Z80-Befehlsreferenz für das Lernapplet

Diese Referenz enthält die Befehle, die im Lernapplet verwendet werden können. Der vollständige Z80-Prozessor kennt noch viele weitere Befehle.

Schreibweisen und Platzhalter

Ein Befehl besteht aus einem Opcode und manchmal aus einem oder zwei weiteren Bytes.

```
n
Ein Wert mit 8 Bit. Er liegt zwischen 0 und 255.

nn
Ein Wert oder eine Adresse mit 16 Bit. Er liegt zwischen 0 und
65535.

e
Ein relativer Sprungabstand von -128 bis 127. Im Assemblerfeld des
Applets wird nicht der Abstand, sondern die Zieladresse angegeben.

(nn)
Der Inhalt der Speicherzelle an der Adresse nn. Die Klammern sind
wichtig: LD A,100 lädt die Zahl 100, LD A,(100) liest dagegen aus
der Speicherzelle 100.
```

Zahlen ohne besondere Kennzeichnung werden als Dezimalzahlen gelesen:

```
42
```

Hexadezimalzahlen können auf verschiedene Arten geschrieben werden:

```
0x2A
$2A
2Ah
```

Enthält eine Zahl die Buchstaben A bis F, erkennt das Applet sie auch ohne Zusatz als Hexadezimalzahl.

Byte-Reihenfolge bei 16-Bit-Werten

Der Z80 speichert einen 16-Bit-Wert in der Reihenfolge „niederwertiges Byte zuerst“. Diese Reihenfolge heißt Little Endian.

Die Dezimalzahl 1056 entspricht der Hexadezimalzahl 0420. Im Maschinencode stehen deshalb zuerst 20 und danach 04:

```
LD HL,1056
Hex: 21 20 04
```

Laden von 8-Bit-Werten

LD A,n

```
Bedeutung: Lade den Wert n in das Register A.
Hex-Code:  3E n
Binär:     00111110 nnnnnnnn
Länge:     2 Byte
Beispiel:  LD A,6
Ergebnis:  A enthält 6.
Hex:       3E 06
Binär:     00111110 00000110
```

LD B,n

```
Bedeutung: Lade den Wert n in das Register B.
Hex-Code:  06 n
Binär:     00000110 nnnnnnnn
Länge:     2 Byte
Beispiel:  LD B,10
Hex:       06 0A
```

LD C,n

```
Bedeutung: Lade den Wert n in das Register C.
Hex-Code:  0E n
Binär:     00001110 nnnnnnnn
Länge:     2 Byte
Beispiel:  LD C,255
Hex:       0E FF
```

LD A,r und LD r,A

```
Bedeutung: Kopiere den Inhalt eines 8-Bit-Registers. Der Quellwert
bleibt dabei erhalten. Als r sind B, C, D, E, H und L möglich.
Länge:     jeweils 1 Byte

Beispiel:
LD B,A        kopiert A nach B
LD A,C        kopiert C nach A

Befehl    Hex        Befehl    Hex
LD A,B     78        LD B,A     47
LD A,C     79        LD C,A     4F
LD A,D     7A        LD D,A     57
LD A,E     7B        LD E,A     5F
LD A,H     7C        LD H,A     67
LD A,L     7D        LD L,A     6F
```

Laden von 16-Bit-Werten

Bei den folgenden Befehlen werden die beiden Bytes von nn in Little-Endian-Reihenfolge gespeichert.

LD HL,nn

```
Bedeutung: Lade den 16-Bit-Wert nn in das Registerpaar HL.
Hex-Code:  21 nn_low nn_high
Binär:     00100001 nnnnnnnn nnnnnnnn
Länge:     3 Byte
Beispiel:  LD HL,1056
Hex:       21 20 04
```

LD DE,nn

```
Bedeutung: Lade den 16-Bit-Wert nn in das Registerpaar DE.
Hex-Code:  11 nn_low nn_high
Binär:     00010001 nnnnnnnn nnnnnnnn
Länge:     3 Byte
Beispiel:  LD DE,1024
Hex:       11 00 04
```

LD BC,nn

```
Bedeutung: Lade den 16-Bit-Wert nn in das Registerpaar BC.
Hex-Code:  01 nn_low nn_high
Binär:     00000001 nnnnnnnn nnnnnnnn
Länge:     3 Byte
Beispiel:  LD BC,500
Hex:       01 F4 01
```

LD SP,nn

```
Bedeutung: Lade nn in den Stackpointer SP.
Hex-Code:  31 nn_low nn_high
Binär:     00110001 nnnnnnnn nnnnnnnn
Länge:     3 Byte
Beispiel:  LD SP,2048
Hex:       31 00 08
```

Zugriff auf den Speicher

LD (nn),A

```
Bedeutung: Speichere den Inhalt von A in der Speicherzelle nn.
Hex-Code:  32 nn_low nn_high
Binär:     00110010 nnnnnnnn nnnnnnnn
```

```
Länge:     3 Byte
Beispiel:  LD (1056),A
Hex:       32 20 04
```

LD A,(nn)

```
Bedeutung: Lade den Inhalt der Speicherzelle nn in A.
Hex-Code:  3A nn_low nn_high
Binär:     00111010 nnnnnnnn nnnnnnnn
Länge:     3 Byte
Beispiel:  LD A,(1024)
Hex:       3A 00 04
```

Rechnen und Vergleichen

INC A

```
Bedeutung: Erhöhe A um 1.
Hex-Code:  3C
Binär:     00111100
Länge:     1 Byte
Beispiel:  Aus A=7 wird A=8.
```

DEC A

```
Bedeutung: Verringere A um 1.
Hex-Code:  3D
Binär:     00111101
Länge:     1 Byte
Beispiel:  Aus A=7 wird A=6.
```

ADD A,n

```
Bedeutung: Addiere n zum Inhalt von A. Das Ergebnis steht in A.
Hex-Code:  C6 n
Binär:     11000110 nnnnnnnn
Länge:     2 Byte
Beispiel:  ADD A,5
Hex:       C6 05
```

ADD A,r

```
Bedeutung: Addiere den Inhalt eines 8-Bit-Registers zu A. Das
           Ergebnis wird wieder in A gespeichert.
Länge:     1 Byte
```

Die offizielle Z80-Schreibweise nennt den Akkumulator A ausdrücklich. Der Befehl heißt also ADD A,B und nicht nur ADD B.

```
Befehl        Hex-Code   Binär
ADD A,B         80       10000000
ADD A,C         81       10000001
ADD A,D         82       10000010
ADD A,E         83       10000011
ADD A,H         84       10000100
ADD A,L         85       10000101
ADD A,(HL)      86       10000110
ADD A,A         87       10000111
```

Bei ADD A,(HL) wird nicht HL selbst addiert. HL enthält eine Adresse; addiert wird das Byte aus der Speicherzelle an dieser Adresse.

Beispiel:

```
LD A,5
LD B,7
ADD A,B
```

Danach enthält A den Wert 12. B bleibt unverändert 7.

XOR A

```
Bedeutung: Verknüpfe A mit sich selbst durch XOR. Dadurch wird A
           auf 0 gesetzt.
Hex-Code:  AF
Binär:     10101111
Länge:     1 Byte
```

CP n

```
Bedeutung: Vergleiche A mit n. A bleibt unverändert. Das
           Zero-Flag Z wird gesetzt, wenn A und n gleich sind.
Hex-Code:  FE n
Binär:     11111110 nnnnnnnn
Länge:     2 Byte
Beispiel:  CP 10
Hex:       FE 0A
```

Sprünge

JP nn

```
Bedeutung: Springe immer zur absoluten Adresse nn.
Hex-Code:  C3 nn_low nn_high
Binär:     11000011 nnnnnnnn nnnnnnnn
Länge:     3 Byte
Beispiel:  JP 300
Hex:       C3 2C 01
```

JP Z,nn und JP NZ,nn

```
Bedeutung: Springe zur absoluten Adresse nn, wenn das Zero-Flag
           gesetzt (Z) beziehungsweise nicht gesetzt (NZ) ist.
Länge:     jeweils 3 Byte
Typisch:   CP n wird vorher zum Vergleichen verwendet.

Befehl     Hex-Code
JP Z,nn    CA nn_low nn_high
JP NZ,nn   C2 nn_low nn_high
```

JR nn

```
Bedeutung: Springe immer zur angegebenen Zieladresse. Im
           Maschinencode wird nur der relative Abstand e gespeichert.
Hex-Code:  18 e
Binär:     00011000 eeeeeeee
Länge:     2 Byte

Beispiel: Der Befehl beginnt an Adresse 100 und soll zur Adresse
          110 springen. Nach dem Lesen des Befehls steht der PC auf
          102. Der Abstand beträgt deshalb 110-102=8.
Befehl:   JR 110
Hex:      18 08
```

JR Z,nn

```
Bedeutung: Springe zur Zieladresse, wenn das Zero-Flag Z gesetzt ist.
Hex-Code:  28 e
Binär:     00101000 eeeeeeee
Länge:     2 Byte
Typisch:   CP n wird vorher zum Vergleichen verwendet.
```

JR NZ,nn

```
Bedeutung: Springe zur Zieladresse, wenn das Zero-Flag Z nicht
           gesetzt ist.
Hex-Code:  20 e
Binär:     00100000 eeeeeeee
Länge:     2 Byte
```

Relative Sprünge können höchstens 128 Byte zurück und 127 Byte vorwärts reichen. Das Applet berechnet den Abstand aus der eingegebenen Zieladresse automatisch.

Unterprogramme

CALL nn

```
Bedeutung: Rufe das Unterprogramm an Adresse nn auf. Die Adresse des
           nächsten Befehls wird auf dem Stack gespeichert.
Hex-Code:  CD nn_low nn_high
Binär:     11001101 nnnnnnnn nnnnnnnn
Länge:     3 Byte
Beispiel:  CALL 300
Hex:       CD 2C 01
```

RET

```
Bedeutung: Kehre aus einem Unterprogramm zurück. Die Rücksprungadresse
           wird vom Stack gelesen.
Hex-Code:  C9
Binär:     11001001
Länge:     1 Byte
```

Stackbefehle

Der Stack arbeitet nach dem LIFO-Prinzip: Der zuletzt abgelegte Wert wird zuerst wieder entnommen.

PUSH AF

```
Bedeutung: Lege das Registerpaar AF auf dem Stack ab.
Hex-Code:  F5
Binär:     11110101
Länge:     1 Byte
```

POP AF

```
Bedeutung: Hole den obersten 16-Bit-Wert vom Stack nach AF.
Hex-Code:  F1
Binär:     11110001
Länge:     1 Byte
```

PUSH BC

```
Bedeutung: Lege das Registerpaar BC auf dem Stack ab.
Hex-Code:  C5
Binär:     11000101
Länge:     1 Byte
```

POP BC

```
Bedeutung: Hole den obersten 16-Bit-Wert vom Stack nach BC.
Hex-Code:  C1
Binär:     11000001
Länge:     1 Byte
```

Programmsteuerung

NOP

```
Bedeutung: Führe keine Operation aus. Der Prozessor geht lediglich
           zum nächsten Befehl weiter.
Hex-Code:  00
Binär:     00000000
Länge:     1 Byte
```

HALT

```
Bedeutung: Beende die Ausführung im Lernapplet.
Hex-Code:  76
Binär:     01110110
Länge:     1 Byte
```

Kurzübersicht

Befehl	Hex-Code	Byte	Kurzbeschreibung
NOP	00	1	keine Operation
HALT	76	1	Ausführung beenden
LD A,n	3E n	2	Wert nach A laden
LD B,n	06 n	2	Wert nach B laden
LD C,n	0E n	2	Wert nach C laden
LD A,r	78-7D	1	Register r nach A kopieren
LD r,A	47-6F	1	A nach Register r kopieren
LD HL,nn	21 lo hi	3	Wert nach HL laden
LD DE,nn	11 lo hi	3	Wert nach DE laden
LD BC,nn	01 lo hi	3	Wert nach BC laden
LD SP,nn	31 lo hi	3	Stackpointer setzen
LD (nn),A	32 lo hi	3	A im Speicher ablegen
LD A,(nn)	3A lo hi	3	A aus dem Speicher laden
INC A	3C	1	A um 1 erhöhen
DEC A	3D	1	A um 1 verringern
ADD A,n	C6 n	2	n zu A addieren
ADD A,B	80	1	B zu A addieren
ADD A,C	81	1	C zu A addieren
ADD A,D	82	1	D zu A addieren
ADD A,E	83	1	E zu A addieren

ADD A,H	84	1	H zu A addieren
ADD A,L	85	1	L zu A addieren
ADD A,(HL)	86	1	Speicherbyte bei HL zu A addieren
ADD A,A	87	1	A verdoppeln
XOR A	AF	1	A auf 0 setzen
CP n	FE n	2	A mit n vergleichen
JP nn	C3 lo hi	3	absolut springen
JP Z,nn	CA lo hi	3	absolut springen, wenn Z gesetzt
JP NZ,nn	C2 lo hi	3	absolut springen, wenn Z nicht gesetzt
JR nn	18 e	2	relativ springen
JR Z,nn	28 e	2	springen, wenn Z gesetzt
JR NZ,nn	20 e	2	springen, wenn Z nicht gesetzt
CALL nn	CD lo hi	3	Unterprogramm aufrufen
RET	C9	1	aus Unterprogramm zurückkehren
PUSH AF	F5	1	AF auf Stack legen
POP AF	F1	1	AF vom Stack holen
PUSH BC	C5	1	BC auf Stack legen
POP BC	C1	1	BC vom Stack holen