

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"

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 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.

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	$A \leftarrow \text{Data}$	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

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	$A \leftarrow A + \text{Data}$	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 ($A \leftarrow$ Lower Byte of $A*B$ and $B \leftarrow$ Higher Byte of $A*B$)	--	1	4
DIV	AB	Divide A by B ($A \leftarrow$ Quotient and $B \leftarrow$ 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
- 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

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	$C \leftarrow 0$ (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 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.

Mnemonic	Instruction	Description	# of Bytes	# of Cycles
ACALL	ADDR11	Absolute Subroutine Call PC + 2 → (SP); ADDR11 → PC	2	2
LCALL	ADDR16	Long Subroutine Call PC + 3 → (SP); ADDR16 → PC	3	2
RET	--	Return from Subroutine (SP) → PC	1	2
RETI	--	Return from Interrupt	1	2
AJMP	ADDR11	Absolute Jump ADDR11 → PC	2	2
LJMP	ADDR16	Long Jump ADDR16 → PC	3	2
SJMP	rel	Short Jump PC + 2 + rel → PC	2	2
JMP	@A + DPTR	A + DPTR → PC	1	2
JZ	rel	If A=0, Jump to PC + rel	2	2
JNZ	rel	If A ≠ 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
			ELECTRONICS HUB	
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

