UNIT 2

8085 Microprocessor

(16 bit address bus, 8 bit data bus)

1. Architecture of 8085 microprocessor -

   INTR INTA 5S 6S 7S TRAP
   RST
   INTERRUPT CONTROL
   SERIAL I/O CONTROL

   8-bit Internal Data Bus

   ACCU. (B)
   TEMP. REGISTER (B)
   FLAG (E)
   FLIP FLOPS
   ARITHMETIC LOGIC UNIT (ALU) (B)
   INSTRUCTION REGISTER (B)
   INSTRUCTION DECODER AND MACHINE CYCLE ENCODING
   MULTIPLEXER

   POWER SUPPLY +5V GND

   IN
   OUT

   CLK GEN
   CONTROL
   STATUS
   DMA
   RESET
   RESET OUT

   CLK READY RD WR AAE S5 S4 S3 S2 S1 S0 HOLO HLOD

   8085 consists of three main sections -

   (1) Arithmetic Logic Unit (ALU)
   (2) Timing and Control Unit
   (3) Set of registers

   Arithmetic Logic Unit (ALU) -

   Sequence of operations in ALU are given below -

   (4) One operand is in the A register
   (2) The other operand may be in the general purpose register or memory locations, which will be transferred to the temporary register...
(3) Then content of the accumulator and temporary registers are considered as input of ALU and the specified operation is carried out in the ALU.
(4) The result of ALU operation is transferred in the A register through internal data bus.
(5) The content of the flag register will be changed depending on the result.

ALU operations are addition, subtraction, logical AND, logical OR, logical EXCLUSIVE OR, complement, increment by 1, decrement by 1, rotate left, rotate right, clear.

→ Timing and Control Unit: (Brain of the microprocessor)
It controls the operations of different units while the CPU generates timing sequence signals for the execution of instructions.
→ Registers:
1. Accumulator (A or ACC) - 8-bit register
   It is used to store the data and to perform arithmetic as well as logical operations, and also store the result.
2. General-Purpose Register - (B, C, D, E, H, and L) - 8-bit register
   Register Group (BC, DE, and HL) - 16-bit data or memory address
   It is used to store operands.
3. Special-Purpose Register
   (i) Program Counter (PC) - 16-bit register
   It is used to hold the memory address of the next instruction which will be executed by incrementing the content of the PC.
   (ii) Stack Pointer (SP) - 16-bit register
   It is used to point the memory location called the stack. The stack is a sequence of memory locations in the R/W memory.
   (iii) Instruction Register (IR) - 8-bit register
   It holds the operation code (opcode) of the current instruction of a program during an arithmetic/ logical operation.
Temporary Register - 8-bit Register
9-bits hold data during arithmetic and logical operations. This register can be used by the microprocessor but is not accessible to the programmer.

Flag / Status Register (SR) -

A flag includes five flip-flops which are set or reset after an AHU operation. The status of each flip-flop is known as a flag.

<table>
<thead>
<tr>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z</td>
<td>S</td>
<td>A</td>
<td>X</td>
<td>P</td>
<td>X</td>
<td>C</td>
<td>Y</td>
</tr>
</tbody>
</table>

Five flags are:

1. Carry Flag (CY) -
A carry is produced (when num is larger than 8 bits) and during subtraction (if borrow is generated), CY is set to 1; otherwise, 0.

2. Parity Flag (P) -
Number of 1's in the result is even (even parity), then P is set to 1; if the number of 1's in the result is odd (odd parity), then P is set to 0.

3. Auxiliary Carry Flag (AC) -
If carry is generated by D3 bit and passed on to D4, then AC is set to 1; otherwise, 0.

4. Zero Flag (Z) -
When an 8-bit AHU operation result in zero, the Z is set to 1; otherwise, 0.

5. Sign Flag (S) -
When a number is negative, the S is set to 1; otherwise, 0.

Pin Diagram of Intel 8085 -

1. A15 - A0 - (Output, 3-state: HOLD, HALT & RESET) → Address Bus
2. AD2 - AD0 - (I/O, 3-state) → Address/Data Bus
3. AE (Address latch enable) -

During the 1st clock rate of a machine cycle, it becomes high and enable the address to get latched either into the memory or internal latch.
(a) Status code and status of 8085 -

<table>
<thead>
<tr>
<th>Machine Cycle Status</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 1</td>
<td>Memory Write</td>
</tr>
<tr>
<td>0 1 0</td>
<td>Memory Read</td>
</tr>
<tr>
<td>0 1 1</td>
<td>Opcode Fetch</td>
</tr>
<tr>
<td>1 0 1</td>
<td>I/O Write</td>
</tr>
<tr>
<td>1 1 0</td>
<td>I/O Read</td>
</tr>
<tr>
<td>1 1 1</td>
<td>Interrupt Acknowledgement</td>
</tr>
</tbody>
</table>

(b) Continuously initiated signals

- TRAP, RST 7.5, RST 6.5, INTA, READY

(c) Continuously initiated_acknowledged

- ALE

(d) Control & Status registers

- HOLD, S0, S1, IO/M, RD, WR

(e) IO/Communication

- SIO
- SID

(f) Serial

- TRAP
- RST 7.5
- RST 6.5
- INTR

(g) Priority

- 1
- 2
- 3
- 4
- 5

(h) INTA

- 10

(i) AD0

- 12

(j) AD1

- 13

(k) AD2

- 14

(l) AD3

- 15

(m) AD4

- 16

(n) AD5

- 17

(o) AD6

- 18

(p) A8

- 19

(q) GND
(5) **RD (Read memory or I/O Devices)** - 3-state, output
(6) **WR (Write memory or I/O Devices)** - 3-state, output
(7) **READY (Input)** - Indicates that the memory or I/O device is ready
(8) **HOLD (Input)** - Requesting the use of address and data bus
(9) **MADA (Output)** - Hold acknowledgement
(10) **INTR (Input)** - Interrupt Request
(11) **INTA (Output)** - Interrupt Acknowledgement
(12) **RST 75, RST 65, RST 55** - Restart Interrupts
(13) **TRAP & Input** - Non-maskable interrupt
(14) **RESET IN (Input)** - Reset PC = 0 and also Interrupt enable 1 MADA flip flop
(15) **RESET OUT (Output)** - Indicates CPU is in RESET condition
(16) **X1, X2 (Input)** - Connected to crystal, 600kHz clock frequency
(17) **CHK (Output)** - Used as system clock & twice X1, X2 input time period
(18) **SIO (Input)** - Serial Input Data line
(19) **SOO (Output)** - Serial Output Data line
(20) **Vcc (+5V supply)**
(21) **GND (Ground reference)**

---

3. **Addressing modes of 8085**

Different techniques of specifying data are called Addressing modes. Four types are:

(i) **Immediate Addressing**
- The immediate data is stored in the destination, which is given in the instruction. Eg: MVI D, FFH

(ii) **Register Addressing**
- Data is provided through the register. Eg.- MOV A, B

(iii) **Direct Addressing**
- It is used to read data from output devices, store in accumulator or write data from accumulator to output devices. Eg: LDA T8000H

(iv) **Indirect Addressing**
- The content of specified register is used to specify the address of the operand. Eg- ADD M
(4) Instruction set of 8085

1. Data Transfer Operations

<table>
<thead>
<tr>
<th>Types of Data Transfer</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Between Registers</td>
<td>MOV A, B</td>
</tr>
<tr>
<td>- Between a specific data to a register or a memory location</td>
<td>MOV A, B</td>
</tr>
<tr>
<td>- Between a memory location &amp; a register</td>
<td>MOV B, [5000H]</td>
</tr>
<tr>
<td>- Between an I/O device &amp; the accumulator</td>
<td>INH, 00H</td>
</tr>
<tr>
<td>- Between Register, Page</td>
<td>LXI H, D</td>
</tr>
</tbody>
</table>

2. Arithmetic Operations

- Performs arithmetic operations such as addition, subtraction and increment/decrement data in registers or memory.
  - Eg: ADD, SUB, INX, DEX

3. Logical Operations

- Performs logical operations such as AND, OR, Exclusive-OR, (left/right) Rotate, complement and complement with the contents of accumulator.
  - Eg: ANA (AND), ORA (OR), XRA (Exclusive-OR), CMP, RLC, RRC, CMA (complement accumulator)

4. Branch Control Operations

- Instructions that change the sequence of program execution, using conditional jump & unconditional jumps, redirected call and return, and branch. Eg: JMP, CALL, RET

5. Stack, I/O and machine control operations

- Stack operations - PUSH, POP, XTHL (Exchange), CALL (Jump), JMP (Jump to)
- I/O operations - IN, OUT
- Machine Control operations - EI (Enable interrupt system), DI (Disable interrupt system), HTI, NOP (No operation)
Assembly language programs of 8085 microprocessor

Stacks –
It is a group of memory locations in Read/Write (R/W) memory of any microcomputer and is used to store the contents of the register, operand and memory locations addresses. The initial address location of the stack is defined by loading a 16 bit address into the stack pointer (LXI SP).

Subroutines –
A group of instructions are known as subroutines. When a main program calls a subroutine, the program execution is transferred to the subroutine and after the completion of the subroutine, the program execution returns to the main program. The microprocessor uses the stack to store the return address of the subroutine.

Types of Subroutines –
Multiple CALL Subroutines, Nested Subroutines, Multiplied ending subroutines.

Time-Delay loops –
When some time delay is required between two operations, a time-delay loop is used to provide it.

Diagram:
- Initialize Delay Register
- Initialize Delay Register Pair
- Decrement Register
- Decrement Register Pair
- If Register = 0?
- If Register Pair = 0?
- Yes
- No
- Load Register B
- Load Register C
- Decrement Register C
- If Register C = 0?
- Yes
- No
- Decrease B Register
- If Register B = 0?
- Yes
- No

Pipelined Programming
9. Modular Programming -
   Complete is divided into sub-problems or small modules.
   Each independent module is separately named and are individually
   maintainable program elements. The size of modules are reduced to a
   humanly comprehensible and manageable level.

   Characteristics of module:
   (1) Each module is independent of other modules.
   (2) Each module has one input and one output.
   (3) A module is small in size.
   (4) Programming a single function per module is a goal.

10. Macro -
    Some new instructions can be developed using a sequence of
    known instructions. These new instructions are always assigned a name
    and known as MACRO, used in assembly-language programming.
    Eg.: LARGE, DELAY, SMALL, MUL, DIV etc.

    General MACRO form:

    Name    MACRO arg
    Statement-1
    Statement-2
    ;
    Statement-n
    ENDM

11. Instruction Format of Assembly Language:

    | MR | MC | L | M | O | C |
    |----|----|---|---|---|---|
    | MR | Memory address | MC | machine code (operation code) |
    | L  | label | M  | Mnemonic (states the operation which will be executed) |
    | O  | Operands | C  | Comments (Optional) (first character must be alphabetical |