

### Embedded System

specialized computer system; specific function; combining hardware and software

Washing Machine (wash cycles, water levels), Automobile Airbag System (detect collision and deploy airbag)

### Assembly language

direct control over hardware and registers

highly efficient + compact code

precise timing + real-time control

low-level device drivers; ISR routines

### RISC vs. CISC

RISC	CISC
small set of simple instructions	large set of complex instructions
fixed-length	variable-length
one instruction in one clock cycle	instructions take multiple cycles
software optimization	hardware optimization

### Harvard vs. Von Neumann

Harvard	Von Neumann
modern architecture based on Harvard Mark I relay based model.	ancient computer architecture based on stored program computer concept
instructions + data → different memory; buses	instructions + data → same physical memory; common bus
an instruction in single cycle	two clock cycles for single instruction

### parallel port programming

control multiple data lines simultaneously

send + receive data in parallel

configure + access I/O port registers (through software → flexibility)

Faster data transfer

4 ports (P0–P3), 8-bit each - port pins → direct interface with external devices alternate functions<sup>a</sup>

used to interface sensors, switches, LEDs, relays, motors, and displays

real time monitoring + control

individual bit control (Bit-addressable capability of ports; precise device control)

direct hardware manipulation; minimal instruction overhead

lower cost + complexity

applications : automation, robotics, and instrumentation

<sup>a</sup> interrupts, timers, and serial communication

### data transfer instructions

Move data between registers, memory, and I/O ports

### Microprocessor vs. Microcontroller

Microprocessor	Microcontroller
single CPU on chip	CPU, I/O, memory on chip
requires external RAM, ROM and I.O devices	all internal
general-purpose	specific embedded applications
higher processing power	moderate

### Microprocessor vs. Microcontroller (cont)

higher power consumption low

more expensive cost effective

### Instruction Set

interface between hardware and software

Controls data movement (I/O and peripheral), arithmetic, and logic operations

program flow control (branching, looping)

### Instructions supported by 8051

**Data** MOV, MOVC, and MOVX

**Transfer** (internal/external, code memory)

**Instru-  
ctions** PUSH and POP (stack)  
XCH and XCHD (between accumulator & registers /memory)

**1. Arithmetic** ADD, ADDC, SUBB, INC, and DEC; MUL AB and DIV AB (8 bit numbers)

**2. Logical** AND, OR, XOR, complement, rotate, NOT  
ANL, ORL, XRL, CLR, CPL, RL, and RRC

**3. Bit Manipulation** SETB, CLR, JB, JNB, and JBC  
Set, clear, complement, and test bits

**Control Instru-  
ctions** Branching; subroutine calls (ACALL, LCALL), returns (RET, RETI), Boolean operations, and NOP for no operation



By **racheleva**  
[cheatography.com/racheleva/](https://cheatography.com/racheleva/)

Not published yet.  
Last updated 29th April, 2026.  
Page 1 of 4.

Sponsored by **CrosswordCheats.com**  
Learn to solve cryptic crosswords!  
<http://crosswordcheats.com>

### Instruction Set (cont)

4. **SJMP, LJMP, DJNZ**  
**Branching** Jump, call, return, and loop control
5. **Boolean** single-bit variables and carry flag

Arithmetic instructions affect Carry (CY), Auxiliary Carry (AC), and Overflow (OV) flags.

### Microcontroller architecture (8051)

- CPU** (Central Processing Unit) 8-bit (arithmetic+logical) ALU; internal registers; program counter
- Program Memory (ROM / Flash)** 4KB; program storage (cannot be modified)
- Data Memory (RAM)** 128 bytes (temporary); 4 register banks (8 registers each)
- Input/Output (I/O) Ports** four 8 bit (32 pins); bidirectional
- Timers and Counters** two 16 bit (for 8051); hardware based delay

#### 1. Select Timer Mode

**2. Load Timer Register** initial value so timer overflows after desired time

**3. Start Timer** set control bits (counting)

**4. Wait for Overflow Flag** monitor overflow flag (TFx)

**5. Stop Timer & Clear Flag** Reset for next use

6. Interrupt Control

### Microcontroller architecture (8051) (cont)

**Interrupt** - temporarily halt normal execution.  
 execute a specific interrupt service routine (ISR)

immediate response to external or internal events; real-time

7. Serial Communication Interfaces

8. Clock / Oscillator

9. Power Supply and Reset Circuit

**Program Status word** 8-bit wide; 6 active flags; 2 user-defined bits

**Accumulator** Stores operands and results of arithmetic and logic operations

**Program Counter** holds address of next instruction to be executed

**Stack Pointer** point to top of the stack

#### Special Function Registers 8051

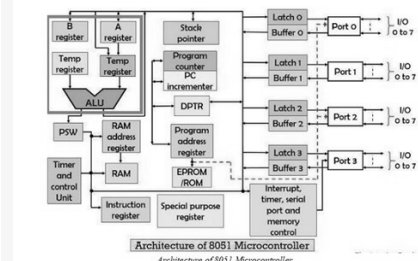
control + configure peripherals (I/O ports, timers/counters, serial communication, and interrupts)

CPU control and status information  
 direct access to hardware  
 special operations beyond general purpose

**Data Pointer** 16-bit user-accessible register; access external memory

In the RAM, only one register bank can be active at a time

fig. 1



### Addressing modes

**Immediate Addressing** transfers data (8 bit constant) directly to destination  
 MOV A, #6AH  
 instructions are 2 bytes long, execute in 1 cycle

**Register Addressing** specifies the memory address  
 MOV A, 04H

**Direct Addressing** uses register names (R0-R7)  
 MOV A, R4  
 1 byte and 1 cycle

**Indirect Addressing** address of data inside a register  
 MOV A, @R0

**Indexed Addressing** accessing data from program memory using MOVC  
 MOVC A, @A+DPTR  
 2 machine cycles



By **racheleva**  
[cheatography.com/racheleva/](http://cheatography.com/racheleva/)

Not published yet.  
 Last updated 29th April, 2026.  
 Page 2 of 4.

Sponsored by **CrosswordCheats.com**  
 Learn to solve cryptic crosswords!  
<http://crosswordcheats.com>

### Timers and counters

Timer0, 16 bit; accessed using two 8-bit registers: THx and TLx.

internal clock → derived from external crystal oscillator

timer clock frequency is 1/12th of the crystal frequency (machine cycle frequency)

### TMOD register

8-bit register; select timer mode + operations

lower 4 → Timer0; upper 4 → Timer1

GATE bit → enables timer (when external interrupt pin + run control bit active)

C/T bit → selects between timer (internal clock) and counter mode (external pulses)

### Modes

Mode 0 - 13-bit timer; 8 bits of THx and 5 bits of TLx

Mode 1 - 16-bit timer; THx and TLx form a full 16-bit counter.

Mode 2 - 8-bit auto-reload; TLx reloads automatically from THx after overflow.

Mode 3 - splits Timer0 into two 8-bit timers; Timer1 stops functioning

### TCON register

controls timer operation and contains overflow flags (TF0, TF1) and run control bits (TR0, TR1).

### Applications:

- Delay generation
- Event counting
- Baud rate generation

fig. 2

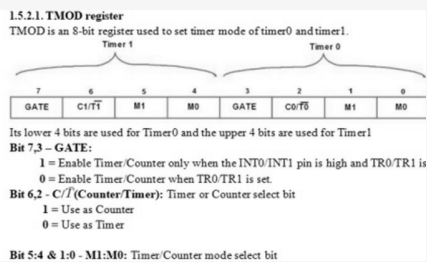
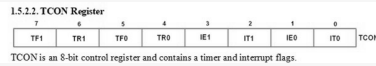


fig. 3



### Serial Communication

**Pins:** TxD, RxD (transmit/receive bit by bit)

**Register:** SBUF

**Flags:** TI → Transmission Interrupt (complete)

RI → Reception Interrupt

**TxD:** LSB sent first; load 8 bit data into register; transmitted serially; TI set; Timer1 or internal clock

**RxD:** stored in register; checked for transmission errors; no error → RI set

**SM2** bit controls error checking:  
when SM2 = 1, error checking is performed before setting RI;  
when SM2 = 0, the interrupt is generated immediately after reception.

**Modes:** Mode 1 → 8-bit UART (variable baud; asynchronous; T&R do not share common clock; RB8 error checking<sup>a</sup>; Timer1)  
Mode 2 → 9-bit UART (9th bit programmable + transmitted using TB8<sup>b</sup>)  
Mode 3 → 9-bit (fixed baud; similar to Mode 2; internal clock instead of Timer 1)

<sup>a</sup> When SM2 = 1, the receiver checks the stop bit: RB8 = 1 indicates correct data, while RB8 = 0 indicates an error, and no interrupt is generated.

<sup>b</sup> When SM2 = 1, the receiver accepts data only if the 9th bit is 1, otherwise the received data is discarded

The TI flag is not cleared automatically and must be cleared by software in the ISR.

RI is also not automatically cleared

T → Transmitter

R → Receiver

fig. 4

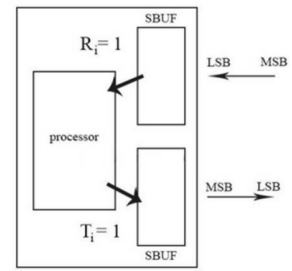


fig. 5 (Mode 1)

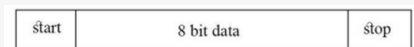
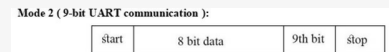


fig. 6



### Interrupt Structure

Interrupt halts normal execution → jumps to ISR

Requires GIE (and PEIE) enabled

Peripheral sets interrupt flag (xxIF) - latched high and must be cleared in ISR

Each peripheral has its own interrupt enable bit (INTCON, PIE1, PIE2, PIE3).

On interrupt - CPU saves context, clears GIE, PC is pushed to stack

Execution jumps to interrupt vector address (0004H).

ISR identifies, executes, clears flag

ends with RETFIE instruction

RETFIE - GIE re-enabled; restores context, resumes program

ISR - Interrupt Service Routine

GIE - Global Interrupt Enable

PEIE - Peripheral Interrupt Enable

xxIF - interrupt request flag



### PART C

#### 8051 Suitability for Real-Time Applications

##### 1. Architecture (8051)

Harvard architecture → faster execution

On-chip peripherals (timers, I/O, serial)

Bit-addressable memory → precise control

Deterministic operation → predictable timing

##### 2. Instruction Set

Simple, control-oriented instructions

Bit-level ops → SETB, CLR, JB, JNB

Fast execution (1–2 cycles)

Efficient for real-time control logic

##### 3. Interrupt Structure

Multiple interrupt sources (external, timer, serial)

Priority-based handling

Fast ISR execution → quick response

Improves real-time performance

---



By **racheleva**

[cheatography.com/racheleva/](https://cheatography.com/racheleva/)

---

Not published yet.

Last updated 29th April, 2026.

Page 4 of 4.

---

Sponsored by **CrosswordCheats.com**

Learn to solve cryptic crosswords!

<http://crosswordcheats.com>