Appendix B
70-Series Family Op Code Listing

HEX	MNEMONIC	
00	NOP	
01	XCH	A, E
02	—	
03	—	
04	—	
05	—	
06	LD	A, S
07	LD	S, A
08	PUSH	EA
09	LD	T, EA
0A	PUSH	A
0B	LD	EA, T
0C	SR	EA
0D	DIV	EA, T
0E	SL	A
0F	SL	EA
10	CALL	0
11	CALL	1
12	CALL	2
13	CALL	3
14	CALL	4
15	CALL	5
16	CALL	6
17	CALL	7
18	CALL	8
19	CALL	9
1A	CALL	10
1B	CALL	11
1C	CALL	12
1D	CALL	13
1E	CALL	14
1F	CALL	15
40	LD	A, E
41	—	
42	—	
43	—	
44	LD	PC, EA
45	LD	SP, EA
46	LD	P2, EA
47	LD	P3, EA
48	LD	E, A
49	—	
4A	—	
4B	—	
4C	XCH	EA, PC
4D	XCH	EA, SP
4E	XCH	EA, P2
4F	XCH	EA, P3
50	AND	A, E
51	—	
52	—	
53	—	
54	PUSH	PC
55	—	
56	PUSH	P2
57	PUSH	P3
58	OR	A, E
59	—	
5A	—	
5B	—	
5C	RET	
5D	—	
5E	POP	P2
5F	POP	P3

HEX	MNEMONIC	
20	JSR	
21	—	
22	PLI	P2, = ADDR
23	PLI	P3, = ADDR
24	JMP	
25	LD	SP, = ADDR
26	LD	P2, = ADDR
27	LD	P3, = ADDR
28	—	
29	—	
2A	—	
2B	—	
2C	MPY	EA, T
2D	RND	
2E	SSM	P2
2F	SSM	P3
30	LD	EA, PC
31	LD	EA, SP
32	LD	EA, P2
33	LD	EA, P3
34	—	
35	—	
36	—	
37	—	
38	POP	A
39	AND	S, = DATA 1
3A	POP	EA
3B	OR	S, = DATA 1
3C	SR	A
3D	SRL	A
3E	RR	A
3F	RRL	A
60	XOR	A, E
61	—	
62	—	
63	—	
64	BP	
65	—	
66	BP	P2
67	BP	P3
68	—	
69	—	
6A	—	
6B	—	
6C	BZ	
6D	—	
6E	BZ	P2
6F	BZ	P3
70	ADD	A, E
71	—	
72	—	
73	—	
74	BRA	
75	—	
76	BRA	P2
77	BRA	P3
78	SUB	A, E
79	—	
7A	—	
7B	—	
7C	BNZ	
7D	—	
7E	BNZ	P2
7F	BNZ	P3

HEX	MNEMONIC	
80	LD	EA,,PC
81	LD	EA,,SP
82	LD	EA,,P2
83	LD	EA,,P3
84	LD	EA, = DATA 2
85	LD	EA,DIR ADDR
86	LD	EA,,@P2
87	LD	EA,,@P3
88	ST	EA,,PC
89	ST	EA,,SP
8A	ST	EA,,P2
8B	ST	EA,,P3
8C	—	
8D	ST	EA,DIR ADDR
8E	ST	EA,,@P2
8F	ST	EA,,@P3
90	ILD	A,,PC
91	ILD	A,,SP
92	ILD	A,,P2
93	ILD	A,,P3
94	—	
95	ILD	A,DIR ADDR
96	ILD	A,,@P2
97	ILD	A,,@P3
98	DLD	A,,PC
99	DLD	A,,SP
9A	DLD	A,,P2
9B	DLD	A,,P3
9C	—	
9D	DLD	A,DIR ADDR
9E	DLD	A,,@P2
9F	DLD	A,,@P3
C0	LD	A,, = DISPL PC
C1	LD	A,, = DISPL SP
C2	LD	A,, = DISPL P2
C3	LD	A,, = DISPL P3
C4	LD	A, = DATA 1
C5	LD	A,DIR ADDR
C6	LD	A,,@P2
C7	LD	A,,@P3
C8	ST	A,,PC
C9	ST	A,,SP
CA	ST	A,,P2
CB	ST	A,,P3
CC	—	
CD	ST	A,DIR ADDR
CE	ST	A,,@P2
CF	ST	A,,@P3
D0	AND	A,,PC
D1	AND	A,,SP
D2	AND	A,,P2
D3	AND	A,,P3
D4	AND	A, = DATA 1
D5	AND	A,DIR ADDR
D6	AND	A,,@P2
D7	AND	A,,@P3
D8	OR	A,,PC
D9	OR	A,,SP
DA	OR	A,,P2
DB	OR	A,,P3
DC	OR	D, = DATA 1
DD	OR	A, DIR ADDR
DE	OR	A,,@P2
DF	OR	A,,@P3

HEX	MNEMONIC	
A0	LD	T,,PC
A1	LD	T,,SP
A2	LD	T,,P2
A3	LD	T,,P3
A4	LD	T, = DATA 2
A5	LD	T,DIR ADDR
A6	LD	T,,@P2
A7	LD	T,,@P3
A8	—	
A9	—	
AA	—	
AB	—	
AC	—	
AD	—	
AE	—	
AF	—	
B0	ADD	EA,,PC
B1	ADD	EA,,SP
B2	ADD	EA,,P2
B3	ADD	EA,,P3
B4	ADD	EA, = DATA 2
B5	ADD	EA,DIR
B6	ADD	EA,,@P2
B7	ADD	EA,,@P3
B8	SUB	EA,,PC
B9	SUB	EA,,SP
BA	SUB	EA,,P2
BB	SUB	EA,,P3
BC	SUB	EA, = DATA 2
BD	SUB	EA,DIR ADDR
BE	SUB	EA,@P2
BF	SUB	EA,@P3
E0	XOR	A,,PC
E1	XOR	A,,SP
E2	XOR	A,,P2
E3	XOR	A,,P3
E4	XOR	A, = DATA 1
E5	XOR	A,DIR ADDR
E6	XOR	A,,@P2
E7	XOR	A,,@P3
E8	—	
E9	—	
EA	—	
EB	—	
EC	—	
ED	—	
EE	—	
EF	—	
F0	ADD	A,,PC
F1	ADD	A,,SP
F2	ADD	A,,P2
F3	ADD	A,,P3
F4	AND	A, = DATA 1
F5	ADD	A,DIR ADDR
F6	ADD	A,,@P2
F7	ADD	A,,@P3
F8	SUB	A,,PC
F9	SUB	A,,SP
FA	SUB	A,,P2
FB	SUB	A,,P3
FC	SUB	A, = DATA 1
FD	SUB	A,DIR ADDR
FE	SUB	A,,@P2
FF	SUB	A,,@P3

NOTES:
1. **ADDR = 2 bytes of address**
2. DATA 1 = 1 byte of data
3. DATA 2 = 2 bytes of data
4. DISPL = 1 byte (-128_D to $+127_D$)
5. DIR ADDR = 1 byte that represents lower byte of direct address. Upper byte of direct address is fixed at address X'FF

Appendix A

70-Series Family Instruction Set

Appendix A contains the instruction set compatible to all the 70-Series devices described in this handbook. The instruction set is defined by function and delineates the mnemonic, operand, address and the operation performed.

INSTRUCTION	MNEMONIC	FIRST OPERAND	SECOND OPERAND	ADDRESSING																				PLAGS AFFECTED							OPERATION PERFORMED
				PC RELATIVE		POINTER RELATIVE						IMMEDIATE		DIRECT		AUTO-INDEXED				IMPLIED		INDEXED		ABSOLUTE or INDIRECT								
						SP		P2		P3						P2		P3				P2		P3									
				OP	CYW	OP	CYW	OP	CYW	OP	CYW	OP	CYW	OP	CYW	OP	CYW	OP	CYW	OP	CYW	OP	CYW	OP	CYW	CY	OV	BB	BA	F3	F2	F1	HE	
LOAD	LD	A		C0 7 2 3		C1 7 2 3		C2 7 2 3		C3 7 2 3		C4 5 2 2		C5 7 2 3		C8 9 2 3		C7 9 2 3														(A) ← (addr)
		EA		90 10 2 4		81 10 2 4		82 10 2 4		83 10 2 4		84 8 3 3		85 10 2 4		86 12 2 4		87 12 2 4														(EA) ← (addr + 1, addr)
		T		A0 10 2 4		A1 10 2 4		A2 10 2 4		A3 10 2 4		A4 8 3 3		A5 10 2 4		A6 12 2 4		A7 12 2 4														(T) ← (addr + 1, addr)
		SP										25 9 3 3																				(SP) ← (PC - 2, PC - 1)
		P2										26 9 3 3																				(P2) ← (PC - 2, PC - 1)
		P3										27 9 3 3																				(P3) ← (PC - 2, PC - 1)
		A	E																	40 4 1 1												(A) ← (E)
		E	A																	48 4 1 1												(E) ← (A)
		A	S																	06 3 1 1												(A) ← (S)
		S	A																	07 3 1 1												(S) ← (A)
		EA	PC																	30 4 1 1												(EA) ← (PC)
		EA	SP																	31 4 1 1												(EA) ← (SP)
		EA	P2																	32 4 1 1												(EA) ← (P2)
		EA	P3																	33 4 1 1												(EA) ← (P3)
		SP	EA																	45 5 1 1												(SP) ← (EA)
		P2	EA																	46 5 1 1												(P2) ← (EA)
		P3	EA																	47 5 1 1												(P3) ← (EA)
		T	EA																	09 4 1 1												(T) ← (EA)
		EA	T																	08 4 1 1												(EA) ← (T)
STORE	ST	A		C8 7 2 3		C9 7 2 3		CA 7 2 3		CB 7 2 3				CD 7 2 3		CE 9 2 3		CF 9 2 3														(A) ← (addr)
		EA		88 10 2 4		89 10 2 4		8A 10 2 4		8B 10 2 4				8D 10 2 4		8E 12 2 4		8F 12 2 4														(EA) ← (addr + 1, addr)
ADD	ADD	A		F0 7 2 3		F1 7 2 3		F2 7 2 3		F3 7 2 3		F4 5 2 2		F5 7 2 3		F8 9 2 3		F7 9 2 3														(A) ← (A) + (addr)
		A	E																	70 4 1 1												(A) ← (A) + (E)
		EA		B0 10 2 4		B1 10 2 4		B2 10 2 4		B3 10 2 4		B4 8 2 3		B5 10 2 4		B6 12 2 4		B7 12 2 4														(EA) ← (EA) + (addr + 1, addr)
SUBTRACT	SUB	A		F8 7 2 3		F9 7 2 3		FA 7 2 3		FB 7 2 3		FC 5 2 2		FD 7 2 3		FE 9 2 3		FF 9 2 3														(A) ← (A) - (addr)
		A	E																	78 4 1 1												(A) ← (A) - (E)
		EA		B8 10 2 4		B9 10 2 4		BA 10 2 4		BB 10 2 4		BC 8 3 3		BD 10 2 4		BE 12 2 4		BF 12 2 4														(EA) ← (EA) - (addr + 1, addr)
MULTIPLY	MPY	EA	T																	2C 37 1 1												(EA) ← [(EA) * (T)] 31:16, (T) ← [(EA) * (T)] 15:0
DIVIDE	DIV	EA	T																	0D 43 1 1												Quotient: (EA) ← [(EA : (T)] 15:0
AND	AND	A		D0 7 2 3		D1 7 2 3		D2 7 2 3		D3 7 2 3		D4 5 2 2		D5 7 2 3		D8 9 2 3		D7 9 2 3														(A) ← (A) (addr)
		A	E																	50 4 1 1												(A) ← (A) (E)
		S										39 5 2 2														•	•			•	•	•	•	(S) ← (S) data
OR	OR	A		D8 7 2 3		D9 7 2 3		DA 7 2 3		DB 7 2 3		DC 7 2 2		DD 7 2 3		DE 9 2 3		DF 9 2 3														(A) ← (A) (addr)
		A	E																	58 4 1 1												(A) ← (A) (E)
		S										3B 5 2 2														•	•			•	•	•	•	(S) ← (S) data
EXCLUSIVE-OR	XOR	A		E0 7 2 3		E1 7 2 3		E2 7 2 3		E3 7 2 3		E4 5 2 2		E5 7 2 3		E6 9 2 3		E7 9 2 3														(A) ← (A) (addr)
		A	E																	60 4 1 1												(A) ← (A) (E)

70-Series Family Instruction Set - Continued

INSTRUCTION	MNEMONIC	FIRST OPERAND	SECOND OPERAND	ADDRESSING																				FLAGS AFFECTED								OPERATION PERFORMED
				PC RELATIVE				POINTER RELATIVE								IMMEDIATE				DIRECT				AUTO-INDEXED				IMPLIED				INDEXED				ABSOLUTE or INDIRECT															
								SP				P2				P3								P2				P3																									
				OP	MC	BY	RW	OP	MC	BY	RW	OP	MC	BY	RW	OP	MC	BY	RW	OP	MC	BY	RW	OP	MC	BY	RW	OP	MC	BY	RW	OP	MC	BY	RW	OP	MC	BY	RW	CY	OV	SB	SA	F3	F2	F1	IE	
EXCHANGE REGISTERS	XCH	A	E																									01	5	1	1																	(A) ↔ (E)
		EA	SP																									4D	7	1	1																	(EA) ↔ (SP)
		EA	P2																									4E	7	1	1																	(EA) ↔ (P2)
		EA	P3																									4F	7	1	1																	(EA) ↔ (P3)
SHIFT RIGHT	SR	A																										3C	3	1	1																	(Ai) ← (Ai-1), ; 1 7 0 ← A7
		EA																										0C	4	1	1																	(EAi) ← (EAi-1), ; 1 15 0 ← EA15
SHIFT RIGHT WITH LINK	SRL	A																										3D	3	1	1											*							(Ai) ← (Ai-1), ; 1 7 CY L ← A7
ROTATE RIGHT	RR	A																										3E	3	1	1																	(Ai) ← (Ai-1), ; 1 7 A0 ← A7
ROTATE RIGHT WITH LINK	RRL	A																										3F	3	1	1											*							(Ai) ← (Ai-1), ; 1 7 A0 ← CY L ← A7
SHIFT LEFT	SL	A																										0E	3	1	1																	(Ai-1) ← (Ai), ; 6 0 0 ← A0
		EA																										0F	4	1	1																	(EAi-1) ← (EAi), ; 14 0 0 ← EA0
SEARCH AND SKIP IF CHARACTER MATCHED	SSM																															2E	1			2F	1											See Text
BRANCH IF NOT DIGIT	BND			2D	2																																										See Text
PUSH	PUSH	A																										0A	5	1	2																	(SP) ← (SP)-1, ((SP)) ← (A)
		EA																										08	8	1	3																	((SP)-1) ← (E), ((SP)-2) ← (A), (SP) ← (SP)-2
		PC																										54	8	1	3																	((SP)-1) ← (PCH), ((SP)-2) ← (PCL), (SP) ← (SP)-2
		P2																										56	8	1	3																	((SP)-1) ← (P2H), ((SP)-2) ← (P2L), (SP) ← (SP)-2
		P3																										57	8	1	3																	((SP)-1) ← (P3H), ((SP)-2) ← (P3L), (SP) ← (SP)-2
PUSH AND LOAD IMMEDIATE	PLI	P2														22	16	3	5																													((SP)-1) ← (P2H), ((SP)-2) ← (P2L), (SP) ← (SP)-2, (P2H) ← byte 3, (P2L) ← byte 2
		P3														23	16	3	5																													((SP)-1) ← (P3H), ((SP)-2) ← (P3L), (SP) ← (SP)-2, (P3H) ← byte 3, (P3L) ← byte 2
POP	POP	A																										38	6	1	2																	(A) ← ((SP)), (SP) ← (SP)-1
		EA																										3A	9	1	3																	(A) ← ((SP)), (E) ← ((SP)-1), (SP) ← (SP)-2
		P2																										5E	10	1	3																	(P2L) ← ((SP)), (P2H) ← ((SP)-1), (SP) ← (SP)-2
		P3																										5F	10	1	3																	(P3L) ← ((SP)), (P3H) ← ((SP)-1), (SP) ← 2
BRANCH UNCONDITIONAL	BRA			74	5	2	2					76	5	2	2	77	5	2	2																													(PC) ← (Pn) + disp
BRANCH POSITIVE	BP			64	5	2	2					66	5	2	2	67	5	2	2																													If (A) ≥ 0 (PC) ← (Pn) + disp
BRANCH ZERO	BZ			6C	5	2	2					6E	5	2	2	6F	5	2	2																													If (A) = 0 (PC) ← (Pn) + disp
BRANCH NOT ZERO	BNZ			7C	5	2	2					7E	5	2	2	7F	5	2	2																													If (A) ≠ 0 (PC) ← (Pn) + disp
JUMP UNCONDITIONAL	JMP																																							24	9	3	3									(PCH) ← byte 3, (PCL) ← byte 2
JUMP TO SUBROUTINE	JSR																																							26	10	3	3									((SP)-1) ← (PCH), byte 3, ((SP)-2) ← (PCL), byte 2, (SP) ← (SP)-2
CALL	CALL	0 - 15																																						10-1F	10	1	5									((SP)-1) ← (PCH), ← (021 - 2*N), ((SP)-2) ← (PCL), ← (020 - 2*N), (SP) ← (SP)-2
RETURN	RET																											5C	10	1	3																	(PCL) ← ((SP)), (PCH) ← ((SP)-1), (SP) ← (SP)-2
LOAD PC	LD	PC	EA																									44	5	1	1																	(PC) ← (EA)
EXCHANGE PC	XCH	PC	EA																									4C	7	1	1																	(PC) ↔ (EA)
INCREMENT AND LOAD	ILD	A		90	8	2	4	91	8	2	4	92	8	2	4	93	8	2	4					95	8	2	4	96	10	2	4	97	10	2	4													(A), (addr) ← (addr) + 1
DECREMENT AND LOAD	DLD	A		98	8	2	4	99	8	2	4	9A	8	2	4	9B	8	2	4					9D	8	2	4	9E	10	2	4	9F	10	2	4													(A), (addr) ← (addr) - 1
NO OPERATION	NOP																											00	3	1	1																	(PC) ← (PC) + 1