ELECTRONICS

(DPHYL21)

(MSC - PHYSICS)

ACHARYA NAGARJUNA UNIVERSITY

CENTRE FOR DISTANCE EDUCATION

NAGARJUNA NAGAR,

GUNTUR

ANDHRA PRADESH

CENTRE FOR DISTANCE EDUCATION
ACHARYA NAGARJUNA UNIVERSITY :: NAGARJUNANAGAR – 522 510

M.Sc Physics (Second year)

Paper XI : Laboratory practical III: Electronics

1. Digital counters
2. Digital shift Registers
3. FET amplifier frequency response curve.
4.Astable multivibrator using 555 timer.
5 Op.Amp as inverting , non-inverting amplifier and, summing amplifier..
6. Wein bridge oscillator (op.Amp)
7. Colpitts Oscillator(Op.Amp)
8. Active filters (Op.Amp)
9.Assembly language programming of 8085 I
10.Assembly language pragramming of 8085 II
11.Assembly language programming of 8085 III
12.Assembly language pragramming of 8085 IV

CENTRE FOR DISTANCE EDUCATION
ACHARYA NAGARJUNA UNIVERSITY :: NAGARJUNANAGAR – 522 510

M.Sc Physics (Second year)

Paper XII : Laboratory practical IV: Solid State Physics

1.Dielectric constant of a ferroelectric material-curie temperature.
2.Specific heat of graphite variation with temperature.
3.Thermal expansion in the crystal –optical method.
4.Lattice dynamics-mono atomic and diatomic lattice.
5.Hall coefficient in semiconductors and density of charge carriers.
6.Fiber optics characteristics: Measurement of Numerical Aperture of mono/multi – step/grade index optical fiber
7. B – H curve.
8.Magnetic susceptibility of a solution of paramagnetic salt-concentration variation (Quinckes method).
9.Band gap of a semiconductor-four probe method.
10.X-ray powder diffraction pattern-determination of lattice constants.
11.Electron spin resonance-determination of g factor for DPPH.
12. Dielectric constant of a (solid or) liquid at microwave frequency (3cm)

Contents

PRACTICAL - III

Page Nos.

PRACTICAL - IV

Experiment No. 1

DIGITAL COUNTERS

Aim : To construct and study the working of (i) binary ripple counters, (ii) decade ripple counter

Apparatus: ICs:7476-three,7400-one,7405-one,7410-one;(2)IC power supply (5V ,1A Stabilized);(3) Pulser switch unit;(4)Lamp monitoring unit;(5)Clock generator;(6)Logical level switch

Counters:

A digital counter is a set of flip-flops (FFs) whose states change in response to pulses applied at the input to the counter. Thus, as its name implies, a counter is used to count pulses. A counter can also be used as a frequency divider to obtain waveforms with frequencies that are specific functions of the clock frequency. They are also used to perform the timing function as in digital watches, to create time delays, to produce non-sequential binary counts, to generate pulse trains, and to act as frequency counters, etc.

Counters may be asynchronous counters or synchronous counters. The term asynchronous refers to events that do not occur at the same time. Asynchronous counters are also called Ripple counters. The asynchronous counter has a disadvantage, in so far as the unwanted spikes are concerned. This limitation is overcome in parallel counters. Propagation delay is a major disadvantage in asynchronous counters because it limits the rate at which the counter can be clocked and creates decoding problem.

The term synchronous as applied to counter operation means that the counter is clocked such that each flip-flop in the counter is triggered at the same time.

Asynchronous counters:

In a asynchronous counter, the flip-flop output transition serves as a source of triggering other flip-flops. In other words, the CP inputs of all flip-flops are triggered not by the incoming pulses but rather by the transition that that occurs in other flip-flops. In this section, we present some asynchronous counters and explain their operation.

Four – Bit Asynchronous Binary Counter:

Four – Bit asynchronous binary counter consists of a series connection of complementing flip-flops, with the output of each flip-flop connected to the CP input of the next higher-order flip-flop. The flip-flop lading the least significant bit receives the incoming count pulses. The diagram of a 4 – bit binary ripple counter is shown in fig 1. All J and K inputs are equal to '1'. The small circle in the CP input indicates that the flip flop complements during a negative-going transition or when the output to which it is connected goes from 1 to 0.

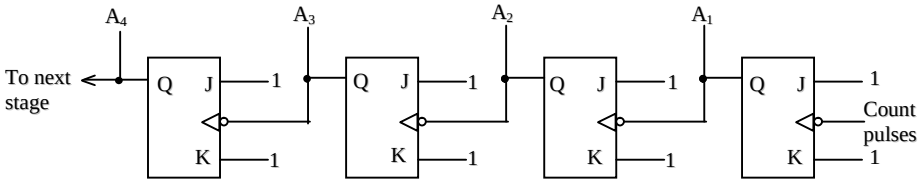


Fig 1 – bit binary ripple counter

Table 1 Count sequence for a binary ripple counter

Count sequence				Conditions for complementing flip-flops	
A_4	A_3	A_2	A_1		
0	0	0	0	Complement A_1	
0	0	0	1	Complement A_1	A_1 will go from 1 to 0 and Complement A_2
0	0	1	0	Complement A_1	
0	0	1	1	Complement A_1	A_1 will go from 1 to 0 and Complement A_2
					A_2 will go from 1 to 0 and Complement A_3
0	1	0	0	Complement A_1	
0	1	0	1	Complement A_1	A_1 will go from 1 to 0 and Complement A_2
0	1	1	0	Complement A_1	
0	1	1	1	Complement A_1	A_1 will go from 1 to 0 and Complement A_2
					A_2 will go from 1 to 0 and Complement A_3
1	0	0	0	and so on …	A_3 will go from 1 to 0 and Complement A_4

To understand the operation of the binary counter, refer to its count sequence given in Table 1. It is obvious that the lowest order bit A_1 must be complemented with each count pulse. Every time A_1 goes from 1 to 0, it complements A_2. Every time A_2 goes from 1 to 0, it complements A_3, and so on. For example, take the transition from count 0111 to 1000. The arrows in the table emphasize the transition in this case. A_1 is complemented with the count pulse. Since A_1 goes from 1 to 0, it triggers A_2 and complements it. As a result, A_2 goes from 1 to 0, which in turn complements A_3. A_3 now goes from 1 to 0, which complements A_4. The output transition of A_4, if connected to a next stage, will not trigger the next flip-flop, since it goes from 0 to 1. The flip-flops change are at a time in rapid success on, and the signal propagates through the counter in a ripple fashion. Hence asynchronous counters are sometimes called asynchronous counters.

The 7493A Four – Bit binary counter:-

The 7493A is presented as an example of a specific integrated circuit asynchronous counter. As the logic diagram in fig 2 shows, this device actually consists of a single flip-flop and a three – bit asynchronous counter. This arrangement is for flexibility. It can be used as a divide – by-2 device using only the single flip-flop, or it can be used as a modulus – 8 counter using only the three-bit counter position. This device also provides gated reset inputs, RO(1) and RO(2). When both of these inputs are HIGH, the counter is RESET to the 0000 state by $\overline{\text{CLR}}$.

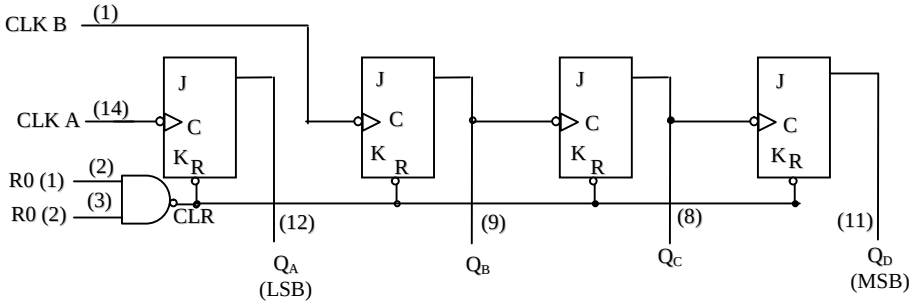
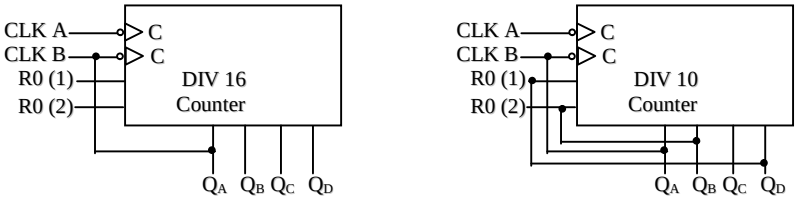


Fig 2 The 7493A four-bit binary counter logic diagram. (Pin numbers are in parentheses, and all J-K inputs are internally connected HIGH.

Additionally, the 7493A can be used as a four-bit modules-16 counter (counts 0 through 15) by connecting Q_A output to the CLK B input as shown in fig 2.



(a) 7493A connected as a modulus-16 counter

(a) 7493A connected as a decade counter

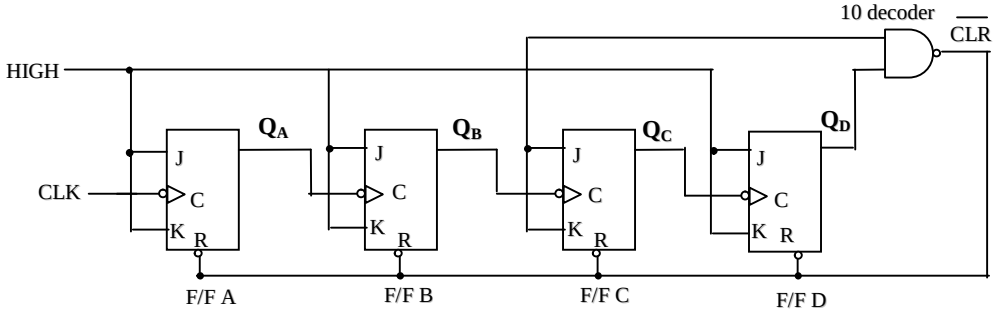
Fig 3 Two configurations of the 7493A asynchronous counter.

It can also be configured as a decade counter with asynchronous recycling by using the gated reset inputs for partial decoding of count 10_{10}, as shown in fig 3.

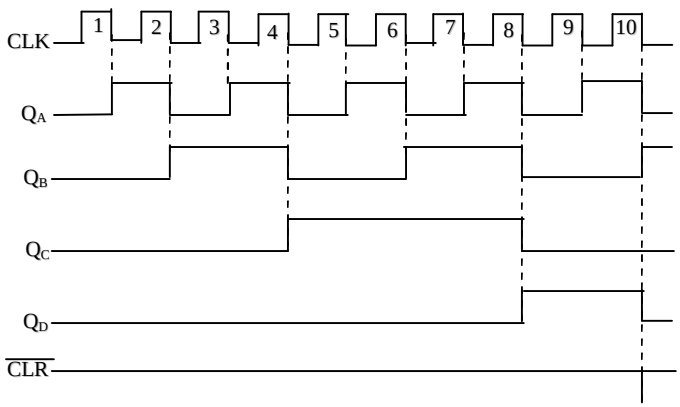
Asynchronous Decade Counters:

Counters with ten states in their sequence are called decade counter. A decade counter with a count sequence of 0(0000) through 9(1001) is a BCD decade counter because its ten-state sequence is the BCD code. This type of counter is very useful in display applications in which BCD is required for conversion to a decimal readout.

A decade counter requires four flip-flops. We will now take a four-bit asynchronous counter and modify its sequence in order to understand the principle of truncated counters. One method of achieving this recycling after the count of 9(1001) is to decode count 10_{10}(1010) with a NAND gate and connect the output of the NAND gate to the clear (CLR) inputs of the flip-flops, as shown in fig 4.



(a)



(b)

Fig 4 An asynchronously clocked decade counter with asynchronous recycling.

Notice that only Q_B and Q_D are connected to the NAND gate inputs. This is an example of partial decoding, in which the two unique states (Q_B = 1and Q_D = 0) are sufficient to decode the count of 10_{10} because none of the other states (0 through 9) have both Q_B and Q_D HIGH at the same time. When the counter goes into count 10_{10}(1010), the decoding gate output goes LOW and asynchronously RESETS all of the flip-flops.

The resulting timing diagrams is shown in fig.4 Notice that there is a glitch on the Q_B wave form. The reason for this glitch is that Q_B must first go HIGH before the count of 10_{10} can be decoded. Not until several nano seconds after the counter goes to the count of 10_{10} does the output of the decoding gate go LOW. Thus, the counter is in the 1010 state for a short time before it is RESET back to 0000, thus producing the glitch on Q_B.

Synchronous Counters:

Synchronous counters are distinguished from ripple counters in that clock pulses are applied to the CP inputs of all flip-flops. The common pulse triggers all the filp-flops simultaneously, rather than one at a time in succession as in a ripple counter. The decision whether a flip-flop is to be complemented or not is determined from the values of the J and K inputs at the time of the pulse. If J = K = 0, the flip-flop remains unchanged. If J = K = 1, the flip-flop complements. In this section, we present some typical MSI synchronous counters and explain their operation.

Three – Bit Synchronous Binary counter:

Three bit synchronous binary counter is shown in fig 5 and its timing diagram in (fig 6). An understanding of this counter can be achieved by a careful examination of its sequence of states as shown in Table 2.

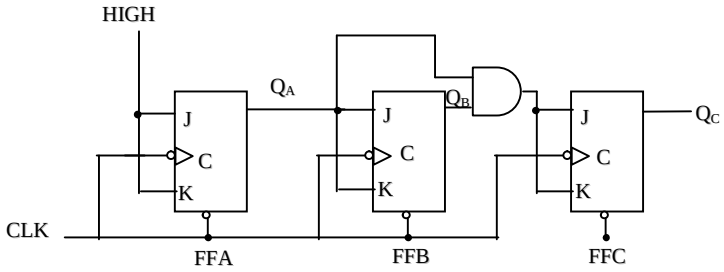


Fig 5 A three – bit synchronous binary counter.

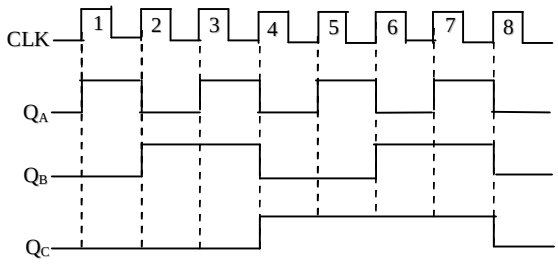


Fig 6) Timing diagram for the counter of figure.

First , let us look at Q_A. Notice that Q_A changes on each clock pulse as we progress from its original state to its final state and then back to its original state. To produce this operation, FFA must be held in the toggle mode by constant HIGH on its J and K inputs. Now let us see what Q_B does. Notice that it goes to the opposite state following each time Q_A is a 1. This occurs at CLK_2, CLK_4, CLK_6 and CLK_8. CLK_8 causes the counter to recycle. To produce this operation, Q_A is connected to the J and K inputs of FFB. When Q_A is a 1 and a clock pulse occurs. FFB is the toggle mode and will change state. The other times when Q_A is a 0, FFB is in the no-change mode and remains in its present state.

Table 2 State sequence for a three-stage binary counter.

Clock Pulse	Q_C	Q_B	Q_A
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

Next, let us see how FFC is made to change at the proper times according to the binary sequence. Notice that both times Q_C changes state, it is preceded by the unique condition of both Q_A and Q_B being HIGH. This condition is detected by the AND gate and applied to the J and K inputs of FFC. Whenever both Q_A and Q_B being HIGH, the output of the AND gate makes the J and K inputs of FFC HIGH, and FFC toggle on the following clock pulse. At all other times, the J and K inputs of FFC are held LOW by the AND gate output, and FFC does not change state.

Experimental Procedure:

a) Asynchronous Binary up Counter:

Wire the four-bit ripple counter as shown in fig.1 using two IC7476s. Monitor the logic levels at D,C,B,A using the lamp monitoring units. Feed the counter with the pulser switch. Tie up all clear inputs and connect to pulser switch. Clear the FFs to read O by momentarily pressing the pulser switch. Apply sixteen pulses one by one and observe the D,C,B,A outputs and fill in the Table3.

Table 3 Asynchronous Binary Up Counter

INPUT	D	C	B	A	
0	0	0	0	0	
1					
2					
-					
-					
-					
-					
-					
-					
-					
-					
-					
-					
-					
-					
16	0	0	0	0	0

Apply 1kHz square wave from the clock generator to the input and observe the output waves at D,C,B,A using a dual trace CRO

b) Asynchronous Binary down Counter:

1.wire the circuit using two IC 7476s

2.Clear the counter . the D,C,B,A outputs show logic level 1.

3.Enter sixteen clock pulses using PSI and observe the outputs after each input pulse. Enter your results in a table similar to table 7 .

4. Observe the relation between the input and output waveforms using a dual trace oscilloscope . Feed the counter with 1kHz signal from the clock generator.

c) Decade Ripple Counter:

Wire the decade ripple counter circuit as shown in fig 4 using two IC7476s and a NAND gate (IC7400). Feed the counter units with the pulser switch(PSI) and monitor the outputs using lamp monitors units. Enter ten pulses one by one and note the output levels after each input pulse. Record your observations in a tabular form.

d)Modulo –3 Counter:

Two FFs are required to construct a modulo-3 counter as shown in fig wire the circuit using IC7476.Monitor B and A outputs using the lamp monitoring unit. Feed the counter with the pulser switch. Verify the count sequence .

(e)Modulo –8 synchronous binary counter:

Wire the circuit shown as in fig 5 using two IC7476s and one IC7410 . Feed the input with positive pulses and monitor C,B,A outputs with the help of the lamp monitoring unit.

Precautions:

1. Verify the proper functioning of individual logic components used in the circuit before using them in the circuit

Results : The working the logic circuit for the decade counter is is as per the truth table.

Experiment No. 2

SHIFT REGISTERS

Aim: To study the working of four bit shift register

Apparatus : Two I C 7476 , One I C 7405 Or 7400 , Power supply , Bread board

Shift Register:

Shift registers are very important in applications involving the storage and transfer of data in digital system. A register, in general, is used solely for storing and shifting data (1s and 0s) entered into it from all external sources and possesses no characteristic internal sequence of states. The storage capability of a register is one of its two basic functional characteristics and makes it an important type of memory device.

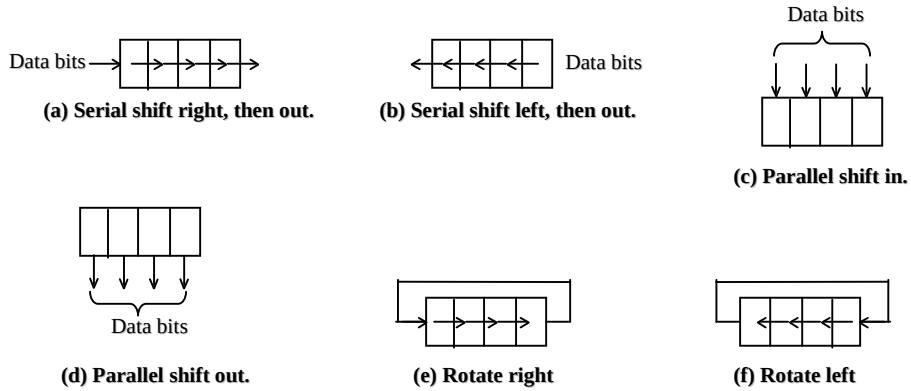


Figure 1 : Basic data movement in registers.

Registers are commonly used for the temporary storage of data within a digital system. The shift capability of a register permits the movement of data from stage to stage within the register or into or out of the register upon application of clock pulses.

Fig 1 shows symbolically the types of data movement in shift register operations. The block represents any arbitrary four-bit register, and the arrow indicates the direction and type of data movement.

Serial In – Serial out Shift Register:

This type of shift register accepts data serially, i.e. one bit at a time, and also outputs data serially. The logic diagram of a 4 – bit serial – in, serial – out, shift register is shown in fig 2. with four stages, i.e., four flip –flops, the register can store up to four bits of data.

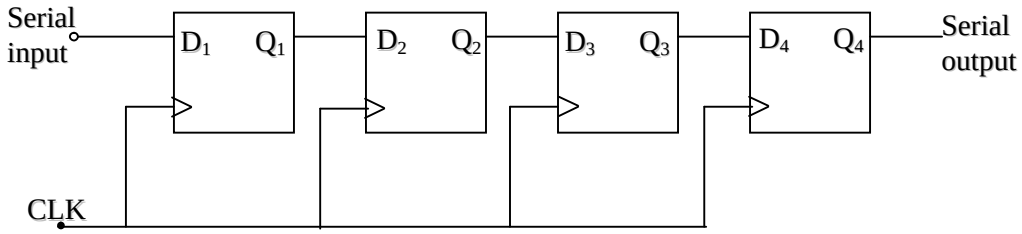


Figure 2 – bit serial – in, serial – out, shift register

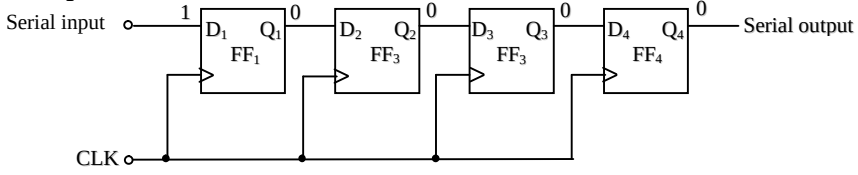
Serial data is applied at the D input of the first Flip-Flop (FF). The Q output of the first FF is connected to the D input of the second FF, the Q output of the second FF is connected to the D input of the third FF and the Q output of the third FF is connected to the D input of the fourth FF. The data is outputted from the Q terminal of the last FF.

When serial data is transferred into a register, each new bit is clocked into the first FF at the positive – going edge of each clock pulse. The bit that was previously stored by the first FF is transferred to the second FF. The bit that was stored by the second FF is transferred to the third FF, and so on. The bit that was stored by the last FF is shifted out.

Fig 3 and table 1 illustrate this process to store the data bits 0101 in the register. Initially all the FFs are reset, i.e., $Q_1 = 0$, $Q_2 = 0$, $Q_3 = 0$ and $Q_4 = 0$.

The right most bit '1' is applied at the D_1 input of FF_1. At the positive – going edge of the first clock pulse, this '1' is shifted into FF_1 and all other FFs store their respective bits at the D inputs. Therefore, $Q_1 = 1$, $Q_2 = 0$, $Q_3 = 0$ and $Q_4 = 0$, after the first clock pulse.

Then a '0' is applied at the D_1 input of FF_1. At the positive – going edge of the second clock pulse, this '0' is shifted to Q_1 of FF_1 and the D inputs of all other FFs are also shifted to their respective outputs. Therefore $Q_1 = 0$, $Q_2 = 1$, $Q_3 = 0$ and $Q_4 = 0$, after the second clock pulse.



(a) Initial states (0000)

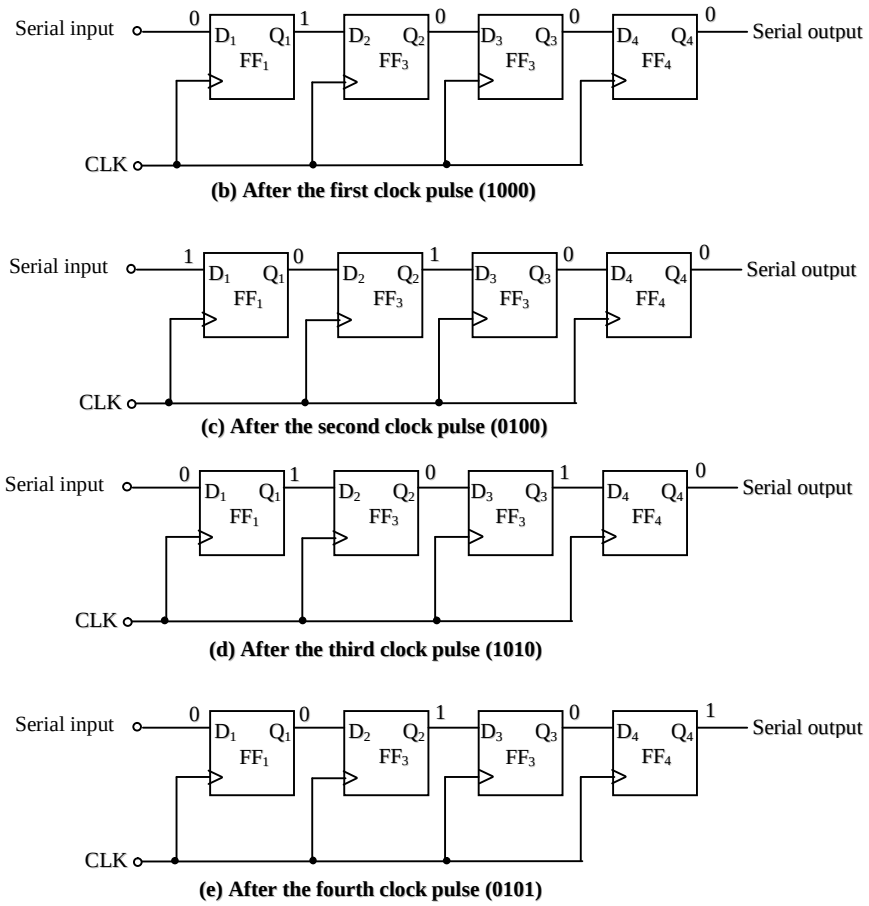


Fig3 Loading of the 4 – bit serial – in, serial – out, shift register

Then a '1' is applied at the D_1 input of FF_1. At the positive – going edge of the third clock pulse, this '1' is shifted into Q_1 of FF_1 and the D inputs of all other FFs are also shifted to their respective outputs. Therefore, $Q_1 = 1$, $Q_2 = 0$, $Q_3 = 1$ and $Q_4 = 0$, after the third clock pulse.

Then a '0' is applied at the D_1 input of FF_1. At the positive – going edge of the fourth clock pulse, this '0' is shifted into Q_1 of FF_1 and the D inputs of all other FFs are also shifted to their respective outputs. Therefore, $Q_1 = 0$, $Q_2 = 1$, $Q_3 = 0$ and $Q_4 = 1$, after the third clock pulse, this '0' is shifted to Q_1 of FF_1 and the D inputs of all other FF_S are also shifted to their respective outputs. Therefore,

After clock pulse	Serial input	Q_1	Q_2	Q_3	Q_4	
0	1	0	0	0	0	Initial states
1	0	1	0	0	0	
2	1	0	1	0	0	
3	0	1	0	1	0	
4	–	0	1	0	1	

Table 1 Shifting in the data 0101 serially.

$Q_1 = 0$, $Q_2 = 1$, $Q_3 = 0$ and $Q_4 = 1$, after the fourth clock pulse.

This completes the serial entry of 0101 into the 4-bit register fig 4 shows the timing diagram of the loading of serial input 0101 into the 4-bit serial-in, serial-out, shift register.

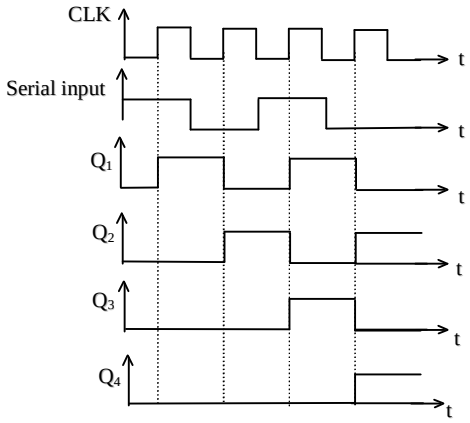


Fig 4 Timing diagram showing the loading of the serial input 0101 into the 4 – bit Serial-in, Serial-out, shift register.

The shifting out of the stored data 0101 serially from the register is illustrated in table 2 . It requires four clock pulses to shift out the 4-bit stored data.

After clock pulse	Serial input	Q_1	Q_2	Q_3	Q_4
0	0	0	1	0	1
1	0	0	0	1	0
2	0	0	0	0	1
3	0	0	0	0	0
4	–	0	0	0	0

Table 2 Shifting in the data 0101 serially.

Serial In, Parallel – out Shift Register:

In this type of register, the data bits are entered into the register serially, but the data stored in the register is shifted out in parallel form. Fig 5 shows the logic diagram and the logic symbol of a 4-bit serial – in, parallel – out shift register.

Once the data bits are stored, each bit appears on its respective output line and all bits are available simultaneously, rather than on a bit-by-bit basis as with the serial output. The serial – in, parallel-out, shift register can be used as a serial-in, serial-out, shift register, if the output is taken from the Q terminal of the last FF.

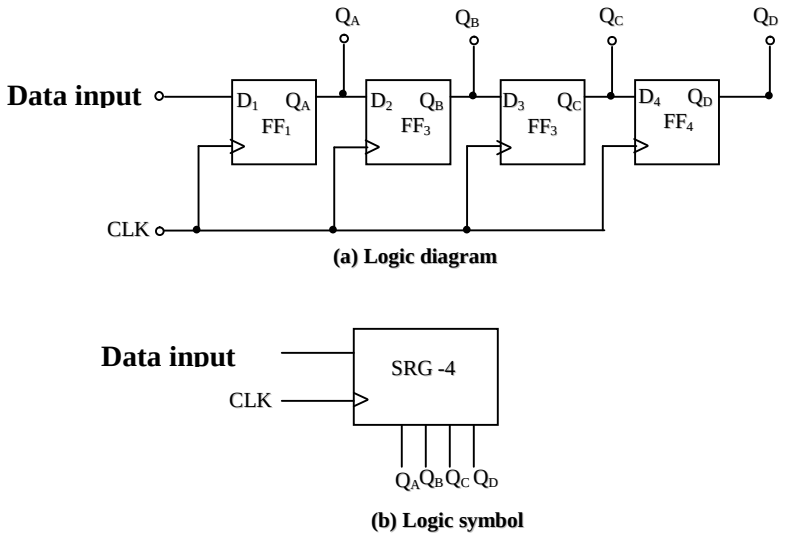


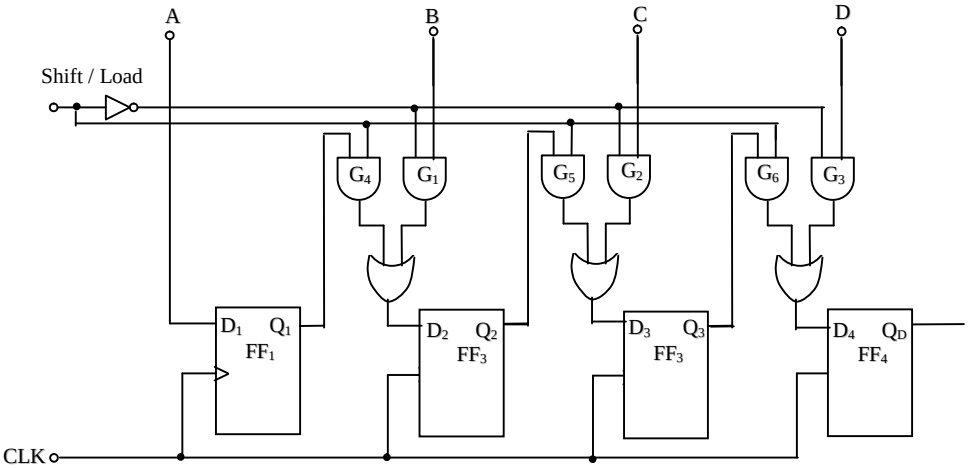
Fig (5) 4 – bit serial - in, parallel - out, shift register.

Parallel In, Serial – Out Shift Register:

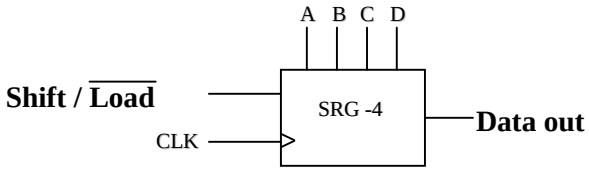
In parallel – in, serial-out, shift register, the data bits are entered simultaneously into their respective stages on parallel lines, rather than on a bit-by-bit basis on one line as with serial data inputs, but the data bits are transferred out of the register serially, i.e., on a bit-by-bit basis over a single line.

Fig 6 illustrates a 4-bit parallel-in, serial out, shift register using D flip-flops. There are four data lines A, B, C and D through which the data is entered into the register in parallel form. The signal shift / load allows (a) the data to be entered in parallel form into the register and (b) the data to be shifted out serially from terminal Q_4.

When Shift / Load line is high, gates G_1, G_2 and G_3 are disabled, but gates G_4, G_5 and G_6 are enabled allowing the data bits to shift – right from one stage to the next. When Shift / Load line is low, gates G_4, G_5 and G_6 are disabled, whereas gates G_1, G_2 and G_3 are enabled allowing the data input to appear at the D inputs of the respective flip-flops. When a clock pulse is applied, these data bits are shifted to the Q output terminals of the flip-flops and therefore, data is inputted in one step. The OR gate allows either the normal shifting operation or the parallel data entry depending on which AND gates are enabled by the level on the Shift / Load input.



(a) Logic diagram



(b) Logic symbol

Fig 6 A 4-bit parallel, serial – out, shift register.

Parallel In, Parallel – out Shift Register:

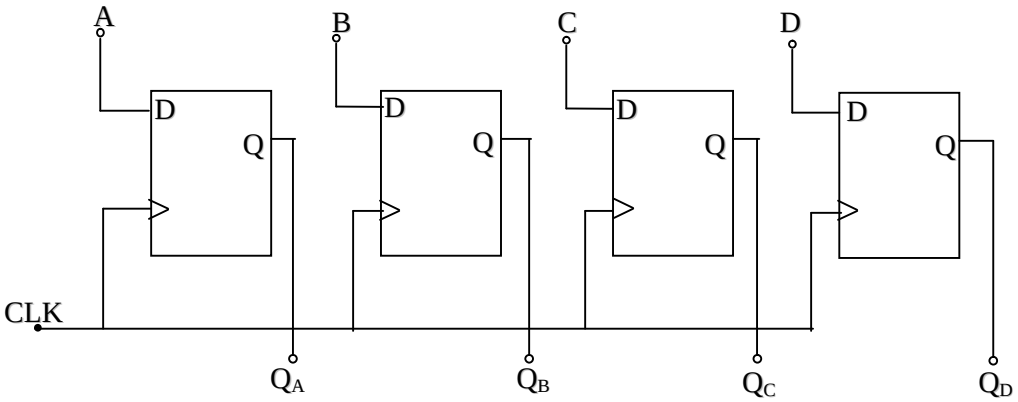


Figure 7 Logic diagram of a 4 – bit parallel – in, parallel – out, shift register

In a parallel – in, parallel – out, shift register, the data is entered into the register in parallel form, and also the data is taken out of the register in parallel form. Immediately following the simultaneous entry of all data bits, the bits appear on the parallel outputs.

Fig 7 shows a 4-bit parallel-in, parallel-out, shift register using D Flip-Flops. Data is applied to the D input terminals of the flip-flops. When a clock pulse is applied, at the positive going edge of that pulse, the D inputs are shifted in to the Q outputs of the Flip-Flops. The register now stores the data. The stored data is available instantaneously for shifting out in parallel form.

Bi-directional Shifter:

A bidirectional shift register is one in which the data bits can be shifted from left to right or from right to left.

Fig 8 shows the logic diagram of a 4-bit serial-in, serial-out, bi-directional shift register. Right/Left is the mode signal.

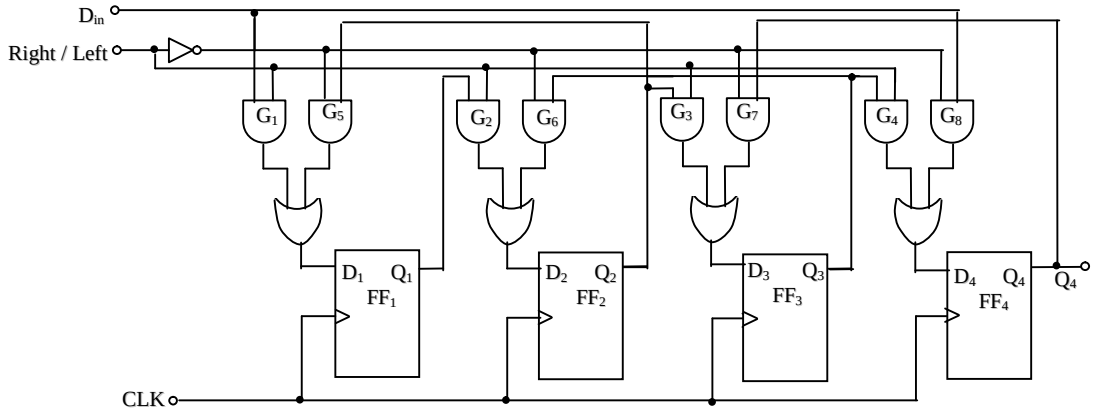


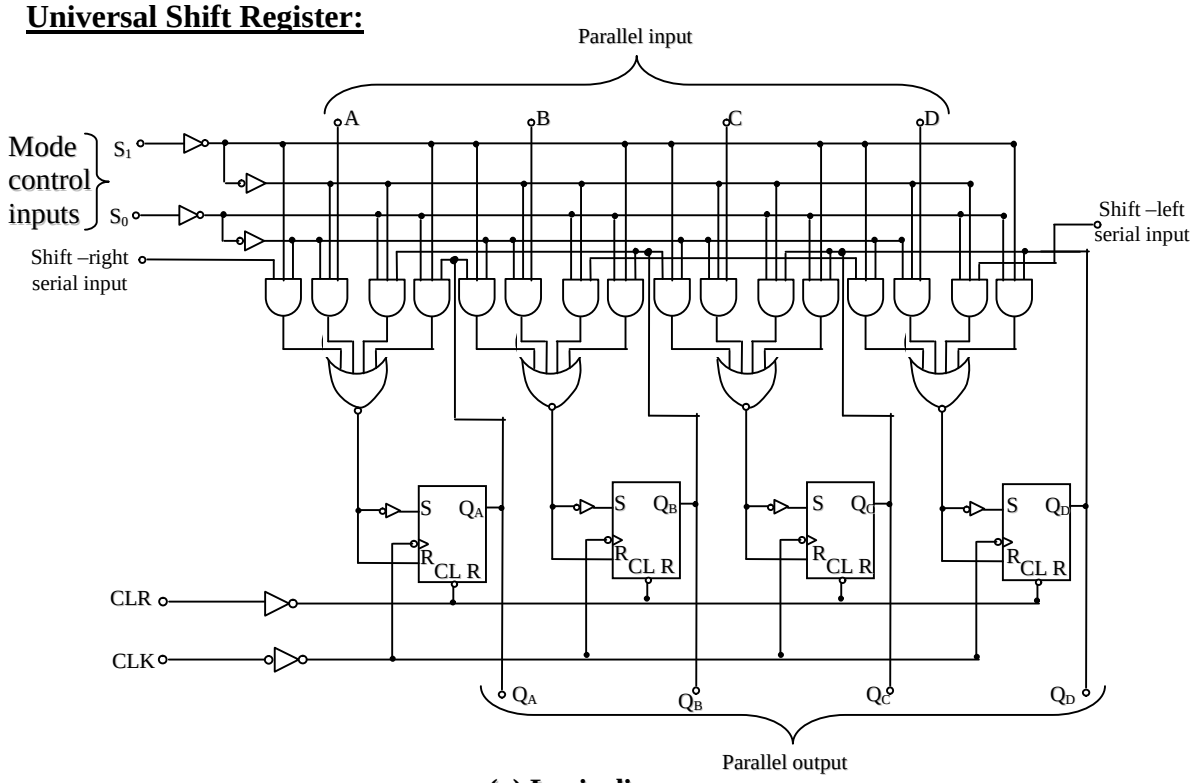
Fig 8 Logic diagram of a 4 – bit bi-directional shift register.

When Right / $\overline{\text{Left}}$ is a 1, the logic circuit works as a shift-right shift register. When Right / $\overline{\text{Left}}$ is a 0, it works as a shift – left register. The bidirectional operation is achieved by using the mode signal and two AND gates and one OR gate for each stage as shown in fig 8.

A HIGH on the Right / $\overline{\text{Left}}$ control input enables the AND gates G_1, G_2, G_3 and G_4 and disables the AND gates G_5, G_6, $G_7$and G_8 and the state of Q output of each FF is passed through the gate to the D input of the following FF. When a clock pulse occurs, the data bits are then effectively shifted one place to the right. A low on the Right / $\overline{\text{Left}}$ control input enables the AND gates G_5, G_6, $G_7$and G_8 and disables the AND gates G_1, G_2, G_3 and G_4 and the Q output of each

Flip-Flop is passed to the D input of the preceding FF. When a clock pulse occurs, the data bits are then effectively shifted one place to the left. Hence, the circuit works as a bidirectional shift register.

Universal Shift Register:



(a) Logic diagram

(b)Truth Table

Inputs S_1	Inputs S_0	Clock	Action
0	0	X	No change
0	1	m	Shift-right
1	0	m	Shift-left
1	1	m	Parallel load

Fig 9 The 74194 **4–bit universal shift register.**

A universal shift Register is a bidirectional register, whose input can be either in serial form or in parallel form and whose output also can be either in serial form or in parallel form.

Fig 9 shows the logic diagrams of the 74194 4-bit universal shift register. The output of each flip-flop is routed through AOI logic to the stage on its right and to the stage on its left. The mode control inputs, S_0 and S_1 are used to enable the left-to-right connections. When it is desired to shift-right, and the right-to-left connections. When it is desired to Shift-Left.

The truth table (tabl3 3) shows that no shifting occurs when S_0 and S_1 are both LOW or both HIGH. When $S_0 = S_1 = 0$, there is no change in the contents of the register, and when $S_0 = S_1 = 1$, the parallel input data A, B, C and D are loaded into the register on the rising edge of the clock pulse. The combination $S_0 = S_1 = 0$ is said to inhibit the loading of serial or parallel data, since the register contents cannot change under that condition. The register has an asynchronous active – LOW clear input, which can be used to reset all the flip-flops irrespective of the clock and any serial or parallel inputs.

Procedure :

Serial in – serial out shift register :

Using two 7476 Ics form 4 D-Flip flops and connect them as shown in Fig 2. to form a serial in serial out 4-bit shift register . Clear all the outputs by entering the reset pulse. All the outputs show level 0. Feed the clock input with the pulser switch `and monitor the logic levels of the output Q3 using the lamp monitoring unit. The J input of the flip – flop (FF0) is connected to the logic switch to provide information bits. Verify the truth table following the method described in theory.

Serial in parallel out shift register :

Using two 7476 Ics form 4 D-Flip flops and connect them as shown in Fig 5. to form a serial in parallel out 4-bit shift register . Clear all the outputs by entering the reset pulse. All the outputs show level 0. Feed the clock input with the pulser switch `and monitor the logic levels of the outputs Q0,Q1,Q2 and Q3 using the lamp monitoring unit. The J input of the flip – flop (FF0) is connected to the logic switch to provide information bits. Verify the truth table following the method described in theory.

Parallel in serial out shift register :

Using two 7476 Ics form 4 D-Flip flops and connect them as shown in Fig 7. to form a parallel in serial out 4-bit shift register . Clear all the outputs by entering the reset pulse. All the outputs show level 0. Feed the clock input with the pulser switch `and monitor the logic levels of the output Q3 using the lamp monitoring unit. Each of the inputs of the flip – flops A,B,C and D are preset with one bit data by connecting them to logic switches to provide information bits. Verify the truth table following the method described in theory.

Parallel in- parallel out shift register :

Using two 7476 ICs form 4 D-Flip flops and connect them as shown in Fig 7. to form a parallel in serial out 4-bit shift register . Clear all the outputs by entering the reset pulse. All the outputs show level 0. Feed the clock input with the pulser switch `and monitor the logic levels of the outputs Q0,Q1,Q2 and Q3 using the lamp monitoring unit. Each of the inputs of the flip – flops A,B,C and D are preset with one bit data by connecting them to logic switches to provide information bits. Verify the truth table following the method described in theory.

Bi directional shift register :

Using two 7476 Ics form 4 D-Flip flops .Using one AND gate IC and two OR gate Ics connect the circuit as shown in Fig 8. to form a 4-bit bi directional shift register . Clear all the outputs by entering the reset pulse. All the outputs show level 0. Feed the clock input with the pulser switch `and monitor the logic levels of the output Q3 using the lamp monitoring unit. The D in input is connected to the logic switch to provide information bits. Verify the truth table following the method described in theory.

Universal shift register : Using a 74194 IC connect various logics to various pins as shown in fig 9. Using S0 and S1 controls and using the method given in theory test the various modes of operation of the Universal shift register and verify the truth table in each mode.

Precautions:

1. Avoid the loose contacts
2. Check proper working of the individual ICs before use
3. Make the perfect contact in breadboard
4. See that all ground points have common ground
5. Use different color wires for clear identification of various signals

Result: The circuits are assembled properly and the results are as per truth tables.

Experiment No. 3

R-C coupled Single stage Common source FET amplifier

Aim: -To construct an RC coupled amp (FET version) and study its frequency response with and with out feedback.

Apparatus: - Field Effect Transistor BFW 10, resistances33kΩ,2.2kΩ,1kΩ,10kΩ, Capacitances .2μf,2μf,50μf/65V, Signal generator, DC Power supply

Theory: -

An amplifier is a device by which one parameter like voltage, current or power of the given signal at input circuit can be increased and obtained at the output terminals by proper selection of the operating point of the transistor. There are several types of classifications of amplifiers basing on

1.Purpose: (Voltage amplifier, current amplifier. Power amplifier)
2. Coupling circuit: RC coupled , Inductance coupled, Transformer coupled
3.Operating point (Class A, Class B, Class C, Class AB)

In order to have very high amplification, we have to use multistage amplifiers. Here we take a single stage amplifier and study its performance. If a fraction of the output (current or voltage) is taken and fed to the input of transistor along with the input it is called feedback amplifier.

RC coupled amplifier can also be formed using Bi junction transistors and Field effect transistors. RC coupled amplifier based on BJT was covered already in the first year course. Here we construct an RC coupled amplifier based on FET and study its performance.

The static characteristics of FET BFW 10 was studied in the first year course. We reproduce here some useful information about FETs

FET is a three terminal device. FET's can be characterized in two main categories like JFET(Junction Field Effect Transistor) and MOSFET(Metal Oxide Field semiconductor). Here we study the JFET characteristics. JFET's are further of two types n-channel type and p-channel type.

Main feature of JFET:-

1. It is a uni polar three terminal device, which solely depends on the conduction of either of electrons or holes.
2. In the operation of this the electric field established by the charges controls the conduction; hence the name Field Effect Transistor.
3. Field Effect Transistor is an voltage control device where as BJT is a current control device. The output current in the BJT is controlled by the input current level where as the output current in FET is controlled by the applied voltage in the input circuit.
4. There are two types of BJT i.e p-n-p and n-p-n. Similarly FET is of two types p-channel FET and n-channel FET.
5. Gate-source junction is generally reverse biased and gate drain junction is forward biased.
6. The effective channel width, which allows current flow, is controlled by reverse biasing the gate source junction, which changes the width of the space charge in the channel.

7. The reverse bias voltage given to gate- source junction, which just prevents the current flow from the source to drain is called “pinch off voltage”.
8. The reverse biasing effect of the gate will be more at the drain end. As drain current is increased by increasing the drain voltage the reverse bias voltage appearing at the drain end (gate - drain end) at some value of V_a this exceeds the reverse breakdown value.Then an avalanche current will flow which is very large This is called breakdown region.

Parameters of FET:-

1. Dynamic Drain resistance (r_d):- Dynamic drain resistance at an operating point is defined as the ratio of small change in drain voltage to the corresponding change in the drain current , when the gate voltage is kept constant.
   $r_d = (\Delta V_{DS}) / (\Delta I_D)$ V_{GS} being constant.
   The typical value of r_d is 400Ω(ohms).
2. Mutual Conductance or Trans conductance (g_m):- The trans- conductance at an operating point is defined as the ratio of a small change in drain current to the corresponding change in gate voltage when drain voltage is kept constant.
   $g_m = (\Delta I_D) / (\Delta V_{GS})$ when V_{DS} is constant. Typical value of g_m is 250μs (micro siemen).
3. Amplification Factor (μ):- Amplification factor is defined as the ratio of small change in drain voltage to the corresponding change in gate voltage when drain current is kept constant.
   $\mu = (\Delta V_{DS}) / (\Delta V_{GS})$ when I is constant.
   μ being the ratio of two voltages it has no units. Typical value of amplification factor of FET is around 10. The above parameters are related by
   $\mu = (r_d) \times g_m$

The drain and the source terminals are taken from n-channel and gate terminal is taken from p type material.

Schematic representation:-

Fig 3shows the schematic representation of FET

(a) (b)

FIG 3

The arrow mark on the gate terminal indicates the direction in which gate current flows when gate junction is forward biased. For n-channel FET arrow is shown into the gate. For p- channel FET arrow is away from gate.

Source:- It is the terminal through majority charge carriers enter the bar.

Drain:- It is the terminal through which majority charge carriers leave the bar.
Gate:- It is the terminal which analogous to base terminal in BJT(Bipolar Junction Transistor) And controls the flow of charge carriers.
Channel:- The region between the source and drain through which majority charge carriers move. The width of this is adjustable by controlling the space charge region in it.

Distinction between BJT and FET amplifiers:
BJT is a Bipolar Device. FET is a Unipolar device.
In BJT amplifiers input junction is forward biased. In FETs it is reverse biased.
BJT is a current controlled device. FET is a field controlled device.
Input current of BJT will be in micro amperes. In FETs it will be in nano to pico amperes.
Input resistance of BJT is about 1kohm .For FET it is about 1 megaohm.
The gain of BJTs is high. Gain of FETs is small.
As the input resistance of FET is high in FET amplifiers to obtain same low frequency response ,it is sufficient if we use very small coupling capacitors when compared to the values coupling capacitors used in BJT amplifiers.
The frequency response of FETS is poor when compared to BJTs. However the technology is being improved to enhance frequency response and gain.
The characteristics of FET are more non linear when compared to BJT characteristics. Hence FET amplifiers produce more distortion in the output signal.
The amplitude of input signal can be larger in FET amplifiers when compared to BJT amplifiers.
In the case of RC coupled amplifier a sinusoidal signal of about 30mV is given and output voltage is measured. The voltage amplification ($A_v = V_{out}/V_{in}$) is calculated at different frequencies and plotted against frequency .

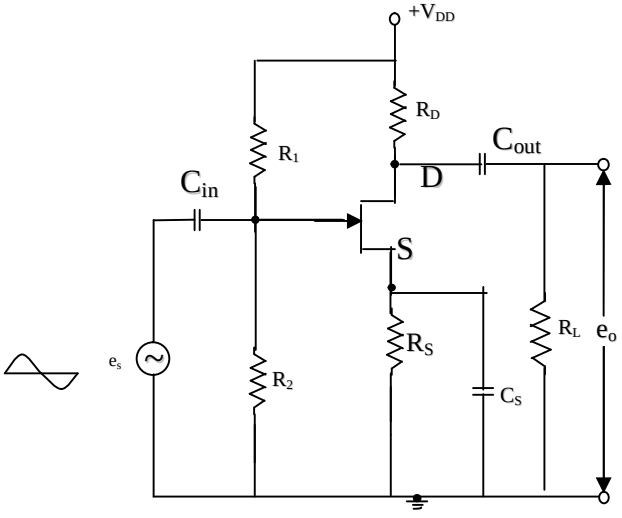
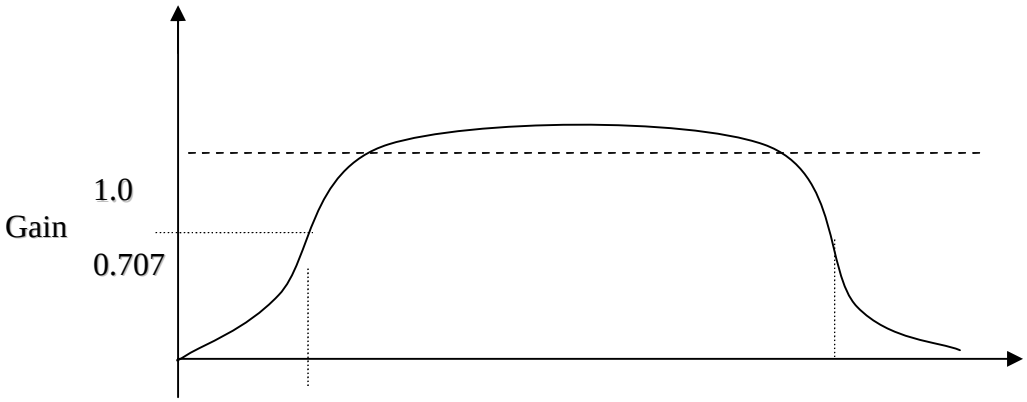


Fig 1 A single stage R-C coupled amplifier.

At low frequencies the amplification increases with increase of frequency and at high frequencies it falls with the increase of frequency. In the midband region gain remains as constant. The circuit diagram is given below (FIG 1). The frequency response curve is shown in fig 2



Normalized gain – frequency response curve

The behavior can be understood as follows. FIG2

Low frequency region: The reactance of coupling capacitor is quite high($1/\omega C$) at low frequencies and thus the output decreases with the decrease of frequency in low frequency region. Further, C_S cannot effectively shunt to source resistance R_S. These are the two reasons that cause the fall in amplification in the low frequency region.

High frequency region: At high frequencies, the reactance of coupling capacitors is very small and they do not offer resistance. If there is a second stage this increases the loading effect and thus decreases the gain. Further, capacitive reactance of Gate source junction at high frequencies becomes low, which increases the gate current. This reduces the amplification factor. By these reasons, gain falls at the high frequency region.

Mid frequency region: The voltage gain in this region remains constant. The coupling capacitors offer zero reactance above the lower cut-off frequency, and at the same time shunt capacitors offer very high and almost constant reactance independent of frequency up to upper cut-off frequency. So in this frequency region gain remains maximum and constant.

The frequencies where the gain is 70.7% of the maximum gain are called cutoff frequencies. There are two such frequencies f_1 and f_2 one on the lower frequency side and the other on the higher frequency side.

Bandwidth $=(f_2 - f_1)$. On decibel scale the power reduction is of three decibels.

Procedure:

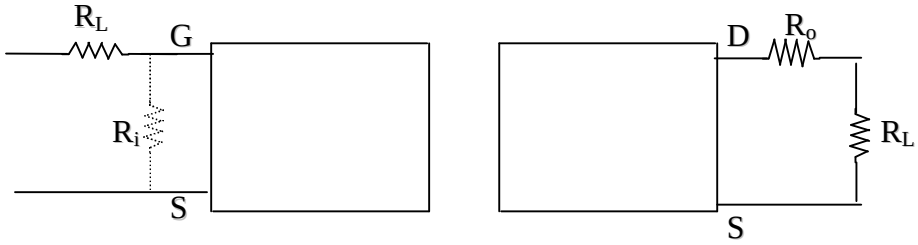
Plotting frequency response curve :

Circuit should be connected as shown in the figure. 30mV (peak to peak) is applied to the gate source using signal generator. Now measure the output voltage by varying the frequency from 50Hz to 1MHz. First the output voltage increases and reaches a constant value. It remains constant up to a certain frequency and then it decreases. Measure the input and out put voltages in the range 50 Hz to 2Mhz and calculate the gain in dB. After completion of taking readings draw graph taking log f on x axis and gain on Y axis. It is called frequency response curve.

Measuring the input and output resistances:

Connect the circuit as shown in fig. Measure the generator voltage and voltage across R_s. as R_s is varied from zero in steps of 100 ohms the voltage across it increases. Note the value of Rs at which V_i is equal to $V_s/2$. At this value $R_i=R_s$

For output resistance put decade box at output and vary that until it comes to $V_o/2$. It is the output resistance R_o.



Observations:

FREQUENCY	OUTPUT VOLTAGE (Vo) volts	Gain Vo/Vi	Gain dB 20 log (Vo/Vi)

Precautions: 1.the coarse and fine knobs of a power supply should be kept in the minimum position before the commencement of the experiment.
2.Loose connections are to be avoided.
3.Readings must be take carefully on C.R.O.
4. At each observations the input voltage magnitude maintain at 30mv peak to peak.

Result:

Band Width obtained from the graph = kHz.
Input resistance Ri = k Ohms.
Output resistance Ro = k Ohms.

Experiment No:4

ASTABLE MULTI VIBRATOR USING 741

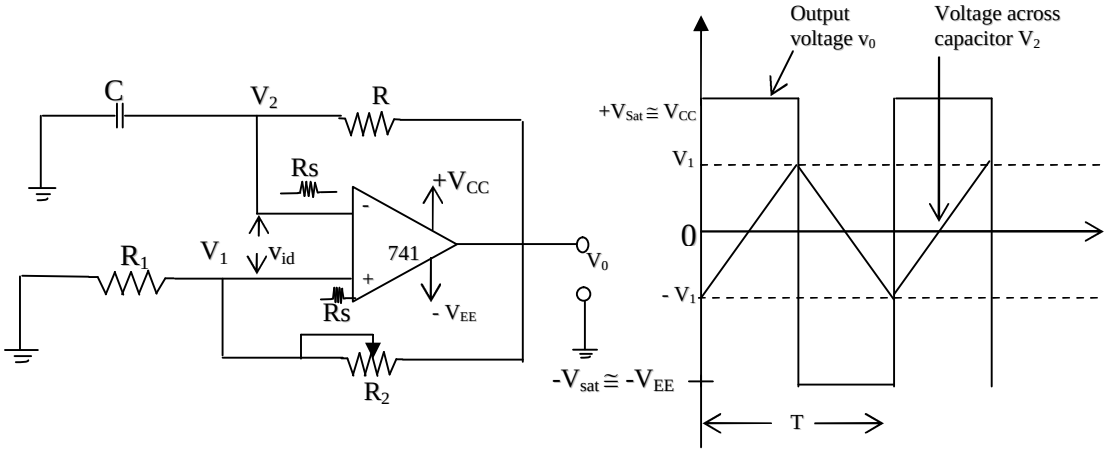
AIM:- To construct an astable multi vibrator with 741 op amp and generate different frequencies with different time constants and compare them with calculated values .

APPARATUS;- IC 741 , power supply , signal generator , CRO, resistances and capacitances of various values.

THEORY:-

Square Wave Generator:

In contrast to sine wave oscillator, square wave outputs are generated when the op-amp is forced to operate in the saturated regions, that is, the output of the op-amp is forced to swing repetitively between positive saturation + V_{sat}($\cong$ + V_{CC}) and negative saturation-V_{sat} ($\cong$ - V_{EE}), resulting in the square wave output. Such a circuit shown in fig.



$$|V_1| = \frac{R_1}{R_1 + R_2}\ |V_{sat}|;\ f_0 = \frac{1}{T} = \frac{1}{2RC} \text{ if } R_2 = 1.16R_1$$

Fig (a) Square wave generator (b) Wave form of output voltage. v_0 and capacitor voltage v_2 of the square wave generator.

This square wave generator is also called a free-running multi-vibrator. Assume that the voltage across the capacitor C is zero volts at the instant the dc supply voltages +V_{CC} and –V_{EE} are applied. This means that the voltage at the inverting terminal is zero initially. At the same instant, however, the voltage V_1 at the non-inverting terminal is very small finite value, that is a function of the output offset voltage V_{001} and the values of R_1 and R_2 resistors. Thus the differential input voltage V_{id} is equal to the voltage V_1 at the non-inverting terminal. Although very small, voltage V_1will start to drive the op-amp into saturation.

For example, suppose that the output offset voltage V_{001} is positive and that, therefore voltage V_1 is also positive. Since initially the capacitor C acts as a short circuit, the gain of the amplifier is very large (A); hence V_1 drives the output of the op-amp to its positive saturation + V_{sat} with the output voltage of the op-amp as $+V_{sat}$, the capacitor C_2 starts charging toward $+V_{sat}$ through the resistor, R. However, as soon as the voltage v_2 across the capacitor is slightly more positive than V_1, the output of op-amp is forced to switch to a negative saturation, $-V_{sat}$. With the op-amps output voltage as negative saturation, $-V_{sat}$ the voltage V_1 across R_1 is also negative, since $V_1 = \frac{R_1}{R_1 + R_2}(-V_{sat})$ ---------- (1)

Thus the net differential voltage $V_{id} = V_1 - V_2$ is negative, which holds the output of the op-amp in negative saturation. The output remains in negative saturation until the capacitor C discharges and then recharges to a negative voltage slightly higher than the $-V_1$ [see Fig.] Now, as soon as the capacitor voltage v_2 becomes more negative than $-V_1$, the net differential voltage V_{id} becomes positive and hence drives the output of the op-amp back to its positive saturation $+V_{sat}$. This completes one cycle. With output at $+V_{sar}$, and voltage V_1, at the non-inverting input is

$$V_1 = \frac{R_1}{R_1 + R}(+V_{sat}) \qquad \text{----------(2)}$$

The time period T of the output waveform is given by

$$T = 2RC \ln\left(\frac{2R_1 + R_2}{R_2}\right) \quad \text{---------} \quad (3)$$

Equation 3 indicates that the frequency of the output f_0 is not only a function of the RC time constant but also of the relationship between R_1 and R_2. For example If $R_2 = 1.16\ R_1$, equation 3 becomes

$$f_0 = \frac{1}{2RC} \quad \text{----------} \quad (4)$$

Equation (4) shows that the smaller the RC time constant, the higher the output frequency f_0 and vice versa. As in the sine wave oscillator, the highest frequency square wave generated is also set by the slew rate of the op-amp. In practice, each inverting and non-inverting terminal needs a series resistor R_s to prevent excessive differential current flow because the input of the op-amp is subjected to large differential voltage. R_s should have a value of 100 KΩ or higher.

The feed back resistance R and the capacitance C provides the integrator action .The op-amp serves as a regenerative comparator . The output of comparator is limited by the saturation level of the op-amp .A function $\beta = R_1/R_1+R_2$ of the output voltage is feed back to the comparator that compares the voltage across the capacitor with the voltage and switches to negative saturation limit .

The moment V_I becomes greater than βV_e , the capacitor begins to discharge through R_b until V_c is equal to βV_0. The output is than switched on to the +ve saturations. The momentj v V_c becomes less than BV_0.

PROCEDURE: The circuit is connected as shown in fig and the output wave form is observed using the CRO. The capacitor is varied and the frequency is measured at each step, The observed frequencies are compared with the calculated ones .

OBSERVATIONS:

DC supply voltage V_{cc} = ……..V

R_1	R_2=1.16R_1	Capacitance C(μf)	Time period	Frequency	
				(Observed)	Calculated

PRECAUTIONS:

1. Distortion should not be in the waves.
2. Connections must be properly verified.
3. Power supply should be adjusted to 15V or less.

RESULT:

The calculated frequencies are found to be in good agreement with the observed values

Experiment :4

Astable multi-vibrator using 555

Aim : To form an astable multivibrator and study its wave forms.

Apparatus : 555 timer IC,0-15 V DC power supply, capacitors, resistors, connecting Wires and bread board.

Description of 555 timer:

The 555 Timer and its pin configurations: Signetic corporation introduced this device as the SE / NE555 in early 1970 for the first time . It is one of the most versatile linear integrated circuit. The applications of 555 timer includes mono-stable and astable multi-vibrators, a.c-d.c converters, digital logic probes, wave form generators, analog frequency meters and tachometers, temperature measurement and control, infrared transmitters, burglar or toxic gas alarm, voltage regulators, electronic eyes and many others. The 555 is a monolithic timing circuit that can produce accurate and highly stable time delays or oscillation. In other words the timer basically operates in one of the two modes: either as a mono stable multi-vibrator (one shot) or as an astable (free running) multi-vibrator. The device is available as an 8 pin metal, as an 8-pin mini DIP or a 14-pin DIP. Fig shows that the pin connection diagram and the block diagram of the SE/NE 555 timer. The SE 555 is designed for the operating temperature range of –55 to $+125^0$c, while NE 555 operates over a temperature range of 0 to $+70^0$c. The important features of the 555 Timer are **(1)** It operates on +5 to +15v supply voltage in both free running (astable) and one shot (mono stable) modes; **(2)** It has an adjustable duty cycle; timing is from micro seconds through hours; **(3)** It has high current output; it can source or sink 200 mA; **(4)** The output can drive TTL and has a temperature stability of 50 parts per million (ppm) per degree Celsius change in temperature.

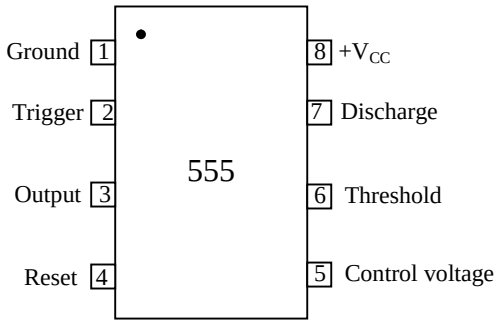


Fig (1a) 555 timer connecting diagram

Before proceeding with the operations of 555 Timer an astable multi-vibrator, it is important to examine its pin functions. The pin number is used in the following discussion refer to the 8 pin mini DIP and 8 pin metal can packages

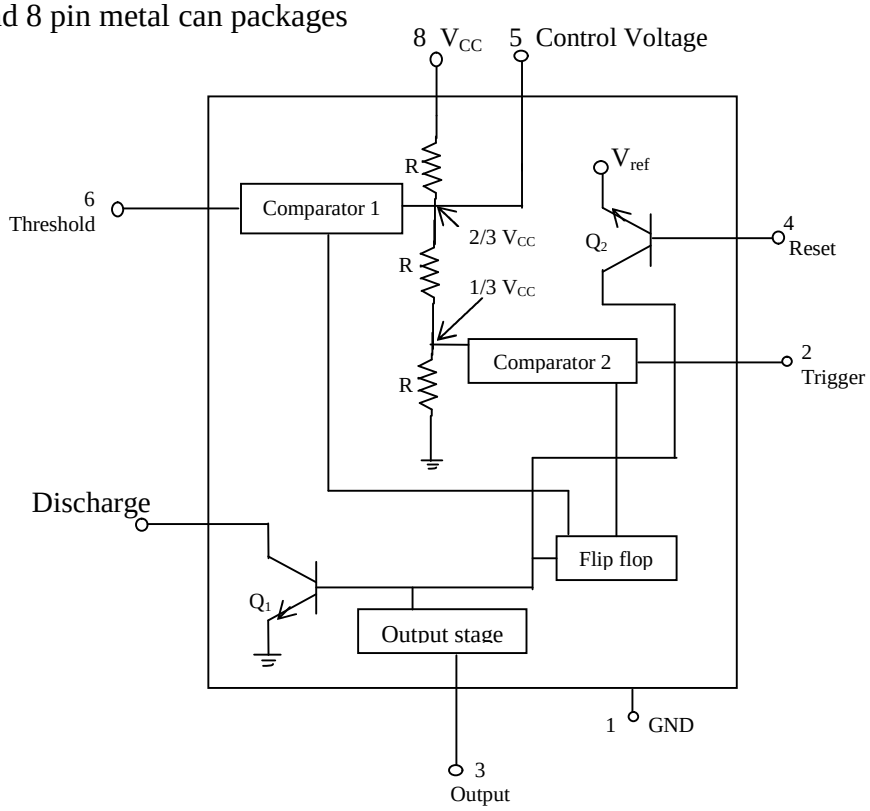


Fig (1b) block diagram

Pin 1: Ground. All voltages are measured w.r.t. this terminal.

Pin 2: trigger. The output of the timer depends on the amplitude of the external trigger pulse applied to this pin. The output is low if the voltage at this pin is greater than 2/3 V_{cc}. However, when a negative going pulse of amplitude larger than 1/3 V_{cc} applied to this pin, the comparator 2 output goes low, which is turn switches the output of the timer high. The output remains high as long as the trigger terminal is held at a low voltage.

Pin 3: there are two ways a load can be connected to the output terminal: either between pin 3 and ground (pin 1) or between pin 3 and supply voltage $+V_{cc}$ (pin 8). When the output is low, the load current flows through the load connected between 3 and $+V_{cc}$ into the output terminal and is called the sink current. However, the current through the grounded load is zero when the output is low. For this reason, the load connected between 3 and $+V_{cc}$ is called normally on load , and that

connected between pin 3 and ground normally off load and that connected between pin 3 and ground is called the normally off load.

Pin 4: Reset. The 555 timer can be reset (disabled) by applying a negative pulse to this pin. When the reset function is not in use, the reset terminal should be connected to $+V_{cc}$ to avoid any possibility of false triggering.

Pin 5: control voltage. An external voltage applied to this terminal changes the threshold as well as the trigger voltage. In other words, by imposing a voltage on this pin or by connecting a pot between this pin and ground, the pulse width of the waveform can be varied. When not used, the control pin should be bypassed to ground with a 0.01μF capacitor to prevent noise problems.

Pin 6: Threshold. This is the non-inverting input terminal of comparator 1, which monitors the voltage across the external capacitor. When the voltage at this point is ≥ threshold voltage $\frac{2}{3}V_{cc}$, the output of comparator 1 goes high, which in turn switches the output of the timer low.

Pin 7: Discharge. This pin is connected internally to the collector of transistor Q_1. When the output is high, Q_1 is off and acts as an open circuit to the external capacitor C connected across it. On the other hand, when the output is low, Q_1 is saturated and acts as a short circuit, shorting out the external capacitor C to ground

Pin 8: $+V_{cc}$. The supply voltage of +5V to +18V is applied to this pin with respect to ground

(pin 1)

The 555 as an astable multi-vibrator:

An astable multi-vibrator, often called a free running multi-vibrator, is a rectangular wave generating circuit. The time during which output is either high or low is determined by the two resistors and a capacitor, which are externally connected to the timer 555

Astable operation: Fig 2(a) shows the 555 timer connected as an astable multi-vibrator. Initially, when the output is high, capacitor C starts charging toward V_{cc} through R_A and R_B.

However as soon as voltage across the capacitor equals $\frac{2}{3}V_{cc}$, comparator 1 triggers the flip-flop, and the output switches low [see fig 2(b)]. Now capacitor C starts discharging through R_B and transistor. When the voltage across C equals $\frac{1}{3}V_{cc}$, comparator 2's output triggers the flip-flop and the output goes high. Then the cycle repeats. The output voltage and capacitor voltage waveforms are shown in figure 2(b).

As shown in this figure, the capacitor C is periodically charged and discharged between $\frac{2}{3}$ V_{cc} and $\frac{1}{3}$ V_{cc} respectively. The time during which the capacitor charges from $\frac{1}{3}$ V_{cc} to $\frac{2}{3}$ V_{cc} is equal to the time the output is high and is given by

$$t_c = 0.69\ (R_A + R_B)C \quad \text{----------} \quad (1)$$

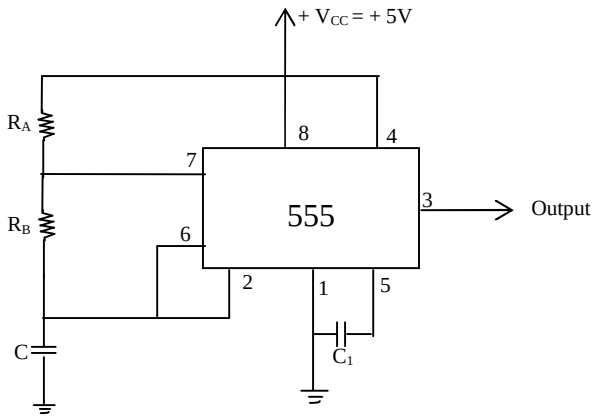


Fig 2(a) The 555 astable multi-vibrator circuit

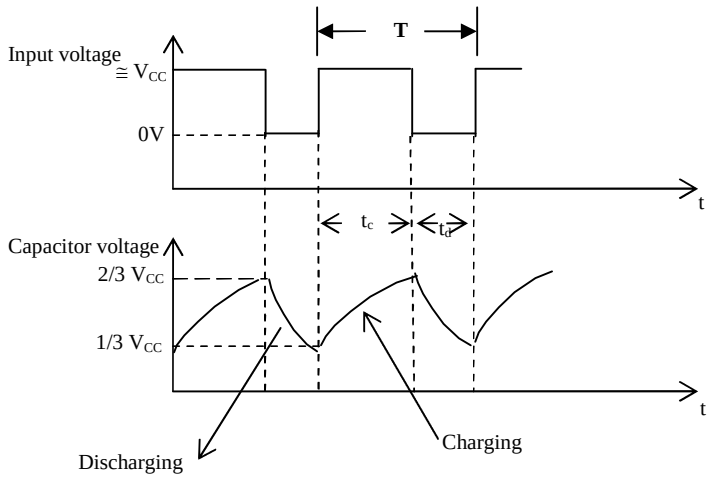


Fig 2(b) The 555 astable multi-vibrator output voltage wave form and voltage across the capacitor.

Where R_A and R_B are is ohms and C is in Farads. Similarly the time during which the capacitor discharges from $\frac{2}{3}$ V_{cc} to $\frac{1}{3}$ V_{cc} is equal to the time, the output is low and is given by

$$t_d = 0.69\ R_B C \quad \text{---------- (2)}$$

Thus the total time period of the output waveforms is

$$T = t_c + t_d = 0.69\ (R_A + 2R_B)\ C \quad \text{----------(3)}$$

This, in turn gives the frequency of oscillation as

$$f_0 = \frac{1}{T} = \frac{1.45}{(R_A + 2R_B)C} \quad \text{---------- (4)}$$

Equation (4) indicates then the frequency f_0 is independent of the supply voltage V_{cc}.

Output is a square wave of period $t = 0.68(R_1 + R_2)C$. $T_2 = 0.698R_2C$:

$T = T_1 + T_2$

$T = 0.68(R_1 + 2R_2)C$.

$R = 1/T = 1.443/(R_1 + 2R_2)C$

PROCEDURE: The circuit is connected using capacitors and the output waveform is observed using the CRO. R_1 and R_2 are chosen 1K and 20K respectively. The supply voltage is given to the fourth and eighth pins respectively. The capacitor is varied and the frequency is measured at each step. The observed frequency is compared with the calculated values. For each combination of R & C the peak value of the output voltage and the voltage across sixth pin are measured. For each setting using a tracing paper the waveform observed on CRO screen is traced. The effect of change in the value are recorded, by varying R_1 the effect of it on the waveform is studied.

Observations:

DC supply voltage V_{cc} = ……..V

R_A	R_B	Capacitance C (μf)	Time period	Frequency	
				(Observed)	Calculated

PRECAUTIONS:
1. Distortion should not be in the waves.
2. Connections must be properly verified.
3. Power supply should be adjusted to 15V or less.

RESULT:
The calculated frequencies are found to be good agreement with the observed values.

Experiment No. 5

INVERTING, NONINVERTING and SUMMING AMPLIFIER

AIM:- To study the working of op-Amp as Inverting, Non-inverting and Summing amplifiers .

APPARATUS:-

S.NO.	NAME OF THE ITEM	RANGE	QUANTITY
1.	DC Regulated Power Supply	±12V	1
2.	Digital multi meter	–	1
3	breadboard	-	1
4.	Op-Amp IC-741		1
5.	Resistances	10kΩ	3
		1kΩ,2.2kΩ	
		3.3kΩ,4.7kΩ	1

Details of 741 OP AMP :

IC 741 is available in two packages. The pin configuration of 741 op-amp in dual in line package (DIP) is given below

It is the most commonly and widely used general-purpose op-amp. It has an integrated 30pF MOS capacitor. It has high input impedance (> 1MΩ), low output impedance (750 Ω) and large voltage gain (200,000).

.

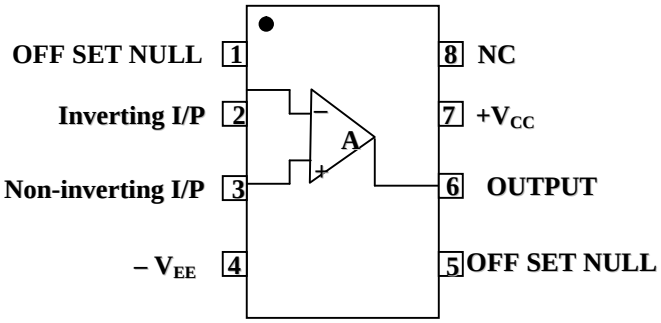


Fig (1.9) pin configuration of μA-741 op-amp

A dot on the top left corner is used to identify pin number 1. It needs a ±12V DC dual power supply. It is basically a high gain differential amplifier with two inputs.

Operational Amplifier:

An operational amplifier is a direct-coupled high gain amplifier usually consists of one or more differential amplifiers and usually followed by a level translator and an output stage. The output

stage is generally a push pull or push pull complementary symmetry pair. An operational amplifier is available as a single integrated circuit package. The operational amplifier is a versatile device that can be used to amplify dc as well as ac input signals and was originally designed for computing such mathematical functions as addition, subtraction, multiplication and integration. Thus the name operational amplifier stems from its original use for doing these

mathematical operations and so is abbreviated to op-amp. With the addition of suitable external feed back component the modern day operational amplifier can be used for a variety of applications. Such as ac and dc signal amplification, active filters, oscillators, comparators, regulators and others.

Schematic symbol:

The most widely used symbol for a circuit with two inputs and one out put is shown in fig (1.6)

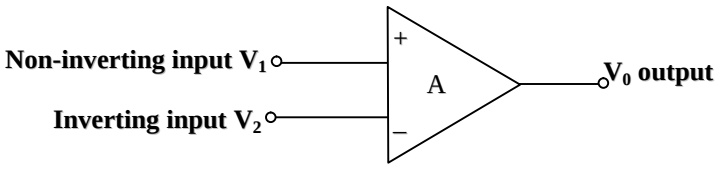


Fig (1.6) Schematic symbol of op-amp

In fig (1.6)

v_1 = voltage at the non-inverting input (volts)

v_2 = voltage at the inverting input (volts)

v_o = output voltage (voltage)

All these are measured w.r.t. ground

A = large signal voltage gain that is specified on the data sheets for an op-amp

power supply and other pin connections are not usually shown in circuits. Since the input differential amplifier stage of the op-amp is designed to be operated in the differential mode, the differential inputs are designated by the (+) and (-) notations, the (+) input is used for non-inverting input. An ac signal (or dc voltage) applied to this input produces an in-phase (or same polarity) signal at the output. On the other hand the (-) input is the inverting input because an ac signal (or dc voltage) applied to this input produces an 180 out of phase (or opposite polarity) signal at the output

Ideal op-amp:

An ideal op-amp exhibits the following electrical characteristics.

(1) Infinite voltage gain A_V.

(2) Infinite input resistance R_i, so that, almost any signal source can drive it and there is no loading of the preceding stage.

(3) Zero output resistance R_o, so that, output can drive an infinite number of other devices

(4) Zero output voltage when the input voltage is zero.

(5) Infinite bandwidth, so that, any frequency signal from 0 to $\propto$ Hz can be amplified with out attenuation.

(6) Infinite common mode rejection ratio so that output common mode noise voltage is zero.

(7) Infinite slew rate so that voltage changes occur simultaneously with input voltage changes.

There are practical op-amps that can be made to achieve some of these characteristic using a negative feedback arrangement. In particular, the input resistance, the output resistance, and bandwidth can be brought close to ideal values by this method.

As the open loop gain of op-amp is very high, only very small signals (of the order of micro-volts or less) having very low frequency may be amplified accurately with out distortion. However, signals this small are very susceptible to noise. Besides being large the open loop gain of the op-amp is not constant. The voltage gain varies with changes in temperature and power supply as well as with mass production techniques. The variations in voltage gain are relatively large in open-loop op-amp, in particular, which makes the open-loop op-amp unsuitable for many linear applications. In most linear applications the output is proportional to the input and is of the same type.

Further the bandwidth (band of frequencies for which the gain remains constant) of most open-loop op-amps is negligibly small - almost a zero. For this reason, the open-loop op-amp is impractical in ac applications. For instance the open loop bandwidth of the 741C is approximately 5Hz. However, in almost all ac applications a bandwidth larger than 5 Hz is needed.

Because of the above stated reasons, the open loop op-amp is generally not used in linear applications. Never the less in certain applications the open-loop op-amp is purposely used as a nonlinear device; that is a square ware output is obtained by deliberately applying a relatively large input signal. Open-loop op-amp configurations are most suitable in such applications.

We will be able to select as well as control the gain of the op-amp, if we introduce a modification in the basic circuit. This modification involves the use of feedback, that is, an output signal is fed back to the input either directly or via another network. If the signal fed back is of opposite polarity or out of phase by 180^0 with respect to input signal, the feedback is called of negative feedback. An amplifier with negative feedback has a self - correcting ability against any change in output voltage caused by changes in environmental conditions. Negative feedback is also known as degenerative feedback because when used it degenerates (reduces) the output voltage amplitude and in turn reduces the output gain.

The inverting amplifier with negative feed back:

In the inverting amplifier only one input is applied and that, to the inverting input terminal. The non-inverting input terminal is grounded. Since $v_1 = 0$ and $v_2 = v_{in}$.

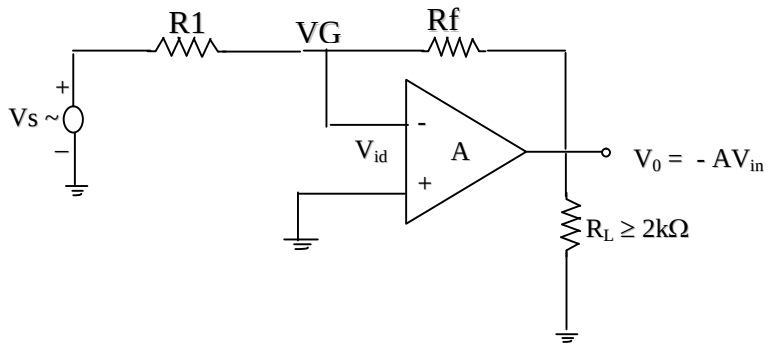


Fig 1.13 Inverting amplifier

The negative sign indicates that the output voltage is out of phase with respect to input by 180^0 or is of opposite polarity. Thus in the inverting amplifier the input signal is amplified by gain A and is also inverted at the output.

Gain of inverting amplifier with feed back:

As the non inverting input of the amplifier is grounded and as no current passes through the input resistance (it is assumed to be infinite in ideal case)

The inverting input terminal also must be at zero potential as there cannot be any voltage drop across R_i.The junction point of R_1 , R_f and inverting input is called virtual ground. Therefore

the input current $I_i = \frac{V_s}{R_1}$

flows through Rf resulting in a voltage drop = $R_f * I_i = -V_o$.

Therefore $V_o = R_f \times (-V_s / R_1) = -(R_f / R_1) \times v_s$

Voltage gain $Av = V_o/V_i = -(R_f / R_1)$

The expression for voltage gain doesn't contain any parameter pertaining to OP AMP. If $R_f = R_1$. The input and output signals have equal amplitude but differ in phase by 180^o .It is called inverting unity gain amplifier. If the ratio R_f/R_1 is varied the output signal amplitude can be scaled up or down. In this application it is called scale changer. Because of virtual ground mentioned above any signal connected to the inverting input through a resistance $R_x = R_1$ supplies current independent of other sources. The current flowing through R_f is algebraic sum of the currents entering in the node. As a result the output voltage can be considered as sum of input signal amplitudes. In this configuration such an amplifier is called summing amplifier. Any number of input signals can be connected like this and the output voltage of the summing amplifier is

$$v_o = -\frac{R_f}{R}(v_1 + v_2 + ...v_x + ...v_n)$$

The non-inverting amplifier:

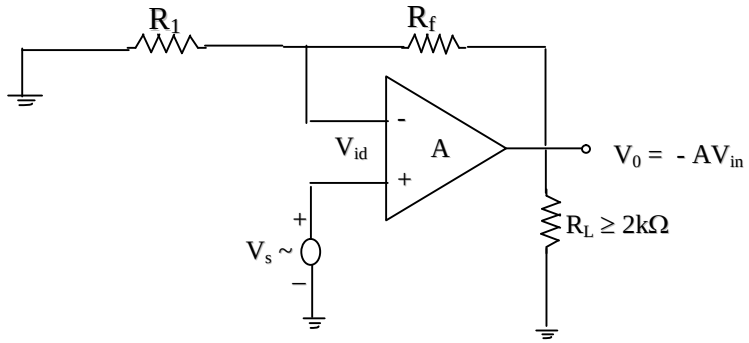


Fig 2 Non – Inverting amplifier

Fig 2 shows the open loop configuration of non inverting amplifier. In this configuration the input is applied to the non inverting input terminal and the inverting input terminal is grounded.

In the circuit shown below $v_1=v_{in}$ and $v_2=0v$. Therefore according to equation $v_0 = Av_{in}$. This means that the output voltage is A times larger than the input voltage and is in phase with input signal. The type of feedback involved is called voltage series feedback. The circuit 2 is redrawn to show feedback.

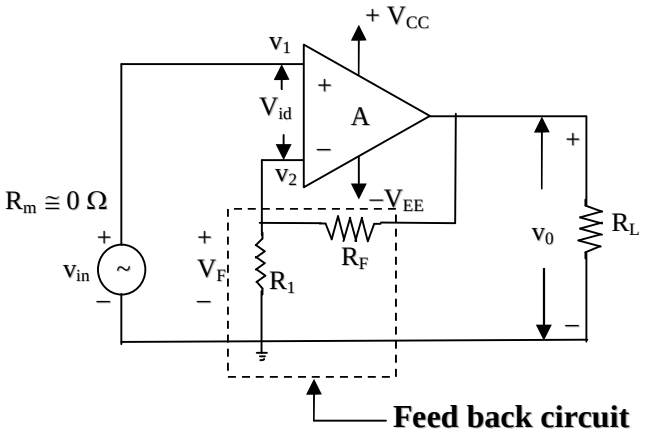


Fig (2.2): voltage series feed back amplifier.

Expression for voltage gain:

As the input resistance of the amplifier is infinite, the inverting terminal will be at same potential as that of non inverting input. The current flowing through the resistance R_f is $(v_o-v_1)/R_f$. It must be equal to v_1/R_1 in magnitude. Equating

$$\frac{v_1}{R_1} = -\frac{(v_1 - v_o)}{R_f}$$

we can rewrite it as

$$v_1\left(\frac{1}{R_f} + \frac{1}{R_1}\right) = \frac{v_o}{R_f}$$

The voltage gain $$A_v = \frac{v_o}{v_s} = 1 + \frac{R_f}{R_1}$$

In this configuration output signal will be in phase with input.. For $R_1=R_f$ the gain is 2.

PROCEDURE:-

INVERTING AMPLIFIER:-

1. Select R_f and R_1 depending on the gain required. Connections are made as per the circuit diagram . Use low wattage resistors preferably one eighth watt so that resistance leads smoothly fit into the breadboard. When high wattage resistors are used in a circuit do not use them in bread board. Use separate group board for them.
2. Fix the IC741 on breadboard such that pins insert in the holes without bending. Enough care must be taken while inserting and removing ICs from bread boards and sockets. Better use IC inserters and pluckers for this.
3. Use minimum length of wire to make connections. The gauge of the wire is selected such that it goes into the whole smoothly. Bent wires should not be inserted. Use a wire cutter cum sleeve remover to cut bent ends and to remove plastic sleeve. Use various colours of wires so that the connections can be identified easily. Use common ground. If several ground points are used ensure that they are all common by checking resistance between those points.

4. Use fixed voltage DC power supply preferably SMPs. The +12 V DC line is given to pin No. 7. –12V DC is given to pin No. 5. Fix the

2.For inverting amplifier vary the input voltage from 1 to 6v in steps of 1v and measure the output voltage as a function of the input voltage .

3.Compare the experimental gain with the total gain of $-R_F/R_1$

S.No.	R_1	Output Voltage		Gain	
		Practical	Theoretical	Practical	Theoretical

SUMMING AMPLIFIER:-

1.Apply $V_1$and varyV_2 from .5 to 2.5v in steps of .5measure the output voltage with the help of multimeter.

2. Compare the experimental values with the theoretical values.

S.No	Input voltage		Output voltage	
	V_1	V_2	Theoretical	Practical

NONINVERTING AMPLIFIER:-

1.Apply the input voltage to the non-inverting terminal from 0.5 to 2.5 in steps of 0.5V measure the output voltages.

2. Compare the practical and theoretical values.

S.No.	R_1	Output Voltage		Gain	
		Practical	Theoretical	Practical	Theoretical

PRECAUTIONS:

1) Check the IC for corking by using it in inverting amplifier.
2) Ensure common ground for all ground connections.
3) Some function generators may have offset null facility. Use it if no output signal is observed even after circuit is assembled properly.

RESULT:-

We observed that the OP-AMP is used as an inverting ,non-inverting and summing amplifiers.

Experiment No.6

WEIN BRIDGE OSCILLATOR

AIM: To construct wean bridge oscillator using IC 741 and to generate sine wave of different frequencies.

APPARATUS: op-amp IC 741, resistance 1 kohm-2, 100 kohm, 10kohm , potentiometer 10k ohm pot, capacitors 0.003Mf, 0.22mf, 0.022mf, 0.002mf, breadboard, DC regulated power supply, function generator, dual trace C.R.O, connecting wires.

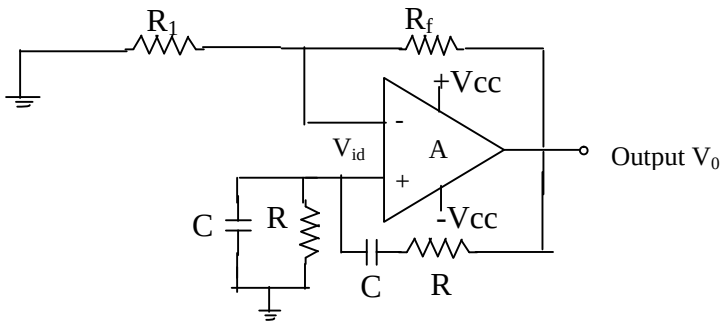


Fig 1 Wein's bridge oscillator

THEORY: An oscillator in which a balanced Wein's bridge is used as the feedback network is called the Wein's bridge oscillator. The active element is an operational amplifier which has a very large positive gain negligible output resistance and very high input resistance. At W_0 frequency, the network introduces zero phase shift. At this frequency the feedback ratio is 1/3. that is it introduces attenuation of 1/3. To maintain oscillations the gain must be unity. This is achieved by introducing negative feedback through R1 and Rf. The values of R_1 and R_f are adjusted such that $R_1 = 2R_f$.

The cuircuit oscillates as long as =1/3. Any deviation from this condition makes the oscillator unstable. The amplitude of the oscillator may be stabilized by varying negative feedback by using a potentiometer in place of R_1. If for any reason, the amplitude of the output voltage increases , the amount of negative feedback increases, there by bringing down the amplitude to a suitable value.

PROCEDURE:

1) connect the circuit as shown in diagram.
2) choose R_1=10kohm potentiometer for R_f connected as variable resistance.
3) Variation of potentiometer allows one to start the oscillations and also to control the distortion.
4) Adjust the potentiomrer such that the oscillations are stable and distortion is minimum.

5) directly read the frequency of the sine wave in C.R.O.
6) by varying the capacitance value repeat the calibration.
7) repeat the same procedure for different capacitance values.

Table

S.No	Capacitance μ f	Time period (T) = λ t Seconds	Observed frequency (1/T) Hz	Calculated frequency 1/2πRC Hz

PRECAUTIONS:

1) bread board should be cheeked before giving the connections.
2) it is checked with millimeter.
3) IC 741 also be checked by gain checking circuit as shown in the figure.
4) Loose connections are avoided in the circuit.
5) The conducting parts of any two wires cannot touch each other.
6) The C.R.O reading should be taken carefully.

RESULT:

Wien bridge oscillator using OP-AMP IC 741 was constructed and sine wave was generated for different frequencies by changing the capacitors for a given value of R.
Experiment is also repeated for different values of R keeping a fixed value for C. The calculated and observed frequencies are agreeing satisfactorily.

Suggestions to teachers: The students may be asked to trace the waveforms for different values of R and C maintaining RC product at a constant value and record their remarks.

Experiment No.7

Colpitts oscillator

AIM: To construct a Colpitts oscillator using Op amp IC 741 and to study its performance for different inductance (L) values and capacitance values.

APPARATUS: Op-amp IC 741, resistance 1 kohm-2, 100 kohm, 10kohm , capacitors 0.003Mf, 0.22mf, 0.022mf, 0.002mf each 2 nos, breadboard, DC regulated power supply(12V DC), function generator, dual trace C.R.O, bread board, single strand connecting wires of different colors variable inductance box.

Theory:

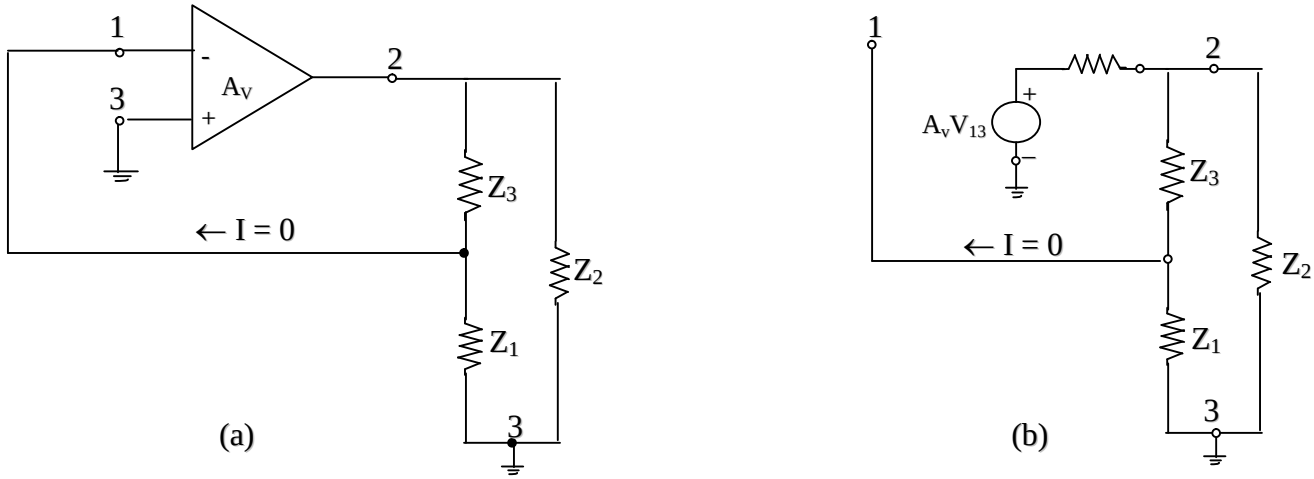


Fig 1(a) The basic configuration for many resonant-circuit oscillators
(b) The linear equivalent circuit using an operational amplifier

Many oscillator circuits fall into the general form shown in fig1a. The active device may be a bipolar transistor, an operational amplifier or an FET. In the analysis that follows we assume an active device with infinite input resistance such as an FET, or an operational amplifier. Fig.2b shows the linear equivalent circuit of Fig.2a, using an amplifier with negative gain $-A_v$ and output resistance R_o. Clearly the topology of fig2 is that of voltage-series feedback.

The Loop Gain: The value of $-A\beta$ will be obtained by considering the circuit of fig to be a feedback amplifier with output taken from terminals 2 and 3 and with input terminals 1 and 3. The load impedance Z_L consists of Z_2 in parallel with the series combination of Z_1 and Z_3. The gain without feedback is $A = -A_vZ_L/(Z_L+R_o)$. The feedback factor is $\beta = -Z_1/(Z_1+Z_3)$. The loop gain is found to be

$$-A\beta = \frac{-A_v Z_1 Z_2}{R_0(Z_1+Z_2+Z_3)+Z_2(Z_1+Z_3)} \qquad \text{----------(1)}$$

Reactive Elements: Z_1,Z_2, and Z_3 If the impedances are pure reactances (either inductive or capacitive), then $Z_1=jX_1$, $Z_2=jX2$, $Z_3=jX_3$. For an inductor, $X=\omega L$, and for a capacitor, $X=-1/\omega C$. Then

$$-A\beta = \frac{+A_v X_1 X_2}{jR_o(X_1+X_2+X_3)-X_2(X_1+X_3)} \qquad \text{----------(2)}$$

For the loop gain to be real (zero phase shift),

$X_1+X_2+X_3 = 0$ ----------(3)

and

$$-A\beta = \frac{A_v X_1 X_2}{-X_2(X_1+X_3)} = \frac{-A_v X_1}{X_1+X_3} \qquad \text{----------(4)}$$

From Eq.3 We see that the circuit will oscillate at the resonant frequency of the series combination of X_1, X_2, and X_3.

Using Eq.3 In Eq. 4 Yields

$$-A\beta = \frac{+A_v X_1}{X_2} \qquad \text{----------(5)}$$

Since $-A\beta$ must be positive and at least unity in magnitude,then X_1 and X_2 must have the same sign (A_v is positive). In other words,they must be the same kind of reactance, either both inductive or both capacitive. Then, from Eq.3, $X_3=-(X_1+X_2)$ must be inductive if X_1 and X_2 are capacitive, or vice versa.

If X_1 and X_2 are capacitors and X_3 is an inductor, the circuit is called a colpitts oscillator. If X_1 and X_2 are inductors and X_3 is a capacitor, the circuit is called a Hartley oscillator.

Procedure :

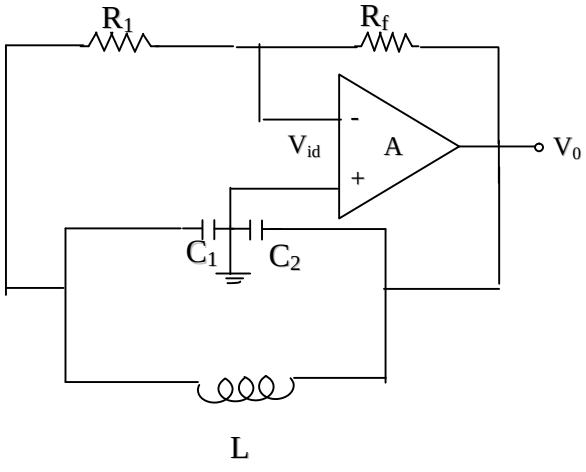


Fig 2: Colpitts oscillator

Before making the connections for the experiment, the functioning of IC should by checking its gain by connecting 741 in inverting configuration as shown in Fig 3.

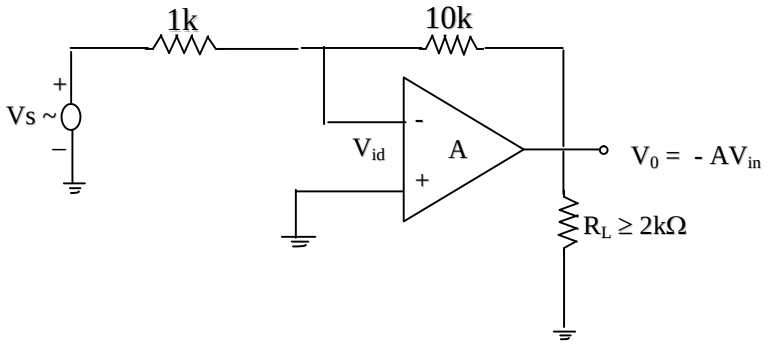


Fig 3: circuit for the proper working of 741mplifier

As shown in the fig 2. an inductance box is connected parallel between the two end terminals of capacitors C_1&C_2. The other two ends of the capacitors are tied together and connected to ground. The ground connection should be made perfectly .The CRO is to be connected between 6^{th} pin(output) and ground. DC power supply power supply with +12V and -12V are connected to 7 and 4 pins. When the connections are made correctly you will see Sine wave output.on CRO screen. The frequency of the wave is determined by following the procedure mentioned in CRO practical. The frequency of the sine wave generated can be varied by changing the inductance in the decade inductance box given. For each value of inductance the frequency of the generated sine wave is determined.

Calculated frequency :

The equivalent capacitance of C_1 and C_2 is obtained by the formula

$$C_{eff} = (C_1.C_2) / (C_1+C_2)$$

By calculating C_{eff} value and the value of inductance kept in inductance box, we calculate the theoretical frequency

$$f_{cal} = 1/ 2\Pi (LC)^{1/2} \text{ Hz}$$

The experiment is repeated for different values of $C_1 = C_2$ .In each case the corresponding frequencies are measured and compared with the calculated values.

We may repeat the experiment with different variable capacitors and compare the Observed and calculated frequencies.

+Vcc = 12V; -Vcc = 12V;

L= ;

Sl No	Capacitance μf	Period T In seconds	Observed frequency	Calculated frequency

C_1 =C_2= 0.01 MFD

Sl .No	Inductance In mH	Period T In seconds	Observed frequency	Calculated frequency

Precautions:
1. The connecting wires should be as small as possible
2. Check the bread board with multimeter before using.
3. Scrap the connecting wires before using.

Result: In Colpitts oscillator the sine wave of different frequencies for various inductance is generated and these frequency values are compared with the calculated values.

Experiment No.8

FIRST ORDER ACTIVE FILTER

Aim: To construct a first order low pass and high pass active filters (using an IC operational amplifier) and to study their frequency response characteristics.

Apparatus: IC 741, signal generator, resistors and capacitors of various values, power supply, C R.O.

Theory:

A filter is an electrical network that prevents some frequencies from appearing at the output. If it allows frequencies starting from zero and up to a frequency called cut-off frequency and prevents all other frequencies above the cut-off frequency it is called a low pass filter. If a network prevents all frequencies starting from zero and up to a frequency called cut-off frequency and allows all other frequencies above the cut-off frequency it is called a high pass filter. A network prevents frequencies starting from zero and up to a frequency and allows a band of frequencies up to a cut-off frequency and prevents all other frequencies above this cut-off frequency is called a band pass filter. The cut off frequency at the lower frequency side is called lower cut-off frequency and the one at higher frequency end is called upper cut-off frequency. The band of frequencies allowed in between f_1 and f_2 is called pass band. And the difference f_2-f_1 is called bandwidth. A band pass filter may be considered as a cascade of high pass filter with cut-off frequency f_1 and a low pass filter with cut-off frequency f_2. A low pass filter with cut off frequency f_1 and a high pass filter with cut-off frequency f_2, connected in cascade will allow frequencies up to f_1 and above f_2. All frequencies between f_1 and f_2 are prevented from appearing at the output. Such filters are called band elimination filters. Filters formed with inductance-resistance combination or capacitance – resistance combination, inductance – capacitance combination are called passive filters. These frequencies attenuate signals even the frequencies in pass band to some extent. If such filters are cascaded, attenuation increases and signal strength falls. It is observed that if including an active component like operational amplifier forms a filter circuit, the signal amplitude can be boosted to the required level. Further the portion below f_1 and above f_2 in a band pass filter can be made steeper. It will reduce the overlapping of neighboring bands in communication systems.

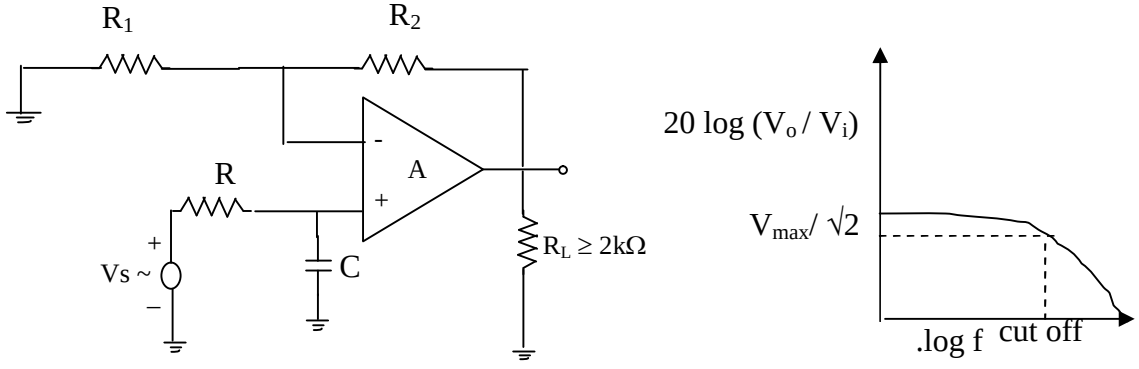
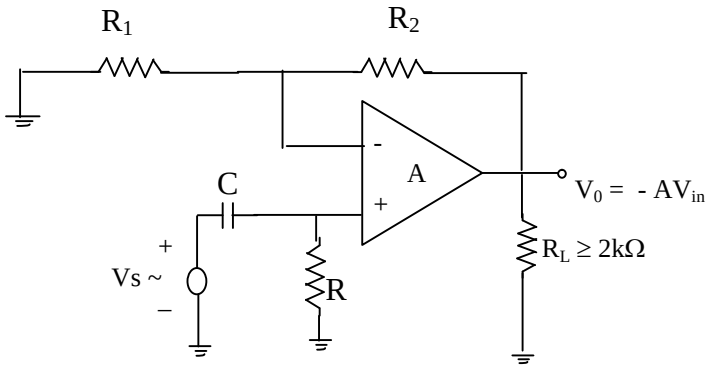


Fig1 Active low pass filter



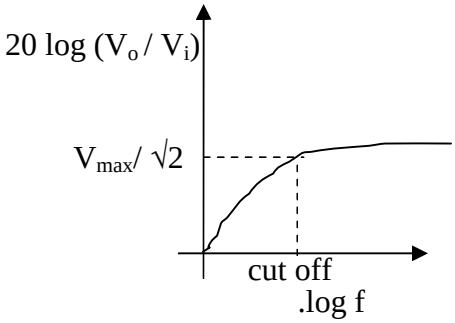


Fig2 Active high pass filter

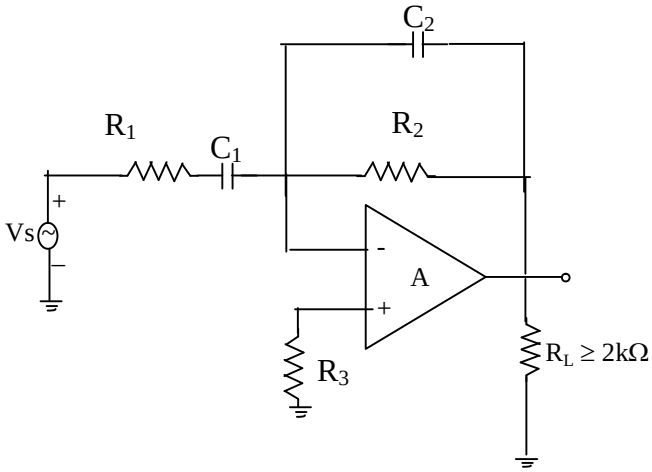


Fig3 Active band pass filter

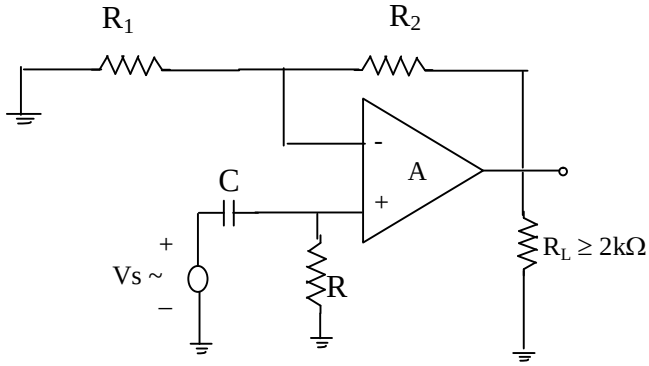


Fig2 Active high pass filter

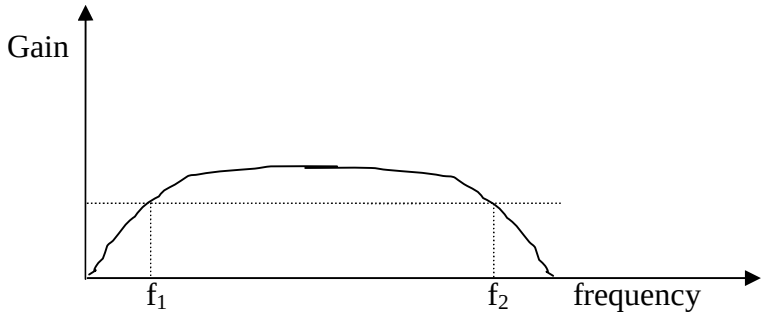


fig 3: Frequency response curve of band- pass filter

Simple RL and RC networks are widely used in filter configurations while these passive filters are quite adequate for many of the applications, they have disadvantage that any ordinary device loads RC low pass filter at low frequencies because of high reactance of capacitor. This interaction has to be taken into account in the design of a multistage filter. It is difficult to achieve no loss condition in the pass band of LC filter because of the heavy inductance used at low frequencies.

Op-amp because of high input impedance and low output impedance provide excellent isolation of different stages of a filter network without actually using inductors. It is possible to realize network functions that can be realized when inductor is used. It is possible to construct active filters with gain. The circuit as shown in fig 1. With a resistor in series with a capacitor is called op amp low pass active filter. The closed loop gain of low frequencies filter is given by.

Where the cut off frequencies $f_{2=}1/2\pi R_2C_2$

The closed loop gain of high pass filter is given by,

An active band pass filter may be realized by combining the two filters. The resulting circuit and the frequencies response as shown in fig 3

Procedure:

Choose a value for the gain. Then R_2= 10Ω and R_1= 1kΩ. Design a low pass filter with f_2 and a high pass filter f_1. Evaluate $C_1$and C_2 using the values of R_1 and R_2 already selected. Vary the frequency of the input signal and measure the corresponding output voltage using a CRO. Change the frequency in suitable steps and measure the output at each setting.

Plot a graph between frequency values and the ratio (v_0/v_i) values. Determine gain and the cut-off frequency of the filter. Compare with design values. Repeat the experiment on a high pass filter. Also repeat the experiment with a band pass filter whose circuit is shown in the fig 3. Compare bandwidth value with design values.

For Low pass filter

Frequency	**Output voltage V_o (Volts)**	**Gain** V_o / V_i	20 log (V_o / V_i)	log f

For High pass filter

Frequency	**Output voltage V_o (Volts)**	**Gain** V_o / V_i	20 log (V_o / V_i)	log f

For Band pass filter

Frequency	**Output voltage V_o (Volts)**	**Gain** V_o / V_i	20 log (V_o / V_i)	log f

From graph:

Low pass filter:

Observed cut off frequency = ; gain =

Calculated cut off frequency = ; gain =

High pass filter :

Observed cut off frequency = ; gain =

Calculated cut off frequency = ; gain =

Band pass filter :

Observed lower cut off frequency = ;

Calculated cut off frequency = ; gain =

Observed upper cut off frequency = ;

Calculated cut off frequency = ; gain =

Precautions:

1) Check the working of 741 OP amp before using it in filter circuit.
2) Keep the input signal amplitude constant through out the experiment
3) Ensure common ground for all ground connections.
4) Some function generators may have offset null facility. Use it if no output signal is observed even after circuit is assembled properly.

RESULT:

First order high pass and low pass filters are constructed with the given cutoff frequency values and their frequency responses are studied. The observed values are agreeing with design values within %.

Exercise:

1) Assemble an active low pass filter and study its gain frequency response curve.
2) Assemble an active high pass filter and study its gain frequency response curve.
3) Assemble an active band pass filter and study its gain frequency response curve.
4) Assemble an active low pass filter to have a cut-off frequency of 4kHz and study its gain frequency response curve
5) Assemble an active high pass filter to have a cut-off frequency of 1kHz and study its gain frequency response curve
6) Assemble an active band pass filter to have a lower cut-off frequency of 1kHz.and to have an upper cut-off frequency of 4kHz. Study its gain frequency response Curve.

.

INTEL 8085 MICROPROCESSOR

Addressing modes:

A microprocessor system operates under the control by a series of instructions that make up a program. An instruction is a command to the microprocessor to perform a specific operation on certain data.

Let us consider the following instructions

a) MOV C, B – This instruction moves the contents of register B into register C.

b) LDA 2050H – Loads the contents of the memory location 2050H into accumulator.

1) Each instruction has two parts: first part indicates the task to be performed and it is called Mneumonic (code in English characters that stand for binary op code) The second part is called operand.

2) In the first example MOV specifies the tasks of instruction (to move the content of one register to another) and is therefore called op code. In the remaining part of the instruction two data registers (source , destination) called operands are specified

In the second instruction, LDA is the op code (in this instruction Acc is one of the operands). A 16-bit number is specified for operand. This number stands for a memory location and represents the second operand. According to INTEL notation if 2050 location is intended the instruction has to be entered in the order as op code, lower byte of address, upper byte of address i.e. 3A 50 20. Here 3A is the hexadecimal equivalent for the mneumonic of LDA op code.

Above examples indicates that each instruction specifies an operation to be performed on certain data. There are certain techniques by which the address of the data to be operated upon may be specified. The various ways of specifying the operand are called the addressing modes.

There are five addressing modes

i) Direct addressing mode ii) Register addressing mode iii) Register indirect addressing mode

iv) Immediate addressing mode v) Implicit addressing mode

1 i) Direct addressing modes: In this mode, the address of the operand is specified directly within the second and third byte of the instruction itself.

Examples:

a) STA 1850H: Load the contents of accumulator into memory location 1850H.

b) LHLD 2015H: Load the contents of memory location 2015H into register L and the contents 2016H into register H.

2 ii) Register addressing mode: In this addressing mode, the source or destination or both operands are located in the microprocessor registers

Examples:

a) MOV A, B – The contents of register B is copied into register A
b) ADD B – The contents of register B is added to the contents of accumulation and the result stored in accumulator.

3 iii) Register indirect addressing mode: In this addressing mode, the address of the operand is specified by a register pair.

Examples:

a) MOV A, M – The contents of the memory location whose address is specified in H-L register pair is copied into accumulator.
b) LDAX B – The contents of the memory location whose address is specified by B-C register pair is copied into register A.

4 iv) Immediate addressing mode: In this addressing mode, the operand is specified in the instruction itself.

Examples:

a) MVI B, 25H – The 8-bit data 25H is moved in to register B
b) LXI H, OF5AH – The 16-bit data OF5AH is moved into H-L register pair such that 5AH is moved into register L and OFH is moved into register H.

5 v) Implicit addressing mode: Some instructions do not require the address of the operand. The operands are implicit in the instruction itself. Most of these instructions operate on the contents of the accumulator.

Examples:

a) CMA – Compliment the contents of accumulator.
b) STC – Set carry flag.

Exercise: Study programs given in material supplied and note the addressing mode of each Instruction in the programs.

INTEL 8085 MICROPROCESSOR

Architecture of 8085microprocessor

Pin configuration of 8085 microprocessor

8085 BUS configuration

The central processing unit of a digital computer built into a single LSI or VLSI chip is called a microprocessor. It is the latest development in the field of computer technology. A VLSI chip contains more than 10,000 transistors. Intel 8085 is an 8-bit NMOS microprocessor. This means the transistors used in the manufacturing of 8085 IC are N-channel Metal Oxide Semiconductor Field Effect Transistors. The 8085 IC has 40 pins and operates on a single +5V DC power supply. Its clock speed is 3MHz. Its memory capacity is 64k Bytes. In the IC, there are 4 functional units: Timing and control unit, to send central signals and synchronize the μp operations with the clock; ALU, to perform arithmetic-logic operations on the data given, Interrupt control unit, to interrupt the execution of program; and serial I/O central unit, to content, the serial form of data into 8-bit, parallel data and vice versa. To reflect the data conditions, 5 Flags are used.

In order to use the microprocessor for a particular application, it has to be connected to various devices. To communicate with the outside world or internal parts, a Microprocessor uses 3 buses namely: address bus, data bus and control bus. The address bus is unidirectional and data bus is bi-directional. Some lines in control bus are input lines and others are output lines. The internal architecture of 8085 Microprocessor determines how and what operations can be performed with data.

The operations are: To transfer 8-bit data, to perform arithmetic and logic operations, to text, for a condition and to sequence the execution of instructions.

The 8085 Microprocessor instruction set, has 74 operation codes that result in 246 instructions. The set includes data transfer, arithmetic, logic, branching and machine control instructions. All these instructions may be one byte, two byte or three byte long. In the instructions, the source and destinations are operands. There are various ways or formats of specifying the operands, which are called the addressing modes. The 8085 microprocessor has 5 types of addressing modes namely, direct, Register, Immediate, Register indirect and Implicit.

Unlike several other manufactures, the INTEL corporation supplied detailed instruction set, technical details and developed a kit for learning about 8085 microprocessor. The 16-bit processors of INTEL namely 8086 and 8088 are similar members with similar internal architecture. IBM adopted INTEL processors as CPUs in its brand of personal computers. Further, all over the world INTEL 8085 is being used as model to acquaint about processors and various interfacing aspects. So, it is still in circulation and it is considered essential to learn about this processor.

Introduction to microcomputer:

Computer being a general-purpose device can be used in different varieties of applications – word processing, computation or controlling industrial process are few of them. But the microcomputers, as the name implies, are small computers having limited capacity, are used for specific applications.

Figure (1) shows a block diagram of a simple microcomputer consists of CPU, memory, input and output devices.

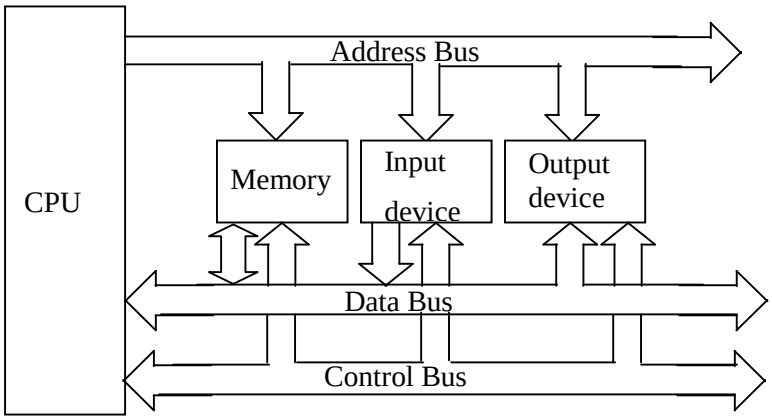


Fig 13.1 Block diagram of a simple microcomputer consists of a CPU

The function of a microcomputer is receiving data and information and processing it. It means performing arithmetic are logical computations on data, store the results or data or information in memory and display the results of the computation.

The means used to receive data are known as input devices. Some of the input devices are: keyboards, switches, mouse etc. A device, which performs computations, is known as arithmetic logic unit (ALU). In microcomputers, a single chip called microprocessors performs this task, together with control of all devices. Some of the well-known microprocessors are: 8085, 8086, Z80, 650Z etc. Storage of data instructions is accomplished using memories. Cassettes, floppy disks, CD ROMs, semiconductor memories are some examples for memories. Presenting of results is done by output devices. Commonly used output devices are monitor (VDU), printers, Seven segment LED displays, LCDs. Figure (1) gives a block diagram of a typical microcomputer.

Various peripherals-I/o devices, memories etc – are connected to the microprocessor by means of three types of buses: Address bus, data bus and control bus. A bus is a group of conducting wires.

Over the address bus, the microprocessor transmits the address of the device, with which it desires to communicate (access). The number of devices or memory locations that a CPU can address is determined by the number of address lines. For example, a CPU with 16 address lines can address 2^{16} or 65,536 or 64K(2^{10} = 1024 = 1K) locations. This bus is unidirectional. It means information can flow in one direction only. The CPU sends the address information on these lines.

Over the data bus, CPU can read in or writes out the data to various memory or I/O devices. Therefore, this bus is bi-directional. Normally data bus widths of sizes 8, 16, 32 etc. are used.

Control bus may consist of some input lines, some output lines and some bi-directional lines depending upon the processor. CPU sends out some control signals like memory read, I/O write etc that are necessary for memory and I/O devices to work.

Architecture of 8085 microprocessor:

The internal logic design of the microprocessor is called its architecture, determines how and what various operations are performed by the microprocessor. As shown in the block diagram (figure 13.2), it consists of arithmetic and logic unit ALU, register section, a timing and control unit and other sections like address / date bus buffers, interrupt control and serial I / O control.

Register Section:

Inside the 8085, there are several internal registers, which are used for executing a program.

a) Accumulator: It is also designated as register 'A' or Acc. It is an 8-bit register used for executing various arithmetic and logical operations. In executing many of the instructions, one of the operands must be in Acc. The accumulated result will also be in Acc, as the name 'Accumulator' implies. For example, during the addition of two 8-bit integers, one of the operands must be in Acc and the result also stored in Acc.

In the I/O mapped I/O mode of data transfer communication with external devices takes place via Accumulator. To send data to output devices in I/O mapped I/O mode the 8-bit data to be sent must be kept in Acc before give OUT instruction. Similarly IN instruction transfers data from input port addressed to Acc.

b) General-purpose registers: The 8085 has six general-purpose 8-bit registers labeled as B, C, D, E, H and L. They can be used individually to store 8-bit information or can be used in pairs to hold 16-bit information or data or address. When used in pairs, only contains combinations viz., B – C,

D – E and H – L are permitted. These registers are programmable, means that a programmer can use them to load or transfer data from the registers by using instructions.

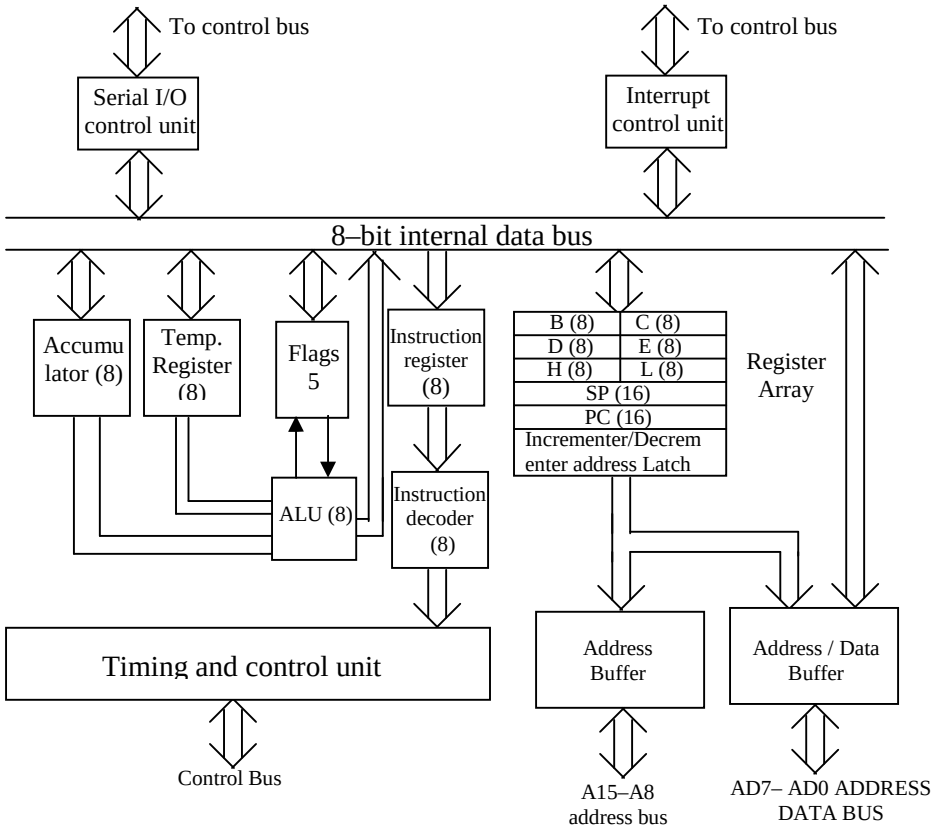


Fig 2 Block diagram of 8085 Microprocessor

c) Program Counter (PC): PC is a 16-bit register used to point the memory address from which the next instruction is to be fetched. Some times it is also called memory pointer. The contents of the PC are automatically incremented by the microprocessor during the execution of the instruction, so that at the end of execution of the present instruction it points to the address of the next instruction in the memory.

The signal of RESET pin of timing and control unit presets the PC to 0000H memory location which is the address of the very first instruction to be executed. The instruction JUMP and subroutine call, also change the contents of PC.

d) Stack Pointer (SP): Stack is a portion of R/W memory set aside by the programmer. The stack is used to save the contents of a register during the execution of a program. Stack pointer holds the address of the top element of the stack. Whenever something is added to the stack, the stack pointer is decremented and whenever something is received from the stack, the stack pointer is incremented. Hence, the SP always points to the top of the stack.

e) Instruction register and instruction decoder: After fetching an instruction code from memory, the microprocessor stores it in the instruction register, which is then decoded by instruction decoder and then microprocessor performs the required operation.

f) Status flag register: The flag register or status register consists of five status flip-flops called flags. Each of these flip-flops holds 1-bit information that indicates certain condition that occurs during an arithmetic or logic operation. The five status flags available in 8085 as shown below are Sign (S), Zero (Z), Auxiliary carry (AC), Parity (P), and carry (Cy).

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
S	Z	X	AC	X	P	X	CY

Fig 3: Bit position of the flags in flag register. (X stands for unused)

Sign flag S – After the execution of a signed arithmetic or logical operation in ALU, if D_7 bit of the result in accumulator is 1, the sign flag is 1 indicating that the result is negative.

Zero flag Z – Zero flag is set to 1, if the ALU operation results in 0, and the flag resets to zero, if the result is not 0.

Auxiliary Carry AC – In an arithmetic operation, when a carry is generated by digit D_3 and passed on to D_4, the AC flag is set.

Parity flag P – If an operation produces even number of 1 in its result, the flag is set to 1. For odd number of 1s in the result, the flag is reset to 0.

Carry flag C – If an ALU operation produces a carry or borrow in its result, the carry flag set to 1, otherwise it is reset.

g) Temporary register: This register is used to hold information from the memory or from the register array of the ALU. The other input to the ALU comes from the accumulator.

h) Address Buffer and Address/Data buffer: Through these buffers microprocessor transfers address as well as data required for memory and input/output devices.

The Arithmetic and Logic Unit (ALU):

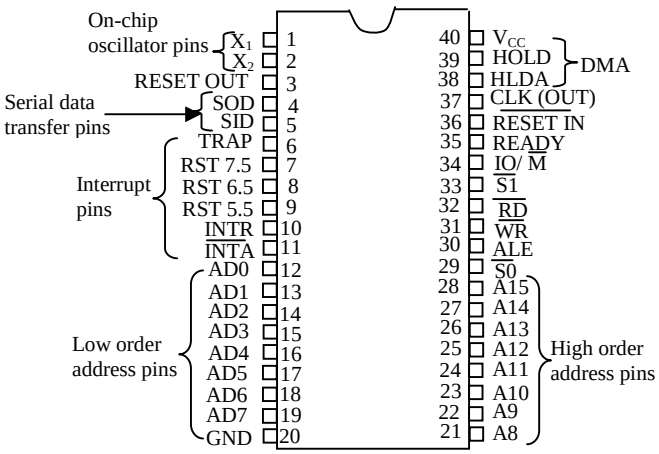
The ALU performs the computing functions like addition, subtraction, logical AND, OR and EX-OR, increment, decrement, complement, compare etc. It includes the accumulator, the temporary register, the arithmetic and logic circuits and flag register. The result is stored in accumulator and flags one set or reset according to the result of the operation.

Timing and Control Unit:

Main function of the microprocessor in the microcomputer system is to perform computations and controlling peripherals like memory and I/o devices. The timing and control unit provides status, control and timing signals, which are necessary for the control of these peripherals.

Pin Configuration of 8085 microprocessor:

The INTEL 8085 is an 8-bit microprocessor fabricated in a 40 Pin DIP package. It operates with single +5VDC supply and with a clock speed of 3 MHz. All the signals corresponding to 40 pins can be divided into six functional groups: 1) Address bus, 2) data bus, 3) Control and status signals, 4) power supply and clock signals, 5) Interrupts and peripheral initiated signals, and 6) serial I/O ports. The pin diagram and signal diagram are shown in Figs15.4 a and 15.4b.



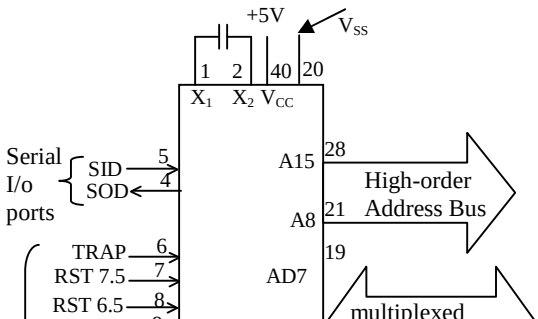
Fig(a) 8085 IC pin diagram

Address Bus:

The 8085 has 16-bit address bus capable of addressing 2^{16} (= 2^6 x 2^{10} = 64 x 1k = 64k) memory locations. The high order 8 address lines $A_8 – A_{15}$ carries high order address of memory through this bus.

Multiplexed Address/data bus:

These 8 lines are time multiplexed low order 8-bit address as well as 8-bit data bus. In the execution of an instruction, during the first clock cycle microprocessor sends out low order 8-bit address and during the remaining clock cycle microprocessor sends out 8-bit data on these pins in multiplexed mode. However, the low – order address bus is separated from data bus using a latch.



Control and status signals:

This group of signals includes two control signals, $\overline{RD}$ and $\overline{WR}$, three status signals, IO/$\overline{M}$, S_1 and S_0, and one special signal ALE. The functions of these signals are as follows:

ALE – Address Latch Enable: A positive going pulse is generated every time when microprocessor sends out address over multiplexed bus. It is used to latch the low-order address from the multiplexed address/data bus on to an octal latch, to generate separate set of eight address lines A_0 to A_1.

$\overline{RD}$ - This active low control signal indicates that the selected I/O or memory device is to be read and data is available on the data bus.

$\overline{WR}$ - This active low write control signal indicates that the data on the data bus, is to be written into a selected memory or I/O device.

IO/$\overline{M}$ - This status signal is used to differentiate between I/O and memory operations. A high on this line indicates an I/O operation whereas a low indicates a memory operation. This signal is combined with $\overline{RD}$ and $\overline{WR}$ to generate I/o and memory control signals.

S_1 and S_0 - These status signals can be used to identify various operations of microprocessor as shown below.

Status signals			Operations
Io/$\overline{M}$	S_1	S_0	
0	1	1	Opcode fetch
0	1	0	Memory Read
0	0	1	Memory write
1	1	0	I/O Read
1	0	1	I/O write
1	1	1	Interrupt Knowledge

Power Supply and clock frequency:

The power supply signals: V_{cc} is +5 volts DC supply and V_{ss} - ground reference X_1 & X_2 – An external crystal connected to these two pins generates a clock frequency by the internal oscillator for the operation of microprocessor. Two divides the oscillator frequency internally; therefore, to operate a system at 3MHz, the crystal should have the frequency of 6MHz.

CLK OUT – This signal can be used as the system clock for other devices.

Interrupts and Externally initiated operations:

Interrupts: When an interrupt is recognized, the next instruction is executed from a fixed location in the memory. The 8085 has five interrupt signals that can be used to interrupt a main program execution. These are TRAP, RST 7.5, RST 6.5 RST5.5 and INTR. The TRAP is a non-maskable interrupt having highest priority among the other interrupts.

INTR – INTR is an interrupt request signal. An interrupt is used by an I/O device to transfer data to the microprocessor without wasting its time.

$\overline{\text{INTA}}$ - $\overline{\text{INTA}}$ is an interrupt acknowledge signal sent by the microprocessor after INTR is received.

$\overline{\text{RESET IN}}$ - It resets the program counter to zero, causes the microprocessor to execute the first instruction at the 0000H location.

READY – Input signal. It is used by slower devices to request the microprocessor to wait until they are ready for data transfer. Microprocessor enters into WAIT states as long as this pin is active.

.6 Serial I/O Ports:

The 8085 have two signals to implement the serial date transmission.

SID – Serial data input line. A serial data bit on this line is loaded into the most significant bit (D7) of accumulator, when RIM instruction is executed.

SOD – Serial data output line. The most significant bit (D7) of accumulator is output on this line when SIM instruction is executed.

13.5 8085 BUS configuration:

The program instructions are stored in memory. To execute an instruction the microprocessor first sends out the address of the memory location on address bus. The address is decoded and the memory location, where the instruction is stored is accessed. The instruction operation code (op code) is fetched through data bus, and placed in instruction decoder. Here the number of bytes further needed is identified and the number of machine cycles needed is determined. Timing and control section issues necessary signals to proceed further with the instruction and the execution of the instruction is completed. These steps are known as "Fetch, Decode and Execute". All these operations are performed in synchronization with system clock. Therefore these operations are performed at a given time period, depending on the clock frequency. During these operations the 8085 signal activations can be visualized by studying its timing diagram. The timing diagrams will give us a clear understanding of how the microprocessor works.

Timing diagram:

Figure 13.5 is the timing diagram of MOV B, A (op code 47H) stored in a memory location say, 3050H. Figure shows the timing activities of various signals like address bus, data bus, $\overline{RD}$, $\overline{WR}$, ALE, IO/$\overline{M}$ etc.

The first one in the figure is the clock waveform. One cycle of this clock is called a state labeled as T.

A basic microprocessor's operation such as reading a byte from memory or writing into a port is called a machine cycle. The time a microprocessor requires to fetch and execute an entire instruction is known as an instruction cycle. An instruction cycle may consist of one or more machine cycles.

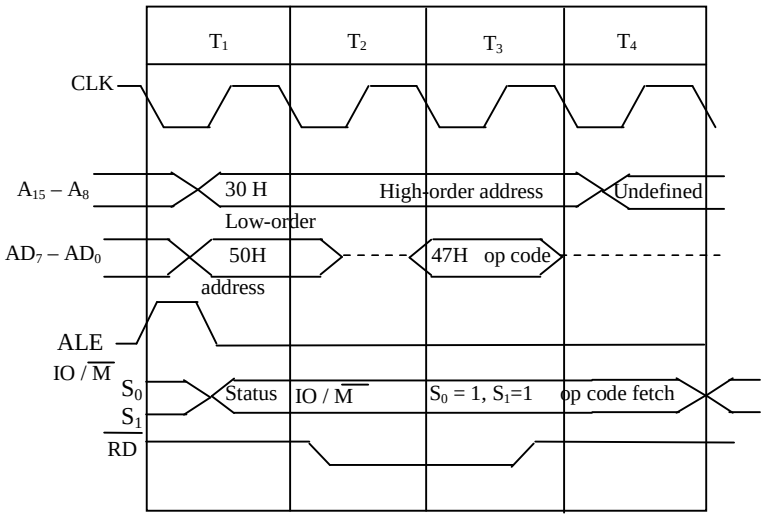


Fig. MOV B,A timing diagram

The sequence of steps involved in the execution of an instruction MOV B, A having the operation code 47H stored in memory location 3050H are as follows (See figure 13.5):

Step 1 – The microprocessor places the contents of program counter 3050H on the address bus, such as 30H on the high order address bus and 50H on the low – order AD_0-AD_1 bus.

Step 2 – ALE goes high indicating $AD_0 – AD_7$ bus contains low order address. This signal is used to store lower byte of address in an octal latch for further use as AD bus gets converted to data bus from second clock cycle.

Step 3 – IO/$\overline{M}$ goes low to indicate the operation relates to memory.

Step 4 – Both S_1 and S_0 goes high to tell the operation is op code fetch. All these operations from step 1 to step 5 is performed in T_1 state.

Step 5 – During T_2, $\overline{RD}$ goes low since it is memory read operation, the program counter is incremented by one i.e.3051H, ALE goes low . The content of memory location 3050H, which is 47H, the op code of MOV B, A is placed on the data bus.

Step 6 – In T_3 state, 47H is read by microprocessor and transferred instruction decoder.

Step 7 – During T_4, microprocessor decodes the instruction and executes the specified operation i.e. transferring of accumulator contents into register B.

Above figure shows a complete machine cycle (in this case it is also equal to one instruction cycle) consisting of four states. If the clock frequency (f) is 2 MHz, the time for one clock period is

$$T = \frac{1}{f} = \frac{1}{2\times10^{6}} = \frac{10^{-6}}{2} = 0.5\mu s$$

$\therefore$ Total execution time = T- states **x** clock period = 4 x 0.5 x 10^{-6} = 20μs.

Suggestion to teachers:

1) It is essential that student understands the architectural details and the purpose of signals at various pins.
2) Student must be in a position to appreciate the significance of the status of signals generated at various clock cycles in a given instruction cycle.
3) Student must be in a position to identify the various ICs and various components used in microprocessor kits.
4) Students must be made to understand the concept of various types of memory and accessing it.

INTEL 8085 MICROPROCESSOR

Various groups of Instructions:

It is clear from the previous studies that the 8085 have a collection of 8-bit and 16-bit registers and 8-bit memory locations that can be used for programming purpose. The collection of these register and memory locations used by the programmer is called programmer's model. The programmer's model of 8085 consists of

i) 8-bit Accumulator ii) Six 8-bit registers: B, C, D, E, H & L or three 16-bit registers: B-C, D-E & H-L .iii) 16-bit program counter PC iv) 16-bit stack pointer SP v) 8-bit flags register vi) 64KBmemory locations divided into ROM, RAM and stack areas and vii) 64 I/O ports.

An instruction is a command to the microprocessor to program a specific task. The entire group of instructions is called its instruction set. The 8085 instruction set has 74 op-codes that result in 246 instructions. Depending on their function all these instructions can be classified into the following categories:

i) Data transfer group ii) Arithmetic group iii) Logical group iv) Branch group

v) Stack, I/o and machine – control group.

i) Data transfer group: These instructions copy data from a source into a destination without modifying the contents of the source and without affecting the flags.

8-bit data transfer as

1) MOV destination, source – Copies the contents of source into destination where the source may be
   - an immediate value (e.g.# 35H)
   - an 8-bit register (A, B, C, D, E, H & L)
   - the contents of a memory location whose address is specified in H-L reg. Pair

   The destination may be
   - an 8-bit register (reg.)
   - the contents of a memory location whose address is specified in H-L reg. pair. But both source and destination should not be memory contents and the destination should not be an immediate value.

Examples:

Between registers:

MOV B, C – data of register C is copied into register B.

Specific data byte into a register or memory

MVI B, 2FH – data 2FH copied into register B

MVI M, 2FH – data 2FH into a memory location whose address is specified by H-L pair (Before this instruction H-L pair should be loaded with a memory address)

Data transfer between memory location and a register

MOV M, B – copy the contents of reg. B into a memory location address by H-L pair.

MOV B, M – copy the contents of memory location address by H-L pair into reg. B.

LDA 16-bit address: Load accumulator directly the contents of memory location whose address is specified in the second and third byte of instruction.

e.g.: LDA 2050 H – Load the contents of memory location 2050H into the acc.

STA 16-bit address: Store accumulator direct.

The contents accumulator is stored into a memory location whose address is specified with second and third byte of instruction.

STA 30A0H – Store accumulator content directly into memory location whose address is 30A0H.

LDAX rp: Load acc . indirect rp = B – C / D – C.

The contents of memory location whose address in the register pair rp is loaded into the acc.

STAX rp: Store acc. Indirect rp = B – C / D – E.

The contents of acc. is stored in memory, whose address is in the register pair rp.

e.g.: STAX B – Store the contents of acc. Into the memory location addressed by B – C.

16-bit data transfer:

1) LXI rp. data 16: Load register pair immediate.

rp = B – C / D – C | H – C | sp.

The 16-bit immediate data is loaded into the specified register pair.

e.g. LXI H, 2450H – Loads 2450H into HL pair such that 50H is in L and 24H in H.

e.g: LDAX B – This instruction will load the contents of the memory location, whose address is in B – C reg. Pair into acc.

2) LHLD 16-bit addr. : Load H – L pair direct.

The contents of memory location, whose address is specified in second and third byte of instruction is loaded into register L. The contents of the next memory location are loaded into register H.

e.g: LHLD 3050H – This instruction loads the contents of 3050H into register L and the contents of 3051H into register H.

3) SHLD 16-bit addr: Store H – L pair direct.

The contents of register L is stored into the memory location whose address is specified in the instruction, and the contents of register H is stored in the next memory location.

e.g: SHLD 3500H – This instruction will store the contents of register L in the memory location 3500H and the contents of register H is the location 3501H.

4) XCHG: Exchange the contents of H – L with D – E.

The contents of register pair H – L are exchanged with that of the register pair D – E.

e.g. if DE=2050;HL=307F ; after the execution of XCHG instruction DE=307F; HL=2050

ii) Arithmetic group:

These instructions perform arithmetic operations such as addition, subtraction, increment and decrement.

i) 8-bit addition and subtraction: The 8085 can execute two types of addition / subtraction instructions – addition or subtraction with out carry and addition or subtraction with carry. Both addition instructions can be used to add the contents of accumulator to the contents of one of the general purpose registers or the contents of memory locations (address in H – L pair) or all immediate data byte.

a) These instructions assume that the acc. is one of the operands.
b) Place the result in the acc.
c) Do not affect the contents of another operand and
d) Modify all the flags according to the data condition in acc.

e.g.1: ADD B – Adds the contents of register B with the contents of the accumulator and the result is stored in the acc. If a=05H and B-0 7H ,execution of ADD B modifies the content of A as 0CH. Reg B content is not altered. As 8085 handles data in binary in the display we see the hexadecimal equivalent of result OCH .Don't expect the decimal answer of 12.

e.g.2: ADD M – The contents of the memory location addressed by the pair is added to the contents of acc. one result is stored in the acc.

e.g.3: ADI 35H – The 8-bit immediate data 35H is added to the contents of the accumulator and the result is stored in acc.

e.g.4: ADC D – The contents of register D and carry flag are added to the contents of acc. and result is stored in the acc.

e.g.5: ADC M – The contents of memory location addressed by H – L and carry are added to the contents of the acc and the result is stored in the acc.

e.g.6: ACI 35H – 35H with carry are added to the contents of acc.

e.g.7: SUB B – The contents of register B is subtracted from the contents of the acc. and the result is stored in the acc.

e.g.8: SBB M – The contents of the memory location addressed by H – L pair and carry are subtracted from the contents of acc. And the result is stored in the acc.

ii) 16-bit addition:

This instruction adds 16-bit data of H – L register to the 16-bit data of another 16-bit register contents. Result is stored in HL pair. Except carry no other flags are effected.

DAD rp: Add the content of register pair specified to the content of H-L pair and store the result in H-L pair.

rp = B-C / D-E / H-L / sp.

The contents of register pair r_p are added to the contents of H – L pair and the result is stored in H – L pair.

e.g.: DAD D – The contents of register pair D – E are added to the contents of HL pair and the result is stored in H – L pair. If the result is more than 16-bits carry flag is set. No other flags are affected.

iii) 8-bit increment/decrement:

These instructions increment or decrement the contents of 8-bit registers or the contents of memory location whose address is specified in H – L register pair.

Carry flag is not affected.

e.g.: INR D – The contents of register D is incremented by one.

INR M – The contents of memory location addressed by H – L pair is incremented by one.

DCR C – The contents of register C is decremented by one.

DCR M – The contents of the memory location addressed by H – L pair is decremented by one.

iv) 16-bit increment/decrement:

1N x rp: Increment register pair contents

rp = B – C / D – E / H – L / sp.

e.g.: 1NX H – The contents of the H – L pair is incremented by one. No flags are affected.

DCX rp: decrement register pair contents.

rp = B – C / D – E / H – L / sp.

e.g.: DCX B – The contents of the B – C pair is decremented by one. No flags are affected.
However it becomes necessary in some programs to check whether the execution of DCX rp resulted in zero or not . After each decrement by checking the logical OR result of the registers involved one can ascertain it See how this is done in program no. .

v) Decimal adjusting data:

DAA: Decimal adjusting accumulator.

DAA instruction is used in the program after an addition instruction. The DAA instruction operates on the result in the accumulator and the final result in decimal form. It uses carry and auxiliary carry for decimal adjustment. To use the DAA instruction the two numbers added should be in BCD form.

iii) Logical Instruction:

These instructions perform logical operations such as AND, OR, EX-OR, compare, rotate and complement of data in register or memory.

1) Logical OR, AND, NOT and EX-OR operations.

a) These instructions implicitly assume that the accumulator is one of the operands
b) Place the result in accumulator.
c) Do not affect the contents of the operand reg. (except NOT operation)
d) Some of the flags are affected according to the result.

e.g.1: ORA D – The contents of the register D is logically OR ed with the contents of the accumulator and the result is stored in the accumulator.

e.g.2: ANI 23H – The data 23H is logically AND ed with the contents of the accumulator and the result is stored in the accumulator.

e.g.3: XRA M – The contents of the memory location addressed by H – L pair is EX – OR ed with the contents of the accumulator and the result is stored in the acc.

e.g.4: CMA – The contents of the accumulator is complemented and the result stored in accumulator.

2) Comparison:

The contents of the accumulator is compared with the contents of the other register or the contents of the memory location addressed by H – L pair or an 8-bit data.

CMP R – Compare register with accumulator

CMP M – Compare memory contents with accumulator.

CPI 8-bit data – Compare immediate data with accumulator

Here comparison is performed through subtraction. However, neither contents are modified, but the result of the subtraction is only reflected in the form of conditional flags.

i) If the content of A is lesser than compared data , carry flag set 1.

ii) If the content of A is greater than compared data, carry flag is zero.

iii) If both are equal, zero flag set to 1.

3) Rotate Instructions:

a) The rotate instructions are register specific, operates only on the accumulator

b) In these instructions, the bit that is shifted out is used as the new bit shifted in

c) There are four rotate instructions to rotate a bit pattern in the accumulator to the left or to the right with or without carry.

RLC – rotate accumulator left

RRC – rotate accumulator right

RAL – rotate all (including carry) left

RAR – rotate all (including carry) right.

iv) Branching Instructions:

The flow of a program proceeds sequentially, from instruction to instruction, unless a control transfer command is executed. The branch instructions allow microprocessor to go to a different memory location, either unconditionally or conditionally (under certain test conditions) and μρ continues executing instructions from that new location.

The branch instructions are classified in three categories:

- Unconditional/conditional jump instructions.
- Unconditional / conditional call and return instructions
- Restart instructions.

Jump Instructions:-

a) Unconditional jump

JMP addr – The program counter is loaded with 16-bit address and the program execution jumps to that address unconditionally.

b) Conditional jump

conditional jump instructions first check for the condition and if the condition is satisfied then only execution jumps to new location. Flag(I), sign flag(S), and purity flag (P).

e.g.1: JC addr – Jump on Carry

JNZ addr – Jump on No zero

JPE addr – Jump on even parity.

Call and Return instructions:-

The call instruction is used in the main program to call a subroutine, and the return instruction is used at the end of the subroutine to return to the main program. When a subroutine is called, the contents of the program counter, which is the address of the instruction following the call instruction, is stored in the stack and the program execution is transferred to the subroutine address. When the return instruction is executed at the end of the subroutine, the memory address stored in the stack is retrieved, and the sequence of execution is resumed in the main program.

a) Unconditional call

CALL addr – unconditional call

b) Conditional call:

CC addr – call if the carry flag set

CNC addr – call of the carry flag not set

CNF addr – call on No zero

a) Unconditional return

RET – unconditional return

b) Conditional return

RC – Return on carry

RNC – Return on No carry

RP – Return on positive

Restart Instructions:

Restart is a one byte CALL instruction. Because these instructions are having specific call addresses for specific restart instructions as specified bellow. The program jumps to the instruction starting at restart location.

Restart Instruction	Location
RST 0	0000H
RST 1	0008H
RST 2	0010H
RST 3	0018H
RST 4	0020H
RST 5	0028H
RST 6	0030H
RST 7	0038H

One interesting use of RST instruction is to create break point. Finding mistakes(bugs) in a big program can be made simple by inserting break points at desired places. Program terminates when RST instruction is executed. If there are no bugs up to that point, the break point can be shifted to some other place . The process of removing bugs is called debugging. Instead of HALT instruction one may use Break point for the safe transfer of control to monitor program.

v) Stack, I/o and machine – control group:

a) Stack instructions

PUSH rp – PUSH register pair onto stack

rp = B – C / D – C / H – C / psw.

The content of the specified register pair is pushed onto stack after the stack pointer is decremented by two.

e.g.: PUSH B – The contents of register pair B – C is pushed on to stack.

POP rp – POP off stack to register pair.

rp = B – C / D – C | H – C | psw.

The top two elements of the stack are POP ed into the specified register pair. The stack pointer is incremented by two.

e.g.: POP D – The contents of stack top POP ed into register pair D – E.

In some programs the available internal registers may not be sufficient to write the program. However some of the registers may not be needed immediately. In such situations the contents of immediately not used register contents are pushed on to the stack. These registers are used for the present need and once the purpose is served, they are reloaded with their earlier values with POP instruction. If a series of push instructions like PUSH B, PUSH D, PUSH H are used, to restore their original contents the POP instructions must be in the order POP H, POP D, POP B as stack operations follow first in last out policy.

b) I/O Instructions

IN port-addr – Input to accumulator from input port

The data available on the port is moved into the accumulator. Port address is 8-bit.

OUT port-addr – Out put from accumulator to output port. The contents of the accumulator is moved to the port whose 8-bit address is specified in the instruction

External devices with suitable interfacing can be connected to input and output ports. The power of microprocessor lies in its ability communicating with external devices. Industrial digital control systems, Robots and other smart devices work under microprocessor control. All together

256 I/O devices can be connected . The devices which use IN and OUT instruction for data transfer are said to be operating in I/o mapped I/o mode. If more than 256 devices are to be connected , Memory mapped I/O mode can be used. In this mode each device is assigned with a 16 bit address.

c) Machine control instructions

These instructions are used to control the microprocessor operation.

e.g.: HLT – Halt

The execution of the HLT instruction stops the microprocessor.

Suggestion to teachers : Students must be trained to appreciate the use instructions in various groups of instructions. Small problems may be given for writing programs and after ascertaining

That students developed good background they may be asked to write the programs for bigger problems. Students have to be at ease while writing programs and should not get by heart the programs given as examples. Students cannot get good programming experience by doing the four practicals given in the list. They have to be taken as examples only and teachers can give much better problems than these.

Experiment No. 9

DATA TRANSFER PROGRAM

Aim: To write an 8085 assembly language program to transfer data from one memory area to another area.

Apparatus: 8085A Microprocessor kit and SMP with 5V DC current 3A. and ±12V DC 100mA Outputs.

Method: The method involves simple algorithm given below

1. Initialize the source memory pointer (mem ptr)
2. Initialize the destination pointer (dest ptr).
3. Initialize a byte counter
4. Increment the source and destination memory pointers
5. Transfer the first byte of source data to destination location pointed to by destination pointer. As there is no direct memory to memory instruction for 8085, transfer content of source location is to accumulator or some other general-purpose register and then transfer it to destination location.
6. Decrement the byte counter. Check the content of the byte counter to see whether it zero or not by verifying the status of ZERO FLAG. It is done using JNZ or JZ instructions. If byte counter (content) is not zero transfer the control to step 4 to repeat transfer operation .If the byte counter is zero, it means all the bytes are transferred from source area to destination area. Go to step 7.
7. Halt or break the program execution and transfer the control to monitor.

To convert this algorithm to a form compatible for 8085A kit we have to define the memory as follows:

Defining program & data area:

Program area: It starts from the address XX00. XX stands for upper byte of address. It is defined by the Microprocessor kit manufacturer. To know about this one has to refer to the technical manual that is supplied along with the kit. In giving examples programs commence from C000.

The data area: Data area starts from the C051. The number of bytes of data available for transfer is given in location C050. The destination area for the transfer of data commences from C071 (It can be changed)

Stack memory area: The present program doesn't use stack.

With these locations defined as above the modified algorithm will be as follows:

Algorithm: The data transfer can be affected in several ways. Algorithm for one such method is given below.

1. Initialize the source memory pointer by loading the address C050 in the HL register pair.
2. Initialize the destination pointer is by loading C070 in DE register pair.
3. Initialize a general purpose register (say C) as byte counter by moving the content of C050 location to C register. It is needed for loop operation.
4. Increment the source and destination memory pointer to C051 and C071 respectively.
5. Transfer a byte of source data is to Accumulator
6. Transfer the content of accumulator to its destination.
7. Decrement the byte counter.
8. If byte counter (content) is not zero go to step 4 to repeat transfer operation .If the byte counter is zero Go to step 9.
9. Halt or break the program execution and transfer the control to monitor.

The problem of transferring a block of data from one memory area to another memory area can be presented by means of a flow chart for better understanding as given in Fig 1

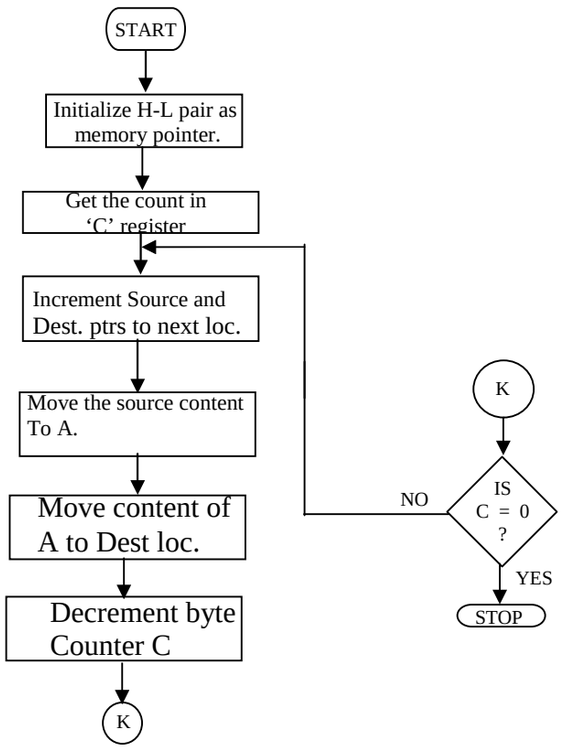


Fig 1

K in circle is used to indicate continuation of flow chart. It is usually done when flowchart Extends from one page to another. Here it is done to save space in the page.

Address	Label	Mnemonics	Operands	machine Code	Comments
C000		LXI	D, C 070	11, 70, C0	: Initialize dest.ptr with C070
C003		LXI	H, C050	21, 50, C0	: Initialize source ptr with C050
C006		MOV	C,M	4E	: Initialize byte counter
C007	loop :	INX	H	23	: Increment source pointer
C008		INX	D	13	:Increment destination pointer.
C009		MOV	A,M	7E	: Move the content of memory location pointed to by source pointer to accumulator(ACC)
C00A		STAX	D	12	: Move the content of ACC Memory location pointed to by destination pointer
C00B		DCR	C	0D	: decrement byte counter by one.
C00C		JNZ	loop	C2 07 C0	: If C is not equal to zero then go to location labeled loop

C00F HLT /Break EF : Halt/ transfer control to monitor.

Procedure:

Identify the RAM area in the kit by referring to the manual given. Identify the Exam Mem/load Memory Key. When this key is pressed Seven-segment display (SSD) of address field (one for each digit of 4-digit address) and data field becomes blank. Enter starting address of program in address field (EX C000) and press NEXT key. Control transfers to data field. Enter data pertaining to the address given (EX 11). Press NEXT key. Monitor increments the address displayed to next address (Ex C001). Enter data corresponding to that address (EX 70). Press NEXT and go on entering data given in column 4 until all the bytes are entered. Do not forget to press NEXT key after each entry of byte of data.

After entering the program verify whether all the bytes are entered as is expected. Press RESET key and EXAM MEM keys one after the other. Enter starting address of source data memory (EX C050) And press NEXT key. Enter corresponding data (EX 05). Press NEXT key. Address increases. Enter data corresponding to that location EX (01). Repeat the procedure until all the data bytes are entered. Before execution of program the destination area may contain any thing called junk data. It is represented by XX. (See table 1).

Sample: Data before execution of the program

TABLE 1

Source Memory Location	data	Dest memory location	data
C050	05	C070	xx
C051	01	C071	xx
C052	02	C072	xx
C053	03	C073	xx
C053	04	C074	xx
C055	05	C075	xx

After entering data ensure whether data was entered correctly or not. After entering the program and data correctly, press RESET key. Press GO key. Monitor transfers the control to some address in permanent memory area .Now enter starting address of the program (EX C000). Press EXEC key. Display changes to the standard logo of kit manufacturer that is displayed whenever RESET key is pressed. Now press EXAM MEM key and enter address of first location in destination area (EXC071), in address field. If the program was written correctly one sees 01(the program is expected to transfer content of the first location in source area (here 01) to first location in destination area). Pressing NEXT key displays next address and its data
(EX C072 02). Verify whether all the bytes are transferred).

Data after execution of the program:

TABLE 2

Source Memory Address	data	Destination Mem address	data
C050	05	C070	XX
C051	01	C071	01
C052	02	C072	02
C053	03	C073	03
C074	04	C074	04
C075	05	C075	05

RESULT:

After execution of the program the data given in table is transferred from memory area starting from C051 to destination area starting from C071 as shown in table 2.

LIMITATIONS: This program can transfer a maximum no. of 256 bytes data only.

PRECAUTIONS: 1. The operational codes should be given correctly.
2.the program must be checked before the execution of program.

Suggestions to teachers: As execution of programs take less amount of time give good number of exercises to students in class room and

1) Encourage the students to write their own programs
2) Make students use their own data without repeating others data.
3) Make students write their program from other starting address.
4) If microprocessor kits of different makes are available let them run their Programs on those kits after making needed modifications.
5) Encourage students to read program given in machine code and write Corresponding assembly language instructions.
6) Encourage students to write the programs using variety of instructions so that The program occupies less number of bytes.
7) Encourage students to write the programs using variety of instructions so that the program takes less execution time.

Exercises:

1) create source data of 10 (Decimal) bytes; make suitable modifications in the given program and execute it and verify the destination area.
2) create source data of 10 (Hexadecimal) bytes; make suitable modifications in the given program and execute it and verify the destination area.
3) make modifications in Ex2 so that data is copied from source data to destination in reverse order.
4) make suitable modifications to program in Ex2 to copy source data to destination if the source area and destination area overlap.
5) modify the program in Ex2 using LDAX rp and STAX rp instructions.
6) Write program of your own using some other register as byte counter.
7) Write a program to copy a byte of data in 12 locations in destination area commencing from XX81 location.
8) Study the program given at the end and include suitable remarks in remarks column.
9) Write a program to interchange the contents data given in two memory blocks. other than the one given below.
10) Write a program that can transfer more than 256 bytes from source to destination area.

PROGRAM FOR ----------
(See Ex 8)

ADDRESS	LABEL	MNEMONICS	OPREANDS	MACHINE CODE	COMMENTS/ REMARKS
XX00		LXI	H XX 50	21 50 XX	
XX03		LXI	D XX 70	11 70 XX	
XX06		MOV	C,M	4E	
XX07	loop :	INX	H	23	
XX08		INX	D	13	
XX09		MOV	B,M	46	
XX0A		LDAX	D	1A	
XX0B		MOV	M,A	77	
XX0C		MOV	M,B	78	
XX0D		STAX	D	12	
XX0E		DCR	C	0D	
XX0F		JNZ loop		C2 07 XX	
XX12		RST 5		EF	

Experiment No.10

LOGICAL GROUP OF INSTRUCTIONS

Aim: Writing programs illustrating the use of programs
1) To find how many negative numbers in a given list

Apparatus: 8085A Microprocessor kit and SMP with 5V DC current 3A and ±12V DC 100mA Outputs.

Method: The method involves simple algorithm given below

In signed binary code most significant bit (d7) is treated as sign bit. If it is 1 the Number is a negative number If it is zero it is a positive number. So the method Involves identifying the most significant bit and testing its status. It may be done as follows

1. Initialize the source memory pointer
2. Initialize the destination pointer.
3. Initialize a byte counter. Initialize another register as number counter.
4. Increment the source and destination memory pointers
5. A (first) byte of source data is transferred to A register and tested for its sign.
6. If it is positive go to step 8 else continue.
7. increment number counter. Move the number to destination area. Increment location address in destination area.
8. Increment source address. Decrement byte counter. Check whether the content of the byte counter is zero or not by verifying the status of ZERO FLAG. It can be done using JNZ or JZ instructions. If byte counter (content) is not zero the control is transferred to step 5 to repeat the operation .If the byte counter is zero, it means all the bytes are transferred from source area to destination area. Go to step 9.
9. Halt or break the program execution and transfer the control to monitor.

To convert this algorithm to a form compatible for 8085A kit we have to define the memory as follows:
Defining program & data area:
Program area: It starts from the address XX00. XX stands for upper byte of address. The Microprocessor kit manufacturer defines it. To know about this one has to refer to the technical manual that is supplied along with the kit. In giving examples programs commence from C000.
The data area: Data area starts from the C051. The number of bytes of data available is given in location C050. The destination area for the data commences from C071
(It can be changed)
Stack memory area: The present program doesn't use stack.

With these locations defined as above the modified algorithm will be as follows:

Algorithm: The listing of negative numbers can be affected in several ways. Algorithm for one such method is given below.

1.The source memory pointer is initialized by loading the address C050 in the HL register pair.
2.The destination pointer is initialized by loading C070 in DE register pair.
3.A general purpose register (say C) is initialized as byte counter by moving the content of C050 location to C register. It is needed for loop operation.
4. Initialize B reg as negative number counter with 00.
5.Increment the source and destination memory pointer to C051 and C071 respectively.
6. A byte of source data is transferred to Accumulator
7. The most significant bit is isolated. The most significant bit (msb) of number in accumulator is tested for sign.
8. If it is positive go to step
9. If it is negative increment number counter.
10. Increment destination pointer
11. Move the content of source location to destination location
12. Increment source address.
13. The byte counter is decremented.
13. If byte counter (content) is not zero go to step 6 to repeat operation .If the byte counter is zero Go to step 14.
14.Halt or break the program execution and transfer the control to monitor.

The problem of identifying negative numbers and copying them into destination area can be presented by means of a flow chart representation for better understanding as given in Fig 1

An assembly language program for listing negative numbers is shown in Fig2. This is not the only way to write a program to list negative numbers. One may use different instructions to achieve the same purpose or they may follow different logic. An alternative program to list negative numbers is shown in fig 3.

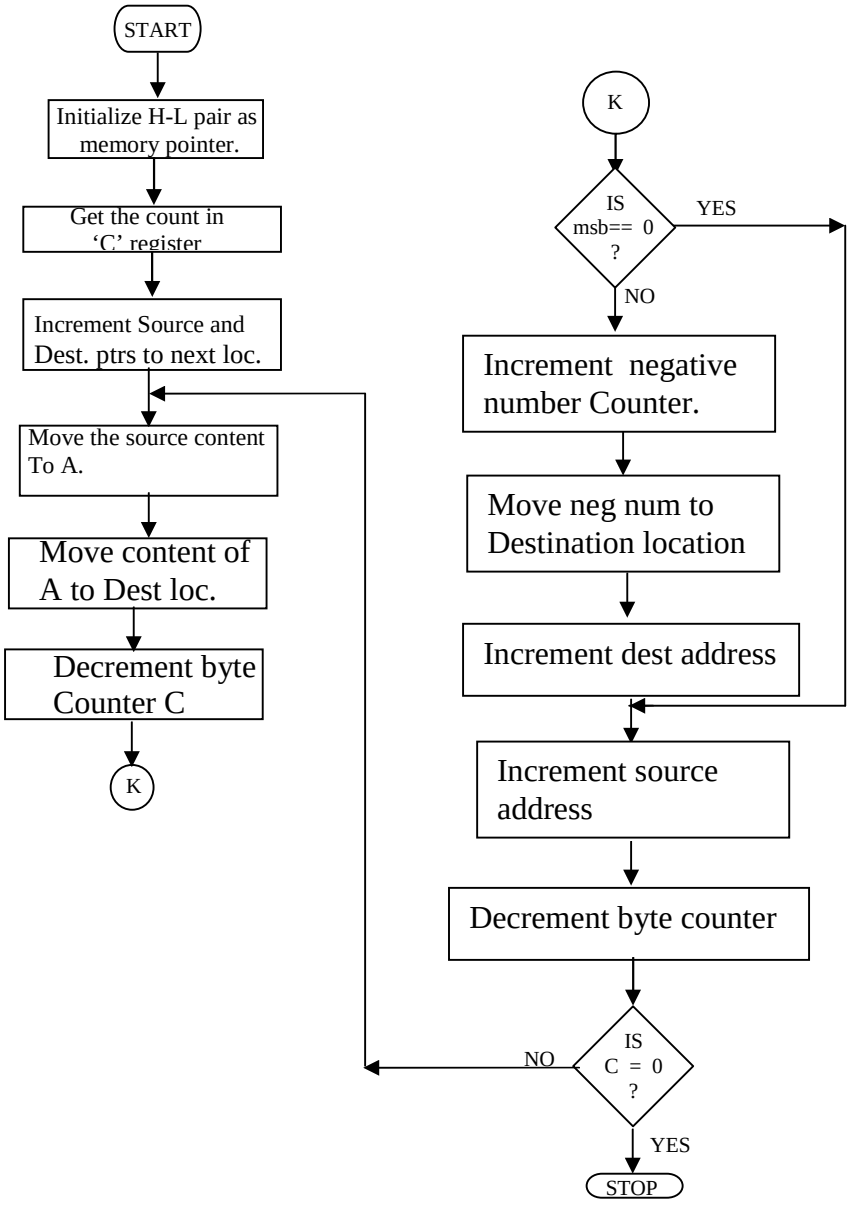


Fig 1

K in circle is used to indicate continuation of flow chart. It is usually done when flowchart Extends from one page to another. Here it is done to save space in the page.

Note: There may be some omissions in op codes, data jumps to wrong addresses etc. in the Programs. The student has to study the programs and logic correct the mistakes and then execute them

Address	Mnemonic	Operand	Op-code	Comments
C000	LXI	H C050	21 50 C0	:Initialize the source area .
C003	LXI	D C070	11 70 C0	: Initialize destination area
C006	MOV	C,M	4E	:Initialize the byte counter.
C007	MVI	B , 00	06 00	:initialize the data counter.
C009	INX	H	23	:Increment the source area .
C00A	INX	D	13	: Increment dest address
C00B loop:	MOV	A,M	7E	: move source loc content to ACC
C00C	ANI	80	E6 80	: isolate msb of ACC content
C00E	JZ (forward)		CA 15 C0	:jump on zero to location Forward.
C011	INR	B	04	:Increment data counter.
C012	MOV	A,M	7E	copy neg. num to ACC
C013	STAX	D	12	save it in dest. loc
C014	INX	D	13	inc . dest. pointer
C015 forward:	INX	H	23	inc. source ptr.
C016	DCR	C	0D	:Decrement byte counter
C017	JNZ loop		C2 0B C0	:If c<>o go to step labeled loop
C01A	RST	5	EF	:Halt/ Break.

We can also use rotate group instructions as shown in program given below to list the negative numbers .

Address	Mnemonic	Operand	Op-code	Comments
C000	LXI	H C050	21 50 C0	:Initialize the source area .
C003	LXI	D C070	11 70 C0	: Initialize destination area
C006	MOV	C,M	4E	:Initialize the byte counter.
C007	MVI	B , 00	06 00	:initialize the data counter.
C009 loop:	INX	H	23	:Increment the source area .
C00A	NOP		00	
C00B	MOV	A,M	7E	
C00C	RLC		07	:Rotate left by one.
C00D	JZ (forward)		CA 14 C0	: if carry bit is zero go to Location labeled Forward.
C010	INR	B	04	:Increment data counter.
C011	MOV	A,M	7E	
C012	STAX	D	12	
C013	INX	D	13	
C014 forward:	NOP		00	
C015	DCR	C	0D	:Decrement counter

C016	JNZ loop		C2 09 C0	:If c<>o go to back steps and continue till C=0
C019	MOV	A,B	78	:transfer memory to Acc.
C01A	STA	C080	32 80 C0	:store the value in C080 address.
C01D	RST	5	EF	:Halt.

Source data		Destination area		
			Before	After execution
C050	05	C071	xx	89
C051	12	C072	xx	A4
C052	89			
C053	41			
C054	A4			
C055	21			

Result:
After executing of the program the number of negative numbers in the given list available in register B is
Limitations:
The program can handle a list having numbers up to 256 bytes data only.

2) To identify the odd numbers and write them separately in another memory area.

An odd number is identified by testing the status of least significant bit of the number present in Accumulator. If least significant bit (lsb) is zero, it is an even number. if lsb is 1 the number is an odd number. Following similar algorithms modify the programs given for finding negative numbers to suit this problem.

Address	Mnemonic	Operand	Op-code	Comments
C000	LXI	H C050	21 50 C0	:Initialize the source area .
C003	LXI	D C070	11 70 C0	: Initialize destination area
C006	MOV	C,M	4E	:Initialize the byte counter.
C007	MVI	B , 00	06 00	:initialize the data counter.
C009	INX	H	23	:Increment the source area .
C00A	INX	D	13	: Increment dest address
C00B loop:	MOV	A,M	7E	
C00C	ANI	01	E6 01	: isolate lsb.
C00E	JZ (forward)		CA 15 C0	: if lsb is zero jump go to Forward.
C011	INR	B	04	:Increment data counter.
C012	MOV	A,M	7E	

```
C013            STAX        D       12
C014            INX         D       13
C015  forward:  INX         H       23
C016            DCR         C       0D          :Decrement counter
C017            JNZ loop            C2  0B  C0  :If c<>o go to to location
                                                Labled loop.else
C01A            RST         5       EF          :Halt/ Break.
```

3) To isolate a bit pattern and compare it with the already available bit pattern we can use ANI data instruction

Example In a 8-bit number d2d1d0 bit pattern corresponds to the password of users, write a program to isolate the password bits and store in a separate array.

4) Write a program to compare whether a given number is in the given list or not.

Note to teachers: Teachers can device new problems to illustrate the use of all the logical group of instructions. Encourage students to come up with their problems and to write their programs. Tell to students that problems for which programs are available in record will not be given in the examination.

Exercise:

1. Modify the program given for negative numbers to list positive numbers.
2. Modify the program given to find odd numbers to list even numbers.
3. Write programs for ex1 and ex 2 using Rotate instructions.
4. Write a program to test the status of d3 bit of every byte in the given list And to put 00 if the bit is zero and to put FF if the bit is one in another array.
5. information based on a seven bit code is stored in an array. Write a program to check its parity and to put 00 if the parity is even and to put FF if the parity is odd in another array.
6. Modify program of Ex5 to convert the all information bytes in to odd parity.

Experiment No. 11

Arithmetic programs 1

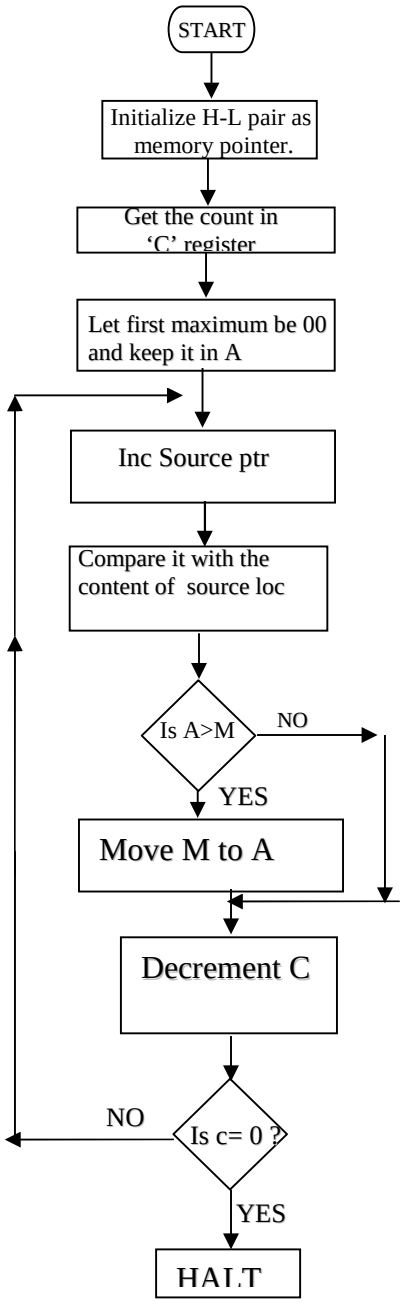
AIM:- To write a program to find the maximum number in a given set of 8-bit numbers .

Program area commences from 3900.

APPARATUS: 8085 Microprocessor kit .

DATA AREA:- Source data starts from 3951 and the number of bytes and data available is given in location 3950.

Flow chart



Method used to find maximum in the given list :

Several methods are available for this. One such method is to put first element in the list in a A register and compare its value with second number in the list .Transfer larger of the two into A .Repeat the process with other bytes until the list is exhausted. If there are n elements in the list the program makes (n-1) comparisons. However by initializing a register with 00 we can make n comparisons and find minimum.

PROCEDURE:-

Initialize the source memory pointer by loading 3950 in the H-L register pair. Initialize Reg C is by moving the content of 3950 into it. Initialize A register as counter with 00. Increment the H-L pair register and compare the content of next location with the content of Accumulator. If the content of memory location is less than the content of Accumulator go to the step to decrement byte counter. otherwise move the content of memory location to Accumulator As one byte is compared decrement byte counter and if the content of byte counter is not zero go to the step to increment the address of memory location. The program loops through n times until all the bytes are compared with the content of accumulator and storing the larger of the two numbers compared in Accumulator. After all the bytes are compared the largest or maximum number in the list will be available in Accumulator and the program halts after transferring the content of Accumulator to 3960 location.

Address	mnemonics	operands	op-code	comments
3900	LXI	H 3900	21 50 39 :	Load H-L pair with 3900
3903	MOV	C,M	4E :	Initialize C-reg.as down counter
3904	MVI	A, 00	3E 00 :	Acc.is initialized with 00
3906 loop	INX	H	23 :	Increment H-L pair reg.
3907	CMP	M	BE :	Compare acc. With content of source.
3908	JNC forward		D2 0C 39 :	If there is no carry decrement the counter else next step.
390B	MOV	A,M	7E :	Move memory content to Acc.
390C forward	DCR	C	0D :	Decrement the counter.
390D	JNZ(loop)		C2 06 39 :	If result is non-zero go to back steps.
3910	STA	3960	32 60 39 :	Store the content of Acc.in 3960.
3913	RST	5	EF :	Halt./break

RESULT:-

After execution of the program the maximum number in the given data is available in memory location 3960 and its value for the given data is ---.

Suggestions to teachers: Students may be trained to use all the arithmetic instructions
By devising new problems. The program given serves only as a sample.

Exercise:

1. Write a program to find minimum in the given list
2. Write a program that stores maximum in the list in c060 and minimum in c061 Locations.
3. Write a program to find the average of maximum and minimum numbers.
4. Write a program that lists separately the numbers greater than are equal to The mean found in ex.4.
5. Write a program using other instructions than those given in the program given.
6. Try new algorithms.
7. Write a program to find whether a given number is present in the list or not. Put FF in C050 location if the number exits other wise put 00.
8. Write a program to find how many times a given number repeated in the list.
9. Write a program to put given number in all the 16 locations starting from C051\
10. Write a program to increment each number in the given list by 1 and store them in the same locations.
11. Write a program to logically OR the content of each location in the given list.
12. Write a program to increment each number in the list by 05.
13. Write a program to add two 32 bit numbers in the given list.

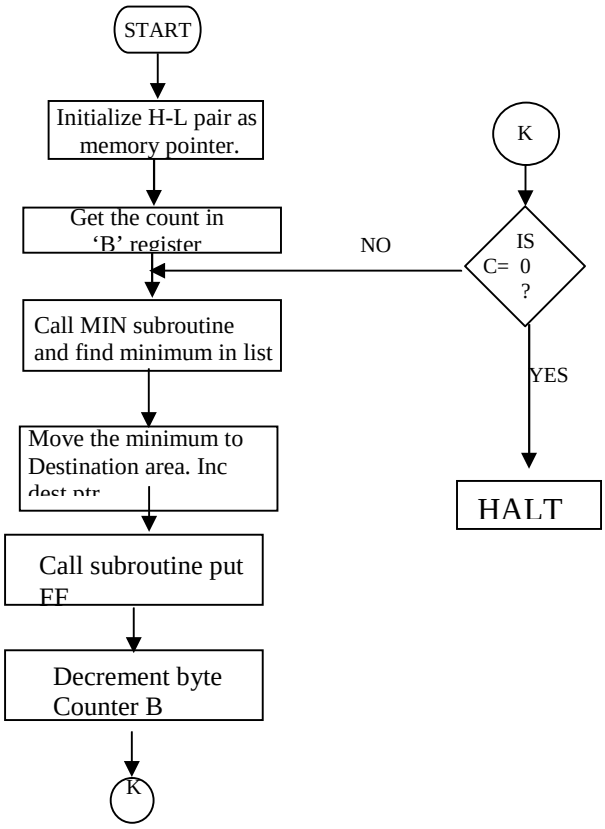
Experiment No. 12

Arithmetic programs 2

Aim:- To write a program to arrange numbers in a given list in ascending order.
Apparatus: 8085 microprocessor kit
Method:
There are several methods like simple sort, bubble sort, shell sort, quick sort etc to sort numbers. Here we follow a simple method.
From the given list smallest number is identified .It is copied in to the first location in destination area . This minimum number is replaced with FF (highest possible maximum number. (It is assumed that the FF is not in the original list) Find next minimum and copy it in to the next location of destination area. These steps are carried out until the list is exhausted. At the end of execution of the program the original list in source area is destroyed and sorted list appears in destination area.
Defining program and data area :-
Program area :-
The main program starts from 3900. The program calls subroutines called MIN and PUTFF. The subroutine for minimum commences from 3916. The subroutine put FF commences from 3930.

Data area:-

Source area starts from 3951 location .The no. of bytes in the given list is stored in 3950 .The destination starts from 3970 .



Procedure:

1. Source memory pointer is initialized by loading the address 3950 in HL register pair. A byte counter (B) is initialized with number of elements in the list. The destination area is initialized by loading 3970 in DE pair . We identify the minimum number in the list by calling the subroutine MIN.
2. After finding the first minimum it is copied to first location in destination area. This minimum number in source area is replaced with a maximum number not present in the list (say FF) by calling the sub routine PUT FF.
3. The B register is decremented once to indicate that one element in the list is sorted out.
4. To continue the process we check the status of Z flag. If it is not set, we loop back (we go back to location 3907) and repeat the steps. This process sorts out all elements in the list in ascending order. At the end, B register becomes zero and the program Halts or breaks.

Subroutine MIN: This sub program is identical to the program of MPPR3 for finding minimum in the list, but for the fact that the starting address is 3916 and it terminates with RET instruction instead of RST 5.

Subroutine PUT FF :

This subroutine commences from 3930. It identifies the memory location of minimum number of the list and replaces it with FF, so that, this minimum will not interfere in the further execution . It is quite easy to follow the steps given in the program.

The source and destination areas before and after the execution are shown in table2&3.

RESULT:-

The given list of numbers are arranged in ascending order as shown in following table.

ADDRESS	MNEMONIC	OPERANDS	Machine code	Comments
3900	LXI	H 3950	21 50 39	LOAD H-L PAIR WITH 3950
3903	LXI	D 3970	11 70 39	LOAD D-E PAIR WITH 3970
3906	MOV	B,M	46	INTIALISE B COUNTER
3907	CALL MIN		CD 16 39	CALL FOR MINIMUM NO.
390A	INX	D	13	INCREMENT DEST. ADD
390B	STAX	D	12	STORE ACC..IN DEST. LOC
390C	CALL	PUT FF	CD 30 39	REPLACE MIN NO FOUND WITH FF IN ORG LIST
390F	DCR	B	05	DECREMENT THE COUNTER B
3910	JNZ	loop	C2 07 39	IF THE LIST IS NOT EXHAUSTED REPEAT LOOP
3913		HLT	EF	HALT /BREAK

Subroutine MIN

Address	mnemonics	operands	op-code	comments
3916	LXI	H 3950	21 50 39	
3919	MOV	C,M	4E	
391A	MVI	A FF	3E FF	
391C loop	INX	H	23	
391D	CMP	M	BE	
391E	JC	forward	DA 22 39	
3921	MOV	A,M	7E	
3922 forward	DCR	C	0D	
3923	JNZ	loop	C2 1C 39	
3926	RET		C9	

SUBROUTINE PUT FF

Address	Label	mnemonics	operands	op-code	comments
3930		LXI	H 3950	21 50 39	
3933	loop1:	INX	H	23	
3934		CMP	M	BE	
3935		JNZ	loop1	C2 33 39	
3938		MVI	A FF	3E FF	
3938		MOV	M,A	77	
3940		RET		C9	

Before execution:

Source Addr	data	destination Addr	data
3950	05	3970	XX
3951	72	3971	XX
3952	21	3972	XX
3953	34	3973	XX
3954	12	3974	XX
3955	B2	3975	XX

After execution:

Source Addr	data	destination Addr	data
3950	05	3970	
3951	72	3971	
3952	21	3972	
3953	34	3973	
3954	12	3974	
3955	B2	3975	

Exercise:

1) Write a program to sort data in descending order
2) Write programs to sort data using other sorting methods.
3) Characters are written in ASCII code. A list of names each containing four characters are given in the memory area starting from C051. The number of names in the list is stored in C050 location. Sort the names in alphabetical order
4) Write a program to add a given list of numbers
5) Wrie a program to add the numbers including carry
6) Write a program to multiply two numbers by repeated addition method.

ELECTRONICS
(DPHYL21)
(MSC PHYSICS)

ACHARYA NAGARJUNA UNIVERSITY

CENTRE FOR DISTANCE EDUCATION

NAGARJUNA NAGAR,

GUNTUR

ANDHRA PRADESH