MICRO LOGIC CORP.
HACKENSACK, NJ

Z80 CPU

MICROPROCESSOR INSTANT REFERENCE CARD

MICRO® CHART

Example of reading instruction set tables: ADC A,A ... ADC A, - entry says to see table; table shows opcode 8F, 4 states; and flag code 'A' which is defined under 'Flag Codes'. ADC HL,BC ... 2 byte opcode is ED,4A; flag code is H; takes 15 states. CALL C, address ... opcode is DC followed by 2 byte address; flag code is Z; states are described by note 5.

Instruction Set

A	ADC	A,—	TABLE	A	L	LD	(IX+d),C	DD71d	Z19		
	ADC	HL,BC	ED4A	H15		LD	(IX+d),D	DD72d	Z19		
	ADC	HL,DE	ED5A	H15		LD	(IX+d),E	DD73d	Z19		
	ADC	HL,HL	ED6A	H15		LD	(IX+d),H	DD74d	Z19		
	ADC	HL,SP	ED7A	H15		LD	(IX+d),L	DD75d	Z19		
	ADD	A,—	TABLE	A		LD	(IX+d),n	DD36dn	Z19		
	ADD	HL,BC	09	G11		LD	(IY+d),A	FD77d	Z19		
	ADD	HL,DE	19	G11		LD	(IY+d),B	FD70d	Z19		
	ADD	HL,HL	29	G11		LD	(IY+d),C	FD71d	Z19		
	ADD	HL,SP	39	G11		LD	(IY+d),D	FD72d	Z19		
	ADD	IX,BC	DD09	G15		LD	(IY+d),E	FD73d	Z19		
	ADD	IX,DE	DD19	G15		LD	(IY+d),H	FD74d	Z19		
	ADD	IX,IX	DD29	G15		LD	(IY+d),L	FD75d	Z19		
	ADD	IX,SP	DD39	G15		LD	(IY+d),n	FD36dn	Z19		
	ADD	IY,BC	FD09	G15		LD	(aa),A	32a	Z13		
	ADD	IY,DE	FD19	G15		LD	(aa),BC	ED43a	Z20		
	ADD	IY,IY	FD29	G15		LD	(aa),DE	ED53a	Z20		
	ADD	IY,SP	FD39	G15		LD	(aa),HL	22a	Z16		
	AND	—	TABLE	C		LD	(aa),IX	DD22a	Z20		
	BIT	—	TABLE	V		LD	(aa),IY	FD22a	Z20		
	CALL	aa	CDa	Z17		LD	(aa) SP	ED73a	Z20		
	CALL	C,aa	DCa	Z(5)		LD	A,(BC)	0A	Z7		
	CALL	M,aa	FCa	Z(5)		LD	A,(DE)	1A	Z7		
	CALL	NC,aa	D4a	Z(5)		LD	A,(aa)	3Aa	Z13		
	CALL	NZ,aa	C4a	Z(5)		LD	A,I	ED57	U9		
	CALL	P,aa	F4a	Z(5)		LD	A,R	ED5F	U9		
C	CALL	PE,aa	ECa	Z(5)		LD	A,—	TABLE	Z		
	CALL	PO,aa	E4a	Z(5)		LD	B,—	TABLE	Z		
	CALL	Z,aa	CCa	Z(5)		LD	BC,(aa)	ED4Ba	Z20		
	CCF		3F	G4		LD	BC,aa	01a	Z10		
	CP	—	TABLE	B		LD	C,—	TABLE	Z		
	CPD		EDA9	T16		LD	D,—	TABLE	Z		
	CPDR		EDB9	T(1)		LD	DE,(aa)	ED5Ba	Z20		
	CPI		EDA1	T16		LD	DE,aa	11a	Z10		
	CPIR		EDB1	T(1)		LD	E,—	TABLE	Z		
	CPL		2F	N4		LD	H,—	TABLE	Z		
	DAA		27	M4		LD	HL,(aa)	2Aa	Z16		
	DEC	(HL)	35	F11		LD	HL,aa	21a	Z10		
	DEC	(IX+d)	DD35d	F23		LD	I,A	ED47	Z9		
	DEC	(IY+d)	FD35d	F23		LD	IX,(aa)	DD2Aa	Z20		
	DEC	A	3D	F4		LD	IX,aa	DD21a	Z14		
	DEC	B	05	F4		LD	IY,(aa)	FD2Aa	Z20		
	DEC	BC	0B	Z6		LD	IY,aa	FD21a	Z14		
	DEC	C	0D	F4		LD	L,—	TABLE	Z		
	DEC	D	15	F4		LD	R,A	ED4F	Z9		
D	DEC	DE	1B	Z6		LD	SP,(aa)	ED7Ba	Z20		
	DEC	E	1D	F4		LD	SP,HL	F9	Z6		
	DEC	H	25	F4		LD	SP,IX	DDF9	Z10		
	DEC	HL	2B	Z6		LD	SP,IY	FDF9	Z10		
	DEC	IX	DD2B	Z10		LD	SP,aa	31a	Z10		
	DEC	IY	FD2B	Z10		LDD		EDA8	R16		
	DEC	L	2D	F4		LDDR		EDB8	S(1)		
	DEC	SP	3B	Z6		LDI		EDA0	R16		
	DI		F3	Z4		LDIR		EDB0	S(1)		
	DJNZ	d	10d	Z(2)		NEG		ED44	B8		
	EI		FB	Z4		NOP		00	Z4		
	EX	(SP),HL	E3	Z19		OR	—	TABLE	D		
E	EX	(SP),IX	DDE3	Z23		OTDR		EDBB	Q(1)		
	EX	(SP),IY	FDE3	Z23		OTIR		EDB3	Q(1)		
	EX	AF,AF'	08	Z4		OUT	(C),A	ED79	Z12		
	EX	DE,HL	EB	Z4		OUT	(C),B	ED41	Z12		
	EXX		D9	Z4		OUT	(C),C	ED49	Z12		
	HALT		76	Z4	O	OUT	(C),D	ED51	Z12		
	IM	0	ED46	Z8		OUT	(C),E	ED59	Z12		
	IM	1	ED56	Z8		OUT	(C),H	ED61	Z12		
	IM	2	ED5E	Z8		OUT	(C),L	ED69	Z12		
	IN	A,(C)	ED78	W12		OUT	(n),A	D3n	Z11		
	IN	A,(n)	DBn	Z11		OUTD		EDAB	P16		
	IN	B,(C)	ED40	W12		OUT I		EDA3	P16		
	IN	C,(C)	ED48	W12		POP	AF	F1	Z10		
	IN	D,(C)	ED50	W12		POP	BC	C1	Z10		
	IN	E,(C)	ED58	W12		POP	DE	D1	Z10		
	IN	H,(C)	ED60	W12		POP	HL	E1	Z10		
	IN	L,(C)	ED68	W12		POP	IX	DDE1	Z14		
	INC	(HL)	34	E11	P	POP	IY	FDE1	Z14		
	INC	(IX+d)	DD34d	E23		PUSH	AF	F5	Z11		
	INC	(IY+d)	FD34d	E23		PUSH	BC	C5	Z11		
I	INC	A	3C	E4		PUSH	DE	D5	Z11		
	INC	B	04	E4		PUSH	HL	E5	Z11		
	INC	BC	03	Z6		PUSH	IX	DDE5	Z15		
	INC	C	0C	E4		PUSH	IY	FDE5	Z15		
	INC	D	14	E4		RES	—	TABLE	Z		
	INC	DE	13	Z6		RET		C9	Z10		
	INC	E	1C	E4		RET	C	D8	Z(4)		
	INC	H	24	E4		RET	M	F8	Z(4)		
	INC	HL	23	Z6		RET	NC	D0	Z(4)		
	INC	IX	DD23	Z10		RET	NZ	C0	Z(4)		
	INC	IY	FD23	Z10		RET	P	F0	Z(4)		
	INC	L	2C	E4		RET	PE	E8	Z(4)		
	INC	SP	33	Z6		RET	PO	E0	Z(4)		
	IND		EDAA	P16		RET	Z	C8	Z(4)		
	INDR		EDBA	Q(1)		RETI		ED4D	Z14		
	INI		EDA2	P16		RETN		ED45	Z14		
	INIR		EDB2	Q(1)		RL	—	TABLE	K		
	JP	(HL)	E9	Z4		RLA		17	J4		
	JP	(IX)	DDE9	Z8	R	RLC	—	TABLE	K		
	JP	(IY)	FDE9	Z8		RLCA		07	J4		
	JP	aa	C3a	Z10		RLD		ED6F	L18		
	JP	C,aa	DAa	Z10		RR	—	TABLE	K		
	JP	M,aa	FAa	Z10		RRA		1F	J4		
	JP	NC,aa	D2a	Z10		RRC	—	TABLE	K		
J	JP	NZ,aa	C2a	Z10		RRCA		0F	J4		
	JP	P,aa	F2a	Z10		RRD		ED67	L18		
	JP	PE,aa	EAa	Z10		RST	00H	C7	Z11		
	JP	PO,aa	E2a	Z10		RST	08H	CF	Z11		
	JP	Z,aa	CAa	Z10		RST	10H	D7	Z11		
	JR	C,d	38d	Z(3)		RST	18H	DF	Z11		
	JR	d	18d	Z12		RST	20H	E7	Z11		
	JR	NC,d	30d	Z(3)		RST	28H	EF	Z11		
	JR	NZ,d	20d	Z(3)		RST	30H	F7	Z11		
	JR	Z,d	28d	Z(3)		RST	38H	FF	Z11		
	LD	(BC),A	02	Z7		SBC	A,—	TABLE	B		
	LD	(DE),A	12	Z7		SBC	HL,BC	ED42	I15		
	LD	(HL),A	77	Z7		SBC	HL,DE	ED52	I15		
	LD	(HL),B	70	Z7		SBC	HL,HL	ED62	I15		
	LD	(HL),C	71	Z7		SBC	HL,SP	ED72	I15		
	LD	(HL),D	72	Z7	S	SCF		37	O4		
L	LD	(HL),E	73	Z7		SET	—	TABLE	Z		
	LD	(HL),H	74	Z7		SLA	—	TABLE	K		
	LD	(HL),L	75	Z7		SRA	—	TABLE	K		
	LD	(HL),n	36n	Z10		SRL	—	TABLE	K		
	LD	(IX+d),A	DD77d	Z19		SUB	—	TABLE	B		
	LD	(IX+d),B	DD70d	Z19		XOR	—	TABLE	D		

	A	B	C	D	E	H	L	(HL)	(IX+d)	(IY+d)	
BIT 0,	CB,47	CB,40	CB,41	CB,42	CB,43	CB,44	CB,45	CB,46	DD,CB,d,46	FD,CB,d,46	V
BIT 1,	CB,4F	CB,48	CB,49	CB,4A	CB,4B	CB,4C	CB,4D	CB,4E	DD,CB,d,4E	FD,CB,d,4E	V
BIT 2,	CB,57	CB,50	CB,51	CB,52	CB,53	CB,54	CB,55	CB,56	DD,CB,d,56	FD,CB,d,56	V
BIT 3,	CB,5F	CB,58	CB,59	CB,5A	CB,5B	CB,5C	CB,5D	CB,5E	DD,CB,d,5E	FD,CB,d,5E	V
BIT 4,	CB,67	CB,60	CB,61	CB,62	CB,63	CB,64	CB,65	CB,66	DD,CB,d,66	FD,CB,d,66	V
BIT 5,	CB,6F	CB,68	CB,69	CB,6A	CB,6B	CB,6C	CB,6D	CB,6E	DD,CB,d,6E	FD,CB,d,6E	V
BIT 6,	CB,77	CB,70	CB,71	CB,72	CB,73	CB,74	CB,75	CB,76	DD,CB,d,76	FD,CB,d,76	V
BIT 7,	CB,7F	CB,78	CB,79	CB,7A	CB,7B	CB,7C	CB,7D	CB,7E	DD,CB,d,7E	FD,CB,d,7E	V
STATES:		8						12	20		

	A	B	C	D	E	H	L	(HL)	(IX+d)	(IY+d)	
RES 0,	CB,87	CB,80	CB,81	CB,82	CB,83	CB,84	CB,85	CB,86	DD,CB,d,86	FD,CB,d,86	Z
RES 1,	CB,8F	CB,88	CB,89	CB,8A	CB,8B	CB,8C	CB,8D	CB,8E	DD,CB,d,8E	FD,CB,d,8E	Z
RES 2,	CB,97	CB,90	CB,91	CB,92	CB,93	CB,94	CB,95	CB,96	DD,CB,d,96	FD,CB,d,96	Z
RES 3,	CB,9F	CB,98	CB,99	CB,9A	CB,9B	CB,9C	CB,9D	CB,9E	DD,CB,d,9E	FD,CB,d,9E	Z
RES 4,	CB,A7	CB,A0	CB,A1	CB,A2	CB,A3	CB,A4	CB,A5	CB,A6	DD,CB,d,A6	FD,CB,d,A6	Z
RES 5,	CB,AF	CB,A8	CB,A9	CB,AA	CB,AB	CB,AC	CB,AD	CB,AE	DD,CB,d,AE	FD,CB,d,AE	Z
RES 6,	CB,B7	CB,B0	CB,B1	CB,B2	CB,B3	CB,B4	CB,B5	CB,B6	DD,CB,d,B6	FD,CB,d,B6	Z
RES 7,	CB,BF	CB,B8	CB,B9	CB,BA	CB,BB	CB,BC	CB,BD	CB,BE	DD,CB,d,BE	FD,CB,d,BE	Z
SET 0,	CB,C7	CB,C0	CB,C1	CB,C2	CB,C3	CB,C4	CB,C5	CB,C6	DD,CB,d,C6	FD,CB,d,C6	Z
SET 1,	CB,CF	CB,C8	CB,C9	CB,CA	CB,CB	CB,CC	CB,CD	CB,CE	DD,CB,d,CE	FD,CB,d,CE	Z
SET 2,	CB,D7	CB,D0	CB,D1	CB,D2	CB,D3	CB,D4	CB,D5	CB,D6	DD,CB,d,D6	FD,CB,d,D6	Z
SET 3,	CB,DF	CB,D8	CB,D9	CB,DA	CB,DB	CB,DC	CB,DD	CB,DE	DD,CB,d,DE	FD,CB,d,DE	Z
SET 4,	CB,E7	CB,E0	CB,E1	CB,E2	CB,E3	CB,E4	CB,E5	CB,E6	DD,CB,d,E6	FD,CB,d,E6	Z
SET 5,	CB,EF	CB,E8	CB,E9	CB,EA	CB,EB	CB,EC	CB,ED	CB,EE	DD,CB,d,EE	FD,CB,d,EE	Z
SET 6,	CB,F7	CB,F0	CB,F1	CB,F2	CB,F3	CB,F4	CB,F5	CB,F6	DD,CB,d,F6	FD,CB,d,F6	Z
SET 7,	CB,FF	CB,F8	CB,F9	CB,FA	CB,FB	CB,FC	CB,FD	CB,FE	DD,CB,d,FE	FD,CB,d,FE	Z
STATES:		8						15	23		

	A(8)	B	C	D	E	H	L	(HL)	(IX+d)	(IY+d)	
RLC	CB,07	CB,00	CB,01	CB,02	CB,03	CB,04	CB,05	CB,06	DD,CB,d,06	FD,CB,d,06	K
RRC	CB,0F	CB,08	CB,09	CB,0A	CB,0B	CB,0C	CB,0D	CB,0E	DD,CB,d,0E	FD,CB,d,0E	K
RL	CB,17	CB,10	CB,11	CB,12	CB,13	CB,14	CB,15	CB,16	DD,CB,d,16	FD,CB,d,16	K
RR	CB,1F	CB,18	CB,19	CB,1A	CB,1B	CB,1C	CB,1D	CB,1E	DD,CB,d,1E	FD,CB,d,1E	K
SLA	CB,27	CB,20	CB,21	CB,22	CB,23	CB,24	CB,25	CB,26	DD,CB,d,26	FD,CB,d,26	K
SRA	CB,2F	CB,28	CB,29	CB,2A	CB,2B	CB,2C	CB,2D	CB,2E	DD,CB,d,2E	FD,CB,d,2E	K
SRL	CB,3F	CB,38	CB,39	CB,3A	CB,3B	CB,3C	CB,3D	CB,3E	DD,CB,d,3E	FD,CB,d,3E	K
STATES:		8						15	23		

	A	B	C	D	E	H	L	(HL)	n	(IX+d)	(IY+d)	
ADC A,	8F	88	89	8A	8B	8C	8D	8E	CE,n	DD,8E,d	FD,8E,d	A
ADD A,	87	80	81	82	83	84	85	86	C6,n	DD,86,d	FD,86,d	A
AND	A7	A0	A1	A2	A3	A4	A5	A6	E6,n	DD,A6,d	FD,A6,d	C
CP	BF	B8	B9	BA	BB	BC	BD	BE	FE,n	DD,BE,d	FD,BE,d	B
OR	B7	B0	B1	B2	B3	B4	B5	B6	F6,n	DD,B6,d	FD,B6,d	D
SBC A,	9F	98	99	9A	9B	9C	9D	9E	DE,n	DD,9E,d	FD,9E,d	B
SUB	97	90	91	92	93	94	95	96	D6,n	DD,96,d	FD,96,d	B
XOR	AF	A8	A9	AA	AB	AC	AD	AE	EE,n	DD,AE,d	FD,AE,d	D
LD A,	7F	78	79	7A	7B	7C	7D	7E	3E,n	DD,7E,d	FD,7E,d	Z
LD B,	47	40	41	42	43	44	45	46	06,n	DD,46,d	FD,46,d	Z
LD C,	4F	48	49	4A	4B	4C	4D	4E	0E,n	DD,4E,d	FD,4E,d	Z
LD D,	57	50	51	52	53	54	55	56	16,n	DD,56,d	FD,56,d	Z
LD E,	5F	58	59	5A	5B	5C	5D	5E	1E,n	DD,5E,d	FD,5E,d	Z
LD H,	67	60	61	62	63	64	65	66	26,n	DD,66,d	FD,66,d	Z
LD L,	6F	68	69	6A	6B	6C	6D	6E	2E,n	DD,6E,d	FD,6E,d	Z
STATES:		4						7	19			

Flag Codes

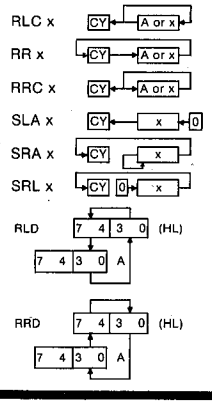
	C	Z	P/V	S	N	H	
A	C	Z	V	S	0	H	
B	C	Z	V	S	1	H	
C	0	Z	P	S	0	1	
D	0	Z	P	S	0	0	
E	=	Z	V	S	0	H	
F	=	Z	V	S	1	H	
G	C	=	=	=	0	U	
H	C	Z	V	S	0	U	
I	C	Z	V	S	1	U	
J	C	=	=	=	0	0	
K	C	Z	P	S	0	0	
L	=	Z	P	S	0	0	
M	C	Z	P	S	=	H	
N	=	=	=	=	1	1	
O	1	=	=	=	0	0	
P	U	F	U	U	1	U	(1)
Q	U	1	U	U	1	U	
R	=	U	F	U	0	0	(2)
S	=	U	0	U	0	0	
T	=	F	F	S	1	U	(3)
U	=	Z	F	S	0	0	(4)
V	=	Z	U	U	0	1	(5)
W	=	Z	P	S	0	U	
X							
Y							
Z	=	=	=	=	=	=	

Codes:
0: reset
1: set
C: Carry*
F: Footnote
H: Half carry*
N: Add/Sub*
P: Parity*
S: Sign*
U: Undefined
V: oVerflow*
Z: Zero*
=: not affected

* Indicated flag affected by result

(1)Z=1 iff B becomes 0
(2)PV=0 iff BC becomes 0
(3)PV=0 iff BC becomes 0 and Z=1 iff A=(HL)
(4)PV=IFF2
(5)Z=bit

Rotates and Shifts



Addressing

n	n is immediate 8-bit data.
aa	aa is immediate 16-bit data or address to CALL/to JP to.
(aa)	aa is address of data.
(rr)	16-bit reg rr holds address of data or address to CALL or to JP to.
(n)	n is port number.
(r)	8-bit reg r holds port number.
(IX+d)	IX+d is address of data (d is a 1 byte signed displacement).
d	In relative jumping, address to jump to is d + address of next instruction (d is signed).

Full 2 byte addresses in code, stack, and data areas are stored low byte followed by high byte. Thus JP 1234H is: C3,34,12.

SP points to used byte at top of stack. PUSH decrements SP by 2.

Notes

(1) 21 except 16 at termination
(2) 13 except 8 at termination
(3) 12 for success; 7 for failure
(4) 11 for success; 5 for failure
(5) 17 for success; 10 for failure
(6) A to A15..A8 and n to A7..A0
(7) B to A15..A8 and C to A7..A0
(8) See faster version of 'Rotate A' instructions

Intentionally Blank

DO NOT PLACE ON HOT SURFACE

AUTHOR: JAMES D LEWIS

#104A

MICRO LOGIC CORP.
HACKENSACK, NJ

Z80 CPU
MICROPROCESSOR INSTANT REFERENCE CARD

MICRO® CHART

Single-Byte-Opcode to Instruction Conversion

LSD ⇒

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0	NOP	LD BC,nn	LD (BC),A	INC BC	INC B	DEC B	LD B,n	RLCA	EX AF,AF'	ADD HL,BC	LD A,(BC)	DEC BC	INC C	DEC C	LD C,n	RRCA	0
1	DJNZ n	LD DE,nn	LD (DE),A	INC DE	INC D	DEC D	LD D,n	RLA	JR n	ADD HL,DE	LD A,(DE)	DEC DE	INC E	DEC E	LD E,n	RRA	1
2	JR NZ,n	LD HL,nn	LD (nn),HL	INC HL	INC H	DEC H	LD H,n	DAA	JR Z,n	ADD HL,HL	LD HL,(nn)	DEC HL	INC L	DEC L	LD L,n	CPL	2
3	JR NC,n	LD SP,nn	LD (nn),A	INC SP	INC (HL)	DEC (HL)	LD (HL),n	SCF	JR C,n	ADD HL,SP	LD A,(nn)	DEC SP	INC A	DEC A	LD A,n	CCF	3
4	LD B,B	LD B,C	LD B,D	LD B,E	LD B,H	LD B,L	LD B,(HL)	LD B,A	LD C,B	LD C,C	LD C,D	LD C,E	LD C,H	LD C,L	LD C,(HL)	LD C,A	4
5	LD D,B	LD D,C	LD D,D	LD D,E	LD D,H	LD D,L	LD D,(HL)	LD D,A	LD E,B	LD E,C	LD E,D	LD E,E	LD E,H	LD E,L	LD E,(HL)	LD E,A	5
6	LD H,B	LD H,C	LD H,D	LD H,E	LD H,H	LD H,L	LD H,(HL)	LD H,A	LD L,B	LD L,C	LD L,D	LD L,E	LD L,H	LD L,L	LD L,(HL)	LD L,A	6
7	LD (HL),B	LD (HL),C	LD (HL),D	LD (HL),E	LD (HL),H	LD (HL),L	HALT	LD (HL),A	LD A,B	LD A,C	LD A,D	LD A,E	LD A,H	LD A,L	LD A,(HL)	LD A,A	7
8	ADD A,B	ADD A,C	ADD A,D	ADD A,E	ADD A,H	ADD A,L	ADD A,(HL)	ADD A,A	ADC A,B	ADC A,C	ADC A,D	ADC A,E	ADC A,H	ADC A,L	ADC A,(HL)	ADC A,A	8
9	SUB B	SUB C	SUB D	SUB E	SUB H	SUB L	SUB (HL)	SUB A	SBC A,B	SBC A,C	SBC A,D	SBC A,E	SBC A,H	SBC A,L	SBC A,(HL)	SBC A,A	9
A	AND B	AND C	AND D	AND E	AND H	AND L	AND (HL)	AND A	XOR B	XOR C	XOR D	XOR E	XOR H	XOR L	XOR (HL)	XOR A	A
B	OR B	OR C	OR D	OR E	OR H	OR L	OR (HL)	OR A	CP B	CP C	CP D	CP E	CP H	CP L	CP (HL)	CP A	B
C	RET NZ	POP BC	JP NZ,nn	JP nn	CALL NZ,nn	PUSH BC	ADD A,n	RST 00H	RET Z	RET	JP Z,nn	table	CALL Z,nn	CALL nn	ADC A,n	RST 08H	C
D	RET NC	POP DE	JP NC,nn	OUT (n), A	CALL NC,nn	PUSH DE	SUB n	RST 10H	RET C	EXX	JP C,nn	IN A,(n)	CALL C,nn	table	SBC A,n	RST 18H	D
E	RET PO	POP HL	JP PO,nn	EX (SP), HL	CALL PO,nn	PUSH HL	AND n	RST 20H	RET PE	JP (HL)	JP PE,nn	EX DE,HL	CALL PE,nn	table	XOR n	RST 28H	E
F	RET P	POP AF	JP P,nn	DI	CALL P,nn	PUSH AF	OR n	RST 30H	RET M	LD SP,HL	JP M,nn	EI	CALL M,nn	table	CP n	RST 38H	F
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	

Multi-Byte-Opcode to Instruction Conversion

CB00	RLC B	ED40	IN B,(C)	%%09	ADD XY,BC	%%CBd06	RLC (XY+d)
CB01	RLC C	ED41	OUT (C),B	%%19	ADD XY,DE	%%CBd0E	RRC (XY+d)
CB02	RLC D	ED42	SBC HL,BC	%%21aa	LD XY,aa	%%CBd16	RL (XY+d)
CB03	RLC E	ED43aa	LD (aa),BC	%%22aa	LD (aa),XY	%%CBd1E	RR (XY+d)
CB04	RLC H	ED44	NEG	%%23	INC XY	%%CBd26	SLA (XY+d)
CB05	RLC L	ED45	RETN	%%29	ADD XY,XY	%%CBd2E	SRA (XY+d)
CB06	RLC (HL)	ED46	IM 0	%%2Aaa	LD XY,(aa)	%%CBd3E	SRL (XY+d)
CB07	RLC A	ED47	LD I,A	%%2B	DEC XY	%%CBd46	BIT 0,(XY+d)
CB08	RRC B	ED48	IN C,(C)	%%34d	INC (XY+d)	%%CBd4E	BIT 1,(XY+d)
CB09	RRC C	ED49	OUT (C),C	%%35d	DEC (XY+d)	%%CBd56	BIT 2,(XY+d)
CB0A	RRC D	ED4A	ADC HL,BC	%%36dn	LD (XY+d),n	%%CBd5E	BIT 3,(XY+d)
CB0B	RRC E	ED4Baa	LD BC,(aa)	%%39	ADD XY,SP	%%CBd66	BIT 4,(XY+d)
CB0C	RRC H	ED4D	RETI	%%46d	LD B,(XY+d)	%%CBd6E	BIT 5,(XY+d)
CB0D	RRC L	ED4F	LD R,A	%%4Ed	LD C,(XY+d)	%%CBd76	BIT 6,(XY+d)
CB0E	RRC (HL)	ED50	IN D,(C)	%%56d	LD D,(XY+d)	%%CBd7E	BIT 7,(XY+d)
CB0F	RRC A	ED51	OUT (C),D	%%5Ed	LD E,(XY+d)	%%CBd86	RES 0,(XY+d)
CB10	RL B	ED52	SBC HL,DE	%%66d	LD H,(XY+d)	%%CBd8E	RES 1,(XY+d)
CB11	RL C	ED53aa	LD (aa),DE	%%6Ed	LD L,(XY+d)	%%CBd96	RES 2,(XY+d)
CB12	RL D	ED56	IM 1	%%70d	LD (XY+d),B	%%CBd9E	RES 3,(XY+d)
CB13	RL E	ED57	LD A,I	%%71d	LD (XY+d),C	%%CBdA6	RES 4,(XY+d)
CB14	RL H	ED58	IN E,(C)	%%72d	LD (XY+d),D	%%CBdAE	RES 5,(XY+d)
CB15	RL L	ED59	OUT (C),E	%%73d	LD (XY+d),E	%%CBdB6	RES 6,(XY+d)
CB16	RL (HL)	ED5A	ADC HL,DE	%%74d	LD (XY+d),H	%%CBdBE	RES 7,(XY+d)
CB17	RL A	ED5Baa	LD DE,(aa)	%%75d	LD (XY+d),L	%%CBdC6	SET 0,(XY+d)
CB18	RR B	ED5E	IM 2	%%77d	LD (XY+d),A	%%CBdCE	SET 1,(XY+d)
CB19	RR C	ED5F	LD A,R	%%7Ed	LD A,(XY+d)	%%CBdD6	SET 2,(XY+d)
CB1A	RR D	ED60	IN H,(C)	%%86d	ADD A,(XY+d)	%%CBdDE	SET 3,(XY+d)
CB1B	RR E	ED61	OUT (C),H	%%8Ed	ADC A,(XY+d)	%%CBdE6	SET 4,(XY+d)
CB1C	RR H	ED62	SBC HL,HL	%%96d	SUB (XY+d)	%%CBdEE	SET 5,(XY+d)
CB1D	RR L	ED67	RRD	%%9Ed	SBC A,(XY+d)	%%CBdF6	SET 6,(XY+d)
CB1E	RR (HL)	ED68	IN L,(C)	%%A6d	AND (XY+d)	%%CBdFE	SET 7,(XY+d)
CB1F	RR A	ED69	OUT (C),L	%%AEd	XOR (XY+d)		
CB20	SLA B	ED6A	ADC HL,HL	%%B6d	OR (XY+d)		
CB21	SLA C	ED6F	RLD	%%BEd	CP (XY+d)	%%E1	POP XY
CB22	SLA D	ED72	SBC HL,SP	%% means DD or FD and for DD, XY means IX for FD, XY means IY		%%E3	EX (SP),XY
CB23	SLA E	ED73aa	LD (aa),SP			%%E5	PUSH XY
CB24	SLA H	ED78	IN A,(C)			%%E9	JP (XY)
CB25	SLA L	ED79	OUT (C),A			%%F9	LD SP,XY
CB26	SLA (HL)	ED7A	ADC HL,SP				
CB27	SLA A	ED7Baa	LD SP,(aa)				
CB28	SRA B	EDA0	LDI				
CB29	SRA C	EDA1	CPI				
CB2A	SRA D	EDA2	INI				
CB2B	SRA E	EDA3	OUTI				
CB2C	SRA H	EDA8	LDD				
CB2D	SRA L	EDA9	CPD				
CB2E	SRA (HL)	EDAA	IND				
CB2F	SRA A	EDAB	OUTD				
CB38	SRL B	EDB0	LDIR				
CB39	SRL C	EDB1	CPIR				
CB3A	SRL D	EDB2	INIR				
CB3B	SRL E	EDB3	OTIR				
CB3C	SRL H	EDB8	LDDR				
CB3D	SRL L	EDB9	CPDR				
CB3E	SRL (HL)	EDBA	INDR				
CB3F	SRL A	EDBB	OTDR				
CB40	see BIT						
...	see RES						
CBFF	see SET						

Hex and Decimal Conversion

LSD ⟶

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	0
1	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	1
2	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	2
3	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	3
4	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	4
5	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	5
6	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	6
7	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	7
8	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	8
9	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	9
A	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	A
B	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	B
C	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	C
D	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	D
E	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	E
F	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	F
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	

Powers of Two

1	2	9	512
2	4	10	1,024
3	8	11	2,048
4	16	12	4,096
5	32	13	8,192
6	64	14	16,384
7	128	15	32,768
8	256	16	65,536

17	131,072
18	262,144
19	524,288
20	1,048,576
21	2,097,152
22	4,194,304
23	8,388,608
24	16,777,216

Unsigned Comparisons

example: CP B

A < B	JP C,YES
A ≤ B	JP C,YES ① JP Z,YES
A = B	JP Z,YES
A ≠ B	JP NZ,YES
A ≥ B	JP NC,YES
A > B	JR C,3 ① JP NZ,YES

YES represents label for code to be executed if condition is true. Internally, A-B is computed to determine flags as for 'SUB B'.

①Requires both instructions.

Status Flags

MSB							LSB
S	Z	-	H	-	P/V	N	C

S = Sign (MSB) of result
Z = 1 when result is Zero
H = Half carry from bit 3
P/V = 1 = Parity even for logic op or oVerflow for arithmetic op
N = 1 when last op was subtract (0 for add)
C = Carry (CY)

Interrupts and Reset

Falling edge sensitive NMI does a RST 66H regardless of IFF1, 2 (Interrupt Flip Flop).

If interrupts are enabled (IFF1=1), low level sensitive INT depends on mode:

MODE 0: Interrupting device puts instruction on bus (e.g. RST or CALL). Takes 2 extra time states.
MODE 1: Does a RST 38H (Z13).
MODE 2: Location pointed to by

15	87	10
I	ivi	0

and next hold vector of service subroutine. ivi (7 bit int vector index) is put on data bus by interrupting device (Z19).

IFF1 and IFF2 are both cleared by INT or DI. Both are set by EI NMI clears IFF1. RETN loads IFF1 from IFF2. LD A,I and LD A,R set P/V flag to IFF2. Reset sets PC=0, IFF1=IFF2=0, I=0, R=0, MODE =0.

General Instruction Description (except shifts)

ADC x, y	Add y+CY to x.
ADD x, y	Add y to x.
AND x	AND x to A.
BIT b, x	Test bit b of x.
CALL c, x	If condition c is true call subroutine at x.
CALL x	Call subroutine at x (push PC and jump to x).
CCF	Complement carry flag.
CP x	Compare A with x (see "Unsigned Comparisons")
CPD	Compare A with (HL); DEC HL; DEC BC.
CPDR	Like CPD, but repeat until A=(HL) or BC=0.
CPI	Compare A with (HL); INC HL; DEC BC.
CPIR	Like CPI, but repeat until A=(HL) or BC=0.
CPL	Complement A (1's comp.).
DAA	Decimal adjust A (after add or sub of BCD data).
DEC x	Decrement x by 1.
DI	Disable interrupts.
DJNZ d	Decrement B; jump relative by d if B not zero.
EI	Enable interrupts after next instruction.
EX x,y	Exchange x with y.
EXX	Exchange BC, DE, HL with BC', DE', HL'.
HALT	Halt (wait for interrupt or reset).
IM x	Set interrupt mode to x.
IN A, (n)	Input port n into A (6).
IN r, (C)	Input port (C) into r (7).
INC x	Increment x by 1.
IND	Load (HL) from port (C); DEC B; DEC HL; (7).
INDR	Like IND, but repeat until B=0 (7).
INI	Load (HL) from port (C); DEC B; INC HL; (7).
INIR	Like INI, but repeat until B=0 (7).
JP c, x	If condition c is true jump to location x.
JP x	Jump to location x.
JR c, d	If condition c is true jump relative by d.
JR d	Jump relative by d.
LD x, y	Load x with y (move y to x).
LDD	Load (DE) with (HL); DEC DE; DEC HL; DEC BC.
LDDR	Like LDD, but repeat until BC=0.
LDI	Load (DE) with (HL); INC DE; INC HL; DEC BC.
LDIR	Like LDI, but repeat until BC=0.
NEG	Negate A (2's comp.).
NOP	No operation.
OR x	OR x to A.
OTDR	Like OUTD, but repeat until B=0 (7).
OTIR	Like OUTI, but repeat until B=0 (7).
OUT (C), r	Output r to port (C) (7).
OUT (n), A	Output A to port n (7).
OUTD	Output (HL) to port (C); DEC B; DEC HL; (7).
OUTI	Output (HL) to port (C); DEC B; INC HL; (7).
POP x	Pop x from top of stack updating SP.
PUSH x	Push x onto top of stack updating SP.
RES b, x	Reset bit b of x (to 0).
RET	Return from subroutine (pop PC).
RET c	If condition c is true return from subroutine.
RETI	Return from interrupt.
RETN	Return from NMI (see "Interrupts").
RST x	Call subroutine at x (1 byte inst).
SBC x, y	Subtract y+CY from x.
SCF	Set carry flag (to 1).
SET b, x	Set bit b of x (to 1).
SUB x	Subtract x from A.
XOR x	XOR x to A.

Pinout

A11	1	40	A10
A12	2	39	A9
A13	3	38	A8
A14	4	37	A7
A15	5	36	A6
Ø	6	35	A5
D4	7	34	A4
D3	8	33	A3
D5	9	32	A2
D6	10	31	A1
+5V	11	30	A0
D2	12	29	GND
D7	13	28	RFSH
D0	14	27	M1
D1	15	26	RESET
INT	16	25	BUSRQ
NMI	17	24	WAIT
HALT	18	23	BUSAK
MREQ	19	22	WR
IORQ	20	21	RD

ASCII Character Set

LSD \ MSD	0 000	1 001	2 010	3 011	4 100	5 101	6 110	7 111
0 0000	NUL	DLE	SP	0	@	P	'	p
1 0001	SOH	DC1	!	1	A	Q	a	q
2 0010	STX	DC2	"	2	B	R	b	r
3 0011	ETX	DC3	#	3	C	S	c	s
4 0100	EOT	DC4	$	4	D	T	d	t
5 0101	ENQ	NAK	%	5	E	U	e	u
6 0110	ACK	SYN	&	6	F	V	f	v
7 0111	BEL	ETB	'	7	G	W	g	w
8 1000	BS	CAN	(	8	H	X	h	x
9 1001	HT	EM	)	9	I	Y	i	y
A 1010	LF	SUB	*	:	J	Z	j	z
B 1011	VT	ESC	+	;	K	[	k	{
C 1100	FF	FS	,	<	L	\	l	|
D 1101	CR	GS	-	=	M	]	m	}
E 1110	SO	RS	.	>	N	↑	n	~
F 1111	SI	US	/	?	O	←	o	DEL

Registers

main		alternate		special	
A	F	A'	F'	I	R
B	C	B'	C'	INDEX IX	
D	E	D'	E'	INDEX IY	
H	L	H'	L'	STCK PTR SP	
small=8 Bit		large=16 bit		PGRM CTR PC	

A=Accumulator
F=Flags
I= Interrupt vector
R=Memory refresh

When AF,BC,DE,HL used as pairs A,B,D,H are high order.

MICRO LOGIC CORP.
HACKENSACK, NJ

8080A & 8085A

MICROPROCESSOR INSTANT REFERENCE CARD

MICRO® CHART

INSTRUCTION SET

	INSTRUCTION		OCT	S&F	DESCRIPTION
A	ADC	A	217	4	Double A with carry (shift left with carry)
	ADC	B	210		Add B and carry to A
	ADC	C	211		Add C and carry to A
	ADC	D	212		Add D and carry to A
	ADC	E	213		Add E and carry to A
	ADC	H	214		Add H and carry to A
	ADC	L	215	Ⓐ	Add L and carry to A
	ADC	M	216	7 Ⓐ	Add LOC (H&L) and carry to A
	ACI	v	316	7 Ⓐ	Add v and carry to A
	ADD	A	207	4	Double A (shift A left)
	ADD	B	200		Add B to A
	ADD	C	201		Add C to A
	ADD	D	202		Add D to A
	ADD	E	203		Add E to A
	ADD	H	204		Add H to A
	ADD	L	205	Ⓐ	Add L to A
	ADD	M	206	7 Ⓐ	Add LOC(H&L) to A
	ADI	v	306	7 Ⓐ	Add v to A
	ANA	A	247	4	Test A and clear carry
	ANA	B	240		AND B to A
	ANA	C	241		AND C to A
	ANA	D	242		AND D to A
	ANA	E	243		AND E to A
	ANA	H	244		AND H to A
	ANA	L	245	Ⓓ	AND L to A
	ANA	M	246	7 Ⓓ	AND LOC(H&L) to A
	ANI	v	346	7 Ⓓ	AND v to A
C	CALL	aa	315	17-18 Ⓝ	Call subroutine aa
	CZ	aa	314	11/17 - 9/18	If zero CALL
	CNZ	aa	304		If not zero CALL
	CP	aa	364		If plus CALL
	CM	aa	374		If minus CALL
	CC	aa	334		If carry CALL
	CNC	aa	324		If no carry CALL
	CPE	aa	354		If even parity CALL
	CPO	aa	344	Ⓝ	If odd parity CALL
	CMA		057	4 Ⓝ	Complement A (1's comp)
	CMC		077	4 Ⓒ	Complement carry
	CMP	A	277	4	Set zero flag
	CMP	B	270		Compare A with B
	CMP	C	271		Compare A with C
	CMP	D	272		Compare A with D
	CMP	E	273		Compare A with E
	CMP	H	274		Compare A with H
	CMP	L	275	Ⓐ	Compare A with L
	CMP	M	276	7 Ⓐ	Compare A with LOC(H&L)
	CPI	v	376	7 Ⓐ	Compare A with v
D	DAA		047	4 Ⓐ	Decimal adjust A
	DAD	B	011	10	Add B&C to H&L
	DAD	D	031		Add D&E to H&L
	DAD	H	051		Double H&L (shift H&L left)
	DAD	SP	071	Ⓒ	Add SP to H&L
	DCR	A	075	5-4	Decrement A
	DCR	B	005		Decrement B
	DCR	C	015		Decrement C
	DCR	D	025		Decrement D
	DCR	E	035		Decrement E
	DCR	H	045		Decrement H
	DCR	L	055	Ⓑ	Decrement L
	DCR	M	065	10 Ⓑ	Decrement LOC(H&L)
	DCX	B	013	5-6	Decrement B&C
	DCX	D	033		Decrement D&E
	DCX	H	053		Decrement H&L
	DCX	SP	073	Ⓝ	Decrement SP
E	DI		363	4	Disable interrupts
	EI		373	Ⓝ	Enable interrupts
H	HLT		166	7-5 Ⓝ	Halt until interrupt
I	IN	v	333	10 Ⓝ	Input from device v to A
	INR	A	074	5-4	Increment A
	INR	B	004		Increment B
	INR	C	014		Increment C
	INR	D	024		Increment D
	INR	E	034		Increment E
	INR	H	044		Increment H
	INR	L	054	Ⓑ	Increment L
	INR	M	064	10 Ⓑ	Increment LOC(H&L)
	INX	B	003	5-6	Increment B&C
	INX	D	023		Increment D&E
	INX	H	043		Increment H&L
	INX	SP	063	Ⓝ	Increment SP
J	IMP	aa	303	10 Ⓝ	Jump to LOC aa
	JZ	aa	312	10 - 7/10	If zero JMP
	JNZ	aa	302		If not zero JMP
	JP	aa	362		If plus JMP
	JM	aa	372		If minus JMP
	JC	aa	332		If carry JMP
	JNC	aa	322		If no carry JMP
	JPE	aa	352		If even parity JMP
	JPO	aa	342	Ⓝ	If odd parity JMP
L	LDA	aa	072	13 Ⓝ	Load A from LOC aa
	LDAX	B	012	7	Load A from LOC(B&C)
	LDAX	D	032	Ⓝ	Load A from LOC(D&E)
	LHLD	aa	052	16 Ⓝ	Load H&L from aa & next
	LXI	B, vv	001	10	Load B&C with vv
	LXI	D, vv	021		Load D&E with vv
	LXI	H, vv	041		Load H&L with vv
	LXI	SP,vv	061	Ⓝ	Load SP with vv
M	MOV	A,B	170	5-4	Move B to A
	MOV	A,C	171		Move C to A
	MOV	A,D	172		Move D to A
	MOV	A,E	173		Move E to A
	MOV	A,H	174		Move H to A
	MOV	A,L	175	Ⓝ	Move L to A
	MOV	A,M	176	7 Ⓝ	Move LOC(H&L) to A
	MOV	B,A	107	5-4	Move A to B
	MOV	B,C	101		Move C to B
	MOV	B,D	102		Move D to B
	MOV	B,E	103		Move E to B
	MOV	B,H	104		Move H to B
	MOV	B,L	105	Ⓝ	Move L to B
	MOV	B,M	106	7 Ⓝ	Move LOC(H&L) to B
	MOV	C,A	117	5-4	Move A to C
	MOV	C,B	110		Move B to C
	MOV	C,D	112		Move D to C
	MOV	C,E	113		Move E to C
	MOV	C,H	114		Move H to C
	MOV	C,L	115	Ⓝ	Move L to C
	MOV	C,M	116	7 Ⓝ	Move LOC(H&L) to C
	MOV	D,A	127	5-4	Move A to D
	MOV	D,B	120		Move B to D
	MOV	D,C	121		Move C to D
	MOV	D,E	123		Move E to D
	MOV	D,H	124		Move H to D
	MOV	D,L	125	Ⓝ	Move L to D
	MOV	D,M	126	7 Ⓝ	Move LOC(H&L) to D
	MOV	E,A	137	5-4	Move A to E
	MOV	E,B	130		Move B to E
	MOV	E,C	131		Move C to E
	MOV	E,D	132		Move D to E
	MOV	E,H	134		Move H to E
	MOV	E,L	135	Ⓝ	Move L to E
	MOV	E,M	136	7 Ⓝ	Move LOC(H&L) to E
	MOV	H,A	147	5-4	Move A to H
	MOV	H,B	140		Move B to H
	MOV	H,C	141		Move C to H
	MOV	H,D	142		Move D to H
	MOV	H,E	143		Move E to H
	MOV	H,L	145	Ⓝ	Move L to H
	MOV	H,M	146	7 Ⓝ	Move LOC(H&L) to H
	MOV	L,A	157	5-4	Move A to L
	MOV	L,B	150		Move B to L
	MOV	L,C	151		Move C to L
	MOV	L,D	152		Move D to L
	MOV	L,E	153		Move E to L
	MOV	L,H	154	Ⓝ	Move H to L
	MOV	L,M	156	7 Ⓝ	Move LOC(H&L) to L
	MOV	M,A	167	7	Move A to LOC(H&L)
	MOV	M,B	160		Move B to LOC(H&L)
	MOV	M,C	161		Move C to LOC(H&L)
	MOV	M,D	162		Move D to LOC(H&L)
	MOV	M,E	163		Move E to LOC(H&L)
	MOV	M,H	164		Move H to LOC(H&L)
	MOV	M,L	165	Ⓝ	Move L to LOC(H&L)
	MVI	A,v	076	7	Move v to A
	MVI	B,v	006		Move v to B
	MVI	C,v	016		Move v to C
	MVI	D,v	026		Move v to D
	MVI	E,v	036		Move v to E
	MVI	H,v	046		Move v to H
	MVI	L,v	056	Ⓝ	Move v to L
	MVI	M,v	066	10 Ⓝ	Move v to LOC(H&L)
N	NOP		000	4 Ⓝ	No operation
O	ORA	A	267	4	Test A and clear carry
	ORA	B	260		OR B to A
	ORA	C	261		OR C to A
	ORA	D	262		OR D to A
	ORA	E	263		OR E to A
	ORA	H	264		OR H to A
	ORA	L	265	Ⓔ	OR L to A
	ORA	M	266	7 Ⓔ	OR LOC(H&L) to A
	ORI	v	366	7 Ⓔ	OR v to A
	OUT	v	323	10 Ⓝ	Output A to device v
P	PCHL		351	5-6 Ⓝ	Jump to LOC(H&L)
	POP	B	301	10	Pop B&C from stack
	POP	D	321		Pop D&E from stack
	POP	H	341	Ⓝ	Pop H&L from stack
	POP	PSW	361	10 Ⓐ	Pop A and flags from stack
P	PUSH	B	305	11-12	Push B&C onto stack
	PUSH	D	325		Push D&E onto stack
	PUSH	H	345		Push H&L onto stack
	PUSH	PSW	365	Ⓝ	Push A and flags onto stack
R	RAL		027	4	Rotate CY & A left
	RAR		037		Rotate CY & A right
	RLC		007		Rotate A left and into carry
	RRC		017	Ⓒ	Rotate A right and into carry
	RIM (8085)		040	4 Ⓝ	Read interrupt mask
	RET		311	10 Ⓝ	Return from subroutine
	RZ		310	5/11 - 6/12	If zero RET
	RNZ		300		If not zero RET
	RP		360		If plus RET
	RM		370		If minus RET
	RC		330		If carry RET
	RNC		320		If no carry RET
	RPE		350		If even parity RET
	RPO		340	Ⓝ	If odd parity RET
	RST	0	307	11-12	Call subroutine at 00H
	RST	1	317		Call subroutine at 08H
	RST	2	327		Call subroutine at 10H
	RST	3	337		Call subroutine at 18H
	RST	4	347		Call subroutine at 20H
	RST	5	357		Call subroutine at 28H
	RST	6	367		Call subroutine at 30H
	RST	7	377	Ⓝ	Call subroutine at 38H
S	SBB	A	237	4	Set A to minus carry
	SBB	B	230		Subtract B & CY from A
	SBB	C	231		Subtract C & CY from A
	SBB	D	232		Subtract D & CY from A
	SBB	E	233		Subtract E & CY from A
	SBB	H	234		Subtract H & CY from A
	SBB	L	235	Ⓐ	Subtract L & CY from A
	SBB	M	236	7 Ⓐ	Subtract LOC (H&L) & CY from A
	SBI	v	336	7 Ⓐ	Subtract v and CY from A
	SHLD	aa	042	16 Ⓝ	Store H&L at aa & next
	SIM (8085)		060	4 Ⓝ	Set interrupt mask
	SPHL		371	5-6 Ⓝ	Load SP from H&L
	STA	aa	062	13 Ⓝ	Store A at LOC aa
	STAX	B	002	7	Store A at LOC(B&C)
	STAX	D	022	Ⓝ	Store A at LOC(D&E)
	STC		067	4 Ⓒ	Set carry (to 1)
	SUB	A	227	4	Clear A
	SUB	B	220		Subtract B from A
	SUB	C	221		Subtract C from A
	SUB	D	222		Subtract D from A
	SUB	E	223		Subtract E from A
	SUB	H	224		Subtract H from A
	SUB	L	225	Ⓐ	Subtract L from A
	SUB	M	226	7 Ⓐ	Subtract LOC(H&L) from A
	SUI	v	326	7 Ⓐ	Subtract v from A
	XCHG		353	4 Ⓝ	Exchange D&E with H&L
X	XRA	A	257	4	Clear A
	XRA	B	250		Exclusive OR B to A
	XRA	C	251		Exclusive OR C to A
	XRA	D	252		Exclusive OR D to A
	XRA	E	253		Exclusive OR E to A
	XRA	H	254		Exclusive OR H to A
	XRA	L	255	Ⓔ	Exclusive OR L to A
	XRA	M	256	7 Ⓔ	Exclusive OR LOC (H&L) to A
	XRI	v	356	7 Ⓔ	Exclusive OR v to A
	XTHL		343	18-16 Ⓝ	Exchange top of stack with H&L

EXAMPLES FROM S & F COLUMN

7	7 STATES FOR 8080 & 8085
5-4	5 STATES 8080; 4 STATES 8085
10-7/10	10 STATES 8080; 7 STATES FOR FAILURE, 10 STATES FOR SUCCESS ON 8085

Intentionally Blank

DO NOT PLACE ON HOT SURFACES

MICRO CHART AUTHOR: JAMES D. LEWIS

#100B

MICRO LOGIC CORP.
HACKENSACK, NJ

8080A & 8085A

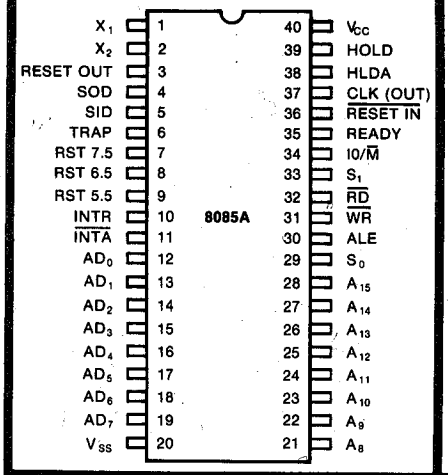
MICROPROCESSOR INSTANT REFERENCE CARD

MICRO® CHART

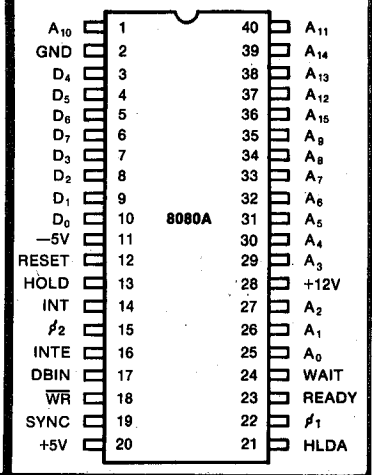
HEX to Instruction Conversion

LSD →	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NOP	LXI B	STAX B	INX B	INR B	DCR B	MVI B	RLC	□	DAD B	LDAX B	DCX B	INR C	DCR C	MVI C	RRC
1	□	LXI D	STAX D	INX D	INR D	DCR D	MVI D	RAL	□	DAD D	LDAX D	DCX D	INR E	DCR E	MVI E	RAR
2	RIM	LXI H	SHLD	INX H	INR H	DCR H	MVI H	DAA	□	DAD H	LHLD	DCX H	INR L	DCR L	MVI L	CMA
3	SIM	LXI SP	STA	INX SP	INR M	DCR M	MVI M	STC	□	DAD SP	LDA	DCX SP	INR A	DCR A	MVI A	CMC
4	MOV B,B	MOV B,C	MOV B,D	MOV B,E	MOV B,H	MOV B,L	MOV B,M	MOV B,A	MOV C,B	MOV C,C	MOV C,D	MOV C,E	MOV C,H	MOV C,L	MOV C,M	MOV C,A
5	MOV D,B	MOV D,C	MOV D,D	MOV D,E	MOV D,H	MOV D,L	MOV D,M	MOV D,A	MOV E,B	MOV E,C	MOV E,D	MOV E,E	MOV E,H	MOV E,L	MOV E,M	MOV E,A
6	MOV H,B	MOV H,C	MOV H,D	MOV H,E	MOV H,H	MOV H,L	MOV H,M	MOV H,A	MOV L,B	MOV L,C	MOV L,D	MOV L,E	MOV L,H	MOV L,L	MOV L,M	MOV L,A
7	MOV M,B	MOV M,C	MOV M,D	MOV M,E	MOV M,H	MOV M,L	HLT	MOV M,A	MOV A,B	MOV A,C	MOV A,D	MOV A,E	MOV A,H	MOV A,L	MOV A,M	MOV A,A
8	ADD B	ADD C	ADD D	ADD E	ADD H	ADD L	ADD M	ADD A	ADC B	ADC C	ADC D	ADC E	ADC H	ADC L	ADC M	ADC A
9	SUB B	SUB C	SUB D	SUB E	SUB H	SUB L	SUB M	SUB A	SBB B	SBB C	SBB D	SBB E	SBB H	SBB L	SBB M	SBB A
A	ANA B	ANA C	ANA D	ANA E	ANA H	ANA L	ANA M	ANA A	XRA B	XRA C	XRA D	XRA E	XRA H	XRA L	XRA M	XRA A
B	ORA B	ORA C	ORA D	ORA E	ORA H	ORA L	ORA M	ORA A	CMP B	CMP C	CMP D	CMP E	CMP H	CMP L	CMP M	CMP A
C	RNZ	POP B	JNZ	JMP	CNZ	PUSH B	ADI	RST 0	RZ	RET	JZ	□	CZ	CALL	ACI	RST 1
D	RNC	POP D	JNC	OUT	CNC	PUSH D	SUI	RST 2	RC	□	JC	IN	CC	□	SBI	RST 3
E	RPO	POP H	JPO	XTHL	CPO	PUSH H	ANI	RST 4	RPE	PCHL	JPE	XCHG	CPE	□	XRI	RST 5
F	RP	POP PSW	JP	DI	CP	PUSH PSW	ORI	RST 6	RM	SPHL	JM	EI	CM	□	CPI	RST 7

8085A Pins



8080A Pins



Registers

A (Acc.)	(8)	Flags	(8)
B	(8)	C	(8)
D	(8)	E	(8)
H	(8)	L	(8)
PC (Program Counter)			(16)
SP (Stack Pointer)			(16)

Note: When BC, DE, HL used as pairs, B,D,H, are high order.

8085 Interrupts

Name	Adr	Type (starting with highest priority)
TRAP	24H	Rising edge & high level till sampled
RST 7.5	3CH	Rising edge (latched)
RST 6.5	34H	High level till sampled
RST 5.5	2CH	High level till sampled
INTR	- - -	High level till sampled

ASCII Character Set

LSD \ MSD	0 000	1 001	2 010	3 011	4 100	5 101	6 110	7 111
0 0000	NUL	DLE	SP	0	@	P	`	p
1 0001	SOH	DC1	!	1	A	Q	a	q
2 0010	STX	DC2	"	2	B	R	b	r
3 0011	ETX	DC3	#	3	C	S	c	s
4 0100	EOT	DC4	$	4	D	T	d	t
5 0101	ENQ	NAK	%	5	E	U	e	u
6 0110	ACK	SYN	&	6	F	V	f	v
7 0111	BEL	ETB	'	7	G	W	g	w
8 1000	BS	CAN	(	8	H	X	h	x
9 1001	HT	EM	)	9	I	Y	i	y
A 1010	LF	SUB	*	:	J	Z	j	z
B 1011	VT	ESC	+	;	K	[	k	{
C 1100	FF	FS	,	<	L	\	l	|
D 1101	CR	GS	-	=	M	]	m	}
E 1110	SO	RS	.	>	N	↑	n	~
F 1111	SI	US	/	?	O	←	o	DEL

Hex and Decimal Conversion

LSD →	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
2	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
3	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
4	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
5	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
6	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
7	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
8	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
9	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
A	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
B	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
C	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
D	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
E	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
F	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

Unsigned Comparisons

example: CMP B

A < B	JC YES
A ⩽ B	JC YES ① JZ YES
A = B	JZ YES
A ≠ B	JNZ YES
A ⩾ B	JNC YES
A > B	JC *+6 ① JNZ YES

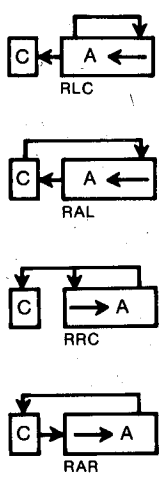
YES represents label for code to be executed if condition is true. Internally, A-B is computed to determine flags as for 'SUB B'.

① Requires both instructions.

8085A RIM & SIM bits

	MSB							LSB
RIM:	SID	I7.5	I6.5	I5.5	IE	M7.5	M6.5	M5.5
SIM:	SOD	SOE	—	R7.5	MSE	M7.5	M6.5	M5.5

Rotate Instructions



Status Flags

MSB							LSB
S	Z	—	AC	—	P	—	C

S = sign (MSB) of result
Z = 1 = result is 0
AC = aux C from bit 3
P = 1 = parity is even
C = 1 = carry (also CY)

Miscellaneous

The 2 byte addresses in code, stack, and data areas are stored low byte followed by high byte. Thus, in hex, JMP 1234H is: C3 34 12.

SP points to used byte at top of stack. Stack push decrements SP by 2. High byte of pair is pushed first.

This card is based on specifications from Intel.®

8080 Timing Example

Using 8224 clock chip:

XTAL = 18 mhz/9 = 2 mhz
= 500 ns = 1 state.
ADD A = 4 states = 2 us

8085 Timing Example

XTAL = 6 mhz/2 = 3 mhz
= 333 ns = 1 state.
ADD A = 4 states = 1.333 us

DAA (Decimal Adjust Accumulator)

The 8 bits in the acc are adjusted to form 2 BCD digits by:

1. If the low 4 bits are > 9 or if the AC flag = 1, 6 is added to the acc.
2. If the high 4 bits are now > 9, or if CY = 1, 6 is added to the high 4 bits.

Flag Codes

Circled letters in Instruction Set indicate which flags are effected as follows:

- Ⓐ All
- Ⓑ All But C
- Ⓒ Just C
- Ⓓ Depends on which MPU:
  8080: All, C=0, AC=OR of bits 3
  8085: All, C=0, AC=1
- Ⓔ All, C=0, AC=0
- Ⓝ None

Abbreviations

aa	= a 2 byte address
v	= a 1 byte value
vv	= a 2 byte value
M	= Memory location that reg pair H&L points to
S&F	= States and Flag Codes
LOC	= Location
MSE	= Mask Set Enable
PSW	= Program Status Word (Acc & Flags)
SID	= Serial Input Data
SOD	= Serial Output Data
SOE	= Serial Output Enable

See also: "Registers" and "Status Flags"

MICRO LOGIC CORP.
HACKENSACK, NJ

8086 & 8088

MICROPROCESSOR INSTANT REFERENCE CARD

MICRO CHART®

Instruction Set

INST	ADDR
AAA	none
AAD	none
AAM	none
AAS	none
ADC	r,r
ADC	m,r
ADC	rr,rr
ADC	mm,rr
ADC	r,m
ADC	rr,mm
ADC	r,i
ADC	m,i
ADC	rr,ii
ADC	mm,ii
ADC	rr,i
ADC	mm,i
ADC	AL,i
ADC	AX,ii
ADD	r,r
ADD	m,r
ADD	rr,rr
ADD	mm,rr
ADD	r,m
ADD	rr,mm
ADD	r,i
ADD	m,i
ADD	rr,ii
ADD	mm,ii
ADD	rr,i
ADD	mm,i
ADD	AL,i
ADD	AX,ii
AND	r,r
AND	m,r
AND	rr,rr
AND	mm,rr
AND	r,m
AND	rr,mm
AND	r,i
AND	m,i
AND	rr,ii
AND	mm,ii
AND	AL,i
AND	AX,ii
CALL	dd
CALL	rr
CALL	mm
CALL	aaaa
CALL	dw
CBW	none
CLC	none
CLD	none
CLI	none
CMC	none
CMP	r,r
CMP	m,r
CMP	rr,rr
CMP	mm,rr
CMP	r,m
CMP	rr,mm
CMP	r,i
CMP	m,i
CMP	rr,ii
CMP	mm,ii
CMP	rr,i
CMP	mm,i
CMP	AL,i
CMP	AX,ii
CMPS	byte
CMPS	word
CWD	none
DAA	none
DAS	none
DEC	m
DEC	mm
DEC	AL
DEC	CL
DEC	DL
DEC	BL
DEC	AH
DEC	CH
DEC	DH
DEC	BH
DEC	AX
DEC	CX
DEC	DX
DEC	BX
DEC	SP
DEC	BP
DEC	SI
DEC	DI
DIV	r
DIV	m
DIV	rr
DIV	mm
ESC	0,rr
ESC	1,rr
ESC	2,rr
ESC	3,rr
ESC	4,rr
ESC	5,rr
ESC	6,rr
ESC	7,rr
ESC	0,mm
ESC	1,mm
ESC	2,mm
ESC	3,mm
ESC	4,mm
ESC	5,mm
ESC	6,mm
ESC	7,mm
HLT	none
IDIV	r
IDIV	m
IDIV	rr
IDIV	mm
IMUL	r
IMUL	m
IMUL	rr
IMUL	mm
IN	AL,i
IN	AX,i
IN	AL,DX
IN	AX,DX
INC	m
INC	mm
INC	AL
INC	CL
INC	DL
INC	BL
INC	AH
INC	CH
INC	DH
INC	BH
INC	AX
INC	CX
INC	DX
INC	BX
INC	SP
INC	BP
INC	SI
INC	DI
INT	3
INT	i
INTO	none
IRET	none
JA	d
JAE	d
JB	d
JBE	d
JC	d
JCXZ	d
JE	d
JG	d
JGE	d
JL	d
JLE	d
JMP	d
JMP	dd
JMP	rr
JMP	mm
JMP	aaaa
JMP	dw
JNA	d
JNAE	d
JNB	d
JNBE	d
JNC	d
JNE	d
JNG	d
JNGE	d
JNL	d
JNLE	d
JNO	d
JNP	d
JNS	d
JNZ	d
JO	d
JP	d
JPE	d
JPO	d
JS	d
JZ	d
LAHF	none
LDS	rr,dw
LEA	rr,m
LES	rr,dw
LOCK	prfx
LODS	byte
LODS	word
LOOP	d
LOOPE	d
LOOPZ	d
LOOPNZ	d
LOOPNE	d
MOV	r,r
MOV	m,r
MOV	rr,rr
MOV	mm,rr
MOV	r,m
MOV	rr,mm
MOV	m,i
MOV	mm,ii
MOV	AL,i
MOV	CL,i
MOV	DL,i
MOV	BL,i
MOV	AH,i
MOV	CH,i
MOV	DH,i
MOV	BH,i
MOV	AX,ii
MOV	CX,ii
MOV	DX,ii
MOV	BX,ii
MOV	SP,ii
MOV	BP,ii
MOV	SI,ii
MOV	DI,ii
MOV	AL,aa
MOV	AX,aa
MOV	aa,AL
MOV	aa,AX
MOV	sr,rr
MOV	sr,mm
MOV	rr,sr
MOV	mm,sr
MOVS	byte
MOVS	word
MOVSB	none
MOVSW	none
MUL	r
MUL	m
MUL	rr
MUL	mm
NEG	r
NEG	m
NEG	rr
NEG	mm
NOP	none
NOT	r
NOT	m
NOT	rr
NOT	mm
OR	r,r
OR	m,r
OR	rr,rr
OR	mm,rr
OR	r,m
OR	rr,mm
OR	r,i
OR	m,i
OR	rr,ii
OR	mm,ii
OR	AL,i
OR	AX,ii
OUT	i,AL
OUT	i,AX
OUT	DX,AL
OUT	DX,AX
POP	mm
POP	AX
POP	CX
POP	DX
POP	BX
POP	SP
POP	BP
POP	SI
POP	DI
POP	ES
POP	CS
POP	SS
POP	DS
POPF	none
PUSH	mm
PUSH	AX
PUSH	CX
PUSH	DX
PUSH	BX
PUSH	SP
PUSH	BP
PUSH	SI
PUSH	DI
PUSH	ES
PUSH	CS
PUSH	SS
PUSH	DS
PUSHF	none
RCL	r,1
RCL	m,1
RCL	rr,1
RCL	mm,1
RCL	r,CL
RCL	m,CL
RCL	rr,CL
RCL	mm,CL
RCR	r,1
RCR	m,1
RCR	rr,1
RCR	mm,1
RCR	r,CL
RCR	m,CL
RCR	rr,CL
RCR	mm,CL
REP	prfx
REPE	prfx
REPZ	prfx
REPNE	prfx
REPNZ	prfx
RET	ws
RET	ii ws
RET	as
RET	ii as
ROL	r,1
ROL	m,1
ROL	rr,1
ROL	mm,1
ROL	r,CL
ROL	m,CL
ROL	rr,CL
ROL	mm,CL
ROR	r,1
ROR	m,1
ROR	rr,1
ROR	mm,1
ROR	r,CL
ROR	m,CL
ROR	rr,CL
ROR	mm,CL
SAHF	none
SAL	r,1
SAL	m,1
SAL	rr,1
SAL	mm,1
SAL	r,CL
SAL	m,CL
SAL	rr,CL
SAL	mm,CL
SAR	r,1
SAR	m,1
SAR	rr,1
SAR	mm,1
SAR	r,CL
SAR	m,CL
SAR	rr,CL
SAR	mm,CL
SBB	r,r
SBB	m,r
SBB	rr,rr
SBB	mm,rr
SBB	r,m
SBB	rr,mm
SBB	r,i
SBB	m,i
SBB	rr,ii
SBB	mm,ii
SBB	AL,i
SBB	AX,ii
SCAS	byte
SCAS	word
SEG	
CS:	prfx
DS:	prfx
ES:	prfx
SS:	prfx
SHL	r,1
SHL	m,1
SHL	rr,1
SHL	mm,1
SHL	r,CL
SHL	m,CL
SHL	rr,CL
SHL	mm,CL
SHR	r,1
SHR	m,1
SHR	rr,1
SHR	mm,1
SHR	r,CL
SHR	m,CL
SHR	rr,CL
SHR	mm,CL
STC	none
STD	none
STI	none
STOS	byte
STOS	word
SUB	r,r
SUB	m,r
SUB	rr,rr
SUB	mm,rr
SUB	r,m
SUB	rr,mm
SUB	r,i
SUB	m,i
SUB	rr,ii
SUB	mm,ii
SUB	AL,i
SUB	AX,ii
TEST	r,r
TEST	r,m
TEST	rr,rr
TEST	rr,mm
TEST	r,i
TEST	m,i
TEST	rr,ii
TEST	mm,ii
TEST	AL,i
TEST	AX,ii
WAIT	none
XCHG	r,r
XCHG	r,m
XCHG	rr,rr
XCHG	rr,mm
XCHG	AX,CX
XCHG	AX,DX
XCHG	AX,BX
XCHG	AX,SP
XCHG	AX,BP
XCHG	AX,SI
XCHG	AX,DI
XLAT	byte
XOR	r,r
XOR	m,r
XOR	rr,rr
XOR	mm,rr
XOR	r,m
XOR	rr,mm
XOR	r,i
XOR	m,i
XOR	rr,ii
XOR	mm,ii
XOR	AL,i
XOR	AX,ii

Cycle Codes

A1	4	A2	11(15)	A3	unused
B1	60		15	B3	12
C1	83	B2	8(12)+	C3	12(16)
D1	3		12+		16
E1	16+	C2	101-112	D3	5:17
F1	16(24)+	D2	107-118+	E3	5:19
	24+	E2	165-184	F3	8+
G1	9+	F2	11+	G3	8(12)+
H1	9(13)+	G2	80-98		12+
	13+	H2	88-104+	H3	18
I1	17+	I2	128-154	I3	18(26)
J1	17(25)+	J2	11(15)+		26
	25+		15+	J3	70-77
K1	19	K2	10	K3	76-83+
	23	L2	10(14)	L3	118-133
L1	16		14	M3	4+5N
	20	M2	8	N3	17(21)+
M1	21(25)+	N2	8(12)		25+
	29+		12	O3	8
N1	28	O2	52		12
	36		72	P3	16(20)+
O1	37(45)+	P2	51		24+
	53+		71	Q3	11
P1	2	Q2	4:53		15
Q1	10+		4:73	R3	10
R1	10(14)+	R2	32		14
	14+		44	S3	8+4N
S1	22	S2	4:16	T3	20+4N+
T1	22(30)	T2	6:18	U3	N/A
	30	U2	15	V3	16
U1	5	V2	11		20
V1	15+	W2	18(22)+	W3	20
W1	15(23)+		22+		24
	23+	X2	24(32)+	X3	26
X1	80-90		32+		34
Y1	86-96+	Y2	16(24)+	Y3	25
Z1	144-162		24+		33
		Z2	2+	Z3	15(19)
A4	171-190(175-194)+				19
	175-194+				
B4	134-160(138-164)+				
	138-164+				
C4	124-139(128-143)+				
	128-143+				
D4	154-172(158-176)+				
	158-176+				
E4	20(28)+4N+				
	28+4N+				
F4	9+17N				
G4	9+17(25)N				
	9+25N				
H4	9+10N				
I4	9+10(14)N				
	9+14N				
J4	9+22N				
K4	9+22(30)N				
	9+30N				
L4	9+15N				
M4	9+15(19)N				
	9+19N				
N4	9+13N				
P4	9+13(17)N				
	9+17N				

Example

After reading "About the Tables", usage of the tables can be verified by assembly and disassembly of:

1)	1406	ADC	AL,6
2)	09D1	OR	CX,DX
3)	C8440708	MOV	(SI+7),8
4)	D1C6	ROL	SI
5)	D104 LP	ROL	(SI)
6)	72FC	JC	LP
7)	D50A	AAD	

The following notes help avoid difficulty (when converting to hex) and correspond to the lines above:

1) Use "AL,i" - not "r,i"
2) Use "rr,rr" - not "r,r". Read about Second Byte Table to convert X9.
3) Parentheses indicate mem pntr and form is "m,i". Form of first operand is "(SI+d)".
4) Use "SI" from reg part of section X.
5) Use "(SI)" from mem part of section X.
6) Read about "Relative Jumps".
7) Special case for disassembly.

Hex and Decimal Conversion

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	0
1	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	1
2	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	2
3	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	3
4	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	4
5	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	5
6	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	6
7	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	7
8	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	8
9	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	9
A	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	A
B	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	B
C	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	C
D	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	D
E	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	E
F	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	F
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	

Memory Locations

00000 - 00003	Type 0 interrupt pointer for divide-error
00004 - 00007	Type 1 interrupt pointer for single-step
00008 - 0000B	Type 2 interrupt pointer for Non-Mask-Int
0000C - 0000F	Type 3 interrupt pointer for 1-byte- inst
00010 - 00013	Type 4 interrupt pointer for INTO inst
00014 - 0007F	Type 5 thru 31 interrupt pointers reserved for Intel products
00080 - 003FF	Type 32 thru 255 available interrupt pointers (or general memory use)
00400 - FFFEF	Main memory space
FFFF0 - FFFFB	CPU jumps to code here upon Reset
FFFFC - FFFFF	Reserved for Intel products

ASCII

LSD \ MSD	0	1	2	3	4	5	6	7
	000	001	010	011	100	101	110	111
0	0000	NUL	DLE	SP	0	@	P	`	p
1	0001	SOH	DC1	!	1	A	Q	a	q
2	0010	STX	DC2	"	2	B	R	b	r
3	0011	ETX	DC3	#	3	C	S	c	s
4	0100	EOT	DC4	$	4	D	T	d	t
5	0101	ENQ	NAK	%	5	E	U	e	u
6	0110	ACK	SYN	&	6	F	V	f	v
7	0111	BEL	ETB	'	7	G	W	g	w
8	1000	BS	CAN	(	8	H	X	h	x
9	1001	HT	EM	)	9	I	Y	i	y
A	1010	LF	SUB	*	:	J	Z	j	z
B	1011	VT	ESC	+	;	K	[	k	{
C	1100	FF	FS	,	<	L	\	l	|
D	1101	CR	GS	-	=	M	]	m	}
E	1110	SO	RS	.	>	N	^	n	~
F	1111	SI	US	/	?	O	--	o	DEL

Unused

Second Byte Table

T		X	0	1	2	3	4	5	6	7	N
7	S	(BX+SI)	00	08	10	18	20	28	30	38	
8	U	(BX+DI)	01	09	11	19	21	29	31	39	
8	M	(BP+SI)	02	0A	12	1A	22	2A	32	3A	
7		(BP+DI)	03	0B	13	1B	23	2B	33	3B	
5	P	(SI)	04	0C	14	1C	24	2C	34	3C	
5	N	(DI)	05	0D	15	1D	25	2D	35	3D	
6	T	(dd)	06	0E	16	1E	26	2E	36	3E	
5	S	(BX)	07	0F	17	1F	27	2F	37	3F	
11		(BX+SI+d)	40	48	50	58	60	68	70	78	
12	T	(BX+DI+d)	41	49	51	59	61	69	71	79	
12	O	(BP+SI+d)	42	4A	52	5A	62	6A	72	7A	
11		(BP+DI+d)	43	4B	53	5B	63	6B	73	7B	
9	D	(SI+d)	44	4C	54	5C	64	6C	74	7C	
9	A	(DI+d)	45	4D	55	5D	65	6D	75	7D	
9	T	(BP+d)	46	4E	56	5E	66	6E	76	7E	
9	A	(BX+d)	47	4F	57	5F	67	6F	77	7F	
11		(BX+SI+dd)	80	88	90	98	A0	A8	B0	B8	
12	I	(BX+DI+dd)	81	89	91	99	A1	A9	B1	B9	
12	N	(BP+SI+dd)	82	8A	92	9A	A2	AA	B2	BA	
11		(BP+DI+dd)	83	8B	93	9B	A3	AB	B3	BB	
9	M	(SI+dd)	84	8C	94	9C	A4	AC	B4	BC	
9	E	(DI+dd)	85	8D	95	9D	A5	AD	B5	BD	
9	M	(BP+dd)	86	8E	96	9E	A6	AE	B6	BE	
9		(BX+dd)	87	8F	97	9F	A7	AF	B7	BF	
0	R	AX or AL	C0	C8	D0	D8	E0	E8	F0	F8	
0	E	CX or CL	C1	C9	D1	D9	E1	E9	F1	F9	
0	G	DX or DL	C2	CA	D2	DA	E2	EA	F2	FA	
0		BX or BL	C3	CB	D3	DB	E3	EB	F3	FB	
0	D	SP or AH	C4	CC	D4	DC	E4	EC	F4	FC	
0	A	BP or CH	C5	CD	D5	DD	E5	ED	F5	FD	
0	T	SI or DH	C6	CE	D6	DE	E6	EE	F6	FE	
0	A	DI or BH	C7	CF	D7	DF	E7	EF	F7	FF	

AL	CL	DL	BL	AH	CH	DH	BH	8
AX	CX	DX	BX	SP	BP	SI	DI	9
ES	CS	SS	DS	—	—	—	—	F
ES	—	SS	DS	—	—	—	—	M

	A	B	C	D	E	F	G	H	I	J	K	
ADD	ADD	ADD	ADD	ROL	ROL	ROL	ROL	ROL	TEST	INC	INC	INC
OR	OR	OR	OR	ROR	ROR	ROR	ROR	ROR	xx,i	DEC	DEC	DEC
ADC	ADC	ADC	ADC	RCL	RCL	RCL	RCL	RCL	NOT	NOT	CALL (2)	CALL (2)
SBB	SBB	SBB	SBB	RCR	RCR	RCR	RCR	RCR	NEG	NEG	CALL (3)	CALL (3)
AND	AND	AND	AND	SHL/SAL	SHL/SAL	SHL/SAL	SHL/SAL	SHL/SAL	MUL	MUL	JMP (2)	JMP (2)
SUB	SUB	SUB	SUB	SHR	SHR	SHR	SHR	SHR	IMUL	IMUL	JMP (3)	JMP (3)
XOR	XOR	XOR	XOR						DIV	DIV	PUSH xx	PUSH xx
CMP	CMP	CMP	CMP	SAR	SAR	SAR	SAR	SAR	IDIV	IDIV		
xx,i	xx,ii	xx,i	xx,ii	x	xx	x	xx,CL	xx,CL	x	xx		

(2) = 'near' transfer indirect via word in reg or mem.
(3) = 'far' transfer indirect via double word in mem.

Pinouts

8086

GND	1	40	VCC
AD14	2	39	AD15
AD13	3	38	A16/S3
AD12	4	37	A17/S4
AD11	5	36	A18/S5
AD10	6	35	A19/S6
AD9	7	34	BHE*/S7
AD8	8	33	MN/MX*
AD7	9	32	RD*
AD6	10	31	HOLD (RQ*/GT0*)
AD5	11	30	HLDA (RQ*/GT1*)
AD4	12	29	WR* (LOCK*)
AD3	13	28	M/IO* (S2*)
AD2	14	27	DT/R* (S1*)
AD1	15	26	DEN* (S0*)
AD0	16	25	ALE (QS0)
NMI	17	24	INTA* (QS1)
INTR	18	23	TEST*
CLK	19	22	READY
GND	20	21	RESET

On 8088 AD8 to AD15 are A8 to A15; pin 28 is IO/M* (S2*); pin 34 is SS0 (HIGH). Max mode is in parenthesis. '*' means active low.

AUTHOR: JAMES D. LEWIS

#107A

MICRO LOGIC CORP.
HACKENSACK, NJ

8086 & 8088

MICROPROCESSOR INSTANT REFERENCE CARD

MICRO CHART®

100% PLASTIC

There are MICRO CHARTS for Z80, 6502, 8080 & 8085, 6048 FAMILY microprocessors, 54/7400 TTL, and BASIC ALGORITHMS.

INSTANT ACCESS

We know of no information at any price produced by anyone (incl. chip mfgs.) which is as accurate, convenient, and fast as this MICRO CHART reference. We only ask that you kindly observe our copyright by using only genuine plastic MICRO CHARTS, easily purchased from your dealer or Micro Logic.

Hex to Instruction Conversion

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0	ADD X8 byte	ADD X9 word	ADD 8X byte	ADD 9X word	ADD AL,i	ADD AX,ii	PUSH ES	POP ES	OR X8 byte	OR X9 word	OR 8X byte	OR 9X word	OR AL,i	OR AX,ii	PUSH CS	POP CS	0
1	ADC X8 byte	ADC X9 word	ADC 8X byte	ADC 9X word	ADC AL,i	ADC AX,ii	PUSH SS	POP SS	SBB X8 byte	SBB X9 word	SBB 8X byte	SBB 9X word	SBB AL,i	SBB AX,ii	PUSH DS	POP DS	1
2	AND X8 byte	AND X9 word	AND 8X byte	AND 9X word	AND AL,i	AND AX,ii	SEG =ES	DAA	SUB X8 byte	SUB X9 word	SUB 8X byte	SUB 9X word	SUB AL,i	SUB AX,ii	SEG =CS	DAS	2
3	XOR X8 byte	XOR X9 word	XOR 8X byte	XOR 9X word	XOR AL,i	XOR AX,ii	SEG =SS	AAA	CMP X8 byte	CMP X9 word	CMP 8X byte	CMP 9X word	CMP AL,i	CMP AX,ii	SEG =DS	AAS	3
4																	4
5																	5
6																	6
7																	7
8	AX	BX	AX	CX	TEST 8X byte	TEST 9X word	XCHG 8X byte	XCHG 9X word	MOV X8 byte	MOV X9 word	MOV 8X byte	MOV 9X word	MOV XL	LEA 9X rr,m	MOV MX	POP OX	8
9	NOP	XCHG AX,CX	XCHG AX,DX	XCHG AX,BX	XCHG AX,SP	XCHG AX,BP	XCHG AX,SI	XCHG AX,DI	CBW	CWD	CALL aaaa	WAIT	PUSHF	POPF	SAHF	LAHF	9
A	MOV AL,aa	MOV AX,aa	MOV aa,AL	MOV aa,AX	MOVS byte	MOVS word	CMPS byte	CMPS word	TEST AL,i	TEST AX,ii	STOS byte	STOS word	LODS byte	LODS word	SCAS byte	SCAS word	A
B	MOV AL,i	MOV CL,i	MOV DL,i	MOV BL,i	MOV AH,i	MOV CH,i	MOV DH,i	MOV BH,i	MOV AX,ii	MOV CX,ii	MOV DX,ii	MOV BX,ii	MOV SP,ii	MOV BP,ii	MOV SI,ii	MOV DI,ii	B
C																	C
D																	D
E																	E
F																	F
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	

Miscellaneous Notes

COMPATIBILITY: The 8086 and 8088 are 100% compatible in machine and assembly languages.

SEGMENTS: Memory segments are 64K byte sections of the full megabyte space. Four segments are assigned to code, stack, data, and extra data. Their physical location is given by their respective registers (CS, SS, DS, ES) times 16. Locations within a segment are specified by a 16-bit offset (or logical) address relative to the beginning of the segment.

ALIGNMENT: On both processors, words can start at even or odd addresses. However, on the 8086, each load or store of an odd aligned word adds 4 cycles to execution time. 8086 programs should at least align the stack.

DESTINATION (,) SOURCE: Instructions that take data from some "source" and put a result at some "destination" are written in the form: MNEMONIC DESTINATION,SOURCE

BYTE ORDER: Two byte and two word values, displacements, and addresses in code, stack, jump-table, and data areas are stored with Least significant half at Lower address.

RELATIVE JUMPS: The destination address of a relative jump is the sum of the signed displacement and the address of the first byte of the next instruction.

STRING POINTERS: For string operations, while SI points into the DATA segment, note that DI points into the EXTRA segment.

BP FOR STACK: When register BP is specified in an instruction, the variable is assumed to reside in the STACK segment.

RESERVED PORTS: Ports 00F8H thru 00FFH of the 64K I/O locations are reserved for Intel products.

INTERRUPT NOTES: When a segment register and another value must be updated together without the possibility of an intervening interrupt (e.g. SS and SP), the segment register should be changed first and followed immediately by the instruction that updates the other value. (Interrupts are not recognized immediately after a move to segment register, POP segment register, or prefix instruction.) Interrupts are accepted and handled properly during repeated string operations provided additional prefixes are not used (and assuming there are no algorithmic conflicts with string data). The NMI and INTR interrupt lines are respectively edge and level triggered.

RESET: A hardware Reset sets CS=FFFF, DS=SS=ES=0000, FLAGS=0, and starts executing code at location FFFF0.

ROTATES AND SHIFTS: All single-bit rotates and shifts set OF=1 if the MSB (sign bit) is changed by the operation. If the sign bit retains its original value, OF is cleared. OF is undefined after multi-bit operations.

PARITY FLAG: The parity flag reflects the parity of only the low order 8 bits of results. (Flag is set if even number of one-bits, cleared if odd.)

BCD TERMS: Packed BCD and Unpacked BCD have respectively two and one binary coded decimal digits per byte. Unpacked BCD + 30H yields ASCII.

LOGICAL INSTRUCTIONS: AND, OR, TEST, and XOR instructions clear the OF and CF flags.

SEGMENT OVERRIDE EXCEPTIONS: A segment override prefix can be attached to instructions (placed just before the opcode byte) to cause data to be accessed at any of the three alternatives to the default segment except for: stack operations; string destinations; and instruction fetches.

DERIVATION: This card is based on Intel publications.

About the Tables

FLAG CODES TABLE: In the FLAG CODES table, 'U' indicates that the flag becomes undefined. Otherwise the listed flag is affected according to the operation.

INSTRUCTION DESCRIPTION TABLE: The single letter column corresponds to the leftmost column of the FLAG CODES table.

HEX COLUMN OF INSTRUCTION SET: Non-HEX values for the second byte refer to sections of the SECOND BYTE TABLE (see below). Following the listed opcode byte(s) go an immediate displacement or address if applicable and finally immediate data if applicable.

'C' COLUMN OF INSTRUCTION SET: These codes refer to the CYCLE CODES table.

CYCLE CODES TABLE: Listed numbers are instruction execution times in CPU cycles. When 8086 and 8088 times differ, the 8086 time is given first and the 8088 is given on the next line. A '+' terminator indicates to add calculation time for the effective address per section 'T' of the SECOND BYTE TABLE. For the 8086, the number in parenthesis applies when word data is at an odd address. 8088 times assume the stack at an even address. A '—' indicates a min to max range. X:Y are times for FAILURE:SUCCESS. Note that several factors can increase execution time over the figures shown. A series of fast executing instructions can drain the instruction queue and increase execution time; and instruction prefetch can conflict with memory data access also increasing execution time. The actual time for a code sequence is claimed to typically be within 5-10% of the theoretical time although in special cases it can be much more. For the 8086, instruction alignment can affect speed in some cases but usually not substantially.

SECOND BYTE TABLE: This table allows conversion to and from hex of the second byte (excluding prefixes) of an instruction. The table is referred to by other parts of this card in such forms as X9, 9X, X0, MX, KX, etc. X9, for example, directs you to find the first operand of the instruction being converted in section X, and the second operand in section 9. (Section 9 is located below number matrix.) The machine code is then found at the intersection. X0 sends you to X for the ONLY operand and across to section 0 for the machine code. For disassembly, first find the machine code in the number matrix and determine the instruction from the associated points in the indicated sections. When assembling, make sure register values are taken from the bottom part of section X while register pointers are taken from the upper parts.

HEX TO INSTRUCTION TABLE: To convert from hex to an instruction, scan down for the first digit (MSD) and across for the second. Two-character codes (upper case) in the table refer to sections of the SECOND BYTE TABLE but only when they appear on the first of the two lines of an entry. On the second line, two-character codes refer to registers.

ADDRESSING COLUMN OF INSTRUCTION SET:

- r = byte register
- rr = word register
- i = immediate byte value
- ii = immediate word value
- d = immediate signed byte displacement
- dd = immediate signed word displacement
- aa = immediate two byte address (offset from segment start) (address can be of byte or word)
- aaaa = immediate four byte address (2 byte offset followed by 2 byte segment address/16)
- m = memory byte specified by memory pointers of section X of SECOND BYTE TABLE
- mm = memory word specified by memory pointers of section X of SECOND BYTE TABLE. (With CALL or JMP instructions memory has 2 byte offset from segment start of point to go to.)
- x = reg or mem byte
- xx = reg or mem word
- sr = segment register
- dw = memory double-word specified by memory pointers of section X of SECOND BYTE TABLE. (With CALL or JMP instructions memory has 2 byte offset from segment start followed by 2 byte segment address/16 of point to go to.)
- ws = within segment
- as = another segment
- () = data in mem

Where 'byte' or 'word' is listed, the assembler may require a dummy reference to labels.

Instruction Description

Flag Codes

- A = A C OU PU SU ZU
- B = AU CU OU P S Z
- C = A C O P S Z
- D = AU C O P S Z
- E = EVERY FLAG
- F = NO OTHERS
- G = A O P S Z
- H = C O
- I = I T
- J = A C P S Z
- K = AU CU OU PU SU ZU
- L = AU C O PU SU ZU
- M = A C OU P S Z
- N = NONE

Flags

- A = Aux carry flag
- C = Carry flag
- D = Direction flag
- I = Interrupt enable
- O = Overflow flag
- P = Parity flag
- S = Sign flag
- T = Trap flag
- Z = Zero flag

Registers

AX	AH	AL
BX	BH	BL
CX	CH	CL
DX	DH	DL

SP	STACK POINTER
BP	BASE POINTER
SI	SOURCE INDEX
DI	DESTINATION INDEX

IP	INSTRUCTION PNTR
F	----ODITSZ-A-P-C

CS	CODE SEGMENT
DS	DATA SEGMENT
SS	STACK SEGMENT
ES	EXTRA SEGMENT

Intentionally Blank

AAA	A	ASCII adjust for addition— Restores AL to unpacked BCD after addition of unpacked BCD number (byte add only)
AAD	B	ASCII adjust for division — Modifies AL *before* dividing unpacked BCD numbers so that result will be unpacked BCD (div by byte only) (AL=AH*OAH+AL;AH=0) AAD followed by DIV only produces unpacked BCD quotients between 0-9. Quotients above 9 will be in normal binary. For unpacked BCD quotients between 00-99 use: AAD; DIV; MOV REM,AH; AAM
AAM	B	ASCII adjust for multiplication — Restores AH and AL to unpacked BCD after multiplication of unpacked BCD numbers (byte mul only)
AAS	A	ASCII adjust for subtraction — Restores AL to unpacked BCD after subtraction of unpacked BCD number (byte sub only)
ADC	C	Add with carry
ADD	C	Add
AND	D	Logical AND (clears CF, OF)
CALL	N	Call procedure (pushes return addr)
CBW	N	Convert byte to word — Extends sign bit of AL throughout AH
CLC	F	Clear carry flag
CLD	F	Clear direction flag — Prepares for auto increment of SI and DI during string-op
CLI	F	Clear interrupt-enable flag — disables interrupts (except NMI and software interrupts)
CMC	F	Complement carry flag
CMP	C	Compare destination with source — JG, for example, will jump if destination is greater than source
CMPS	C	Compare string — Compares bytes or words pointed to by DI and SI and updates DI and SI by 1 or 2 accordingly - JG, for example, will jump if destination is greater than source
CWD	N	Convert word to double-word — Extends sign bit of AX throughout DX
DAA	M	Decimal adjust for addition — Restores AL to packed BCD after addition of packed BCD number (byte add only)
DAS	M	Decimal adjust for subtraction — Restores AL to packed BCD after subtraction of packed BCD number (byte sub only)
DEC	G	Decrement by one
DIV	K	Divide unsigned — (AL=AX/source; AH=rem) or (AX = DX:AX / source; DX = rem) Type 0 interrupt if div by 0 or quotient too large
ESC	N	Escape — for instructions to coprocessor
HALT	N	Halt and wait for interrupt
IDIV	K	Divide signed — see DIV
IMUL	L	Multiply signed — (AX = AL * source) or (DX:AX = AX * source) CF and OF are set when upper half of result has significant bits.
IN	N	Input from port
INC	G	Increment by one
INT	I	Interrupt — Activates 1 of 256 interrupt routines by software (clears IF, TF) (As with hardware Int, flags are saved on stack)
INTO	N	Interrupt if overflow — If OF=1 then INT 4 (clears IF, TF if successful)
IRET	E	Interrupt return - Returns from interrupt procedure whether activated by hardware or software (flags are restored from stack)
JA	N	Jump if above - unsigned
JAE	N	Jump if above or equal - unsigned
JB	N	Jump if below - unsigned
JBE	N	Jump if below or equal - unsigned
JC	N	Jump if carry - If CF=1
JCXZ	N	Jump if CX register zero
JE	N	Jump if equal - If ZF=1
JG	N	Jump if greater - signed
JGE	N	Jump if greater or equal - signed
JL	N	Jump if less - signed
JLE	N	Jump if less or equal - signed
JMP	N	Jump unconditionally
JNA	N	Jump if not above - unsigned
JNAE	N	Jump if not above nor equal - unsigned
JNB	N	Jump if not below - unsigned
JNBE	N	Jump if not below nor equal - unsigned
JNC	N	Jump if no carry - If CF=0
JNE	N	Jump if not equal - If ZF=0
JNG	N	Jump if not greater - signed
JNGE	N	Jump if not greater nor equal - signed
JNL	N	Jump if not less - signed
JNLE	N	Jump if not less nor equal - signed
JNO	N	Jump if no overflow - If OF=0
JNP	N	Jump if not parity - If PF=0
JNS	N	Jump if not sign - If SF=0
JNZ	N	Jump if not zero - If ZF=0
JO	N	Jump if overflow - If OF=1
JP	N	Jump if parity - If PF=1
JPE	N	Jump if parity even - If PF=1
JPO	N	Jump if parity odd - If PF=0
JS	N	Jump if sign - If SF=1
JZ	N	Jump if zero - If ZF=1
LAHF	N	Load AH from low byte of flags
LDS	N	Load pointer using DS - A double word pointer located in memory is moved into a register (first word) and register DS (second word)
LEA	N	Load effective address - The address (offset from beginning of segment) of the source operand (as opposed to its value) is loaded into a register
LES	N	Load pointer using ES - Similar to LDS
LOCK	N	Lock bus - A prefix causing CPU to assert LOCK signal during execution of prefixed instruction
LODS	N	Load string - Loads byte or word pointed to by SI into AL or AX and updates SI by 1 or 2 accordingly
LOOP	N	Loop - Decrement CX by one and jump if CX not zero
LOOPE	N	Loop while equal - Decrement CX and jump if CX not zero and ZF=1
LOOPNE	N	Loop while not equal - Same as LOOPNZ
LOOPNZ	N	Loop while not zero - Decrement CX and jump if CX not zero and ZF=0
LOOPZ	N	Loop while zero - Same as LOOPE
MOV	N	Move - Moves to destination from source
MOVS	N	Move string - Moves byte or word pointed to by SI to location pointed to by DI and updates pointers by 1 or 2 accordingly
MOVSB	N	Move string byte - See MOVS
MOVSW	N	Move string word - See MOVS
MUL	L	Multiply unsigned - See IMUL
NEG	C	Negate - two's complement (multiply by -1)
NOP	N	No operation
NOT	N	Logical NOT - one's complement
OR	D	Logical OR (clears CF, OF)
OUT	N	Output to port
POP	N	Pop word from stack - SP increases after access
POPF	E	Pop flags from stack
PUSH	N	Push word onto stack - SP decreases first
PUSHF	N	Push flags onto stack
RCL	H	Rotate thru carry left - by 1 or by CL
RCR	H	Rotate thru carry right - by 1 or by CL
REP	N	Repeat prefix - See below
RET	N	Return from procedure - Not for use with interrupt procedures - Optional pop-value (usually even #) is added to SP to dump passed parameters
ROL	H	Rotate left - by 1 or by CL - CF = LSB of result
ROR	H	Rotate right - by 1 or by CL - CF = MSB of result
SAHF	J	Store AH into low byte of flags
SAL	D	Shift arithmetic left - zero fill - by 1 or by CL — CF = last bit shifted out
SAR	D	Shift arithmetic right - sign extension - by 1 or by CL - CF = last bit shifted out - note that negative numbers are rounded differently from IDIV by 2
SBB	C	Subtract with borrow - destination minus source
SCAS	C	Scan string - Compares AL or AX with byte or word pointed to by DI and updates DI by 1 or 2 accordingly - JG, for example, will jump if AL or AX is greater than string element
SHL	D	Shift logical left - Same as SAL
SHR	D	Shift logical right - zero fill - by 1 or by CL - CF = last bit shifted out
STC	F	Set carry flag
STD	F	Set direction flag - Prepares for auto decrement of SI and DI during string-op
STI	F	Set interrupt-enable flag - enables interrupts after next instruction
STOS	N	Store string - Stores AL or AX into location pointed to by DI and updates DI by 1 or 2 accordingly
SUB	C	Subtract - destination minus source
TEST	D	Test by logical AND - affects only flags (also clears CF, OF)
WAIT	N	CPU enters WAIT state
XCHG	N	Exchange - switches contents of destination and source
XLAT	N	Translate - Replaces AL with a byte from a table pointed to by BX. AL initially gives the position in the table. The first element is at position 0.
XOR	D	Logical Exclusive-OR - Differing bits yield one - Like bits yield zero (clears CF, OF)

REPEAT INSTRUCTIONS

REP LODS	Repeats LODS CX times (1) - N4 cycles for bytes, P4 for words
REP MOVS	Repeats MOVS CX times (1) - F4 cycles for bytes, G4 for words
REP STOS	Repeats STOS CX times (1) - H4 cycles for bytes, I4 for words
REPE CMPS	Repeats CMPS until strings mismatch but not more than CX times (1) - J4 cycles for bytes, K4 for words
REPE SCAS	Repeats SCAS until AL or AX mismatches string but not more than CX times (1) - L4 cycles for bytes, M4 for words
REPNE CMPS	Repeats CMPS until strings match but not more than CX times (1) - J4 cycles for bytes, K4 for words
REPNE SCAS	Repeats SCAS until AL or AX matches string but not more than CX times (1) - L4 cycles for bytes, M4 for words
REPNZ CMPS	Same as REPNE CMPS
REPNZ SCAS	Same as REPNE SCAS
REPZ CMPS	Same as REPE CMPS
REPZ SCAS	Same as REPE SCAS

(1) CX is decremented each time. DI (and SI) end up pointing one position past end of string(s) or past first point of match/mismatch. ZF does not have to be setup before using these instructions.