5th -Semester Electrical Engineering

Digital Electronics and Microprocessor Detailed Notes by Bibhu Prasad Das

1.1 Binary, Octal, Hexadecimal Number Systems and Comparison with Decimal System

Decimal Number System (Base-10):

- Uses 10 digits: 0 to 9.
- Each digit has a positional value based on powers of 10.
- Example: $325 = 3\times10^2 + 2\times10^1 + 5\times10^0 = 300 + 20 + 5 = 325$

Binary Number System (Base-2):

- Uses only two digits: 0 and 1.
- Each bit has a weight of power of 2.
- Example: $1011 = 1\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0 = 8 + 0 + 2 + 1 = 11$ (Decimal)

Octal Number System (Base-8):

- Uses digits: 0 to 7.
- Each digit's place value is a power of 8.
- Example: $157_8 = 1\times8^2 + 5\times8^1 + 7\times8^0 = 64 + 40 + 7 = 111$ (Decimal)

Hexadecimal Number System (Base-16):

- Uses digits: 0 to 9 and A to F (A=10, ..., F=15)
- Each digit's weight is power of 16.
- Example: $1F = 1\times16 + 15 = 31$ (Decimal)

Comparison Table:

Decimal	Binary	Octal	Hexadecimal
0	0000	0	0

1	0001	1	1
10	1010	12	A
15	1111	17	F
31	11111	37	1F

1.2 Binary Arithmetic

Binary Addition Rules:

- 0 + 0 = 0
- 0 + 1 = 1
- 1 + 0 = 1
- 1 + 1 = 10 (0 with carry 1)

Binary Subtraction Rules:

- 0 - 0 = 0
- 1 - 0 = 1
- 1 - 1 = 0
- 0 - 1 = 1 (with borrow 1)

Binary Multiplication:

- Same as decimal: Multiply and shift.
- Example:

```
  101 (5)
x  11 (3)
-----
  101
 1010
-----
 1111 (15)
```

Binary Division:

- Use long division method.
- Example: 1011 ÷ 10 = 101 (5 remainder 1)

1.3 1's and 2's Complement of a Binary Number

1's Complement:

- Invert each bit.
- Example: 1010 → 0101

2's Complement:

- Add 1 to 1’s complement.
- Example: 1010 → 1’s comp = 0101 → +1 = 0110

1.4 Binary Subtraction using 2's Complement Method

Steps:

1. Take 2's complement of subtrahend.
2. Add it to the minuend.
3. Discard carry if present.

Example: 5 - 3

- 5 = 0101
- 3 = 0011 → 2's comp = 1101
- Add: 0101 + 1101 = 10010 → Discard carry → 0010 = 2

1.5 Weighted and Un-weighted Codes

Weighted Codes:

- Each digit position has a specific weight.
- Example: BCD (8421 code)
  - Decimal 9 = 1001 in BCD

Un-weighted Codes:

- No weight assigned to bit positions.
- Examples: Excess-3, Gray code

Excess-3 Code:

- Add 3 to BCD value
- Decimal 4 = BCD 0100 → +3 = 0111 (Excess-3)

Gray Code:

- Only one bit changes at a time.
- Binary to Gray:
  - MSB remains same
  - Other bits: current bit XOR previous bit

1.6 Importance of Parity Bit

- Used in error detection
- Even parity: Total 1s are even
- Odd parity: Total 1s are odd
- If parity mismatch occurs, error is detected

1.7 Logic Gates with Truth Tables

| Gate | Symbol | Logic Expression | Truth Table (A, B → Output) |

| ----- | ------- | ------------------- | ------------------------------- |
AND	A·B	A AND B	00→0, 01→0, 10→0, 11→1
OR	A+B	A OR B	00→0, 01→1, 10→1, 11→1
NOT	~A	NOT A	0→1, 1→0
NAND	¬(A·B)	NOT AND	00→1, 01→1, 10→1, 11→0
NOR	¬(A+B)	NOT OR	00→1, 01→0, 10→0, 11→0
XOR	A⊕B	Exclusive OR	00→0, 01→1, 10→1, 11→0

1.8 Logic Gates Realization using NAND & NOR

Using NAND Gates Only:

- NOT A = A NAND A
- A AND B = (A NAND B) NAND (A NAND B)
- A OR B = (A NAND A) NAND (B NAND B)

Using NOR Gates Only:

- NOT A = A NOR A
- A OR B = (A NOR B) NOR (A NOR B)
- A AND B = (A NOR A) NOR (B NOR B)

1.9 Boolean Algebra Postulates & De Morgan's Theorems

Postulates:

- A + 0 = A
- A + 1 = 1
- A · 0 = 0
- A · 1 = A
- A + A = A
- A · A = A
- A + A' = 1
- A · A' = 0

De Morgan's Theorems:

1. ¬(A·B) = ¬A + ¬B
2. ¬(A + B) = ¬A · ¬B

1.10 Boolean Algebra Simplification

Example:

- F = A·B + A·B'
- F = A(B + B') = A · 1 = A

1.11 Karnaugh Maps (K-Map)

2-Variable K-Map: 4 cells
3-Variable K-Map: 8 cells
4-Variable K-Map: 16 cells
The Karnaugh map or k-map is a graphical technique for simplifying Boolean function. The k-map in a two dimensional representation of a truth table it provides a simpler method for minimizing logic expressions. The map method is ideally suited for four or less variables. But it becomes cumbersome for five or more variable. A karnaugh map is a diagram consisting of squares. Each square of the map represent either min term or max term. Any logic expression can be written as either sum of product term or product of sum term which is also called sum of min term and product of max term respectively. Therefore a logic expression can be easily represented on a karnaugh map.

A KARNAUGGH map for n variables made up of 2^n squares. Each square designate a product term in Boolean expression. For product term which are present in the expression. In case of sum of product, 1s can be written in Present Square and in case of product of sum term, os can be written in Present Square. Blank Square or opposite square indicate. That then contains nothing.
We can construct KARNAUGH map in two different forms for sum of product term and for product of sum term. Because in sum of product contains zero as complement value and one as un-complement value similarly in the product of sum term we use 1 as complement and 0 as un-complemented value function of both sum of product and product of sum about same except that the complement and un-complement value.

2 variable k-map: – In the 2 variable k-map, four squares are constructed. Each square contains one term of expression with two variables. **A 2 variable k-map for SOP and POS form are as follows: –**

A. SOP: -

A \ B	$\overline{B}$ 0	B 1
$\overline{A}$ 0	$\overline{A}.\overline{B}$	$\overline{A}.B$
A 1	$A.\overline{B}$	A.B

B. POS: -

A \ B	B 0	$\overline{B}$ 1
A 0	A+B	$A+\overline{B}$
$\overline{A}$ 1	$\overline{A}+B$	$\overline{A}+\overline{B}$

3 variables k-map: – In the three variable k-map, 8 square is required. The 3-variables k-map can content either horizontally or vertically. **An example of 3 variables k-map for SOP and POS form are as follows: –**

A. SOP: -

A \ BC	$\overline{B}\,\overline{C}$ 00	$\overline{B}C$ 01	BC 11	$B\overline{C}$ 10
$\overline{A}$ 0	$\overline{A}\,\overline{B}\,\overline{C}$ 0	$\overline{A}\,\overline{B}C$ 1	$\overline{A}BC$ 3	$\overline{A}B\overline{C}$ 2
A 1	$A\overline{B}\,\overline{C}$ 4	$A\overline{B}C$ 5	ABC 7	$AB\overline{C}$ 6

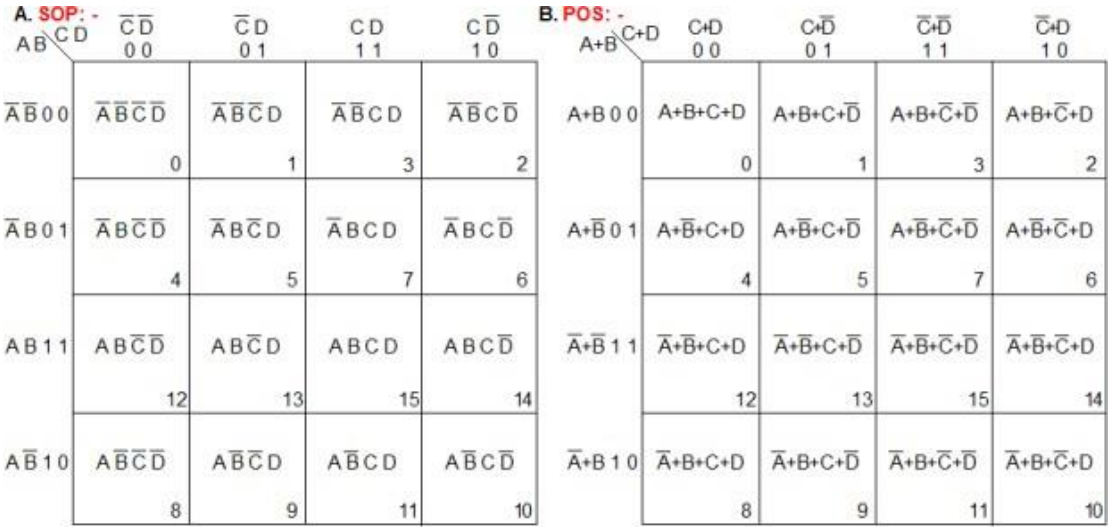
AB \ C	$\overline{C}$ 0	C 1
$\overline{A}\,\overline{B}$ 00	$\overline{A}\,\overline{B}\,\overline{C}$ 0	$\overline{A}\,\overline{B}C$ 1
$\overline{A}B$ 01	$\overline{A}B\overline{C}$ 2	$\overline{A}BC$ 3
AB 11	$AB\overline{C}$ 6	ABC 7
$A\overline{B}$ 10	$A\overline{B}\,\overline{C}$ 4	$A\overline{B}C$ 5

B. POS: -

A \ B+C	B+C 00	B+$\overline{C}$ 01	$\overline{B}$+$\overline{C}$ 11	$\overline{B}$+C 10
A 0	A+B+C 0	A+B+$\overline{C}$ 1	A+$\overline{B}$+$\overline{C}$ 3	A+$\overline{B}$+C 2
$\overline{A}$ 1	$\overline{A}$+B+C 4	$\overline{A}$+B+$\overline{C}$ 5	$\overline{A}$+$\overline{B}$+$\overline{C}$ 7	$\overline{A}$+$\overline{B}$+C 6

A+B \ C	C 0	$\overline{C}$ 1
A+B 0 0	A+B+C 0	A+B+$\overline{C}$ 1
A+$\overline{B}$ 0 1	A+$\overline{B}$+C 2	A+$\overline{B}$+$\overline{C}$ 3
$\overline{A}$+$\overline{B}$ 1 1	$\overline{A}$+$\overline{B}$+C 6	$\overline{A}$+$\overline{B}$+$\overline{C}$ 7
$\overline{A}$+B 1 0	$\overline{A}$+B+C 4	$\overline{A}$+B+$\overline{C}$ 5

4 variable k-map: – In the 4 variable k-map, 16 square are required. **An example of 4 variable k-map for SOP and POS form are as follows: –**



2. COMBINATIONAL LOGIC CIRCUITS

2.1 Concept of Combinational Logic Circuits

- Output depends only on present input.
- No memory element involved.
- Examples: Adders, Subtractors, Multiplexers, etc.

2.2 Half Adder Circuit

- Adds two binary digits (A and B).
- **Sum = A ⊕ B**
- **Carry = A · B**

A B Sum Carry

0 0 0 0

0 1 1 0

1 0 1 0

1 1 0 1

2.3 Half Adder using NAND/NOR Gates

Using NAND:

- Use NAND-based XOR and AND circuits **Using NOR:**
- Use NOR equivalents to realize XOR and AND

2.4 Full Adder Circuit

- Adds three bits: A, B, and Cin (carry-in)
- **Sum = A ⊕ B ⊕ Cin**
- **Carry = AB + BCin + ACin**

A	B	Cin	Sum	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

2.5 Full Adder using Two Half Adders and OR Gate

- 1st HA: A, B → S1, C1
- 2nd HA: S1, Cin → Sum, C2
- OR gate: C1 + C2 = Cout

2.6 Full Subtractor

- Inputs: A, B, Bin
- Outputs: Diff = A ⊕ B ⊕ Bin
- Borrow = A'·B + (A ⊕ B)'·Bin

A	B	Bin	Diff	Bout
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

2.7 4×1 Multiplexer (MUX)

- Selects one of 4 inputs (I0 to I3) based on 2 select lines (S1, S0)
- Output = I0·S1'·S0' + I1·S1'·S0 + I2·S1·S0' + I3·S1·S0

2.8 1×4 Demultiplexer (DEMUX)

- Routes input to one of 4 outputs based on 2 select lines.
- Example: D input goes to Y0, Y1, Y2, or Y3.

2.9 Binary to Decimal Encoder

- Converts active input line into binary code.
- Example: Input = 0100 → Output = 10 (for line D2 active)

2.10 3×8 Decoder

- Converts 3-bit input to 1 of 8 outputs.
- Each output represents one binary combination.

2.11 Two-Bit Magnitude Comparator

- Compares two 2-bit numbers (A1A0 and B1B0)
- Outputs:
  - A>B
  - A=B
  - A<B
- Uses logic expressions to derive conditions.

3. SEQUENTIAL LOGIC CIRCUITS

3.1 Concept of Sequential Logic Circuits

- Output depends on present input and past history.
- Memory elements are used (flip-flops).

3.2 Necessity of Clock, Level Clocking, and Edge Triggering

- Clock synchronizes operations.
- **Level Clocking**: Triggering during a HIGH or LOW level.
- **Edge Triggering**: Triggering occurs at the rising or falling edge of the clock.

3.3 Clocked SR Flip-Flop with Preset and Clear

- S (Set), R (Reset), Clock, Preset, Clear inputs.
- Preset → Set output Q to 1.
- Clear → Reset output Q to 0.

3.5 Level Clocked JK Flip-Flop using SR

- Overcomes invalid state of SR Flip-Flop.

- J = S, K = R with feedback.
- **J = K = 1** → Toggle condition.

J K Q(next)
0 0 Q
0 1 0
1 0 1
1 1 ~Q

3.6 Race Around Condition and Master-Slave JK FF

- Occurs when J=K=1 with level triggering.
- **Solution**: Master-Slave JK Flip-Flop.
- Two JK flip-flops in series with opposite clock phases.

3.7 Edge Triggered D and T Flip-Flop

- **D Flip-Flop**: Stores input D on clock edge.

D Q(next)
0 0
1 1

- **T Flip-Flop**: Toggles output when T = 1.

T Q(next)
0 Q
1 ~Q

3.8 Applications of Flip-Flops

- Counters
- Registers
- Memory Storage
- Frequency Division

3.9 Modulus of a Counter

- Number of unique states before repeating.
- Example: Mod-4 counter counts 0 to 3.

3.10 4-Bit Asynchronous Counter

- Ripple counter.

- Each flip-flop triggers the next.
- Output changes asynchronously.

3.11 Asynchronous Decade Counter

- Mod-10 counter.
- Counts from 0 to 9 and resets.
- Requires logic to reset after 1010 (decimal 10).

3.12 4-Bit Synchronous Counter

- All flip-flops clocked simultaneously.
- No ripple delay.
- More reliable than asynchronous.

3.13 Synchronous vs Asynchronous Counters

Feature	Synchronous	Asynchronous
Triggering	Same clock	Ripple from previous
Speed	Faster	Slower
Complexity	More complex	Simple

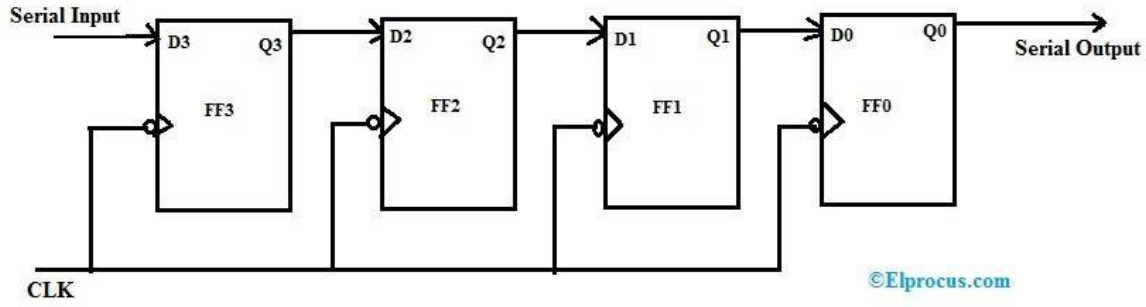
3.14 Need for Register and Types

- Used to store multi-bit data.
- Types:
    1. SISO (Serial-In Serial-Out)
    2. SIPO (Serial-In Parallel-Out)
    3. PISO (Parallel-In Serial-Out)
    4. PIPO (Parallel-In Parallel-Out)

3.15 Working of SISO, SIPO, PISO, PIPO using Flip-Flop

- Flip-flops connected in desired configuration.
- **Truth tables** show shifting of bits per clock pulse.
- # What is SISO Shift Register?
- The term “SISO” stands for “Serial-In Serial-Out”. The SISO shift register circuit accepts serial data on its input pin and shifts it out serially on its output pin. The number of bits that can be shifted out before the next bit arrives depends on the speed of the clock signal that controls the operation of the shift register. This type of shift register can be used as a buffer between two asynchronous devices that communicate with each other using signals with different frequencies or phases.

- The **SISO shift register block diagram** is shown below which includes 3-D flip-flops. The connection of these FFs can be done by connecting the one Flip Flops output to the next flip flop input. So these FFs are synchronous through each other because the equal CLK signal is applied in each Flip Flop.

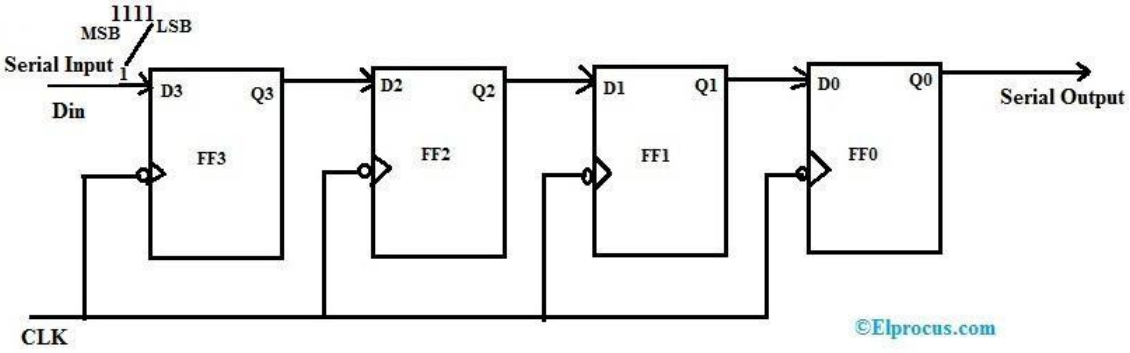


SISO Shift Register Block Diagram

- This shift register includes simply three connections the SI (serial input), SO (serial output), and CLK (clock signal). Here, the SI determines the input enters into the left-hand side flip-flop, the SO is the output taken from the right-hand side flip-flop & the sequencing CLK signal.
- In this type of shift register, the bits can be transmitted serially from the serial input of the flip flop. For each clock signal in the FFs, the data can be transmitted from one phase to the next phase. As a result, we can get the bits in sequence from the D-FFs output which is known as serial output.

SISO Shift Register Circuit Diagram

- We know that the input in this type of shift register is serially fed whereas the output is received in a serial way. The SISO shift register circuit diagram is shown below where the D FFs like D0 to D3 are connected serially as shown in the following diagram.



SISO Shift Register Circuit Diagram

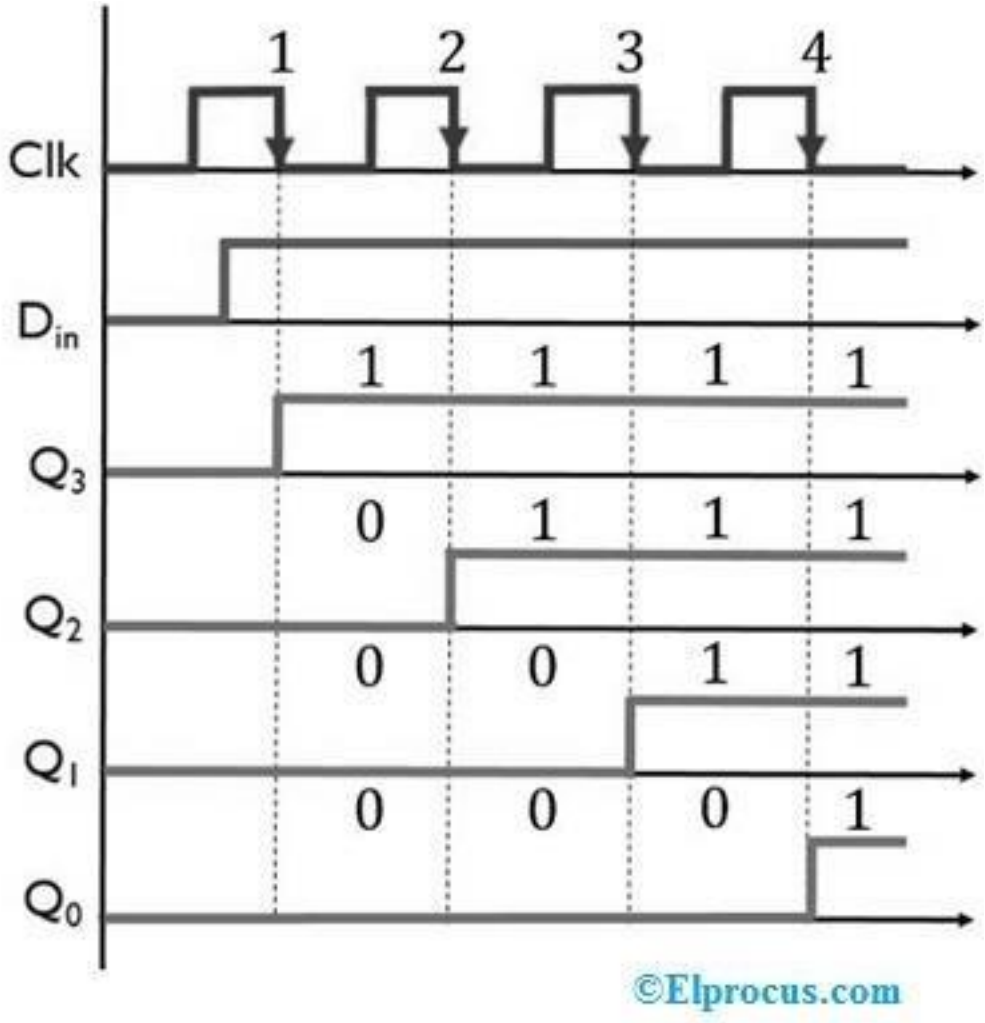
- At first, all the four D flip-flops are set to reset mode so that each flip-flop's output within the circuit is low which is '0'.
- This is a shift right mode circuit which means when the data i/p is given at the left end of the flip flop then the stored bit can be shifted to the right side to generate serial output. Now let us discuss how the given data is stored within this register.

Working of SISO Shift Register

- Let us take an example of the 1011 binary number. Before that, the circuit must be set to reset mode so the output of every register will be '0', so the output provided by all the registers will be "0000".
- In this 4-bit shift register example like "1111", the LSB bit is '1' and the MSB bit is '1'. First, the high signal (LSB bit) is used as an input to the first D3 flip flop, then D3=1. But primarily all the D FFs outputs will be 0. So, D2=D1=D0=0. When D3 input is high signal (1) then D3 will cause 'Q3' to be '1'. Therefore the overall o/p for 1st falling edge will become 1000.
- Similarly, when the next data i/p bit in the above 4-bit like high signal (1) is given at flip flop D3, again this 'D3' will cause 'Q3' to be 1, however, 'Q3' is given as input to FF 'D2'. So, this 'D2'will cause 'Q2' to be 1 when all the remaining outputs will become 0.
- As a result, we will obtain '11' for a 2nd falling edge; so will obtain '11' at the stored bit in the shift register, thus the overall o/p for the 2nd falling edge will get o/p as"1100".
- When the third input bit like high signal (1) is applied at the 'D3' FF then earlier 'Q2' o/p will cause the 'D1' i/p to be '0'. This will give the output Q3, Q2 & Q1 as '1' whereas 'Q0' will be '0'. So the overall o/p for the 3rd falling edge will get o/p as "1110".
- In addition, an MSB bit like high signal (1) is given as input, after that '1' at 'Q1' will cause input 'D0' to be '1', thus, this will make 'Q0' be '1'. Therefore, finally, SISO shift register store 1111 bit & shows in the o/p.
- The **SISO shift register truth table** is shown below.

CLK	'Q3'	'Q2'	'Q1'	'Q0'
Initially (Reset)	0	0	0	0
1st Falling Edge	1	0	0	0
2nd Falling Edge	1	1	0	0
3rd Falling Edge	1	1	1	0
4th Falling Edge	1	1	1	1

- By considering the above truth table, the SISO shift register waveform representation will be like the following.



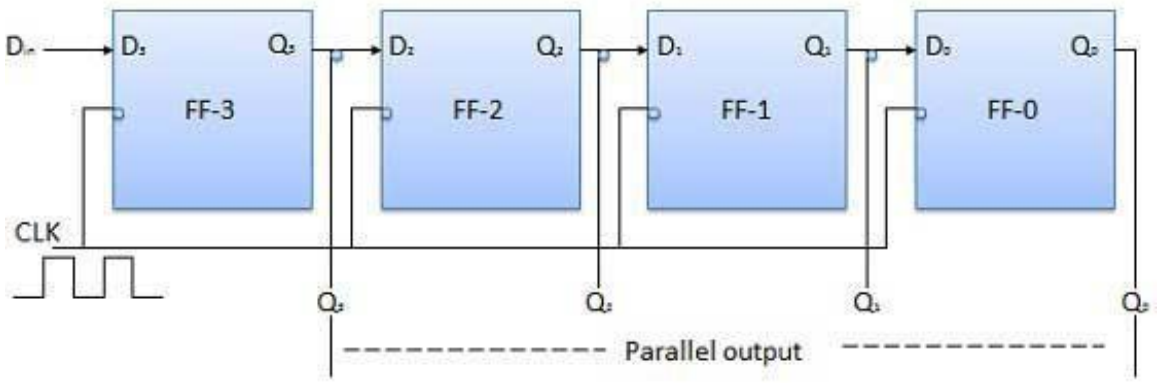
- Waveform Representation
- In the above waveform, the 1st waveform is the CLK i/p signal whereas the 2nd waveform shows the data i/p to be stored as ‘1111’. So the waveform will be a constant high signal. In addition, the above-shown waveform will represent the 4 data o/p of the FFs.
- Firstly, all the FFs o/ps were ‘0’ which is very clearly given in the above representation of the waveform. But, the ‘Q3’ output will vary from ‘0’ to ‘1’ once the 1st CLK signal arrives while the remaining o/ps are still ‘0’.
- In this way, the 2nd CLK signal ‘Q2’ will vary from ‘0’ to ‘1’. As a result, both ‘Q2’ & ‘Q3’ will show logic high within the above waveform.

- That how a SISO shift register operates. Once the 4th CLK signal arrives then all the four registers outputs will become '1'. So the storage can be performed by shifting each bit on the arrival of every CLK signal so it is called a SISO shift register.

Serial In - Parallel Out (SIPO) Shift Register

- In such types of operations, the data is entered serially and taken out in parallel fashion.
- Data is loaded bit by bit. The outputs are disabled as long as the data is loading.
- As soon as the data loading gets completed, all the flip-flops contain their required data, the outputs are enabled so that all the loaded data is made available over all the output lines at the same time.
- 4 clock cycles are required to load a four bit word. Hence the speed of operation of SIPO mode is same as that of SISO mode.

Block Diagram



Parallel In - Serial Out (PISO) Shift Register

- Data bits are entered in parallel fashion.
- The circuit shown below is a four bit parallel input serial output register.
- Output of previous Flip Flop is connected to the input of the next one via a combinational circuit.
- The binary input word B_0, B_1, B_2, B_3 is applied though the same combinational circuit.
- There are two modes in which this circuit can work namely - shift mode or load mode.

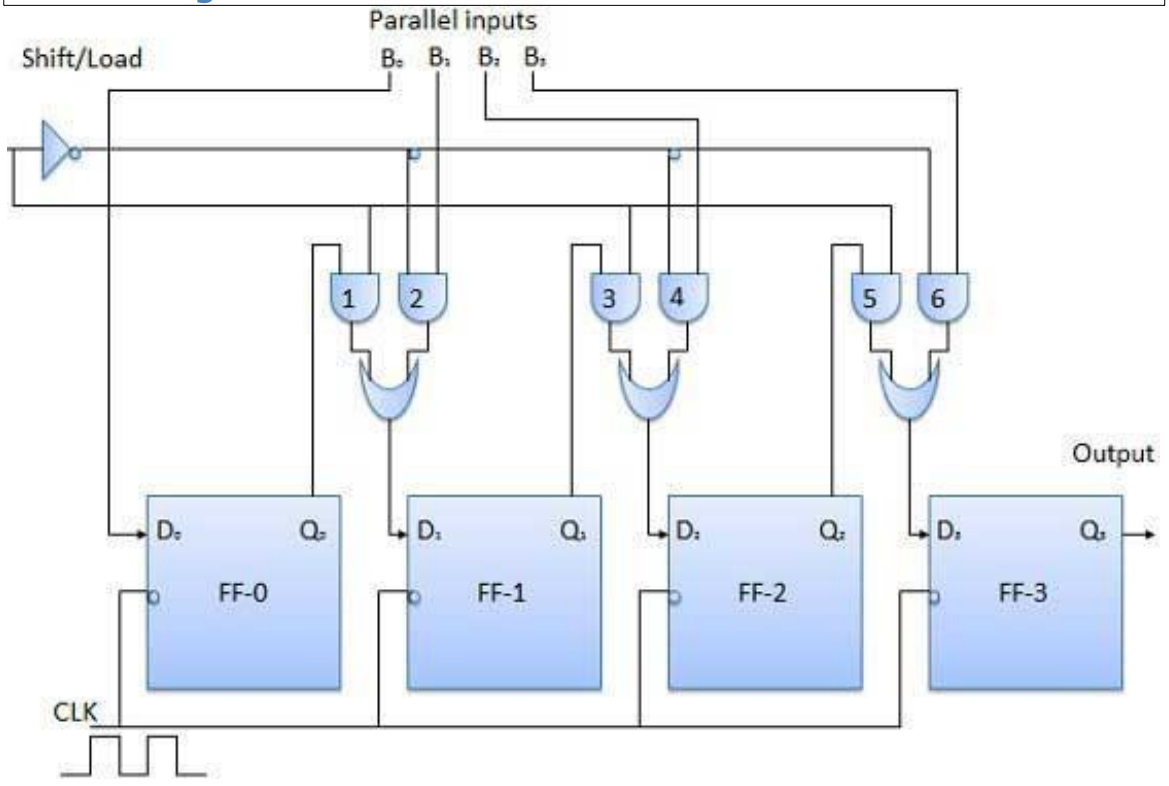
Load Mode

When the shift/load bar line is low (0), the AND gate 2, 4 and 6 become active they will pass B_1, B_2, B_3 bits to the corresponding flip-flops. On the low going edge of clock, the binary input B_0, B_1, B_2, B_3 will get loaded into the corresponding flip-flops. Thus parallel loading takes place.

Shift Mode

When the shift/load bar line is low (1), the AND gate 2, 4 and 6 become inactive. Hence the parallel loading of the data becomes impossible. But the AND gate 1,3 and 5 become active. Therefore the shifting of data from left to right bit by bit on application of clock pulses. Thus the parallel in serial out operation takes place.

Block Diagram

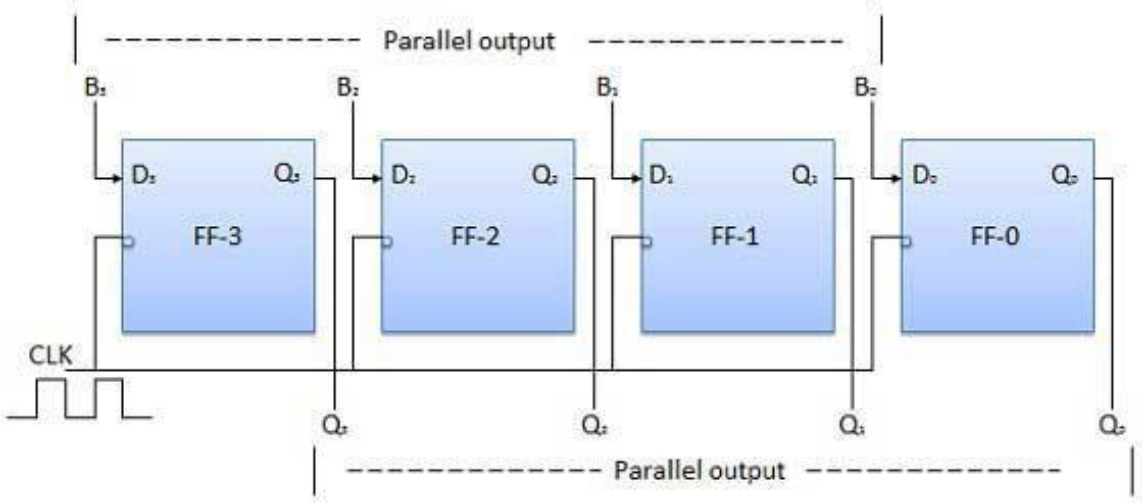


Parallel In - Parallel Out (PIPO) Shift Register

In this mode, the 4 bit binary input B_0, B_1, B_2, B_3 is applied to the data inputs D_0, D_1, D_2, D_3 respectively of the four flip-flops. As soon as a

negative clock edge is applied, the input binary bits will be loaded into the flip-flops simultaneously. The loaded bits will appear simultaneously to the output side. Only clock pulse is essential to load all the bits.

Block Diagram



INTRODUCTION TO 8085 MICROPROCESSOR

MICROCOMPUTER ARCHITECTURE:

A *microprocessor* is a programmable electronics chip that has computing and decision making capabilities similar to central processing unit of a computer. Any microprocessor-based systems having limited number of resources are called *microcomputers*. Nowadays, microprocessor can be seen in almost all types of electronics devices like mobile phones, printers, washing machines etc. Microprocessors are also used in advanced applications like radars, satellites and flights. Due to the rapid advancements in electronic industry and large scale integration of devices results in a significant cost reduction and increase application of microprocessors and their derivatives.

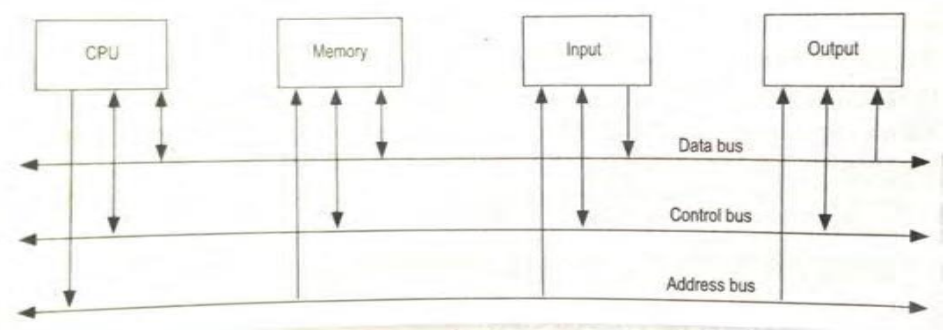


Fig.1 Microprocessor-based system

- **Bit**: A bit is a single binary digit.
- **Word**: A word refers to the basic data size or bit size that can be processed by the arithmetic and logic unit of the processor. A 16-bit binary number is called a word in a 16-bit processor.
- **Bus**: A bus is a group of wires/lines that carry similar information.
- **System Bus**: The system bus is a group of wires/lines used for communication between the microprocessor and peripherals.
- **Memory Word**: The number of bits that can be stored in a register or memory element is called a memory word.
- **Address Bus**: It carries the address, which is a unique binary pattern used to identify a memory location or an I/O port. For example, an eight bit address bus has eight lines and thus it can address 2^8 = 256 different locations. The locations in hexadecimal format can be written as 00H – FFH.
- **Data Bus**: The data bus is used to transfer data between memory and processor or between I/O device and processor. For example, an 8-bit processor will generally have an 8-bit data bus and a 16-bit processor will have 16-bit data bus.
- **Control Bus**: The control bus carry control signals, which consists of signals for selection of memory or I/O device from the given address, direction of data transfer and synchronization of data transfer in case of slow devices.

A typical microprocessor consists of arithmetic and logic unit (ALU) in association with control unit to process the instruction execution. Almost all the microprocessors are based on the principle of store-program concept. In *store-program concept*, programs or instructions are sequentially stored in the memory locations that are to be executed. To do any task using a microprocessor, it is to be programmed by the user. So the programmer must have idea about its internal resources, features and supported instructions. Each microprocessor has a set of instructions, a list which is provided by the microprocessor manufacturer. The instruction set of a microprocessor is provided in two forms: *binary machine code and mnemonics*.

Microprocessor communicates and operates in binary numbers 0 and 1. The set of instructions in the form of binary patterns is called a *machine language* and it is difficult for us to understand. Therefore, the binary patterns are given abbreviated names, called mnemonics, which forms the *assembly language*. The conversion of assembly-level language into binary machine-level language is done by using an application called *assembler*.

Technology Used:

The semiconductor manufacturing technologies used for chips are:

- Transistor-Transistor Logic (TTL)
- Emitter Coupled Logic (ECL)
- Complementary Metal-Oxide Semiconductor (CMOS)

Classification of Microprocessors:

Based on their specification, application and architecture microprocessors are classified.

Based on size of data bus:

- 4-bit microprocessor
- 8-bit microprocessor
- 16-bit microprocessor
- 32-bit microprocessor

Based on application:

- General-purpose microprocessor- used in general computer system and can be used by programmer for any application. Examples, 8085 to Intel Pentium.
- Microcontroller- microprocessor with built-in memory and ports and can be programmed for any generic control application. Example, 8051.
- Special-purpose processors- designed to handle special functions required for an application. Examples, digital signal processors and application-specific integrated circuit (ASIC) chips.

Based on architecture:

- Reduced Instruction Set Computer (RISC) processors
- Complex Instruction Set Computer (CISC) processors

8085 MICROPROCESSOR ARCHITECTURE

The salient features of 8085 µp are:

a. 1.It is a 8 bit microprocessor.
b. It is manufactured with N-MOS technology.
c. It has 16-bit address bus and hence can address up to 216 = 65536 bytes (64KB) memory locations through A_0-A_{15}.
d. The first 8 lines of address bus and 8 lines of data bus are multiplexed AD_0 – AD_7.
e. Data bus is a group of 8 lines D_0 – D_7.
f. It supports external interrupt request.
g. A 16 bit program counter (PC)
h. A 16 bit stack pointer (SP)
i. Six 8-bit general purpose register arranged in pairs: BC, DE, HL.
j. It requires a signal +5V power supply and operates at 3.2 MHZ single phase clock.
k. It is enclosed with 40 pins DIP (Dual in line package).

8085 MICROPROCESSOR ARCHITECTURE

The 8085 microprocessor is an 8-bit processor available as a 40-pin IC package and uses +5 V for power. It can run at a maximum frequency of 3 MHz. Its data bus width is 8-bit and address bus width is 16-bit, thus it can address 2^{16} = 64 KB of memory. The internal architecture of 8085 is shown is Fig. 2.

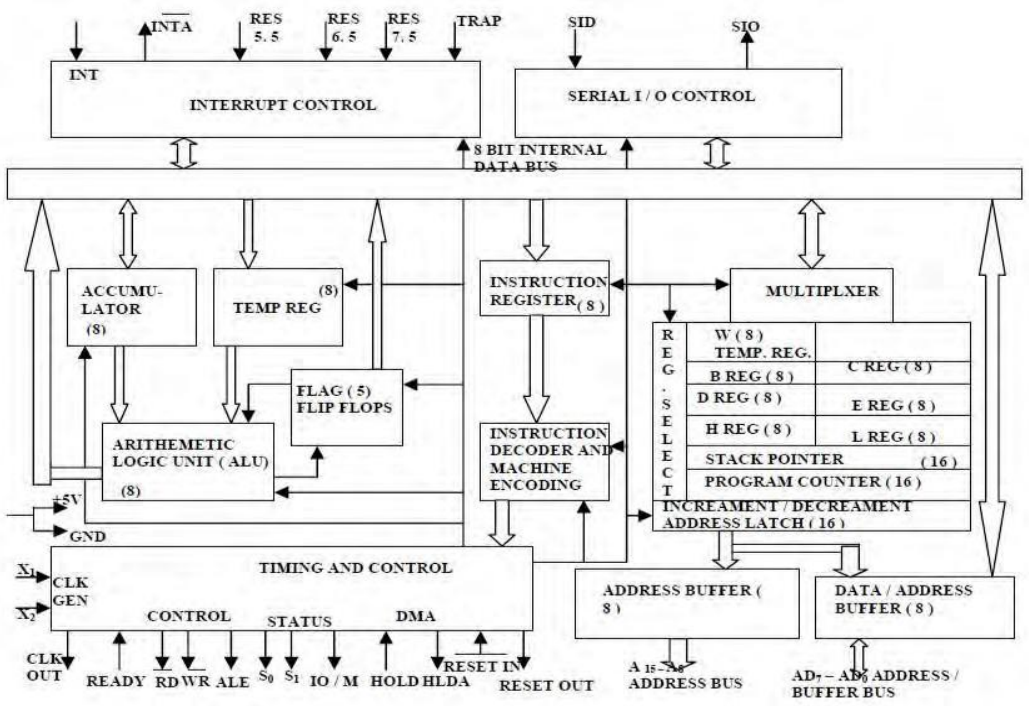


Fig. 2 Internal Architecture of 8085

Arithmetic and Logic Unit

The ALU performs the actual numerical and logical operations such as Addition (ADD), Subtraction (SUB), AND, OR etc. It uses data from memory and from Accumulator to perform operations. The results of the arithmetic and logical operations are stored in the accumulator.

Registers

The 8085 includes six registers, one accumulator and one flag register, as shown in Fig. 3. In addition, it has two 16-bit registers: stack pointer and program counter. They are briefly described as follows.

The 8085 has six general-purpose registers to store 8-bit data; these are identified as B, C, D, E, H and L. they can be combined as register pairs - BC, DE and HL to perform some 16-bit operations. The programmer can use these registers to store or copy data into the register by using data copy instructions.

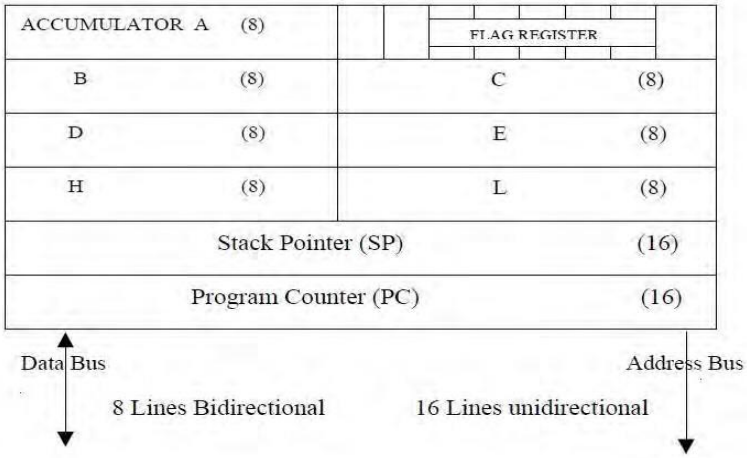


Fig. 3 Register organisation

Accumulator

The accumulator is an 8-bit register that is a part of ALU. This register is used to store 8-bit data and to perform arithmetic and logical operations. The result of an operation is stored in the accumulator. The accumulator is also identified as register A.

Flag register

The ALU includes five flip-flops, which are set or reset after an operation according to data condition of the result in the accumulator and other registers. They are called Zero (Z), Carry (CY), Sign (S), Parity (P) and Auxiliary Carry (AC) flags. Their bit positions in the flag register are shown in Fig. 4. The microprocessor uses these flags to test data conditions.

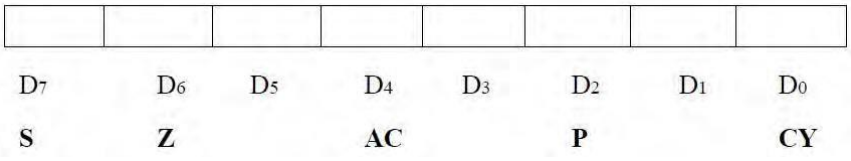


Fig. 4 Flag register

For example, after an addition of two numbers, if the result in the accumulator is larger than 8-bit, the flip-flop uses to indicate a carry by setting CY flag to 1. When an arithmetic operation results in zero, Z flag is set to 1. The S flag is just a copy of the bit D7 of the accumulator. A negative number has a 1 in bit D7 and a positive number has a 0 in 2's complement representation. The AC flag is set to 1, when a carry result from bit D3 and passes to bit D4. The P flag is set to 1, when the result in accumulator contains even number of 1s.

Program Counter (PC)

This 16-bit register deals with sequencing the execution of instructions. This register is a memory pointer. The microprocessor uses this register to sequence the execution of the instructions. The function of the program counter is to point to the memory address from which the next byte is to be fetched. When a byte is being fetched, the program counter is automatically incremented by one to point to the next memory location.

Stack Pointer (SP)

The stack pointer is also a 16-bit register, used as a memory pointer. It points to a memory location in R/W memory, called stack. The beginning of the stack is defined by loading 16-bit address in the stack pointer.

Instruction Register/Decoder

It is an 8-bit register that temporarily stores the current instruction of a 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.

Control Unit

Generates signals on data bus, address bus and control bus within microprocessor to carry out the instruction, which has been decoded. Typical buses and their timing are described as follows:

- *Data Bus*: Data bus carries data in binary form between microprocessor and other external units such as memory. It is used to transmit data i.e. information, results of arithmetic etc between memory and the microprocessor. Data bus is bidirectional in nature. The data bus width of 8085 microprocessor is 8-bit i.e. 2^8 combination of binary digits and are typically identified as D0 – D7. Thus size of the data bus determines what arithmetic can be done. If only 8-bit wide then largest number is 11111111 (255 in decimal). Therefore, larger numbers have to be broken down into chunks of 255. This slows microprocessor.
- *Address Bus*: The address bus carries addresses and is one way bus from microprocessor to the memory or other devices. 8085 microprocessor contain 16-bit address bus and are generally identified as A0 - A15. The higher order address lines (A8 – A15) are unidirectional and the lower order lines (A0 – A7) are multiplexed (time-shared) with the eight data bits (D0 – D7) and hence, they are bidirectional.
- *Control Bus*: Control bus are various lines which have specific functions for coordinating and controlling microprocessor operations. The control bus carries control signals partly unidirectional and partly bidirectional. The following control and status signals are used by 8085 processor:
  I. ALE (output): Address Latch Enable is a pulse that is provided when an address appears on the AD0 – AD7 lines, after which it becomes 0.
  II. $\overline{RD}$ (active low output): The Read signal indicates that data are being read from the selected I/O or memory device and that they are available on the data bus.
  III. $\overline{WR}$ (active low output): The Write signal indicates that data on the data bus are to be written into a selected memory or I/O location.
  IV. IO/$\overline{M}$ (output): It is a signal that distinguished between a memory operation and an I/O operation. When IO/$\overline{M}$ = 0 it is a memory operation and IO/$\overline{M}$ = 1 it is an I/O operation.
  V. S1 and S0 (output): These are status signals used to specify the type of operation being performed; they are listed in Table 1.

Table 1 Status signals and associated operations

S1	S0	States
0	0	Halt
0	1	Write
1	0	Read
1	1	Fetch

The schematic representation of the 8085 bus structure is as shown in Fig. 5. The microprocessor performs primarily four operations:

I. Memory Read: Reads data (or instruction) from memory.
II. Memory Write: Writes data (or instruction) into memory.
III. I/O Read: Accepts data from input device.
IV. I/O Write: Sends data to output device.

The 8085 processor performs these functions using address bus, data bus and control bus as shown in Fig. 5.

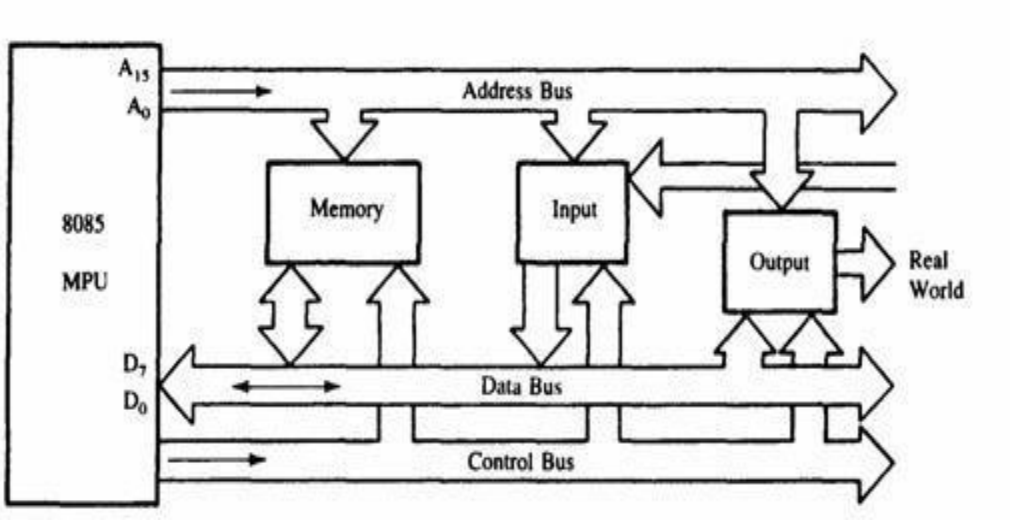


Fig. 5 The 8085 bus structure

3. 8085 PIN DESCRIPTION

Properties:

- It is a 8-bit microprocessor
- Manufactured with N-MOS technology
- 40 pin IC package
- It has 16-bit address bus and thus has 2^{16} = 64 KB addressing capability.
- Operate with 3 MHz single-phase clock
- +5 V single power supply

The logic pin layout and signal groups of the 8085nmicroprocessor are shown in Fig. 6. All the signals are classified into six groups:

- Address bus
- Data bus
- Control & status signals

- Power supply and frequency signals
- Externally initiated signals
- Serial I/O signals

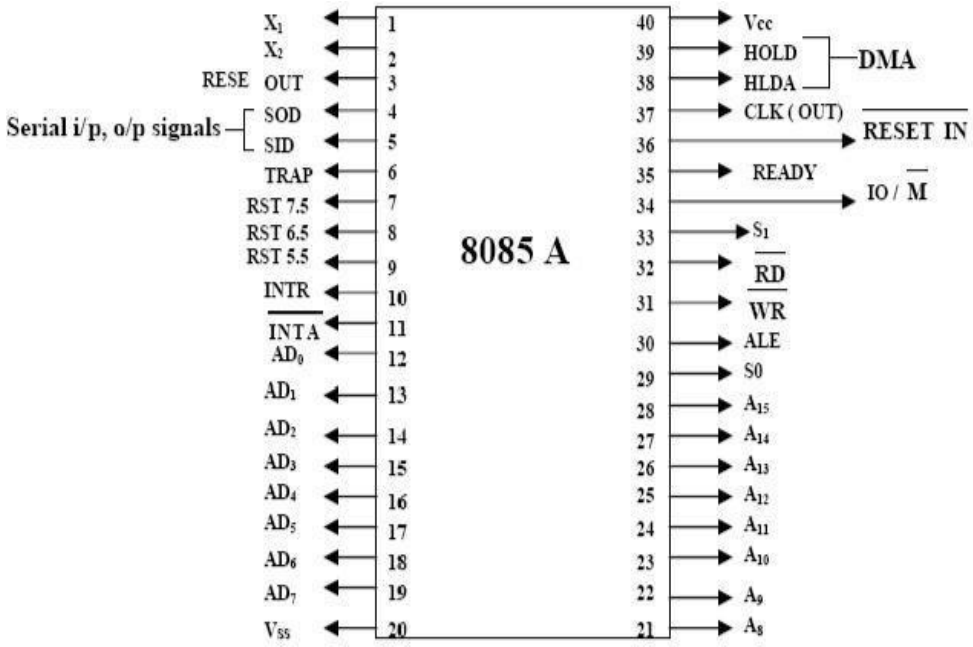


Fig. 6 8085 microprocessor pin layout and signal groups

Address and Data Buses:

- A8 – A15 (output, 3-state): Most significant eight bits of memory addresses and the eight bits of the I/O addresses. These lines enter into tri-state high impedance state during HOLD and HALT modes.
- AD0 – AD7 (input/output, 3-state): Lower significant bits of memory addresses and the eight bits of the I/O addresses during first clock cycle. Behaves as data bus during third and fourth clock cycle. These lines enter into tri-state high impedance state during HOLD and HALT modes.

Control & Status Signals:

- ALE: Address latch enable
- $\overline{RD}$: Read control signal.
- $\overline{WR}$: Write control signal.
- IO/$\overline{M}$, S1 and S0 : Status signals.

Power Supply & Clock Frequency:

- Vcc: +5 V power supply
- Vss: Ground reference
- X1, X2: A crystal having frequency of 6 MHz is connected at these two pins
- CLK: Clock output

Externally Initiated and Interrupt Signals:

- $\overline{\text{RESET IN}}$: When the signal on this pin is low, the PC is set to 0, the buses are tri-stated and the processor is reset.
- RESET OUT: This signal indicates that the processor is being reset. The signal can be used to reset other devices.
- READY: When this signal is low, the processor waits for an integral number of clock cycles until it goes high.
- HOLD: This signal indicates that a peripheral like DMA (direct memory access) controller is requesting the use of address and data bus.
- HLDA: This signal acknowledges the HOLD request.
- INTR: Interrupt request is a general-purpose interrupt.
- $\overline{\text{INTA}}$: This is used to acknowledge an interrupt.
- RST 7.5, RST 6.5, RST 5,5 – restart interrupt: These are vectored interrupts and have highest priority than INTR interrupt.
- TRAP: This is a non-maskable interrupt and has the highest priority.

Serial I/O Signals:

- SID: Serial input signal. Bit on this line is loaded to D7 bit of register A using RIM instruction.
- SOD: Serial output signal. Output SOD is set or reset by using SIM instruction.

4. INSTRUCTION SET AND EXECUTION IN 8085

Based on the design of the ALU and decoding unit, the microprocessor manufacturer provides instruction set for every microprocessor. The instruction set consists of both machine code and mnemonics.

An instruction is a binary pattern designed inside a microprocessor to perform a specific function. The entire group of instructions that a microprocessor supports is called instruction set. Microprocessor instructions can be classified based on the parameters such functionality, length and operand addressing.

Classification based on functionality:

I. Data transfer operations: This group of instructions copies data from source to destination. The content of the source is not altered.
II. Arithmetic operations: Instructions of this group perform operations like addition, subtraction, increment & decrement. One of the data used in arithmetic operation is stored in accumulator and the result is also stored in accumulator.
III. Logical operations: Logical operations include AND, OR, EXOR, NOT. The operations like AND, OR and EXOR uses two operands, one is stored in accumulator and other can be any register or memory location. The result is stored in accumulator. NOT operation requires single operand, which is stored in accumulator.
IV. Branching operations: Instructions in this group can be used to transfer program sequence from one memory location to another either conditionally or unconditionally.
V. Machine control operations: Instruction in this group control execution of other instructions and control operations like interrupt, halt etc.

Classification based on length:

I. One-byte instructions: Instruction having one byte in machine code. Examples MOV A,B.
I. Two-byte instructions: Instruction having two byte in machine code. Examples MVI A,08H
II. Three-byte instructions: Instruction having three byte in machine code. Examples LDA2004H

Addressing Modes in Instructions:

The process of specifying the data to be operated on by the instruction is called addressing. The various formats for specifying operands are called addressing modes. The 8085 has the following five types of addressing:

I. Immediate addressing
II. Memory direct addressing
III. Register direct addressing
IV. Indirect addressing
V. Implicit addressing

1. Immediate Addressing:

In this mode, the operand given in the instruction - a byte or word – transfers to the destination register or memory location.

Ex: MVI A, 9AH

- The operand is a part of the instruction.
- The operand is stored in the register mentioned in the instruction.

2. Memory Direct Addressing:

Memory direct addressing moves a byte or word between a memory location and register. The memory location address is given in the instruction.

Ex: LDA 850FH

This instruction is used to load the content of memory address 850FH in the accumulator.

3. Register Direct Addressing:

Register direct addressing transfer a copy of a byte or word from source register to destination register.

Ex: MOV B, C

It copies the content of register C to register B.

4. Indirect Addressing:

Indirect addressing transfers a byte or word between a register and a memory location.

Ex: MOV A, M

Here the data is in the memory location pointed to by the contents of HL pair. The data is moved to the accumulator.

5. Implicit Addressing

In this addressing mode the data itself specifies the data to be operated upon.

Ex: CMA

The instruction complements the content of the accumulator. No specific data or operand is mentioned in the instruction.

5. INSTRUCTION SET OF 8085

Data Transfer Instructions:

Opcode	Operand	Description
Copy from source to destination		
MOV	Rd, Rs M, Rs Rd, M	This instruction copies the contents of the source register into the destination register; the contents of the source register are not altered. If one of the operands is a memory location, its location is specified by the contents of the HL registers Example: MOV B, C or MOV B, M
Move immediate 8-bit		
MVI	Rd, data M, data	The 8-bit data is stored in the destination register or memory. If the operand is a memory location, its location is specified by the contents of the HL registers Example: MVI B, 57 or MVI M, 57
Load accumulator		
LDA	16-bit address	The contents of a memory location, specified by a 16-bit address in the operand, are copied to the accumulator. The contents of the source are not altered Example: LDA 2034 or LDA XYZ
Load accumulator indirect		
LDAX	B/D Reg. pair	The contents of the designated register pair point to a memory location. This instruction copies the contents of that memory location into the accumulator. The contents of either the register pair or the memory location are not altered. Example: LDAX B
Load register pair immediate		
LXI	Reg. pair, 16-bit data	The instruction loads 16-bit data in the register pair designated in the operand Example: LXI H, 2034
Load H and L registers direct		
LHLD	16-bit address	The instruction copies the contents of the memory location pointed out by the 16-bit address into register L and copies the contents of the next memory location into register H. The contents of source memory locations are not altered. Example: LHLD 2040