Study Material

Subject : Microprocessor and Microcontroller

V Semester Electronics

By

Prof. Mrs. J. S. Gawai

Unit I: Intel 8086/8088 microprocessor & Programming:

8086/8088 microprocessor:

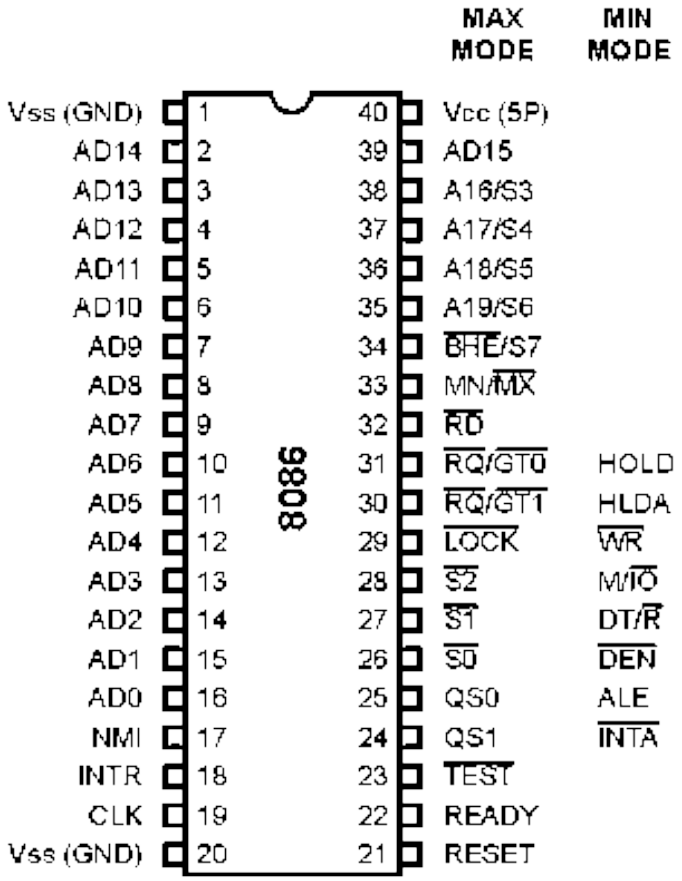
- It is a 16 bit µp.
- 8086 has a 20 bit address bus can access upto 220 memory locations (1 MB).
- It can support upto 64K I/O ports.
- It provides 14, 16-bit registers.
- It has multiplexed address and data bus AD 0- AD15 and A16 – A19.
- It requires single phase clock with 33% duty cycle to provide internal timing.
- 8086 is designed to operate in two modes, Minimum and Maximum.
- It can prefetches upto 6 instruction bytes from memory and queues them in order to speed up instruction execution.
- It requires +5V power supply.
- A 40 pin dual in line package.

Minimum and Maximum Modes:

• The minimum mode is selected by applying logic 1 to the MN / MX# input pin. This is a single microprocessor configuration.

• The maximum mode is selected by applying logic 0 to the MN / MX# input pin. This is a multi micro processors configuration.

Pin Diagram of Microprocessor-8086



A0-A7: The address/ data bus lines are the multiplexed address data bus and contain the right most eight bit of memory address or data. The address and data bits are separated by using ALE signal.

A8-A15: The address/data bus lines compose the upper multiplexed address/data bus. This lines contain address bit or data bus . The address and data bits are separated by using ALE signal.

A16/S3-A19/S6: The address/status bus bits are multiplexed to provide address signals and also status bits . The address bits are separated from the status bits using the ALE signals. The status bit is always a logic 0, bit indicates the condition of the interrupt flag bit.

BHE/S7: The bus high enable (BHE) signal is used to indicate the transfer of data over the higher order data bus. It goes low for the data transfer over and is used to derive chip select of odd address memory bank or peripherals.

Read : Whenever the read signal is at logic 0, the data bus receives the data from the memory

READY: This is the acknowledgement from the slow devices or memory that they have completed the data transfer operation. This signal is active high.

INTR: Interrupt Request: Interrupt request is used to request a hardware interrupt of INTR is held high when interrupt enable flag is set, the 8086 enters an interrupt acknowledgement cycle after the current instruction has completed its execution.

TEST : This input is tested by "WAIT" instruction. If the TEST input goes low; execution will continue. Else the processor remains in an idle state.

NMI (Non-maskable Interrupt): The non-maskable interrupt input is similar to INTR except that the NMI interrupt does not check for interrupt enable flag is at logic 1, i.e, NMI is not maskable internally by software. If NMI is activated, the interrupt input uses interrupt vector 2.

RESET: The reset input causes the microprocessor to reset itself. When 8086 reset, it restarts the execution from memory location FFFF0H. The reset signal is active high and must be active for at least four clock cycles.

CLK : Clock input: The clock input signal provides the basic timing input signal for processor and bus control operation. It is asymmetric square wave with 33% duty cycle. MN/MX: The minimum/maximum mode signal to select the mode of operation either in minimum or maximum mode configuration. Logic 1 indicates minimum mode.

Memory/IO: Signal selects either memory operation or operation. This line indicates that the microprocessor address bus contains either a memory address or an port address. Signal high at this pin indicates a memory operation. This line is logically equivalent to in maximum mode.

Interrupt acknowledge: The interrupt acknowledge signal is a response to the INTR input signal. The signal is normally used to gate the interrupt vector number onto the data bus in response to an interrupt request.

ALE- Address Latch Enable: This output signal indicates the availability of valid address on the address/data bus, and is connected to latch enable input of latches.

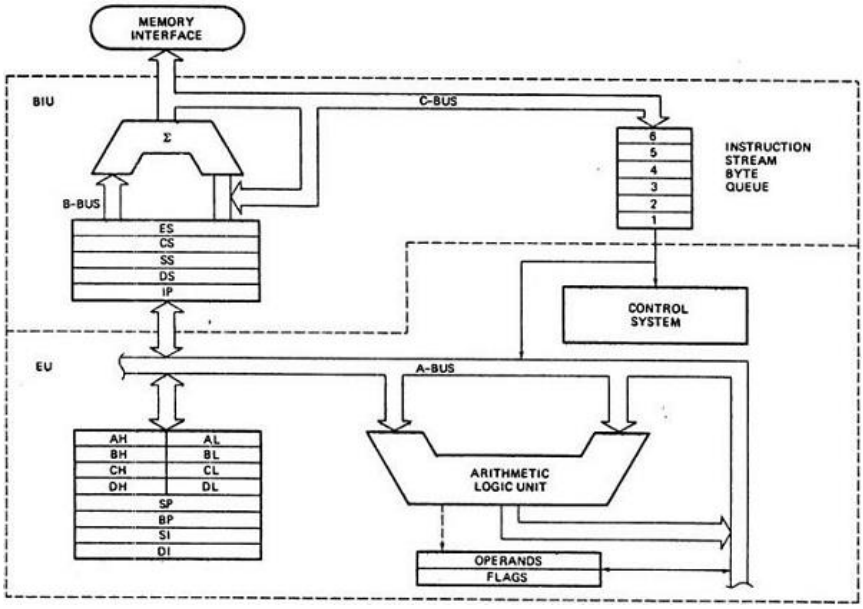
Data transmit/Receive: This output signal is used to decide the direction of date flow through the bi-directional buffer. indicates transmitting and indicates receiving the data.

Data Enable: Data bus enable signal indicates the availability of valid data over the address/data lines.

HOLD: The hold input request a direct memory access(DMA). If the hold signal is at logic 1, the micro process stops its normal execution and places its address, data and control bus at the high impedance state. HLDA: Hold acknowledgement indicates that 8086 has entered into the hold state.

QS1 and QS2- Queue status: The queue status bits shows the status of the internal instruction queue. **RQ/GT1 and RQ/GT2 - request/Grant:** The request/grant pins are used by other local bus masters to force the processor to release the local bus at the end of the processors current bus cycle. These lines are bidirectional and are used to both request and grant a DMA operation.

Architecture of 8086:



- 8086 has two blocks BIU and EU.
- The BIU performs all bus operations such as instruction fetching, reading and writing operands for memory and calculating the addresses of the memory operands.
- The instruction bytes are transferred to the instruction queue.
- EU executes instructions from the instruction system byte queue.
- Both units operate asynchronously to give the 8086 an overlapping instruction fetch and execution mechanism which is called as Pipelining. This results in efficient use of the system bus and system performance.
- BIU contains Instruction queue, Segment registers, Instruction pointer, Address adder. • EU contains Control circuitry, Instruction decoder, ALU, Pointer and Index register, Flag register.

Bus Interfacr Unit:

• It provides a full 16 bit bidirectional data bus and 20 bit address bus.

• The bus interface unit is responsible for performing all external bus operations. Specifically it has the following functions:

• Instruction fetch, Instruction queuing, Operand fetch and storage, Address relocation and Bus control.

• The BIU uses a mechanism known as an instruction stream queue to implement a pipeline architecture.

- This queue permits prefetch of up to six bytes of instruction code. When ever the queue of the BIU is not full, it has room for at least two more bytes and at the same time the EU is not requesting it to read or write operands from memory, the BIU is free to look ahead in the program by prefetching the next sequential instruction.
- These prefetching instructions are held in its FIFO queue. With its 16 bit data bus, the BIU fetches two instruction bytes in a single memory cycle. • After a byte is loaded at the input end of the queue, it automatically shifts up through the FIFO to the empty location nearest the output.
- The EU accesses the queue from the output end. It reads one instruction byte after the other from the output of the queue. If the queue is full and the EU is not requesting access to operand in memory.
- These intervals of no bus activity, which may occur between bus cycles are known as Idle state.
- If the BIU is already in the process of fetching an instruction when the EU request it to read or write operands from memory or I/O, the BIU first completes the instruction fetch bus cycle before initiating the operand read / write cycle.
- The BIU also contains a dedicated adder which is used to generate the 20 bit physical address that is output on the address bus. This address is formed by adding an appended 16 bit segment address and a 16 bit offset address.
- For example, the physical address of the next instruction to be fetched is formed by combining the current contents of the code segment CS register and the current contents of the instruction pointer IP register.
- The BIU is also responsible for generating bus control signals such as those for memory read or write and I/O read or write.

EXECUTION UNIT :

The Execution unit is responsible for decoding and executing all instructions.

• The EU extracts instructions from the top of the queue in the BIU, decodes them, generates operands if necessary, passes them to the BIU and requests it to perform the read or write bys cycles to memory or I/O and perform the operation specified by the instruction on the operands.

• During the execution of the instruction, the EU tests the status and control flags and updates them based on the results of executing the instruction.

- If the queue is empty, the EU waits for the next instruction byte to be fetched and shifted to top of the queue.
- When the EU executes a branch or jump instruction, it transfers control to a location corresponding to another set of sequential instructions.
- Whenever this happens, the BIU automatically resets the queue and then begins to fetch instructions from this new location to refill the queue.

Addressing Modes:

- Implied - the data value/data address is implicitly associated with the instruction.
- Register - references the data in a register or in a register pair.
- Immediate - the data is provided in the instruction.
- Direct - the instruction operand specifies the memory address where data is located.
- Register indirect - instruction specifies a register containing an address, where data is located. This addressing mode works with SI, DI, BX and BP registers.
- Based :- 8-bit or 16-bit instruction operand is added to the contents of a base register (BX or BP), the resulting value is a pointer to location where data resides. Indexed :- 8-bit or 16-bit instruction operand is added to the contents of an index register (SI or DI), the resulting value is a pointer to location where data resides.
- Based Indexed :- the contents of a base register (BX or BP) is added to the contents of an index register (SI or DI), the resulting value is a pointer to location where data resides.
- Based Indexed with displacement :- 8-bit or 16-bit instruction operand is added to the contents of a base register (BX or BP) and index register (SI or DI), the resulting value is a pointer to location where data resides.

Unit II: 8086 & Peripheral Interfacing I:

Assembly language programming of 8086:

Machine Language:

• Set of fundamental instructions the machine can execute.

• Expressed as a pattern of 1's and 0's „ Assembly Language:

• Alphanumeric equivalent of machine language

• Mnemonics more human-oriented than 1's and 0's „ Assembler:

• Computer program that transliterates (one-to-one mapping) assembly to machine language

• Computer's native language is machine/assembly language

„ Faster and shorter programs.

• Compilers do not always generate optimum code. „

Instruction set knowledge is important for machine designers. „ Compiler writers must be familiar with details of machine language. „ Small controllers embedded in many products

• Have specialized functions,

• Rely so heavily on input/output functionality,

• HLLs inappropriate for product development.

Instruction Set Architecture:„

Instruction set:

collection of all machine operations. „ Programmer sees set of instructions, and machine resources manipulated by them. ISA includes

• Instruction set,

• Memory, and

• Programmer-accessible registers. „

Temporary or scratch-pad memory used to implement some functions is not part of ISA

• Not programmer accessible.

CPU Registers: „

Fourteen 16-bit registers „ Data Registers

• AX (Accumulator Register): AH and AL
• BX (Base Register): BH and BL
• CX (Count Register): CH and CL
• DX (Data Register): DH and DL „ Pointer and Index Registers
• SI (Source Index)
• DI (Destination Index)
• SP (Stack Pointer)
• BP (Base Pointer)
• IP (Instruction Pointer)

Segment Registers

• CS (Code Segment)
• DS (Data Segment)
• SS (Stack Segment)
• ES (Extra Segment) „

FLAGS Register:

• Zero flag • Sign flag • Parity flag • Carry flag • Overflow flag

Fetch-Execute Process „ Program Counter (PC) or Instruction Pointer (IP)

• Holds address of next instruction to fetch „ Instruction Register (IR)
• Stores the instruction fetched from memory. „

Fetch-Execute process

• Read an instruction from memory addressed by PC
• Increment program counter
• Execute fetched instruction in IR
• Repeat process

Assembly Language Syntax „ Program consists of statement per line. „ Each statement is an instruction or assembler directive „ Statement syntax

• Name operation operand(s) comment „ Name field
• Used for instruction labels, procedure names, and variable names.
• Assembler translates names into memory addresses
• Names are 1-31 characters including letters, numbers and special characters ? . @ _ $ %
• Names may not begin with a digit.
    - If a period is used, it must be first character
• Case insensitive

Examples of legal names

• COUNTER1
• @character
• SUM_OF_DIGITS

• $1000
• Done?
• TEST „
Examples of illegal names
• TWO WORDS
• 2abc
• A45.28
• You&Me
Operation field
• instruction
    • Symbolic operation code (opcode)
    • Symbolic opcodes translated into machine language opcode
    • Describes operation's function; e.g. MOV, ADD, SUB, INC.
• Assembler directive
    • Contains pseudo-operation code (pseudo-op)
    • Not translated into machine code
    • Tell the assembler to do something. „
Operand field
• Specifies data to be acted on
• Zero, one, or two operands
    • NOP
    • INC AX
    • ADD AX, 2
Comment field
    • A semicolon marks the beginning of a comment
    • A semicolon in beginning of a line makes it all a comment line
    • Good programming practice dictates comment on every line „
    Examples • MOV CX, 0 ; move 0 to CX
        • Do not say something obvious
    • MOV CX, 0 ; CX counts terms, initially 0
    • Put instruction in context of program
    • ; initialize registers

Variable Declaration „

Each variable has a type and assigned a memory address. „
Data-defining pseudo-ops
• DB define byte
• DW define word
• DD define double word (two consecutive words)
• DQ define quad word (four consecutive words)
• DT define ten bytes (five consecutive words) „
Each pseudo-op can be used to define one or more data items of given type.

Byte Variables „

Assembler directive format defining a byte variable

• name DB initial value

• a question mark (“?”) place in initial value leaves variable uninitialized „

I DB 4 define variable I with initial value 4 „

J DB ? Define variable J with uninitialized value „

Name DB “Course” allocate 6 bytes for Name „

K DB 5, 3, -1 allocates 3 bytes

Instruction Types „

Data transfer instructions

• Transfer information between registers and memory locations or I/O ports.

• MOV, XCHG, LEA, PUSH, POP, PUSHF, POPF, IN, OUT. „

Arithmetic instructions

• Perform arithmetic operations on binary or binary-coded-decimal (BCD) numbers.

• ADD, SUB, INC, DEC, ADC, SBB, NEG, CMP, MUL, IMUL, DIV, IDIV, CBW, CWD. „

Bit manipulation instructions

• Perform shift, rotate, and logical operations on memory locations and registers.

• SHL, SHR, SAR, ROL, ROR, RCL, RCR, NOT, AND, OR, XOR, TEST.

Control transfer instructions

• Control sequence of program execution; include jumps and procedure transfers.

• JMP, JG, JL, JE, JNE, JGE, JLE, JNG, JNL, JC, JS, JA, JB, JAE, JBE, JNB, JNA, JO, JZ, JNZ, JP, JCXZ, LOOP, LOOPE, LOOPZ, LOOPNE, LOOPNZ, CALL, RET. „

String instructions

• Move, compare, and scan strings of information.

• MOVS, MOVSB, MOVSW, CMPS, CMPSB, CMPSW. SCAS, SCASB, SCASW, LODS, LODSB, LODSW, STOS, STOSB, STOSW. „

Interrupt instructions

• Interrupt processor to service specific condition.

• INT, INTO, IRET. „ Processor control instructions

• Set and clear status flags, and change the processor execution state.

• STC, STD, STI. „ Miscellaneous instructions

• NOP, WAIT.

1. **Program Structure: An Example**

```
TITLE PRGM1
.MODEL SMALL
.STACK 100H
DATA
        A DW 2
        B DW 5
        SUM DW ?
.CODE
```

```
MAIN PROC
; initialize DS
        MOV AX, @DATA
        MOV DS, AX
```

2. Program Structure: An Example

```
; add the numbers
        MOV AX, A
        ADD AX, B
        MOV SUM, AX
        ; exit to DOS
        MOV AX, 4C00H
        INT 21H
        MAIN ENDP
        END MAIN
```

I/O Interface

- We have four common types of memory:
- Read only memory (ROM)
- Flash memory (EEPROM)
- Static Random access memory (SARAM)
- Dynamic Random access memory (DRAM).
- Pin connections common to all memory devices are:

The address input, data output or input/outputs, selection input and control input used to select a read or write operation.

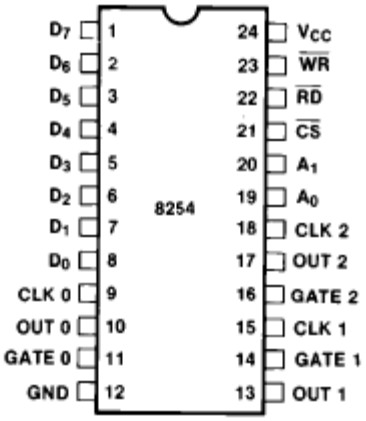
- Address connections: All memory devices have address inputs that select a memory location within the memory device. Address inputs are labeled from A0 to An.
- Data connections: All memory devices have a set of data outputs or input/outputs. Today many of them have bi-directional common I/O pins.
- Selection connections: Each memory device has an input, that selects or enables the memory device. This kind of input is most often called a chip select (CS bar), chip enable (CE bar) or simply select (S bar) input. RAM memory generally has at least one CS bar or S bar input and ROM at least one CE bar .
- If the CE bar, CS bar , S bar input is active the memory device perform the read or write.
- If it is inactive the memory device cannot perform read or write operation.
- If more than one CS bar connection is present, all most be active to perform read or write data.
- Control connections: A ROM usually has only one control input, while a RAM often has one or two control inputs.
- The control input most often found on the ROM is the output enable (OE bar) or gate (G bar), this allows data to flow out of the output data pins of the ROM.
- If OE bar and the selected input are both active, then the output is enable, if OE bar is inactive, the output is disabled at its high-impedance state.

- The OE bar connection enables and disables a set of three-state buffer located within the memory device and must be active to read data.
- A RAM memory device has either one or two control inputs. If there is one control input it is often called R/ W .
- This pin selects a read operation or a write operation only if the device is selected by the selection input (CS bar).
- If the RAM has two control inputs, they are usually labeled WE bar or W bar and OE bar or G bar .
- (WE bar) write enable must be active to perform a memory write operation and OE bar must be active to perform a memory read operation.
- When these two controls WE and OE bar are present, they must never be active at the same time.
- The ROM read only memory permanently stores programs and data and data was always present, even when power is disconnected.
- It is also called as nonvolatile memory.
- EPROM (erasable programmable read only memory) is also erasable if exposed to high intensity ultraviolet light for about 20 minutes or less, depending upon the type of EPROM.
- We have PROM (programmable read only memory)
- RMM (read mostly memory) is also called the flash memory.
- The flash memory is also called as an EEPROM (electrically erasable programmable ROM), EAROM (electrically alterable ROM), or a NOVROM (nonvolatile ROM).
- These memory devices are electrically erasable in the system, but require more time to erase than a normal RAM.
- EPROM contains the series of 27XXX contains the following part numbers : 2704(512 * 8), 2708(1K * 8), 2716(2K * 8), 2732(4K * 8), 2764(8K * 8), 27128(16K * 8) etc.
- Each of these parts contains address pins, eight data connections, one or more chip selection inputs (CE) and an output enable pin (OE).
- This device contains 11 address inputs and 8 data outputs.
- If both the pin connection CE and OE are at logic 0, data will appear on the output connection . If both the pins are not at logic 0, the data output connections remains at their high impedance or off state.
- To read data from the EPROM Vpp pin must be placed at a logic 1.

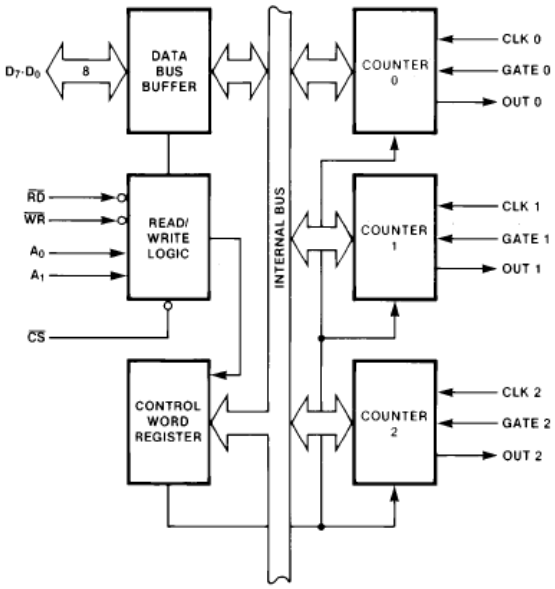
Unit III: 8086 & Peripheral Interfacing II:

Programmable interval timer/counter 8254; Architecture, working modes

- Compatible with All Intel and Most other Microprocessors
- Handles Inputs from DC to 10 MHz 8 MHz 8254 10 MHz 8254-2
- Status Read-Back Command
- Six Programmable Counter Modes
- Three Independent 16-Bit Counters
- Binary or BCD Counting
- Single a 5V Supply



Pin Configuration



8254 Block Diagram

- Standard Temperature Range.
- The Intel 8254 is a counter/timer device designed to solve the common timing control problems in microcomputer system design.
- It provides three independent 16-bit counters, each capable of handling clock inputs up to 10 MHz.
- All modes are software programmable. The 8254 is a superset of the 8253.
- The 8254 uses HMOS technology and comes in a 24-pin plastic or CERDIP package.

- READ/WRITE LOGIC

  The Read/Write Logic accepts inputs from the system bus and generates control signals for the other functional blocks of the 8254. A1 and A0 select one of the three counters or the Control Word Register to be read from/written into. A ``low" on the RD input tells the 8254 that the CPU is reading one of the counters. A ``low" on the WR input tells the 8254 that the CPU is writing either a Control Word or an initial count. Both RD and WR are qualified by CS; RD and WR are ignored unless the 8254 has been selected by holding CS low.
- CONTROL WORD REGISTER

  The Control Word Register (see Figure 4) is selected by the Read/Write Logic when A1,A0 e 11. If the CPU then does a write operation to the 8254, the data is stored in the Control Word Register and is interpreted as a Control Word used to define the operation

of the Counters. The Control Word Register can only be written to; status information is available with the Read-Back Command.

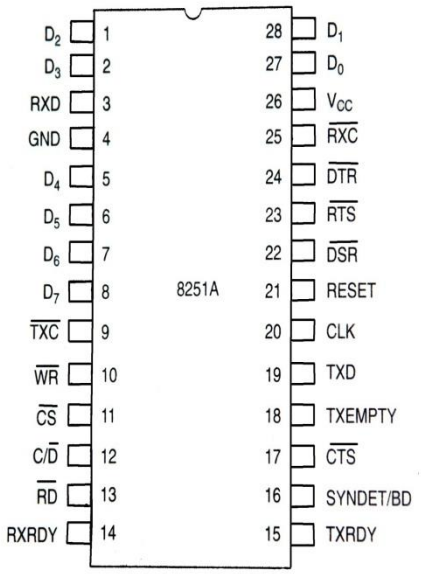
- COUNTER 0, COUNTER 1, COUNTER 2
  These three functional blocks are identical in operation, so only a single Counter will be described. The internal block diagram of a single counter is shown in Figure. The Counters are fully independent. Each Counter may operate in a different Mode. The Control Word Register is shown in the figure; it is not part of the Counter itself, but its contents determine how the Counter operates. The status register, shown in Figure 5, when latched, contains the current contents of the Control Word Register and status of the output and null count flag. (See detailed explanation of the Read- Back command.) The actual counter is labelled CE (for ``Counting Element"). It is a 16-bit presettable synchronous down counter. OLM and OLL are two 8-bit latches. OL stands for ``Output Latch"; the subscripts M and L stand for ``Most significant byte" and ``Least significant byte".

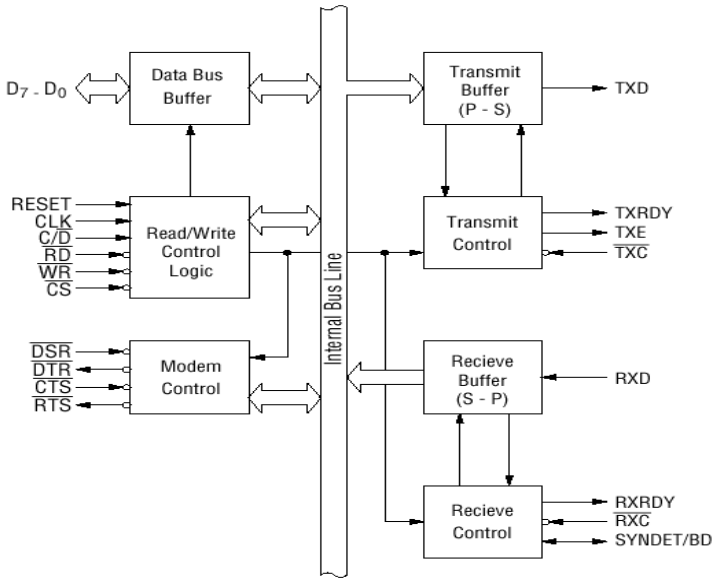
8251A programmable Communication Interface:

Introduction:

- 8251A is a USART (Universal Synchronous Asynchronous Receiver Transmitter) for serial data communication.
- Programmable peripheral designed for synchronous /asynchronous serial data communication, packaged in a 28-pin DIP.
- Receives parallel data from the CPU & transmits serial data after conversion.
- Also receives serial data from the outside & transmits parallel data to the CPU after conversion.

Pin diagram and Block diagram of the 8251 USART:





Sections of 8251A

- Data Bus buffer
- Read/Write Control Logic
- Modem Control
- Transmitter
- Receiver

1. Data Bus Buffer
    - D0-D7 : 8-bit data bus used to read or write status, command word or data from or to the 8251A
2. Read/Write Control logic
    - Includes a control logic, six input signals & three buffer registers: Data register, control register & status register.
    - Control logic : Interfaces the chip with MPU, determines the functions of the chip according to the control word in the control register & monitors the data flow.
3. Input signals
    - CS – Chip Select : When signal goes low, the 8251A is selected by the MPU for communication.
    - C/D – Control/Data : When signal is high, the control or status register is addressed; when it is low, data buffer is addressed. (Control register & status register are differentiated by WR and RD signals)
    - WR : When signal is low, the MPU either writes in the control register or sends output to the data buffer.
    - RD : When signal goes low, the MPU either reads a status from the status register or accepts data from data buffer.
    - RESET : A high on this signal reset 8252A & forces it into the idle mode.
    - CLK : Clock input, usually connected to the system clock for communication with the microprocessor.
4. Control Register
    - 16-bit register for a control word consist of two independent bytes namely mode word & command word.
    - Mode word : Specifies the general characteristics of operation such as baud, parity, number of bits etc.
    - Command word : Enables the data transmission and reception.
    - Register can be accessed as an output port when the Control/Data pin is high.
5. Status register
    - Checks the ready status of the peripheral.
    - Status word in the status register provides the information concerning register status and transmission errors.
6. Data register
    - Used as an input and output port when the C/D is low

CS	C/D	WR	RD	Operation
0	0	1	0	MPU reads data from data buffer
0	0	0	1	MPU writes data from data buffer
0	1	0	1	MPU writes a word to control register
0	1	1	0	MPU reads a word from status register
1	×	×	×	Chip is not selected for any operation

7. Modem Control
    - DSR - Data Set Ready : Checks if the Data Set is ready when communicating with a modem.
    - DTR - Data Terminal Ready : Indicates that the device is ready to accept data when the 8251 is communicating with a modem.
    - CTS - Clear to Send : If its low, the 8251A is enabled to transmit the serial data provided the enable bit in the command byte is set to '1'.
    - RTS - Request to Send Data : Low signal indicates the modem that the receiver is ready to receive a data byte from the modem.
8. **Transmitter section**
    - Accepts parallel data from MPU & converts them into serial data.
    - Has two registers:
        i. Buffer register : To hold eight bits
        ii. Output register : To convert eight bits into a stream of serial bits.
    - The MPU writes a byte in the buffer register.
    - Whenever the output register is empty; the contents of buffer register are transferred to output register.
    - Transmitter section consists of three output & one input signals
        i. TxD - Transmitted Data Output : Output signal to transmit the data to peripherals
        ii. TxC - Transmitter Clock Input : Input signal, controls the rate of transmission.
        iii. TxRDY - Transmitter Ready : Output signal, indicates the buffer register is empty and the USART is ready to accept the next data byte.
        iv. TxE - Transmitter Empty : Output signal to indicate the output register is empty and the USART is ready to accept the next data byte.
9. **Receiver Section**
    - Accepts serial data on the RxD pin and converts them to parallel data.
    - Has two registers :

i. Receiver input register
ii. Buffer register

- When RxD goes low, the control logic assumes it is a start bit, waits for half bit time, and samples the line again. If the line is still low, the input register accepts the following data, and loads it into buffer register at the rate determined by the receiver clock.
- RxRDY - Receiver Ready Output: Output signal, goes high when the USART has a character in the buffer register & is ready to transfer it to the MPU.
- RxD - Receive Data Input : Bits are received serially on this line & converted into a parallel byte in the receiver input register.
- RxC - Receiver Clock Input : Clock signal that controls the rate at which bits are received by the USART.

Unit – V: 8051 microcontroller & programming:

Introduction to 8051 microcontroller; Pin diagram, architecture:

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

Difference between Microprocessor and Microcontroller

The following table highlights the differences between a microprocessor and a microcontroller –

Microcontroller	Microprocessor
Microcontrollers are used to execute a single task within an application.	Microprocessors are used for big applications.
Its designing and hardware cost is low.	Its designing and hardware cost is high.
Easy to replace.	Not so easy to replace.
It is built with CMOS technology, which requires less power to operate.	Its power consumption is high because it has to control the entire system.
It consists of CPU, RAM, ROM, I/O ports.	It doesn't consist of RAM, ROM, I/O ports. It uses its pins to interface to peripheral devices.

Types of Microcontrollers

Microcontrollers are divided into various categories based on memory, architecture, bits and instruction sets. Following is the list of their types −

Bit

Based on bit configuration, the microcontroller is further divided into three categories.

- 8-bit microcontroller − This type of microcontroller is used to execute arithmetic and logical operations like addition, subtraction, multiplication division, etc. For example, Intel 8031 and 8051 are 8 bits microcontroller.
- 16-bit microcontroller − This type of microcontroller is used to perform arithmetic and logical operations where higher accuracy and performance is required. For example, Intel 8096 is a 16-bit microcontroller.
- 32-bit microcontroller − This type of microcontroller is generally used in automatically controlled appliances like automatic operational machines, medical appliances, etc.

Memory

Based on the memory configuration, the microcontroller is further divided into two categories.

- External memory microcontroller − This type of microcontroller is designed in such a way that they do not have a program memory on the chip. Hence, it is named as external memory microcontroller. For example: Intel 8031 microcontroller.
- Embedded memory microcontroller − This type of microcontroller is designed in such a way that the microcontroller has all programs and data memory, counters and timers, interrupts, I/O ports are embedded on the chip. For example: Intel 8051 microcontroller.

Instruction Set

Based on the instruction set configuration, the microcontroller is further divided into two categories.

- CISC − CISC stands for complex instruction set computer. It allows the user to insert a single instruction as an alternative to many simple instructions.
- RISC − RISC stands for Reduced Instruction Set Computers. It reduces the operational time by shortening the clock cycle per instruction.

Applications of Microcontrollers

Microcontrollers are widely used in various different devices such as −

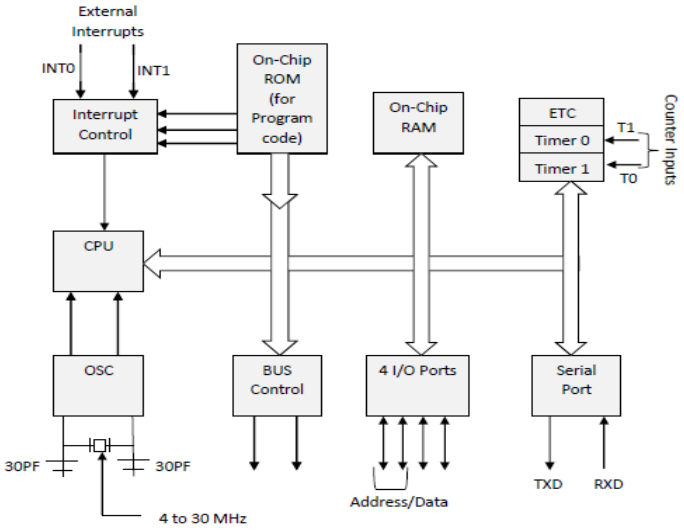
- Light sensing and controlling devices like LED.
- Temperature sensing and controlling devices like microwave oven, chimneys.
- Fire detection and safety devices like Fire alarm.

- Measuring devices like Volt Meter.

8051 microcontroller is designed by Intel in 1981. It is an 8-bit microcontroller. It is built with 40 pins DIP (dual inline package), 4kb of ROM storage and 128 bytes of RAM storage, 2 16-bit timers. It consists of are four parallel 8-bit ports, which are programmable as well as addressable as per the requirement. An on-chip crystal oscillator is integrated in the microcontroller having crystal frequency of 12 MHz.

Let us now discuss the architecture of 8051 Microcontroller.

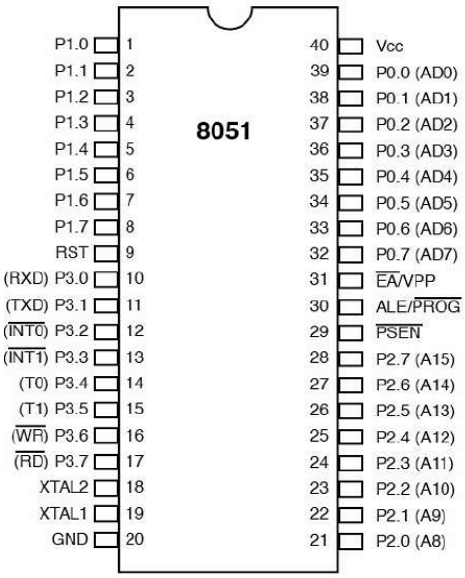
In the following diagram, the system bus connects all the support devices to the CPU. The system bus consists of an 8-bit data bus, a 16-bit address bus and bus control signals. All other devices like program memory, ports, data memory, serial interface, interrupt control, timers, and the CPU are all interfaced together through the system bus.



The pin diagram of 8051 microcontroller looks as follows –

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

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



8051 microcontrollers have 4 I/O ports each of 8-bit, which can be configured as input or output. Hence, total 32 input/output pins allow the microcontroller to be connected with the peripheral devices.

- Pin configuration, i.e. the pin can be configured as 1 for input and 0 for output as per the logic state.
    - Input/Output (I/O) pin − All the circuits within the microcontroller must be connected to one of its pins except P0 port because it does not have pull-up resistors built-in.
    - Input pin − Logic 1 is applied to a bit of the P register. The output FE transistor is turned off and the other pin remains connected to the power supply voltage over a pull-up resistor of high resistance.
- Port 0 − The P0 (zero) port is characterized by two functions −
    - When the external memory is used then the lower address byte (addresses A0A7) is applied on it, else all bits of this port are configured as input/output.

- When P0 port is configured as an output then other ports consisting of pins with built-in pull-up resistor connected by its end to 5V power supply, the pins of this port have this resistor left out.

Input Configuration

If any pin of this port is configured as an input, then it acts as if it "floats", i.e. the input has unlimited input resistance and in-determined potential.

Output Configuration

When the pin is configured as an output, then it acts as an "open drain". By applying logic 0 to a port bit, the appropriate pin will be connected to ground (0V), and applying logic 1, the external output will keep on "floating".

In order to apply logic 1 (5V) on this output pin, it is necessary to build an external pullup resistor.

Port 1

P1 is a true I/O port as it doesn't have any alternative functions as in P0, but this port can be configured as general I/O only. It has a built-in pull-up resistor and is completely compatible with TTL circuits.

Port 2

P2 is similar to P0 when the external memory is used. Pins of this port occupy addresses intended for the external memory chip. This port can be used for higher address byte with addresses A8-A15. When no memory is added then this port can be used as a general input/output port similar to Port 1.

Port 3

In this port, functions are similar to other ports except that the logic 1 must be applied to appropriate bit of the P3 register.

Pins Current Limitations

- When pins are configured as an output (i.e. logic 0), then the single port pins can receive a current of 10mA.
- When these pins are configured as inputs (i.e. logic 1), then built-in pull-up resistors provide very weak current, but can activate up to 4 TTL inputs of LS series.
- If all 8 bits of a port are active, then the total current must be limited to 15mA (port P0: 26mA).
- If all ports (32 bits) are active, then the total maximum current must be limited to 71mA.

Interrupts are the events that temporarily suspend the main program, pass the control to the external sources and execute their task. It then passes the control to the main program where it had left off.

8051 has 5 interrupt signals, i.e. INT0, TFO, INT1, TF1, RI/TI. Each interrupt can be enabled or disabled by setting bits of the IE register and the whole interrupt system can be disabled by clearing the EA bit of the same register.

IE (Interrupt Enable) Register

This register is responsible for enabling and disabling the interrupt. EA register is set to one for enabling interrupts and set to 0 for disabling the interrupts. Its bit sequence and their meanings are shown in the following figure.

EA	-	-	ES	ET1	EX1	ET0	EX0

EA	IE.7	It disables all interrupts. When EA = 0 no interrupt will be acknowledged and EA = 1 enables the interrupt individually.
-	IE.6	Reserved for future use.
-	IE.5	Reserved for future use.
ES	IE.4	Enables/disables serial port interrupt.
ET1	IE.3	Enables/disables timer1 overflow interrupt.
EX1	IE.2	Enables/disables external interrupt1.
ET0	IE.1	Enables/disables timer0 overflow interrupt.
EX0	IE.0	Enables/disables external interrupt0.

IP (Interrupt Priority) Register

We can change the priority levels of the interrupts by changing the corresponding bit in the Interrupt Priority (IP) register as shown in the following figure.

- A low priority interrupt can only be interrupted by the high priority interrupt, but not interrupted by another low priority interrupt.

- If two interrupts of different priority levels are received simultaneously, the request of higher priority level is served.
- If the requests of the same priority levels are received simultaneously, then the internal polling sequence determines which request is to be serviced.

-	-	PT2	PS	PT1	PX1	PT0	PX0
bit7	bit6	bit5	bit4	bit3	bit2	bit1	

-	IP.6	Reserved for future use.
-	IP.5	Reserved for future use.
PS	IP.4	It defines the serial port interrupt priority level.
PT1	IP.3	It defines the timer interrupt of 1 priority.
PX1	IP.2	It defines the external interrupt priority level.
PT0	IP.1	It defines the timer0 interrupt priority level.
PX0	IP.0	It defines the external interrupt of 0 priority level.

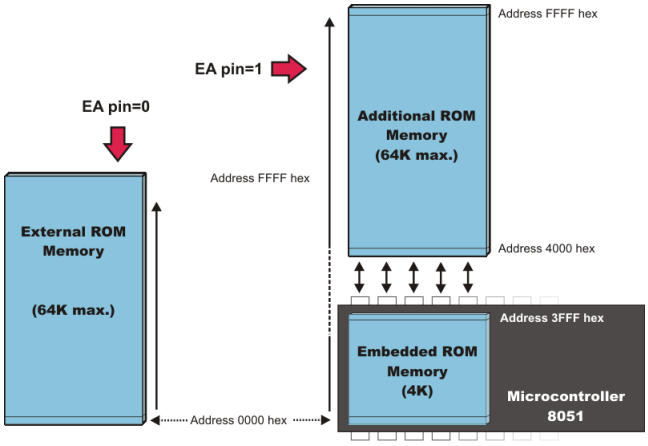
TCON register specifies the type of external interrupt to the microcontroller.

8051 Memory Organization

The 8051 microcontroller's memory is divided into Program Memory and Data Memory. Program Memory (ROM) is used for permanent saving program being executed, while Data Memory (RAM) is used for temporarily storing and keeping intermediate results and variables.

Program Memory (ROM)

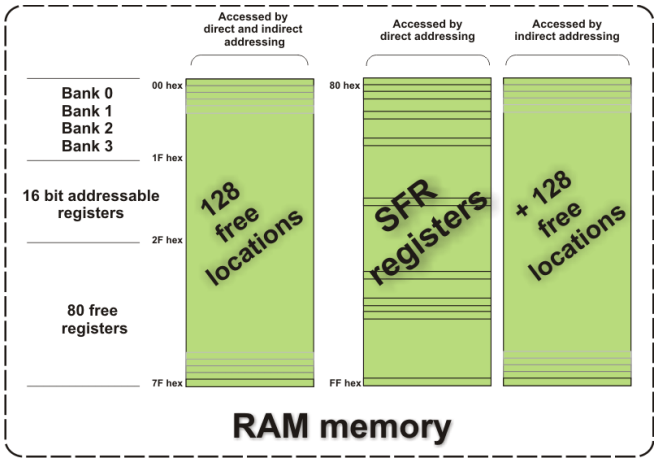
Program Memory (ROM) is used for permanent saving program (CODE) being executed. The memory is read only. Depending on the settings made in compiler, program memory may also used to store a constant variables. The 8051 executes programs stored in program memory only. code memory type specifier is used to refer to program memory.
8051 memory organization alows external program memory to be added. How does the microcontroller handle external memory depends on the pin EA logical state.



Internal Data Memory

Up to 256 bytes of internal data memory are available depending on the 8051 derivative. Locations available to the user occupy addressing space from 0 to 7Fh, i.e. first 128 registers and this part of RAM is divided in several blocks. The first 128 bytes of internal data memory are both directly and indirectly addressable. The upper 128 bytes of data memory (from 0x80 to 0xFF) can be addressed only indirectly.

Since internal data memory is used for CALL stack also and there is only 256 bytes splited over few different memory areas fine utilizing of this memory is crucial for fast and compact code. See types efficiency also.
Memory block in the range of 20h to 2Fh is bit-addressable, which means that each bit being there has its own address from 0 to 7Fh. Since there are 16 such registers, this block contains in total of 128 bits with separate addresses (Bit 0 of byte 20h has the bit address 0, and bit 7 of byte 2Fh has the bit address 7Fh).
Three memory type specifiers can be used to refer to the internal data memory: **data**, **idata**, and **bdata**.



External Data Memory

Access to external memory is slower than access to internal data memory. There may be up to 64K Bytes of external data memory. Several 8051 devices provide on-chip XRAM space that is accessed with the same instructions as the traditional external data space. This XRAM space is typically enabled via proper setting of SFR register and overlaps the external memory space. Setting of that register must be manually done in code, before any access to external memory or

XRAM space is made.
The mikroC PRO for 8051 has two memory type specifiers that refers to external memory space: **xdata** and **pdata**.

SFR Memory

The 8051 provides 128 bytes of memory for Special Function Registers (SFRs). SFRs are bit, byte, or word-sized registers that are used to control timers, counters, serial I/O, port I/O, and peripherals.

8051 Serial Port and Timer/Counter:
Serial Port
Timer Counter

8051 Timer/Counter Two internal Timers/Counters 16-bit timer/counter Timer uses system clock as source of input pulses Counter uses external input pulses from port 3 (T0,T1) If associated interrupt is enabled, when count overflow an interrupt is generated Registers TH0, TL0 : timer/counter register of timer 0 TH1, TL1 : timer/counter register of timer 1 TMOD : Mode Select register TCON : Control Register

8051 Timer/Counter
Timer/Counter Operations

- Input Source
- Operation Control
- Update Mode

8051 Timer/Counter
Operation Modes
Mode 0

- 13-bit counter, an interrupt is generated when counter overflows.
- It takes 8192 input pulses to generate the next interrupt.

Mode 1

- 16-bit counter, similar to mode 0, but take 65536 input pulses

Mode 2

- 8-bit reload
- TLi operates as timer/counter
- THi store a number and reload to TLi when overflows

Mode 3

- Timer 1 is inactive, hold count value.
- TL0 and TH0 operate as two separate 8-bit timer/counter
- TL0 control by timer 0 control bits
- TH0 operate as timer driven by system clock, prescaled by 12 and cause timer 1 interrupt overflows