UNIT -2
MICROCONTROLLERS

Advantages of microcontrollers

- The overall system cost is low, as the peripherals are integrated in a single chip.
- The product is of small size as compared to the microprocessor based system and is very handy.
- The system is more reliable.
- The system is easy to troubleshoot and maintain.
- If required additional RAM, ROM and I/O ports may be interfaced

ARCHITECTURE OF 8051

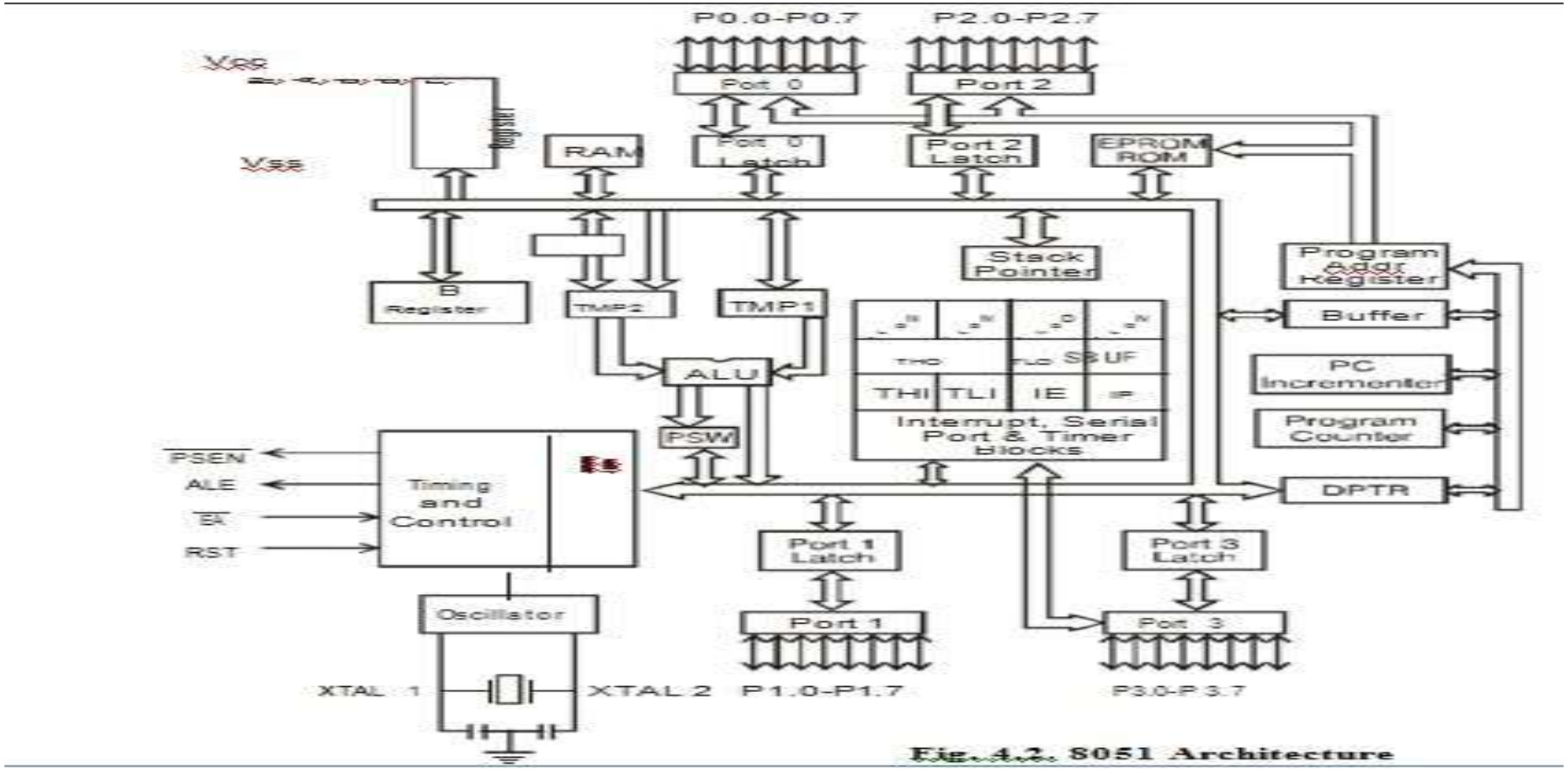


Fig. 4.2. 8051 Architecture

The features of the 8051 are :

- 8 bit CPU with registers A (the accumulator) and B
- 16 bit Program Counter (PC) and Data Pointer (DPTR)
- 8 bit Program Status Word (PSW)
- 64K Program memory address space
- 64K Data memory address space
- 128 bytes of on chip data memory
- 32 I/O pins for four 8 bit ports : Port 0, Port 1, Port 2, Port 3
- Two 16 bit timers / counters : T_0 and T_1
- Full duplex UART : SBUF
- Two external and three internal interrupt sources
- On chip clock oscillator.

Central processing unit

- The CPU is the brain of the microcontrollers reading user's programs and executing the expected task as per instructions stored there in. It's primary elements are an Accumulator (ACC), B register (B), Stack pointer (SP), Program counter (PC), Program status word (PSW), Data pointer register (DPTR) and few more 8 bit registers.

Accumulator

- The accumulator performs arithmetic and logic functions on 8 bit input variables.
- Arithmetic operations include basic addition, subtraction, multiplication and division.
- Logical operations are AND, OR XOR as well as rotate, clear, complement etc.
- Apart from all the above, accumulator is responsible for conditional branching decisions and provides a temporary place in a data transfer operations within the device.

B Register

- B register is used in multiply and divide operations.
- During execution B register either keeps one of the two inputs and then retains a portion of the result.
- For other instructions it is used as general purpose register.

Stack Pointer

- Stack Pointer (SP) is an 8 bit register.
- This pointer keeps track of memory space where the important register information are stored when the program flow gets into executing a subroutine.
- The stack portion may be placed in anywhere in the onchip RAM.
- But normally SP is initialized to 07H after a device reset and grows up from the location 08H.
- The SP is automatically incremented or decremented for all PUSH or POP instructions and for all subroutine calls and returns.

Program Counter

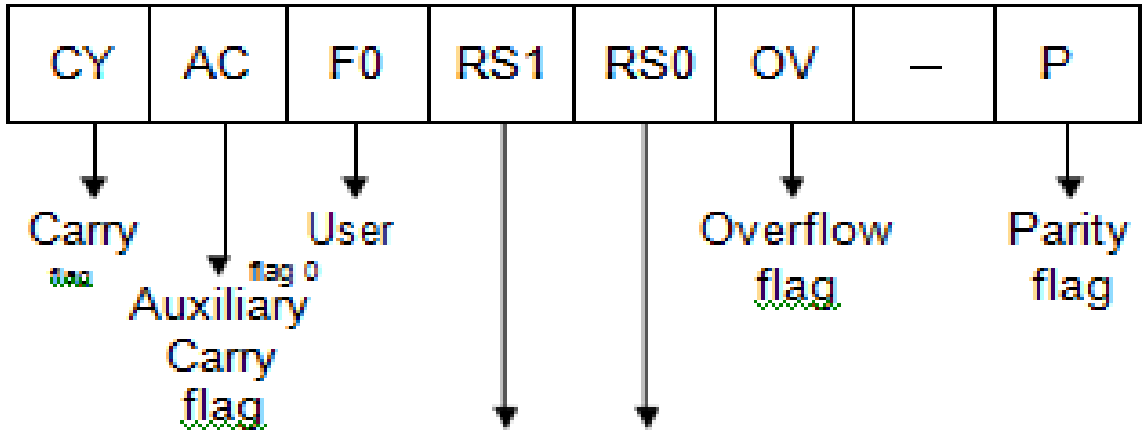
- The Program Counter (PC) is the 16 bit register giving address of next instruction to be executed during program execution.
- It always points to the program memory space.

Data Pointer Register

- The Data Pointer Register (DPTR) is the 16 bit addressing register that can be used to fetch any 8 bit data from the data memory space.
- When it is not being used for this purpose, it can be used as two eight bit registers, DPH and DPL.

Program Status Word

- The Program Status Word (PSW) keeps the current status of the arithmetic and logic operations in different bits.
- The 8051 has four math flags that respond automatically to the outcomes of arithmetic and logic operations and 3 general purpose user flags that can be set 1 or cleared to 0 by the programmer as desired.
- The math flags are carry (C), auxiliary carry (AC), overflow (OV) and parity (P).
- User flags are named flag 0 (F0), Register bank select bits RS0 and RS1.



0 0 - Select register bank 0

0 1- Select register bank 1

1 0- Select register bank 2

1 1 - Select register bank 3

Input / Output Ports

- 8051 has 32 I/O pins configured as 4 eight bit parallel ports (P0, P1, P2 and P3).
- Each pin can be used as an input or as an output under the software control.
- These I/O pins can be accessed directly by memory instructions during program execution to get require flexibility.

Timers / Counters

- 8051 has two 16 bit Timers / Counters, T0 and T1 capable of working in different modes.
- Each consists of a ‘HIGH’ byte and a ‘LOW’ byte which can be accessed under software.
- There is a mode control register (TMOD) and a control register (TCON) to configure these timers / counters in number of ways.
- These timers are used to measure time intervals, determine pulse widths or initiate events with one microsecond resolution upto a maximum 65ms.

Serial Port

- The 8051 has a high speed full duplex serial port which is software configurable in 4 basic modes :
- Shift register mode
- Standard UART mode
- Multiprocessor mode
- 9 bit UART mode

Interrupts

- The 8051 has five interrupt sources : One from the serial port (RI / TI) when a transmission or reception operation is executed : two from the timers (TF0, TF1) when overflow occurs and two come from the two input pins INT0, INT1.
- Each interrupt may be independently enabled or disabled to allow polling on same sources and each may be classified as high or low priority.
- These operations are selected by Interrupt Enable (IE) and Interrupt Priority (IP) registers.

Oscillator and Clock

- The 8051 generates the clock pulses by which all internal operations are synchronized.
- Pins XTAL 1 and XTAL 2 are provided for connecting a resonant network to form an oscillator.
- A quartz crystal is used for oscillator.
- The crystal frequency is the basic internal clock frequency of the microcontroller.

SPECIAL FUNCTION REGISTERS (SFRS)

- The address of the Special Function Registers are above 80H, since the addresses 00H to 7FH are the addresses of RAM memory.
- The SFRs have addresses between 80H and FFH.
- But all the address space of 80H to FFH is not used by the SFRs.
- The unused locations are reserved and must not be used by the programmer.

Table 4.3: Special Function Registers

Name	Function	Address (Hex)
Acc (A)	Accumulator	E0
B	Arithmetic	F0
DPH	(Data Pointer High byte) Addressing external memory	83
DPL	Data Pointer Low byte	82
IE	Interrupt Enable Control	A8
IP	Interrupt Priority Control	B8
P0	I/O Port 0 Latch	80
P1	I/O Port 1 Latch	90
P2	I/O Port 2 Latch	A0
P3	I/O Port 3 Latch	B0
PCON	Power Control	87
PSW	Program Status Word	D0
SCON	Serial Port Control	98
SBUF	Serial Port Data Buffer	99
SP	Stack Pointer	81
TMOD	Timer / Counter Mode Control	89
TCON	Timer / Counter Control	88
TL0	Timer 0 low byte	8A
TH0	Timer 0 high byte	8C
TL1	Timer 1 low byte	8B
TH1	Timer 1 high byte	8D

ADDRESSING MODES

- Immediate addressing mode
- Register addressing mode
- Direct addressing mode
- Register indirect addressing mode
- Indexed addressing mode

Immediate Addressing Mode

- When a source operand is a constant rather than a variable, then the constant can be embedded into the instruction itself.
- This kind of instructions take two bytes and first one specifies the opcode and second byte gives the required constant.
- The operand comes immediately after the opcode. The mnemonic for immediate data is the pound sign (#).
- This addressing mode can be used to load information into any of the registers including DPTR register.

Examples :

MOV A, # 18H

MOV B, # 65H

MOV DPTR, #2040H

DPL ← 40H

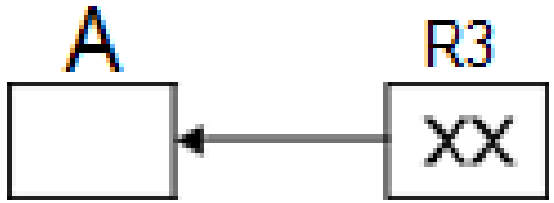
DPH ← 20H

Register Addressing Mode

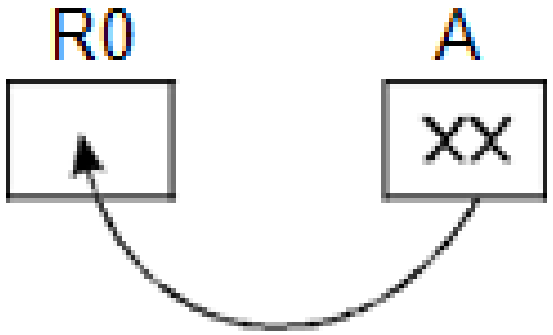
- Register addressing accesses the eight working registers (R_0 - R_7) of the selected register bank.
- The least significant three bits of the instruction opcode indicate which register is to be used for the operation.
- One of the four banks of registers is to be predefined in the PSW before using register addressing instruction.
- ACC, B and DPTR can also be addressed in this mode.

Examples :

MOV A, R3



MOV R0, A



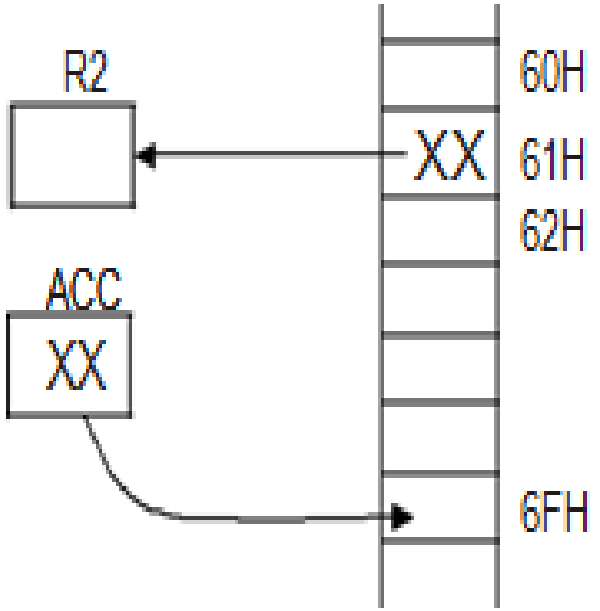
Direct Addressing Mode

- In the direct addressing mode, all 128 bytes of internal RAM and the SFRs may be addressed directly using the single - byte address assigned to each RAM location and each SFR.
- Internal RAM uses address from 00H to 7FH to address each byte.

Examples

MOV R2, 61H

MOV 6F H, A



Register Indirect Addressing Mode

- In this mode a register is used as a pointer to the data.
- If the data is inside the CPU, only registers R0 and R1 are used for this purpose.
- When R0 and R1 hold the addresses of RAM locations, they must be preceded by the “@” sign.

Examples

MOV @ R1, A : Move contents of A into RAM location whose address is held by R1.

MOV B, @ R0 : Move contents of RAM location whose address is held by R0 into B.

Indexed Addressing Mode

- Only the program memory can be accessed by this mode.
- This mode is intended for reading lookup tables in the program memory.
- A 16 bit base register (DPTR or PC) points to the base of the lookup tables and accumulator carries the constant indicating table entry number.
- The address of the exact location of the table is formed by adding the accumulator data to the base pointer.

Example

MOVC A, @A + DPTR

- The contents of A are added to the DPTR to form the 16 bit address of the needed data. ‘C’ means code.

I/OPORTS port 0 (P0.0 - 0.7)

- Port 0 is used for both address and data bus ($AD_0 - AD_7$).
- When the microcontroller chip is connected to an external memory, Port 0 provides both address and data.
- ALE pin indicates if Port 0 has address or data.
- When ALE = 0, Port 0 provides data ($D_0 - D_7$)
  = 1, Port 0 provides address ($A_0 - A_7$)
- ALE is used for demultiplexing address and data with the help of a latch

Port 1 (P1.0 - P1.7)

- Port 1 pins are used as input or output.
- To make port 1 as an input port, write 1 to all its 8 bits.
- To make port 1 as output port, write 0 to all its 8 bits.
- Thus port 1 pins have no dual functions.

Port 2 (P2.0 - P2.7)

- Port 2 pins are used as input / output pins similar in operation to port 1.
- The alternate use of port 2 is to supply a high order address byte (A_8 – A_{15}) when the microcontroller is connected to external memory

Port 3 (P3.0 – P3.7)

- Port 3 pins are used as input or output

Pin	Function
P3.0 - RXD	Serial data input
P3.1 - TXD	Serial data output
P3.2 - $\overline{\text{INT0}}$	External interrupt 0
P3.3 - $\overline{\text{INT1}}$	External interrupt 1
P3.4 - T0	External timer 0 input
P3.5 - T1	External timer 1 input
P3.6 - $\overline{\text{WR}}$	External memory write pulse
P3.7 - $\overline{\text{RD}}$	External memory read pulse

INSTRUCTION SET

- An instruction is a command given to the computer to perform a specified operation on given data.
- The instruction set is the collection of instructions that the microcontroller is designed to execute.
- The programmer can write the program in assembly language using these instructions.

- Data transfer group
- Arithmetic group
- Logical group
- Boolean variable manipulation
- Program branching

Data

Mnemonic	Description	Operation
MOV A, Rn	A ← Rn	Move register to accumulator
MOV A, direct	A ← (addr)	Move direct byte to accumulator
MOV A, @ Ri	A ← (Ri)	Move indirect RAM to accumulator
MOV A, # data	A ← data	Move immediate datato accumulator
MOV Rn, A	Rn ← A	Move accumulator to register
MOV Rn, direct	Rn ← (addr)	Move direct byte to register
MOV Rn, #data	Rn ← data	Move immediate datato register
MOV direct, A	(addr) ← A	Move accumulator to direct byte
MOV direct, Rn	(addr) ← Rn	Move register to direct byte
MOV direct, direct	(addr 1) ← (addr 2)	Move direct byte to direct byte
MOV direct, @Ri	(addr) ← (Ri)	Move indirect RAM to direct byte
MOV direct, #data	(addr) ← data	Move immediate datato direct byte
MOV, @ Ri, A	(Ri) ← A	Move accumulator to indirect RAM
MOV @ Ri, direct	(Ri) ← (addr)	Move direct byte into indirect RAM
MOV DPTR, # data 16	DPTR ← data 16	Load data pointerwith 16 bit constant
MOV C A, @A + DPTR	A ← (A + DPTR)	Move code byte relative to DPTR to accumulator
MOVC A, @A + PC	A ← (A + PC)	Move code byte relative to PC to accumulator.
MOV X A, @ Ri	A ← (Ri)^	Move external RAM (8 bit address) to accumulator
MOV X A, @ DPTR	A ← (DPTR)^	Move external RAM (16 bit address) to accumulator.
MOV X @ Ri, A	(Ri)^ ← A	Move accumulator to external RAM (8 bit address)
MOV X @ DPTR, A	(DPTR)^ ← A	Move accumulator to external RAM (16 bit address)
PUSH direct	(SP) ← ADDR	Push direct byte onto stack

Mnemonic	Description	Operation
POP direct	(addr) ← (SP)	POP direct byte from stack
XCH A, Rn	A ↔ Rn	Exchange register with accumulator
XCH A, direct	A↔ (addr)	Exchange direct byte with accumulator
XCH A, @Ri	A ↔ (Ri)	Exchange indirect RAM with accumulator
XCHD A, @Ri	AL ↔ (Ri)L	Exchange low order digit indirect RAM with accumulator

ARITHMETIC INSTRUCTIONS

Mnemonic	Description	Operation
ADD, A, Rn	A ← A + Rn	Add register to accumulator
ADD A, direct	A ← A + (addr)	Add direct byte to accumulator
ADD A, @Ri	A ← A + (Ri)	Add indirect RAM to accumulator
ADD A, # data	A ← A + data	Add immediate data to accumulator
ADDC A, Rn	A ← A + Rn + C	Add register to accumulator with carry
ADDC A, direct	A ← A + (addr) + C	Add direct byte to accumulator with carry
ADDC A, @Ri	A ← A + (Ri) + C	Add indirect RAM to accumulator with carry
ADDC A, # data	A ← A + data	Add immediate data to accumulator with carry
SUBB A, Rn	A ← A - Rn -C	Subtract register from accumulator with borrow
SUBB A, direct	A ← A -(addr) - C	Subtract direct byte from accumulator with borrow
SUBB A, @ Ri	A ← A -(Ri)-C	Subtract indirect RAM from accumulator with borrow
SUBB A, # data	A ← A - data - C	Subtract immediate data from accumulator with borrow
INC A	A ← A + 1	Increment accumulator
INC Rn	Rn ← Rn + 1	Increment register
INC direct	(addr) ← (addr) + 1	Increment direct byte
INC @Ri	(Ri) ← (Ri) + 1	Increment indirect RAM
INC DPTR	DPTR ← DPTR +1	Increment data pointer
DEC A	A ← A - 1	Decrement accumulator
DEC Rn	Rn ← Rn - 1	Decrement register
DEC direct	(addr) ← (addr) -1	Decrement direct byte
DEC @ Ri	(Ri) ← (Ri) -1	Decrement indirect RAM
MUL AB	AB ← A x B	Multiply A and B
DIV AB	AB ← A/B	Divide A by B
DA A	$A_{dec} \leftarrow A_{bin}$	Decimal adjustaccumulator

Mnemonic	Description	Operation
ANL, A, Rn	(A) AND (Rn)	AND register to accumulator
ANL A, direct	(A) AND (addr)	AND direct byte to accumulator
ANL A, @Ri	(A) AND ((Ri))	AND indirect RAM to accumulator
ANL A, #data	(A) AND data	AND immediate data to accumulator
ANL direct, A	(addr) AND (A)	AND accumulator to direct byte
ANL direct, #data	(addr) AND data	AND immediate data to direct byte
ORL A, Rn	(A) OR (Rn)	OR register to accumulator
ORL A, direct	(A) OR (addr)	OR direct byte to accumulator
ORL A, @Ri	(A) OR ((Ri))	OR indirect RAM to accumulator
ORL A, #data	(A) OR data	OR immediate data to accumulator
ORL direct, A	(addr) OR (A)	OR accumulator to direct byte
ORL direct, # data	(addr) OR data	OR immediate data to direct byte
XRL A, Rn	(A) XOR (Rn)	Ex - OR register to accumulator
XRL A, direct	(A) XOR (addr)	EX - OR direct byte to accumulator
XRL A, @Ri	(A) XOR ((Ri))	EX - OR indirect RAM to accumulator
XRL A, #data	(A) XOR data	EX - OR immediate data to accumulator
XRL direct, A	(addr) XOR (A)	EX - OR accumulator to direct byte
XRL direct, #data	(addr) XOR data	EX - OR immediate data to direct byte
RL A	$A_0 \leftarrow A_7 \leftarrow A_6 \ldots \leftarrow A_1 \leftarrow A_0$	Rotate accumulator left
RLC A	$C \leftarrow A_7 \leftarrow A_6 \ldots \leftarrow A_0 \leftarrow C$	Rotate accumulator left through carry
RR A	$A_0 \rightarrow A_7 \rightarrow A_6 \ldots \rightarrow A_1 \rightarrow A_0$	Rotate accumulator right
RRC A	$C \rightarrow A_7 \rightarrow A_6 \ldots \rightarrow A_0 \rightarrow C$	Rotate accumulator right through carry
CLR A	$A \leftarrow 00$	Clear accumulator
CPL A	$A \leftarrow \overline{A}$	Complement accumulator
SWAP A	$A_L \leftrightarrow A_H$	Swap nibbles within the accumulator.

Mnemonic	Description	Operation
CLR C	C ← 0	Clear carry
CLR bit	bit ← 0	Clear direct bit
SETB C	C ← 1	Set carry
SETB bit	bit ← 1	Set direct bit

CPL C	C ← $\overline{C}$	Complement carry
CPL bit	bit ← $\overline{bit}$	Complement direct bit
ANL C bit	(C)AND bit	AND direct bit to carry
ANL C, $\overline{bit}$	(C)AND $\overline{bit}$	AND complement of direct bit to carry
ORL C, bit	(C) OR bit	OR direct bit to carry
ORL C, $\overline{bit}$	(C) OR $\overline{bit}$	OR complement of direct bit to carry
MOV C, bit	C ← bit	Move direct bit to carry
MOV bit, C	bit ← C	Move carry to direct bit
JC radd	[C = 1]; PC ← PC + 2 + radd	Jump if carry is set
JNC radd	[C = 0]; PC ← PC+2 + radd	Jump if carry is not set.
JB bit, radd	[bit = 1]; PC ← PC+3 + radd	Jump if direct bit is set
JNB bit, radd	[bit = 0]; PC ← PC+3 + radd	Jump if direct bit is not set
JBC bit, radd	[bit = 1]; PC ← PC+3 + radd	Jump if direct bit is set and clear bit

Mnemonic	Description	Operation
ACALL sadd	(SP) ← PC + 2; PC ← sadd	Absolute subroutine call
LCALL ladd	(SP) ← PC + 3; PC ← ladd	Long subroutine call
RET	PC ← (SP)	Return from sub - routine
RETI	PC ← (SP); EI	Return from interrup
AJUMP sadd	PC ← sadd	Absolute jump
LJUMP ladd	PC ← ladd	Long jump
SJUMP radd	PC ← PC + 2 + radd	Short jump (relative address)
JMP @ A + DPTR	PC ← DPTR + A	Jump indirect relative to the DPTR
JZ radd	[A = 00]; PC ← PC + 2 + radd	Jump if accumulator is zero
JNZ radd	[A > 00]; PC ← PC + 2 + radd	Jump if accumulator is not zero.
CJNE A, direct, radd	[A <> (addr)]; PC ← PC + 3 + radd	Compare direct byte to Acc and jump if not equal.
CJNE A, # data, radd	[A <> (data)]; PC ← PC + 3 + radd	Compare immediate data to Acc and jump if not equal.
CJNE Rn, # data, radd	[(R_n) <> data]; PC ← PC + 3 + radd	Compare immediate data to register and jump if not equal.
DJNZ Rn, radd	[R_n-1 <> 00]; PC ← PC + 3 + radd	Decrement register and jump if not zero.
DJNZ direct, radd	[(add) -1 <> 00]; PC ← PC + 3 + radd	Decrement direct byte and jump if not zero.
NOP	PC ← PC +1	No operation.