8051 Microcontroller Instruction Set

NOVEMBER 23, 2017 BY RAVI — 12 COMMENTS

In the previous tutorial on 8051 Microcontroller, we have seen the Introduction of 8051, the Architecture of 8051 and the Memory Organization of the 8051 Microcontroller. Continuing further, we will take a look at the 8051 Microcontroller Instruction Set and the 8051 Addressing Modes in this tutorial.

Table of Contents

Introduction to 8051 Microcontroller Instruction Set

Writing a Program for any Microcontroller consists of giving commands to the Microcontroller in a particular order in which they must be executed in order to perform a specific task. The commands to the Microcontroller are known as a Microcontroller's Instruction Set.

Just as our sentences are made of words, a Microcontroller's (for that matter, any computer) program is made of Instructions. Instructions written in a program tell the Microcontroller which operation to carry out.

An Instruction Set is unique to a family of computers. This tutorial introduces the 8051 Microcontroller Instruction Set also called as the MCS-51 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 $2^8 = 256$ Instructions.

> Before going into details of 8051 Microcontroller Instruction Set, read this: **8051 MICROCONTROLLER MEMORY ORGANIZATION**.

A Brief Look at 8051 Microcontroller Instructions and Groups

Before going into the details of the 8051 Microcontroller Instruction Set, Types of Instructions and the Addressing Mode, let us take a brief look at the instructions and the instruction groups of the 8051 Microcontroller Instruction Set (the MCS-51 Instruction Set).

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

8051 Addressing Modes

What is an Addressing Mode?

An Addressing Mode is a way to locate a target Data, which is also called as Operand. The 8051 Family of Microcontrollers allows five types of Addressing Modes for addressing the Operands. They are:

- Immediate Addressing
- Register Addressing
- Direct Addressing
- Register – Indirect Addressing
- Indexed Addressing

Immediate Addressing

In Immediate Addressing mode, the operand, which follows the Opcode, is a constant data of either 8 or 16 bits. The name Immediate Addressing came from the fact that the constant data to be stored in the memory immediately follows the Opcode.

The constant value to be stored is specified in the instruction itself rather than taking from a register. The destination register to which the constant data must be copied should be the same size as the operand mentioned in the instruction.

Example: `MOV A, #030H`

Here, the Accumulator is loaded with 30 (hexadecimal). The # in the operand indicates that it is a data and not the address of a Register.

Immediate Addressing is very fast as the data to be loaded is given in the instruction itself.

Register Addressing

In the 8051 Microcontroller Memory Organization Tutorial, we have seen the organization of RAM and four banks of Working Registers with eight Registers in each bank.

In Register Addressing mode, one of the eight registers (R0 – R7) is specified as Operand in the Instruction.

It is important to select the appropriate Bank with the help of PSW Register. Let us see a example of Register Addressing assuming that Bank0 is selected.

Example: `MOV A, R5`

Here, the 8-bit content of the Register R5 of Bank0 is moved to the Accumulator.

Direct Addressing

In Direct Addressing Mode, the address of the data is specified as the Operand in the instruction. Using Direct Addressing Mode, we can access any register or on-chip variable. This includes general purpose RAM, SFRs, I/O Ports, Control registers.

Example: `MOV A, 47H`

Here, the data in the RAM location 47H is moved to the Accumulator.

Register Indirect Addressing

In the Indirect Addressing Mode or Register Indirect Addressing Mode, the address of the Operand is specified as the content of a Register. This will be clearer with an example.

Example: `MOV A, @R1`

The @ symbol indicates that the addressing mode is indirect. If the contents of R1 is 56H, for example, then the operand is in the internal RAM location 56H. If the contents of the RAM location 56H is 24H, then 24H is moved into accumulator.

Only R0 and R1 are allowed in Indirect Addressing Mode. These register in the indirect addressing mode are called as Pointer registers.

Indexed Addressing Mode

With Indexed Addressing Mode, the effective address of the Operand is the sum of a base register and an offset register. The Base Register can be either Data Pointer (DPTR) or Program

Counter (PC) while the Offset register is the Accumulator (A).

In Indexed Addressing Mode, only MOVC and JMP instructions can be used. Indexed Addressing Mode is useful when retrieving data from look-up tables.

Example: MOVC A, @A+DPTR

Here, the address for the operand is the sum of contents of DPTR and Accumulator.

NOTE: Some authors and textbooks add few other Addressing Modes like Absolute Addressing Mode, Relative Addressing Mode and Long Addressing Mode.

> Also read: **8051 MICROCONTROLLER ARCHITECTURE**.

Types of Instructions in 8051 Microcontroller Instruction Set

Before seeing the types of instructions, let us see the structure of the 8051 Microcontroller Instruction. An 8051 Instruction consists of an Opcode (short of Operation – Code) followed by Operand(s) of size Zero Byte, One Byte or Two Bytes.

The Op-Code part of the instruction contains the Mnemonic, which specifies the type of operation to be performed. All Mnemonics or the Opcode part of the instruction are of One Byte size.

Coming to the Operand part of the instruction, it defines the data being processed by the instructions. The operand can be any of the following:

- No Operand
- Data value
- I/O Port
- Memory Location
- CPU register

There can multiple operands and the format of instruction is as follows:

MNEMONIC DESTINATION OPERAND, SOURCE OPERAND

A simple instruction consists of just the opcode. Other instructions may include one or more operands. Instruction can be one-byte instruction, which contains only opcode, or two-byte instructions, where the second byte is the operand or three byte instructions, where the operand makes up the second and third byte.

Based on the operation they perform, all the instructions in the 8051 Microcontroller Instruction Set are divided into five groups. They are:

- Data Transfer Instructions
- Arithmetic Instructions
- Logical Instructions
- Boolean or Bit Manipulation Instructions
- Program Branching Instructions

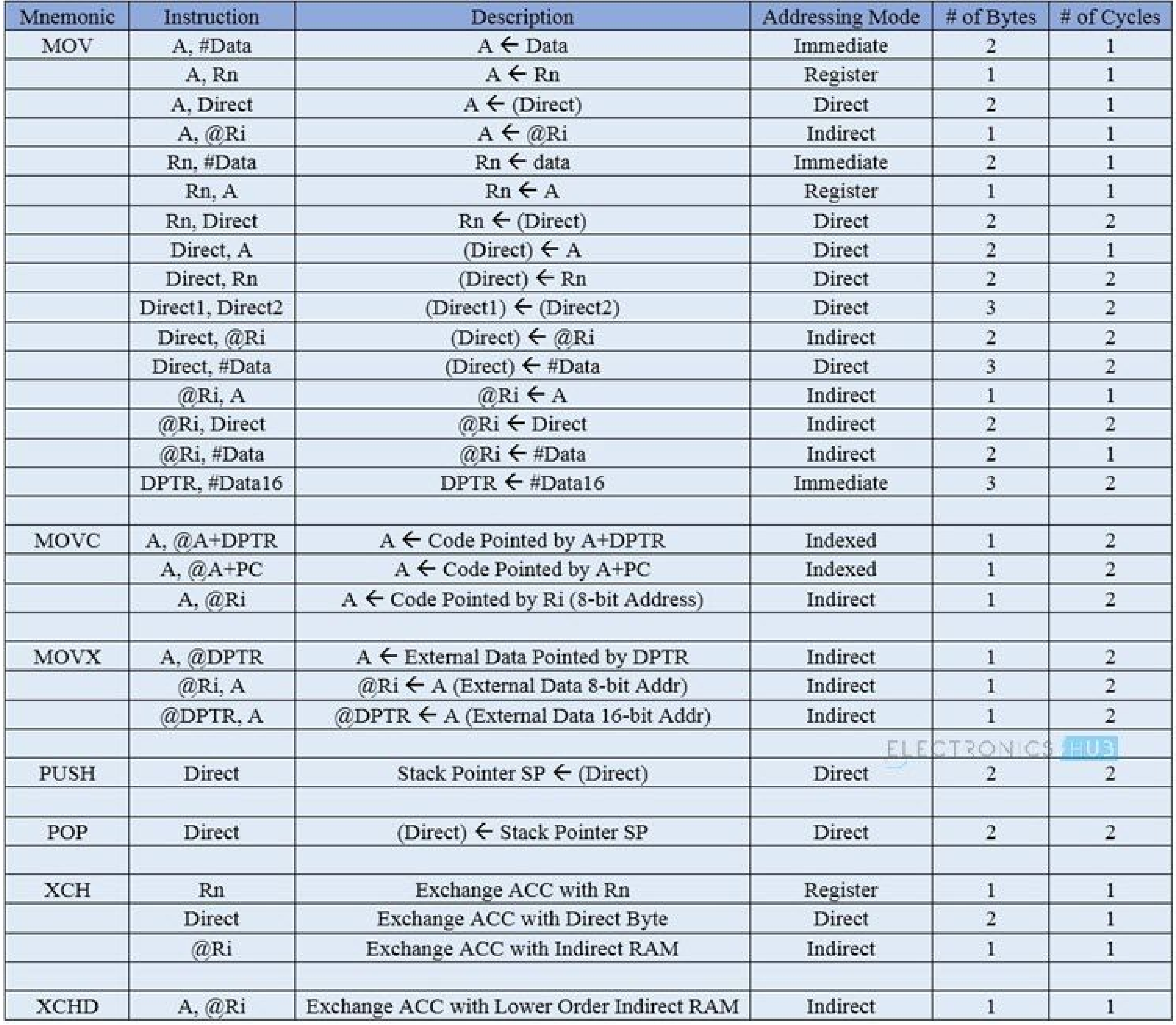
We will now see about these instructions briefly.

Data Transfer Instructions

The Data Transfer Instructions are associated with transfer with data between registers or external program memory or external data memory. The Mnemonics associated with Data Transfer are given below.

- *MOV*
- *MOVC*
- *MOVX*
- *PUSH*
- *POP*
- *XCH*
- *XCHD*

The following table lists out all the possible data transfer instruction along with other details like addressing mode, size occupied and number machine cycles it takes.



Mnemonic	Instruction	Description	Addressing Mode	# of Bytes	# of Cycles
MOV	A, #Data	A ← Data	Immediate	2	1
	A, Rn	A ← Rn	Register	1	1
	A, Direct	A ← (Direct)	Direct	2	1
	A, @Ri	A ← @Ri	Indirect	1	1
	Rn, #Data	Rn ← data	Immediate	2	1
	Rn, A	Rn ← A	Register	1	1
	Rn, Direct	Rn ← (Direct)	Direct	2	2
	Direct, A	(Direct) ← A	Direct	2	1
	Direct, Rn	(Direct) ← Rn	Direct	2	2
	Direct1, Direct2	(Direct1) ← (Direct2)	Direct	3	2
	Direct, @Ri	(Direct) ← @Ri	Indirect	2	2
	Direct, #Data	(Direct) ← #Data	Direct	3	2
	@Ri, A	@Ri ← A	Indirect	1	1
	@Ri, Direct	@Ri ← Direct	Indirect	2	2
	@Ri, #Data	@Ri ← #Data	Indirect	2	1
	DPTR, #Data16	DPTR ← #Data16	Immediate	3	2
MOVC	A, @A+DPTR	A ← Code Pointed by A+DPTR	Indexed	1	2
	A, @A+PC	A ← Code Pointed by A+PC	Indexed	1	2
	A, @Ri	A ← Code Pointed by Ri (8-bit Address)	Indirect	1	2
MOVX	A, @DPTR	A ← External Data Pointed by DPTR	Indirect	1	2
	@Ri, A	@Ri ← A (External Data 8-bit Addr)	Indirect	1	2
	@DPTR, A	@DPTR ← A (External Data 16-bit Addr)	Indirect	1	2
PUSH	Direct	Stack Pointer SP ← (Direct)	Direct	2	2
POP	Direct	(Direct) ← Stack Pointer SP	Direct	2	2
XCH	Rn	Exchange ACC with Rn	Register	1	1
	Direct	Exchange ACC with Direct Byte	Direct	2	1
	@Ri	Exchange ACC with Indirect RAM	Indirect	1	1
XCHD	A, @Ri	Exchange ACC with Lower Order Indirect RAM	Indirect	1	1

Arithmetic Instructions

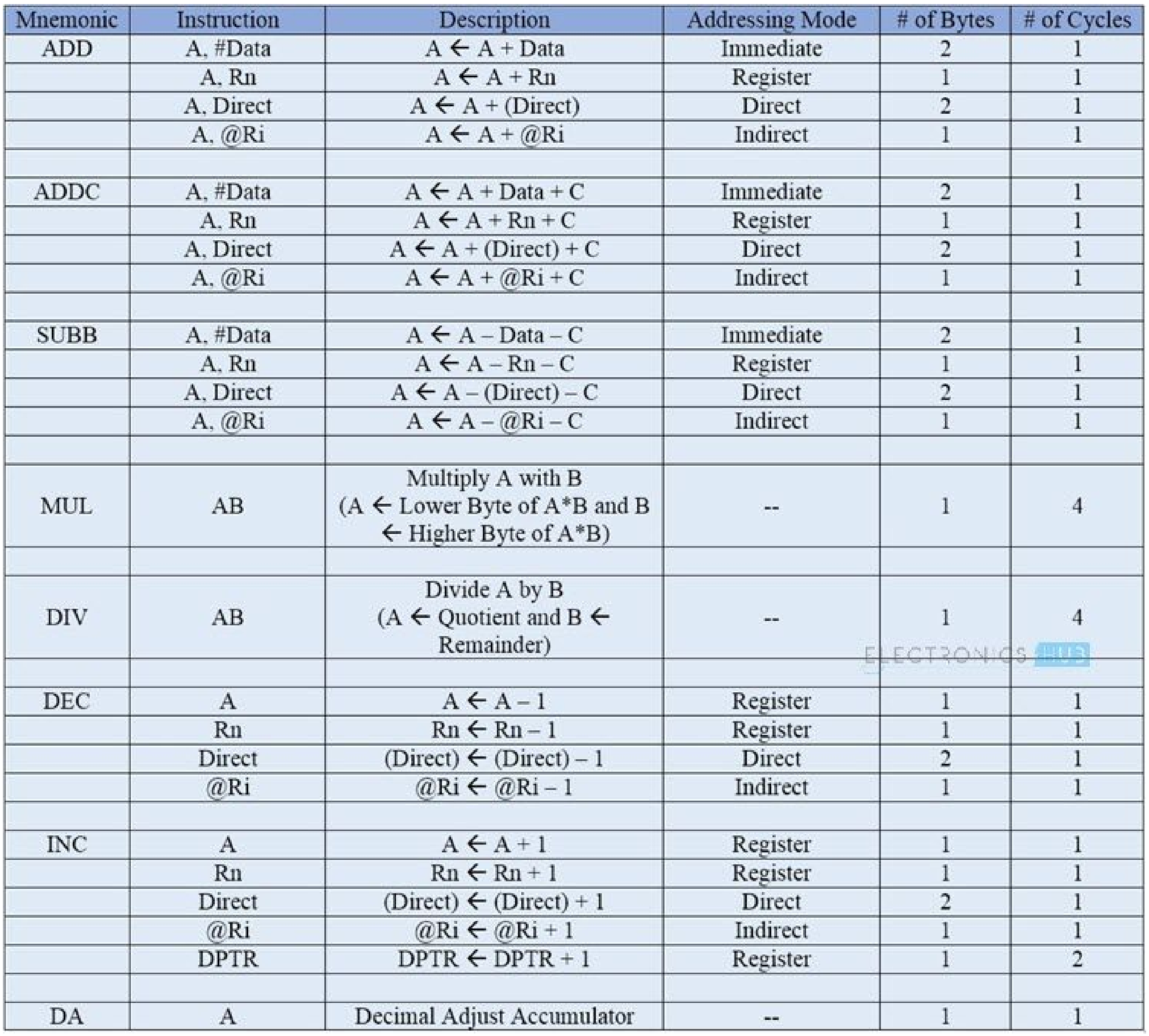
Using Arithmetic Instructions, you can perform addition, subtraction, multiplication and division. The arithmetic instructions also include increment by one, decrement by one and a special instruction called Decimal Adjust Accumulator.

The Mnemonics associated with the Arithmetic Instructions of the 8051 Microcontroller Instruction Set are:

- *ADD*
- *ADDC*
- *SUBB*
- *INC*
- *DEC*
- *MUL*
- *DIV*
- *DA A*

The arithmetic instructions has no knowledge about the data format i.e. signed, unsigned, ASCII, BCD, etc. Also, the operations performed by the arithmetic instructions affect flags like carry, overflow, zero, etc. in the PSW Register.

All the possible Mnemonics associated with Arithmetic Instructions are mentioned in the following table.



Mnemonic	Instruction	Description	Addressing Mode	# of Bytes	# of Cycles
ADD	A, #Data	A ← A + Data	Immediate	2	1
	A, Rn	A ← A + Rn	Register	1	1
	A, Direct	A ← A + (Direct)	Direct	2	1
	A, @Ri	A ← A + @Ri	Indirect	1	1
ADDC	A, #Data	A ← A + Data + C	Immediate	2	1
	A, Rn	A ← A + Rn + C	Register	1	1
	A, Direct	A ← A + (Direct) + C	Direct	2	1
	A, @Ri	A ← A + @Ri + C	Indirect	1	1
SUBB	A, #Data	A ← A – Data – C	Immediate	2	1
	A, Rn	A ← A – Rn – C	Register	1	1
	A, Direct	A ← A – (Direct) – C	Direct	2	1
	A, @Ri	A ← A – @Ri – C	Indirect	1	1
MUL	AB	Multiply A with B (A ← Lower Byte of A*B and B ← Higher Byte of A*B)	--	1	4
DIV	AB	Divide A by B (A ← Quotient and B ← Remainder)	--	1	4
DEC	A	A ← A – 1	Register	1	1
	Rn	Rn ← Rn – 1	Register	1	1
	Direct	(Direct) ← (Direct) – 1	Direct	2	1
	@Ri	@Ri ← @Ri – 1	Indirect	1	1
INC	A	A ← A + 1	Register	1	1
	Rn	Rn ← Rn + 1	Register	1	1
	Direct	(Direct) ← (Direct) + 1	Direct	2	1
	@Ri	@Ri ← @Ri + 1	Indirect	1	1
	DPTR	DPTR ← DPTR + 1	Register	1	2
DA	A	Decimal Adjust Accumulator	--	1	1

Logical Instructions

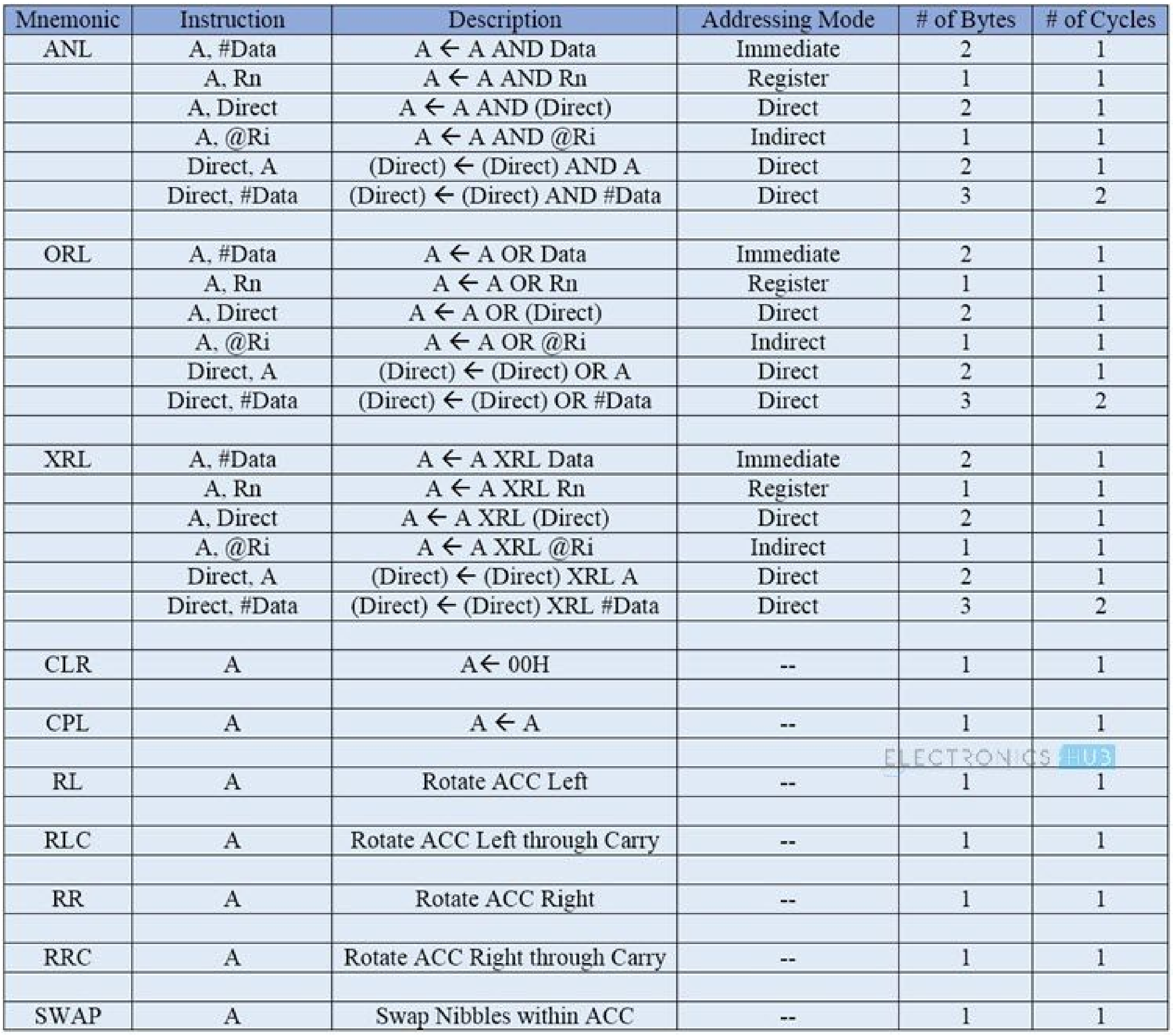
The next group of instructions are the Logical Instructions, which perform logical operations like AND, OR, XOR, NOT, Rotate, Clear and Swap. Logical Instruction are performed on Bytes of data on a bit-by-bit basis.

Mnemonics associated with Logical Instructions are as follows:

- *ANL*
- *ORL*
- *XRL*

- *CLR*
- *CPL*
- *RL*
- *RLC*
- *RR*
- *RRC*
- *SWAP*

The following table shows all the possible Mnemonics of the Logical Instructions.



Mnemonic	Instruction	Description	Addressing Mode	# of Bytes	# of Cycles
ANL	A, #Data	A ← A AND Data	Immediate	2	1
	A, Rn	A ← A AND Rn	Register	1	1
	A, Direct	A ← A AND (Direct)	Direct	2	1
	A, @Ri	A ← A AND @Ri	Indirect	1	1
	Direct, A	(Direct) ← (Direct) AND A	Direct	2	1
	Direct, #Data	(Direct) ← (Direct) AND #Data	Direct	3	2
ORL	A, #Data	A ← A OR Data	Immediate	2	1
	A, Rn	A ← A OR Rn	Register	1	1
	A, Direct	A ← A OR (Direct)	Direct	2	1
	A, @Ri	A ← A OR @Ri	Indirect	1	1
	Direct, A	(Direct) ← (Direct) OR A	Direct	2	1
	Direct, #Data	(Direct) ← (Direct) OR #Data	Direct	3	2
XRL	A, #Data	A ← A XRL Data	Immediate	2	1
	A, Rn	A ← A XRL Rn	Register	1	1
	A, Direct	A ← A XRL (Direct)	Direct	2	1
	A, @Ri	A ← A XRL @Ri	Indirect	1	1
	Direct, A	(Direct) ← (Direct) XRL A	Direct	2	1
	Direct, #Data	(Direct) ← (Direct) XRL #Data	Direct	3	2
CLR	A	A← 00H	--	1	1
CPL	A	A ← A	--	1	1
RL	A	Rotate ACC Left	--	1	1
RLC	A	Rotate ACC Left through Carry	--	1	1
RR	A	Rotate ACC Right	--	1	1
RRC	A	Rotate ACC Right through Carry	--	1	1
SWAP	A	Swap Nibbles within ACC	--	1	1

Boolean or Bit Manipulation Instructions

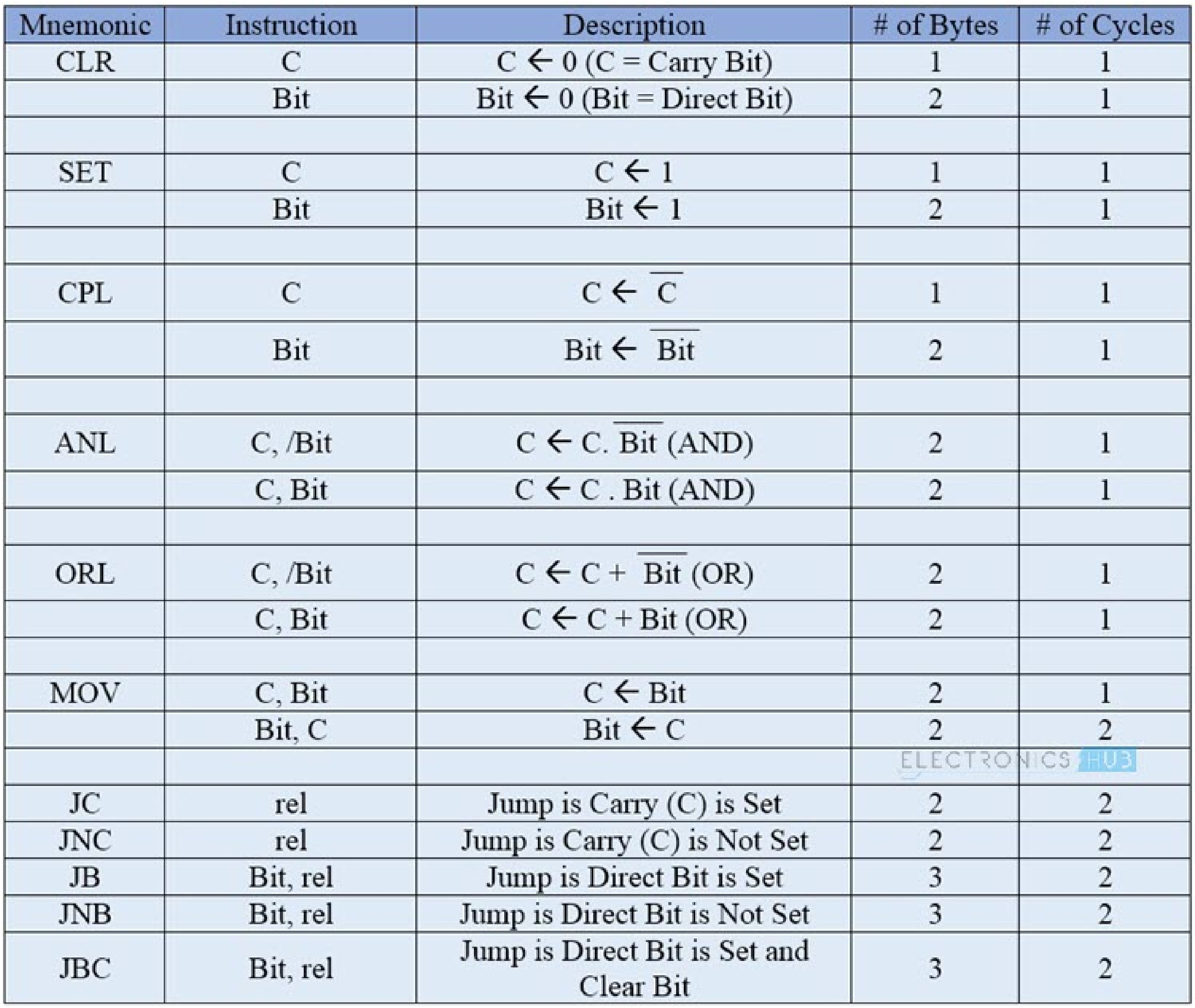
As the name suggests, Boolean or Bit Manipulation Instructions will deal with bit variables. We know that there is a special bit-addressable area in the RAM and some of the Special Function Registers (SFRs) are also bit addressable.

The Mnemonics corresponding to the Boolean or Bit Manipulation instructions are:

- *CLR*

- *SETB*
- *MOV*
- *JC*
- *JNC*
- *JB*
- *JNB*
- *JBC*
- *ANL*
- *ORL*
- *CPL*

These instructions can perform set, clear, and, or, complement etc. at bit level. All the possible mnemonics of the Boolean Instructions are specified in the following table.



Mnemonic	Instruction	Description	# of Bytes	# of Cycles
CLR	C	C ← 0 (C = Carry Bit)	1	1
	Bit	Bit ← 0 (Bit = Direct Bit)	2	1
SET	C	C ← 1	1	1
	Bit	Bit ← 1	2	1
CPL	C	$C \leftarrow \overline{C}$	1	1
	Bit	$Bit \leftarrow \overline{Bit}$	2	1
ANL	C, /Bit	$C \leftarrow C.\ \overline{Bit}$ (AND)	2	1
	C, Bit	C ← C . Bit (AND)	2	1
ORL	C, /Bit	$C \leftarrow C + \overline{Bit}$ (OR)	2	1
	C, Bit	C ← C + Bit (OR)	2	1
MOV	C, Bit	C ← Bit	2	1
	Bit, C	Bit ← C	2	2
JC	rel	Jump is Carry (C) is Set	2	2
JNC	rel	Jump is Carry (C) is Not Set	2	2
JB	Bit, rel	Jump is Direct Bit is Set	3	2
JNB	Bit, rel	Jump is Direct Bit is Not Set	3	2
JBC	Bit, rel	Jump is Direct Bit is Set and Clear Bit	3	2

Program Branching Instructions

The last group of instructions in the 8051 Microcontroller Instruction Set are the Program Branching Instructions. These instructions control the flow of program logic. The mnemonics of the Program Branching Instructions are as follows.

- *LJMP*
- *AJMP*
- *SJMP*
- *JZ*
- *JNZ*
- *CJNE*
- *DJNZ*
- *NOP*
- *LCALL*
- *ACALL*
- *RET*
- *RETI*
- *JMP*

All these instructions, except the NOP (No Operation) affect the Program Counter (PC) in one way or other. Some of these instructions has decision making capability before transferring control to other part of the program.

The following table shows all the mnemonics with respect to the program branching instructions.