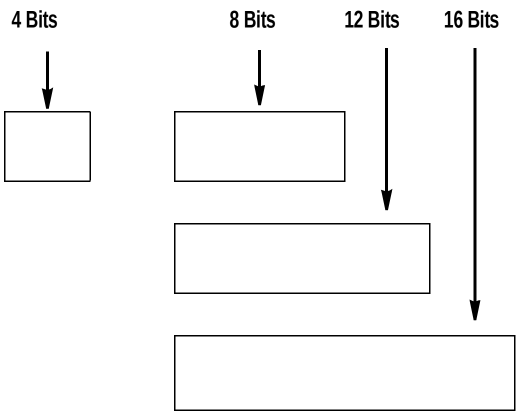
INSTRUCTION DESCRIPTION AND FORMATS

The following section lists each instruction set, and describes the:

- Instruction Format
- Operation performed
- Flag Conditions
- Examples of the code

The format for the instruction uses the following conventions:



NOTE: The bytes shown in the boxes are in machine code order. The ZiLOG assembler always requires the format `OPC, dst, src`.

Address modes `R` or `IR` can be used to specify a 4-bit working register. In this format, the source or destination working-register operand is specified by adding `1110B` (`EH`) to the High nibble of the operand. For example, if working register `R12` (`CH`) is the destination operand, then `ECH` is used as the destination operand in the Op Code.

E	src	or	E	dst

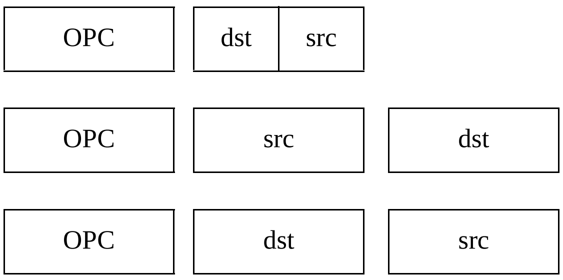
Address mode `IRR` can be used to specify a 4-bit working register Pair. In this format, the destination working register Pair operand is specified by adding `1110B` (`EH`) to the High nibble of the operand. For example, if working register Pair `RR12` (`CH`) is the destination operand, then `ECH` is used as the destination operand in the Op Code.

E	dst

ADC
Add with Carry

Instruction Format:

`ADC dst, src`



	Address Mode	
OPC (Hex)	dst	src
12	r	r
13	r	Ir
14	R	R
15	R	IR
16	R	IM
17	IR	IM

Operation:

`dst ← dst + src + C`

The source operand, along with the setting of the Carry (`C`) Flag, is added to the destination operand. Two's complement addition is performed. The sum is stored in the destination operand. The contents of the source operand are not changed. In multiple precision arithmetic, this instruction permits the carry from the addition of low order operands to be carried into the addition of high order operands.

Flags:

When the instruction is executed, the flags are set as follows:

C:	1 if a value is carried from the most signigicant bit of the result; otherwise, 0.
Z:	1 if the result is `0`; otherwise, 0.
S:	1 if the result is a negative value; otherwise, 0.
V:	1 if an arithmetic overflow occurs (both operands have the same sign and the result has the opposite sign; otherwise, 0.
D:	0.
H:	1 if a value is carried from the most significant bit of the low-order four bits of the result; otherwise, 0.

ADC
Add with Carry

Example: Working register R3 contains 16H. The C flag is set to 1. Working register R11 contains 20H. The following statement leaves the value 37H in working register R3, and the C, Z, S, V, D, and H flags are set to 0.

```
ADC R3, R11
Op Code: 12 3B.
```

Example: Working register R16 contains 16H. The C flag is not set. Working register R10 contains 20H. Register 20H contains 11H. The following statements leave the value 27H in working register R16; the C, Z, S, V, D, and H flags are set to 0.

```
ADC R16, @R10
Op Code: 13 FA
```

Example: Register 34H contains 2EH. The C flag is set. Register 12H contains 1BH. The following statement leaves the value 4AH in register 34H. The H flag is set, and the C, Z, S, V, and D flags are set to 0.

```
ADC 34H, 12H
Op Code: 14 12 34
```

Example: Register 4BH contains 82H. The C flag is set. Working register R3 contains 10H. Register 10H contains 01H. The following statement leaves the value 84H in register 4BH. The S flag is set to 1, and the C, Z, V, D, and H flags are set to 0.

```
ADC 4BH, @R3
Op Code: 15 E3 4B
```

ADC
Add with Carry

Example: Register 6CH contains 2AH. The C flag is not set. The following statement leaves the value 2DH in register 6CH. The C, Z, S, V, D, and H flags are set to 0.

```
ADC 6CH, #03H
Op Code: 16 6C 03
```

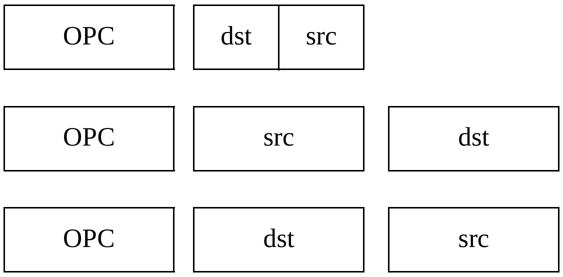
Example: Register D4H contains 5FH. Register 5FH contains 4CH. The C flag is set. The following statement leaves the value 4FH in register 5FH. The C, Z, S, V, D, and H flags are set to 0.

```
ADC @D4H, #02H
Op Code: 17 D4 02
```

ADD
Add

Instruction Format:

```
ADD dst, src
```



	Address Mode	
OPC (Hex)	**dst**	**src**
02	r	r
03	r	Ir
04	R	R
05	R	IR
06	R	IM
07	IR	IM

Operation:

```
dst ← dst + src
```

The source operand is added to the destination operand. Two's complement addition is performed. The sum is stored in the destination operand. The contents of the source operand are not changed.

Flags:

When the instruction is executed, the flags are set as follows:

C:	1 if a value is carried from the most significant bit of the result; otherwise, 0.
Z:	1et if the result is 0; otherwise, 0.
S:	1 if the result is negative; otherwise, 0.
V:	1 if an arithmetic overflow occurs(both operands have the same sign and the result has the opposite sign); otherwise, 0.
D:	0.
H:	1 if a value is carried from the most significant bit of the result's low-order four bits; otherwise, 0.

ADD
Add

Example: Working register R3 contains 16H. Working register R11 contains 20H. The following statement leaves the value 36H in working register R3. The C, Z, S, V, D, and H flags are set to 0.

```
ADD R3, R11
Op Code: 02 3B
```

Example: Working register R16 contains 16H. Working register R10 contains 20H. Register 20H contains 11H. The following statement leaves the value 27H in working register R16. The C, Z, S, V, D, and H flags are set to 0.

```
ADD R16,@R10
Op Code: 03 FA
```

Example: Register 34H contains 2EH. Register 12H contains 1BH. The following statement leaves the value 49H in register 34H. The H flag is set to 1, and the C, Z, S, V, and D flags are set to 0.

```
ADD 34H,12H
Op Code: 04 12 34
```

Example: Register 4BH contains 82H. Working register R3 contains 10H. Register 10H contains 01H. The following statement leaves the value 83H in register 4BH. The S flag is set, and the C, Z, V, D, and H flags are set to 0.

```
ADD 3EH, @R3
Op Code: 05 E3 4B
```

Example: Register 6CH contains 2AH. The following statement leaves the value 2DH in register 6CH. The C, Z, S, V, D, and H flags are set to 0.

```
ADD 6CH, #03H
Op Code: 06 6C 03
```

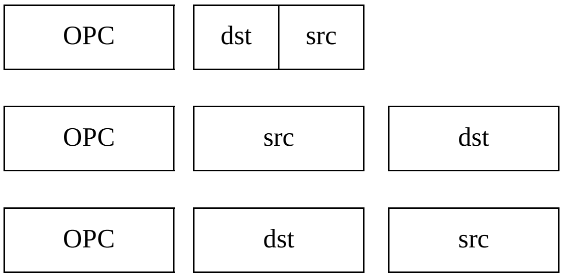
Example: Register D4H contains 5FH. Register 5FH contains 4CH. The following statement leaves the value 4EH in register 5FH. The C, Z, S, V, D, and H flags are set to 0.

```
ADD @D4H, #02H
Op Code: 07 D4 02
```

AND
Logical AND

Instruction Format:

```
AND dst, src
```



	Address Mode	
OPC (Hex)	dst	src
52	r	r
53	r	Ir
54	R	R
55	R	IR
56	R	IM
57	IR	IM

Operation:

```
dst ← dst AND src
```

The source operand and the destination operandare processed with a logical AND operation. The result is a `1` stored whenever the corresponding bits in the two operands are both `1`; otherwise, a `0` is stored. The result is stored in the destination operand. The contents of the source register are unchanged.

Flags:

When the instruction is executed, the flags are set as follows:

C:	The value set by the preceding instruction.
Z:	1 if the result is `0`; otherwise, 0.
S:	1 if bit `7` of the result is 1; otherwise, 0.
V:	`0`
D:	The value set by the preceding instruction.
H:	The value set by the preceding instruction.

Example: Working register `R1` contains `34H` (`00111000B`) and working register `R14` contains `4DH` (`10001101`). The following statement leaves the value `04H` (`00001000`) in working register `R1`. The `Z`, `V`, and `S` flags are set to 0.

```
AND R1, R14
Op Code: 52 1E
```

AND
Logical AND

Example: Working register R4 contains F9H (11111001B). Working register R13 contains 7BH. Register 7BH contains 6AH (01101010B). The following statement leaves the value 68H (01101000B) in working register R4. The Z, V, and S flags are set to 0.

```
AND R4, @R13
Op Code: 53 4D
```

Example: Register 3AH contains the value F5H (11110101B). Register 42H contains the value 0AH (00001010). The following statement leaves the value 00H (00000000B) in register 3AH. The Z flag is setto 1, and the V and S flags are cleared.

```
AND 3AH, 42H
Op Code: 54 42 3A
```

Example: If working register R5 contains F0H (11110000B). Register 45H contains 3AH. Register 3AH contains 7FH (01111111B). The following statement leaves the value 70H (01110000B) in working register R5. The Z, V, and S flags are set to 0.

```
AND R5, @45H
Op Code: 55 45 E5
```

Example: Register 7AH contains the value F7H (11110111B). The following statement leaves the value F0H (11110000B) in register 7AH. The S flag is set to 1, and the Z and V flags are set to 0.

```
AND 7AH, #F0H
Op Code: 56 7A F0
```

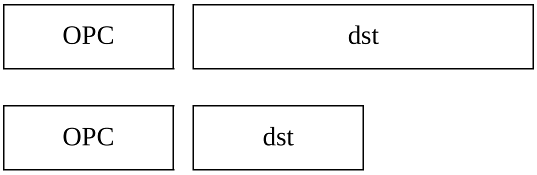
Example: Working register R3 contains the value 3EH. Register 3EH contains the value ECH (11101100B). The following statement leaves the value 04H (00000100B) in register 3EH. The Z, V, and S flags are set to 0.

```
AND @R3, #05H
Op Code: 57 E3 05
```

CALL
Call Procedure

Instruction Format:

```
CALL dst
```



OPC (Hex)	Address Mode dst
D6	DA
D4	IRR

Operation:

```
SP ← SP - 2
@SP ← PC
PC ← dst
```

The Stack pointer (SP) is decremented by 2. The current contents of the program counter (PC) (the address of the first instruction following the `CALL` instruction) are pushed onto the top of the Stack. The specified destination address is then loaded into the PC, which points to the first instruction of the procedure.

At the end of the procedure a return (`RET`) instruction can be used to return to the original program flow. `RET` pops the top of the Stack and replaces the original value into the PC.

Flags:

When the instruction is executed, the flags are set as follows:

C:	The value set by the preceding instruction.
Z:	The value set by the preceding instruction.
S:	The value set by the preceding instruction.
V:	The value set by the preceding instruction.
D:	The value set by the preceding instruction.
H:	The value set by the preceding instruction.

CALL
Call Procedure

Example: The contents of the PC are 1A47H and the contents of the SP (registers FEH and FFH) are 3002H. The following statements cause the SP to be decremented to 3000H, 1A4AH.The address following the CALL instructionis stored in external data memory at addresses 3000 and 3001H. The PC is loaded with 3521H and now points to the address of the first statement in the procedure to be executed.

```
CALL 3521H
Op Code: D6 35 21
```

Example: The contents of the PC are 1A47H. The contents of the SP (register FFH) are 72H. The contents of register A4H are 34H. The contents of register pair 34H are 3521H. The following statements cause the SP to be decremented to 70H, 1A4AH.The address following the CALL instructionis stored in R70H and 71H. The PC is loaded with 3521H and now points to the address of the first statement in the procedure to be executed

```
CALL @A4H
Op Code: D4 A4
```

CCF
Complement Carry Flag

Instruction Format:

```
CCF
```

OPC	OPC (Hex)
	EF

Operation:

```
C ← NOT C
```

The `C` flag is complemented. If `C = 1`, then it is changed to `C = 0`; or, if `C = 0`, then it is changed to `C = 1`.

Flags:

When the instruction is executed, the flags are set as follows:

C:	The value set by the preceding instruction is complemented.
Z:	The value set by the preceding instruction.
S:	The value set by the preceding instruction.
V:	The value set by the preceding instruction.
D:	The value set by the preceding instruction.
H:	The value set by the preceding instruction.

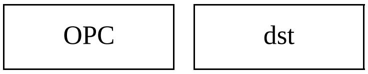
Example: The `C` flag contains a `0`. The following statement changes the `C` flag from `C = 0` to `C = 1`.

```
CCF
Op Code: EF
```

CLR
Clear

Instruction Format:

```
CLR dst
```



OPC (Hex)	Address Mode dst
B0	R
B1	IR

Operation:

```
dst ← 0
```

The destination operand is set to 00H.

Flags

When the instruction is executed, the flags are set as follows:

C:	The value set by the preceding instruction.
Z:	The value set by the preceding instruction.
S:	The value set by the preceding instruction.
V:	The value set by the preceding instruction.
D:	The value set by the preceding instruction.
H:	The value set by the preceding instruction.

Example: Working register `R6` contains `AFH`. The following statement leaves the value `00H` in working register `R6.`

```
CLR R6
Op Code: B0 E6
```

Example: Register `A5H` contains the value `23H`. Register `23H` contains the value `FCH`. The following statement leaves the value `00H` in register `23H`.

```
CLR @A5H
Op Code: B1 A5
```

COM
Complement

Instruction Format:

```
COM dst
```

OPC	dst

OPC (Hex)	Address Mode dst
60	R
61	IR

Operation:

```
dst ← NOT dst
```

The contents of the destination operand are complemented (one’s complement). All `1` bits are changed to `0`, and all `0` bits are changed to `1`.

Flags:

When the instruction is executed, the flags are set as follows:

C:	The value set by the preceding instruction.
Z:	1 if the result is `0`; otherwise, 0.
S:	1 if result bit `7` is set; otherwise, 0.
V:	`0`
D:	The value set by the preceding instruction.
H:	The value set by the preceding instruction.

Example: Register `08H` contains `24H` (`00100100B`). The following statement leaves the value `DBH` (`11011011`) in register `08H`. The `S` flag is set to 1, and the `Z` and `V` flags are set to 0.

```
COM 08
Op Code: 60 08
```

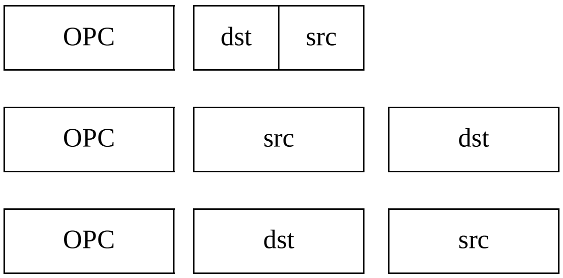
Example: Register `08H` contains `24H`, and register `24H` contains `FFH` (`11111111B`). The following statement leaves the value `00H` (`00000000B`) in register `24H`. The `Z` flag is set to 1, and the `V` and `S` flags are set to 0.

```
COM @08H
Op Code: 61 08
```

CP
Compare

Instruction Format:

```
CP dst, src
```



	Address Mode	
OPC (Hex)	**dst**	**src**
A2	r	r
A3	r	Ir
A4	R	R
A5	R	IR
A6	R	IM
A7	IR	IM

Operation:

```
dst - src
```

The source operand is compared to (subtracted from) the destination operand, and the appropriate flags are set accordingly. The contents of both operands are unchanged.

Flags:

When the instruction is executed, the flags are set as follows:

C:	1 if a value is carried from the most significant bit of the result, otherwise, 0.
Z:	1 if the result is 0; otherwise, 0.
S:	1 if bit 7 of the result is 1 (negative); otherwise, 0.
V:	1 if arithmetic overflow occurs; otherwise, 0.
D:	The value set by the preceding instruction.
H:	The value set by the preceding instruction.

CP
Compare

Example: Working register R3 contains 16H. Working register R11 contains 20H. The following statement sets the C and S flags to 1, and the Z and V flags are set to 0.

```
CP R3, R1
Op Code: A2 3B
```

Example: Working register R15 contains 16H. Working register R10 contains 20H. Register 20H contains 11H. The following statement sets the C, Z, S, and V flags to 0.

```
CP R16, @R10
Op Code: A3 FA
```

Example: Register 34H contains 2EH. Register 12H contains 1BH. The following statement sets the C, Z, S, and V flags to 0.

```
CP 34H,12H
Op Code: A4 12 34
```

Example: Register 4BH contains 82H. Working register R3 contains 10H. Register 10H contains 01H. The following statement sets the S flag to 1, and the C, Z, and V flags are set to 0.

```
CP 4BH, @R3
Op Code: A5 E3 4B
```

Example: Register 6CH contains 2AH. The following statement sets the Z flag to 1, and the C, S, and V flags are se to 0.

```
CP 6CH, #2AH
Op Code: A6 6C 2A
```

Example: Register D4H contains FCH. Register FCH contains 8FH. The following statement sets the V flag to 1, and the C, Z, and S flags are set to 0.

```
CP @D4H, 7FH
Op Code: A7 D4 FF
```

DA
Decimal Adjust

Instruction Format:

```
DA dst
```

OPC	dst

OPC (Hex)	Address Mode dst
40	R
41	IR

Operation:

```
dst ← DA dst
```

The destination operand is adjusted to two 4-bit BCD digits following a binary addition or subtraction operation on BCD-encoded bytes. For addition (ADD and ADC) or subtraction (SUB and SBC), Table 3-14 indicates the operation performed.

Table 3-16. DA Operation Reference

Prior Instruction	Flags Before DA			Result Before		Adjustment Added	Result After		C Flag After
	C	H	D	[7...4]	[3...0]		[7...4]	[3...0]	
ADD or ADC	0	0	0	0-9	0-9	00	0-9	0-9	0
	0	0	0	0-8	A-F	06	1-9	0-5	0
	0	1	0	1-9	0-3	06	1-9	6-9	0
	0	0	0	A-F	0-9	60	0-5	0-9	1
	1	0	0	0-2	0-9	60	6-8	0-9	1
	0	0	0	9-F	A-F	66	0-5	0-5	1
	0	1	0	A-F	0-3	66	0-5	6-9	1
	1	0	0	0-2	A-F	66	6-9	0-5	1
	1	1	0	0-3	0-3	66	6-9	6-9	1
SUB or SBC	0	0	1	0-9	0-9	00	0-9	0-9	0
	0	1	1	0-8	6-F	FA	0-8	0-9	0
	1	0	1	7-F	0-9	A0	1-9	0-9	1
	1	1	1	6-F	6-F	9A	0-9	0-9	1

Note: If the destination operand is not the result of a valid addition or subtraction of BCD digits, the result is meaningless.

DA
Decimal Adjust

Flags:

When the instruction is executed, the flags are set as follows:

C:	1 if a value is carried or borrowed during the prior addition or subtaction.
Z:	1 if the result is 0; otherwise, 0.
S:	1 if bit 7 of the result is 1 (negative); otherwise, 0.
V:	The value set by the preceding instruction.
D:	The value set by the preceding instruction.
H:	The value set by the preceding instruction.

Example: Addition is performed using the BCD values 15 and 27, the result should be 42. The sum actually obtained is incorrect, however, when the binary representations are added in the destination location using standard binary arithmetic.

```
    0001 0101 = 15H
+   0010 0111 = 27H
-------------------
    0011 1100 = 3CH
```

When the result of the addition is stored in Register 5FH, the following statement adjusts this result so the correct BCD representation is obtained.

```
DA 5FH
Op Code: 41 45
```

```
    0011 1100 = 3CH
+   0000 0110 = 06H
-------------------
    0100 0010 = 42H
```

Register 5F now contains the value 42H. The C, Z, and S flags are set to 0.

DA
Decimal Adjust

Example: A subtraction is performed on BCD values to subtract 17 from 25, the result should be 8. The result is incorrect when standard binary subtraction is performed on the binary representations of the BCD numbers.

```
     0010 0101 = 25H
  +  0001 0111 = 17H
     ______________
     0000 1110 = 0EH
```

Register 45H contains the value 5FH. The result of the subtraction is stored in 5FH. The following statements adjust the result so the correct BCD representation is obtained.

```
DA @45H
Op Code: 40 45
```

```
     0000 1110 = 0EH
  +  1111 1010 = FAH
  _________________
     0000 1000 = 08H
```

Register `5FH` now contains the value `08H`. The `C`, `Z`, and `S` flags are set to 0.

DEC
Decrement

Instruction Format:

```
DEC dst
```

OPC	dst

OPC (Hex)	Address Mode dst
00	R
01	IR

Operation:

```
dst ← dst - 1
```

The contents of the destination operand are decremented by one.

Flags:

When the instruction is executed, the flags are set as follows:

C:	The value set by the preceding instruction.
Z:	1 if the result is 0; otherwise, 0.
S:	1 if bit 7 of the result is 1 (negative); otherwise, 0.
V:	1 if arithmetic overflow occurs; otherwise, 0.
D:	The value set by the preceding instruction.
H:	The value set by the preceding instruction.

Example: Working register R10 contains 2AH. The following statement leaves the value 29H in working register R10. The Z, V, and S flags are set to 0.

```
DEC R10
Op Code: 00 EA
```

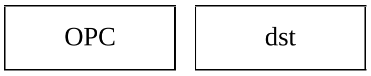
Example: Register B3H contains CBH. Register CBH contains 01H. The following statement leaves the value 00H in Register CBH. The Z flag is set to 1, and the V and S flags are set to 0.

```
DEC @B3H
Op Code: 01 B3
```

DECW
Decrement Word

Instruction Format:

```
DECW dst
```



OPC (Hex)	Address Mode dst
80	RR
81	IR

Operation:

```
dst ← dst - 1
```

The contents of the destination (which must be an even address) operand are decremented by one. The destination operand can be a Register Pair or a working register Pair.

Flags:

When the instruction is executed, the flags are set as follows:

C:	The value set by the preceding instruction.
Z:	1 if the result is 0; otherwise, 0
S:	1 if bit 7 of the result is 1 (negative); otherwise, 0
V:	1 if arithmetic overflow occurs; otherwise, 0
D:	The value set by the preceding instruction.
H:	The value set by the preceding instruction.

Example: Register pair 30H and 31H contain the value 0AF2H. The statement leaves the value 0AF1H in register pair 30H and 31H. The Z, V, and S flags are set to 0.

```
DECW 30H
Op Code: 80 30
```

Example: Working register R0 contains 30H. Register Pair 30H and 31H contain the value FAF3H. The following statement leaves the value FAF2H in Register Pair 30H and 31H. The S flag is set, and the Z and V flags are cleared.

```
DECW @R0
Op Code: 81 E0
```

DI
Disable Interrupts

Instruction Format:

```
DI
```

OPC	**OPC (Hex)**
	8F

Operation:

```
IMASK (7) ← 0
```

Bit `7` of control register `FBH` (the Interrupt Mask Register) is reset to `0`. All interrupts are disabled, although they remain potentially enabled. For example, the Global Interrupt Enable is cleared, but not the individual interrupt level enables.

Flags:

When the instruction is executed, the flags are set as follows:

C:	The value set by the preceding instruction.
Z:	The value set by the preceding instruction.
S:	The value set by the preceding instruction.
V:	The value set by the preceding instruction.
D:	The value set by the preceding instruction.
H:	The value set by the preceding instruction.

Example: Control register `FBH` contains `8AH` (`10001010B`) (interrupts `IRQ1` and `IRQ3` are enabled). The following statement sets control register `FBH` to `0AH` (`00001010B`) and disables all interrupts.

```
DI
Op Code: 8F
```

DJNZ
Decrement And Jump If Non-zero

Instruction Format:

```
DJNZ r, dst
```

r	OPC	dst

OPC (Hex)	Address Mode dst
rA (r=0 to F)	RA

Operation:

```
r ← r - 1;
If r ≠ 0, PC ← PC + dst
```

The specified working register serves as a counter and is decremented. If the contents of the specified working register are not `0` after decrementing, then the relative address is added to the Program Counter (PC) and control passes to the statement whose address is now in the PC. The range of the relative address is `+127` to `–128`. The original value of the PC is the address of the instruction byte following the `DJNZ` statement. When the specified working register counter reaches `0`, control falls through to the statement following the `DJNZ` instruction.

Flags:

When the instruction is executed, the flags are set as follows:

C:	The value set by the preceding instruction.
Z:	The value set by the preceding instruction.
S:	The value set by the preceding instruction.
V:	The value set by the preceding instruction.
D:	The value set by the preceding instruction.
H:	The value set by the preceding instruction.

Example: `DJNZ` is typically used to control a loop of instructions. In this example, 12 bytes are moved from one buffer area in the register file to another. The steps involved are:

1. Load `12` into the counter (working register `R6`).
2. Set up the loop to perform the moves.
3. End the loop with a `DJNZ` instruction.

DJNZ
Decrement And Jump If Non-zero

The assembly listing required for this routine is as follows:

```
         Assembly          Op Code

         LD R6, #12        6E 0C

LOOP:    LD R9 %20(R6)     C7 56 30

         LD %14(R6), R9    D7 56 10

         DJNZ R6, LOOP     6A F8
```

EI
Enable Interrupts

Instruction Format:

```
EI
```

	OPC (Hex)
OPC	9F

Operation:

```
IMASK (7) ← 1
```

Bit `7` of Control Register `FBH` (the Interrupt Mask Register) is set to `1`. This allows potentially enabled interrupts to become enabled.

Flags:

When the instruction is executed, the flags are set as follows:

C:	The value set by the preceding instruction.
Z:	The value set by the preceding instruction.
S:	The value set by the preceding instruction.
V:	The value set by the preceding instruction.
D:	The value set by the preceding instruction.
H:	The value set by the preceding instruction.

Example: Control Register `FBH` contains `0AH` (`00001010`) (interrupts `IRQ1` and `IRQ3` are selected). The following statement sets Control Register `FBH` to `8AH` (`10001010B`) enabling `IRQ1` and `IRQ3`.

```
EI
Op Code: 9F
```

HALT
Halt

Instruction Format:
HALT

OPC	OPC (Hex)
OPC	7F

Operation:

The HALT instruction turns off the internal CPU clock, but not the XTAL oscillation. The peripherals and interrupt logic remain active. Operation can be restarted by an interrupt or a reset.

Flags:

When the instruction is executed, the flags are set as follows

C:	The value set by the preceding instruction.
Z:	The value set by the preceding instruction.
S:	The value set by the preceding instruction.
V:	The value set by the preceding instruction.
D:	The value set by the preceding instruction.
H:	The value set by the preceding instruction.

Example: Assuming the Z8 is in normal operation, the following statements place the Z8 into HALT mode.

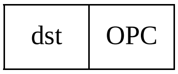
```
HALT
Op Codes: 7F
```

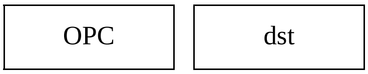
NOTE: Unlike the Z8, the Z8^{PLUS} does not require a NOP before the HALT instruction.

INC
Increment

Instruction Format:

```
INC dst
```





	Address Mode
OPC (Hex)	dst
rE r = 0–15	r
20 21	R IR

Operation:

```
dst ← dst + 1
```

The contents of the destination operand are incremented by one.

Flags:

When the instruction is executed, the flags are set as follows:

C:	The value set by the preceding instruction.
Z:	1 if the result is 0; otherwise, 0.
S:	1 if bit 7 of the result is 1 (negative); otherwise, 0.
V:	1 if arithmetic overflow occurs; otherwise, 0.
D:	The value set by the preceding instruction.
H:	The value set by the preceding instruction.

INC
Increment

Example: Working register R10 contains 2AH. The following statement leaves the value 2BH in working register R10. The Z, V, and S flags are set to 0.

```
INC R10
Op Code: AE
```

Example: Register B3H contains CBH. The following statement leaves the value CCH in register CBH. The S flag is set to 1, and the Z and V flags are set to 0.

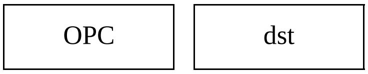
```
INC B3H
Op Code: 20 B3
```

Example: Register B3H contains CBH. Register CBH contains FFH The following statement leaves the value 00H in register CBH. The Z flag is set to 1, and the V and S flags are set to 0.

```
INC @B3H
Op Code: 21 B3
```

INCW
Increment Word

Instruction Format:
`INCW dst`



	Address Mode
OPC (Hex)	dst
A0	RR
A1	IR

Operation:

```
dst ← dst + 1
```

The contents of the destination (which must be an even address) operand is incremented by one. The destination operand can be a Register Pair or a working register Pair.

Flags:

When the instruction is executed, the flags are set as follows:

`C:`	The value set by the preceding instruction.
`Z:`	1 if the result is `0`; otherwise, 0.
`S:`	1 if bit 7 of the result is 1 (negative); otherwise, 0.
`V:`	1 if arithmetic overflow occurs; otherwise, 0.
`D:`	The value set by the preceding instruction.
`H:`	The value set by the preceding instruction.

Example: Register pairs `30H` and `31H` contain the value `0AF2H`. The following statement leaves the value `0AF3H` in register pair `30H` and `31H`. The `Z`, `V`, and `S` flags are set to 0.

```
INCW 30H
Op Code: A0 30
```

Example: Working register `R0` contains `30H`. Register pairs `30H` and `31H` contain the value `FAF3H`. The following statement leaves the value `FAF4H` in register pair `30H` and `31H`. The `S` flag is set, and the `Z` and `V` flags are set to 0.

```
INCW @R0
Op Code: A1 E0
```

IRET
Interrupt Return

Instruction Format:

```
IRET
```

	OPC (Hex)
OPC	BF

Operation:

```
FLAGS ← @SP
SP ← SP + 1
PC ← @SP
SP ← SP + 2
IMR (7) ← 1
```

This instruction is issued at the end of an interrupt service routine. It restores the Flag Register (Control Register FCH) and the PC. It also re-enables any interrupts that are potentially enabled.

Flags:

When the instruction is executed, the flags are set as follows:

C:	The value prior to the issuance of the interrupt.
Z:	The value prior to the issuance of the interrupt.
S:	The value prior to the issuance of the interrupt.
V:	The value prior to the issuance of the interrupt.
D:	The value prior to the issuance of the interrupt.
H:	The value prior to the issuance of the interrupt.

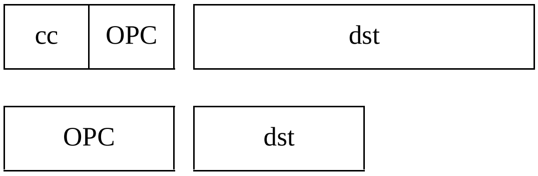
Example: Stack Pointer Low (register FFH) currently contains the value 45H. Register 45H contains the value 00H. Register 46H contains 6FH. Register 47 Contains E4H. The following statement restores the Flags Register (FCH) with the value 00H, restores the PC with the value 6FE4H, re-enables the interrupts, and sets the Stack Pointer Low to 48H. The next instruction to be executed is at location 6FE4H.

```
IRET
Op Code: BF
```

JP
Jump

Instruction Format:

```
JP cc, dst
```



OPC (Hex)	Address Mode dst
ccD (cc = 0 to F)	DA
30	IRR

Operation:

If condition code is true, then `PC` ← `dst`

A conditional jump (JP) transfers program control to the destination address if the condition specified by `cc` is true. Otherwise, the instruction following the `JP` instruction is executed. See page 3-8 for a list of condition codes.

NOTE: Op Code `30H` (`JP IRR`) is ***unconditional*** only.

An unconditional jump simply replaces the contents of the Program Counter with the contents of the register pair specified by the destination operand. Program Control then passes to the instruction addressed by the PC.

Flags:

When the instruction is executed, the flags are set as follows:

C:	The value set by the preceding instruction.
Z:	The value set by the preceding instruction.
S:	The value set by the preceding instruction.
V:	The value set by the preceding instruction.
D:	The value set by the preceding instruction.
H:	The value set by the preceding instruction.

JP
Jump

Example: The Carry flag is 1. The following statement replaces the contents of the Program Counter with `1520H` and transfers program control to that location. If the Carry flag had not been 1, control would have fallen through to the statement following the `JP` instruction.

```
JP C, 1520H
Op Code: 7D 15 20
```

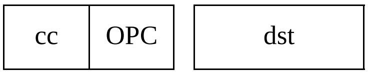
Example:Working register pair `RR2` contains the value `3F45H`. The following statement replaces the contents of the PC with the value `3F45H` and transfers program control to that location.

```
JP @RR2
Op Code: 30 E2
```

JR
Jump Relative

Instruction Format:

```
JR cc, dst
```



OPC (Hex)	Address Mode dst
ccB (cc=0 to F)	RA

Operation:

```
If cc is true, PC ← PC + dst
```

If the condition specified by the `cc` is true, the relative address is added to the PC and control passes to the instruction located at the address specified by the PC (See page 3-8 for a list of condition codes). Otherwise, the instruction following the `JR` instruction is executed. The range of the relative address is `+127` to `-128`, and the original value of the PC is taken to be the address of the first instruction byte following the `JR` instruction.

Flags:

When the instruction is executed, the flags are set as follows:

C:	The value set by the preceding instruction.
Z:	The value set by the preceding instruction.
S:	The value set by the preceding instruction.
V:	The value set by the preceding instruction.
D:	The value set by the preceding instruction.
H:	The value set by the preceding instruction.

Example: The result of the last arithmetic operation executed is negative. The next nine bytes are skipped with the following statement. If the result is not negative, execution continues with the instruction following the `JR` instruction.

```
JR MI, 9
Op Code: 5B 09
```

Example: A short form of a jump `-45` is:

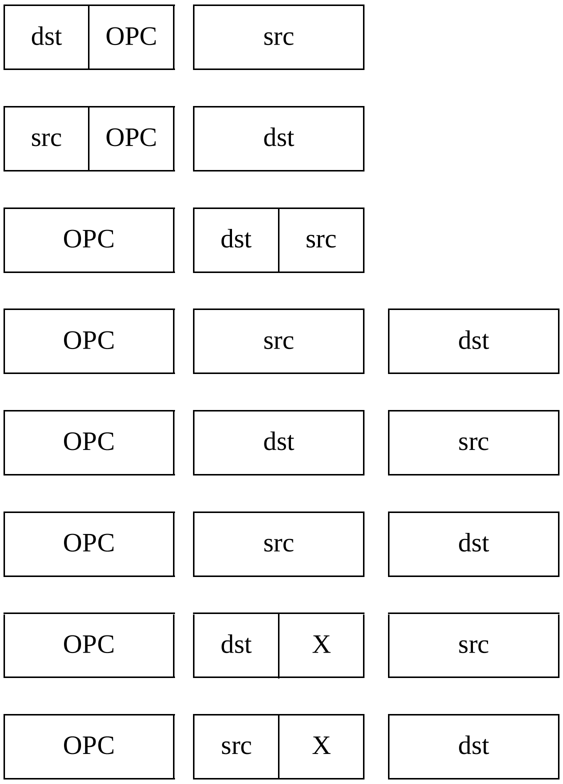
```
JR -45
Op Code: 8B D3
```

The instruction jumps backwards 45 bytes, unconditionally. The condition code is blank in this case, and is assumed to be always true.

LD
Load

Instruction Format:

```
LD dst, src
```



	Address Mode	
OPC (Hex)	**dst**	**src**
rC	r	IM
r8	r	R
r9 r=0 to F	R*	r
E3	r	Ir
F3	Ir	r
E4	R	R
E5	R	IR
E6	R	IM
E7	IR	IM
F5	IR	R
C7	r	X
D7	X	r

*For OPC r9H, only a full 8-bit register can be used. The ZiLOG assember automatically uses the `r8` Op Code for an instruction like:

`LD R0,R1`.

LD
Load

Operation:

```
dst ← src
```

The contents of the source operand are loaded into the destination operand. The contents of the source operand are not changed.

Flags:

When the instruction is executed, the flags are set as follows:

C:	The value set by the preceding instruction.
Z:	The value set by the preceding instruction.
S:	The value set by the preceding instruction.
V:	The value set by the preceding instruction.
D:	The value set by the preceding instruction.
H:	The value set by the preceding instruction.

Example: The following statement loads the value 34H into working register R15.

```
LD R15, #34H
Op Code: FC 34
```

Example: Register 34H contains the value FCH. The following statement loads the value FCH into working register R14. The contents of register 34H are not changed.

```
LD R14, 34H
Op Code: F8 34
```

Example: Working register R14 contains the value 45H. The following statement loads the value 45H into register 34H. The contents of working register R14 are not changed.

```
LD 34H, R14
Op Code: E9 34
```

Example: Working register R12 contains the value 34H. Register 34H contains the value FFH. The following statement loads the value FFH into working register R13. The contents of working register R12 and register 34H are not changed.

```
LD R13, @R12
Op Code: E3 DC
```

LD
Load

Example: Working register R13 contains the value 45H. Working register R12 contains the value 00H. The following statement loads the value 00H into register 45H. The contents of working register R12 and working register R13 are not changed.

```
LD @R13, R12
Op Code: F3 DC
```

Example: Register 45H contains the value CFH. The following statement loads the value CFH into register 34H. The contents of register 45H are not changed.

```
LD 34H, 45H
Op Code: E4 45 34
```

Example: Register 45H contains the value CFH. Register CFH contains the value FFH. The following statement loads the value FFH into register 34H. The contents of register 45H and register CFH are not changed.

```
LD 34H, @45H
Op Code: E5 45 34
```

Example: The following statement loads the value A4H into Register 34H.

```
LD 34H, #0A4H
Op Code: E6 34 A4
```

Example: Working register R14 contains the value 7FH. The following statement loads the value FCH into Register 7FH. The contents of working register R14 are not changed.

```
LD @R14, #0FCH
Op Code: E7 EE FC
```

LD
Load

Example: Register 34H contains the value CFH. Register 45H contains the value FFH. The following statement loads the value FFH into register CFH. The contents of register 34H and register 45H are not changed.

```
LD @34H, 45H
Op Code: F5 45 34
```

Example: Working register R0 contains the value 08H. Register 2CH (24H + 08H = 2CH) contains the value 4FH. The following statement loads working register R10 with the value 4FH. The contents of working register R0 and Register 2CH are not changed.

```
LD R10, 24H(R0)
Op Code: C7 A0 24
```

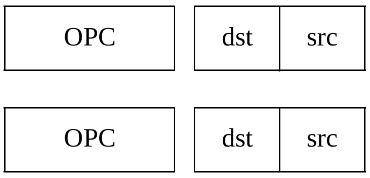
Example: Working register R0 contains the value 0BH. Working register R10 contains 03H. The following statement loads the value 03H into register FBH (F0H + 0BH = FBH). Since this is the Interrupt Mask Register, the LOAD statement has the effect of enabling IRQ0 and IRQ1. The contents of working registers R0 and R10 are unchanged by the load.

```
LD F0H(R0), R10
Op Code: D7 A0 F0
```

LDC
Load Constant

Instruction Format:

```
LDC dst, src
```



OPC (Hex)	Address Mode dst	Address Mode src
C2	r	Irr
D2	Irr	r

Operation:

```
dst ← src
```

This instruction is used to load a byte constant from program memory into a working register, or vice versa. The address of the program memory location is specified by a working register pair. The contents of the source operand are not changed.

Flags

When the instruction is executed, the flags are set as follows:

C:	The value set by the preceding instruction.
Z:	The value set by the preceding instruction.
S:	The value set by the preceding instruction.
V:	The value set by the preceding instruction.
D:	The value set by the preceding instruction.
H:	The value set by the preceding instruction.

Example: Working register pairs `R6` and `R7` contain the value `30A2H` and program memory location `30A2H` contains the value `22H`. The following statement loads the value `22H` into working register `R2`. The value of program memory location `30A2H` is unchanged by the load.

```
LDC R2, @RR6
Op Code: C2 26
```

LDC
Load Constant

Example: Working register `R2` contains the value `22H`. Working register pair `R6` and `R7` contains the value `10A2H`. The following statement loads the value `22H` into program memory location `10A2H`. The value of working register `R2` is unchanged by the load.

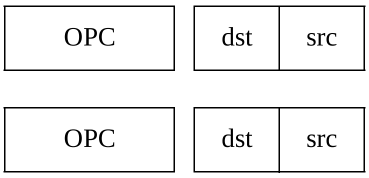
```
LDC @RR6, R2
Op Code: D2 26
```

NOTE: This instruction format is valid only for MCUs which can write to program memory.

LDCI
Load Constant Auto Increment

Instruction Format:

```
LDCI dst, src
```



OPC	Address Mode	
(Hex)	dst	src
C3	Ir	Irr
D3	Irr	Ir

Operation:

```
dst ← src
r ← r + 1
rr ← rr + 1
```

This instruction is used for block transfers of data between program memory and the Register File. The address of the program memory location is specified by a working register Pair, and the address of the Register File location is specified by working register. The contents of the source location are loaded into the destination location. Both addresses in the working registers are then incremented automatically. The contents of the source operand are not changed.

Flags:

When the instruction is executed, the flags are set as follows:

C:	The value set by the preceding instruction.
Z:	The value set by the preceding instruction.
S:	The value set by the preceding instruction.
V:	The value set by the preceding instruction.
D:	The value set by the preceding instruction.
H:	The value set by the preceding instruction.

LDCI
Load Constant Auto-increment

Example: Working register pair R6-R7 contains 30A2H, program memory location 30A2H and 30A3H contain 22H and BCH respectively, and working register R2 contains 20H. The following statement loads the value 22H into Register 20H. working register Pair RR6 is incremented to 30A3H and working register R2 is incremented to 21H.

```
LDCI @R2, @RR6
Op Code: C3 26
```

A second statement loads the value BCH into register 21H. working register pair RR6 is incremented to 30A4H and working register R2 is incremented to 22H.

```
LDCI @R2, @RR6
Op Code: C3 26
```

Example: Working register R2 contains 20H. Register 20H contains 22H. Register 21H contains BCH. Working register pair R6-R7 contains 30A2H. The following statement loads the value 22H into program memory location 30A2H. working register R2 is incremented to 21H and working register Pair R6-R7 is incremented to 30A3H.

```
LDCI @RR6, @R2
Op Code: D3 26
```

A second statement loads the value BCH into program memory location 30A3H. working register R2 is incremented to 22H and working register pair R6-R7 is incremented to 30A4H.

```
LDCI @RR6, @R2
Op Code: D3 26
```

NOP
No Operation

Instruction Format:

NOP

$\boxed{\text{OPC}}$

OPC (Hex)
FF

Operation:

No action is performed by this instruction. It is typically used for timing delays or clearing the pipeline.

Flags:

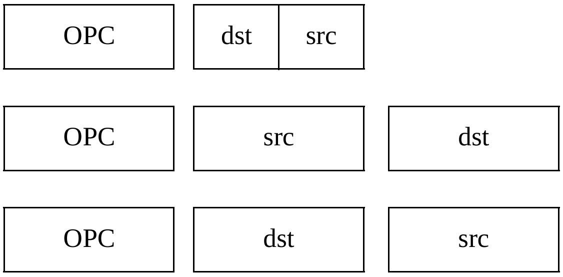
When the instruction is executed, the flags are set as follows:

C:	The value set by the preceding instruction.
Z:	The value set by the preceding instruction.
S:	The value set by the preceding instruction.
V:	The value set by the preceding instruction.
D:	The value set by the preceding instruction.
H:	The value set by the preceding instruction.

OR
Logical OR

Instruction Format:

```
OR dst, src
```



	Address Mode	
OPC (Hex)	**dst**	**src**
42	r	r
43	r	Ir
44	R	R
45	R	IR
46	R	IM
47	IR	IM

Operation:

```
dst ← dst OR src
```

The source operand is logically `OR`ed with the destination operand and the result is stored in the destination operand. The contents of the source operand are not changed. The `OR` operation stores a 1 bit whenever either of the corresponding bits in the two operands is a 1. Otherwise, a `0` bit is stored.

Flags:

When the instruction is executed, the flags are set as follows:

C:	The value set by the preceding instruction.
Z:	1 if the result is `0`; otherwise, 0.
S:	1 if bit 7 of the result is 1; otherwise, 0.
V:	0.
D:	The value set by the preceding instruction.
H:	The value set by the preceding instruction.

OR
Logical OR

Example: Working register R1 contains 34H (00111000B). Working register R14 contains 4DH (10001101). The following statement leaves the value BDH (10111101B) in working register R1. The S flag is set to 1, and the Z and V flags are set to 0.

```
OR R1, R14
Op Code: 42 1E
```

Example: Working register R4 contains F9H (11111001B). Working register R13 contains 7BH. Register 7B contains 6AH (01101010B). The following statement leaves the value FBH (11111011B) in working register R4. The S flag is set to 1, and the Z and V flags are set to 0.

```
OR R4, @R13
Op Code: 43 4D
```

Example: Register 3AH contains the value F5H (11110101B. Register 42H contains the value 0AH (00001010B). The following statement leaves the value FFH (11111111B) in register 3AH. The S flag is setto 1, and the Z and V flags are set to 0.

```
OR 3AH, 42H
Op Code: 44 42 3A
```

Example: Working register R5 contains 70H (01110000B). Register 45H contains 3AH. Register 3AH contains 7FH (01111111B). The following statement leaves the value 7FH (01111111B) in working register R5. The Z, V, and S flags are set to 0.

```
OR R5, @45H
Op Code: 45 45 E5
```

Example: Register 7AH contains the value F3H (11110111B). The following statement leaves the value F3H (11110111B) in register 7AH. The S flag is set to 1, and the Z and V flags are set to 0.

```
OR 7AH, #F0H
Op Code: 46 7A F0
```

Example: Working register R3 contains the value 3EH. Register 3EH contains the value 0CH (00001100B). The following statement leaves the value 0DH (00001101B) in register 3EH. The Z, V, and S flags are set to 0.

```
OR @R3, #05H
Op Code: 57 E3 05
```

POP
Pop

Instruction Format:

```
POP dst
```

OPC	dst

OPC (Hex)	Address Mode dst
50	R
51	IR

Operation:

```
dst ← @SP
SP ← SP + 1
```

The contents of the location specified by the Stack Pointer (SP) are loaded into the destination operand. The SP is then incremented automatically.

Flags:

When the instruction is executed, the flags are set as follows:

C:	The value set by the preceding instruction.
Z:	The value set by the preceding instruction.
S:	The value set by the preceding instruction.
V:	The value set by the preceding instruction.
D:	The value set by the preceding instruction.
H:	The value set by the preceding instruction.

Example: The SP (Control Registers FEH and FFH) contains the value 70H. Register 70H contains 44H. The following statement loads the value 44H into register 34H. After the POP operation, the SP contains 71H. The contents of register 70 are not changed.

```
POP 34H
Op Code: 50 34
```

Example: The SP (Control Registers FEH and FFH) contains the value 1000H. Memory location 1000H contains 55H. Working register R6 contains 22H. The following statement loads the value 55H into register 22H. After the POP operation, the SP contains 1001H. The contents of working register R6 are not changed.

```
POP @R6
Op Code: 51 E6
```

PUSH
Push

Instruction Format:

```
PUSH src
```

OPC	dst

OPC (Hex)	Address Mode dst
70	R
71	IR

Operation:

```
SP ← SP - 1
@SP ← src
```

The contents of the SP (stack pointer) are decremented by one. Then, the contents of the source operand are loaded into the location addressed by the updated SP, adding a new element to the stack.

Flags

:When the instruction is executed, the flags are set as follows:

C:	The value set by the preceding instruction.
Z:	The value set by the preceding instruction.
S:	The value set by the preceding instruction.
V:	The value set by the preceding instruction.
D:	The value set by the preceding instruction.
H:	The value set by the preceding instruction.

Example: The SP contains 1001H. The following statement stores the contents of Register FCH (the Flag Register) in location 1000H. After the PUSH operation, the SP contains 1000H.

```
PUSH FCH
Op Code: 70 FC
```

Example: The SP contains 61H. Working register R4 contains FCH. The following statement stores the contents of register FCH (the Flag Register) in location 60H. After the PUSH operation, the SP contains 60H.

```
PUSH @R4
Op Code: 71 E4
```

RCF
Reset Carry Flag

Instruction Format:

```
RCF
```

$\boxed{\text{OPC}}$

OPC (Hex)
CF

Operation:

`C ← 0`

The `C` flag is reset to `0`, regardless of its previous value.

Flags:

When the instruction is executed, the flags are set as follows:

C:	`0`
Z:	The value set by the preceding instruction.
S:	The value set by the preceding instruction.
V:	The value set by the preceding instruction.
D:	The value set by the preceding instruction.
H:	The value set by the preceding instruction.

Example: The `C` flag is currently set to 1. The following statement resets the Carry flag to `0`.

```
RCF
Op Code: CF
```

RET
Return

Instruction Format:

```
RET
```

OPC	OPC (Hex)
	AF

Operation:

```
PC ← @SP
SP ← SP + 2
```

This instruction is used to return from a procedure entered by a CALL instruction. The contents of the location addressed by the stack pointer (SP) are popped into the Program Control. The next statement executed is the one addressed by the new contents of the PC. The stack pointer is also incremented by 2.

Flags:

When the instruction is executed, the flags are set as follows:

C:	The value set by the preceding instruction.
Z:	The value set by the preceding instruction.
S:	The value set by the preceding instruction.
V:	The value set by the preceding instruction.
D:	The value set by the preceding instruction.
H:	The value set by the preceding instruction.

NOTE: Each `PUSH` instruction executed within the subroutine should be countered with a `POP` instruction in order to guarantee the `SP` is at the correct location when the `RET` instruction is executed. Otherwise the wrong address is loaded into the PC and the program does not operate as desired.

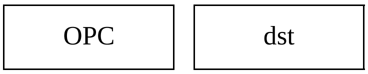
Example: `SP` contains `200H`. Memory location `200H` contains `18H`. Location `201H` contains `B5H`. The following statement leaves the value `202H` in the `SP`, and the PC contains `18B5H`, the address of the next instruction to be executed.

```
RET
Op Code: AF
```

RL
Rotate Left

Instruction Format:

```
RL dst
```

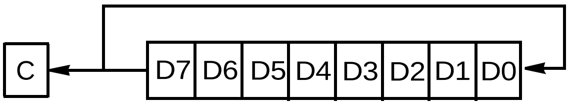


OPC (Hex)	Address Mode dst
90	R
91	IR

Operation:

```
C ← dst(7)
dst(0) ← dst(7)
dst(1) ← dst(0)
dst(2) ← dst(1)
dst(3) ← dst(2)
dst(4) ← dst(3)
dst(5) ← dst(4)
dst(6) ← dst(5)
dst(7) ← dst(6)
```

The contents of the destination operand are rotated left by one bit position. The value from bit `7` is moved to the bit `0` position and also into the Carry flag.



Flags:

When the instruction is executed, the flags are set as follows:

C:	1 if the bit rotated from the most significant bit position was `1` (that is, bit `7` was previously set to `1`).
Z:	1 if the result is `0`; otherwise, 0.
S:	1 if bit `7` of the result is 1; otherwise, 0.
V:	1 if arithmetic overflow occurred (if the sign of the destination operand changed during rotation); otherwise, 0.
D:	The value set by the preceding instruction.
H:	The value set by the preceding instruction.

RL
Rotate Left

Example: The contents of register C6H are 88H (10001000B). The following statement leaves the value 11H (00010001B) in register C6H. The C and V flags are setto 1, and the S and Z flags are set to 0.

```
RL C6H
Op Code: 80 C6
```

Example: The contents of register C6H are 88H. The contents of register 88H are 44H (01000100B). The following statement leaves the value 88H in register 88H (10001000B). The S and V flags are set to 1, and the C and Z flags are set to 0.

```
RL @C6H
Op Code: 81 C6
```

RLC
Rotate Left Through Carry

Instruction Format:

```
RLC dst
```

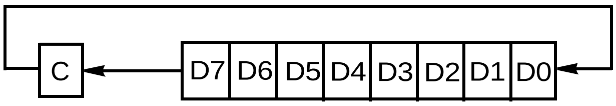
OPC	dst

	Address Mode
OPC (Hex)	dst
10	R
11	IR

Operation:

```
C  ← dst(7)
dst(0) ← C
dst(1) ← dst(0)
dst(2) ← dst(1)
dst(3) ← dst(2)
dst(4) ← dst(3)
dst(5) ← dst(4)
dst(6) ← dst(5)
dst(7) ← dst(6)
```

The contents of the destination operand along with the `C` flag are rotated left by one bit position. The initial value of bit `7` becomes the value of the `C` flag and the previous value of the `C` flag becomes the value of bit `0`.



RLC
Rotate Left Through Carry

Flags:

When the instruction is executed, the flags are set as follows:

C: 1 if the bit rotated from the most significant bit position was 1 (that is, bit 7 was previously set to 1).

Z: 1 if the result is 0; otherwise, 0.

S: 1 if bit 7 of the result is 1; otherwise, 0.

V: 1 if arithmetic overflow occurred (if the sign of the destination operand changed during rotation); otherwise, 0.

D: The value set by the preceding instruction.

H: The value set by the preceding instruction.

Example: The C flag is reset. Register C6 contains 8F (10001111B). The following statement leaves register C6 with the value 1EH (00011110B). The C and V flags are set to 1, and S and Z flags are set to 0.

```
RLC C6
Op Code: 10 C6
```

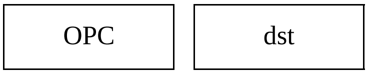
Example: The C flag is reset. Working register R4 contains C6H. Register C6 contains 8F (10001111B). The following statement leaves register C6 with the value 1EH (00011110B). The C and V flags are set to 1, and S and Z flags are set to 0.

```
RLC @R4
Op Code: 11 E4
```

RR
Rotate Right

Instruction Format:

```
RR dst
```

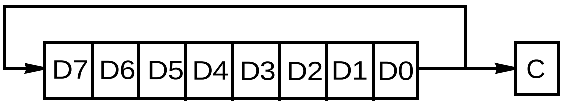


	Address Mode
OPC (Hex)	dst
E0	R
E1	IR

Operation:

```
C ← dst(0)
dst(0) ← dst(1)
dst(1) ← dst(2)
dst(2) ← dst(3)
dst(3) ← dst(4)
dst(4) ← dst(5)
dst(5) ← dst(6)
dst(6) ← dst(7)
dst(7) ← dst(0)
```

The contents of the destination operand are rotated to the right by one bit position. The initial value of bit `0` becomes the value of bit `7` and the `C` flag.



Flags:

When the instruction is executed, the flags are set as follows:

C:	1 if the value rotated from the least significant bit position (bit 1) was `1`.
Z:	1 if the result is `0`; otherwise, 0.
S:	1 if bit `7` of the result is 1; otherwise, 0.
V:	1 if arithmetic overflow occurred (if the sign of the destination operand changed during rotation); otherwise, 0.
D:	The value set by the preceding instruction.
H:	The value set by the preceding instruction.

RR
Rotate Right

Example: The contents of working register R6 are 31H (00110001B). The following statement leaves the value 98H (10011000B) in working register R6. The C, V, and S flags are set to 1, and the Z flag is set to 0.

```
RR R6
Op Code: E0 E6
```

Example: The contents of register C6 are 31H. The contents of register 31H are 7EH (01111110B). The following statement leaves the value 4FH (00111111B) in register 31H. The C, Z, V, and S flags are set to 0.

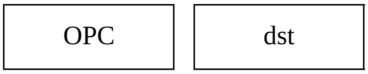
```
RR @C6
Op Code: E1 C6
```

RRC
Rotate Right Through Carry

Instruction Format:

```
RRC dst
```

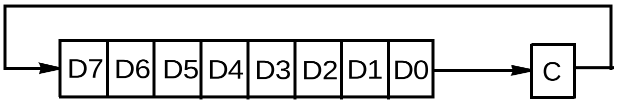
Operation:



OPC (Hex)	Address Mode dst
C0	R
C1	IR

```
C ← dst(0)
dst(0) ← dst(1)
dst(1) ← dst(2)
dst(2) ← dst(3)
dst(3) ← dst(4)
dst(4) ← dst(5)
dst(5) ← dst(6)
dst(6) ← dst(7)
dst(7) ← C
```

The contents of the destination operand with the `C` flag are rotated right by one bit position. The value of the C flag becomes the value of bit 7; the value of bit `0` becomes the value of the `C` flag .



RRC
Rotate Right Through Carry

Flags:

When the instruction is executed, the flags are set as follows:

C:	1 if the bit rotated from the least significant bit position was `1` (that is, bit `0` was `1`).
Z:	1 if the result is `0`; otherwise, 0.
S:	1 if bit `7` of the result is 1; otherwise, 0.
V:	1 if an arithmetic overflow occurs (the sign of the destination operand changed during rotation); otherwise, 0.
D:	The value set by the preceding instruction.
H:	The value set by the preceding instruction.

Example: The contents of register `C6H` are `DDH` (`11011101B`). The `C` flag is 0. The following statement leaves the value `6EH` (`01101110B`) in register `C6H`. The `C` and `V` flags are set to 1, and the `Z` and `S` flags are set to 0.

```
RRC C6H
Op Code: C0 C6
```

Example: The contents of register `2C` are `EDH`. The contents of register `EDH` is `02H` (`00000010B`. The `C` flag is 0. The following statement leaves the value `01H` (`00000001B`) in register `EDH`. The `C`, `Z`, `S`, and `V` flags are reset to 0.

```
RRC @2CH
Op Code: C1 2C
```

SBC
Subtract with Carry

Instruction Format:

```
SBC dst, src
```

		Address Mode	
	OPC (Hex)	dst	src
OPC | dst | src	32	r	r
	33	r	Ir
OPC | src | dst	34	R	R
	35	R	IR
OPC | dst | src	36	R	IM
	37	IR	IM

Operation:

```
dst ← dst - src - C
```

The value of the source operand, and the value of the `C` flag, are subtracted from the destination operand. The result is stored in the destination operand. The contents of the source operand do not change. Subtraction is performed by adding the two's complement of the source operand to the destination operand. In multiple precision arithmetic, this instruction permits the carry (borrow) from the subtraction of low-order operands to be subtracted from the subtraction of high-order operands.

Flags:

When the instruction is executed, the flags are set as follows:

C:	0 if a value is carried from the most significant bit of the result; otherwise, 1 (indicating a borrow).
Z:	1 if the result is `0`; otherwise, 0.
V:	1 if an arithmetic overflow occurs (the operands have opposite signs, and the sign of the result is the same as the sign of the source); otherwise, 0.
S:	1 if bit `7` of the result is 1; otherwise, 0.
H:	0 if if a value is carried from the most significant bit of the low-order four bits of the result; otherwise, 1 (indicating a borrow).
D:	1.

SBC
Subtract with Carry

Example: Working register R3 contains 16H. The C flag is set to 1. Working register R11 contains 20H. The following statement leaves the value F5H in working register R3. The C, S, and D flags are set to 1, and the Z, V, and H flags are set to 0.

```
SBC R3, R11
Op Code: 32 3B
```

Example: Working register R15 contains 16H. The C flag is not set. Working register R10 contains 20H. Register 20H contains 11H. The following statement leaves the value 05H in working register R15. The D flag is set to 1, and the C, Z, S, V, and H flags are set to 0.

```
SBC R16, @R10
Op Code: 33 FA
```

Example: Register 34H contains 2EH. The C flag is set. Register 12H contains 1BH. The following statement leaves the value 12H in register 34H. The D flag is set, and the C, Z, S, V, and H flags are cleared.

```
SBC 34H, 12H
Op Code: 34 12 34
```

Example: Register 4BH contains 82H. The C flag is set. Working register R3 contains 10H. Register 10H contains 01H. The following statement leaves the value 80H in register 4BH. The D and S flags are set to 1, and the C, Z, V, and H flags are set to 0.

```
SBC 4BH, @R3
Op Code: 35 E3 4B
```

Example: Register 6CH contains 2AH. The C flag is not set. The following statement leaves the value 27H in register 6CH. The D flag is set to 1, and the C, Z, S, V, and H flags are set to 0.

```
SBC 6CH, #03H
Op Code: 36 6C 03
```

Example: Register D4H contains 5FH. Register 5FH contains 4CH. The C flag is set. The following statement leaves the value 49H in register 5FH. The D flag is set to 1, and the C, Z, S, V, and H flags are set to 0.

```
SBC @D4H, #02H
Op Code: 37 D4 02
```

SCF
Set Carry Flag

Instruction Format:

```
SRC
```

OPC	**OPC (Hex)**
	DF

Operation:

```
C ← 1
```

The `C` flag is set to 1, regardless of its previous value.

Flags:

When the instruction is executed, the flags are set as follows:

C	1.
Z	The value set by the preceding instruction.
S	The value set by the preceding instruction.
V	The value set by the preceding instruction.
D	The value set by the preceding instruction.
H	The value set by the preceding instruction.

Example: The `C` flag is currently 0. The following statement sets the Carry flag to `1`.

```
SCF
Op Code: DF
```

SRA
Shift Right Arithmetic

Instruction Format:

```
SRA dst
```

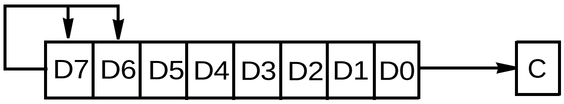
OPC	dst

	Address Mode
OPC (Hex)	**dst**
D0	R
D1	IR

Operation:

```
C ← dst(0)
dst(0) ← dst(1)
dst(1) ← dst(2)
dst(2) ← dst(3)
dst(3) ← dst(4)
dst(4) ← dst(5)
dst(5) ← dst(6)
dst(6) ← dst(7)
dst(7) ← dst(7)
```

An arithmetic right shift by one bit position is performed on the destination operand. Bit 0 replaces the C flag. The value of Bit 7 (the sign bit) is unchanged.Bit 6 becomes the same as the value of bit 7. The result is a signed divide by two holding the half-bit remainder stored in the Carry (C) flag.



Flags:

When the instruction is executed, the flags are set as follows:

C:	1 if the value rotated from the least-significant bit (bit 0) position was 1.
Z:	1 if the result is 0; otherwise, 0.
S:	1 if bit 7 of the result is 1; otherwise, 0.
V:	0.
D:	The value set by the preceding instruction.
H:	The value set by the preceding instruction.

SRA
Shift Right Arithmetic

Example: The contents of working register R6 are 31H (00110001B). The following statement leaves the value 98H (00011000B) in working register R6. The C flag is set to 1, and the Z, V, and S flags are set to 0.

```
SRA R6
Op Code: D0 E6
```

Example: Register C6 contains the value DFH. Register DFH contains the value B8H (10111000B). The following statement leaves the value DCH (11011100B) in Register DFH. The C, Z, and V flags are reset to 0, and the S flag is set to 1.

```
SRA @C6
Op Code: D1 C6
```

SRP
Set Register Pointer

Instruction Format:

```
SRP src
```

OPC	dst

OPC (Hex)	Address Mod dst
31	IM

Operation:

```
RP ← src
```

The specified value is loaded into the Register Pointer (RP) Control Register (FDH). Bits 7-4 determine the working register group. Bits 3-0 selects the Memory Page. Addressing non-existent working register groups and memory pages results in undefined behavior.

Table 3-17. Register Pointers, Working Register Groups, and Actual Registers

Register Pointer (FDH) Contents (Bin)	Working Register Group (Hex)	Actual Registers (Hex)
1111 0000	F	F0-FF
1110 0000	E	E0-EF
1101 0000	D	D0-DF
1100 0000	C	C0-CF
1011 0000	B	B0-BF
1010 0000	A	A0-AF
1001 0000	9	90-9F
1000 0000	8	80-8F
0111 0000	7	70-7F
0110 0000	6	60-6F
0101 0000	5	50-5F
0100 0000	4	40-4F
0011 0000	3	30-3F
0010 0000	2	20-2F
0001 0000	1	10-1F
0000 0000	0	00-0F

SRP
Set Register Pointer

Flags:

When the instruction is executed, the flags are set as follows:

C:	The value set by the preceding instruction.
Z:	The value set by the preceding instruction.
S:	The value set by the preceding instruction.
V:	The value set by the preceding instruction.
D:	The value set by the preceding instruction.
H:	The value set by the preceding instruction.

Example: The following statement `SRP %70` assigns registers `070H` through `07FH` to be the current working register group, and, therefore, accessable as R0 through R15 in four bit addressing modes. The active memory page is set to page 0, and all eight-bit addressed register accesses are on page 0.

```
SRP %70
Op Code: 31 F0
```

STOP
Stop

Instruction Format:

```
STOP
```

OPC	**OPC (Hex)**
	6F

Operation:

This instruction turns off the internal system clock (`SCLK`) and external crystal (`XTAL`) oscillator, and draws only standby current. The `STOP` mode is terminated by a `RESET` or Stop Mode Recovery (`SMR`) which causes the processor to restart the application program at address `0020H`. The waken up source can be determined by reading the `FLAGS` register, specifically the `SMR` and `WDT` flags (see page 3–5 for more information).

Flags:

When the instruction is executed, the flags are set as follows:

C:	The value set by the preceding instruction.
Z:	The value set by the preceding instruction.
S:	The value set by the preceding instruction.
V:	The value set by the preceding instruction.
D:	The value set by the preceding instruction.
H:	The value set by the preceding instruction.

Example: The following statements place the Z8 into `STOP` mode.

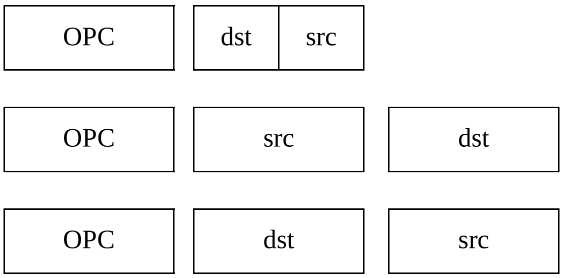
```
STOP
Op Codes: 6F
```

NOTE: Unlike the Z8, the Z8^{PLUS} does not require a `NOP` before the `STOP` instruction.

SUB
Subtract

Instruction Format:

```
SUB dst, src
```



	Address Mode	
OPC (Hex)	**dst**	**src**
22	r	r
23	r	Ir
24	R	R
25	R	IR
26	R	IM
27	IR	IM

Operation:

```
dst ← dst - src
```

The source operand is subtracted from the destination operand and the result is stored in the destination operand. The contents of the source operand are not changed. Subtraction is performed by adding the two's complement of the source operand to the destination operand.

Flags:

When the instruction is executed, the flags are set as follows:

C:	0 if a value is carried from the most significant bit of the result; otherwise, 1, indicating a borrow.
Z:	1 if the result is `0`; otherwise, 0.
V:	1 if arithmetic overflow occurred (if the operands have opposite sign and the sign of the result has the same as the source); reset otherwise.
S:	1 if the result is negative; otherwise, 0.
H:	0 if there is a carry from the most significant bit of the low-order four bits of the result; otherwise, 1, indicating a borrow.
D:	1.

SUB
Subtract

Example: Working register R3 contains 16H. Working register R11 contains 20H. The following statement leaves the value F6H in working register R3. The C, S, and D flags are set to 1, and the Z, V, and H flags are set to 0.

```
SUB R3, R11
Op Code: 22 3B
```

Example: Working register R15 contains 16H. Working register R10 contains 20H. Register 20H contains 11H. The following statement leaves the value 05H in working register R15. The D flag is set to 1, and the C, Z, S, V, and H flags are set to 0.

```
SUB R16, @R10
Op Code: 23 FA
```

Example: Register 34H contains 2EH. Register 12H contains 1BH. The following statement leaves the value 13H in register 34H. The D flag is set to 1, and the C, Z, S, V, and H flags are set to 0.

```
SUB 34H, 12H
Op Code: 24 12 34
```

Example: Register 4BH contains 82H. Working register R3 contains 10H. Register 10H contains 01H. The following statement leaves the value 81H in register 4BH. The D and S flags are set to 1, and the C, Z, V, and H flags are set to 0.

```
SUB 4BH, @R3
Op Code: 25 E3 4B
```

Example: Register 6CH contains 2AH. The following statement leaves the value 27H in register 6CH. The D flag is set to 1, and the C, Z, S, V, and H flags are set to 0.

```
SUB 6CH, #03H
Op Code: 26 6C 03
```

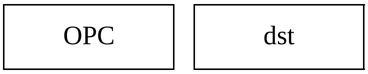
Example: Register D4H contains 5FH. Register 5FH contains 4CH. The following statement leaves the value 4AH in register 5FH. The D flag is set to 1, and the C, Z, S, V, and H flags are set to 0.

```
SUB @D4H, #02H
Op Code: 17 D4 02
```

SWAP
Swap Nibbles

Instruction Format:

```
SWAP dst
```



OPC (Hex)	Address Mode dst
F0	R
F1	IR

Operation:

```
dst(7-4) ↔ dst(3-0)
```

The contents of the lower four bits and upper four bits of the destination operand are swapped.

Flags:

When the instruction is executed, the flags are set as follows:

C:	The value set by the preceding instruction.
Z:	1 if the result is 0; otherwise, 0.
S:	1 if bit 7 of the result is 1; otherwise, 0.
V:	The value set by the preceding instruction.
D:	The value set by the preceding instruction.
H:	The value set by the preceding instruction.

Example: Register BCH contains B3H (10110011B). The following statement leaves the value 3BH (00111011B) in register BCH. The Z and S flags are set to 0.

```
SWAP B3H
Op Code: F0 B3
```

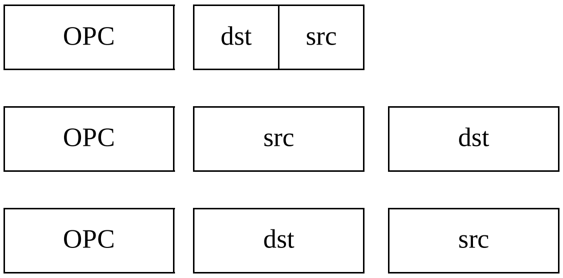
Example: Working register R5 contains BCH and register BCH contains B3H (10110011B). The following statement leaves the value 3BH (00111011B) in register BCH. The Z and S flags are set to 0.

```
SWAP @R5H
Op Code: F1 E5
```

TCM
Test Complement Under Mask

Instruction Format:

```
TCM dst, src
```



	Address Mode	
OPC (Hex)	dst	src
62	r	r
63	r	Ir
64	R	R
65	R	IR
66	R	IM
67	IR	IM

Operation:

```
(NOT dst) AND src
```

This instruction tests selected bits in the destination operand for a logical `1` value. The bits to be tested are specified by setting a `1` bit in the corresponding bit position in the source operand (the mask). The `TCM` instruction complements the destination operand, and then perforoms a logingal AND operation using `AND`s with the mask (source operand). The Zero (`Z`) flag can then be read to check the result. If the `Z` flag is set, then the tested bits were `1`. When the `TCM` operation is complete, the destination and source operands still contain their previous values.

Flags:

When the instruction is executed, the flags are set as follows::

C:	The value set by the preceding instruction.
Z:	1 if the result is `0`; otherwise, 0.
S:	1 if bit `7` of the result is 1; otherwise, 0.
V:	`0`.
D:	The value set by the preceding instruction.
H:	The value set by the preceding instruction.

Example: Working register `R3` contains `45H` (`01000101B`). Working register `R7` contains the value `01H` (`00000001B`) (bit `0` is being tested if it is `1`). The following statement sets the `Z` flag indicating bit `0` in the destination operand is `1`. The `V` and `S` flags are set to 0.

```
TCM R3, R7
Op Code: 62 37
```

TCM
Test Complement Under Mask

Example: Working register R14 contains the value F3H (11110011B). Working register R5 contains CBH. Register CBH contains 88H (10001000B) (bit 7 and bit 3 are tested if they are 1). The following statement resets the Z flag to 0, because bit 3 in the destination operand is not a 1. The V and S flags are also set to 0.

```
TCM R14, @R5
Op Code: 63 E5
```

Example: Register D4H contains the value 04H (000001000B). Working register R0 contains the value 80H (10000000B) (bit 7 istested if it is 1). The following statement resets the Z flag to 0, because bit 7 in the destination operand is not a 1. The S flag is set to 1, and the V flag is set to 0.

```
TCM D4H, R0
Op Code: 64 E0 D4
```

Example: Register DFH contains the value FFH (11111111B). Register 07H contains the value 1FH. Register 1FH contains the value BDH (10111101B) (bit 7, bit 5, bit 4, bit 3, bit 2, and bit 0 are tested if they are 1), The following statement sets the Z flag to 1 indicating the tested bits in the destination operand are 1. The S and V flags are set to 0.

```
TCM DFH, @07H
Op Code: 65 07 DF
```

Example: Working register R13 contains the value F2H (11110010B). The following statement tests bit 1 of the destination operand for 1. The Z flag is set to 1 indicating bit 1 in the destination operand was 1. The S and V flags are set to 0.

```
TCM R13, #02H
Op Code: 66 ED, 02
```

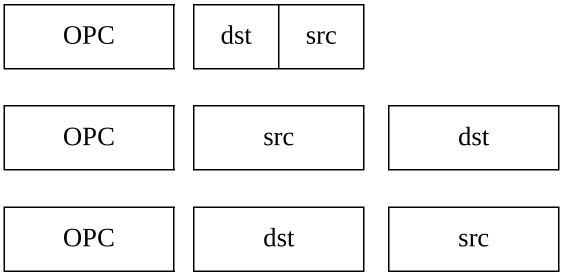
Example: Register 5DH contains A0H. Register A0H contains 0FH (00001111B). The statement tests bit 4 of the Register A0H for 1. The Z flag is reset to 0 indicating bit 1 in the destination operand was not 1. The S and V flags are set to 0.

```
TCM @5D, #10H
Op Code: 67 5D 10
```

TM
Test Under Mask

Instruction Format:

```
TM dst, src
```



	Address Mode	
OPC (Hex)	**dst**	**src**
72	r	r
73	r	Ir
74	R	R
75	R	IR
76	R	IM
77	IR	IM

Operation: `dst AND src`

This instruction tests selected bits in the destination operand for a logical `0` value. The bits to be tested are specified by setting a `1` bit in the corresponding bit position in the source operand (the mask). The `TM` instruction `AND`s the destination operand with the mask (the source operand). The Zero (`Z`) flag can then be read to check the result. If the `Z` flag is set, then the tested bits were `0`. When the `TM` operation is complete, the destination and source operands still contain their previous values.

Flags:

When the instruction is executed, the flags are set as follows:

C:	The value set by the preceding instruction.
Z:	1 if the result is `0`; otherwise, 0.
S:	1 if bit 7 of the result is 1; otherwise, 0.
V:	0.
D:	The value set by the preceding instruction.
H:	The value set by the preceding instruction.

Example: Working register `R3` contains `45H` (`01000101B`. Working register `R7` contains the value `02H` (`00000010B`) (bit `1` is tested if it is `0`). The following statement sets the `Z` flag to 1 indicating bit `1` in the destination operand is `0`. The `V` and `S` flags are set to 0.

```
TM R3, R7
Op Code: 72 37
```

TM
Test Under Mask

Example: Working register R14 contains the value F3H (11110011B). Working register R5 contains CBH. Register CBH contains 88H (10001000B) (bit 7 a bit 3 are tested if they are 0). The following statement resets the Z flag to 0, because bit 7 in the destination operand is not a 0. The S flag is set to 1, and the V flag is set to 0.

```
TM R14, @R5
Op Code: 73 E5
```

Example: Register D4H contains the value 08H (00001000B). Working register R0 contains the value 04H (00000100B) (bit 2 is tested if it is 0). The statement sets the Z flag to 1, because bit 2 in the destination operand is a 0. The S and V flags are set to 0.

```
TM D4H, R0
Op Code: 74 E0 D4
```

Example: Register DFH contains the value 00H (00000000B). Register 07H contains the value 1FH. Register 1FH contains the value BDH (10111101B) (bit 7, bit 5, bit 4, bit 3, bit 2, and bit 0 are tested if they are 0). The following statement sets the Z flag to 1, indicating the tested bits in the destination operand are 0. The S is set to 1, and the V flag is set to 0.

```
TM DFH, @07H
Op Code: 75 07 DF
```

Example: Working register R13 contains the value F1H (11110001B). The following statement tests bit 1 of the destination operand for 0. The Z flag is set to 1, indicating bit 1 in the destination operand was 0. The S and V flags are set to 0.

```
TM R13, #02H
Op Code: 76 ED, 02
```

Example: Register 5DH contains A0H. Register A0H contains 0FH (00001111B). The following statement tests bit 4 of the register A0H for 0. The Z flag is set to 1, indicating bit 4 in the destination operand was 0. The S and V flags are set to 0.

```
TM @5D, #10H
Op Code: 77 5D 10
```

WDT
Watch-Dog Timer

Instruction Format:

```
WDT
```

OPC	**OPC (Hex)**
	5F

Operation:

The Watch-Dog Timer (`WDT`) is a retriggerable one-shot timer that resets the device if it reaches its terminal count. Each execution of the `WDT` instruction refreshes the timer and prevents the `WDT` from timing out.

Flags:

When the instruction is executed, the flags are set as follows:

C:	The value set by the preceding instruction.
Z:	The value set by the preceding instruction.
S:	The value set by the preceding instruction.
V:	The value set by the preceding instruction.
D:	The value set by the preceding instruction.
H:	The value set by the preceding instruction.

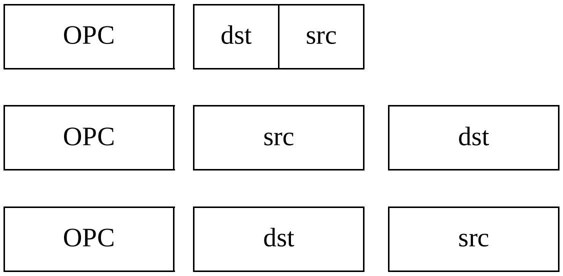
Example: The `WDT` is enabled. The following statement refreshes the Watch-Dog Timer.

```
WDT
Op Code:5F
```

XOR
Logical Exclusive OR

Instruction Format:

```
XOR dst, src
```



OPC (Hex)	Address Mode dst	Address Mode src
B2	r	r
B3	r	Ir
B4	R	R
B5	R	IR
B6	R	IM
B7	IR	IM

Operation:

```
dst ← dst XOR src
```

The source operand performs a logical `EXCLUSIVE OR`ed operation, which stores a `1` in the destination operand whenever the corresponding bits in the two operands are different. The destination operand is set to 1; otherwise, a `0` is stored. The contents of the source operand are not changed.

Flags:

When the instruction is executed, the flags are set as follows:

C:	The value set by the preceding instruction.
Z:	1 if the result is `0`; otherwise, 0.
S:	1 if bit `7` of the result is 1; otherwise, 0.
V:	0.
D:	The value set by the preceding instruction.
H:	The value set by the preceding instruction.

XOR
Logical Exclusive OR

Example: Working register R1 contains 38H (00111000B). Working register R14 contains 8DH (10001101B). The following statement leaves the value B5H (10110101B) in working register R1. The Z, and V flags are set to 0, and the S flag is set to 1.

```
XOR R1, R14
Op Code: B2 1E
```

Example: Working register R4 contains F9H (11111001B). Working register R13 contains 7BH. Register 7B contains 6AH (01101010B). The following statement leaves the value 93H (10010011B) in working register R4. The S flag is set to 1, and the Z and V flags are set to 0.

```
XOR R4, @R13
Op Code: B3 4D
```

Example: Register 3AH contains the value F5H (11110101B). Register 42H contains the value 0AH (00001010B). The following statement leaves the value FFH (11111111B) in register 3AH. The S flag is set to 1, and the C and V flags are set to 0.

```
XOR 3AH, 42H
Op Code: B4 42 3A
```

Example: Working register R5 contains F0H (11110000B). Register 45H contains 3AH. Register 3A contains 7F (01111111B). The statement leaves the value 8FH (10001111B) in working register R5. The S flag is set to 1, and the C and V flags are set to 0.

```
XOR R5, @45H
Op Code: B5 45 E5
```

Example: Register 7AH contains the value F7H (11110111B). The following statement leaves the value 07H (00000111B) in register 7AH. The Z, V, and S flags are set to 0.

```
XOR 7AH, #F0H
Op Code: B6 7A F0
```

Example: Working register R3 contains the value 3EH. Register 3EH contains the value 6CH (01101100B). The following statement leaves the value 69H (01101001B) in register 3EH. The Z, V, and S flags are set to 0.

```
XOR @R3, #05H
Op Code: B7 E3 05
```