UNIT-IV Introduction to 8051Microcontroller

MICROPROCESSORS AND MICROCONTROLLERS

A digital computer typically consists of three major components—the Central Processing Unit (CPU), program and data memory, and an Input/Output (I/O) system. The CPU controls the flow of information among the components of the computer. It also processes the data with Arithmetic-Logic Unit (ALU) within the CPU. A microprocessor is a CPU that is compacted into a single-chip semiconductor device. Microprocessors are general-purpose devices, suitable for many applications. A computer built with a microprocessor is called a microcomputer.

A microcontroller is an entire computer manufactured on a single chip. Microcontrollers are usually dedicated devices embedded within an application. A microcontroller is a highly integrated chip, which includes all or most of the parts needed for a controller on one chip. The microcontroller could be called a "one-chip solution". It typically includes: CPU (central processing unit), RAM (Random Access Memory), EPROM/ PROM/ROM (Erasable Programmable Read Only Memory), I/O (input/output), serial and parallel timers, interrupt controller.

Differences between Micro-processor and Micro-controller: -

S.NO	Micro-processor	Micro-controller
1	Micro-processor contains ALU, GPRs, PC, SP, Control and Timing circuit, and interrupt circuit.	Micro-controller contains the CPU, in built ROM, RAM, I/O ports, and Timer &Counter circuits.
2	It has many instructions to move data between Memory and CPU.	It has few instructions to move data between Memory and CPU.
3	Access time for memory and I/O devices are more.	Less access time for built-in memory and I/O devices.
4	Microprocessor based system requires more hardware.	Microcontroller based system requires less hardware.
5	Microprocessor based system is more flexible in design point of view	Microcontroller based system is less flexible in design point of view
6	Less number of pins are multi- functioned.	More number of pins are multi- functioned.
7	CPU on a single chip is called Microprocessor.	Computer on a single chip is called Microcontroller.

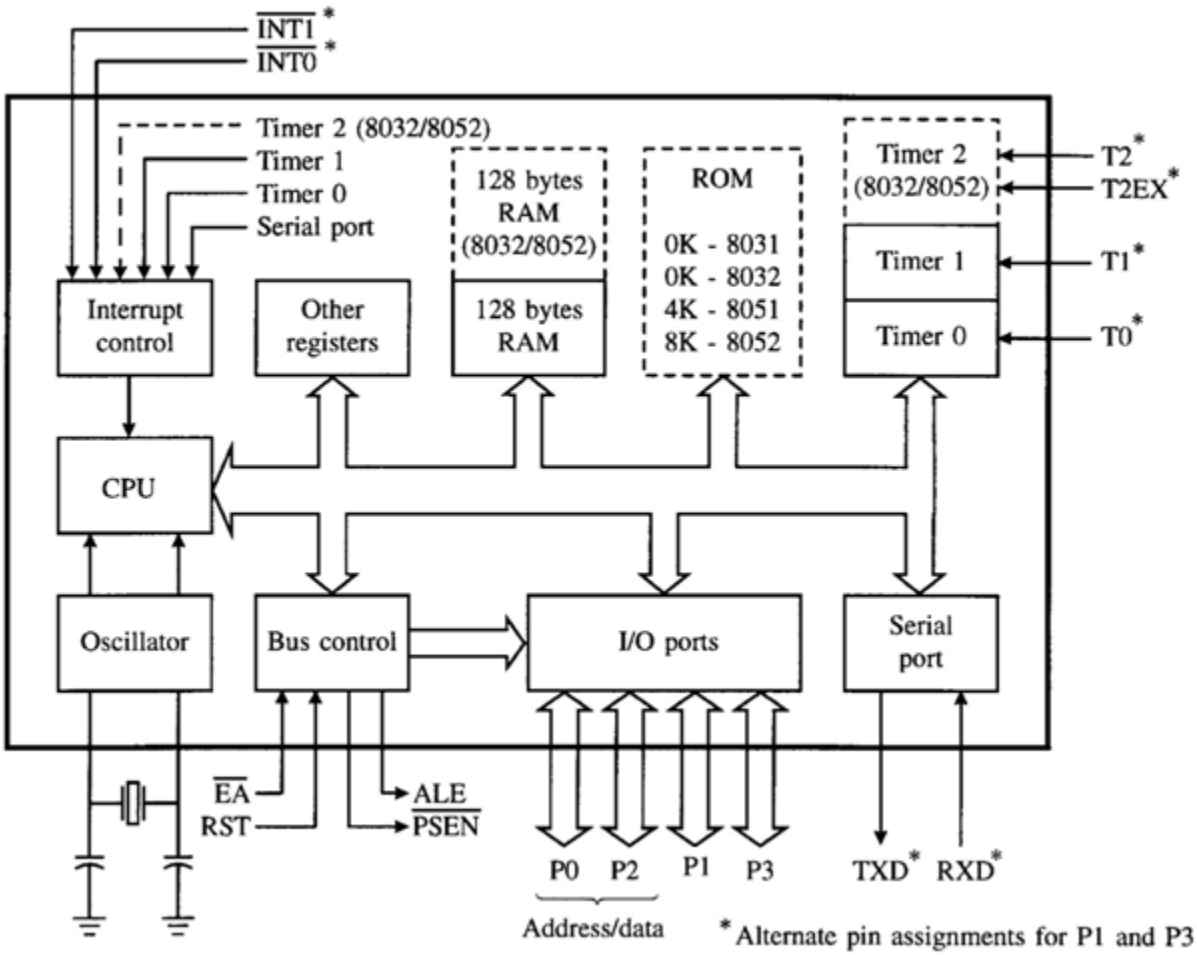
Features of 8051 Micro-controller: -

1. It is an 8-bit Micro-controller.
2. It has 8 data lines (D7-D0) and 16 address lines (A15-A0). All data lines are multiplexed with the lower order 8- address lines (A7-A0), and then the multiplexed bus is (AD7- AD0).
3. It has 34 8-bit GPRS, and four 16-bit registers (PC, DPTR, T0, & T1).
4. It has four I/O ports, and each port contains eight I/O pins.
5. It has 128 bytes of Internal RAM and 4Kbytes of internal ROM
6. It supports 64KB Data memory address space and 64KB program memory address space.
7. Its operating clock frequency of 1MHz -16 MHz, but for commercial applications frequency is 12MHz theoretically but practically it is 11.0592MHz.
8. It supports full duplex serial communication.
9. It supports five interrupts in that two are external interrupts and three are internal interrupts. All are **vectored interrupts and Maskable interrupts**.
10. Four mathematical flags and three user defined flags.
11. Two 16-bit Timer/Counter circuits.

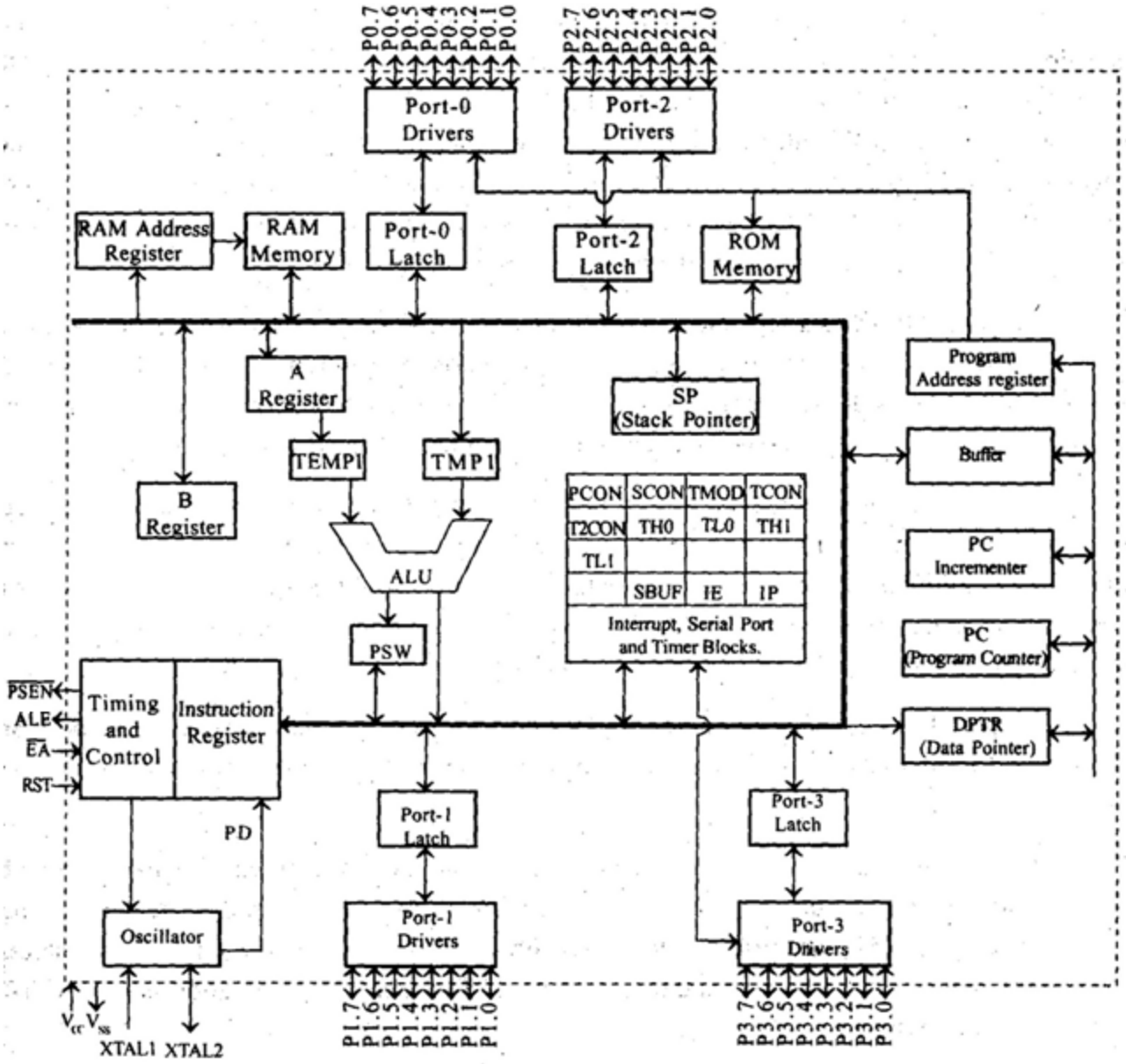
Intel 8051 Family

Microcontroller	*On-chip Code Memory*	*On-chip Data Memory*	*Timers*
8051	4K ROM	128 bytes	2
8031	0	128 bytes	2
8751	4K EPROM	128 bytes	2
8052	8K ROM	256 bytes	3
8032	0	256 bytes	3
8752	8K EPROM	256 bytes	3

Block Diagram of 8051 Micro-controller



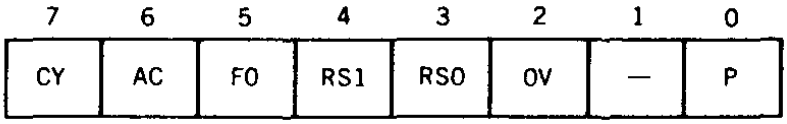
Architecture of 8051 Micro-controller: -



A Register (Accumulator): - The A (accumulator) register is the most versatile of the two CPU registers and is used for many operations, including addition, subtraction, integer multiplication and division, and Boolean bit manipulations. The A register is also used for all data transfers between the 8051 and any external memory.

B Register: - The B register is used with the A register for multiplication and division operations and has no other function other than as a location where data may be stored.

Program Status Word Register (PSW): -



It contains several status bits that reflect the current state of the CPU. Besides, this register contains four mathematical flags (Carry flag, Auxiliary Carry, Overflow flag, parity bit) two register bank select bits (RS1 & RS0), and one user-definable status flag (F0) and one bit is not defined.

P - Parity bit: - If a number stored in the accumulator A contains odd number of 1's then this bit will be automatically set (1), otherwise it will be cleared (0). It is mainly used during data transmit and receive via serial communication.

OV Overflow: - Overflow occurs when the result of an arithmetical operation is larger than 255 and cannot be stored in one register. Overflow condition causes the OV bit to be set (1). Otherwise, it will be cleared (0).

RS0, RS1 - Register bank select bits. These two bits are used to select one of four register banks of RAM. By setting and clearing these bits, registers R0-R7 are stored in one of four banks of RAM.

RS1	RS0	Space in RAM
0	0	Bank0 (00H-07H)
0	1	Bank1 (08H-0FH)
1	0	Bank2 (10H-17H)
1	1	Bank3 (18H-1FH)

F0 – User Flag 0. This is a general-purpose user defined flag the use of this flag is decided by the user.

AC - Auxiliary Carry Flag: - It is used for BCD operations only. This flag is set to '1' when in the addition operation the carry is generated at bit position D3 or in subtraction operation borrow is needed at the bit position D3.

CY - Carry Flag: - This flag is set to '1' when in the addition operation the final carry is generated or in subtraction operation the Minuend is less than the Subtrahend.

PC (Program Counter): - The 8051 contains two 16-bit registers: the program counter (PC) and the data pointer (DPTR). Each is used to hold the address of a byte in memory. The program counter points to the address of the next instruction to be executed. As the CPU fetches the opcode from the program ROM, the program counter is increasing to point to the next. The PC is the only register that does not have an internal address. The program counter is 16 bits wide, this means that it can access program addresses 0000 to FFFFH, a total of 64Kbytes of code.

DPTR (Data Pointer): - The DPTR register is made up of two 8-bit registers, named DPH and DPL, that are used to hold memory addresses for internal and external code access and external data access. The DPTR is under the control of program instructions and can be specified by its 16-bit name, DPTR, or by each individual byte name, DPH and DPL. DPTR does not have a single internal address; DPH and DPL are each assigned an address.

Internal Memory: - The 8051 Microcontroller has internal program memory (ROM) and internal data memory (RAM). Unlike microcontrollers with Von Neumann architectures, which can use a single memory address for either program code or data, but not for both, the 8051 has a Harvard architecture, which uses the same address, in different memories, for code and data. Internal circuitry accesses the correct memory based upon the nature of the operation in progress.

Internal RAM: - The 8051 microcontroller has 128 bytes of internal RAM, its address range from 00H to 07FH. From 80H to 0FFH address space are assigned to SFRs (Special Function Registers). The internal RAM 128Bytes can divide into three parts. Those are

1. Register Banks – 32 Bytes (00H – 1FH)
2. Bit/Byte addressable memory – 16 Bytes (20H – 2FH)
3. User memory or General purpose memory—80 Bytes (30H – 7FH)

A. Thirty two bytes addresses from 00H to 1FH that make up 32 working registers organized as four register banks of eight registers each. The register banks are numbered from 0 to 3 and are made up of eight registers named R0 to R7. Each register can be called in to the program by its name or by its RAM address. The register bank selection bits RS1 & RS0 in the PSW decides which register bank currently used for the operation of the microcontroller. The register bank-0 is selected when the microcontroller is reset.

B. The Bit / Byte addressable area of 16-Bytes occupies RAM addresses from 20H-2FH forming a total of 128 addressable bits. A bit addressable bit can be specified by its bit addresses from 00H to 7FH or 8- bits may form a byte address from 20H-2FH. Addressable bits are useful when the

program need only remember a binary equivalent.

C. A general purpose RAM area starts from 30H-7FH. This is addressable as bytes.

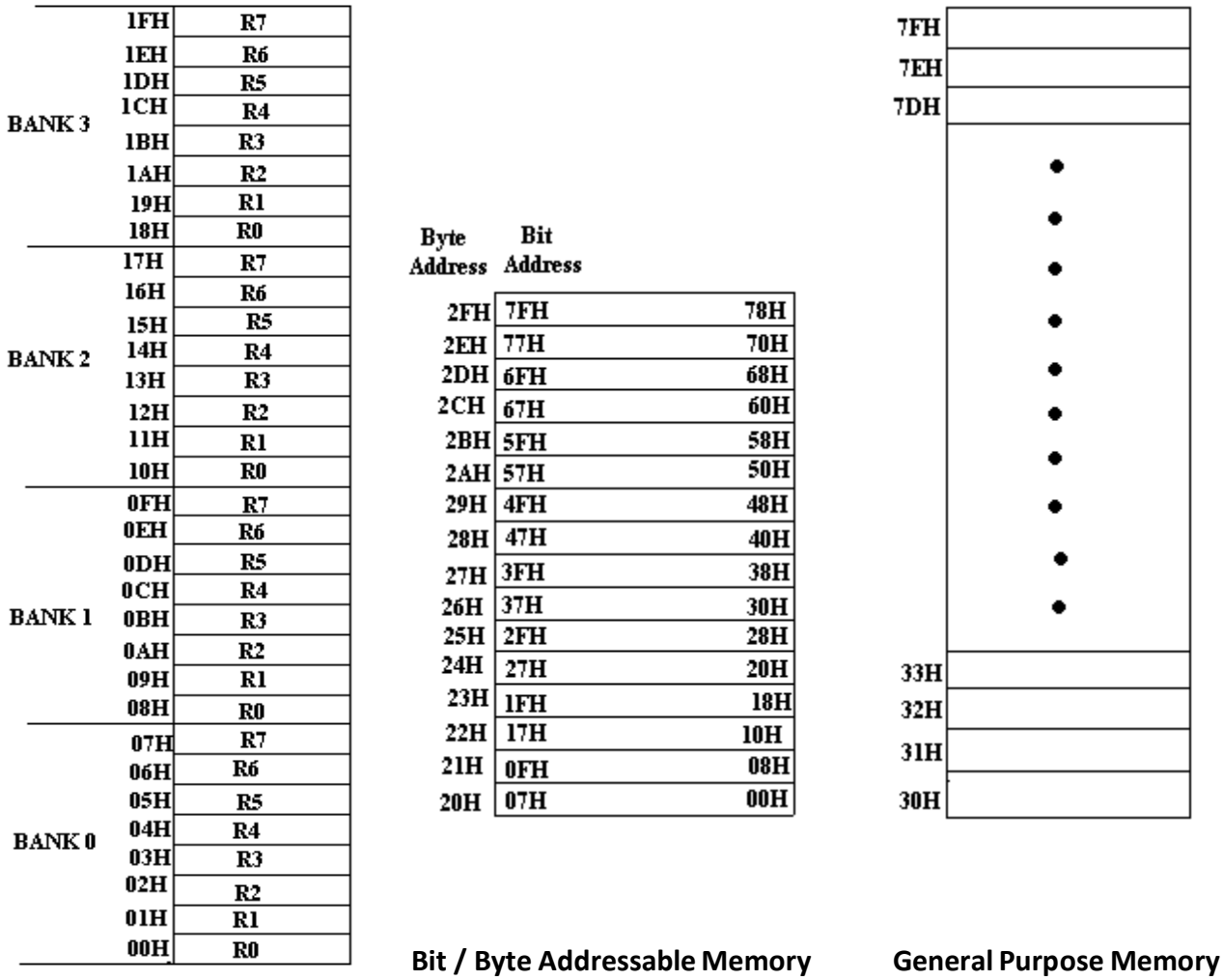


FIG: INTERNAL RAM ORGANISATION

The Stack and the Stack Pointer: The stack refers to an area of internal RAM that is used in conjunction with certain op-codes to store and retrieve data quickly.

The 8-bit stack pointer (SP) register is used by the 8051 to hold an internal RAM address that is called the "top of the stack."

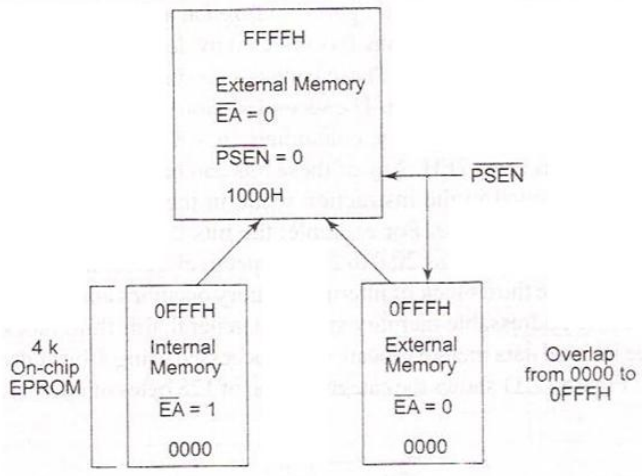
The address held in the SP register is the location in internal RAM where the last byte of data was stored by a stack operation.

When data is to be placed on the stack, the SP increments before storing data on the stack so that the stack grows up as data is stored.

As data is retrieved from the stack, the byte is read from the stack, and then the SP decrements to

point to the next available byte of stored data.

Internal ROM: -The 8051 microcontroller has an internal ROM of 4Kbytes, its addresses from 0000H to 0FFFH. This is used to store the system files or system code information.
The program addresses higher than the 0FFFH, which exceeds the internal ROM capacity, will cause the 8051to automatically fetch the program codes from external program memory.
Code bytes can also be fetched exclusively from the external program memory addresses from 0000H to0FFFFH, by connecting the EA(External Active or External Enable) pin to the ground.



Program Memory Map of an 8051 System

Special Function Registers (SFR): -

8051 Microcontroller contains several special function registers, which can perform different functions in the controller. In these SFRs some registers are Bit Addressable Registers and some are Byte Addressable Registers.
Bit Addressable Registers means these registers a byte address is allotted for the entire register and each and every individual bit also have their own address. We can call these register bits individually by their bit addresses as well as register byte addresses.
The addresses for these SFRs in 8051 microcontroller are starts from 80H to 0FFH.

Register Name	Function	Internal Address in Hex
A	Accumulator *	0E0H
B	Arithmetic *	0F0H
DPL	Data Pointer Lower	82H

DPH	Data Pointer Higher	83H
SP	Stack Pointer	81H
TCON	Timer Control Register *	88H
TMOD	Timer Mode Register	89H
SCON	Serial Control Register *	98H
SBUF	Serial Buffer Register	99H
PCON	Power Control Register	87H
IP	Interrupt Priority Register *	0A8H
IE	Interrupt Enable Register *	0B8H
PORT 0	Input/output Port Latch – 0 *	80H
PORT 1	Input/output Port Latch – 1 *	90H
PORT 2	Input/output Port Latch – 2 *	0A0H
PORT 3	Input/output Port Latch – 3 *	0B0H
TL0	Timer - 0 Lower	8AH
TH0	Timer - 0 Higher	8CH
TL1	Timer - 1 Lower	8BH
TH0	Timer - 1 Higher	8DH
PSW	Program Status Word *	0D0H

*Indicate the registers are Bit Addressable registers

MEMORY ORGANISATION

The 8051 devices have 4 Kbytes of on-chip program memory and 128 bytes of on-chip data random access memory. In addition to internal memory, 64 Kbytes of program memory and 64 Kbytes of data memory can be interfaced with 8051 using port 0, port 2, port 3.6, port 3.7, PSEN and EA pins.

Program memory is accessed using the program counter (PC) and A, or DPTR and A.

Data memory is accessed using DPTR, R0 and R1 register.

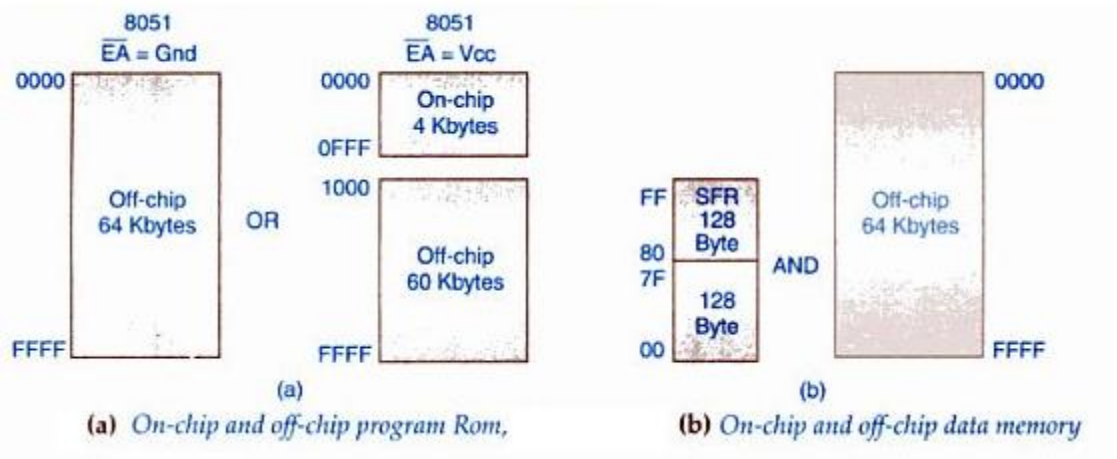
In 8051, port 0 and port 2 provide 16 bit address to access external memory. Port 0 provides the lower 8 bit address (A0-A7) and 8 bit data (D0-D7). This is called address/data multiplexing. Port 2 provides the upper 8 bit address (A8-A15). ALE pin of 8051 and 74LS373 latch are used to demultiplex AD0-AD7.

When EA is connected to the ground, 8051 fetches instructions from external ROM by using PSEN. Then, the address of external ROM is 0000H to FFFFH.

When EA is connected to Vcc, then it fetches instructions from on-chip ROM 4 Kbytes. The

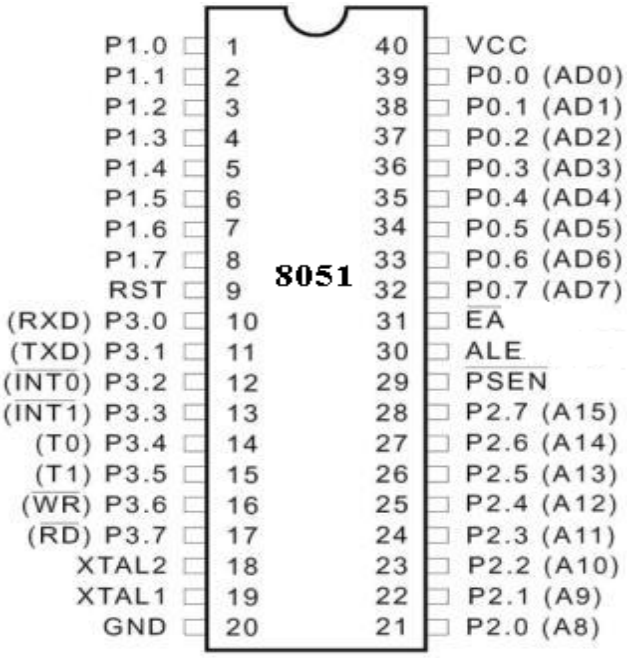
address of on-chip ROM is 0000H to 0FFFH. For address 1000H to FFFFH, it fetches instructions from external ROM. In this mode, 60 Kbytes of external ROM can be interfaced with 8051 microcontroller as shown in Fig. (a).

The data can be stored both in internal and 64 Kbytes of external data memory as shown in Fig. (b).



(a) *On-chip and off-chip program Rom,* **(b)** *On-chip and off-chip data memory*

Pin Diagram of 8051 Microcontroller: -



Pin description of 8051 Microcontroller:-

Pins 1 – 8 (Port 1 Pins P1.0 – P1.7): - These pins can be used for the connection of I/O devices in

Single Chip Mode operation as well as in **Expanded Mode** of operation of 8051. It means these port pins can used as Input Output pins.

Pin 9 (RESET): -It is active high signal. If it is active then Microcontroller starts its operation from initial point, all internal registers are loaded with their default values, internal buses operates at tri-state and all IO ports can be declared as Input Ports.

Pins 10 – 17 (Port 3 Pins P3.0 – P3.7): -These pins can be used for the connection of I/O devices in **Single Chip Mode** operation. But in **Expanded Mode** of operation this port pins perform multiple functions; those are

a. Serial Communication Signals (RXD & TXD)

b. External Interrupt Request Input Signals ($\overline{INT0}$ $\overline{INT1}$)

c. Timers / Counters Input Signals (T0 & T1)

d. Control Signals ($\overline{RD}$&$\overline{WR}$**)**

Pin 10 (RXD): - By using this pin the microcontroller receives data from external world in serial form.

Pin 11 (TXD): - By using this pin the microcontroller send data to external world in serial form.

Pin 12& 13 ($\overline{INT0}$ **&** $\overline{INT1}$ **): -** By using these two signals external devices send their interrupt request signals to microcontroller**.**

Pin 14 (T0): - By using this pin clock pulses are applied to the Timer-0, when the timer can used as Counter.

Pin 15 (T1): -By using this pin clock pulses are applied to the Timer-1, when the timer can used as Counter.

Pins 16 & 17 ($\overline{RD}$**&**$\overline{WR}$**): -**These signals indicate which type operation is going to performed with the selected device by the microcontroller. i.e. either Read or Write.

Pins 18 & 19 (XTAL2 & XTAL1): -These pins are provided for to connect a resonant circuit or crystal

to form an oscillator.

Pin 20 (GND): - Ground**.**

Pins 21 – 28 (Port 2 Pins P2.0 – P2.7): - These pins can be used for the connection of I/O devices in **Single Chip Mode** operation. But in **Expanded Mode** of operation this port pins can be used as dedicated higher order address lines **(A15 – A8).**

Pin 29 ($\overline{PSEN}$): -Program Store Enable. This is an output pin and is connected to the OE pin of the external ROM. It means it is used as enable signal for the external program memory ROM.

Pin 30 (ALE): - Address Latch Enable. The ALE pin is used for de-multiplexing the address and data by connecting to the enable pin of the 74LS373 octal latch. If this signal is active then the information the multiplexed bus (AD7 – AD0) is address otherwise the information is data.

Pin 31 ($\overline{EA}$): - External Access: If this signal is active, the microcontroller is communicating with the external memories or external world.

Pins 32 – 39 (Port 0 Pins P0.0 – P0.7): -These pins can be used for the connection of I/O devices in **Single Chip Mode** operation. But in **Expanded Mode** of operation this port pins can be used as multiplexed address and data lines **(AD8 – AD0).**

Pin 40 (Vcc): - +5V power supply.

Addressing Modes of 8051 Microcontroller:-

An Addressing Mode indicates how the data is represented in the instruction. 8051 supports 6 types of Addressing Modes, those are

1. Immediate Addressing mode
2. Register Addressing Mode
3. Register Indirect Addressing Mode
4. Direct Addressing Mode
5. Indexed Addressing Mode
6. Implicit Addressing Mode

1.Immediate Addressing Mode: -In these addressing mode instructions the data is directly placed in the source operand field of the instruction, and a '#' symbol must prefix for the data.

Ex: - MOV A, #38H
ADD A, #67H

2. Register Addressing Mode: -In these addressing mode instructions the data is directly placed in the operand field of the instruction through a GPR (General Purpose Register).

EX: - MOV A, R6
ADD A, R5

3. Register Indirect Addressing Mode: -In these addressing mode instructions the address of the data is indirectly placed in the operand field of the instruction through a GPR (General Purpose Register) R0 or R1.

Ex: - MOV A, @R0
ADD A, @R1

4. Direct Addressing Mode: - In these addressing mode instructions the address of the data is directly placed in the operand field of the instruction. The address is the internal RAM or internal SFR (Special Function Register).

Ex: - MOV A, 20H
MOV R1, 70H

5. Indexed Addressing Mode: -In these addressing mode instructions the address of the data is indirectly placed in the operand field of the instruction through combination 'A' register PC or DPTR.

Ex: - MOVC A, @A+DPTR
MOVC A, @A+PC

6.Implied Addressing Mode: -In this addressing mode instruction the operand is implicitly represented in the operation code of the instruction.

Ex: - RET, RETI

Instruction Set of 8051 Microcontroller: -

All instructions of 8051 can be divided in to four different groups, those are

1. Data Transfer Instructions
2. Arithmetic Instructions
3. Logical Instructions
4. Branching Instructions

1. Data Transfer Instructions: -Data transfer instructions are also called as Data Movement Instructions or Data Copying Instructions. Data transfer instruction execution doesn't effect on the Mathematical flags in PSW. In this group of instructions the Xerox copy of data is transferred from source to destination, because of this after execution of the instruction the content of source is equal to the content of destination. In this microcontroller data transfer instruction three types.

A. Move Related Instructions
B. Stack Related Instructions
C. Exchange Related Instructions

A. Move Related Instructions: - There are three types move related instructions.

i. MOV ii. MOVX iii. MOVC

i. MOV Instruction:- This instruction is used for onboard data transfers in 8051 Microcontroller.

Ex: - MOV R1, #68H
MOV A, #88H
MOV A, R0
MOV DPTR, #0896H

ii. MOVX Instruction: - This instruction is used for data transfers between external RAM and Microcontroller through 'A' register. This instruction only supports Register Indirect Addressing Mode.

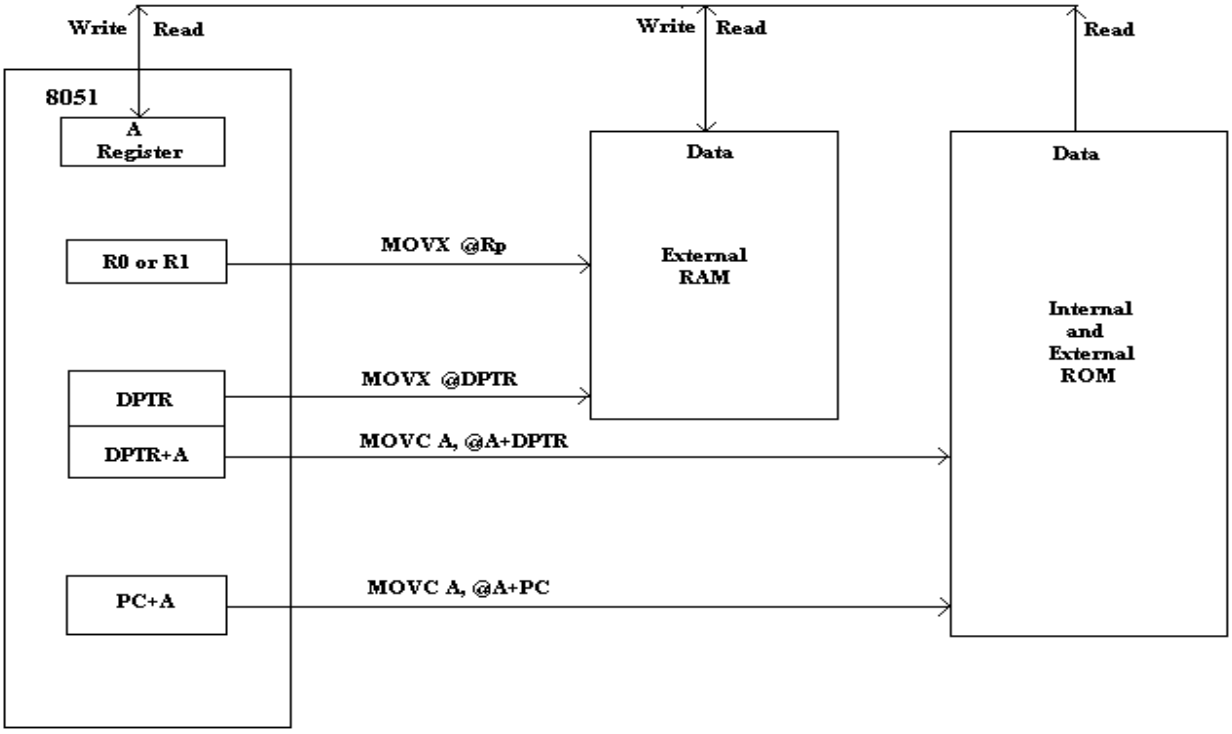
Ex: - MOVX A, @Rp
MOVX A, @DPTR MOVX @Rp, A
MOVX @DPTR, A

iii. MOVC Instruction: - This instruction is used for data transfers between internal or external ROM and the Microcontroller through 'A' register. This instruction only supports Indexed Addressing Mode it is one type of Register Addressing Mode.

Ex: - MOVC A, @A+DPTR

MOVC A, @A+PC

External Addressing Using MOVX and MOVC Instructions: -



B. Stack Related Instructions: -There are two instructions are there in this group. These instructions are used for data transfer between the internal RAM and the specified Direct Address in the instruction. The internal RAM can be used as Stack. The instructions are

i. PUSH ii. POP

i.PUSH Instruction: - This instruction is used for transferring the data from specified Direct Address into Top of the Stack. For this instruction execution the SP content is increment by '1'. In this instruction execution the internal operations are performed in the following order. It support only Direct Addressing Mode

1. SP content increment by '1'

2. Specified Direct Address content is pushed on to SP specified location in the Stack.

Ex: - PUSH 76H

ii. POP Instruction: -This instruction is used for transferring the data from Top of the Stack to specified Direct Address. For this instruction execution the SP content is decrement by '1'. In this instruction execution the internal operations are performed in the following order. It supports only Indirect Addressing Mode.

1. SP specified location in the Stack content is copied to Specified Direct Address in instruction.
2. SP content decrement by '1'

Ex: - POP 86H

C. Exchange Related Instructions: -In this group there are two instructions. In these instructions one operand is accumulator register 'A'. These instructions don't support Immediate Addressing Mode.

i. XCH ii. XCHD

i. XCH Instruction: - This instruction is used for data exchanging between Accumulator register 'A' and the specified other operand in the instruction. Ex: - XCH A, R0 XCH A, @R1
XCH A, 46H

ii. XCHD Instruction: -This instruction is used for Least Significant Nibbles data exchanging between Accumulator register 'A' and the specified other operand in the instruction.

Ex: - XCHD A, @R1

2. Arithmetic Instructions: -The microcontroller 8051 supports the following arithmetic operations. If any arithmetic instruction is executed by controller based on the result the mathematical flags are modified. For these arithmetic instructions one operand must be the content of accumulator 'A'and after completion of operation the result is stored in register 'A'.

i. Addition ii. Subtraction iii. Multiplication iv. Division
v. Increment vi. Decrement vii. Decimal Adjust

i. Addition Instructions: -There are two instructions for to perform addition operation. Those are

a. ADD b. ADDC

a. ADD Instruction: -This instruction is used to perform addition between Accumulator and specified another operand in the instruction. It supports all addressing modes.

Ex: - ADD A, R0 ADD A, #24H ADD A, @R0 ADD A, 68H

b. ADDC Instruction: -This instruction is used to perform addition between Accumulator specified another operand in the instruction and previous operation generated carry. It supports all addressing modes.

Ex: - ADDC A, R0 ADDC A, #24H
ADDC A, @R0 ADDC A, 68H

ii. Subtraction Instruction: -Only one instruction is there for performing the subtraction operation. That is subtraction with borrow 'SUBB'. For this Accumulator content is Minuend.

SUBB Instruction: -This instruction is used to perform subtraction between Accumulator specified another operand in the instruction and previous operation generated carry. It supports all addressing modes.

Ex: - SUBB A, R0 SUBB A, #24H SUBB A, @R0 SUBB A, 68H

iii. Multiplication Instruction: -Only one instruction is there for performing the Multiplication operation. This instruction syntax is fixed, and it support only two 8-bits numbers Multiplication.

MUL Instruction: - This instruction is used to perform Multiplication between 'A' register and 'B' register. The result of the Multiplication is stored in B & A registers. MSByte is stored in 'B' register and LSByte is stored in 'A' register. Ex: - MUL AB

iv. Division Instruction: -Only one instruction is there for performing the Division operation. This instruction syntax is fixed, and it support only two 8-bits numbers Division.

DIV Instruction: -- This instruction is used to perform Division between 'A' register and 'B' register. The result of the Division is stored in A & B registers. Quotient is stored in 'A' register and Remainder is stored in 'B' register.

Ex: - DIV AB

v. Increment Instruction: - Only one instruction is there for performing the Increment operation. It supports all types of addressing modes except Immediate addressing Mode. No mathematical flag is affected for this instruction execution.

INC Instruction: -This instruction is used to increment the specified operand content by '1'. The operand may not an immediate data.

Ex: - INC R0 INC 70H INC @R0 INC DPTR INC A

vi. Decrement Instruction: -Only one instruction is there for performing the decrement operation. It supports all types of addressing modes except Immediate addressing Mode. No mathematical flag is affected for this instruction execution.

DEC Instruction: -This instruction is used to decrement the specified operand content by '1'. The operand may not an immediate data.

Ex: - DEC R0 DEC 70H DEC @R0 DEC DPTR DEC A

vii. Decimal Adjust Instruction: -Only one instruction is there for performing the decimal adjust operation. It supports only register addressing mode. For this Instruction the operand is Accumulator register 'A' only.

DA A Instruction: -This instruction is used to decimal adjust the accumulator content, it means convert the binary content of register 'A' into BCD form.

Ex: - DA A

3. Logical Instructions: -The 8051 support two types of logical operations based on the size of applied data. Those are Byte level Logical Instructions and Bit Level Logical Instructions.
8051 support the following logical operations,

i. AND ii. OR iii. XOR iv. NOT

A. Byte level Logical Instructions: -

i. AND Instructions: -Only one instruction is there for performing the logical AND operation. It supports all addressing modes.

ANL Instruction: - This instruction is used to perform Logical AND operation between the source operand and destination operands. The destination operand is either 'A' register or

an Address but the source is a register, or an immediate data, or an address.

Ex: - ANL A, R0 ANL 70H, A ANL A, #80H ANL 76H, #74H ANL A, @R0

ii. OR Instruction: -Only one instruction is there for performing the logical OR operation. It supports all addressing modes.

ORL Instruction: - This instruction is used to perform Logical OR operation between the source operand and destination operands. The destination operand is either 'A' register or an Address but the source is a register, or an immediate data, or an address.
Ex: - ORL A, R0 ORL 70H, A ORL A, #80H ORL 76H, #74H ORL A, @R0

iii. XOR Instruction: -Only one instruction is there for performing the logical XOR operation. It supports all addressing modes.

XRL Instruction: - This instruction is used to perform Logical XOR operation between the source operand and destination operands. The destination operand is either 'A' register or an Address but the source is a register, or an immediate data, or an address.

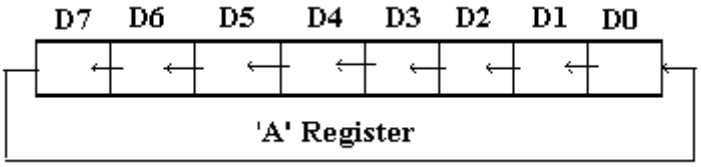
Ex: - XRL A, R0 XRL 70H, A XRL A, #80H XRL 76H, #74H XRL A, @R0

iv. NOT Instruction: -Only one instruction is there for performing the logical NOT operation. For this instruction the operand is Accumulator register 'A'.

CPL Instruction: - This instruction is used to complement the content of Accumulator register. Ex: - CPL A
Rotation Related Instructions: -All these instructions are designed based on the 'A' register is an operand of the instruction, and all these instructions support register addressing mode only.

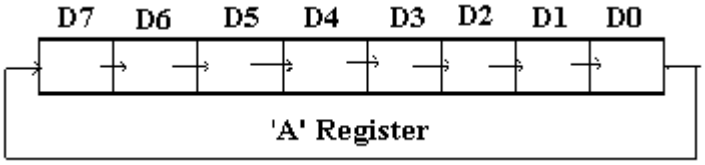
RL Instruction: - This instruction is used to rotate the accumulator content bit by bit to left side. MSbit is copied into LSbit.

Ex: - RL A



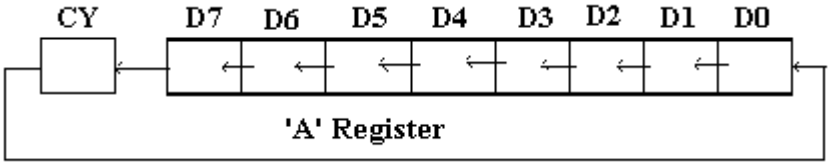
RR Instruction: -This instruction is used to rotate the accumulator content bit by bit to right side. LSbit is copied into MSbit.

Ex: - RR A



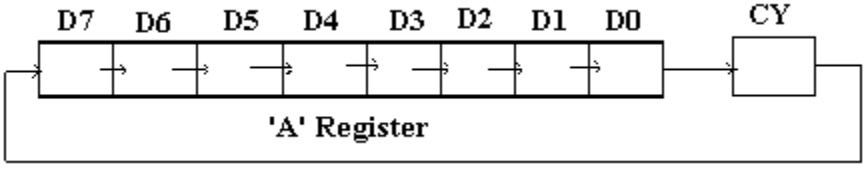
RLC Instruction: -This instruction is used to rotate the accumulator content bit by bit to left side through Carry. MSbit is copiedto Carry flag position and Carry bit is copied to LSbit position.

Ex: - RLC A



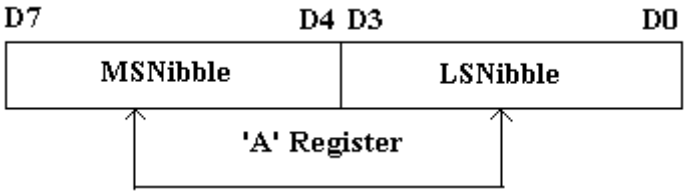
RRC Instruction: -This instruction is used to rotate the accumulator content bit by bit to right side through Carry. LSbit is copiedto Carry flag position and Carry bit is copied to MSbit position.

Ex: - RRC A



SWAP Instruction: - This instruction is used to exchange the nibbles of Accumulator register.

Ex: - SWAP A



B. Bit level Logical Instructions: -These instructions perform operations on any Bit Addressable RAM or SFR bits. The Carry flag in the PSW can be used as destination for maximum number of instructions.

We have bit related instructions for the operations like Logical AND, OR, Complement, Clear, Set and Move.

AND Related Instructions: -

i. ANL C, bInstruction: - This instruction is used logical AND between Carry Flag (CY) and the specified direct addressed bit in the instruction. Ex: - ANL C, 64H.

ii. ANL C, $\overline{b}$ Instruction: -This instruction is used logical AND between Carry Flag (CY) and the complement of specified direct addressed bit in the instruction. Ex: - ANL C, $\overline{64H}$

OR Related Instructions: -

i. ORL C, bInstruction: - This instruction is used logical OR between Carry Flag (CY) and the specified direct addressed bit in the instruction. Ex: - ORL C, 64H.

ii. ORL C, $\overline{b}$ Instruction: -This instruction is used logical OR between Carry Flag (CY) and the complement of specified direct addressed bit in the instruction. Ex: - ORL C, $\overline{64H}$

Complement Related Instructions: -

i. CPL C Instruction: - This instruction is used to Complement the Carry flag content.

ii. CPL b Instruction: - This instruction is used to Complement the specified direct

addressed bit content.

Clear Related Instructions: -

i. CLR C Instruction: - This instruction is used to Clear the Carry flag content.

ii. CLR b Instruction: - This instruction is used to Clear the specified direct addressed bit content.

Set Related Instructions: -

i. SETB C Instruction: - This instruction is used to Set the Carry flag content.

ii. SETB b Instruction: - This instruction is used to Set the specified direct addressed bit content.

Move Related Instructions: -

i. MOV C, b Instruction: - This instruction is used Move the content of the specified direct addressed bit in the instruction to the Carry Flag (CY). Ex: - MOV C, 64H.

i. MOV b, C Instruction: - This instruction is used Move the content of the Carry Flag (CY) to the specified direct addressed bit in the instruction. Ex: - MOV 64H, C.

4. Branching Instructions: - These instructions are also called as Transfer of control Instructions, or Program flow control instructions. By using these instructions the program flow control is transferred from one location to another location conditionally or unconditionally. Basically these are two types of instructions

A. Unconditional Branching Instructions

B. Conditional Branching Instructions

A. Unconditional Branching Instructions: -These instructions are used to change the program flow control from one location to another location without taking any condition from the flags of PSW or Accumulator content. In this mainly there are two types of instructions.

a. Jump Related Instructions,

b. Call and Return Related Instructions.

a. Jump Related Instructions: -These instructions are used to change the program execution from one location to another location unconditionally. In this different instructions are available in 8051 Microcontroller instruction set. Those are

i. SJMP ii. AJMP iii. LJMP iv. JMP

i. SJMP Instruction: -It is a Short Jump Instruction. By using this jump instruction execution is moved to the previous locations or next locations. The destination must be in the range -128 to
+127 from the current instruction. For this instruction the operand is an 8-bit relative address. The operand address is added to or subtracted from the current content of PC. This instruction is also used as termination instruction in 8051 programs.

Ex: - SJMP 8-bit Relative address

ii. AJMP Instruction: -It is an Absolute Jump Instruction. By using this jump instruction execution is moved to only next locations. The destination must be within 2K locations from the current instruction. For this instruction the operand is an 11-bit address. The operand address is added to the current content of PC.

Ex: - AJMP 11-bit address

iii. LJMP Instruction: -It is a Long Jump Instruction. By using this jump instruction execution is moved to only next locations. The destination is any location in the 64K locations of the memory. For this instruction the operand is a 16-bit address. The operand address is added to the current content of PC.

Ex: - LJMP 16-bit address

iv. JMP Instruction: -It is a simple Jump Instruction. By using this jump instruction execution is moved to the location address is provided by the combination of ‘A’ register and DPTR register.

Ex: - JMP @A + DPTR

b. Call and Return Related Instructions: - By using the Call and Return instructions the subprogram is called into the main program. Call instruction is used to change the program execution from the main program to subprogram, and by using Ret instruction the program execution is shifted from the subprogram to the main program. There are two Call related instructions, those are

i. ACALL ii. LCALL

i. ACALL Instruction: -By using this instruction to call a subprogram into the main program. But the subprogram must be within the 2K locations. In another way the subprogram and the main program must be in the same page and the page size is 2KB.

Ex: - ACALL 11-bit address

ii. LCALL Instruction: -By using this instruction to call a subprogram into the main program. The subprogram is in any location in the 64K locations. Ex: - LCALL 16-bit address

There are two Return Related Instructions, those are i. RET ii. RETI

i. RET Instruction: -By using this instruction the program execution is shift from the subprogram to the main program, and this instruction is used as last instruction in the subprogram. This instruction is comes under implicit addressing mode. Ex: - RET

ii. RETI Instruction: -By using this instruction the program execution is shift from theInterrupt Service Routine (ISR) to the main program, and this instruction is used as last instruction in the ISR. This instruction also comes under implicit addressing mode.

Ex: - RETI

B. Conditional Branching Instructions: -These instructions are used to change the program flow control from one location to another location with taking condition from the flags of PSW or Accumulator content. All Conditional Jump instructions are Relative Jump instructions or Short Jump Instructions. In this mainly there are three types of instructions.

a. Conditional Jump Instructions

b. Comparison Related Instructions

c. Decrement Related Instructions

a. Conditional Jump Instructions: -These Instructions are designed based on the conditions of
Carry flag in the PSW, Accumulator content and the specified bit.

i. JC Instruction: -By using this instruction the program execution is shift from one location to other location only if the CY = '1'. Otherwise the instruction is ignored.

Ex: - JC 8-bit rel add

ii. JNC Instruction: -By using this instruction the program execution is shift from one location to other location only if the CY = '0'. Otherwise the instruction is ignored.

Ex: - JNC 8-bit rel add

iii. JZ Instruction: -By using this instruction the program execution is shift from one location to other location only if the Accumulator (A) content = '0'. Otherwise the instruction is ignored.

Ex: - JZ 8-bit rel add

iv. JNZ Instruction: -By using this instruction the program execution is shift from one location to other location only if the Accumulator (A) content not equal to'0'. Otherwise the instruction is ignored.
Ex: - JNZ 8-bit rel add

v. JB instruction: -By using this instruction the program execution is shift from one location to other location only if the Specified bit = '1'. Otherwise the instruction is ignored.

Ex: - JB b, 8-bit rel add where 'b' indicates the Direct Address of the Bit.

vi. JNB instruction: -By using this instruction the program execution is shift from one location to other location only if the Specified bit = '0'. Otherwise the instruction is ignored.

Ex: - JNB b, 8-bit rel add where 'b' indicates the Direct Address of the Bit.

vii. JBC instruction: -By using this instruction the program execution is shift from one location to other location only if the Specified bit = '1', and clear that bit also. Otherwise the instruction
is ignored.

Ex: - JBC b, 8-bit rel add where 'b' indicates the Direct Address of the Bit.

b. Comparison Related Instruction: -This instruction are used compare the accumulator content with any memory location content or an immediate data or a memory location content is compare with an immediate data or any GPR content is compare with an immediate data. After comparison if both operands are not equal then jump to the specified relative address. Only this instruction has three operand fields.

Ex:- CJNE A, #70H, CJNE A, 80H, CJNE Rn, #78H, CJNE @Ri, #32H
Where Rn = any GPR (R0 to R7) and Ri = either R0 or R1

c. Decrement Related Instructions: -This instruction is used to Decrement the specified GPR content or the Direct Address content by '1' and that is not equal to zero then jump to the specified relative address.

Ex: - DJNZ Rn, 8-bit rel add, DJNZ 64H, 8-bit rel add

Sample Programs of 8051 Micro-Controller: -

1. **Writ an 8051 ALP to save the status of bits P1.1 and P1.2 on RAM bit locations 08H and 09H respectively**.

```
MOV C, P1.1
MOV 08H, C
MOV C, P1.2
MOV 09H, C
UP: SJMP UP
```

2. **Writ an 8051 ALP to find the sum of values 79H, 0F5H, and 0E2H. Put the sum in register R0 (Lower order Byte) and R1 (Higher order Byte).**

```
MOV A, 00H
MOV R1, A
MOV R0, A
ADD A, #79H
JNC NEXT1
INC R1
NEXT1: ADD A, #0F5H
JNC NEXT 2
INC R1
NEXT 2: ADD A, #0E2H
JNC NEXT3
INC R1
NEXT 3: MOV R0, A
HERE: SJMP HERE
```

3. **Writ an 8051 ALP to add '2' to the accumulator by 600H times.**

```
MOV A, #77H
MOV R2, #10H
NEXT: MOV R3, #60H
AGAIN: ADD A, #02H
DJNZ R3, AGAIN
DJNZ R2, NEXT
STOP: SJMP STOP
```

4. Writ an 8051 ALP to toggle all the bits of Port 1 by sending to it the values 66H and 0BBH continuously. Put a delay in between each issuing of data to port.

```
UP: MOV A, #66H
MOV P1, A
LCALL DELAY
MOV A, #0BBH
MOV P1, A
LCALL DELAY
SJMP UP
DELAY: MOV R3, #0FFH
AGAIN: DJNZ R3, AGAIN
 RET
```

5. Writ an 8051 ALP to find the number of 1's in a 0FFH.

```
MOV R2, #00H
MOV R3, #08H
MOV A, #0FFH
REPEAT: RLC A
JNC NEXT
INC R2
NEXT: DJNZ R3, REPEAT
HERE: SJMP HERE
```

6. Writ an 8051 ALP to find the Largest / Maximum number in the array.

```
        MOV R6, #08H; COUNT
        MOV R1, #40H; STARTING ADDRESS
        MOV A, #00H
UP:     MOV B, @R1
        CJNE A, B, DOWN
DOWN:   JNC SKIP
        MOV A, B
```

```
SKIP:   INC R1
        DJNZ R6, UP
        MOV 60H, A; STORE RESULT
        END
```

7. Put the number 34H in registers R5, R6 and R7 using at least three different methods?

Ans:

```
        CASE1:
        MOV R5, #34H
        MOV R6, #34H
        MOV R7, #34H

        CASE2:
        MOV A, #34H
        MOV R5, A
        MOV R6, A
        MOV R7, A

        CASE3:
        MOV 20H, #34H
        MOV R5, 20H
        MOV R6, 20H
        MOV R7, 20H
```

9. Write a program to multiply the data in RAM location 22h by the data in RAM location 15h; Put the result in RAM location 19h (low byte) and 1A h (high byte).

```
        MOV A, 22H

        MOV B, 15H

        MUL AB

        MOV 19H, A

        MOV 1AH, B

        END
```

10. Write a program in assembly language that accepts a number in register and complement least significant bit without affecting other bits in the register?

```
MOV A, 40H
XRL A, #01H
MOV 40H, A
END
```