

Architecture & Instruction set of 8085 microprocessor

8085 is pronounced as "eighty-eighty-five" microprocessor. It is an 8-bit microprocessor designed by Intel in 1977 using NMOS technology.

It has the following configuration –

- 8-bit data bus
- 16-bit address bus, which can address upto 64KB
- A 16-bit program counter
- A 16-bit stack pointer
- Six 8-bit registers arranged in pairs: BC, DE, HL
- Requires +5V supply to operate at 3.2 MHZ single phase clock

It is used in washing machines, microwave ovens, mobile phones, etc.

8085 Microprocessor – Functional Units

8085 consists of the following functional units –

Accumulator

It is an 8-bit register used to perform arithmetic, logical, I/O & LOAD/STORE operations. It is connected to internal data bus & ALU.

Arithmetic and logic unit

As the name suggests, it performs arithmetic and logical operations like Addition, Subtraction, AND, OR, etc. on 8-bit data.

General purpose register

There are 6 general purpose registers in 8085 processor, i.e. B, C, D, E, H & L. Each register can hold 8-bit data.

These registers can work in pair to hold 16-bit data and their pairing combination is like B-C, D-E & H-L.

Program counter

It is a 16-bit register used to store the memory address location of the next instruction to be executed. Microprocessor increments the program whenever an instruction is being executed, so that the program counter points to the memory address of the next instruction that is going to be executed.

Stack pointer

It is also a 16-bit register works like stack, which is always incremented/decremented by 2 during push & pop operations.

Temporary register

It is an 8-bit register, which holds the temporary data of arithmetic and logical operations.

Flag register

It is an 8-bit register having five 1-bit flip-flops, which holds either 0 or 1 depending upon the result stored in the accumulator.

These are the set of 5 flip-flops –

- Sign (S)
- Zero (Z)
- Auxiliary Carry (AC)
- Parity (P)
- Carry (C)

Its bit position is shown in the following table –

D7	D6	D5	D4	D3	D2	D1	D0
S	Z		AC		P		CY

Instruction register and decoder

It is an 8-bit register. When an instruction is fetched from memory then it is stored in the Instruction register. Instruction decoder decodes the information present in the Instruction register.

Timing and control unit

It provides timing and control signal to the microprocessor to perform operations. Following are the timing and control signals, which control external and internal circuits –

- Control Signals: READY, RD', WR', ALE
- Status Signals: S0, S1, IO/M'
- DMA Signals: HOLD, HLDA
- RESET Signals: RESET IN, RESET OUT

Interrupt control

As the name suggests it controls the interrupts during a process. When a microprocessor is executing a main program and whenever an interrupt occurs, the microprocessor shifts the control from the main program to process the incoming request. After the request is completed, the control goes back to the main program.

There are 5 interrupt signals in 8085 microprocessor: INTR, RST 7.5, RST 6.5, RST 5.5, TRAP.

Serial Input/output control

It controls the serial data communication by using these two instructions: SID (Serial input data) and SOD (Serial output data).

Address buffer and address-data buffer

The content stored in the stack pointer and program counter is loaded into the address buffer and address-data buffer to communicate with the CPU. The memory and I/O chips are connected to these buses; the CPU can exchange the desired data with the memory and I/O chips.

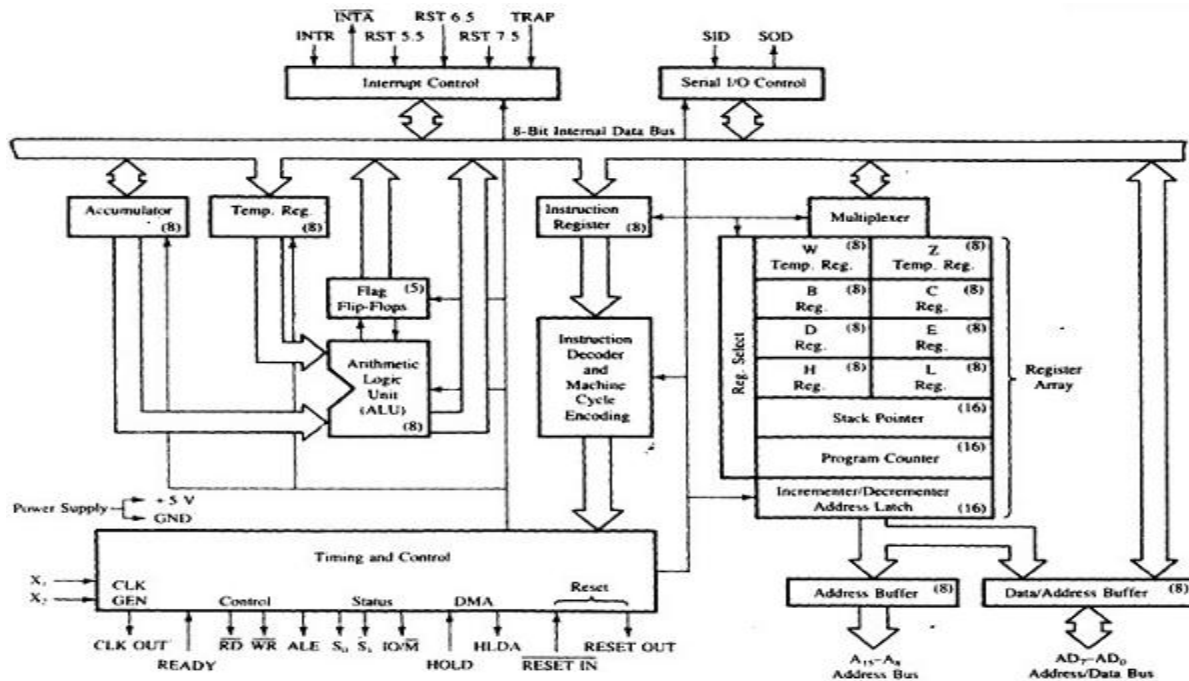
Address bus and data bus

Data bus carries the data to be stored. It is bidirectional, whereas address bus carries the location to where it should be stored and it is unidirectional. It is used to transfer the data & Address I/O devices.

8085 Architecture

We have tried to depict the architecture of 8085 with this following image –

Architecture & Instruction set of 8085 Microprocessor and 8051 Micro Controller



Let us take a look at the programming of 8085 Microprocessor.

Instruction sets are instruction codes to perform some task. It is classified into five categories.

S.No.	Instruction & Description
1	<p><u>Control Instructions</u></p> <p>Following is the table showing the list of Control instructions with their meanings.</p>

2	<u>Logical Instructions</u> Following is the table showing the list of Logical instructions with their meanings.
3	<u>Branching Instructions</u> Following is the table showing the list of Branching instructions with their meanings.
4	<u>Arithmetic Instructions</u> Following is the table showing the list of Arithmetic instructions with their meanings.
5	<u>Data Transfer Instructions</u> Following is the table showing the list of Data-transfer instructions with their meanings.

8085–Demo Programs

Now, let us take a look at some program demonstrations using the above instructions –

Adding Two 8-bit Numbers

Write a program to add data at 3005H & 3006H memory location and store the result at 3007H memory location.

Problem demo –

(3005H) = 14H

(3006H) = 89H

Result –

$14H + 89H = 9DH$

The program code can be written like this –

LXI H 3005H : "HL points 3005H"

MOV A, M : "Getting first operand"

INX H : "HL points 3006H"

ADD M : "Add second operand"

INX H : "HL points 3007H"

MOV M, A : "Store result at 3007H"

HLT : "Exit program"

Exchanging the Memory Locations

Write a program to exchange the data at 5000M& 6000M memory location.

LDA 5000M : "Getting the contents at 5000M location into accumulator"

MOV B, A : "Save the contents into B register"

LDA 6000M : "Getting the contents at 6000M location into accumulator"

STA 5000M : "Store the contents of accumulator at address 5000M"

MOV A, B : "Get the saved contents back into A register"

STA 6000M : "Store the contents of accumulator at address 6000M"

Arrange Numbers in an Ascending Order

Write a program to arrange first 10 numbers from memory address 3000H in an ascending order.

MVI B, 09 : "Initialize counter"

START : "LXI H, 3000H: Initialize memory pointer"

MVI C, 09H : "Initialize counter 2"

BACK: MOV A, M : "Get the number"

INX H : "Increment memory pointer"

CMP M : "Compare number with next number"

JC SKIP : "If less, don't interchange"

JZ SKIP : "If equal, don't interchange"
MOV D, M
MOV M, A
DCX H
MOV M, D
INX H : "Interchange two numbers"
SKIP:DCR C : "Decrement counter 2"
JNZ BACK : "If not zero, repeat"
DCR B : "Decrement counter 1"
JNZ START
HLT : "Terminate program execution"

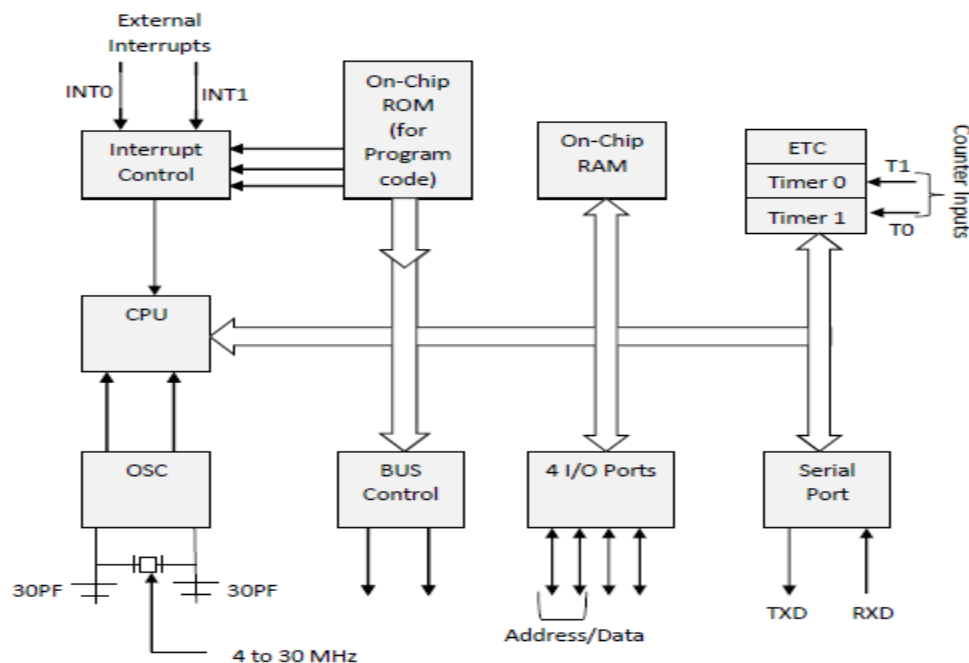
Architecture & Instruction set of 8051 microcontroller

8051 microcontroller is designed by Intel in 1981. It is an 8-bit microcontroller. It is built with 40 pins DIP (dual inline package), 4kb of ROM storage and 128 bytes of RAM storage, 2 16-bit timers. It consists of are four parallel 8-bit ports, which are programmable as well as addressable as per the requirement. An on-chip crystal oscillator is integrated in the microcontroller having crystal frequency of 12 MHz.

Architecture & Instruction set of 8085 Microprocessor and 8051 Micro Controller

Let us now discuss the architecture of 8051 Microcontroller.

In the following diagram, the system bus connects all the support devices to the CPU. The system bus consists of an 8-bit data bus, a 16-bit address bus and bus control signals. All other devices like program memory, ports, data memory, serial interface, interrupt control, timers, and the CPU are all interfaced together through the system bus.



8051 Microcontroller Instruction Set

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.

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.

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.

Architecture & Instruction set of 8085 Microprocessor and 8051 Micro Controller

<i>DATA TRANSFER</i>	<i>ARITHMETIC</i>	<i>LOGICAL</i>	<i>BOOLEAN</i>	<i>PROGRAM BRANCHING</i>
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

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 an 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 registers 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.

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:

Architecture & Instruction set of 8085 Microprocessor and 8051 Micro Controller

- 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.

Architecture & Instruction set of 8085 Microprocessor and 8051 Micro Controller

Mnemonic	Instruction	Description	Addressing Mode	# of Bytes	# of Cycles
MOV	A, #Data	$A \leftarrow \text{Data}$	Immediate	2	1
	A, Rn	$A \leftarrow Rn$	Register	1	1
	A, Direct	$A \leftarrow (\text{Direct})$	Direct	2	1
	A, @Ri	$A \leftarrow @Ri$	Indirect	1	1
	Rn, #Data	$Rn \leftarrow \text{data}$	Immediate	2	1
	Rn, A	$Rn \leftarrow A$	Register	1	1
	Rn, Direct	$Rn \leftarrow (\text{Direct})$	Direct	2	2
	Direct, A	$(\text{Direct}) \leftarrow A$	Direct	2	1
	Direct, Rn	$(\text{Direct}) \leftarrow Rn$	Direct	2	2
	Direct1, Direct2	$(\text{Direct1}) \leftarrow (\text{Direct2})$	Direct	3	2
	Direct, @Ri	$(\text{Direct}) \leftarrow @Ri$	Indirect	2	2
	Direct, #Data	$(\text{Direct}) \leftarrow \#Data$	Direct	3	2
	@Ri, A	$@Ri \leftarrow A$	Indirect	1	1
	@Ri, Direct	$@Ri \leftarrow \text{Direct}$	Indirect	2	2
	@Ri, #Data	$@Ri \leftarrow \#Data$	Indirect	2	1
	DPTR, #Data16	$DPTR \leftarrow \#Data16$	Immediate	3	2
MOVC	A, @A+DPTR	$A \leftarrow \text{Code Pointed by } A+DPTR$	Indexed	1	2
	A, @A+PC	$A \leftarrow \text{Code Pointed by } A+PC$	Indexed	1	2
	A, @Ri	$A \leftarrow \text{Code Pointed by } Ri \text{ (8-bit Address)}$	Indirect	1	2
MOVX	A, @DPTR	$A \leftarrow \text{External Data Pointed by } DPTR$	Indirect	1	2
	@Ri, A	$@Ri \leftarrow A \text{ (External Data 8-bit Addr)}$	Indirect	1	2
	@DPTR, A	$@DPTR \leftarrow A \text{ (External Data 16-bit Addr)}$	Indirect	1	2
PUSH	Direct	Stack Pointer $SP \leftarrow (\text{Direct})$	Direct	2	2
POP	Direct	$(\text{Direct}) \leftarrow \text{Stack Pointer } SP$	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

Architecture & Instruction set of 8085 Microprocessor and 8051 Micro Controller

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 have 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.

Architecture & Instruction set of 8085 Microprocessor and 8051 Micro Controller

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

Architecture & Instruction set of 8085 Microprocessor and 8051 Micro Controller

- *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	$A \leftarrow A \text{ AND Data}$	Immediate	2	1
	A, Rn	$A \leftarrow A \text{ AND Rn}$	Register	1	1
	A, Direct	$A \leftarrow A \text{ AND (Direct)}$	Direct	2	1
	A, @Ri	$A \leftarrow A \text{ AND @Ri}$	Indirect	1	1
	Direct, A	$(\text{Direct}) \leftarrow (\text{Direct}) \text{ AND A}$	Direct	2	1
	Direct, #Data	$(\text{Direct}) \leftarrow (\text{Direct}) \text{ AND \#Data}$	Direct	3	2
ORL	A, #Data	$A \leftarrow A \text{ OR Data}$	Immediate	2	1
	A, Rn	$A \leftarrow A \text{ OR Rn}$	Register	1	1
	A, Direct	$A \leftarrow A \text{ OR (Direct)}$	Direct	2	1
	A, @Ri	$A \leftarrow A \text{ OR @Ri}$	Indirect	1	1
	Direct, A	$(\text{Direct}) \leftarrow (\text{Direct}) \text{ OR A}$	Direct	2	1
	Direct, #Data	$(\text{Direct}) \leftarrow (\text{Direct}) \text{ OR \#Data}$	Direct	3	2
XRL	A, #Data	$A \leftarrow A \text{ XRL Data}$	Immediate	2	1
	A, Rn	$A \leftarrow A \text{ XRL Rn}$	Register	1	1
	A, Direct	$A \leftarrow A \text{ XRL (Direct)}$	Direct	2	1
	A, @Ri	$A \leftarrow A \text{ XRL @Ri}$	Indirect	1	1
	Direct, A	$(\text{Direct}) \leftarrow (\text{Direct}) \text{ XRL A}$	Direct	2	1
	Direct, #Data	$(\text{Direct}) \leftarrow (\text{Direct}) \text{ XRL \#Data}$	Direct	3	2
CLR	A	$A \leftarrow 00H$	--	1	1
CPL	A	$A \leftarrow A$	--	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

Architecture & Instruction set of 8085 Microprocessor and 8051 Micro Controller

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.

Architecture & Instruction set of 8085 Microprocessor and 8051 Micro Controller

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

Architecture & Instruction set of 8085 Microprocessor and 8051 Micro Controller

Mnemonic	Instruction	Description	# of Bytes	# of Cycles
ACALL	ADDR11	Absolute Subroutine Call $PC + 2 \rightarrow (SP); ADDR11 \rightarrow PC$	2	2
LCALL	ADDR16	Long Subroutine Call $PC + 3 \rightarrow (SP); ADDR16 \rightarrow PC$	3	2
RET	--	Return from Subroutine $(SP) \rightarrow PC$	1	2
RETI	--	Return from Interrupt	1	2
AJMP	ADDR11	Absolute Jump $ADDR11 \rightarrow PC$	2	2
LJMP	ADDR16	Long Jump $ADDR16 \rightarrow PC$	3	2
SJMP	rel	Short Jump $PC + 2 + rel \rightarrow PC$	2	2
JMP	@A + DPTR	$A + DPTR \rightarrow PC$	1	2
JZ	rel	If $A=0$, Jump to $PC + rel$	2	2
JNZ	rel	If $A \neq 0$, Jump to $PC + rel$		
CJNE	A, Direct, rel	Compare (Direct) with A. Jump to $PC + rel$ if not equal	3	2
	A, #Data, rel	Compare #Data with A. Jump to $PC + rel$ if not equal	3	2
	Rn, #Data, rel	Compare #Data with Rn. Jump to $PC + rel$ if not equal	3	2
	@Ri, #Data, rel	Compare #Data with @Ri. Jump to $PC + rel$ if not equal	3	2
DJNZ	Rn, rel	Decrement Rn. Jump to $PC + rel$ if not zero	2	2
	Direct, rel	Decrement (Direct). Jump to $PC + rel$ if not zero	3	2
NOP		No Operation	1	1

Architecture & Instruction set of 8085 Microprocessor and 8051 Micro Controller
