## Unit V

## 8085 Microprocessor

## Introduction to Microprocessor 8085

Microprocessor is a electronic chip, that functions as the central processing unit of a  computer.

For example: Washing machines, microwave ovens, mobile phones etc.

Its advance applications are Radar, Satellites, flights.

All processors are use the basic concept of stored program execution. program or instructions are stored sequentially in the memory. Every microprocessor has its own associated set of instructions. [Instruction set](http://mvn.edu.in/mvnlms/mod/assign/view.php?id=904) for microprocessor is in two forms one in mnemonic, which is comparatively easy to understand and the other is binary machine code.

[**8085 microprocessor**](http://mvn.edu.in/mvnlms/mod/quiz/view.php?id=909)**:**

The **Intel 8085** ("*eighty-eighty-five*") is an [8-bit](http://en.wikipedia.org/wiki/8-bit) [microprocessor](http://en.wikipedia.org/wiki/Microprocessor) introduced by [Intel](http://en.wikipedia.org/wiki/Intel) in 1977. The 8085 is a conventional [von Neumann](http://en.wikipedia.org/wiki/Von_Neumann_architecture)design based on the Intel 8080. It is designed by using nmos technology. The "5" in the model number came from the fact that the 8085 requires only a +5-[Volt](http://en.wikipedia.org/wiki/Volt) (V) power supply. rather than requiring the +5 V, −5 V and +12 V supplies the 8080 needed. It has 8 bit data bus and 16 bit address bus. it can work upto 5 MHz frequency. It has 40 pins in its chip. Lower order address bus is multiplexed with data bus to minimize the chip size.

The 8085 has extensions to support new interrupts, with three maskable interrupts (RST 7.5, RST 6.5 and RST 5.5), one [non-maskable interrupt](http://en.wikipedia.org/wiki/Non-maskable_interrupt) (TRAP), and one externally serviced interrupt (INTR). The RST n.5 interrupts refer to actual pins on the processor, a feature which permitted simple systems to avoid the cost of a separate interrupt controller.

### 2. Architecture of 8085

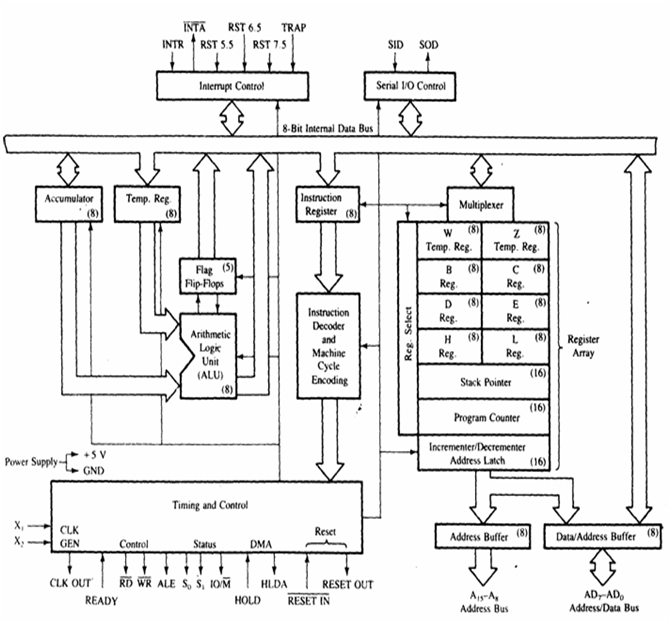


Fig : block diagram of 8085

8085 architecture consists of five functional units:

1. **arithmatic and logic unit** :

The ALU performs the actual numerical and logic operation such as add’, ‘subtract’, ‘AND’, ‘OR’, etc. Uses data from memory and from Accumulator to perform arithmetic. Always stores result of operation in Accumulator.

**2. genaral purpose registers**:

8-bit B and 8-bit C registers can be used as one 16-bit BC register pair. When used as a pair the C register contains low-order byte. Some instructions may use BC register as a data pointer.

8-bit D and 8-bit E registers can be used as one 16-bit DE register pair. When used as a pair the E register contains low-order byte. Some instructions may use DE register as a data pointer.

8-bit H and 8-bit L registers can be used as one 16-bit HL register pair. When used as a pair the L register contains low-order byte. HL register usually contains a data pointer used to reference memory addresses.

3.   **special purpose registers:**

**a) Accumulator** or A register is an 8-bit register used for arithmetic, logic, I/O and load/store operations.

**b) Flag** is an 8-bit register containing 5 1-bit flags:

* Sign - set if the most significant bit of the result is set.
* Zero - set if the result is zero.
* Auxiliary carry - set if there was a carry out from bit 3 to bit 4 of the result.
* Parity - set if the parity (the number of set bits in the result) is even.
* Carry - set if there was a carry during addition, or borrow during subtraction/comparison.

**c)Stack pointer** is a 16 bit register. This register is always incremented/decremented by 2

**d)Program counter** is a 16-bit register.

**4. instruction register and decoder**:

Temporary store for the current instruction ofa program. Latest instruction sent here from memory prior to execution. Decoder then takes instruction and ‘decodes’ or interprets the instruction. Decoded instruction then passed to next stage.

**5. Timing and control unit**: Generates signals within uP to carry out the instruction, which has been decoded. In reality causes certain connections between blocks of the uP to be opened or closed, so that data goes where it is required, and so that ALU operations occur.

### 3. Pin Diagram of 8085

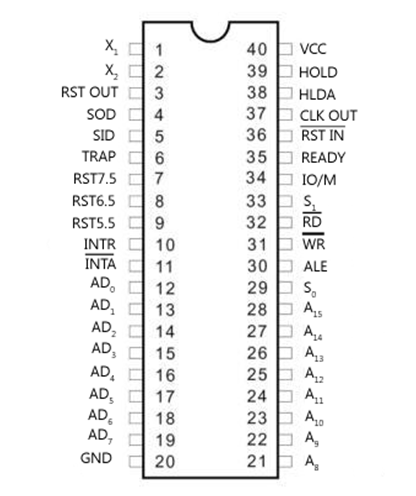


                                     Fig: [pin diagram of 8085](http://mvn.edu.in/mvnlms/mod/assign/view.php?id=902)

8085 up is an 8-bit general purpose microprocessor capable of addressing 64Kb of memory.

The device has 40 pins,+5 V power supply, operate on 3 to 5 MHZ frequency single phase clock.

All the signals can be classified in 8085 up pin diagram into six groups –

1)  Address Bus:  in this 16 signals lines. These lines are splits into two segments -

a) A15-A8– unidirectional and used for the higher order address (MSB).

b)AD7-AD0 – Dual purpose such as data bus as well as lower address data bus(LSB).

**2)**Control and status signals: these signals are used to identify the nature of operation.

**Three control signals that are-**

**RD** – it is a active low signal. Which indicate that the selected IO or Memory device is to be read and data is available on the data bus.

**WR**-it is a active low signal which indicate that the data on the data bus are to be written into a selected memory or IO location.

**ALE**- it is a +ve going pulse generated everytime the 8085 begins an operation (machine cycle): which indicate that the bits on AD7-AD0are address bits.

**Three status signals that are**–

**IO/M-**this is a status signal used to differentiate between IO and Memory operations.when it is hign then IO operation and When it is low then Memory operation.

**S1 and S0-** status signals,similar to IO/M,can identify various operations.that are rarely used in the systems.

**Instruction Set**

An instruction is a binary pattern designed inside a microprocessor to perform a specific function. Each instruction is represented by 8 bit binary value.

Types of [instruction set](http://mvn.edu.in/mvnlms/mod/assign/view.php?id=904):

**1)      Data transfer instructions:**

Instructions, which are used to transfer data from one register to another register, from memory to register or register to memory, come under this group. Examples are: MOV, MVI, LXI, LDA, STA etc. When an instruction of data transfer group is executed, data is transferred from the source to the destination without altering the contents of the source. For example, when MOV A, B is executed the content of the register B is copied into the register A, and the content of register B remains unaltered. Similarly, when LDA 2500 is executed the content of the memory location 2500 is loaded into the accumulator. But the content of the memory location 2500 remains unaltered.

#### EXAMPLES:

1. MOV r1, r2 (Move Data; Move the content of the one register to another).  [r1] <-- [r2]

2. MOV r, m (Move the content of memory register). r <-- [M]

3. MOV M, r. (Move the content of register to memory). M <-- [r]

4. MVI r, data. (Move immediate data to register). [r] <-- data.

5. MVI M, data. (Move immediate data to memory). M <-- data.

6. LXI rp, data 16. (Load register pair immediate). [rp] <-- data 16 bits, [rh] <-- 8 LSBs of data.

7. LDA addr. (Load Accumulator direct). [A] <-- [addr].

8. STA addr. (Store accumulator direct). [addr] <-- [A].

9. LHLD addr. (Load H-L pair direct). [L] <-- [addr], [H] <-- [addr+1].

10.SHLD addr. (Store H-L pair direct) [addr] <-- [L], [addr+1] <-- [H].

11.LDAX rp. (LOAD accumulator indirect) [A] <-- [[rp]]

12.STAX rp. (Store accumulator indirect) [[rp]] <-- [A].

13.XCHG. (Exchange the contents of H-L with D-E pair) [H-L] <-->  [D-E].

**2) Arithmatic instructions**:

The instructions of this group perform arithmetic operations such as addition, subtraction; increment or decrement of the content of a register or memory.

**Examples:**

1). ADD r. (Add register to accumulator) [A] <-- [A] + [r].

2) .ADD M. (Add memory to accumulator) [A] <-- [A] + [[H-L]].

3).ADC r. (Add register with carry to accumulator). [A] <-- [A] + [r] + [CS].

4). ADC M. (Add memory with carry to accumulator) [A] <-- [A] + [[H-L]] [CS].

5) .ADI data (Add immediate data to accumulator) [A] <-- [A] + data.

6) .ACI data (Add with carry immediate data to accumulator). [A] <-- [A] + data + [CS].

7).DAD rp. (Add register paid to H-L pair). [H-L] <-- [H-L] + [rp].

8).SUB r. (Subtract register from accumulator). [A] <-- [A] – [r].

9).SUB M. (Subtract memory from accumulator). [A] <-- [A] – [[H-L]].

10).SBB r. (Subtract register from accumulator with borrow). [A] <-- [A] – [r] – [CS].

11).SBB M. (Subtract memory from accumulator with borrow). [A] <-- [A] – [[H-L]] – [CS].

12).SUI data. (Subtract immediate data from accumulator) [A] <-- [A] – data.

13).SBI data. (Subtract immediate data from accumulator with borrow). [A] <-- [A] – data – [CS].

14).INR r (Increment register content) [r] <-- [r] +1.

15). INR M. (Increment memory content) [[H-L]] <-- [[H-L]] + 1.

16).DCR r. (Decrement register content). [r] <-- [r] – 1.

17).DCR M. (Decrement memory content) [[H-L]] <-- [[H-L]] – 1.

18).INX rp. (Increment register pair) [rp] <-- [rp] – 1.

19).DCX rp (Decrement register pair) [rp] <-- [rp] -1.

20).DAA (Decimal adjust accumulator) .

**3) Logical instructions:**

The Instructions under this group perform logical operation such as AND, OR, compare, rotate etc.

**Examples:**

1). ANA r. (AND register with accumulator) [A] <-- [A] ^ [r].

2).ANA M. (AND memory with accumulator). [A] <-- [A] ^ [[H-L]].

3).ANI data. (AND immediate data with accumulator) [A] <-- [A] ^ data.

4).ORA r. (OR register with accumulator) [A] <-- [A] v [r].

5).ORA M. (OR memory with accumulator) [A] <-- [A] v [[H-L]]

6).ORI data. (OR immediate data with accumulator) [A] <-- [A] v data.

7).XRA r. (EXCLUSIVE – OR register with accumulator) [A] <-- [A] v   [r]

8).XRA M. (EXCLUSIVE-OR memory with accumulator) [A] <-- [A] v  [[H-L]]

9).XRI data. (EXCLUSIVE-OR immediate data with accumulator) [A] <-- [A]

10).CMA. (Complement the accumulator) [A] <-- [A]

11).CMC. (Complement the carry status) [CS] <-- [CS]

12).STC. (Set carry status) [CS] <-- 1.

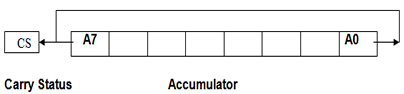
13).CMP r. (Compare register with accumulator) [A] – [r]

14).CMP M. (Compare memory with accumulator) [A] – [[H-L]]

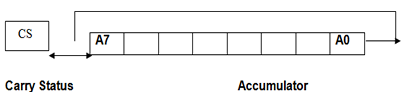
15)CPI data. (Compare immediate data with accumulator) [A] – data.

The 2nd byte of the instruction is data, and it is subtracted from the content of the accumulator. The status flags are set according to the result of subtraction. But the result is discarded. The content of the accumulator remains unchanged.

**3) Logical instructions cont..:**

16).RLC (Rotate accumulator left) [An+1] <-- [An], [A0] <-- [A7],[CS] <-- [A7]**.**

The content of the accumulator is rotated left by one bit. The seventh bit of the accumulator is moved to carry bit as well as to the zero bit of the accumulator. Only CS flag is affected.

17).RRC. (Rotate accumulator right) [A7] <-- [A0], [CS] <-- [A0], [An] <-- [An+1].

The content of the accumulator is rotated right by one bit. The zero bit of the accumulator is moved to the seventh bit as well as to carry bit. Only CS flag is affected.

18). RAL. (Rotate accumulator left through carry) [An+1] <-- [An], [CS] <-- [A7], [A0] <-- [CS].

19).RAR. (Rotate accumulator right through carry) [An] <-- [An+1], [CS] <-- [A0], [A7] <-- [CS].

**4) Branching Instructions:**

This group includes the instructions for conditional and unconditional jump, subroutine call and return, and restart.

**Examples:**

1. MP addr (label). (Unconditional jump: jump to the instruction specified by the address). [PC] <-- Label.
2. Conditional Jump addr (label): After the execution of the conditional jump instruction the program jumps to the instruction specified by the address (label) if the specified condition is fulfilled. The program proceeds further in the normal sequence if the specified condition is not fulfilled. If the condition is true and program jumps to the specified label, the execution of a conditional jump takes 3 machine cycles: 10 states. If condition is not true, only 2 machine cycles; 7 states are required for the execution of the instruction.
   1. **JZ** addr (label). (Jump if the result is zero)
   2. **JNZ** addr (label) (Jump if the result is not zero)
   3. **JC** addr (label). (Jump if there is a carry)
   4. **JNC** addr (label). (Jump if there is no carry)
   5. **JP** addr (label). (Jump if the result is plus)
   6. **JM** addr (label). (Jump if the result is minus)
   7. **JPE** addr (label) (Jump if even parity)
   8. **JPO** addr (label) (Jump if odd parity)
3. CALL addr (label) (Unconditional CALL: call the subroutine identified by the operand)

CALL instruction is used to call a subroutine. Before the control is transferred to the subroutine, the address of the next instruction of the main program is saved in the stack. The content of the stack pointer is decremented by two to indicate the new stack top. Then the program jumps to subroutine starting at address specified by the label.

1. RET (Return from subroutine).
2. RST n (Restart) Restart is a one-word CALL instruction. The content of the program counter is saved in the stack. The program jumps to the instruction starting at restart location.

### Instruction Set Contd. With sample prog.

**5). Stack,I/O and Machine control instructions:**

1. N port-address. (Input to accumulator from I/O port) [A] <-- [Port]
2. OUT port-address (Output from accumulator to I/O port) [Port] <-- [A]
3. PUSH rp (Push the content of register pair to stack)
4. PUSH PSW (PUSH Processor Status Word)
5. POP rp (Pop the content of register pair, which was saved, from the stack)
6. POP PSW (Pop Processor Status Word)
7. HLT (Halt)
8. XTHL (Exchange stack-top with H-L)
9. SPHL (Move the contents of H-L pair to stack pointer)
10. EI (Enable Interrupts)
11. DI (Disable Interrupts)
12. SIM (Set Interrupt Masks)
13. RIM (Read Interrupt Masks)
14. NOP (No Operation).

**Sample programs of instruction sets:**

**1).program for adding two 8 bit numbers.**

 // Load accumulator from memory address

        // Move the content from accumulator to *B*

     //  Load accumulator from memory address 1001*H*

           // Add the content of B to accumulator and store the result in accumulator

     // Store the content of accumulator in memory location 1002*H*

                             // Halt

**Another assembly language instruction sequence to solve the same problem.**

               // Load lower order byte of address to register

            // Load higher order byte of address to register

            // Move the contents of memory location addressed by pair to

            //  Load the accumulator from memory location

              // Add the content of to accumulator

            // Store the content of accumulator to memory location

               // Halt.

**1).program for Multiplying two 8 bit numbers.**

            MVI A,00  // Load immediate data into accumulator.

            MVI B,02//  Load immediate data into register B.

            MVI C,04// Load immediate data into register C.

LOOP:    ADD B  // Add the content of to accumulator.

            DCR C // Decrement the content of register C by 1.

JNZ  LOOP

            STA 1051H// Store the content of accumulator to memory location 1051H

            HLT    // Halt