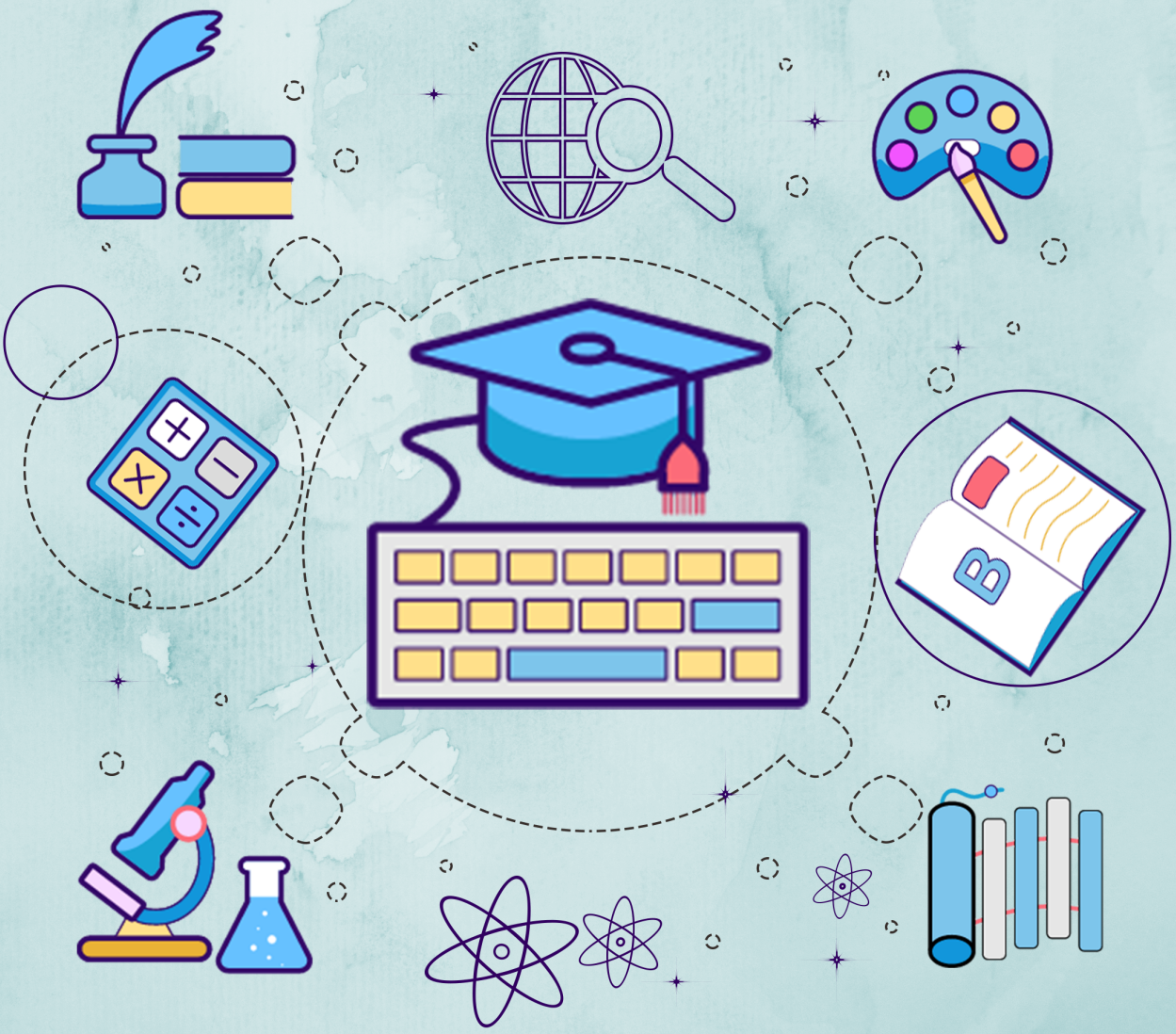
APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

KeralaNotes



SYLLABUS | STUDY MATERIALS | TEXTBOOK

PDF | SOLVED QUESTION PAPERS

www.keralanotes.com

KeralaNotes

KTU STUDY MATERIALS

MICROPROCESSORS AND MICROCONTROLLERS

CST 307

Module 5

Related Link :

- KTU S5 STUDY MATERIALS
- KTU S5 NOTES
- KTU S5 SYLLABUS
- KTU S5 TEXTBOOK PDF
- KTU S5 PREVIOUS YEAR SOLVED QUESTION PAPER

Microcontrollers are programmable devices which can access instructions, decode them and execute to perform particular task. They are also able to interface with other hardware devices.

Features of 8051 MC

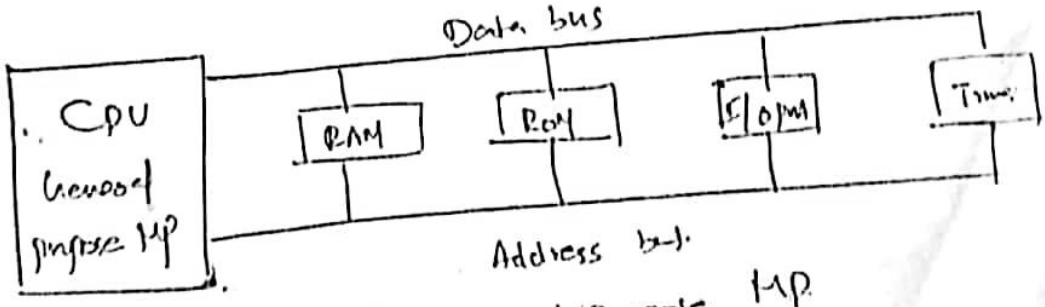
→ 8 bit CPU with reg A and B
→ 16 bit Program counter (P.C) and Data pointer (DPTR)
→ 8 bit PSW
→ 8 bit Stack pointer (SP)
→ Embedded ROM
→ Internal RAM of 128 bytes.
- Four register banks
- 16 bytes, which addressed at bit level
- .

→ 32 I/O pins arranged as four 8 bit ports : P0-P3
→ Two 16 bit timer/counters : T0 and T1
→ Oscillator and clock circuit

Comparison : MP Vs MC

MP's are also programmable devices but they contain no RAM, ROM, I/O ports on the chip itself. For this reason, they are commonly referred as general purpose MP. Using this types of MP, it is necessary to add RAM, ROM, I/O ports, timers externally to make functional. The addition of these components make the system more bulky and expensive.

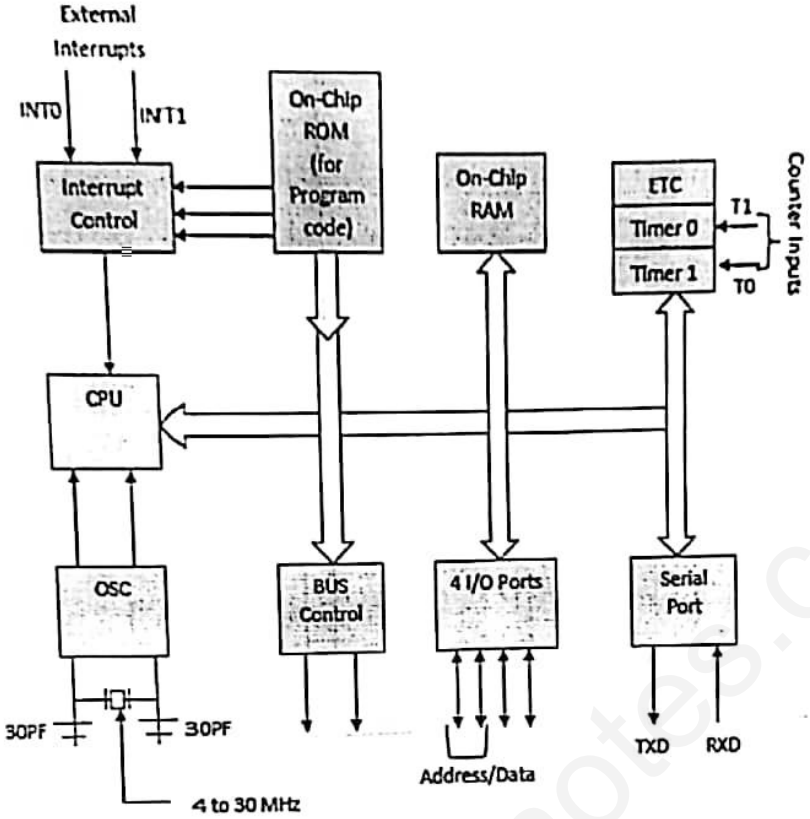
A MC has a CPU, in addition to this fixed amount of RAM, ROM, I/O ports and timer on a single chip. i.e a processor, RAM, ROM, I/O ports, timer all are embedded on single chip. The designer cannot add an external RAM, ROM, I/O ports or timer to it.



(a) General purpose MP

CPU	RAM	ROM
I/O	Timer	Serial

Architecture or Block diagram 8051 (2)



Program counter and Data pointer.

→ 8051 contains two 16 bit registers.
→ PC and DPTR.
→ Each is used to hold the address of a byte in memory.
→ Program instruction bytes are fetched from locations in memory that are addressed by the P.C.
→ The P.C is the only register that does not have an internal address.

④ → DPTR made up of two 8 bit regis

- DPH
- DPL

→ DPTR doesn't have a single internal address

A and B CPU Registers.

→ 8051 contains 34 general purpose or working register.

→ Two of these, A and B hold results of many instructions, ie math and logical operation of CPU

→ The other 32 are arranged as a part of internal RAM in four banks, B0–B3 of eight registers.

→ A (Accumulator) register is the most versatile.

→ Used for many operations such as addition, subtraction, boolean operations.

→ The reg B is used with A register for multiplication and division operations.

→ A is also used for data transfer in 8051.

I/O ports.

→ There are total of 4 ports for I/O operations.

→ The four ports are P0, P1, P2, P3 each use 8 pins, make the 8 bit ports.

→ When '0' is written to a port, it becomes an o/p port.

→ To use as an i/p port '1' must be sent to the port.

Interrupt.

→ A single Mc can control several devices.

→ One of the ways to do that is through interrupts.

→ In interrupt method, if any device need its service, the device sends an interrupt signal.

→ On receiving this MC stops currently executing program and store them in stack.

→ Then it execute the interrupt service routine to handle the work of the other device.

Serial ports

→ Serial communication is a slow process.

→ It is used to transmit and receive simultaneously.

→ Port 3 can be used as I/O port.

→ 0th bit of port 3 used as Serial I/P line RXD

→ 1st bit of port 3 used as Serial O/P line TXD.

Timer / Counter.

→ It has 2-16 bit timer/counter

→ Each counter is used to count External clk pulse. other wise count external pulse.

Flags and program status word.

→ Flags are 1 bit register provided to store the results of certain program instructions

→ 8051 has 4 math flags and three general purpose flags can be set to 1 or 0.

→ Math flags → C, AC, OV, P

→ Program status word is as show below.

7	6	5	4	3	2	1	0
CY	AC	F0	RS1	RS0	V	–	P

CY — carry flag

AC — Auxiliary Carry flag, Used for BCD operands

F0 → User flag 0

RS1 → Register bank select bit 1

RS0 → " " " " 0

OV → overflow flag, Used for arithmetic operations

– → Reserved for future use

P → parity flag.

RS1	RS0	
0	0	Bank 0
0	1	Bank 1
1	0	Bank 2
1	1	Bank 3

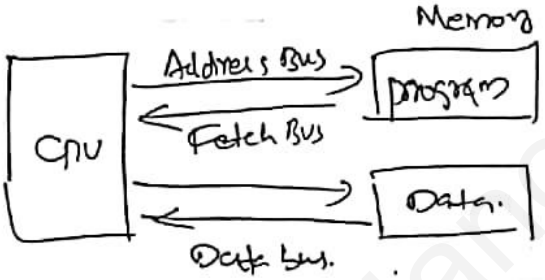
(6) Memory organisation.

(1) Internal memory.

→ A functioning computer must have memory for program code bytes, commonly ROM and RAM memory.

→ Additional memory can be added externally using suitable circuits.

→ 8051 has Harvard architecture.

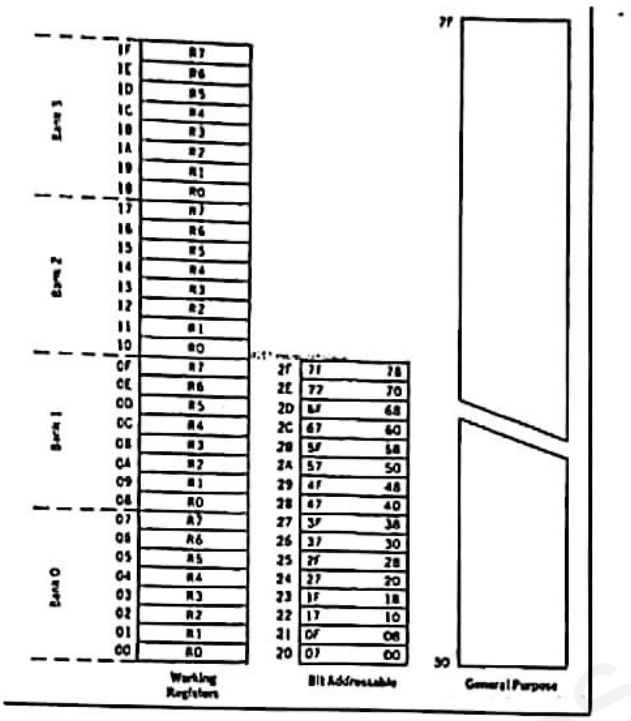


→ It uses same address, to different memory locations

(a) Internal RAM.

→ Internal RAM is organised into 3 distinct areas.

1). 32 bytes from address 00 H to 1F H that make up 32 working registers organised as four banks of 8 registers each. The four banks numbered as B0 to B3



and are made up of 8 registers named R0 to R7. Bit RS0 and RS1 determine which bank of registers is currently use.

2) A bit addressable area of 16 bytes occupies RAM byte addresses 20H to 2FH forming a total 128 addressable bits. It is specified by its address at 00H to 7FH.

3) A general purpose RAM area above the bit area, from 30H to 7FH, addressable as bytes.

2) Internal RAM.

→ 8051 organized data memory and program code memory can be in two entirely different physical memory entities.

→ It occupies code address space 0000 H to 0FFF H

→ PC occupies address 0000 H to FFFF H

→ Program address higher than 0FFF H, which exceed the internal ROM capacity, will cause the 8051 automatically fetch code bytes from external program memory.

→ Code bytes also be fetched exclusively from an external memory, by connecting the External access pin (EA pin 31 of 8051)

The Stack and Stack pointer

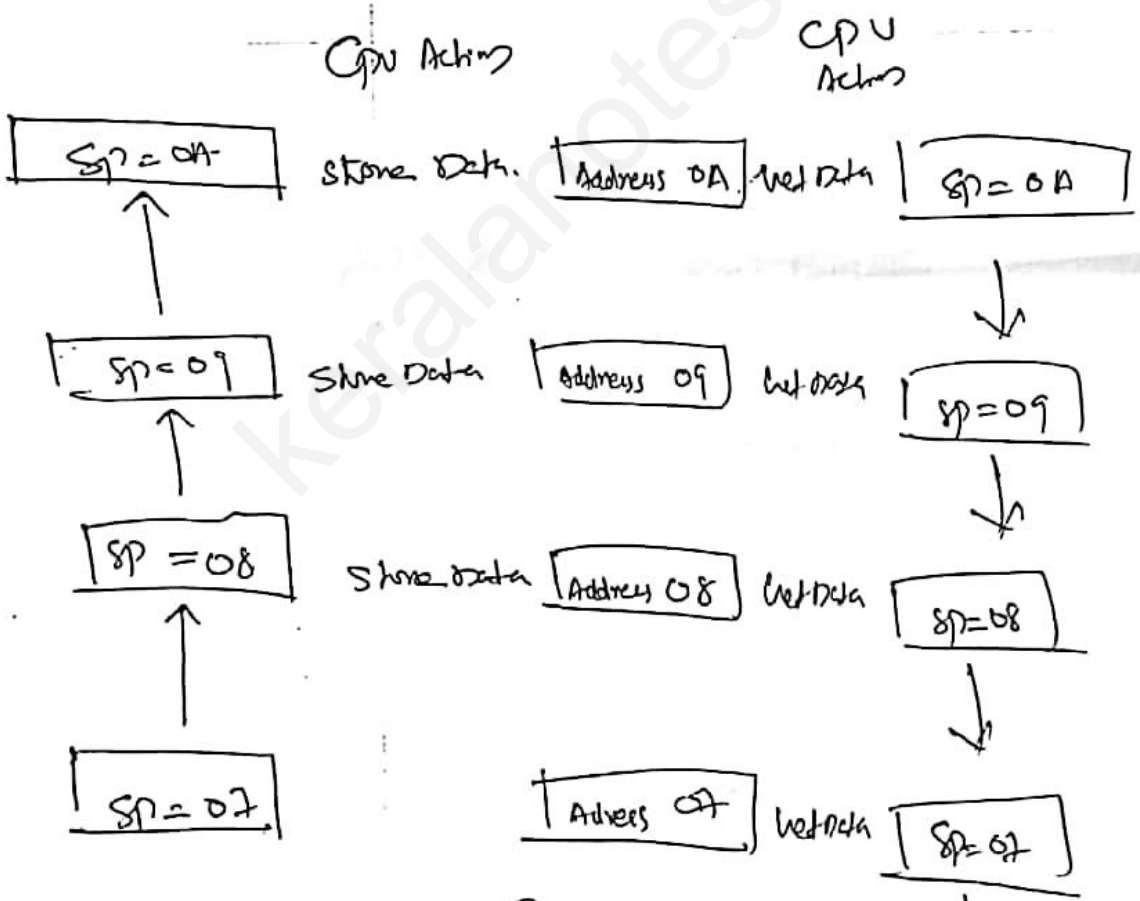
→ Stack refers to an area of internal RAM that is used to store data and retrieve data quickly

The 8 bit stack pointer is used by 8051 to hold an internal RAM address that is called top of the stack.

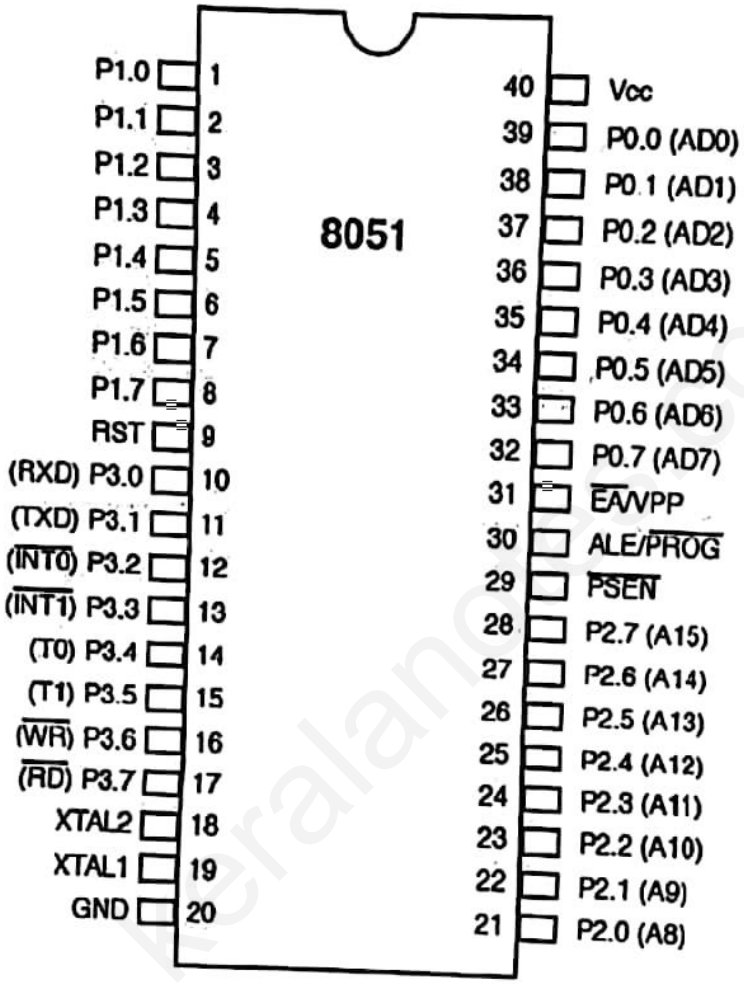
→ When data is placed on the stack, SP increments before storing data on the stack.

→ As data is retrieved from the stack, the byte is read from stack, SP decrements

→ Stack operations as follows.



The pin diagram of 8051 microcontroller



- **Pins 1 to 8** – These pins are known as Port 1. This port doesn't serve any other functions. It is internally pulled up, bi-directional I/O port.
- **Pin 9** – It is a RESET pin, which is used to reset the microcontroller to its initial values.
- **Pins 10 to 17** – These pins are known as Port 3. This port serves some functions like interrupts, timer input, control signals, serial communication signals RxD and TxD, etc.
- **Pins 18 & 19** – These pins are used for interfacing an external crystal to get the system clock.
- **Pin 20** – This pin provides the power supply to the circuit.

- **Pins 21 to 28** – These pins are known as Port 2. It serves as I/O port. Higher order address bus signals are also multiplexed using this port.
- **Pin 29** – This is PSEN pin which stands for Program Store Enable. It is used to read a signal from the external program memory.
- **Pin 30** – This is EA pin which stands for External Access input. It is used to enable/disable the external memory interfacing.
- **Pin 31** – This is ALE pin which stands for Address Latch Enable. It is used to demultiplex the address-data signal of port.
- **Pins 32 to 39** – These pins are known as Port 0. It serves as I/O port. Lower order address and data bus signals are multiplexed using this port.
- **Pin 40** – This pin is used to provide power supply to the circuit.

Addressing Modes.

→ The way the data sources and destinations addrels are specified in the mnemonic that moves determines the addressing mode.

(1) Immediate addressing mode.

→ The data source is immediately available as a part of the instruction itself.

→ When 8051 executes an immediate data move, the P.C is automatically incremented.

→ The mnemonic for immediate data is pound sign (#)

→ Three mnemonics can copy immediate numbers from the opcode into registers R0–R7, A and DPTR.

MOV Rr, #n — Copy the 8 bit number n into register Rr

MOV A, #n — Copy the 8 bit number n into register A (Accumulator)

MOV DPTR, #nn — Copy the 16 bit number nn into DPTR register.

Eg.

MOV R0, #:00H put the immediate 8 bit number 00 H to reg. R0.

MOV R1, #:01 H " 01H to reg. R1

MOV A, #:0AA h put the immediate 8 bit no. AA H to register A.

MOV DPTR, #1234 H.

(2) Register Addressing Mode.

→ Register names may be used as part of the opcode mnemonic as source and destination data.

→ Registers A, DPTR and R0–R7 are used as part of Mnemonic.

Register to Register moves as follows.

MOV A, R5 . copy data from reg R5 to A

MOV R7, A " " " A to reg. R0

ect addressing mode.

→ All 128 bytes of internal RAM and SFR may be addressed directly using the single-byte address assigned to each RAM location.

→ Internal RAM addresses uses ~~addr~~ from 00H to 7FH.

→ SFR addresses exist from 80H to FFH.

Mov A, addr — Copy data from direct address to reg. A

Mov addr, A — Copy data from reg. A to direct address

Mov Rr, addr

Mov addr, Rr

Mov add, #n

Mov addr1, addr2 — Copy data from direct address 1 to address 2.

Eg:

Mov R0, #3AH
Mov 3AH, #2BH

Copy data from Internal RAM location 3AH into R0.

2BH data is stored in Ram location 3AH

MOV R0, 12H. Copy data from RAM location 12H to regis

(4) Indirect Addressing Mode.

→ The indirect addressing mode uses a register to hold the actual address that will finally be used in the data move.

→ Indirect addressing for MOV opcodes uses R0 or R1 registers often called as data pointer

→ Used to hold the address of one of the RAM locations in RAM from address 00H to 7FH

MOV @Rp, #n — Copy the immediate data n to address in Rp

MOV @Rp, addr — Copy the contents of address to address in Rp

MOV @Rp, A. — Copy the data in A to the address in Rp.

MOV A, @Rp

MOV addr, @Rp

Instruction Set - 8051

a) Data movement operations

b) Arithmetic operations

c) Logical operations.

d) Jump and Call Instructions.

a) Data movement.

→ This is used to move data from source to destination.

→ 28 different mnemonics are used.

→ MOV is used for data transfer with 8051

→ The memory is divided into 4 physical parts.

1) Internal RAM
2) Internal SFR
3) External ROM
4) Internal and external ROM

→ The ~~four~~ five different types of opcodes used for movement

1. MOV : MOV destination, source → Internal data move
2. MOVX : MOVX dest^n, source → External data move
3. MOVC : MOVC dest^n, source → ROM data move
4. PUSH & POP.
5. XCH

→ Destination is mentioned first followed by source

copied/moved to destn.

1. Internal Data move.

MOV A,#n — Copy/move the 8 bit number n into Accumulator

MOV A,Rx — Copy data from register Rx to reg. A

MOV A, addr — Copy data from direct address to reg.

MOV addr, A

MOV addr, #n

MOV addr1, addr2 — Copy data from direct address 2 to direct address 1.

MOV @Rp, #n — Copy data n to the address of Rp.

2) External Datamove.

MOVX A, @Rp — Copy contents of external address in Rp to reg. A

MOVX A, @DPTR — Copy contents of external address in DPTR to reg. A.

MOVX @Rp, A.

MOVX @DPTR, A.

MOVX A, @R1

3) ROM data move.

MOVC A, @A+DPTR — Copy contents of ROM where address is formed by adding A and DPTR to A.

MOVC A, @A+PC

~~MOV~~

PUSH & POP

→ These are used to move data from register to stack and vice versa.

PUSH → storing data to stack.

POP → Retrieving data from stack.

Mnemonic	Operation
PUSH addr	Increment SP, Copy data from address to internal RAM address contain SP.
POP addr	Copy data from internal RAM address contained in SP to address; decrement SP

5) Data Exchange operations.

→ Data is mutually transferred b/w source and destination. All exchanges use reg. A.

Mnemonic	Operation
XCH A, Rn	Exchange data b/w reg. Rn and A.
XCH A, addr	Exchange data b/w address and A.

5) Arithmetic operations

→ MC performs Mathematical operations also.

→ For that arithmetic operations are needed.

→ There are 24 opcodes grouped into following lines.

INC dest^n → Increment dest^n by

DEC dest^n → Decrement dest^n by 1.

ADD dest^n, source → Add contents of source with dest^n without carry

ADDC dest^n, source → // with carry

SUBB dest^n, source → Sub with borrow, the source from dest^n.

MUL AB → Multiply contents of reg A with reg. B.

DIV AB → Divide contents of reg. A by contents of reg B.
Quotient stored in A, remainder in reg B.

DAA → Decimal adjust accumulator (BCD additions)

Increment and Decrement.

INC A → Add 1 to reg A or accumulator.

INC Rr

INC addr

INC @Rp

INC DPTR → Add 1 to the 16 bit DPTR

Mnemonic Operation

(5)

Hexa
3A -
1
39

MOV A, #3A H ⇒ A = 3A H.

DEC A ⇒ A = 39 H

MOV R0, #15 H ⇒ R0 = 15 H.

MOV 15H, #12 H ⇒ Internal RAM addr ~~location~~ 15 H = 12.

INC @ R0 ⇒ Internal RAM address 15 H = 13

DEC 15 H ⇒ Internal RAM address 15H = 12

INC R0 ⇒ R0 = 16 H.

MOV 16H, A ⇒ Internal RAM address 16 H = 39H

INC @R0 ⇒ Internal RAM address 16 H = 3A

Addition

All addition is done with A register as destination of result

ADD A, #n → Add contents of A with number n and result is stored in A.

ADD A, Rx

ADD A, addr

ADDC A, #n → Add content of A with no. 'n' with carry, then sum is in A

ADDC A, addr

ADDC A, Rx

```
MOV A, #1Bh        A = 1Bh.
MOV R5, #0Ah       R5 = 0Ah.
ADD A, R5          A = 25h, C=0,
```

Subtraction.

Subtraction can be done by taking 2's complement of subtrahend and adding it to minuend.

```
SUBB A, #n   ⟶ Subtract immediate no. n and C-flag from A. Put result in A.
SUBB A, addr
SUBB A, Rr.
SUBB A, @Rp
```

Multiplication and Division

8 bit MUL and DIV can be performed by 8051. Reg A and B can be used. B is exclusively used for these operations. The two no's are stored in A and B. After operation result is stored in A and B.

Multiplication

A and B is used as source and destination. The OV flag will set if result is more than FF H.

MUL AB → Multiply A by B. put low byte in A and high byte in B. (6)
(No comma b/w A and B).

Division

Reg A and B used as source and destination. The number A is divided by B. Quotient stored in A and remainder in B.

DIV AB → Divide A by B. store quotient in A and remainder B.

(c) Logical Group Instructions

→ To perform logical operations AND, OR, NOT etc.

	Mnemonic	
AND →	AND	ie AND logical
OR →	ORL	
NOT →	CPL	
XOR →	XRL	

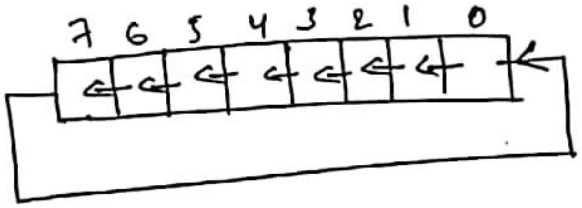
Eg.

ANL A, #n → And each bit of n with same bit of immediate no. n; put result in A.
ANL A, addr
ORL A, #n
CLR A → clear each bit of A.
CPL A → Complement each bit of A.

Rotate and SWAP.

→ Reg. A can be rotated on bit position to left or right with or without carry.

RLA ⇒ Rotate Accumulator left one bit position without carry

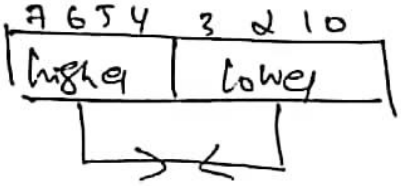


RLCA ⇒ Rotate Accumulator left one bit position with carry.

RRA ⇒ Rotate acc. right one bit position without carry

RRCA ⇒ " with carry.

SWAP A ⇒ Interchange lower nibble & higher nibble of A



Mnemonic	Operation result	A 5 1010 0101
MOV A, #0A5h	A = 1010 0101 b = A5h	
RR A	A = 1101 0010 b = D2h.	
RR A	A = 0110 1001 b = 69h	
RR A	A = 1011 0100 b = B4h.	
RR A	A = 0101 1010 b = 5Ah.	
SWAP A	A = 1010 0101 = A5h	

d) Jump and Call Instructions

1) Conditional Jump

JZ ⇒ Jump if A = 0

JNZ ⇒ Jump if A ≠ 0

DJNZ ⇒ Decrement and Jump if reg ≠ 0

eg. DJNZ R2, loop.

CJNE A, data ⇒ Jump if A ≠ data.

JC ⇒ Jump if CY = 1

JNC ⇒ Jump if CY = 0

JB ⇒ Jump if bit = 1

JNB ⇒ Jump if bit = 0.

2) Unconditional

JMP @A+DPTR ⇒ Jump to address formed by A + DPTR

NOP ⇒ Do nothing and go to next instruction.

3) Call Instructions

ACALL addr → Call subroutine located on same page (within 2K)

LCALL addr → Call subroutine located anywhere in pgm memory (within 64K)