MODULE-3
Microcontroller Architecture: Intel 8051

V.PRAKASH
Asst. Professor(Sr.), SENSE,
VIT Chennai

MODULE-3

Microcontroller Architecture: Intel 8051

Microcontroller 8051 - Organization and Architecture, RAM-ROM Organization, Machine Cycle, Instruction set: Addressing modes, Data Processing - Stack, Arithmetic, Logical; Branching – Unconditional and Conditional, Assembly programming.

INTRODUCTION TO MICROCONTROLLERS

INTRODUCTION TO MICROCONTROLLERS

WHAT IS MICROCONTROLLER?

- A microcontroller is a compact integrated circuit designed to govern a specific operation in an embedded system.
- A typical microcontroller includes a processor, memory and input/output (I/O) peripherals on a single chip.

Block Diagram of Microcontroller

List of microcontrollers?

webpage

8051 Microcontroller IC

Inside view of 8051 IC

INTRODUCTION TO MICROCONTROLLERS

WHAT IS MICROCONTROLLER?

- Microcontroller can be easily interfaced to external peripheral devices like ADC, DAC, serial ports, Bluetooth, Wi-Fi, etc.

- Examples of the microcontrollers are microchip's PIC, Intel's 80196, 8051, Motorola's 68HCxx.

- Applications of microcontrollers:
  - ✓ Automobile
  - ✓ Aeronautics & Defence application
  - ✓ Mobile communications
  - ✓ Industrial processing
  - ✓ Robotics
  - ✓ Consumer electronics , music players
  - ✓ Security (e-commerce, smart cards)
  - ✓ Medical electronics (hospital equipment, and mobile monitoring) and

TMS-1802-NC Microcontroller Chip

One of the first applications of microcontrollers was in gas pumps.

History of first microcontroller

Gary Boone - Inventor of the Microcontroller

INTRODUCTION TO MICROCONTROLLERS

MICROPROCESSOR vs MICROCONTROLLERS

Microprocesser: CPU
and several supporting chips.

Microcontroller: CPU
on a single chip.

INTRODUCTION TO MICROCONTROLLERS

MICROPROCESSOR vs MICROCONTROLLERS

Microprocessor Based Temperature Monitoring System

Microcontroller Based Temperature Monitoring System

INTRODUCTION TO MICROCONTROLLERS

MICROPROCESSOR vs MICROCONTROLLERS

Parameters	µP	µC
Architecture	Von-Neumann and RISC architecture	Harvard and CISC architectures.
Flexibility	designer can decide on the amount of ROM, RAM and I/O ports.	fixed amount of on-chip ROM, RAM, I/O ports
Processing Power	High	Low
Design Time	More	Less
Power	High	Low
Protection	Less	More
Cost	High	Low
Examples	8085, 8086	8051

INTRODUCTION TO MICROCONTROLLERS

MICROPROCESSOR vs MICROCONTROLLERS

Von Neumann Architecture

Harvard Architecture

INTRODUCTION TO MICROCONTROLLERS

MICROPROCESSOR vs MICROCONTROLLERS

CISC	RISC
Emphasis on hardware	Emphasis on software
Multiple instruction sizes and formats	Instructions of same set with few formats
Less registers	Uses more registers
More addressing modes	Fewer addressing modes
Extensive use of microprogramming	Complexity in compiler
Instructions take a varying amount of cycle time	Instructions take one cycle time
Pipelining is difficult	Pipelining is easy

INTRODUCTION TO INTEL MCS 51 µC

INTRODUCTION TO INTEL MCS 51 µC

8051 FAMILY SERIES

- The Intel MCS-51 (commonly termed 8051) is a Harvard architecture, complex instruction set computing (CISC) Architecture, single chip microcontroller (µC) series developed by Intel in 1980 for use in embedded systems
- Intel's original MCS-51 family was developed using N-type metal-oxide-semiconductor (NMOS) technology
- But later versions, identified by a letter C in their name (e.g., 80C51) used complementary metal–oxide–semiconductor (CMOS) technology and consume less power
- Although the 8051 is the most popular member of the 8051 family, you will not see "8051" in the part number.
- This is because the 8051 is available in different memory types, such as UV-EPROM, flash, and NV-RAM, all of which have different part numbers.

INTRODUCTION TO INTEL MCS 51 µC

8051 FAMILY SERIES

- The UV-EPROM version of the 8051 is the 8751. The flash ROM version is marketed by many companies including Atmel Corp. and Dallas Semiconductor.

- The Atmel Flash 8051 is called AT89C51, while Dallas Semiconductor calls theirs DS89C4xO (DS89C420/430/440).

Feature	8051	8052	8031
ROM(bytes)	4K	8K	0K
RAM(bytes)	128	256	128
Timers	2	3	2
I/O pins	32	32	32
Serial port	1	1	1
Interrupt sources	6	8	6

INTRODUCTION TO INTEL MCS 51 µC

8051 ATMEL SERIES

Part Number	ROM	RAM	I/O pins	Timer	Interrupt	Vcc	Packaging
AT89C51	4K	128	32	2	6	5V	40
AT89C52	8K	256	32	3	8	5V	40
AT89C1051	1K	64	15	1	3	3V	20
AT89C2051	2K	128	32	3	8	3V	20
AT89LV51	4K	128	32	2	6	3V	40
AT89LV52	8K	128	32	3	8	3V	40

- 8051 Major Manufacturers: ATMEL, ANALOG DEVICES, ST MICROELECTRONICS, DALLAS, MAXIM, SILICON LABS, TEXAS INSTRUMENTS, MICROCHIP, ZILOG

INTRODUCTION TO INTEL MCS 51 µC

FEATURES OF 8051

- 8-bit CPU, 8-bit data bus, 16-bit address bus
- 4K bytes on-chip program memory (ROM), 128 bytes on-chip data memory (RAM)
- 32 I/O pins arranged as four 8-bit ports (P0 - P3)
- 32 general purpose registers each of 8-bit
- Special Function Registers (SFRs) of 128 bytes
- 16-bit Program Counter
- 8-bit Processor Status Word (PSW) & Stack Pointer
- Two 16-bit timer/counters : T0 and T1
- Two external and three internal vectored interrupts
- One full duplex serial I/O (UART)

INTEL 8051 PIN DETAILS

INTEL 8051 PIN DETAILS

INTEL 8051 PIN DETAILS

8051 PIN DETAILS

- Although 8051 family members (e.g., 8751, 89C51, 89C52, DS89C4xO) come in different packages, such as DIP (dual in-line package), QFP (quad flat package), and LLC (leadless chip carrier)

- They all have 40 pins that are dedicated to various functions such as I/O, RD, WR, address, data, and interrupts.

- Some companies provide a 20-pin version of the 8051 with a reduced number of I/O ports for less demanding applications.

- In 40 pin package, a total of 32 pins are set aside for the four ports P0, P1, P2, and P3, where each port takes 8 pins. The rest of the pins are designated as Vcc, GND, XTAL1, XTAL2, RST, EA, PSEN, and ALE.

INTEL 8051 PIN DETAILS

8051 PIN DETAILS

- Of these pins, six (Vcc, GND, XTAL1, XTAL2, RST, and EA) are used by all members of the 8051 and 8031 families.
- The other two pins, PSEN and ALE, are used mainly in 8031-based systems.

INTEL 8051 PIN DETAILS

8051 PIN DETAILS

- Pin-40 : Vcc is the main power source. Usually its +5V DC.

- Pins 32-39: Port 0 (P0.0 to P0.7) – In addition to serving as I/O port, lower order address and data bus signals are multiplexed with this port (to serve the purpose of external memory interfacing). This is a bi directional I/O port (the only one in 8051) and external pull up resistors are required to function this port as I/O.

- Pin-31:- ALE - Address Latch Enable. It is especially used for 8031 IC to connect it to the external memory. It can be used while deciding whether P0 pins will be used as Address bus or Data bus. When ALE = 1, then the P0 pins work as Data bus and when ALE = 0, then the P0 pins act as Address bus.

INTEL 8051 PIN DETAILS

8051 PIN DETAILS

- Pin-30:- EA - External Access input is used to enable or disallow external memory interfacing. If there is no external memory requirement, this pin is connect it to Vcc.

- Pin- 29:- PSEN or Program Store Enable. This is an active low pin, i.e., it gets activated after applying a low pulse. It is an output pin and used along with the EA pin in 8031 based (i.e. ROMLESS) Systems to allow storage of program code in external ROM.

- Pins- 21-28: Port 2 (P 2.0 to P 2.7) – in addition to serving as I/O port, higher order address bus signals are multiplexed with this quasi bi directional port.

- Pin 20:- Vss – it represents ground (0 V) connection.

- Pins 18 and 19:- XTAL1 & XTAL2 Used for interfacing an external crystal to provide system clock

INTEL 8051 PIN DETAILS

8051 PIN DETAILS

- Pins 10 – 17:- Port 3 (P 3.0 to P 3.7) It is also of 8 bits and can be used as Input/Output. This port provides some extremely important signals. P3.0 and P3.1 are RxD (Receiver) and TxD (Transmitter) respectively and are collectively used for Serial Communication. P3.2 and P3.3 pins are used for external interrupts. P3.4 and P3.5 are used for timers T0 and T1 respectively. P3.6 and P3.7 are Write (WR) and Read (RD) pins.

- Pin 9:- RESET pin is used to set the 8051 microcontroller to its initial values, while the microcontroller is working or at the initial start of application. The RESET pin must be set high for 2 machine cycles.

- Pins 1 – 8:- Port 1 (P 1.0 to P 1.7). It is an 8-bit port (pin 1 through 8) and can be used either as input or output. Unlike other ports, this port does not serve any other functions. Port 1 is an internally pulled up, quasi bi directional I/O port.

8051 MICROCONTROLLER ARCHITECTURE

8051 MICROCONTROLLER ARCHITECTURE

8051 ARCHITECTURE (Simplified)

8051 MICROCONTROLLER ARCHITECTURE

8051 ARCHITECTURE (Detailed)

8051 MICROCONTROLLER ARCHITECTURE

1. CENTRAL PROCESSING UNIT (CPU):

- CPU is the brain of any processing device of the microcontroller.
- It monitors and controls all operations that are performed on the Microcontroller units.
- It reads program written in ROM memory and executes them and do the expected task of that application. The User has no control over the work of the CPU directly.

2. INTERRUPT CONTROL:

- There are five interrupt sources in 8051 μC and interrupt control section control all these interrupts.
- Two external interrupts (INT0 & INT1), two timer (TF0 & TF1) interrupts and one serial port (RI/TI) interrupt.
- The Microcontroller 8051 can be configured in such a way that it temporarily terminates or pause the main program at the occurrence of interrupt. When subroutine is completed then the execution of main program starts as usual.

8051 MICROCONTROLLER ARCHITECTURE

3. RAM & ROM:

- 8051 has 4K of Code Memory or Program memory that is it has 4KB ROM and RAM of 128 bytes.
- The memory which is used to store the program of Microcontroller, is known as code memory or Program memory . It is known as 'ROM'(Read Only Memory).
- Microcontroller also requires a memory to store data or operands temporarily. This memory is known as Data Memory and we use 'RAM'(Random Access Memory) for this purpose.

4. BUS CONTROL:

- Bus control section of 8051 is responsible for controlling the operation of address and data bus.
- Bus: Basically Bus is a collection of wires which work as a communication channel or medium for transfer of Data. Buses are of two types:
  - Address Bus: Microcontroller 8051 has a 16 bit address bus. It used to address memory locations.
  - Data Bus: Microcontroller 8051 has 8 bits data bus. It is used to carry data.

8051 MICROCONTROLLER ARCHITECTURE

5. CRYSTAL OSCILLATOR:

- Since Microcontroller is a digital circuit device, therefore it requires clock for its operation.
- For this purpose, 8051 has oscillator circuitry section which works as a clock source for CPU.
- As the output pulses of oscillator are stable therefore it enables synchronized work of all parts of 8051 Microcontroller.

6. I/O PORTS:

- To connect any external devices or peripherals we require I/O interfacing ports in the microcontroller.
- All 8051 microcontrollers have 4 I/O ports each comprising 8 bits which can be configured as input (1) or an output (0), depends on its logic state.
- Accordingly, in total of 32 input/output pins enabling the microcontroller to be connected to peripheral devices are available for use.

8051 MICROCONTROLLER ARCHITECTURE

7. TIMERS/COUNTERS:

- Timers/Counters: 8051 Microcontroller 8051 two 16-bit timers and counters: Timer 0 and Timer 1.
- They can be used either as timers to generate a time delay or as counters to count events happening outside the microcontroller.
- Since the 8051 has an 8-bit architecture, each 16-bit is accessed as two separate registers of low byte and high byte.

8. SERIAL PORTS:

- The 8051 contains one Serial port or UART (Universal Asynchronous Receiver Transmitter). The serial port is full-duplex so, it can transmit and receive simultaneously.
- Two port pins are used for serial interface, P3.0 is the receive pin (RXD) P3.1 is the transmit pin (TXD).
- Serial port can be operated in one of four different modes and at a range of frequencies

RAM-ROM ORGANIZATION

RAM-ROM ORGANIZATION

PROGRAM MEMORY

- The 8051 has two types of memory and these are Program Memory and Data Memory.
- Program Memory (ROM) is used to permanently save the program being executed, while Data Memory (RAM) is used for temporarily storing data and intermediate results created and used during the operation of the microcontroller.
- Depending on the model in use (8051 microcontroller family in general) at most a few Kb of ROM and 128 or 256 bytes of RAM is used.
- The 8051 has 4K (2^{12} = 4096 locations = 0000H to 0FFFH) of internal or on-chip ROM and it can be expanded up to 64K.
- Even though such an amount of memory is sufficient for writing most of the programs, there are situations when it is necessary to use additional memory as well.

RAM-ROM ORGANIZATION

PROGRAM MEMORY

- EA=0: In this case, the microcontroller completely ignores internal program memory and executes only the program stored in external memory.
- EA=1: In this case, the microcontroller executes first the program from built-in ROM, then the program stored in external memory

RAM-ROM ORGANIZATION

DATA MEMORY

- In the MCS-51 family, 8051 has 128 bytes of internal data memory and it allows interfacing external data memory of maximum size up to 64K.

- So the total size of data memory in 8051 can be upto 64K (external) + 128 bytes (internal).

- So there are 3 divisions of the data memory:-

  1) Register banks

  2) Bit addressable area

  3) Scratch pad area

  4) Special Function Registers (SFR)

RAM-ROM ORGANIZATION

DATA MEMORY

RAM-ROM ORGANIZATION

DATA MEMORY

1. Register Bank:

- Registers are used to store data or operands during executions. Register banks form the lowest 32 bytes on internal RAM memory.
- There are 4 register banks designated bank #0,#1, #2 and #3. Each bank has 8 registers which are designated as R0,R1…R7.
- At a time only one register bank is selected (using RS1 & RS0 bits in PSW register) for operations and the registers inside the selected bank are accessed using mnemonics R0..R1.. etc.
- By default register bank #0 is selected (after a system reset).

RAM-ROM ORGANIZATION

DATA MEMORY

Register banks and their RAM address

Bank 0		Bank 1		Bank 2		Bank 3	
7	R7	F	R7	17	R7	1F	R7
6	R6	E	R6	16	R6	1E	R6
5	R5	D	R5	15	R5	1D	R5
4	R4	C	R4	14	R4	1C	R4
3	R3	B	R3	13	R3	1B	R3
2	R2	A	R2	12	R2	1A	R2
1	R1	9	R1	11	R1	19	R1
0	R0	8	R0	10	R0	18	R0

RAM-ROM ORGANIZATION

DATA MEMORY

2. Bit Addressable Area:

- The 8051 supports a special feature which allows access to bit variables.
- This is where individual memory bits in Internal RAM can be set or cleared.
- The Bit Addressable area of the RAM is 16 bytes (128 bits) next to register banks of Internal RAM located between 20h and 2Fh.
- In all there are 128 bits numbered 00h to 7Fh.
- Being bit variables any one variable can have a value 0 or 1.
- A bit variable can be set with a command such as SETB and cleared with a command such as CLR.

RAM-ROM ORGANIZATION

DATA MEMORY

- Example instructions are

  SETB 25h ; sets the bit 25h (becomes 1)

  CLR 25h ; clears bit 25h (becomes 0)

- Bit addressable area is mainly used to store bit variables from application program, like status of an output device like LED or Motor (ON/OFF) etc.

- We need only a bit to store this status and using a complete byte addressable area for storing this is really bad programming practice, since it results in wastage of memory.

RAM-ROM ORGANIZATION

DATA MEMORY

3. Scratch-pad Area:

- 80 bytes of Internal RAM memory scratch pad area are available for general-purpose data storage (from 30H to 7FH) and this includes stack too.
- However, these 80 bytes are used by the system stack and in practice little space is left for general storage.
- The scratch pad RAM can be accessed using direct or indirect addressing modes.
- Examples of direct addressing:

```
MOV A, 6Ah      ;reads contents of address 6Ah to accumulator
```

- Examples for indirect addressing (use registers R0 or R1):

```
MOV R1, #6Ah    ; move immediate 6Ah to R1
MOV A, @R1      ; R1 contains address of RAM which contains data that is moved to A.
```

RAM-ROM ORGANIZATION

DATA MEMORY

4. Special Function Register (SFR):

- SFRs are accessed just like normal Internal RAM locations and they are located within the Internal Memory in the address range 80h to FFH.
- Each SFR has a very specific function as well as specific memory locations. Note some of the SFR registers are bit addressable.
- Although 128 byes of the SFR address space is defined only 21 SFR registers are defined in the standard 8051.
- Rest of locations are intentionally left unoccupied in order to enable the manufacturers to further develop microcontrollers keeping them compatible with the previous versions.
- Main function of SFR is to control timers, counters, serial I/O, port I/O, and peripherals that are present in 8051 microcontroller.

RAM-ROM ORGANIZATION

DATA MEMORY

Name of the Register	*Function*	*Internal RAM Address (HEX)*
ACC	Accumulator	E0H
B	B Register (for Arithmetic)	F0H
DPH	Addressing External Memory	83H
DPL	Addressing External Memory	82H
IE	Interrupt Enable Control	A8H
IP	Interrupt Priority	B8H
P0	PORT 0 Latch	80H
P1	PORT 1 Latch	90H
P2	PORT 2 Latch	A0H
P3	PORT 3 Latch	B0H
PCON	Power Control	87H
PSW	Program Status Word	D0H
SCON	Serial Port Control	98H
SBUF	Serial Port Data Buffer	99H
SP	Stack Pointer	81H
TMOD	Timer / Counter Mode Control	89H
TCON	Timer / Counter Control	88H
TL0	Timer 0 LOW Byte	8AH
TH0	Timer 0 HIGH Byte	8CH
TL1	Timer 1 LOW Byte	8BH
TH1	Timer 1 HIGH Byte	8DH

SFRs register layout

RAM-ROM ORGANIZATION

SFR

- 21 Special Function Registers of 8051 Microcontroller are categorized in to seven groups.:

1. Math or CPU Registers: A and B
2. Status Register: PSW (Program Status Word)
3. I/O Port Registers: P0 (Port 0), P1 (Port 1), P2 (Port 2) and P3 (Port 3)
4. Pointer Registers: DPTR (Data Pointer – DPL,DPH), SP (Stack Pointer)
5. Peripheral Control Registers: PCON, SCON, TCON, TMOD, IE and IP
6. Peripheral Data Registers: TL0, TH0, TL1, TH1 and SBUF

RAM-ROM ORGANIZATION

SFR (A & B REGISTER)

- The most important of all SFR is Accumulator which is also known as ACC or A. It is an 8-bit register and it is both byte and bit addressable.
- The Accumulator holds the result of most of arithmetic and logic operations.
- It is also used to store 8-bit data and to hold one of operand of ALU during arithmetical and logical operations.
- More than half instructions used by the 8051 microcontroller use somehow the accumulator.
- ACC is usually accessed by direct addressing and its physical address is E0H.
- To access the individual bits of accumulator, use the format ACC.X in the instruction where "X" denotes bit to be accessed.

RAM-ROM ORGANIZATION

SFR (A & B REGISTER)

- It is special 8-bit math register and it is bit and byte accessible.
- It is used in conjunction with A register as an input operand for ALU to perform multiplication and division operation.
- It can also be used as general purpose register to store 8-bit data.

	(MSB) ACC.7	ACC.6	ACC.5	ACC.4	ACC.3	ACC.2	ACC.1	(LSB) ACC.0
Direct Address E0H								
Bit Address	E7	E6	E5	E4	E3	E2	E1	E0

	(MSB) B.7	B.6	B.5	B.4	B.3	B.2	B.1	(LSB) B.0
Direct Address F0H								
Bit Address	F7	F6	F5	F4	F3	F2	F1	F0

RAM-ROM ORGANIZATION

SFR (PSW REGISTER)

- Program Status Word(PSW) is a 8-bit register and it is bit and byte accessible.
- It contains several status bits that reflects the status of the operation that is being carried out in the processor.
- It has 4 conditional flags (CY, AC, OV, P) which sets or resets according to condition of result.
- It has 3 control flags (F0, RS0, RS1) by setting or resetting bit required operation or function can be achieved.

RAM-ROM ORGANIZATION

SFR (PSW REGISTER)

- CY, the carry flag: This flag is set whenever there is a carry out from the D7 bit. This flag bit is affected after an 8-bit addition or subtraction.

- AC, the auxiliary carry flag: If there is a carry from D3 to D4 during an ADD or SUB operation, this bit is set; otherwise, it is cleared. This flag is used by instructions that perform BCD (binary coded decimal) arithmetic.

- P, the parity flag: The parity flag reflects the number of 1 s in the A (accumulator) register only. If the A register contains an odd number of 1's, then P = 1. Therefore, P = 0 if A has an even number of 1s.

- F0, the Flag 0: The PSW.5 and PSW.1 bits are general-purpose status flag bits and can be used by the programmer for any purpose. In other words, they are user definable.

RAM-ROM ORGANIZATION

SFR (PSW REGISTER)

- **OV, overflow flag:** This flag is set whenever the result of a signed number operation is too large, causing the high-order bit to overflow into sign bit. In general, the carry flag is used to detect errors in unsigned arithmetic operations. The overflow flag is only **used to detect errors in signed arithmetic operations.** Overflow flag is set to 1(C6 XOR C7), if either of the following condition occurs.
  - There is a carry from D6 to D7 but no carry out from D7.
  - There is a carry out from D7, but no carry from D6 to D7.

- **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.

PSW bank selection

	RS1(PSW.4)	RS0(PSW.3)
Bank 0	0	0
Bank 1	0	1
Bank 2	1	0
Bank 3	1	1

RAM-ROM ORGANIZATION

SFR (I/O PORT REGISTER)

- There are 4 ports with in total of 32 input/output pins are available for connection to peripheral environment.
- So 4 Input/Output ports named P0, P1, P2 and P3 has got four corresponding port registers P0, P1, P2 and P3. All 4 port registers are bit as well as byte addressable.
- Data must be written into port registers first to send it out to any other external device through ports. Similarly any data received through ports must be read from port registers for performing any operation.
- If a bit is cleared (0), the appropriate pin will be configured as an output, while if it is set (1), the appropriate pin will be configured as an input.
- Upon reset and power-on, all port bits are set (1), which means that all appropriate pins will be configured as inputs.

RAM-ROM ORGANIZATION

SFR (I/O PORT REGISTER)

Input Output Port P0

	(MSB) P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	(LSB) P0.0
Direct Address 80H								
Bit Address	87	86	85	84	83	82	81	80

Input Output Port P1

	(MSB) P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	(LSB) P1.0
Direct Address 90H								
Bit Address	97	96	95	94	93	92	91	90

Input Output Port P2

	(MSB) P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	(LSB) P2.0
Direct Address A0H								
Bit Address	A7	A6	A5	A4	A3	A2	A1	A0

Input Output Port P3

	(MSB) P3.7	P3.6	P3.5	P3.4	P3.3	P3.2	P3.1	(LSB) P3.0
Direct Address B0H								
Bit Address	B7	B6	B5	B4	B3	B2	B1	B0

RAM-ROM ORGANIZATION

SFR (DPTR REGISTER)

- It is a 16-bit register used to hold address of external or internal RAM where data is stored or result is to be stored.
- It can be divided into two 8-bit registers, DPH-data pointer higher order and DPL-data pointer lower order.
- Each register can be used as general purpose register to store 8-bit data and can also be used to store memory location.
- It functions as Base register in base relative addressing mode and indirect jump.

RAM-ROM ORGANIZATION

SFR (DPTR REGISTER)

RAM-ROM ORGANIZATION

SFR (STACK POINTER)

- It is 8-bit register and byte addressable. It is used to hold the internal RAM memory location addresses which are used as stack memory.
- When the data is to be placed on stack by push instruction, the content of stack pointer is incremented by 1.
- When data is retrieved from stack, content of stack of stack pointer is decremented by 1. A value stored in the Stack Pointer points to the first free stack address and permits stack availability.
- Upon any reset and power-on, the value 7 is stored in the Stack Pointer, which means that the space of RAM reserved for the stack starts at this location.

	0	0	0	0	0	1	1	1	Value after Reset
SP									Bit name
	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	

RAM-ROM ORGANIZATION

PROGRAM COUNTER (Not a part of SFR)

- It is used to hold 16 bit address of internal RAM, external RAM or external ROM locations.
- The Program Counter (PC) is a 2-byte address which tells the 8051 where the next instruction to be executed from memory.
- When the 8051 is initialized PC always starts at 0000h and is incremented each time an instruction is executed.

ADDRESSING MODES

ADDRESSING MODES

- Addressing mode is a way to address an operand.
- Operand means the data we are operating upon.
- There are eight addressing modes available in the 8051:
  - Immediate
  - Direct
  - Register
  - Register indirect
  - Indexed

ADDRESSING MODES

IMMEDIATE ADDRESSING MODE

- This addressing mode is named as "immediate" because it transfers an 8-bit data immediately to the accumulator (destination operand).

- In general we can write MOV A, #data. Example: MOV A, #6AH. The '#' symbol before 6AH indicates that operand is a data (8 bit).

- The opcode for MOV A, # data is 74H. The opcode is saved in program memory at 0202 address. The data 6AH is saved in program memory 0203.

- When the opcode 74H is read, the next step taken would be to transfer whatever data at the next program memory address (here at 0203) to accumulator A (E0H is the address of accumulator).

- This instruction is of two bytes and is executed in one cycle. So after the execution of this instruction, program counter will add 2 and move to 0204 of program memory.

ADDRESSING MODES

IMMEDIATE ADDRESSING MODE

Immediate Addressing Mode

Instruction	Opcode	Bytes	Cycles
MOV A, #6AH	74H	2	1

ADDRESSING MODES

DIRECT ADDRESSING MODE

- Here the address of the data (source data) is given as operand. Lets take an example.

  MOV A, 04H

- Here 04H is the address of register 4 of register bank#0. When this instruction is executed, what ever data is stored in register 04H is moved to accumulator.
- In the figure register 04H holds the data 1FH. So the data 1FH is moved to accumulator.
- The opcode for instruction MOV A, address is E5H. When the instruction at 0202 is executed (E5H), accumulator is made active and ready to receive data.
- Then program control goes to next address that is 0203 and look up the address of the location (04H) where the source data (to be transferred to accumulator) is located.
- At 04H the control finds the data 1F and transfers it to accumulator and hence the execution is completed.

ADDRESSING MODES

DIRECT ADDRESSING MODE

Direct Addressing Mode

Instruction	Opcode	Bytes	Cycles
MOV A, 04H	E5	2	1

ADDRESSING MODES

REGISTER ADDRESSING MODE

- In this addressing mode we use the register name directly (as source operand).
- At a time registers can take value from R0,R1…to R7. An example is shown below.

  MOV A, R4
- In register direct addressing mode, data is transferred to accumulator from the register (based on which register bank is selected).
- PSW.3 and PSW.4 bits are known as register bank select bits as they are used to select register banks.

ADDRESSING MODES

REGISTER ADDRESSING MODE

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

Register bank Select bit 0

Register bank Select bit 1

RS1	RS0	Register Bank	Register Bank Status
0	0	0	Register Bank 0 is selected
0	1	1	Register Bank 1 is selected
1	0	2	Register Bank 2 is selected
1	1	3	Register Bank 3 is selected

Instruction	Opcode	Bytes	Cycles
MOV A, R4	ECH	1	1

ADDRESSING MODES

REGISTER INDRECT ADDRESSING MODE

- In this addressing mode, address of the data (source data to transfer) is given in the register operand.

MOV A, @R0

- Here the value inside R0 is considered as an address, which holds the data to be transferred to accumulator. If R0 is 20H, and a data 2FH stored at the address 20H, then the value 2FH will get transferred to accumulator after executing this instruction.

- The opcode for MOV A, @R0 is E6H. Assuming that register bank #0 is selected. So the R0 of register bank #0 holds the data 20H.

- This is a single byte instruction and the program counter increments 1 and moves to 0203 of program memory. Only R0 and R1 are allowed to form a register indirect addressing instruction.

ADDRESSING MODES

REGISTER INDRECT ADDRESSING MODE

Instruction	Opcode	Bytes	Cycles
MOV A, @ R0	E6H	1	1

ADDRESSING MODES

INDEXED ADDRESSING MODE

MOVC A, @A+DPTR and MOVC A, @A+PC

- where DPTR is data pointer and PC is program counter (both are 16 bit registers).
- The source operand is @A+DPTR and we will get the source data (to transfer) from this location.
- It is nothing but adding contents of DPTR with present content of accumulator. This addition will result a new data which is taken as the address of source data (to transfer). The data at this address is then transferred to accumulator.
- The opcode for the instruction is 93H. DPTR holds the value 01FE, where 01 is located in DPH (higher 8 bits) and FE is located in DPL (lower 8 bits)

ADDRESSING MODES

INDEXED ADDRESSING MODE

- Accumulator now has the value 02H. A 16 bit addition is performed and now 01FE H+02 H results in 0200 H.

- Whatever data is in 0200 H will get transferred to accumulator. The previous value inside accumulator (02H) will get replaced with new data from 0200H. New data in the accumulator is shown in dotted line box.

- This is a 1 byte instruction with 2 cycles needed for execution. So, the execution time required for this instruction is high compared to other addressing modes (which all were 1 cycle).

- The other example MOVC A, @A+PC works the same way as above example. The only difference is, instead of adding DPTR with accumulator, here data inside program counter (PC) is added with accumulator to obtain the target address.

MACHINE CYCLE

MACHINE CYCLE

- The CPU takes a certain number of clock cycles to execute an instruction. In the 8051 family, these clock cycles are referred to as machine cycles.

- A single machine cycle is the minimum amount of time in which a single 8051 instruction can be executed, although many instructions take multiple cycles.

- Pulse - One complete oscillation of the clock source; State - Two pulses; Machine Cycle - six states.

MACHINE CYCLE

- If an instruction takes one machine cycle to execute, it will take 12 pulses of the crystal to execute. 8051 is designed to operate between 4MHz to 40MHz and generally operates with a crystal frequency 11.0592 MHz.
- Since we know the crystal is pulsing 11,059,000 times per second and that one machine cycle is 12 pulses. we can calculate how many instruction cycles the 8051 can execute per second: 11,059,000 / 12 = 921,583 and 1/ 921,583=1.085μs
- This means that the 8051 can execute 921,583 single-cycle instructions per second.
- For example, if you are using exclusively 2-cycle instructions you would find that the 8051 would execute 460,791 instructions per second.
- The 8051 also has two really slow instructions that require a full 4 cycles to execute-those instructions you'd find performance to be about 230,395 instructions per second.

MACHINE CYCLE

EXAMPLE-1

- Lets find the time period of the machine cycle in each case for the following crystal frequency of different 8051 based systems: 11.0592 MHz, 16 MHz, 20 MHz.

- Answer:
  - 11.0592 MHz:

    11.0592/12 = 921.6 KHz

    Machine cycle = 1/921.6 KHz = 1.085us [us=microsecond]
  - 16 MHz:

    16MHz/12 = 1.333 MHz

    Machine cycle = 1/1.333 MHz = 0.75us [us=microsecond]
  - 20MHz:

    20MHz/12 = 1.66 MHz

    Machine Cycle = 1/1.66 MHz = 0.60us [us=microsecond]

MACHINE CYCLE

EXAMPLE-2

- Lets find how long it takes to execute each of the following instructions, for a crystal frequency of 11.0592 MHz. The machine cycle of a system of 11.0592.z is 1.085 us

INSTRUCTION	MACHINE CYCLE	TIME TO EXECUTE
MOV R2,#55H	1	1x1.085 us = 1.085 us
DEC R2	1	1x1.085 us = 1.085 us
DJNZ R2,target	2	2x1.085 us = 2.17 us
LJMP	2	2x1.085 us = 2.17 us
SJMP	2	2x1.085 us = 2.17 us
NOP	1	1x1.085 us = 1.085 us
MUL AB	4	4x1.085 us = 4.34 us

INSTRUCTION SET

INSTRUCTION SET

- All microcontrollers compatible with the 8051 have in total of 255 instructions, many instructions are considered to be "different", even though they perform the same operation, so there are only 111 truly different commands.

- For example: ADD A,R0, ADD A,R1, ... ADD A,R7 are instructions that perform the same operation (addition of the accumulator and register). Taking into account that all instructions perform only 53 operations (addition, subtraction, copy etc.)

- 8051 instructions have 8-bit opcode
  - There are 256 possible instructions of which 255 are implemented
  - Every 8-bit opcode from 00 to FF is used except for A5.
  - Some instructions have one or two additional bytes for data or address
  - There are 139 1-byte instructions, 92 2-byte instructions, and 24 3-byte instruction

INSTRUCTION SET

INSTRUCTION SET

IMPORTANT NOTATIONS USED IN THE INSTRUCTIONS

Rn	Register R7-R0 of the currently selected Register Bank.
Direct	8-bit internal data location's address. This could be an Internal Data RAM location (0-127) or a SFR [i.e., I/O port, control register, status register, etc. (128-255)].
@Ri	8-bit internal data RAM location (0-255) addressed indirectly through register R1or R0.
#Data	8-bit constant included in instruction.
#Data16	16-bit constant included in instruction.
ADDR11	11-bit destination address. Used by ACALL and AJMP. The branch will be within the same 2K byte page of program memory as the first byte of the following instruction
ADDR16	16-bit destination address. Used by LCALL and LJMP. A branch can be anywhere within the 64K byte Program Memory address space.
rel	Signed (two's complement) 8-bit offset byte. Used by SJMP and all conditional jumps. Range is -128 to +127 bytes relative to first byte of the following instruction.
Bit	Direct Addressed bit in Internal Data RAM or Special Function Register

INSTRUCTION SET

DATA TRANSFER INSTRUCTIONS

- These instructions are associated with transfer with data between registers or external program code or external data memory.

Mnemonic	Meaning	Format	Operation	Example	# Byte	# Cycles
MOVC	Move Code Byte to Accumulator	MOVC A,@A+DPTR	A ← Code pointed by A+DPTR	MOVC A,@A+DPTR	1	2
		MOVC A,@A+PC	A ← Code pointed by A+PC	MOVC A,@A+PC	1	2
		MOVC A,@Ri	A ← Code pointed by Ri (8-bit address)	MOVC A,@R0	1	2
MOVX	Move Data To/From External Memory (XRAM)	MOVX A,@DPTR	A ← External Data pointed by DPTR	MOVX A,@DPTR	1	2
		MOVX @Ri, A	@Ri ← (External Data 8-Bit address)	MOVX @R0, A	1	2
		MOVX @DPTR,A	@DPTR ← (External Data 16-Bit address)	MOVX @DPTR,A	1	2

INSTRUCTION SET

Mnemonic	Meaning	Format	Operation	Example	# Byte	# Cycles
MOV	Move data from source to destination	MOV A, #Data	A ← Data	MOV A,#45H	2	1
		MOV A, Rn	A ← Rn	MOV A,R2	1	1
		MOV A, Direct	A ← (Direct)	MOV A,45H	2	1
		MOV A, @Ri	A ← @Ri	MOV A,@R0	1	1
		MOV Rn, #Data	Rn ← Data	MOV R2,#45H	2	1
		MOV Rn, A	Rn ← A	MOV R2,A	1	1
		MOV Rn, Direct	Rn ← (Direct)	MOV A,45H	2	2
		MOV Direct, A	(Direct) ← A	MOV 45H,A	2	1
		MOV Direct, Rn	(Direct) ← Rn	MOV 45H,R2	2	2
		MOV Direct1, Direct2	(Direct1) ← (Direct2)	MOV 45H,55H	3	2
		MOV Direct, @Ri	(Direct) ← @Ri	MOV 45H,@R0	2	2
		MOV Direct, #Data	(Direct) ← Data	MOV 45H,#45H	3	2
		MOV @Ri, A	@Ri ← A	MOV @R0,A	1	1
		MOV @Ri, Direct	@Ri ← (Direct)	MOV @R0,45H	2	2
		MOV @Ri, #Data	@Ri ← Data	MOV @R0,#45H	2	1
		MOV DPTR,#Data16	DPTR ← Data	MOV DPTR,#3000H	3	2

INSTRUCTION SET

DATA TRANSFER INSTRUCTIONS

Mnemonic	Meaning	Format	Operation	Example	# Byte	# Cycles
PUSH	Push Value Onto Stack	PUSH Direct	Stack Pointer ← (Direct)	PUSH 45h	2	2
POP	Pop Value From Stack	POP Direct	(Direct) ← Stack Pointer	POP 45H	2	2
XCH	Exchange Bytes	XCH Rn	Exchange Accumulator with Rn	XCH R2	1	1
		XCH Direct	Exchange Accumulator with Direct Byte	XCH 45H	2	1
		XCH @Ri	Exchange Accumulator with Indirect RAM	XCH @R0	1	1
XCHD	Exchange lower order nibble of the Accumulator with Lower order nibble of the Indirect RAM	XCHD A,@Ri		XCHD A,@R0	1	1

INSTRUCTION SET

ARITHMETIC INSTRUCTIONS

Mnemonic	Meaning	Format	Operation	Example	# Byte	# Cycles
ADD	Add Accumulator	ADD A, #Data	A ← A + Data	ADD A,#45H	2	1
		ADD A, Rn	A ← A + Rn	ADD A,R2	1	1
		ADD A, Direct	A ← A + (Direct)	ADD A, 45H	2	1
		ADD A, @Ri	A ← A + @Ri	ADD A, @R0	1	1
ADDC	Add Accumulator with Carry	ADD A, #Data	A ← A + Data + C	ADD A,#45H	2	1
		ADD A, Rn	A ← A + Rn + C	ADD A,R2	1	1
		ADD A, Direct	A ← A + (Direct) + C	ADD A, 45H	2	1
		ADD A, @Ri	A ← A + @Ri +C	ADD A, @R0	1	1
SUBB	Subtract from Accumulator With Borrow	SUBB A, #Data	A ← A – Data - C	SUBB A, #45H	2	1
		SUBB A, Rn	A ← A – Rn - C	SUBB A, R2	1	1
		SUBB A, Direct	A ← A - (Direct) - C	SUBB A, 45H	2	1
		SUBB A, @Ri	A ← A - @Ri - C	SUBB A, @R0	1	1

INSTRUCTION SET

ARITHMETIC INSTRUCTIONS

Mnemonic	Meaning	Format	Operation	Example	# Byte	# Cycles
MUL	Multiply Accumulator by B	MUL AB	A*B, A ← Lower byte of A*B, B ← Higher byte of A*B	MUL AB	1	4
DIV	Divide Accumulator by B	DIV AB	A/B, A ← Quotient, B ← Reminder	DIV AB	1	4
INC	Increment	INC A	A ← A + 1	INC A	1	1
		INC Rn	Rn ← Rn + 1	INC R2	1	1
		INC Direct	(Direct) ← (Direct) + 1	INC 45H	2	1
		INC @Ri	@Ri ← @Ri + 1	INC @R1	1	1
		INC DPTR	DPTR ← DPTR + 1	INC DPTR	1	2
DEC	Decrement	DEC A	A ← A - 1	DEC A	1	1
		DEC Rn	Rn ← Rn - 1	DEC R2	1	1
		DEC Direct	(Direct) ← (Direct) - 1	DEC 45H	2	1
		DEC @Ri	@Ri ← @Ri - 1	DEC @R1	1	1

INSTRUCTION SET

ARITHMETIC INSTRUCTIONS

Mnemonic	Meaning	Format	Operation	Example	# Byte	# Cycles
DA	Decimal Adjust Accumulator	DA A	Convert Accumulator value into proper two digit BCD number	DA A	1	1
CLR	Clear Accumulator	CLR A	A ← 00H	CLR A	1	1
CPL	Complement Accumulator	CPL A	A ← A’	CPL A	1	1
RL	Rotate Accumulator Left	RL A		RL A	1	1
RLC	Rotate Accumulator Left Through Carry	RLC A		RLC A	1	1
RR	Rotate Accumulator Right	RR A		RR A	1	1
RRC	Rotate Accumulator Right Through Carry	RRC A		RRC A	1	1
SWAP	Swap Accumulator Nibbles	SWAP A		SWAP A	1	1

INSTRUCTION SET

LOGICAL INSTRUCTIONS

- Logical Instruction (AND, OR, XOR) are performed on Bytes of data on a bit-by-bit basis.

Mnemonic	Meaning	Format	Operation	Example	# Byte	# Cycles
ANL	Perform bit-by-bit AND operation	ANL A, #Data	A ← A AND Data	ANL A, #45H	2	1
		ANL A, Rn	A ← A AND Rn	ANL A,R2	1	1
		ANL A, Direct	A ← A AND (Direct)	ANL A, 45H	2	1
		ANL A, @Ri	A ← A AND @Ri	ANL A, @R0	1	1
		ANL Direct, A	(Direct) ← (Direct) AND A	ANL 45H,A	2	1
		ANL Direct, #Data	(Direct) ← (Direct) AND Data	ANL 45H,#45H	3	2

INSTRUCTION SET

LOGICAL INSTRUCTIONS

Mnemonic	Meaning	Format	Operation	Example	# Byte	# Cycles
ORL	Perform bit-by-bit OR operation	ORL A, #Data	A ← A OR Data	ORL A, #45H	2	1
		ORL A, Rn	A ← A OR Rn	ORL A,R2	1	1
		ORL A, Direct	A ← A OR (Direct)	ORL A, 45H	2	1
		ORL A, @Ri	A ← A OR @Ri	ORL A, @R0	1	1
		ORL Direct, A	(Direct) ← (Direct) OR A	ORL 45H,A	2	1
		ORL Direct, #Data	(Direct) ← (Direct) OR Data	ORL 45H,#45H	3	2
XRL	Perform bit-by-bit XOR operation	XRL A, #Data	A ← A XOR Data	XRL A, #45H	2	1
		XRL A, Rn	A ← A XOR Rn	XRL A,R2	1	1
		XRL A, Direct	A ← A XOR (Direct)	XRL A, 45H	2	1
		XRL A, @Ri	A ← A XOR @Ri	XRL A, @R0	1	1
		XRL Direct, A	(Direct) ← (Direct) XOR A	XRL 45H,A	2	1
		XRL Direct, #Data	(Direct) ← (Direct) XOR Data	XRL 45H,#45H	3	2

INSTRUCTION SET

BOOLEAN OR BIT-ORIENTED INSTRUCTIONS

- These Instructions will deal with special bit-addressable area in the RAM and some of the bit addressable Special Function Registers (SFRs).

Mnemonic	Meaning	Format	Operation	Example	# Byte	# Cycles
MOV	Move a bit addressable register bit to Carry bit	MOV C, Bit	C ← Bit (C = Carry bit), (Bit = Direct Bit)	MOV C, P1.0	2	1
	Move Carry bit to bit addressable register bit	MOV Bit, C	Bit ← C	MOV P1.0, C	2	2
CLR	Clear Carry bit	CLR C	C ← 0	CLR C	1	1
	Clear bit addressable register bit	CLR Bit	Bit ← 0	CLR P1.0	2	1
SETB	Set Carry bit	SETB C	C ← 1	SETB C	1	1
	Set bit addressable register bit	SETB Bit	Bit ← 1	SETB P1.0	2	1

INSTRUCTION SET

BOOLEAN OR BIT-ORIENTED INSTRUCTIONS

Mnemonic	Meaning	Format	Operation	Example	# Byte	# Cycles
CPL	Complement Carry bit	CPL C	C ← C'	CPL C	1	1
	Complement bit addressable register bit	CPL Bit	Bit ← Bit'	CPL P1.0	2	1
ANL	Perform AND operation on Carry bit and bit addressable register bit	ANL C, Bit	C ← C AND Bit	ANL C, P1.0	2	1
	Perform AND operation on Carry bit and Complement of bit addressable register bit	ANL C, /Bit	C ← C AND Bit'	ANL C, /P1.0	2	1
ORL	Perform OR operation on Carry bit and bit addressable register bit	OR C, Bit	C ← C OR Bit	ORL C, P1.0	2	1
	Perform OR operation on Carry bit and Complement of bit addressable register bit	OR C, /Bit	C ← C OR Bit'	ORL C, /P1.0	2	1

INSTRUCTION SET

PROGRAM BRANCHING INSTRUCTIONS

- These instructions control the flow of program logic. 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

- There are two categories of program branching instructions in the 8051.

1. CALL Instruction

2. JUMP Instruction
   - Conditional - control is transferred based on fulfilling certain condition before jumping to the target location
   - Unconditional – control is transferred unconditionally to the target location

INSTRUCTION SET

PROGRAM BRANCHING INSTRUCTIONS

List of Conditional JUMP instructions in 8051:

- JZ - Jump if Zero
- JNZ - Jump of Not Zero
- JC - Jump if Carry is set.
- JNC - Jump if Not Carry is set
- JB - Jump if Bit is set.
- JNB - Jump if Bit is Not set.
- JBC - Jump if Bit is set and Clear the bit.
- CJNE - Compare and Jump if Not Equal
- DJNZ - Decrement and Jump if Not Zero

INSTRUCTION SET

PROGRAM BRANCHING INSTRUCTIONS

List of Unconditional JUMP instructions in 8051:

- JMP - JuMP
- SJMP - Short JuMP
- LJMP - Long JuMP
- AJMP - Absolute JuMP

List of CALL instructions in 8051:

- ACALL - Absolute CALL
- LCALL - Long CALL

Other types of control instructions in 8051:

- NOP - No OPeration
- RET - RETurn
- RETI - RETurn from Interrupt

INSTRUCTION SET

PROGRAM BRANCHING INSTRUCTIONS

Mnemonic	Meaning	Format	Operation	Example	# Byte	# Cycles
ACALL	Absolute Call Within 2K Block	ACALL ADDR11	PC+2 → (SP); ADDR11→ PC	ACALL 0200H	2	2
LCALL	Long Call any 64k address location	LCALL ADDR16	PC+3 → (SP); ADDR16→ PC	LCALL 3000H	3	2
RET	Return From Subroutine	RET	(SP) → (PC)	RET	1	2
RETI	Return From Interrupt	RETI	(SP) → (PC)	RETI	1	2
AJMP	Absolute Jump Within 2K Block	AJMP ADDR11	ADDR11→ (PC)	AJMP 0200H	2	2
LJMP	Long Jump any 64k address location	AJMP ADDR16	ADDR16 → (PC)	LJMP 3000H	3	2
SJMP	Short Jump within -128 or +127 bytes	SJMP rel	PC+2+rel → PC	SJMP 002FH	2	2
JMP	Jump to Data Pointer + Accumulator	JMP @A+DPTR	A+DPTR → PC	JMP @A+DPTR	1	2

INSTRUCTION SET

PROGRAM BRANCHING INSTRUCTIONS

Mnemonic	Meaning	Format	Operation	Example	# Byte	# Cycles
JZ	Jump if Accumulator Zero	JZ rel	If A=0, Jump to PC+rel	JZ 0200H	2	2
JNZ	Jump if Accumulator Not Zero	JNZ rel	If A≠0, Jump to PC+rel	JNZ 3000H	2	2
JC	Jump if Carry Set	JC rel	If C=1, Jump to PC+rel	JC 0200H	2	2
JNC	Jump if Carry Not Set	JNC rel	If C=0, Jump to PC+rel	JNC 3000H	2	2
JB	Jump if Bit Set	JB Bit,rel	If Bit=1, Jump to PC+rel	JB P1.0,0200H	3	2
JNB	Jump if Bit Not Set	JNB Bit,rel	If Bit=0, Jump to PC+rel	JNB P1.0, 3000H	3	2
JBC	Jump if Bit Set and Clear Bit	JBC Bit,rel	If Bit=1, Jump to PC+rel and clear the bit	JBC P1.0, 0200H	3	2

INSTRUCTION SET

PROGRAM BRANCHING INSTRUCTIONS

Mnemonic	Meaning	Format	Operation	Example	# Byte	# Cycles
CJNE	Compare and Jump If Not Equal	CJNE A, Direct, rel	Compare (Direct) with A. Jump to PC+rel if not equal	CJNE A,45H, 3000H	3	2
		CJNE A, #Data, rel	Compare #Data with A. Jump to PC+rel if not equal	CJNE A,#45H, 3000H	3	2
		CJNE Rn, #Data, rel	Compare #Data with Rn. Jump to PC+rel if not equal	CJNE R2,#45H, 3000H	3	2
		CJNE @Ri, #Data, rel	Compare #Data with @Ri. Jump to PC+rel if not equal	CJNE @R0,#45H, 3000H	3	2
DJNZ	Decrement and Jump if Not Zero	DJNZ Rn, rel	Decrement Rn and Jump to PC+rel if Not Zero	DJNZ R2,0200H	2	2
		DJNZ Direct, rel	Decrement (Direct) and Jump to PC+rel if Not Zero	DJNZ 45H,0200H	3	2
NOP	No Operation	NOP	-	NOP	1	1

INSTRUCTION SET

INSTRUCTIONS THAT AFFECT FLAG SETTINGS

Instruction	Flag			Instruction	Flag		
	C	OV	AC		C	OV	AC
ADD	X	X	X	CLR C	0		
ADDC	X	X	X	CPL C	X		
SUBB	X	X	X	ANL C,Bit	X		
MUL	0	X		ANL C,/Bit	X		
DIV	0	X		OR C,Bit	X		
DA	X			OR C,/Bit	X		
RRC	X			MOV C,Bit	X		
RLC	X			CJNE	X		
SETB C	1						

INSTRUCTION SET

ASSEMBLER DIRECTIVES

- ORG (origin):
  - ✓ This directive is used to indicate the beginning of the addresses the number that comes after ORG can be either in hex or in decimal.

Ex:- ORG 0000H

- EQU (equate):
  - ✓ This is used to define a constant without occupying a memory location.
  - ✓ The EQU directive does not set aside storage for a data item but associates a constant value with a data label
  - ✓ So that when the label appears in the program it constant value will be substituted for the label.

Ex:- COUNT EQU 25
MOV R3, # COUNT

INSTRUCTION SET

ASSEMBLER DIRECTIVES

- END:
  - ✓ END directive indicates to the assembler at the end of the source (asm) file
  - ✓ END directive is the last line of an 8051 program meaning that in the source code anything after the END directive is ignored by the assembler.

- DB (Define byte):-
  - ✓ It is used to define the 8-bit data when DB is used to define data, the numbers can be in decimal binary, hex .
  - ✓ DB directive is the only directive that can be used to define ASCII strings larger than the character therefore it should be used for all ASCII data definitions.

Ex: -

DATA 1: DB 28 (Decimal)
DATA 2: DB 39H (HEX)
DATA 3: DB 0101001 B (Binary)
DATA 4: DB "Hello World!!!"

ASSEMBLY PROGRAMMING

ASSEMBLY PROGRAMMING

EXAMPLE-1

Write a program to move a block of FIVE data starting from RAM 40H to external RAM memory 3000H onwards and perform complement operation before storing to RAM.

```
        ORG 0000H
        MOV R0, #40H
        MOV DPTR, #3000H
        MOV R2, #05H
LOOP:   MOV A,@R0
        CPL A
        MOVX @DPTR,A
        INC R0
        INC DPTR
        DJNZ R2, LOOP
        END
```

ASSEMBLY PROGRAMMING

EXAMPLE-2

Assume that RAM bit location 24H holds the status of PIR sensor to indicate whether there has been a human presence or not inside a hall. Write an 8051 assembly language program to set port P1.0 bit as high if 24H is high. If it is low, set port P1.0 bit as low.

```
        ORG 0000H
        MOV P1, #00H
AGAIN:  MOV C, 24H
        JNC NO
        SETB P1.0
        SJMP AGAIN
NO:     CLR P1.0
        SJMP AGAIN
        END
```

ASSEMBLY PROGRAMMING

EXAMPLE-3

Write an 8051 ASM program to read a temperature value (T) from RAM location 55H. According to the test results, place the temperature value into the registers indicated below:

- If T=30H then A=30H
- If T<30H then R1=T
- If T>30H then R2=T

```
ORG 0000H
        MOV A, 55H
        CJNE A, #30H, OVER
        SJMP EXIT
OVER:   JNC NEXT
        MOV R1, A
        SJMP EXIT
NEXT:   MOV R2, A
EXIT:   NOP
        END
```

ASSEMBLY PROGRAMMING

EXAMPLE-4

Write an assembly language program to count number of 1's and 0's in an eight bit number. Store the number of 1's in R1 and Number of zeros in R2.

```
MOV   R1,#00H    // TO COUNT NUMBER OF 0S
MOV   R2,#00H    // TO COUNT NUMBER OF 1S
MOV   R7,#08H    // COUNTER FOR 8-BITS
MOV   A,#97H    // DATA TO COUNT NO. OF 1S & 0S
AGAIN:  RLC   A
JC    NEXT
INC   R2
SJMP   HERE
NEXT:  INC   R1
HERE:  DJNZ   R7,AGAIN
END
```

EXAMPLE-5

Show the status of the CY, AC, OV and P flag after executing following instructions

(a) MOV A, #38H
ADD A, #2FH

(b) MOV A, #76H
SUBB A, #97H

(a) MOV A,#38H ;A=38H= 0011 1000 B

ADD A, #2FH ;A=38H + 2FH =67 = 0110 0111 B

```
   38     0011 1000
 + 2F     0010 1111
          0110 0111
```

CY=0; AC=1; OV=0; P=1

(b) MOV A,#76H ;A=76H= 0111 0110 B

SUBB A, #97H ;A=76H – 97H =-21H

```
   76     0111 0110
 - 97     1001 0111
      1)  1101 1111
```

CY=1; AC=1; OV=1; P=1

ASSEMBLY PROGRAMMING

EXAMPLE-6

(a) Find the size of the delay in the following program if the crystal frequency is 11.0592 MHz.

```
DELAY:   MOV R3, #250   ; 1MC
HERE:    NOP            ; 1MC
         NOP            ; 1MC
         NOP            ; 1MC
         NOP            ; 1MC
         DJNZ R3, HERE  ; 2MC
         RET            ; 2MC
```

Solution:

- The time delay inside HERE loop is [250x(1+1+1+1+2)]x1.085 μs = 1627.5 μs
- Adding two instructions outside loop, total time delay is 1627.5 μs + (3x1.085 μs) = 1630.755 μs

(b) Find the size of the delay in the following program if the crystal frequency is 11.0592 MHz.

```
DELAY:   MOV R2, #200    ; 1MC
AGAIN:   MOV R3, #250    ; 1MC
HERE:    NOP             ; 1MC
         NOP             ; 1MC
         DJNZ R3, HERE   ; 2MC
         DJNZ R2,AGAIN   ; 2MC
         RET             ; 2MC
```

Solution:

- HERE loop Time delay : [250x(1+1+2)]x1.085=1085μs
- For AGAIN loop, repeat HERE loop 200 times, so 200x1085 μs = 217000 μs
- But, "MOV R3,#250" and "DJNZ R2, AGAIN" at the start and end of AGAIN loop [200x(1+2)]x1.085 μs = 651 μs
- As a result, total time delay 217000 + 651 = 217651 μs

ASSEMBLY PROGRAMMING

EXAMPLE-7

(a) Show the stack and stack pointer for the following instruction execution

```
MOV R6, #25H
MOV R1, #12H
MOV R4, #0F3H
PUSH 6
PUSH 1
PUSH 4
```

START SP=07H

0B	
0A	
09	
08	

AFTER PUSH 6

SP=08H

0B	
0A	
09	
08	25

AFTER PUSH 1

SP=09H

0B	
0A	
09	12
08	25

AFTER PUSH 4

SP=0AH

0B	
0A	F3
09	12
08	25

(b) Show the content of register and stack pointer after execution of following instruction

```
POP 3      ; POP stack into R3
POP 5      ; POP stack into R5
POP 2      ; POP stack into R2
```

START SP=0BH

0B	54
0A	F9
09	76
08	6C

AFTER POP 3

SP=0AH

0B	
0A	F9
09	76
08	6C

AFTER POP 5

SP=09H

0B	
0A	
09	76
08	6C

AFTER POP 2

SP=08H

0B	
0A	
09	
08	6C

ASSEMBLY PROGRAMMING

EXAMPLE-8

The cost prices of 10 items are stored in RAM location starting from 50H onwards. When these items are sold out as indicated by a "HIGH" on a switch, the cost price is replaced by the selling price by adding a constant profit of 19 to each value. This changed data is sent out through port 1. Write a program for this scenario.

```
ORG 0000H
SW EQU P0.1
DATA EQU P1
SETB SW
MOV DATA,#00H
NEXT: MOV C,SW
JNC NEXT
MOV R1, #10
……….
……….
MOV R0, #50H
BACK: MOV A, @R0
ADD A, #19
MOV @R0, A
MOV DATA, A
INC R0
DJNZ R1, BACK
END
```

ASSEMBLY PROGRAMMING

EXAMPLE-9

Write a program to check the port P1 whether it has a value of 45H. If so, send 99H to P2; otherwise it stays cleared.

```
ORG 0000H
MOV P2, #00H
MOV P1, #0FFH
MOV R3, #45H
MOV A, P1
XRL A,R3
JNZ EXIT
MOV P2, #99H
EXIT:NOP
END
```

ASSEMBLY PROGRAMMING

EXAMPLE-10

Write a program to arrange 5 numbers stored in location starting from 30H in descending order.

```
ORG 0000H
MOV R4,#05H
AGAIN: MOV R3,#05H
MOV R0,#30H
CLR C
UP:MOV A,@R0
MOV R1,A
INC R0
MOV A,@R0
SUBB A,R1
.................
```

```
...........
JC SKIP
MOV A,@R0
DEC R0
MOV @R0,A
MOV A,R1
INC R0
MOV @R0,A
SKIP:DJNZ R3,UP
DJNZ R4,AGAIN
RET
```

Bubble Sort – Descending Order

	0	1	2	3	4
Pass 1	2	3	4	5	1
Pass 2	3	4	5	2	1
Pass 3	4	5	3	2	1
Pass 4	5	4	3	2	1

* By changing "JC SKIP" instruction into "JNC SKIP" ascending order can be achieved

ASSEMBLY PROGRAMMING

EXERCISE

1. Write an 8051 assembly language program to add the first 10 natural numbers

2. Write an ALP to compare two eight bit numbers NUM1 and NUM2 stored in external memory locations 8000h and 8001h respectively. Reflect your result as:
   - If NUM1<NUM2, SET LSB of data RAM location 2FH (bit address 78H).
   - If NUM1>NUM2, SET MSB of location 2FH (bit address 7FH).
   - If NUM1 = NUM2, then CLR both LSB & MSB of bit addressable memory location 2FH.

3. Write an 8051 ALP to find whether given eight bit number is odd or even. If odd store 00h in accumulator. If even store FFH in accumulator.

4. Write an assembly language program to find the smallest/largest element in a given string of n = 5 bytes at location 30h. Store the smallest/largest element at location 38h.

THANK YOU

By
prakash v