

Z80 Hardware Reference

This file contains information on how to program the Z80 microprocessor. No responsibility is taken for the information presented here.

This is based on Magnus Hagander's (mha@ticalc.org) quick reference.

Registers

The following registers are available:

A, B, C, D, E	8-bit registers
AF	16-bit register containing A and flags
BC	16-bit register containing B and C
DE	16-bit register containing D and E
HL	16-bit register used for addressing
F	8-bit Flag register
I	8-bit Interrupt Page address register
IX, IY	16-bit index registers
PC	16-bit Program Counter register
R	8-bit Memory Refresh register
SP	16-bit Stack Pointer register

Flags

The following flags are available:

S	Sign flag (bit 7)
Z	Zero flag (bit 6)
H	Half Carry flag (bit 4)
P	Parity/Overflow flag (bit 2)
N	Add/Subtract flag (bit 1)
C	Carry flag (bit 0)

Addressing methods

The following addressing methods are available:

Mnemonic	Description
n	Immediate addressing (8-bit)
nn	Immediate extending addressing (16-bit)
e	Relative addressing (8-bit; $PC=PC+2+offset$)
[nn]	Extended addressing (16-bit)
[xx+d]	Indexed addressing (16-bit + 8-bit)
r	Register addressing (8-bit)
[rr]	Register indirect addressing (16-bit)
b	Bit addressing
p	Modified page 0 addressing

Symbol descriptions

Here is a short description of the symbols that are used in the instruction list:

b	One bit (0 to 7)
cc	Condition (C,M,NC,NZ,P,PE,PO,Z)
d	One-byte expression (-128 to +127)
dst	Destination s, ss, [BC], [DE], [HL], [nn]
e	One-byte expression (-126 to +129)
m	Any register r, [HL] or [xx+d]
n	One-byte expression (0 to 255)
nn	Two-byte expression (0 to 65535)
pp	Register pair BC, DE, IX or SP
qq	Register pair AF, BC, DE or HL
qq'	Alternative register pair AF, BC, DE or HL
r	Register A, B, C, D, E, H or L
rr	Register pair BC, DE, IY or SP
s	Any register r, value n, [HL] or [xx+d]
src	Source s, ss, [BC], [DE], [HL], nn, [nn]
ss	Register pair BC, DE, HL or SP
xx	Index register IX or IY

Instructions

Flag	Notes	Field meaning	Notes
S	Sign flag (bit 7)	-	Flag unaffected
Z	Zero flag (bit 6)	*	Flag affected
H	Half Carry flag (bit 4)	0	Flag reset
P	Parity/Overflow flag (bit 2)	1	Flag set
N	Add/Subtract flag (bit 1)	?	Unknown
C	Carry flag (bit 0)		

Mnemonic	Flags	Description	Notes
	SZHPNC		
ADC A, s	***V0*	Add with Carry	A=A+s+CY
ADC HL, ss	**?V0*	Add with Carry	HL=HL+ss+CY
ADD A, s	***V0*	Add	A=A+s
ADD HL, ss	--?-0*	Add	HL=HL+ss
ADD IX, pp	--?-0*	Add	IX=IX+pp
ADD IY, rr	--?-0*	Add	IY=IY+rr
AND s	***P00	Logical AND	A=A & s
BIT b, m	?*1?0-	Test Bit	m&{2~b}
CALL cc, nn	-----	Conditional Call	If cc CALL
CALL nn	-----	Unconditional Call	- [SP]=PC, PC=nn
CCF	--?-0*	Complement Carry Flag	CY=~CY
CP s	***V1*	Compare	A-s
CPD	****1-	Compare and Decrement	A-[HL], HL=HL-1, BC=BC-1
CPDR	****1-	Compare, Dec., Repeat	CPD till A=[HL] or BC=0
CPI	****1-	Compare and Increment	A-[HL], HL=HL+1, BC=BC-1
CPIR	****1-	Compare, Inc., Repeat	CPI till A=[HL] or BC=0
CPL	--1-1-	Complement	A=~A
DAA	***P-*	Decimal Adjust Acc.	A=BCD format
DEC s	***V1-	Decrement	s=s-1
DEC xx	-----	Decrement	xx=xx-1
DEC ss	-----	Decrement	ss=ss-1
DI	-----	Disable Interrupts	
DJNZ e	-----	Dec., Jump Non-Zero	B=B-1 till B=0
EI	-----	Enable Interrupts	
EX [SP], HL	-----	Exchange	[SP]<->HL
EX [SP], xx	-----	Exchange	[SP]<->xx
EX AF, AF'	-----	Exchange	AF<->AF'
EX DE, HL	-----	Exchange	DE<->HL
EXX	-----	Exchange	qq<->qq' (except AF)
HALT	-----	Halt	
IM n	-----	Interrupt Mode	(n=0, 1, 2)
IN A, [n]	-----	Input	A=[n]
IN r, [C]	***P0-	Input	r=[C]
INC r	***V0-	Increment	r=r+1
INC [HL]	***V0-	Increment	[HL]=[HL]+1
INC xx	-----	Increment	xx=xx+1
INC [xx+d]	***V0-	Increment	[xx+d]=[xx+d]+1
INC ss	-----	Increment	ss=ss+1
IND	?*??1-	Input and Decrement	[HL]=[C], HL=HL-1, B=B-1
INDR	?1??1-	Input, Dec., Repeat	IND till B=0
INI	?*??1-	Input and Increment	[HL]=[C], HL=HL+1, B=B-1
INIR	?1??1-	Input, Inc., Repeat	INI till B=0

Mnemonic	Flags	Description	Notes
	SZHPNC		
JP [HL]	-----	Unconditional Jump	PC=[HL]
JP [xx]	-----	Unconditional Jump	PC=[xx]
JP nn	-----	Unconditional Jump	PC=nn
JP cc,nn	-----	Conditional Jump	If cc JP
JR e	-----	Unconditional Jump	PC=PC+e
JR cc,e	-----	Conditional Jump	If cc JR(cc=C,NC,NZ,Z)
LD dst,src	-----	Load	dst=src
LD A,i	**0*0-	Load	A=i (i=I,R)
LDD	--0*0-	Load and Decrement	[DE]=[HL],HL=HL-1,#
LDDR	--000-	Load, Dec., Repeat	LDD till BC=0
LDI	--0*0-	Load and Increment	[DE]=[HL],HL=HL+1,#
LDIR	--000-	Load, Inc., Repeat	LDI till BC=0
NEG	***V1*	Negate	A=-A
NOP	-----	No Operation	
OR s	***P00	Logical inclusive OR	A=A s
OTDR	?1??1-	Output, Dec., Repeat	OUTD till B=0
OTIR	?1??1-	Output, Inc., Repeat	OUTI till B=0
OUT [C],r	-----	Output	[C]=r
OUT [n],A	-----	Output	[n]=A
OUTD	?*??1-	Output and Decrement	[C]=[HL],HL=HL-1,B=B-1
OUTI	?*??1-	Output and Increment	[C]=[HL],HL=HL+1,B=B-1
POP xx	-----	Pop	xx=[SP]+
POP qq	-----	Pop	qq=[SP]+
PUSH xx	-----	Push	-[SP]=xx
PUSH qq	-----	Push	-[SP]=qq
RES b,m	-----	Reset bit	m=m&{~2~b}
RET	-----	Return	PC=[SP]+
RET cc	-----	Conditional Return	If cc RET
RETI	-----	Return from Interrupt	PC=[SP]+
RETN	-----	Return from NMI	PC=[SP]+
RL m	**0P0*	Rotate Left	m={CY,m}<-
RLA	--0-0*	Rotate Left Acc.	A={CY,A}<-
RLC m	**0P0*	Rotate Left Circular	m=m<-
RLCA	--0-0*	Rotate Left Circular	A=A<-
RLD	**0P0-	Rotate Left 4 bits	{A,[HL]}={A,[HL]}<- ##
RR m	**0P0*	Rotate Right	m=->{CY,m}
RRA	--0-0*	Rotate Right Acc.	A=->{CY,A}
RRC m	**0P0*	Rotate Right Circular	m=->m
RRCA	--0-0*	Rotate Right Circular	A=->A
RRD	**0P0-	Rotate Right 4 bits	{A,[HL]}=->{A,[HL]} ##
RST p	-----	Restart	(p=0H,8H,10H,...,38H)
SBC A,s	***V1*	Subtract with Carry	A=A-s-CY
SBC HL,ss	***V1*	Subtract with Carry	HL=HL-ss-CY
SCF	--0-01	Set Carry Flag	CY=1
SET b,m	-----	Set bit	m=mv{2~b}
SLA m	**0P0*	Shift Left Arithmetic	m=m*2
SRA m	**0P0*	Shift Right Arith.	m=m/2
SRL m	**0P0*	Shift Right Logical	m=->{0,m,CY}
SUB s	***V1*	Subtract	A=A-s
XOR s	***P00	Logical Exclusive OR	A=AxS