	15	A ← (XSP+)	------	1	4. -. -
	-WL	POP R	38+zz+R	R ← (XSP+)	------	1	-. 4. 6
	BWL	POP r	C8+zz+r :05	r ← (XSP+)	------	2	6. 6. 8
	BW-	POP<W> (mem)	B0+mem :04+z	(mem) ← (XSP+)	------	2+M	6. 6. -
LDA	-WL	LDA R,mem	B0+mem :10+zz+R	R ← mem	------	2+M	-. 4. 4
LDAR	-WL	LDAR R,$+4+d16	F3:13:d16:20+zz+R	R ← PC+d16	------	5	-.11.11

(2) 交換

命令群	サイズ	ニモニック	コード (16進)	機 能	SZHVNC	命令長	ステート
EX	B--	EX F,F'	16	F ↔ F'	******	1	2. -. -
	BW-	EX R,r	C8+zz+r :B8+R	R ↔ r	------	2	5. 5. -
	BW-	EX (mem),R	80+zz+mem:30+R	(mem) ↔ R	------	2+M	6. 6. -
MIRR	-W-	MIRR r	D8+r :16	r<0:MSB>←r<MSB:0>	------	2	-. 4. -

■ 900/L命令一覧表 (2/10)

(3) ブロック転送/ブロックサーチ

命令群	サイズ	ニモニック	コード (16進)	機　能	SZHVNC	命令長	ステート
LDxx	BW-	LDI<W> [(XDE+),(XHL+)]	83+zz :10	(XDE+) ← (XHL+) BC ← BC-1	--0①0-	2	10.10. -
	BW-	LDI<W> (XIX+),(XIY+)	85+zz :10	(XIX+) ← (XIY+) BC ← BC-1	--0①0-	2	10.10. -
	BW-	LDIR<W> [(XDE+),(XHL+)]	83+zz :11	repeat (XDE+) ← (XHL+) BC ← BC-1 until BC=0	--000-	2	10.10. - (end) 14.14. - (repeat)
	BW-	LDIR<W> (XIX+),(XIY+)	85+zz :11	repeat (XIX+) ← (XIY+) BC ← BC-1 until BC=0	--000-	2	10.10. - (end) 14.14. - (repeat)
	BW-	LDD<W> [(XDE-),(XHL-)]	83+zz :12	(XDE-) ← (XHL-) BC ← BC-1	--0①0-	2	10.10. -
	BW-	LDD<W> (XIX-),(XIY-)	85+zz :12	(XIX-) ← (XIY-) BC ← BC-1	--0①0-	2	10.10. -
	BW-	LDDR<W> [(XDE-),(XHL-)]	83+zz :13	repeat (XDE-) ← (XHL-) BC ← BC-1 until BC=0	--000-	2	10.10. - (end) 14.14. - (repeat)
	BW-	LDDR<W> (XIX-),(XIY-)	85+zz :13	repeat (XIX-) ← (XIY-) BC ← BC-1 until BC=0	--000-	2	10.10. - (end) 14.14. - (repeat)
CPxx	BW-	CPI [A/WA,(R+)]	80+zz+R :14	A/WA - (R+) BC ← BC-1	*②*①1-	2	8. 8. -
	BW-	CPIR [A/WA,(R+)]	80+zz+R :15	repeat A/WA - (R+) BC ← BC-1 until A/WA=(R) or BC=0	*②*①1-	2	10.10. - (end) 14.14. - (repeat)
	BW-	CPD [A/WA,(R-)]	80+zz+R :16	A/WA - (R-) BC ← BC-1	*②*①1-	2	8. 8. -
	BW-	CPDR [A/WA,(R-)]	80+zz+R :17	repeat A/WA - (R-) BC ← BC-1 until A/WA=(R) or BC=0	*②*①1-	2	10.10. - (end) 14.14. - (repeat)

(注1) ①： 命令実行後BC＝0ならP/Vフラグは“0”に、それ以外は“1”にセットされます。
②： A/WA＝(R)ならZフラグは“1”に、それ以外は“0”にセットされます。

(注2) CPI, CPIR, CPD, CPDR命令でオペランドを省略したときは、“A, (XHL+/−)”を指定したことになります。

■ 900/L命令一覧表 (3/10)

(4) 算術演算

命令群	サイズ	ニモニック	コード (16進)	機 能	SZHVNC	命令長	ステート
ADD	BWL	ADD R,r	C8+zz+r :80+R	R ← R + r	***V0*	2	4. 4. 7
	BWL	ADD r,#	C8+zz+r :C8:#	r ← r + #	***V0*	2+#	4. 4. 7
	BWL	ADD R,(mem)	80+zz+mem:80+R	R ← R + (mem)	***V0*	2+M	4. 4. 6
	BWL	ADD (mem),R	80+zz+mem:88+R	(mem) ← (mem) + R	***V0*	2+M	6. 6.10
	BW-	ADD<W> (mem),#	80+zz+mem:38:#	(mem) ← (mem) + #	***V0*	2+M#	7. 8. -
ADC	BWL	ADC R,r	C8+zz+r :90+R	R ← R + r + CY	***V0*	2	4. 4. 7
	BWL	ADC r,#	C8+zz+r :C9:#	r ← r + # + CY	***V0*	2+#	4. 4. 7
	BWL	ADC R,(mem)	80+zz+mem:90+R	R ← R+(mem)+CY	***V0*	2+M	4. 4. 6
	BWL	ADC (mem),R	80+zz+mem:98+R	(mem) ←(mem)+R+CY	***V0*	2+M	6. 6.10
	BW-	ADC<W> (mem),#	80+zz+mem:39:#	(mem) ←(mem)+#+CY	***V0*	2+M#	7. 8. -
SUB	BWL	SUB R,r	C8+zz+r :A0+R	R ← R - r	***V1*	2	4. 4. 7
	BWL	SUB r,#	C8+zz+r :CA:#	r ← r - #	***V1*	2+#	4. 4. 7
	BWL	SUB R,(mem)	80+zz+mem:A0+R	R ← R - (mem)	***V1*	2+M	4. 4. 6
	BWL	SUB (mem),R	80+zz+mem:A8+R	(mem) ← (mem) - R	***V1*	2+M	6. 6.10
	BW-	SUB<W> (mem),#	80+zz+mem:3A:#	(mem) ← (mem) - #	***V1*	2+M#	7. 8. -
SBC	BWL	SBC R,r	C8+zz+r :B0+R	R ← R - r - CY	***V1*	2	4. 4. 7
	BWL	SBC r,#	C8+zz+r :CB:#	r ← r - # - CY	***V1*	2+#	4. 4. 7
	BWL	SBC R,(mem)	80+zz+mem:B0+R	R ←R-(mem)-CY	***V1*	2+M	4. 4. 6
	BWL	SBC (mem),R	80+zz+mem:B8+R	(mem) ←(mem)-R-CY	***V1*	2+M	6. 6.10
	BW-	SBC<W> (mem),#	80+zz+mem:3B:#	(mem) ←(mem)-#-CY	***V1*	2+M#	7. 8. -
CP	BWL	CP R,r	C8+zz+r :F0+R	R - r	***V1*	2	4. 4. 7
	BW-	CP r,#3	C8+zz+r :D8+#3	r - #3	***V1*	2	4. 4. -
	BWL	CP r,#	C8+zz+r :CF:#	r - #	***V1*	2+#	4. 4. 7
	BWL	CP R,(mem)	80+zz+mem:F0+R	R - (mem)	***V1*	2+M	4. 4. 6
	BWL	CP (mem),R	80+zz+mem:F8+R	(mem) - R	***V1*	2+M	6. 6. 6
	BW-	CP<W> (mem),#	80+zz+mem:3F:#	(mem) - #	***V1*	2+M#	6. 6. -
INC	B--	INC #3,r	C8+r :60+#3	r ← r + #3	***V0-	2	4. -. -
	-WL	INC #3,r	C8+zz+r :60+#3	r ← r + #3	------	2	-. 4. 4
	BW-	INC<W> #3,(mem)	80+zz+mem:60+#3	(mem) ← (mem) + #3	***V0-	2+M	6. 6. -
DEC	B--	DEC #3,r	C8+r :68+#3	r ← r - #3	***V1-	2	4. -. -
	-WL	DEC #3,r	C8+zz+r :68+#3	r ← r - #3	------	2	-. 4. 5
	BW-	DEC<W> #3,(mem)	80+zz+mem:68+#3	(mem) ← (mem) - #3	***V1-	2+M	6. 6. -
NEG	BW-	NEG r	C8+zz+r :07	r ← 0 - r	***V1*	2	5. 5. -
EXTZ	-WL	EXTZ r	C8+zz+r :12	r<high> ← 0	------	2	-. 4. 4
EXTS	-WL	EXTS r	C8+zz+r :13	r<high> ← r<low. MSB>	------	2	-. 5. 5
DAA	B--	DAA r	C8+r :10	加減算後の10進補正	***P-*	2	6. -. -
PAA	-WL	PAA r	C8+zz+r :14	if r<0>=1 then INC r	------	2	-. 4. 4

(注1) INC/DEC命令では、#3のコード値が0のときは、+8/−8として機能します。
(注2) TLCS-90の “ADD R,r” 命令 (ワード型) は、S/Z/Vフラグが変化しません。

■ 900/L命令一覧表(4/10)

命令群	サイズ	ニモニック	コード(16進)	機　能	SZHVNC	命令長	ステート

MUL	BW-
	BW-
	BW-
		MUL RR,r
		MUL rr,#
		MUL RR,(mem)
			C8+zz+r :40+R
			C8+zz+r :08:#
			80+zz+mem:40+R
				RR ← R×r
				rr ← r×#
				RR ← R×(mem)
					------ 	
					------ 	
					------ 	
						2
						2+#
						2+M
							18.26. -
							18.26. -
							18.26. -

MULS	BW-
	BW-
	BW-
		MULS RR,r
		MULS rr,#
		MULS RR,(mem)
			C8+zz+r :48+R
			C8+zz+r :09:#
			80+zz+mem:48+R
				RR ← R×r ;signed
				rr ← r×# ;signed
				RR ←R×(mem);signed
					------ 	
					------ 	
					------ 	
						2
						2+#
						2+M
							18.26. -
							18.26. -
							18.26. -

DIV	BW- BW- BW-	DIV RR,r DIV rr,# DIV RR,(mem)	C8+zz+r :50+R C8+zz+r :0A:# 80+zz+mem:50+R	R ← RR÷r r ← rr÷# R ← RR÷(mem)	---V-- ---V-- ---V--	2 2+# 2+M	22.30. - 22.30. - 22.30. -
DIVS	BW- BW- BW-	DIVS RR,r DIVS rr,# DIVS RR,(mem)	C8+zz+r :58+R C8+zz+r :0B:# 80+zz+mem:58+R	R ← RR÷r ;signed r ← rr÷# ;signed R ← RR÷(mem);signed	---V-- ---V-- ---V--	2 2+# 2+M	24.32. - 24.32. - 24.32. -
MULA	-W-	MULA rr	D8+r :19	符号付き積和演算 rr(32bit) ← rr(32bit)+(XDE)(16bit)×(XHL)(16bit) XHL ← XHL-2	**-V--	2	-.31. -
MINC	-W-	MINC1 #,r (#=2**n) (1<=n<=15)	D8+r :38:#-1	モジュロインクリメント ;+1 if (r mod #)=(#-1) then r←r-(#-1) else r←r+1	------	4	-. 8. -
	-W-	MINC2 #,r (#=2**n) (2<=n<=15)	D8+r :39:#-2	モジュロインクリメント ;+2 if (r mod #)=(#-2) then r←r-(#-2) else r←r+2	------	4	-. 8. -
	-W-	MINC4 #,r (#=2**n) (3<=n<=15)	D8+r :3A:#-4	モジュロインクリメント ;+4 if (r mod #)=(#-4) then r←r-(#-4) else r←r+4	------	4	-. 8. -
MDEC	-W-	MDEC1 #,r (#=2**n) (1<=n<=15)	D8+r :3C:#-1	モジュロデクリメント ;-1 if (r mod #)=0 then r←r+(#-1) else r←r-1	------	4	-. 7. -
	-W-	MDEC2 #,r (#=2**n) (2<=n<=15)	D8+r :3D:#-2	モジュロデクリメント ;-2 if (r mod #)=0 then r←r+(#-2) else r←r-2	------	4	-. 7. -
	-W-	MDEC4 #,r (#=2**n) (3<=n<=15)	D8+r :3E:#-4	モジュロデクリメント ;-4 if (r mod #)=0 then r←r+(#-4) else r←r-4	------	4	-. 7. -

(注) MUL, MULS, DIV, DIVS 命令のオペランド“RR”は、演算サイズの倍のレジスタを指定することを示しています。演算サイズがバイトのとき(8×8ビット, 16÷8ビット)は、ワードレジスタ(16ビット)指定し、演算サイズがワードのとき(16×16ビット, 32÷16ビット)は、ロングワードレジスタ(32ビット)を指定します。

■ 900/L命令一覧表 (5/10)

(5) 論理演算

命令群	サイズ	ニモニック	コード (16進)	機 能	SZHVNC	命令長	ステート
AND	BWL	AND R,r	C8+zz+r :C0+R	R ← R and r	**1P00	2	4. 4. 7
	BWL	AND r,#	C8+zz+r :CC:#	r ← r and #	**1P00	2+#	4. 4. 7
	BWL	AND R,(mem)	80+zz+mem:C0+R	R ← R and (mem)	**1P00	2+M	4. 4. 6
	BWL	AND (mem),R	80+zz+mem:C8+R	(mem) ← (mem) and R	**1P00	2+M	6. 6. 10
	BW-	AND<w> (mem),#	80+zz+mem:3C:#	(mem) ← (mem) and #	**1P00	2+M#	7. 8. -
OR	BWL	OR R,r	C8+zz+r :E0+R	R ← R or r	**0P00	2	4. 4. 7
	BWL	OR r,#	C8+zz+r :CE:#	r ← r or #	**0P00	2+#	4. 4. 7
	BWL	OR R,(mem)	80+zz+mem:E0+R	R ← R or (mem)	**0P00	2+M	4. 4. 6
	BWL	OR (mem),R	80+zz+mem:E8+R	(mem) ← (mem) or R	**0P00	2+M	6. 6. 10
	BW-	OR<W> (mem),#	80+zz+mem:3E:#	(mem) ← (mem) or #	**0P00	2+M#	7. 8. -
XOR	BWL	XOR R,r	C8+zz+r :D0+R	R ← R xor r	**0P00	2	4. 4. 7
	BWL	XOR r,#	C8+zz+r :CD:#	r ← r xor #	**0P00	2+#	4. 4. 7
	BWL	XOR R,(mem)	80+zz+mem:D0+R	R ← R xor (mem)	**0P00	2+M	4. 4. 6
	BWL	XOR (mem),R	80+zz+mem:D8+R	(mem) ← (mem) xor R	**0P00	2+M	6. 6. 10
	BW-	XOR<W> (mem),#	80+zz+mem:3D:#	(mem) ← (mem) xor #	**0P00	2+M#	7. 8. -
CPL	BW-	CPL r	C8+zz+r :06	r ← not r	--1-1-	2	4. 4. -

■ 900/L命令一覧表 (6/10)

(6) ビット操作

命令群	サイズ	ニモニック	コード (16進)	機 能	SZHVNC	命令長	ステート
LDCF	BW-	LDCF #4,r	C8+zz+r :23:#4	CY ← r<#4>	-----*	3	4.4.-
	BW-	LDCF A ,r	C8+zz+r :2B	CY ← r<A>	-----*	2	4.4.-
	B--	LDCF #3,(mem)	B0+men :98+#3	CY ← (mem)<#3>	-----*	2+M	8.-.-
	B--	LDCF A ,(mem)	B0+men :2B	CY ← (mem)<A>	-----*	2+M	8.-.-
STCF	BW-	STCF #4,r	C8+zz+r :24:#4	r<#4> ← CY	------	3	4.4.-
	BW-	STCF A ,r	C8+zz+r :2C	r<A> ← CY	------	2	4.4.-
	B--	STCF #3,(mem)	B0+mem :A0+#3	(mem)<#3> ← CY	------	2+M	8.-.-
	B--	STCF A ,(mem)	B0+mem :2C	(mem)<A> ← CY	------	2+M	8.-.-
ANDCF	BW-	ANDCF #4,r	C8+zz+r :20:#4	CY ← CY and r<#4>	-----*	3	4.4.-
	BW-	ANDCF A ,r	C8+zz+r :28	CY ← CY and r<A>	-----*	2	4.4.-
	B--	ANDCF #3,(mem)	B0+mem :80+#3	CY ← CY and (mem)<#3>	-----*	2+M	8.-.-
	B--	ANDCF A ,(mem)	B0+mem :28	CY ← CY and (mem)<A>	-----*	2+M	8.-.-
ORCF	BW-	ORCF #4,r	C8+zz+r :21:#4	CY ← CY or r<#4>	-----*	3	4.4.-
	BW-	ORCF A ,r	C8+zz+r :29	CY ← CY or r<A>	-----*	2	4.4.-
	B--	ORCF #3,(mem)	B0+mem :88+#3	CY ← CY or (mem)<#3>	-----*	2+M	8.-.-
	B--	ORCF A ,(mem)	B0+mem :29	CY ← CY or (mem)<A>	-----*	2+M	8.-.-
XORCF	BW-	XORCF #4,r	C8+zz+r :22:#4	CY ← CY xor r<#4>	-----*	3	4.4.-
	BW-	XORCF A ,r	C8+zz+r :2A	CY ← CY xor r<A>	-----*	2	4.4.-
	B--	XORCF #3,(mem)	B0+mem :90+#3	CY ← CY xor (mem)<#3>	-----*	2+M	8.-.-
	B--	XORCF A ,(mem)	B0+mem :2A	CY ← CY xor (mem)<A>	-----*	2+M	8.-.-
RCF	---	RCF	10	CY ← 0	--0-00	1	2
SCF	---	SCF	11	CY ← 1	--0-01	1	2
CCF	---	CCF	12	CY ← not CY	--X-0*	1	2
ZCF	---	ZCF	13	CY ← not "Z" フラグ	--X-0*	1	2
BIT	BW-	BIT #4,r	C8+zz+r :33:#4	Z ← not r<#4>	X*1X0-	3	4.4.-
	B--	BIT #3,(mem)	B0+mem :C8+#3	Z ← not (mem)<#3>	X*1X0-	2+M	8.-.-
RES	BW-	RES #4,r	C8+zz+r :30:#4	r<#4> ← 0	------	3	4.4.-
	B--	RES #3,(mem)	B0+mem :B0+#3	(mem)<#3> ← 0	------	2+M	8.-.-
SET	BW-	SET #4,r	C8+zz+r :31:#4	r<#4> ← 1	------	3	4.4.-
	B--	SET #3,(mem)	B0+mem :B8+#3	(mem)<#3> ← 1	------	2+M	8.-.-
CHG	BW-	CHG #4,r	C8+zz+r :32:#4	r<#4> ← not r<#4>	------	3	4.4.-
	B--	CHG #3,(mem)	B0+mem :C0+#3	(mem)<#3>←not (mem)<#3>	------	2+M	8.-.-
TSET	BW-	TSET #4,r	C8+zz+r :34:#4	Z←not r<#4> : r<#4>←1	X*1X0-	3	6.6.-
	B--	TSET #3,(mem)	B0+mem :A8+#3	Z ← not (mem)<#3> (mem)<#3> ← 1	X*1X0-	2+M	10.-.-
BS1	-W-	BS1F A,r	D8+r :0E	A ← 1 サーチ r ; Forward	---①--	2	-.4.-
	-W-	BS1B A,r	D8+r :0F	A ← 1 サーチ r ; Backward	---①--	2	-.4.-

(注) ① : サーチするビットが見つかったとき“0”にセットされ、それ以外のときは“1”にセットされAレジスタは不定値になります。

■ 900/L命令一覧表 (7/10)

(7) 特別演算、CPU制御

命令群	サイズ	ニモニック	コード (16進)	機 能	SZHVNC	命令長	ステート
NOP	---	NOP	00	no operation	------	1	2
MIN	---	MIN	04	ミニマムモードに移行 MAX←0	------	1	4
EI	---	EI [#3]	06 :#3	割り込み許可フラグの設定 IFF←#3	------	2	5
DI	---	DI	06 :07	割り込み禁止 IFF←7	------	2	5
PUSH	-W-	PUSH SR	02	(-XSP)←SR	------	1	-.4.-
POP	-W-	POP SR	03	SR←(XSP+)	******	1	-.6.-
SWI	---	SWI [#3]	F8+#3	ソフトウエア割り込み PUSH PC&SR JP 8000H+4H×#3	------	1	22
HALT	---	HALT	05	CPU停止	------	1	8
LDC	BWL BWL	LDC cr,r LDC r,cr	C8+zz+r :2E:cr C8+zz+r :2F:cr	cr ← r r ← cr	------ ------	3 3	8.8.8 8.8.8
LDX	B--	LDX (#8),#	F7:00:#8:00:#:00	(#8) ← #	------	6	9.-.-

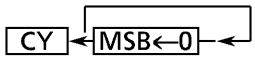
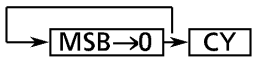
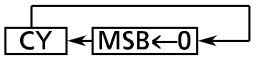
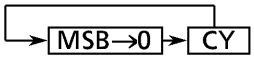
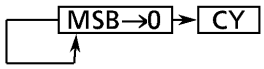
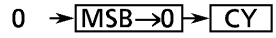
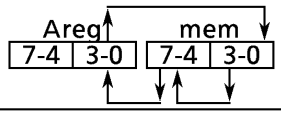
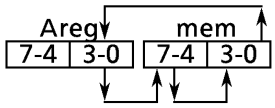
LINK	--L	LINK r,d16	E8+r :0C:d16	PUSH r
				LD r,XSP
				ADD XSP,d16
					------	4	-.-.10

UNLK	--L	UNLK r	E8+r :0D	LD XSP,r POP r	------	2	-.-.8
LDF	---	LDF #3	17 :#3	レジスタバンクの設定 RFP← #3 (リセット時"0")	------	2	2
INCF	---	INCF	0C	レジスタバンクの切り替え RFP← RFP + 1	------	1	2
DECF	---	DECF	0D	レジスタバンクの切り替え RFP← RFP − 1	------	1	2
SCC	BW-	SCC cc,r	C8+zz+r :70+cc	if cc then r ← 1 else r ← 0	------	2	6.-6.-

(注1) EI命令でオペランド#3の記述を省略すると“0”を指定したものとみなされます。
(注2) SWI命令でオペランド#3の記述を省略すると“7”を指定したものとみなされます。
(注3) SWI命令のステート数は、CPUがマキシマムモード時の値です。ミニマムモード時は−2してください。

■ 900/L命令一覧表 (8/10)

(8) ローテート、シフト

命令群	サイズ	ニモニック	コード (16進)	機 能	SZHVNC	命令長	ステート
RLC	BWL	RLC #4,r	C8+zz+r :E8:#4		**0P0*	3	6.6.8+2n
	BWL	RLC A,r	C8+zz+r :F8		**0P0*	2	6.6.8+2n
	BW-	RLC<W> (mem)	80+zz+mem:78		**0P0*	2+M	8.8.-
RRC	BWL	RRC #4,r	C8+zz+r :E9:#4		**0P0*	3	6.6.8+2n
	BWL	RRC A,r	C8+zz+r :F9		**0P0*	2	6.6.8+2n
	BW-	RRC<W> (mem)	80+zz+mem:79		**0P0*	2+M	8.8.-
RL	BWL	RL #4,r	C8+zz+r :EA:#4		**0P0*	3	6.6.8+2n
	BWL	RL A,r	C8+zz+r :FA		**0P0*	2	6.6.8+2n
	BW-	RL<W> (mem)	80+zz+mem:7A		**0P0*	2+M	8.8.-
RR	BWL	RR #4,r	C8+zz+r :EB:#4		**0P0*	3	6.6.8+2n
	BWL	RR A,r	C8+zz+r :FB		**0P0*	2	6.6.8+2n
	BW-	RR<W> (mem)	80+zz+mem:7B		**0P0*	2+M	8.8.-
SLA	BWL	SLA #4,r	C8+zz+r :EC:#4		**0P0*	3	6.6.8+2n
	BWL	SLA A,r	C8+zz+r :FC		**0P0*	2	6.6.8+2n
	BW-	SLA<W> (mem)	80+zz+mem:7C		**0P0*	2+M	8.8.-
SRA	BWL	SRA #4,r	C8+zz+r :ED:#4		**0P0*	3	6.6.8+2n
	BWL	SRA A,r	C8+zz+r :FD		**0P0*	2	6.6.8+2n
	BW-	SRA<W> (mem)	80+zz+mem:7D		**0P0*	2+M	8.8.-
SLL	BWL	SLL #4,r	C8+zz+r :EE:#4		**0P0*	3	6.6.8+2n
	BWL	SLL A,r	C8+zz+r :FE		**0P0*	2	6.6.8+2n
	BW-	SLL<W> (mem)	80+zz+mem:7E		**0P0*	2+M	8.8.-
SRL	BWL	SRL #4,r	C8+zz+r :EF:#4		**0P0*	3	6.6.8+2n
	BWL	SRL A,r	C8+zz+r :FF		**0P0*	2	6.6.8+2n
	BW-	SRL<W> (mem)	80+zz+mem:7F		**0P0*	2+M	8.8.-
RLD	B--	RLD [A,](mem)	80+mem :06		**0P0-	2+M	12.-.-
RRD	B--	RRD [A,](mem)	80+mem :07		**0P0-	2+M	12.-.-

(注1) #4/Aによるシフト回数の指定では、下位4ビットの値(0~15)が使われます。コード0は、16回シフトになります。

(注2) TLCS-90のRLCA/RRCA/RLA/RRA/SLAA/SRAA/SLLA/SRLA命令は、S/Z/Vフラグが変化しません。

■ 900/L命令一覧表 (9/10)

(9) ジャンプ、コール、リターン

命令群	サイズ	ニモニック	コード (16進)	機 能	SZHVNC	命令長	ステート
JP	--- --- --- --- ---	JP #16 JP #24 JR [cc,]$+2+d8 JRL [cc,]$+3+d16 JP [cc,]mem	1A :#16 1B :#24 60+cc :d8 70+cc :d16 B0+mem :D0+cc	PC ← #16 PC ← #24 if cc then PC ← PC+d8 if cc then PC ← PC+d16 if cc then PC ← mem	------ ------ ------ ------ ------	3 4 2 3 2+M	7 7 8/4 (T/F) 8/4 (T/F) 9/6 (T/F)
CALL	--- --- --- ---	CALL #16 CALL #24 CALR $+3+d16 CALL [cc,]mem	1C :#16 1D :#24 1E :d16 B0+mem :E0+cc	PUSH PC : JP #16 PUSH PC : JP #24 PUSH PC : JR $+3+d16 if cc then PUSH PC : JP mem	------ ------ ------ ------	3 4 3 2+M	14 14 14 14/6 (T/F)
DJNZ	BW-	DJNZ [r,]$+3/4+d8	C8+zz+r :1C:d8	r←r-1 (r省略でBレジスタ) if r≠0 then JR $+3+d8	------	3	11 (r≠0) 7 (r=0)
RET	--- --- --- ---	RET RET cc RETD d16 RETI	0E B0 :F0+cc 0F :d16 07	POP PC if cc then POP PC RET : ADD XSP,d16 POP SR&PC	------ ------ ------ ******	1 2 3 1	11 14/6 (T/F) 11 12

(注1) CALL/CALR/RET/RETD/RETI 命令のステート数は、CPUがマキシマムモード時の値です。ミニマムモード時は−2してください。

(注2) (T/F) はTrue/False時のステート数を表しています。

■ 900/L命令一覧表 (10/10)

(10) アドレッシングモード

分類	モード	ステート (追加分)
R	R	+0
r	r	+1
(mem)	(R)	+0
	(R + d8)	+2
	(#8)	+2
	(#16)	+2
	(#24)	+3
	(r)	+5
	(r + d16)	+5
	(r + r8)	+8
	(r + r16)	+8
	(− r)	+3
	(r +)	+3

(11) 割り込み

モード		動作	ステート
汎用割り込み処理		PUSH PC PUSH SR IFF ← 受け付けレベル + 1 INTNEST ← INTNEST + 1 JP (FFFF00H + ベクタ)	25 (MAXモード) 23 (MINモード)
μDMA	I/O to MEM	(DMADn +) ← (DMASn)	16. 16. −
	I/O to MEM	(DMADn −) ← (DMASn)	16. 16. −
	MEM to I/O	(DMADn) ← (DMASn +)	16. 16. −
	MEM to I/O	(DMADn) ← (DMASn −)	16. 16. −
	I/O to I/O	(DMADn) ← (DMASn)	16. 16. −
	DRAM Refresh	Dummy ← (DMASn +)	−. −. 14
	Counter	DMASn ← DMASn + 1	−. −. 11

(注) 割り込み動作の詳細は、各製品の「割り込み」の章をご覧ください。

付録C. 命令コードマップ (1/4)

1バイトオペコード命令

H/L	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NOP		PUSH SR	POP SR	MIN	HALT	EI n	RETI	LD (n) , n	PUSH n	LDW (n) , nn	PUSHW nn	INCF	DECF	RET	RETD dd
1	RCF	SCF	CCF	ZCF	PUSH A	POP A	EX F, F'	LDF n	PUSH F	POP F	JP nn	JP nnn	CALL nn	CALL nnn	CALR PC + dd	
2	LD R, n								PUSH RR							
3	LD RR, nn								PUSH XRR							
4	LD XRR, nnnn								POP RR							
5									POP XRR							
6	F	LT	LE	ULE	PE/OV	M/MI	Z	JR C	JR cc,PC + d (T)	GE	GT	UGT	PO/NOV	P/PL	NZ	NC
7								JRL cc,PC + dd ↑								
8	scr. B (XWA)	(XBC)	(XDE)	(XHL)	(XIX)	(XIY)	(XIZ)	(XSP)	scr. B (XWA + d)	(XBC + d)	(XDE + d)	(XHL + d)	(XIX + d)	(XIY + d)	(XIZ + d)	(XSP + d)
9	scr. W ↑								scr. W ↑							
A	scr. L ↑								scr. L ↑							
B	dst ↑								dst ↑							
C	scr. B (n)	(nn)	(nnn)	(mem)	(– xrr)	(xrr +)		reg. B r	reg. B W	A	B	C	D	E	H	L
D	scr. W ↑							reg. W rr	reg. W WA	BC	DE	HL	IX	IY	IZ	SP
E	scr. L ↑							reg. L xrr	reg. L XWA	XBC	XDE	XHL	XIX	XIY	XIZ	XSP
F	dst ↑							LDX (n), n	0	1	2	SWI 3	n 4	5	6	7

(注1) 空白部分のコードは、未定義命令です。
(注2) コード01Hにはダミーの命令が実装されています。使用しないでください。

付録C. 命令コードマップ (2/4)

1バイト目: reg

H/L	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0				LD r,#	PUSH r	POP r	CPL BW r	NEG BW r	MUL rr,#	MULS rr,#	DIV rr,#	DIVS BW rr,#	LINK L r, dd	UNLK L r	BS1F A, r	BS1B W A, r
1	DAA B r		EXTZ WL r	EXTS WL r	PAA WL r		MIRR W r			MULA W r			DJNZ BW r, d			
2	ANDCF #, r	ORCF #, r	XORCF #, r	LDCF #, r	STCF BW #, r				ANDCF A, r	ORCF A, r	XORCF A, r	LDCF A, r	STCF BW A, r		LDC cr, r	LDC r, cr
3	RES #, r	SET #, r	CHG #, r	BIT #, r	TSET BW #, r				MINC1	MINC2 #, r	MINC4 W		MDEC1	MDEC2 #, r	MDEC4 W	
4				MUL	R, r			BW				MULS	R, r			BW
5				DIV	R, r			BW				DIVS	R, r			BW
6	8	1	2	INC 3	#3, r 4	5	6	7	8	1	2	DEC 3	#3, r 4	5	6	7
7	F	LT	LE	ULE	PE/OV	M/MI	Z	SCC C	cc. r (T)	GE	GT	UGT	PO/NOV	P/PL	NZ	BW NC
8				ADD	R, r							LD	R, r			
9				ADC	R, r							LD	r, R			
A				SUB	R, r				0	1	2	LD 3	r, #3 4	5	6	7
B				SBC	R, r							EX	R, r			BW
C				AND	R, r				ADD r, #	ADC r, #	SUB r, #	SBC r, #	AND r, #	XOR r, #	OR r, #	CP r, #
D				XOR	R, r				0	1	2	CP 3	r, #3 4	5	6	BW 7
E				OR	R, r				RLC #, r	RRC #, r	RL #, r	RR #, r	SLA #, r	SRA #, r	SLL #, r	SRL #, r
F				CP	R, r				RLC A, r	RRC A, r	RL A, r	RR A, r	SLA A, r	SRA A, r	SLL A, r	SRL A, r

r : 1バイト目のコードで指定されるレジスタを示します (すべてのCPU内部レジスタが指定可能です)。
R : 2バイト目のコードで指定されるレジスタを示します (カレントレジスタ8本のみ指定可能です)。
B : オペランドサイズは、バイト型です。
W : オペランドサイズは、ワード型です。
L : オペランドサイズは、ロング型です。

(注) コード1AH, 1BH, 3BH, 3FHにはダミーの命令が実装されています。使用しないでください。

付録C. 命令コードマップ (3/4)

1バイト目:src (mem)

H/L	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0					PUSH BW (mem)		RLD A, (mem)	RRD B								
1	LDI	LDIR	LDD	LDDR BW	CPI	CPIR	CPD	CPDR BW		LD BW (nn),(m)						
2	W	A	B	LD C	R, (mem) D	E	H	L								
3				EX	(mem), R			BW	ADD	ADC	SUB	SBC (mem) ,#	AND	XOR	OR	CP BW
4				MUL	R, (mem)			BW				MULS	R, (mem)			BW
5				DIV	R, (mem)			BW				DIVS	R, (mem)			BW
6	8	1	2	INC 3	#3, (mem) 4	5	6	BW 7	8	1	2	DEC 3	#3, (mem) 4	5	6	BW 7
7									RLC	RRC	RL	RR (mem)	SLA	SRA	SLL	SRL BW
8				ADD	R, (mem)							ADD	(mem), R			
9				ADC	R, (mem)							ADC	(mem), R			
A				SUB	R, (mem)							SUB	(mem), R			
B				SBC	R, (mem)							SBC	(mem), R			
C				AND	R, (mem)							AND	(mem), R			
D				XOR	R, (mem)							XOR	(mem), R			
E				OR	R, (mem)							OR	(mem), R			
F				CP	R, (mem)							CP	(mem), R			

B : オペランドサイズは、バイト型です。
W : オペランドサイズは、ワード型です。

付録C. 命令コードマップ (4/4)

1バイト目: dst (mem)

H/L	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	LD B (m), #		LD W (m), #		POP B (mem)		POP W (mem)									
1					LD B (m),(nn)		LD W (m),(nn)									
2	LDA R, mem W								ANDCF A, (mem)	ORCF	XORCF	LDCF	STCF B			
3	LDA R, mem L															
4	LD (mem), R B															
5	LD (mem), R W															
6	LD (mem), R L															
7																
8	ANDCF #3, (mem) B								ORCF #3, (mem) B							
	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
9	XORCF #3, (mem) B								LDCF #3, (mem) B							
	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
A	STCF #3, (mem) B								TSET #3, (mem) B							
	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
B	RES #3, (mem) B								SET #3, (mem) B							
	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
C	CHG #3, (mem) B								BIT #3, (mem) B							
	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
D								JP cc, mem								
	F	LT	LE	ULE	PE/OV	M/MI	Z	C	(T)	GE	GT	UGT	PO/NOV	P/PL	NZ	NC
E	CALL cc, mem ↑															
F	RET cc (1バイト目のコードはB0H) ↑															

B : オペランドサイズは、バイト型です。
W : オペランドサイズは、ワード型です。
L : オペランドサイズは、ロング型です。

付録 D. TLCS-90との相違点

項目＼シリーズ	TLCS-90	TLCS-900/L
CPU アーキテクチャ	8ビットCPU	16ビットCPU
内蔵 ROM / 内蔵 RAM	8ビットデータバス	16ビットデータバス
内蔵 I/O	8ビットデータバス	8ビットデータバス
外部データバス	8ビットデータバス	8ビット/16ビットデータバス (混在可能)
プログラム空間 (MMU付を除く) データ 空間	64KB 16MB (バンク)	16MB (リニア) 16MB (リニア)
命令セット / 命令ニモニック	TLCS-90	TLCS-90 + α α = 16ビット乗除算命令, ビット操作命令の強化, 32ビット転送 / 演算命令, Cコンパイラ用命令, レジスタバンク操作命令 etc.
命令コード (オブジェクト コード)	TLCS-90 独自	TLCS-900 独自 (TLCS-90と違います。)
アドレッシングモード	TLCS-90	TLCS-90 + α α = (– Reg), (Reg +), (Reg + disp16), (Reg + Reg16), (nnn)
汎用レジスタ	TLCS-90	TLCS-90 + α α = 32ビット化 レジスタバンク化
フラグ (F)	S Z I H X V N C	S Z "0" H "0" V N C Iフラグは, SRレジスタのIFF2～0に拡張。 Xフラグは, 削除。
リセット	PC ← 0000H (SPは不変)	PC ← (ベクタベース番地) XSP ← 100H
内蔵 ROM アドレス 内蔵 RAM アドレス 内蔵 I/O アドレス ダイレクトアドレッシング 領域 (n)	0000H～ ～FFxxH FFxxH～FFFFH FF00H～FFFFH	不定 0080H～ 0000H～007FH 0000H～00FFH
割り込み 割り込みスタートアドレス セーブされるレジスタ マスクレジスタ マスクレベル	 0000H + 8×V PC と AF IFF 0～1	 (ベクタベース番地 + 4×V) PC と SR IFF2～0 0～7

項目＼シリーズ	TLCS-90	TLCS-900
命令		
① ADD R, r (ワード型)	S, Z, Vフラグは変化しません。 [16ビットのレジスタ間加算以外では、S, Z, Vフラグは変化します。]	S, Z, Vフラグは変化します。
② Aレジスタのシフト	RLCA RRCA RLA RRA SLAA SRAA SLLA SRLA 以上の命令では、S, Z, Vフラグは変化しません。 RLC A RRC A RL A RR A SLA A SRA A SLL A SRL A 以上の命令では、S, Z, Vフラグは変化します。	S, Z, Vフラグは変化します。

補足： 900/Lは、基本的にTLCS-90との互換性を考慮しつつ、TLCS-90のCPUを16ビット化したシリーズです。

合計6種類の直接対応する命令がないので、TLCS-90用に作られたプログラムを移植するときは、下記のような等価命令に置き換えが必要です。

TLCS-90にあり TLCS-900/Lにない命令	TLCS-900/Lでの等価命令
EXX	EX BC, BC' EX DE, DE' EX HL, HL'
EX AF, AF'	EX A, A' EX F, F'
PUSH AF	PUSH A PUSH F
POP AF	POP F POP A
INCX	(32ビット INC命令)
DECX	(32ビット DEC命令)

また、900/Lは、TLCS-90と同じ命令でも一部機能強化されている命令があり、オペランドでの指定項目が増えているものがあります。下記に、それらを示します。

TLCS-90	TLCS-900
INC reg INC mem	INC imm3, reg INC imm3, mem
DEC reg DEC mem	DEC imm3, reg DEC imm3, mem
RLC reg RRC reg RL reg RR reg SLA reg SRA reg SLL reg SRL reg	RLC imm, reg RRC imm, reg RL imm, reg RR imm, reg SLA imm, reg SRA imm, reg SLL imm, reg SRL imm, reg