8051 Instruction Set

As the 8051 family of Microcontrollers are 8-bit processors, the 8051 Microcontroller Instruction Set is optimized for 8-bit control applications. As a typical 8-bit processor, the 8051 Microcontroller instructions have 8-bit Opcodes. As a result, the 8051 Microcontroller instruction set can have up to 28 = 256 Instructions.

The following table shows the 8051 Instruction Groups and Instructions in each group. There are **49 Instruction Mnemonics** in the 8051 Microcontroller Instruction Set and these 49 Mnemonics are divided into five groups.

DATA TRANSFER	***ARITHMETIC***	***LOGICAL***	***BOOLEAN***	***PROGRAM BRANCHING***
MOV	ADD	ANL	CLR	LJMP
MOVC	ADDC	ORL	SETB	AJMP
MOVX	SUBB	XRL	MOV	SJMP
PUSH	INC	CLR	JC	JZ
POP	DEC	CPL	JNC	JNZ
XCH	MUL	RL	JB	CJNE
XCHD	DIV	RLC	JNB	DJNZ
	DA A	RR	JBC	NOP
		RRC	ANL	LCALL
		SWAP	ORL	ACALL
			CPL	RET
				RETI
				JMP

DATA TRANSFER:

MOV A, Rn - Moves the Rn register to the accumulator
The instruction moves the Rn register to the accumulator. The Rn register is not affected.

Syntax:
MOV A,Rn
Before execution:
R3=58h
After execution:
R3=58h A=58h

MOV A,@Ri - Moves the indirect RAM to the accumulator
Instruction moves the indirectly addressed register of RAM to the accumulator. The register address is stored in the Ri register (R0 or R1). The result is stored in the accumulator. The register is not affected.

Syntax:
MOV A,@Ri
Register Address SUM=F2h R0=F2h
Before execution:
SUM=58h
After execution:
A=58h SUM=58h

MOV A,#data - Moves the immediate data to the accumulator
Instruction moves the immediate data to the accumulator.
Syntax:
MOV A, #28
After execution:
A=28h

MOV direct,@Ri - Moves the indirect RAM to the direct byte
Instruction moves the indirectly adressed register of RAM to the direct byte. The register is not affected.

Syntax:
MOV Rx,@Ri
Register Address SUM=F3
Before execution:
SUM=58h R1=F3
After execution:
SUM=58h TEMP=58h

MOVC A,@A+DPTR - Moves the code byte relative to the DPTR to the accumulator
Instruction first adds the 16-bit DPTR register to the accumulator. The result of addition is then used as a memory address from which the 8-bit data is moved to the accumulator.

Arithmetic:

1. ADD A,Rn;
Adds the register Rn to the accumulator

Description:
Instruction adds the register Rn (R0-R7) to the accumulator. After addition, the result is stored in the accumulator
Before execution:
A=2Eh R4=12h
After execution:
A=40h R4=12h

2. ADD A,@Ri-
Adds the indirect RAM to the accumulator.
Ri: Register R0 or R1

Description:
Instruction adds the indirect RAM to the accumulator. Address of indirect RAM is stored in the Ri register (R0 or R1). After addition, the result is stored in the accumulator. **Register address:**
R0=4Fh
Before execution:
A= 16h SUM= 33h
After execution :
A= 49h

3. ADDA,#DATA
Data: constant within 0-255 (0-FFh)
Description:
Instruction adds data (0-255) to the accumulator. After addition, the result is stored in the accumulator.

ADD A,#33h
Before execution:
A= 16h
After execution:
A= 49h

4. SUBB A,direct-
Subtracts the direct byte from the accumulator witha borrow
Direct: arbitrary register with address 0-255(0-FFh)
Description:
Instruction subtracts the direct byte from the accumulator with a borrow. If the higher bit is subtracted from the lower bit then the carry flag is set. As it is direct addressing, the direct byte can be any SFRs or general-purpose register with address 0-7Fh. (0-127 dec.). The result is stored in the accumulator.

SUBB A,Rx
Before execution:
A=C9h, DIF=53h, C=0
After execution:
A=76h, C=0

5. INC A - Increments the accumulator by1
Description:
This instruction increments the value in the accumulator by 1. If the accumulator includes the number 255, the result of the operation will be 0.
Before execution:
A=E4h
After execution:
A=E5h

6. DEC A - Decrements the accumulator by1
Description: Instruction decrements the value in the accumulator by 1. If there is a 0 in the accumulator, the result of the operation is FFh. (255 dec.)
Syntax: DEC A;
Byte: 1 (instruction code);

STATUS register flags:
No flags are affected;
Before execution:
A=E4h
After execution: A=E3h

7. DIV AB - Divides the accumulator by the register B
Description:
Instruction divides the value in the accumulator by the value in the B register. After division the integer part of result is stored in the accumulator while the register contains the remainder. In case of dividing by 1, the flag OV is set and the result of division is unpredictable. The 8-bit quotient is stored in the accumulator and the 8-bit remainder is stored in the B register.
Before execution:
A=FBh (251dec.) B=12h (18 dec.)
After execution:
A=0Dh (13dec.) B=11h (17dec.)
13·18 + 17 =251

8. DA A - Decimal adjust accumulator
Description: Instruction adjusts the contents of the accumulator to correspond to a BCD number after two BCD numbers have been added by the ADD and ADDC instructions. The result in form of two 4-digit BCD numbers is stored in the accumulator.
Before execution:
A=56h (01010110) 56BCD
B=67h (01100111)67BCD
After execution:
A=BDh (10111101)
After BCD conversion:
A=23h (00100011), C=1 (Overflow)
(C+23=123) = 56+67

9. MUL AB - Multiplies A andB
Description: Instruction multiplies the value in the accumulator with the value in the B register. The low-order byte of the 16-bit result is stored in the

accumulator, while the high byte remains in the B register. If the result is larger than 255, the overflow flag is set. The carry flag is not affected.

Before execution:
A=80 (50h) B=160 (A0h)
After execution:
A=0 B=32h A·B=80·160=12800 (3200h)

Logical:

ANL A,Rn - AND register to the accumulator A: accumulator Rn: any R register(R0-R7)
Instruction performs logic AND operation between the accumulator and Rn register. The result is stored in the accumulator.
Syntax:
ANL A,Rn
Before execution:
A= C3h (11000011 Bin.) R5= 55h (01010101 Bin.)
After execution:
A= 41h (01000001 Bin.)

ORL A,Rn - OR register to the accumulator Rn: any R register (R0-R7)
Instruction performs logic OR operation between the accumulator and Rn register. The result is stored in the accumulator.
Syntax:
ORLA,Rn
Before execution:
A= C3h (11000011 Bin.) R5= 55h (01010101 Bin.)
After execution:
A= D7h (11010111 Bin.)

XRL A,Rn - Exclusive OR register to accumulator Rn: any R register (R0-R7)
Instruction performs exclusive OR operation between the accumulator and the Rn register. The result is stored in the accumulator.
Syntax:
XRL A,Rn
Before execution:

A= C3h (11000011 Bin.) R3= 55h (01010101 Bin.)
After execution:
A= 96h (10010110 Bin.)

CLR A - Clears the accumulator A: accumulator
Instruction clears the accumulator.
Syntax:
CLR A
After execution:
A=0

CPL A - Complements the accumulator
Instruction complements all the bits in the accumulator (1==>0, 0==>1).
Syntax:
CPL A
Before execution:
A=(00110110)
After execution:
A=(11001001)

RL A - Rotates the accumulator one bit left A: accumulator
Eight bits in the accumulator are rotated one bit left, so that the bit 7 is rotated into the bit 0 position.
Syntax:
RL A
Before execution:
A= C2h (11000010 Bin.)
After execution:
A=85h (10000101 Bin.)

RR A - Rotates the accumulator one bit right
A: accumulator All eight bits in the accumulator are rotated one bit right so that the bit 0 is rotated into the bit 7 position.
Syntax:
RR A Before execution:
A= C2h (11000010Bin.)
After execution:
A= 61h (01100001Bin.)

RLC A - Rotates the accumulator one bit left through the carry flag A: accumulator
All eight bits in the accumulator and carry flag are rotated one bit left. After this operation, the bit 7 is rotated into the carry flag position and the carry flag is rotated into the bit 0 position.
Syntax:
RLC A
Before execution:
A= C2h(11000010 Bin.) C=0
After execution:
A= 85h(10000100Bin.) C=1

RRC A - Rotates the accumulator one bit right through the carry flag A: accumulator
All eight bits in the accumulator and carry flag are rotated one bit right. After this operation, the carry flag is rotated into the bit 7 position and the bit 0 is rotated into the carry flag position.
Syntax:
RRC A
Before execution:
A= C2h(11000010 Bin.) C=0
After execution:
A= 61h(01100001Bin.) C=0

SWAP A - Swaps nibbles within the accumulator
A: accumulator
A nibble refers to a group of 4 bits within one register (bit0-bit3 and bit4-bit7). This instruction interchanges high and low nibbles of the accumulator.
Syntax: SWAPA
Before execution:
A=E1h (11100001)bin.
After execution:
A=1Eh (00011110)bin.