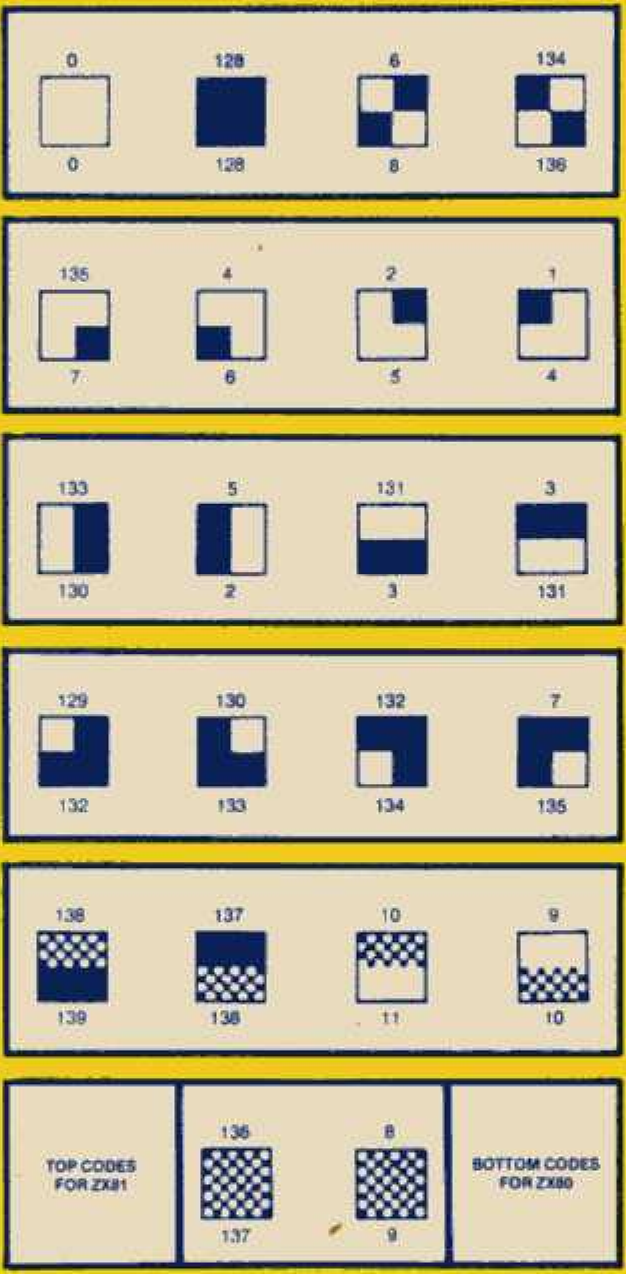
NANOS SYSTEMS CORP.

REFERENCE CARD FOR THE ZX80®, ZX81® AND TIMEX SINCLAIR 1000™



☆ = ZX80 ONLY **BASIC STATEMENTS** * = ZX81 ONLY

STATEMENT • DESCRIPTION

CLEAR • Clear all program variables.
CLS • Clear screen.
DIM *var (num)* • Define array "var" with "num" entries.
DIM *var (num [, num...]) • Define array "var" with "num" entries.
FOR *var*=*num1* TO *num2* • Loop thru NEXT until var exceeds num2.
FOR *var*=*num1* TO *num2* STEP *incr • Loop thru NEXT, stepping "incr".
GOSUB *line* • Perform rtn at "line" until RETURN.
GOTO *line* • Branch to "line".
IF *cond* THEN *action* • Execute action based on condition.
INPUT *var* • Capture keyed reponse into "var".
LET *var*=*value* • Assign a value to variable.
LPRINT *item [; [,[*item*. . .] • Print data.
LPRINT TAB *num;item [; [, [**TAB**. . .] • Print with tabs.
LPRINT AT *line,col;item [; [, [**AT** . . .] • Print at position.
NEXT *var* • End of FOR loop. Increment var.
PAUSE *frames • Halt execution until time-out.
***PAUSE 40000** • Halt execution until key pressed.
PLOT *(horiz, vert) • Turn on graphics block at this location.
POKE *addr, num* • Store number at address.
PRINT *item* [; [, [*item* . . .] • Display data.
PRINT TAB *num;item [; [, [**TAB** . . .] • Display with tabs.
PRINT AT *line,col;item [; [, [**AT** . . .] • Display at position.
RAND *num* • Re-seed random number generator.
REM • Remarks follow this statement.
RETURN • Return from a GOSUB.
***SCROLL** • Scroll 1 line toward top of screen.
STOP • Interrupt program execution.
UNPLOT *(horiz, vert) • Turn off graphics block at this location.

PARENS NOT REQUIRED FOR PLOT/UNPLOT.

☆ = ZX80 ONLY **BASIC COMMANDS** * = ZX81 ONLY

COMMAND	DESCRIPTION
BREAK	Interrupt execution until CONT.
CONT	Continue after **BREAK** or **STOP**.
***COPY**	Print screen contents on the printer.
EDIT	Edit current program line.
***FAST**	Enter high-speed execution mode.
***FUNCTION**	Enter function mode.
***GRAPHICS**	Enter or exit graphics mode.
LIST [lineno]	Display program lines on screen.
***LLIST [lineno]**	Print program lines on printer.
☆**LOAD**	Load first program found on cassette.
***LOAD "name"**	Load program "name" from cassette.
NEW	Clear memory and restart.
RUN [lineno]	Execute program.
☆**SAVE**	Save program from memory to tape.
***SAVE "name"**	Save program called "name" to tape.
***SLOW**	Enter slow-speed execution mode.
EDIT SUBCOMMANDS	
⇨	Forward-space cursor.
⇦	Backspace cursor.
⇧	Scroll up.
⇩	Scroll down.
lineno	Delete line "lineno".
***DELETE**	Backspace and erase.
☆**HOME**	Move cursor to line 0.
☆**RUBOUT**	Backspace and erase.
CURSORS AND MARKERS	
>	Current program line marker.
*F	Function prompter.
*G	Graphics prompter.
K	Keyword prompter.
L	Data/character prompter.
S	Syntax error marker.

☆ = ZX80 ONLY	BASIC FUNCTIONS * = ZX81 ONLY
FUNCTION	**RETURNS**
	CHARACTER/STRING FUNCTIONS
CHR$ (num)	Character whose decimal value is num. ("A" = 38)
*LEN (string)	Length of string.
STR$ (num)	Character string of num.
☆TL$ (string)	String without the first character.
*string ([n1] [TO] [n2])	Substr of string from pos. n1 to n2.
	NUMERIC FUNCTIONS
ABS (num)	Absolute value of number. (−3.25 = 3.25)
CODE (string)	Decimal value of first char in string.
*INT (num)	Nearest whole number after rounding down. (2.5 = 2) (−2.5 = −3)
*RND	Random number between 0 and 1.
☆RND (num)	Random number between 0 and num +1.
☆RND (0)	1.
*SGN (num)	0 (Zero), 1 (Positive), − 1 (Negative).
*VAL (String)	Number extracted from character string.
	MATH FUNCTIONS
*ACS (num)	Angle whose cosine is num, in radians.
*ASN (num)	Angle whose sine is num, in radians.
*ATN (num)	Angle whose tangent is num, in radians.
*COS (angle)	Cosine of angle radians.
*EXP (num)	Inverse LN of num.
*LN (num)	Natural logarithm of num. (Base 2.718281828)
*PI	3.14159265.
*SIN (angle)	Sine of angle radians.
*SQR (num)	Square root of num.
*TAN (angle)	Tangent of angle radians.
Radians = Degrees/57.29577951	Degrees = Radians * 57.29577951
	I/O AND OTHER FUNCTIONS
*INKEY$	Character value of key pressed on keyboard.
PEEK (addr)	Decimal value of byte at location "addr".
USR (addr)	Branch to machine language rtn at "addr".
PARENS IN FUNCTIONS ABOVE ARE OPTIONAL IN MOST CASES.	

DERIVED FUNCTIONS (X is in radians.)

FUNCTION	EXPRESSION
SECANT	1/COS(X)
COSECANT	1/SIN(X)
COTANGENT	1/TAN(X)
	INVERSE
SINE	ATN(X/SQR(−X*X + 1))
COSINE	−ATN(X/SQR(−X*X + 1)) + 1.570796
SECANT	ATN(SQR(X*X − 1)) + (SGN(X) −1) * 1.570796
COSECANT	ATN(1/SQR(X*X −1)) + (SGN(X) −1)*1.570796
COTANGENT	−ATN(X) + 1.570796
	HYPERBOLIC
SINE	(EXP(X) − EXP(−X))/2
COSINE	(EXP(X) + EXP(−X))/2
TANGENT	−EXP(−X)/(EXP(X) + EXP(−X))*2 + 1
SECANT	2/(EXP(X) + EXP(−X))
COSECANT	2/(EXP(X) − EXP(−X))
COTANGENT	EXP(−X)/(EXP(X) − EXP(−X))*2 + 1
	INVERSE HYPERBOLIC
SINE	LN (X + SQR(X*X + 1))
COSINE	LN (X + SQR(X*X − 1))
TANGENT	LN ((1 +X)/(1 −X))/2
SECANT	LN ((SQR(−X*X + 1) + 1)/X)
COSECANT	LN ((SGN(X)*SQR(X*X + 1) + 1)/X)
COTANGENT	LN ((X + 1)/(X − 1))/2

BASIC SPECIAL CHARACTERS AND OPERATORS

;	Suppress tab after PRINT.
,	Tab to next column after PRINT.
blank	Tab to next line after PRINT.
"" ""	Double-quote char. in a string.
""	Character-string delimiter.
$	Identifies character-string variable. (A$)
blank	Identifies a numeric variable. (A)
E	Scientific notation. (1.7E38)
( )	Denotes priority in order of operations.
=	Equal or assignment.
+	Addition, concatenation.
−	Subtraction or minus sign.
*	Multiplication.
**	Exponentiation.
/	Division.
>	Greater than.
<	Less than.
>=	Greater than or equal to.
<=	Less than or equal to.
<>	Not equal.
NOT	Reverses true/false result.
AND	If both expr are true, result is true.
OR	If either or both expr are true, result is true.
NON-ZERO	True.
ZERO	False.

Order of operations: ( ), **, − **(negation)**, *, /, + −, =, >, <, **NOT, AND, OR.**

SCREEN LAYOUT

LINE	DISPLACEMENT	Y-COORD	LINE	DISPLACEMENT	Y-COORD
0	0	43 42	12	396	19 18
1	33	41 40	13	429	17 16
2	66	39 38	14	462	15 14
3	99	37 36	15	495	13 12
4	132	35 34	16	528	11 10
5	165	33 32	17	561	9 8
6	198	31 30	18	594	7 6
7	231	29 28	19	627	5 4
8	264	27 26	20	660	3 2
9	297	25 24	21	693	1 0
10	330	23 22	22	726	
11	363	21 20	23	759	

DEC	HEX	ZX80	ZX81	DEC	HEX	ZX80	ZX81
0	00	SPACE	SPACE	64	40		RND
1	01	"	▘	65	41		INKEY$
2	02	▌	▝	66	42		PI
3	03	▄	▀	67	43		
4	04	▘	▖	68	44		
5	05	▝	▌	69	45		
6	06	▖	▞	70	46		
7	07	▗	▛	71	47		
8	08	▞	▒	72	48		
9	09	▒	⬓	73	49		
10	0A	⬓	⬒	74	4A		
11	0B	⬒	"	75	4B		
12	0C	£	£	76	4C		
13	0D	$	$	77	4D		
14	0E	:	:	78	4E		
15	0F	?	?	79	4F		
16	10	(	(	80	50		
17	11	)	)	81	51		
18	12	−	>	82	52		
19	13	+	<	83	53		
20	14	*	=	84	54		
21	15	/	+	85	55		
22	16	=	−	86	56		
23	17	>	*	87	57		
24	18	<	/	88	58		
25	19	;	;	89	59		
26	1A	,	,	90	5A		
27	1B	.	.	91	5B		
28	1C	0	0	92	5C		
29	1D	1	1	93	5D		
30	1E	2	2	94	5E		
31	1F	3	3	95	5F		
32	20	4	4	96	60		
33	21	5	5	97	61		
34	22	6	6	98	62		
35	23	7	7	99	63		
36	24	8	8	100	64		
37	25	9	9	101	65		
38	26	A	A	102	66		
39	27	B	B	103	67		
40	28	C	C	104	68		
41	29	D	D	105	69		
42	2A	E	E	106	6A		
43	2B	F	F	107	6B		
44	2C	G	G	108	6C		
45	2D	H	H	109	6D		
46	2E	I	I	110	6E		
47	2F	J	J	111	6F		
48	30	K	K	112	70		⇧
49	31	L	L	113	71		⇩
50	32	M	M	114	72		⇦
51	33	N	N	115	73		⇨
52	34	O	O	116	74	HOME	GRAPHICS
53	35	P	P	117	75	EDIT	EDIT
54	36	Q	Q	118	76	NEWLINE	ENTER
55	37	R	R	119	77	RUBOUT	DELETE
56	38	S	S	120	78		K/L mode
57	39	T	T	121	79		FUNCTION
58	3A	U	U	122	7A		
59	3B	V	V	123	7B		
60	3C	W	W	124	7C		
61	3D	X	X	125	7D		
62	3E	Y	Y	126	7E		number
63	3F	Z	Z	127	7F		cursor

ZX80 ONLY : 16 – 26 , 64 – 212 Not available from keyboard

DEC	HEX	ZX80	ZX81	DEC	HEX	ZX80	ZX81
128	80	█	█	192	C0		""
129	81	"	▟	193	C1		AT
130	82	▐	▙	194	C2		TAB
131	83	▀	▄	195	C3		
132	84	▟	▜	196	C4		CODE
133	85	▙	▐	197	C5		VAL
134	86	▜	▚	198	C6		LEN
135	87	▛	▗	199	C7		SIN
136	88	▚	▒	200	C8		COS
137	89	▒	⬒	201	C9		TAN
138	8A	⬒	⬓	202	CA		ASN
139	8B	⬓	"	203	CB		ACS
140	8C	£	£	204	CC		ATN
141	8D	$	$	205	CD		LN
142	8E	:	:	206	CE		EXP
143	8F	?	?	207	CF		INT
144	90	(	(	208	D0		SQR
145	91	)	)	209	D1		SGN
146	92	−	>	210	D2		ABS
147	93	+	<	211	D3		PEEK
148	94	*	=	212	D4	"	USR
149	95	/	+	213	D5	THEN	STR$
150	96	=	−	214	D6	TO	CHR$
151	97	>	*	215	D7	;	NOT
152	98	<	/	216	D8	,	**
153	99	;	;	217	D9	)	OR
154	9A	,	,	218	DA	(	AND
155	9B	.	.	219	DB	NOT	<=
156	9C	0	0	220	DC	−	>=
157	9D	1	1	221	DD	+	<>
158	9E	2	2	222	DE	*	THEN
159	9F	3	3	223	DF	/	TO
160	A0	4	4	224	E0	AND	STEP
161	A1	5	5	225	E1	OR	LPRINT
162	A2	6	6	226	E2	**	LLIST
163	A3	7	7	227	E3	=	STOP
164	A4	8	8	228	E4	>	SLOW
165	A5	9	9	229	E5	<	FAST
166	A6	A	A	230	E6	LIST	NEW
167	A7	B	B	231	E7	RET	SCROLL
168	A8	C	C	232	E8	CLS	CONT
169	A9	D	D	233	E9	DIM	DIM
170	AA	E	E	234	EA	SAVE	REM
171	AB	F	F	235	EB	FOR	FOR
172	AC	G	G	236	EC	GOTO	GOTO
173	AD	H	H	237	ED	POKE	GOSUB
174	AE	I	I	238	EE	INPUT	INPUT
175	AF	J	J	239	EF	RAND	LOAD
176	B0	K	K	240	F0	LET	LIST
177	B1	L	L	241	F1		LET
178	B2	M	M	242	F2		PAUSE
179	B3	N	N	243	F3	NEXT	NEXT
180	B4	O	O	244	F4	PRINT	POKE
181	B5	P	P	245	F5		PRINT
182	B6	Q	Q	246	F6	NEW	PLOT
183	B7	R	R	247	F7	RUN	RUN
184	B8	S	S	248	F8	STOP	SAVE
185	B9	T	T	249	F9	CONT	RAND
186	BA	U	U	250	FA	IF	IF
187	BB	V	V	251	FB	GOSUB	CLS
188	BC	W	W	252	FC	LOAD	UNPLOT
189	BD	X	X	253	FD	CLEAR	CLEAR
190	BE	Y	Y	254	FE	REM	RETURN
191	BF	Z	Z	255	FF		COPY

ZX81 - SELECTED ROM CALLS

TO USE, POKE BYTES INTO ANY SAFE RAM, AND CALL VIA USR FUNCTION (LET A=USR(addr)). RESULTS RETURNED AS FUNCTION VALUE, AND IN BC REGISTER.

TO SCAN KEYBOARD FASTER THAN INKEY$

HEX	DEC	CODE
CD BB 02	205 187 2	CALL 02BBH
7C	124	LD A,H
C6 02	198 2	ADD A,2
38 09	56 9	JR C,+9
44	68	LD B,H
4D	77	LD C,L
CD BD 07	205 189 7	CALL 07BDH
06 00	6 0	LD B,0
4E	78	LD C,(HL)
D8	216	RET C
01 00 00	1 0 0	LD BC,0
C9	201	RET

TO MOVE CURSOR TO A ROW, COLUMN

01 cl rw	1 cl rw	LD BC,row col
CD F5 08	205 245 8	CALL 08F5H
C9	201	RET

TO OUTPUT A CHARACTER TO SCREEN

3E nn	62 nn	LD A,nn (nn=character)
D7	215	RST 0010H
C9	201	RET

TO OUTPUT CHARACTER STRING TO SCREEN

11 dd dd	17 dd dd	LD DE,addr of string (low byte first)
01 dd dd	1 dd dd	LD BC,length of string (low byte first)
CD 6B 0B	205 107 11	CALL 0B6BH
C9	201	RET

TO PLOT

01 xxyy	1 xx yy	LD BC,yyxx
3E 9B	62 155	LD A,9BH
CD B2 0B	205 178 11	CALL 0BB2H
C9	201	RET

TO UNPLOT

01 xx yy	1 xx yy	LD BC,yyxx
3E A0	62 160	LD A,A0H
CD B2 0B	205 178 11	CALL 0BB2H
C9	201	RET

TO SET "FAST"

CD 20 0F	205 32 15	CALL 0F20H
C9	201	RET

TO SET "SLOW"

CD 28 0F	205 40 15	CALL 0F28H
C9	201	RET

HOW TO USE *FOR . . . NEXT*

The *FOR* statement sets up conditions for executing a series of BASIC lines over and over again until the ending conditions are met.

```
  A SAMPLE OF USAGE:  10 FOR A = 0 TO 30
                      20 Z = Z + Y
                      30 R = R + Z
                      40 NEXT A
                      50 PRINT M
```

When you enter: FOR A = 0 TO 30
You are really saying:
    Assign the value "0" to A.
    Execute code beginning at line 20.
    When you get to the "NEXT A" statement, add 1 to A.
    Then, if A is greater than 30, go to the next line (50).
    Otherwise, go back to line 20.
YOU CAN ENTER THE STATEMENT MANY WAYS:

```
        FOR C = A TO B
        FOR X = 1 TO W
        FOR P = L TO 10
```

HOW TO USE *FOR . . . STEP . . . NEXT*

USAGE IS THE SAME AS **FOR . . . NEXT** ABOVE,
EXCEPT: **FOR A = 0 TO 30 STEP 2**
    WHEN YOU GET TO THE "NEXT A" STATEMENT,
    2 IS ADDED TO A.
    OTHER FORMS CAN STEP IN A NEGATIVE INCREMENT,
    FOR EXAMPLE:
        FOR A = 30 TO 0 STEP −2

HOW TO USE MORE THAN 1 *FOR* AT A TIME

WHEN YOU USE MORE THAN ONE **FOR** AT A TIME, IT IS CALLED "NESTING".
WHEN NESTING, EACH **FOR** HAS ITS OWN **NEXT**.
THE LAST **FOR** ENTERED MUST FIND ITS OWN **NEXT** FIRST, OR YOU HAVE BAD PROCESSING.

```
        FOR A=1 TO 4
        FOR B =1 TO 7
        FOR C=5 TO 10
        NEXT C
        NEXT B
        NEXT A
```

What happens here, is:
    The **A** loop is entered. It loops 4 times.
    The **B** loop is called 4 times by the **A** loop.
    Each time the **A** loop calls it, the **B** loop goes 7 times.
    The **C** loop is called 7 times by the **B** loop.
    Each time the **B** loop calls it, the **C** loop goes 6 times.
    So, **A** loops 4 times, **B** loops 28 times, **C** loops 168 times.

HOW TO USE *IF . . . THEN*

The **IF** statement allows you to compare items against each other and **THEN** take an action based upon the results of the compare.
You can test for equal, less than, greater than, or any combination of the three.
You can also combine tests for a complex compare.
You can also combine **IF**s by making the next one the action.
This is called "NESTING".

Example of a simple **IF** statement:

```
        IF A = B THEN LET C = D
        IF A NOT = B THEN GOTO 1000
        IF A > B THEN PRINT L$;
        IF A < B THEN STOP
        IF A<>B THEN GOSUB 2000
```

Example of a complex **IF** statement:

```
     IF A = B AND C = D THEN GOSUB 3000        (Both required)
     IF A = B OR C = D THEN GOTO 4000          (Only one required)
     IF A = B AND C = D OR E = F THEN PRINT M$;
                  (First two, or last one required)
```

When using **AND**, then every test must be true to take the action.
When using **OR**, then only one of the tests must be true.
Of course, if there are **AND**s, then they must all be true before the **OR** makes its decision.
When none of the tests fits the condition for the **THEN**, control of your program falls through to the next line.

Example of combined **IF**s:

```
        IF A = B THEN IF C = D THEN IF E = F THEN GOTO 5000
```

This is easier to see if written as follows:

```
    IF A = B THEN
      IF C = D THEN
        IF E = F THEN
          GOTO 5000
```

At each level, when the condition is not true, then the next **IF** is not tested. Instead, all of this code is skipped, and the computer goes to the next line.
When "NESTING" **IF**s, it is the same as using **AND** without the nesting.

Another example:

```
        IF A = B THEN IF C NOT = D OR E > F THEN PRINT R$;
```

If **A** equals **B** then we ask the next **IF** question, otherwise we go to the next line.
In the next **IF** question, if **C** is not equal to **D** then we will print.
If **C** is equal to **D**, then we ask if **E** is greater than **F**.
If **E** is greater than **F**, then we will print.
Otherwise, we will go to the next line.

ZX80 MEMORY MAP

ADDRESS		
DECIMAL	HEX	DESCRIPTION
16384	4000H	ERROR-CODE MINUS ONE.
16385	4001H	*BASIC SYSTEM CONTROL FLAG BITS.
16386	4002H	CURRENT BASIC STATEMENT NUMBER.
16388	4004H	ADDRESS OF K OR L CURSOR.
16390	4006H	BASIC STATEMENT NUMBER AT ❯ CURSOR.
16392	4008H	*ADDRESS OF PROGRAM VARIABLES.
16394	400AH	*ADDRESS OF WORKING STORAGE (KEY INPUT)
16396	400CH	*ADDRESS OF UPPER SCREEN.
16398	400EH	*ADDRESS OF LOWER SCREEN.
16400	4010H	*ADDRESS OF END-OF-SCREEN.
16402	4012H	*NUMBER OF LOWER-SCREEN LINES.
16403	4013H	NUMBER OF FIRST BASIC STMT ON SCREEN.
16405	4015H	ADDRESS OF S MARK MINUS ONE.
16407	4017H	NUMBER OF STMT TO "CONTINUE" AT.
16409	4019H	SYNTAX FLAG BITS.
16410	401AH	SYNTAX TABLE POINTER.
16412	401CH	RANDOM NUMBER SEED.
16414	401EH	SCREEN FRAME DISPLAY COUNT.
16416	4020H	ADDRESS OF FIRST CHAR OF FIRST VAR NAME IN LAST DIM, FOR, INPUT, LET, NEXT.
16418	4022H	VALUE OF LAST VAR OR EXPRESSION.
16420	4024H	*LINE POS OF NEXT SCREEN CHAR: FROM 33 (LEFT) TO 2 (RIGHT) 1 = FIRST COL, NEXT LINE (LINE FULL) 0 = FIRST COL, (E-O-LINE.)
16421	4025H	*CURRENT SCREEN LINE (0=BOT, 23=TOP)
16422	4026H	*ADDRESS OF CHAR AFTER PEEK OR POKE STMT.
16424	4028H	USER PROGRAM AREA.

* = DO NOT POKE. UNPREDICTABLE RESULTS.

ERROR CODES

ERROR CODES APPEAR AS: xx/yy
WHERE: xx is the error code,
AND: yy is the number of the last statement executed.

CODE	MEANING
0	Successful execution, or, GOTO-line too big.
1	NEXT has invalid variable, but, variable is assigned.
2	Variable not assigned, or, DIMensioned.
3	Bad subscript.
4	Memory exhausted.
5	Screen full.
6	Arithmetic number too large.
7	RETURN before GOSUB.
8	INPUT attempted in command mode. Illegal.
9	STOP was executed.
	ZX81 ONLY
A	Invalid parameter.
B	Invalid integer.
C	Invalid data in VAL string.
D	BREAK was pressed.
E	Unused.
F	SAVE name is a null string. Illegal.

ZX81 MEMORY MAP

ADDRESS		
DECIMAL	HEX	DESCRIPTION
0	0000H	MONITOR ROM.
8192	2000H	NOTHING. (USED FOR ROM IN SOME ADD-ON DEVICES).
16384	4000H	ERROR-CODE MINUS ONE.
16385	4001H	*BASIC SYSTEM CONTROL FLAG BITS.
16386	4002H	*ADDR OF NEXT INSTR AFTER "RETURN"ING.
16388	4004H	ADDR OF LAST AVAIL BASIC BYTE +1.
16390	4006H	CURSOR MODE — K, L, F, OR G.
16391	4007H	CURRENT BASIC STMT NUMBER.
16393	4009H	ROM VERSION CODE (0 = 8K).
16394	400AH	BASIC STMT NUMBER AT ❯ CURSOR.
16396	400CH	*ADDRESS OF SCREEN.
16398	400EH	ADDRESS OF NEXT SCREEN PRINT POS.
16400	4010H	*ADDRESS OF PROGRAM VARIABLES.
16402	4012H	ADDRESS OF ASSIGNMENT VARIABLE.
16404	4014H	*ADDR OF WORKING STORAGE (KEY INPUT).
16406	4016H	*ADDR OF BYTE AFTER PEEK OR POKE.
16408	4018H	ADDRESS OF S MARK MINUS ONE.
16410	401AH	*ADDRESS OF MATH CALC STACK.
16412	401CH	*ADDR OF END OF MATH CALC STACK.
16414	401EH	B-REGISTER OF CALCULATOR.
16415	401FH	ADDRESS OF CALCULATOR MEMORY.
16417	4021H	NOT USED.
16418	4022H	*NUMBER OF LOWER-SCREEN LINES.
16419	4023H	NUMBER OF FIRST BASIC STMT ON SCREEN.
16421	4025H	LAST KEY PRESSED.
16423	4027H	KEYBOARD DEBOUNCE STATUS.
16424	4028H	NUMBER OF BLANK LINES ABOVE AND BELOW MOVING GRAPHICS.
16425	4029H	*ADDR OF NEXT BASIC STMT LINE.
16427	402BH	NUMBER OF STMT TO "CONT"INUE AT.
16429	402DH	SYSTEM FLAG BITS.
16430	402EH	STRING-TYPE LENGTH IN ASSIGNMENT.
16432	4030H	ADDR OF NEXT SYNTAX TABLE ENTRY.
16434	4032H	RANDOM NUMBER SEED.
16436	4034H	SCREEN FRAME DISPLAY COUNT.
16438	4036H	LAST "PLOT" X-COORDINATE.
16439	4037H	LAST "PLOT" Y-COORDINATE.
16440	4038H	LSB OF ADDR OF NEXT "LPRINT" POSITION.
16441	4039H	*"PRINT" COLUMN NUMBER.
16442	403AH	*"PRINT" LINE NUMBER.
16443	403BH	INTERNAL FLAG BITS.
16444	403CH	PRINTER BUFFER.
16477	405DH	CALCULATOR AUXILIARY MEMORY AREA.
16507	407BH	NOT USED.
16509	407DH	USER PROGRAM AREA.
17407	43FFH	END OF 1K SYSTEMS.
18431	47FFH	END OF 2K SYSTEMS.
32767	7FFFH	END OF 16K SYSTEMS.

* = DO NOT POKE. UNPREDICTABLE RESULTS.

Addr 16393 (4009H) thru 16508 (407BH) are always SAVEd with the program.

Published 1982 by Nanos Systems Corp., P.O. Box 24344, Speedway, IN 46224 (317)244-4078.
Printed in U.S.A.

TIMING (ASSUMING 4MHZ)

MACHINE CYCLES		CLOCK PERIODS	MICRO SECONDS	YOUR TIME
1	A	4	1.00	
	B	5	1.25	
	C	6	1.50	
2	A	7=4+3	1.75	
	B	8=4+4	2.00	
	C	8=5+3	2.00	
	D	9=4+5	2.25	
	E	10=4+6	2.50	
3	A	10=4+3+3	2.50	
	A	11=4+3+4	2.75	
	C	11=4+4+3	2.75	
	D	11=5+3+3	2.75	
	E	12=4+3+5	3.00	
	F	12=4+4+4	3.00	
	G	13=5+3+5	3.25	
4	A	13=4+3+3+3	3.25	
	B	14=4+4+3+3	3.50	
	C	15=4+4+4+3	3.75	
	D	15=4+5+3+3	3.75	
	E	16=4+4+3+5	4.00	
	F	16=4+5+3+4	4.00	
5	A	16=4+3+3+3+3	4.00	
	B	17=4+3+4+3+3	4.25	
	C	18=4+4+3+4+3	4.50	
	D	19=4+3+4+3+5	4.75	
	E	19=4+4+3+5+3	4.75	
	F	20=4+4+3+5+4	5.00	
	G	21=4+4+3+5+5	5.25	
	H	21=4+5+3+4+5	5.25	
6	A	20=4+4+3+3+3+3	5.00	
	B	23=4+4+3+4+3+5	5.75	
	C	23=4+4+3+5+4+3	5.75	

To Calculate Your Own Timing:

(4MHZ/Your MHZ)∗Micro Sec. = Actual Microsec For Your Computer

(Make entry on right in chart above)

MATH INSTRUCTIONS

ADC—Add with Carry.
SBC—Subtract with Carry.

OPERANDS: *(r = A, B, C, D, E, H, or L)*

A,r^{1A} *HL,BC*^{4C}
A,imm^{2A} *HL,DE*^{4C}
A,(HL)^{2A} *HL,HL*^{4C}
A,(IX+d)^{5E} *HL,SP*^{4C}
A,(IY+d)^{5E}

Condition Set: **YES**

ADD—Add.

OPERANDS: *(r = A, B, C, D, E, H, or L)*

A,r^{1A} *HL,BC*^{3C} *IX,BC*^{4C} *IY,BC*^{4C}
A,imm^{2A} *HL,DE*^{3C} *IX,DE*^{4C} *IY,DE*^{4C}
A,(HL)^{2A} *HL,HL*^{3C} *IX,IX*^{4C} *IY,IY*^{4C}
A,(IX+d)^{5E} *HL,SP*^{3C} *IX,SP*^{4C} *IY,SP*^{4C}
A,(IY+d)^{5E}

Condition Set: **YES**

SUB—Subtract from Accumulator.

OPERANDS: *(r = A, B, C, D, E, H, or L)*

r^{1A} *imm*^{2A} *(HL)*^{2A} *(IX+d)*^{5E} *(IY+d)*^{5E}

Condition Set: **YES**

DEC—Decrement.
INC—Increment.

OPERANDS: *(r = A, B, C, D, E, H, or L)*

r^{1A} *BC*^{1C}
(HL)^{3C} *DE*^{1C}
(IX+d)^{6C} *HL*^{1C}
(IY+d)^{6C} *IX*^{2E}
IY^{2E}
SP^{1C}

Condition Set: **YES**
(Not for Register Pairs)

STORE REGISTER-INTO-MEMORY INSTRUCTIONS

LD—Store Register Into Memory.

OPERANDS: *(r = A, B, C, D, E, H, or L)*

(HL),r^{2A} *(addr),A*^{4A} *(addr),IX*^{6A} *(BC),A*^{2A}
(IX+d),r^{5E} *(addr),BC*^{6A} *(addr),IY*^{6A} *(DE),A*^{2A}
(IY+d),r^{5E} *(addr),DE*^{6A} *(addr),SP*^{6A}
(addr),HL^{5A}

Condition Set: **NO**

PUSH—Store Register Into Stack.

OPERANDS:

AF^{3D} *BC*^{3D} *DE*^{3D} *HL*^{3D} *IX*^{4D} *IY*^{4D}

Condition Set: **NO**

LOAD REGISTER INSTRUCTIONS

LD—Load Register.

OPERANDS: *(r = A, B, C, D, E, H, or L)*

r,r^{1A} *BC,imm*^{3A} *A,(addr)*^{4A} *SP,HL*^{1C} *A,(BC)*^{2A} *A,I*^{2D}
r,imm^{2A} *DE,imm*^{3A} *BC,(addr)*^{6A} *SP,IX*^{2E} *A,(DE)*^{2A} *A,R*^{2D}
r,(HL)^{2A} *HL,imm*^{3A} *DE,(addr)*^{6A} *SP,IY*^{2E} *I,A*^{2D}
r,(IX+d)^{5E} *IX,imm*^{4B} *HL,(addr)*^{5A} *R,A*^{2D}
r,(IY+d)^{5E} *IY,imm*^{4B} *IX,(addr)*^{6A}
SP,imm^{3A} *IY,(addr)*^{6A}
SP,(addr)^{6A}

Condition Set: **Yes**
(Only for LD A,I and LD A,R)

POP—Load Register from Stack.

OPERANDS:

AF^{3A} *BC*^{3A} *DE*^{3A} *HL*^{3A} *IX*^{4B} *IY*^{4B}

Condition Set: **NO**

MOVE MEMORY-TO-MEMORY INSTRUCTIONS

LD—Move to Memory from Immediate.

OPERANDS:

(HL),imm^{3A} *(IX+d),imm*^{5E} *(IY+d),imm*^{5E} Condition Set: **NO**

LDD —Move (HL) to (DE). Decrement BC, DE, and HL.^{4E}
LDDR—Move (HL) to (DE). Decrement BC, DE, and HL.
Repeat if: BC NOT = 0.$^{5G\ (IF\ BC\ =\ 0\ THEN\ 4E)}$

LDI —Move (HL) to (DE). Decrement BC. Increment DE and HL.^{4E}
LDIR—Move (HL) to (DE). Decrement BC. Increment DE and HL.
Repeat if: BC NOT = 0.$^{5G\ (IF\ BC\ =\ 0\ THEN\ 4E)}$

OPERANDS: ***None Required.*** Condition Set: **YES**

EXCHANGE INSTRUCTIONS

EX—Exchange Register Data with Register or Stack.

OPERANDS:

AF,AF'^{1A} *DE,HL*^{1A} *(SP),HL*^{5D} *(SP),IX*^{6B} *(SP),IY*^{6B}
Condition Set: **NO**

EXX—Exchange Multiple Registers.
BC with BC'. DE with DE'. HL with HL'.^{1A}

OPERANDS: ***None Required.*** Condition Set: **NO**

SHIFT INSTRUCTIONS

RL —Shift Left thru Carry Flag.
Bit 7 goes to Carry Flag. Carry Flag goes to Bit 0.
RR —Shift Right thru Carry Flag.
Bit 0 goes to Carry Flag. Carry Flag goes to Bit 7.

RLC —Shift Left thru Carry Flag.
Bit 7 goes to Carry Flag and Bit 0.
RRC —Shift Right thru Carry Flag.
Bit 0 goes to Carry Flag and Bit 7.

SLA —Shift Left Arithmetic.
Zero Forced into Bit 0. Bit 7 goes to Carry Flag.
SRA —Shift Right Arithmetic.
Bit 7 not changed. Bit 0 goes to Carry Flag.

SRL —Shift Right Logical.
Zero Forced into Bit 7. Bit 0 goes to Carry Flag.

OPERANDS: *(r = A, B, C, D, E, H, or L)*

r^{2B} *(HL)*^{4C} *(IX+d)*^{6C} *(IY+d)*^{6C} Condition Set: **YES**

RLA —Shift Accumulator Left thru Carry Flag.^{1A}
Bit 7 goes to Carry Flag. Carry Flag goes to Bit 0.
RRA —Shift Accumulator Right thru Carry Flag.^{1A}
Bit 0 goes to Carry Flag. Carry Flag goes to Bit 7.

RLCA—Shift Accumulator Left thru Carry Flag.^{1A}
Bit 7 goes to Carry Flag and Bit 0.
RRCA—Shift Accumulator Right thru Carry Flag.^{1A}
Bit 0 goes to Carry Flag and Bit 7.

RLD —Shift Left Half-Byte.^{5C}
Bits 0-3 of (HL) go into Bits 4-7 of (HL).
Bits 4-7 of (HL) go into Bits 0-3 of A.
Bits 0-3 of A go into Bits 0-3 of (HL).
RRD —Shift Right Half-Byte.^{5C}
Bits 4-7 of (HL) go into Bits 0-3 of (HL).
Bits 0-3 of (HL) go into Bits 0-3 of A.
Bits 0-3 of A go into Bits 4-7 of (HL).

OPERANDS: ***Not Required.*** Condition Set: **YES**

COMPARE INSTRUCTIONS

BIT—Test Bit.

OPERANDS: *(b = 0, 1, 2, 3, 4, 5, 6, or 7)*
(r = A, B, C, D, E, H, or L)

b,r^{2B} *b,(HL)*^{3F} *b,(IX+d)*^{5F} *b,(IY+d)*^{5F} Condition Set: **YES**

CP—Compare to Accumulator.

OPERANDS: *(r = A, B, C, D, E, H, or L)*

r^{1A} *imm*^{2A} *(HL)*^{2A} *(IX+d)*^{5E} *(IY+d)*^{5E} Condition Set: **YES**

CPD —Compare (HL) to Accumulator. Decrement HL and BC.^{4E}
CPDR—Compare (HL) to Accumulator. Decrement HL and BC.
Repeat if: BC NOT = 0 AND ACCUMULATOR NOT = (HL).
5G IF REPEAT, ELSE 4E

CPI —Compare (HL) to Accumulator. Increment HL. Decrement BC.^{4E}
CPIR—Compare (HL) to Accumulator. Increment HL. Decrement BC.
Repeat if: BC NOT = 0 AND ACCUMULATOR NOT = (HL).
5G IF REPEAT, ELSE 4E

OPERANDS: ***None Required.*** Condition Set: **YES**

BRANCH INSTRUCTIONS

CALL—Branch and Link for Return.

OPERANDS:

UNCOND^{5B}	COND$^{5B\ IF\ TRUE,\ 3A\ IF\ NOT\ TRUE}$			
addr	*C,addr*	*Z,addr*	*PO,addr*	*P,addr*
	NC,addr	*NZ,addr*	*PE,addr*	*M,addr*

Condition Set: **NO**

DJNZ—Decrement B. Branch if B NOT = 0.

OPERANDS: *addr*$^{3G\ IF\ TRUE,\ 2C\ IF\ NOT\ TRUE}$ Condition Set: **NO**

JP—Branch.

OPERANDS:

UNCOND	COND			
addr^{3A}	*C,addr*^{3A}	*Z,addr*^{3A}	*PO,addr*^{3A}	*P,addr*^{3A}
(HL)^{1A}	*NC,addr*^{3A}	*NZ,addr*^{3A}	*PE,addr*^{3A}	*M,addr*^{3A}
(IX)^{2B}				
(IY)^{2B}				

Condition Set: **NO**

JR—Branch.

OPERANDS:

UNCOND^{3E}	COND$^{3E\ IF\ TRUE,\ 2A\ IF\ NOT\ TRUE}$	
addr	*C,addr*	*Z,addr*
	NC,addr	*NZ,addr*

Condition Set: **NO**

RET—Return from Call.

OPERANDS:

UNCOND^{3A}	COND$^{3D\ IF\ TRUE,\ 1B\ IF\ NOT\ TRUE}$			
None	*C*	*Z*	*PO*	*P*
Required	*NC*	*NZ*	*PE*	*M*

Condition Set: **NO**

RST—Branch to Special Address.

OPERANDS:

00H -or- 0	20H -or- 32	3D
08H -or 8	28H -or- 40	
10H -or- 16	30H -or- 48	
18H -or 24	38H -or- 56	

Condition Set: **NO**

DATA ALTERATION INSTRUCTIONS

AND —'AND' the Accumulator.
OR —'OR' the Accumulator.
XOR —Exclusive 'OR' the Accumulator.

OPERANDS: *(r = A, B, C, D, E, H, or L)*

r[1A] *imm*[2A] *(HL)*[2A] *(IX+d)*[5E] *(IY+d)*[5E] Condition Set: **YES**

RES —Reset Bit. (Set Bit off or Set Bit to 0).
SET —Set Bit. (Set Bit on or Set Bit to 1).

OPERANDS: *(b = 0, 1, 2, 3, 4, 5, 6, or 7)*
(r = A, B, C, D, E, H, or L)

b,r[2B] *b,(HL)*[4C] *b,(IX+d)*[6C] *b,(IY+d)*[6C] Condition Set: **NO**

CCF —Reverse the Carry Flag Bit.[1A] Condition Set: **YES**
CPL —Reverse the Accum Bits.[1A] Condition Set: **NO**
DAA —Convert Accum from Binary to BCD.[1A] Condition Set: **YES**
NEG —Reverse the Accum Numeric Value.[2B] Condition Set: **YES**
NOP —No Operation.[1A] Condition Set: **NO**
SCF —Turn on Carry Flag Bit (Set Bit to 1).[1A] Condition Set: **NO**

OPERANDS: *None Required.*

I/O INSTRUCTIONS

DI —Disable Maskable Interrupts.[1A]
EI —Enable Maskable Interrupts.[1A]
HALT—Halt CPU until Interrupt or Reset is Received.[1A]
IM0 —Set Interrupt Mode 0.[2B]
IM1 —Set Interrupt Mode 1.[2B]
IM2 —Set Interrupt Mode 2.[2B]
RETI —Return from Interrupt.[4B]
(EI Must Be Executed First to Re-Enable Interrupts.)
RETN—Return from Non-Maskable Interrupt.[4B]

OPERANDS: *None Required.* Condition Set: **NO**

IND —Read Device (C) into (HL). Decrement B and HL.[4F]
OUTD—Write (HL) to Device (C). Decrement B and HL.[4F]

INI —Read Device (C) into (HL). Decrement B. Increment HL.[4F]
OUTI —Write (HL) to Device (C). Decrement B. Increment HL.[4F]

INDR —Read Device (C) into (HL). Decrement B and HL.
Repeat If: B NOT = 0.[5H (IF B = 0, 4F)]
OTDR—Write (HL) to Device (C). Decrement B and HL.
Repeat If: B NOT = 0.[5H (IF B = 0, 4F)]

INIR —Read Device (C) into (HL). Decrement B. Increment HL.
Repeat If: B NOT = 0.[5H (IF B = 0, 4F)]
OTIR—Write (HL) to Device (C). Decrement B. Increment HL.
Repeat If: B NOT = 0.[5H (IF B = 0, 4F)]

OPERANDS: *Not Required.* Condition Set: **YES**

IN—Read Device (C) into Specified Register.

OPERANDS: *(r = A, B, C, D, E, H, or L)*

r,(C)[3F] Condition Set: **YES**

OUT—Write to Device (C) from Specified Register.

OPERANDS: *(r = A, B, C, D, E, H, or L)*

(C),r[3F] Condition Set: **NO**

IN—Read Device Specified into Accumulator.

OPERANDS: *A,(addr)*[3B] Condition Set: **NO**

OUT—Write to Specified Device from Accumulator.

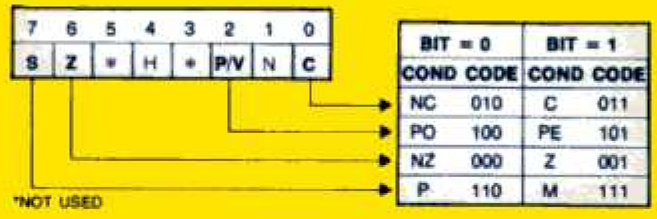
OPERANDS: *(addr),A*[3B] Condition Set: **NO**

FLAGS

S	=	Sign Flag
Z	=	Zero Flag
H	=	Half-Carry Flag
P/V	=	Parity/Oflo Flag
N	=	Add/Subtract Flag
C	=	Carry Flag

CONDITIONS

NC	=	No Carry
C	=	Carry
PO	=	Parity Odd/No Oflo
PE	=	Parity Even/Oflo
NZ	=	Not Zero
Z	=	Zero
P	=	Positive
M	=	Negative



BIT = 0		BIT = 1	
COND	CODE	COND	CODE
NC	010	C	011
PO	100	PE	101
NZ	000	Z	001
P	110	M	111

TYPE OF INSTRUCTION	INSTRUCTIONS WHICH SET FLAGS	CONDITIONS TO TEST							
		NC	C	PO	PE	NZ	Z	P	M
MATH	ADC, ADD, SBC, SUB	●	●	●	●	●	●	●	●
	DEC, INC (Excluding register pairs)			●	●	●	●	●	●
COMPARE	BIT					●	●		
	CP	●	●	●	●	●	●	●	●
	CPD, CPDR, CPI, CPIR			●	●	●	●	●	●
LOAD	LD A,I			●	●	●	●	●	●
	LD A,R			●	●	●	●	●	●
MOVE	LDD, LDI			●	●				
	LDDR, LDIR			●					
DATA	AND, OR, XOR	●		●	●	●	●	●	●
	CCF, RLA, RLCA, RRA, RRCA	●	●						
	DAA, NEG, RL, RLC, RR, RRC	●	●	●	●	●	●	●	●
	SLA, SRA, SRL	●	●	●	●	●	●	●	●
	RLD, RRD			●	●	●	●	●	●
I/O	IN (Except when dev. not spec. by (C))			●	●	●	●	●	●
	IND, INI, OUTD, OUTI					●	●		
	INDR, INIR, OTDR, OTIR						●		

HEX/DEC CONVERSION CHART

8 4 2 1		8 4 2 1		8 4 2 1		8 4 2 1	
HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC
0	0	0	0	0	0	0	0
1	4096	1	256	1	16	1	1
2	8192	2	512	2	32	2	2
3	12288	3	768	3	48	3	3
4	16384	4	1024	4	64	4	4
5	20480	5	1280	5	80	5	5
6	24576	6	1536	6	96	6	6
7	28672	7	1792	7	112	7	7
8	32768	8	2048	8	128	8	8
9	36864	9	2304	9	144	9	9
A	40960	A	2560	A	160	A	10
B	45056	B	2816	B	176	B	11
C	49152	C	3072	C	192	C	12
D	53248	D	3328	D	208	D	13
E	57344	E	3584	E	224	E	14
F	61440	F	3840	F	240	F	15
65535		4095		255		15	

INSTRUCTION	HEX	DECIMAL
NOP	00	000
LD BC,imm	01iiii	001,iii,iii
LD (BC),A	02	002
INC BC	03	003
INC B	04	004
DEC B	05	005
LD B,imm	06ii	006,iii
RLCA	07	007
EX AF,AF'	08	008
ADD HL,BC	09	009
LD A,(BC)	0A	010
DEC BC	0B	011
INC C	0C	012
DEC C	0D	013
LD C,imm	0Eii	014,iii
RRCA	0F	015
DJNZ addr	10aa	016,aaa
LD DE,imm	11iiii	017,iii,iii
LD (DE),A	12	018
INC DE	13	019
INC D	14	020
DEC D	15	021
LD D,imm	16ii	022,iii
RLA	17	023
JR addr	18aa	024,aaa
ADD HL,DE	19	025
LD A,(DE)	1A	026
DEC DE	1B	027
INC E	1C	028
DEC E	1D	029
LD E,imm	1Eii	030,iii
RRA	1F	031
JR NZ,addr	20aa	032,aaa
LD HL,imm	21iiii	033,iii,iii
LD (addr),HL	22aaaa	034,aaa,aaa
INC HL	23	035
INC H	24	036
DEC H	25	037
LD H,imm	26ii	038,iii
DAA	27	039
JR Z,addr	28aa	040,aaa
ADD HL,HL	29	041
LD HL,(addr)	2Aaaaa	042,aaa,aaa
DEC HL	2B	043
INC L	2C	044
DEC L	2D	045
LD L,imm	2Eii	046,iii
CPL	2F	047
JR NC,addr	30aa	048,aaa
LD SP,imm	31iiii	049,iii,iii
LD (addr),A	32aaaa	050,aaa,aaa
INC SP	33	051
INC (HL)	34	052
DEC (HL)	35	053
LD (HL),imm	36ii	054,iii
SCF	37	055
JR C,addr	38aa	056,aaa
ADD HL,SP	39	057
LD A,(addr)	3Aaaaa	058,aaa,aaa
DEC SP	3B	059
INC A	3C	060
DEC A	3D	061
LD A,imm	3Eii	062,iii
CCF	3F	063
LD B,B	40	064
LD B,C	41	065
LD B,D	42	066
LD B,E	43	067
LD B,H	44	068
LD B,L	45	069
LD B,(HL)	46	070
LD B,A	47	071
LD C,B	48	072
LD C,C	49	073
LD C,D	4A	074
LD C,E	4B	075
LD C,H	4C	076
LD C,L	4D	077
LD C,(HL)	4E	078
LD C,A	4F	079
LD D,B	50	080
LD D,C	51	081
LD D,D	52	082
LD D,E	53	083
LD D,H	54	084
LD D,L	55	085
LD D,(HL)	56	086
LD D,A	57	087

INSTRUCTION	HEX	DECIMAL
LD E,B	58	088
LD E,C	59	089
LD E,D	5A	090
LD E,E	5B	091
LD E,H	5C	092
LD E,L	5D	093
LD E,(HL)	5E	094
LD E,A	5F	095
LD H,B	60	096
LD H,C	61	097
LD H,D	62	098
LD H,E	63	099
LD H,H	64	100
LD H,L	65	101
LD H,(HL)	66	102
LD H,A	67	103
LD L,B	68	104
LD L,C	69	105
LD L,D	6A	106
LD L,E	6B	107
LD L,H	6C	108
LD L,L	6D	109
LD L,(HL)	6E	110
LD L,A	6F	111
LD (HL),B	70	112
LD (HL),C	71	113
LD (HL),D	72	114
LD (HL),E	73	115
LD (HL),H	74	116
LD (HL),L	75	117
HALT	76	118
LD (HL),A	77	119
LD A,B	78	120
LD A,C	79	121
LD A,D	7A	122
LD A,E	7B	123
LD A,H	7C	124
LD A,L	7D	125
LD A,(HL)	7E	126
LD A,A	7F	127
ADD A,B	80	128
ADD A,C	81	129
ADD A,D	82	130
ADD A,E	83	131
ADD A,H	84	132
ADD A,L	85	133
ADD A,(HL)	86	134
ADD A,A	87	135
ADC A,B	88	136
ADC A,C	89	137
ADC A,D	8A	138
ADC A,E	8B	139
ADC A,H	8C	140
ADC A,L	8D	141
ADC A,(HL)	8E	142
ADC A,A	8F	143
SUB B	90	144
SUB C	91	145
SUB D	92	146
SUB E	93	147
SUB H	94	148
SUB L	95	149
SUB (HL)	96	150
SUB A	97	151
SBC A,B	98	152
SBC A,C	99	153
SBC A,D	9A	154
SBC A,E	9B	155
SBC A,H	9C	156
SBC A,L	9D	157
SBC A,(HL)	9E	158
SBC A,A	9F	159
AND B	A0	160
AND C	A1	161
AND D	A2	162
AND E	A3	163
AND H	A4	164
AND L	A5	165
AND (HL)	A6	166
AND A	A7	167
XOR B	A8	168
XOR C	A9	169
XOR D	AA	170
XOR E	AB	171
XOR H	AC	172
XOR L	AD	173
XOR (HL)	AE	174
XOR A	AF	175

INSTRUCTION	HEX	DECIMAL
OR B	B0	176
OR C	B1	177
OR D	B2	178
OR E	B3	179
OR H	B4	180
OR L	B5	181
OR (HL)	B6	182
OR A	B7	183
CP B	B8	184
CP C	B9	185
CP D	BA	186
CP E	BB	187
CP H	BC	188
CP L	BD	189
CP (HL)	BE	190
CP A	BF	191
RET NZ	C0	192
POP BC	C1	193
JP NZ,addr	C2aaaa	194,aaa,aaa
JP addr	C3aaaa	195,aaa,aaa
CALL NZ,addr	C4aaaa	196,aaa,aaa
PUSH BC	C5	197
ADD A,imm	C6ii	198,iii
RST 00H	C7	199
RET Z	C8	200
RET	C9	201
JP Z,addr	CAaaaa	202,aaa,aaa
RLC B	CB00	203,000
RLC C	CB01	203,001
RLC D	CB02	203,002
RLC E	CB03	203,003
RLC H	CB04	203,004
RLC L	CB05	203,005
RLC (HL)	CB06	203,006
RLC A	CB07	203,007
RRC B	CB08	203,008
RRC C	CB09	203,009
RRC D	CB0A	203,010
RRC E	CB0B	203,011
RRC H	CB0C	203,012
RRC L	CB0D	203,013
RRC (HL)	CB0E	203,014
RRC A	CB0F	203,015
RL B	CB10	203,016
RL C	CB11	203,017
RL D	CB12	203,018
RL E	CB13	203,019
RL H	CB14	203,020
RL L	CB15	203,021
RL (HL)	CB16	203,022
RL A	CB17	203,023
RR B	CB18	203,024
RR C	CB19	203,025
RR D	CB1A	203,026
RR E	CB1B	203,027
RR H	CB1C	203,028
RR L	CB1D	203,029
RR (HL)	CB1E	203,030
RR A	CB1F	203,031
SLA B	CB20	203,032
SLA C	CB21	203,033
SLA D	CB22	203,034
SLA E	CB23	203,035
SLA H	CB24	203,036
SLA L	CB25	203,037
SLA (HL)	CB26	203,038
SLA A	CB27	203,039
SRA B	CB28	203,040
SRA C	CB29	203,041
SRA D	CB2A	203,042
SRA E	CB2B	203,043
SRA H	CB2C	203,044
SRA L	CB2D	203,045
SRA (HL)	CB2E	203,046
SRA A	CB2F	203,047
SRL B	CB38	203,056
SRL C	CB39	203,057
SRL D	CB3A	203,058
SRL E	CB3B	203,059
SRL H	CB3C	203,060
SRL L	CB3D	203,061
SRL (HL)	CB3E	203,062
SRL A	CB3F	203,063
BIT 0,B	CB40	203,064
BIT 0,C	CB41	203,065
BIT 0,D	CB42	203,066
BIT 0,E	CB43	203,067
BIT 0,H	CB44	203,068

INSTRUCTION	HEX	DECIMAL
BIT 0,L	CB45	203,069
BIT 0,(HL)	CB46	203,070
BIT 0,A	CB47	203,071
BIT 1,B	CB48	203,072
BIT 1,C	CB49	203,073
BIT 1,D	CB4A	203,074
BIT 1,E	CB4B	203,075
BIT 1,H	CB4C	203,076
BIT 1,L	CB4D	203,077
BIT 1,(HL)	CB4E	203,078
BIT 1,A	CB4F	203,079
BIT 2,B	CB50	203,080
BIT 2,C	CB51	203,081
BIT 2,D	CB52	203,082
BIT 2,E	CB53	203,083
BIT 2,H	CB54	203,084
BIT 2,L	CB55	203,085
BIT 2,(HL)	CB56	203,086
BIT 2,A	CB57	203,087
BIT 3,B	CB58	203,088
BIT 3,C	CB59	203,089
BIT 3,D	CB5A	203,090
BIT 3,E	CB5B	203,091
BIT 3,H	CB5C	203,092
BIT 3,L	CB5D	203,093
BIT 3,(HL)	CB5E	203,094
BIT 3,A	CB5F	203,095
BIT 4,B	CB60	203,096
BIT 4,C	CB61	203,097
BIT 4,D	CB62	203,098
BIT 4,E	CB63	203,099
BIT 4,H	CB64	203,100
BIT 4,L	CB65	203,101
BIT 4,(HL)	CB66	203,102
BIT 4,A	CB67	203,103
BIT 5,B	CB68	203,104
BIT 5,C	CB69	203,105
BIT 5,D	CB6A	203,106
BIT 5,E	CB6B	203,107
BIT 5,H	CB6C	203,108
BIT 5,L	CB6D	203,109
BIT 5,(HL)	CB6E	203,110
BIT 5,A	CB6F	203,111
BIT 6,B	CB70	203,112
BIT 6,C	CB71	203,113
BIT 6,D	CB72	203,114
BIT 6,E	CB73	203,115
BIT 6,H	CB74	203,116
BIT 6,L	CB75	203,117
BIT 6,(HL)	CB76	203,118
BIT 6,A	CB77	203,119
BIT 7,B	CB78	203,120
BIT 7,C	CB79	203,121
BIT 7,D	CB7A	203,122
BIT 7,E	CB7B	203,123
BIT 7,H	CB7C	203,124
BIT 7,L	CB7D	203,125
BIT 7,(HL)	CB7E	203,126
BIT 7,A	CB7F	203,127
RES 0,B	CB80	203,128
RES 0,C	CB81	203,129
RES 0,D	CB82	203,130
RES 0,E	CB83	203,131
RES 0,H	CB84	203,132
RES 0,L	CB85	203,133
RES 0,(HL)	CB86	203,134
RES 0,A	CB87	203,135
RES 1,B	CB88	203,136
RES 1,C	CB89	203,137
RES 1,D	CB8A	203,138
RES 1,E	CB8B	203,139
RES 1,H	CB8C	203,140
RES 1,L	CB8D	203,141
RES 1,(HL)	CB8E	203,142
RES 1,A	CB8F	203,143
RES 2,B	CB90	203,144
RES 2,C	CB91	203,145
RES 2,D	CB92	203,146
RES 2,E	CB93	203,147
RES 2,H	CB94	203,148
RES 2,L	CB95	203,149
RES 2,(HL)	CB96	203,150
RES 2,A	CB97	203,151
RES 3,B	CB98	203,152
RES 3,C	CB99	203,153
RES 3,D	CB9A	203,154
RES 3,E	CB9B	203,155
RES 3,H	CB9C	203,156

OP-CODE SEQUENCE

INSTRUCTION	HEX	DECIMAL
RES 3,L	CB9D	203,157
RES 3,(HL)	CB9E	203,158
RES 3,A	CB9F	203,159
RES 4,B	CBA0	203,160
RES 4,C	CBA1	203,161
RES 4,D	CBA2	203,162
RES 4,E	CBA3	203,163
RES 4,H	CBA4	203,164
RES 4,L	CBA5	203,165
RES 4,(HL)	CBA6	203,166
RES 4,A	CBA7	203,167
RES 5,B	CBA8	203,168
RES 5,C	CBA9	203,169
RES 5,D	CBAA	203,170
RES 5,E	CBAB	203,171
RES 5,H	CBAC	203,172
RES 5,L	CBAD	203,173
RES 5,(HL)	CBAE	203,174
RES 5,A	CBAF	203,175
RES 6,B	CBB0	203,176
RES 6,C	CBB1	203,177
RES 6,D	CBB2	203,178
RES 6,E	CBB3	203,179
RES 6,H	CBB4	203,180
RES 6,L	CBB5	203,181
RES 6,(HL)	CBB6	203,182
RES 6,A	CBB7	203,183
RES 7,B	CBB8	203,184
RES 7,C	CBB9	203,185
RES 7,D	CBBA	203,186
RES 7,E	CBBB	203,187
RES 7,H	CBBC	203,188
RES 7,L	CBBD	203,189
RES 7,(HL)	CBBE	203,190
RES 7,A	CBBF	203,191
SET 0,B	CBC0	203,192
SET 0,C	CBC1	203,193
SET 0,D	CBC2	203,194
SET 0,E	CBC3	203,195
SET 0,H	CBC4	203,196
SET 0,L	CBC5	203,197
SET 0,(HL)	CBC6	203,198
SET 0,A	CBC7	203,199
SET 1,B	CBC8	203,200
SET 1,C	CBC9	203,201
SET 1,D	CBCA	203,202
SET 1,E	CBCB	203,203
SET 1,H	CBCC	203,204
SET 1,L	CBCD	203,205
SET 1,(HL)	CBCE	203,206
SET 1,A	CBCF	203,207
SET 2,B	CBD0	203,208
SET 2,C	CBD1	203,209
SET 2,D	CBD2	203,210
SET 2,E	CBD3	203,211
SET 2,H	CBD4	203,212
SET 2,L	CBD5	203,213
SET 2,(HL)	CBD6	203,214
SET 2,A	CBD7	203,215
SET 3,B	CBD8	203,216
SET 3,C	CBD9	203,217
SET 3,D	CBDA	203,218
SET 3,E	CBDB	203,219
SET 3,H	CBDC	203,220
SET 3,L	CBDD	203,221
SET 3,(HL)	CBDE	203,222
SET 3,A	CBDF	203,223
SET 4,B	CBE0	203,224
SET 4,C	CBE1	203,225
SET 4,D	CBE2	203,226
SET 4,E	CBE3	203,227
SET 4,H	CBE4	203,228
SET 4,L	CBE5	203,229
SET 4,(HL)	CBE6	203,230
SET 4,A	CBE7	203,231
SET 5,B	CBE8	203,232
SET 5,C	CBE9	203,233
SET 5,D	CBEA	203,234
SET 5,E	CBEB	203,235
SET 5,H	CBEC	203,236
SET 5,L	CBED	203,237
SET 5,(HL)	CBEE	203,238
SET 5,A	CBEF	203,239
SET 6,B	CBF0	203,240
SET 6,C	CBF1	203,241
SET 6,D	CBF2	203,242
SET 6,E	CBF3	203,243
SET 6,H	CBF4	203,244

INSTRUCTION	HEX	DECIMAL
SET 6,L	CBF5	203,245
SET 6,(HL)	CBF6	203,246
SET 6,A	CBF7	203,247
SET 7,B	CBF8	203,248
SET 7,C	CBF9	203,249
SET 7,D	CBFA	203,250
SET 7,E	CBFB	203,251
SET 7,H	CBFC	203,252
SET 7,L	CBFD	203,253
SET 7,(HL)	CBFE	203,254
SET 7,A	CBFF	203,255
CALL Z,addr	CCaaaa	204,aaa,aaa
CALL addr	CDaaaa	205,aaa,aaa
ADC A,imm	CEii	206,iii
RST 08H	CF	207
RET NC	D0	208
POP DE	D1	209
JP NC,addr	D2aaaa	210,aaa,aaa
OUT (addr),A	D3aa	211,aaa
CALL NC,addr	D4aaaa	212,aaa,aaa
PUSH DE	D5	213
SUB imm	D6ii	214,iii
RST 10H	D7	215
RET C	D8	216
EXX	D9	217
JP C,addr	DAaaaa	218,aaa,aaa
IN A,addr	DBaa	219,aaa
CALL C,addr	DCaaaa	220,aaa,aaa
ADD IX,BC	DD09	221,009
ADD IX,DE	DD19	221,025
LD IX,imm	DD21iiii	221,033,iii,iii
LD (addr),IX	DD22aaaa	221,034,aaa,aaa
INC IX	DD23	221,035
ADD IX,IX	DD29	221,041
LD IX,(addr)	DD2Aaaaa	221,042,aaa,aaa
DEC IX	DD2B	221,043
INC (IX+d)	DD34dd	221,052,ddd
DEC (IX+d)	DD35dd	221,053,ddd
LD (IX+d),imm	DD36ddii	221,054,ddd,iii
ADD IX,SP	DD39	221,057
LD B,(IX+d)	DD46dd	221,070,ddd
LD C,(IX+d)	DD4Edd	221,078,ddd
LD D,(IX+d)	DD56dd	221,086,ddd
LD E,(IX+d)	DD5Edd	221,094,ddd
LD H,(IX+d)	DD66dd	221,102,ddd
LD L,(IX+d)	DD6Edd	221,110,ddd
LD (IX+d),B	DD70dd	221,112,ddd
LD (IX+d),C	DD71dd	221,113,ddd
LD (IX+d),D	DD72dd	221,114,ddd
LD (IX+d),E	DD73dd	221,115,ddd
LD (IX+d),H	DD74dd	221,116,ddd
LD (IX+d),L	DD75dd	221,117,ddd
LD (IX+d),A	DD77dd	221,119,ddd
LD A,(IX+d)	DD7Edd	221,126,ddd
ADD A,(IX+d)	DD86dd	221,134,ddd
ADC A,(IX+d)	DD8Edd	221,142,ddd
SUB (IX+d)	DD96dd	221,150,ddd
SBC A,(IX+d)	DD9Edd	221,158,ddd
AND (IX+d)	DDA6dd	221,166,ddd
XOR (IX+d)	DDAEdd	221,174,ddd
OR (IX+d)	DDB6dd	221,182,ddd
CP (IX+d)	DDBEdd	221,190,ddd
RLC (IX+d)	DDCBdd06	221,203,ddd,006
RRC (IX+d)	DDCBdd0E	221,203,ddd,014
RL (IX+d)	DDCBdd16	221,203,ddd,022
RR (IX+d)	DDCBdd1E	221,203,ddd,030
SLA (IX+d)	DDCBdd26	221,203,ddd,038
SRA (IX+d)	DDCBdd2E	221,203,ddd,046
SRL (IX+d)	DDCBdd3E	221,203,ddd,062
BIT 0,(IX+d)	DDCBdd46	221,203,ddd,070
BIT 1,(IX+d)	DDCBdd4E	221,203,ddd,078
BIT 2,(IX+d)	DDCBdd56	221,203,ddd,086
BIT 3,(IX+d)	DDCBdd5E	221,203,ddd,094
BIT 4,(IX+d)	DDCBdd66	221,203,ddd,102
BIT 5,(IX+d)	DDCBdd6E	221,203,ddd,110
BIT 6,(IX+d)	DDCBdd76	221,203,ddd,118
BIT 7,(IX+d)	DDCBdd7E	221,203,ddd,126
RES 0,(IX+d)	DDCBdd86	221,203,ddd,134
RES 1,(IX+d)	DDCBdd8E	221,203,ddd,142
RES 2,(IX+d)	DDCBdd96	221,203,ddd,150
RES 3,(IX+d)	DDCBdd9E	221,203,ddd,158
RES 4,(IX+d)	DDCBddA6	221,203,ddd,166
RES 5,(IX+d)	DDCBddAE	221,203,ddd,174
RES 6,(IX+d)	DDCBddB6	221,203,ddd,182
RES 7,(IX+d)	DDCBddBE	221,203,ddd,190
SET 0,(IX+d)	DDCBddC6	221,203,ddd,198
SET 1,(IX+d)	DDCBddCE	221,203,ddd,206
SET 2,(IX+d)	DDCBddD6	221,203,ddd,214

OP-CODE SEQUENCE

INSTRUCTION	HEX	DECIMAL
SET 3,(IX+d)	DDCBddDE	221,203,ddd,222
SET 4,(IX+d)	DDCBddE6	221,203,ddd,230
SET 5,(IX+d)	DDCBddEE	221,203,ddd,238
SET 6,(IX+d)	DDCBddF6	221,203,ddd,246
SET 7,(IX+d)	DDCBddFE	221,203,ddd,254
POP IX	DDE1	221,225
EX (SP),IX	DDE3	221,227
PUSH IX	DDE5	221,229
JP (IX)	DDE9	221,233
LD SP,IX	DDF9	221,249
SBC A,imm	DEii	222,iii
RST 18H	DF	223
RET PO	E0	224
POP HL	E1	225
JP PO,addr	E2aaaa	226,aaa,aaa
EX (SP),HL	E3	227
CALL PO,addr	E4aaaa	228,aaa,aaa
PUSH HL	E5	229
AND imm	E6ii	230,iii
RST 20H	E7	231
RET PE	E8	232
JP (HL)	E9	233
JP PE,addr	EAaaaa	234,aaa,aaa
EX DE,HL	EB	235
CALL PE,addr	ECaaaa	236,aaa,aaa
IN B,(C)	ED40	237,064
OUT (C),B	ED41	237,065
SBC HL,BC	ED42	237,066
LD (addr),BC	ED43aaaa	237,067,aaa,aaa
NEG	ED44	237,068
RETN	ED45	237,069
IM 0	ED46	237,070
LD I,A	ED47	237,071
IN C,(C)	ED48	237,072
OUT (C),C	ED49	237,073
ADC HL,BC	ED4A	237,074
LD BC,(addr)	ED4Baaaa	237,075,aaa,aaa
RETI	ED4D	237,077
LD R,A	ED4F	237,079
IN D,(C)	ED50	237,080
OUT (C),D	ED51	237,081
SBC HL,DE	ED52	237,082
LD (addr),DE	ED53aaaa	237,083,aaa,aaa
IM 1	ED56	237,086
LD A,I	ED57	237,087
IN E,(C)	ED58	237,088
OUT (C),E	ED59	237,089
ADC HL,DE	ED5A	237,090
LD DE,(addr)	ED5Baaaa	237,091,aaa,aaa
IM 2	ED5E	237,094
LD A,R	ED5F	237,095
IN H,(C)	ED60	237,096
OUT (C),H	ED61	237,097
SBC HL,HL	ED62	237,098
RRD	ED67	237,103
IN L,(C)	ED68	237,104
OUT (C),L	ED69	237,105
ADC HL,HL	ED6A	237,106
RLD	ED6F	237,111
SBC HL,SP	ED72	237,114
LD (addr),SP	ED73aaaa	237,115,aaa,aaa
IN A,(C)	ED78	237,120
OUT (C),A	ED79	237,121
ADC HL,SP	ED7A	237,122
LD SP,(addr)	ED7Baaaa	237,123,aaa,aaa
LDI	EDA0	237,160
CPI	EDA1	237,161
INI	EDA2	237,162
OUTI	EDA3	237,163
LDD	EDA8	237,168
CPD	EDA9	237,169
IND	EDAA	237,170
OUTD	EDAB	237,171
LDIR	EDB0	237,176
CPIR	EDB1	237,177
INIR	EDB2	237,178
OTIR	EDB3	237,179
LDDR	EDB8	237,184
CPDR	EDB9	237,185
INDR	EDBA	237,186
OTDR	EDBB	237,187
XOR imm	EEii	238,iii
RST 28H	EF	239
RET P	F0	240
POP AF	F1	241
JP P,addr	F2aaaa	242,aaa,aaa
DI	F3	243
CALL P,addr	F4aaaa	244,aaa,aaa

INSTRUCTION	HEX	DECIMAL
PUSH AF	F5	245
OR imm	F6ii	246,iii
RST 30H	F7	247
RET M	F8	248
LD SP,HL	F9	249
JP M,addr	FAaaaa	250,aaa,aaa
EI	FB	251
CALL M,addr	FCaaaa	252,aaa,aaa
ADD IY,BC	FD09	253,009
ADD IY,DE	FD19	253,025
LD IY,imm	FD21iiii	253,033,iii,iii
LD (addr),IY	FD22aaaa	253,034,aaa,aaa
INC IY	FD23	253,035
ADD IY,IY	FD29	253,041
LD IY,(addr)	FD2Aaaaa	253,042,aaa,aaa
DEC IY	FD2B	253,043
INC (IY+d)	FD34dd	253,052,ddd
DEC (IY+d)	FD35dd	253,053,ddd
LD (IY+d),imm	FD36ddii	253,054,ddd,iii
ADD IY,SP	FD39	253,057
LD B,(IY+d)	FD46dd	253,070,ddd
LD C,(IY+d)	FD4Edd	253,078,ddd
LD D,(IY+d)	FD56dd	253,086,ddd
LD E,(IY+d)	FD5Edd	253,094,ddd
LD H,(IY+d)	FD66dd	253,102,ddd
LD L,(IY+d)	FD6Edd	253,110,ddd
LD (IY+d),B	FD70dd	253,112,ddd
LD (IY+d),C	FD71dd	253,113,ddd
LD (IY+d),D	FD72dd	253,114,ddd
LD (IY+d),E	FD73dd	253,115,ddd
LD (IY+d),H	FD74dd	253,116,ddd
LD (IY+d),L	FD75dd	253,117,ddd
LD (IY+d),A	FD77dd	253,119,ddd
LD A,(IY+d)	FD7Edd	253,126,ddd
ADD A,(IY+d)	FD86dd	253,134,ddd
ADC A,(IY+d)	FD8Edd	253,142,ddd
SUB (IY+d)	FD96dd	253,150,ddd
SBC A,(IY+d)	FD9Edd	253,158,ddd
AND (IY+d)	FDA6dd	253,166,ddd
XOR (IY+d)	FDAEdd	253,174,ddd
OR (IY+d)	FDB6dd	253,182,ddd
CP (IY+d)	FDBEdd	253,190,ddd
RLC (IY+d)	FDCBdd06	253,203,ddd,006
RRC (IY+d)	FDCBdd0E	253,203,ddd,014
RL (IY+d)	FDCBdd16	253,203,ddd,022
RR (IY+d)	FDCBdd1E	253,203,ddd,030
SLA (IY+d)	FDCBdd26	253,203,ddd,038
SRA (IY+d)	FDCBdd2E	253,203,ddd,046
SRL (IY+d)	FDCBdd3E	253,203,ddd,062
BIT 0,(IY+d)	FDCBdd46	253,203,ddd,070
BIT 1,(IY+d)	FDCBdd4E	253,203,ddd,078
BIT 2,(IY+d)	FDCBdd56	253,203,ddd,086
BIT 3,(IY+d)	FDCBdd5E	253,203,ddd,094
BIT 4,(IY+d)	FDCBdd66	253,203,ddd,102
BIT 5,(IY+d)	FDCBdd6E	253,203,ddd,110
BIT 6,(IY+d)	FDCBdd76	253,203,ddd,118
BIT 7,(IY+d)	FDCBdd7E	253,203,ddd,126
RES 0,(IY+d)	FDCBdd86	253,203,ddd,134
RES 1,(IY+d)	FDCBdd8E	253,203,ddd,142
RES 2,(IY+d)	FDCBdd96	253,203,ddd,150
RES 3,(IY+d)	FDCBdd9E	253,203,ddd,158
RES 4,(IY+d)	FDCBddA6	253,203,ddd,166
RES 5,(IY+d)	FDCBddAE	253,203,ddd,174
RES 6,(IY+d)	FDCBddB6	253,203,ddd,182
RES 7,(IY+d)	FDCBddBE	253,203,ddd,190
SET 0,(IY+d)	FDCBddC6	253,203,ddd,198
SET 1,(IY+d)	FDCBddCE	253,203,ddd,206
SET 2,(IY+d)	FDCBddD6	253,203,ddd,214
SET 3,(IY+d)	FDCBddDE	253,203,ddd,222
SET 4,(IY+d)	FDCBddE6	253,203,ddd,230
SET 5,(IY+d)	FDCBddEE	253,203,ddd,238
SET 6,(IY+d)	FDCBddF6	253,203,ddd,246
SET 7,(IY+d)	FDCBddFE	253,203,ddd,254
POP IY	FDE1	253,225
EX (SP),IY	FDE3	253,227
PUSH IY	FDE5	253,229
JP (IY)	FDE9	253,233
LD SP,IY	FDF9	253,249
CP imm	FEii	254,iii
RST 38H	FF	255