#	オペレーション	オペランド	機械語 (16進数)	フラグ	挙動	
1	ADD	reg, CS[PC + disp16]	871r dddd	SVZC	reg ← reg + CS[CS[PC++] + PC]	
2	ADD	reg, CS[addr16]	861r aaaa	SVZC	reg ← reg + CS[CS[PC++]]	
3	ADD	reg, CS[src0]	84ur	SVZC	reg ← reg + CS[src0]	※src0 / tgt0 は偶数番レジスタのみ (以下同じ)
4	ADD	reg, DS[--src]	83sr	SVZC	reg ← reg + DS[--src]	
5	ADD	reg, DS[addr16]	862r aaaa	SVZC	reg ← reg + DS[CS[PC++]]	
6	ADD	reg, DS[src + disp16]	85sr dddd	SVZC	reg ← reg + DS[CS[PC++] + src]	
7	ADD	reg, DS[src++]	82sr	SVZC	reg ← reg + DS[src++]	
8	ADD	reg, DS[src]	81sr	SVZC	reg ← reg + DS[src]	
9	ADD	reg, ES[addr16]	863r aaaa	SVZC	reg ← reg + ES[CS[PC++]]	
10	ADD	reg, ES[src1]	84vr	SVZC	reg ← reg + ES[src1]	※src1 / tgt1 は奇数番レジスタのみ (以下同じ)
11	ADD	reg, PC + disp16	870r dddd	SVZC	reg ← reg + (CS[PC++] + PC)	
12	ADD	reg, imm16	860r nnnn	SVZC	reg ← reg + CS[PC++]	
13	ADD	reg, src	80sr	SVZC	reg ← reg + src	
14	ADDC	reg, CS[PC + disp16]	8F1r dddd	SVZC	reg ← reg + CF + CS[CS[PC++] + PC]	
15	ADDC	reg, CS[addr16]	8E1r aaaa	SVZC	reg ← reg + CF + CS[CS[PC++]]	
16	ADDC	reg, CS[src0]	8Cur	SVZC	reg ← reg + CF + CS[src0]	
17	ADDC	reg, DS[--src]	8Bsr	SVZC	reg ← reg + CF + DS[--src]	
18	ADDC	reg, DS[addr16]	8E2r aaaa	SVZC	reg ← reg + CF + DS[CS[PC++]]	
19	ADDC	reg, DS[src + disp16]	8Dsr dddd	SVZC	reg ← reg + CF + DS[CS[PC++] + src]	
20	ADDC	reg, DS[src++]	8Asr	SVZC	reg ← reg + CF + DS[src++]	
21	ADDC	reg, DS[src]	89sr	SVZC	reg ← reg + CF + DS[src]	
22	ADDC	reg, ES[addr16]	8E3r aaaa	SVZC	reg ← reg + CF + ES[CS[PC++]]	
23	ADDC	reg, ES[src1]	8Cvr	SVZC	reg ← reg + CF + ES[src1]	
24	ADDC	reg, PC + disp16	8F0r dddd	SVZC	reg ← reg + CF + (CS[PC++] + PC)	
25	ADDC	reg, imm16	8E0r nnnn	SVZC	reg ← reg + CF + CS[PC++]	
26	ADDC	reg, src	88sr	SVZC	reg ← reg + CF + src	
27	ADDQ	reg, -imm4m	02nr	SVZC	※SUBQ の別名 reg ← reg – SignExt (imm4m) ※imm4m の範囲は -1 〜-16 ※ただし埋め込まれるのは ~imm4m した値 (0x0〜0xF)	
28	ADDQ	reg, -imm4p	0Anr	SVZC	※SUBQ の別名 reg ← reg – ZeroExt (imm4p) ※imm4p の範囲は 0 〜+15	
29	AND	reg, CS[PC + disp16]	C71r dddd	S0Z0	reg ← reg & CS[CS[PC++] + PC]	
30	AND	reg, CS[addr16]	C61r aaaa	S0Z0	reg ← reg & CS[CS[PC++]]	
31	AND	reg, CS[src0]	C4ur	S0Z0	reg ← reg & CS[src0]	
32	AND	reg, DS[--src]	C3sr	S0Z0	reg ← reg & DS[--src]	
33	AND	reg, DS[addr16]	C62r aaaa	S0Z0	reg ← reg & DS[CS[PC++]]	
34	AND	reg, DS[src + disp16]	C5sr dddd	S0Z0	reg ← reg & DS[CS[PC++] + src]	
35	AND	reg, DS[src++]	C2sr	S0Z0	reg ← reg & DS[src++]	
36	AND	reg, DS[src]	C1sr	S0Z0	reg ← reg & DS[src]	
37	AND	reg, ES[addr16]	C63r aaaa	S0Z0	reg ← reg & ES[CS[PC++]]	
38	AND	reg, ES[src1]	C4vr	S0Z0	reg ← reg & ES[src1]	
39	AND	reg, imm16	C60r nnnn	S0Z0	reg ← reg & CS[PC++]	
40	AND	reg, src	C0sr	S0Z0	reg ← reg & src	

#	オペレーション	オペランド	機械語 (16進数)	フラグ	挙動
41	ANDN	reg, CS[PC + disp16]	CF1r dddd	S0Z0	reg ← reg & ~CS[CS[PC++] + PC]
42	ANDN	reg, CS[addr16]	CE1r aaaa	S0Z0	reg ← reg & ~CS[CS[PC++]]
43	ANDN	reg, CS[src0]	CCur	S0Z0	reg ← reg & ~CS[src0]
44	ANDN	reg, DS[--src]	CBsr	S0Z0	reg ← reg & ~DS[--src]
45	ANDN	reg, DS[addr16]	CE2r aaaa	S0Z0	reg ← reg & ~DS[CS[PC++]]
46	ANDN	reg, DS[src + disp16]	CDsr dddd	S0Z0	reg ← reg & ~DS[CS[PC++] + src]
47	ANDN	reg, DS[src++]	CAsr	S0Z0	reg ← reg & ~DS[src++]
48	ANDN	reg, DS[src]	C9sr	S0Z0	reg ← reg & ~DS[src]
49	ANDN	reg, ES[addr16]	CE3r aaaa	S0Z0	reg ← reg & ~ES[CS[PC++]]
50	ANDN	reg, ES[src1]	CCvr	S0Z0	reg ← reg & ~ES[src1]
51	ANDN	reg, imm16	CE0r nnnn	S0Z0	reg ← reg & ~CS[PC++]
52	ANDN	reg, src	C8sr	S0Z0	reg ← reg & ~src
53	ASC2NUM	reg, src	0150 DCsr	S0Z0	reg ← ASC2NUM (src) ※ASCII文字を36進数化:'0'～'9', 'A'～'Z', 'a'～'z' 以外の時は 0xFFFF になる
54	ASL / LSL	CS[PC + disp16]	671C dddd	SVZC	wk ← CS[PC++] + PC ; CS[wk] ← ArithShiftLeft (CS[wk], 1)
55	ASL / LSL	CS[addr16]	661C aaaa	SVZC	wk ← CS[PC++] ; CS[wk] ← ArithShiftLeft (CS[wk], 1)
56	ASL / LSL	CS[tgt0]	64uC	SVZC	wk ← tgt0 ; CS[wk] ← ArithShiftLeft (CS[wk], 1)
57	ASL / LSL	DS[--tgt]	63tC	SVZC	wk ← --tgt ; DS[wk] ← ArithShiftLeft (DS[wk], 1)
58	ASL / LSL	DS[addr16]	662C aaaa	SVZC	wk ← CS[PC++] ; DS[wk] ← ArithShiftLeft (DS[wk], 1)
59	ASL / LSL	DS[tgt + disp16]	65tC dddd	SVZC	wk ← CS[PC++] + tgt ; DS[wk] ← ArithShiftLeft (DS[wk], 1)
60	ASL / LSL	DS[tgt++]	62tC	SVZC	wk ← tgt++ ; DS[wk] ← ArithShiftLeft (DS[wk], 1)
61	ASL / LSL	DS[tgt]	61tC	SVZC	wk ← tgt ; DS[wk] ← ArithShiftLeft (DS[wk], 1)
62	ASL / LSL	ES[addr16]	663C aaaa	SVZC	wk ← CS[PC++] ; ES[wk] ← ArithShiftLeft (ES[wk], 1)
63	ASL / LSL	ES[tgt1]	64vC	SVZC	wk ← tgt1 ; ES[wk] ← ArithShiftLeft (ES[wk], 1)
64	ASL / LSL	tgt	60tC	SVZC	tgt ← ArithShiftLeft (tgt, 1)
65	ASLM / LSLM	reg, src, imm4	0156 Cnsr	S0Z0	reg ← ArithShiftLeft (src, imm4) ※VFは常に0、CFも常に0
66	ASLM / LSLM	reg, src, qty	01Dq C0sr	S0Z0	reg ← ArithShiftLeft (src, LOW4(qty)) ※VFは常に0、CFも常に0
67	ASR	CS[PC + disp16]	671A dddd	S0ZC	wk ← CS[PC++] + PC ; CS[wk] ← ArithShiftRight (CS[wk], 1)
68	ASR	CS[addr16]	661A aaaa	S0ZC	wk ← CS[PC++] ; CS[wk] ← ArithShiftRight (CS[wk], 1)
69	ASR	CS[tgt0]	64uA	S0ZC	wk ← tgt0 ; CS[wk] ← ArithShiftRight (CS[wk], 1)
70	ASR	DS[--tgt]	63tA	S0ZC	wk ← --tgt ; DS[wk] ← ArithShiftRight (DS[wk], 1)
71	ASR	DS[addr16]	662A aaaa	S0ZC	wk ← CS[PC++] ; DS[wk] ← ArithShiftRight (DS[wk], 1)
72	ASR	DS[tgt + disp16]	65tA dddd	S0ZC	wk ← CS[PC++] + tgt ; DS[wk] ← ArithShiftRight (DS[wk], 1)
73	ASR	DS[tgt++]	62tA	S0ZC	wk ← tgt++ ; DS[wk] ← ArithShiftRight (DS[wk], 1)
74	ASR	DS[tgt]	61tA	S0ZC	wk ← tgt ; DS[wk] ← ArithShiftRight (DS[wk], 1)
75	ASR	ES[addr16]	663A aaaa	S0ZC	wk ← CS[PC++] ; ES[wk] ← ArithShiftRight (ES[wk], 1)
76	ASR	ES[tgt1]	64vA	S0ZC	wk ← tgt1 ; ES[wk] ← ArithShiftRight (ES[wk], 1)
77	ASR	tgt	60tA	S0ZC	tgt ← ArithShiftRight (tgt, 1)
78	ASRM	reg, src, imm4	0156 Ansr	S0Z0	reg ← ArithShiftRight (src, imm4) ※VFは常に0、CFも常に0
79	ASRM	reg, src, qty	01Dq A0sr	S0Z0	reg ← ArithShiftRight (src, LOW4(qty)) ※VFは常に0、CFも常に0
80	BAND	CF, CS[PC+disp16].bpimm	7F1p dddd 3p7p	?0?C	wk ← CS[PC++] + PC ; CF ← CF & <CS[wk].bpimm>

#	オペレーション	オペランド	機械語 (16進数)	フラグ	挙動
81	BAND	CF, CS[addr16].bpimm	7E1p aaaa 3p7p	?0?C	wk ← CS[PC++] ; CF ← CF & <CS[wk].bpimm>
82	BAND	CF, CS[src0].bpimm	7Cup 3p7p	?0?C	wk ← src0 ; CF ← CF & <CS[wk].bpimm>
83	BAND	CF, DS[--src].bpimm	7Bsp 3p7p	?0?C	wk ← --src ; CF ← CF & <DS[wk].bpimm>
84	BAND	CF, DS[addr16].bpimm	7E2p aaaa 3p7p	?0?C	wk ← CS[PC++] ; CF ← CF & <DS[wk].bpimm>
85	BAND	CF, DS[src++].bpimm	7Asp 3p7p	?0?C	wk ← src++ ; CF ← CF & <DS[wk].bpimm>
86	BAND	CF, DS[src+disp16].bpimm	7Dsp dddd 3p7p	?0?C	wk ← CS[PC++] + src ; CF ← CF & <DS[wk].bpimm>
87	BAND	CF, DS[src].bpimm	79sp 3p7p	?0?C	wk ← src ; CF ← CF & <DS[wk].bpimm>
88	BAND	CF, ES[addr16].bpimm	7E3p aaaa 3p7p	?0?C	wk ← CS[PC++] ; CF ← CF & <ES[wk].bpimm>
89	BAND	CF, ES[src1].bpimm	7Cvp 3p7p	?0?C	wk ← src1 ; CF ← CF & <ES[wk].bpimm>
90	BAND	CF, src.bpimm	78sp 3p7p	?0?C	CF ← CF & <src.bpimm>
91	BAND	CS[PC+disp16].bpimm, CF	771p dddd 3p7p	----	wk ← CS[PC++] + PC ; <CS[wk].bpimm> ← <CS[wk].bpimm> & CF
92	BAND	CS[addr16].bpimm, CF	761p aaaa 3p7p	----	wk ← CS[PC++] ; <CS[wk].bpimm> ← <CS[wk].bpimm> & CF
93	BAND	CS[tgt0].bpimm, CF	74up 3p7p	----	wk ← tgt0 ; <CS[wk].bpimm> ← <CS[wk].bpimm> & CF
94	BAND	DS[--tgt].bpimm, CF	73tp 3p7p	----	wk ← --tgt ; <DS[wk].bpimm> ← <DS[wk].bpimm> & CF
95	BAND	DS[addr16].bpimm, CF	762p aaaa 3p7p	----	wk ← CS[PC++] ; <DS[wk].bpimm> ← <DS[wk].bpimm> & CF
96	BAND	DS[tgt++].bpimm, CF	72tp 3p7p	----	wk ← tgt++ ; <DS[wk].bpimm> ← <DS[wk].bpimm> & CF
97	BAND	DS[tgt+disp16].bpimm, CF	75tp dddd 3p7p	----	wk ← CS[PC++] + tgt ; <DS[wk].bpimm> ← <DS[wk].bpimm> & CF
98	BAND	DS[tgt].bpimm, CF	71tp 3p7p	----	wk ← tgt ; <DS[wk].bpimm> ← <DS[wk].bpimm> & CF
99	BAND	ES[addr16].bpimm, CF	763p aaaa 3p7p	----	wk ← CS[PC++] ; <ES[wk].bpimm> ← <ES[wk].bpimm> & CF
100	BAND	ES[tgt1].bpimm, CF	74vp 3p7p	----	wk ← tgt1 ; <ES[wk].bpimm> ← <ES[wk].bpimm> & CF
101	BAND	tgt.bpimm, CF	70tp 3p7p	----	<tgt.bpimm> ← <tgt.bpimm> & CF
102	BANDN	CF, CS[PC+disp16].bpimm	7F1p dddd Bp7p	?0?C	wk ← CS[PC++] + PC ; CF ← CF & ~<CS[wk].bpimm>
103	BANDN	CF, CS[addr16].bpimm	7E1p aaaa Bp7p	?0?C	wk ← CS[PC++] ; CF ← CF & ~<CS[wk].bpimm>
104	BANDN	CF, CS[src0].bpimm	7Cup Bp7p	?0?C	wk ← src0 ; CF ← CF & ~<CS[wk].bpimm>
105	BANDN	CF, DS[--src].bpimm	7Bsp Bp7p	?0?C	wk ← --src ; CF ← CF & ~<DS[wk].bpimm>
106	BANDN	CF, DS[addr16].bpimm	7E2p aaaa Bp7p	?0?C	wk ← CS[PC++] ; CF ← CF & ~<DS[wk].bpimm>
107	BANDN	CF, DS[src++].bpimm	7Asp Bp7p	?0?C	wk ← src++ ; CF ← CF & ~<DS[wk].bpimm>
108	BANDN	CF, DS[src+disp16].bpimm	7Dsp dddd Bp7p	?0?C	wk ← CS[PC++] + src ; CF ← CF & ~<DS[wk].bpimm>
109	BANDN	CF, DS[src].bpimm	79sp Bp7p	?0?C	wk ← src ; CF ← CF & ~<DS[wk].bpimm>
110	BANDN	CF, ES[addr16].bpimm	7E3p aaaa Bp7p	?0?C	wk ← CS[PC++] ; CF ← CF & ~<ES[wk].bpimm>
111	BANDN	CF, ES[src1].bpimm	7Cvp Bp7p	?0?C	wk ← src1 ; CF ← CF & ~<ES[wk].bpimm>
112	BANDN	CF, src.bpimm	78sp Bp7p	?0?C	CF ← CF & ~<src.bpimm>
113	BANDN	CS[PC+disp16].bpimm, CF	771p dddd 7p3p	----	wk ← CS[PC++] + PC ; <CS[wk].bpimm> ← <CS[wk].bpimm> & ~CF
114	BANDN	CS[addr16].bpimm, CF	761p aaaa 7p3p	----	wk ← CS[PC++] ; <CS[wk].bpimm> ← <CS[wk].bpimm> & ~CF
115	BANDN	CS[tgt0].bpimm, CF	74up 7p3p	----	wk ← tgt0 ; <CS[wk].bpimm> ← <CS[wk].bpimm> & ~CF
116	BANDN	DS[--tgt].bpimm, CF	73tp 7p3p	----	wk ← --tgt ; <DS[wk].bpimm> ← <DS[wk].bpimm> & ~CF
117	BANDN	DS[addr16].bpimm, CF	762p aaaa 7p3p	----	wk ← CS[PC++] ; <DS[wk].bpimm> ← <DS[wk].bpimm> & ~CF
118	BANDN	DS[tgt++].bpimm, CF	72tp 7p3p	----	wk ← tgt++ ; <DS[wk].bpimm> ← <DS[wk].bpimm> & ~CF
119	BANDN	DS[tgt+disp16].bpimm, CF	75tp dddd 7p3p	----	wk ← CS[PC++] + tgt ; <DS[wk].bpimm> ← <DS[wk].bpimm> & ~CF
120	BANDN	DS[tgt].bpimm, CF	71tp 7p3p	----	wk ← tgt ; <DS[wk].bpimm> ← <DS[wk].bpimm> & ~CF
121	BANDN	ES[addr16].bpimm, CF	763p aaaa 7p3p	----	wk ← CS[PC++] ; <ES[wk].bpimm> ← <ES[wk].bpimm> & ~CF

#	オペレーション	オペランド		機械語 (16進数)	フラグ	挙動
122	BANDN	ES[tgt1].bpimm,	CF	74vp 7p3p	----	wk ← tgt1 ; <ES[wk].bpimm> ← <ES[wk].bpimm> & ~CF
123	BANDN	tgt.bpimm,	CF	70tp 7p3p	----	<tgt.bpimm> ← <tgt.bpimm> & ~CF
124	BCD2NUM0A	reg, src		0156 D0sr	S0Z0	reg ← BCD2NUM0 (src).octA ※BCD値0000～9999 を符号無し16bit値に変換する
125	BCD2NUM1A	reg, src		0156 D1sr	S0Z0	reg ← BCD2NUM1 (src).octA ※BCD値0000_0000～9999_0000 を符号無し32bit値に変換する
126	BCD2NUM1B	reg, src		0156 D2sr	S0Z0	reg ← BCD2NUM1 (src).octB ※BCD値0000_0000～9999_0000 を符号無し32bit値に変換する
127	BCD2NUM2A	reg, src		0156 D3sr	S0Z0	reg ← BCD2NUM2 (src).octA ※BCD値0000_0000_0000～9999_0000_0000 を符号無し48bit値に変換する
128	BCD2NUM2B	reg, src		0156 D4sr	S0Z0	reg ← BCD2NUM2 (src).octB ※BCD値0000_0000_0000～9999_0000_0000 を符号無し48bit値に変換する
129	BCD2NUM2C	reg, src		0156 D5sr	S0Z0	reg ← BCD2NUM2 (src).octC ※BCD値0000_0000_0000～9999_0000_0000 を符号無し48bit値に変換する
130	BCDADD	reg, src		015E 01sr	00ZC	BCD(二進化十進数) : reg ← reg + src
131	BCDADDC	reg, src		5Esr	00ZC	BCD(二進化十進数) : reg ← reg + src + CF
132	BCDADJ	reg, src		5Dsr	----	reg ← BCDAdj (reg) ※regを4桁のBCDと見なし、それぞれの桁が5以上の時は3を加算する
133	BCDDEC	reg		015F 000r	00ZB	BCD(二進化十進数) : reg ← reg – 1
134	BCDINC	reg		015E 000r	00ZC	BCD(二進化十進数) : reg ← reg + 1
135	BCDSUB	reg, src		015F 01sr	00ZB	BCD(二進化十進数) : reg ← reg – src
136	BCDSUBB	reg, src		5Fsr	00ZB	BCD(二進化十進数) : reg ← reg – src – BF
137	BCLR	CS[PC + disp16].bpimm		771p dddd 7p7p	----	wk ← CS[PC++] + PC ; <CS[wk].bpimm> ← 0
138	BCLR	CS[addr16].bpimm		761p aaaa 7p7p	----	wk ← CS[PC++] ; <CS[wk].bpimm> ← 0
139	BCLR	CS[tgt0].bpimm		74up 7p7p	----	wk ← tgt0 ; <CS[wk].bpimm> ← 0
140	BCLR	DS[--tgt].bpimm		73tp 7p7p	----	wk ← --tgt ; <DS[wk].bpimm> ← 0
141	BCLR	DS[addr16].bpimm		762p aaaa 7p7p	----	wk ← CS[PC++] ; <DS[wk].bpimm> ← 0
142	BCLR	DS[tgt + disp16].bpimm		75tp dddd 7p7p	----	wk ← CS[PC++] + tgt ; <DS[wk].bpimm> ← 0
143	BCLR	DS[tgt++].bpimm		72tp 7p7p	----	wk ← tgt++ ; <DS[wk].bpimm> ← 0
144	BCLR	DS[tgt].bpimm		71tp 7p7p	----	wk ← tgt ; <DS[wk].bpimm> ← 0
145	BCLR	ES[addr16].bpimm		763p aaaa 7p7p	----	wk ← CS[PC++] ; <ES[wk].bpimm> ← 0
146	BCLR	ES[tgt1].bpimm		74vp 7p7p	----	wk ← tgt1 ; <ES[wk].bpimm> ← 0
147	BCLR	SF		0173	0---	SF ← 0
148	BCLR	VF		0172	-0--	VF ← 0
149	BCLR	ZF		0171	--0-	ZF ← 0
150	BCLR	CF		0170	---0	CF ← 0
151	BCLR	BF		01F0	---1	BF ← 0 すなわち CF ← 1 ※BSET CF の別名 ※TD16RM1は JC ≠ JB 方式
152	BCLR	tgt.bpimm		55tp	----	<tgt.bpimm> ← 0
153	BCLR	tgt.bpimm		70tp 7p7p	----	<tgt.bpimm> ← 0 ※機械語が短い方を使う事
154	BITREV	reg, src		0155 DBsr	S0Z0	reg ← BITREV (src) ※16bit 全体でbit 位置を反転させる
155	BITREV4	reg, src		0156 DAsr	S0Z0	reg ← BITREV4 (src) ※ 4bit 単位でbit 位置を反転させる
156	BITREV8	reg, src		0156 DBsr	S0Z0	reg ← BITREV8 (src) ※ 8bit 単位でbit 位置を反転させる
157	BMOV	CF, CS[PC+disp16].bpimm		7F1p dddd 3p3p	?0?C	wk ← CS[PC++] + PC ; CF ← <CS[wk].bpimm>
158	BMOV	CF, CS[addr16].bpimm		7E1p aaaa 3p3p	?0?C	wk ← CS[PC++] ; CF ← <CS[wk].bpimm>

#	オペレーション	オペランド	機械語 (16進数)	フラグ	挙動	
159	BMOV	CF, CS[src0].bpimm	7Cup 3p3p	?0?C	wk ← src0	; CF ← <CS[wk].bpimm>
160	BMOV	CF, DS[--src].bpimm	7Bsp 3p3p	?0?C	wk ← --src	; CF ← <DS[wk].bpimm>
161	BMOV	CF, DS[addr16].bpimm	7E2p aaaa 3p3p	?0?C	wk ← CS[PC++]	; CF ← <DS[wk].bpimm>
162	BMOV	CF, DS[src++].bpimm	7Asp 3p3p	?0?C	wk ← src++	; CF ← <DS[wk].bpimm>
163	BMOV	CF, DS[src+disp16].bpimm	7Dsp dddd 3p3p	?0?C	wk ← CS[PC++] + src	; CF ← <DS[wk].bpimm>
164	BMOV	CF, DS[src].bpimm	79sp 3p3p	?0?C	wk ← src	; CF ← <DS[wk].bpimm>
165	BMOV	CF, ES[addr16].bpimm	7E3p aaaa 3p3p	?0?C	wk ← CS[PC++]	; CF ← <ES[wk].bpimm>
166	BMOV	CF, ES[src1].bpimm	7Cvp 3p3p	?0?C	wk ← src1	; CF ← <ES[wk].bpimm>
167	BMOV	CF, src.bpimm	78sp 3p3p	?0?C	CF ← <src.bpimm>	※機械語が短い方を使う事
168	BMOV	CF, src.bpimm	5Csp	?0?C	CF ← <src.bpimm>	
169	BMOV	CS[PC+disp16].bpimm, CF	771p dddd Fp7p	----	wk ← CS[PC++] + PC	; <CS[wk].bpimm> ← CF
170	BMOV	CS[addr16].bpimm, CF	761p aaaa Fp7p	----	wk ← CS[PC++]	; <CS[wk].bpimm> ← CF
171	BMOV	CS[tgt0].bpimm, CF	74up Fp7p	----	wk ← tgt0	; <CS[wk].bpimm> ← CF
172	BMOV	DS[--tgt].bpimm, CF	73tp Fp7p	----	wk ← --tgt	; <DS[wk].bpimm> ← CF
173	BMOV	DS[addr16].bpimm, CF	762p aaaa Fp7p	----	wk ← CS[PC++]	; <DS[wk].bpimm> ← CF
174	BMOV	DS[tgt++].bpimm, CF	72tp Fp7p	----	wk ← tgt++	; <DS[wk].bpimm> ← CF
175	BMOV	DS[tgt+disp16].bpimm, CF	75tp dddd Fp7p	----	wk ← CS[PC++] + tgt	; <DS[wk].bpimm> ← CF
176	BMOV	DS[tgt].bpimm, CF	71tp Fp7p	----	wk ← tgt	; <DS[wk].bpimm> ← CF
177	BMOV	ES[addr16].bpimm, CF	763p aaaa Fp7p	----	wk ← CS[PC++]	; <ES[wk].bpimm> ← CF
178	BMOV	ES[tgt1].bpimm, CF	74vp Fp7p	----	wk ← tgt1	; <ES[wk].bpimm> ← CF
179	BMOV	tgt.bpimm, CF	70tp Fp7p	----	<tgt.bpimm> ← CF	※機械語が短い方を使う事
180	BMOV	tgt.bpimm, CF	54tp	----	<tgt.bpimm> ← CF	
181	BMOVN	CF, CS[PC+disp16].bpimm	7F1p dddd BpBp	?0?C	wk ← CS[PC++] + PC	; CF ← ~<CS[wk].bpimm>
182	BMOVN	CF, CS[addr16].bpimm	7E1p aaaa BpBp	?0?C	wk ← CS[PC++]	; CF ← ~<CS[wk].bpimm>
183	BMOVN	CF, CS[src0].bpimm	7Cup BpBp	?0?C	wk ← src0	; CF ← ~<CS[wk].bpimm>
184	BMOVN	CF, DS[--src].bpimm	7Bsp BpBp	?0?C	wk ← --src	; CF ← ~<DS[wk].bpimm>
185	BMOVN	CF, DS[addr16].bpimm	7E2p aaaa BpBp	?0?C	wk ← CS[PC++]	; CF ← ~<DS[wk].bpimm>
186	BMOVN	CF, DS[src++].bpimm	7Asp BpBp	?0?C	wk ← src++	; CF ← ~<DS[wk].bpimm>
187	BMOVN	CF, DS[src+disp16].bpimm	7Dsp dddd BpBp	?0?C	wk ← CS[PC++] + src	; CF ← ~<DS[wk].bpimm>
188	BMOVN	CF, DS[src].bpimm	79sp BpBp	?0?C	wk ← src	; CF ← ~<DS[wk].bpimm>
189	BMOVN	CF, ES[addr16].bpimm	7E3p aaaa BpBp	?0?C	wk ← CS[PC++]	; CF ← ~<ES[wk].bpimm>
190	BMOVN	CF, ES[src1].bpimm	7Cvp BpBp	?0?C	wk ← src1	; CF ← ~<ES[wk].bpimm>
191	BMOVN	CF, src.bpimm	78sp BpBp	?0?C	CF ← ~<src.bpimm>	
192	BMOVN	CS[PC+disp16].bpimm, CF	771p dddd 7pFp	----	wk ← CS[PC++] + PC	; <CS[wk].bpimm> ← ~CF
193	BMOVN	CS[addr16].bpimm, CF	761p aaaa 7pFp	----	wk ← CS[PC++]	; <CS[wk].bpimm> ← ~CF
194	BMOVN	CS[tgt0].bpimm, CF	74up 7pFp	----	wk ← tgt0	; <CS[wk].bpimm> ← ~CF
195	BMOVN	DS[--tgt].bpimm, CF	73tp 7pFp	----	wk ← --tgt	; <DS[wk].bpimm> ← ~CF
196	BMOVN	DS[addr16].bpimm, CF	762p aaaa 7pFp	----	wk ← CS[PC++]	; <DS[wk].bpimm> ← ~CF
197	BMOVN	DS[tgt++].bpimm, CF	72tp 7pFp	----	wk ← tgt++	; <DS[wk].bpimm> ← ~CF
198	BMOVN	DS[tgt+disp16].bpimm, CF	75tp dddd 7pFp	----	wk ← CS[PC++] + tgt	; <DS[wk].bpimm> ← ~CF
199	BMOVN	DS[tgt].bpimm, CF	71tp 7pFp	----	wk ← tgt	; <DS[wk].bpimm> ← ~CF

#	オペレーション	オペランド	機械語 (16進数)	フラグ	挙動
200	BMOVN	ES[addr16].bpimm, CF	763p aaaa 7pFp	----	wk ← CS[PC++] ; <ES[wk].bpimm> ← ~CF
201	BMOVN	ES[tgt1].bpimm, CF	74vp 7pFp	----	wk ← tgt1 ; <ES[wk].bpimm> ← ~CF
202	BMOVN	tgt.bpimm, CF	70tp 7pFp	----	<tgt.bpimm> ← ~CF
203	BNOT	CS[PC + disp16].bpimm	771p dddd BpBp	----	wk ← CS[PC++] + PC ; <CS[wk].bpimm> ← ~<CS[wk].bpimm>
204	BNOT	CS[addr16].bpimm	761p aaaa BpBp	----	wk ← CS[PC++] ; <CS[wk].bpimm> ← ~<CS[wk].bpimm>
205	BNOT	CS[tgt0].bpimm	74up BpBp	----	wk ← tgt0 ; <CS[wk].bpimm> ← ~<CS[wk].bpimm>
206	BNOT	DS[--tgt].bpimm	73tp BpBp	----	wk ← --tgt ; <DS[wk].bpimm> ← ~<DS[wk].bpimm>
207	BNOT	DS[addr16].bpimm	762p aaaa BpBp	----	wk ← CS[PC++] ; <DS[wk].bpimm> ← ~<DS[wk].bpimm>
208	BNOT	DS[tgt + disp16].bpimm	75tp dddd BpBp	----	wk ← CS[PC++] + tgt ; <DS[wk].bpimm> ← ~<DS[wk].bpimm>
209	BNOT	DS[tgt++].bpimm	72tp BpBp	----	wk ← tgt++ ; <DS[wk].bpimm> ← ~<DS[wk].bpimm>
210	BNOT	DS[tgt].bpimm	71tp BpBp	----	wk ← tgt ; <DS[wk].bpimm> ← ~<DS[wk].bpimm>
211	BNOT	ES[addr16].bpimm	763p aaaa BpBp	----	wk ← CS[PC++] ; <ES[wk].bpimm> ← ~<ES[wk].bpimm>
212	BNOT	ES[tgt1].bpimm	74vp BpBp	----	wk ← tgt1 ; <ES[wk].bpimm> ← ~<ES[wk].bpimm>
213	BNOT	SF	01B3	*---	SF ← ~SF
214	BNOT	VF	01B2	-*--	VF ← ~VF
215	BNOT	ZF	01B1	--*-	ZF ← ~ZF
216	BNOT	CF	01B0	---*	CF ← ~CF
217	BNOT	BF	01B0	---*	BF ← ~BF ※BNOT CF の別名
218	BNOT	tgt.bpimm	56tp	----	<tgt.bpimm> ← ~<tgt.bpimm>
219	BNOT	tgt.bpimm	70tp BpBp	----	<tgt.bpimm> ← ~<tgt.bpimm> ※機械語が短い方を使う事
220	BOR	CF, CS[PC+disp16].bpimm	7F1p dddd Fp3p	?0?C	wk ← CS[PC++] + PC ; CF ← CF | <CS[wk].bpimm>
221	BOR	CF, CS[addr16].bpimm	7E1p aaaa Fp3p	?0?C	wk ← CS[PC++] ; CF ← CF | <CS[wk].bpimm>
222	BOR	CF, CS[src0].bpimm	7Cup Fp3p	?0?C	wk ← src0 ; CF ← CF | <CS[wk].bpimm>
223	BOR	CF, DS[--src].bpimm	7Bsp Fp3p	?0?C	wk ← --src ; CF ← CF | <DS[wk].bpimm>
224	BOR	CF, DS[addr16].bpimm	7E2p aaaa Fp3p	?0?C	wk ← CS[PC++] ; CF ← CF | <DS[wk].bpimm>
225	BOR	CF, DS[src++].bpimm	7Asp Fp3p	?0?C	wk ← src++ ; CF ← CF | <DS[wk].bpimm>
226	BOR	CF, DS[src+disp16].bpimm	7Dsp dddd Fp3p	?0?C	wk ← CS[PC++] + src ; CF ← CF | <DS[wk].bpimm>
227	BOR	CF, DS[src].bpimm	79sp Fp3p	?0?C	wk ← src ; CF ← CF | <DS[wk].bpimm>
228	BOR	CF, ES[addr16].bpimm	7E3p aaaa Fp3p	?0?C	wk ← CS[PC++] ; CF ← CF | <ES[wk].bpimm>
229	BOR	CF, ES[src1].bpimm	7Cvp Fp3p	?0?C	wk ← src1 ; CF ← CF | <ES[wk].bpimm>
230	BOR	CF, src.bpimm	78sp Fp3p	?0?C	CF ← CF | <src.bpimm>
231	BOR	CS[PC+disp16].bpimm, CF	771p dddd Fp3p	----	wk ← CS[PC++] + PC ; <CS[wk].bpimm> ← <CS[wk].bpimm> | CF
232	BOR	CS[addr16].bpimm, CF	761p aaaa Fp3p	----	wk ← CS[PC++] ; <CS[wk].bpimm> ← <CS[wk].bpimm> | CF
233	BOR	CS[tgt0].bpimm, CF	74up Fp3p	----	wk ← tgt0 ; <CS[wk].bpimm> ← <CS[wk].bpimm> | CF
234	BOR	DS[--tgt].bpimm, CF	73tp Fp3p	----	wk ← --tgt ; <DS[wk].bpimm> ← <DS[wk].bpimm> | CF
235	BOR	DS[addr16].bpimm, CF	762p aaaa Fp3p	----	wk ← CS[PC++] ; <DS[wk].bpimm> ← <DS[wk].bpimm> | CF
236	BOR	DS[tgt++].bpimm, CF	72tp Fp3p	----	wk ← tgt++ ; <DS[wk].bpimm> ← <DS[wk].bpimm> | CF
237	BOR	DS[tgt+disp16].bpimm, CF	75tp dddd Fp3p	----	wk ← CS[PC++] + tgt ; <DS[wk].bpimm> ← <DS[wk].bpimm> | CF
238	BOR	DS[tgt].bpimm, CF	71tp Fp3p	----	wk ← tgt ; <DS[wk].bpimm> ← <DS[wk].bpimm> | CF
239	BOR	ES[addr16].bpimm, CF	763p aaaa Fp3p	----	wk ← CS[PC++] ; <ES[wk].bpimm> ← <ES[wk].bpimm> | CF
240	BOR	ES[tgt1].bpimm, CF	74vp Fp3p	----	wk ← tgt1 ; <ES[wk].bpimm> ← <ES[wk].bpimm> | CF

#	オペレーション	オペランド	機械語 (16進数)	フラグ	挙動
241	BOR	tgt.bpimm, CF	70tp Fp3p	----	<tgt.bpimm> ← <tgt.bpimm> | CF
242	BORN	CF, CS[PC+disp16].bpimm	7F1p dddd FpBp	?0?C	wk ← CS[PC++] + PC ; CF ← CF | ~<CS[wk].bpimm>
243	BORN	CF, CS[addr16].bpimm	7E1p aaaa FpBp	?0?C	wk ← CS[PC++] ; CF ← CF | ~<CS[wk].bpimm>
244	BORN	CF, CS[src0].bpimm	7Cup FpBp	?0?C	wk ← src0 ; CF ← CF | ~<CS[wk].bpimm>
245	BORN	CF, DS[--src].bpimm	7Bsp FpBp	?0?C	wk ← --src ; CF ← CF | ~<DS[wk].bpimm>
246	BORN	CF, DS[addr16].bpimm	7E2p aaaa FpBp	?0?C	wk ← CS[PC++] ; CF ← CF | ~<DS[wk].bpimm>
247	BORN	CF, DS[src++].bpimm	7Asp FpBp	?0?C	wk ← src++ ; CF ← CF | ~<DS[wk].bpimm>
248	BORN	CF, DS[src+disp16].bpimm	7Dsp dddd FpBp	?0?C	wk ← CS[PC++] + src ; CF ← CF | ~<DS[wk].bpimm>
249	BORN	CF, DS[src].bpimm	79sp FpBp	?0?C	wk ← src ; CF ← CF | ~<DS[wk].bpimm>
250	BORN	CF, ES[addr16].bpimm	7E3p aaaa FpBp	?0?C	wk ← CS[PC++] ; CF ← CF | ~<ES[wk].bpimm>
251	BORN	CF, ES[src1].bpimm	7Cvp FpBp	?0?C	wk ← src1 ; CF ← CF | ~<ES[wk].bpimm>
252	BORN	CF, src.bpimm	78sp FpBp	?0?C	CF ← CF | ~<src.bpimm>
253	BORN	CS[PC+disp16].bpimm, CF	771p dddd 3pFp	----	wk ← CS[PC++] + PC ; <CS[wk].bpimm> ← <CS[wk].bpimm> | ~CF
254	BORN	CS[addr16].bpimm, CF	761p aaaa 3pFp	----	wk ← CS[PC++] ; <CS[wk].bpimm> ← <CS[wk].bpimm> | ~CF
255	BORN	CS[tgt0].bpimm, CF	74up 3pFp	----	wk ← tgt0 ; <CS[wk].bpimm> ← <CS[wk].bpimm> | ~CF
256	BORN	DS[--tgt].bpimm, CF	73tp 3pFp	----	wk ← --tgt ; <DS[wk].bpimm> ← <DS[wk].bpimm> | ~CF
257	BORN	DS[addr16].bpimm, CF	762p aaaa 3pFp	----	wk ← CS[PC++] ; <DS[wk].bpimm> ← <DS[wk].bpimm> | ~CF
258	BORN	DS[tgt++].bpimm, CF	72tp 3pFp	----	wk ← tgt++ ; <DS[wk].bpimm> ← <DS[wk].bpimm> | ~CF
259	BORN	DS[tgt+disp16].bpimm, CF	75tp dddd 3pFp	----	wk ← CS[PC++] + tgt ; <DS[wk].bpimm> ← <DS[wk].bpimm> | ~CF
260	BORN	DS[tgt].bpimm, CF	71tp 3pFp	----	wk ← tgt ; <DS[wk].bpimm> ← <DS[wk].bpimm> | ~CF
261	BORN	ES[addr16].bpimm, CF	763p aaaa 3pFp	----	wk ← CS[PC++] ; <ES[wk].bpimm> ← <ES[wk].bpimm> | ~CF
262	BORN	ES[tgt1].bpimm, CF	74vp 3pFp	----	wk ← tgt1 ; <ES[wk].bpimm> ← <ES[wk].bpimm> | ~CF
263	BORN	tgt.bpimm, CF	70tp 3pFp	----	<tgt.bpimm> ← <tgt.bpimm> | ~CF
264	BSET	CS[PC + disp16].bpimm	771p dddd FpFp	----	wk ← CS[PC++] + PC ; <CS[wk].bpimm> ← 1
265	BSET	CS[addr16].bpimm	761p aaaa FpFp	----	wk ← CS[PC++] ; <CS[wk].bpimm> ← 1
266	BSET	CS[tgt0].bpimm	74up FpFp	----	wk ← tgt0 ; <CS[wk].bpimm> ← 1
267	BSET	DS[--tgt].bpimm	73tp FpFp	----	wk ← --tgt ; <DS[wk].bpimm> ← 1
268	BSET	DS[addr16].bpimm	762p aaaa FpFp	----	wk ← CS[PC++] ; <DS[wk].bpimm> ← 1
269	BSET	DS[tgt + disp16].bpimm	75tp dddd FpFp	----	wk ← CS[PC++] + tgt ; <DS[wk].bpimm> ← 1
270	BSET	DS[tgt++].bpimm	72tp FpFp	----	wk ← tgt++ ; <DS[wk].bpimm> ← 1
271	BSET	DS[tgt].bpimm	71tp FpFp	----	wk ← tgt ; <DS[wk].bpimm> ← 1
272	BSET	ES[addr16].bpimm	763p aaaa FpFp	----	wk ← CS[PC++] ; <ES[wk].bpimm> ← 1
273	BSET	ES[tgt1].bpimm	74vp FpFp	----	wk ← tgt1 ; <ES[wk].bpimm> ← 1
274	BSET	SF	01F3	1---	SF ← 1
275	BSET	VF	01F2	-1--	VF ← 1
276	BSET	ZF	01F1	--1-	ZF ← 1
277	BSET	CF	01F0	---1	CF ← 1
278	BSET	BF	0170	---0	BF ← 1 すなわち CF ← 0 ※BCLR CF の別名 ※TD16RM1は JC ≠ JB 方式
279	BSET	tgt.bpimm	57tp	----	<tgt.bpimm> ← 1
280	BSET	tgt.bpimm	70tp FpFp	----	<tgt.bpimm> ← 1 ※機械語が短い方を使う事
281	BXOR	CF, CS[PC+disp16].bpimm	7F1p dddd Bp3p	?0?C	wk ← CS[PC++] + PC ; CF ← CF ^ <CS[wk].bpimm>

#	オペレーション	オペランド	機械語 (16進数)	フラグ	挙動
282	BXOR	CF, CS[addr16].bpimm	7E1p aaaa Bp3p	?0?C	wk ← CS[PC++] ; CF ← CF ^ <CS[wk].bpimm>
283	BXOR	CF, CS[src0].bpimm	7Cup Bp3p	?0?C	wk ← src0 ; CF ← CF ^ <CS[wk].bpimm>
284	BXOR	CF, DS[--src].bpimm	7Bsp Bp3p	?0?C	wk ← --src ; CF ← CF ^ <DS[wk].bpimm>
285	BXOR	CF, DS[addr16].bpimm	7E2p aaaa Bp3p	?0?C	wk ← CS[PC++] ; CF ← CF ^ <DS[wk].bpimm>
286	BXOR	CF, DS[src++].bpimm	7Asp Bp3p	?0?C	wk ← src++ ; CF ← CF ^ <DS[wk].bpimm>
287	BXOR	CF, DS[src+disp16].bpimm	7Dsp dddd Bp3p	?0?C	wk ← CS[PC++] + src ; CF ← CF ^ <DS[wk].bpimm>
288	BXOR	CF, DS[src].bpimm	79sp Bp3p	?0?C	wk ← src ; CF ← CF ^ <DS[wk].bpimm>
289	BXOR	CF, ES[addr16].bpimm	7E3p aaaa Bp3p	?0?C	wk ← CS[PC++] ; CF ← CF ^ <ES[wk].bpimm>
290	BXOR	CF, ES[src1].bpimm	7Cvp Bp3p	?0?C	wk ← src1 ; CF ← CF ^ <ES[wk].bpimm>
291	BXOR	CF, src.bpimm	78sp Bp3p	?0?C	CF ← CF ^ <src.bpimm>
292	BXOR	CS[PC+disp16].bpimm, CF	771p dddd Bp3p	----	wk ← CS[PC++] + PC ; <CS[wk].bpimm> ← <CS[wk].bpimm> ^ CF
293	BXOR	CS[addr16].bpimm, CF	761p aaaa Bp3p	----	wk ← CS[PC++] ; <CS[wk].bpimm> ← <CS[wk].bpimm> ^ CF
294	BXOR	CS[tgt0].bpimm, CF	74up Bp3p	----	wk ← tgt0 ; <CS[wk].bpimm> ← <CS[wk].bpimm> ^ CF
295	BXOR	DS[--tgt].bpimm, CF	73tp Bp3p	----	wk ← --tgt ; <DS[wk].bpimm> ← <DS[wk].bpimm> ^ CF
296	BXOR	DS[addr16].bpimm, CF	762p aaaa Bp3p	----	wk ← CS[PC++] ; <DS[wk].bpimm> ← <DS[wk].bpimm> ^ CF
297	BXOR	DS[tgt++].bpimm, CF	72tp Bp3p	----	wk ← tgt++ ; <DS[wk].bpimm> ← <DS[wk].bpimm> ^ CF
298	BXOR	DS[tgt+disp16].bpimm, CF	75tp dddd Bp3p	----	wk ← CS[PC++] + tgt ; <DS[wk].bpimm> ← <DS[wk].bpimm> ^ CF
299	BXOR	DS[tgt].bpimm, CF	71tp Bp3p	----	wk ← tgt ; <DS[wk].bpimm> ← <DS[wk].bpimm> ^ CF
300	BXOR	ES[addr16].bpimm, CF	763p aaaa Bp3p	----	wk ← CS[PC++] ; <ES[wk].bpimm> ← <ES[wk].bpimm> ^ CF
301	BXOR	ES[tgt1].bpimm, CF	74vp Bp3p	----	wk ← tgt1 ; <ES[wk].bpimm> ← <ES[wk].bpimm> ^ CF
302	BXOR	tgt.bpimm, CF	70tp Bp3p	----	<tgt.bpimm> ← <tgt.bpimm> ^ CF
303	BXORN	CF, CS[PC+disp16].bpimm	7F1p dddd 3pBp	?0?C	wk ← CS[PC++] + PC ; CF ← CF ^ ~<CS[wk].bpimm>
304	BXORN	CF, CS[addr16].bpimm	7E1p aaaa 3pBp	?0?C	wk ← CS[PC++] ; CF ← CF ^ ~<CS[wk].bpimm>
305	BXORN	CF, CS[src0].bpimm	7Cup 3pBp	?0?C	wk ← src0 ; CF ← CF ^ ~<CS[wk].bpimm>
306	BXORN	CF, DS[--src].bpimm	7Bsp 3pBp	?0?C	wk ← --src ; CF ← CF ^ ~<DS[wk].bpimm>
307	BXORN	CF, DS[addr16].bpimm	7E2p aaaa 3pBp	?0?C	wk ← CS[PC++] ; CF ← CF ^ ~<DS[wk].bpimm>
308	BXORN	CF, DS[src++].bpimm	7Asp 3pBp	?0?C	wk ← src++ ; CF ← CF ^ ~<DS[wk].bpimm>
309	BXORN	CF, DS[src+disp16].bpimm	7Dsp dddd 3pBp	?0?C	wk ← CS[PC++] + src ; CF ← CF ^ ~<DS[wk].bpimm>
310	BXORN	CF, DS[src].bpimm	79sp 3pBp	?0?C	wk ← src ; CF ← CF ^ ~<DS[wk].bpimm>
311	BXORN	CF, ES[addr16].bpimm	7E3p aaaa 3pBp	?0?C	wk ← CS[PC++] ; CF ← CF ^ ~<ES[wk].bpimm>
312	BXORN	CF, ES[src1].bpimm	7Cvp 3pBp	?0?C	wk ← src1 ; CF ← CF ^ ~<ES[wk].bpimm>
313	BXORN	CF, src.bpimm	78sp 3pBp	?0?C	CF ← CF ^ ~<src.bpimm>
314	BXORN	CS[PC+disp16].bpimm, CF	771p dddd 3pBp	----	wk ← CS[PC++] + PC ; <CS[wk].bpimm> ← <CS[wk].bpimm> ^ ~CF
315	BXORN	CS[addr16].bpimm, CF	761p aaaa 3pBp	----	wk ← CS[PC++] ; <CS[wk].bpimm> ← <CS[wk].bpimm> ^ ~CF
316	BXORN	CS[tgt0].bpimm, CF	74up 3pBp	----	wk ← tgt0 ; <CS[wk].bpimm> ← <CS[wk].bpimm> ^ ~CF
317	BXORN	DS[--tgt].bpimm, CF	73tp 3pBp	----	wk ← --tgt ; <DS[wk].bpimm> ← <DS[wk].bpimm> ^ ~CF
318	BXORN	DS[addr16].bpimm, CF	762p aaaa 3pBp	----	wk ← CS[PC++] ; <DS[wk].bpimm> ← <DS[wk].bpimm> ^ ~CF
319	BXORN	DS[tgt++].bpimm, CF	72tp 3pBp	----	wk ← tgt++ ; <DS[wk].bpimm> ← <DS[wk].bpimm> ^ ~CF
320	BXORN	DS[tgt+disp16].bpimm, CF	75tp dddd 3pBp	----	wk ← CS[PC++] + tgt ; <DS[wk].bpimm> ← <DS[wk].bpimm> ^ ~CF
321	BXORN	DS[tgt].bpimm, CF	71tp 3pBp	----	wk ← tgt ; <DS[wk].bpimm> ← <DS[wk].bpimm> ^ ~CF
322	BXORN	ES[addr16].bpimm, CF	763p aaaa 3pBp	----	wk ← CS[PC++] ; <ES[wk].bpimm> ← <ES[wk].bpimm> ^ ~CF

#	オペレーション	オペランド		機械語 (16進数)	フラグ	挙動
323	BXORN	ES[tgt1].bpimm,	CF	74vp 3pBp	----	wk ← tgt1 ; <ES[wk].bpimm> ← <ES[wk].bpimm> ^ ~CF
324	BXORN	tgt.bpimm,	CF	70tp 3pBp	----	<tgt.bpimm> ← <tgt.bpimm> ^ ~CF
325	CADD	reg, CS[PC + disp16]		BF1r dddd	SVZC	if (CF == 1) then reg ← reg + CS[CS[PC++] + PC]
					----	※CF == 0 の時は何もしない(フラグも無変化)
326	CADD	reg, CS[addr16]		BE1r aaaa	SVZC	if (CF == 1) then reg ← reg + CS[CS[PC++]]
					----	※CF == 0 の時は何もしない(フラグも無変化)
327	CADD	reg, CS[src0]		BCur	SVZC	if (CF == 1) then reg ← reg + CS[src0]
					----	※CF == 0 の時は何もしない(フラグも無変化)
328	CADD	reg, DS[--src]		BBsr	SVZC	if (CF == 1) then reg ← reg + DS[--src]
					----	※CF == 0 の時は何もしない(フラグも無変化)
329	CADD	reg, DS[addr16]		BE2r aaaa	SVZC	if (CF == 1) then reg ← reg + DS[CS[PC++]]
					----	※CF == 0 の時は何もしない(フラグも無変化)
330	CADD	reg, DS[src + disp16]		BDsr dddd	SVZC	if (CF == 1) then reg ← reg + DS[CS[PC++] + src]
					----	※CF == 0 の時は何もしない(フラグも無変化)
331	CADD	reg, DS[src++]		BAsr	SVZC	if (CF == 1) then reg ← reg + DS[src++]
					----	※CF == 0 の時は何もしない(フラグも無変化)
332	CADD	reg, DS[src]		B9sr	SVZC	if (CF == 1) then reg ← reg + DS[src]
					----	※CF == 0 の時は何もしない(フラグも無変化)
333	CADD	reg, ES[addr16]		BE3r aaaa	SVZC	if (CF == 1) then reg ← reg + ES[CS[PC++]]
					----	※CF == 0 の時は何もしない(フラグも無変化)
334	CADD	reg, ES[src1]		BCvr	SVZC	if (CF == 1) then reg ← reg + ES[src1]
					----	※CF == 0 の時は何もしない(フラグも無変化)
335	CADD	reg, PC + disp16		BF0r dddd	SVZC	if (CF == 1) then reg ← reg + (CS[PC++] + PC)
					----	※CF == 0 の時は何もしない(フラグも無変化)
336	CADD	reg, imm16		BE0r nnnn	SVZC	if (CF == 1) then reg ← reg + CS[PC++]
					----	※CF == 0 の時は何もしない(フラグも無変化)
337	CADD	reg, src		B8sr	SVZC	if (CF == 1) then reg ← reg + src
					----	※CF == 0 の時は何もしない(フラグも無変化)
338	CALL	CS[PC + disp16]		6F10 dddd	----	wk ← CS[CS[PC++] + PC] ; DS[--SP] ← PC ; PC ← wk
339	CALL	CS[addr16]		6E10 aaaa	----	wk ← CS[CS[PC++]] ; DS[--SP] ← PC ; PC ← wk
340	CALL	CS[src0]		6Cu0	----	wk ← CS[src0] ; DS[--SP] ← PC ; PC ← wk
341	CALL	DS[--src]		6Bs0	----	wk ← DS[--src] ; DS[--SP] ← PC ; PC ← wk
342	CALL	DS[addr16]		6E20 aaaa	----	wk ← DS[CS[PC++]] ; DS[--SP] ← PC ; PC ← wk
343	CALL	DS[src + disp16]		6Ds0 dddd	----	wk ← DS[CS[PC++] + src] ; DS[--SP] ← PC ; PC ← wk
344	CALL	DS[src++]		6As0	----	wk ← DS[src++] ; DS[--SP] ← PC ; PC ← wk
345	CALL	DS[src]		69s0	----	wk ← DS[src] ; DS[--SP] ← PC ; PC ← wk
346	CALL	ES[addr16]		6E30 aaaa	----	wk ← ES[CS[PC++]] ; DS[--SP] ← PC ; PC ← wk
347	CALL	ES[src1]		6Cv0	----	wk ← ES[src1] ; DS[--SP] ← PC ; PC ← wk
348	CALL	PC + disp16		6F00 dddd	----	wk ← CS[PC++] + PC ; DS[--SP] ← PC ; PC ← wk
349	CALL	addr16		6E00 aaaa	----	wk ← CS[PC++] ; DS[--SP] ← PC ; PC ← wk
350	CALL	src		68s0	----	wk ← src ; DS[--SP] ← PC ; PC ← wk
351	CMP	reg, CS[PC + disp16]		A71r dddd	SVZB	0 ← reg - CS[CS[PC++] + PC]
352	CMP	reg, CS[addr16]		A61r aaaa	SVZB	0 ← reg - CS[CS[PC++]]

#	オペレーション	オペランド	機械語 (16進数)	フラグ	挙動
353	CMP	reg, CS[src0]	A4ur	SVZB	0 ← reg - CS[src0]
354	CMP	reg, DS[--src]	A3sr	SVZB	0 ← reg - DS[--src]
355	CMP	reg, DS[addr16]	A62r aaaa	SVZB	0 ← reg - DS[CS[PC++]]
356	CMP	reg, DS[src + disp16]	A5sr dddd	SVZB	0 ← reg - DS[CS[PC++] + src]
357	CMP	reg, DS[src++]	A2sr	SVZB	0 ← reg - DS[src++]
358	CMP	reg, DS[src]	A1sr	SVZB	0 ← reg - DS[src]
359	CMP	reg, ES[addr16]	A63r aaaa	SVZB	0 ← reg - ES[CS[PC++]]
360	CMP	reg, ES[src1]	A4vr	SVZB	0 ← reg - ES[src1]
361	CMP	reg, PC + disp16	A70r dddd	SVZB	0 ← reg - (CS[PC++] + PC)
362	CMP	reg, imm16	A60r nnnn	SVZB	0 ← reg - CS[PC++]
363	CMP	reg, src	A0sr	SVZB	0 ← reg – src
364	CMPB	reg, CS[PC + disp16]	AF1r dddd	SVZB	0 ← reg - BF - CS[CS[PC++] + PC]
365	CMPB	reg, CS[addr16]	AE1r aaaa	SVZB	0 ← reg - BF - CS[CS[PC++]]
366	CMPB	reg, CS[src0]	ACur	SVZB	0 ← reg - BF - CS[src0]
367	CMPB	reg, DS[--src]	ABsr	SVZB	0 ← reg - BF - DS[--src]
368	CMPB	reg, DS[addr16]	AE2r aaaa	SVZB	0 ← reg - BF - DS[CS[PC++]]
369	CMPB	reg, DS[src + disp16]	ADsr dddd	SVZB	0 ← reg - BF - DS[CS[PC++] + src]
370	CMPB	reg, DS[src++]	AAsr	SVZB	0 ← reg - BF - DS[src++]
371	CMPB	reg, DS[src]	A9sr	SVZB	0 ← reg - BF - DS[src]
372	CMPB	reg, ES[addr16]	AE3r aaaa	SVZB	0 ← reg - BF - ES[CS[PC++]]
373	CMPB	reg, ES[src1]	ACvr	SVZB	0 ← reg - BF - ES[src1]
374	CMPB	reg, PC + disp16	AF0r dddd	SVZB	0 ← reg - BF - (CS[PC++] + PC)
375	CMPB	reg, imm16	AE0r nnnn	SVZB	0 ← reg - BF - CS[PC++]
376	CMPB	reg, src	A8sr	SVZB	0 ← reg - BF – src
377	CMPQ	reg, imm4m	03nr	SVZB	0 ← reg – SignExt (imm4m) ※imm4m の範囲は -1 〜-16 ※ただし埋め込まれるのは ~imm4m した値 (0x0〜0xF)
378	CMPQ	reg, imm4p	0Bnr	SVZB	0 ← reg – ZeroExt (imm4p) ※imm4p の範囲は 0 〜+15
379	CPUVER		0100	----	R1 ← TD16RM1バージョン番号 ※常に固定値を返す:RRCを16回で16bit生成する(実際は上下8bitずつの2分割)
380	CSBNKRAMSEL		0761	----	SYS[3] ← 1 ※CodeSegBank SRAM Select CS / ICSの前半はSRAM (後半もSRAM)
381	CSBNKROMSEL		0760	----	SYS[3] ← 0 ※CodeSegBank FlashROM Select CS / ICSの前半はFlashROM (後半はSRAM)
382	CSBNKSEL	reg	066r	----	SYS[3] ← LOW1(reg) ※CodeSegBank Select ※SYSCTL SYS[3], reg の別名
383	DEC	CS[PC + disp16]	6719 dddd	SVZB	wk ← CS[PC++] + PC ; CS[wk] ← CS[wk] - 1
384	DEC	CS[addr16]	6619 aaaa	SVZB	wk ← CS[PC++] ; CS[wk] ← CS[wk] - 1
385	DEC	CS[tgt0]	64u9	SVZB	wk ← tgt0 ; CS[wk] ← CS[wk] - 1
386	DEC	DS[--tgt]	63t9	SVZB	wk ← --tgt ; DS[wk] ← DS[wk] - 1
387	DEC	DS[addr16]	6629 aaaa	SVZB	wk ← CS[PC++] ; DS[wk] ← DS[wk] - 1
388	DEC	DS[tgt + disp16]	65t9 dddd	SVZB	wk ← CS[PC++] + tgt ; DS[wk] ← DS[wk] - 1
389	DEC	DS[tgt++]	62t9	SVZB	wk ← tgt++ ; DS[wk] ← DS[wk] - 1
390	DEC	DS[tgt]	61t9	SVZB	wk ← tgt ; DS[wk] ← DS[wk] - 1
391	DEC	ES[addr16]	6639 aaaa	SVZB	wk ← CS[PC++] ; ES[wk] ← ES[wk] - 1
392	DEC	ES[tgt1]	64v9	SVZB	wk ← tgt1 ; ES[wk] ← ES[wk] - 1

#	オペレーション	オペランド	機械語 (16進数)	フラグ	挙動
393	DEC	tgt	60t9	SVZB	tgt ← tgt - 1
394	DECB	CS[PC + disp16]	6711 dddd	SVZB	wk ← CS[PC++] + PC ; CS[wk] ← CS[wk] - BF
395	DECB	CS[addr16]	6611 aaaa	SVZB	wk ← CS[PC++] ; CS[wk] ← CS[wk] - BF
396	DECB	CS[tgt0]	64u1	SVZB	wk ← tgt0 ; CS[wk] ← CS[wk] - BF
397	DECB	DS[--tgt]	63t1	SVZB	wk ← --tgt ; DS[wk] ← DS[wk] - BF
398	DECB	DS[addr16]	6621 aaaa	SVZB	wk ← CS[PC++] ; DS[wk] ← DS[wk] - BF
399	DECB	DS[tgt + disp16]	65t1 dddd	SVZB	wk ← CS[PC++] + tgt ; DS[wk] ← DS[wk] - BF
400	DECB	DS[tgt++]	62t1	SVZB	wk ← tgt++ ; DS[wk] ← DS[wk] - BF
401	DECB	DS[tgt]	61t1	SVZB	wk ← tgt ; DS[wk] ← DS[wk] - BF
402	DECB	ES[addr16]	6631 aaaa	SVZB	wk ← CS[PC++] ; ES[wk] ← ES[wk] - BF
403	DECB	ES[tgt1]	64v1	SVZB	wk ← tgt1 ; ES[wk] ← ES[wk] - BF
404	DECB	tgt	60t1	SVZB	tgt ← tgt - BF
405	DELAY1FD		0141	----	WCK512(正論理)の立ち下がり(FallDown)エッジ検出するまで 0 ～(0.98+α) ミリ秒待つ ※SYSCK=32k の時は、NOP 命令だけでも約2.2ms 消費するので余り意味がない命令
406	DELAY1RU		01C1	----	WCK512(正論理)の立ち上がり(RiseUp)エッジ検出するまで 0 ～(0.98+α) ミリ秒待つ ※SYSCK=32k の時は、NOP 命令だけでも約2.2ms 消費するので余り意味がない命令
407	DELAY4FD		0142	----	WCK128(正論理)の立ち下がり(FallDown)エッジ検出するまで 0 ～(3.91+α) ミリ秒待つ ※32kHz の方をカウントするが(内部処理の関係で)SYSCK の選択により、+α分の時間が増減する
408	DELAY4RU		01C2	----	WCK128(正論理)の立ち上がり(RiseUp)エッジ検出するまで 0 ～(3.91+α) ミリ秒待つ ※32kHzの方をカウントするが(内部処理の関係で)SYSCKの選択により、+α分の時間が増減する
409	DELAY16FD		0143	----	WCK32 (正論理)の立ち下がり(FallDown)エッジ検出するまで 0 ～(15.6+α) ミリ秒待つ ※32kHz の方をカウントするが(内部処理の関係で)SYSCK の選択により、+α分の時間が増減する
410	DELAY16RU		01C3	----	WCK32 (正論理)の立ち上がり(RiseUp)エッジ検出するまで 0 ～(15.6+α) ミリ秒待つ ※32kHzの方をカウントするが(内部処理の関係で)SYSCKの選択により、+α分の時間が増減する
411	DELAY63FD		0144	----	WCK8 (正論理)の立ち下がり(FallDown)エッジ検出するまで 0 ～(62.5+α) ミリ秒待つ ※32kHz の方をカウントするが(内部処理の関係で)SYSCK の選択により、+α分の時間が増減する
412	DELAY63RU		01C4	----	WCK8 (正論理)の立ち上がり(RiseUp)エッジ検出するまで 0 ～(62.5+α) ミリ秒待つ ※32kHzの方をカウントするが(内部処理の関係で)SYSCKの選択により、+α分の時間が増減する
413	DELAYINIT		0140	----	Delay Initialization
414	DIV16STGn		013n	----	符号無し16bit 除算:第1～16ステージ:R1 ÷ R3.R2 ⇒ ER1 … R1
415	DJNZ	reg, $ + 1 - imm4p	09nr	----	--reg ; if (reg != 0) then PC ← PC - ZeroExt **(**imm4p**)** ※imm4p の範囲は 0 ～+15
416	DLJNZ	reg, $ + 2 + disp16	019r dddd	----	--reg ; if (reg != 0) then PC ← CS[PC++] + PC
417	FABS	FPr	01A4 01rr	----	TD16float: 絶対値(正) ※FPX は FXABS を使う事。これでも出来るが実行時間が長い。
418	FABSNEG	FPr	01A4 03rr	----	TD16float: 絶対値(負) ※FPX は FXABSNEG を使う事。これでも出来るが実行時間が長い。
419	FADDEXP	FPr, Rs	01A6 0srr	SVZC	TD16float: FPr の指数部に、汎用レジスタ Rs を加算 ※FPX でもこれを使う。
420	FBAS10	FPd, Rs	0165 0sdd	SVZC	TD16float: 底2から底10に変換する係数 ※FPd は特殊なフォーマットになるので要注意。 また、Rsの上位8bitがFLGにコピーされる(∞判定用)
421	FBAS10NEG	FPd, Rs	0165 2sdd	SVZC	TD16float: 底2から底10に変換する係数(負) ※FPd は特殊なフォーマットになるので要注意。 また、Rsの上位8bitがFLGにコピーされる(∞判定用)
422	FCHKSGN	FPr	01A4 00rr	S???	TD16float: 符号check のみ ※FPX は FXCHKSGN を使う事。これでも出来るが実行時間が長い。
423	FCLR	FPr	01E5 01rr	----	TD16float: 0.0 ⇒ FPr ※FPX は FXCLR を使う事。これだと仮数部末尾16bitがクリアされない
424	FCMPEXP	FPr, FPs	01AC ssrr	SVZC	TD16float: 指数部(FPr)と指数部(FPs)のどちらが大きいか？(符号有り比較) ※$FPWK に差の値が格納されるので注意

#	オペレーション	オペランド	機械語 (16進数)	フラグ	挙動
425	FCMPMAN	FPr, FPs	01AD ssrr	??ZC	TD16float: 仮数部(FPr)と仮数部(FPs)のどちらが大きいか？(符号無し比較)
426	FCONST	FPd, imm8	016D nndd	----	TD16float: 定数 ⇒ FPd ※FPX は FXCONST を使う事。こちらだと仮数部末尾16bitがクリアされない
427	FCONST_M_INF	FPr	01E5 03rr	----	TD16float: -2**384 ⇒ FPr : 0xFF ⇒ FPrSgn ; 0x0180 ⇒ FPrExp ; 0x0080_0000 ⇒ FPrMan0～3
428	FCONST_P_INF	FPr	01E5 02rr	----	TD16float: +2**384 ⇒ FPr : 0x00 ⇒ FPrSgn ; 0x0180 ⇒ FPrExp ; 0x0080_0000 ⇒ FPrMan0～3
429	FCVT16I	Rt, FPr	0166 0trr	0VZ?	TD16float: FPr を sint16 に変換(小数点以下は切り捨て) ※変換できなかった時は VF をセット(Rtは無変化)
430	FCVT32F	Rt_Rw, FPr	0167 0trr	----	TD16float: FPr を float32 に変換
431	FEXP16I	FPd, Rs	016E 0sdd	----	TD16float: Rs ⇒ FPd 展開 ※直後に FXNORM が必要 ※FPX の場合、FXEXP16 を使う事。こちらだと仮数部の末尾16bitが無変化になるので要注意。
432	FEXP32F	FPd, Rs_Rw	016F 0sdd	----	TD16float: Rs_Rw ⇒ FPd 展開 ※FPX の場合、FXEXP32 を使う事。こちらだと仮数部の末尾16bitが無変化になるので要注意。
433	FEXPELM	FPr	01A5 01rr	----	TD16float: 指数部のみ化: 0⇒FPr.Sgn8, 1.0⇒FPr.Man32
434	FILLBLK	xS[tgt++], reg	52tr	----	do { xS[tgt++] ← reg } while (--ER0); ※ER0==0 の時は65536回になる xS はtgt 番号の下位2bit から、CS / ICS / DS / ES のいずれか1つが選ばれる。
435	FLDMAN	FPr, Rs_Rw	01E6 1srr	----	TD16float: Rs_Rw ⇒ FPrMan1～3
436	FMANELM	FPr	01A5 03rr	----	TD16float: 仮数部のみ化: 0⇒FPr.Sgn8, 0⇒FPr.Exp16
437	FMKHALF	FPr	01E5 00rr	----	TD16float: 四捨五入用の±0.5 作成:0xFFFF ⇒ FPrExp ; 0x0080_0000 ⇒ FPrMan0～3 ※FPrSgnはそのまま
438	FMOV	FPd, FPs	016C ssdd	----	TD16float: FPs ⇒ FPd ※FPs＝FPX の時は TD16float形式に変換(chopする)。 FPd＝FPX の時は仮数部末尾16bitがクリアされない(なので FXEXP を使うべき)
439	FNCALL	imm8	10nn	----	ファンクション・コール: DS[--SP] ← PC ; PC ← ZeroExt (imm8) * 2
440	FNEG	FPr	01A4 02rr	S???	TD16float: 符号反転※FPX は FXNEG を使う事。これでも出来るが実行時間が長い。
441	FPOW10	FPd, Rs	0164 0sdd	----	TD16float: 10 の冪乗(+10**±n) ※指定できる冪数n は -128～0～+127 の整数のみ
442	FPOW10NEG	FPd, Rs	0164 2sdd	----	TD16float: 10 の冪乗(+10**±n)の負数 ※指定できる冪数n は -128～0～+127 の整数のみ
443	FSGNELM	FPr	01A5 00rr	----	TD16float: 符号部のみ化: 0⇒FPr.Exp16, 1.0⇒FPr.Man32
444	FSGNEXPELM	FPr	01A5 02rr	----	TD16float: 符号・指数部のみ化: 1.0⇒FPr.Man32
445	FSTMAN	Rt_Rw, FPr	01E6 0trr	----	TD16float: FPrMan1～3 :: 0x00 ⇒ Rt_Rw
446	FXABS		01A1	----	TD16floatX: 絶対値(正)
447	FXABSNEG		01A3	----	TD16floatX: 絶対値(負)
448	FXADD1	FPs	01E7 ssFC	??ZC	TD16floatX: 加算命令#1:FPX + FPs ⇒ FPX ※絶対値(FPs) ≧ 絶対値(FPX) である事 ※符号部はここで調整する。内部作業用に $FPWK を使用する
449	FXADD2		01E8	??ZC	TD16floatX: 加算命令#2:FPX + FPs ⇒ FPX ※絶対値(FPs) ≧ 絶対値(FPX) である事
450	FXADDNORM1		01E9	----	TD16floatX: 加算時のみの正規化(1段目)
451	FXADDNORM2		01EA	----	TD16floatX: 加算時のみの正規化(2段目):8bit単位×2
452	FXADDNORM3		01EB	----	TD16floatX: 加算時のみの正規化(3段目):8bit単位×2
453	FXALIGN1		0160	00ZC	TD16floatX: FPX の指数部が FPs に一致するよう調整する:1回目 ※この結果、FPX は 非正規化状態になるので要注意, CF==1:シフト量が大きすぎる, ZF==1:32bitシフトは不要
454	FXALIGN2		0161	00ZC	TD16floatX: FPX の指数部が FPs に一致するよう調整する:2回目 ※この結果、FPX は 非正規化状態になるので要注意, CF==1:シフト量が大きすぎる, ZF==1:16bitシフトは不要
455	FXALIGN3		0162	00ZC	TD16floatX: FPX の指数部が FPs に一致するよう調整する:3回目 ※この結果、FPX は 非正規化状態になるので要注意, CF==1:シフト量が大きすぎる, ZF==1: 8bitシフトは不要

#	オペレーション	オペランド	機械語 (16進数)	フラグ	挙動
456	FXALIGN4		0163	00ZC	TD16floatX: FPX の指数部が FPs に一致するよう調整する:4回目 ※この結果、FPX は 非正規化状態になるので要注意, CF==1:シフト量が大きすぎる, ZF==1:シフト量が0(シフトは不要)
457	FXBAS10	Rs	0165 1sFC	SVZC	TD16floatX: 底2から底10に変換する係数 ※FPX は特殊なフォーマットになるので要注意。また、Rsの上位8bitがFLGにコピーされる(∞判定用)
458	FXBAS10NEG	Rs	0165 3sFC	SVZC	TD16floatX: 底2から底10に変換する係数(負) ※FPX は特殊なフォーマットになるので要注意。また、Rsの上位8bitがFLGにコピーされる(∞判定用)
459	FXBOUND		016A	----	TD16floatX: 加減乗算後、overflow/underflow 発生時に正規化する
460	FXCHKSGN		01A0	S???	TD16floatX: 符号check のみ ※使える cc はNS / S のみ。VF にはゴミ値が入るので SGT / SGE / SLE / SLT は使用不可。
461	FXCLR		01E1	----	TD16floatX: 0.0 ⇒ FPX
462	FXCONST	imm8	01ED nn6D	----	TD16floatX: 定数 ⇒ FPX
463	FXCONST_M_INF		01E5 13FC	----	TD16floatX: -2**384 ⇒ FPX : 0xFF ⇒ FPXSgn ; 0x0180 ⇒ FPXExp ; 0x0080_0000_0000 ⇒ FPXMan0〜5 ※機械語2word目の0xFCは必須
464	FXCONST_P_INF		01E5 12FC	----	TD16floatX: +2**384 ⇒ FPX : 0x00 ⇒ FPXSgn ; 0x0180 ⇒ FPXExp ; 0x0080_0000_0000 ⇒ FPXMan0〜5 ※機械語2word目の0xFCは必須
465	FXDIVINI1	FPnum, FPden	0127 ddnn	?VZ0	TD16floatX: 除算命令初期化#1 : FPnum ÷ FPden ⇒ FPX
466	FXDIVINI2		0128	----	TD16floatX: 除算命令初期化#2
467	FXDIVSTG1		0120	??ZB	TD16floatX: 除算命令処理段#1
468	FXDIVSTG2		0121	----	TD16floatX: 除算命令処理段#2
469	FXDIVSTG3		0122	----	TD16floatX: 除算命令処理段#3
470	FXEXP16I	Rs	01EE 0s6E	----	TD16floatX: Rs ⇒ FPX 展開 ※直後に FXNORM が必要
471	FXEXP32F	Rs_Rw	01EF 0s6F	----	TD16floatX: Rs_Rw ⇒ FPX 展開
472	FXEXPELM		01A5 11FC	----	TD16floatX: 指数部のみ化: 0⇒FPX.Sgn8, 1.0⇒FPX.Man48 ※機械語2word目の0xFCは必須
473	FXLDMAN	Rs_Rw	01E6 3sFC	----	TD16floatX: Rs_Rw :: 0x00 ⇒ FPXMan1〜5
474	FXMANELM		01A5 13FC	----	TD16floatX: 仮数部のみ化: 0⇒FPX.Sgn8, 0⇒FPX.Exp16 ※機械語2word目の0xFCは必須
475	FXMKHALF		01E5 10FC	----	TD16floatX: 四捨五入用の±0.5 作成:0xFFFF ⇒ FPXExp ; 0x0080_0000_0000 ⇒ FPXMan0〜5 ※FPXSgnはそのまま※機械語2word目の0xFCは必須
476	FXMOV	FPs	01EC ssFC	----	TD16floatX: FPs ⇒ FPX ※TD16floatx形式に変換(FPX の仮数部の末尾16bitは0クリアする)
477	FXMUL1	FPr, FPs	01A7 ssrr	00Z0	TD16floatX: 乗算命令#1:FPr * FPs ⇒ FPX ※符号部はここで調整する。 ZF==1の時は FXCLR を呼ぶ事。内部作業用に$FPWKを使用する
478	FXMUL2		01A8	----	TD16floatX: 乗算命令#2
479	FXMUL3		01A9	----	TD16floatX: 乗算命令#3
480	FXMUL4		01AA	----	TD16floatX: 乗算命令#4
481	FXMUL5		01AB	----	TD16floatX: 乗算命令#5
482	FXNEG		01A2	S???	TD16floatX: 符号反転
483	FXNORM		0168	----	TD16floatX: 正規化:1bit単位
484	FXPOW10	Rs	0164 1sFC	----	TD16floatX: 10 の冪乗(+10**±n) ※指定できる冪数n は -128〜0〜+127 の整数のみ
485	FXPOW10NEG	Rs	0164 3sFC	----	TD16floatX: 10 の冪乗(+10**±n)の負数 ※指定できる冪数n は -128〜0〜+127 の整数のみ
486	FXROUND24		0129	----	TD16floatX: 24bit目の0捨1入処理(さらに仮数部の末尾16bitを0クリアする) ※一種の加算処理
487	FXROUND32		012B	----	TD16floatX: 32bit目の0捨1入処理(さらに仮数部の末尾 8bitを0クリアする) ※一種の加算処理
488	FXROUNDUP		0169	?VZ?	TD16floatX: 切り上げ処理:FPX.Man48 + LUT[FPX.Exp16] ⇒ FPX.Man48 ※非正規化状態のまま:2.0以上になる可能性があるので、FXADDNORM1/FXNORM必須
489	FXSGNELM		01A5 10FC	----	TD16floatX: 符号部のみ化: 0⇒FPX.Exp16, 1.0⇒FPX.Man48 ※機械語2word目の0xFCは必須

#	オペレーション	オペランド	機械語 (16進数)	フラグ	挙動
490	FXSGNEXPELM		01A5 12FC	----	TD16floatX: 符号・指数部のみ化: 1.0⇒FPX.Man48 ※機械語2word目の0xFCは必須
491	FXSTMAN	Rt_Rw	01E6 2tFC	----	TD16floatX: FPXMan1～4 ⇒ Rt_Rw
492	FXZCHKMAN		01E2	00Z1	TD16floatX: FPX の仮数部が 0.0(オール0) か？0.0 なら指数部を 0xFE81 にする。
493	FXZCMP		01E0	S0Z1	TD16floatX: ゼロと比較　※符号部と指数部のみで判断する。仮数部が 0.0 かどうかはチェックしない。
494	FZCMP	FPr	01E4 00rr	S0Z1	TD16float: ゼロと比較　※FPX は FXZCMP を使う事。これでも出来るが実行時間が若干長い
495	HLT	(imm7)	0800 ~ 087F	----	HALT状態になる(割込み処理後かCont.すると処理継続する)。imm7 省略時は0x00
496	I2CCHKACK		0183	???C	i2c : check ACK condition: CF=0:ACK, CF=1:NACK
497	I2CINP	reg	0C6r	----	LOW8(reg) ← INP[6] ※i2c用入力Reg, i2c機能不使用時は唯の入力port
498	I2COUTP	imm8	46nn	----	OUTP[6] ← imm8 ※i2c用出力Reg, i2c機能不使用時は唯の出力port
499	I2COUTP	reg	046r	----	OUTP[6] ← LOW8(reg) ※i2c用出力Reg, i2c機能不使用時は唯の出力port
500	I2CRCVDAT		0184	----	i2c : receive data from slave dev.
501	I2CREPSTART		0181	----	i2c : repeat start condition
502	I2CSNDACK		0185	----	i2c : send ACK to slave dev.
503	I2CSNDDAT		0182	----	i2c : send data to slave dev.
504	I2CSNDNACK		0186	----	i2c : send NACK to slave dev.
505	I2CSTART		0180	----	i2c : start condition
506	I2CSTOP		0187	----	i2c : stop condition
507	IABLE	reg	060r	----	SYS[0] ← LOW1(reg) ※割込み禁止 / 許可 ※SYSCTL SYS[0], reg の別名
508	IDIS		0700	----	SYS[0] ← 0 ※割込み禁止 ※SYSCTL SYS[0], imm1 の別名
509	IEMASK	imm4	072n	----	SYS[1] ← imm4 ※bit0:PushSW, bit1:Timer(15ms/125ms/1s), bit2:(VSS_VBI), bit3:Keyboard
510	IEMASK	reg	062r	----	SYS[1] ← LOW4(reg) ※bit0:PushSW, bit1:Timer(15ms/125ms/1s), bit2:(VSS_VBI), bit3:Keyboard
511	IEN		0701	----	SYS[0] ← 1 ※割込み許可 ※SYSCTL SYS[0], imm1 の別名
512	INC	CS[PC + disp16]	6718 dddd	SVZC	wk ← CS[PC++] + PC ; CS[wk] ← CS[wk] + 1
513	INC	CS[addr16]	6618 aaaa	SVZC	wk ← CS[PC++] ; CS[wk] ← CS[wk] + 1
514	INC	CS[tgt0]	64u8	SVZC	wk ← tgt0 ; CS[wk] ← CS[wk] + 1
515	INC	DS[--tgt]	63t8	SVZC	wk ← --tgt ; DS[wk] ← DS[wk] + 1
516	INC	DS[addr16]	6628 aaaa	SVZC	wk ← CS[PC++] ; DS[wk] ← DS[wk] + 1
517	INC	DS[tgt + disp16]	65t8 dddd	SVZC	wk ← CS[PC++] + tgt ; DS[wk] ← DS[wk] + 1
518	INC	DS[tgt++]	62t8	SVZC	wk ← tgt++ ; DS[wk] ← DS[wk] + 1
519	INC	DS[tgt]	61t8	SVZC	wk ← tgt ; DS[wk] ← DS[wk] + 1
520	INC	ES[addr16]	6638 aaaa	SVZC	wk ← CS[PC++] ; ES[wk] ← ES[wk] + 1
521	INC	ES[tgt1]	64v8	SVZC	wk ← tgt1 ; ES[wk] ← ES[wk] + 1
522	INC	tgt	60t8	SVZC	tgt ← tgt + 1
523	INCC	CS[PC + disp16]	6710 dddd	SVZC	wk ← CS[PC++] + PC ; CS[wk] ← CS[wk] + CF
524	INCC	CS[addr16]	6610 aaaa	SVZC	wk ← CS[PC++] ; CS[wk] ← CS[wk] + CF
525	INCC	CS[tgt0]	64u0	SVZC	wk ← tgt0 ; CS[wk] ← CS[wk] + CF
526	INCC	DS[--tgt]	63t0	SVZC	wk ← --tgt ; DS[wk] ← DS[wk] + CF
527	INCC	DS[addr16]	6620 aaaa	SVZC	wk ← CS[PC++] ; DS[wk] ← DS[wk] + CF
528	INCC	DS[tgt + disp16]	65t0 dddd	SVZC	wk ← CS[PC++] + tgt ; DS[wk] ← DS[wk] + CF
529	INCC	DS[tgt++]	62t0	SVZC	wk ← tgt++ ; DS[wk] ← DS[wk] + CF

#	オペレーション	オペランド	機械語 (16進数)	フラグ	挙動
530	INCC	DS[tgt]	61t0	SVZC	wk ← tgt ; DS[wk] ← DS[wk] + CF
531	INCC	ES[addr16]	6630 aaaa	SVZC	wk ← CS[PC++] ; ES[wk] ← ES[wk] + CF
532	INCC	ES[tgt1]	64v0	SVZC	wk ← tgt1 ; ES[wk] ← ES[wk] + CF
533	INCC	tgt	60t0	SVZC	tgt ← tgt + CF
534	INP	reg, PIO[port4]	0Cpr	----	LOW8(reg) ← INP[port4]
535	INP	reg, PIO[src]	0Dsr	----	LOW8(reg) ← INP[LOW4(src)]
536	INP0	reg	0C0r	----	LOW8(reg) ← INP[0] ※Joypad or Push SWs, 別名、PADINP
537	INP1	reg	0C1r	----	LOW8(reg) ← INP[1]
538	INP2	reg	0C2r	----	LOW8(reg) ← INP[2]
539	INP3	reg	0C3r	----	LOW8(reg) ← INP[3]
540	INP4	reg	0C4r	----	LOW8(reg) ← INP[4]
541	INP5	reg	0C5r	----	LOW8(reg) ← INP[5] ※別名、KBINP
542	INP6	reg	0C6r	----	LOW8(reg) ← INP[6] ※別名、I2CINP
543	INP7	reg	0C7r	----	LOW8(reg) ← INP[7] ※別名、SPIINP
544	INP8	reg	0C8r	----	LOW8(reg) ← INP[8]
545	INP9	reg	0C9r	----	LOW8(reg) ← INP[9]
546	INP10	reg	0CAr	----	LOW8(reg) ← INP[10]
547	INP11	reg	0CBr	----	LOW8(reg) ← INP[11]
548	INP12	reg	0CCr	----	LOW8(reg) ← INP[12]
549	INP13	reg	0CDr	----	LOW8(reg) ← INP[13]
550	INP14	reg	0CEr	----	LOW8(reg) ← INP[14]
551	INP15	reg	0CFr	----	LOW8(reg) ← INP[15]
552	IRET	reg	065r	----	IUNLOCK値をセットして、割込み処理ルーチン#nを終了(nIntEndT#アサート) ※直値imm4 の方の IRET命令を使うべき
553	IRET0		0751	----	IUNLOCK値をセットして、割込み処理ルーチン#0を終了(nIntEndT#アサート)
554	IRET1		0752	----	IUNLOCK値をセットして、割込み処理ルーチン#1を終了(nIntEndT#アサート)
555	IRET2		0754	----	IUNLOCK値をセットして、割込み処理ルーチン#2を終了(nIntEndT#アサート)
556	IRET3		0758	----	IUNLOCK値をセットして、割込み処理ルーチン#3を終了(nIntEndT#アサート)
557	IUNLOCK	imm4	074n	----	SYS[2] ← imm4 ※IRET 命令の内部で使用。ユーザーが直接叩く事はない
558	IUNLOCK	reg	064r	----	SYS[2] ← LOW4(reg) ※IRET 命令の内部で使用。ユーザーが直接叩く事はない
559	J	$ + 1 + disp8	11dd	----	PC ← SignExt (disp8) + PC ※always
560	JB	$ + 1 + disp8	19dd	----	if (BF == 1) then { PC ← SignExt (disp8) + PC } ※ボロー有り
561	JC	$ + 1 + disp8	18dd	----	if (CF == 1) then { PC ← SignExt (disp8) + PC } ※キャリー有り
562	JEQ	$ + 1 + disp8	13dd	----	if (ZF == 1) then { PC ← SignExt (disp8) + PC } ※＝
563	JMP	CS[PC + disp16]	4F10 dddd	----	PC ← CS[CS[PC] + PC + 1]
564	JMP	CS[addr16]	4E10 aaaa	----	PC ← CS[CS[PC]]
565	JMP	CS[src0]	4Cu0	----	PC ← CS[src0]
566	JMP	DS[--src]	4Bs0	----	PC ← DS[--src]
567	JMP	DS[addr16]	4E20 aaaa	----	PC ← DS[CS[PC]]
568	JMP	DS[src + disp16]	4Ds0 dddd	----	PC ← DS[CS[PC] + src]
569	JMP	DS[src++]	4As0	----	PC ← DS[src++] ※ JMP DS[SP++] は RET
570	JMP	DS[src]	49s0	----	PC ← DS[src]

#	オペレーション	オペランド	機械語 (16進数)	フラグ	挙動		
571	JMP	ES[addr16]	4E30 aaaa	----	PC ← ES[CS[PC]]		
572	JMP	ES[src1]	4Cv0	----	PC ← ES[src1]		
573	JMP	addr16	4E00 aaaa	----	PC ← CS[PC]		
574	JMP	src	48s0	----	PC ← src		
575	JNB	$ + 1 + disp8	18dd	----	if (BF == 0)	then { PC ← SignExt (disp8) + PC }	※ボロー無し
576	JNC	$ + 1 + disp8	19dd	----	if (CF == 0)	then { PC ← SignExt (disp8) + PC }	※キャリー無し
577	JNEQ	$ + 1 + disp8	12dd	----	if (ZF == 0)	then { PC ← SignExt (disp8) + PC }	※≠
578	JNS	$ + 1 + disp8	16dd	----	if (SF == 0)	then { PC ← SignExt (disp8) + PC }	※正(＋)
579	JNV	$ + 1 + disp8	14dd	----	if (VF == 0)	then { PC ← SignExt (disp8) + PC }	※Overflow無し
580	JNZ	$ + 1 + disp8	12dd	----	if (ZF == 0)	then { PC ← SignExt (disp8) + PC }	※≠
581	JS	$ + 1 + disp8	17dd	----	if (SF == 1)	then { PC ← SignExt (disp8) + PC }	※負(－)
582	JSGE	$ + 1 + disp8	1Cdd	----	if (SF == VF)	then { PC ← SignExt (disp8) + PC }	※符号有り≧
583	JSGT	$ + 1 + disp8	1Edd	----	if ((SF == VF) && (ZF == 0))	then { PC ← SignExt (disp8) + PC }	※符号有り＞
584	JSLE	$ + 1 + disp8	1Fdd	----	if ((SF != VF) || (ZF == 1))	then { PC ← SignExt (disp8) + PC }	※符号有り≦
585	JSLT	$ + 1 + disp8	1Ddd	----	if (SF != VF)	then { PC ← SignExt (disp8) + PC }	※符号有り＜
586	JUGE	$ + 1 + disp8	18dd	----	if (BF == 0)	then { PC ← SignExt (disp8) + PC }	※符号無し≧
587	JUGT	$ + 1 + disp8	1Add	----	if ((BF == 0) && (ZF == 0))	then { PC ← SignExt (disp8) + PC }	※符号無し＞
588	JULE	$ + 1 + disp8	1Bdd	----	if ((BF == 1) || (ZF == 1))	then { PC ← SignExt (disp8) + PC }	※符号無し≦
589	JULT	$ + 1 + disp8	19dd	----	if (BF == 1)	then { PC ← SignExt (disp8) + PC }	※符号無し＜
590	JV	$ + 1 + disp8	15dd	----	if (VF == 1)	then { PC ← SignExt (disp8) + PC }	※Overflow発生
591	JZ	$ + 1 + disp8	13dd	----	if (ZF == 1)	then { PC ← SignExt (disp8) + PC }	※＝
592	KBINP	reg	0C5r	----	LOW8(reg) ← INP[5]	※KBC用入力Reg, KBC機能不使用時は唯の入力port	
593	LCDOUTP	imm8	40nn	----	OUTP[0] ← imm8	※キャラ LCD	
594	LCDOUTP	reg	040r	----	OUTP[0] ← LOW8(reg)	※キャラ LCD	
595	LCDSNDDAT		0107	----	キャラ LCD の E信号に1パルス出力		
596	LDFLG	imm4	077n	設定値	FLG ← imm4		
597	LDFLG	reg	067r	設定値	FLG ← LOW4(reg)		
598	LDNMFLG	imm4	07Fn	設定値	NormalModeFLG ← imm4	※割込みモード時のみ	
599	LDNMFLG	reg	06Fr	設定値	NormalModeFLG ← LOW4(reg)	※割込みモード時のみ	
600	LDNMPC	src	48s2	----	NormalModePC ← src ※割込みモード時のみ。ノーマルモード時では JMP src と同じ挙動。		
601	LEA	reg, CS[PC + disp16]	3F1r dddd	----	reg ← CS[PC++] + PC ※セグメントの指定はできるが、得られる実効アドレスの計算値には無関係		
602	LEA	reg, CS[addr16]	3E1r aaaa	----	reg ← CS[PC++] ※セグメントの指定はできるが、得られる実効アドレスの計算値には無関係		
603	LEA	reg, CS[src0]	3Cur	----	reg ← src0 ※セグメントの指定はできるが、得られる実効アドレスの計算値には無関係		
604	LEA	reg, DS[--src]	3Bsr	----	--src ; reg ← src ※セグメントの指定はできるが、得られる実効アドレスの計算値には無関係		
605	LEA	reg, DS[addr16]	3E2r aaaa	----	reg ← CS[PC++] ※セグメントの指定はできるが、得られる実効アドレスの計算値には無関係		
606	LEA	reg, DS[src + disp16]	3Dsr dddd	----	reg ← CS[PC++] + src ※セグメントの指定はできるが、得られる実効アドレスの計算値には無関係		

#	オペレーション	オペランド	機械語 (16進数)	フラグ	挙動
607	LEA	reg, DS[src++]	3Asr	----	reg ← src ; src++ ※セグメントの指定はできるが、得られる実効アドレスの計算値には無関係
608	LEA	reg, DS[src]	39sr	----	reg ← src ※セグメントの指定はできるが、得られる実効アドレスの計算値には無関係
609	LEA	reg, ES[addr16]	3E3r aaaa	----	reg ← CS[PC++] ※セグメントの指定はできるが、得られる実効アドレスの計算値には無関係
610	LEA	reg, ES[src1]	3Cvr	----	reg ← src1 ※セグメントの指定はできるが、得られる実効アドレスの計算値には無関係
611	LJ	$ + 2 + disp16	0111 dddd	----	PC ← CS[PC++] + PC ※always
612	LJB	$ + 2 + disp16	0119 dddd	----	if (BF == 1) then { PC ← CS[PC++] + PC } ※ボロー有り
613	LJC	$ + 2 + disp16	0118 dddd	----	if (CF == 1) then { PC ← CS[PC++] + PC } ※キャリー有り
614	LJEQ	$ + 2 + disp16	0113 dddd	----	if (ZF == 1) then { PC ← CS[PC++] + PC } ※＝
615	LJNB	$ + 2 + disp16	0118 dddd	----	if (BF == 0) then { PC ← CS[PC++] + PC } ※ボロー無し
616	LJNC	$ + 2 + disp16	0119 dddd	----	if (CF == 0) then { PC ← CS[PC++] + PC } ※キャリー無し
617	LJNEQ	$ + 2 + disp16	0112 dddd	----	if (ZF == 0) then { PC ← CS[PC++] + PC } ※≠
618	LJNS	$ + 2 + disp16	0116 dddd	----	if (SF == 0) then { PC ← CS[PC++] + PC } ※正(＋)
619	LJNV	$ + 2 + disp16	0114 dddd	----	if (VF == 0) then { PC ← CS[PC++] + PC } ※Overflow無し
620	LJNZ	$ + 2 + disp16	0112 dddd	----	if (ZF == 0) then { PC ← CS[PC++] + PC } ※≠
621	LJS	$ + 2 + disp16	0117 dddd	----	if (SF == 1) then { PC ← CS[PC++] + PC } ※負(－)
622	LJSGE	$ + 2 + disp16	011C dddd	----	if (SF == VF) then { PC ← CS[PC++] + PC } ※符号有り≧
623	LJSGT	$ + 2 + disp16	011E dddd	----	if ((SF == VF) && (ZF == 0)) then { PC ← CS[PC++] + PC } ※符号有り＞
624	LJSLE	$ + 2 + disp16	011F dddd	----	if ((SF != VF) || (ZF == 1)) then { PC ← CS[PC++] + PC } ※符号有り≦
625	LJSLT	$ + 2 + disp16	011D dddd	----	if (SF != VF) then { PC ← CS[PC++] + PC } ※符号有り＜
626	LJUGE	$ + 2 + disp16	0118 dddd	----	if (BF == 0) then { PC ← CS[PC++] + PC } ※符号無し≧
627	LJUGT	$ + 2 + disp16	011A dddd	----	if ((BF == 0) && (ZF == 0)) then { PC ← CS[PC++] + PC } ※符号無し＞
628	LJULE	$ + 2 + disp16	011B dddd	----	if ((BF == 1) || (ZF == 1)) then { PC ← CS[PC++] + PC } ※符号無し≦
629	LJULT	$ + 2 + disp16	0119 dddd	----	if (BF == 1) then { PC ← CS[PC++] + PC } ※符号無し＜
630	LJV	$ + 2 + disp16	0115 dddd	----	if (VF == 1) then { PC ← CS[PC++] + PC } ※Overflow発生
631	LJZ	$ + 2 + disp16	0113 dddd	----	if (ZF == 1) then { PC ← CS[PC++] + PC } ※＝
632	LOCNT	reg, src	015B 01sr	0VZ0	reg ← LOCNT(src) : Leading Ones Count: MSB から連続する '1' の個数をカウント: 0 個ならZF←1, ALL '1' なら reg←16 かつ VF←1
633	LSR	CS[PC + disp16]	671E dddd	00ZC	wk ← CS[PC++] + PC ; CS[wk] ← LogicShiftRight (CS[wk], 1)
634	LSR	CS[addr16]	661E aaaa	00ZC	wk ← CS[PC++] ; CS[wk] ← LogicShiftRight (CS[wk], 1)
635	LSR	CS[tgt0]	64uE	00ZC	wk ← tgt0 ; CS[wk] ← LogicShiftRight (CS[wk], 1)
636	LSR	DS[--tgt]	63tE	00ZC	wk ← --tgt ; DS[wk] ← LogicShiftRight (DS[wk], 1)
637	LSR	DS[addr16]	662E aaaa	00ZC	wk ← CS[PC++] ; DS[wk] ← LogicShiftRight (DS[wk], 1)
638	LSR	DS[tgt + disp16]	65tE dddd	00ZC	wk ← CS[PC++] + tgt ; DS[wk] ← LogicShiftRight (DS[wk], 1)
639	LSR	DS[tgt++]	62tE	00ZC	wk ← tgt++ ; DS[wk] ← LogicShiftRight (DS[wk], 1)
640	LSR	DS[tgt]	61tE	00ZC	wk ← tgt ; DS[wk] ← LogicShiftRight (DS[wk], 1)
641	LSR	ES[addr16]	663E aaaa	00ZC	wk ← CS[PC++] ; ES[wk] ← LogicShiftRight (ES[wk], 1)
642	LSR	ES[tgt1]	64vE	00ZC	wk ← tgt1 ; ES[wk] ← LogicShiftRight (ES[wk], 1)
643	LSR	tgt	60tE	00ZC	tgt ← LogicShiftRight (tgt, 1)
644	LSRM	reg, src, imm4	0156 Ensr	S0Z0	reg ← LogicShiftRight (src, imm4) ※VFは常に0、CFも常に0

#	オペレーション	オペランド		機械語 (16進数)	フラグ	挙動
645	LSRM	reg, src, qty		01Dq E0sr	S0Z0	reg ← LogicShiftRight (src, LOW4(qty))　※VFは常に0、CFも常に0
646	LZCNT	reg, src		015B 00sr	0VZ0	reg ← LZCNT(src) : Leading Zeros Count: MSB から連続する '0' の個数をカウント: 0 個ならZF←1, ALL '0' なら reg←16 かつ VF←1
647	MOV	CS[PC + disp16],	imm16	3710 dddd nnnn	----	wk ← CS[PC++] + PC ; CS[wk] ← CS[PC++]
648	MOV	CS[PC + disp16],	reg	271r dddd	----	wk ← CS[PC++] + PC ; CS[wk] ← reg
649	MOV	CS[addr16],	imm16	3610 aaaa nnnn	----	wk ← CS[PC++] ; CS[wk] ← CS[PC++]
650	MOV	CS[addr16],	reg	261r aaaa	----	wk ← CS[PC++] ; CS[wk] ← reg
651	MOV	CS[tgt0],	imm16	34u0 nnnn	----	CS[tgt0] ← CS[PC++]
652	MOV	CS[tgt0],	reg	24ur	----	CS[tgt0] ← reg
653	MOV	DS[--tgt],	imm16	33t0 nnnn	----	DS[--tgt] ← CS[PC++]
654	MOV	DS[--tgt],	reg	23tr	----	DS[--tgt] ← reg
655	MOV	DS[addr16],	imm16	3620 aaaa nnnn	----	wk ← CS[PC++] ; DS[wk] ← CS[PC++]
656	MOV	DS[addr16],	reg	262r aaaa	----	wk ← CS[PC++] ; DS[wk] ← reg
657	MOV	DS[tgt + disp16],	imm16	35t0 dddd nnnn	----	wk ← CS[PC++] + tgt ; DS[wk] ← CS[PC++]
658	MOV	DS[tgt + disp16],	reg	25tr dddd	----	wk ← CS[PC++] + tgt ; DS[wk] ← reg
659	MOV	DS[tgt++],	imm16	32t0 nnnn	----	DS[tgt++] ← CS[PC++]
660	MOV	DS[tgt++],	reg	22tr	----	DS[tgt++] ← reg
661	MOV	DS[tgt],	imm16	31t0 nnnn	----	DS[tgt] ← CS[PC++]
662	MOV	DS[tgt],	reg	21tr	----	DS[tgt] ← reg
663	MOV	ES[addr16],	imm16	3630 aaaa nnnn	----	wk ← CS[PC++] ; ES[wk] ← CS[PC++]
664	MOV	ES[addr16],	reg	263r aaaa	----	wk ← CS[PC++] ; ES[wk] ← reg
665	MOV	ES[tgt1],	imm16	34v0 nnnn	----	ES[tgt1] ← CS[PC++]
666	MOV	ES[tgt1],	reg	24vr	----	ES[tgt1] ← reg
667	MOV	reg, CS[PC + disp16]		2F1r dddd	----	reg ← CS[CS[PC++] + PC]
668	MOV	reg, CS[addr16]		2E1r aaaa	----	reg ← CS[CS[PC++]]
669	MOV	reg, CS[src0]		2Cur	----	reg ← CS[src0]
670	MOV	reg, DS[--src]		2Bsr	----	reg ← DS[--src]
671	MOV	reg, DS[addr16]		2E2r aaaa	----	reg ← DS[CS[PC++]]
672	MOV	reg, DS[src + disp16]		2Dsr dddd	----	reg ← DS[CS[PC++] + src]
673	MOV	reg, DS[src++]		2Asr	----	reg ← DS[src++]
674	MOV	reg, DS[src]		29sr	----	reg ← DS[src]
675	MOV	reg, ES[addr16]		2E3r aaaa	----	reg ← ES[CS[PC++]]
676	MOV	reg, ES[src1]		2Cvr	----	reg ← ES[src1]
677	MOV	tgt, imm16		30t0 nnnn	----	tgt ← CS[PC++]
678	MOV	tgt, reg		20tr	----	tgt ← reg
679	MOVBLK	xS[--reg], yS[--src]		5Bsr	----	do { xS[--reg] ← yS[--src] } while (--ER0);　※ER0==0 の時は65536回になる xS はreg 番号の下位2bit から、CS / ICS / DS / ES のいずれか1つが選ばれる。 yS はsrc 番号の下位2bit から、CS / ICS / DS / ES のいずれか1つが選ばれる。
680	MOVBLK	xS[reg++], yS[src++]		5Asr	----	do { xS[reg++] ← yS[src++] } while (--ER0);　※ER0==0 の時は65536回になる xS はreg 番号の下位2bit から、CS / ICS / DS / ES のいずれか1つが選ばれる。 yS はsrc 番号の下位2bit から、CS / ICS / DS / ES のいずれか1つが選ばれる。
681	MOVMM	DS[dst], DS[src]		012C ssdd	----	DS[dst] ← DS[src]　※1ワード分　dst, src は 0x0000～0x00FF の範囲のみ

#	オペレーション	オペランド	機械語 (16進数)	フラグ	挙動
682	MOVMM	DS[dst], ES[src]	012D ssdd	----	DS[dst] ← ES[src] ※1ワード分 dst, src は 0x0000～0x00FF の範囲のみ
683	MOVMM	ES[dst], DS[src]	012E ssdd	----	ES[dst] ← DS[src] ※1ワード分 dst, src は 0x0000～0x00FF の範囲のみ
684	MOVMM	ES[dst], ES[src]	012F ssdd	----	ES[dst] ← ES[src] ※1ワード分 dst, src は 0x0000～0x00FF の範囲のみ
685	MOVQ	DS[tgt], sint8	0124 nntt	----	DS[tgt] ← sint8 ※符号拡張される tgt は 0x0000～0x00FF の範囲のみ
686	MOVQ	ES[tgt], sint8	0125 nntt	----	ES[tgt] ← sint8 ※符号拡張される tgt は 0x0000～0x00FF の範囲のみ
687	MOVQ	reg, -imm4m	28nr	----	reg ← -SignExt (imm4m) ※imm4m の範囲は -1 ～-16 ※ただし埋め込まれるのは ~imm4m した値 (0x0～0xF)
688	MOVQ	reg, -imm4p	38nr	----	reg ← -SignExt (imm4p) ※imm4p の範囲は 0 ～+15
689	MOVSX	reg, src	0150 DFsr	S0Z0	符号拡張:reg ← 0xFFCD ※src==0xABCD の場合
690	MOVZX	reg, src	0150 D7sr	00Z0	ゼロ拡張:reg ← 0x00CD ※src==0xABCD の場合
691	MUL16	reg, src	0Esr	----	符号無し16bit乗算: exreg_reg ← reg * src ※積の上位16bitは、reg と同じ番号の exreg に代入する
692	MUL8	reg, CS[PC + disp16]	FF1r dddd	----	符号無し8bit乗算: reg ← LOW8U(reg) * LOW8U(CS[CS[PC++] + PC])
693	MUL8	reg, CS[addr16]	FE1r aaaa	----	符号無し8bit乗算: reg ← LOW8U(reg) * LOW8U(CS[CS[PC++]])
694	MUL8	reg, CS[src0]	FCur	----	符号無し8bit乗算: reg ← LOW8U(reg) * LOW8U(CS[src0])
695	MUL8	reg, DS[--src]	FBsr	----	符号無し8bit乗算: reg ← LOW8U(reg) * LOW8U(DS[--src])
696	MUL8	reg, DS[addr16]	FE2r aaaa	----	符号無し8bit乗算: reg ← LOW8U(reg) * LOW8U(DS[CS[PC++]])
697	MUL8	reg, DS[src + disp16]	FDsr dddd	----	符号無し8bit乗算: reg ← LOW8U(reg) * LOW8U(DS[CS[PC++] + src])
698	MUL8	reg, DS[src++]	FAsr	----	符号無し8bit乗算: reg ← LOW8U(reg) * LOW8U(DS[src++])
699	MUL8	reg, DS[src]	F9sr	----	符号無し8bit乗算: reg ← LOW8U(reg) * LOW8U(DS[src])
700	MUL8	reg, ES[addr16]	FE3r aaaa	----	符号無し8bit乗算: reg ← LOW8U(reg) * LOW8U(ES[CS[PC++]])
701	MUL8	reg, ES[src1]	FCvr	----	符号無し8bit乗算: reg ← LOW8U(reg) * LOW8U(ES[src1])
702	MUL8	reg, imm8u	FE0r 00nn	----	符号無し8bit乗算: reg ← LOW8U(reg) * LOW8U(CS[PC++])
703	MUL8	reg, src	F8sr	----	符号無し8bit乗算: reg ← LOW8U(reg) * LOW8U(src)
704	NEG	CS[PC + disp16]	671B dddd	SVZB	wk ← CS[PC++] + PC ; CS[wk] ← 0 - CS[wk]
705	NEG	CS[addr16]	661B aaaa	SVZB	wk ← CS[PC++] ; CS[wk] ← 0 - CS[wk]
706	NEG	CS[tgt0]	64uB	SVZB	wk ← tgt0 ; CS[wk] ← 0 - CS[wk]
707	NEG	DS[--tgt]	63tB	SVZB	wk ← --tgt ; DS[wk] ← 0 - DS[wk]
708	NEG	DS[addr16]	662B aaaa	SVZB	wk ← CS[PC++] ; DS[wk] ← 0 - DS[wk]
709	NEG	DS[tgt + disp16]	65tB dddd	SVZB	wk ← CS[PC++] + tgt ; DS[wk] ← 0 - DS[wk]
710	NEG	DS[tgt++]	62tB	SVZB	wk ← tgt++ ; DS[wk] ← 0 - DS[wk]
711	NEG	DS[tgt]	61tB	SVZB	wk ← tgt ; DS[wk] ← 0 - DS[wk]
712	NEG	ES[addr16]	663B aaaa	SVZB	wk ← CS[PC++] ; ES[wk] ← 0 - ES[wk]
713	NEG	ES[tgt1]	64vB	SVZB	wk ← tgt1 ; ES[wk] ← 0 - ES[wk]
714	NEG	tgt	60tB	SVZB	tgt ← 0 - tgt
715	NEGB	CS[PC + disp16]	6713 dddd	SVZB	wk ← CS[PC++] + PC ; CS[wk] ← 0 - CS[wk] - BF
716	NEGB	CS[addr16]	6613 aaaa	SVZB	wk ← CS[PC++] ; CS[wk] ← 0 - CS[wk] - BF
717	NEGB	CS[tgt0]	64u3	SVZB	wk ← tgt0 ; CS[wk] ← 0 - CS[wk] - BF
718	NEGB	DS[--tgt]	63t3	SVZB	wk ← --tgt ; DS[wk] ← 0 - DS[wk] - BF
719	NEGB	DS[addr16]	6623 aaaa	SVZB	wk ← CS[PC++] ; DS[wk] ← 0 - DS[wk] - BF
720	NEGB	DS[tgt + disp16]	65t3 dddd	SVZB	wk ← CS[PC++] + tgt ; DS[wk] ← 0 - DS[wk] - BF
721	NEGB	DS[tgt++]	62t3	SVZB	wk ← tgt++ ; DS[wk] ← 0 - DS[wk] - BF

#	オペレーション	オペランド	機械語 (16進数)	フラグ	挙動
722	NEGB	DS[tgt]	61t3	SVZB	wk ← tgt ; DS[wk] ← 0 - DS[wk] - BF
723	NEGB	ES[addr16]	6633 aaaa	SVZB	wk ← CS[PC++] ; ES[wk] ← 0 - ES[wk] - BF
724	NEGB	ES[tgt1]	64v3	SVZB	wk ← tgt1 ; ES[wk] ← 0 - ES[wk] - BF
725	NEGB	tgt	60t3	SVZB	tgt ← 0 - tgt - BF
726	NOP		0000	----	何もしない
727	NOT	CS[PC + disp16]	6716 dddd	S0Z0	wk ← CS[PC++] + PC ; CS[wk] ← ~CS[wk]
728	NOT	CS[addr16]	6616 aaaa	S0Z0	wk ← CS[PC++] ; CS[wk] ← ~CS[wk]
729	NOT	CS[tgt0]	64u6	S0Z0	wk ← tgt0 ; CS[wk] ← ~CS[wk]
730	NOT	DS[--tgt]	63t6	S0Z0	wk ← --tgt ; DS[wk] ← ~DS[wk]
731	NOT	DS[addr16]	6626 aaaa	S0Z0	wk ← CS[PC++] ; DS[wk] ← ~DS[wk]
732	NOT	DS[tgt + disp16]	65t6 dddd	S0Z0	wk ← CS[PC++] + tgt ; DS[wk] ← ~DS[wk]
733	NOT	DS[tgt++]	62t6	S0Z0	wk ← tgt++ ; DS[wk] ← ~DS[wk]
734	NOT	DS[tgt]	61t6	S0Z0	wk ← tgt ; DS[wk] ← ~DS[wk]
735	NOT	ES[addr16]	6636 aaaa	S0Z0	wk ← CS[PC++] ; ES[wk] ← ~ES[wk]
736	NOT	ES[tgt1]	64v6	S0Z0	wk ← tgt1 ; ES[wk] ← ~ES[wk]
737	NOT	tgt	60t6	S0Z0	tgt ← ~tgt
738	NPOW2	reg, src	0151 D8sr	S0Z0	reg ← ~ArithShiftLeft (1, LOW4(src)) ※2 の src 冪乗の否定
739	NSWP	reg, src	0155 D6sr	S0Z0	reg ← 0xDCBA ※src==0xABCD の場合
740	NSWP8	reg, src	0156 D6sr	S0Z0	reg ← 0xBADC ※src==0xABCD の場合
741	NSWP8SX	reg, src	0150 DEsr	S0Z0	符号拡張:reg ← 0xFFDC ※src==0xABCD の場合
742	NSWP8ZX	reg, src	0150 D6sr	00Z0	ゼロ拡張:reg ← 0x00DC ※src==0xABCD の場合
743	NSWPSX	reg, src	0152 DEsr	S0Z0	符号拡張:reg ← 0xFFBA ※src==0xABCD の場合
744	NSWPZX	reg, src	0152 D6sr	00Z0	ゼロ拡張:reg ← 0x00BA ※src==0xABCD の場合
745	NX2ASC0	reg, src	0150 51sr	0000	reg ← HEX2ASC (0x000D) ※src==0xABCD の場合
746	NX2ASC1	reg, src	0151 51sr	0000	reg ← HEX2ASC (0x000C) ※src==0xABCD の場合
747	NX2ASC2	reg, src	0152 51sr	0000	reg ← HEX2ASC (0x000B) ※src==0xABCD の場合
748	NX2ASC3	reg, src	0153 51sr	0000	reg ← HEX2ASC (0x000A) ※src==0xABCD の場合
749	NX2SSE0	reg, src	0150 52sr	?000	reg ← HEX2SSE (0x000D) ※src==0xABCD の場合 ※結果の下位8bitが7segエレメント値、上位8bitは不定値
750	NX2SSE1	reg, src	0151 52sr	?000	reg ← HEX2SSE (0x000C) ※src==0xABCD の場合 ※結果の下位8bitが7segエレメント値、上位8bitは不定値
751	NX2SSE2	reg, src	0152 52sr	?000	reg ← HEX2SSE (0x000B) ※src==0xABCD の場合 ※結果の下位8bitが7segエレメント値、上位8bitは不定値
752	NX2SSE3	reg, src	0153 52sr	?000	reg ← HEX2SSE (0x000A) ※src==0xABCD の場合 ※結果の下位8bitが7segエレメント値、上位8bitは不定値
753	NXTRCT0	reg, src	0150 50sr	00Z0	reg ← 0x000D ※src==0xABCD の場合
754	NXTRCT1	reg, src	0151 50sr	00Z0	reg ← 0x000C ※src==0xABCD の場合
755	NXTRCT2	reg, src	0152 50sr	00Z0	reg ← 0x000B ※src==0xABCD の場合
756	NXTRCT3	reg, src	0153 50sr	00Z0	reg ← 0x000A ※src==0xABCD の場合
757	OCT2BCD0A	reg, src	0151 55sr	00Z0	reg ← OCT2BCD (0x0000_0000_009A).octA ※src==0x123456789A の場合
758	OCT2BCD1A	reg, src	0153 56sr	00Z0	reg ← OCT2BCD (0x0000_0000_7800).octA ※src==0x123456789A の場合
759	OCT2BCD1B	reg, src	0153 57sr	00Z0	reg ← OCT2BCD (0x0000_0000_7800).octB ※src==0x123456789A の場合

#	オペレーション	オペランド	機械語 (16進数)	フラグ	挙動	
760	OCT2BCD2A	reg, src	0151 5Csr	00Z0	reg ← OCT2BCD (0x0000_0056_0000).octA	※src==0x123456789A の場合
761	OCT2BCD2B	reg, src	0151 5Dsr	00Z0	reg ← OCT2BCD (0x0000_0056_0000).octB	※src==0x123456789A の場合
762	OCT2BCD3A	reg, src	0153 9Csr	00Z0	reg ← OCT2BCD (0x0000_3400_0000).octA	※src==0x123456789A の場合
763	OCT2BCD3B	reg, src	0153 9Dsr	00Z0	reg ← OCT2BCD (0x0000_3400_0000).octB	※src==0x123456789A の場合
764	OCT2BCD3C	reg, src	0153 9Esr	00Z0	reg ← OCT2BCD (0x0000_3400_0000).octC	※src==0x123456789A の場合
765	OCT2BCD4A	reg, src	0151 DCsr	00Z0	reg ← OCT2BCD (0x0012_0000_0000).octA	※src==0x123456789A の場合
766	OCT2BCD4B	reg, src	0151 DDsr	00Z0	reg ← OCT2BCD (0x0012_0000_0000).octB	※src==0x123456789A の場合
767	OCT2BCD4C	reg, src	0151 DEsr	00Z0	reg ← OCT2BCD (0x0012_0000_0000).octC	※src==0x123456789A の場合
768	OCT2BCD4D	reg, src	0151 DFsr	00Z0	reg ← OCT2BCD (0x0012_0000_0000).octD	※src==0x123456789A の場合
769	OR	reg, CS[PC + disp16]	D71r dddd	S0Z0	reg ← reg | CS[CS[PC++] + PC]	
770	OR	reg, CS[addr16]	D61r aaaa	S0Z0	reg ← reg | CS[CS[PC++]]	
771	OR	reg, CS[src0]	D4ur	S0Z0	reg ← reg | CS[src0]	
772	OR	reg, DS[--src]	D3sr	S0Z0	reg ← reg | DS[--src]	
773	OR	reg, DS[addr16]	D62r aaaa	S0Z0	reg ← reg | DS[CS[PC++]]	
774	OR	reg, DS[src + disp16]	D5sr dddd	S0Z0	reg ← reg | DS[CS[PC++] + src]	
775	OR	reg, DS[src++]	D2sr	S0Z0	reg ← reg | DS[src++]	
776	OR	reg, DS[src]	D1sr	S0Z0	reg ← reg | DS[src]	
777	OR	reg, ES[addr16]	D63r aaaa	S0Z0	reg ← reg | ES[CS[PC++]]	
778	OR	reg, ES[src1]	D4vr	S0Z0	reg ← reg | ES[src1]	
779	OR	reg, imm16	D60r nnnn	S0Z0	reg ← reg | CS[PC++]	
780	OR	reg, src	D0sr	S0Z0	reg ← reg | src	
781	OUTP	PIO[port3], imm8	4pnn	----	OUTP[port3] ← imm8 ※Port#0 ～Port#7のみ (Port#8 ～#15 は OUTPn reg を使用する事)	
782	OUTP	PIO[port4], reg	04pr	----	OUTP[port4] ← LOW8(reg)	※reg の上位8bit は無視される
783	OUTP	PIO[tgt], reg	05tr	----	OUTP[LOW4(tgt)] ← LOW8(reg)	※reg の上位8bit は無視される
784	OUTP0	imm8	40nn	----	OUTP[0] ← imm8	※キャラLCD ※別名、LCDOUTP
785	OUTP0	reg	040r	----	OUTP[0] ← LOW8(reg)	※キャラLCD ※別名、LCDOUTP
786	OUTP1	imm8	41nn	----	OUTP[1] ← imm8	
787	OUTP1	reg	041r	----	OUTP[1] ← LOW8(reg)	
788	OUTP2	imm8	42nn	----	OUTP[2] ← imm8	
789	OUTP2	reg	042r	----	OUTP[2] ← LOW8(reg)	
790	OUTP3	imm8	43nn	----	OUTP[3] ← imm8	
791	OUTP3	reg	043r	----	OUTP[3] ← LOW8(reg)	
792	OUTP4	imm8	44nn	----	OUTP[4] ← imm8	※別名、PWM0OUTP
793	OUTP4	reg	044r	----	OUTP[4] ← LOW8(reg)	※別名、PWM0OUTP
794	OUTP5	imm8	45nn	----	OUTP[5] ← imm8	※別名、PWM1OUTP
795	OUTP5	reg	045r	----	OUTP[5] ← LOW8(reg)	※別名、PWM1OUTP
796	OUTP6	imm8	46nn	----	OUTP[6] ← imm8	※別名、I2COUTP
797	OUTP6	reg	046r	----	OUTP[6] ← LOW8(reg)	※別名、I2COUTP
798	OUTP7	imm8	47nn	----	OUTP[7] ← imm8	※別名、SPIOUTP
799	OUTP7	reg	047r	----	OUTP[7] ← LOW8(reg)	※別名、SPIOUTP

#	オペレーション	オペランド	機械語 (16進数)	フラグ	挙動
800	OUTP8	reg	048r	----	OUTP[8] ← LOW8(reg)
801	OUTP9	reg	049r	----	OUTP[9] ← LOW8(reg)
802	OUTP10	reg	04Ar	----	OUTP[10] ← LOW8(reg)
803	OUTP11	reg	04Br	----	OUTP[11] ← LOW8(reg)
804	OUTP12	reg	04Cr	----	OUTP[12] ← LOW8(reg)
805	OUTP13	reg	04Dr	----	OUTP[13] ← LOW8(reg)
806	OUTP14	reg	04Er	----	OUTP[14] ← LOW8(reg)
807	OUTP15	reg	04Fr	----	OUTP[15] ← LOW8(reg)
808	PACKHH	reg, srcH, srcL	015C 3rhl	----	reg.H ← srcH.H ; reg.L ← srcL.H
809	PACKHL	reg, srcH, srcL	015C 2rhl	----	reg.H ← srcH.H ; reg.L ← srcL.L
810	PACKLH	reg, srcH, srcL	015C 1rhl	----	reg.H ← srcH.L ; reg.L ← srcL.H
811	PACKLL	reg, srcH, srcL	015C 0rhl	----	reg.H ← srcH.L ; reg.L ← srcL.L
812	PADINP	reg	0C0r	----	LOW8(reg) ← INP[0] ※Joypad or Push SWs
813	POP	reg	2A0r	----	reg ← DS[SP++] ※ MOV reg, DS[SP++] の別名
814	POPCNT	reg, src	0157 9Fsr	00Z0	reg ← POPCNT(src) ※Population Count: '1' の個数をカウント(0 〜16)
815	POPF		012A	復帰	FLG ← DS[SP++]
816	POW2	reg, src	0150 D8sr	S0Z0	reg ← ArithShiftLeft (1, LOW4(src)) ※2 の src 冪乗
817	PSUB	reg, CS[PC + disp16]	B71r dddd	SVZB	wk ← CS[CS[PC++] + PC] ; if (reg >= wk) then { reg ← reg – wk } ※引けなかった時でもフラグは変化
818	PSUB	reg, CS[addr16]	B61r aaaa	SVZB	wk ← CS[CS[PC++]] ; if (reg >= wk) then { reg ← reg – wk } ※引けなかった時でもフラグは変化
819	PSUB	reg, CS[src0]	B4ur	SVZB	wk ← CS[src0] ; if (reg >= wk) then { reg ← reg – wk } ※引けなかった時でもフラグは変化
820	PSUB	reg, DS[--src]	B3sr	SVZB	wk ← DS[--src] ; if (reg >= wk) then { reg ← reg – wk } ※引けなかった時でもフラグは変化
821	PSUB	reg, DS[addr16]	B62r aaaa	SVZB	wk ← DS[CS[PC++]] ; if (reg >= wk) then { reg ← reg – wk } ※引けなかった時でもフラグは変化
822	PSUB	reg, DS[src + disp16]	B5sr dddd	SVZB	wk ← DS[CS[PC++] + src] ; if (reg >= wk) then { reg ← reg – wk } ※引けなかった時でもフラグは変化
823	PSUB	reg, DS[src++]	B2sr	SVZB	wk ← DS[src++] ; if (reg >= wk) then { reg ← reg – wk } ※引けなかった時でもフラグは変化
824	PSUB	reg, DS[src]	B1sr	SVZB	wk ← DS[src] ; if (reg >= wk) then { reg ← reg – wk } ※引けなかった時でもフラグは変化
825	PSUB	reg, ES[addr16]	B63r aaaa	SVZB	wk ← ES[CS[PC++]] ; if (reg >= wk) then { reg ← reg – wk } ※引けなかった時でもフラグは変化
826	PSUB	reg, ES[src1]	B4vr	SVZB	wk ← ES[src1] ; if (reg >= wk) then { reg ← reg – wk } ※引けなかった時でもフラグは変化
827	PSUB	reg, PC + disp16	B70r dddd	SVZB	wk ← CS[PC++] + PC ; if (reg >= wk) then { reg ← reg – wk } ※引けなかった時でもフラグは変化
828	PSUB	reg, imm16	B60r nnnn	SVZB	wk ← CS[PC++] ; if (reg >= wk) then { reg ← reg – wk } ※引けなかった時でもフラグは変化
829	PSUB	reg, src	B0sr	SVZB	wk ← src ; if (reg >= wk) then { reg ← reg – wk } ※引けなかった時でもフラグは変化
830	PULSEP0	1	0108	----	PULSEP0 端子に負論理パルスを 1 回出力する
831	PULSEP0	2	0148	----	PULSEP0 端子に負論理パルスを 2 回出力する

#	オペレーション	オペランド	機械語 (16進数)	フラグ	挙動
832	PULSEP0	4	0188	----	PULSEP0 端子に負論理パルスを 4 回出力する
833	PULSEP0	8	01C8	----	PULSEP0 端子に負論理パルスを 8 回出力する
834	PULSEP1	1	0109	----	PULSEP1 端子に負論理パルスを 1 回出力する
835	PULSEP1	2	0149	----	PULSEP1 端子に負論理パルスを 2 回出力する
836	PULSEP1	4	0189	----	PULSEP1 端子に負論理パルスを 4 回出力する
837	PULSEP1	8	01C9	----	PULSEP1 端子に負論理パルスを 8 回出力する
838	PULSEP2	1	010A	----	PULSEP2 端子に負論理パルスを 1 回出力する
839	PULSEP2	2	014A	----	PULSEP2 端子に負論理パルスを 2 回出力する
840	PULSEP2	4	018A	----	PULSEP2 端子に負論理パルスを 4 回出力する
841	PULSEP2	8	01CA	----	PULSEP2 端子に負論理パルスを 8 回出力する
842	PULSEP3	1	010B	----	PULSEP3 端子に負論理パルスを 1 回出力する
843	PULSEP3	2	014B	----	PULSEP3 端子に負論理パルスを 2 回出力する
844	PULSEP3	4	018B	----	PULSEP3 端子に負論理パルスを 4 回出力する
845	PULSEP3	8	01CB	----	PULSEP3 端子に負論理パルスを 8 回出力する
846	PULSEP4	1	010C	----	PULSEP4 端子に負論理パルスを 1 回出力する
847	PULSEP4	2	014C	----	PULSEP4 端子に負論理パルスを 2 回出力する
848	PULSEP4	4	018C	----	PULSEP4 端子に負論理パルスを 4 回出力する
849	PULSEP4	8	01CC	----	PULSEP4 端子に負論理パルスを 8 回出力する
850	PULSEP5	1	010D	----	PULSEP5 端子に負論理パルスを 1 回出力する
851	PULSEP5	2	014D	----	PULSEP5 端子に負論理パルスを 2 回出力する
852	PULSEP5	4	018D	----	PULSEP5 端子に負論理パルスを 4 回出力する
853	PULSEP5	8	01CD	----	PULSEP5 端子に負論理パルスを 8 回出力する
854	PULSEP6	1	010E	----	PULSEP6 端子に負論理パルスを 1 回出力する
855	PULSEP6	2	014E	----	PULSEP6 端子に負論理パルスを 2 回出力する
856	PULSEP6	4	018E	----	PULSEP6 端子に負論理パルスを 4 回出力する
857	PULSEP6	8	01CE	----	PULSEP6 端子に負論理パルスを 8 回出力する
858	PULSEP7	1	010F	----	PULSEP7 端子に負論理パルスを 1 回出力する
859	PULSEP7	2	014F	----	PULSEP7 端子に負論理パルスを 2 回出力する
860	PULSEP7	4	018F	----	PULSEP7 端子に負論理パルスを 4 回出力する
861	PULSEP7	8	01CF	----	PULSEP7 端子に負論理パルスを 8 回出力する
862	PUSH	imm16	3300 nnnn	----	DS[--SP] ← CS[PC++] ※ MOV DS[--SP], imm16 の別名
863	PUSH	reg	230r	----	DS[--SP] ← reg ※ MOV DS[--SP], reg の別名
864	PUSHF		0123	----	DS[--SP] ← FLG
865	PWM0OUTP	imm8	44nn	----	OUTP[4] ← imm8 ※PWM用出力Reg, PWM機能不使用時は唯の出力port
866	PWM0OUTP	reg	044r	----	OUTP[4] ← LOW8(reg) ※PWM用出力Reg, PWM機能不使用時は唯の出力port
867	PWM1OUTP	imm8	45nn	----	OUTP[5] ← imm8 ※PWM用出力Reg, PWM機能不使用時は唯の出力port
868	PWM1OUTP	reg	045r	----	OUTP[5] ← LOW8(reg) ※PWM用出力Reg, PWM機能不使用時は唯の出力port
869	RET		4A00	----	PC ← DS[SP++] ※ JMP DS[SP++] の別名
870	RLC	CS[PC + disp16]	6714 dddd	S0ZC	wk ← CS[PC++] + PC ; CS[wk] ← RotateLeftWithCF (CS[wk], 1)
871	RLC	CS[addr16]	6614 aaaa	S0ZC	wk ← CS[PC++] ; CS[wk] ← RotateLeftWithCF (CS[wk], 1)
872	RLC	CS[tgt0]	64u4	S0ZC	wk ← tgt0 ; CS[wk] ← RotateLeftWithCF (CS[wk], 1)

#	オペレーション	オペランド	機械語 (16進数)	フラグ	挙動
873	RLC	DS[--tgt]	63t4	S0ZC	wk ← --tgt ; DS[wk] ← RotateLeftWithCF (DS[wk], 1)
874	RLC	DS[addr16]	6624 aaaa	S0ZC	wk ← CS[PC++] ; DS[wk] ← RotateLeftWithCF (DS[wk], 1)
875	RLC	DS[tgt + disp16]	65t4 dddd	S0ZC	wk ← CS[PC++] + tgt ; DS[wk] ← RotateLeftWithCF (DS[wk], 1)
876	RLC	DS[tgt++]	62t4	S0ZC	wk ← tgt++ ; DS[wk] ← RotateLeftWithCF (DS[wk], 1)
877	RLC	DS[tgt]	61t4	S0ZC	wk ← tgt ; DS[wk] ← RotateLeftWithCF (DS[wk], 1)
878	RLC	ES[addr16]	6634 aaaa	S0ZC	wk ← CS[PC++] ; ES[wk] ← RotateLeftWithCF (ES[wk], 1)
879	RLC	ES[tgt1]	64v4	S0ZC	wk ← tgt1 ; ES[wk] ← RotateLeftWithCF (ES[wk], 1)
880	RLC	tgt	60t4	S0ZC	tgt ← RotateLeftWithCF (tgt, 1)
881	RLM	reg, src, imm4	0156 4nsr	S0Z0	reg ← RotateLeft (src, imm4) ※VFは常に0、CFも常に0
882	RLM	reg, src, qty	01Dq 40sr	S0Z0	reg ← RotateLeft (src, LOW4(qty)) ※VFは常に0、CFも常に0
883	RRC	CS[PC + disp16]	6712 dddd	S0ZC	wk ← CS[PC++] + PC ; CS[wk] ← RotateRightWithCF (CS[wk], 1)
884	RRC	CS[addr16]	6612 aaaa	S0ZC	wk ← CS[PC++] ; CS[wk] ← RotateRightWithCF (CS[wk], 1)
885	RRC	CS[tgt0]	64u2	S0ZC	wk ← tgt0 ; CS[wk] ← RotateRightWithCF (CS[wk], 1)
886	RRC	DS[--tgt]	63t2	S0ZC	wk ← --tgt ; DS[wk] ← RotateRightWithCF (DS[wk], 1)
887	RRC	DS[addr16]	6622 aaaa	S0ZC	wk ← CS[PC++] ; DS[wk] ← RotateRightWithCF (DS[wk], 1)
888	RRC	DS[tgt + disp16]	65t2 dddd	S0ZC	wk ← CS[PC++] + tgt ; DS[wk] ← RotateRightWithCF (DS[wk], 1)
889	RRC	DS[tgt++]	62t2	S0ZC	wk ← tgt++ ; DS[wk] ← RotateRightWithCF (DS[wk], 1)
890	RRC	DS[tgt]	61t2	S0ZC	wk ← tgt ; DS[wk] ← RotateRightWithCF (DS[wk], 1)
891	RRC	ES[addr16]	6632 aaaa	S0ZC	wk ← CS[PC++] ; ES[wk] ← RotateRightWithCF (ES[wk], 1)
892	RRC	ES[tgt1]	64v2	S0ZC	wk ← tgt1 ; ES[wk] ← RotateRightWithCF (ES[wk], 1)
893	RRC	tgt	60t2	S0ZC	tgt ← RotateRightWithCF (tgt , 1)
894	RRM	reg, src, imm4	0156 2nsr	S0Z0	reg ← RotateRight (src, imm4) ※VFは常に0、CFも常に0
895	RRM	reg, src, qty	01Dq 20sr	S0Z0	reg ← RotateRight (src, LOW4(qty)) ※VFは常に0、CFも常に0
896	RVG1SEL		07A2	----	SYS[5] ← 0b10 ※RegView Group#1 Selected ※SYSCTL SYS[5], imm2 の別名
897	RVG2SEL		07A3	----	SYS[5] ← 0b11 ※RegView Group#2 Selected ※SYSCTL SYS[5], imm2 の別名
898	RVHWSEL		07A0	----	SYS[5] ← 0b0x ※RegView H/W Selected ※SYSCTL SYS[5], imm2 の別名
899	RVSEL	reg	06Ar	----	SYS[5] ← LOW2(reg) ※RegView Select bit pattern ※SYSCTL SYS[5], reg の別名
900	RVWAITL14		01C5	----	RegView の Line#14 を検出するまで待つ ※RegView 無しの時は無限に待ってしまうので要注意
901	SETB	reg	0F9r	----	if (BF == 1) then { reg ← 0xFFFF } else { reg ← 0x0000 } ※ボロー有り
902	SETC	reg	0F8r	----	if (CF == 1) then { reg ← 0xFFFF } else { reg ← 0x0000 } ※キャリー有り
903	SETEQ	reg	0F3r	----	if (ZF == 1) then { reg ← 0xFFFF } else { reg ← 0x0000 } ※＝
904	SETNB	reg	0F8r	----	if (BF == 0) then { reg ← 0xFFFF } else { reg ← 0x0000 } ※ボロー無し
905	SETNC	reg	0F9r	----	if (CF == 0) then { reg ← 0xFFFF } else { reg ← 0x0000 } ※キャリー無し
906	SETNEQ	reg	0F2r	----	if (ZF == 0) then { reg ← 0xFFFF } else { reg ← 0x0000 } ※≠
907	SETNS	reg	0F6r	----	if (SF == 0) then { reg ← 0xFFFF } else { reg ← 0x0000 } ※正(＋)
908	SETNV	reg	0F4r	----	if (VF == 0) then { reg ← 0xFFFF } else { reg ← 0x0000 } ※Overflow無し
909	SETNZ	reg	0F2r	----	if (ZF == 0) then { reg ← 0xFFFF } else { reg ← 0x0000 } ※≠
910	SETS	reg	0F7r	----	if (SF == 1) then { reg ← 0xFFFF } else { reg ← 0x0000 } ※負(－)
911	SETSGE	reg	0FCr	----	if (SF == VF) then { reg ← 0xFFFF } else { reg ← 0x0000 } ※符号有り≧
912	SETSGT	reg	0FEr	----	if ((SF == VF) && (ZF == 0)) then { reg ← 0xFFFF } else { reg ← 0x0000 } ※符号有り＞

#	オペレーション	オペランド	機械語 (16進数)	フラグ	挙動
913	SETSLE	reg	0FFr	----	if ((SF != VF) || (ZF == 1)) then { reg ← 0xFFFF } else { reg ← 0x0000 } ※符号有り≦
914	SETSLT	reg	0FDr	----	if (SF != VF) then { reg ← 0xFFFF } else { reg ← 0x0000 } ※符号有り＜
915	SETUGE	reg	0F8r	----	if (BF == 0) then { reg ← 0xFFFF } else { reg ← 0x0000 } ※符号無し≧
916	SETUGT	reg	0FAr	----	if ((BF == 0) && (ZF == 0)) then { reg ← 0xFFFF } else { reg ← 0x0000 } ※符号無し＞
917	SETULE	reg	0FBr	----	if ((BF == 1) || (ZF == 1)) then { reg ← 0xFFFF } else { reg ← 0x0000 } ※符号無し≦
918	SETULT	reg	0F9r	----	if (BF == 1) then { reg ← 0xFFFF } else { reg ← 0x0000 } ※符号無し＜
919	SETV	reg	0F5r	----	if (VF == 1) then { reg ← 0xFFFF } else { reg ← 0x0000 } ※Overflow発生
920	SETZ	reg	0F3r	----	if (ZF == 1) then { reg ← 0xFFFF } else { reg ← 0x0000 } ※＝
921	SPIFREE		07E0	----	SYS[7] ← 0b0000 ※SPI All Slave Released ※SYSCTL SYS[7], imm4 の別名
922	SPIINP	reg	0C7r	----	LOW8(reg) ← INP[7] ※SPI用入力Reg, SPI機能不使用時は唯の入力port
923	SPIMMCSEL		07E1	----	SYS[7] ← 0b0001 ※SPI Slave#0 Select = micro MMC Select
924	SPIOUTP	imm8	47nn	----	OUTP[7] ← imm8 ※SPI用出力Reg, SPI機能不使用時は唯の出力port
925	SPIOUTP	reg	047r	----	OUTP[7] ← LOW8(reg) ※SPI用出力Reg, SPI機能不使用時は唯の出力port
926	SPIS1SEL		07E2	----	SYS[7] ← 0b0010 ※SPI Slave#1 Selected ※3.3V/5V選択可能(jumpper)
927	SPIS2SEL		07E4	----	SYS[7] ← 0b0100 ※SPI Slave#2 Selected ※SYSCTL SYS[7], imm4 の別名
928	SPIS3SEL		07E8	----	SYS[7] ← 0b1000 ※SPI Slave#3 Selected ※SYSCTL SYS[7], imm4 の別名
929	SPISEL	reg	06Er	----	SYS[7] ← LOW4(reg) ※SPI Slave#n Select bit pattern SYSCTL SYS[7], reg の別名
930	SPIXCHG		01C7	----	INP7 / OUTP7 をSPI操作(8bit交換)
931	SSVG1SEL		07C0	----	SYS[6] ← 0 ※7segView Group#1 Selected ※SYSCTL SYS[6], imm1 の別名
932	SSVG2SEL		07C1	----	SYS[6] ← 1 ※7segView Group#2 Selected ※SYSCTL SYS[6], imm1 の別名
933	SSVSEL	reg	06Cr	----	SYS[6] ← LOW1(reg) ※7segView Select bit pattern ※SYSCTL SYS[6], reg の別名
934	STFLG	reg	063r	----	LOW8(reg) ← FLG ※ reg の上位8bitは無変化
935	STNMFLG	reg	06Br	----	LOW8(reg) ← NormalModeFLG ※ reg の上位8bitは無変化 ※割込みモード時のみ
936	STNMPC	tgt	48t3	----	tgt ← NormalModePC ※割込みモード時のみ。ノーマルモード時では STPC tgt と同じ挙動。
937	STOP	(imm7)	0880 ~ 08FF	----	STOP 状態になる(割込み処理後かCont.しても処理継続しない)。imm7 省略時は0x00
938	STOP_W_IDIS		01C0	----	割込み禁止:SYS[0] ← 0 してから、STOP 状態になる。解除はReset のみ。
939	STPC	tgt	48t1	----	tgt ← PC
940	SUB	reg, CS[PC + disp16]	971r dddd	SVZB	reg ← reg - CS[CS[PC++] + PC]
941	SUB	reg, CS[addr16]	961r aaaa	SVZB	reg ← reg - CS[CS[PC++]]
942	SUB	reg, CS[src0]	94ur	SVZB	reg ← reg - CS[src0]
943	SUB	reg, DS[--src]	93sr	SVZB	reg ← reg - DS[--src]
944	SUB	reg, DS[addr16]	962r aaaa	SVZB	reg ← reg - DS[CS[PC++]]
945	SUB	reg, DS[src + disp16]	95sr dddd	SVZB	reg ← reg - DS[CS[PC++] + src]
946	SUB	reg, DS[src++]	92sr	SVZB	reg ← reg - DS[src++]
947	SUB	reg, DS[src]	91sr	SVZB	reg ← reg - DS[src]
948	SUB	reg, ES[addr16]	963r aaaa	SVZB	reg ← reg - ES[CS[PC++]]
949	SUB	reg, ES[src1]	94vr	SVZB	reg ← reg - ES[src1]
950	SUB	reg, PC + disp16	970r dddd	SVZB	reg ← reg - (CS[PC++] + PC)
951	SUB	reg, imm16	960r nnnn	SVZB	reg ← reg - CS[PC++]
952	SUB	reg, src	90sr	SVZB	reg ← reg – src

#	オペレーション	オペランド	機械語 (16進数)	フラグ	挙動
953	SUBB	reg, CS[PC + disp16]	9F1r dddd	SVZB	reg ← reg - BF - CS[CS[PC++] + PC]
954	SUBB	reg, CS[addr16]	9E1r aaaa	SVZB	reg ← reg - BF - CS[CS[PC++]]
955	SUBB	reg, CS[src0]	9Cur	SVZB	reg ← reg - BF - CS[src0]
956	SUBB	reg, DS[--src]	9Bsr	SVZB	reg ← reg - BF - DS[--src]
957	SUBB	reg, DS[addr16]	9E2r aaaa	SVZB	reg ← reg - BF - DS[CS[PC++]]
958	SUBB	reg, DS[src + disp16]	9Dsr dddd	SVZB	reg ← reg - BF - DS[CS[PC++] + src]
959	SUBB	reg, DS[src++]	9Asr	SVZB	reg ← reg - BF - DS[src++]
960	SUBB	reg, DS[src]	99sr	SVZB	reg ← reg - BF - DS[src]
961	SUBB	reg, ES[addr16]	9E3r aaaa	SVZB	reg ← reg - BF - ES[CS[PC++]]
962	SUBB	reg, ES[src1]	9Cvr	SVZB	reg ← reg - BF - ES[src1]
963	SUBB	reg, PC + disp16	9F0r dddd	SVZB	reg ← reg - BF - (CS[PC++] + PC)
964	SUBB	reg, imm16	9E0r nnnn	SVZB	reg ← reg - BF - CS[PC++]
965	SUBB	reg, src	98sr	SVZB	reg ← reg - BF – src
966	SUBQ	reg, imm4m	02nr	SVZB	reg ← reg – SignExt (imm4m) ※imm4m の範囲は -1 〜-16 ※ただし埋め込まれるのは ~imm4m した値 (0x0〜0xF)
967	SUBQ	reg, imm4p	0Anr	SVZB	reg ← reg – ZeroExt (imm4p) ※imm4p の範囲は 0 〜+15
968	SWP	reg, src	0155 D7sr	S0Z0	reg ← 0xCDAB ※src==0xABCD の場合
969	SWPSX	reg, src	0152 DFsr	S0Z0	符号拡張:reg ← 0xFFAB ※src==0xABCD の場合
970	SWPZX	reg, src	0152 D7sr	00Z0	ゼロ拡張:reg ← 0x00AB ※src==0xABCD の場合
971	SYSCOND1H2FLG		01F4	SVZC	FLG ← HIGH4 (LOW8 (SysCond1)) ※bit7(SF):WCK512, bit6(VF):WCK128, bit5(ZF):WCK32, bit4(CF):WCK8
972	SYSCOND1L2FLG		0174	SVZC	FLG ← LOW4 (SysCond1) ※bit3(SF):RegViewL14, bit2(VF):nMMC_Det#, bit1(ZF):nIdlingCond#, bit0(CF):RegViewSeg
973	SYSCOND2H2FLG		01F5	SVZC	FLG ← HIGH4 (LOW8 (SysCond2)) ※bit7(SF):未使用, bit6(VF):未使用, bit5(ZF):未使用, bit4(CF):未使用
974	SYSCOND2L2FLG		0175	SVZC	FLG ← LOW4 (SysCond2) ※bit3(SF):nVSS_ABusy#, bit2(VF):nVSS_CBusy#, bit1(ZF):nVSS_Ack#, bit0(CF):nVSS_VBI#
975	SYSCOND3H2FLG		01F6	SVZC	FLG ← HIGH4 (LOW8 (SysCond3)) ※bit7(SF):IntEnable, bit6(VF):CSBnkSel0, bit5(ZF):未使用, bit4(CF):nKBC_Notify#
976	SYSCOND3L2FLG		0176	SVZC	FLG ← LOW4 (SysCond3) ※bit3(SF):DIPSW4, bit2(VF):DIPSW3, bit1(ZF):DIPSW2, bit0(CF):DIPSW1
977	SYSCTL	SYS[port3], imm4	07pn	----	SYS[port3] ← imm4
978	SYSCTL	SYS[port3], reg	06pr	----	SYS[port3] ← LOW4 (reg)
979	TENFOLD	reg, src	0156 D9sr	S0Z0	reg ← src * 10 ※符号無し乗算, 下位16bitのみ
980	TENFOLD0A	reg, src	0150 D9sr	S0Z0	reg ← (src.L * 10).octA ※符号無し乗算
981	TENFOLD1A	reg, src	0153 D9sr	S0Z0	reg ← (src.H * 10).octA ※符号無し乗算
982	TENFOLD1B	reg, src	0152 DDsr	S0Z0	reg ← (src.H * 10).octB ※符号無し乗算
983	TOCNT	reg, src	015A 01sr	0VZ0	reg ← TOCNT(src) : Trailing Ones Count: LSB から連続する '1' の個数をカウント: 0 個ならZF←1, ALL '1' なら reg←16 かつ VF←1
984	TST	reg, CS[PC + disp16]	E71r dddd	S0Z0	0 ← reg & CS[CS[PC++] + PC]
985	TST	reg, CS[addr16]	E61r aaaa	S0Z0	0 ← reg & CS[CS[PC++]]
986	TST	reg, CS[src0]	E4ur	S0Z0	0 ← reg & CS[src0]

#	オペレーション	オペランド	機械語 (16進数)	フラグ	挙動
987	TST	reg, DS[--src]	E3sr	S0Z0	0 ← reg & DS[--src]
988	TST	reg, DS[addr16]	E62r aaaa	S0Z0	0 ← reg & DS[CS[PC++]]
989	TST	reg, DS[src + disp16]	E5sr dddd	S0Z0	0 ← reg & DS[CS[PC++] + src]
990	TST	reg, DS[src++]	E2sr	S0Z0	0 ← reg & DS[src++]
991	TST	reg, DS[src]	E1sr	S0Z0	0 ← reg & DS[src]
992	TST	reg, ES[addr16]	E63r aaaa	S0Z0	0 ← reg & ES[CS[PC++]]
993	TST	reg, ES[src1]	E4vr	S0Z0	0 ← reg & ES[src1]
994	TST	reg, imm16	E60r nnnn	S0Z0	0 ← reg & CS[PC++]
995	TST	reg, src	E0sr	S0Z0	0 ← reg & src
996	TSTN	reg, CS[PC + disp16]	EF1r dddd	S0Z0	0 ← reg & ~CS[CS[PC++] + PC]
997	TSTN	reg, CS[addr16]	EE1r aaaa	S0Z0	0 ← reg & ~CS[CS[PC++]]
998	TSTN	reg, CS[src0]	ECur	S0Z0	0 ← reg & ~CS[src0]
999	TSTN	reg, DS[--src]	EBsr	S0Z0	0 ← reg & ~DS[--src]
1000	TSTN	reg, DS[addr16]	EE2r aaaa	S0Z0	0 ← reg & ~DS[CS[PC++]]
1001	TSTN	reg, DS[src + disp16]	EDsr dddd	S0Z0	0 ← reg & ~DS[CS[PC++] + src]
1002	TSTN	reg, DS[src++]	EAsr	S0Z0	0 ← reg & ~DS[src++]
1003	TSTN	reg, DS[src]	E9sr	S0Z0	0 ← reg & ~DS[src]
1004	TSTN	reg, ES[addr16]	EE3r aaaa	S0Z0	0 ← reg & ~ES[CS[PC++]]
1005	TSTN	reg, ES[src1]	ECvr	S0Z0	0 ← reg & ~ES[src1]
1006	TSTN	reg, imm16	EE0r nnnn	S0Z0	0 ← reg & ~CS[PC++]
1007	TSTN	reg, src	E8sr	S0Z0	0 ← reg & ~src
1008	TZCNT	reg, src	015A 00sr	0VZ0	reg ← TZCNT(src) : Trailing Zeros Count: LSB から連続する '0' の個数をカウント: 0 個ならZF←1, ALL '0' なら reg←16 かつ VF←1
1009	UNPKHH	dstH, dstL, src	015D 3shl	----	dstH.H ← src.H ; dstL.H ← src.L
1010	UNPKHL	dstH, dstL, src	015D 2shl	----	dstH.H ← src.H ; dstL.L ← src.L
1011	UNPKLH	dstH, dstL, src	015D 1shl	----	dstH.L ← src.H ; dstL.H ← src.L
1012	UNPKLL	dstH, dstL, src	015D 0shl	----	dstH.L ← src.H ; dstL.L ← src.L
1013	VSSCLRACK		01B5	----	VideoSubSystem の ACK をクリアする。クリアされるまで何度も再トライする。 ※VideoSubSystem 無しの時は即終了する
1014	VSSCMDH	VSS[tgt], reg	51tr	----	垂直Blank期間(VBI)を検出するまで待ってから VSS[tgt].H ← reg.H する。 ACK が返ってくるまで何度も再トライする。 ※VideoSubSystem 無しの時は無限に繰り返すので要注意
1015	VSSCMDL	VSS[tgt], reg	50tr	----	垂直Blank期間(VBI)を検出するまで待ってから VSS[tgt].L ← reg.L する。 ACK が返ってくるまで何度も再トライする。 ※VideoSubSystem 無しの時は無限に繰り返すので要注意
1016	VSSWAITCBSY		01B4	----	VideoSubSystem が Cmd Busy の間、待つ。 ※VideoSubSystem 無しの時は即終了する
1017	VSSWAITVBI		01B6	----	VideoSubSystem の 垂直Blank期間(VBI)を検出するまで待つ ※VideoSubSystem 無しの時は無限に待ってしまうので要注意
1018	XCHG	reg, CS[PC + disp16]	F71r dddd	----	交換: reg ⇔ CS[CS[PC++] + PC]
1019	XCHG	reg, CS[addr16]	F61r aaaa	----	交換: reg ⇔ CS[CS[PC++]]
1020	XCHG	reg, CS[src0]	F4ur	----	交換: reg ⇔ CS[src0]

#	オペレーション	オペランド	機械語 (16進数)	フラグ	挙動
1021	XCHG	reg, DS[--src]	F3sr	----	交換: reg ⇔ DS[--src]
1022	XCHG	reg, DS[addr16]	F62r aaaa	----	交換: reg ⇔ DS[CS[PC++]]
1023	XCHG	reg, DS[src + disp16]	F5sr dddd	----	交換: reg ⇔ DS[CS[PC++] + src]
1024	XCHG	reg, DS[src++]	F2sr	----	交換: reg ⇔ DS[src++]
1025	XCHG	reg, DS[src]	F1sr	----	交換: reg ⇔ DS[src]
1026	XCHG	reg, ES[addr16]	F63r aaaa	----	交換: reg ⇔ ES[CS[PC++]]
1027	XCHG	reg, ES[src1]	F4vr	----	交換: reg ⇔ ES[src1]
1028	XCHG	reg, src	F0sr	----	交換: reg ⇔ src
1029	XOR	reg, CS[PC + disp16]	DF1r dddd	S0Z0	reg ← reg ^ CS[CS[PC++] + PC]
1030	XOR	reg, CS[addr16]	DE1r aaaa	S0Z0	reg ← reg ^ CS[CS[PC++]]
1031	XOR	reg, CS[src0]	DCur	S0Z0	reg ← reg ^ CS[src0]
1032	XOR	reg, DS[--src]	DBsr	S0Z0	reg ← reg ^ DS[--src]
1033	XOR	reg, DS[addr16]	DE2r aaaa	S0Z0	reg ← reg ^ DS[CS[PC++]]
1034	XOR	reg, DS[src + disp16]	DDsr dddd	S0Z0	reg ← reg ^ DS[CS[PC++] + src]
1035	XOR	reg, DS[src++]	DAsr	S0Z0	reg ← reg ^ DS[src++]
1036	XOR	reg, DS[src]	D9sr	S0Z0	reg ← reg ^ DS[src]
1037	XOR	reg, ES[addr16]	DE3r aaaa	S0Z0	reg ← reg ^ ES[CS[PC++]]
1038	XOR	reg, ES[src1]	DCvr	S0Z0	reg ← reg ^ ES[src1]
1039	XOR	reg, imm16	DE0r nnnn	S0Z0	reg ← reg ^ CS[PC++]
1040	XOR	reg, src	D8sr	S0Z0	reg ← reg ^ src
1041	ZCACLR / CLR	CS[PC + disp16]	6717 dddd	S0Z1	wk ← CS[PC++] + PC ; 0 ← CS[wk] – 0 ; CS[wk] ← 0x0000
1042	ZCACLR / CLR	CS[addr16]	6617 aaaa	S0Z1	wk ← CS[PC++] ; 0 ← CS[wk] – 0 ; CS[wk] ← 0x0000
1043	ZCACLR / CLR	CS[tgt0]	64u7	S0Z1	wk ← tgt0 ; 0 ← CS[wk] – 0 ; CS[wk] ← 0x0000
1044	ZCACLR / CLR	DS[--tgt]	63t7	S0Z1	wk ← --tgt ; 0 ← DS[wk] – 0 ; DS[wk] ← 0x0000
1045	ZCACLR / CLR	DS[addr16]	6627 aaaa	S0Z1	wk ← CS[PC++] ; 0 ← DS[wk] – 0 ; DS[wk] ← 0x0000
1046	ZCACLR / CLR	DS[tgt + disp16]	65t7 dddd	S0Z1	wk ← CS[PC++] + tgt ; 0 ← DS[wk] – 0 ; DS[wk] ← 0x0000
1047	ZCACLR / CLR	DS[tgt++]	62t7	S0Z1	wk ← tgt++ ; 0 ← DS[wk] – 0 ; DS[wk] ← 0x0000
1048	ZCACLR / CLR	DS[tgt]	61t7	S0Z1	wk ← tgt ; 0 ← DS[wk] – 0 ; DS[wk] ← 0x0000
1049	ZCACLR / CLR	ES[addr16]	6637 aaaa	S0Z1	wk ← CS[PC++] ; 0 ← ES[wk] – 0 ; ES[wk] ← 0x0000
1050	ZCACLR / CLR	ES[tgt1]	64v7	S0Z1	wk ← tgt1 ; 0 ← ES[wk] – 0 ; ES[wk] ← 0x0000
1051	ZCACLR / CLR	tgt	60t7	S0Z1	0 ← tgt – 0 ; tgt ← 0x0000
1052	ZCASET / SET	CS[PC + disp16]	671F dddd	S0Z1	wk ← CS[PC++] + PC ; 0 ← CS[wk] – 0 ; CS[wk] ← 0xFFFF
1053	ZCASET / SET	CS[addr16]	661F aaaa	S0Z1	wk ← CS[PC++] ; 0 ← CS[wk] – 0 ; CS[wk] ← 0xFFFF
1054	ZCASET / SET	CS[tgt0]	64uF	S0Z1	wk ← tgt0 ; 0 ← CS[wk] – 0 ; CS[wk] ← 0xFFFF
1055	ZCASET / SET	DS[--tgt]	63tF	S0Z1	wk ← --tgt ; 0 ← DS[wk] – 0 ; DS[wk] ← 0xFFFF
1056	ZCASET / SET	DS[addr16]	662F aaaa	S0Z1	wk ← CS[PC++] ; 0 ← DS[wk] – 0 ; DS[wk] ← 0xFFFF
1057	ZCASET / SET	DS[tgt + disp16]	65tF dddd	S0Z1	wk ← CS[PC++] + tgt ; 0 ← DS[wk] – 0 ; DS[wk] ← 0xFFFF
1058	ZCASET / SET	DS[tgt++]	62tF	S0Z1	wk ← tgt++ ; 0 ← DS[wk] – 0 ; DS[wk] ← 0xFFFF
1059	ZCASET / SET	DS[tgt]	61tF	S0Z1	wk ← tgt ; 0 ← DS[wk] – 0 ; DS[wk] ← 0xFFFF
1060	ZCASET / SET	ES[addr16]	663F aaaa	S0Z1	wk ← CS[PC++] ; 0 ← ES[wk] – 0 ; ES[wk] ← 0xFFFF
1061	ZCASET / SET	ES[tgt1]	64vF	S0Z1	wk ← tgt1 ; 0 ← ES[wk] – 0 ; ES[wk] ← 0xFFFF

#	オペレーション	オペランド	機械語 (16進数)	フラグ	挙動
1062	ZCASET / SET	tgt	60tF	S0Z1	0 ← tgt – 0 ; tgt ← 0xFFFF
1063	ZCMP	CS[PC + disp16]	671D dddd	S0Z1	wk ← CS[PC++] + PC ; 0 ← CS[wk] - 0
1064	ZCMP	CS[addr16]	661D aaaa	S0Z1	wk ← CS[PC++] ; 0 ← CS[wk] - 0
1065	ZCMP	CS[tgt0]	64uD	S0Z1	wk ← tgt0 ; 0 ← CS[wk] - 0
1066	ZCMP	DS[--tgt]	63tD	S0Z1	wk ← --tgt ; 0 ← DS[wk] - 0
1067	ZCMP	DS[addr16]	662D aaaa	S0Z1	wk ← CS[PC++] ; 0 ← DS[wk] - 0
1068	ZCMP	DS[tgt + disp16]	65tD dddd	S0Z1	wk ← CS[PC++] + tgt ; 0 ← DS[wk] - 0
1069	ZCMP	DS[tgt++]	62tD	S0Z1	wk ← tgt++ ; 0 ← DS[wk] - 0
1070	ZCMP	DS[tgt]	61tD	S0Z1	wk ← tgt ; 0 ← DS[wk] - 0
1071	ZCMP	ES[addr16]	663D aaaa	S0Z1	wk ← CS[PC++] ; 0 ← ES[wk] - 0
1072	ZCMP	ES[tgt1]	64vD	S0Z1	wk ← tgt1 ; 0 ← ES[wk] - 0
1073	ZCMP	tgt	60tD	S0Z1	0 ← tgt - 0
1074	ZCMPB	CS[PC + disp16]	6715 dddd	SVZB	wk ← CS[PC++] + PC ; 0 ← CS[wk] - 0 - BF
1075	ZCMPB	CS[addr16]	6615 aaaa	SVZB	wk ← CS[PC++] ; 0 ← CS[wk] - 0 - BF
1076	ZCMPB	CS[tgt0]	64u5	SVZB	wk ← tgt0 ; 0 ← CS[wk] - 0 - BF
1077	ZCMPB	DS[--tgt]	63t5	SVZB	wk ← --tgt ; 0 ← DS[wk] - 0 - BF
1078	ZCMPB	DS[addr16]	6625 aaaa	SVZB	wk ← CS[PC++] ; 0 ← DS[wk] - 0 - BF
1079	ZCMPB	DS[tgt + disp16]	65t5 dddd	SVZB	wk ← CS[PC++] + tgt ; 0 ← DS[wk] - 0 - BF
1080	ZCMPB	DS[tgt++]	62t5	SVZB	wk ← tgt++ ; 0 ← DS[wk] - 0 - BF
1081	ZCMPB	DS[tgt]	61t5	SVZB	wk ← tgt ; 0 ← DS[wk] - 0 - BF
1082	ZCMPB	ES[addr16]	6635 aaaa	SVZB	wk ← CS[PC++] ; 0 ← ES[wk] - 0 - BF
1083	ZCMPB	ES[tgt1]	64v5	SVZB	wk ← tgt1 ; 0 ← ES[wk] - 0 - BF
1084	ZCMPB	tgt	60t5	SVZB	0 ← tgt - 0 - BF