



Shifa Tameer-e-Millat University

شفا تعمیرِ ملّت یونیورسٹی

SHIFA TAMEER-E-MILLAT UNIVERSITY

Lab Manual

Introduction to Computer Science

Course Instructor

Lab Instructor

DEPARTMENT OF COMPUTING

Table of Contents

Lab 1: Look inside the box (Hardware).....	4
Lab 2: Installation of Ubuntu/Windows.....	16
Lab 3: Basic Linux Commands.....	32
Lab 4: Basic Linux Commands.....	47
Lab 5: Basic Linux Commands.....	59
Lab 6: LMC Programming.....	69
Lab 7: LMC Programming.....	81
Lab 8: LMC Programming.....	93
Lab 10: Bash Scripting.....	105
Lab 11: Bash Scripting.....	115
Lab 12: Bash Scripting.....	125
Lab 13: LaTeX.....	133
Lab 14: LaTeX.....	145
Lab 15: LaTeX.....	158

INTRODUCTION TO COMPUTER SCIENCE

Lab 1

Look inside the box (Hardware)

Lab 1: Look inside the box (Hardware)

1. Introduction

In this lab, we will learn about computers, the components of computers, and the different parts of computers. Most of you use computers for different purposes but most probably you haven't tried to look inside the computers. At the start of this lab, there will be informal activity in which Instructor and every student will introduce himself. Then, in the second part of the lab, you will be given an overview of computers and their structure and parts. Then in the third section, we will do some practical work we will make the groups of students and will open the computer box in front of each group one by one and then everyone from the group will do the practice to open every part of the computer and then fix each part to its right place.

In next labs you will be provided with a lab manual before lab so that you should read it and have an idea about the things that we will do in the lab.

2. Activity Timeboxing

Task No	Activity Name	Activity Time	Total Time
	Introduction	30 minutes	30 minutes
4	Lab Outline	20 minutes	50 minutes
5	Grading	15 minutes	65 minutes
6	Concept Map	40 minutes	105 minutes
7	Walkthrough task	45 minutes	150 minutes

Table 1: Activity Timeboxing

3. Objective

- To get a basic understanding of Computers
- To get a basic understanding of the structure of Computers
- To get a basic understanding of parts of Computers
- To practice identifying any part and how to replace it.

4. Outline for ICS Labs

The outline for the labs of this course is uploaded on LMS. You can download and see it. We will discuss it in the lab.

5. Grading Criteria

ICS course has 25% Lab work. Every Lab is very important. Each lab will have assignments, Unseen Tasks during Labs and Quizes. We will discuss it in details in during Outline presentation.

Each lab has individual work. Sharing the solution will be considered as a violation of the honor code and any suspicious activity may be referred to the disciplinary committee. In case of any help, please consult the instructor. Also late submission will not be accepted.

6. Concept Map

In this lab, we will look inside the computer box and familiarize ourselves with the components of the computer and its functionalities.

6.1. Computer

A computer is an electronic device that accepts some data, perform mathematical and logical operations or process information at high speed and displays or store the results of these operations.

6.2. Components of a Computer System

All types of computers have three basic types of components to perform operations for converting input data into useful information. These components are the Input Unit, CPU, Output Unit as shown in Figure 1.

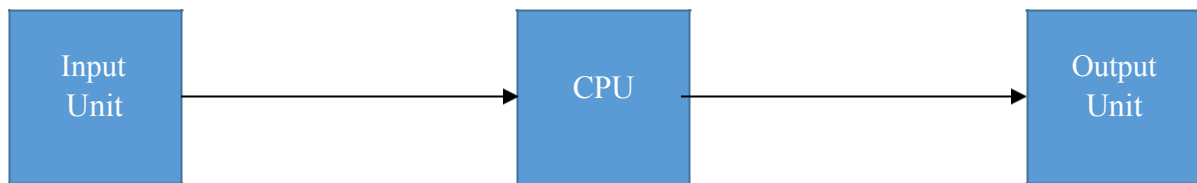


Figure 1: Components of a computer

6.2.1. Input Unit

The input unit contains those devices that help the user to enter data and commands into a computer. In other words, we can say that this unit links users and computers. Few input devices used in computers are given below and we explain them one by one.

- Keyboard
- Mouse
- Microphone
- Scanner
- Bar Code Reader
- Optical Character Reader

6.2.1.1. Keyboard



Figure 2: Keyboard

A keyboard shown in [Figure 2](#) is a very common input device. It is used to input data to the computer. Using a keyboard, a user can type a document, play games, and use keyset shortcuts. Most keyboards have between 80 and 110 keys.

6.2.1.2. Mouse



Figure 2: Mouse

A mouse is also an input unit for entering directions and commands ([Figure 2](#)). It is a small palm-size box. Generally, it has two buttons i.e. left and right buttons and a scroll bar that is present at mid. It can't be used to enter the text rather it is used to control the position of the cursor on the screen. It makes it easy to use a computer and moves the cursor faster than the arrow keys of the keyboard.

6.2.1.3. Microphone



Figure 3: Microphone

It is an input device to input sound to computers ([Figure 3](#)). There are two types of computer Microphones: **Internal** and **External**. Microphones can be used for voice recording and online chatting. These are also used for computer gaming.

6.2.1.4. Scanner



Figure 4: Scanner

It is also an input device, shown in Figure 4, which is used when some information is available on paper and we want to save it on the computer's hard disk. It works like a photocopy machine. It captures the images of paper and then converts them into a digital form that can be stored on a computer.

6.2.1.5. Barcode Reader



Figure 5: Barcode Reader

It is an input device capable of reading a bar code using a laser (Figure 5). The most common use of barcode reader is in supermarkets where it is used to read the bar code and log the price of a product.

6.2.2. Central Processing Unit (CPU)

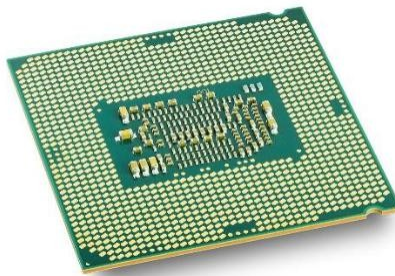


Figure 6: CPU

CPU is also known as the brain of the computer (Figure 6). After receiving data or commands from the user through the input unit, the computer has to process it according to the instructions provided. So

CPU is the component of the computer that does this job. It controls the operations of all parts of a computer. It has three components: (1) Memory Unit (2) Arithmetic Logic Unit (3) Control Unit

6.2.2.1. Memory Unit



Figure 7: Memory Unit

This unit can store data, instructions and results. All inputs and outputs are transmitted through the main memory (Figure 7). Once a user inputs data the computer stores this data in its memory unit then the CPU process data. Internal Storage Unit, Random Access Memory (RAM), Main memory are also used for this term and all of them are the same thing.

Its main functions are:

- It stores the data to be processed and required instructions to process data.
- It stores intermediate results for processing.
- Final results are also stored on it before transmitting to the output device.
- All inputs and outputs are transmitted through the memory unit.

6.2.2.2. Control Unit

The Control Unit is also known as the backbone of computers. Coordination of tasks of different components of a computer is done by the Control Unit.

Its main functions are:

- It is responsible for controlling the transfer of data and instructions among other units of a computer.
- It is responsible for the coordination of all units of a computer.
- It gets instructions from the main memory, interprets instructions and directs the operation of the computer.
- It is also responsible to communicate with output devices for the transfer of results from memory.
- It does not store data, it just controls the operations of all parts of the computer.

6.2.2.3. Arithmetic Logic Unit (ALU)

It is that part of the CPU that performs arithmetic and logical operations. Based upon its functionality it is further categorized into two subsections: Arithmetic and logic section.

Arithmetic Section:

As the name suggests, this section performs arithmetic operations like addition, subtraction, multiplication and division. All other complex operations are done by making repetitive use of the above operations.

Logic Section:

This section performs logic operations such as comparing, matching, selecting and merging data.

6.2.3. Output Unit

The output Unit contains those devices that help the user to get information or result from the computer. As we studied earlier in the input unit section the Input unit was a link between user and computer, here the output unit is the link between computer and user. These devices translate the output and results of a computer into human-readable. . Few output devices are given below and we explain them one by one.

- Monitor
- Printer
- Speakers
- Headphones
- Projector
- Video Card

6.2.3.1. Monitor



Figure 8: Monitor

A monitor, as shown in [Figure 8](#), is the most common and main output device. It is also known as the Visual Display Unit (VDU). It creates a visual display in front of the user by taking data from the computer.

6.2.3.2. Printer



Figure 9: Printer

It is also an output device to print the processed data/result on paper (Figure 9). It creates a hard copy of the electronic data sent from a computer.

6.2.3.3. Speakers



Figure 10: Speakers

These are the devices that transform signals from the computer's sound card into audio (Figure10)

6.2.3.4. Headphones



Figure 11: Headphones

These are also the output device that output audio from computers (Figure 11). They are also known as an earphone. Similarly, Headset is a combination of speakers and mic that are used for communication with family and friends over the internet.

6.2.3.5. Projector



Figure 12: Projector

This output device projects computer video or image onto another surface (Figure 12). That surface is usually a whiteboard, screen, or wall. Video or image data is transmitted by the computer to its video card, which then sends it to the projector. It is used for presentations, teaching, watching movies because it enables all the people in a room to see the image or video.

6.2.3.6. Video Card



It generates output images to display. This is also known as a graphic card.

6.3. Some other Components of Computer

6.3.1. Motherboard



Figure 13: Motherboard

A motherboard, as shown in Figure 13, is the computer's main circuit board. It holds the CPU, memory, connectors for the hard drive, expansion cards to control audio and video and all other connections to computer ports such as USB ports. It is connected directly or indirectly with every other part of the computer.

6.3.2. Hard Drive

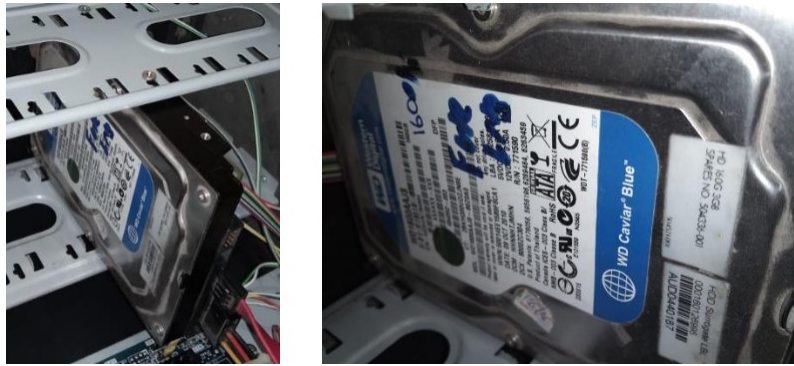


Figure 14: Hard Drive

A hard drive is a data storage device in a computer where the operating system, software, and other documents and files are stored (Figure 14). Unlike Main Memory (RAM) the data is still saved even if you restart your computer.

6.3.3. Power Supply Unit



Figure 15: Power Supply Unit

A power supply converts the main AC electricity to low voltage DC voltage to provide power to the motherboard and other components in the computer (Figure 15).

7. Walkthrough Tasks

7.1. Group Formation

It is an activity in the lab in which groups will be formed depending upon the computer boxes that will be provided for this lab. All the students will work in groups for this lab.

7.2. Walkthrough tasks

After this activity, the lab instructor will discuss the components and different parts of the computer by opening the box in front of every group. After that students will discuss it in their group.

8. Practice Tasks

Open the computer box and discuss the different parts of the computer in your group.

9. Related Material

1. [How To Identify The Components Inside Your Computer](#)
2. [What does what in your computer? Computer parts Explained](#)
3. [How Computers Work: What Makes a Computer, a Computer?](#)



SHIFA TAMEER-E-MILLAT UNIVERSITY

Lab Manual Introduction to Computer Science

DEPARTMENT OF COMPUTING

Shifa Tameer-e-Millat University

Department Of Computing

INTRODUCTION TO COMPUTER SCIENCE

Lab 2

Installation of Ubuntu/Windows

Lab 2: Installation of Ubuntu/Windows

1. Introduction

In this lab, we will learn that how can we install an operating system (Ubuntu/Window) on a computer. We will install these operating system in VirtualBox. Installing an operating system in VirtualBox has a lot of benefits we will discuss them in our Lab. VirtualBox is virtualization software for creating virtual machines. With the help of a virtual machine, you can run an operating system like another application inside your current operating system. More simply we can say it as a computer inside a computer.

2. Activity Timeboxing

Heading No	Activity Name	Activity Time	Total Time
	Recap Lab1	30 minutes	30 minutes
5	Walkthrough Task	30 minutes	60 minutes
6	Submission	90 minutes	150 minutes

Table 1: Activity Timeboxing

3. Objective

- To install an operating system on your computer.
- Installation of the operating system inside the operating system

4. Concept Map

An Operating System (OS) is an interface between a user and computer hardware. An operating system is a software that performs all the basic tasks like file management, memory management, process management, handling input and output, and controlling peripheral devices such as disk drives and printers. There are three most common operating systems for PCs: Microsoft Windows, Linux, and macOS. In this lab, we will install Ubuntu and windows in VirtualBox that will help you to install these operating systems on your Laptops and PCs. Ubuntu is a Linux-based operating system. Ubuntu is a distribution or distro of Linux.

5. Walkthrough Task

5.1. Installation of Ubuntu in VirtualBox

As already told in the introduction that we will install Operating System in VirtualBox and it is placed on the server as well as on Lab PCs. If you are working in lab then you can skip **Step 1** and **Step 2** but if you want to install it on your computer and don't have access to the server or lab PCs then follow this manual from **Step 1**.

Step 1: Download VirtualBox

Go to the VirtualBox website and download the binary for your operating system (Figure 1)

For downloading it [click here](#). The following screen will be opened. Note: you can also access it from the server or Lab PC. It is available in the software folder on the server.

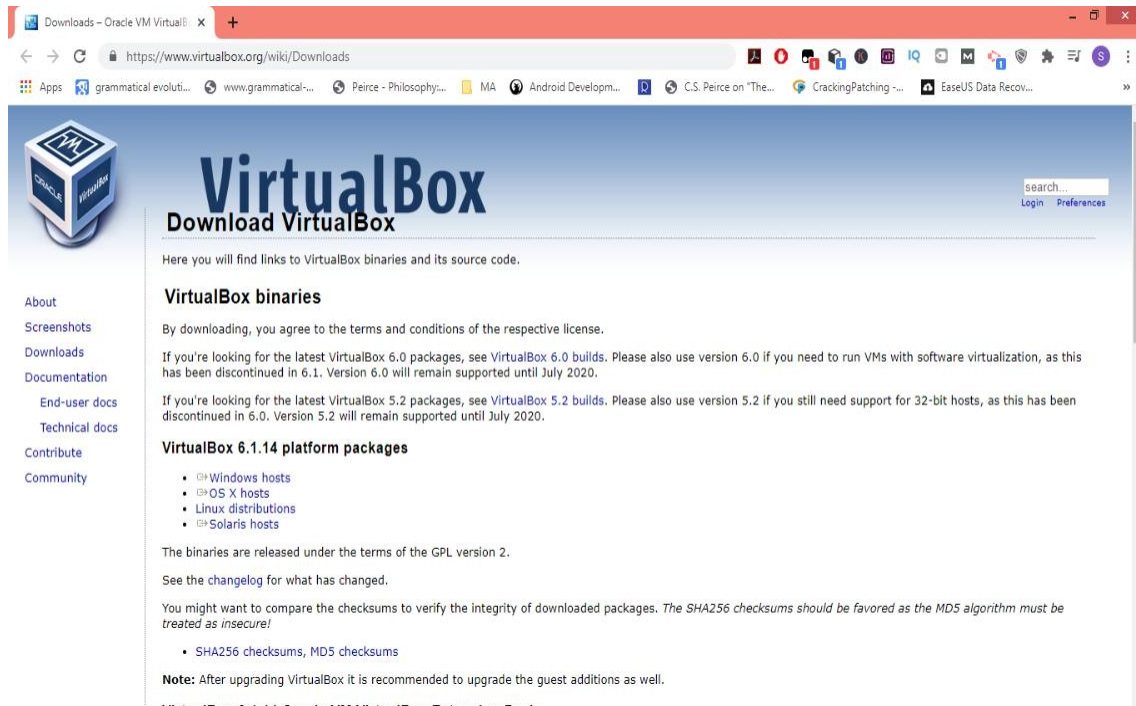


Figure 1: Download VirtualBox

Step 2: Download Ubuntu File

Now download the file of the operating system you want to install on VirtualBox. In our case, we are installing Ubuntu so visit the official Ubuntu website and download the file of Ubuntu Installer ([Figure2](#)). To download it [click here](#). (You can also copy it from the server or Lab PC).

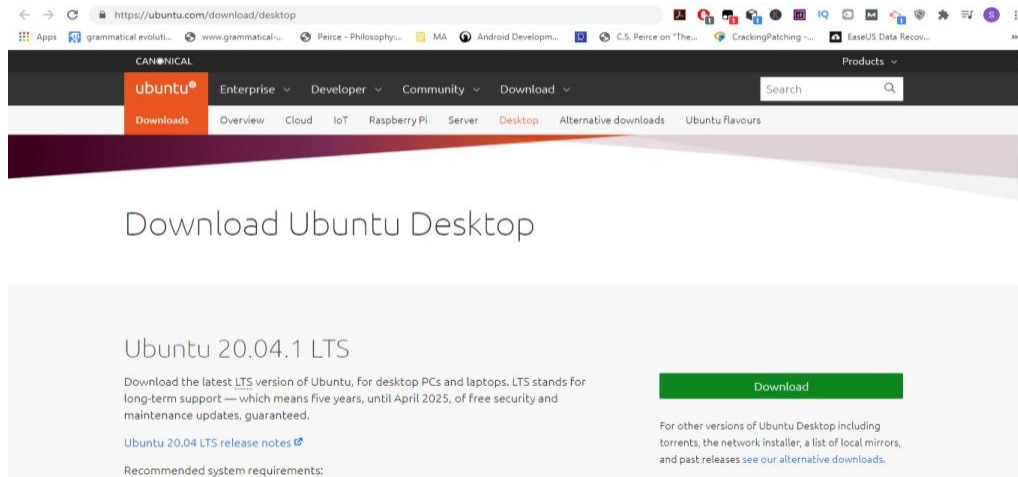


Figure 2: Download Ubuntu

Step 3: Install VirtualBox

Now double-click the Virtualbox file that you have downloaded or copied from the server in step 1. Then follow the instructions shown on the screen. You need not be an expert for it. You have to only press the next button for 2, 3 screens, and then press the install button on one screen and it will ask for permission you have to give permission and then click the “Finish” button. Then the following screen will be displayed as shown in (Figure 3).

(Note: This step was for window users that already use the window on their computers. The User of other operating systems should follow the installation procedure given on the internet or may contact the lab instructor).



Figure 3: installation of VirtualBox

Step 4: Create a Virtual Machine for Ubuntu

To create a new virtual machine for installing Ubuntu on VirtualBox, click the 'New' button to open a dialog as shown in Figure 4.



Figure 4: Creating a Virtual Machine

Enter a name for the new virtual machine. As we are going to install Ubuntu 20.04, we'll enter 'ubuntu2004'. VirtualBox automatically changes 'Type' to Linux and 'Version' to 'Ubuntu (64 bit)'. These fields are exactly what we need as shown in [Figure 5](#).



Figure 5: Selecting the type of operating system

After pressing next in [Figure 5](#), allocate the memory (RAM) for Ubuntu. You can allocate any value in the range of the green bar. It is recommended to allocate within the range of the green bar. In our case, we will provide half of our total memory. In my case, it is 8192 MB so I will provide 4096 MB. But if you have a total of 4096 MB memory then you should provide 2048 MB. And press the “Next” button ([Figure 6](#))

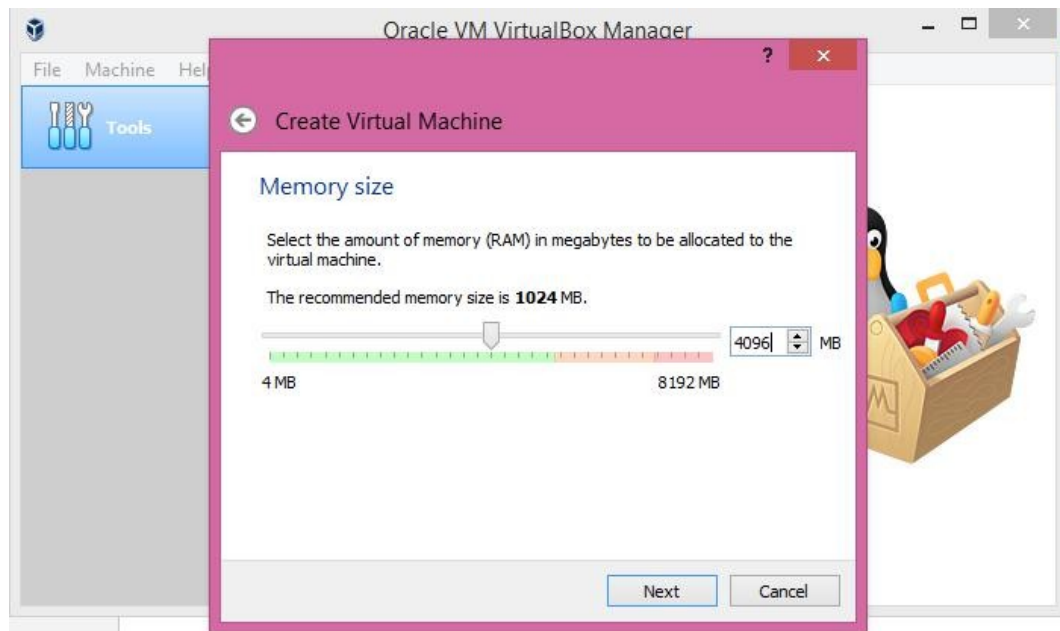


Figure 6: Allocating memory size

After pressing “Next” in Figure 6, select the default 'Create a virtual hard drive now' and click the 'Create' button (Figure 7)

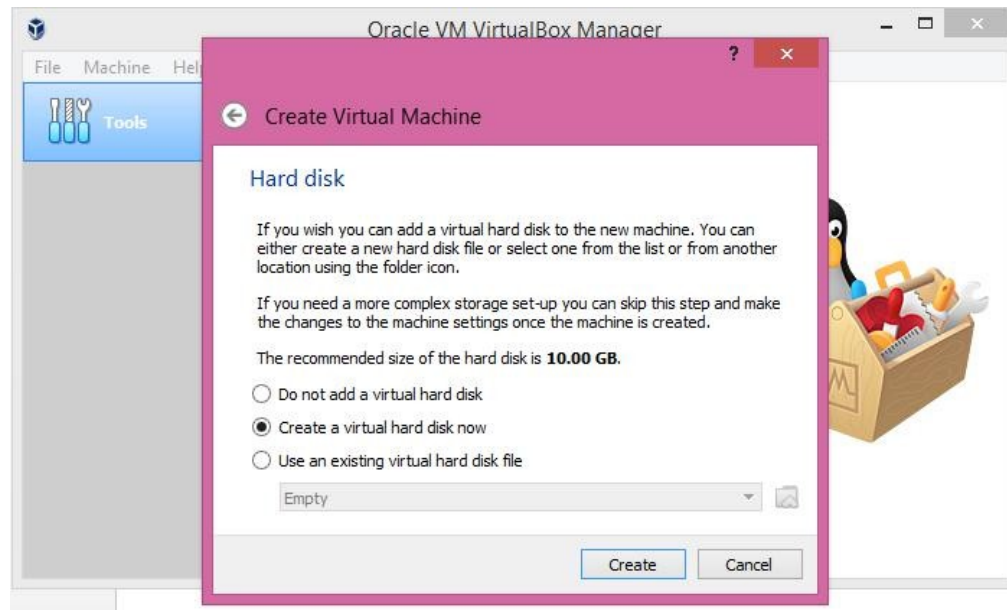


Figure 7: Creating a Virtual Hard Disk

In Figure 8, select the default 'VDI (VirtualBox Disk Image)' drive file type and click the 'Next' button.

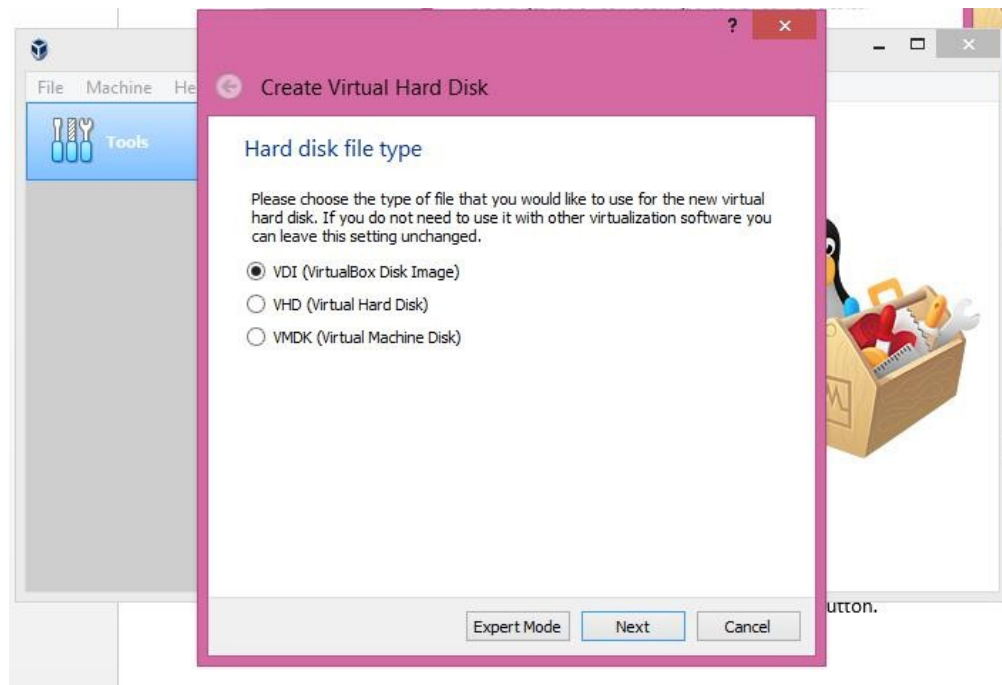


Figure 8: Hard Disk File Type

In Figure 9, let the options be the default (Dynamically allocated)

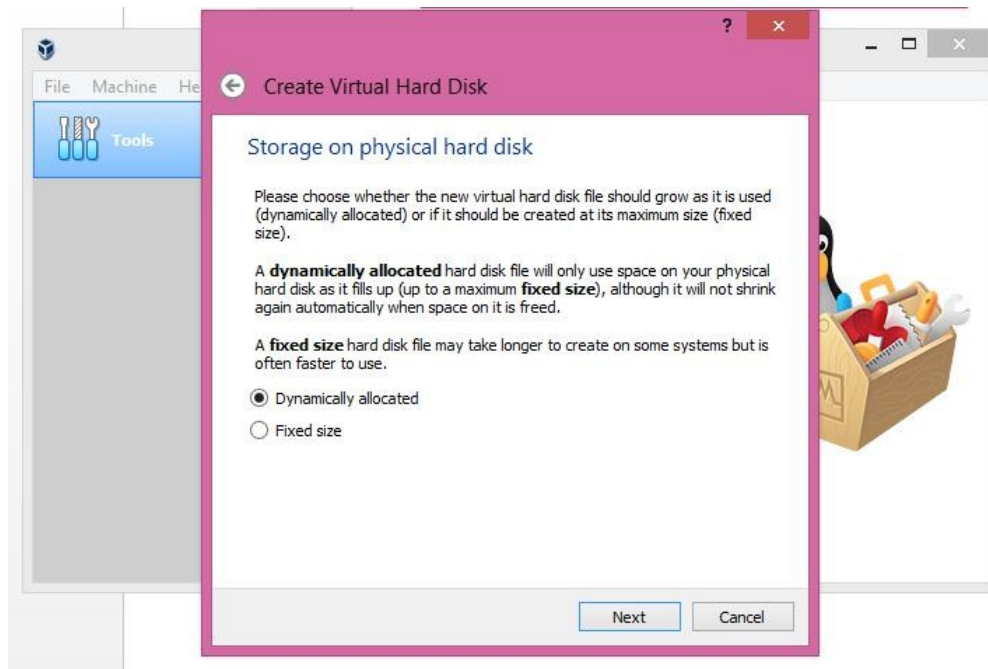


Figure 9: Storage of Hard Disk

In Figure 10, Enter the size of the virtual hard disk whatever you want (depending upon available size in your partition where you are installing VirtualBox) and Click “Create”

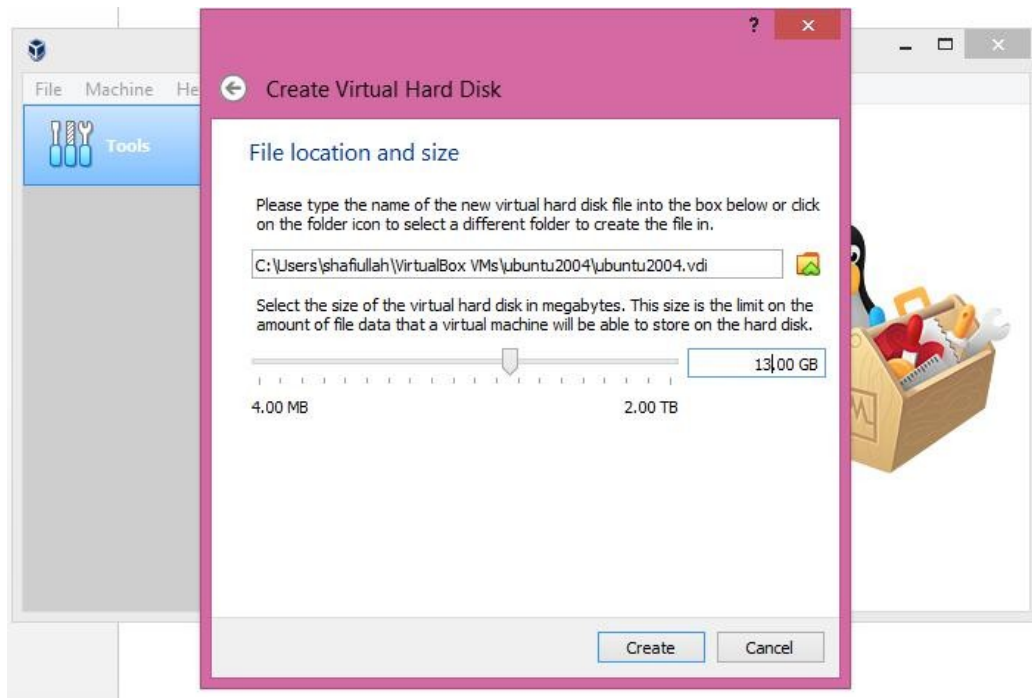


Figure 10: Size of Virtual Hard Disk

Now the virtual machine is created and we can install Ubuntu in this virtual machine.

Step 5: Ubuntu Installation

We have created a virtual machine in the above steps. In this part, we will install Ubuntu in this virtual machine. Select the VM that we have just created and select the “Settings” button and anew window will be opened as shown in Figure 11.

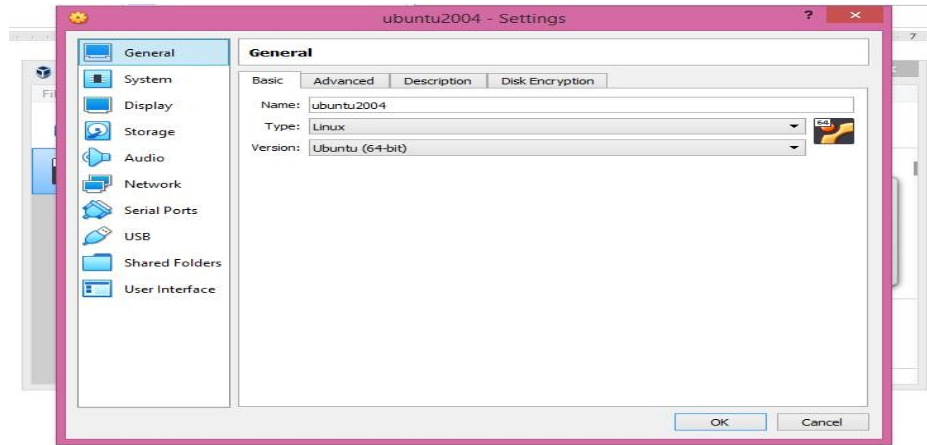


Figure 11: VM Settings (a)

Select the Storage from Figure 11, then 'Empty' under Controller: IDE. Click the "CD/DVD" icon on the right-hand side and select the Ubuntu ISO file to mount and press the “OK” button. (Ubuntu ISO file is that file that we have downloaded from Ubuntu official website in Step 2) and now the VM settings window will look like Figure 12.

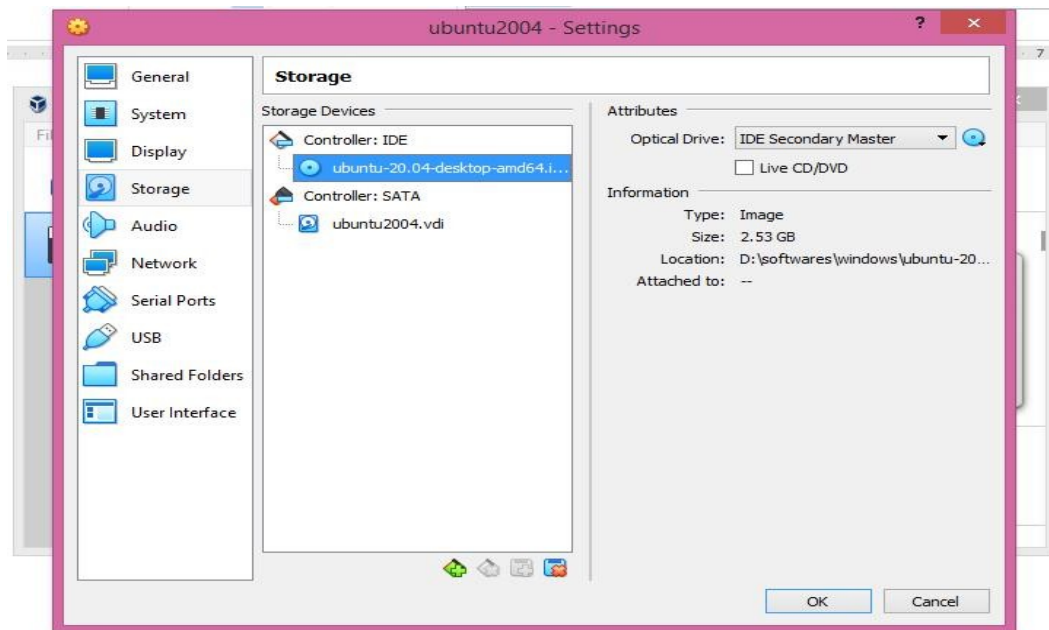


Figure 12: VM settings (b)

In Figure 13, select your Virtual Machine and Press Start Button

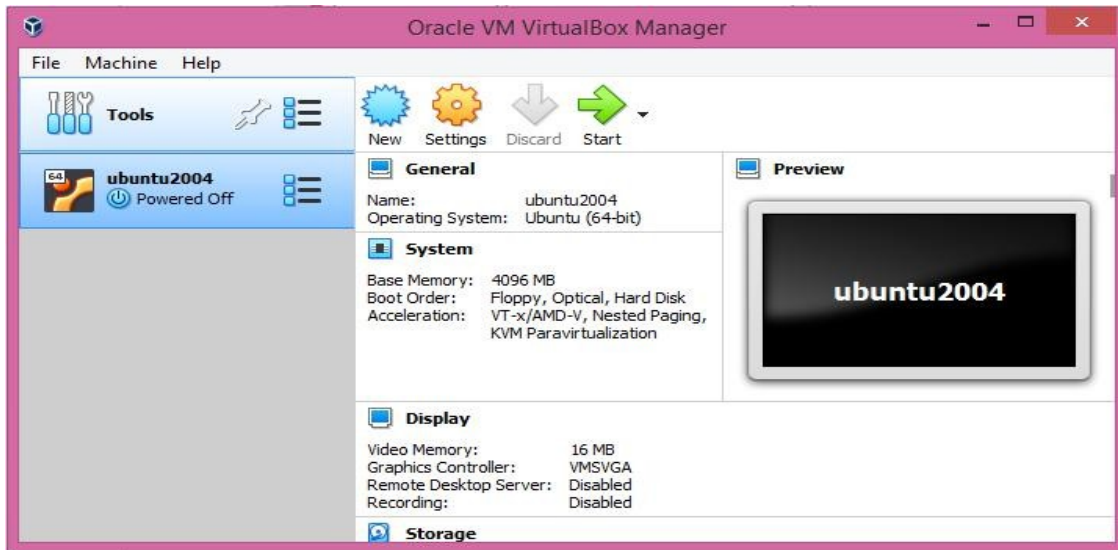


Figure 13: starting VM

After pressing the “start” button in Figure 13, a new window will be open where Ubuntu will be installed in your Virtual Machine (Figure 14).

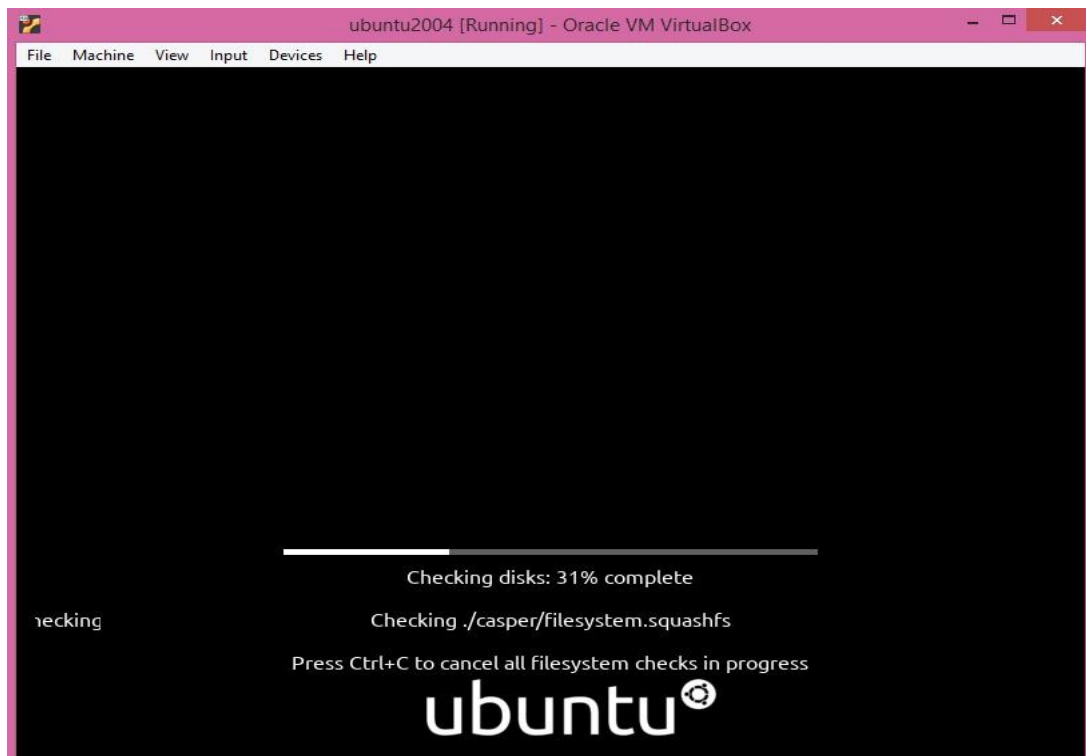


Figure 14: Installing Ubuntu (a)

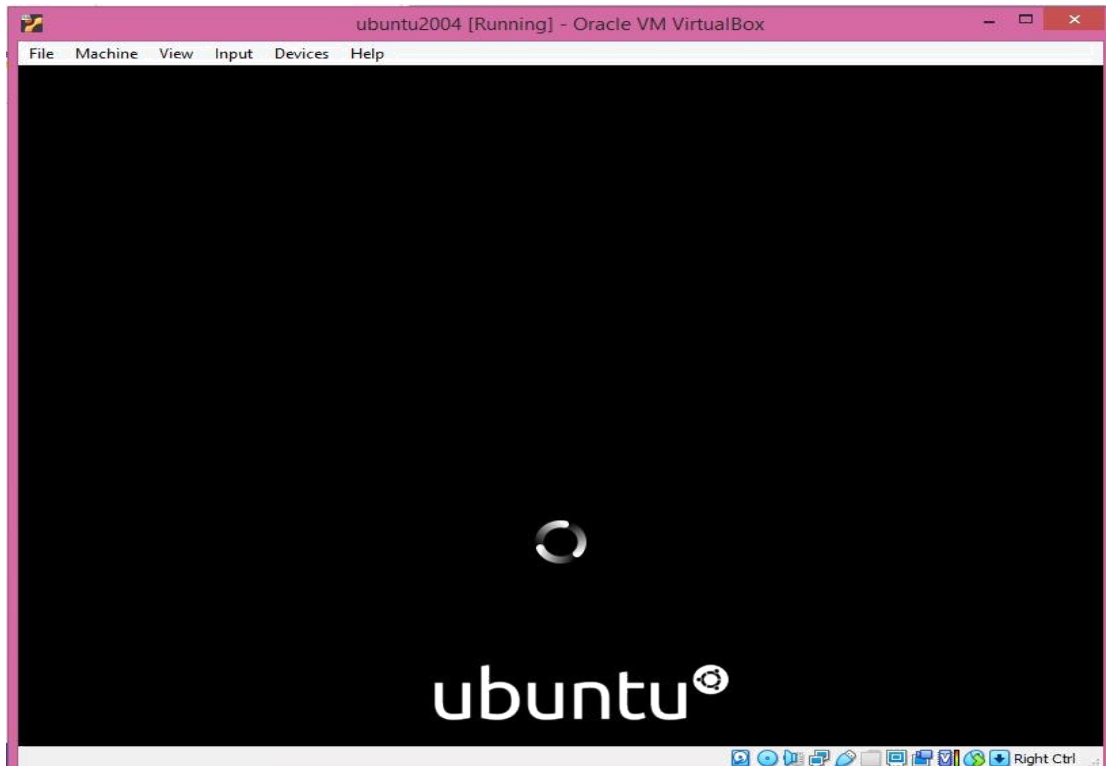


Figure 15: Installing Ubuntu (b)

To install Ubuntu press “Install Ubuntu” highlighted in Figure 16: Installing Ubuntu (c)

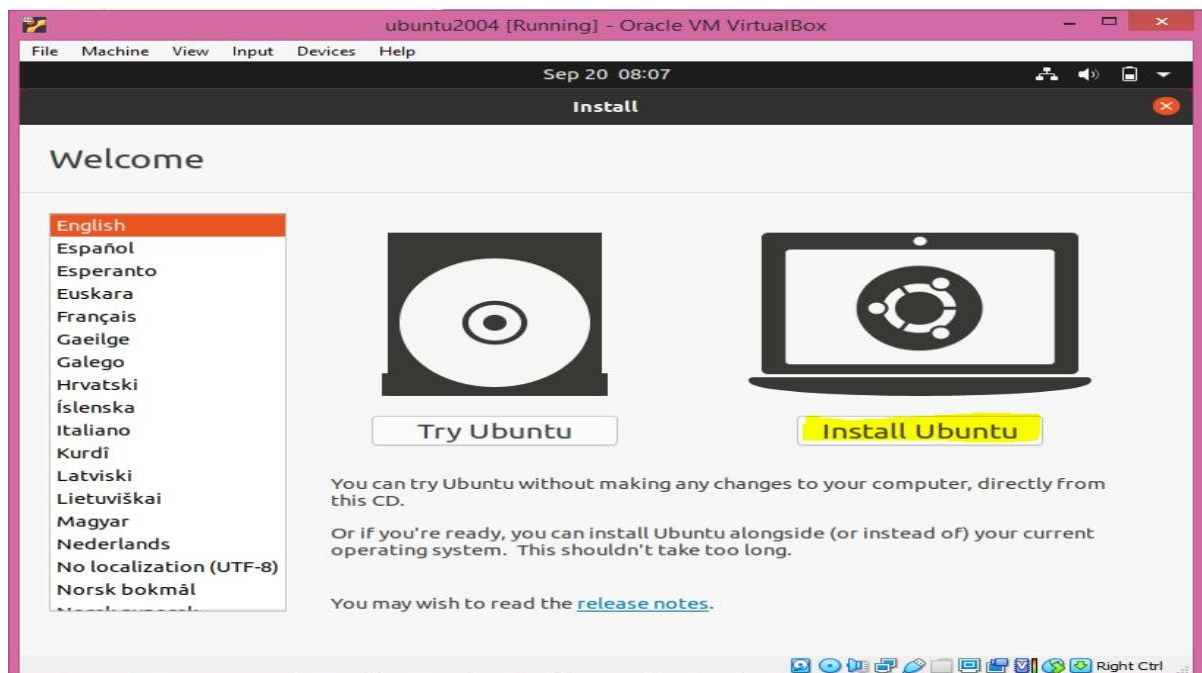


Figure 16: Installing Ubuntu (c)

Let the keyboard layout be the same as default and press the “Continue” button (Figure 17).

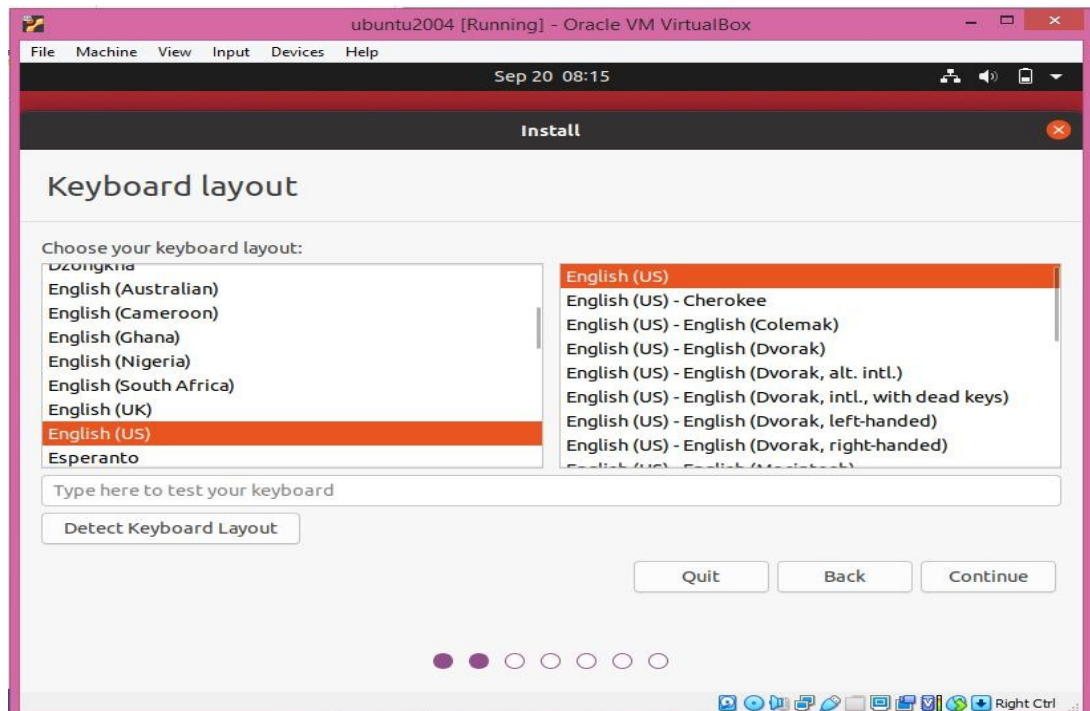


Figure 17: Installing Ubuntu (d)

Now select the minimal installation and press the “Continue” button (Figure 18)

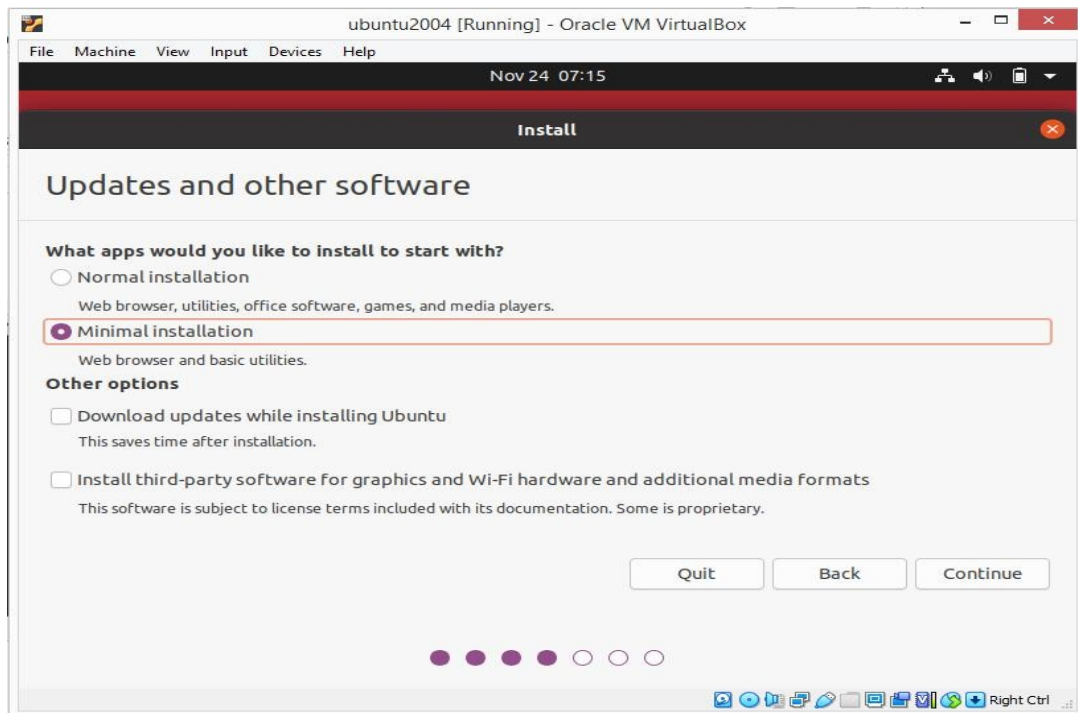


Figure 18: Installing Ubuntu (e)

Let the settings be the default for next following screen and press “Install Now” (Figure 19)

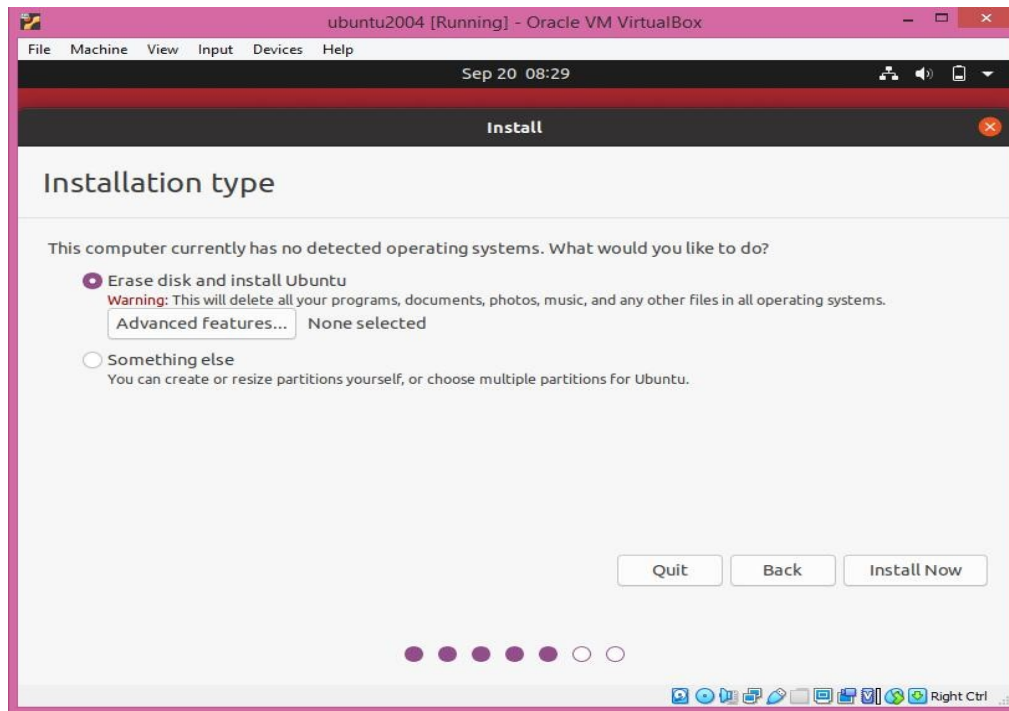


Figure 19: Installing Ubuntu (f)

Press Continue for the following screen (Figure 20)

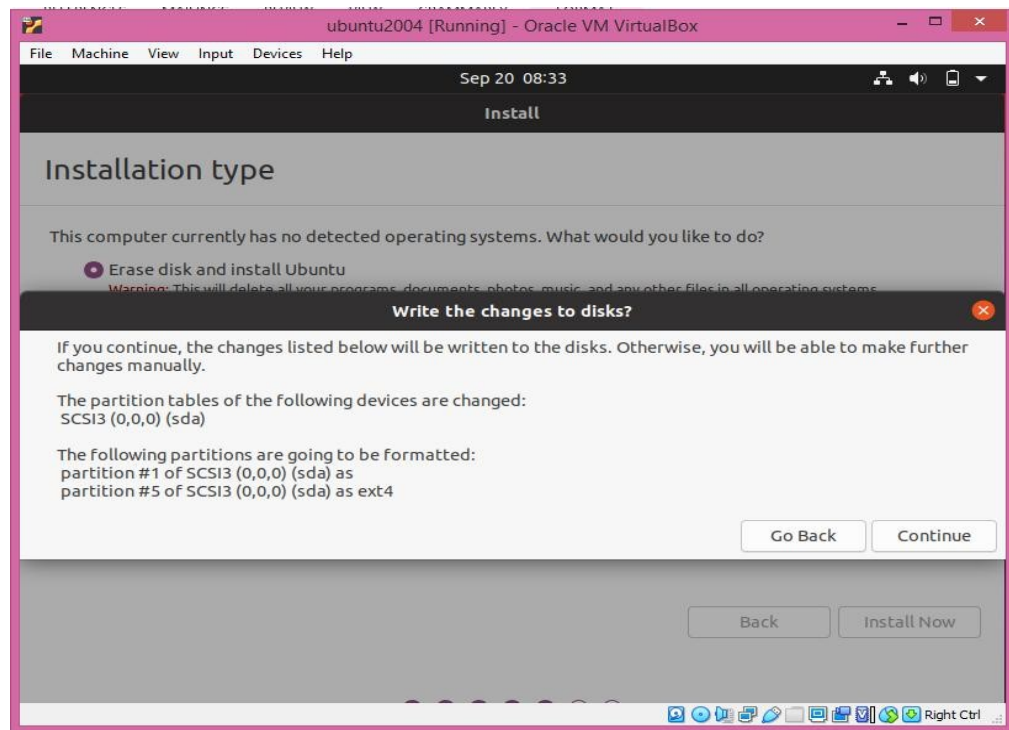


Figure 20: Installing Ubuntu (g)

Select your location and press “Continue” (Figure 21).

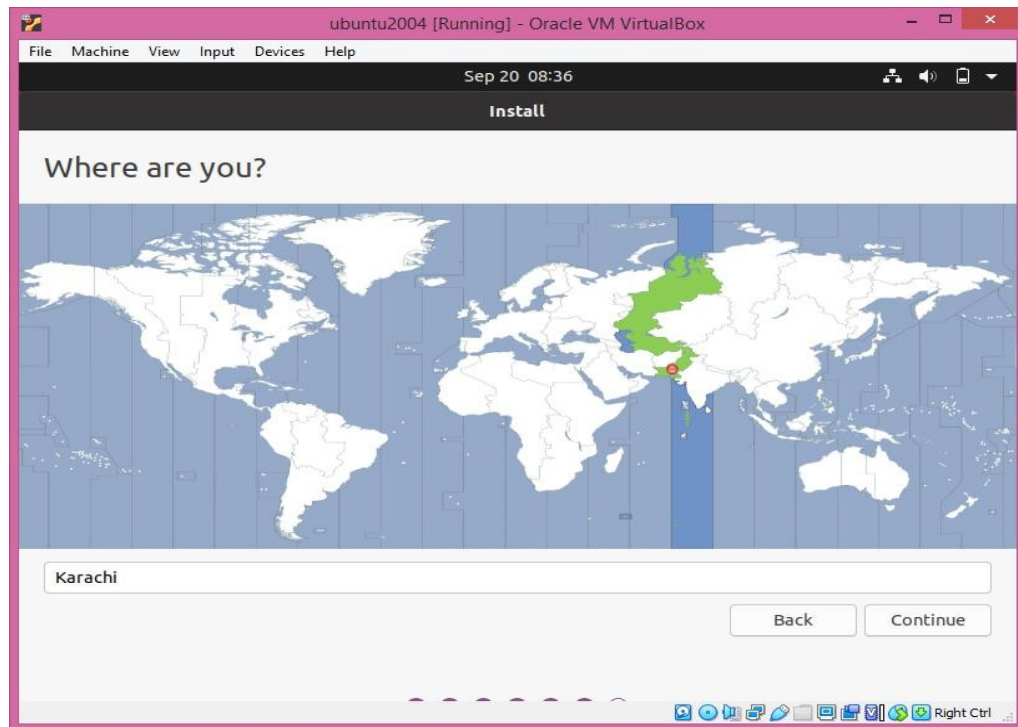


Figure 21: Installing Ubuntu (h)

Fill all the details in “Who are Screen” and click “Continue” (Figure 22)

Note: in the following screen fill your details. Your name should be like “YourName”, Computer’s Name should be like “YourNameRollNo-PC” and username be like “YourName”.

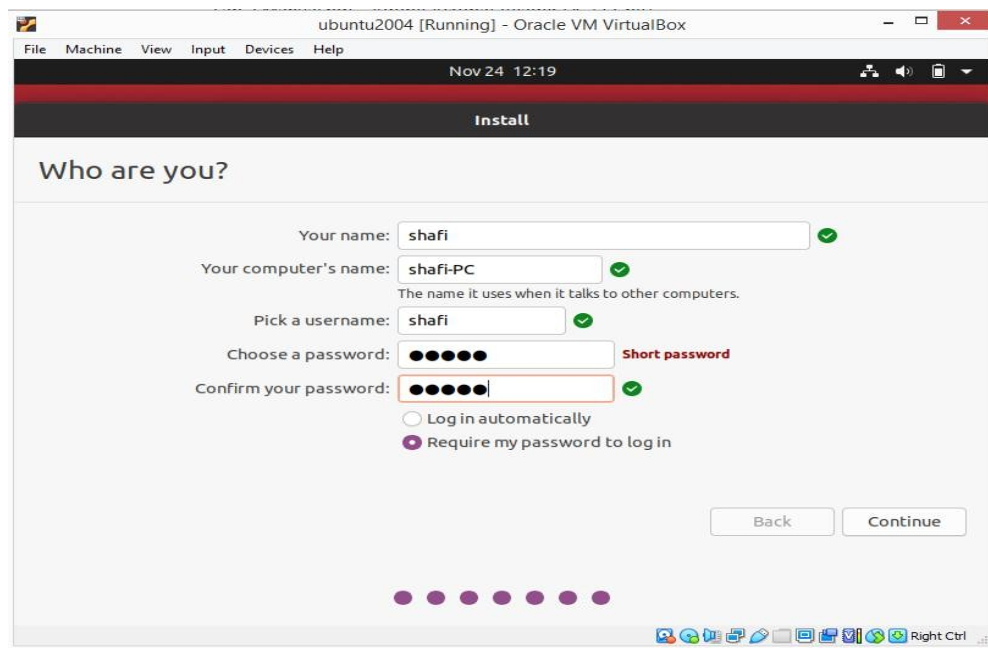


Figure 22: Installing Ubuntu (i)

Now the installation will start (Figure 23)

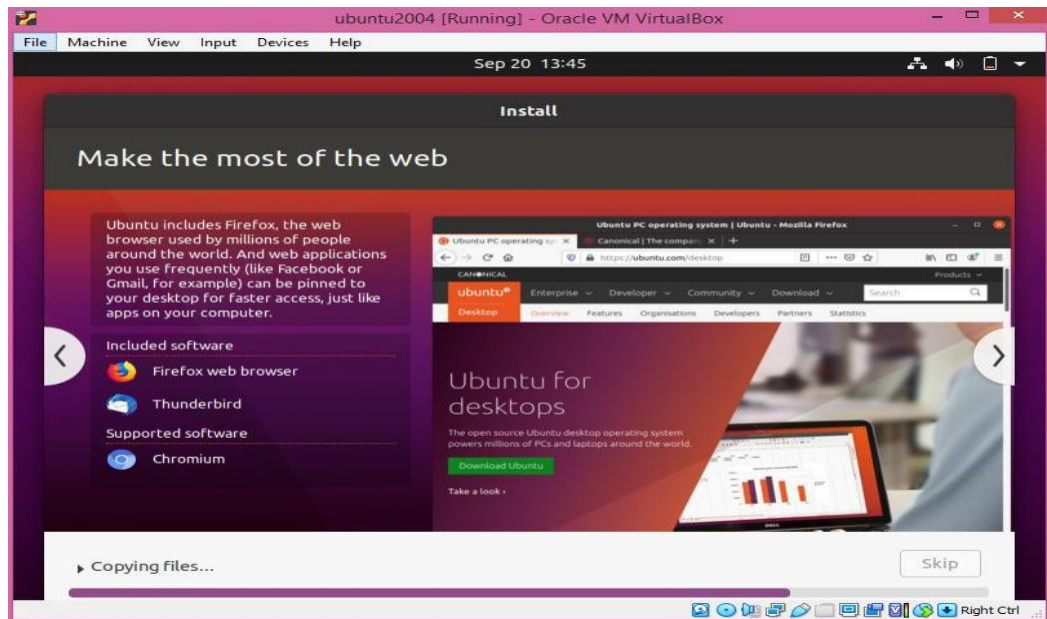


Figure 23: Installing Ubuntu (j)

After the installation is completed, a message will be displayed regarding the completion of installation and it will ask for the restarting. Press “Restart Now” (Figure 24)

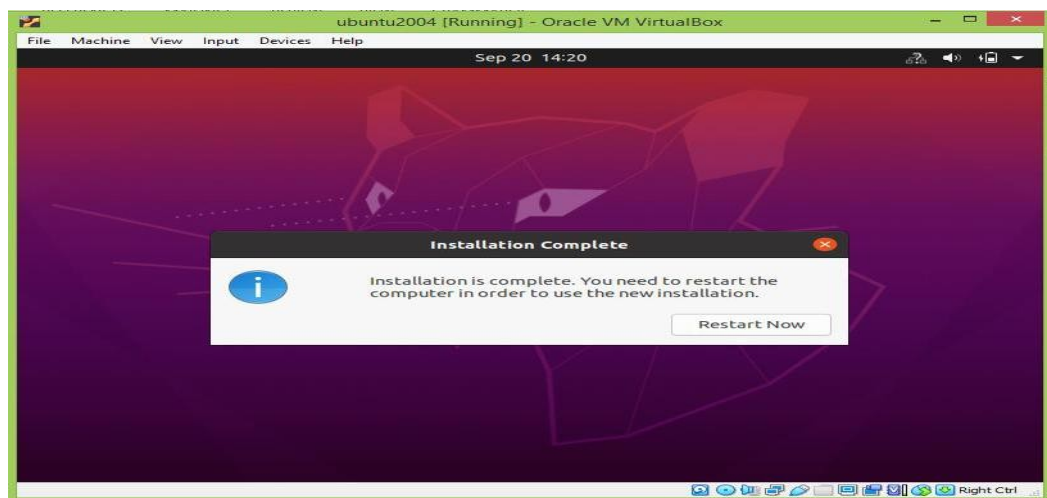


Figure 24: Installing Ubuntu (k)

Now the Ubuntu is installed.

6. Submission

Task 1: Install Ubuntu during lab and inform the instructor.

Task 2:

- i. Try to explore Ubuntu.
- ii. Create a directory/folder on desktop named “*Your_RollNo*”
- iii. Create a file named “**Your name**” inside the “*Your_RollNo*” folder and write about your self. i.e. your name, city, school/college, hobbies etc.
- iv. Make a compress/zip file of the directory created on desktop.
- v. Upload the compress/zip file on LMS inside the “**Lab 2 Submission**” submission folder of Lab 2.

7. Assignment

Assignment for this Lab will be uploaded on LMS and will be explained in Lab. Deadline will also be mentioned on LMS.

8. Evaluation Criteria

The evaluation criteria for this lab will be based on the completion of the following tasks. Each task is assigned the marks which will be evaluated by the instructor, whether the student has finished the complete/partial task(s).

Sr. No.	Heading No	Description	Marks
1	6	Submission Tasks	5
2	7	Assignment	5

Table 2: Evaluation of the Lab



SHIFA TAMEER-E-MILLAT UNIVERSITY

Lab Manual Introduction to Computer Science

DEPARTMENT OF COMPUTING

Shifa Tameer-e-Millat University

Department of Computing

INTRODUCTION TO COMPUTER SCIENCE

Lab 3

Basic Linux Commands

Lab 3: Basic Linux Commands

1. Introduction

Linux: The Linux operating system (or Linux OS) is an open source, freely available OS that can be installed on a wide range of devices such as desktops, servers, smartphones and tablets.

Linux Shell is a program that receives commands from the user and gives it to the Operating System to process and it shows the output. Linux has a CLI (command-line interface) known as **Terminal**. In this lab, we will cover the basic commands that we use in the terminal. To open a terminal in Ubuntu, press “**Ctrl + Alt +T**”.

2. Activity Timeboxing

Heading No	Activity Name	Activity Time	Total Time
4,5	Concept Map/Walkthrough Task	40 minutes	40 minutes
6	Practice Task in Lab	60 minutes	100 minutes
8	Unseen Task	40 minutes	140 minutes
	Assignment Discussion	10	150 minutes

Table 1: Activity Timeboxing

3. Objective

- To understand the Linux commands used by the Linux community
- To learn the basic Linux Commands to perform different tasks through the terminal.

4. Concept Map

In this lab, we learn basic Linux commands that are frequently used while using the Linux operating system. It is important to know Linux commands as it makes it easy to complete tasks. There are a lot of Linux Commands but we will see the more frequent and important commands.

4.1. Basic Linux Commands

1. pwd

To know about your present working directory.

Syntax: \$ pwd

When you first open the terminal, you are in the home directory of your user. To know which directory you are in, you can use the “pwd” command. It gives us the absolute path, which means the path that starts from the root. The root is the base of the Linux file system. It is denoted by a forward slash (/). The user directory is usually something like “/home/username”.

2. ls

To view the list of files/directories in your current directory.

Syntax: \$ ls

It is important to note that:

- Directories are denoted in blue color.
- Files are denoted in white.

i. *ls -a*

To view the list of hidden files/directories in your current directory.

Hidden items in Linux begin with the '.' symbol at the start of the file or directory. Any Directory/file starting with a '.' will not be seen unless you request for it.

ii. *ls -al*

To view the detailed information about the files/directories.

This command provides information in a column format e.g. owner, size, date and time of file.

iii. *ls -R*

To view all the content of directory not only in the current directories but also in sub-directories.

This command list subdirectories recursively. This command not only displays the files or directories of the current directory but also the files and directories of subdirectories.

iv. *ls -d */*

To view all the directories only

This command list only directories in the current directory.

3. cd

To go/change to another directory.

Syntax: \$ cd new_file_location

For example, if you are in the home directory, and you want to go to the Downloads directory, then you can type: cd Downloads.

It is important to note that:

- This command is case sensitive, and you have to type in the name of the directory exactly as it is.
- If you just type cd and press enter, it takes you to the home directory.
- To go back from a folder to the folder before that, you can type: "**cd ..**". The two dots represent back.
- Remember, if you get lost, **type cd**. It will return to your home-directory

4. your home directory (~)

To go/change to another directory.

Home directories can also be referred to by the tilde ~ character. It can be used to specify paths starting at your home directory.

So typing

```
ls ~/Desktop
```

will list the contents of your Desktop directory, no matter where you currently are in the file system. Similarly you can go back to any other location directly no matter where you are like:

```
cd ~/Downloads
```

will take you to the Downloads Directory.

5. mkdir

To make a new directory.

Syntax: \$ mkdir directory_name

For example, if you want to create a directory named as “**Pakistan**”, then you can type:

```
mkdir Pakistan
```

It is important to note that:

- If you want to create a new directory named “**Namal Institute**”, then you can type:

```
mkdir Namal\ Institute
```
- The above point states that if there is a blank character between words of directory name then you should put a backslash (\) between words. You can also create the directory/Folder name having space using **Quotation Marks** i.e

```
mkdir “Namal Institute” OR mkdir ‘Namal Institute’
```
- You can also create **more than one directory/Folder** at a time e.g.

```
mkdir dir1 dir2 dir3
```
- This command is also used to **create nested directories**. For example, if you want to create three directories dirA, dirB, dirC in such a way dirC is inside dirB and dirB is inside dirA then write the following command:

```
mkdir -p dirA/dirB/dirC
```

The “-p” that we used is called an option or a switch (in this case it means “create the parent directories, too”). Options are used to modify how a command operates, allowing a single command to behave in a variety of different ways.

6. rmdir

To delete an existing directory.

Syntax: \$ rmdir directory_name

For example, if you want to delete a directory named “**Pakistan**”, then you can type:

```
rmdir Pakistan
```

It is important to note that:

- rmdir can only be used to delete an empty directory.
- To delete a directory containing files and directories, you can type: **rm -rf**

Be Careful while using **rm -rf** because deleted files cannot be recovered.

7. touch

To create a new file without any content.

Syntax: \$ touch new_file_name

If you want to create a text file named newfile.txt, then you can type:

touch newfile.txt

8. rm

To delete a file.

Syntax: \$ rm file_name

If you want to delete a file named **newfile.txt**, then you can type:

rm newfile.txt

9. mv

To move a file/directory | to rename a file/directory.

This command is used to move a file or directory to another location. This command is also used to rename the file or directory name.

i. Move File/Directory

Syntax: \$ mv filename new_file_location

This command is used to move the file from one place to another location.

Let's move a file. Now create a file named "myfile" on desktop try the following commands

cd ~/Desktop

touch myfile ls

Now move the myfile.txt file to Downloads, you can type:

mv myfile.txt /home/Downloads

ls

ii. Move Directory

Syntax: \$ mv dir_Name new_location

This command is used to move the directory from one place to another location. For example,

to move the directory first we create a directory on Desktop by following commands:

cd ~/Desktop

mkdir newDir ls

Now move the newDir directory to Downloads by the following commands

mv newDir /home/Downloads

ls

iii. *Rename file*

Syntax: \$ mv filename new_file_name

This command is used to rename the file. Lets create and rename the file

Create a file named named “personalfile.txt”

touch personalfile.txt

ls

Now rename this file

mv personalfile.txt personal.txt ls

iv. *Rename Directory*

Syntax: \$ mv dir_name new_dir_name

This command is used to rename the directories.

For example, create a directory named

ICS_Lab. mkdir ICS_Lab

To rename a directory from “ICS_Lab” to “ICS”, use the command: mv ICS_Lab ICS

10. cp

To copy a file/directory to a new location.

Syntax: \$ cp filename new_file_location

Syntax: \$ cp -R dir_name new_file_location

The cp command is used to copy files and/or directories from one place to another.

📁 To copy a file “personal.txt” to Music, use the command: cp personal /home/Music

📁 To copy a directory “ICS” to Music, use the command: cp -R ICS /home/Music

11. Paste any command

To paste any command or text in the terminal.

Syntax: \$ Ctrl + Shift + V

Sometimes it becomes annoying to write long commands in the terminal. To paste the copied command in the terminal press Ctrl + Shift + V.

12. clear

To clear the terminal screen.

Syntax: \$ clear

This clear command is used to clear the terminal screen.

13. man

To display the user manual of any command.

Syntax: \$ man command_name

This command is used to display the user manual of any command that we can run on the terminal. It provides a detailed view of the command. For example, the “man ls” command will give a detailed view of the “ls” command.

14. history

To view the previously executed commands.

Syntax: \$ history

This command is used to view the previously executed commands in the terminal. It will display all the commands. You can view the limited number of commands by giving some specified number. For example to show the last 10 recently used commands you can type: history 10.

Commands can be executed using event numbers also. The event number is the number that is displayed on the left side of the command when we run the history command.

For example, if we want to run the command that has 10 event number then we will type: “!10”. We can also print any command by its event number i.e. “!10:p”

5. Walkthrough Tasks

5.1. Writing commands in Terminal

Login to the Ubuntu operating system you will see the following screen as shown in **Figure 1**. Click the “**show applications**” icon that is highlighted by a yellow rectangle in **Figure 1**.

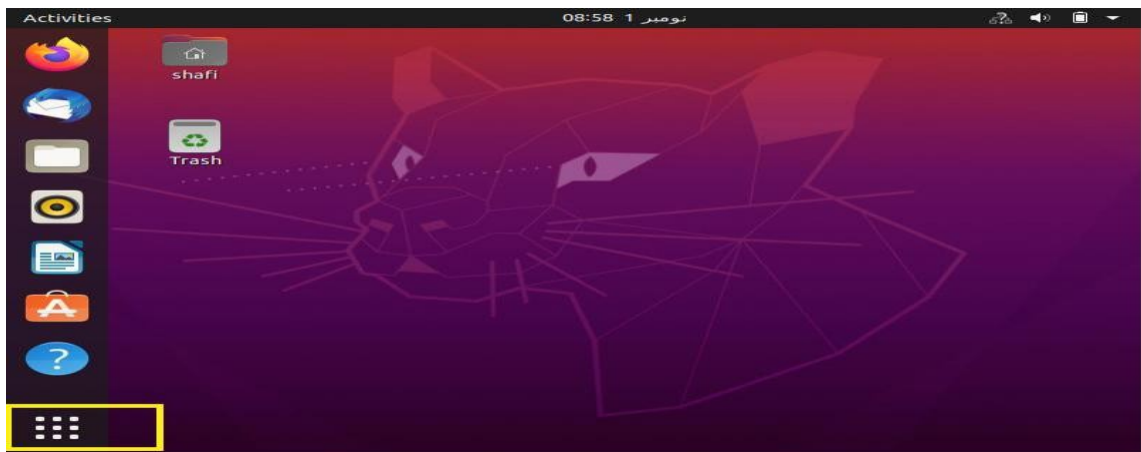


Figure 1

When you will click on the “**Show Applications**” terminal you will see the following screen as shown in **Figure 2** then search “**Terminal**” from the search field as highlighted by a rectangle in Figure 2. Click on the **Terminal** icon then terminal will be opened on screen as shown in **Figure 3**.

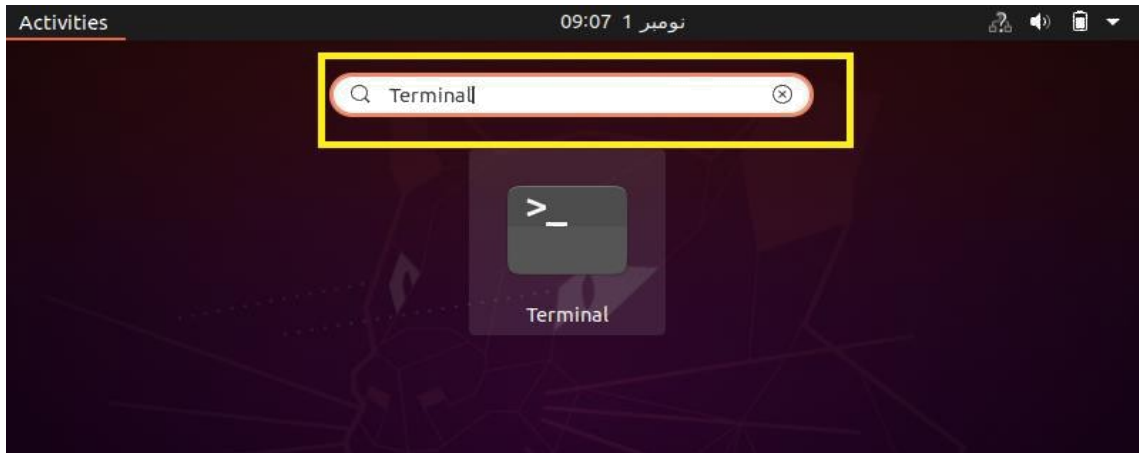


Figure 2

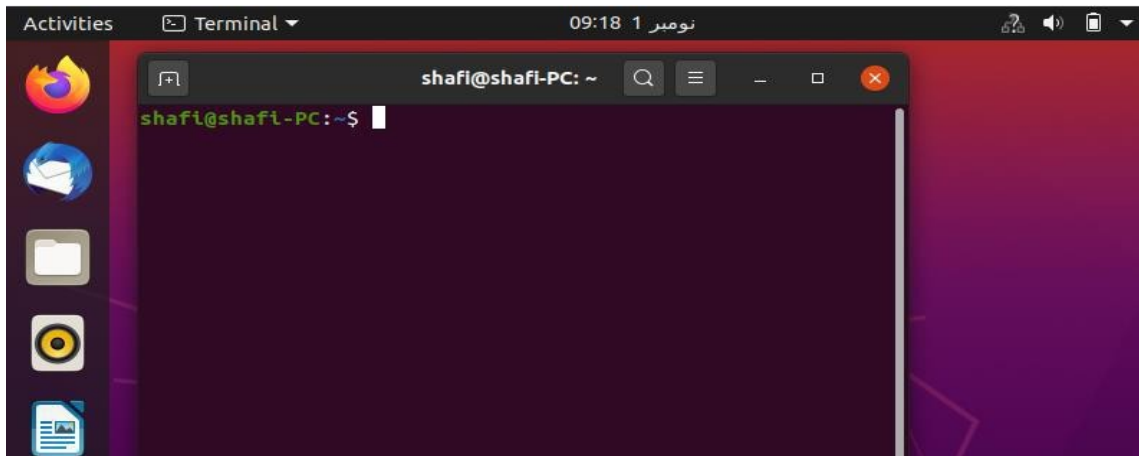


Figure 3

When Terminal is opened write a Linux command “ls” as shown in **Figure 4**.

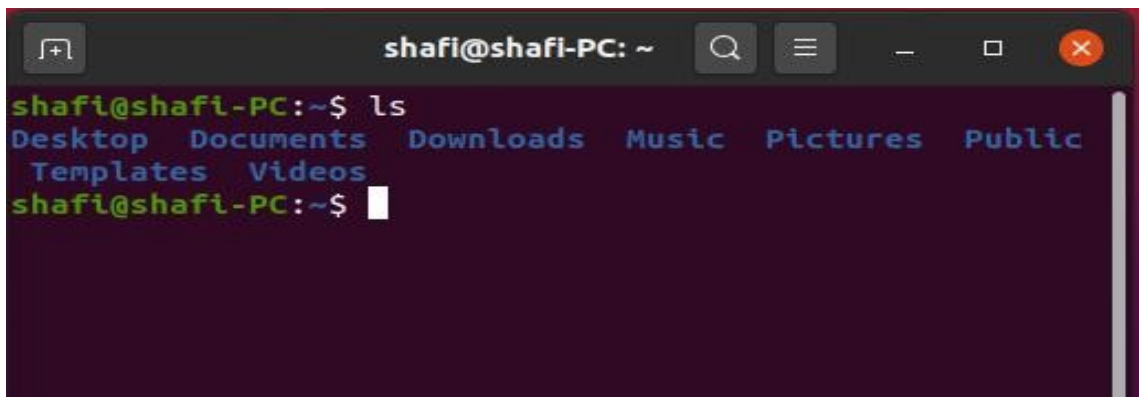


Figure 4

When we write a command “ls” in Terminal, we can see all the content (directories and files) of the current directory. Here in our example, we were in the “Home” directory and this command list all the content of this directory.

Similarly, we can use other commands in the terminal and get their results. In this lab manual, some commands have already been discussed in the concept map heading, Write them in Terminal and try to perform Practice tasks given in the next heading.

6. Practice Tasks

In this part, you are provided some practice tasks that will help you to understand these commands. You have to perform all the given tasks below and when you write any command to perform a task then copy that command or write that command in the “YourRollNo_YourName.txt” file.

You have to write commands in the following syntax in the “YourRollNo_YourName.txt” file. For example, we want to write the commands for Task 1 and Task 2 then write:

Task 1:

```
cd
```

```
cd Desktop
```

```
mkdir Class2025
```

Task 2:

```
cd
```

```
cd Desktop
```

```
cd Class2025
```

```
mkdir ICS_Lab
```

And same for Task 3, Task 4, etc.

Note: It is necessary to write each command in this file because you have to submit it at the end of this lab.

Note: Before starting any Task write the “**cd**” command because it will take you to the “home” directory every time. Then start writing commands according to the Task. You can see from the given below Tasks that every Task is started from the “home” directory.

6.1. Task 1

Go to Desktop and make a directory named “Class2025”

```
cd
```

```
cd Desktop
```

```
mkdir Class2025
```

6.2. Task2

Make another directory named “ICS_Lab” inside the “Class2025” directory.

```
cd  
cd Desktop  
cd Class2025  
mkdir ICS_Lab
```

6.3. Task3

Make a file inside the “ICS_Lab” directory named “file1.txt”. Add some text to it. Like “This is file1”

```
cd  
cd Desktop/Class2025/ICS_Lab  
touch file1.txt
```

6.4. Task4

Make another file inside the “ICS_Lab” directory named “file2.txt” and add some text to it. Like “This is file2”

```
cd  
cd Desktop/Class2025/ICS_Lab  
touch file2.txt
```

6.5. Task5

Now make a copy of “file1.txt” to another file named “file3.txt” (file3.txt should also be created in the same directory)

```
cd  
cd Desktop/Class2025/ICS_Lab  
cp file1.txt file3.txt
```

6.6. Task6

Now make another directory named “ICS_Lab_2” inside the “Class2025” directory

```
cd  
cd Desktop/Class2025  
mkdir ICS_Lab_2
```

6.7. Task7

Now copy “file3.txt” present in “ICS_Lab” to the “ICS_Lab_2” directory.

Here are multiple ways to do this task. Understand all methods and choose any of these. You can ask instructor if you have any confusion.

```
cp ~/Desktop/Class2025/ICS_Lab/file3.txt ~/Desktop/Class2025/ICS_Lab_2
```


OR

```
cd  
cd Desktop/Class2025/ICS_Lab  
cp file3.txt ~/Desktop/Class2025/ICS_Lab_2
```

OR

```
cd  
cd Desktop/Class2025/ICS_Lab  
cp file3.txt ../ICS_Lab_2
```

6.8. Task8

Now rename the file named “file3.txt” located in the “ICS_Lab_2” directory to “renamed_file.txt”.

```
mv ~/Desktop/Class2025/ICS_Lab_2/file3.txt ~/Desktop/Class2025/ICS_Lab_2/renamed_file.txt
```

OR

```
cd  
cd Desktop/Class2025/ICS_Lab_2  
mv file3.txt renamed_file.txt
```

6.9. Task9

Now go to the Desktop and list/display all the files and directories.

```
cd  
cd Desktop  
ls  
OR  
ls ~/Desktop
```

6.10. Task10

Now make a copy of ICS_Lab directory named “**ICS_Lab_3**”.

```
cp -R ~/Desktop/Class2025/ICS_Lab ~/Desktop/Class2025/ICS_Lab_3  
move this ICS_Lab_3 directory to the Music  
mv ~/Desktop/Class2025/ICS_Lab_3 ~/Music
```

6.11. Task11

Now print the history of your commands.

```
history
```

6.12. Task12

Now create directories named as test1, test2 and test3 on Desktop using a single command. (Don't use command three times to create directories one by one).

```
cd  
cd Desktop  
mkdir test1 test2 test3
```

6.13. Task13

Now delete the "file1.txt" from the directory named "ICS_Lab"

```
cd  
cd Desktop/Class2025/ICS_Lab  
rm file1.txt  
OR  
rm ~/Desktop/Class2025/ICS_Lab/file1.txt
```

6.14. Task14

Now delete the directory named "test3"

```
cd  
cd Desktop  
rmdir test3  
rmdir ~/Desktop/test3
```

6.15. Task15

Now clear the terminal.

```
clear
```

6.16. Task16

Now move to the "Class2025" directory and show detailed information about the files/directories and also sub-directories.

```
cd  
cd Desktop/Class2025  
ls -Rl  
OR  
ls ~/Desktop/Class2025
```

7. Submission [Practice Tasks]

Make a file named as your “YourRollNo_YourName” and Submit it on LMS in Lab 3 Practice Task Submission folder.

8. Unseen Task

You will be given an unseen task you have to perform and submit it during the lab.

9. Assignment

Assignment for this Lab will be uploaded on LMS after the Lab session. The deadline will also be mentioned on LMS.

10. Quiz

A quiz about this lab will be conducted at the end of the all linux commands labs.

11. Further Readings

The reading material and other helping material regarding this Lab will be uploaded on LMS if any. You can access it from the LMS.

12. Evaluation Criteria

■ **Method of Evaluation:** VivaFile submitted on LMS.

	Excellent 10	Good 9-7	Satisfactory 6-4	Unsatisfactory 3-1	Poor 0	Marks Obtained
Unseen Tasks	All tasks completed correctly in given time.	Most tasks completed correctly. Tasks could be improved further.	Some tasks completed correctly.	Most tasks incomplete or incorrect	All tasks incomplete or incorrect. Didn't perform tasks	
Assignment	All tasks completed correctly in given time and have complete understanding	Most tasks completed correctly. Tasks could be improved further and have complete understanding	Some tasks completed correctly and have incomplete understanding	Most tasks incomplete or incorrect and have no understanding	All tasks incomplete or incorrect. Didn't perform tasks and have no understanding	
Total						



SHIFA TAMEER-E-MILLAT UNIVERSITY

Lab Manual **Introduction to Computer Science**

DEPARTMENT OF COMPUTING

Shifa Tameer-e-Millat University

Department of Computing

INTRODUCTION TO COMPUTER SCIENCE

Lab 4

Basic Linux Commands

Lab 4: Basic Linux Commands

1. Introduction

Linux: The Linux operating system (or Linux OS) is an open-source, freely available OS that can be installed on a wide range of devices such as desktops, servers, smartphones and tablets.

Linux Shell is a program that receives commands from the user and gives it to the Operating System to process and it shows the output. Linux has a CLI (command-line interface) known as **Terminal**. In this lab, we will cover the basic commands that we use in the terminal. To open a terminal in Ubuntu, press “**Ctrl + Alt +T**”.

2. Activity Timeboxing

Heading No	Activity Name	Activity Time	Total Time
	Unseen Task from Previous Lab	30 minutes	30 minutes
5	Walkthrough Task	45 minutes	75 minutes
6	Practice Task in Lab	45 minutes	120 minutes
7	Unseen Task Lab 4	30 minutes	150 minutes

Table 1: Activity Timeboxing

3. Objective

- To understand the Linux commands used by the Linux community
- To learn the basic Linux Commands to perform different tasks through the terminal.

4. Concept Map

In this lab we learn basic Linux commands that are frequently used while using Linux operating system. It is important to know Linux commands as it makes it easy to complete tasks. There are a lot of Linux Commands but we will see the more frequent and important commands.

4.1. Basic Linux Commands

1. echo

To display line of text/string.

Syntax: \$ echo [optional options] [string]

echo command in Linux is used to display line of text/string that are passed as an argument. This is a built in command that is mostly used in shell scripts and batch files to output status text to the screen or a file.

The echo command in Linux is used to display a string provided by the user.

For example, use the following command to print Hello, World! as the output:

```
echo Hello, World
```

Echo Command Options

The echo command uses the following options:

- **-n**: Displays the output while omitting the newline after it.
- **-E**: The default option, disables the interpretation of escape characters.
- **-e**: Enables the interpretation of the following escape characters:
 - o ****: Displays a backslash character (\).
 - o **\c**: Omits any output following the escape character.
 - o **\t**: Creates horizontal tab spaces.
 - o **\v**: Creates vertical tab spaces.

Run the following commands and see the results of the echo command

Note: *If you are using the -e option, enter your string enclosed in single quotation marks. This ensures that any escape characters are interpreted correctly.*

```
echo Hello, World!
```

Changing the Output Format

Using the -e option allows you to use escape characters. These special characters make it easy to customize the output of the echo command.

For instance, using \c let you shorten the output by omitting the part of the string that follows the escape character:

```
echo -e 'Hello, World! \c This is a Book'
```

Use \n any time you want to move the output to a new line:

```
echo -e 'Hello, \nWorld, \nthis \nis \nBook!'
```

Add horizontal tab spaces by using \t:

```
echo -e 'Hello, \tWorld!'
```

Use \v to create vertical tab spaces:

```
echo -e 'Hello, \vWorld, \vthis \vis \va \vBook!'
```

2. cat

[To display/view the content of a file.](#)

Syntax: \$ cat filename

The cat command is used to display the content of the text files.

For example, create a file named “file1.txt” on desktop using the touch command and add some text like this is file 1.

Write the following commands to create a file on the desktop.

```
touch ~/Desktop/file1.txt
```


Now you have created a file with some text. Try the following command to display the content of the file1.txt file.

```
cat file1.txt
```

The “cat” command can also be used for copying, combining and creating new text files. Let's see how it works.

i. Creating a new file with content:

Syntax: `$ cat > filename.txt`

This command is used to create a new text file. To create a new text file named file2.txt, you can type:

```
cat > file2.txt
```

The cursor will blink on the terminal to enter the text in the newly created file (Type “this is file 2” in this file). After entering the text press “ctrl + d” to exit the editing mode.

Now you can see the content of this file by following the command

```
cat file2.txt
```

Again create the new file named file3.txt by the following command and enter the text “this is file3”.

```
cat > file3.txt
```

ii. Combining files:

Syntax: `$ cat file2.txt file3.txt > file4.txt`

This syntax is used to combine text files. To combine text files named “file2.txt” and “file3.txt” into “file4.txt”, you can type

```
cat file2.txt file3.txt > file4.txt
```

As soon as you insert this command and hit enter, the files are concatenated, but you do not see a result. This is because Linux Shell (Terminal) is a silent type. It will never give you a confirmation message like "OK" or "Command Successfully Executed". It will only show a message when something goes wrong or when an error has occurred.

To see if the file is created or not enter the following command in the terminal

```
ls
```

If the file4.txt is created and to check if the text of both files is combined or not write the following command.

```
cat file4.txt
```

iii. Copying file:

Syntax: `$ cat file1.txt > file3.txt`

This command is used to copy the text inside one text file to another. To copy a text file named “file1.txt” into “file5.txt”, you can type:

```
cat file1.txt > file5.txt
```

iv. Append file:

Syntax: `$ cat >> file1.txt`

This command is used to append a text into the already created file. If the file is not already created it will create a new file.

To append some text “this is appended text” into file1.txt. write the following command.

```
cat >> file1.txt
```

and enter “this is appended text” in the terminal to enter text in file1.

Note: Only text files can be displayed, created, combined and copied using the cat command.

3. date

To display current time and date.

Syntax: \$ date

If you want to get the current time and date type ‘date’ in the terminal, it will print time with the date but if you are only interested to see only Time the type “date +%T” in the terminal, it will print only time but if you are interested to see only Date the type “date +%D” in terminal.

Different commands for the date are available on the following link. Try those commands to get your desired results.

<https://phoenixnap.com/kb/linux-date-command>

4. cal

To display the calendar of the current month.

Syntax: \$ cal

Cal command is used to display the calendar of the current month. You can also get the calendar of specified month and year. The syntax for this command will be:

Syntax: \$ cal month year

For example, if you want to print the calendar for July 2012 the type: “cal 7 2012”.

5. whoami

To display the currently logged-in

user. **Syntax:** \$ whoami

This command displays the user who is currently logged in. To check the currently logged-in user type “whoami” in the terminal.

6. whatis

To give one-line description of the command.

Syntax: \$ whatis [command]

This command display a one-line description of the command. It can be used as a quick reference for any command.

For example, see the following commands and try for other commands if you are interested to get a one-line description of any command

```
shafi@shafi10-PC:~/Desktop$ whatis mkdir
mkdir (1)          - make directories
shafi@shafi10-PC:~/Desktop$ whatis ls
ls (1)            - list directory contents
shafi@shafi10-PC:~/Desktop$ whatis cp
cp (1)            - copy files and directories
shafi@shafi10-PC:~/Desktop$ whatis cat
cat (1)           - concatenate files and print on the standard output
shafi@shafi10-PC:~/Desktop$
```

7. file

To determine the file type of a given file.

Syntax: \$ file [filename]

This command is used to determine the type of file. You can provide one or more than one file as an argument to the file command. To determine the type of files of multiple files the syntax is:

“file file1Name file2Name”

For example, if I have 2 files and a directory on my desktop and I want to know their file type then I will write the following commands.

```
shafi@shafi10-PC:~/Desktop$ ls
Class2025 file10.txt file1.txt
shafi@shafi10-PC:~/Desktop$ file file1.txt file10.txt Class2025/
file1.txt: ASCII text
file10.txt: ASCII text
Class2025/: directory
shafi@shafi10-PC:~/Desktop$
```

8. head

To display the first few lines of a text

Syntax: \$ head file_name

This command is used to display the first few lines of a file. For example, if you want to show the first few lines of the file named as “file1.txt” type: head file1.txt.

By default, it will display the first 10 lines of the file. But if you are interested to show a specified number of lines then use the -n option with this command. For example, now you are interested to display only the first 4 lines of “file1.txt” type: head -4 file1.txt.

Currently, you have not much data in file1.txt. To check the results of the head command, add at least 20 lines such as:

This is line

1 This is

line 2

This is line 3

This is line 4

.

.

.

This is line 18

This is line 19

This is line 20

9. tail

To display the last few lines of a text

file. **Syntax:** \$ tail file_name

This command is used to display the last few lines of a file. For example, if you want to show the last few lines of the file named as “file1.txt” type: tail file1.txt. By default, it will display the last 10 lines of the file. But if you are interested to show a specified number of lines then use the -n option with this command. For example, now you are interested to display only the last 4 lines of the “file1.txt” type:

```
tail -4 file1.txt.
```

10. alias

Shortcut to commands.

Syntax: \$ alias short_name_of_command = Actual command

The ‘alias’ is another name for a command. Aliases can be used for short names of commands. For example, you might use the clear command frequently. You can create an alias for it like
alias cls =clear

Next time you enter 'cls ' on the terminal, your screen will get clear.

If no argument is given, it shows current aliases. Current aliases can be checked with the following command:

```
alias
```

5. Walkthrough Tasks

5.1. Writing commands in Terminal

Login to the Ubuntu operating system you will see the following screen as shown in **Figure 1**. Click the “**show applications**” icon that is highlighted by a yellow rectangle in **Figure 1**.

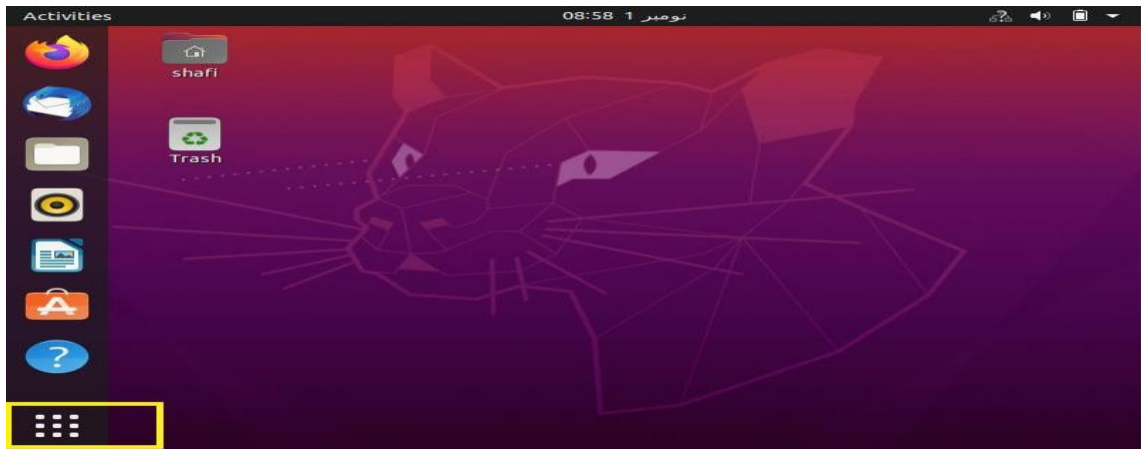


Figure 1

When you will click on the “**Show Applications**” terminal you will see the following screen as shown in **Figure 2** then search “**Terminal**” from the search field as highlighted by a rectangle in Figure 2. Click on the **Terminal** icon then the terminal will be opened on screen as shown in **Figure 3**.

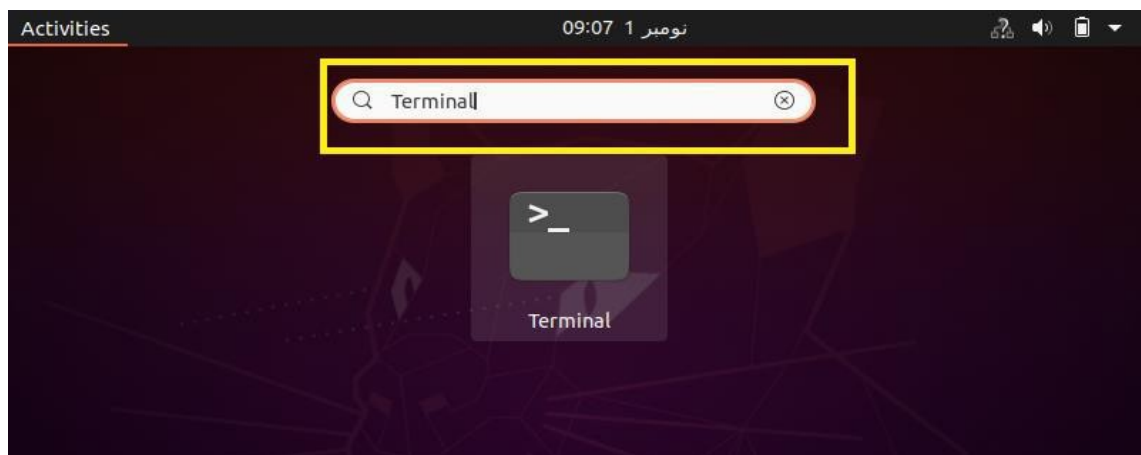


Figure 2

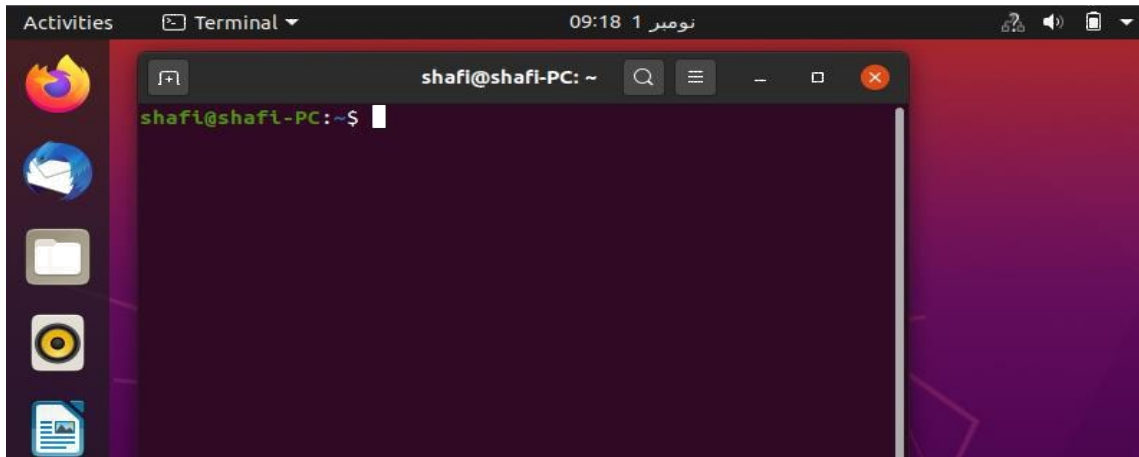


Figure 3

When Terminal is opened write a Linux command “ls” as shown in **Figure 4**.

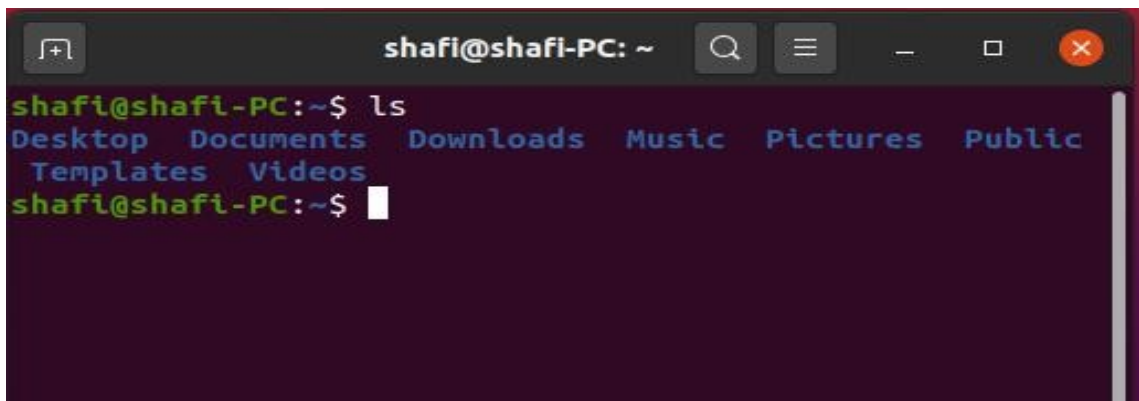


Figure 4

When we write a command “ls” in Terminal, we can see all the content (directories and files) of the current directory. Here in our example, we were in the “Home” directory and this command list all the content of this directory.

Similarly, we can use other commands in the terminal and get their results. In this lab manual, some commands have already been discussed in the concept map heading, Write them in Terminal and try to perform Practice tasks given in the next heading.

6. Practice Tasks in Lab

Practice tasks for all commands given in this lab are given with the description of the commands. Try all given examples to know the working of all commands.

7. Unseen Task

You will be given unseen tasks you have to perform and submit during the lab.

8. Assignment

Assignment for this Lab will be uploaded on LMS after the Lab session. The deadline will also be mentioned on LMS.

9. Quiz

A quiz about this lab will be conducted at the end of all Linux commands labs.

10. Further Readings

The reading material and other helping material regarding this Lab will be uploaded on LMS if any. You can access it from the LMS.

11. Evaluation Criteria

■ **Method of Evaluation:** Viva, File submitted on LMS.

	Excellent 10	Good 9-7	Satisfactory 6-4	Unsatisfactory 3-1	Poor 0	Marks Obtained
Unseen Tasks	All tasks completed correctly in given time.	Most tasks completed correctly. Tasks could be improved further.	Some tasks completed correctly.	Most tasks incomplete or incorrect	All tasks incomplete or incorrect. Didn't perform tasks	
Assignment	All tasks completed correctly in given time and have complete understanding	Most tasks completed correctly. Tasks could be improved further and have complete understanding	Some tasks completed correctly and have incomplete understanding	Most tasks incomplete or incorrect and have no understanding	All tasks incomplete or incorrect. Didn't perform tasks and have no understanding	
Total						



SHIFA TAMEER-E-MILLAT UNIVERSITY

Lab Manual **Introduction to Computer Science**

DEPARTMENT OF COMPUTING

INTRODUCTION TO COMPUTER SCIENCE

Lab 5

Basic Linux Commands

Lab 5: Basic Linux Commands

1. Introduction

Linux: The Linux operating system (or Linux OS) is an open-source, freely available OS that can be installed on a wide range of devices such as desktops, servers, smartphones and tablets.

Linux Shell is a program that receives commands from the user and gives it to the Operating System to process and it shows the output. Linux has a CLI (command-line interface) known as **Terminal**. In this lab, we will cover the basic commands that we use in the terminal. To open a terminal in Ubuntu, press “**Ctrl + Alt +T**”.

2. Activity Timeboxing

Heading No	Activity Name	Activity Time	Total Time
	Unseen task from Previous Lab	40 minutes	40 minutes
5	Walkthrough Task	40 minutes	80 minutes
6	Practice Task in Lab	70 minutes	150 minutes

Table 1: Activity Timeboxing

3. Objective

- To understand the Linux commands used by the Linux community
- To learn the basic Linux Commands to perform different tasks through the terminal.

4. Concept Map

In this lab, we learn basic Linux commands that are frequently used while using Linux operating system. It is important to know Linux commands as it makes it easy to complete tasks. There are a lot of Linux Commands but we will see the more frequent and important commands.

4.1. Basic Linux Commands

1. zip

To compress the files | to reduce file size | to put one or more files into a single zip archive.

Syntax [files]: \$ zip zipfilename.zip filename

(if there are multiple files use space between them)

Syntax [directories]: \$ zip -r zipfilename.zip filename/directoryName (If there are multiple files/directories use space between them)

Zip is the most widely used archive file format that supports lossless data compression. A Zip file is a data container containing one or more compressed files or directories. Compressed (zipped) files take up less disk space can be transferred from one to another machine more quickly than uncompressed files. Zip files can be easily extracted in Windows, macOS, and Linux using the utilities available for all operating systems.

To unzip a zip file using unzip command for example to unzip a file named “myzipfile.zip” use the following command.
unzip myzipfile.zip

Practice Task 1

Download a file named “dummy_text.txt” from the LMS in the Lab 5 Section. Or If the file is not downloaded but opened in the browser then copy all the text from the file and paste it into a file “dummy_text.txt” (create dummy_text.txt file before pasting the copied text).

Move this file to the desktop.

Then make a zip file of “**dummy_text.txt**” file named “**myzipfile.zip**” Now create another file name “**file2.txt**” and add some text to it.

Now make a copy of the “**dummy_text.txt**” file using the **cp** command and name it as “**copied_file.txt**”

Now create a zip file of “dummy_text.txt”, “copied_file.txt” and “file2.txt” named as “**multiZip.zip**”

Practice Task 2

Now create a **directory/folder** on the desktop named “**myfolder**”

Copy **dummy_text.txt** file using cat or **cp** command inside the **myfolder**

Now make a zip file of “myfolder” named “**myfolder.zip**”

2. less

[To view text files.](#)

Syntax: \$ less fileName

With the less command, one does not have to use an editor to view files. It allows users to view files without fearing them being modified. The process is faster and best for those who believe their files might mistakenly be edited.

Once the file opens, users can view the entire document using the Up and Down key on the keyboard. Once viewed, users can quit less by pressing the ‘Q’

Practice Task 3

Open the “**dummy_text.txt**” file using **less** command

3. sudo

To access restricted files and operations.

Syntax: \$ sudo [command]

“sudo” stands for SuperUser **DO** and is used to access restricted files and operations. By default, Linux restricts access to certain parts of the system preventing sensitive files from being compromised. The sudo command temporarily elevates privileges allowing users to complete sensitive tasks without logging in as the root user.

For example:

Open a terminal window, and try the following command:

```
apt-get update
```

You should see an error message. You do not have the necessary permissions to run the command.

Try the same command with sudo:

```
sudo apt-get update
```

Type your password when prompted. The system executes the command and updates the repositories.

To stop update enter “Ctrl + c” in the terminal

4. chmod

To sets the permissions of files or directories.

Syntax: \$ chmod options permissions file_name

This command sets the permissions of files or directories. Permissions define the permissions for the owner of the file (the "user"), members of the group who owns the file (the "group"), and anyone else ("others"). There are two ways to represent these permissions: with symbols (alphanumeric characters), or with octal numbers (the digits 0 through 7).

Let's say you are the owner of a file named myfile, and you want to set its permissions so that:

- i. The user can read, write, and execute it;
- ii. Members of your group can read and execute it; and
- iii. Others may only read it.

This command will look like: `chmod u=rwx,g=rx,o=r myfile`

This example uses symbolic permissions notation. The letters u, g, and o stand for "user", "group", and "other". The equals sign ("=") means "set the permissions exactly like this," and the letters "r", "w", and "x" stand for "read", "write", and "execute", respectively. The commas separate the different classes of permissions, and there are no spaces in between them.

Here is the equivalent command using octal permissions notation:

chmod 754 myfile

Here the digits 7, 5, and 4 each individually represent the permissions for the user, group, and others, in that order. Each digit is a combination of the numbers 4, 2, 1, and 0:

- 4 stands for "read",
- 2 stands for "write",
- 1 stands for "execute", and
- 0 stands for "no permission."

So 7 is the combination of permissions 4+2+1 (read, write, and execute), 5 is 4+0+1 (read, no write, and execute), and 4 is 4+0+0 (read, no write, and no execute).

Note: You can check the permissions using “ls-l command”

Practice Task 4

Now make a file named “testfile.txt” on the Desktop using the cat command and write “this is test file” inside this file.

Now check the permissions of this file.

Show the content of this file using the cat command

Now append the text inside the file “this is appended text”

Again check the permissions of this file.

Now change the permissions of this file as

- **User** can only **write** inside the file, **group** and **others** could only **read** the file.

Now try to display the content of the file using the cat command. (if you correctly changed the permissions then you would not be able to display the content of the file.)

Now Append the file with the text “again appended text”

Now change the permissions of the file as

- **User** can only **read** inside the file. Also, **group** and **others** could only **read** the file.

Now display the content of the file. This time you can see the content provided you changed the permissions correctly.

Now try to append the file (But you will not be able to append the text and you will get a message “permission denied” if you changed the permissions correctly)

Now change the permissions for the user to both **read** and **write** the content. And try to append some new text and display it on the terminal.

5. Walkthrough Tasks

5.1. Writing commands in Terminal

Login to the Ubuntu operating system you will see the following screen as shown in **Figure 1**. Click the “**show applications**” icon that is highlighted by a yellow rectangle in **Figure 1**.

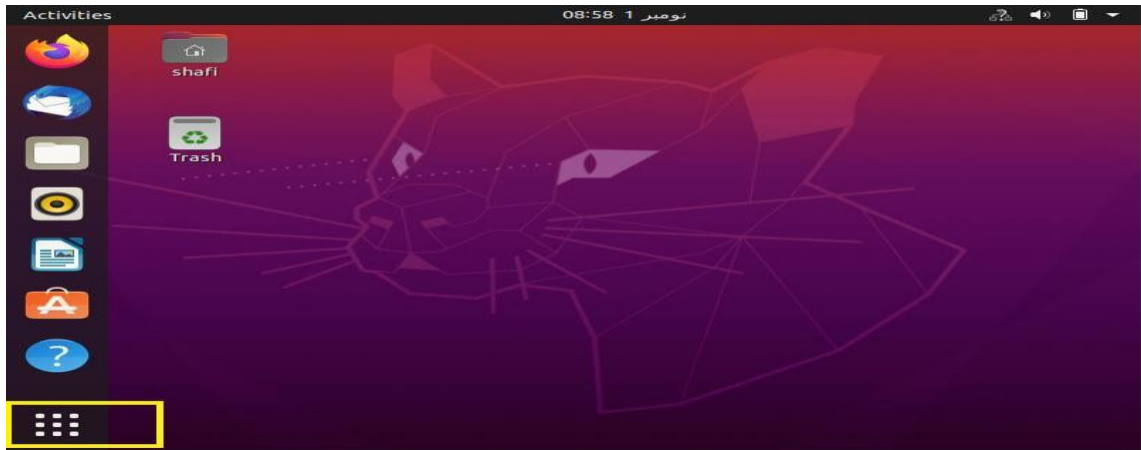


Figure 1

When you will click on the “**Show Applications**” terminal you will see the following screen as shown in **Figure 2** then search “**Terminal**” from the search field as highlighted by a rectangle in Figure 2. Click on the **Terminal** icon then the terminal will be opened on screen as shown in **Figure 3**.

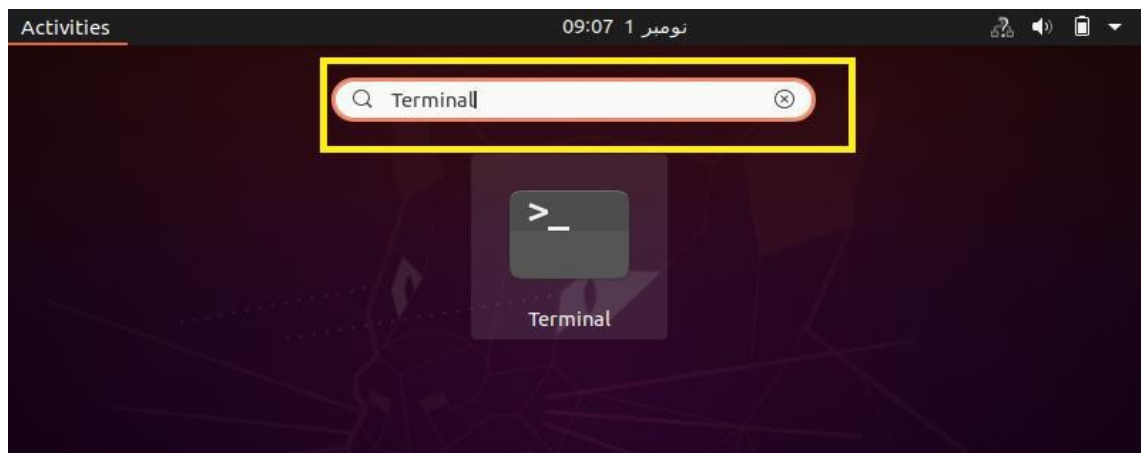


Figure 2

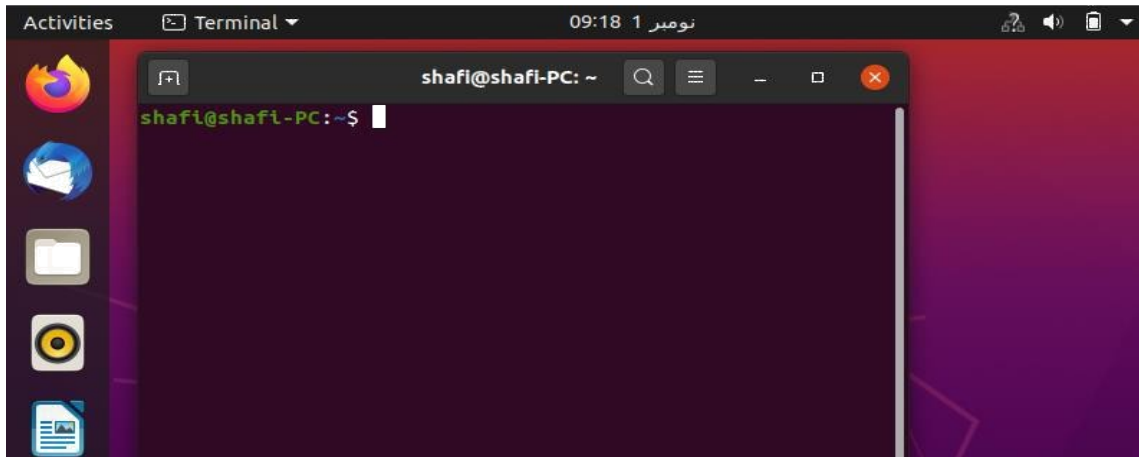


Figure 3

When Terminal is opened write a Linux command “ls” as shown in **Figure 4**.

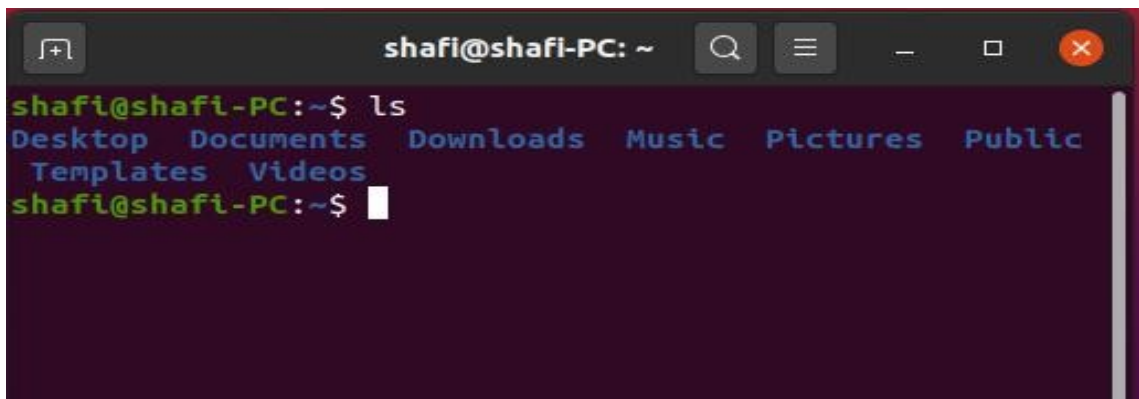


Figure 4

When we write a command “ls” in Terminal, we can see all the content (directories and files) of the current directory. Here in our example, we were in the “Home” directory and this command list all the content of this directory.

Similarly, we can use other commands in the terminal and get their results. In this lab manual, some commands have already been discussed in the concept map heading, Write them in Terminal and try to perform Practice tasks given in the next heading

6. Practice Tasks in Lab

Practice tasks for all commands given in this lab are given with the description of the commands. Try all given examples to know the working of all commands.

7. Unseen Task

You will be given unseen tasks you have to perform and submit during the lab.

8. Assignment

Assignment for this Lab will be uploaded on LMS after the Lab session. The deadline will also be mentioned on LMS.

9. Quiz

A quiz about this lab will be conducted at the end of all Linux commands labs.

Note: This was the last lab for the Linux commands. You will have the quiz for Lab 3, Lab 4, Lab 5 in the start of the next Lab.

10. Further Readings

The reading material and other helping material regarding this Lab will be uploaded on LMS if any. You can access it from the LMS.

11. Evaluation Criteria

📁 **Method of Evaluation:** Viva, File submitted on LMS.

	Excellent 10	Good 9-7	Satisfactory 6-4	Unsatisfactory 3-1	Poor 0	Marks Obtained
Unseen Tasks	All tasks were completed correctly in the given time.	Most tasks were completed correctly. Tasks could be improved further.	Some tasks were completed correctly.	Most tasks were incomplete or incorrect	All tasks were incomplete or incorrect. Didn't perform tasks	
Assignment	All tasks completed correctly in given time and have a complete understanding	Most tasks were completed correctly. Tasks could be improved further and have a complete understanding	Some tasks were completed correctly and have an incomplete understanding	Most tasks were incomplete or incorrect and have no understanding	All tasks were incomplete or incorrect. Didn't perform tasks and have no understanding	
Total						



SHIFA TAMEER-E-MILLAT UNIVERSITY

Lab Manual **Introduction to Computer Science**

DEPARTMENT OF COMPUTING

Shifa Tameer-e-Millat University

Department of Computing

INTRODUCTION TO COMPUTER SCIENCE

Lab 6

LMC Programming

Lab 6: LMC Programming

1. Introduction

A Little Man Computer (LMC) is a simulator which has many of the basic features of a modern computer that uses the Von Neumann architecture. The LMC is based on the idea of a ‘Little Man’ acting as the control unit of a CPU, fetching instructions from RAM, decoding and executing them as well as managing the input and output mechanisms. LMC can be programmed by using a basic set of 10 assembly code instructions which are then assembled into machine code (although in decimal not binary).

2. Activity Timeboxing

Heading No	Activity Name	Activity Time	Total Time
5	Unseen task from Previous Lab	30 minutes	30 minutes
	Quiz from the Previous Labs	20 minutes	50 minutes
6	Walkthrough Task	40 minutes	90 minutes
7	Practice Task in Lab	60 minutes	150 minutes

Table 1: Activity Timeboxing

3. Objective

- To understand the Little Man Computer (LMC).
- To understand the working of a modern computer using Little Man Computer (LMC).
- To write the basic programs in LMC

4. Concept Map

In this lab, we learn about the LMC and its working. We will also relate its working with the working of modern computers. We will also write some basic programs in LMC.

4.1. Little Man Computer (LMC)

A Little Man Computer (LMC) is a simulator that has many of the basic features of modern computers. The LMC is based on the idea of a ‘Little Man’ acting as the control unit of a CPU, fetching instructions from RAM, decoding and executing them as well as managing the input and output mechanisms.

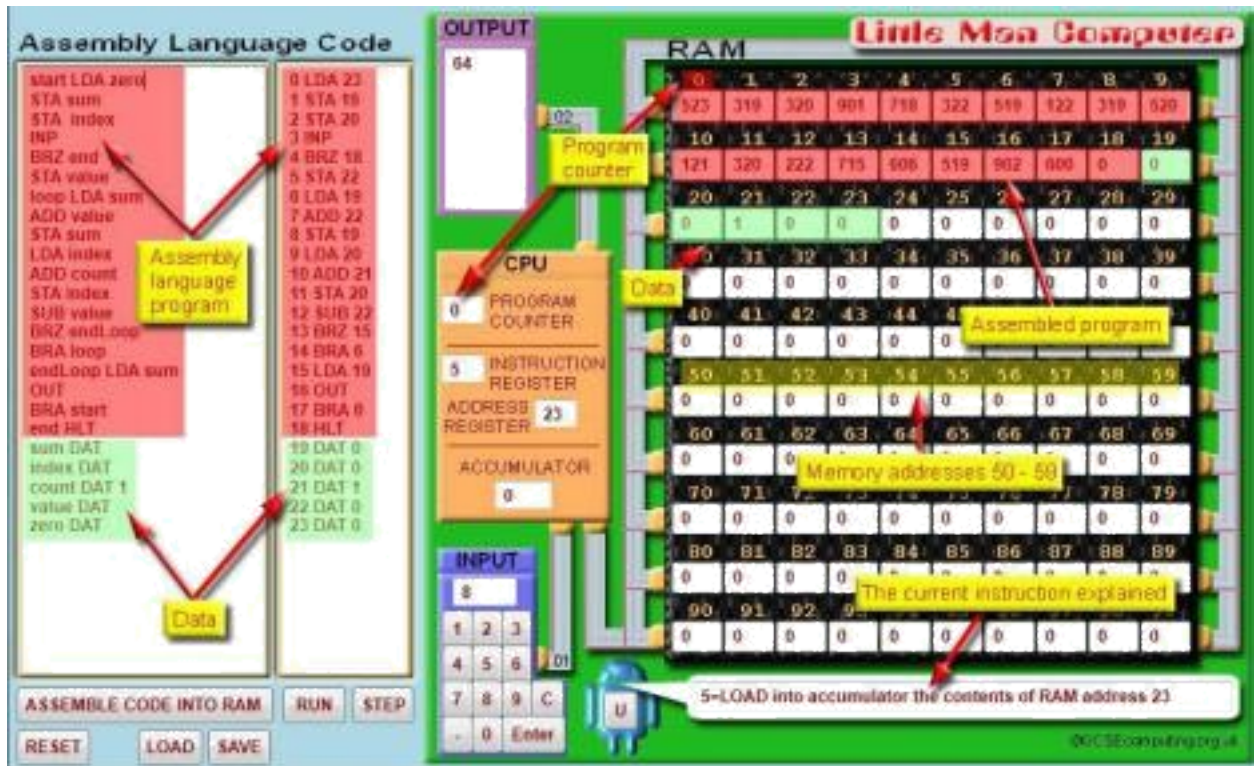


Figure 1: LMC

4.2. Understanding the LMC Simulator

- The **100 memory addresses (Mailboxes)** in the computer memory are numbered 0 to 99 and can each contain a '**machine code**' instruction or data.
- Each **assembly language instruction** is made up of a 3-letter mnemonic (which represents the operation code), usually followed by the memory address of the data the CPU is to act on (this is called absolute memory addressing).
- The **Input box** allows the user to enter numerical data (-999 to 999) while a program is running and load it into the **accumulator**.
- The **Output box** can output the contents of the accumulator while a program is running.
- The results of any **ADD** or **SUBTRACT** instructions are stored in the **Accumulator**.
- The **Program Counter** stores the memory address of the instruction being carried out. It will automatically increment by 1 after each instruction is completed.
- If the CPU receives a **non-sequential** instruction to **branch** (BRA, BRP or BRZ) then the Program Counter is set to the memory address of that instruction.
- To **restart** a program, the Program Counter is reset to 0.

4.3. How LMC works?

1. Check the **Program Counter** so it knows the memory address to look at.
2. **Fetch** the **instruction** from the memory address that matches the program counter.
3. **Increment** the Program Counter (so that it contains the next memory).
4. **Decode** the instruction (includes finding the memory address for any data it refers to).
5. **Execute** the instruction

4.4. Instruction Set

The best way to learn the LMC is by running a set of codes, from the simplest to the more advanced gradually, rather than making an effort to understand the simulator fully at first. This is the approach adopted in this tutorial.

Before that, however, you have to be familiar with the set of instructions: there are not many; just 11 of them. They are as follows:

INSTRUCTION	MNEMONIC	MACHINE CODE
Input	INP	901
Output	OUT	902
End	HLT	000
Add	ADD	1xx
Subtract	SUB	2xx
Store	STA	3xx
Load	LDA	5xx
Branch Always	BRA	6xx
Branch if Zero	BRZ	7xx
Branch if Zero or Positive	BRP	8xx
Data Location	DAT	-

Table 2: Instruction Set

Instruction	Mnemonic	Code	What it does
INPUT	INP	901	Copy the value from the "input box" onto the accumulator (calculator).
OUTPUT	OUT	902	Copy the value from the accumulator (calculator) to the "output box."
END	HLT	000	Causes the Little Man Computer to stop executing your program.
ADD	ADD	1xx	Add the contents of the given mailbox onto the accumulator (calculator).
SUBTRACT	SUB	2xx	Subtract the contents of the given mailbox from the accumulator (calculator)
STORE	STA	3xx	Store the contents of the accumulator (calculator) to the mailbox of the given address
LOAD	LDA	5xx	Load the contents of the given mailbox onto the accumulator (calculator). Note: the contents of the mailbox are not changed.
BRANCH ALWAYS	BRA	6xx	Set the contents of the accumulator (calculator) to the given address
BRANCH IF ZERO	BRZ	7xx	If the contents of the accumulator (calculator) are 000, the PC (program counter) will be set to the given address
BRANCH IF ZERO OR POSITIVE	BRP	8xx	If the contents of the accumulator (calculator) are 000 or positive (i.e. the negative flag is not set), the PC (program counter) will be set to the given address.
DATA LOCATION	DAT	-	When compiled, a program converts each instruction into a three-digit code. These codes are placed in sequential mailboxes. Instead of a program component, this instruction will reserve the next mailbox for data storage.

Table 3: Instruction Set

5. Quiz of Previous Labs

At the start of this lab, you will have a quiz from the previous labs (Lab3, Lab4, Lab5).

6. Walkthrough Tasks

6.1. First LMC Program

1. Click [here](#) to open the LMC simulator. When you will open this link then the following screen will appear as shown in **Figure 2**.

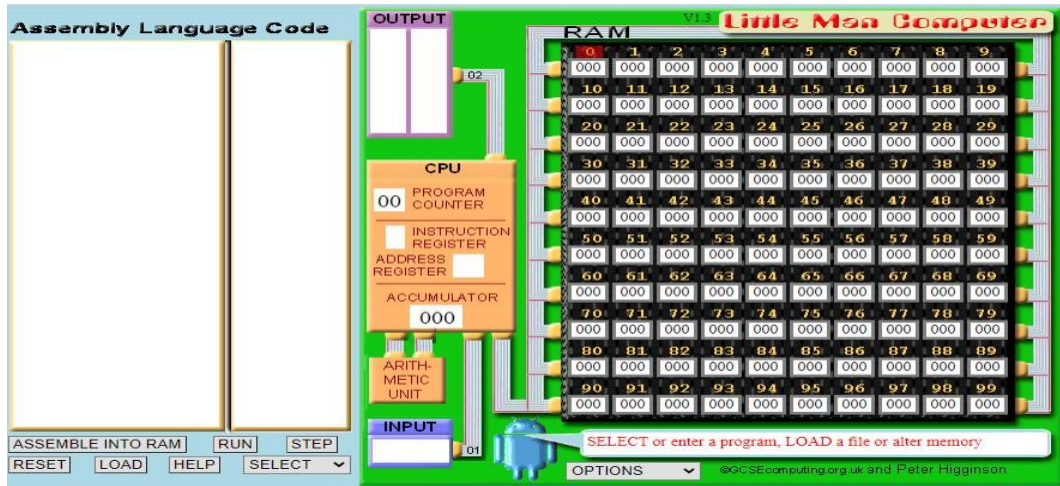


Figure 2: LMC

2. Now start writing the LMC program by double-clicking the area that is showing the text “Start writing the LMC Program hereby double-clicking here” in **Figure 3**.

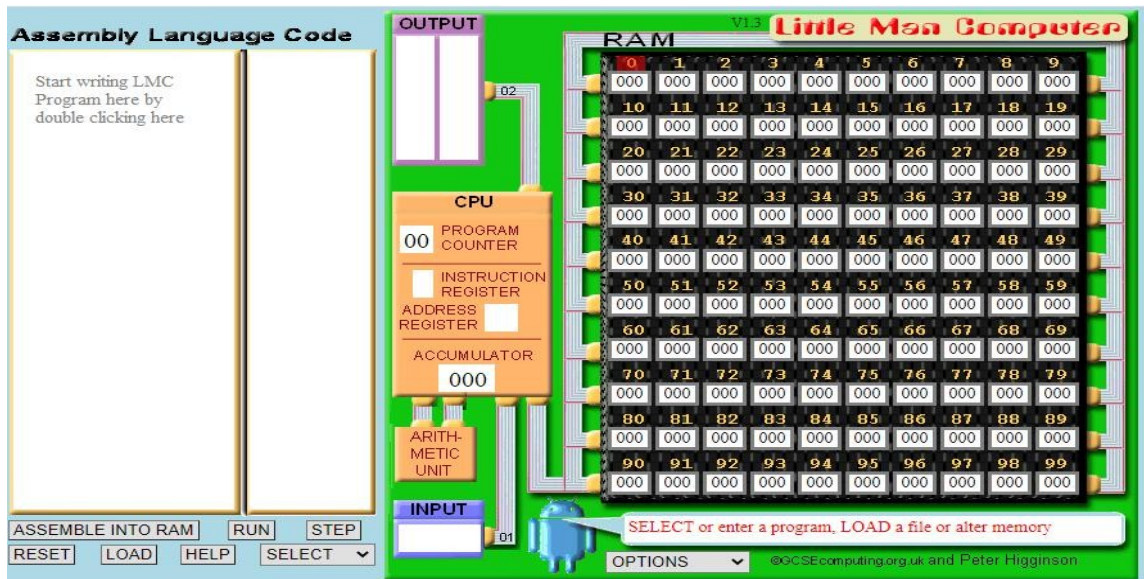


Figure 3: LMC [writing program]

3. Now start writing code with the help of the instruction set that we have discussed in **heading 4.4**. For example, if we want to add two numbers input by the user. Then we will write the following instructions. For now, see the instructions in **Figure 4**. These instructions will be explained in the upcoming steps.

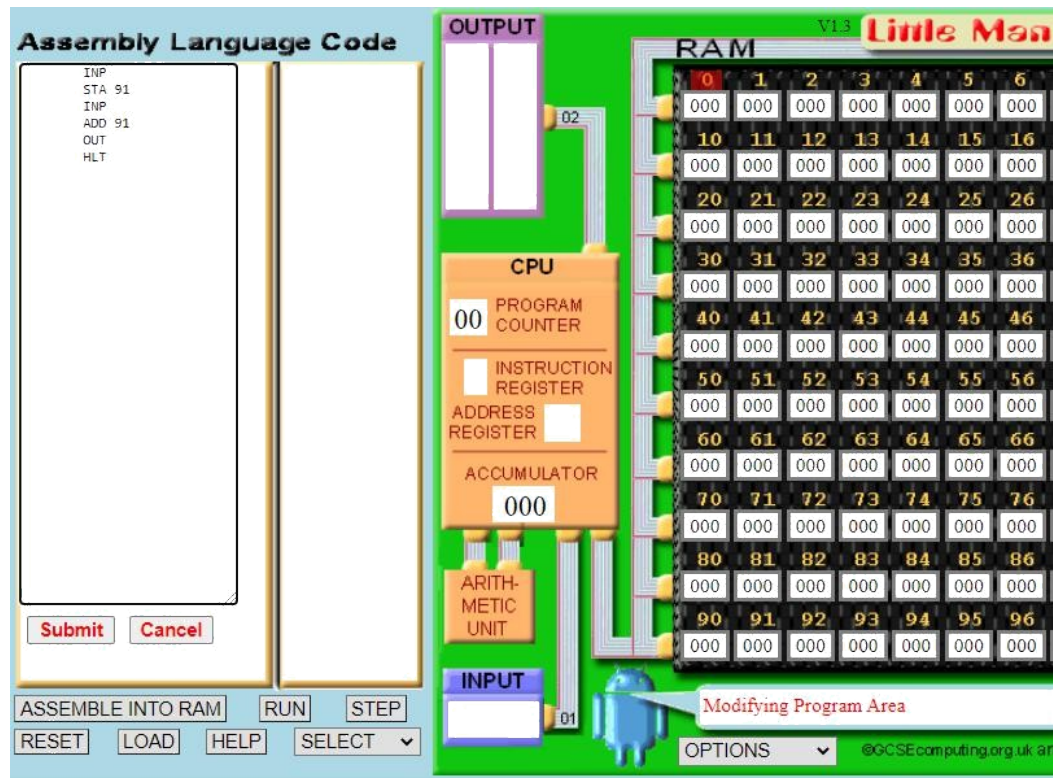


Figure 4: LMC[First Program: Adding two input numbers]

- After writing the code click on the submit button as shown in Figure 4. Then you will see the same code highlighted in Figure 5 and at the same time you can see some values in mailboxes and it is also highlighted in Figure 5. In other words, we have assembled code to the RAM.

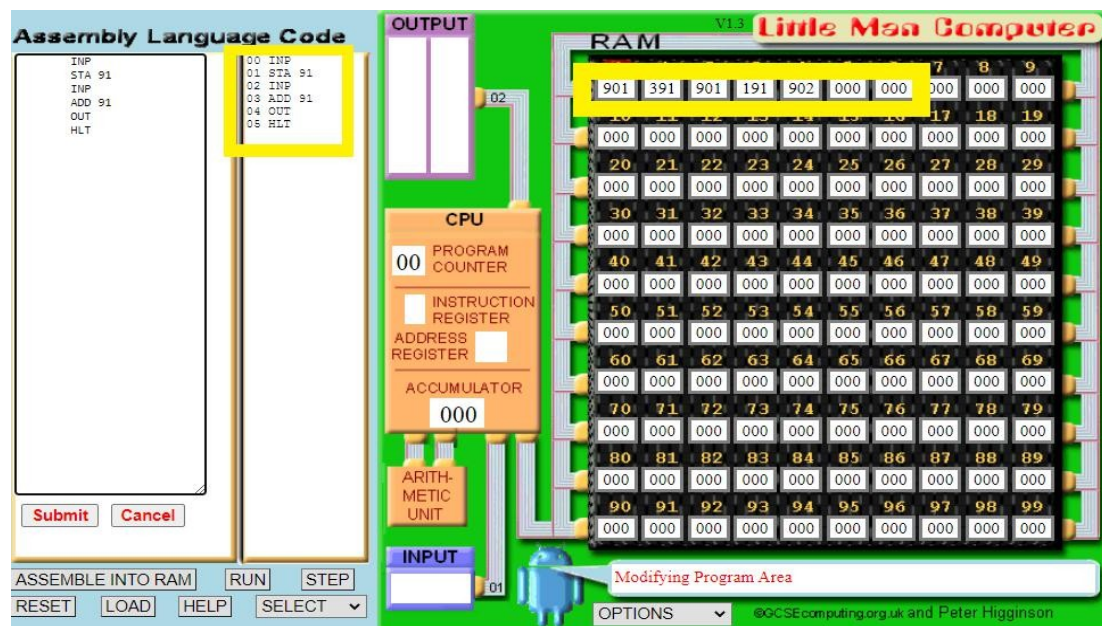


Figure 5: LMC [assembling code to the RAM]

5. Now we try to understand the code.

- i. In the first line of the code, Instruction “**INP**” is written. You can see the description of this instruction from Table 2 or Table 3 of this manual. As described by “Table 3” of this manual the mnemonic INP has a machine code 901 and its function is to take input the value from the input box and save it in an accumulator. When the submit button is pressed then **901** is also set to mailbox **0**. **901** is a machine code, when the program will start executing then it will read 901 instead of INP.
- ii. Then the second instruction is “**STA**” and “**91**” is also with STA instruction. This means that store the value of accumulator (input value) in mailbox 91. And machine code for this instruction will be **391** (mailbox 1) and here **3** is for store instruction and **91** is the mailbox address. So it will store the value of the accumulator at mailbox **91**.
- iii. The third instruction is again **INP** instruction and it will input the value from the user and machine code **901** will be placed at mailbox 2 and the input value will be set in the accumulator.
- iv. The fourth instruction is **ADD** instruction and the machine code for the **ADD** instruction is **1xx**. **1** is for **ADD** instruction and **xx** represents the address. Here in the fourth instruction **ADD 91** means to add the value of accumulator (here in our case the value entered by user after third instruction) to the value stored at the address **91**. The addition will be done by the Arithmetic unit and the result of the addition will be stored in the accumulator.
- v. The fifth instruction is **OUT** instruction and its machine code is 902 (mailbox 4). The **OUT** instruction reads the value from the accumulator and sends it to the output box. We will see the output (addition result of two inputs) in the output box.
- vi. The last instruction is **HLT** instruction and its purpose is to stop the program. The machine code for this instruction is **000**.

6.2. Important LMC components

Program Counter:

The Program Counter holds the address of the next instruction the Little Man will carry out. This Program Counter is incremented by 1 after each instruction is executed, allowing the Little Man to work through a program sequentially.

Instruction register:

Instruction Register is used to hold the top digit of the instruction read from memory. For example, in our case the second instruction was **391** then **3** will be placed in the instruction register.

Address register

Address Register to hold the bottom two digits of the instruction read from memory. . For example in our case the second instruction was **391** then **91** will be placed in the address register.

Accumulator

An Accumulator to store the result of the last operation or calculation.

Arithmetic Unit

An Arithmetic Unit to do calculations.

Input box

An Input area into which a number can be typed when needed.

Output box

An Output area where any numbers output is printed.

7. Practice Tasks in Lab

In this part, you are provided with some practice tasks that will help you to understand LMC. You have to complete all the given tasks below and when you complete any task, save it in a text file and also mention the task no before the LMC instructions of that task.

7.1. Task 1

Write a program that takes two numbers as input and output their sum.

7.2. Task2

Write a program that takes 2 numbers as input and subtracts the first number from the second number. The result of this subtraction should be stored on the “99” mailbox and also be shown on the output box.

7.3. Task3

Write a program that takes three numbers as input and output their sum.

7.4. Task4

Write a program that saves any number (for example **50**) into the RAM using DAT mnemonic and loads that number (in our case **50**) into the accumulator and then shows it in the output box.

7.5. Task5

Write a program that takes one number as input and add it to the number “31”. You have to store the result in the mailbox “99”. You are also required to show the result in the output box.

Hint: number 31 is stored in the mailbox using a **DAT** mnemonic.

7.6. Task 6

Write a program to copy the value of one variable to another.

7.7. Task 7

Write a program to swap the values of two variables.

8. Unseen Task

You will be given unseen tasks you have to perform and submit during the lab.

9. Assignment

Assignment for this Lab will be uploaded on LMS after the Lab session. The deadline will also be mentioned on LMS.

10. Quiz

A quiz about this lab will be conducted at the end of the LMC Labs.

11. Further Readings

The reading material and other helping material regarding this Lab will be uploaded on LMS if any. You can access it from the LMS.

12. Evaluation Criteria

📁 **Method of Evaluation:** Viva, File submitted on LMS.

	Excellent 10	Good 9-7	Satisfactory 6-4	Unsatisfactory 3-1	Poor 0	Marks Obtained
Unseen Tasks	All tasks were completed correctly in the given time.	Most tasks were completed correctly. Tasks could be improved further.	Some tasks were completed correctly.	Most tasks were incomplete or incorrect	All tasks were incomplete or incorrect. Didn't perform tasks	
Assignment	All tasks completed correctly in given time and have a complete understanding	Most tasks were completed correctly. Tasks could be improved further and have a complete understanding	Some tasks were completed correctly and have an incomplete understanding	Most tasks were incomplete or incorrect and have no understanding	All tasks were incomplete or incorrect. Didn't perform tasks and have no understanding	
Total						



SHIFA TAMEER-E-MILLAT UNIVERSITY

Lab Manual **Introduction to Computer Science**

DEPARTMENT OF COMPUTING

Shifa Tameer-e-Millat University

Department of Computing

INTRODUCTION TO COMPUTER SCIENCE

Lab 7

LMC Programming

Lab 7: LMC Programming

1. Introduction

A Little Man Computer (LMC) is a simulator which has many of the basic features of a modern computer that uses the Von Neumann architecture. The LMC is based on the idea of a ‘Little Man’ acting as the control unit of a CPU, fetching instructions from RAM, decoding and executing them as well as managing the input and output mechanisms. LMC can be programmed by using a basic set of 10 assembly code instructions which are then assembled into machine code (although in decimal not binary).

2. Activity Timeboxing

Heading No	Activity Name	Activity Time	Total Time
5	Unseen task from Previous Lab	30 minutes	30 minutes
6	Walkthrough Task	50 minutes	80 minutes
7	Practice Task in Lab	70 minutes	150 minutes

Table 1: Activity Timeboxing

3. Objective

- To understand the Little Man Computer (LMC).
- To understand the working of a modern computer using Little Man Computer (LMC).
- To write the basic programs in LMC

4. Concept Map

In this lab, we learn about the LMC and its working. We will also relate its working with the working of modern computers. We will also write some basic programs in LMC.

4.1. Little Man Computer (LMC)

A Little Man Computer (LMC) is a simulator that has many of the basic features of modern computers. The LMC is based on the idea of a ‘Little Man’ acting as the control unit of a CPU, fetching instructions from RAM, decoding and executing them as well as managing the input and output mechanisms.

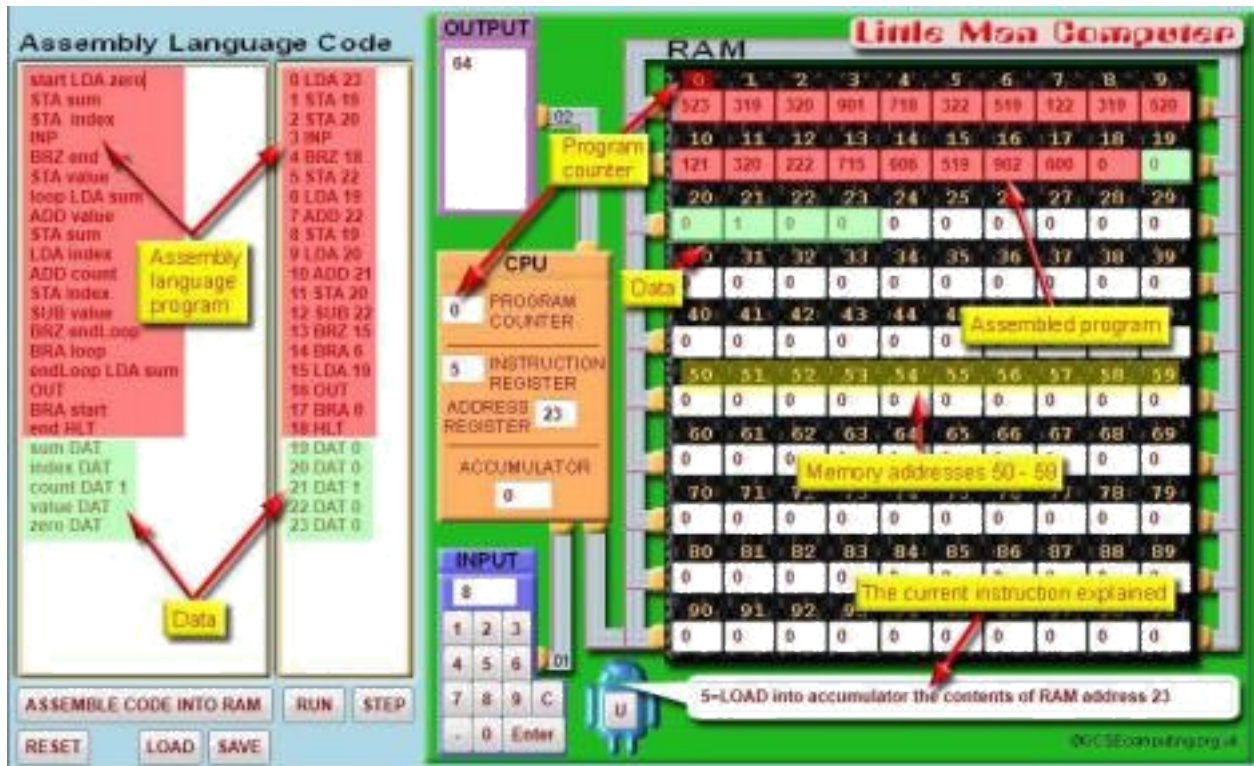


Figure 1: LMC

4.2. Understanding the LMC Simulator

- The **100 memory addresses (Mailboxes)** in the computer memory are numbered 0 to 99 and can each contain a '**machine code**' instruction or data.
- Each **assembly language instruction** is made up of a 3-letter mnemonic (which represents the operation code), usually followed by the memory address of the data the CPU is to act on (this is called absolute memory addressing).
- The **Input box** allows the user to enter numerical data (-999 to 999) while a program is running and load it into the **accumulator**.
- The **Output box** can output the contents of the accumulator while a program is running.
- The results of any **ADD** or **SUBTRACT** instructions are stored in the **Accumulator**.
- The **Program Counter** stores the memory address of the instruction being carried out. It will automatically increment by 1 after each instruction is completed.
- If the CPU receives a **non-sequential** instruction to **branch** (BRA, BRP or BRZ) then the Program Counter is set to the memory address of that instruction.
- To **restart** a program, the Program Counter is reset to 0.

4.3. How LMC works?

1. Check the **Program Counter** so it knows the memory address to look at.
2. **Fetch** the **instruction** from the memory address that matches the program counter.
3. **Increment** the Program Counter (so that it contains the next memory).
4. **Decode** the instruction (includes finding the memory address for any data it refers to).
5. **Execute** the instruction

4.4. Instruction Set

The best way to learn the LMC is by running a set of codes, from the simplest to the more advanced gradually, rather than making an effort to understand the simulator fully at first. This is the approach adopted in this tutorial.

Before that, however, you have to be familiar with the set of instructions: there are not many; just 11 of them. They are as follows:

INSTRUCTION	MNEMONIC	MACHINE CODE
Input	INP	901
Output	OUT	902
End	HLT	000
Add	ADD	1xx
Subtract	SUB	2xx
Store	STA	3xx
Load	LDA	5xx
Branch Always	BRA	6xx
Branch if Zero	BRZ	7xx
Branch if Zero or Positive	BRP	8xx
Data Location	DAT	-

Table 2: Instruction Set

Instruction	Mnemonic	Code	What it does
INPUT	INP	901	Copy the value from the "input box" onto the accumulator (calculator).
OUTPUT	OUT	902	Copy the value from the accumulator (calculator) to the "output box."
END	HLT	000	Causes the Little Man Computer to stop executing your program.
ADD	ADD	1xx	Add the contents of the given mailbox onto the accumulator (calculator).
SUBTRACT	SUB	2xx	Subtract the contents of the given mailbox from the accumulator (calculator)
STORE	STA	3xx	Store the contents of the accumulator (calculator) to the mailbox of the given address
LOAD	LDA	5xx	Load the contents of the given mailbox onto the accumulator (calculator). Note: the contents of the mailbox are not changed.
BRANCH ALWAYS	BRA	6xx	Set the contents of the accumulator (calculator) to the given address
BRANCH IF ZERO	BRZ	7xx	If the contents of the accumulator (calculator) are 000, the PC (program counter) will be set to the given address
BRANCH IF ZERO OR POSITIVE	BRP	8xx	If the contents of the accumulator (calculator) are 000 or positive (i.e. the negative flag is not set), the PC (program counter) will be set to the given address.
DATA LOCATION	DAT	-	When compiled, a program converts each instruction into a three-digit code. These codes are placed in sequential mailboxes. Instead of a program component, this instruction will reserve the next mailbox for data storage.

Table 3: Instruction Set

5. Walkthrough Tasks

5.1. First LMC Program

1. Click [here](#) to open the LMC simulator. When you will open this link then the following screen will appear as shown in **Figure 2**.

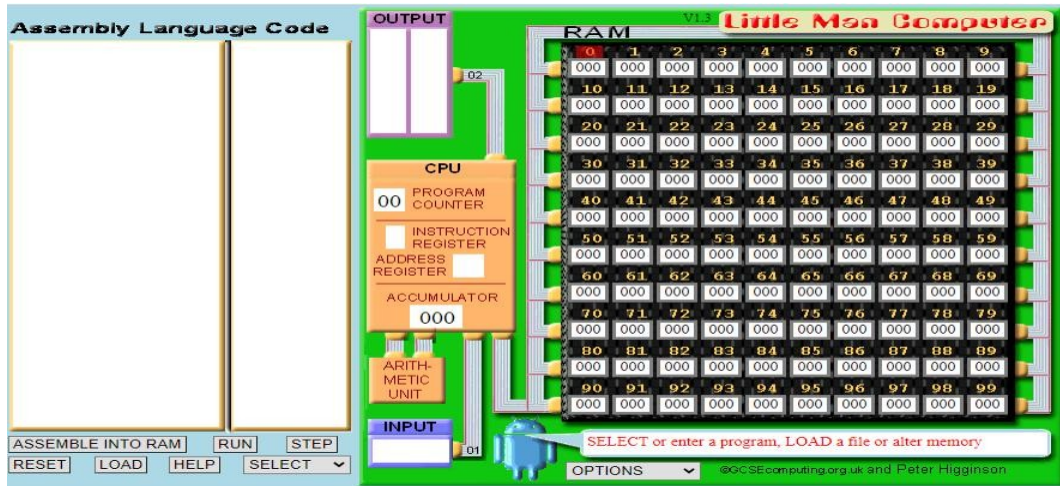


Figure 2: LMC

2. Now start writing the LMC program by double-clicking the area that is showing the text “Start writing the LMC Program hereby double-clicking here” in **Figure 3**.

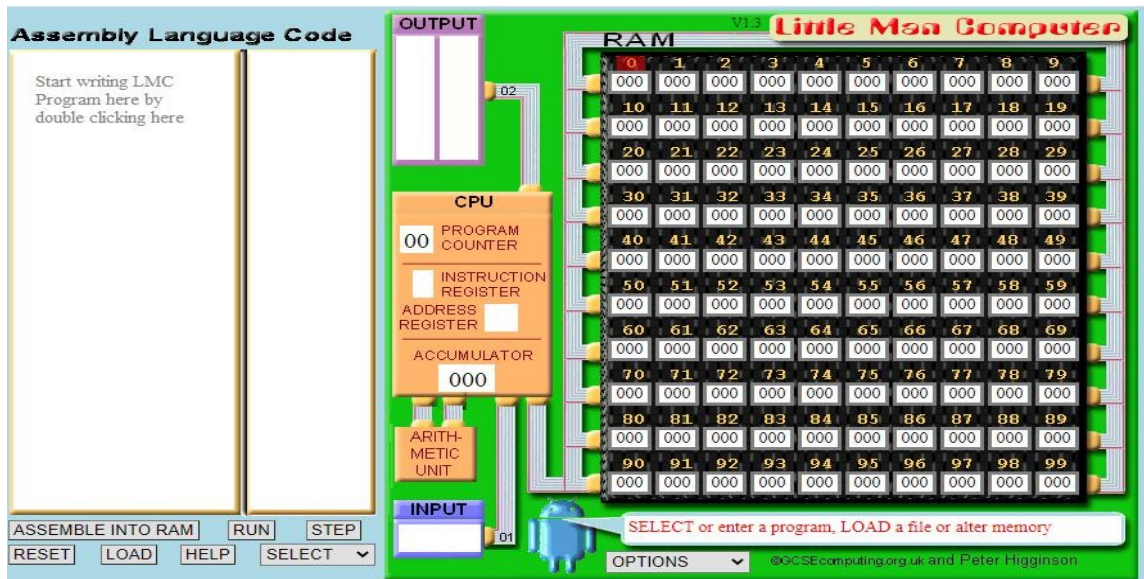


Figure 3: LMC [writing program]

3. Now start writing code with the help of the instruction set that we have discussed in **heading 4.4**. For example, if we want to add two numbers input by the user. Then we will write the following instructions. For now, see the instructions in **Figure 4**. These instructions will be explained in the upcoming steps.

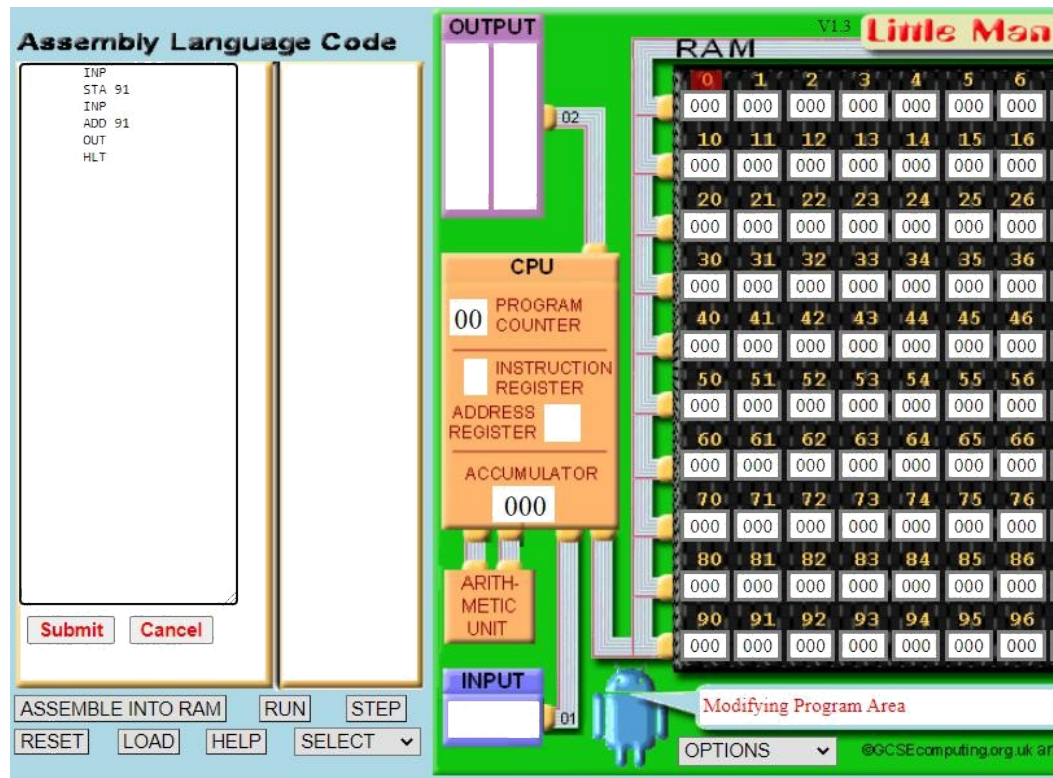


Figure 4: LMC[First Program: Adding two input numbers]

- After writing the code click on the submit button as shown in **Figure 4**. Then you will see the same code highlighted in **Figure 5** and at the same time you can see some values in mailboxes and it is also highlighted in **Figure 5**. In other words, we have assembled code to the RAM.

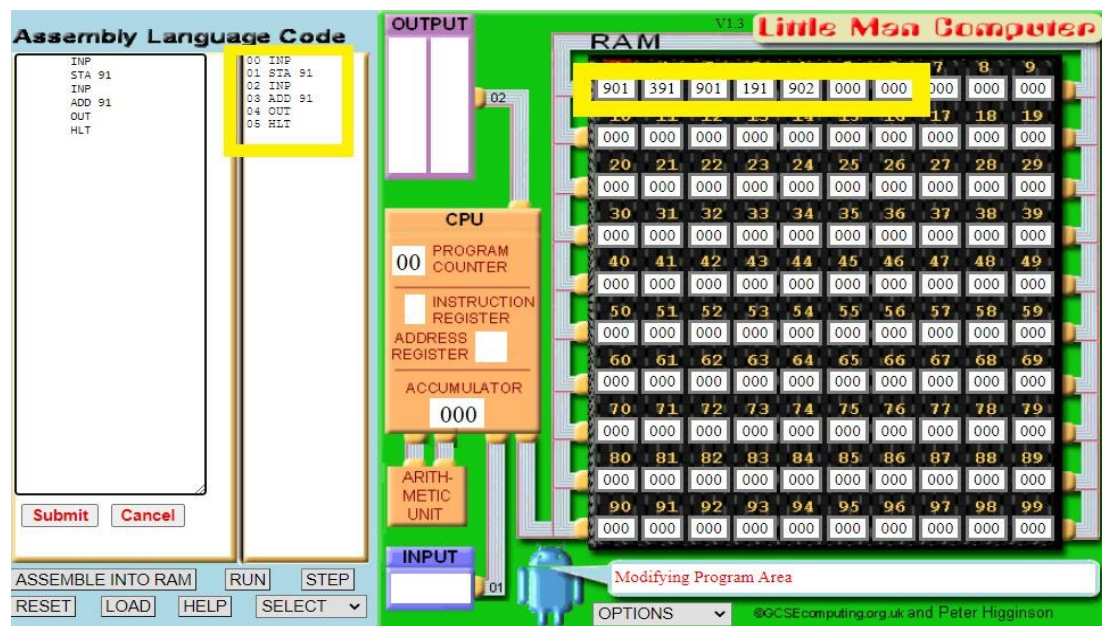


Figure 5: LMC [assembling code to the RAM]

5. Now we try to understand the code.

- i. In the first line of the code, Instruction “**INP**” is written. You can see the description of this instruction from Table 2 or Table 3 of this manual. As described by “Table 3” of this manual the mnemonic INP has a machine code 901 and its function is to take input the value from the input box and save it in an accumulator. When the submit button is pressed then **901** is also set to mailbox **0**. **901** is a machine code, when the program will start executing then it will read 901 instead of INP.
- ii. Then the second instruction is “**STA**” and “**91**” is also with STA instruction. This means that store the value of accumulator (input value) in mailbox 91. And machine code for this instruction will be **391** (mailbox 1) and here **3** is for store instruction and **91** is the mailbox address. So it will store the value of the accumulator at mailbox **91**.
- iii. The third instruction is again **INP** instruction and it will input the value from the user and machine code **901** will be placed at mailbox 2 and the input value will be set in the accumulator.
- iv. The fourth instruction is **ADD** instruction and the machine code for the **ADD** instruction is **1xx**. **1** is for **ADD** instruction and **xx** represents the address. Here in the fourth instruction **ADD 91** means to add the value of accumulator (here in our case the value entered by user after third instruction) to the value stored at the address **91**. The addition will be done by the Arithmetic unit and the result of the addition will be stored in the accumulator.
- v. The fifth instruction is **OUT** instruction and its machine code is 902 (mailbox 4). The **OUT** instruction reads the value from the accumulator and sends it to the output box. We will see the output (addition result of two inputs) in the output box.
- vi. The last instruction is **HLT** instruction and its purpose is to stop the program. The machine code for this instruction is **000**.

5.2. Important LMC components

Program Counter:

The Program Counter holds the address of the next instruction the Little Man will carry out. This Program Counter is incremented by 1 after each instruction is executed, allowing the Little Man to work through a program sequentially.

Instruction register:

Instruction Register is used to hold the top digit of the instruction read from memory. For example, in our case the second instruction was **391** then **3** will be placed in the instruction register.

Address register

Address Register to hold the bottom two digits of the instruction read from memory. . For example in our case the second instruction was **391** then **91** will be placed in the address register.

Accumulator

An Accumulator to store the result of the last operation or calculation.

Arithmetic Unit

An Arithmetic Unit to do calculations.

Input box

An Input area into which a number can be typed when needed.

Output box

An Output area where any numbers output is printed.

6. Practice Tasks in Lab

In this part, you are provided with some practice tasks that will help you to understand LMC. You have to complete all the given tasks below and when you complete any task, save it in a text file and also mention the task no before the LMC instructions of that task.

6.1. Task 1

Write a program that takes two numbers as input and output the maximum number.

6.2. Task2

Write a program that prints the number from 1 to 10.

6.3. Task3

Write a program that prints the table of any number up to 10. (that number is already saved in the RAM)

6.4. Task4

Write a program that prints the table of any number up to 10. And that number is taken as an input from the user.

6.5. Task5

Write a program that takes two numbers as input and output their multiplication.

7. Unseen Task

You will be given unseen tasks you have to perform and submit during the lab.

8. Assignment

Assignment for this Lab will be uploaded on LMS after the Lab session. The deadline will also be mentioned on LMS.


9. Quiz

A quiz about this lab will be conducted at the end of the LMC Labs.

10. Further Readings

The reading material and other helping material regarding this Lab will be uploaded on LMS if any.
You can access it from the LMS.

11. Evaluation Criteria

 **Method of Evaluation:** Viva, File submitted on LMS.

	Excellent 10	Good 9-7	Satisfactory 6-4	Unsatisfactory 3-1	Poor 0	Marks Obtained
Unseen Tasks	All tasks were completed correctly in the given time.	Most tasks were completed correctly. Tasks could be improved further.	Some tasks were completed correctly.	Most tasks were incomplete or incorrect	All tasks were incomplete or incorrect. Didn't perform tasks	
Assignment	All tasks completed correctly in given time and have a complete understanding	Most tasks were completed correctly. Tasks could be improved further and have a complete understanding	Some tasks were completed correctly and have an incomplete understanding	Most tasks were incomplete or incorrect and have no understanding	All tasks were incomplete or incorrect. Didn't perform tasks and have no understanding	
Total						



SHIFA TAMEER-E-MILLAT UNIVERSITY

Lab Manual **Introduction to Computer Science**

DEPARTMENT OF COMPUTING

Shifa Tameer-e-Millat University

Department of Computing

INTRODUCTION TO COMPUTER SCIENCE

Lab 8

LMC Programming

Lab 8: LMC Programming

1. Introduction

A Little Man Computer (LMC) is a simulator which has many of the basic features of a modern computer that uses the Von Neumann architecture. The LMC is based on the idea of a ‘Little Man’ acting as the control unit of a CPU, fetching instructions from RAM, decoding and executing them as well as managing the input and output mechanisms. LMC can be programmed by using a basic set of 10 assembly code instructions which are then assembled into machine code (although in decimal not binary).

2. Activity Timeboxing

Heading No	Activity Name	Activity Time	Total Time
5	Walkthrough Task	30 minutes	30 minutes
6	Practice Task in Lab	120 minutes	150 minutes

Table 1: Activity Timeboxing

3. Objective

- To understand the Little Man Computer (LMC).
- To understand the working of a modern computer using Little Man Computer (LMC).
- To write the basic programs in LMC

4. Concept Map

In this lab, we learn about the LMC and its working. We will also relate its working with the working of modern computers. We will also write some basic programs in LMC.

4.1. Little Man Computer (LMC)

A Little Man Computer (LMC) is a simulator that has many of the basic features of modern computers. The LMC is based on the idea of a ‘Little Man’ acting as the control unit of a CPU, fetching instructions from RAM, decoding and executing them as well as managing the input and output mechanisms.

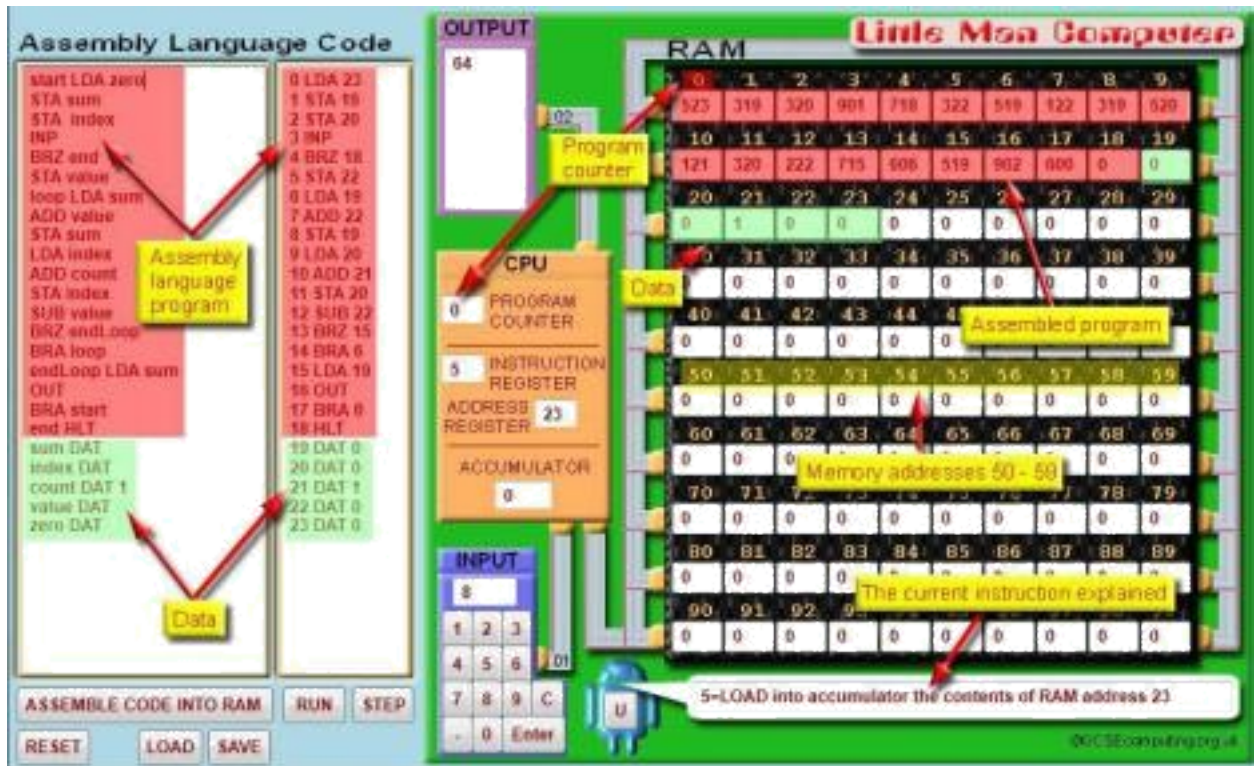


Figure 1: LMC

4.2. Understanding the LMC Simulator

- The **100 memory addresses (Mailboxes)** in the computer memory are numbered 0 to 99 and can each contain a '**machine code**' instruction or data.
- Each **assembly language instruction** is made up of a 3-letter mnemonic (which represents the operation code), usually followed by the memory address of the data the CPU is to act on (this is called absolute memory addressing).
- The **Input box** allows the user to enter numerical data (-999 to 999) while a program is running and load it into the **accumulator**.
- The **Output box** can output the contents of the accumulator while a program is running.
- The results of any **ADD** or **SUBTRACT** instructions are stored in the **Accumulator**.
- The **Program Counter** stores the memory address of the instruction being carried out. It will automatically increment by 1 after each instruction is completed.
- If the CPU receives a **non-sequential** instruction to **branch** (BRA, BRP or BRZ) then the Program Counter is set to the memory address of that instruction.
- To **restart** a program, the Program Counter is reset to 0.

4.3. How LMC works?

1. Check the **Program Counter** so it knows the memory address to look at.
2. **Fetch** the **instruction** from the memory address that matches the program counter.
3. **Increment** the Program Counter (so that it contains the next memory).
4. **Decode** the instruction (includes finding the memory address for any data it refers to).
5. **Execute** the instruction

4.4. Instruction Set

The best way to learn the LMC is by running a set of codes, from the simplest to the more advanced gradually, rather than making an effort to understand the simulator fully at first. This is the approach adopted in this tutorial.

Before that, however, you have to be familiar with the set of instructions: there are not many; just 11 of them. They are as follows:

INSTRUCTION	MNEMONIC	MACHINE CODE
Input	INP	901
Output	OUT	902
End	HLT	000
Add	ADD	1xx
Subtract	SUB	2xx
Store	STA	3xx
Load	LDA	5xx
Branch Always	BRA	6xx
Branch if Zero	BRZ	7xx
Branch if Zero or Positive	BRP	8xx
Data Location	DAT	-

Table 2: Instruction Set

Instruction	Mnemonic	Code	What it does
INPUT	INP	901	Copy the value from the "input box" onto the accumulator (calculator).
OUTPUT	OUT	902	Copy the value from the accumulator (calculator) to the "output box."
END	HLT	000	Causes the Little Man Computer to stop executing your program.
ADD	ADD	1xx	Add the contents of the given mailbox onto the accumulator (calculator).
SUBTRACT	SUB	2xx	Subtract the contents of the given mailbox from the accumulator (calculator)
STORE	STA	3xx	Store the contents of the accumulator (calculator) to the mailbox of the given address
LOAD	LDA	5xx	Load the contents of the given mailbox onto the accumulator (calculator). Note: the contents of the mailbox are not changed.
BRANCH ALWAYS	BRA	6xx	Set the PC (program counter) to the given address
BRANCH IF ZERO	BRZ	7xx	If the contents of the accumulator (calculator) are 000, the PC (program counter) will be set to the given address
BRANCH IF ZERO OR POSITIVE	BRP	8xx	If the contents of the accumulator (calculator) are 000 or positive (i.e. the negative flag is not set), the PC (program counter) will be set to the given address.
DATA LOCATION	DAT	-	When compiled, a program converts each instruction into a three-digit code. These codes are placed in sequential mailboxes. Instead of a program component, this instruction will reserve the next mailbox for data storage.

Table 3: Instruction Set

5. Walkthrough Tasks

5.1. First LMC Program

1. Click [here](#) to open the LMC simulator. When you will open this link then the following screen will appear as shown in **Figure 2**.

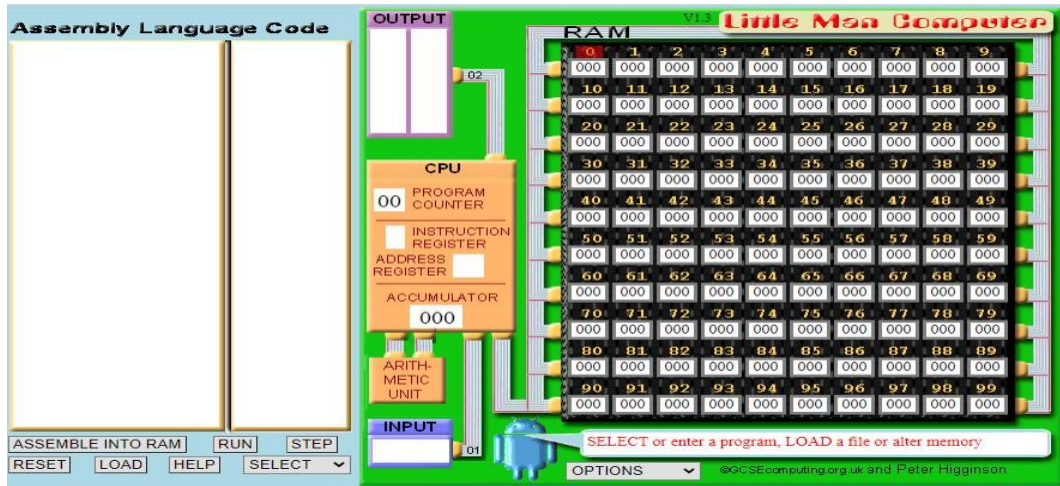


Figure 2: LMC

2. Now start writing the LMC program by double-clicking the area that is showing the text “Start writing the LMC Program hereby double-clicking here” in **Figure 3**.

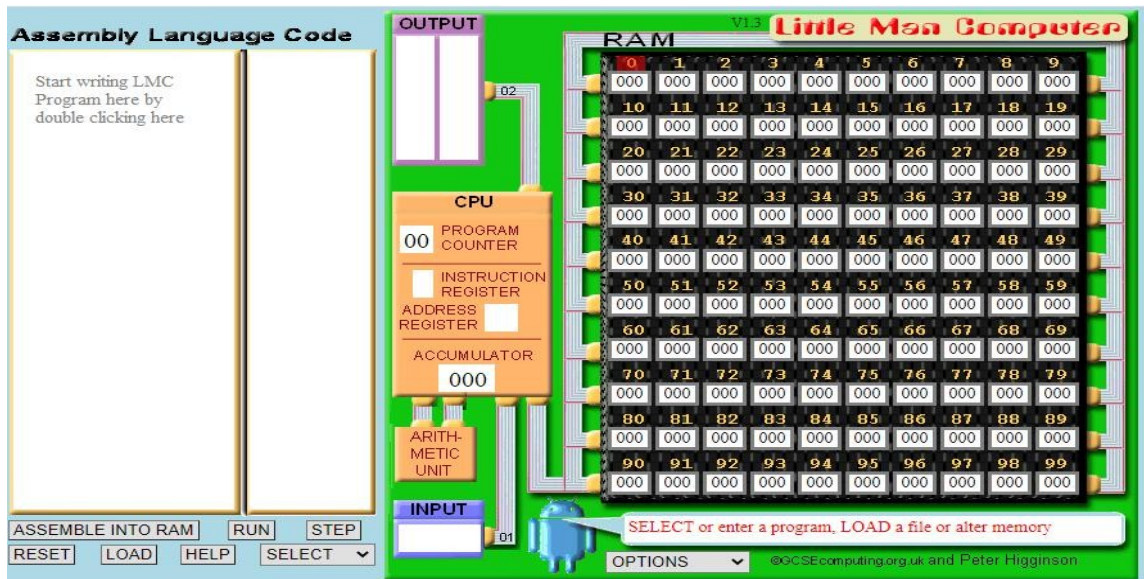


Figure 3: LMC [writing program]

3. Now start writing code with the help of the instruction set that we have discussed in **heading 4.4**. For example, if we want to add two numbers input by the user. Then we will write the following instructions. For now, see the instructions in **Figure 4**. These instructions will be explained in the upcoming steps.

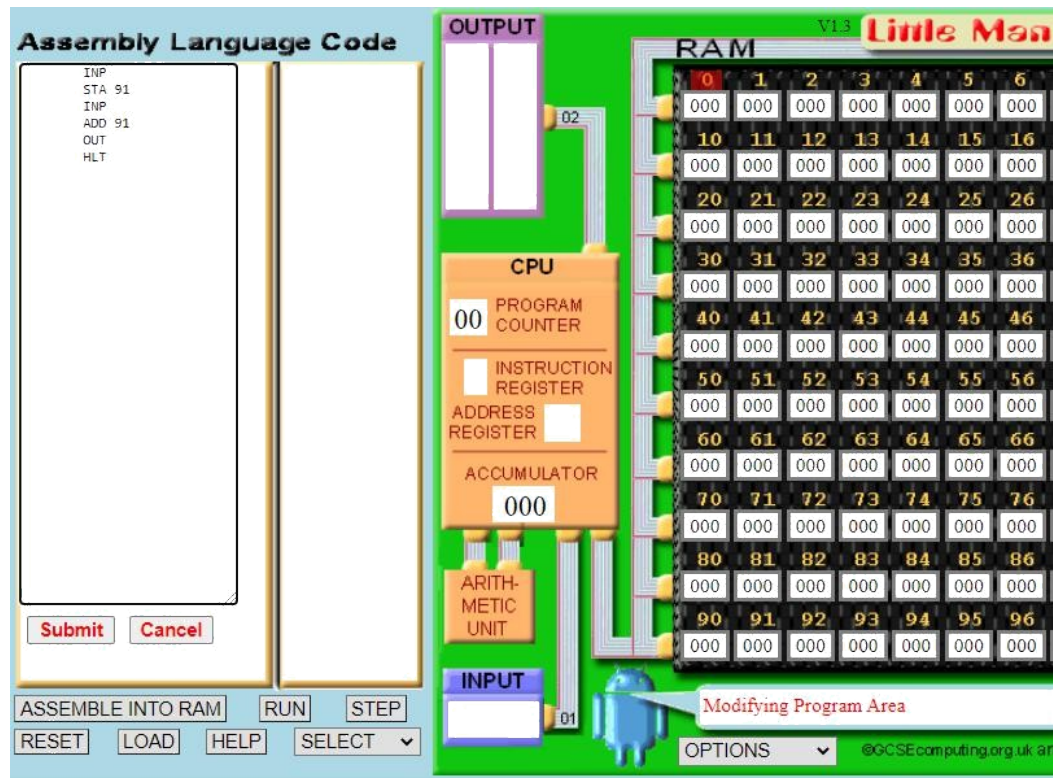


Figure 4: LMC[First Program: Adding two input numbers]

- After writing the code click on the submit button as shown in Figure 4. Then you will see the same code highlighted in Figure 5 and at the same time you can see some values in mailboxes and it is also highlighted in Figure 5. In other words, we have assembled code to the RAM.

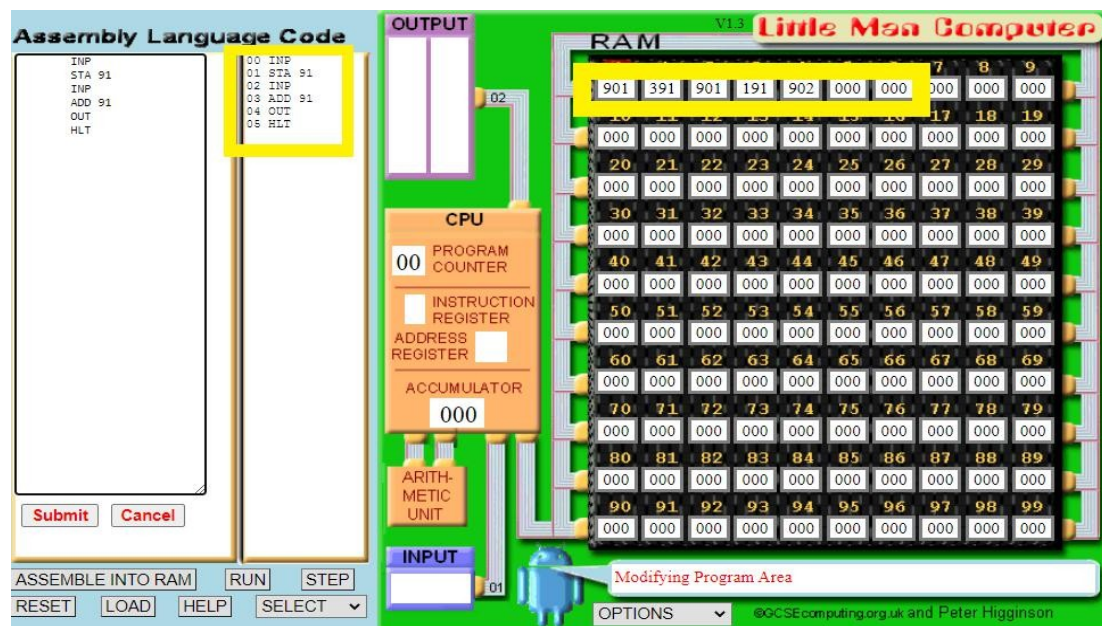


Figure 5: LMC [assembling code to the RAM]

5. Now we try to understand the code.

- i. In the first line of the code, Instruction “**INP**” is written. You can see the description of this instruction from Table 2 or Table 3 of this manual. As described by “Table 3” of this manual the mnemonic INP has a machine code 901 and its function is to take input the value from the input box and save it in an accumulator. When the submit button is pressed then **901** is also set to mailbox **0**. **901** is a machine code, when the program will start executing then it will read 901 instead of INP.
- ii. Then the second instruction is “**STA**” and “**91**” is also with STA instruction. This means that store the value of accumulator (input value) in mailbox 91. And machine code for this instruction will be **391** (mailbox 1) and here **3** is for store instruction and **91** is the mailbox address. So it will store the value of the accumulator at mailbox **91**.
- iii. The third instruction is again **INP** instruction and it will input the value from the user and machine code **901** will be placed at mailbox 2 and the input value will be set in the accumulator.
- iv. The fourth instruction is **ADD** instruction and the machine code for the **ADD** instruction is **1xx**. **1** is for **ADD** instruction and **xx** represents the address. Here in the fourth instruction **ADD 91** means to add the value of accumulator (here in our case the value entered by user after third instruction) to the value stored at the address **91**. The addition will be done by the Arithmetic unit and the result of the addition will be stored in the accumulator.
- v. The fifth instruction is **OUT** instruction and its machine code is 902 (mailbox 4). The **OUT** instruction reads the value from the accumulator and sends it to the output box. We will see the output (addition result of two inputs) in the output box.
- vi. The last instruction is **HLT** instruction and its purpose is to stop the program. The machine code for this instruction is **000**.

5.2. Important LMC components

Program Counter:

The Program Counter holds the address of the next instruction the Little Man will carry out. This Program Counter is incremented by 1 after each instruction is executed, allowing the Little Man to work through a program sequentially.

Instruction register:

Instruction Register is used to hold the top digit of the instruction read from memory. For example, in our case the second instruction was **391** then **3** will be placed in the instruction register.

Address register

Address Register to hold the bottom two digits of the instruction read from memory. . For example in our case the second instruction was **391** then **91** will be placed in the address register.

Accumulator

An Accumulator to store the result of the last operation or calculation.

Arithmetic Unit

An Arithmetic Unit to do calculations.

Input box

An Input area into which a number can be typed when needed.

Output box

An Output area where any numbers output is printed.

6. Practice Tasks in Lab

In this part, you are provided with some practice tasks that will help you to understand LMC. You have to complete all the given tasks below and when you complete any task, save it in a text file and also mention the task no before the LMC instructions of that task.

1. Write a program that takes any number as input and output that number.
2. Write a program that takes an input and output the result as three times of that number. For example, if user enters 4 as an input the program should output the 12 as an output.
3. Write a program that takes two numbers as input and subtracts the second number from first number.
4. Write a program that should add two numbers and both two numbers are stored in variables. You don't need to take input; the numbers are saved as data variable in the memory.
5. Write a program that takes two numbers as input and tells if both numbers are equal or not. If numbers are equal, then program should print 105 as output otherwise 205.
6. Write a program that takes two numbers as input and check if both numbers are same or not if the numbers are same then program should output that number otherwise it should again take input two numbers until they are the same numbers.
7. Write a program that takes input from user and prints 1 if the numbers are positive or zero, - 1 if the number is negative.
8. Write a program that takes input from user and prints 1 if the number is positive, 0 if the number is zero, -1 if the number is negative.
9. Write a program takes two numbers and prints the maximum number as an output.
10. Write a program that prints the number from 1 to 10.
11. Write a program that prints the table 3 up to 10
12. Write a program that prints the table of any number up to 10.
13. Write a program that prints out the odd numbers from 1 to 25.
14. Write a program that takes two numbers as input and output their multiplication.
15. Write a program that 5 numbers as input and output the maximum number.
16. Write a program that will add the numbers (we don't know how many numbers), program will ask for the input and the first input of the program will decide that how many numbers the program has to add. If the first input by the user is five, then program should take five numbers as an input and output their addition.

7. Unseen Task

You will be given unseen tasks you have to perform and submit during the next lab.

8. Assignment

Assignment for this Lab will be uploaded on LMS after the Lab session. The deadline will also be mentioned on LMS.


9. Quiz

A quiz about this lab will be conducted at the end of the LMC Labs.

10. Further Readings

The reading material and other helping material regarding this Lab will be uploaded on LMS if any. You can access it from the LMS.

11. Evaluation Criteria

 **Method of Evaluation:** Viva, File submitted on LMS.

	Excellent 10	Good 9-7	Satisfactory 6-4	Unsatisfactory 3-1	Poor 0	Marks Obtained
Unseen Tasks	All tasks were completed correctly in the given time.	Most tasks were completed correctly. Tasks could be improved further.	Some tasks were completed correctly.	Most tasks were incomplete or incorrect	All tasks were incomplete or incorrect. Didn't perform tasks	
Assignment	All tasks completed correctly in given time and have a complete understanding	Most tasks were completed correctly. Tasks could be improved further and have a complete understanding	Some tasks were completed correctly and have an incomplete understanding	Most tasks were incomplete or incorrect and have no understanding	All tasks were incomplete or incorrect. Didn't perform tasks and have no understanding	
Total						



SHIFA TAMEER-E-MILLAT UNIVERSITY

Lab Manual **Introduction to Computer Science**

DEPARTMENT OF COMPUTING

Shifa Tameer-e-Millat University

Department of Computing

INTRODUCTION TO COMPUTER SCIENCE

Lab 10

Bash Scripting

Lab 10: Bash Scripting

1. Introduction

A Bash script is a plain text file that contains a series of commands. These commands are a mixture of commands we would normally type ourselves on the command line or terminal (such as `ls` or `cp` for example). The basic concept of a bash script is a list of commands, which are listed in the order of execution. Think of a script for a play, a movie, or a TV show. The script tells the actors what they should say and do. A script for a computer tells the computer what it should do or say. In the context of Bash scripts, we are telling the Bash shell what it should do.

2. Activity Timeboxing

Heading No	Activity Name	Activity Time	Total Time
	About this Lab	20 minutes	20 minutes
5	Walkthrough Task	40 minutes	60 minutes
6	Practice Task in Lab	80 minutes	140 minutes
	Q/A session	10 minutes	150 minutes

Table 1: Activity Timeboxing

3. Objective

- To create a bash script
- To learn about strings, variables, shell execution.
- To learn about user input, comparison, conditions, loops

4. Concept Map

In this lab, we learn about bash scripting, what is bash scripting, few basic scripts. First of all, try to learn about some important terms that we should know while writing bash scripts.

4.1. Kernel

Every operating system (OS) has a kernel. It is the layer of the Operating system that bridges the hardware with the main programs that run on a computer. The kernel is the core of the operating system and is the first to load when the computer boots up. It remains in the computer's memory throughout a session. It is responsible for providing an interface for all applications, controlling the hardware and allowing processes to get information from each other.

4.2. Shell

The shell forms the layer between the user and the kernel so the user can enter commands. The kernel 'understands' only binary language, which is composed exclusively of ones and zeros. In early computing, any instructions/commands from the users were entered in binary language, but this evolved so that the user can enter commands in a more recognizable language. It is the shell that acts as the go-between, accepting the commands entered in the language recognizable by the user, and translating them to binary language for the kernel.

4.3. Bash

Bash stands for Bourne Again SHell, and is a type of shell found in Linux, which is the default shell in several versions ('distributions') of Linux. Other common types of shells are cshell and kshell, though there are others. The most primitive type of shell in Linux is sh.

4.4. Terminal

The terminal is the GUI window that you see on the screen. It takes commands and shows the output.

4.5. Revision

Quickly revise basic Linux commands, which were used in the previous Labs.

- Present working directory
- Change Directory
- Listing files and folders
- Removing a file from the directory
- Move or rename a file
- Showing the content of a file on the Terminal
- Copy a file from one location to another
- Creating a new directory
- Removing/deleting a directory

5. Walkthrough Tasks

5.1. Create Your First Script

It is a lot simpler than you might think to make a bash script.

Go to the terminal in Ubuntu or use the [online linux terminal](#).

Make a file called first_script, using the touch command. i.e. `$ touch first_script` inside a folder named "yourname" on Desktop.

Edit the file with the program of your choice by typing the following line of script into it

```
#!/bin/bash
```

```
pwd
```

```
ls
```

```
echo "Hello, world!"
```

Now from the command line, run the script using the bash interpreter:

```
$ bash first_script
```

You'll see the script has run successfully from the output.

Output: you will see your current working directory, files available in the current directory and the Hello, World message as an output.

That's it, you've created your first script!

5.2. Executable Scripts

So far, we've learned how to run a script from the command line prefixed with the bash interpreter. However, if you want to run the script by name alone, it won't work. Try to run the file simply by typing the name of the file and pressing enter. Note that we're prefixing the file with `./`, which means a file in the current directory.

```
$ ./first_script
```

Output: bash: ./hello-world: Permission denied

To run a file directly, we'll need to change the permissions to allow the script to be executable for the user. **chmod** is a command that changes permissions on a file, and `+x` will add execute rights to the script.

```
$ chmod +x first_script
```

There are various other shell interpreters available, such as Korn shell, C shell and more. For this reason, it is a good practice to define the shell interpreter to be used explicitly to interpret the script's content.

To define your script's interpreter as Bash, first locate a full path to its executable binary using which command, prefix it with a shebang `#!` and insert it as the first line of your script.

All our scripts will include shell interpreter definition `#!/bin/bash`.

Now you can run `first_script` directly.

```
$ ./first_script
```

Output: Hello, world!

5.3. Use of comment

A comment is a human-readable explanation that is written in the shell script.

`#` symbol is used to add a single-line comment in the bash script. For example

```
#!/bin/bash
```

```
#This script prints the Hello World String
```

```
echo Hello World
```

`:'` and `“ ”` symbols are used to add a multi-line comment in the bash script.

```
#!/bin/bash
```

```
:'
```

```
This script prints the
```

```
Hello World String
```

```
,
```

```
echo Hello World
```

5.4. Use of echo command

The “echo” command is used to create a bash script that displays some text on the terminal. Make a file named “myscript.sh” and write the following commands that use echo commands.

```
#!/bin/bash
```

```
echo "Printing text with a newline (by default)"
```

```
echo -n "Printing text without newline"
```

```
echo -e "\nenabling \t interpretation of backslash \t escapes\n"
```

Now run this file using the bash command or giving executable permissions to this file.

The command “echo” can be used with various options. When “echo” is used without any option the new line is added by default. “-n” option is used to print any text without a new line. Another option “-e” is used to enable interpretation of backslash escapes.

For detailed use of the echo command you can also see the manuals of the previous Linux basic commands labs.

5.5. Variables

Variables can be used in bash scripting to save some data or value. We save our data or value in variables and use them when we need them. We can assign any value to a variable. For example, if we want to assign any name i.e. “Ali” to a variable called “name” then write `name="Ali"` in your script file(**Note:** Note that `name = "Ali"` with a space between the assignment is not valid. There must not be a space between variable and value). To use this variable write another command `echo "My name is $name"`. have you noticed the “\$” symbol? This is used before the variable name to access the value of the variable. Similarly, you can use curly brackets around a variable name like `“My name is ${name}”` to get a variable name. There is no difference if you don’t use curly brackets. The purpose of using these brackets is to make variables more visible.

5.6. Use of Read command

Read command is used to get input from the user from the terminal. In the bash script, we can use this command to get input from the user and use it for further use. For example, if we want to get the name from the user as an input and use it in the script then we will use the read command and will save input in any variable. This variable will be used for further use of input. Copy the following commands in the your script file and run this script.

```
#!/bin/bash
```

```
echo What is your name?
```

```
read name
```

```
echo "Hi $name!"
```

```
echo "Hello ${name}"
```

5.7. Strings

- i. Strings can be directly used and don't require double quotes like `echo this is a string`
- ii. With the `-e` flag, bash will interpret strings with backslash-escaped characters, such as `\n` for newline. This also requires a quoted string.
`echo -e "This string has a \nnew line"`
- iii. A single or double quote will expect a closing match, so in order to use one in such a string, you would need to escape the quote using a backslash.

```
echo I\'m a string echo
```

```
I\'m a string
```

- iv. If you want to use a single or double quote in a string without escaping characters, you can do so by wrapping your string in quotes.
`echo 'A single quoted "Hello World"' echo "A double quoted 'Hello World'"`
- v. Double quoted strings are required for variables.
`name=Shafi Ullah Khan`
`echo "I am $name"`
- vi. Within a single-quoted string, the dollar sign would be interpreted literally
`echo 'I am $name'`
- vii. If you would like to use the output of a shell execution within a string, you can do so with a dollar sign followed by parentheses. `$()`. For example, the `pwd` command will print out the present working directory. To use it within a string, wrap `pwd` in the shell execution syntax.
`echo "present working directory is, $(pwd)!"`

5.8. Number comparison and String comparison operators

For comparison of strings, you can use operators, such as `">"` for greater than.

For comparison of numbers, you can use operators, such as `"-gt"` for greater than.

Some comparison operators are listed below and their use is also provided on LMS by a URL of a web page.

Number Comparison	String Comparison	Description
<code>-eq</code>	<code>==</code>	Equal
<code>-ne</code>	<code>!=</code>	Not equal
<code>-gt</code>	<code>?</code>	Greater than
<code>-ge</code>	<code>>=</code>	Greater than or equal
<code>-lt</code>	<code><</code>	Less than
<code>-le</code>	<code><=</code>	Less than or equal

Table 2: Comparison operators

5.9. If else conditions

You can use if condition with single or multiple conditions. Starting and ending block of this statement is defined by `'if'` and `'fi'`. Create the following script to know the use of if statement in bash.

```
#!/bin/bash

echo enter your age

read age

if [ $age -lt 30 ];
then
echo "you are under 30 years age"
else
echo "you are not under 30 years age"
fi
```

Here, age is a variable that is saving the user input. if the value of age is less than 30 then the output will be “You are under 30 years age”, otherwise the output will be “you are not under 30 years age”. For comparison, ‘-lt’ is used here. For comparison, you can also use other operators from “Table 2”.

5.10. While Loop

Three types of loops are used in bash programming. While loop is one of them. Like other loops, a while loop is used to do repetitive tasks.

Syntax:

```
while [ condition ]
do
    commands
done
```

The termination condition is defined at the starting of the loop. The starting and ending block of the while loop is defined by **do** and **done** keywords in the bash script. Write the following script in a file and run it.

```
i=0
while [ $i -lt 5 ]
do
    echo "iteration $i "
    (( i++ ))
done
```

In the above-given script, the loop will iterate 5 times and print the text which is defined inside the loop.

6. Practice Tasks in Lab

Few practice tasks and their solutions are available on LMS under the Lab 10 to give you a better understanding of bash scripts. You can get the idea from that solutions to complete the following practice tasks.

6.1. Task1

Write a script that prints the following line.

“I can’t go to Karachi.”

6.2. Task2

Write a script that makes a directory named “script1” and then makes another directory named “Task No 3” inside “script1” and then makes a file named “test.txt” inside “Task No 3”.

6.3. Task3

Write a script that takes an input from the user and rename the file test.txt under “Task No 3” with the user-entered input.

6.4. Task4

Write a script that takes two numbers and tells which number is greater.

6.5. Task5

Write a program to print the table of any number up to 10.

6.6. Task6

Write a program that prints from 20 to 1. i.e. 20, 19, 18,17.....3,2,1

6.7. Task7

Write a program that takes any number as input and prints the next 5 numbers. For example, input number is 75 then it should print 76, 77, 78,79,80

7. Unseen Task

You will be given unseen tasks you have to perform and submit during the next lab.

8. Assignment

Assignment for this Lab will be uploaded on LMS after the Lab session. The deadline will also be mentioned on LMS.

9. Quiz

A quiz about this lab will be conducted at the end of the bash scripting Labs.

10. Further Readings

The reading material and other helping material regarding this Lab will be uploaded on LMS if any. You can access it from the LMS.

11. Evaluation Criteria

📁 **Method of Evaluation:** Viva, File submitted on LMS.

	Excellent 10	Good 9-7	Satisfactory 6-4	Unsatisfactory 3-1	Poor 0	Marks Obtained
Unseen Tasks	All tasks were completed correctly in the given time.	Most tasks were completed correctly. Tasks could be improved further.	Some tasks were completed correctly.	Most tasks were incomplete or incorrect	All tasks were incomplete or incorrect. Didn't perform tasks	
Assignment	All tasks completed correctly in given time and have a complete understanding	Most tasks were completed correctly. Tasks could be improved further and have a complete understanding	Some tasks were completed correctly and have an incomplete understanding	Most tasks were incomplete or incorrect and have no understanding	All tasks were incomplete or incorrect. Didn't perform tasks and have no understanding	
Total						



SHIFA TAMEER-E-MILLAT UNIVERSITY

Lab Manual **Introduction to Computer Science**

DEPARTMENT OF COMPUTING

Shifa Tameer-e-Millat University

Department of Computing

INTRODUCTION TO COMPUTER SCIENCE

Lab 11

Bash Scripting

Lab 11: Bash Scripting

1. Introduction

A Bash script is a plain text file that contains a series of commands. These commands are a mixture of commands we would normally type ourselves on the command line or terminal (such as ls or cp for example). The basic concept of a bash script is a list of commands, which are listed in the order of execution. Think of a script for a play, a movie, or a TV show. The script tells the actors what they should say and do. A script for a computer tells the computer what it should do or say. In the context of Bash scripts, we are telling the Bash shell what it should do.

2. Activity Timeboxing

Heading No	Activity Name	Activity Time	Total Time
	About this Lab	20 minutes	20 minutes
5	Walkthrough Task	40 minutes	60 minutes
6	Practice Task in Lab	80 minutes	140 minutes
	Q/A session	10 minutes	150 minutes

Table 1: Activity Timeboxing

3. Objective

- To create a bash script
- To learn about strings, variables, shell execution.
- To learn about user input, comparison, conditions, loops

4. Concept Map

In this lab, we learn about bash scripting, what is bash scripting, few basic scripts. First of all, try to learn about some important terms that we should know while writing bash scripts.

4.1. Kernel

Every operating system (OS) has a kernel. It is the layer of the Operating system that bridges the hardware with the main programs that run on a computer. The kernel is the core of the operating system and is the first to load when the computer boots up. It remains in the computer's memory throughout a session. It is responsible for providing an interface for all applications, controlling the hardware and allowing processes to get information from each other.

4.2. Shell

The shell forms the layer between the user and the kernel so the user can enter commands. The kernel 'understands' only binary language, which is composed exclusively of ones and zeros. In early computing, any instructions/commands from the users were entered in binary language, but this evolved so that the user can enter commands in a more recognizable language. It is the shell that acts as the go-between, accepting the commands entered in the language recognizable by the user, and translating them to binary language for the kernel.

4.3. Bash

Bash stands for Bourne Again SHell, and is a type of shell found in Linux, which is the default shell in several versions ('distributions') of Linux. Other common types of shells are `cshell` and `kshell`, though there are others. The most primitive type of shell in Linux is `sh`.

4.4. Terminal

The terminal is the GUI window that you see on the screen. It takes commands and shows the output.

5. Walkthrough Tasks

5.1. Create Your First Script

It is a lot simpler than you might think to make a bash script.

Go to the terminal in Ubuntu or use the [online linux terminal](#).

Make a file called `first_script`, using the `touch` command. i.e. `$ touch first_script` inside a folder named "yourname" on Desktop.

Edit the file with the program of your choice by typing the following line of script into it

```
#!/bin/bash
```

```
pwd
```

```
ls
```

```
echo "Hello, world!"
```

Now from the command line, run the script using the bash interpreter:

```
$ bash first_script
```

You'll see the script has run successfully from the output.

Output: you will see your current working directory, files available in the current directory and the Hello, World message as an output.

That's it, you've created your first script!

5.2. Executable Scripts

So far, we've learned how to run a script from the command line prefixed with the bash interpreter. However, if you want to run the script by name alone, it won't work. Try to run the file simply by typing the name of the file and pressing enter. Note that we're prefixing the file with `./`, which means a file in the current directory.

```
$ ./first_script
```

Output: bash: ./hello-world: Permission denied

To run a file directly, we'll need to change the permissions to allow the script to be executable for the user. **chmod** is a command that changes permissions on a file, and `+x` will add execute rights to the script.

```
$ chmod +x first_script
```

There are various other shell interpreters available, such as Korn shell, C shell and more. For this reason, it is a good practice to define the shell interpreter to be used explicitly to interpret the script's content.

To define your script's interpreter as Bash, first locate a full path to its executable binary using which command, prefix it with a shebang `#!/` and insert it as the first line of your script.

All our scripts will include shell interpreter definition `#!/bin/bash`.

Now you can run `first_script` directly.

```
$ ./first_script
```

Output: Hello, world!

5.3. else if statement:

The use of else if condition is a little different in bash than other programming languages. 'elif' is used to define else if condition in bash. Else if is used for multiple conditions. For example:

```
#!/bin/bash
echo "Enter your marks in percentage"
read n
if [ $n -gt 60 ];
then
echo "Grade A"
elif [ $n -gt 45 ];
then
echo "Grade B"
elif [ $n -ge 40 ];
then
echo "Grade C"
else
echo "Grade F"
fi
```

5.4. if statement with AND logic

Different types of logical conditions can be used in if statement with two or more conditions. How you can define multiple conditions in if statement using **AND** logic is shown in the following example. '&&' is used to apply **AND** logic of if statement. Save the following script in a file and run

```
#!/bin/bash
echo "Enter username"
read username
echo "Enter password"
read password
if [[ $username == "admin" && $password == "12345" ]]; then
echo "valid user"
else
```

```
echo "invalid user"
fi
```

Here, the value of username and password variables will be taken from the user and compared with username '**admin**' and password '**12345**'. If both values match then the output will be "**valid user**", otherwise the output will be "**invalid user**".

5.5. if statement with OR logic

symbol '**||**' is used to define **OR** logic in if condition. Save the following script in a file and execute it.

```
#!/bin/bash
echo "Enter any number"
read n
if [[ ( $n -eq 15 || $n -eq 45 ) ]] then
echo "Great. You guessed the number"
else
echo "Sorry...! Try Again"
fi
```

Here, the value of **n** will be taken from the user. If the value is equal to **15** or **45** then the output will be "**Great. You guessed the number**", otherwise the output will be "**Sorry...! Try Again**".

5.6. For Loop

Loops are used in any programming language to execute the same code repeatedly. For loop is also used to do some repeatedly in Bash scripting.

Syntax:

```
for variable_name in lists
do
commands
done
```

How many times a for loop will iterate depends on the declared lists variable. The loop will take one item from the lists and store the value on a variable which can be used within the loop. The starting and ending block of for loop are defined by do and done keywords in bash script. Let's understand from an example.

```
for i in 1 2 3 6 7
do
echo "list item = $i"
done
```

In this example, 5 static values are declared in the lists part. This loop will iterate 5 times and each time It will receive a value from the lists and store it in the variable named "**i**" which will print inside the loop.

Look another example:

```
#!/bin/bash
for (( counter=1; counter<=10; counter++ ))
do
echo "$counter "
done
```

in this example, the loop will iterate 10 times as the counter is starting from 1 and it is iterating till 10. After every iteration counter is incremented by 1.

Here is another example.

```
for counter in $(seq 10)
do
echo "$counter "
done
```

5.7. Arrays

An array is defined inside parenthesis. There are **no commas** between the items of the array. Items are separated with space. For example

```
arr=(22 24 10 45 98)
```

Arrays are indexed from “0”. 22 is placed on the “0” index, 24 is on index “1” and so on. To print an element on the specific index for example 2nd index, write the following line in the script.

```
echo ${arr[2]}
```

it will print “10” as output.

To print all elements of an Array using @ or * instead of the specific index number.

```
echo ${arr[@]}
```

```
echo ${arr[*]}
```

To append a new element in an array “+=” is used. For example to add 101 at the end of the array “**arr**” use the following command.

```
arr+=(99)
```

To update an element on specific index for example 3rd index, write the following line in the script.

```
arr[3]=88
```

This will update 3rd index. “45” will be replaced with 88.

To delete an element on specific index for example 1st index, write the following line in the script.

```
unset arr[1]
```

To print or check the indexes of the array, write the following line in the script

```
echo ${!arr[*]}
```

To check the size of an array, write the following line in the script

```
echo ${#arr[*]}
```

To delete the whole array, write the following line in the script.

```
unset arr
```

You can use for loop to iterate the values of an array.

```
ColorList=("Blue Green Pink White Red")
for color in $ColorList
do
echo "My favorite color is $color"
done
```

You can also use while loop.

```
i=0
while [ $i -lt ${#ColorList[@]} ]
do
echo My favorite color is ${ColorList[i]}
(( i++ ))
done
```

6. Practice Tasks in Lab

Few practice tasks and their solutions are available on LMS under the Lab 10 to give you a better understanding of bash scripts. You can get the idea from that solutions to complete the following practice tasks.

6.1. Task1

Write a script that takes two numbers and tells which number is greater or both numbers are equal.

6.2. Task2

Write a script that takes any number and tells that number is even or odd.

6.3. Task3

Write a script that prints those numbers that are divisible by 3 and 5. (Less than 100)

6.4. Task4

Write a script that prints those numbers that are divisible by 3 or 5. (Less than 100).

6.5. Task5

Write a program to print the table of any number up to 10 using for loop.

6.6. Task6

Write a Script that takes your percentage marks and tells:

- A+ if percentage > 80
- A if percentage > 60
- B if percentage > 50
- C if percentage > 40
- D if percentage > 33
- F if percentage < 33

6.7. Task7

Write a bash script that helps to find factorial of any number using For Loop (Number should be taken as input from the user). For example, if the user enters 4. Then the answer should be 24 as $4! = 4 \times 3 \times 2 \times 1 = 24$. You have to ask and show the output as follows:

```
shafi@shafi-PC:~/Desktop$ bash Task1
Enter a Number
4
factorial of 4 is = 24
shafi@shafi-PC:~/Desktop$ bash Task1
Enter a Number
0
factorial of 0 is = 1
shafi@shafi-PC:~/Desktop$
```

7. Unseen Task

You will be given unseen tasks you have to perform and submit during the next lab.

8. Assignment

Assignment for this Lab will be uploaded on LMS after the Lab session. The deadline will also be mentioned on LMS.

9. Quiz

A quiz about this lab will be conducted at the end of the bash scripting Labs.

10. Further Readings

The reading material and other helping material regarding this Lab will be uploaded on LMS if any. You can access it from the LMS.

11. Evaluation Criteria

📁 **Method of Evaluation:** Viva, File submitted on LMS.

	Excellent 10	Good 9-7	Satisfactory 6-4	Unsatisfactory 3-1	Poor 0	Marks Obtained
Unseen Tasks	All tasks were completed correctly in the given time.	Most tasks were completed correctly. Tasks could be improved further.	Some tasks were completed correctly.	Most tasks were incomplete or incorrect	All tasks were incomplete or incorrect. Didn't perform tasks	
Assignment	All tasks completed correctly in given time and have a complete understanding	Most tasks were completed correctly. Tasks could be improved further and have a complete understanding	Some tasks were completed correctly and have an incomplete understanding	Most tasks were incomplete or incorrect and have no understanding	All tasks were incomplete or incorrect. Didn't perform tasks and have no understanding	
Total						



SHIFA TAMEER-E-MILLAT UNIVERSITY

Lab Manual **Introduction to Computer Science**

DEPARTMENT OF COMPUTING

Shifa Tameer-e-Millat University

Department of Computing

INTRODUCTION TO COMPUTER SCIENCE

Lab 12

Bash Scripting

Lab 12: Bash Scripting

1. Introduction

A Bash script is a plain text file that contains a series of commands. These commands are a mixture of commands we would normally type ourselves on the command line or terminal (such as `ls` or `cp` for example). The basic concept of a bash script is a list of commands, which are listed in the order of execution. Think of a script for a play, a movie, or a TV show. The script tells the actors what they should say and do. A script for a computer tells the computer what it should do or say. In the context of Bash scripts, we are telling the Bash shell what it should do.

2. Activity Timeboxing

Heading No	Activity Name	Activity Time	Total Time
	About this Lab	20 minutes	20 minutes
5	Walkthrough Task	40 minutes	60 minutes
6	Practice Task in Lab	80 minutes	140 minutes
	Q/A session	10 minutes	150 minutes

Table 1: Activity Timeboxing

3. Objective

- To create a bash script
- To learn about strings, variables, shell execution.
- To learn about user input, comparison, conditions, loops

4. Concept Map

In this lab, we learn about bash scripting, what is bash scripting, few basic scripts. First of all, try to learn about some important terms that we should know while writing bash scripts.

4.1. Kernel

Every operating system (OS) has a kernel. It is the layer of the Operating system that bridges the hardware with the main programs that run on a computer. The kernel is the core of the operating system and is the first to load when the computer boots up. It remains in the computer's memory throughout a session. It is responsible for providing an interface for all applications, controlling the hardware and allowing processes to get information from each other.

4.2. Shell

The shell forms the layer between the user and the kernel so the user can enter commands. The kernel 'understands' only binary language, which is composed exclusively of ones and zeros. In early computing, any instructions/commands from the users were entered in binary language, but this evolved so that the user can enter commands in a more recognizable language. It is the shell that acts as the go-between, accepting the commands entered in the language recognizable by the user, and translating them to binary language for the kernel.

4.3. Bash

Bash stands for Bourne Again SHell, and is a type of shell found in Linux, which is the default shell in several versions ('distributions') of Linux. Other common types of shells are cshell and kshell, though there are others. The most primitive type of shell in Linux is sh.

4.4. Terminal

The terminal is the GUI window that you see on the screen. It takes commands and shows the output.

5. Walkthrough Tasks

5.1. Create Your First Script

It is a lot simpler than you might think to make a bash script.

Go to the terminal in Ubuntu or use the [online linux terminal](#).

Make a file called first_script, using the touch command. i.e. `$ touch first_script` inside a folder named "yourname" on Desktop.

Edit the file with the program of your choice by typing the following line of script into it

```
#!/bin/bash
```

```
pwd
```

```
ls
```

```
echo "Hello, world!"
```

Now from the command line, run the script using the bash interpreter:

```
$ bash first_script
```

You'll see the script has run successfully from the output.

Output: you will see your current working directory, files available in the current directory and the Hello, World message as an output.

That's it, you've created your first script!

5.2. Executable Scripts

So far, we've learned how to run a script from the command line prefixed with the bash interpreter. However, if you want to run the script by name alone, it won't work. Try to run the file simply by typing the name of the file and pressing enter. Note that we're prefixing the file with `./`, which means a file in the current directory.

```
$ ./first_script
```

Output: bash: ./hello-world: Permission denied

To run a file directly, we'll need to change the permissions to allow the script to be executable for the user. **chmod** is a command that changes permissions on a file, and `+x` will add execute rights to the script.

```
$ chmod +x first_script
```

There are various other shell interpreters available, such as Korn shell, C shell and more. For this reason, it is a good practice to define the shell interpreter to be used explicitly to interpret the script's content.

To define your script's interpreter as Bash, first locate a full path to its executable binary using which command, prefix it with a shebang `#!/` and insert it as the first line of your script.

All our scripts will include shell interpreter definition `#!/bin/bash`.

Now you can run `first_script` directly.

```
$ ./first_script
```

Output: Hello, world!

5.3. Get Arguments from Command Line

Bash script can read input from command line arguments like other programming languages. Command Line Arguments are specified with the shell script on the terminal during the run time. Each variable passed to a shell script at the command line is stored in corresponding shell variables including the shell script name.

Syntax:

```
./myscript.sh ARG1 ARG2 ARG3 ARG4 ARG5 ARG6 ARG7 ARG8 ARG9 ARG10
```

See the below image to understand the command line values and variables. Here ARG1, ARG2 to ARG10 are command-line values, which are assigned to corresponding shell variables.



For example, copy the following script and save and run it. i.e.

```
bash myscript.sh 200 500
```

In the following script, `$1` and `$2` variable are used to read first and second command line arguments.

```
#!/bin/bash
```

```
echo "All arguments : $@"
```

```
echo "Total arguments : $#"
```

```
echo "First Argument = $1"
echo "Second argument = $2"
```

5.4. Function

Function is created using function keyword. Function named as “HelloFunction” is created below.

```
#!/bin/bash

function HelloFunction () {
    echo 'Hello World'
}

HelloFunction
```

You can call any function by name only without using any bracket in bash script like in above example function is called by its name “HelloFunction” in the last line of the script.

You can also create a function without function keyword like:

```
#!/bin/bash

HelloFunction () {
    echo 'Hello World'
}

HelloFunction
```

5.5. Create a function with Parameters

Bash can’t declare function parameters or arguments at the time of function declaration. But you can use parameters in function by using other variables. If two values are passed at the time of function call then \$1 and \$2 variables are used for reading the values. For example, an “Add” function is created below that will add two values.

```
#!/bin/bash

Add() {
    sum=$(( $1 + $2 ))
    echo "Sum is : $sum"
}

Add 11 22
```


6. Practice Tasks in Lab

6.1. Task1

Write a general paragraph about yourself such that when you provide a name, city, university name then it becomes the paragraph about that person. (Provide name, city etc as command line argument)

6.2. Task2

Write a function named “printEven” that prints the first 10 even numbers.

6.3. Task3

Write a function named “printOdd” that prints the first 10 odd numbers.

6.4. Task4

Write two functions **printEven** and **printOdd** (task 2 and task 3) and ask the user if he wants to print even numbers then enter 1. To print, the odd numbers enter 2 otherwise show wrong input.

6.5. Task5

Write a script that has a function named “subtract” that receives two numbers from the user and output their subtraction.

6.6. Task6

Write a script that has a function named “multiply” that receives two numbers from the user and output their multiplication.

7. Assignment

Assignment for this Lab will be uploaded on LMS after the Lab session. The deadline will also be mentioned on LMS.

8. Quiz

A quiz about this lab will be conducted at the end of the bash scripting Labs.

9. Further Readings

The reading material and other helping material regarding this Lab will be uploaded on LMS if any. You can access it from the LMS.

10. Evaluation Criteria

📁 **Method of Evaluation:** Viva, File submitted on LMS.

	Excellent 10	Good 9-7	Satisfactory 6-4	Unsatisfactory 3-1	Poor 0	Marks Obtained
Assignment	All tasks completed correctly in given time and have a complete understanding	Most tasks were completed correctly. Tasks could be improved further and have a complete understanding	Some tasks were completed correctly and have an incomplete understanding	Most tasks were incomplete or incorrect and have no understanding	All tasks were incomplete or incorrect. Didn't perform tasks and have no understanding	
Total						



SHIFA TAMEER-E-MILLAT UNIVERSITY

Lab Manual **Introduction to Computer Science**

DEPARTMENT OF COMPUTING

Shifa Tameer-e-Millat University

Department of Computing

INTRODUCTION TO COMPUTER SCIENCE

Lab 13

LaTeX

Lab 13: LaTeX

1. Introduction

Latex is a software system for document preparation. LaTeX is widely used in academia for the communication and publication of scientific documents in many fields, including mathematics, statistics, computer science, engineering, physics, economics, linguistics, quantitative psychology, philosophy, and political science.

Overleaf is a great on-line LaTeX edition tool that allows you to create LaTeX documents directly in your web browser. This lab explains how to create a new project in Overleaf, either starting from scratch or using one of the many templates available.

2. Activity Timeboxing

Heading No	Activity Name	Activity Time	Total Time
	Quiz from previous Lab	35 minutes	35 minutes
	About this Lab	10 minutes	45 minutes
6	Walkthrough Task	45 minutes	90 minutes
7	Practice Task in Lab	60 minutes	150 minutes

Table 1: Activity Timeboxing

3. Objective

- To create an account over overleaf
- To create a LaTeX document
- To learn about document structure and formating
- To learn about the basic commands of LaTeX.

4. Concept Map

In this lab, we learn about Latex. LaTeX is a tool used to create professional-looking documents. It is based on the WYSIWYM (what you see is what you mean) idea, meaning you only have to focus on the contents of your document and the computer will take care of the formatting. Instead of spacing out text on a page to control formatting, as with Microsoft Word or LibreOffice Writer, users can enter plain text and let LATEX take care of the rest.

4.1. Overleaf

Overleaf is a great on-line LaTeX edition tool that allows you to create LaTeX documents directly in your web browser. To start using Overleaf, go to www.overleaf.com. If you don't have an account enter your e-mail address and set a password in the corresponding boxes. Get started now, click Register and that's it, you will be redirected to the project management page where you will be guided into how to create a new project. If you already have an account, click Login in the upper right corner, then type in your email and password and click the Login button.

Once you are logged in, you will see the Overleaf Project Management page.

4.2. Starting a new project

To start a new project from scratch, on the main page click the New Project button, you will see the next drop-down menu then click Blank Project. A box will open where you should enter the name of your new project, then click Create. After that, you will be redirected to the editor. There, a new document is created with some basic information already filled in. You can start editing your .tex file now, to view the changes click Recompile or

5. Walkthrough Tasks

5.1. First document

5.1.1 Creating a Blank Project

In the Project Management page click New project => Blank Project, enter a name for your project then click Create.

Detailed Explanation to Create a Blank Project: To start a new project from scratch, on the main page click the New Project button, you will see the next drop-down menu then click Blank Project as shown in **Figure 1**.

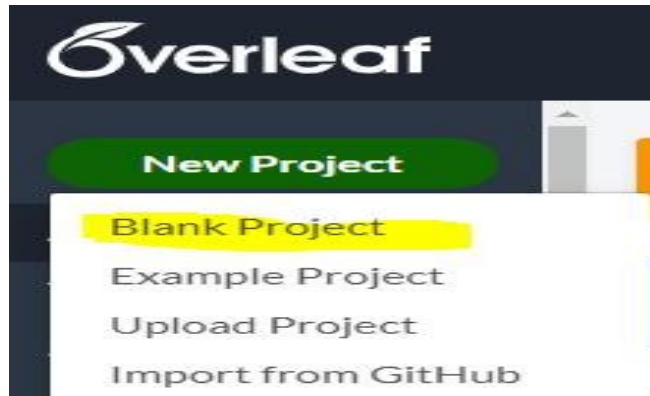


Figure 1

A box will open where you should enter the name of your new project, then click **Create** as shown in **Figure 2**.

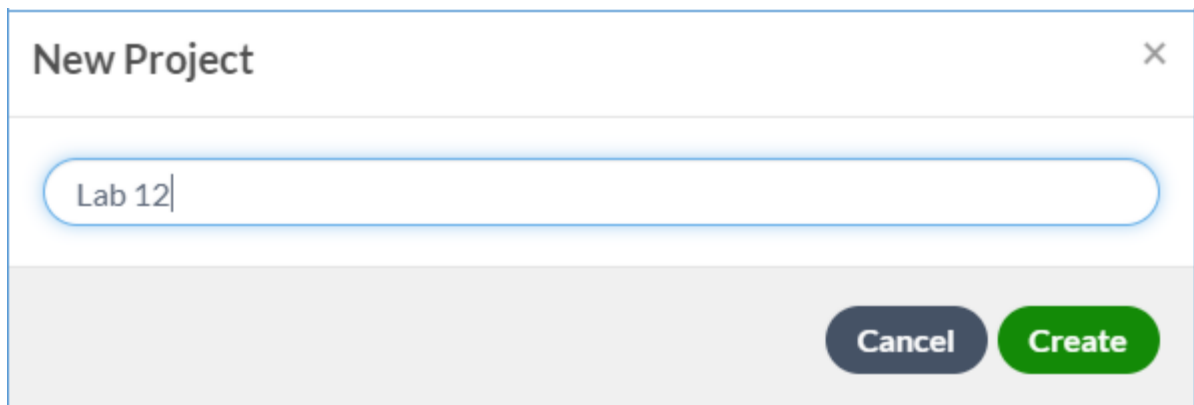
A screenshot of a 'New Project' dialog box. The title bar says 'New Project' with a close button (X) on the right. Below the title bar is a text input field containing the text 'Lab 12'. At the bottom right of the dialog box are two buttons: 'Cancel' and 'Create'. The 'Create' button is highlighted with a green background.

Figure 2

After that, you will be redirected to the editor. There, a new document is created with some basic information already filled in. You can start editing your .tex file now, to view the changes click Recompile **Figure 3**.

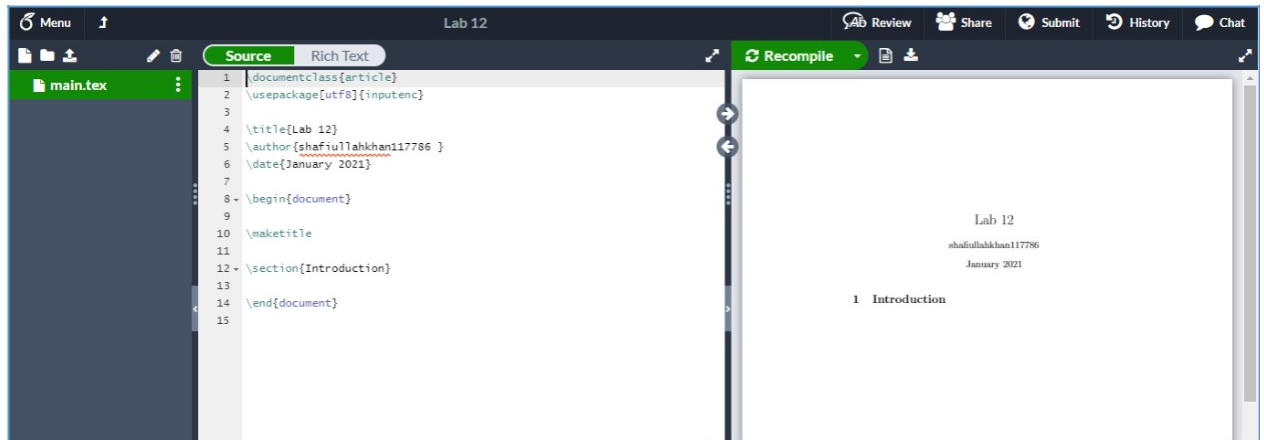


Figure 3

5.1.2 Creating a project from a template

Using a template to start a new project will save you a lot of time. Overleaf provides a vast set of templates.

To create a document from a template, in the Project Management page click New Project, a drop-down menu will display, below Templates click on the document type you are about to write. **Figure 4**. In our case, we are going to use the Resume/CV template **Figure 4**.

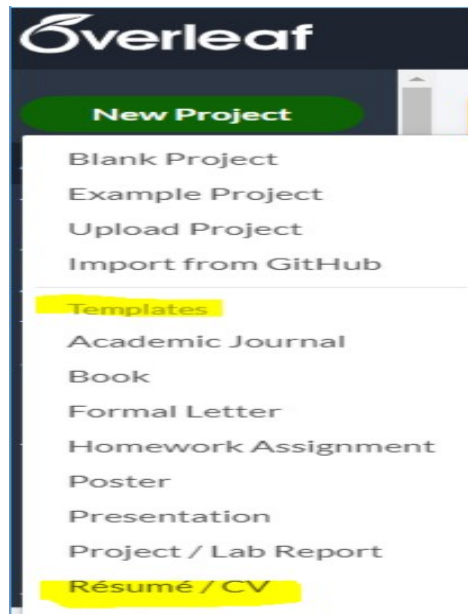


Figure 4

You will be redirected to a list of templates where you can select one that is suitable **Figure 5**.

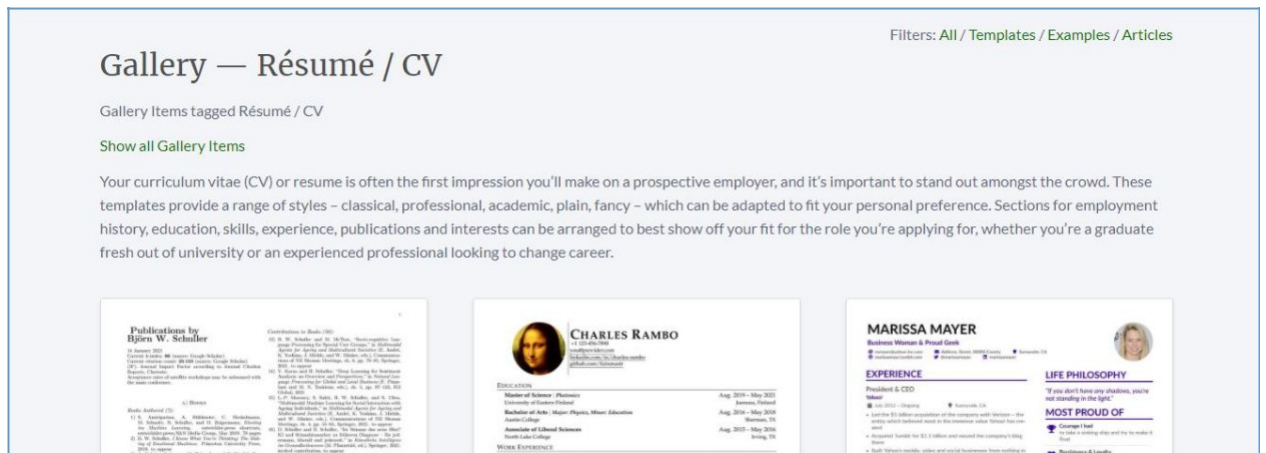


Figure 5

Clicking on the template thumbnail will display some further information about the template, and you can start editing your copy of the template on Overleaf right away by clicking 'Open as Template' **Figure 6**.

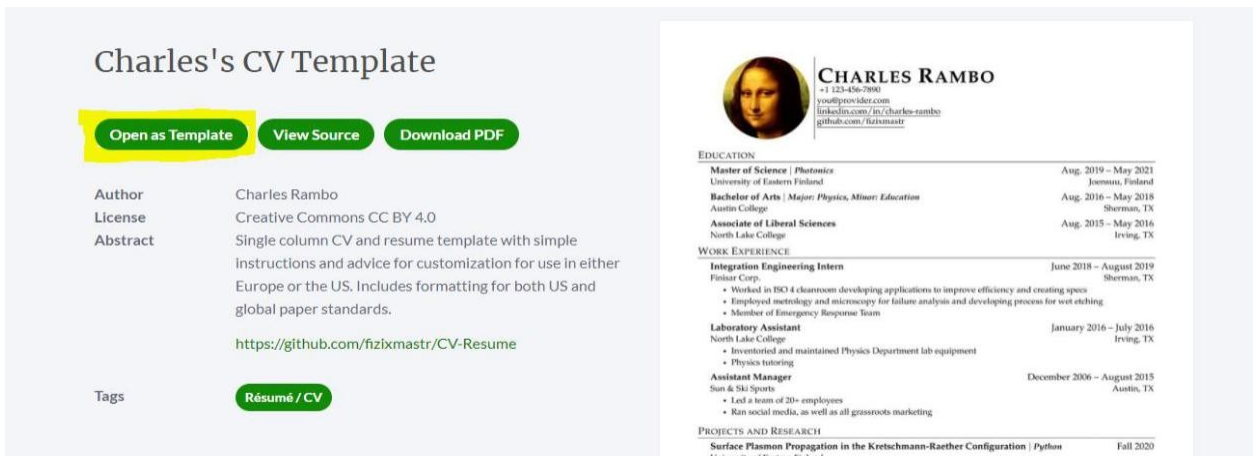


Figure 6

You can start editing the .tex file of your template now, to view the changes click Recompile **Figure 7**.

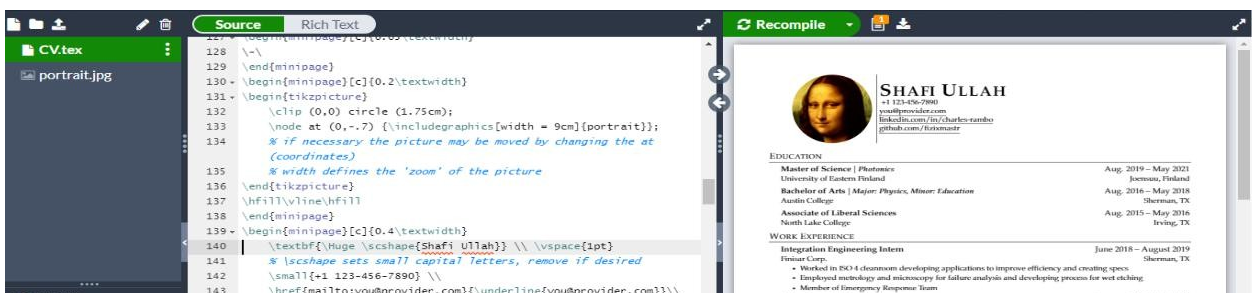


Figure 7

5.2. The preamble of a document

In the previous example, while creating a blank project, the text was entered after the `\begin{document}` command. The part of your .tex file before this point is called the preamble. In the preamble, you define the type of document you are writing and the language, load extra packages you will need, and set several parameters. For instance, a normal document preamble would look like this:

```
\documentclass[12pt, letterpaper]{article} \usepackage[utf8]
{inputenc}
```

```
\title{First document}
```

```
\author{Hubert Farnsworth}
```

```
\date{February 2014}
```

You can learn more about preamble of document from [here](#).

5.3. Displaying the title of your document

To display the title of your document you have to declare its components in the preamble and then use some additional code:

```
\documentclass[12pt, letterpaper]{article} \usepackage[utf8]
{inputenc}
```

```
\title{First document}
```

```
\author{Hubert Farnsworth }
```

```
\date{February 2014}
```

```
\begin{document}
```

```
\begin{titlepage}
```

```
\maketitle
```

```
\end{titlepage}
```

```
Text after the title. Text after the title. Text after the title.
```

```
Text after the title. Text after the title.
```

```
\end{document}
```

There is a block with highlighted three lines in the preamble that defines the information to be included on the title page.

`\title{First document}`

This is the title.

`\author{Hubert Farnsworth}`

Here you put the name(s) of the author(s) and, as an optional parameter, you can add the next command:

`\date{February 2014}`

You can enter the date manually or use the command `\today` so the date will be updated automatically at the time you compile your document.

Once you have that in the preamble now in the body of your document you can use the next commands for the information to be printed.

`\begin{titlepage} \end{titlepage}`

This declares an environment, a block of code with a specific behaviour depending on its type. In this case whatever you include in this titlepage environment will appear in the first page of your document.

`\maketitle`

This command will print the title, the author and the date in the format shown in the example. If it's not enclosed in a titlepage environment, it will be shown at the beginning of the document, above the first line.

5.4. Comments

Sometimes it's necessary to add comments to your LATEX code for readability. This is straightforward, put a % before the comment and LATEX will ignore that text.

```
\documentclass{article}
```

```
\usepackage[utf8]{inputenc} %codification of the document
```

```
\usepackage{comment}
```

```
%Here begins the body of the document \
```

```
begin{document}
```

This document contains a lot of comments, none of them will appear here, only this text.

```
\begin{comment}
```

This text won't show up in the compiled pdf this is just a multi-line comment. Useful

to, for instance, comment out slow-rendering parts while working on a draft. \end{comment}

```
\end{document}
```

In the last part of the example, you can see a comment environment. This helps in multi-line comments instead of putting a % at the beginning of each line. For this to work you must add the following line to your preamble:

```
\usepackage{comment}
```

The % symbol is used for single-line comments. It is a reserved character, if you need this symbol to be printed in your document, use \%.

5.5. Quotation marks

Quotation marks are a bit tricky, use a backtick ` on the left and an apostrophe ' on the right.

Single quotes: `text'.

Double quotes: ``text''.

Single quotes: ‘text’.

Double quotes: “text”.

5.6. Paragraphs and new lines

To start a new paragraph in LATEX, you must leave a blank line in between. There's another way to start a new paragraph, the `\par` command also starts a new paragraph. Look at the following code snippet.

This is the text in first paragraph. This is the text in first paragraph. This is the text in first paragraph. \par

This is the text in second paragraph. This is the text in second paragraph. This is the text in second paragraph.

To start a new line without actually starting a new paragraph insert a break line point, this can be done by `\\` (a double backslash as in the example) or the `\newline` command. Look at the following code snippet.

This is the `\\` text in the paragraph. `\\` This is the text in `\newline` paragraph. This is the text in the paragraph.

You can learn more about paragraphs and new lines from [here](#).

5.7. Paragraph Alignment

Paragraphs in LaTeX are fully justified, i.e. flush with both the left and right margins. If you would like to change the justification of a paragraph, LATEX has the following three environments: **center**, **flushleft** and **flushright**

```
\begin{flushleft}
```

` This is the text in paragraph Alignment. This is the text in paragraph Alignment. This is the text in paragraph Alignment. This is the text in paragraph Alignment. This is the text in paragraph Alignment. This is the text in paragraph Alignment. This is the text in paragraph Alignment."

```
\end{flushleft}
```

The flushleft environment left-justifies the paragraph. To right-justify use flushright instead.

You can learn more about paragraphs and new lines from [here](#).

5.8. Bold text

To make a text bold use `\textbf` command:

Some of the `\textbf{greatest}` discoveries in science were made by accident.

5.9. Italic text

To make a text italic is straightforward, use the `\textit` command.

Some of the greatest discoveries in science were made by `\textit{accident}`.

5.10. Underlined text

Underlining text is very simple too, use the `\underline` command:

Some of the greatest discoveries in `\underline{science}` were made by accident.

5.11. Emphasizing text

Text can be emphasized by using `\emph` command. Sometimes the `\emph` command behaves just as `\textit`, but is not the same:

Some of the greatest `\emph{discoveries}` in science were made by accident.

```
\textit{Some of the greatest \emph{discoveries} in science were made by accident.}
```

```
\textbf{Some of the greatest \emph{discoveries} in science were made by accident.}
```

What the `\emph` command does with its argument depends on the context - inside the normal text, the emphasized text is italicized, but this behavior is reversed if used inside an italicized text-see example above

6. Practice Tasks in Lab

In this part, you are provided some practice tasks that will help you to create documents in LaTeX.

6.1. Task 1

Practice all the tasks explained in the Walkthrough task on overleaf.

6.2. Task2

Create your Resume/CV using any template from overleaf.

7. Assignment

Assignment for this Lab will be uploaded on LMS after the Lab session. The deadline will also be mentioned on LMS.

8. Further Readings

The reading material and other helping material regarding this Lab will be uploaded on LMS if any. You can access it from the LMS.

9. Evaluation Criteria

📁 **Method of Evaluation:** Viva, File submitted on LMS.

	Excellent 10	Good 9-7	Satisfactory 6-4	Unsatisfactory 3-1	Poor 0	Marks Obtained
Assignment	All tasks completed correctly in given time and have a complete understanding	Most tasks were completed correctly. Tasks could be improved further and have a complete understanding	Some tasks were completed correctly and have an incomplete understanding	Most tasks were incomplete or incorrect and have no understanding	All tasks were incomplete or incorrect. Didn't perform tasks and have no understanding	
Total						



SHIFA TAMEER-E-MILLAT UNIVERSITY

Lab Manual **Introduction to Computer Science**

DEPARTMENT OF COMPUTING

Shifa Tameer-e-Millat University

Department of Computing

INTRODUCTION TO COMPUTER SCIENCE

Lab 14

LaTeX

Lab 14: LaTeX

1. Introduction

Latex is a software system for document preparation. LaTeX is widely used in academia for the communication and publication of scientific documents in many fields, including mathematics, statistics, computer science, engineering, physics, economics, linguistics, quantitative psychology, philosophy, and political science.

Overleaf is a great on-line LaTeX edition tool that allows you to create LaTeX documents directly in your web browser. This lab explains how to create a new project in Overleaf, either starting from scratch or using one of the many templates available.

2. Activity Timeboxing

Heading No	Activity Name	Activity Time	Total Time
	About this Lab	10 minutes	10 minutes
6	Walkthrough Task	50 minutes	60 minutes
7	Practice Task in Lab	90 minutes	150 minutes

Table 1: Activity Timeboxing

3. Objective

- To create an account over overleaf
- To create a LaTeX document
- To learn about document structure and formating
- To learn about the basic commands of LaTeX.

4. Concept Map

In this lab, we learn about Latex. LaTeX is a tool used to create professional-looking documents. It is based on the WYSIWYM (what you see is what you mean) idea, meaning you only have to focus on the contents of your document and the computer will take care of the formatting. Instead of spacing out text on a page to control formatting, as with Microsoft Word or LibreOffice Writer, users can enter plain text and let LATEX take care of the rest.

4.1. Overleaf

Overleaf is a great on-line LaTeX edition tool that allows you to create LaTeX documents directly in your web browser. To start using Overleaf, go to www.overleaf.com. If you don't have an account enter your e-mail address and set a password in the corresponding boxes. Get started now, click Register and that's it, you will be redirected to the project management page where you will be guided into how to create a new project. If you already have an account, click Login in the upper right corner, then type in your email and password and click the Login button.

Once you are logged in, you will see the Overleaf Project Management page.

4.2. Starting a new project

To start a new project from scratch, on the main page click the New Project button, you will see the next drop-down menu then click Blank Project. A box will open where you should enter the name of your new project, then click Create. After that, you will be redirected to the editor. There, a new document is created with some basic information already filled in. You can start editing your .tex file now, to view the changes click Recompile or

5. Walkthrough Tasks

5.1. First document

5.1.1 Creating a Blank Project

In the Project Management page click New project => Blank Project, enter a name for your project then click Create.

Detailed Explanation to Create a Blank Project: To start a new project from scratch, on the main page click the New Project button, you will see the next drop-down menu then click Blank Project as shown in **Figure 1**.

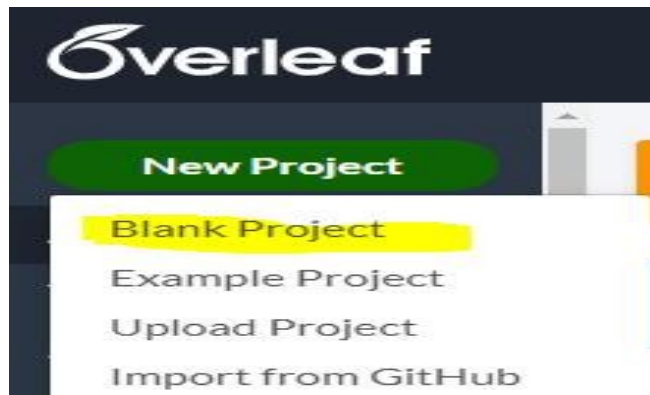


Figure 1

A box will open where you should enter the name of your new project, then click **Create** as shown in **Figure 2**.

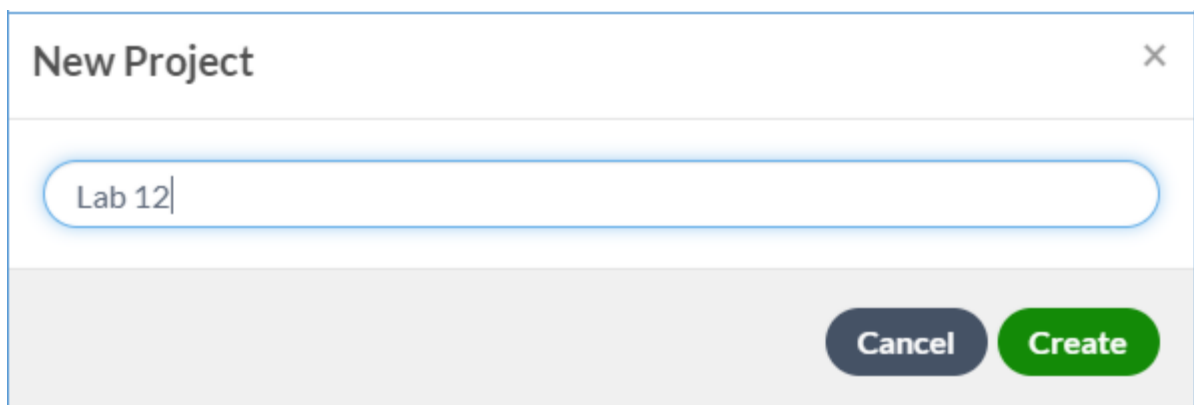
A screenshot of a 'New Project' dialog box. The title bar says 'New Project' with a close button (X) on the right. Below the title bar is a text input field containing the text 'Lab 12'. At the bottom right of the dialog box are two buttons: 'Cancel' and 'Create'. The 'Create' button is highlighted in green.

Figure 2

After that, you will be redirected to the editor. There, a new document is created with some basic information already filled in. You can start editing your .tex file now, to view the changes click Recompile **Figure 3**.

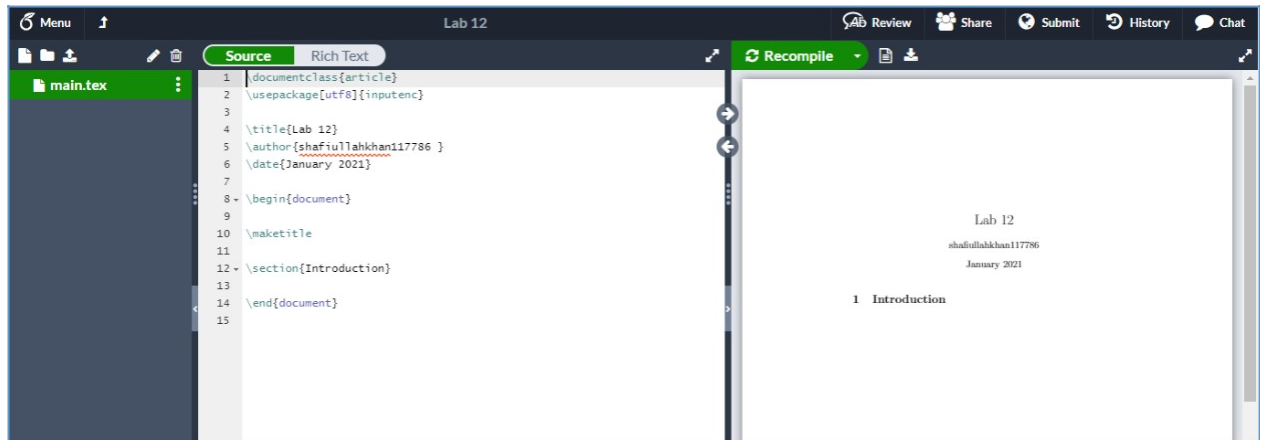


Figure 3

5.1.2 Creating a project from a template

Using a template to start a new project will save you a lot of time. Overleaf provides a vast set of templates.

To create a document from a template, in the Project Management page click New Project, a drop-down menu will display, below Templates click on the document type you are about to write. **Figure 4**. In our case, we are going to use the Resume/CV template **Figure 4**.

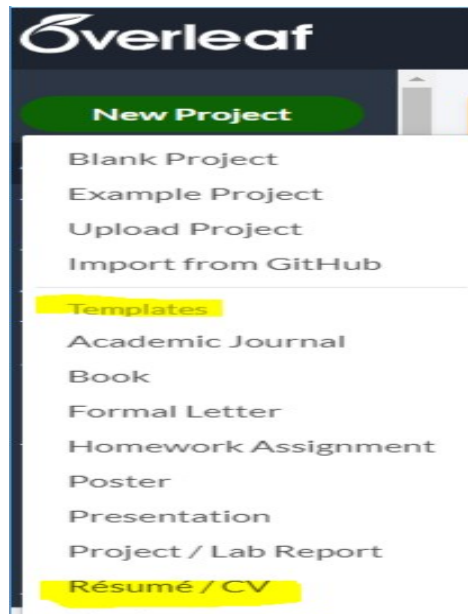


Figure 4

You will be redirected to a list of templates where you can select one that is suitable **Figure 5**.

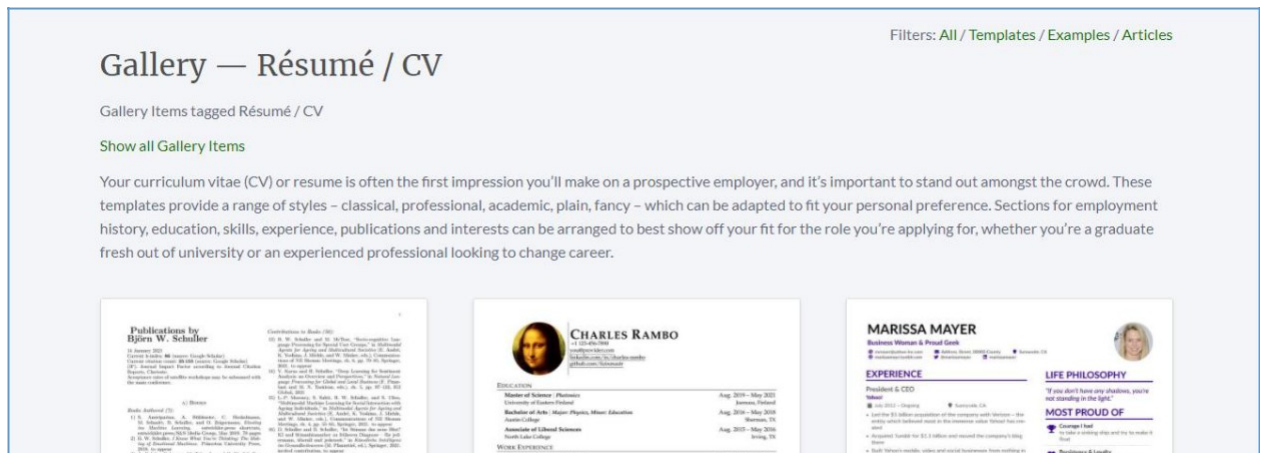


Figure 5

Clicking on the template thumbnail will display some further information about the template, and you can start editing your copy of the template on Overleaf right away by clicking 'Open as Template' **Figure 6**.

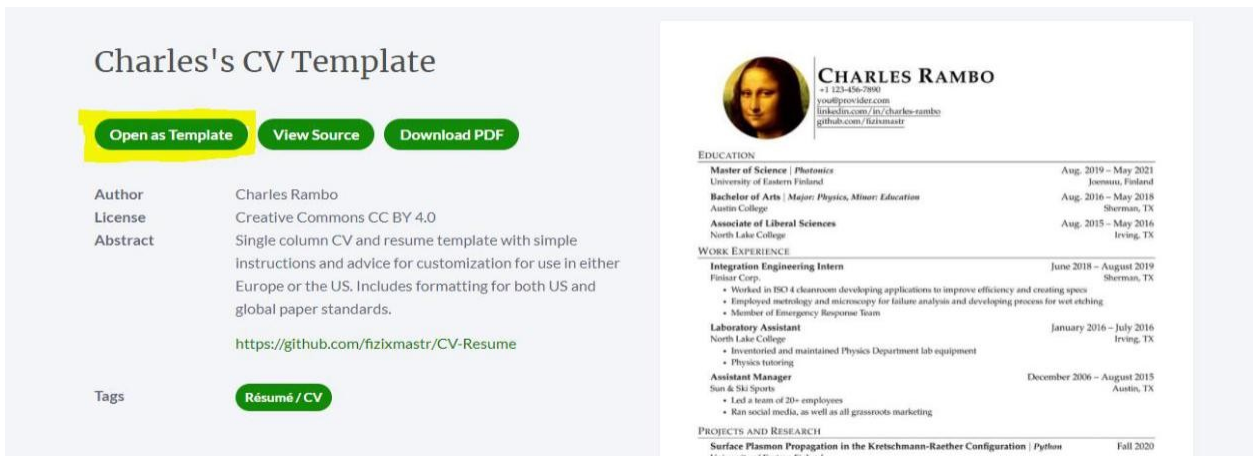


Figure 6

You can start editing the .tex file of your template now, to view the changes click Recompile **Figure 7**.

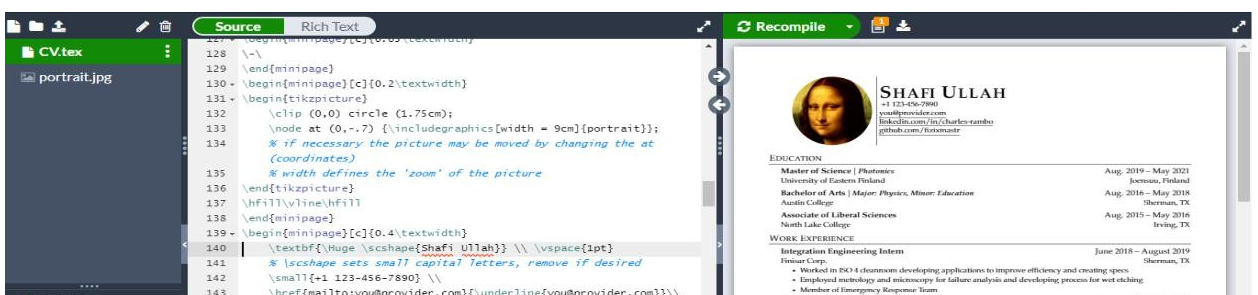


Figure 7

5.2. Adding Images

On Overleaf, you will first have to upload the images.

To upload an image, in the editor go to the upper left corner and click the upload icon, **Figure 8**.

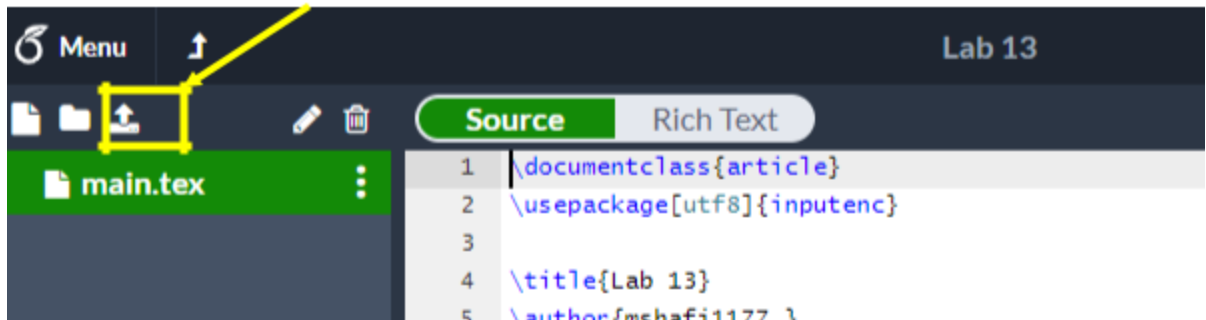


Figure 8

A dialogue box will pop up for you to upload your files, **Figure 9**.

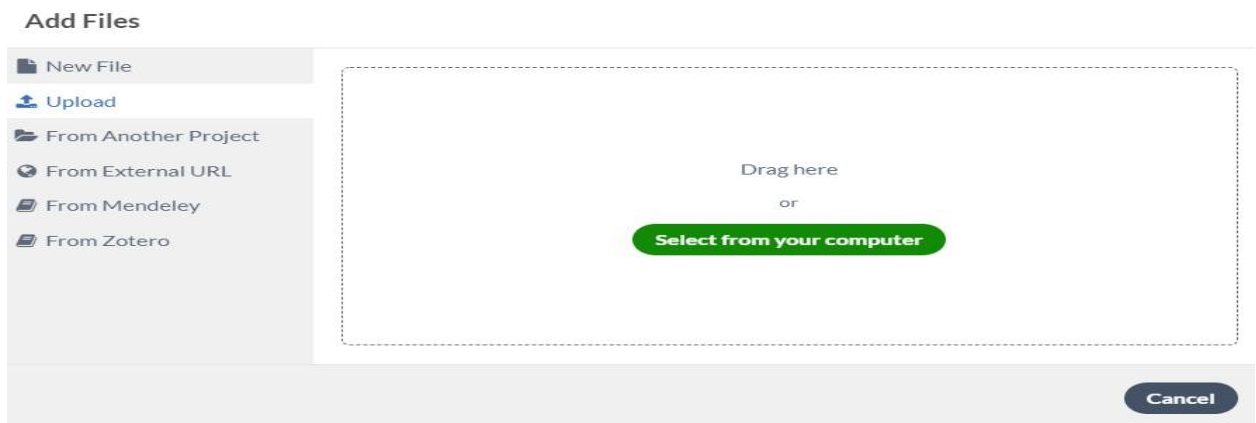


Figure 9

There you can either drag and drop your files or click Select files(s) to open a file system window. Navigate to the right folder and select the images to upload. You can upload several files at once. After the upload process is complete you will see the files in the left panel. **Figure 10**.

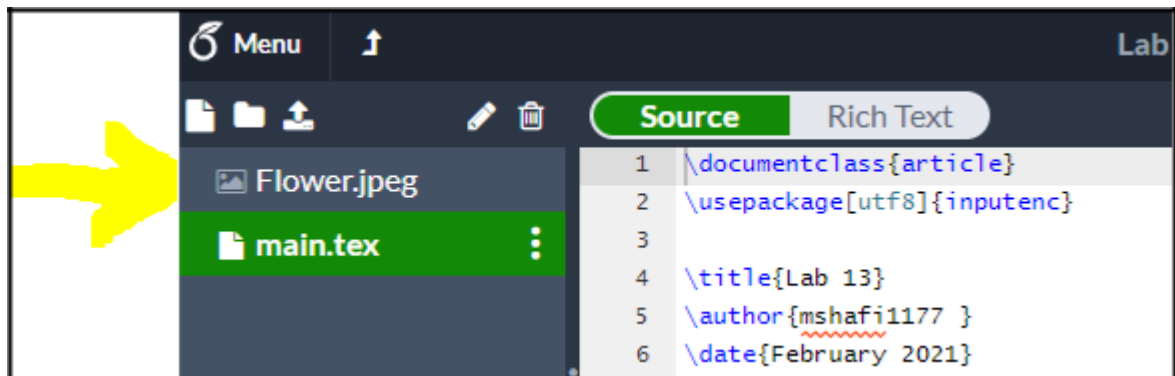


Figure 10

Below is a example on how to include a picture.

```
\documentclass{article}
\usepackage[utf8]{inputenc}
\usepackage{graphicx}

\begin{document}

  \includegraphics[width=4cm]{Flower.jpeg}

\end{document}
```

5.3. Creating Lists

Lists are very simple to create in LATEX. You can create lists using different list environments. Environments are sections of our document that we want to present in a different way to the rest of the document. They start with a `\begin{...}` command and end with an `\end{...}` command.

There are two main different types of lists, **ordered lists** and **unordered lists**. Each will use a different environment.

5.3.1. Unordered Lists

Unordered lists are produced by the `itemize` environment. Each entry must be preceded by the control sequence `\item` as shown below.

```
\documentclass{article}
\usepackage[utf8]{inputenc}
\begin{document}

\begin{itemize}
  \item item 1.
  \item Item 2.
  \item item 3.
  \item so on.
\end{itemize}

\end{document}
```

5.3.2. Ordered Lists

Ordered lists have the same syntax inside a different environment. We make ordered lists using the **enumerate** environment:

```
\documentclass{article}

\usepackage[utf8]{inputenc}

\begin{document}

\begin{enumerate}

\item This is the first entry in our list

\item The list numbers increase with each entry we add \end{enumerate}

\end{document}
```

5.4. Abstracts

In scientific documents, it's a common practice to include a brief overview of the main subject of the paper. In LATEX there's the `abstract` environment for this. The `abstract` environment will put the text in a special format at the top of your document.

```
\documentclass{article}

\usepackage[utf8]{inputenc}

\begin{document}

\begin{abstract}

This is a simple paragraph at the beginning of the

document. A brief introduction about the main subject.

\end{abstract}

\end{document}
```

5.5. Chapters and Sections

Documents usually have some levels of chapters and/or sections to keep their contents organized. LATEX supports this type of organization. Usually, `\section` is the top-level document command in most documents. However, in reports, books, and alike, this would be `\chapter`.

The command `\section{}` marks the beginning of a new section, inside the braces is set the title. Section numbering is automatic and can be disabled by including a `*` in the section command as `\section*{}`. We can also have `\subsection{}`s, and indeed `\subsubsection{}`s.

Note that `\chapter` is only available in report and book document classes. `\chapter` is used to add chapters in a document.

```
\documentclass{report}
```

```
\begin{document}
```

```
\tableofcontents{}
```

```
\chapter{Editing compile}
```

```
\section{First Compile}
```

how to compile basic hello world into a pdf.

Write your favorite text editor create file and copy/paste the following (with `hello.tex`):

```
\subsection{Output formats}
```

different output formats (dvi, pdf)

The output of this command `\latex hello.tex` will be a dvi file (`hello.dvi`). This file (`.dvi`) can be converted by `\Sdvipdf` `hello.dvi` The get an pdf file from tex file, run this command `\texi2pdf` `hello.tex`

```
\chapter{Document Structure}
```

```
\section{Reserved Characters}
```


The following symbols characters are reserved by LATEX because they introduce a command and have a special meaning.

```
\end{document}
```

```
\documentclass{report}
```

```
\begin{document}
```

```
\chapter{Editing compile}
```

```
\section{First Compile}
```

how to compile basic hello world into a pdf.

Write your favorite text editor create file and copy/paste the following (with hello.tex):

```
\subsection{Output formats}
```

different output formats (dvi, pdf)

The output of this command `\latex hello.tex` will be a dvi file (hello.dvi). This file (.dvi) can be converted by `\dvipdf` hello.dvi The get an pdf file from tex file, run this command `\texi2pdf` hello.tex

```
\chapter{Document Structure}
```

```
\section{Reserved Characters}
```

The following symbols characters are reserved by LATEX because they introduce a command and have a special meaning.

```
\end{document}
```

5.6. Adding a Table of Contents

To create the table of contents is straightforward, the command `\tableofcontents` does all the work for you. Use this command in the above code after the `\begin{document}` command used in the Chapters and Sections heading.

6. Practice Tasks in Lab

In this part, you are provided some practice tasks that will help you to create documents in LaTeX.

6.1. Task 1

Practice all the tasks explained in the Walkthrough task on overleaf.

6.2. Task2

Create an overleaf document similar to the document provided on LMS.

7. Assignment

Assignment for this Lab will be uploaded on LMS after the Lab session. The deadline will also be mentioned on LMS.

8. Further Readings

The reading material and other helping material regarding this Lab will be uploaded on LMS if any. You can access it from the LMS.

9. Evaluation Criteria

📁 **Method of Evaluation:** Viva, File submitted on LMS.

	Excellent 10	Good 9-7	Satisfactory 6-4	Unsatisfactory 3-1	Poor 0	Marks Obtained
Assignment	All tasks completed correctly in given time and have a complete understanding	Most tasks were completed correctly. Tasks could be improved further and have a complete understanding	Some tasks were completed correctly and have an incomplete understanding	Most tasks were incomplete or incorrect and have no understanding	All tasks were incomplete or incorrect. Didn't perform tasks and have no understanding	
Total						



SHIFA TAMEER-E-MILLAT UNIVERSITY

Lab Manual **Introduction to Computer Science**

DEPARTMENT OF COMPUTING

Shifa Tameer-e-Millat University

Department of Computing

INTRODUCTION TO COMPUTER SCIENCE

Lab 15

LaTeX

Lab 15: LaTeX

1. Introduction

Latex is a software system for document preparation. LaTeX is widely used in academia for the communication and publication of scientific documents in many fields, including mathematics, statistics, computer science, engineering, physics, economics, linguistics, quantitative psychology, philosophy, and political science.

Overleaf is a great online LaTeX edition tool that allows you to create LaTeX documents directly in your web browser. This lab explains how to create a new project in Overleaf, either starting from scratch or using one of the many templates available.

2. Activity Timeboxing

Heading No	Activity Name	Activity Time	Total Time
	About this Lab	10 minutes	10 minutes
5	Walkthrough Task	50 minutes	60 minutes
6	Practice Task in Lab	90 minutes	150 minutes

Table 1: Activity Timeboxing

3. Objective

- To create a LaTeX document
- Create tables in LaTeX, tables with borders
- To learn about document structure and formatting
- To learn about the basic commands of LaTeX.

4. Concept Map

In this lab, we learn about Latex. LaTeX is a tool used to create professional-looking documents. It is based on the WYSIWYM (what you see is what you mean) idea, meaning you only have to focus on the contents of your document and the computer will take care of the formatting. Instead of spacing out text on a page to control formatting, as with Microsoft Word or LibreOffice Writer, users can enter plain text and let LATEX take care of the rest.

4.1. Overleaf

Overleaf is a great online LaTeX edition tool that allows you to create LaTeX documents directly in your web browser. To start using Overleaf, go to www.overleaf.com. If you don't have an account enter your e-mail address and set a password in the corresponding boxes. Get started now, click Register and that's it, you will be redirected to the project management page where you will be guided into how to create a new project. If you already have an account, click Login in the upper right corner, then type in your email and password and click the Login button.

Once you are logged in, you will see the Overleaf Project Management page.

4.2. Starting a new project

To start a new project from scratch, on the main page click the New Project button, you will see the next drop-down menu then click Blank Project. A box will open where you should enter the name of your new project, then click Create. After that, you will be redirected to the editor. There, a new document is created with some basic information already filled in. You can start editing your .tex file now, to view the changes click Recompile or

5. Walkthrough Tasks

5.1. First document

5.1.1 Creating a Blank Project

On the Project, Management page click New project => Blank Project, enter a name for your project then click Create.

Detailed Explanation to Create a Blank Project: To start a new project from scratch, on the main page click the New Project button, you will see the next drop-down menu then click Blank Project as shown in **Figure 1**.

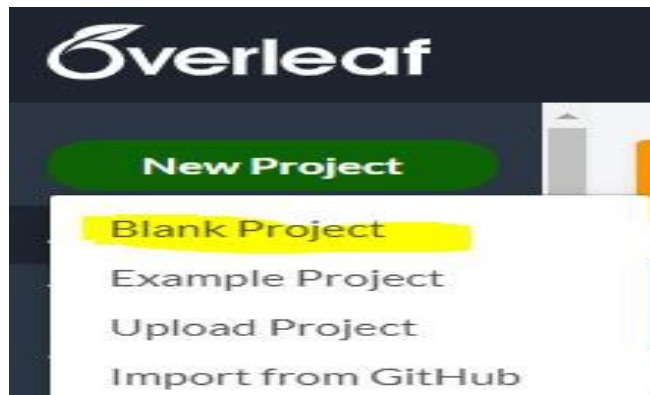


Figure 1

A box will open where you should enter the name of your new project, then click **Create** as shown in **Figure 2**.

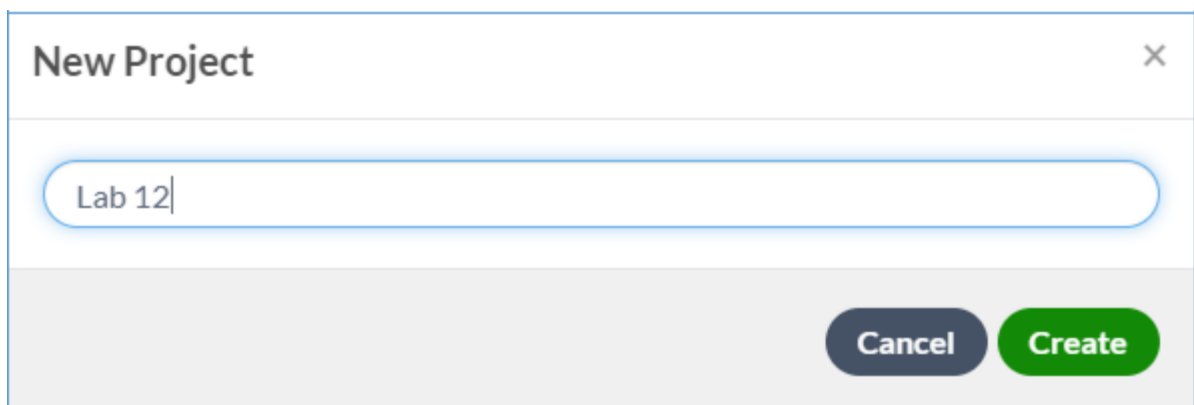
A screenshot of a 'New Project' dialog box. The title bar says 'New Project' with a close button (X) on the right. Inside the dialog, there is a text input field containing the text 'Lab 12'. At the bottom right of the dialog, there are two buttons: 'Cancel' (dark blue) and 'Create' (green).

Figure 2

After that, you will be redirected to the editor. There, a new document is created with some basic information already filled in. You can start editing your .tex file now, to view the changes click Recompile **Figure 3**.



Figure 3

5.1.2 Creating a project from a template

Using a template to start a new project will save you a lot of time. Overleaf provides a vast set of templates.

To create a document from a template, in the Project Management page click New Project, a drop-down menu will display, below Templates click on the document type you are about to write. **Figure 4**. In our case, we are going to use the Resume/CV template **Figure 4**.

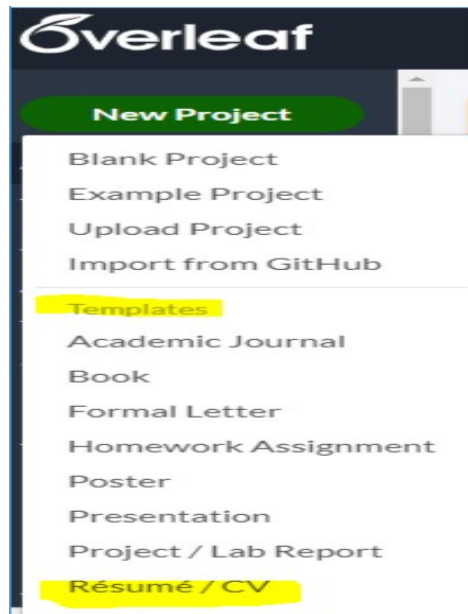


Figure 4

You will be redirected to a list of templates where you can select one that is suitable **Figure 5**.

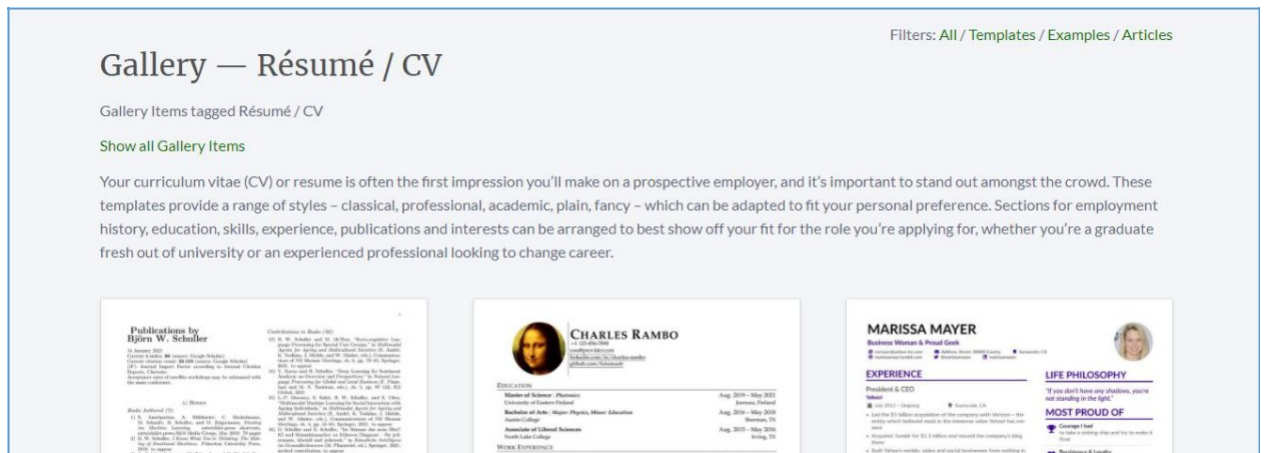


Figure 5

Clicking on the template thumbnail will display some further information about the template, and you can start editing your copy of the template on Overleaf right away by clicking 'Open as Template' **Figure 6**.

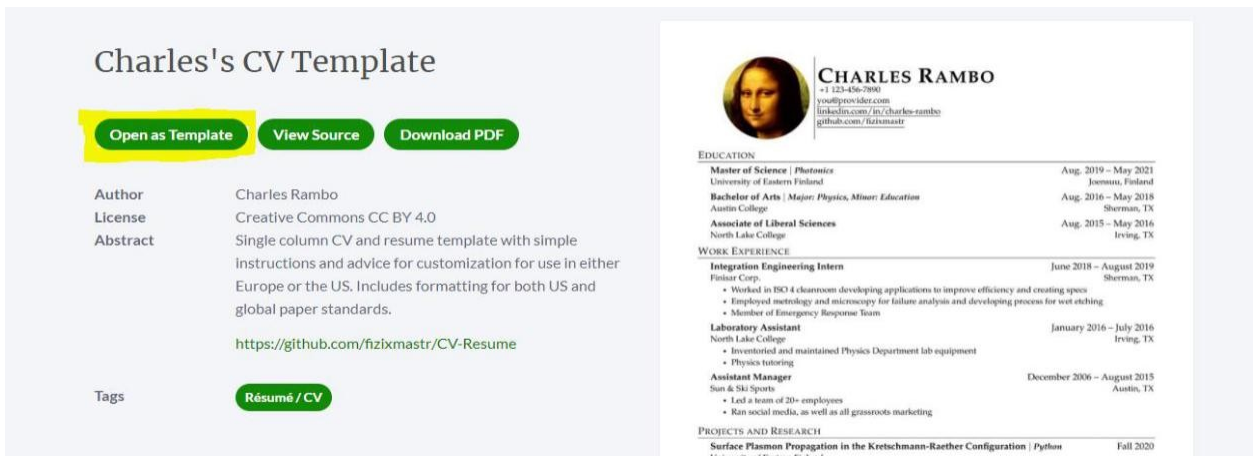


Figure 6

You can start editing the .tex file of your template now, to view the changes click Recompile **Figure 7**.

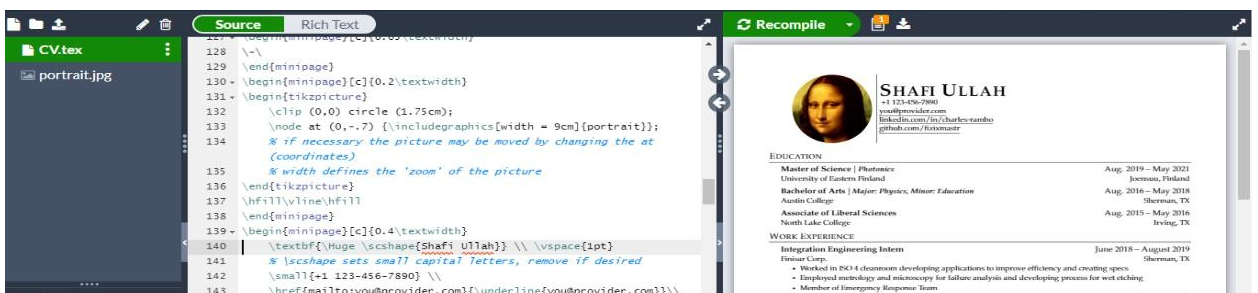


Figure 7

5.2. Creating a simple table in LATEX

Below you can see the simplest working example of a table.

```
\begin{center}
\begin{tabular}{c c c}
cell1 & cell2 & cell3 \\
cell4 & cell5 & cell6 \\
cell7 & cell8 & cell9
\end{tabular}
\end{center}
```

The tabular environment is the default LATEX method to create tables. You must specify a parameter to this environment, in this case `{c c c}`. This tells LATEX that there will be three columns and that the text inside each one of them must be centered. You can also use `r` to align the text to the right and `l` for left alignment. The alignment symbol `&` is used to specify the breaks in the table entries. There must always be one less alignment symbol in each line than the number of columns. To go to the next line of your table, we use the new line command `\\`. We wrap the entire table inside the center environment so that it will appear in the center of the page.

5.3. Adding borders

The tabular environment is more flexible, you can put separator lines in between each column.

```
\begin{center}
\begin{tabular}{|c|c|c|}
\hline
cell1 & cell2 & cell3 \\
cell4 & cell5 & cell6 \\
cell7 & cell8 & cell9 \\
\hline
\end{tabular}
\end{center}
```

You can add borders using the horizontal line command `\hline` and the vertical line parameter `|`.

- `{|c|c|c|}`: This declares that three columns, separated by a vertical line, are going to be used in the table. The `|` symbol specifies that these columns should be separated by a vertical line.

■ `\hline`: This will insert a horizontal line. We have included horizontal lines at the top and bottom of the table here. There is no restriction on the number of times you can use `\hline`.

Below you can see a second example.

```
\begin{center}
\begin{tabular}{||c c c c||}
\hline
Col1 & Col2 & Col2 & Col3 \\
\hline\hline
1&6&87837&787\\
\hline
2&7&78&5415\\
\hline
3 & 545 & 778 & 7507 \\
\hline
4 & 545 & 18744 & 7560 \\
\hline
5&88&788&6344\\
\hline
\end{tabular}
\end{center}
```

You can learn more about the tables in latex from [here](#).

5.4. Include a bibliography

We will see in the lab how we can add bibliography (references) in our latex document or you can see from [here](#) how you can add bibliography (references) in your latex document.

6. Practice Tasks in Lab

Create an overleaf report document. You can choose any template from the overleaf templates or you can create your blank document. A sample report document is available on LMS. You have to create a similar report document in which create separate files for each chapter and then include them in the main file.

7. Assignment

Assignment for this Lab will be uploaded on LMS after the Lab session. The deadline will also be mentioned on LMS.

8. Further Readings

The reading material and other helping material regarding this Lab will be uploaded on LMS if any. You can access it from the LMS.

9. Evaluation Criteria

📁 **Method of Evaluation:** Viva, File submitted on LMS.

	Excellent 10	Good 9-7	Satisfactory 6-4	Unsatisfactory 3-1	Poor 0	Marks Obtained
Assignment	All tasks completed correctly in given time and have a complete understanding	Most tasks were completed correctly. Tasks could be improved further and have a complete understanding	Some tasks were completed correctly and have an incomplete understanding	Most tasks were incomplete or incorrect and have no understanding	All tasks were incomplete or incorrect. Didn't perform tasks and have no understanding	
Total						