CHAPTER 12
INSTRUCTION SET

12.1 Z8 FUNCTIONAL SUMMARY

Z8 instructions can be divided functionally into the following eight groups:

- Load
- Bit Manipulation
- Arithmetic
- Block Transfer
- Logical
- Rotate and Shift
- Program Control
- CPU Control

The following summary shows the instructions belonging to each group and the number of operands required for each. The source operand is src, the destination operand is dst, and a condition code is cc.

Table 12-1. Load Instructions

Mnemonic	Operands	Instruction
CLR	dst	Clear
LD	dst, src	Load
LDC	dst, src	Load Constant
LDE	dst, src	Load External
POP	dst	Pop
PUSH	src	Push

Table 12-2. Arithmetic Instructions

Mnemonic	Operands	Instruction
ADC	dst, src	Add with Carry
ADD	dst, src	Add
CP	dst, src	Compare
DA	dst	Decimal Adjust
DEC	dst	Decrement
DECW	dst	Decrement Word
INC	dst	Increment
INCW	dst	Increment Word
SBC	dst, src	Subtract with Carry
SUB	dst, src	Subtract

Table 12-3. Logical Instructions

Mnemonic	Operands	Instruction
AND	dst, src	Logical AND
COM	dst	Complement
OR	dst, src	Logical OR
XOR	dst, src	Logical Exclusive OR

Table 12-4. Program Control Instructions

Mnemonic	Operands	Instruction
CALL	dst	Call Procedure
DJNZ	dst, src	Decrement and Jump Non-Zero
IRET		Interrupt Return
JP	cc, dst	Jump
JR	cc, dst	Jump Relative
RET		Return

Table 12-5. Bit Manipulation Instructions

Mnemonic	Operands	Instruction
TCM	dst, src	Test Complement Under Mask
TM	dst, src	Test Under Mask
AND	dst, src	Bit Clear
OR	dst, src	Bit Set
XOR	dst, src	Bit Complement

Table 12-6. Block Transfer Instructions

Mnemonic	Operands	Instruction
LDCI	dst, src	Load Constant Auto Increment
LDEI	dst, src	Load External Auto Increment

Table 12-7. Rotate and Shift Instructions

Mnemonic	Operands	Instruction
RL	dst	Rotate Left
RLC	dst	Rotate Left Through Carry
RR	dst	Rotate Right
RRC	dst	Rotate Right Through Carry
SRA	dst	Shift Right Arithmetic
SWAP	dst	Swap Nibbles

Table 12-8. CPU Control Instructions

Mnemonic	Operands	Instruction
CCF		Complement Carry Flag
DI		Disable Interrupts
EI		Enable Interrupts
HALT		Halt
NOP		No Operation
RCF		Reset Carry Flag
SCF		Set Carry Flag
SRP	src	Set Register Pointer
STOP		Stop
WDH		WDT Enable During HALT
WDT		WDT Enable or Refresh

12.2 PROCESSOR FLAGS

The Flag Register (FCH) informs the user of the current status of the Z8. The flags and their bit positions in the Flag Register are shown in Figure 12-1.

The Z8 Flag Register contains six bits of status information which are set or cleared by CPU operations. Four of the bits (C, V, Z and S) can be tested for use with conditional Jump instructions. Two flags (H and D) cannot be tested and are used for BCD arithmetic. The two remaining bits in the Flag Register (F1 and F2) are available to the user, but they must be set or cleared by instructions and are not usable with conditional Jumps.

As with bits in the other control registers, the Flag Register bits can be set or reset by instructions; however, only those instructions that do not affect the flags as an outcome of the execution should be used (Load Immediate).

Note: The Watch-Dog Timer (WDT) instruction effects the Flags accordingly: Z=1, S=0, V=0.

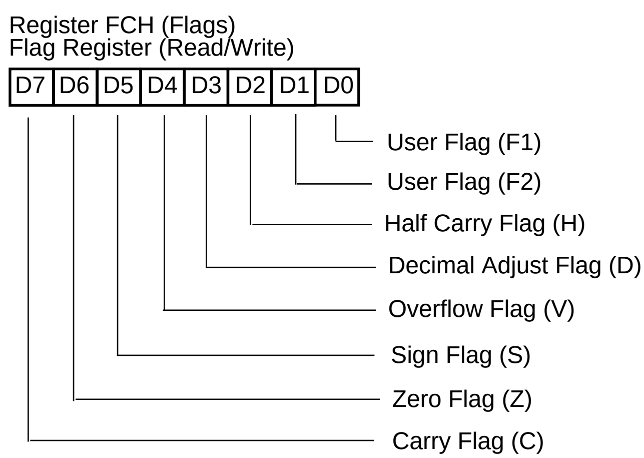


Figure 12-1. Z8 Flag Register

12.2.1 Carry Flag (C)

The Carry Flag is set to 1 whenever the result of an arithmetic operation generates a “carry out of” or a “borrow into” the high order bit 7. Otherwise, the Carry Flag is cleared to 0.

Following Rotate and Shift instructions, the Carry Flag contains the last value shifted out of the specified register.

An instruction can set, reset, or complement the Carry Flag.

IRET may change the value of the Carry Flag when the Flag Register, saved in the Stack, is restored.

12.2.2 Zero Flag (Z)

For arithmetic and logical operations, the Zero Flag is set to 1 if the result is zero. Otherwise, the Zero Flag is cleared to 0.

If the result of testing bits in a register is 00H, the Zero Flag is set to 1. Otherwise the Zero Flag is cleared to 0.

If the result of a Rotate or Shift operation is 00H, the Zero Flag is set to 1. Otherwise, the Zero Flag is cleared to 0.

IRET changes the value of the Zero Flag when the Flag Register saved in the Stack is restored. The WDT Instruction sets the Zero Flag to a 1.

12.2.3 Sign Flag (S)

The Sign Flag stores the value of the most significant bit of a result following an arithmetic, logical, Rotate, or Shift operation.

When performing arithmetic operations on signed numbers, binary two’s-complement notation is used to represent and process information. A positive number is identified by a 0 in the most significant bit position (bit 7); therefore, the Sign Flag is also 0.

A negative number is identified by a 1 in the most significant bit position (bit 7); therefore, the Sign Flag is also 1.

IRET changes the value of the Sign Flag when the Flag Register saved in the Stack is restored.

12.2.4 Overflow Flag (V)

For signed arithmetic, Rotate, and Shift operations, the Overflow Flag is set to 1 when the result is greater than the maximum possible number (>127) or less than the minimum possible number (<–128) that can be represented in two’s-complement form. The Overflow Flag is set to 0 if no overflow occurs.

Following logical operations the Overflow Flag is set to 0.

IRET changes the value of the Overflow Flag when the Flag Register saved in the Stack is restored.

12.2.5 Decimal Adjust Flag (D)

The Decimal Adjust Flag is used for BCD arithmetic. Since the algorithm for correcting BCD operations is different for addition and subtraction, this flag specifies what type of instruction was last executed so that the subsequent Decimal Adjust (DA) operation can function properly. Normally, the Decimal Adjust Flag cannot be used as a test condition.

After a subtraction, the Decimal Adjust Flag is set to 1. Following an addition it is cleared to 0.

IRET changes the value of the Decimal Adjust Flag when the Flag Register saved in the Stack is restored.

12.2.6 Half Carry Flag (H)

The Half Carry Flag is set to 1 whenever an addition generates a “carry out of” bit 3 (Overflow) or a subtraction generates a “borrow into” bit 3. The Half Carry Flag is used by the Decimal Adjust (DA) instruction to convert the binary result of a previous addition or subtraction into the correct decimal (BCD) result. As in the case of the Decimal Adjust Flag, the user does not normally access this flag.

IRET changes the value of the Half Carry Flag when the Flag Register saved in the Stack is restored.

12.3 CONDITION CODES

The C, Z, S, and V Flags control the operation of the 'Conditional' Jump instructions. Sixteen frequently useful functions of the flag settings are encoded in a 4-bit field called the condition code (cc), which forms bits 4-7 of the conditional instructions.

Condition codes and flag settings are summarized in Tables 12-9, 12-10, and 12-11. Notation for the flags and how they are affected are as follows:

Table 12-9. Z8 Flag Definitions

Flag	Description
C	Carry Flag
Z	Zero Flag
S	Sign Flag
V	Overflow Flag
D	Decimal Adjust Flag
H	Half Carry Flag

Table 12-10. Flag Settings Definitions

Symbol	Definition
0	Cleared to 0
1	Set to 1
*	Set or cleared according to operation
–	Unaffected
X	Undefined

Table 12-11. Condition Codes

Binary	HEX	Mnemonic	Definition	Flag Settings
0000	0	F	Always False	–
1000	8	(blank)	Always True	–
0111	7	C	Carry	C = 1
1111	F	NC	No Carry	C = 0
0110	6	Z	Zero	Z = 1
1110	E	NZ	Non-Zero	Z = 0
1101	D	PL	Plus	S = 0
0101	5	MI	Minus	S = 1
0100	4	OV	Overflow	V = 1
1100	C	NOV	No Overflow	V = 0
0110	6	EQ	Equal	Z = 1
1110	E	NE	Not Equal	Z = 0
1001	9	GE	Greater Than or Equal	(S XOR V) = 0
0001	1	LT	Less Than	(S XOR V) = 1
1010	A	GT	Greater Than	(Z OR (S XOR V)) = 0
0010	2	LE	Less Than or Equal	(Z OR (S XOR V)) = 1
1111	F	UGE	Unsigned Greater Than or Equal	C = 0
0111	7	ULT	Unsigned Less Than	C = 1
1011	B	UGT	Unsigned Greater Than	(C = 0 AND Z = 0) = 1
0011	3	ULE	Unsigned Less Than or Equal	(C OR Z) = 1

12.4 NOTATION AND BINARY ENCODING

In the detailed instruction descriptions that make up the rest of this chapter, operands and status flags are represented by a notational shorthand. Operands, condition codes, address modes, and their notations are as follows (Table 12-12):

Table 12-12. Notational Shorthand

Notation	Address Mode	Operand	Range *
cc	Condition Code		See condition codes
r	Working Register	Rn	n = 0 – 15
R	Register or Working Register	Reg Rn	Reg. represents a number in the range of 00H to FFH n = 0 – 15
RR	Register Pair or Working Register Pair	Reg RRp	Reg. represents an even number in the range of 00H to FEH p = 0, 2, 4, 6, 8, 10, 12, or 14
Ir	Indirect Working Register	@Rn	n = 0 –15
IR	Indirect Register or Indirect Working Register	@Reg @Rn	Reg. represents a number in the range of 00H to FFH n = 0– 15
Irr	Indirect Working Register Pair	@RRp	p = 0, 2, 4, 6, 8, 10, 12, or 14
IRR	Indirect Register Pair or Working Register Pair	@Reg @RRp	Reg. represents an even number in the range 00H to FFH p = 0, 2, 4, 6, 8, 10, 12, or 14
X	Indexed	Reg (Rn)	Reg. represents a number in the range of 00H to FFH and n = 0 – 15
DA	Direct Address	Addrs	Addrs. represents a number in the range of 00H to FFH
RA	Relative Address	Addrs	Addrs. represents a number in the range of +127 to –128 which is an offset relative to the address of the next instruction
IM	Immediate	#Data	Data is a number between 00H to FFH

*See the device product specification to determine the exact register file range available. The register file size varies by the device type.

12.4 NOTATION AND BINARY ENCODING (Continued)

Additional symbols used are:

Table 12-13. Additional Symbols

Symbol	Definition
dst	Destination Operand
src	Source Operand
@	Indirect Address Prefix
SP	Stack Pointer
PC	Program Counter
FLAGS	Flag Register (FCH)
RP	Register Pointer (FDH)
IMR	Interrupt Mask Register (FBH)
#	Immediate Operand Prefix
%	Hexadecimal Number Prefix
H	Hexadecimal Number Suffix
B	Binary Number Suffix
OPC	Opcode

Assignment of a value is indicated by the symbol “¨”. For example,

dst ¨ dst + src

indicates the source data is added to the destination data and the result is stored in the destination location.

The notation 'addr(n)' is used to refer to bit'n' of a given location. For example,

dst (7)

refers to bit 7 of the destination operand.

12.4.1 Assembly Language Syntax

For proper instruction execution, Z8 assembly language syntax requires ‘dst, src’ be specified, in that order. The following instruction descriptions show the format of the object code produced by the assembler. This binary format should be followed by users who prefer manual program coding or who intend to implement their own assembler.

Example: If the contents of registers 43H and 08H are added and the result is stored in 43H, the assembly syntax and resulting object code is:

ASM:	ADD	43H,	08H	(ADD dst, src)
OBJ:	04	08	43	(OPC src, dst)

In general, whenever an instruction format requires an 8-bit register address, that address can specify any register location in the range 0 - 255 or a Working Register R0 - R15. If, in the above example, register 08H is a Working Register, the assembly syntax and resulting object code would be:

ASM:	ADD	43H,	08H	(ADD dst, src)
OBJ:	04	08	43	(OPC src, dst)

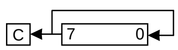
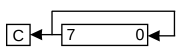
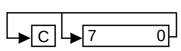
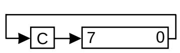
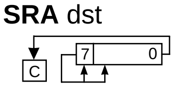
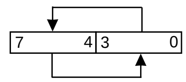
Note: See the device product specification to determine the exact register file range available. The register file size varies by device type

12.5 Z8 INSTRUCTION SUMMARY

Instruction and Operation	Address Mode dst	Address Mode src	Opcode Byte (Hex)	C	Z	S	V	D	H
ADC dst, src dst ← dst + src +C	†		1[ ]	*	*	*	*	0	*
ADD dst, src dst ← dst + src	†		0[ ]	*	*	*	*	0	*
AND dst, src dst ← dst AND src	†		5[ ]	–	*	*	0	–	–
CALL dst	DA		D6	–	–	–	–	–	–
SP ← SP – 2	IRR		D4						
and PC ← dst or @ SP ← PC									
CCF C ←NOT C			EF	*	–	–	–	–	–
CLR dst	R		B0	–	–	–	–	–	–
dst ← 0	IR		B1						
COM dst	R		60	–	*	*	0	–	–
dst ← NOT dst	IR		61						
CP dst, src dst – src	†		A[ ]	*	*	*	*	–	–
DA dst	R		40	*	*	*	X	–	–
dst ← DA dst	IR		41						
DEC dst	R		00	–	*	*	*	–	–
dst ← dst – 1	IR		01						
DECW dst	R R		80	–	*	*	*	–	–
dst ← dst – 1	IR		81						
DI dst IMR(7) ← 0			8 F	–	–	–	–	–	–
DJNZr, dst	RA		rA	–	–	–	–	–	–
r ← r – 1 if r ≠ 0 PC ← PC + dst Range:+127, -128			r=0-F						
EI IMR(7) ← 1			9 F	–	–	–	–	–	–
HALT			7 F	–	–	–	–	–	–
INC dst dst ← dst + 1	r		rE r=0-F	–	*	*	*	–	–
	R		20						
	IR		21						
INCW dst	RR		A0	–	*	*	*	–	–
dst ← dst + 1	IR		A1						

Instruction and Operation	Address Mode dst	Address Mode src	Opcode Byte (Hex)	C	Z	S	V	D	H
IRET FLAGS←@SP; SP ← SP + 1 PC ← @SP; SP ← SP + 2; and IMR(7) – 1			B F	*	*	*	*	*	*
JP cc, dst	DA		cD	–	–	–	–	–	–
if cc is true,			c = 0 – F						
then PC ← dst	IRR		30						
JR cc, dst	RA		cB	–	–	–	–	–	–
if cc is true, PC ← PC + dst Range: +127, –128			c = 0 – F						
LD dst, src	r	Im	r C	–	–	–	–	–	–
dst ← src	r	R	r 8						
	R	r	r 9 r = 0 – F						
	r	X	C 7						
	X	r	D 7						
	r	Ir	E 3						
	Ir	r	F 3						
	R	R	E 4						
	R	IR	E 5						
	R	IM	E 6						
	IR	IM	E 7						
	IR	R	F 5						
LDC dst, src	r	Irr	C 2	–	–	–	–	–	–
dst ← src	Irr	r	D 2						
LDCI dst, src	Ir	Irr	C 3	–	–	–	–	–	–
dst ← src r ← r + 1 or rr ←rr + 1	Irr	r	D 3						
LDE dst, src	r	Irr	82	–	–	–	–	–	–
dst ← src	Irr	r	92						
LDEI dst, src	r	Irr	C 2	–	–	–	–	–	–
dst ← src and r ← r + 1 or rr ←rr + 1	Irr	r	D 2						
NOP			FF	–	–	–	–	–	–
OR dst, src dst ← dst OR src	†		4[ ]	–	*	*	0	–	–

12.5 Z8 INSTRUCTION SUMMARY (Continued)

Instruction and Operation	Address Mode dst	Address Mode src	Opcode Byte (Hex)	Flags Affected C Z S V D H
POP dst dst ← @SP and SP ← SP + 1	R IR		50 51	– – – – – –
PUSH src SP ← SP – 1 and @SP ← src	R IR		70 71	– – – – – –
RCF C ← 0			C F	0 – – – – –
RET PC ← @SP; SP ← SP + 2			A F	– – – – – –
RL dst 	R IR		90 91	* * * * – –
RLC dst 	R IR		10 11	* * * * – –
RR dst 	R IR		E 0 E 1	* * * * – –
RRC dst 	R IR		C 0 C 1	* * * * – –
SBC dst, src dst ← dst – src – C	†		3[ ]	* * * * 1 *
SCF C ← 1			D F	1 – – – – –
SRA dst 	R IR		D 0 D 1	* * * 0 – –
SRP dst RP ← src	Im		31	– – – – – –
STOP			6 F	– – – – – –
SUB dst, src dst ← dst – src	†		2[ ]	* * * * 1 *
SWAP dst 	R IR		F0 F1	X * * X – –
TCM dst, src (NOT dst) AND src	†		6[ ]	– * * 0 – –
TM dst, src dst AND src	†		7[ ]	– * * 0 – –
WDH			4 F	– X X X – –
WDT			5 F	– X X X – –
XOR dst, src dst AND src XOR src	†		7[ ]	– * * 0 – –

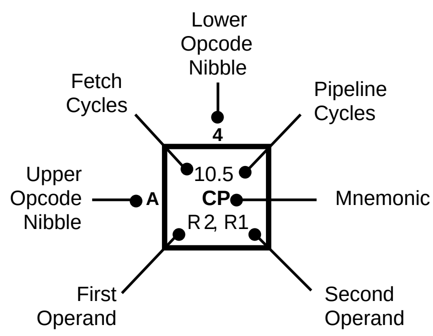
Note: † These instructions have an identical set of addressing modes, which are encoded for brevity. The first opcode nibble is found in the instruction set table above. The second nibble is expressed symbolically by a '[ ]' in this table, and its value is found in the following table to the left of the applicable addressing mode pair. For example, the opcode of an ADC instruction using the addressing modes r (destination) and Ir (source) is 13.

Address dst	Mode src	Lower Opcode Nibble
r	r	[2]
r	Ir	[3]
R	R	[4]
R	IR	[5]
R	IM	[6]
IR	IM	[7]

12.5.1 OPCODE MAP

Lower Nibble (Hex) across the top; Upper Nibble (Hex) down the side.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	6.5 **DEC** R1	6.5 **DEC** IR1	6.5 **ADD** r1, r2	6.5 **ADD** r1, Ir2	10.5 **ADD** R2, R1	10.5 **ADD** IR2, R1	10.5 **ADD** R1, IM	10.5 **ADD** IR1, IM	6.5 **LD** r1, R2	6.5 **LD** r2, R1	12/10.5 **DJNZ** r1, RA	12/10.0 **JR** cc, RA	6.5 **LD** r1, IM	12.10.0 **JP** cc, DA	6.5 **INC** r1	
1	6.5 **RLC** R1	6.5 **RLC** IR1	6.5 **ADC** r1, r2	6.5 **ADC** r1, Ir2	10.5 **ADC** R2, R1	10.5 **ADC** IR2, R1	10.5 **ADC** R1, IM	10.5 **ADC** IR1, IM								
2	6.5 **INC** R1	6.5 **INC** IR1	6.5 **SUB** r1, r2	6.5 **SUB** r1, Ir2	10.5 **SUB** R2, R1	10.5 **SUB** IR2, R1	10.5 **SUB** R1, IM	10.5 **SUB** IR1, IM								
3	8.0 **JP** IRR1	6.1 **SRP** IM	6.5 **SBC** r1, r2	6.5 **SBC** r1, Ir2	10.5 **SBC** R2, R1	10.5 **SBC** IR2, R1	10.5 **SBC** R1, IM	10.5 **SBC** IR1, IM								
4	8.5 **DA** R1	8.5 **DA** IR1	6.5 **OR** r1, r2	6.5 **OR** r1, Ir2	10.5 **OR** R2, R1	10.5 **OR** IR2, R1	10.5 **OR** R1, IM	10.5 **OR** IR1, IM								6.0 **WDH**
5	10.5 **POP** R1	10.5 **POP** IR1	6.5 **AND** r1, r2	6.5 **AND** r1, Ir2	10.5 **AND** R2, R1	10.5 **AND** IR2, R1	10.5 **AND** R1, IM	10.5 **AND** IR1, IM								6.0 **WDT**
6	6.5 **COM** R1	6.5 **COM** IR1	6.5 **TCM** r1, r2	6.5 **TCM** r1, Ir2	10.5 **TCM** R2, R1	10.5 **TCM** IR2, R1	10.5 **TCM** R1, IM	10.5 **TCM** IR1, IM								6.0 **STOP**
7	10/12.1 **PUSH** R2	12/14.1 **PUSH** IR2	6.5 **TM** r1, r2	6.5 **TM** r1, Ir2	10.5 **TM** R2, R1	10.5 **TM** IR2, R1	10.5 **TM** R1, IM	10.5 **TM** IR1, IM								7.0 **HALT**
8	10.5 **DECW** RR1	10.5 **DECW** IR1	12.0 **LDE** r1, Irr2	18.0 **LDEI** Ir1, Irr2												6.1 **DI**
9	6.5 **RL** R1	6.5 **RL** IR1	12.0 **LDE** r2, Irr1	18.0 **LDEI** Ir2, Irr1												6.1 **EI**
A	10.5 **INCW** RR1	10.5 **INCW** IR1	6.5 **CP** r1, r2	6.5 **CP** r1, Ir2	10.5 **CP** R2, R1	10.5 **CP** IR2, R1	10.5 **CP** R1, IM	10.5 **CP** IR1, IM								14.0 **RET**
B	6.5 **CLR** R1	6.5 **CLR** IR1	6.5 **XOR** r1, r2	6.5 **XOR** r1, Ir2	10.5 **XOR** R2, R1	10.5 **XOR** IR2, R1	10.5 **XOR** R1, IM	10.5 **XOR** IR1, IM								16.0 **IRET**
C	6.5 **RRC** R1	6.5 **RRC** IR1	12.0 **LDC** r1, Irr2	18.0 **LDCI** Ir1, Irr2				10.5 **LD** r1,x,R2								6.5 **RCF**
D	6.5 **SRA** R1	6.5 **SRA** IR1	12.0 **LDC** Irr1, r2	18.0 **LDCI** Irr1, Ir2	20.0 **CALL*** IRR1		20.0 **CALL** DA	10.5 **LD** r2,x,R1								6.5 **SCF**
E	6.5 **RR** R1	6.5 **RR** IR1		6.5 **LD** r1, IR2	10.5 **LD** R2, R1	10.5 **LD** IR2, R1	10.5 **LD** R1, IM	10.5 **LD** IR1, IM								6.5 **CCF**
F	8.5 **SWAP** R1	8.5 **SWAP** IR1		6.5 **LD** Ir1, r2		10.5 **LD** R2, IR1										6.0 **NOP**
Bytes per Instruction	2				3				2					3	1	



Legend:
R = 8-bit Address
r = 4-bit Address
R1 or r1 = Dst Address
R2 or r2 = Src Address

Sequence:
Opcode, First Operand,
Second Operand

Note: Blank areas are reserved.

*2-byte instruction appears as a 3-byte instruction