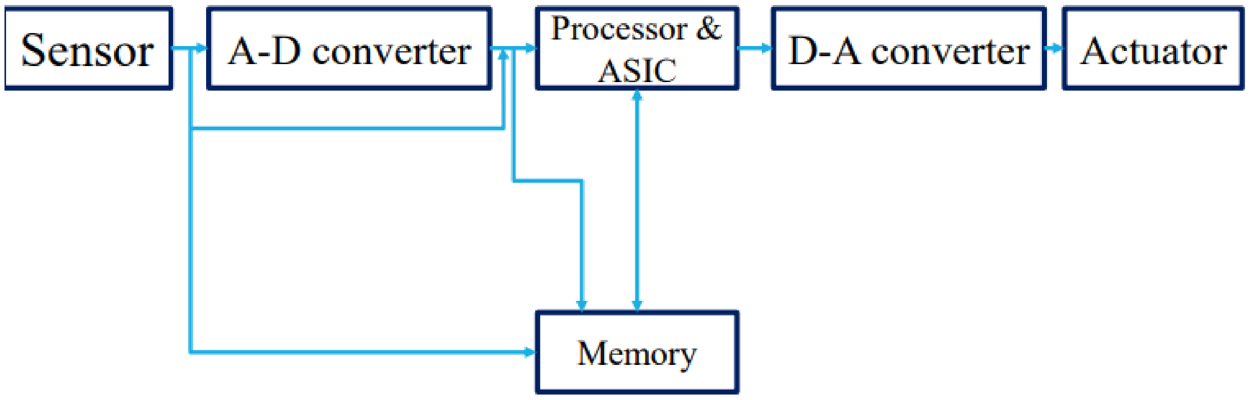
Embedded Systems

An embedded system is a microprocessor or microcontroller-based, software-driven, reliable, real-time control system. A system is an arrangement in which all its units work together according to a set of rules. All its sub-components depend on each other.

Components: Hardware, Software, Real-time OS (RTOS) supervising the application software.
Types of RTOS are Hard and Soft. Missing deadlines causes failure in hard, while it causes degraded performance in soft real-time OS.

Characteristics:

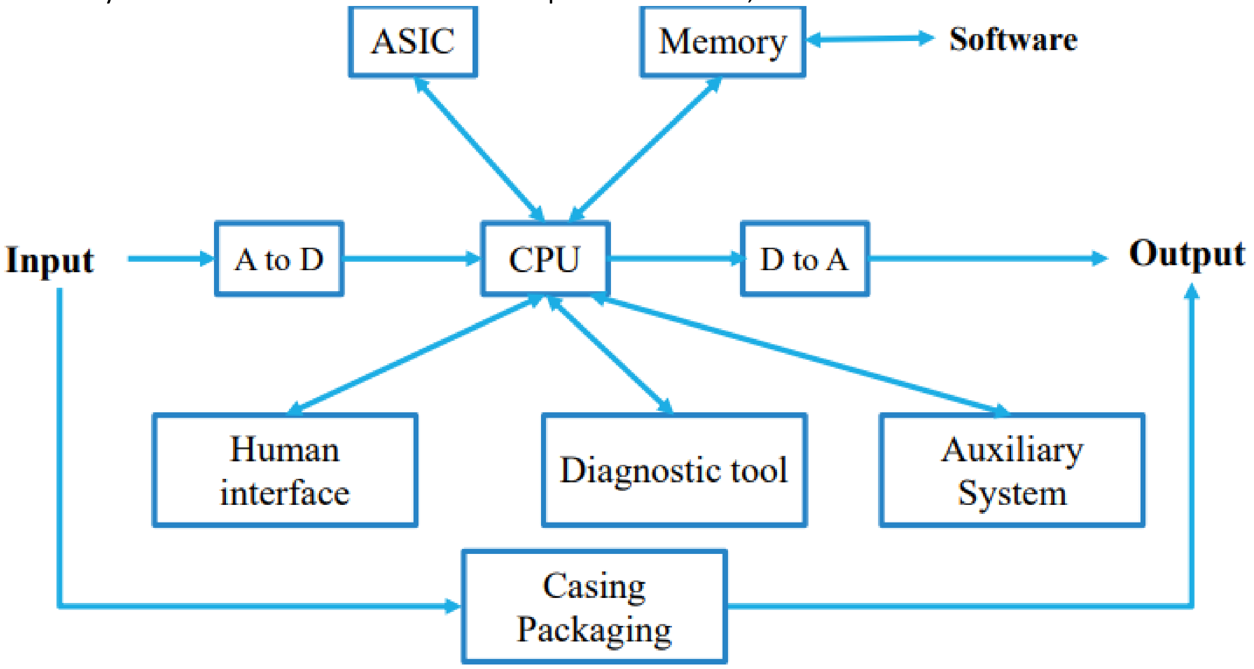
- Single functioned: Performs a specified and repeated operation.
- Tightly constrained: Design metrics are a measure of an implementation's features such as its cost, size, power, and performance.
- Reactive and real-time: It must continually react to changes in the system's environment and compute specific results without delay.
- Microprocessor or microcontroller based.
- Memory: It must have memory, as its software usually embeds in ROM. No need for secondary memories in the computer.
- Peripherals: They must be connected to input and output devices.
- Software and Hardware: For more features and flexibility. For performance and security.



Sensor – It measures a physical quantity and converts it to an electrical signal which can be read by an A-D converter. It stores the measured quantity to the memory.
A-D Converter – Analog to Digital converter converts analog signal sent by sensor into digital signal.
Processor & ASIC (Application Specific Integrated Circuit) – Processes data to measure and store output to memory.
D-A Converter – Converts digital data fed by the processor to analog data.
Actuator – Compares the output given by D-A converter to the expected output and stores it.

Auxiliary system provides additional help and support.

Hardware evolved which resulted in faster clock rate and higher degree of integration.
The fastest being System on Chip 🡐 Application specific processors 🡐 Digital Signal Processor 🡐 General purpose microprocessor / microcontroller.



Expectations:

- Program must be logically and temporally (delay) correct.
- Must deal with inherent physical concurrency. It should be a reactive system.
- Reliability and fault-tolerance are critical issues.
- Application specific and single purpose.

Some of the challenges faced are unfixed hardware requirements, meeting deadlines, and minimizing power.

Design goals are performance, functionality and UI, manufacturing cost, power consumption, size etc.

Development lifecycle is Requirements, Specifications, Architecture, Component design, System integration.

Microprocessor

An integrated circuit (IC) that incorporates core functions of a computer's central processing unit (CPU).

It's a programmable multipurpose silicon chip, clock driven, register based, accepts binary data as input and provides output after processing it as per the instructions stored in the memory. It contains an ALU, Control Unit and a Register Array.

Common terms

Bus: A set of conductors intended to transmit data, address or control information to different elements in a microprocessor. Data bus, Control bus, and Address buses are three types of buses.

Instruction set: A group of commands (Assembly) that a microprocessor can understand.

Word length: The number of bits a processor can process simultaneously.

Cache memory: A random access memory integrated into the processor.

Clock speed (Hertz): The speed at which a microprocessor executes instructions. The microprocessor uses a clock signal to control the rate at which instructions are executed, synchronize other internal components, and control the data transfer between them.

Advantages

Low cost, High speed, Small size, Low power consumption, Less heat generation, Portable, Versatile (used for no.of applications), Reliable (low failure rate).

History

1st Generation. 4-bit microprocessor 4004. Range -8 to +7. Range is 2 ^ (word size - 1).
2nd Generation. 8-bit microprocessor 8080. Range -128 to +127.
3rd Generation. 16-bit microprocessor 8086. Range -32,768 to +32,767.
4th Generation. 32-bit microprocessor 80386. Range $\pm 2\times10^9$.
5th Generation. 64-bit microprocessor.

Different types of microprocessors

CISC – Complex Instruction Set Computer
RISC – Reduced Instruction Set Computer
Special microprocessors:
ASIC – Application Specific Integrated Circuits
Superscalar Processor
DSP – Digital Signal Processor
Coprocessors
Transputer (Transistor Computer)
Input/Output Processor

CISC

- Complex Instruction Set Computer.
- Minimizes the no. of instructions per program, ignoring the no. of cycles per instruction.
- The emphasis is on building complex instructions directly into the hardware.
- The length of the code is relatively short, so the compiler's work to translate the high-level language to assembly-level/machine code is little. So, minimal RAM is required to store instructions.
- Examples are IBM 370/168, VAX 11/780, and Intel 80486.

Characteristics:

- Large no. of short and complex instructions.
- Variable length of instruction formats.
- Several cycles may be required to execute one instruction.
- Instruction-decoding logic is complex.
- Variety of addressing modes.
- One instruction should support multiple addressing modes.

RISC

- Reduced Instruction Set Computer.
- Reduces execution time by simplifying the instruction set. Each instruction requires only one clock cycle to execute results in uniform execution time.
- There are more lines of code, so more RAM is needed to store them, reducing the efficiency.
- Compiler has to work more to convert high-level language instructions to machine code.
- Examples are Power PC: 601, 604, 615, 620; DEC Alpha; PA-RISC: HP 7100LC, MIPS: TS (R10000).

Characteristics:

- Large number of long and simple instructions.
- One cycle execution time.
- Supports various data-type formats.
- Simple addressing modes and fixed length of instructions for pipelining.
- Supports register to use in any context.
- Consists of larger no. of registers, less no. of transistors.
- “LOAD” and “STORE” instructions are used to access the memory location.

Special Processors

ASIC: An Application-specific integrated circuit, is a microchip designed for a special application.

Superscalar: A CPU that implements a form of parallelism called instruction-level parallelism within a single processor.

- Can execute more than one instruction during a clock cycle by simultaneously dispatching multiple instructions to different execution units on the processor.
- Each execution unit is not a separate processor or a core if it’s a multi-core processor. It is an execution resource within a single CPU such as an ALU.

DSP: Specially designed to process analog signals to a digital form.

- Samples the voltage level at regular time intervals and converts voltage at that instant into a digital form.
- This process is performed by a circuit called an analogue to digital converter, i.e AD converter.
- Its applications are audio and video compression, graphics acceleration, video signal processing.

Coprocessor: Used to supplement the functions of th primary processor (CPU).

- By offloading main processor from intensive tasks, they accelerate system performance.
- Operations performed are arithmetic, graphics, signal processing, cryptography, I/O interfacing.
- Allows a line of computers to be customized, so that customers who don’t need the extra performance don’t need to pay for it.

Transputer (Transistor Computer): Has own local memory and links to have inter-processor communications between other transputers.

- First designed in 1980 by nmos and targeted to the utilization of VLSI technology.
- Can be used as a single processor system or can be connected to external links, which reduces the construction cost and increases the performance.

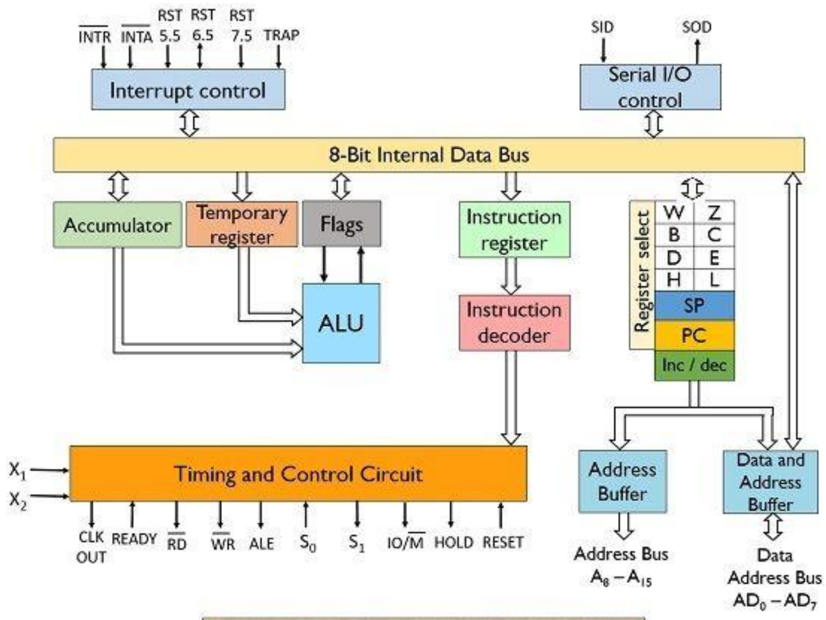
I/O processor: Local memory. Controls I/O devices with minimum CPU involvement. Ex: Keyboard, DMA.

8085 Microprocessor

Intel 8085 is 8-bit processor introduced in 1976. Address bus is of 16-bit wide. Data bus and ALU are of 8-bit. Requires 5V. Has 40 pins. Has 80 instructions. Operates with 3MHz up to 5MHz.

The processors **ALU** performs data processing.
In addition to arithmetic and logic circuits, it includes the accumulator, which is a part of every AL operation. The accumulator stores 8-bit data, result of an operation, 8-bit data during I/O transfer.

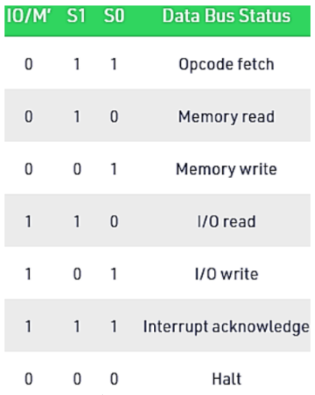
It also includes two temporary registers (ACT and TMP) used for holding data temporarily during the execution of the operation. The programmer can't access these.



Architecture of 8085 Microprocessor

Timing and control unit provides signals to the processor to perform operations. Some of them are:
Control signals: READY, RD, WR, ALE; Status Signals: IO/M, S0, S1; DMA Signals: HOLD, HLDA;
RESET Signals: RESET IN, RESET OUT.

- READY: Senses whether a peripheral is ready to transfer data or not. If READY is high (1), the peripheral is ready. If it's low (0), the processor waits till it goes high. Useful for interfacing low speed devices.
- RD: Controls the READ operation. When low, memory or I/O device in the selected location is read.
- WR: Controls the WRITE operation. When low, data on the data bus is written into the selected location.
- ALE: Address Latch Enable signal. If its value is 1, it goes high during the first T state of a machine cycle and enables the lower 8 bits of the address. Otherwise, data bus is activated.



IO/M'	S1	S0	Data Bus Status
0	1	1	Opcode fetch
0	1	0	Memory read
0	0	1	Memory write
1	1	0	I/O read
1	0	1	I/O write
1	1	1	Interrupt acknowledge
0	0	0	Halt

- IO/M: Determines whether the address is for I/O or memory. When 1, address bus is for I/O. When 0, it's for memory.
- S0, S1: Status signals to distinguish various types of operations such as halt, reading, instruction fetching or writing.
- HOLD: It indicates that another device is requesting the use of the address and data bus. If a HOLD request is received, the processor relinquishes the use of the buses as soon as the current machine cycle is completed. Internal processing may continue. After the signal is removed, the processor regains the bus.

- HLDA: This signal indicates that the hold request has been received. HLDA is set to low after the removal of HOLD signal.
- RESET IN: When the signal is low, the program-counter is set to zero. The buses are tristate, and the microprocessor unit is reset.
- RESET OUT: This indicates that the MPU is being reset. It is used to reset other devices.

General purpose registers

B, C, D, E, H and L (8-bit registers) and W, Z are temporary 8-bit registers.

- They can be used as single registers or paired as 16-bit registers (BC, DE, HL and WZ).
- HL can be used as data pointer (holds memory address).

Special Registers

Program counter

A 16-bit register used to store the memory address location of the next instruction to be executed. Microprocessor increments the program counter whenever an instruction is being executed, so that the program counter points to the memory address of the next instruction that is going to be executed.

Stack pointer

A 16-bit register works like stack, which is incremented/decremented by 2 during push & pop operations.

Program status word (Flag register (lower byte))

Flag register: 8-bit register having five 1-bit flip-flops, that holds either 0/1 depending on result stored in accumulator. The 5 flip-flops are: Sign (S), Zero(Z), Auxiliary Carry (AC), Parity (P), and Carry (C).

Sign flag: This flag is set when MSB (Most Significant Bit) of the result is 1. Negative binary numbers are represented in the 8085 CPU in standard two's complement notation, S indicates the sign of the result.
1 – MSB is 1 (negative)
0 – MSB is 0 (positive)

Zero flag: This flag is set when the result of operation is zero, else it is reset.
1 – zero result
0 – non-zero result

Auxiliary carry flag: This flag is set whenever there has been a carry out of lower nibble into higher nibble or a borrow from higher nibble into lower nibble of an 8-bit quantity, else AC is reset. Used by decimal arithmetic instructions.
1 – carry out from bit 3 on addition or borrow into bit 3 on subtraction
0 – otherwise

Parity flag: This flag is set whenever the result has even parity, an even number of 1 bit. If parity is odd, P is cleared.

Carry flag: This flag is set whenever there has been a carry out of, or a borrow into, the higher order bit of the result.

Instruction register and decoder

An 8-bit register which stores the instruction fetched from the memory. Instruction decoder decodes information present in the Instruction register.

Serial I/O control

It controls serial data communication using these two instructions: SID (Serial input data) and SOD (Serial output data).

Address bus and data bus

Data bus carries the data to be stored. Its bidirectional, whereas address bus carries the location to where it should be stored and its unidirectional. It is used to transfer the data and address I/O devices.

Interrupts in 8085

Interrupts are signals generated by the external devices to request the microprocessor to perform a task. There are 5 interrupt signals in 8085: INTR, RST 7.5, RST 6.5, RST 5.5, TRAP.

Vector interrupt: Interrupt address is known to processor. Ex: TRAP, RST 7.5, RST 6.5, RST 5.5.

Non-vector interrupt: Interrupt address is not known to processor. So, the interrupt address must be sent externally by the device to perform interrupt. Ex: INTR.

Maskable interrupt: Can disable this interrupt by writing instructions in the program. Ex: RST 7.5, 6.5, 5.5; INTR.

Non-maskable interrupt: Cannot disable this interrupt. Ex: TRAP.

Hardware interrupt: There are 5 interrupt pins in 8085 used as hardware interrupts. Ex: TRAP, INTR, RST 5.5, 6.5, 7.5.

Interrupt	Type	Instructions	Trigger	Vector
TRAP	Non-maskable	Independent of EI and DI	Level & Edge sensitive	0024 H
RST7.5	Maskable	Controlled by EI and DI	Positive edge sensitive	003C H
RST6.5	Maskable	Controlled by EI and DI	Level sensitive	0034 H
RST5.5	Maskable	Controlled by EI and DI	Level sensitive	002C H
INTR	maskable	Controlled by EI and DI	Level sensetive	0038 H

Software interrupt: Programmer must add instructions to execute this interrupt. There are 8 software interrupts in 8085. Ex: RST0, 1, 2, 3, 4, 5, 6, 7.

Software Interrupt	RST0	RST1	RST2	RST3	RST4	RST5	RST6	RST7
Vector	0000 H	0008 H	0010 H	0018 H	0020 H	0028 H	0030 H	0038 H

Interrupt Service Routine (ISR): A small program/routine, when executed, services corresponding interrupting source.

Interrupt control

As the name suggests, it controls the interrupts during a process. When a processor executes a main program and whenever an interrupt occurs, the processor shifts the control from the main program to process the incoming request. After the request is completed, the control returns to the main program.

The signals of 8085 can be **classified** into six groups: Address bus, Data bus, Control and status signals, Power supply and frequency signals, externally initiated signals, Serial I/O ports.

Microprocessor operations

It performs three types of functions:

1. Microprocessor-initiated operations
2. Internal operations
3. Peripheral operations

Microprocessor-initiated operations

- Memory read
- Memory write
- I/O read
- I/O write

Internal operations

- Store 8-bit data
- Performs arithmetic and logical operations
- Test for conditions
- Sequence the execution of instructions
- Store data temporarily during execution in the defined R/W memory locations

Peripheral operations

- Reset
- Interrupt
- Ready
- Hold

Machine Language

The number of bits that form the "word" of a microprocessor is fixed for that particular microprocessor. These bits define a maximum number of combinations. For example, an 8-bit microprocessor can have at most 2^8 = 256 different combinations. However not all these combinations are used. Certain patterns are chosen and assigned specific meanings. Specific patterns form an instruction for the microprocessor. The complete set of patterns makes up the microprocessor's machine language.

The 8085 uses a total of 246-bit patterns to form its instruction set. These 246 patterns represent only 80 instructions. The vast difference is because of the multiple different formats of most instructions. We can't use the bit patterns correctly, so hexadecimal is used instead of binary to make things easier.
Example: 0011 1100 is 3C in hexadecimal (OPCODE), which translates to "increment the number in accumulator".

But a program written in hexadecimal is difficult to understand. So, each company defines a symbolic code called "mnemonics" for these instructions. These mnemonics are usually a group of letters suggesting the operation performed by an instruction. Ex: INR is a mnemonic assigned for Increment. Machine language and the assembly language associated with it are completely **machine dependent**.

Translating assembly language into machine language can be done in two ways. First one is called "hand assembly" where the programmer translates each assembly language instruction into its equivalent hex code (machine language), which is then entered into the memory. The second one uses a program called "assembler", which does this translation automatically.

8085 Instructions

Each instruction has two parts. The first part is the operation to be performed, which is called the "**opcode**" (**op**eration **code**). The second part is the data to be operated on, which is called the "operand".

The instructions of 8085 can be grouped into five different groups:

- Data Transfer
- Arithmetic
- Logic
- Branch
- Machine Control

Data Transfer: These operations copy the data from source to destination. Ex: MOV, MVI, LDA, STA etc.
Arithmetic: Performs operations on 8-bit numbers, contents of registers/memory locations. Ex: DAA (Dec Adj Acc) etc.

Logical: These perform operations like comparing and complementing contents of accumulator, etc.
Branching: These consist of operations like JUMP, which are used to skip to different lines in a program.
Control: These include operations like HLT, enabling and disabling interrupts, etc.

Addressing Modes

1. Immediate Addressing Mode
2. Register Addressing Mode
3. Direct Addressing Mode
4. Register Indirect Addressing Mode
5. Implied/Implicit Addressing Mode

Immediate: The source operand is always data. If the data is 8-bit/16-bit, instruction will be 2/3 bytes, respectively. Example – MVI B, 45H (move data 45H immediately to register B).

Register: Register content is the operand. Example – MOV A, B (Move contents of B to A).

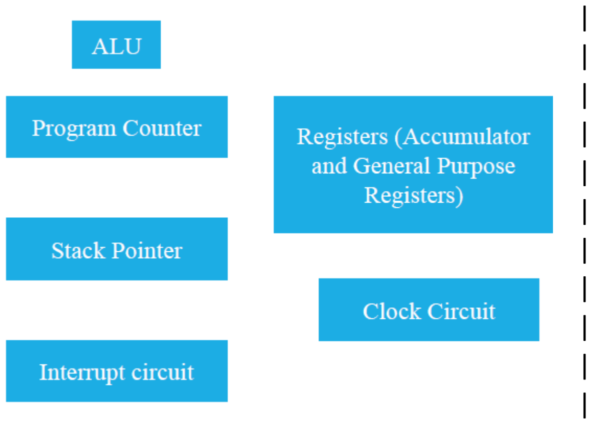
Direct: The register contains the memory location of the operand. Example – LDA 2050H (Load contents of memory location into accumulator A).

Indirect: The operand data is inside a memory location which is indirectly specified by a register pair. Example – MOV A, M (move contents of memory location pointed by the H-L pair to the accumulator).

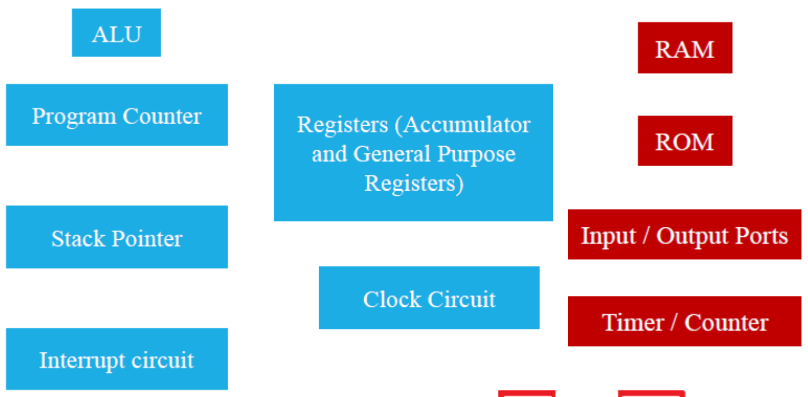
Implied/Implicit: The operand is hidden, and its data is available in the instruction itself. Example – RRC (rotate accumulator A right by one bit).

8051

Block Diagram of a Microprocessor

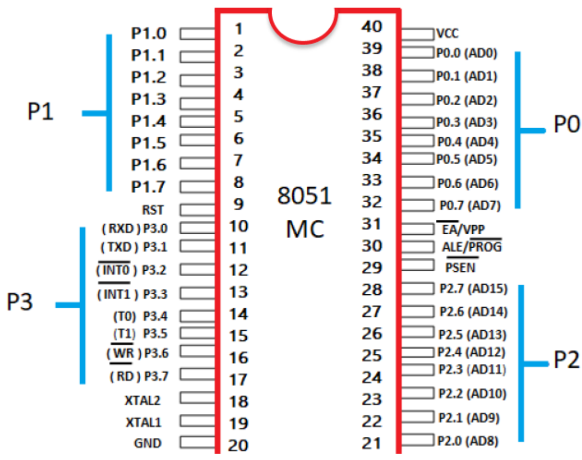


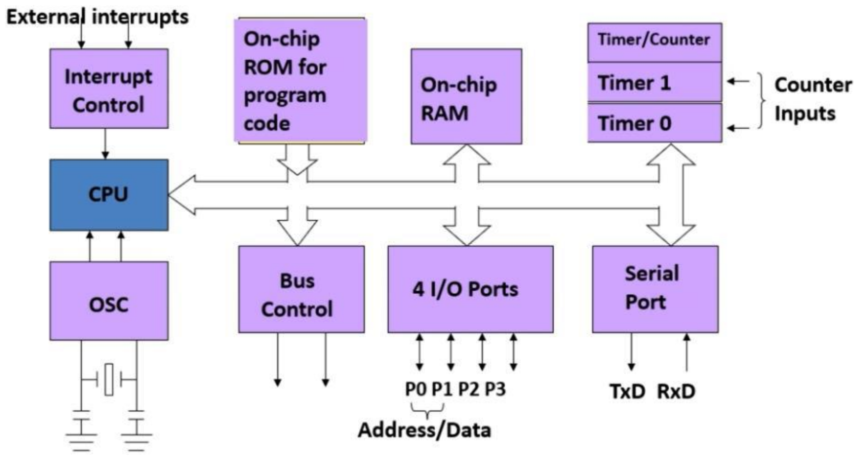
Block Diagram of a Microcontroller

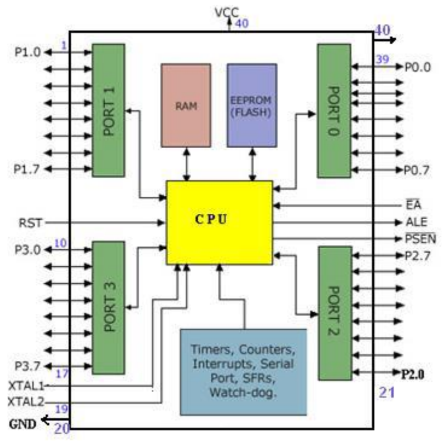


8051 microcontroller is used in consumer applications like remote controls, musical instruments, printer, robotics, etc.

Features:
→8-bit microcontroller. Separate program and data memory.
→8-bit ALU and accumulator.
→128 bytes RAM, 4 KB on-chip ROM.
→4 I/O ports of 8-bit each (bidirectional and bit addressable)
→Interrupts, on-chip oscillator, Boolean processor, power down mode, etc.







Block Diagrams of 8051

→32 (00h to 1Fh) bytes are devoted to 4 register banks, 8 register bundles each.
→16 (20h to 2Fh) bytes of bit addressable area.
→A general-purpose RAM area above the bit area, from 30h to 7Fh, addressable as bytes.
→Special function registers (SFR) are addressed like internal RAM, using addresses from 80h to FFh.
→Program counter is not an SFIR. It is a 16-bit register.

Assembly vs. C

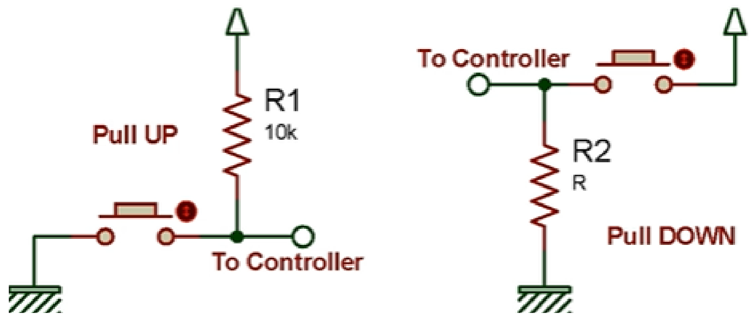
→C compiler is less efficient than Assembly.
→C is easy to understand and has strong library support, while Assembly is difficult to learn and remember.
→One can't change platform from Assembly easily.
→Compilers produce HEX files that are downloaded into the microcontroller's ROM.
→The file size is the primary concern, as microcontrollers have limited on-chip ROM.
→C programming is less time-consuming but will produce larger HEX files.

The reason why programs are written in C:

→It is easier and less time-consuming to write in C than in Assembly.
→Function libraries can be used, and C is easier to modify and update.
→C code is portable to other microcontrollers with little or no modification.
→A good understanding of C data types can help create programs with smaller HEX files.

Pull up and Pull down registers

Pull up register's is connected to the ground with default state OFF. If a switch is pressed, it completes the circuit, giving an output of HIGH or ON. Pull down register is connected to the voltage source with a default state ON. If switch is pressed, it gives an output of LOW or OFF. https://www.youtube.com/watch?v=5vnW4U5Vj0k



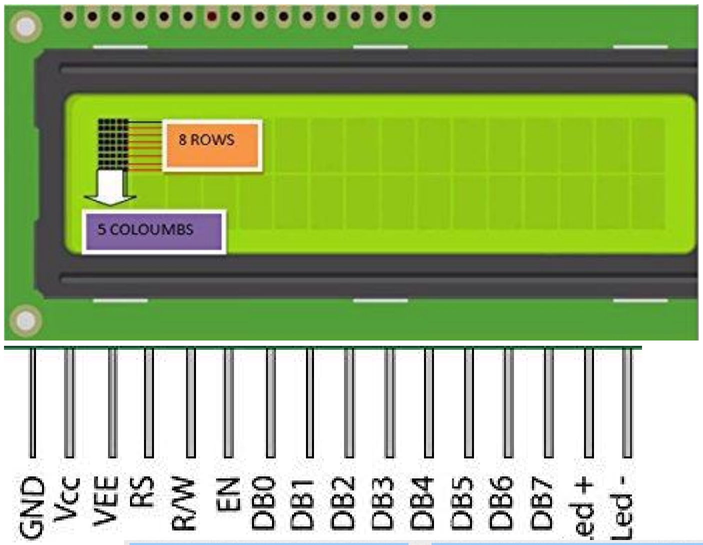
The four ports P(0,1,2,3) are 8-bit, bidirectional, and bit addressable. They are internal pull-up registers except for P0. They are configured with the default input mode as ON.

Liquid Crystal Display (LCD)

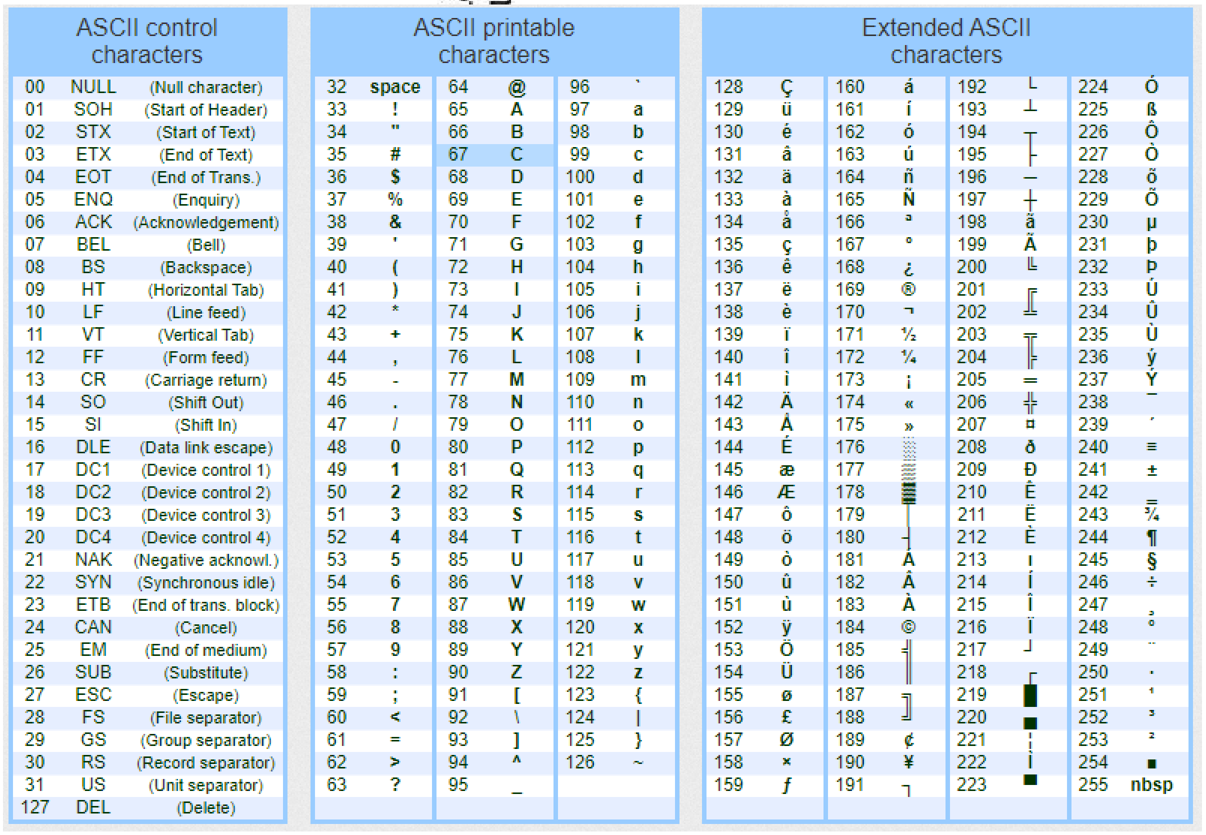
LCD screen is an electronic display module used in various applications. Different LCD types are Text LED, Graphics LED, TFT, and OLED.

A 16x2 LCD displays 16 characters per line, with 2 such lines. Each character is displayed in a 5x7 pixel matrix.

This LCD has two registers, Command and Data. Command registers store commands, Data registers store data to be displayed on the LCD.



GND and VCC: Supply Voltage
VEE: Contrast control
LED+ and LED-: ON and OFF backlight LED
RS: Register select (0 for command, 1 for data)
R/W: Read and write operation (1 for read, 0 for write)
EN: Perform R/W operation giving high to low transition on the pin
DB0 – DB1: Data pins



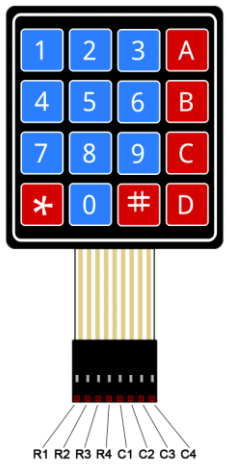
ASCII control characters		
00	NULL	(Null character)
01	SOH	(Start of Header)
02	STX	(Start of Text)
03	ETX	(End of Text)
04	EOT	(End of Trans.)
05	ENQ	(Enquiry)
06	ACK	(Acknowledgement)
07	BEL	(Bell)
08	BS	(Backspace)
09	HT	(Horizontal Tab)
10	LF	(Line feed)
11	VT	(Vertical Tab)
12	FF	(Form feed)
13	CR	(Carriage return)
14	SO	(Shift Out)
15	SI	(Shift In)
16	DLE	(Data link escape)
17	DC1	(Device control 1)
18	DC2	(Device control 2)
19	DC3	(Device control 3)
20	DC4	(Device control 4)
21	NAK	(Negative acknowl.)
22	SYN	(Synchronous idle)
23	ETB	(End of trans. block)
24	CAN	(Cancel)
25	EM	(End of medium)
26	SUB	(Substitute)
27	ESC	(Escape)
28	FS	(File separator)
29	GS	(Group separator)
30	RS	(Record separator)
31	US	(Unit separator)
127	DEL	(Delete)

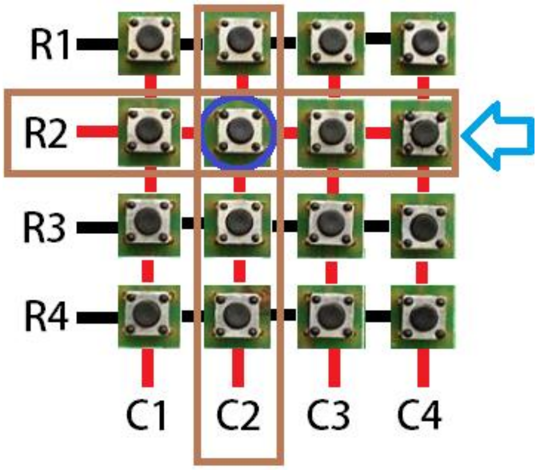
ASCII printable characters					
32	space	64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g
40	(	72	H	104	h
41	)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[	123	{
60	<	92	\	124	|
61	=	93	]	125	}
62	>	94	^	126	~
63	?	95	_		

Extended ASCII characters							
128	Ç	160	á	192	└	224	Ó
129	ü	161	í	193	┴	225	ß
130	é	162	ó	194	┬	226	Ô
131	â	163	ú	195	├	227	Ò
132	ä	164	ñ	196	─	228	õ
133	à	165	Ñ	197	┼	229	Õ
134	å	166	ª	198	ã	230	µ
135	ç	167	º	199	Ã	231	þ
136	ê	168	¿	200	╚	232	Þ
137	ë	169	®	201	╔	233	Ú
138	è	170	¬	202	╩	234	Û
139	ï	171	½	203	╦	235	Ù
140	î	172	¼	204	╠	236	ý
141	ì	173	¡	205	═	237	Ý
142	Ä	174	«	206	╬	238	¯
143	Å	175	»	207	¤	239	´
144	É	176	░	208	ð	240	≡
145	æ	177	▒	209	Ð	241	±
146	Æ	178	▓	210	Ê	242	‗
147	ô	179	│	211	Ë	243	¾
148	ö	180	┤	212	È	244	¶
149	ò	181	Á	213	ı	245	§
150	û	182	Â	214	Í	246	÷
151	ù	183	À	215	Î	247	¸
152	ÿ	184	©	216	Ï	248	°
153	Ö	185	╣	217	┘	249	¨
154	Ü	186	║	218	┌	250	·
155	ø	187	╗	219	█	251	¹
156	£	188	╝	220	▄	252	³
157	Ø	189	¢	221	¦	253	²
158	×	190	¥	222	Ì	254	■
159	ƒ	191	┐	223	▀	255	nbsp

ASCII Table

Two types of Matrix keypads: 4x3, 4x4. https://embedjournal.com/interface-4x4-matrix-keypad-with-microcontroller/





Black (rows) – 0
Red (columns) – 1

The circuit below the button will be shorted when the button is pressed. It results in the button's column getting 0 voltage.

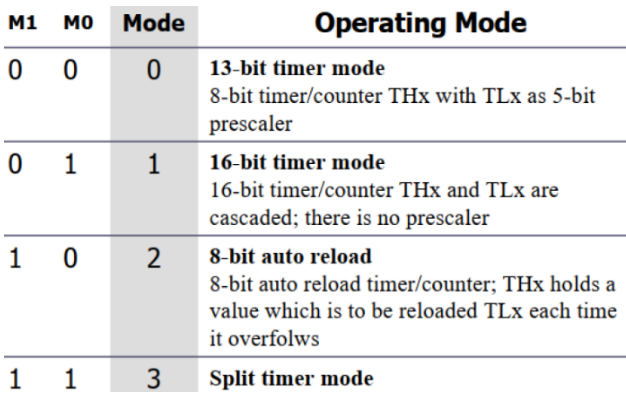
Then, each row is given a high voltage, when it reaches the row where the button is pressed, the column will return the high voltage or 1.

Timer / Counter: There are two of them, Timer/Counter 0 and 1.
A timer is used to generate a time delay. The clock source is the internal crystal frequency of the 8051,
An event counter is an external input pin to count the number of events on registers like the number of people passing through an entrance or others which can be converted to pulses.

TH0, TL0, TH1, TL1, TMOD (Timer mode register), TCON (Timer control register).

TCON is a register controlling the operations of counters and timers. It is an 8-bit register where four upper bits are for timers and counters, lower ones are for interrupts. They are TF1, TR1, TF0, TR0, IE1, IT1, IE0, IT0.

TMOD is an 8-bit register for selecting a timer or counter and mode of timers. GATE, C/Tbar, M1, M0, GATE, C/Tbar, M1, M0. The top half controls timer 1 and the bottom half controls timer 0. If the gate is set to 0, starting and stopping is done by software, if it is 1, we can perform a hardware timer. If C/T is 1, it is acting as a counter mode, else timer mode.



M1	M0	Mode	Operating Mode
0	0	0	**13-bit timer mode** 8-bit timer/counter THx with TLx as 5-bit prescaler
0	1	1	**16-bit timer mode** 16-bit timer/counter THx and TLx are cascaded; there is no prescaler
1	0	2	**8-bit auto reload** 8-bit auto reload timer/counter; THx holds a value which is to be reloaded TLx each time it overfolws
1	1	3	**Split timer mode**

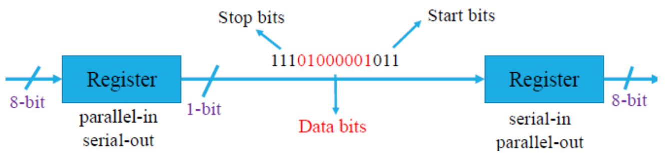
Serial Communication

Serial transfer sends and receives only from one pin like D0. Parallel transfer can transfer data via many pins D0-D7 at once.

8051 module connects to the PC using RS232, which supports half-duplex, synchronous/asynchronous, serial communication.

In Simplex, data can be sent only in one direction. In a Duplex, data can be transmitted and received. In Half duplex, data is transmitted one way at a time. In Full duplex, data is transmitted and received at the same time. Synchronous data is sent in blocks of bytes. Asynchronous data is sent in individual bytes.

The sender is a parallel-in-serial-out shift register that takes in bytes of data and converts them into bits for transmission over a single data line. The receiver is a serial-in-parallel-out shift register that receives data bits and packs them into a byte.



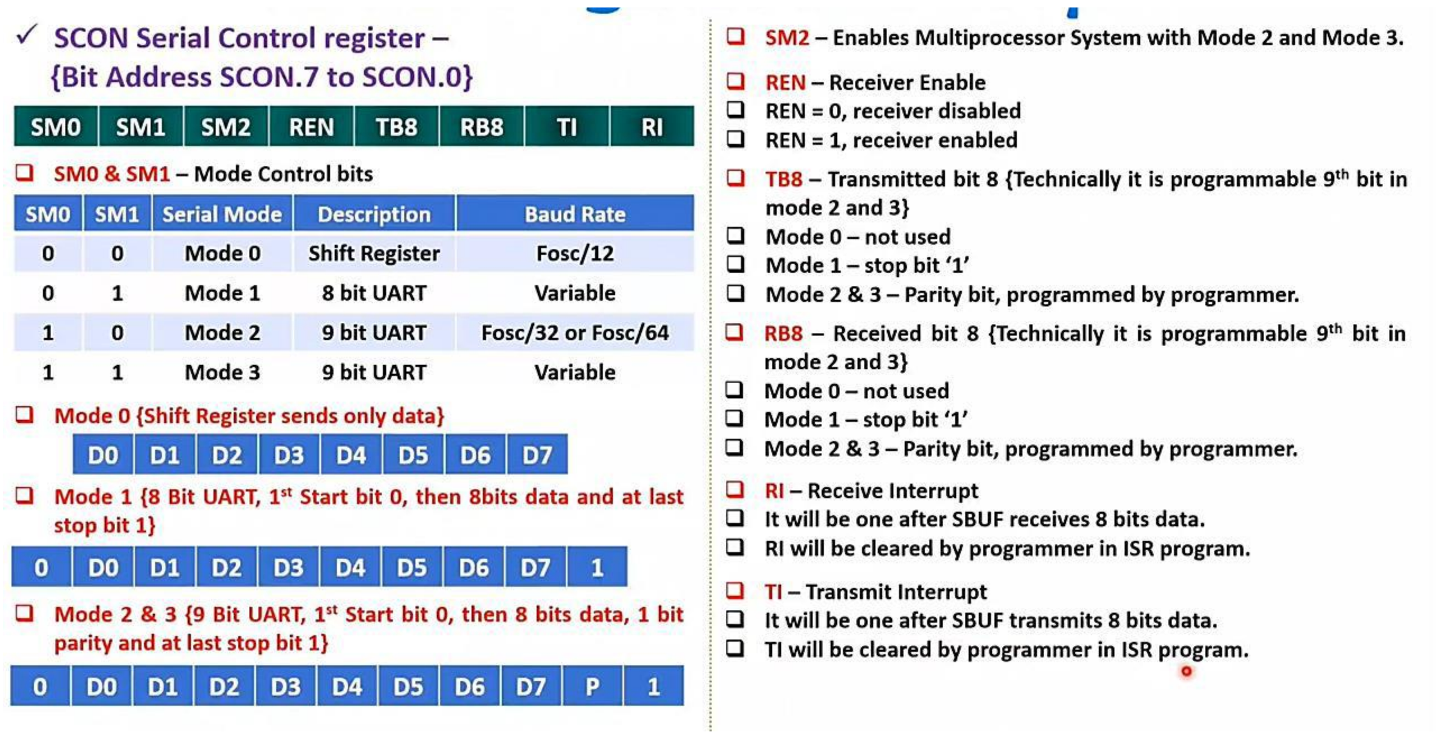
UART and USART are special Integrated Control chips made for serial communication. Writing software using synchronous and asynchronous methods is possible but tedious and long. UART is a universal asynchronous receiver-transmitter, and USART is a universal synchronous-asynchronous receiver-transmitter. 8051 has an inbuilt UART.

TXD(transmitted data) and RXD(received data) are TTL compatible. The 8051 requires a line driver like the MAX232 chip to make them RS232 compatible. PC supports several baud rates, but 9600 is the default. The hyper terminal supports baud rates higher than 19200 bps. The 8052 transfers and receives data serially at many baud rates by using UART. Timer 1 sets the baud rate.

Baud Rate Calculation

Take the frequency (12MHz) and divide it by 12 and then 32. You will get 1MHz and 31250Hz respectively. If the baud rate required is 9600, the 31250/9600 which is 3.25. It is not a perfect integer, so 12MHz is unsuitable for serial communication. If the frequency is 11.0592MHz, you will get 3. So, 11.0592MHz is suitable for serial communication.

Registers used for serial transfer circuits are PCON (Power Control Register), SBUF (Serial Data Buffer), and SCON (Serial Control Register). PCON forces the 8051 into power-saving mode. It consists of two power-saving mode bits and one for serial baud rate control. PCON is not bit addressable.



✓ SCON Serial Control register – {Bit Address SCON.7 to SCON.0}

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

❑ SM0 & SM1 – Mode Control bits

SM0	SM1	Serial Mode	Description	Baud Rate
0	0	Mode 0	Shift Register	Fosc/12
0	1	Mode 1	8 bit UART	Variable
1	0	Mode 2	9 bit UART	Fosc/32 or Fosc/64
1	1	Mode 3	9 bit UART	Variable

❑ Mode 0 {Shift Register sends only data}

D0	D1	D2	D3	D4	D5	D6	D7

❑ Mode 1 {8 Bit UART, 1st Start bit 0, then 8bits data and at last stop bit 1}

0	D0	D1	D2	D3	D4	D5	D6	D7	1

❑ Mode 2 & 3 {9 Bit UART, 1st Start bit 0, then 8 bits data, 1 bit parity and at last stop bit 1}

0	D0	D1	D2	D3	D4	D5	D6	D7	P	1

❑ SM2 – Enables Multiprocessor System with Mode 2 and Mode 3.

❑ REN – Receiver Enable
❑ REN = 0, receiver disabled
❑ REN = 1, receiver enabled

❑ TB8 – Transmitted bit 8 {Technically it is programmable 9th bit in mode 2 and 3}
❑ Mode 0 – not used
❑ Mode 1 – stop bit '1'
❑ Mode 2 & 3 – Parity bit, programmed by programmer.

❑ RB8 – Received bit 8 {Technically it is programmable 9th bit in mode 2 and 3}
❑ Mode 0 – not used
❑ Mode 1 – stop bit '1'
❑ Mode 2 & 3 – Parity bit, programmed by programmer.

❑ RI – Receive Interrupt
❑ It will be one after SBUF receives 8 bits data.
❑ RI will be cleared by programmer in ISR program.

❑ TI – Transmit Interrupt
❑ It will be one after SBUF transmits 8 bits data.
❑ TI will be cleared by programmer in ISR program.