

Practical-1

Aim: Assembling of Computer.

Computer:

- An electronic device for storing and processing data, typically in binary form, according to instructions given to it in a variable program.
- A computer is a device that accepts information (in the form of digitalized data) and manipulates it for some result based on a program, software, or sequence of instructions on how the data is to be processed.

History of Computer:

The computer's full name is Common Operating Machine Purposely Used for Technological and Educational Research.

Charles Babbage (26 December 1791 – 18 October 1871) was an English polymath A mathematician, philosopher, inventor and mechanical engineer, Babbage originated the concept of a digital programmable computer.

Babbage is considered by some to be "father of the computer".



Fig 1.1 Computer

Generation of Computer's:

This long period is often conveniently divided into the subsequent phases called computer generations:

There are total of five generation of computer's:

- First Generation Computers (1940-1956)
- Second Generation Computers (1956-1963)
- Third Generation Computers (1964-1971)
- Fourth Generation Computers (1971-Present)
- Fifth Generation Computers (Present and Beyond)

First Generation Computers: Vacuum Tubes (1940-1956)

- The technology behind the primary generation computers was a fragile glass device, which was called vacuum tubes.
- These computers were very heavy and really large in size. These weren't very reliable and programming on them was a really tedious task as they used high-level programming language and used no OS.
- First-generation computers were used for calculation, storage, and control purpose. They were too bulky and large that they needed a full room and consume rot of electricity.



Fig 2.1 1st

Main first generation computers are:

- **ENIAC:** Electronic Numerical Integrator and Computer, built by J. Presper Eckert and John V. Mauchly was a general-purpose computer. It had been very heavy, large, and contained 18,000 vacuum tubes.
- **EDVAC:** Electronic Discrete Variable Automatic Computer was designed by von Neumann. It could store data also as instruction and thus the speed was enhanced.
- **UNIVAC:** Universal Automatic Computer was developed in 1952 by Eckert and Mauchly.

Main characteristics of first generation computers are:

Main electronic component	Vacuum tube.
Programming language	Machine language.
Main memory	Magnetic tapes and magnetic drums.
Input/output devices	Paper tape and punched cards.
Speed and size	Very slow and very large in size (often taking up entire room).
Examples of the first generation	IBM 650, IBM 701, ENIAC, UNIVAC1, etc.

Second Generation Computers: Transistors (1956-1963)

Second-generation computers used the technology of transistors rather than bulky vacuum tubes. Another feature was the core storage. A transistor may be a device composed of semiconductor material that amplifies a sign or opens or closes a circuit.

Transistors were invented in Bell Labs. The use of transistors made it possible to perform powerfully and with due speed. It reduced the dimensions and price and thankfully the warmth too, which was generated by vacuum tubes. Central Processing Unit (CPU), memory, programming language and input, and output units also came into the force within the second generation.

Programming language was shifted from high level to programming language and made programming comparatively a simple task for programmers. Languages used for programming during this era were FORTRAN (1956), ALGOL (1958), and COBOL (1959).



Fig 3.1 2nd

Main characteristics of second generation computers are:-

Main electronic component	Transistor.
Programming language	Machine language and assembly language.
Memory	Magnetic core and magnetic tape/disk.
Input/output devices	Magnetic tape and punched cards.
Power and size	Smaller in size, low power consumption, and generated less heat (in comparison with the first generation computers).
Examples of second generation	PDP-8, IBM1400 series, IBM 7090 and 7094, UNIVAC 1107, CDC 3600 etc.

Third Generation Computers: Integrated Circuits. (1964-1971)

During the third generation, technology envisaged a shift from huge transistors to integrated circuits, also referred to as IC.

Here a variety of transistors were placed on silicon chips, called semiconductors.

The most feature of this era's computer was the speed and reliability.

IC was made from silicon and also called silicon chips.



Fig 4.1 3rd

A single IC, has many transistors, registers, and capacitors built on one thin slice of silicon.

The value size was reduced and memory space and dealing efficiency were increased during this generation. Programming was now wiped out Higher level languages like BASIC (Beginners All-purpose Symbolic Instruction Code). Minicomputers find their shape during this era.

Main characteristics of third generation computers are:

Main electronic component	Integrated circuits (ICs)
Programming language	High-level language
Memory	Large magnetic core, magnetic tape/disk
Input / output devices	Magnetic tape, monitor, keyboard, printer, etc.
Examples of third generation	IBM 360, IBM 370, PDP-11, NCR 395, B6500, UNIVAC 1108, etc.

Fourth Generation Computers: Micro-processors (1971-Present)

In 1971 First microprocessors were used, the large scale of integration LSI circuits built on one chip called microprocessors.



Fig 5.1 4th

The most advantage of this technology is that one microprocessor can contain all the circuits required to perform arithmetic, logic, and control functions on one chip.

The computers using microchips were called microcomputers. This generation provided the even smaller size of computers, with larger capacities.

That's not enough, then Very Large Scale Integrated (VLSI) circuits replaced LSI circuits.

The Intel 4004chip, developed in 1971, located all the components of the pc from the central processing unit and memory to input/ output controls on one chip and allowed the dimensions to reduce drastically.

Technologies like multiprocessing, multiprogramming, time-sharing, operating speed, and virtual memory made it a more user-friendly and customary device.

The concept of private computers and computer networks came into being within the fourth generation.

Main characteristics of fourth generation computers are:

Main electronic component	Very large-scale integration (VLSI) and the microprocessor (VLSI has thousands of transistors on a single microchip).
Memory	semiconductor memory (such as RAM, ROM, etc.)
Input/output devices	pointing devices, optical scanning, keyboard, monitor, printer, etc.
Examples of fourth generation	IBM PC, STAR 1000, APPLE II, Apple Macintosh, Alter 8800, etc.

Fifth Generation Computers

The technology behind the fifth generation of computers is AI. It allows computers to behave like humans. It is often seen in programs like voice recognition, area of medicines, and entertainment. Within the field of games playing also it's shown remarkable performance where computers are capable of beating human competitors.

The speed is highest, size is that the smallest and area of use has remarkably increased within the fifth generation computers. Though not a hundred percent AI has been achieved to date but keeping in sight the present developments, it is often said that this dream also will become a reality very soon.

In order to summarize the features of varied generations of computers, it is often said that a big improvement has been seen as far because the speed and accuracy of functioning care, but if we mention the dimensions, it's being small over the years. The value is additionally diminishing and reliability is in fact increasing.

Fig 6.1 5th**Main characteristics of fifth generation computers are:**

Main electronic component	Based on artificial intelligence, uses the Ultra Large-Scale Integration (ULSI) technology and parallel processing method (ULSI has millions of transistors on a single microchip and Parallel processing method use two or more microprocessors to run tasks simultaneously).
Language	Understand natural language (human language).
Size	Portable and small in size.
Input / output device	Trackpad (or touchpad), touchscreen, pen, speech input (recognize voice/speech), light scanner, printer, keyboard, monitor, mouse, etc.
Example of fifth generation	Desktops, laptops, tablets, smartphones, etc.

Types Of Computers:-

There are total of four types of computers.

1. Micro Computer
2. Mini Computer
3. Mainframe Computer
4. Super Computer

1. Micro Computer:-

Microcomputer is also known as a personal computer. It is a general-purpose computer that is designed for individual use.

It has a microprocessor as a central processing unit, memory, storage area, input unit and output unit. Laptops and desktop computers are examples of microcomputers.

They are suitable for personal work that may be making an assignment, watching a movie, or at office for office work.



Fig 6.1 Micro Computer

Characteristics of a microcomputer:

- It is the smallest in size among all types of computers.
- A limited number of software can be used.
- It is designed for personal work and applications. Only one user can work at a time.
- It is less expensive and easy to use.
- It does not require the user to have special skills or training to use it.
- Generally, comes with single semiconductor chip.
- It is capable of multitasking such as printing, scanning, browsing, watching videos, etc.

Types of Micro Computers:-

1. Personal
2. Workstation
3. Laptops
4. Mobile
5. Embedded Computer

Personal:

Personal computers are used to create spreadsheets, write papers, play games, track our finances, account, run databases, and many other tasks.

Also, at home, it is widely used for playing PC games, multimedia entertainment, accessing the Internet and more.



Fig 6.2 Personal

If your PC is connected to the Internet, you can use it for communicating with friends via instant messaging programs, browsing the Web, checking e-mail, and downloading data or files.

It is normal for a personal computer to create a network by connecting more than one PC together, even though they are designed to use as single-user systems.

Workstation:

Workstation is a single user computer that is designed for technical or scientific applications.

It has a faster microprocessor, a large amount of RAM and high speed graphic adapters.

It generally performs a specific job with great expertise; accordingly, they are of different types such as graphics workstation, music workstation and engineering design workstation.



Fig 6.3 Workstation

Characteristics of workstation computer:

- It is a high-performance computer system designed for a single user for business or professional use.
- It has larger storage capacity, better graphics, and more powerful CPU than a personal computer.
- It can handle animation, data analysis, CAD, audio and video creation and editing.

Laptop:

A Laptop is a small portable personal computer consisting of a screen, an alphanumeric keyboard, and a touchpad. Sometimes, the laptop is also called a notebook computer.



Fig 6.4 Laptop

The first laptop was invented by Osborne Computer Corporation in April 1981 and its name was Osborne I. At the time of release the cost of the Osborne I was \$1795. Osborne I consists of 5 inches long screen, 64 KB of memory, and the keyboard which was on the lid of the computer. The weight of this first laptop is about 25 pounds.

Laptops are used in every field such as research, education, Internet surfing, playing games, and homes.

We can easily fold flat laptops for transportation. You can take laptops with you and use them in different locations. Laptops have a battery which should be charged via sockets. As compared to desktop computers, laptops are more expensive because they are more difficult to design.

Mobile:

Mobile technology is technology that goes where the user goes. It consists of portable two-way communications devices, computing devices and the networking technology that connects them.



Fig 6.5 Mobile

Currently, mobile technology is typified by internet-enabled devices like smartphones, tablets and watches. These are the latest in a progression that includes two-way pagers, notebook computers, mobile telephones (flip phones), GPS-navigation devices and more.

The communications networks that connect these devices are loosely termed wireless technologies. They enable mobile devices to share voice, data and applications (mobile apps).

Embedded Computer:

An embedded computer, which is an integral component of most embedded systems, is a combination of hardware and software that is designated to perform a highly specific function.



Fig 6.6 Embedded Computer

For example, the type of embedded computer in a washing machine will not be the same as the embedded computer in a Nikon camera.

2. Mini Computers:

It is a midsize multiprocessing computer. It consists of two or more processors and can support 4 to 200 users at one time. Miniframe computers are used in institutes and departments for tasks such as billing, accounting and inventory management. A minicomputer lies between the mainframe and microcomputer as it is smaller than mainframe but larger than a microcomputer.



Fig 7.1 Mini

Characteristics of miniframe or minicomputer:

- It is light weight that makes it easy to carry and fit anywhere.
- It is less expensive than mainframe computers.
- It is very fast compared to its size.
- It remains charged for a long time.
- It does not require a controlled operational environment.

Applications of minicomputers:

- Process control: It was used for process control in manufacturing. It mainly performs two primary functions that are collecting data and feedback. If any abnormality occurs in the process, it is detected by the minicomputer and necessary adjustments are made accordingly.
- Data management: It is an excellent device for small organizations to collect, store and share data. Local hospitals and hotels can use it to maintain the records of their patients and customers respectively.
- Communications Portal: It can also play the role of a communication device in larger systems by serving as a portal between a human operator and a central processor or computer.

3.Mainframe Computer:

Mainframe computers are designed to support hundreds or thousands of users simultaneously. They can support multiple programs at the same time. It means they can execute different processes simultaneously.

These features of mainframe computers make them ideal for big organizations like banking and telecom sectors, which need to manage and process high volume of data.

Mainframe computers are designed to support hundreds or thousands of users simultaneously. They can support multiple programs at the same time. It means they can execute different processes simultaneously.

These features of mainframe computers make them ideal for big organizations like banking and telecom sectors, which need to manage and process a high volume of data that requires integer operations such as indexing, comparisons, etc.



Fig 8.1 MainFrame Computer

Characteristics of Mainframe Computers:

- It can process huge amount of data, e.g. millions of transactions in a second in the banking sector.
- It has a very long life. It can run smoothly for up to 50 years after proper installation.
- It gives excellent performance with large scale memory management.
- It has the ability to share or distribute its workload among other processors and input/output terminals.
- There are fewer chances of error or bugs during processing in mainframe computers. If any error occurs it can fix it quickly without affecting the performance.
- It has the ability to protect the stored data and other ongoing exchange of information **and data.**

Applications of mainframe computers:

- In health care, it enabled hospitals to maintain a record of their millions of patients in order to contact them for treatment or related to their appointment, medicine updates or disease updates.
- In the field of defence, it allows the defence departments to share a large amount of sensitive information with other branches of defence.
- In the field of education, it helps big universities to store, manage and retrieve data related to their courses, admissions, students, teachers, employees and affiliated schools and colleges.

- In the retail sector, the retail companies that have a huge customer base and branches use mainframe computers to handle and execute information related to their inventory management, customer management, and huge transactions in a short duration.

4. Super Computer:

Supercomputers are the biggest and fastest computers. They are designed to process huge amount of data. A supercomputer can process trillions of instructions in a second. It has thousands of interconnected processors.



Fig 9.1 Computer

Supercomputers are particularly used in scientific and engineering applications such as weather forecasting, scientific simulations and nuclear energy research. The first supercomputer was developed by Roger Cray in 1976.

Characteristics or applications of supercomputers:

- It has the ability to decrypt your password to enhance protection for security reasons.
- It produces excellent results in animations.
- It is used for virtual testing of nuclear weapons and critical medical tests.
- It can study and understand climate patterns and forecast weather conditions. It can run in NOAA's system (National Oceanic and Atmospheric Administration) that can execute any type of simple and logical data.
- It helps in designing the flight simulators for pilots at the beginner level for their training.
- It helps in extracting useful information from data storage centres or cloud system. For example, in insurance companies.

- It has played a vital role in managing the online currency world such as stock market and bitcoin.
- It helps in the diagnosis of various critical diseases and in producing accurate results in brain injuries, strokes, etc.
- It helps in scientific research areas by accurately analysing data obtained from exploring the solar system, satellites, and movement of Earth.
- It also used in a smog control system where it predicts the level of fog and other pollutants in the atmosphere.

Practical-2

Aim: Write a program to convert a given number system to other number system.

```
#include <stdio.h>
#include <math.h>
#include <string.h>
#include <conio.h>

long int Bin_to_Dec(long int);
long int Bin_to_Oct(long int);
long int Bin_to_Hex(long int);
long int Dec_to_Bin(long int);
long int Dec_to_Oct(long int);
long int Dec_to_Hex(long int);
long int Oct_to_Bin(long int);
long int Oct_to_Dec(long int);
long int Oct_to_Hex(long int);
void Hex_to_Bin(char []);
void Hex_to_Dec(char []);
void Hex_to_Oct(char []);
int main()
{
    int op,num=1,check;
    long int bin,oct,dec;
    char hex[100];
    int i,j,space;
    printf("\t\tWELCOME TO NUMBER SYSTEM CONVERSION\n\n");
    while(num!=0)
    {
        printf("\t\t>>>>>> CHOOSE THE CONVERSION <<<<<<\n\n");
        printf("=> BINARY <=<\n");
        printf("1: Binary to Decimal.\n2: Binary to Octal.\n3: Binary to Hexa-Decimal.\n");
```

```

printf("\n=> DECIMAL <=\n");
printf("4: Decimal to Binary.\n5: Decimal to Octal.\n6: Decimal to Hexa-
Decimal.\n");
printf("\n=> OCTAL <=\n");
printf("7: Octal to Binary.\n8: Octal to Decimal.\n9: Octal to Hexa-Decimal.\n");
printf("\n=> HEXA-DECIMAL <=\n");
printf("10: Hexa-Decimal to Binary.\n11: Hexa-Decimal to Decimal.\n12: Hexa-
Decimal to Octal.\n");
printf("\nENTER YOUR CHOICE: ");
scanf("%d",&op);
switch(op)
{
    case 1:
        printf("\n***BINARY TO DECIMAL***\n");
        D:
        printf("\nEnter the Number in Binary form (0s & 1s): ");
        scanf("%ld",&bin);
        check=bin;
        while(check!=0)
        {
            num=check%10;
            if(num>1)
            {
                printf("\n%d IS NOT BINARY NUMBER.\n",bin);
                printf("***TRY AGAIN***\n");
                goto D;
            }
            else
                check=check/10;
        }
        Bin_to_Dec(bin); break;

```

case 2:

```
printf("\n***BINARY TO OCTAL***\n");  
E:  
printf("\nEnter the Number in Binary form (0s & 1s): ");  
scanf("%ld",&bin);  
check=bin;  
while(check!=0)  
{  
    num=check%10;  
    if(num>1)  
    {  
        printf("\n%d IS NOT BINARY NUMBER.\n",bin);  
        printf("***TRY AGAIN***\n");  
        goto E;  
    }  
    else  
        check=check/10;  
}  
Bin_to_Oct(bin); break;
```

case 3:

```
printf("\n***BINARY TO HEXA-DECIMAL***\n");  
F:  
printf("\nEnter the Number in Binary form (0s & 1s): ");  
scanf("%ld",&bin);  
check=bin;  
while(check!=0)  
{  
    num=check%10;  
    if(num>1)  
    {  
        printf("\n%d IS NOT BINARY NUMBER.\n",bin);  
        printf("***TRY AGAIN***\n");  
    }  
}
```

```
        goto F;
    }
    else
        check=check/10;
    }
    Bin_to_Hex(bin); break;
case 4:
    printf("\n***DECIMAL TO BINARY***\n");
    printf("\nEnter the Number in Decimal form (0 to 9): ");
    scanf("%ld",&dec);
    Dec_to_Bin(dec); break;
case 5:
    printf("\n***DECIMAL TO OCTAL***\n");
    printf("\nEnter the Number in Decimal form (0 to 9): ");
    scanf("%ld",&dec);
    Dec_to_Oct(dec); break;
case 6:
    printf("\n***DECIMAL TO HEXA-DECIMAL***\n");
    printf("\nEnter the Number in Decimal form (0 to 9): ");
    scanf("%ld",&dec);
    Dec_to_Hex(dec); break;
case 7:
    printf("\n***OCTAL TO BINARY***\n");
    A:
    printf("\nEnter the Number in Octal form (0 to 7): ");
    scanf("%ld",&oct);
    check=oct;
    while(check!=0)
    {
        num=check%10;
        if(num>7)
        {
```

```
        printf("\n%d IS NOT OCTAL NUMBER.\n",num);
        goto A;
    }
    else
    {
        check=check/10;
        i++;
    }
}
Oct_to_Bin(oct); break;
case 8:
    printf("\n***OCTAL TO DECIMAL***\n");
    B:
    printf("\nEnter the Number in Octal form (0 to 7): ");
    scanf("%ld",&oct);
    check=oct;
    while(check!=0)
    {
        num=check%10;
        if(num>7)
        {
            printf("\n%d IS NOT OCTAL NUMBER.\n",num);
            goto B;
        }
        else
        {
            check=check/10;
            i++;
        }
    }
    Oct_to_Dec(oct); break;
```

case 9:

```
printf("\n***OCTAL TO HEXA-DECIMAL***\n");  
C:  
printf("\nEnter the Number in Octal form (0 to 7): ");  
scanf("%ld",&oct);  
check=oct;  
while(check!=0)  
{  
    num=check%10;  
    if(num>7)  
    {  
        printf("\n%d IS NOT OCTAL NUMBER.\n",num);  
        goto C;  
    }  
    else  
    {  
        check=check/10;  
        i++;  
    }  
}  
Oct_to_Hex(oct); break;
```

case 10:

```
printf("\n***HEXA-DECIMAL TO BINARY***\n");  
X:  
printf("\nEnter the Number in Hexa-Decimal form: ");  
scanf("%s",&hex);  
//check  
for(i=strlen(hex)-1;i>=0;i--)  
{  
    if(hex[i]>'f' && hex[i]<='z' || hex[i]>'F' && hex[i]<='Z')  
    {  
        printf("\nYou have to Enter Hexa-Decimal Number.\n");  
    }  
}
```

```

        printf("'%c' IS NOT Hexa-Decimal Number.\n",hex[i]);
        goto X;
    }
}

Hex_to_Bin(hex); break;
case 11:
    printf("\n***HEXA-DECIMAL TO DECIMAL***\n");
    Y:
    printf("\nEnter the Number in Hexa-Decimal form: ");
    scanf("%s",&hex);
    //check
    for(i=strlen(hex)-1;i>=0;i--)
    {
        if(hex[i]>'f' && hex[i]<='z' || hex[i]>'F'&& hex[i]<='Z')
        {
            printf("\nYou have to Enter Hexa-Decimal Number.\n");
            printf("'%c' IS NOT Hexa-Decimal Number.\n",hex[i]);
            goto Y;
        }
    }
    Hex_to_Dec(hex); break;
case 12:
    printf("\n***HEXA-DECIMAL TO OCTAL***\n");
    Z:
    printf("\nEnter the Number in Hexa-Decimal form: ");
    scanf("%s",&hex);
    //check
    for(i=strlen(hex)-1;i>=0;i--)
    {
        if(hex[i]>'f' && hex[i]<='z' || hex[i]>'F'&& hex[i]<='Z')
        {
            printf("\nYou have to Enter Hexa-Decimal Number.\n");

```



```

        printf("'%' IS NOT Hexa-Decimal Number.\n",hex[i]);
        goto Z;
    }
}
Hex_to_Oct(hex); break;
default:
    printf("\n***INVALID NUMBER***\n");
    break;
}
printf("\n\nDO YOU WANT TO CONTINUE = (1/0) :\n");
scanf("%d",&num);
}
space = 3+35;
for( i=1;i<=3;i++)
{
    for( j=1;j<=space;j++)
    {
        printf(" ");
    }
    space--;
    for( j=1;j<=2*i-1;j++)
    {
        printf("*");
    }
    printf("\n");
}
space = 37;
for( i=1;i<=3;i++)
{
    for( j=1;j<=space;j++)
    {
        printf(" ");
    }

```

```
    }
    space++;
    for( j=1;j<=2*(3-i)-1;j++)
    {
        printf("*");
    }
    printf("\n");
}
}

long int Bin_to_Dec(long int bin)
{
    int rem,sum=0,i=0;
    while(bin!=0)
    {
        rem=bin%10;
        bin=bin/10;
        sum=sum+rem*pow(2,i);
        i++;
    }
    printf("\nEquivalent Decimal Number : %d",sum);
}

long int Bin_to_Oct(long int bin)
{
    int i=0,rem,sum=0,remain[100],len=0;
    while(bin!=0)
    {
        rem=bin%10;
        bin=bin/10;
        sum=sum+rem*pow(2,i);
        i++;
    }
    i=0;
```

```
while(sum!=0)
{
    remain[i]=sum%8;
    sum=sum/8;
    i++;
    len++;
}
printf("\nEquivalent Octal Number : ");
for(i=len-1;i>=0;i--)
{
    printf("%d",remain[i]);
}
}
long int Bin_to_Hex(long int bin)
{
    int rem,i=0,sum=0,remain[100],len=0;
    while(bin!=0)
    {
        rem=bin%10;
        bin=bin/10;
        sum=sum+rem*pow(2,i);
        i++;
    }
    i=0;
    while(sum!=0)
    {
        remain[i]=sum%16;
        sum=sum/16;
        i++;
        len++;
    }
    printf("\nEquivalent Hexa-Decimal Number : ");
```

```
for(i=len-1;i>=0;i--)
{
    switch(remain[i])
    {
        case 10:
            printf("A"); break;
        case 11:
            printf("B"); break;
        case 12:
            printf("C"); break;
        case 13:
            printf("D"); break;
        case 14:
            printf("E"); break;
        case 15:
            printf("F"); break;
        default:
            printf("%d",remain[i]);
    }
}

long int Dec_to_Bin(long int dec)
{
    int rem[50],i,len=0;
    do
    {
        rem[i]=dec%2;
        dec=dec/2;
        i++;
        len++;
    }
    while(dec!=0);
```

```
printf("\nEquivalent Binary Number : ");
for(i=len-1;i>=0;i--)
{
    printf("%d",rem[i]);
}
}

long int Dec_to_Oct(long int dec)
{
    int rem[50],i,len=0;
    do
    {
        rem[i]=dec%8;
        dec=dec/8;
        i++;
        len++;
    }
    while(dec!=0);
    printf("\nEquivalent Octal Number : ");
    for(i=len-1;i>=0;i--)
    {
        printf("%d",rem[i]);
    }
}

long int Dec_to_Hex(long int dec)
{
    int rem[50],i,len=0;
    do
    {
        rem[i]=dec%16;
        dec=dec/16;
        i++;
    }
```

```
len++;
}
while(dec!=0);
printf("\nEquivalent Hexa-Decimal Number : ");
for(i=len-1;i>=0;i--)
{
    switch(rem[i])
    {
        case 10:
            printf("A"); break;
        case 11:
            printf("B"); break;
        case 12:
            printf("C"); break;
        case 13:
            printf("D"); break;
        case 14:
            printf("E"); break;
        case 15:
            printf("F"); break;
        default:
            printf("%d",rem[i]);
    }
}
}
long int Oct_to_Bin(long int oct)
{
    int rem[50],len=0,decimal=0,i=0,num,ans;
    while(oct!=0)
    {
        ans=oct % 10;
        decimal = decimal + ans * pow(8,i);
```

```
        i++;
        oct = oct/10;
    }
    i=0;
    do
    {
        rem[i]=decimal%2;
        decimal=decimal/2;
        i++;
        len++;
    }
    while(decimal!=0);
    printf("\nEquivalent Binary Number : ");
    for(i=len-1;i>=0;i--)
    {
        printf("%d",rem[i]);
    }
}

long int Oct_to_Dec(long int oct)
{
    int decimal=0,i=0,num,ans;
    while(oct!=0)
    {
        ans=oct % 10;
        decimal = decimal + ans * pow(8,i);
        i++;
        oct = oct/10;
    }
    printf("\nEquivalent Decimal Number : %d",decimal);
}

long int Oct_to_Hex(long int oct)
{

```

```
int rem[50],len=0,decimal=0,i=0,num,ans=0;
while(oct!=0)
{
    ans=oct % 10;
    decimal = decimal + ans * pow(8,i);
    i++;
    oct = oct/10;
}
i=0;
while(decimal!=0)
{
    rem[i]=decimal%16;
    decimal=decimal/16;
    i++;
    len++;
}
printf("\nEquivalent Hexa-Decimal Number : ");
for(i=len-1;i>=0;i--)
{
    switch(rem[i])
    {
        case 10:
            printf("A"); break;
        case 11:
            printf("B"); break;
        case 12:
            printf("C"); break;
        case 13:
            printf("D"); break;
        case 14:
            printf("E"); break;
```



```
        case 15:
            printf("F"); break;
        default:
            printf("%d",rem[i]);
        }
    }
}

void Hex_to_Bin(char hex[])
{
    int i=0;
    printf("\nEquivalent Binary Number : ");
    for(i=0;i<strlen(hex);i++)
    {
        switch (hex[i])
        {
            case '0':
                printf("0000"); break;
            case '1':
                printf("0001"); break;
            case '2':
                printf("0010"); break;
            case '3':
                printf("0011"); break;
            case '4':
                printf("0100"); break;
            case '5':
                printf("0101"); break;
            case '6':
                printf("0110"); break;
            case '7':
                printf("0111"); break;
            case '8':
```

```

        printf("1000"); break;
    case '9':
        printf("1001"); break;
    case 'A':
    case 'a':
        printf("1010"); break;
    case 'B':
    case 'b':
        printf("1011"); break;
    case 'C':
    case 'c':
        printf("1100"); break;
    case 'D':
    case 'd':
        printf("1101"); break;
    case 'E':
    case 'e':
        printf("1110"); break;
    case 'F':
    case 'f':
        printf("1111"); break;
    default:
        printf("\n Invalid hexa digit %c ", hex[i]);
    }
}
}

void Hex_to_Dec(char hex[])
{
    int i,num=0,power=0,decimal=0;
    for(i=strlen(hex)-1;i>=0;i--)
    {
        if(hex[i]=='A' || hex[i]=='a')

```

```
{
    num=10;
}
else if(hex[i]=='B' || hex[i]=='b')
{
    num=11;
}
else if(hex[i]=='C' || hex[i]=='c')
{
    num=12;
}
else if(hex[i]=='D' || hex[i]=='d')
{
    num=13;
}
else if(hex[i]=='E' || hex[i]=='e')
{
    num=14;
}
else if(hex[i]=='F' || hex[i]=='f')
{
    num=15;
}
else
{
    num=hex[i]-48;
}
decimal=decimal+num*pow(16,power);
power++;
}
printf("\nEquivalent Decimal Number : %d",decimal);
```

```
}  
void Hex_to_Oct(char hex[])  
{  
    int i,len,num=0,power=0,decimal=0,rem[100];  
    for(i=strlen(hex)-1;i>=0;i--)  
    {  
        if(hex[i]=='A' || hex[i]=='a')  
        {  
            num=10;  
        }  
        else if(hex[i]=='B' || hex[i]=='b')  
        {  
            num=11;  
        }  
        else if(hex[i]=='C' || hex[i]=='c')  
        {  
            num=12;  
        }  
        else if(hex[i]=='D' || hex[i]=='d')  
        {  
            num=13;  
        }  
        else if(hex[i]=='E' || hex[i]=='e')  
        {  
            num=14;  
        }  
        else if(hex[i]=='F' || hex[i]=='f')  
        {  
            num=15;  
        }  
        else  
        {
```

```
        num=hex[i]-48;
    }
    decimal=decimal+num*pow(16,power);
    power++;
}
i=0,len=0;
while(decimal!=0)
{
    rem[i]=decimal%8;
    decimal=decimal/8;
    i++;
    len++;
}
printf("\nEquivalent Octal Number : ");
for(i=len-1;i>=0;i--)
{
    printf("%d",rem[i]);
}

}
```

Output:

```
>>>>> CHOOSE THE CONVERSION <<<<<<

=> BINARY <=
1: Binary to Decimal.
2: Binary to Octal.
3: Binary to Hexa-Decimal.

=> DECIMAL <=
4: Decimal to Binary.
5: Decimal to Octal.
6: Decimal to Hexa-Decimal.

=> OCTAL <=
7: Octal to Binary.
8: Octal to Decimal.
9: Octal to Hexa-Decimal.

=> HEXA-DECIMAL <=
10: Hexa-Decimal to Binary.
11: Hexa-Decimal to Decimal.
12: Hexa-Decimal to Octal.

ENTER YOUR CHOICE: 1

***BINARY TO DECIMAL***

Enter the Number in Binary form (0s & 1s): 0001

Equivalent Decimal Number : 1
```

Practical-3

Aim: Implement a circuit in Logisim to display given binary number in decimal on to seven segment display.

Truth Table:

w	x	y	z	a	b	c	d	e	f	g	dp
0	0	0	0	1	1	1	1	1	1	0	1
0	0	0	1	0	1	1	0	0	0	0	1
0	0	1	0	1	1	0	1	1	0	1	1
0	0	1	1	1	1	1	1	0	0	1	1
0	1	0	0	0	1	1	0	0	1	1	1
0	1	0	1	1	0	1	1	0	1	1	1
0	1	1	0	1	0	1	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0	1
1	0	0	0	1	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1	1
1	0	1	0	0	0	0	0	0	0	0	0
1	0	1	1	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0	0

Fig 3. 1

K-MAP:

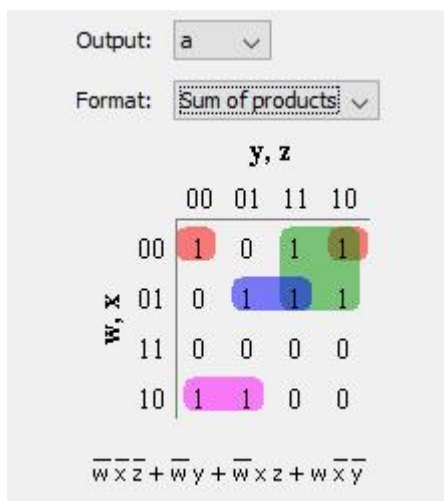


Fig 3. 2 K-MAP OF A
MAP OF B

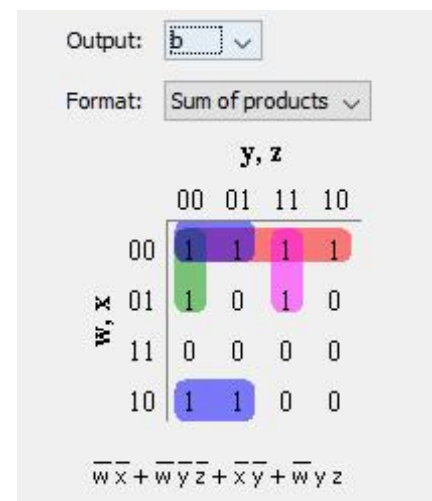


Fig 3.3 K-

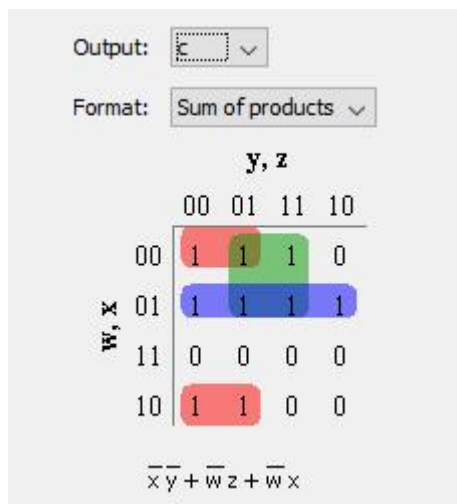


Fig 3.4 K-MAP OF C

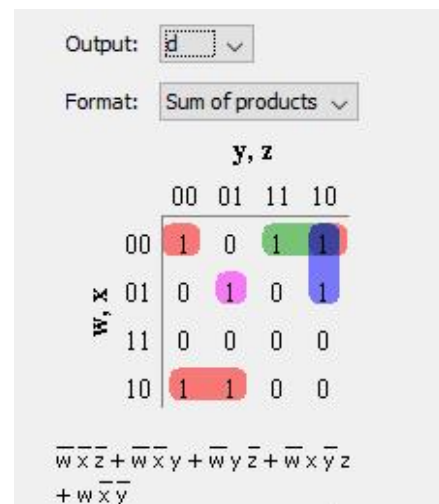


Fig 3.5 K-MAP OF D

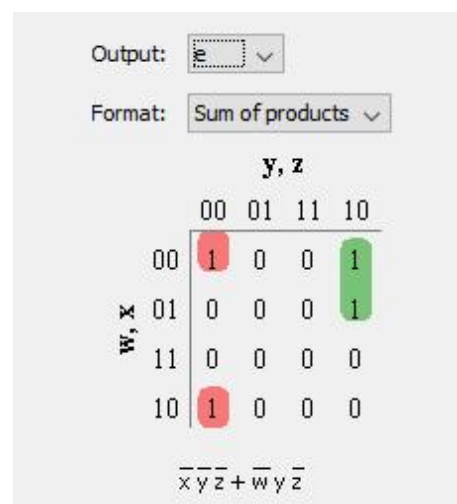


Fig 3.6 K-MAP OF E

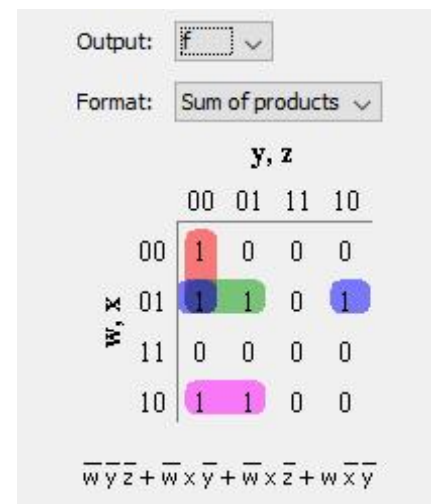


Fig 3.7 K-MAP OF F

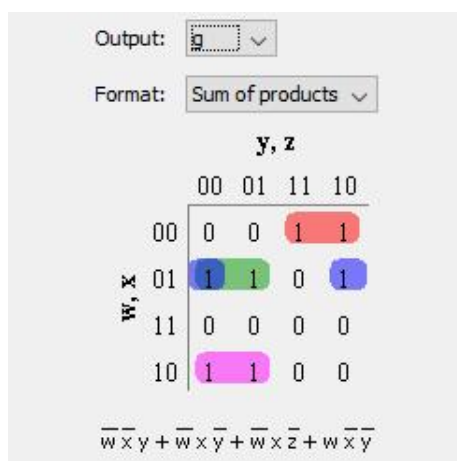


Fig 3.8 K-MAP OF G

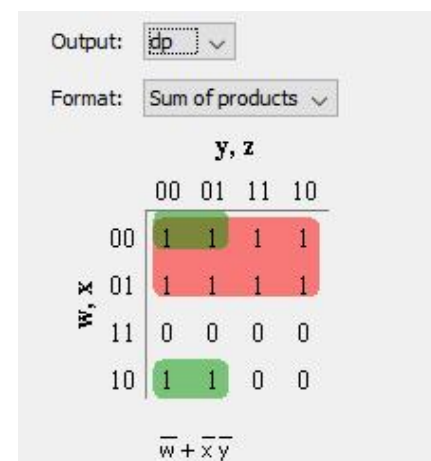


Fig 3.9 K-MAP OF DP

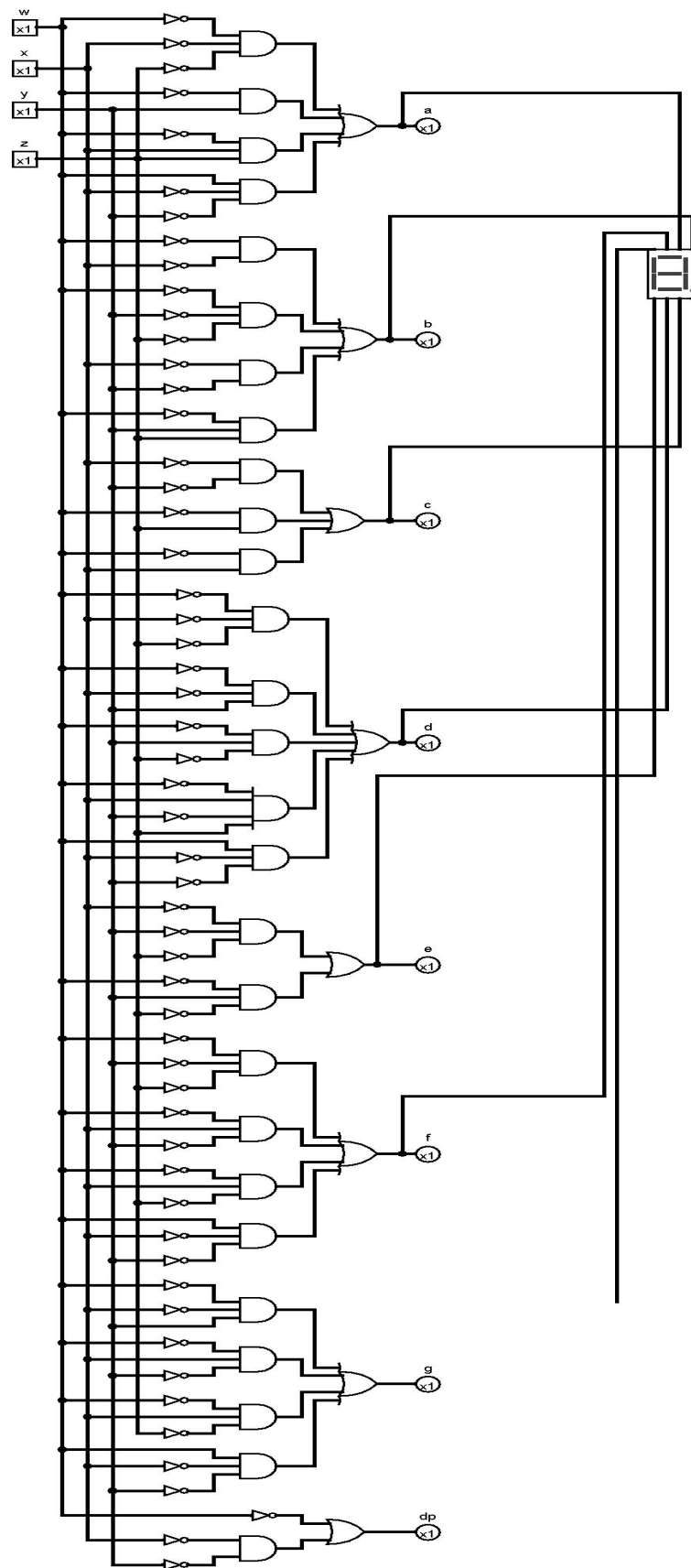
Circuit Diagram:

Fig 3.10 Circuit Diagram

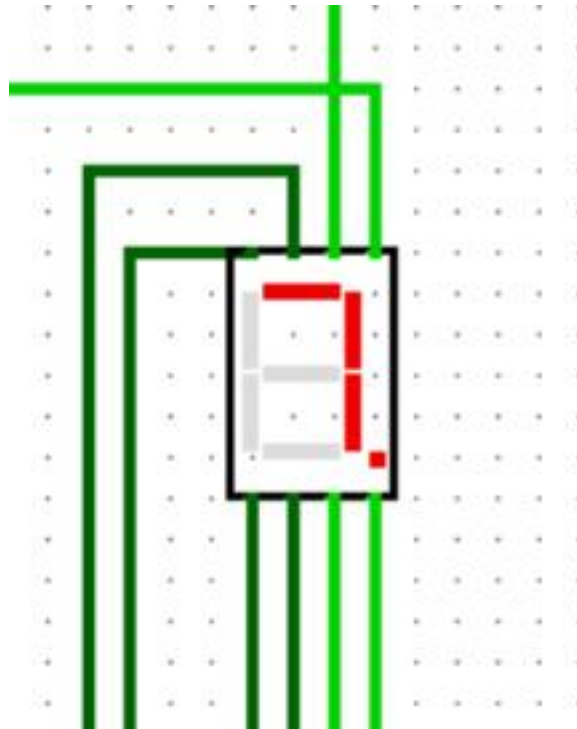
Output:

Fig 3.11 Output

Conclusion:

In this practical we have learnt how to implement circuit in a Logisim to display given binary number in decimal on to seven segment display.

Practical-4

Aim: Implement a circuit in Logisim which perform Addition and Subtraction of signnumber.

Practical:

Circuit Diagram:

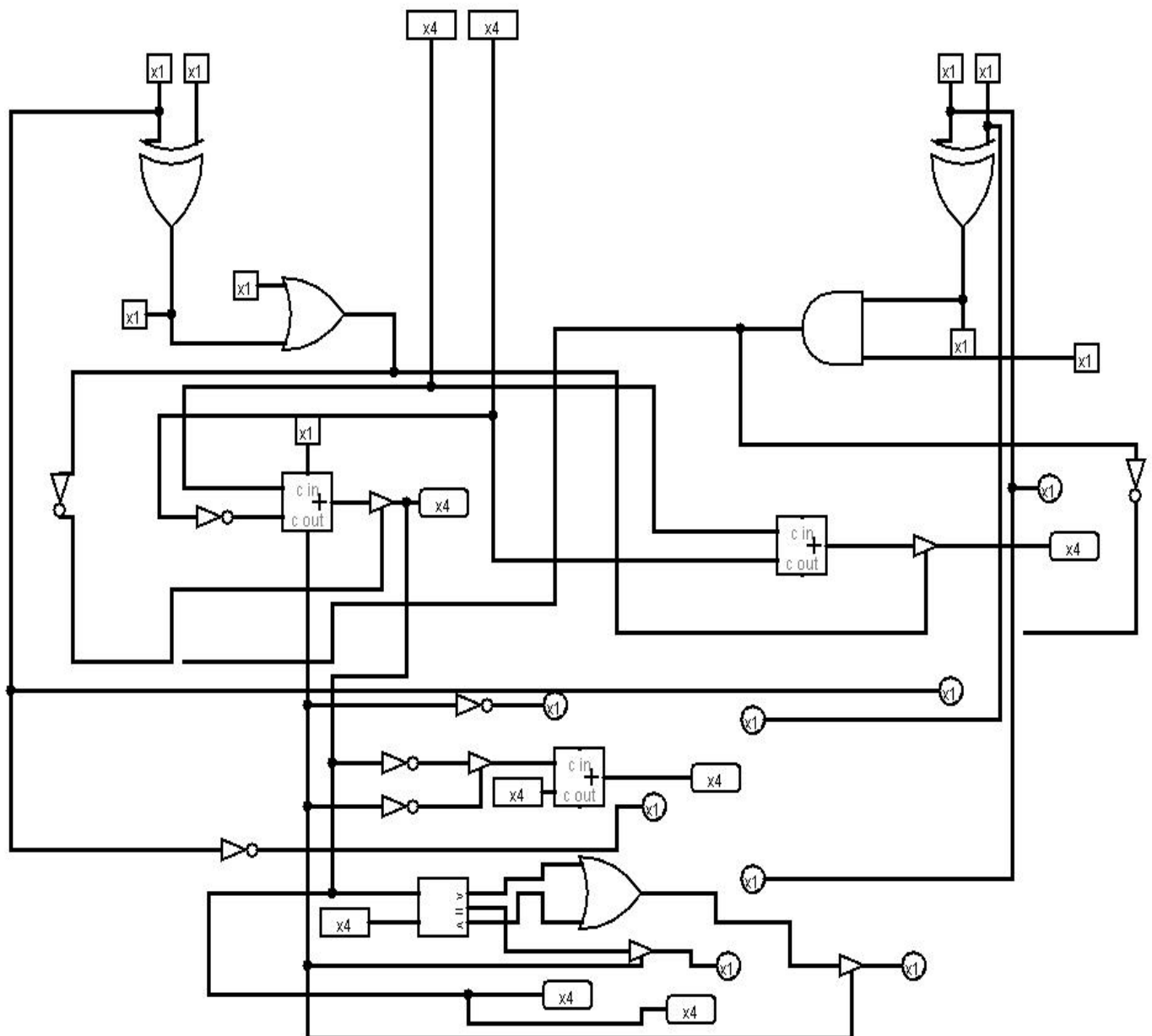


Fig 4.1 Circuit Diahgram

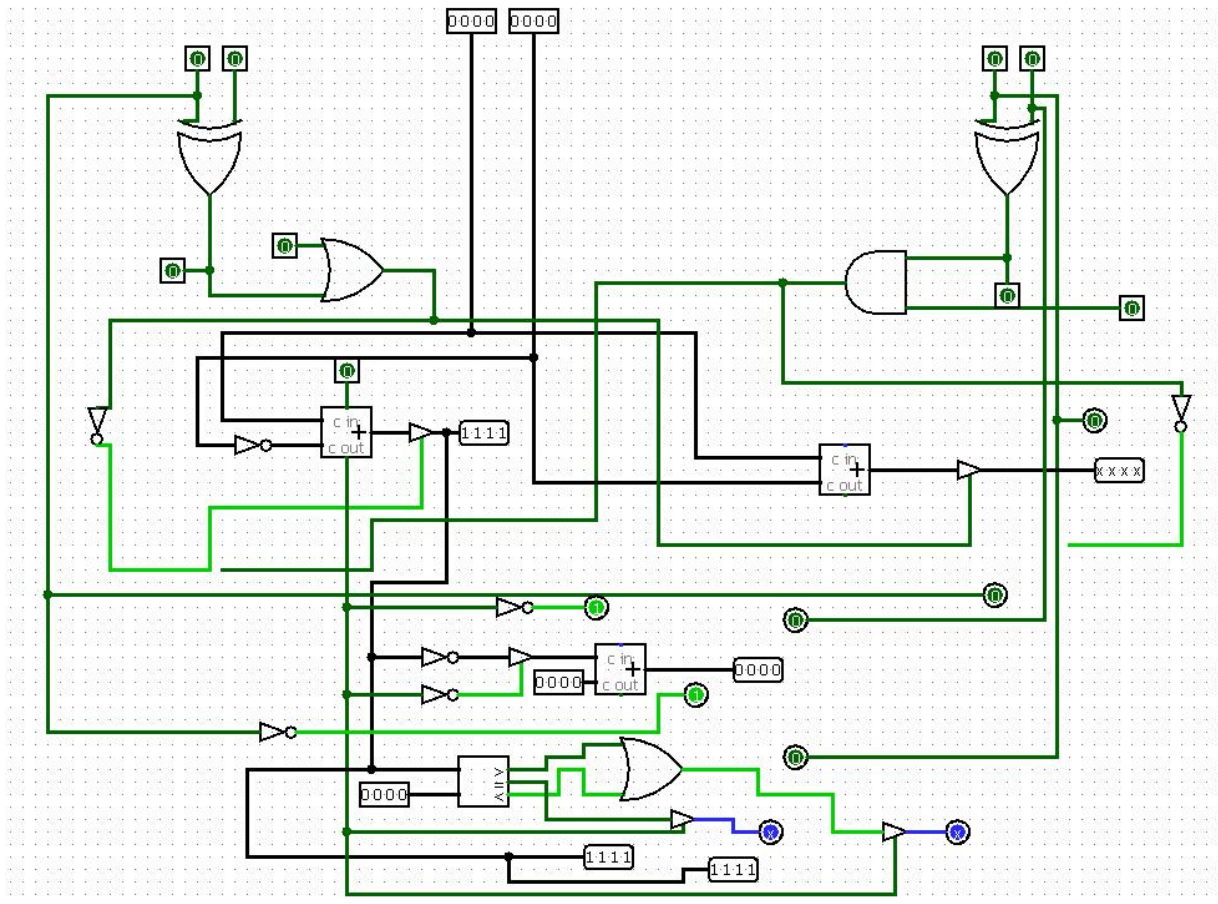
Output:

Fig 4.2 Circuit Diagram

Conclusion: in this practical we learn to implement a circuit in Logisim that performs addition and subtraction of signed numbers, the following steps can be taken:

- Design the circuit using basic gates, such as AND, OR, and NOT gates, to perform the bit-wise operations of addition and subtraction.
- Use a multiplexer to select between addition and subtraction operations.
- Implement a circuit to handle overflow and underflow conditions.
- Use a circuit to handle the sign bit, such as a circuit to add or subtract the sign bit depending on the operation.
- Test the circuit thoroughly to ensure correct operation and debug any issues.
- Finally, document the circuit and its operation thoroughly for future reference

Practical – 5

Aim: Write a program which perform multiplication using booth algorithm.

Program:

```
#include <stdio.h>
#include <math.h>

int a = 0, b = 0, c = 0, a1 = 0, b1 = 0, com[5] = {1, 0, 0, 0, 0};
int anum[5] = {0}, anumcp[5] = {0}, bnum[5] = {0};
int acomp[5] = {0}, bcomp[5] = {0}, pro[5] = {0}, res[5] = {0};

void binary()
{
    a1 = fabs(a);
    b1 = fabs(b);
    int r, r2, i, temp;
    for (i = 0; i < 5; i++)
    {
        r = a1 % 2;
        a1 = a1 / 2;
        r2 = b1 % 2;
        b1 = b1 / 2;
        anum[i] = r;
        anumcp[i] = r;
        bnum[i] = r2;
        if (r2 == 0)
        {
            bcomp[i] = 1;
        }
        if (r == 0)
        {
            acomp[i] = 1;
        }
    }
    // part for two's complementing
    c = 0;
```

```
for (i = 0; i < 5; i++)
{
res[i] = com[i] + bcomp[i] + c;
if (res[i] >= 2)
{
c = 1;
}
else
c = 0;
res[i] = res[i] % 2;
}
for (i = 4; i >= 0; i--)
{
bcomp[i] = res[i];
}
// in case of negative inputs
if (a < 0)
{
c = 0;
for (i = 4; i >= 0; i--)
{
res[i] = 0;
}
for (i = 0; i < 5; i++)
{
res[i] = com[i] + acomp[i] + c;
if (res[i] >= 2)
{
c = 1;
}
else
c = 0;
res[i] = res[i] % 2;
}
for (i = 4; i >= 0; i--)
```

```
{
anum[i] = res[i];
anumcp[i] = res[i];
}
}
if (b < 0)
{
for (i = 0; i < 5; i++)
{
temp = bnum[i];
bnum[i] = bcomp[i];
bcomp[i] = temp;
}
}
}
void add(int num[])
{
int i;
c = 0;
for (i = 0; i < 5; i++)
{
res[i] = pro[i] + num[i] + c;
if (res[i] >= 2)
{
c = 1;
}
else
{
c = 0;
}
res[i] = res[i] % 2;
}
for (i = 4; i >= 0; i--)
{
pro[i] = res[i];
```

```
printf("%d", pro[i]);
}
printf(".");
for (i = 4; i >= 0; i--)
{
printf("%d", anumcp[i]);
}
}

void arshift()
{ // for arithmetic shift right
int temp = pro[4], temp2 = pro[0], i;
for (i = 1; i < 5; i++)
{ // shift the MSB of product
pro[i - 1] = pro[i];
}
pro[4] = temp;
for (i = 1; i < 5; i++)
{ // shift the LSB of product
anumcp[i - 1] = anumcp[i];
}
anumcp[4] = temp2;
printf("\nAR-SHIFT: "); // display together
for (i = 4; i >= 0; i--)
{
printf("%d", pro[i]);
}
printf(".");
for (i = 4; i >= 0; i--)
{
printf("%d", anumcp[i]);
}
}

void main()
{
int i, q = 0;
```



```

printf("\t\tBOOTH'S MULTIPLICATION ALGORITHM");
printf("\nEnter two numbers to multiply: ");
printf("\nBoth must be less than 16");
// simulating for two numbers each below 16
do
{
printf("\nEnter A: ");
scanf("%d", &a);
printf("Enter B: ");
scanf("%d", &b);
} while (a >= 16 || b >= 16);
printf("\nExpected product = %d", a * b);
binary();
printf("\n\nBinary Equivalentents are: ");
printf("\nA = ");
for (i = 4; i >= 0; i--)
{
printf("%d", anum[i]);
}
printf("\nB = ");
for (i = 4; i >= 0; i--)
{
printf("%d", bnum[i]);
}
printf("\nB'+ 1 = ");
for (i = 4; i >= 0; i--)
{
printf("%d", bcomp[i]);
}
printf("\n\n");
for (i = 0; i < 5; i++)
{
if (anum[i] == q)
{ // just shift for 00 or 11
printf("\n-->");

```

```
arshift();
q = anum[i];
}
else if (anum[i] == 1 && q == 0)
{ // subtract and shift for 10
printf("\n-->");
printf("\nSUB B: ");
add(bcomp); // add two's complement to implement subtraction
arshift();
q = anum[i];
}
else
{ // add ans shift for 01
printf("\n-->");
printf("\nADD B: ");
add(bnum);
arshift();
q = anum[i];
}
}
printf("\nProduct is = ");
for (i = 4; i >= 0; i--)
{
printf("%d", pro[i]);
}
for (i = 4; i >= 0; i--)
{
printf("%d", anumcp[i]);
}
printf("\nD22DCE173");
}
```

Output:

```
⊗ Enter A: cd "/Users/ninadvyas/Desktop/" && gcc p-5.c -o p-5 && "/Users/ninadvyas/Desktop/"p-5
Enter B:
Expected product = 0

Binary Equivalents are:
A = 00000
B = 00000
B'+ 1 = 00000

-->
AR-SHIFT: 00000:00000
-->
AR-SHIFT: 00000:00000
-->
AR-SHIFT: 00000:00000
-->
AR-SHIFT: 00000:00000
-->
AR-SHIFT: 00000:00000
Product is = 0000000000
D22DCE190%
```

Fig:5.1

Conclusion:-

In this practical i have learned about how to perform booth algorithm using multiplication in c language.

Practical – 6

Aim: Implement a circuit in Logisim which perform Arithmetic and Logic unit.

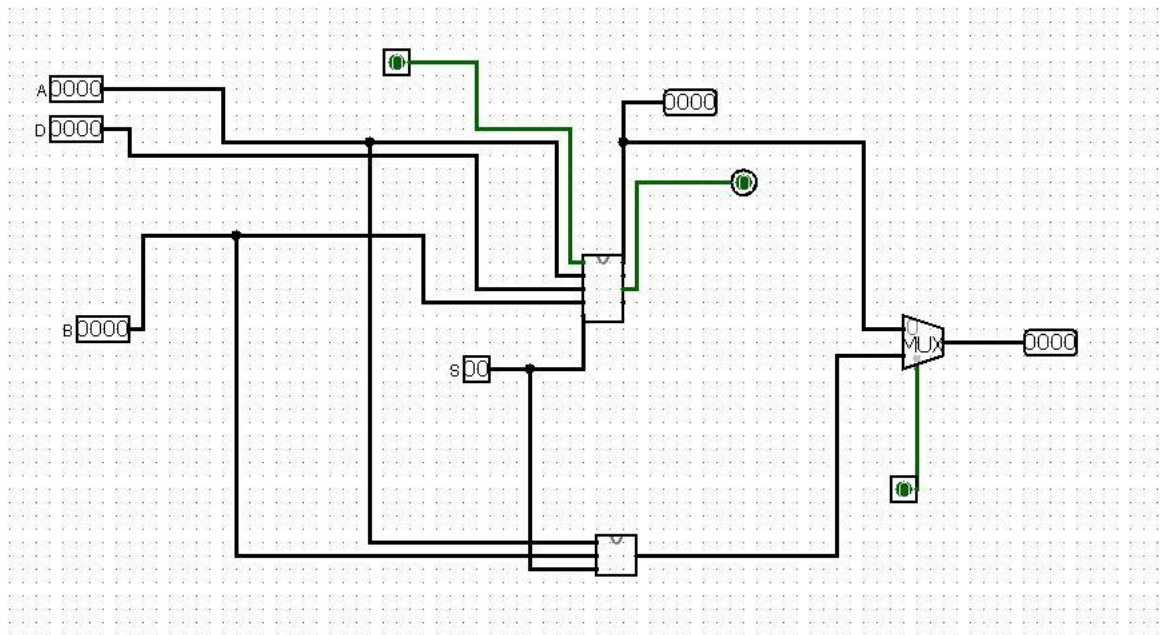


FIG 6.1 Main Circuit

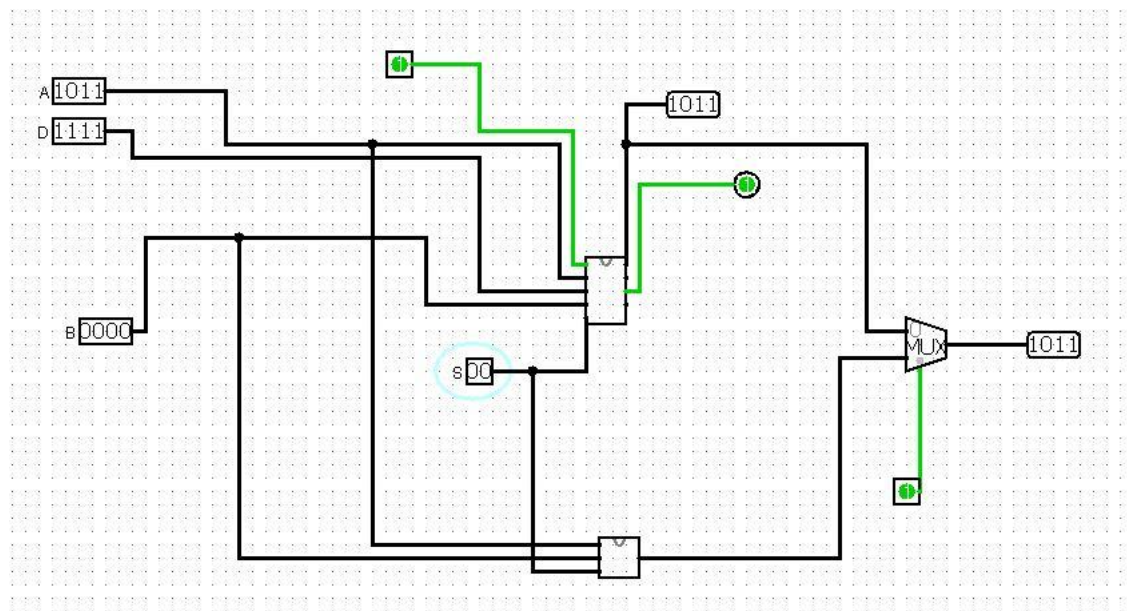


FIG 6.2 Output

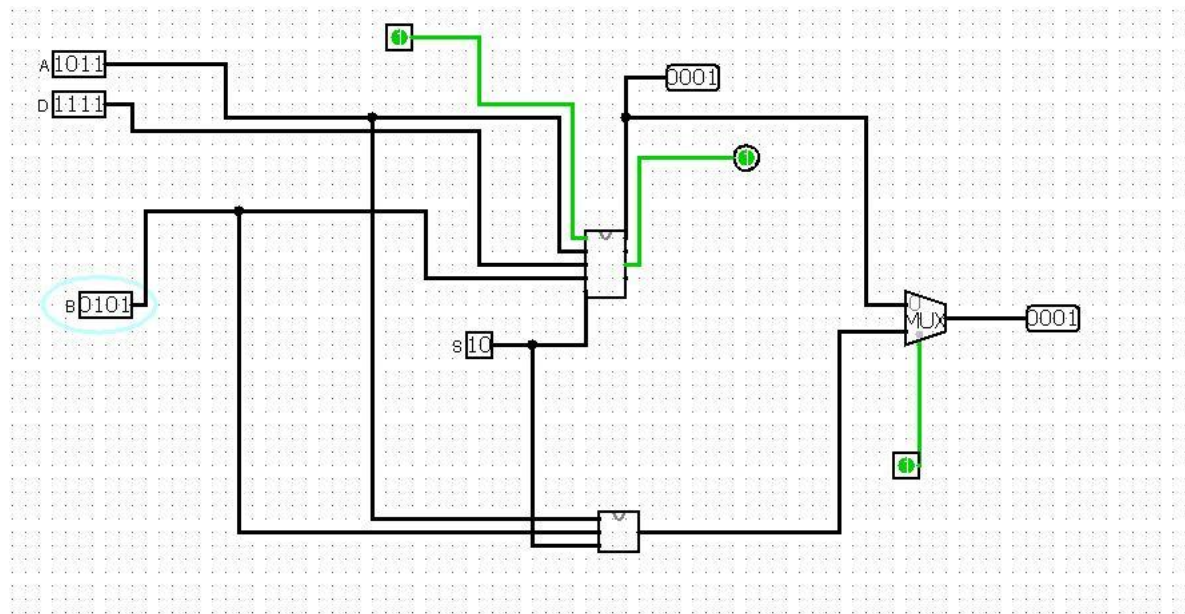


FIG 6.3 Output

Conclusion:

In this practical I have learnt how to design ALU (Arithmetic & Logical circuit) in Logisim.

Practical – 7

Aim: (Basics of assembly level programming) Perform following operations on 8-bit data

1. Addition:-

```
org 100h
MOV AL, 5
ADD AL, 3
ret
```



Fig:7.1

2.Addition With Carry:-

```
org 100h
STC
MOV AL, 5
ADD AL, 7
ret
```



Fig:7.2

2. Subtraction:-

```
org 100h
MOV AL, 5
SUB AL, 3
ret
```

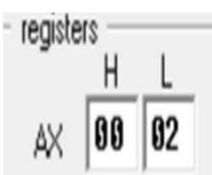


Fig:7.3

4.Subtract With Borrow:-

```
org 100h
MOV AL, 9
SBB AL, 5
ret
```

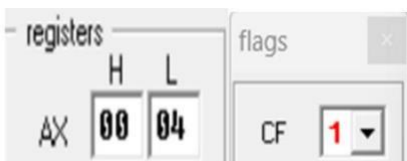


Fig:7.4

5.Multiplication:-

```
org 100h
MOV AL, 20
MOV BL, 4
MUL BL
RET
```



Fig:7.5

6.Signed Multiplication:-

```
org 100h
MOV AX, -20
MOV BL, -1
IMUL BL
RET
```



Fig:7.6

7.Division:-

```
org 100h
MOV AX, 200
MOV BL, 5
DIV BL
RET
```



Fig:7.8

8.Signed Division:-

```
org 100h
MOV AX, -20
MOV BL, 4
IDIV BL
RET
```

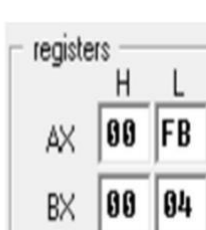


Fig:7.9

9.AND:-

Fig:7.10

10.OR:-

Fig:7.11

11.XOR:-

Fig:7.12

12.NOT:-

Fig:7.13

13.Logical Left Shift:-

Fig:7.14

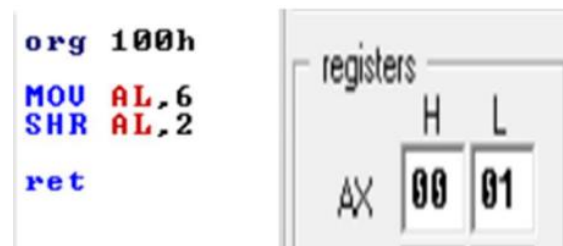
14.Logical Right Shift:-

Fig:7.15

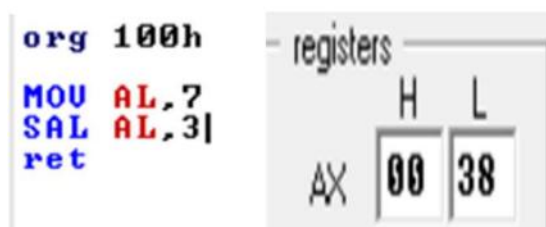
15.Arithmetic Left Shift:-

Fig:7.16

16.Arithmetic Right Shift:-

Fig:7.17

17.Rotate Left With Carry:-

Fig:7.18

18.Rotate Right With Carry:-

Fig:7.19

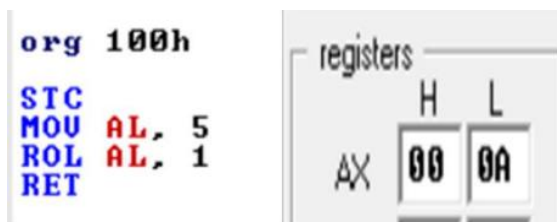
19.Rotate Left Without Carry:-

Fig:7.20

20.Rotate Right Without Carry:-

Fig:7.21

Conclusion:

I have learned how to give instruction in 8086 microprocessor.

I have also learned about how to perform various operation on 8086 microprocessor.

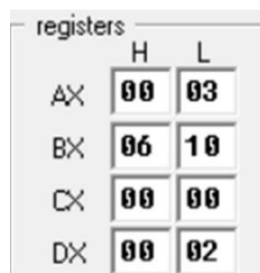
Practical – 8

AIM: (Array handling in Assembly level programming)

Create an array. Perform addition of all even numbers from array and save answer in one variable.

Code:

```
org 100h
mov [1000h],05h;
mov [1001h],02h;
mov [1002h],01h;
mov [1003h],08h;
mov [1004h],06h;
mov si,1000h;
mov cx,05h;
mov dl,02h;
mov bl,00h;
l1:
mov bh,[si];
mov al,bh;
div dl;
cmp ah,01h;
jz exit:
add bl,bh;
exit:
inc si;
loop l1
ret
```

Output:

	H	L
AX	00	03
BX	06	10
CX	00	00
DX	00	02

Fig:8.1 Output

Conclusion:

In this practical I learnt how to store data in array form and how to deal with loop instruction.

Practical – 9

Aim: (String Handling in Assembly level language) Find out whether the given string is palindrome or not and print appropriate message. Don't use procedure.

Code:

```
org 100h

jmp start

m1:
s db 'saras'
s_size = $ - m1
    db 0Dh,0Ah,'$'

start:

mov ah, 9
mov dx, offset s
int 21h

lea di, s
mov si, di
add si, s_size
dec si

mov cx, s_size
cmp cx, 1
je is_palindrome

shr cx, 1

next_char:
    mov al, [di]
    mov bl, [si]
    cmp al, bl
    jne not_palindrome
    inc di
    dec si
loop next_char

is_palindrome:

    mov ah, 9
    mov dx, offset msg1
    int 21h
```

```
jmp stop
```

```
not_palindrome:
```

```
    mov ah, 9
```

```
    mov dx, offset msg2
```

```
    int 21h
```

```
stop:
```

```
mov ah, 0
```

```
int 16h
```

```
ret
```

```
msg1 db " this is palindrome!$"
```

```
msg2 db " this is not a palindrome!$"
```

Output:

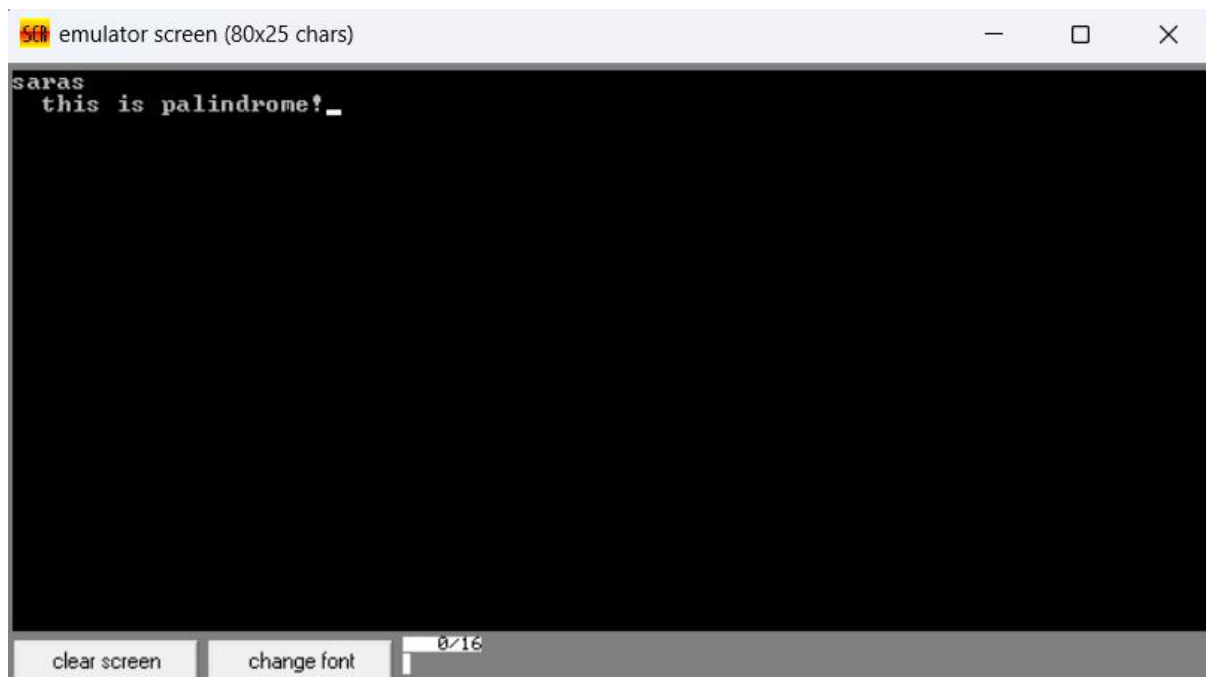


Fig 9.1:Output

Conclusion:

In this practical I have learnt how to Find out whether the given string is palindrome or not and print appropriate message.

Practical – 10

Aim: (Procedure in Assembly Level Language) Write an assembly code to evaluate the answer of below given series and store the answer in ANS variable. Program should have only one procedure to compute factorial of number. Series: $1! - 2 + 3! - 4 + 5! - 6 + 7! - 8 + 9! - 10$.

Code:

```
org 100h
```

```
MOV BL,02H
MOV SI,2000H
MOV CX,06H
MOV DL,00H
```

```
L1:
MOV AL,[SI]
DIV BL
CMP AH,00H
JZ L2
JNZ L3
JMP EXIT
```

```
L2:
SUB DL,[SI]
INC SI
LOOP L1
JMP EXIT
```

```
L3:
MOV BH,[SI]
MOV AX, 0001H
```

```
LOOP_START:
MUL BH
DEC BH
JNZ LOOP_START
```

```
ADD DX,AX
INC SI
LOOP L1
```

```
EXIT:
Ret
```

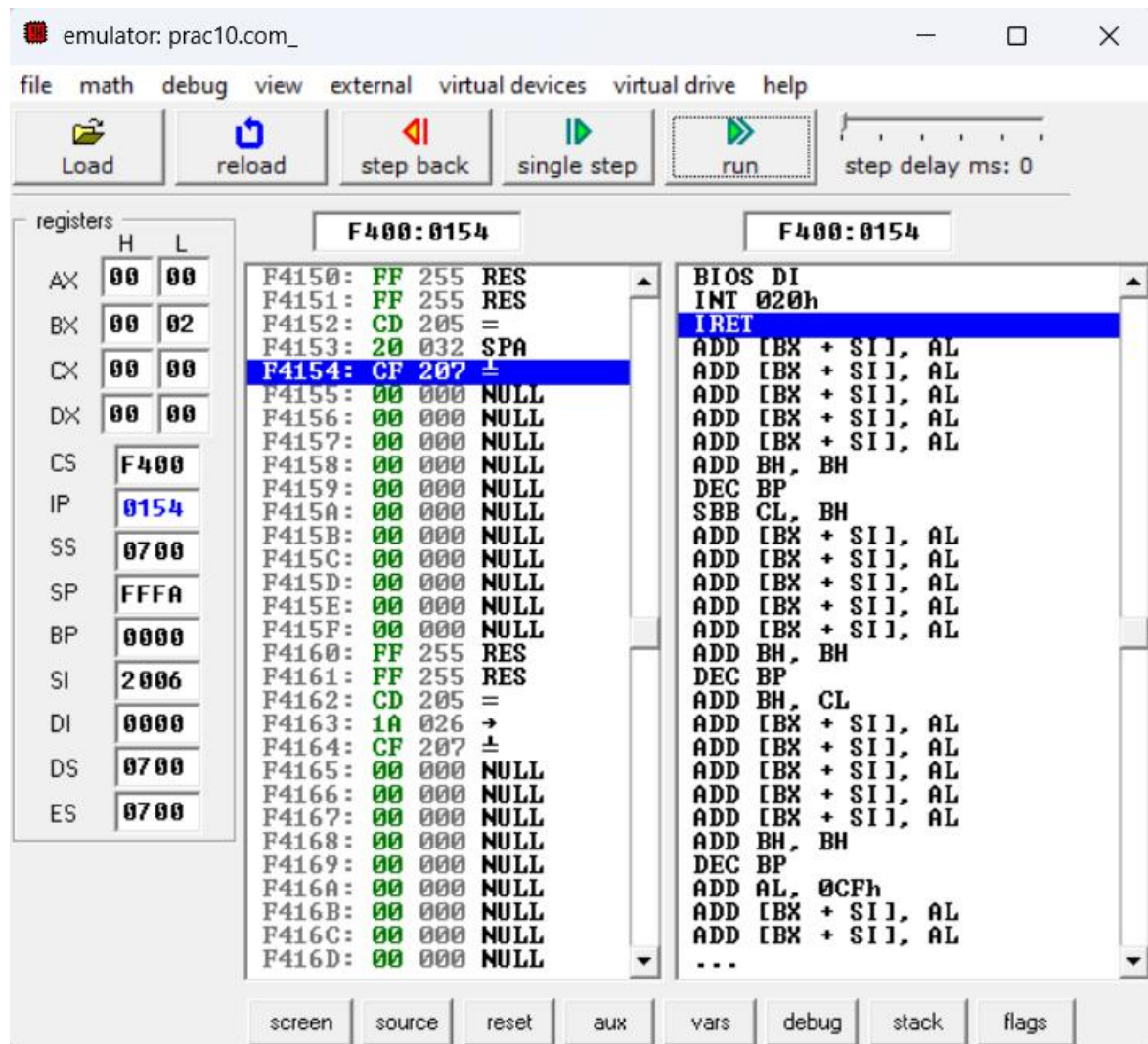
Output:

Fig 10.1:Output

Conclusion:

In this practical I have learnt how to Write an assembly code to evaluate the answer of blow given series and store the answer in ANS variable. Where series is $1! - 2+3! - 4+5! - 6+7! - 8+9! - 10$.