SNS COLLEGE OF TECHNOLOGY

(An Autonomous Institution)

COIMBATORE-35

Accredited by NBA-AICTE and Accredited by NAAC – UGC with A+ Grade

Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

COURSE NAME: 19EEB303 - MICROCONTROLLER AND ITS APPLICATIONS

III YEAR / VI SEMESTER

Unit 1 – INTRODUCTION

UNIT-1
INTRODUCTION

Introduction to Microprocessors and Microcontrollers, Architecture of 8086, Intel MCS-51 family features – ATMEL Processor – 8051 -organization and architecture, Addressing modes, Instruction set format, Interrupts.

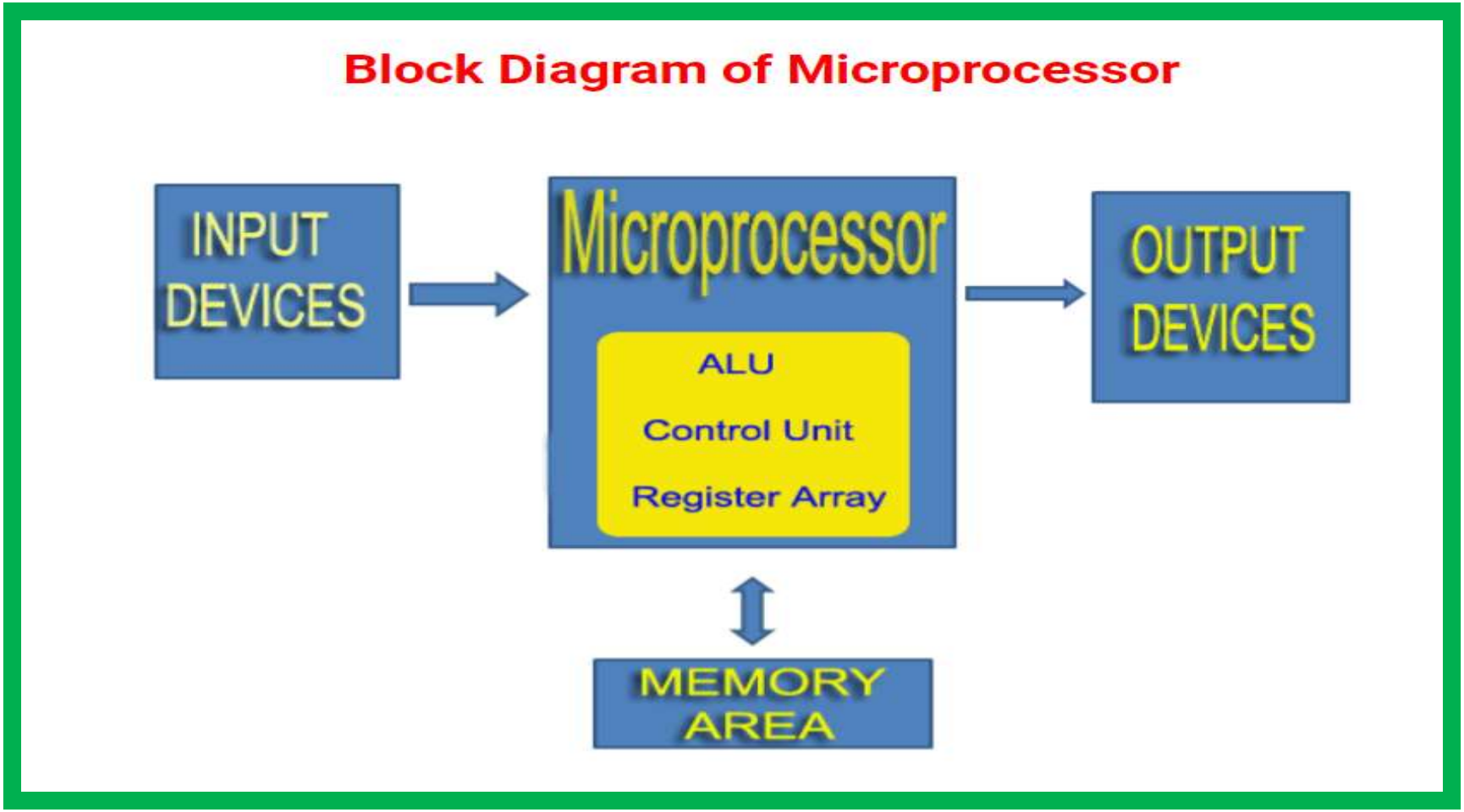
CONTENTS

- Introduction to Microprocessors and Microcontrollers
- Architecture of 8086
- Intel MCS-51 family features
- ATMEL Processor
- 8051 -organization and architecture
- Addressing modes
- Instruction set format
- Interrupts

MICROPROCESSOR

- controlling unit of a micro-computer, fabricated on a small chip capable of performing ALU operations and communicating with the other devices connected to it.
- consists of
  - ALU - arithmetical and logical operations on the data received from the memory or an input device.
  - Register array- consists of registers identified by letters like B, C, D, E, H, L and accumulator.
  - Control unit- controls the flow of data and instructions within the computer.

MICROPROCESSOR

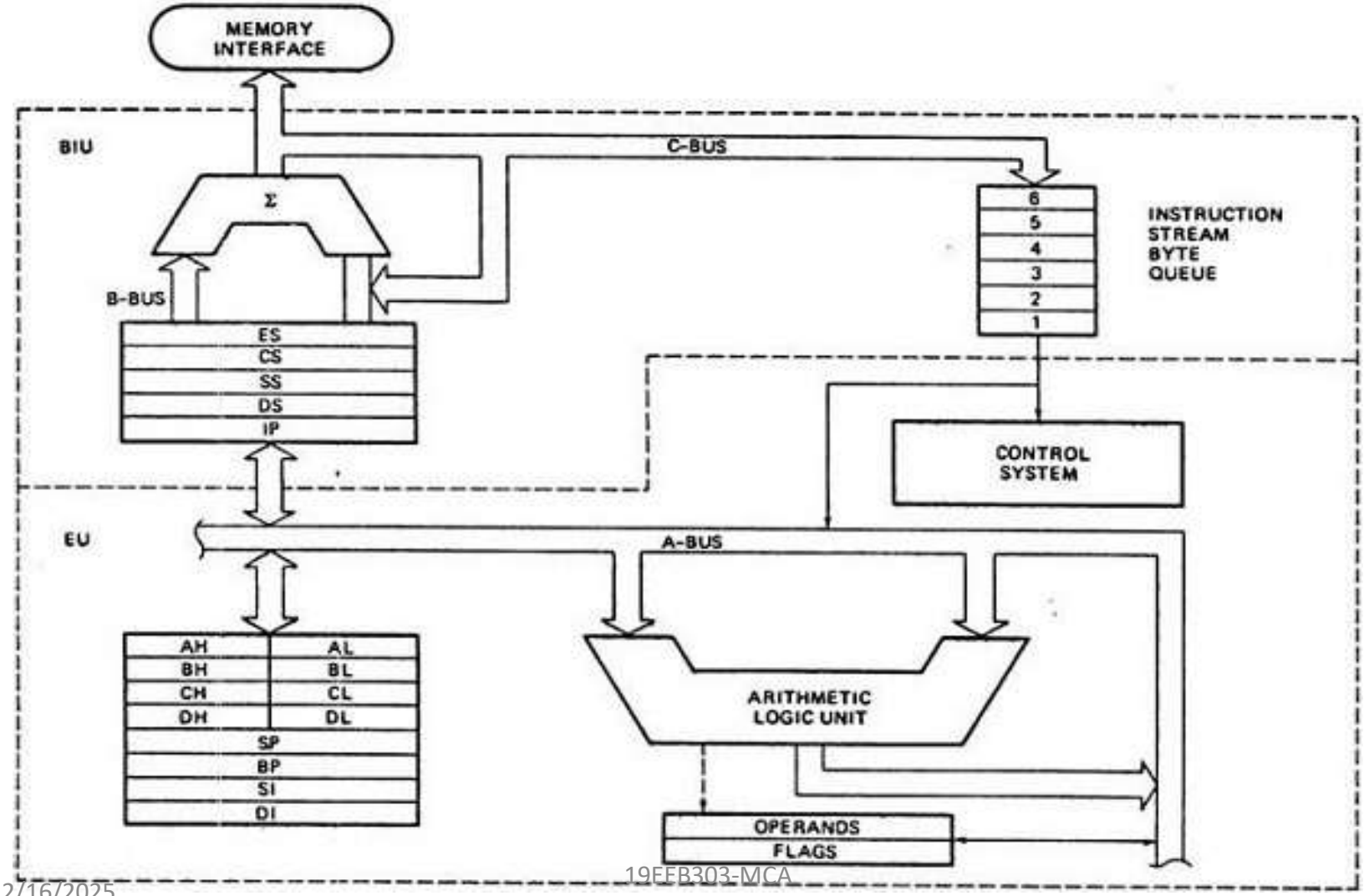


WORKING OF MICROPROCESSOR

- The microprocessor follows a sequence:
  - Fetch,
  - Decode
  - Execute.
- Initially, the instructions are stored in the memory in a sequential order.
- The microprocessor fetches those instructions from the memory, then decodes it and executes those instructions till STOP instruction is reached.
- Later, it sends the result in binary to the output port.
- Between these processes, the register stores the temporarily data and ALU performs the computing functions.

ARCHITECTURE OF 8086

Two functional units-**EU** (Execution Unit) and **BIU** (Bus Interface Unit)



EU (Execution Unit)

- EU gives instructions to BIU stating from where to fetch the data and then decode and execute those instructions.
- Function-to control operations on data using the instruction decoder & ALU.
- EU has no direct connection with system buses, it performs operations over data through BIU.

FUNCTIONAL UNIT OF 8086

- ALU- handles all arithmetic and logical operations, like +, −, ×, /, OR, AND, NOT operations.
- Stack pointer register-16-bit register, holds the address from the start of the segment to the memory location, where a word was most recently stored on the stack.
- Flag Register-16-bit register that behaves like a flip-flop, i.e. it changes its status according to the result stored in the accumulator.
  - It has 9 flags and they are divided into 2 groups −
  1. Conditional Flags- represents the result of the last arithmetic or logical instruction executed.
  2. Control Flags- controls the operations of the execution unit.

FLAG REGISTER- CONDITIONAL FLAG

- **Carry flag** −indicates an overflow condition for arithmetic operations.
- **Auxiliary flag** − When an operation is performed at ALU, it results in a carry/barrow from lower nibble (i.e. D0 – D3) to upper nibble (i.e. D4 – D7), then this flag is set
- **Parity flag** − used to indicate the parity of the result
- **Zero flag** − set to 1 when the result of arithmetic or logical operation is zero else it is set to 0.
- **Sign flag** − holds the sign of the result, i.e. when the result of the operation is negative, then the sign flag is set to 1 else set to 0.
- **Overflow flag** − represents the result when the system capacity is exceeded.

FLAG REGISTER- CONTROL FLAG

- **Trap flag** − used for single step control and allows the user to execute one instruction at a time for debugging. If it is set, then the program can be run in a single step mode.
- **Interrupt flag** − It is an interrupt enable/disable flag, i.e. used to allow/prohibit the interruption of a program. It is set to 1 for interrupt enabled condition and set to 0 for interrupt disabled condition.
- **Direction flag** − used in string operation. When it is set then string bytes are accessed from the higher memory address to the lower memory address and vice-a-versa.

FUNCTIONAL UNIT OF 8086

- General purpose register- 8 general purpose registers, i.e., AH, AL, BH, BL, CH, CL, DH, and DL.
- The valid register pairs are AH and AL, BH and BL, CH and CL, and DH and DL. It is referred to the AX, BX, CX, and DX respectively.
- **AX register** − It is also known as accumulator register. It is used to store operands for arithmetic operations.
- **BX register** − It is used as a base register. It is used to store the starting base address of the memory area within the data segment.
- **CX register** − It is referred to as counter. It is used in loop instruction to store the loop counter.
- **DX register** − This register is used to hold I/O port address for I/O instruction

BIU (Bus Interface Unit)

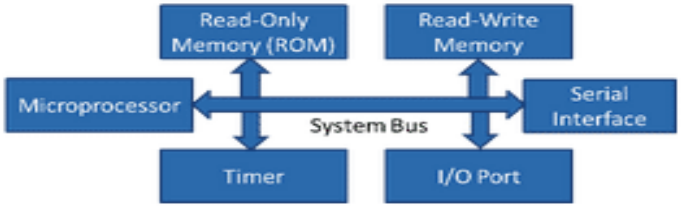
- BIU takes care of all data and addresses transfers on the buses for the EU like sending addresses, fetching instructions from the memory, reading data from the ports and the memory as well as writing data to the ports and the memory.
- EU and BIU are connected with the Internal Bus.

BIU (Bus Interface Unit)-FUNCTIONAL PARTS

- **Instruction queue** − BIU contains the instruction queue. BIU gets upto 6 bytes of next instructions and stores them in the instruction queue.
- Fetching the next instruction while the current instruction executes is called **pipelining**.
- **Instruction pointer** − 16-bit register used to hold the address of the next instruction to be executed.

BIU (Bus Interface Unit)-FUNCTIONAL PARTS

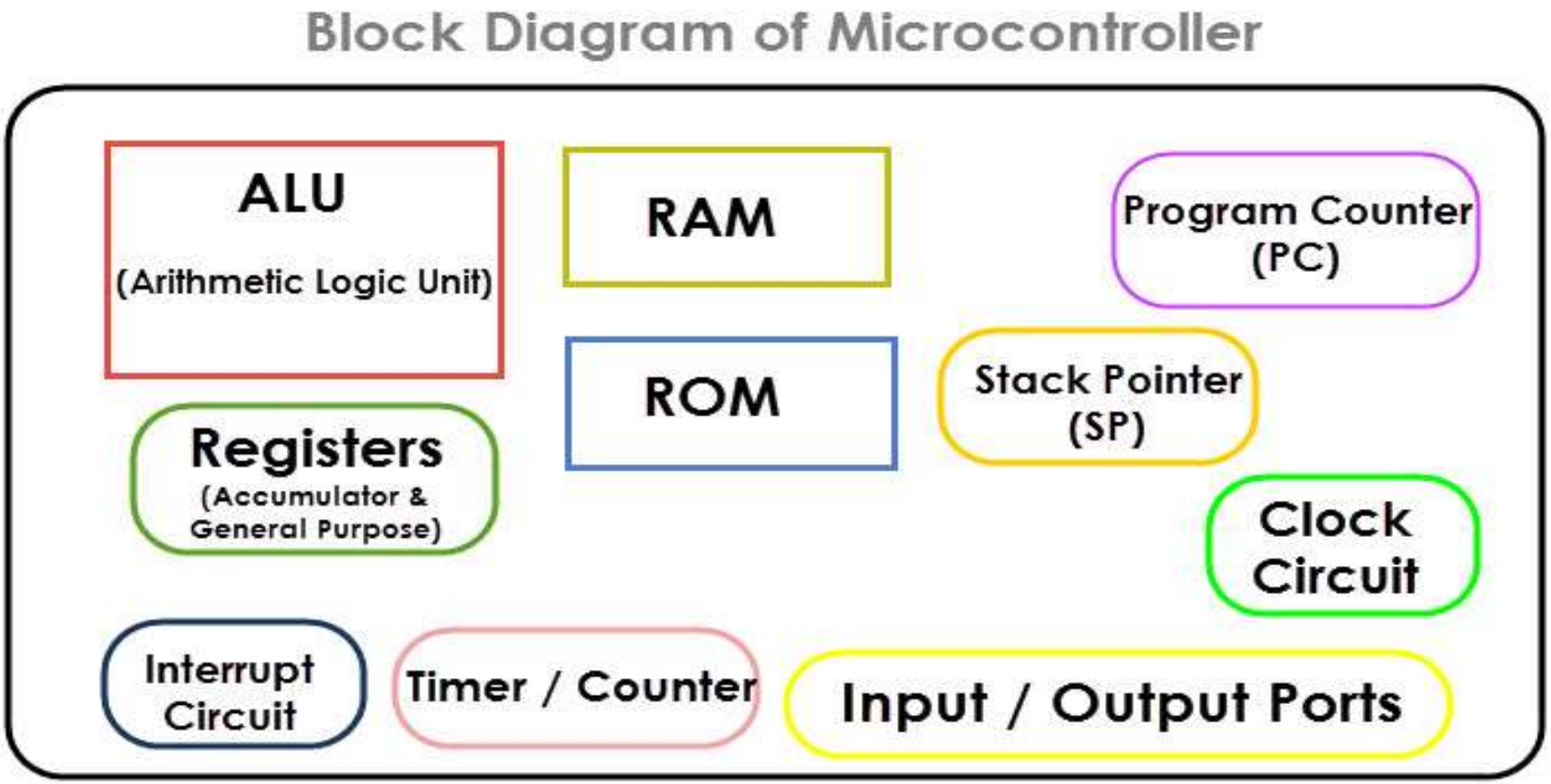
- **Segment register** − BIU has 4 segment buses, i.e. CS, DS, SS& ES. It holds the addresses of instructions and data in memory, which are used by the processor to access memory locations.
  - **CS** (Code Segment)-used for addressing a memory location in the code segment of the memory, where the executable program is stored.
  - **DS** (Data Segment)- consists of data used by the program and is accessed in the data segment by an offset address or the content of other register that holds the offset address.
  - **SS** (Stack Segment)-handles memory to store data and addresses during execution.
  - **ES** (Extra Segment)-additional data segment, used by the string to hold the extra destination data.

Microprocessor	Micro Controller
	
Microprocessor is heart of Computer system.	Micro Controller is a heart of embedded system.
It is just a processor. Memory and I/O components have to be connected externally	Micro controller has external processor along with internal memory and i/O components
Since memory and I/O has to be connected externally, the circuit becomes large.	Since memory and I/O are present internally, the circuit is small.
Cannot be used in compact systems and hence inefficient	Can be used in compact systems and hence it is an efficient technique
Cost of the entire system increases	Cost of the entire system is low
Due to external components, the entire power consumption is high. Hence it is not suitable to used with devices running on stored power like batteries.	Since external components are low, total power consumption is less and can be used with devices running on stored power like batteries.
Most of the microprocessors do not have power saving features.	Most of the micro controllers have power saving modes like idle mode and power saving mode. This helps to reduce power consumption even further.
Since memory and I/O components are all external, each instruction will need external operation, hence it is relatively slower.	Since components are internal, most of the operations are internal instruction, hence speed is fast.
Microprocessor have less number of registers, hence more operations are memory based.	Micro controller have more number of registers, hence the programs are easier to write.
Microprocessors are based on von Neumann model/architecture where program and data are stored in same memory module	Micro controllers are based on Harvard architecture where program memory and Data memory are separate
Mainly used in personal computers	Used mainly in washing machine, MP3 players

MICROCONTROLLER

- small and low-cost microcomputer
- designed to perform the specific tasks of embedded systems(displaying microwave's information, receiving remote signals, etc).
- The general microcontroller consists of
  - processor
  - memory (RAM, ROM, EPROM)
  - Serial ports
  - peripherals (timers, counters)

MICROCONTROLLER



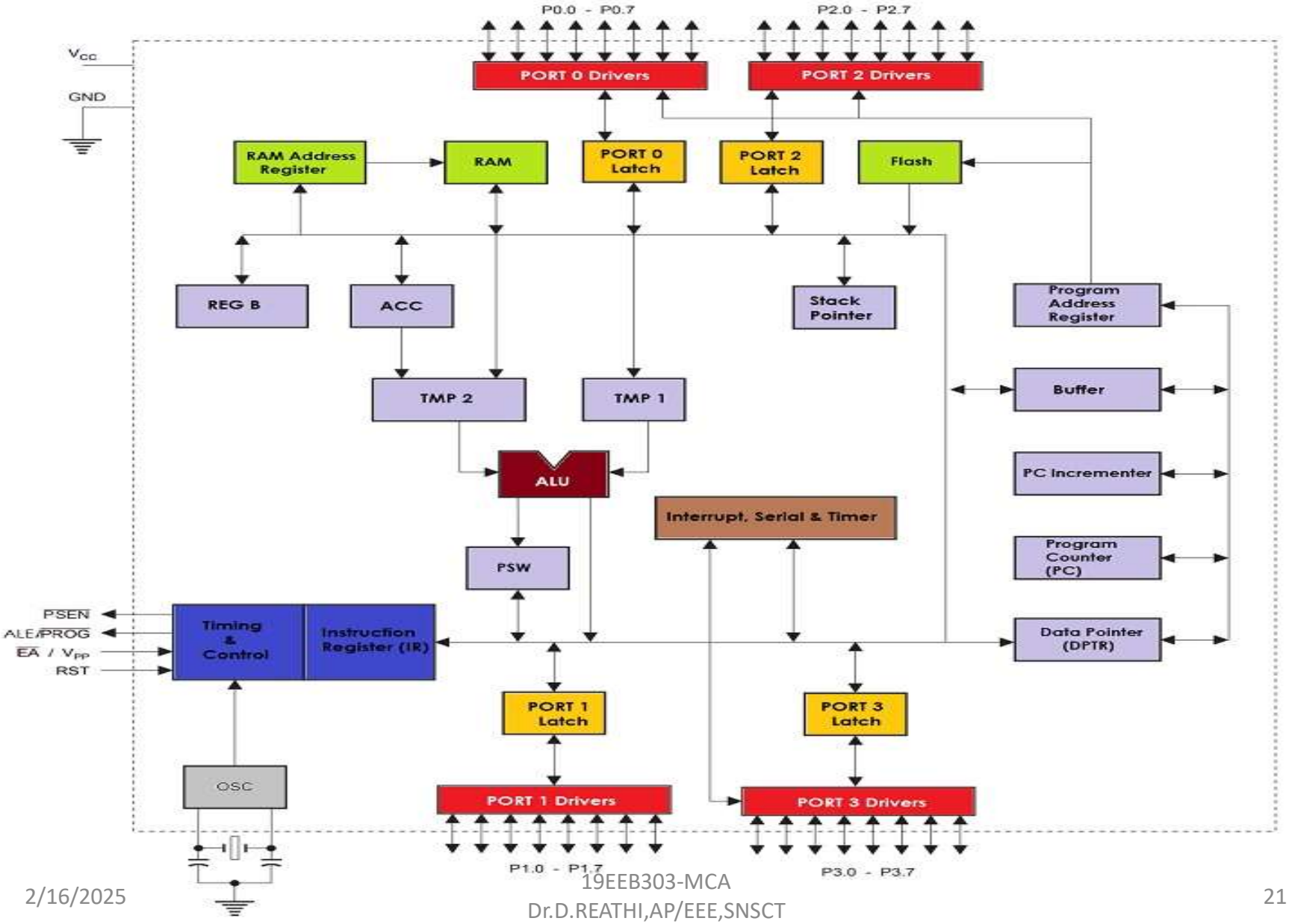
MICROCONTROLLER-8051 ARCHITECTURE

- 8051 microcontroller is designed by Intel in 1981.
- On-chip crystal oscillator is integrated in the microcontroller having crystal frequency of 12 MHz.
- Technically called as Intel MCS-51 Architecture, the 8051 microcontroller series was developed by Intel in the year 1980.
- 8 – bit CPU with two Registers A (Accumulator) and B.
- Internal ROM of 8K Bytes – It is a flash memory that supports in – system programming.
- Internal RAM of 256 Bytes – The first 128 Bytes of the RAM i.e. 00H to 7FH is again divided in to 4 banks with 8 registers (R0 – R7) in each bank, 16 bit addressable registers and 80 general purpose registers. The higher 128 Bytes of the RAM i.e. 80H to FFH consists of SFRs or Special Function Registers.

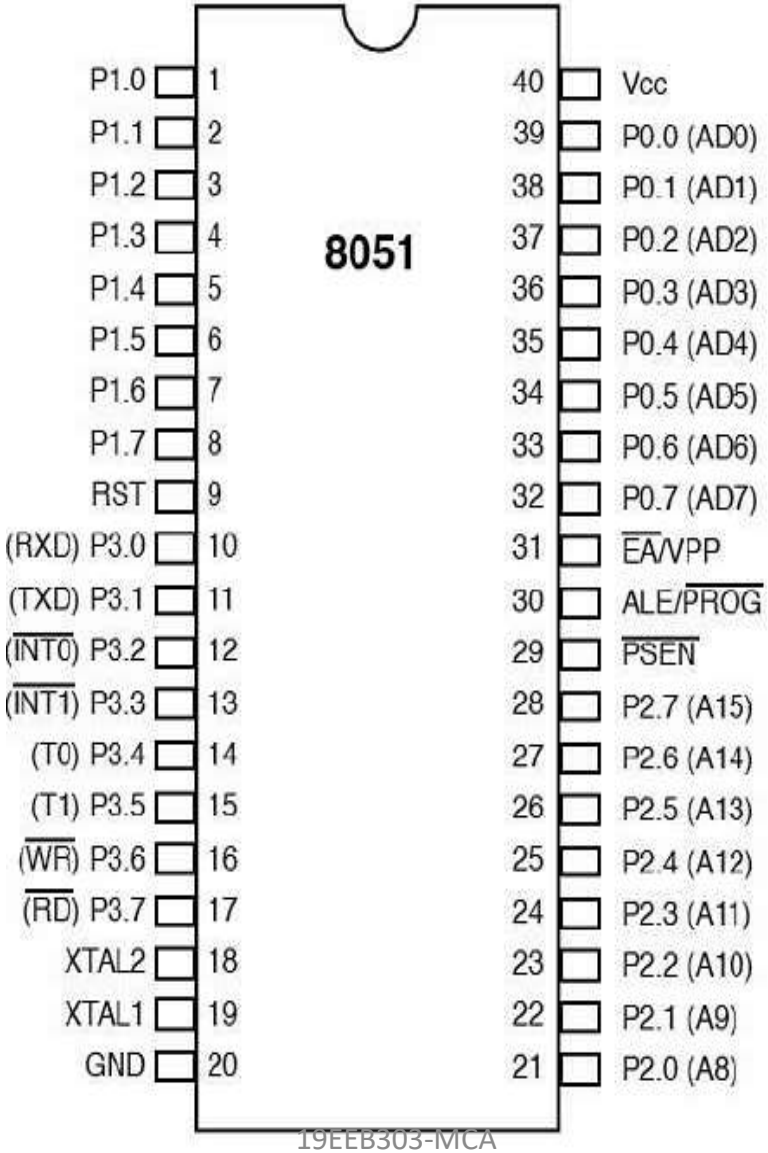
CONTD….

- Using SFRs we can control different peripherals like Timers, Serial Port, all I/O Ports, etc.
- consists of are four parallel 8-bit ports, which are programmable as well as addressable as per the requirement.
- 32 I/O Pins (Input / Output Pins) – Arranged as 4 Ports: P0, P1, P2 and P3.
- 8- bit Stack Pointer (SP) and Processor Status Word (PSW).
- 16 – bit Program Counter (PC) and Data Pointer (DPTR).
- Two 16 – bit Timers / Counters – T0 and T1.
- Control Registers – SCON, PCON, TCON, TMOD, IP and IE.
- Serial Data Transmitter and Receiver for Full – Duplex Operation – SBUF.
- Interrupts: Two External and Three Internal.
- Oscillator and Clock Circuit.

8051 ARCHITECTURE



8051 PIN DIAGRAM



8051 PIN DIAGRAM

- **Pins 1 to 8** − known as Port 1 and doesn't serve any other functions. It is internally pulled up, bi-directional I/O port.
- **Pin 9** − RESET pin, which is used to reset the microcontroller to its initial values.
- **Pins 10 to 17** − known as Port 3 and serves some functions like interrupts, timer input, control signals, serial communication signals RxD and TxD, etc.
- **Pins 18 & 19** − used for interfacing an external crystal to get the system clock.
- **Pin 20** − provides the power supply to the circuit.
- **Pins 21 to 28** − known as Port 2. It serves as I/O port. Higher order address bus signals are also multiplexed using this port.

8051 PIN DIAGRAM

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

8051 ADDRESSING MODE

- Immediate Addressing Mode
- Register Addressing Mode
- Direct Addressing Mode
- Register Indirect Addressing Mode
- Indexed Addressing Mode
- Implied Addressing Mode

IMMEDIATE ADDRESSING MODE

- The operand, which follows the Opcode, is a constant data of either 8 or 16 bits.
- constant data to be stored in the memory immediately follows the Opcode.
- The constant value to be stored is specified in the instruction itself rather than taking from a register.
- The destination register to which the constant data must be copied should be the same size as the operand mentioned in the instruction.
- **Example:** MOV A, #030H
- Here, the Accumulator is loaded with 30 (hexadecimal). The # in the operand indicates that it is a data and not the address of a Register.

REGISTER ADDRESSING MODE

- In the 8051, four banks of Working Registers with eight Registers in each bank.
- In Register Addressing mode, one of the eight registers (R0 – R7) is specified as Operand in the Instruction.
- It is important to select the appropriate Bank with the help of PSW Register.
- **Example:** MOV A, R5
- Here, the 8-bit content of the Register R5 of Bank0 is moved to the Accumulator.

DIRECT ADDRESSING MODE

- In Direct Addressing Mode, the address of the data is specified as the Operand in the instruction.
- access any register or on-chip variable.
- includes general purpose RAM, SFRs, I/O Ports, Control registers.
- **Example:** MOV A, 47H
- Here, the data in the RAM location 47H is moved to the Accumulator.

REGISTER INDIRECT ADDRESSING MODE

- The address of the Operand is specified as the content of a Register.
- **Example:** MOV A, @R1
- The @ symbol indicates that the addressing mode is indirect.
- If the contents of R1 is 56H, for example, then the operand is in the internal RAM location 56H.
- If the contents of the RAM location 56H is 24H, then 24H is moved into accumulator.
- Only R0 and R1 are allowed in Indirect Addressing Mode. These register are called as Pointer registers.

INDEXED ADDRESSING MODE

- The effective address of the Operand is the sum of a base register and an offset register.
- The Base Register can be either Data Pointer (DPTR) or Program Counter (PC) while the Offset register is the Accumulator (A).
- only MOVC and JMP instructions can be used.
- useful when retrieving data from look-up tables.
- **Example:** MOVC A, @A+DPTR
- Here, the address for the operand is the sum of contents of DPTR and Accumulator.

IMPLIED ADDRESSING MODE

- there will be a single operand.
- work on specific registers only.
- also known as register specific instruction.
- **Example:** RLA; SWAPA;
- These are 1- byte instruction. The first one is used to rotate the A register content to the Left. The second one is used to swap the nibbles in A.

8051 INSTRUCTION SET

- Data Transfer Instructions
- Arithmetic Instructions
- Logical Instructions
- Boolean or Bit Manipulation Instructions
- Program Branching Instructions

DATA TRANSFER INSTRUCTION

- associated with transfer of data between registers or external program memory or external data memory.

Mnemonic	Description
MOV	Move Data
MOVC	Move Code
MOCX	Move External Data PUSH Move Data to Stack
PUSH	Move Data to Stack
POP	Copy Data from Stack
XCH	Exchange Data between two Registers
XCHD	Exchange Lower Order Data between two Registers

ARITHMETIC INSTRUCTION

- perform addition, subtraction, multiplication and division.
- include increment by one, decrement by one and a special instruction called Decimal Adjust Accumulator.
- performed by the arithmetic instructions affect flags like carry, overflow, zero, etc. in the PSW Register.

Mnemonic	Description
ADD	Addition without Carry
ADDC	Addition with Carry
SUBB	Subtract with Carry
INC	Increment by 1
DEC	Decrement by 1
MUL	Multiply
DIV	Divide

LOGICAL INSTRUCTION

- perform logical operations like AND, OR, XOR, NOT, Rotate, Clear and Swap.
- performed on Bytes of data on a bit-by-bit basis.

Mnemonic	Description
ANL	Logical AND
ORL	Logical OR
XRL	Ex-OR
CLR	Clear Register
CPL	Complement the Register
RL	Rotate a Byte to Left
RLC	Rotate a Byte and Carry Bit to Left
RR	Rotate a Byte to Right
RRC	Rotate a Byte and Carry Bit to Right
SWAP	Exchange lower and higher nibbles in a Byte

BOOLEAN OR BIT MANIPULATION INSTRUCTION

- deal with bit variables.
- special bit-addressable area in the RAM and some of the Special Function Registers (SFRs) are also bit addressable.

Mnemonic	Description
Mnemonic	**Description**
CLR	Clear a Bit (Reset to 0)
SETB	Set a Bit (Set to 1)
MOV	Move a Bit
JC	Jump if Carry Flag is Set
JNC	Jump if Carry Flag is Not Set
JB	Jump if specified Bit is Set
JNB	Jump if specified Bit is Not Set
JBC	Jump if specified Bit is Set and also clear the Bit
ANL	Bitwise AND
ORL	Bitwise OR
CPL	Complement the Bit

PROGRAM BRANCHING INSTRUCTIONS

- control the flow of program logic.

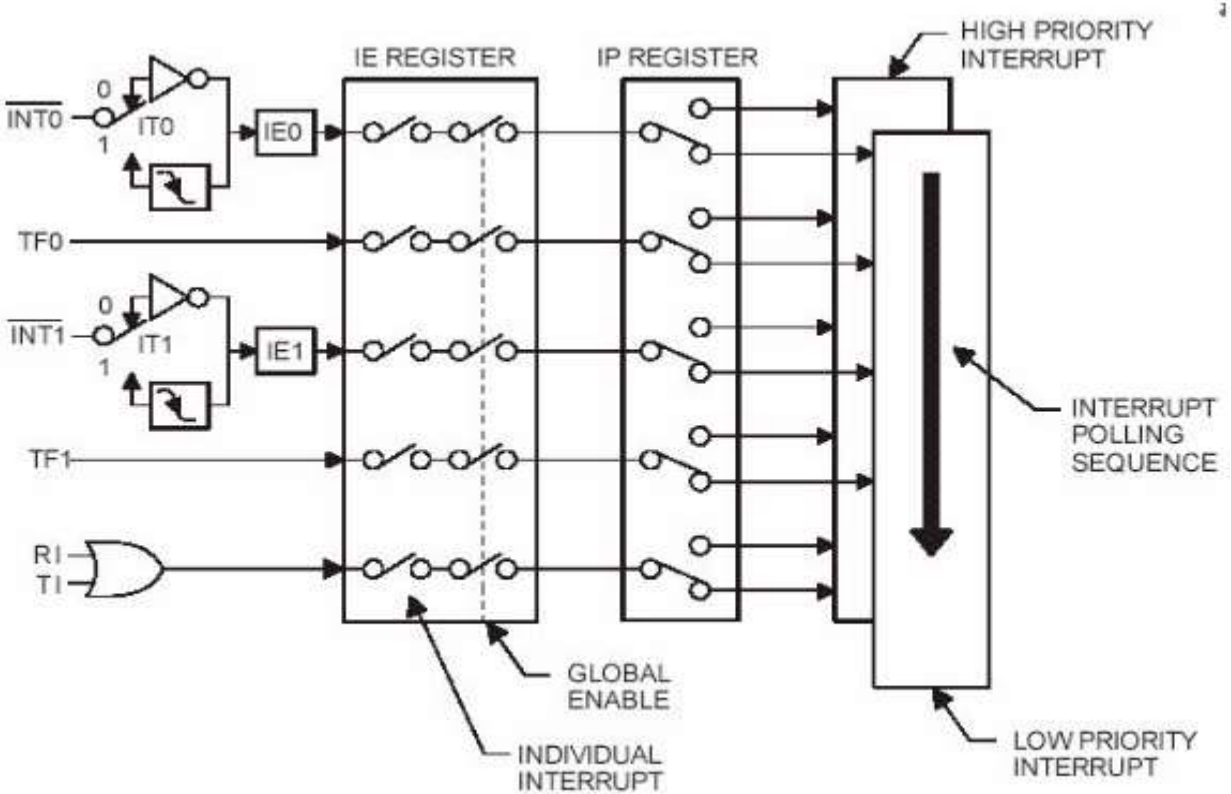
Mnemonic	Description
Mnemonic	**Description**
LJMP	Long Jump (Unconditional)
AJMP	Absolute Jump (Unconditional)
SJMP	Short Jump (Unconditional)
JZ	Jump if A is equal to 0
JNZ	Jump if A is not equal to 0
CJNE	Compare and Jump if Not Equal
DJNZ	Decrement and Jump if Not Zero
NOP	No Operation
LCALL	Long Call to Subroutine
ACALL	Absolute Call to Subroutine (Unconditional)
RET	Return from Subroutine
RETI	Return from Interrupt
JMP	Jump to an Address (Unconditional)

8051 INTERRUPT

- It is a sub-routine calls that given by the microcontroller when some other program with high priority is request for acquiring the system buses than interrupt occur in current running program.
- Interrupts provide a method to postpone or delay the current process, performs a sub-routine task and then restart the standard program again.
- five sources of interrupts are
  - Timer 0 overflow interrupt - TF0
  - External hardware interrupt - INT0
  - Timer 1 overflow interrupt - TF1
  - External hardware interrupt - INT1
  - Serial communication interrupt - RI/TI
- The timer and serial interrupts are internally produced by the microcontroller
- the external interrupts are produced by additional interfacing devices or switches that are externally connected with the microcontroller.
- These external interrupts can be level triggered or edge triggered.

8051 INTERRUPT STRUCTURE

- After 'RESET' all the interrupts get disabled, and therefore, all the interrupts is enabled by software.
- From all the five interrupts, if anyone or all interrupt are activated, this will sets the corresponding interrupt flags.



INTERRUPT ENABLE (IE) REGISTER

- used for enabling and disabling the interrupt.
- This is a bit addressable register in which EA value must be set to one for enabling interrupts.
- The individual bits in this register enables the particular interrupt like timer, serial and external inputs.

EA	--	--	ES	ET1	EX1	ET0	EX0

EA	IE.7	Disables all interrupts, If EA=0, no interrupt will be acknowledged. If EA=1, interrupt source is individually enable or disabled by setting or clearing its enable bit.
--	IE.6	Not implemented, reserved for future use*.
--	IE.5	Not implemented, reserved for future use*.
ES	IE.4	Enable or disable the Serial port interrupt.
ET1	IE.3	Enable or disable the Timer 1 overflow interrupt.
EX1	IE.2	Enable or disable External interrupt 1.
ET0	IE.1	Enable or disable the Timer 0 overflow interrupt.
EX0	IE.0	Enable or disable External interrupt 0.

INTERRUPT PRIORITY (IP) REGISTER

- possible to change the priority levels of an interrupts by clearing or setting the individual bit in (IP) register.
- allows the low priority interrupt can interrupt the high-priority interrupt, but it prohibits the interruption by using another low-priority interrupt.
- If the priorities of interrupt are not programmed, then microcontroller executes the instruction in a predefined manner and its order are INT0, TF0, INT1, TF1, and SI.