University of Mosul / College of Engineering / Department of Computer Engineering

Level 2 Laboratories

Description of the Microprocessor I Laboratory

1. **General Information:**	
Laboratory Name:	Microprocessor I (Lab-312)
Associated Course Name:	Microprocessor I
Department:	Computer Engineering
Weekly Lab Hours:	3
Number of Weeks in the Semester:	15
Academic Level:	2
Lab Supervisor:	Dr. Mazin Hashim Aziz

2. General Description of the Laboratory:

The Microprocessor I Lab aims to reinforce the theoretical concepts covered in the Microprocessor I course lectures, through real-life practical applications using tools and software dedicated to microprocessors. The lab provides an interactive environment that enables students to acquire the skills of writing and executing software in assembly language, and tracking, detecting, and debugging errors.

3. Laboratory Objectives:

- Train students to write software in assembly language.
- Learn about different instructions.
- Practice using programming tools.
- Use a simulation environment to track, detect, and debug errors.
- Write and develop assembly codes, display results, and improve them.
- Enhance project implementation skills.

4. Learning Outcomes:

By the end of the lab, the student should be able to:

- Write programs in assembly language.
- Trace, detect, and debug errors.
- Execute programs in assembly language.
- Write and implement intermediate-level software applications in assembly language.
- Acquire a sense of teamwork.

5. Weekly Experiment Schedule:

Week	Experiment Title	Tools / Software Used	Main Objective
1	Introduction to the Tools Used	Computer	Learn Assembly Language Concepts
2	Setting up the DOSBOX Environment	DOSBOX Environment Installation	Use A Virtual Environment
3	Setting up the Debug Program	Debug Software Installation	Learn How To Use A Program To Detect, Trace, And Debug Errors
4	Setting up the Masm & Link Program	Masm & Link Software Installation	Learn How To Write In Assembly Language Until Execution
5	Writing a Simple Assembly Program	Text Editor + The Software tools	Simplified Application
6	Data Transfer Instructions	Same	Use Data Transfer Instructions

7	Logical Instructions	Same	Use Logical Instructions
8	Shft and Rotate Instructions	Same	Use Scroll And Rotate Instructions
9	Branching Instructions	Same	Use Branch Instructions
10	Arithmetic Instructions	Same	Use Arithmetic Instructions
11	String Instructions	Same	Use String Instructions
12	Control Instructions	Same	Use Control Instructions
13	Manipulating Video Memory (VRAM)	Same	Interactive Application Using Video Memory
14	Applications Using Interrupt Service Routines	Same	Integrate Acquired Skills Into A Final Project

6. Tools and Equipment Used:

• Computers.

• Software.

7. Safety Guidelines:

● Ensure the power is off when connecting devices.

● Do not touch electrical outlets or network components without the supervisor's permission.

● Remain calm and organize cables to avoid accidents.

● Use the simulator for routing experiments before testing on real devices.

8. Evaluation Method:

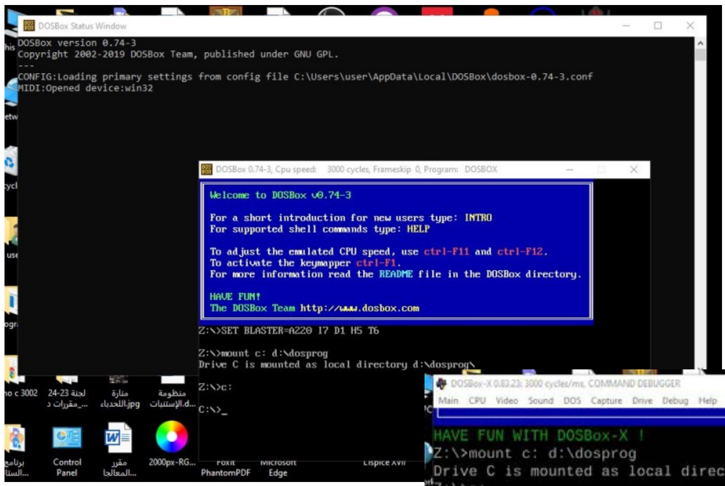
Evaluation Item	Percentage
Attendance and Participation	10%
Weekly Experiment Reports	30%
Quizzes	20%
Final Practical Exam	30%
Practical Project	10%

9. References and Sources:

- Walter Triebel and Avtar Singh, The 8088 and 8086 Microprocessors: programming, Interfacing, software, Hardware, Applications, 4th edition, prentice-Hall, 2002.
- Lectures, experiment manual, and notes
- The Intel microprocessors 8086/8088, 80186/80188, 80286, 80386, 80486, Pentium, Pentium Pro processor, Pentium II, Pentium III, Pentium 4, and Core2 with 64-bit extensions: architecture, programming, and interfacing by: Barry B. Brey—8th ed.
- https://classroom.google.com/c/NzEyMjE0ODcyMTUx
- https://www.eng.auburn.edu/~sylee/ee2220/8086_instruction_set.html

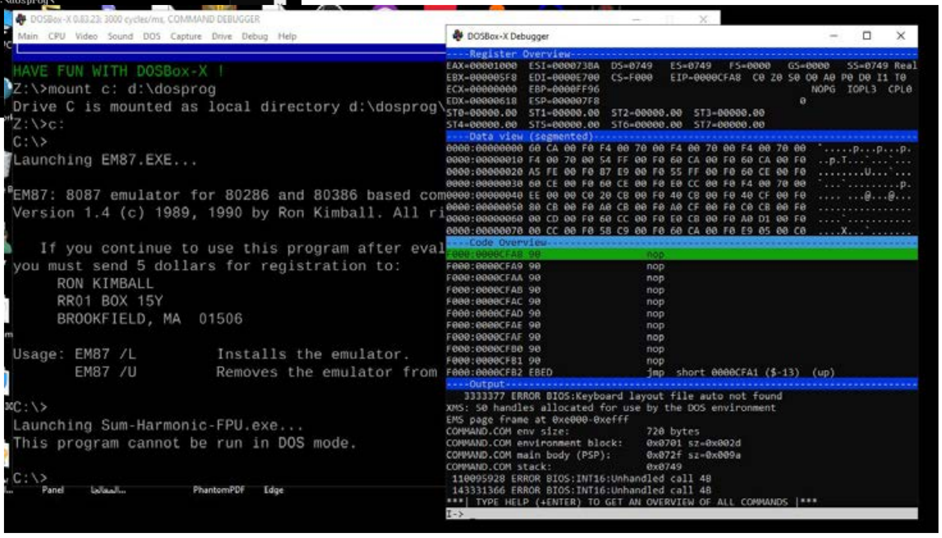
10. Attachments:

• Experiment report form


DOSBox Status Window
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
Control Panel
Foxit PhantomPDF
Microsoft Edge

```
DOSBox version 0.74-3
Copyright 2002-2019 DOSBox Team, published under GNU GPL.
---
CONFIG:Loading primary settings from config file C:\Users\user\AppData\Local\DOSBox\dosbox-0.74-3.conf
MIDI:Opened device:win32
```

```
Welcome to DOSBox v0.74-3

For a short introduction for new users type: INTRO
For supported shell commands type: HELP

To adjust the emulated CPU speed, use ctrl-F11 and ctrl-F12.
To activate the keymapper ctrl-F1.
For more information read the README file in the DOSBox directory.

HAVE FUN!
The DOSBox Team http://www.dosbox.com
Z:\>SET BLASTER=A220 I7 D1 H5 T6

Z:\>mount c: d:\dosprog
Drive C is mounted as local directory d:\dosprog\

Z:\>c:

C:\>_
```



```
HAVE FUN WITH DOSBox-X !
Z:\>mount c: d:\dosprog
Drive C is mounted as local directory d:\dosprog\
Z:\>c:
C:\>
Launching EM87.EXE...

EM87: 8087 emulator for 80286 and 80386 based com
Version 1.4 (c) 1989, 1990 by Ron Kimball. All ri

   If you continue to use this program after eval
you must send 5 dollars for registration to:
     RON KIMBALL
     RR01 BOX 15Y
     BROOKFIELD, MA  01506

Usage: EM87 /L        Installs the emulator.
       EM87 /U        Removes the emulator from

C:\>
Launching Sum-Harmonic-FPU.exe...
This program cannot be run in DOS mode.

C:\>
```

```
----Register Overview--------------------------------------------------------
EAX=00001000  ESI=0000738A  DS=0749   ES=0749   FS=0000   GS=0000   SS=0749 Real
EBX=000005F8  EDI=0000E700  CS=F000   EIP=0000CFA8  C0 Z0 S0 O0 A0 P0 D0 I1 T0
ECX=00000000  EBP=0000FF96                                   NOPG  IOPL3  CPL0
EDX=00000618  ESP=000007F8                                 0
ST0=00000.00  ST1=00000.00  ST2=00000.00  ST3=00000.00
ST4=00000.00  ST5=00000.00  ST6=00000.00  ST7=00000.00
----Data view (segmented)----------------------------------------------------
0000:00000000 60 CA 00 F0 F4 00 70 00 F4 00 70 00 F4 00 70 00  `.....p...p...p.
0000:00000010 F4 00 70 00 54 FF 00 F0 60 CA 00 F0 60 CA 00 F0  ..p.T...`...`...
0000:00000020 A5 FE 00 F0 87 E9 00 F0 55 FF 00 F0 60 CE 00 F0  ........U...`...
0000:00000030 60 CE 00 F0 60 CE 00 F0 E0 CC 00 F0 F4 00 70 00  `...`.........p.
0000:00000040 EE 00 00 C0 20 CB 00 F0 40 C8 00 F0 40 CF 00 F0  .... ...@...@...
0000:00000050 80 CB 00 F0 A0 CB 00 F0 A0 CF 00 F0 C0 CB 00 F0  ................
0000:00000060 00 CD 00 F0 60 CC 00 F0 E0 C8 00 F0 A0 D1 00 F0  ....`...........
0000:00000070 00 CC 00 F0 58 C9 00 F0 60 CA 00 F0 E9 05 00 C0  ....X...`.......
----Code Overview------------------------------------------------------------
F000:0000CFA8 90                       nop
F000:0000CFA9 90                       nop
F000:0000CFAA 90                       nop
F000:0000CFAB 90                       nop
F000:0000CFAC 90                       nop
F000:0000CFAD 90                       nop
F000:0000CFAE 90                       nop
F000:0000CFAF 90                       nop
F000:0000CFB0 90                       nop
F000:0000CFB1 90                       nop
F000:0000CFB2 EBED                     jmp  short 0000CFA1 ($-13)  (up)
----Output-------------------------------------------------------------------
   3333377 ERROR BIOS:Keyboard layout file auto not found
XMS: 50 handles allocated for use by the DOS environment
EMS page frame at 0xe000-0xefff
COMMAND.COM env size:              720 bytes
COMMAND.COM environment block:     0x0701 sz=0x002d
COMMAND.COM main body (PSP):       0x072f sz=0x009a
COMMAND.COM stack:                 0x0749
 110095928 ERROR BIOS:INT16:Unhandled call 48
 143331366 ERROR BIOS:INT16:Unhandled call 48
***| TYPE HELP (+ENTER) TO GET AN OVERVIEW OF ALL COMMANDS |***
I->
```

Experiment No. (6)

Student Name:

Data Transfer Instructions

Objective:-

• Learn how to use data transfer instructions.

Equipments:-

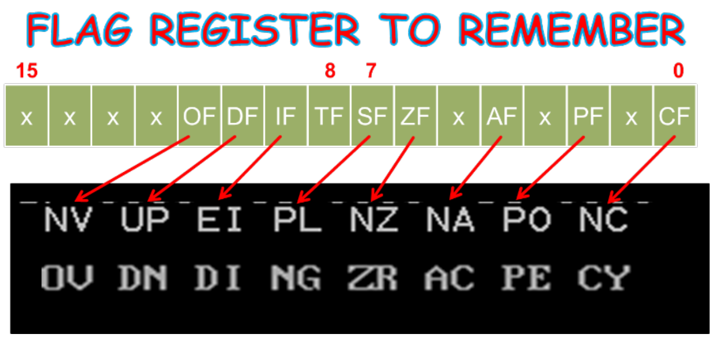
•PC

Procedure:-

For the steps (1, 4) write the instructions using (A) command, then execute them step by step using (T) command.

1. Use (D) command after each step from (h-k) to observe the contents of the specified memory locations:

```
a)      -R
b)      -A 100
c)      XXXX:0100    MOV   CX,1122          CX=.. ..
d)      XXXX:0103    MOV   AX,3344         AX=.. ..
e)      XXXX:0106    XCHG CL,AH           AX=.. ..                    ,CX=.. ..
f)      XXXX:0108    MOV   BX,10            BX=.. ..
g)      XXXX:010B    MOV   DI,200          DI=.. ..
h)      XXXX:010E    MOV   DX,5566         DX=.. ..
i)XXXX:0111 MOV   AL,[BX]          AL=………
j)XXXX:0113 LEA  SI,[BX]          SI=.....
k)      XXXX:0115    LDS  DI,[BX]          DI=……            ,DS=……
l)XXXX:0117 LES  BX,[DI]           BX=.....                 ,ES=……
```


FLAG REGISTER TO REMEMBER
15
8
7
0
x x x x OF DF IF TF SF ZF x AF x PF x CF
NV UP EI PL NZ NA PO NC
OV DN DI NG ZR AC PE CY

2. Apply each of the following steps (a through g), then state the results.

```
a)      -RF
b)      –RAX                          AH=.. .. ..
c)      -A 100
d)      XXXX:0100      LAHF      <ENTER>
e)      -T=100
f)      -RAX                          AH=.. .. ..
g)      -RF
```

Apply each of the following steps (a through f), then state the results.

```
a)     -A 100
b)     XXXX:0100      MOV AH,0
c)     XXXX:01XX      SAHF <ENTER>
d)     -T=100
e)     -RAX
f)     -RF                        Flag register=.. .. .. .. .. .. .. .. .. .
```

3. Apply each of the following steps (a through o), then use (D) command prior to each XLAT instruction to observe the contents of the specified memory locations:

```
a)     -R
b)     -F 100 120 “MICROPROCESSOR LAB”
c)     -F 200 220 “microprocessor lab”
d)     -A 100
e)     XXXX:0100     MOV   AL,0             AL=.. ..
f)     XXXX:01XX     MOV   BX,100           BX=.. ..
g)     XXXX:01XX     MOV   DI,200          DI=.. ..
h)     XXXX:01XX     XLAT                  AL=.. ..
i)XXXX:01XX          MOV   AL,9             AL=.. ..
j)XXXX:01XX          XLAT                  AL=.. ..
k)     XXXX:01XX     XCHG DI,BX            DI=.. .. .. ,BX= .. .. ..
l)XXXX:01XX          MOV   AL,9            AL=.. ..
m)     XXXX:01XX     XLAT                  AL=.. ..
n)     XXXX:01XX     MOV   AL,F             AL=.. ..
o)     XXXX:01XX     XLAT                  AL=.. ..
```

Description of the Microprocessor II Laboratory

1. General Information:

Laboratory Name:	Microprocessor II (Lab-312)
Associated Course Name:	Microprocessor II
Department:	Computer Engineering
Weekly Lab Hours:	3
Number of Weeks in the Semester:	15
Academic Level:	2
Lab Supervisor:	Dr. Mazin Hashim Aziz

2. General Description of the Laboratory:

The Microprocessor II Lab aims to reinforce the theoretical concepts covered in the Microprocessor II lectures, through real-life practical applications using tools and software dedicated to microprocessors. The lab provides an interactive environment that enables students to acquire the skills of designing circuits compatible with the 8086 microprocessor, writing programs in assembly language, executing them in a virtual environment, viewing timelines, and analyzing results.

3. Laboratory Objectives:

- Train students in design.
- Learn about various digital elements.
- Practice using virtual tools.
- Use a simulation environment to prepare and modify designs.
- Write software in assembly language, implement it using the prepared designs, and display and improve the results.
- Enhance project implementation skills.

4. Learning Outcomes:

By the end of the lab, the student should be able to:

- Use the Proteus simulation environment.
- Prepare and implement designs.
- Execute assembly language programs and apply them to the prepared designs.
- Analyze results, identify errors, and improve designs.
- Acquire a sense of teamwork.

5. Weekly Experiment Schedule:

Week	Experiment Title	Tools / Software Used	Main Objective
1	Introduction to the Tools Used	Computer	Learn the concepts of using simulation tools in design
2	Setting Up the Proteus Environment	Installing the Proteus Environment	Gain software installation and configuration skills
3	Available Design Tools and	Proteus	Learn the tools

	Libraries		provided by Proteus and how to employ them
4	Design and Implementation for Simulating Simple Logic Circuits	Proteus	Use design tools to implement simple logic circuits and learn about input signal simulation tools and results display tools
5	Design and Implementation of a Circuit to Simulate the 8086 Microprocessor Standalone	Proteus	Implement a microprocessor circuit and write its first program to display processor signals
6	Design and Implementation of a Circuit to Simulate the Address and Data Buses of the 8086 Microprocessor	Proteus	Apply theoretical knowledge to identify processor buses
7	Design and Implementation of a Circuit to Simulate the Address Signal Demultiplexing of the 8086 Microprocessor	Proteus	Apply theoretical knowledge to address signal separation
8	Design and Implementation of a Circuit to Simulate the Data Bus Buffers of the 8086 Microprocessor	Proteus	Apply theoretical knowledge to build data signal buffers
9	Design and Implementation	Proteus	Apply theoretical

	of a Circuit to Simulate the Address Decoding of the 8086 Microprocessor		knowledge to build a decoder circuit
10	Design and Implementation of a Circuit to Simulate the 8086 Microprocessor's Memory Interface - 1	Proteus	Apply theoretical knowledge to build a compatibility circuit with a specific memory size
11	Design and Implementation of a Circuit to Simulate the 8086 Microprocessor's Memory Interface - 2	Proteus	Apply theoretical knowledge to modify the memory compatibility circuit when changing the memory size.
12	Design and Implementation of a Circuit to Simulate the Interoperability of the 8086 Microprocessor with Peripherals - 1	Proteus	Apply theoretical knowledge to build a compatibility circuit with the input and output terminals.
13	Design and Implementation of a Circuit to Simulate the Interoperability of the 8086 Microprocessor with Peripherals - 2	Proteus	Apply theoretical knowledge to build a compatibility circuit with the input and output terminals when increasing their capacity.
14	Design and Implementation of a Circuit to Simulate the	Proteus	Integrate the acquired skills into a final

	Interoperability of the 8086 Microprocessor with Memory and Peripherals		project.

6. **Tools and Equipment Used:**
• Computers. • Software.

7. **Safety Guidelines:**
● Ensure the power is off when connecting devices. ● Do not touch electrical outlets or network components without the supervisor's permission. ● Remain calm and organize cables to avoid accidents. ● Use the simulator for routing experiments before testing on real devices.

8. Evaluation Method:

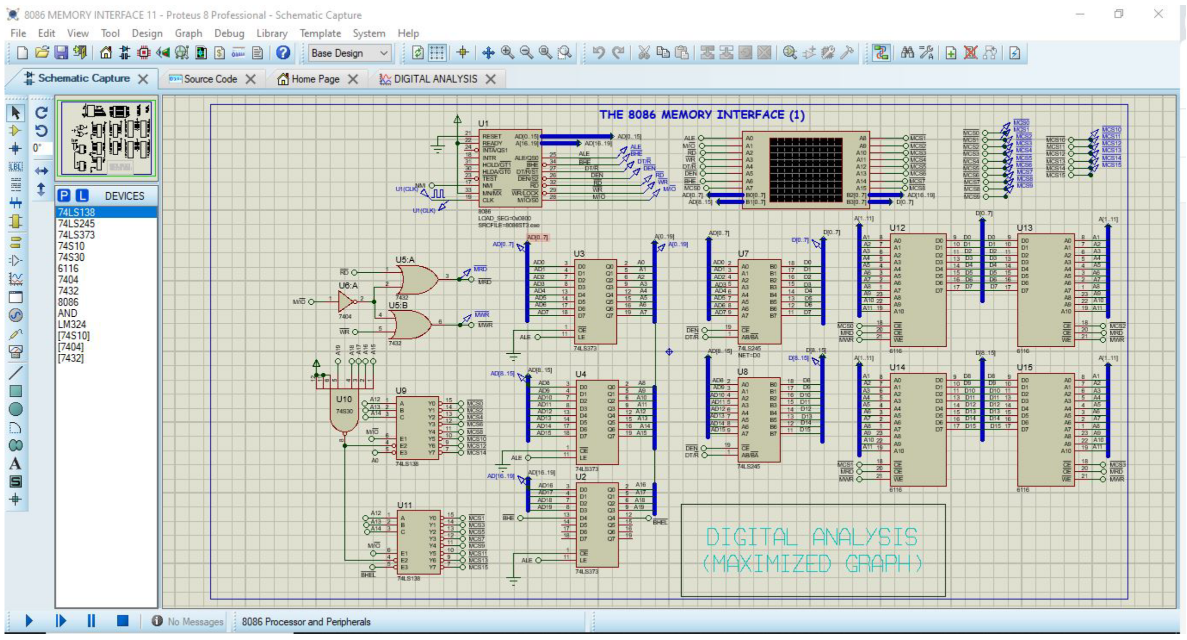
Evaluation Item	**Percentage**
Attendance and Participation	10%
Weekly Experiment Reports	30%
Quizzes	20%
Final Practical Exam	30%
Practical Project	10%

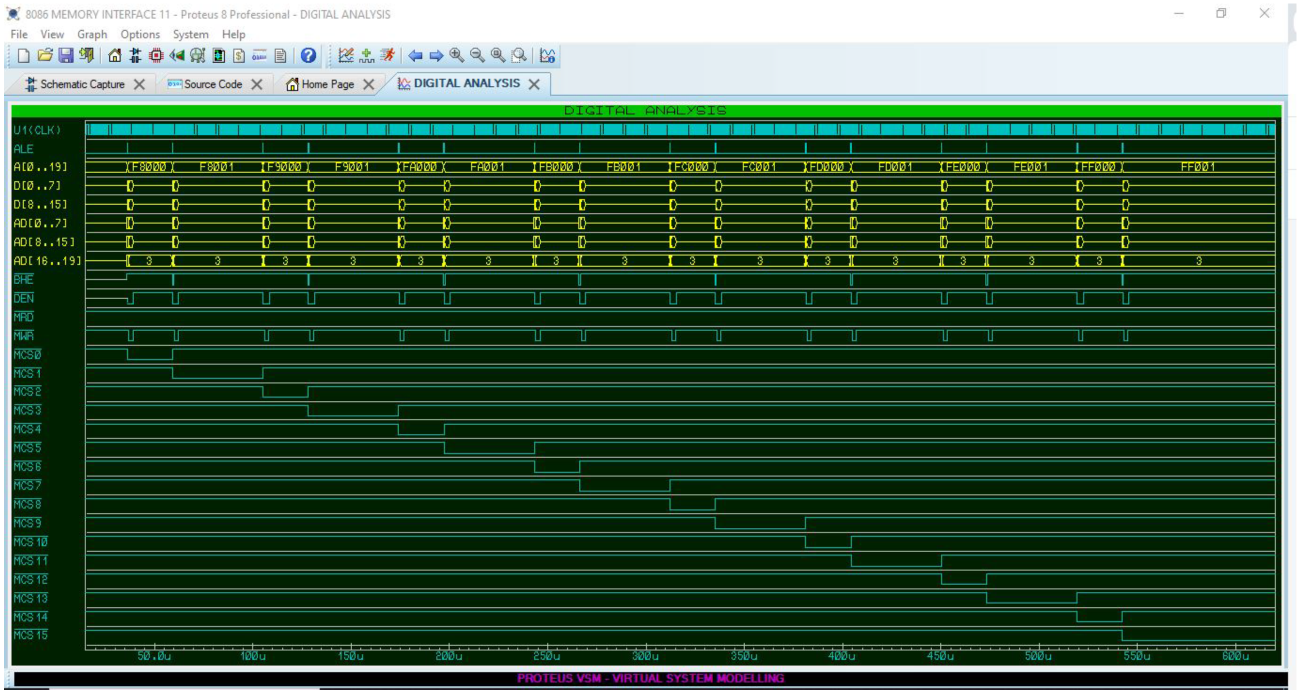
9. References and Sources:

- The Intel microprocessors 8086/8088, 80186/80188, 80286, 80386, 80486, Pentium, Pentium Pro processor, Pentium II, Pentium III, Pentium 4, and Core2 with 64-bit extensions: architecture, programming, and interfacing by: Barry B. Brey—8th ed.
- https://classroom.google.com/c/NzQ1Nzk2ODA2NjYx

10. Attachments:

• Experiment report form


8086 MEMORY INTERFACE 11 - Proteus 8 Professional - Schematic Capture
File Edit View Tool Design Graph Debug Library Template System Help
Schematic Capture
Source Code
Home Page
DIGITAL ANALYSIS
THE 8086 MEMORY INTERFACE (1)
DEVICES
74LS138
74LS245
74LS373
74S10
74S30
6116
7404
7432
8086
AND
LM324
[74S10]
[7404]
[7432]
DIGITAL ANALYSIS (MAXIMIZED GRAPH)
No Messages
8086 Processor and Peripherals


8086 MEMORY INTERFACE 11 - Proteus 8 Professional - DIGITAL ANALYSIS
File View Graph Options System Help
Schematic Capture
Source Code
Home Page
DIGITAL ANALYSIS
DIGITAL ANALYSIS
U1(CLK)
ALE
A[0..19]
D[0..7]
D[8..15]
AD[0..7]
AD[8..15]
AD[16..19]
BHE
DEN
MRD
MWR
MCS0
MCS1
MCS2
MCS3
MCS4
MCS5
MCS6
MCS7
MCS8
MCS9
MCS10
MCS11
MCS12
MCS13
MCS14
MCS15
F8000
F8001
F9000
F9001
FA000
FA001
FB000
FB001
FC000
FC001
FD000
FD001
FE000
FE001
FF000
FF001
50.0u
100u
150u
200u
250u
300u
350u
400u
450u
500u
550u
600u
PROTEUS VSM - VIRTUAL SYSTEM MODELLING

Experiment No. (11)

Student Name:

THE MEMORY INTERFACE (1)
Chip Selects Generation

Objective:-

- Learning how to simulate an 8086-microprocessor using **Proteus8.1 suit** and perform memory chip select generation.
- Observing the effects of software changes on the outputs of the chips selection signals.

equipment:-

- PC with **Proteus 8.1 Professional Design Suite** package.

Procedure:-

1) Open the Proteus program as **administrator**.
2) Start a new project and name it (**8086 MEMORY INTERFACE 1++.pdsprj**).
3) Design the 8086 Microprocessor memory interface circuit as shown in (Fig. 1).
(**Hint. You can start from the previous experiment design**.)
4) The 8086- assembly program should have the following commands:
HINT: Assume the used memory chips are 2Kbytes each.

```
CODE    SEGMENT PUBLIC 'CODE'
        ASSUME CS: CODE

START:
1.      CONT:
2.              MOV   AX,0F800h             ;Segment Base Address
3.              MOV   DS, AX
4.              MOV   SI,0                  ;Byte Pointer
5.              MOV   BX,0                  ;Memory Chip Pointer
6.      NXT:
7.              MOV   DL,0AAh               ;Data for Even Byte
8.              MOV   [SI+BX], DL
9.              MOV   DL,0BBh               ;Data for Odd Byte
10.             MOV   [SI+BX+1], DL
11.             ADD   BX,1000h              ;Chip Pointer Increment (2X2Kbyte)
12.             JMP   NXT

CODE    ENDS
        END START
```

5) Start the simulation:
(A) Either from the "Main-Menu-Bar", click on "Debug", then "Run Simulation", Or from the MDI control bar ▶ ▶ ❚❚ ■ at the left button of the screen.

(B) If there were no errors, the simulation will start and a colored small square will flash near each pin of the working chips of the circuit, which indicate the state of each pin.

6) Right click on the Virtual Logic Analyzer, then go-down to the bottom of the menu and click on "VSM Logic Analyzer". The VSM Logic Analyzer window will be opened.

7) Ensure that the "Capture Resolution" knob setting is around 25ns, if not you can control the knob to get the required setting.

8) Click-on the "Capture " push-button and wait for the signals to be displayed.

9) Now the "Display Scale" knob can be used to control the display range for better observations.

10) Note that the signals will be displayed according to its connection sequence to the VSM Logic Analyzer in the circuit diagram.

11) You can watch the signals (ALE, RD, WR, BHE, DT/R, DEN, M/IO, AD0-19, MSC0, MCS1, ………).

12) Add a Digital Analysis Graph and add a Voltage Probe to monitor the above signals and buses.

Lab work:

(1) Add the digital analysis graph and add all buses and signals to it using the voltage probes and use it to watch the signals.

(2) Add and connect the necessary signals to the logic analyzer and to the digital analysis graph and write the above code in the source code tab.

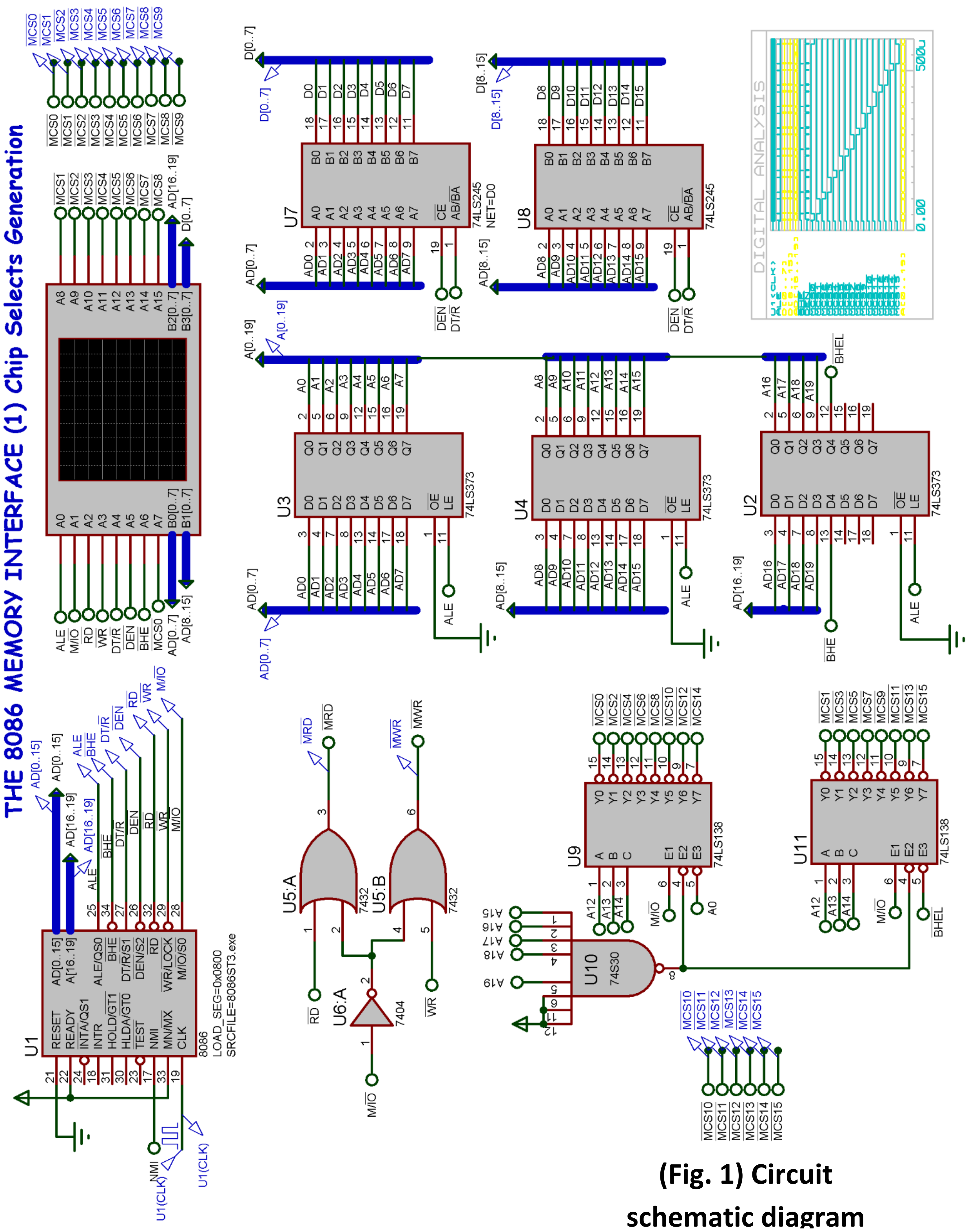
(3) Run the simulator and draw the timing diagram showing all the important signals and buses.

(4) Stop the simulation, make the following changes, and run the simulator after each of them, watch their effects on different signals and record your notes:

(A) What will happen if you change the instruction in line 1 to (MOV AX, 0FC00h), and try to fix any problem you will face.

(B) Convert the program to move 16-bit in each cycle.

(5) Draw the signal timing diagram for each case and state the affected signals?



(Fig. 1) Circuit schematic diagram

Description of the Programmable Logic Design Laboratory

1. General Information:	
Laboratory Name:	Programmable Logic Design Laboratory (Lab 210)
Associated Course Name:	Programmable Logic Design
Department:	Computer Engineering
Weekly Lab Hours:	3 hours
Number of Weeks in the Semester:	15 weeks
Academic Level:	Second level
Lab Supervisor:	Dr. Shawkat Sabah Khairullah

2. General Description of the Laboratory:

Design and analysis of clocked sequential digital circuits such as flip-flops, shift registers, counters, and pattern detectors; the architectural concepts of different programmable logic devices (PLDs); Hazards in combinational logic circuits and eliminating techniques; field programmable gate array (FPGA) design techniques using very high-speed circuit hardware description language (VHDL) and introduction to modeling, simulation, synthesis (with Xilinx, Altera, or Intel FPGAs). This course will present the syntax, structure, and data types used in HDLs and gain proficiency in writing basic HDL code.

3. Laboratory Objectives:

The basic objective of this course is to instruct the students the basic principles of modern digital systems, VHDL and programmable logic design.

4. Learning Outcomes:

Here are five Course Learning Outcomes (CLOs) for the course "Digital System Fundamentals":

CLO 1: An ability to identify, analyze, and solve complex engineering problems according to principles of engineering, science, and mathematics.

CLO 2: An ability to acquire and apply new knowledge and using appropriate learning strategies.

CLO 3: An ability to participate and work professionally and ethically in different projects to function on multi-disciplinary teams.

CLO 4: Students will design and analyze clocked sequential and interactive digital circuits such as flip-flops, shift registers, counters, pattern detectors and understand their applications.

CLO 5: Students will learn how to model basic combinational logic circuits such as logic gates, multiplexers, decoders, and other combinational circuits using hardware description languages (HDLs), such as VHDL. They will also be able to use VHDL for coding the basic sequential logic circuits, including latches, flip-flops, state machines, registers, and counters.

CLO 6: Students will develop a solid understanding of the architectural concepts and programmable technologies for different types of programmable logic devices (PLDs) such as programmable read only memory (PROM), programmable array logic (PAL), programmable logic array (PLA), generic array logic (GAL), complex programmable logic design (CPLD), and field programmable gate array (FPGA).

CLO 7: Students will understand the hazard in combinational logic circuits that caused by a deficiency in the digital system or external influences. There are basically two types of hazards: static and dynamic that can be eliminated using different mitigation methods.

CLO 8: Students will gain skills in simulating and verifying HDL designs. They will learn to use simulation tools to test and validate their designs before implementation. They will gain an understanding of design abstraction techniques in HDL using structural modeling. They will learn how to divide a complex design into hierarchical modules and describe the interconnections between modules.

CLO 9: Students will explore the process of hardware synthesis, which converts HDL descriptions into gate-level representations suitable for implementation on programmable logic devices. They will learn synthesis techniques and optimization strategies to achieve desired design performance and resource utilization.

CLO 10: Students will learn how to realize combinational logic functions and implement synchronous sequential logic systems on PLD digital devices that typically use logic arrays as hardware platform, logic storage and programmable input output (I/O) capability.

5. Weekly Experiment Schedule:

Week	Experiment Title	Tools / Software Used	Main Objective
1, 2	Experiment (1): Synthesis and	Proteus 8.13	Understand the basic

	Understanding the Operation of Latch, D Flip-Flop, J-K Flip-Flop, and T Flip-Flop	software	concepts of flip-flops as elementary basic units of sequential logic circuits.
3, 4	Experiment (2): Design and Understanding the Operation of Shift Registers and Synchronous Counter Logic Circuits	Proteus 8.13 software	Understand the design and operation of shift registers and synchronous counter logic circuits.
5, 6	Experiment (3): Using (Xilinx, Altera, or Intel FPGAs) Simulator as a Modeling Design, Synthesis, and Analysis Tool for Modeling Basic Combinational Components: Logic Gates, Multiplexers in VHDL	Xilinx ISE 14.7 software	Introduce students to using the Xilinx ISE 14.7 VHDL simulator.
7, 8	Experiment (4): Modeling Basic Combinational Components: Decoders and Encoders in VHDL	Xilinx ISE 14.7 software	Implementing logic functions using decoder and encoder logic circuits using VHDL.
9, 10	Experiment (5): Structural VHDL	Xilinx ISE 14.7 software	Writing the structural VHDL code using component
11, 12	Experiment (6): Modeling Basic Sequential Components: Latches, Flip-Flops, Shift Registers, Counters in VHDL	Xilinx ISE 14.7 software	Write VHDL code descriptions that implements the Latches, Flip-Flops, Shift Registers, Counters in VHDL.
13, 14	Experiment (7): VHDL State Machine	Xilinx ISE 14.7 software	Write VHDL code descriptions that implements the state machines.

15	Final Exam		

6. Tools and Equipment Used:

- Xilinx ISE 14.7 software
- Proteus 8.13 software
- Spartan-3 FPGA
- Spartan-3E FPGA
- Wires
- Voltmeter

7. Safety Guidelines:

- Ensure the power is disconnected when connecting devices.

Do not touch electrical outlets or main device components without the supervisor’s permission.

Maintain quiet and organize cables and devices to avoid accidents.

8. Evaluation Method:

Percentage	Evaluation Item
10%	Reports
6%	Projects/Lab
10%	Final Exam

9. References and Sources:

- Modern digital design by Richard S. Sandige (McGraw-Hill)
- Voinci A. pedroni, "Circuit design with VHDL", MIT press, Cambridge, London 2004.
- Thom A.S. "digital with CPLA application and VHDL.
- Introduction to Logic Design, 3rd edition, Alan Marcovitz, McGraw-Hill, 2010.

10. Attachments:

- Experiment Report Template
- Experiment Assignment

Description of the Analog Electronics Laboratory

1. General Information:	
Laboratory Name:	**Analog Electronics Lab**
Associated Course Name:	**Analog Electronics**
Department:	**Computer Engineering**
Weekly Lab Hours:	**6**
Number of Weeks in the Semester:	**15**
Academic Level:	**2nd level**
Lab Supervisor:	**Dr. Rabee M. Hagem**

2. General Description of the Laboratory:

The Analog Electronics Lab is an experimental environment used to teach and apply the principles of analog electronics, which allows students to measure continuous values such as electric current or voltage. It includes a lab that teaches students how to design electronic circuits using analog components such as motors, capacitors, transistors, diodes, and process inductors.

3. Laboratory Objectives:

● In the Analog Electronics Lab, numerous hands-on experiments are conducted that help students understand how circuits work and how analog components respond to different signals. Students can measure currents and voltages and analyze the characteristics of the generated waves using devices such as multimeters, oscilloscopes, signal generators, and frequency meters.

4. Learning Outcomes:

By the end of the lab, the student should be able to implement the following circuits:

● Diode clipping, diode clamping, and rectifier circuits.

● Transistor amplifier circuits (CC, CB, CE).

● Op amp circuits.

● Basic op amp amplification circuits (voltage follower, adder, subtractor, differential, and integrator circuits).

5. Weekly Experiment Schedule:

Week	Experiment Title	Tools / Software Used	Main Objective
1	Introduction to Diode characteristics (PN junction) diode.	Kl-200 linear circuit lab Experiment module: Kl-23001 kit digital multimeter, oscilloscope and signal generator and basic hand tools.	Understanding the characteristics of each type (kind) of diode device. Recognize the specification of each type (kind) of diode device. Learning how to test the characteristics of each type.
2	Diode clipping circuit.	Kl-200 linear circuit lab Experiment module: Kl-23001 kit digital multimeter, oscilloscope and signal generator and basic hand tools.	Understanding the operation principle of diode clipping circuit
3	Diode clamping circuit.	Kl-200 linear circuit lab Experiment module: Kl-23001 kit digital multimeter, oscilloscope and signal generator and basic hand tools.	Understanding the operation principle of diode clamping circuit
4	Rectifying circuit.	Kl-200 linear circuit lab Experiment module: Kl-23021 kit digital multimeter,	Understanding the principles and features of half-wave, full-wave and bridge rectifier.

		oscilloscope and signal generator and basic hand tools.	
5	Transistor characteristics.	Kl-200 linear circuit lab Experiment module: Kl-23002 kit digital multimeter, oscilloscope and signal generator and basic hand tools.	Understanding the structure and symbols of the transistors and the characteristics of the transistors
6	Transistor amplifying circuit.	Kl-200 linear circuit lab Experiment module: kl-23003 kit digital multimeter, oscilloscope and signal gererator and basic hand tools.	Understanding the basic characteristics of each amplifying ciruit and the meaning of three modes of operation of the transistor (CE, CC, CB amplification)
7	Transistor amplifying circuit common emitter amplifier (CE).	Kl-200 linear circuit lab Experiment module: kl-23003 kit digital multimeter, oscilloscope and signal gererator and basic hand tools.	Understanding the characteristics of common emitter amplifier ciruit transistor (CE).
8	Emitter self-bias.	Kl-200 linear circuit lab Experiment module: kl-23003 kit digital multimeter, oscilloscope and signal generator and basic hand tools.	Understanding the characteristics Emitter self-bias

9	Transistor amplifying circuit common base amplifier (CB).	Kl-200 linear circuit lab Experiment module: kl-23003 kit digital multimeter, oscilloscope and signal generator and basic hand tools.	Understanding the characteristics common base amplifier (CB)
10	Transistor amplifying circuit common collector amplifier (CC).	Kl-200 linear circuit lab Experiment module: kl-23003 kit digital multimeter, oscilloscope and signal generator and basic hand tools	Understanding the characteristics common collector amplifier (CC).
11	Multistage amplifying circuit.	Kl-200 linear circuit lab Experiment module: kl-23005 kit digital multimeter, oscilloscope and signal generator and basic hand tools	Understanding the principle of amplifiers with various types of coupling, OTL, OCL amplifier circuits.
12	basic characteristics of OP AMP.	Kl-200 linear circuit lab Experiment module: kl-23012 kit digital multimeter, oscilloscope and signal generator and basic hand tools	Understanding the basic characteristics of OP AMP ZI, ZO, BW and the adjustment method for offset voltage of OP AMP

13	Basic amplification of OP AMP (voltage follower, adder and subtractor).	Kl-200 linear circuit lab Experiment module: kl-23013 kit digital multimeter, oscilloscope and signal generator and basic hand tools	Understanding the basic characteristics of OP AMP (voltage follower, adder and subtractor).
14	Basic amplification of OP AMP (differential circuit and integrator circuit).	Kl-200 linear circuit lab Experiment module: kl-23013 kit digital multimeter, oscilloscope and signal generator and basic hand tools	Understanding the basic characteristics of OP AMP (differential circuit and integrator circuit).
15	Preparatory week before the final Exam	All equipments	Comprehensive review of all lab topics

6. Tools and Equipment Used:

• Linear Circuits Lab Kl-200

• Experiment Unit: Kl-23013 Kit

• Experiment Unit: Kl-23012 Kit

• Experiment Unit: Kl-23005 Kit

• Experiment Unit: Kl-23003 Kit

• Experiment Unit: Kl-23002 Kit

• Experiment Unit: Kl-23001 Kit

• Digital multimeter, oscilloscope, signal generator, and basic hand tools

7. Safety Guidelines:

- Ensure the power is off when connecting devices.
- Do not touch electrical outlets or network components without the supervisor's permission.
- Remain calm and organize cables to avoid accidents.
- Use the simulator for routing experiments before testing on real devices.

8. Evaluation Method:

Percentage	Evaluation Item
Attendance and Participation	10%
Weekly Lab Reports	30%
Quizzes	20%
Final Practical Exam	30%
Practical Project	10%

9. References and Sources:

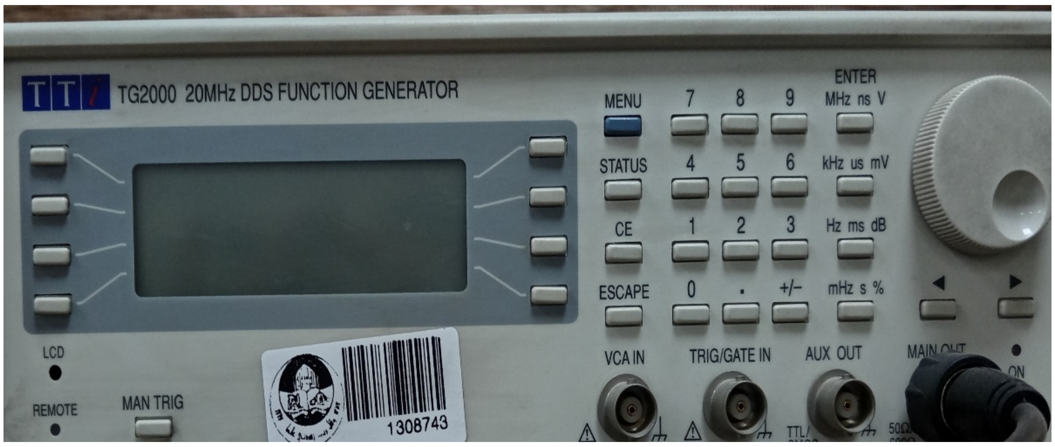
- Electronic Devices, Thomas L. Floyd, 7th edition, 2017

10. Attachments:

Analog Electronics Lab

Linear Circuits Lab Kl-200:

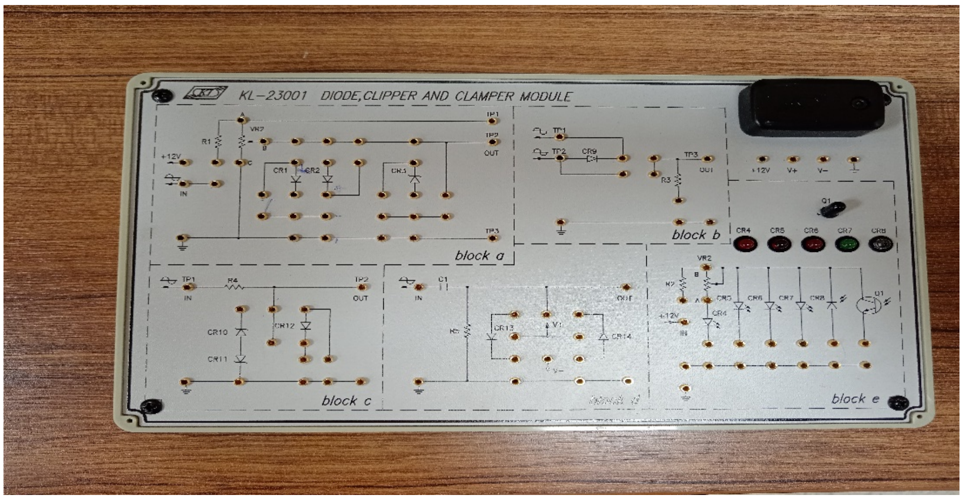
Signal generator:



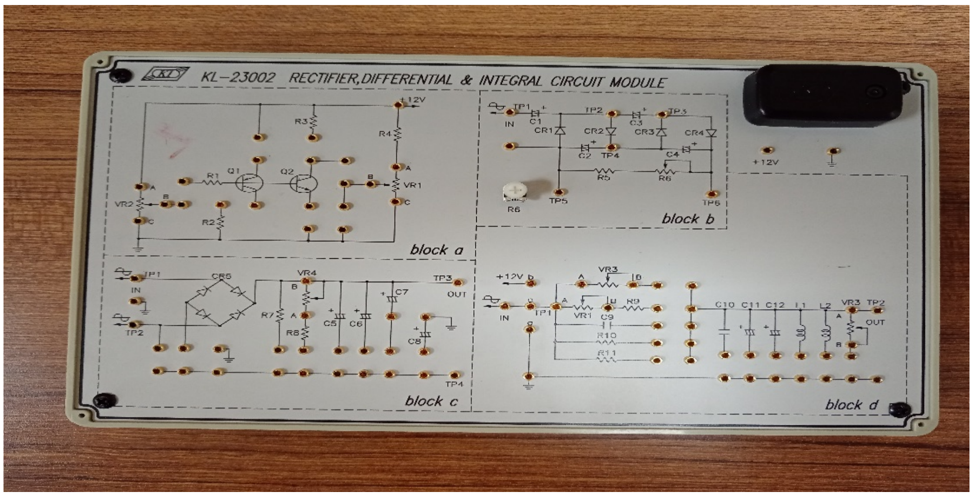
Oscilloscope:

digital multimeter:

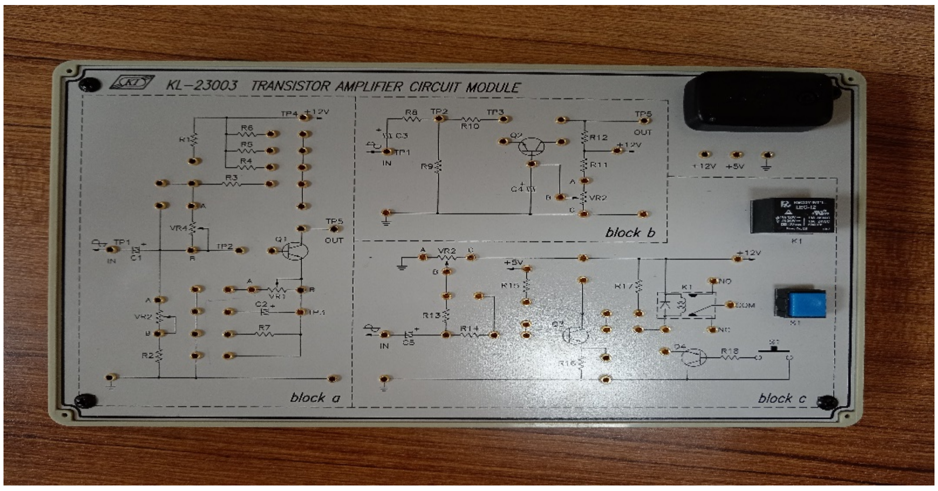
Experiment Unit: Kl-23001 Kit:



Experiment Unit: Kl-23002 Kit:



Experiment Unit: Kl-23003 Kit:



Description of the Digital Electronics Laboratory

1. **General Information:**	
Laboratory Name:	Digital Electronics
Associated Course Name:	Digital Electronics
Department:	Computer Engineering
Weekly Lab Hours:	3 H
Number of Weeks in the Semester:	15
Academic Level:	Second Stage
Lab Supervisor:	Modhar Ahmed Hammoudy

2. General Description of the Laboratory:

The Digital Electronics Lab aims to reinforce the theoretical concepts covered in the Digital Electronics course lectures through practical applications using the tools necessary for analyzing and designing digital circuits and systems. The lab provides an interactive environment that enables students to gain a solid foundation in both digital and electronic fields.

3. Laboratory Objectives:

- Train students in designing and developing digital circuits and systems.
- Identify all types of digital electronic circuits and the differences between them.
- Use basic concepts of electrical and electronic analysis to determine the power consumption, number of load circuits, and logic voltage levels of a logic gate.
- Select the appropriate logic design from various types of logic gate families.
- Be able to evaluate the electrical and logical quantities of any digital logic circuit.

4. Learning Outcomes:

By the end of the lab, the student should be able to:

- Name all types of digital electronic circuits and the differences between them.
- Select an appropriate logic design from the various types of logic gate families.
- Design a new digital logic circuit to perform a specific task.
- Document experiments and prepare professional reports.
- Work as part of a team to solve digital systems problems.

5. Weekly Experiment Schedule:

Week	Experiment Title	Tools / Software Used	Main Objective
1	Introduction to LTSpice	LTSpice	Learn about the LTSpice simulation environment and its use in the analysis and design of digital electronic circuits.
2	Design Logic Gates Using Resistor Diode Logic (RDL)	LTSpice	Build and simulate simple logic circuits that use resistors and diodes to implement logic functions.
3	Design Logic Gates Using Resistor Transistor Logic (RTL)	LTSpice	Design logic circuits that use resistors and transistors to produce basic gates such as AND, OR, and NOT.
4	Design Logic Gates Using Diode Transistor Logic (DTL)	LTSpice	Simulate logic circuits that use diodes for coupling and transistors for amplification to implement logic functions.
5	Design Logic Gates Using Transistor Transistor Logic (TTL)	LTSpice	Build logic circuits that rely solely on transistors to perform fast and efficient logic operations.
6	Logic Gate Design Using LTSpice (Emitter-Coupled Logic ECL, I2L)	LTSpice	Explore the design of high-speed logic gates using ECL and I2L techniques.
7	Logic Gate Design Using (NMOS and PMOS Logic	LTSpice	Implement and simulate logic gates using NMOS and PMOS transistors.

	Circuits)		
8	Logic Gate Design Using (Complementary Metal Oxide CMOS Logic Circuits)	LTSpice	Design power-efficient logic gates using CMOS technology, which combines NMOS and PMOS.
9	The term exam	LTSpice	comprehensively assesses the student's skills in designing and analyzing logic circuits using the techniques covered in the lab.
10	Logic Circuit Design Using (Sequential MOS Logic Circuits)	LTSpice	Construct and simulate sequential logic circuits using MOS transistors to implement memory functions.
11	Design logic circuits using (Regenerative Logic Circuits).	LTSpice	Analyze and design feedback-based logic circuits to achieve stability and binary logic.
12	Design logic circuits using (Semiconductor Memories).	LTSpice	Learn about the structure and operation of semiconductor memory circuits such as RAM and ROM and design them using simulation tools.
13	Design logic circuits using (Semiconductor Memories).	LTSpice	Learn about the structure and operation of semiconductor memory circuits such as RAM and ROM and design them using simulation tools.
14	Comprehensive review and discussion of the practical project.	LTSpice	All tools integrate the acquired skills into a final project.
15	Final exam:	LTSpice	A comprehensive assessment of the student's skills in designing and analyzing logic circuits using the techniques covered in the lab.

6. Tools and Equipment Used:

• Laboratory computers

• LTSpice software

7. Safety Guidelines:

• Use your computer safely and correctly:

Sit in a comfortable position, avoid eye strain by using adequate lighting, and take short breaks during extended periods of program work.

• Adhere to the instructions of your supervisor or instructor carefully:

Follow the instructional steps carefully when designing circuits and avoid running simulations without reviewing the connections and parameters to avoid false results or confusion during the experiment.

• Maintain and organize your work files:

Make sure to save your experiment files with an appropriate name and in an organized location on your computer for easy review later. Avoid deleting or modifying your colleagues' files without permission.

8. Evaluation Method:

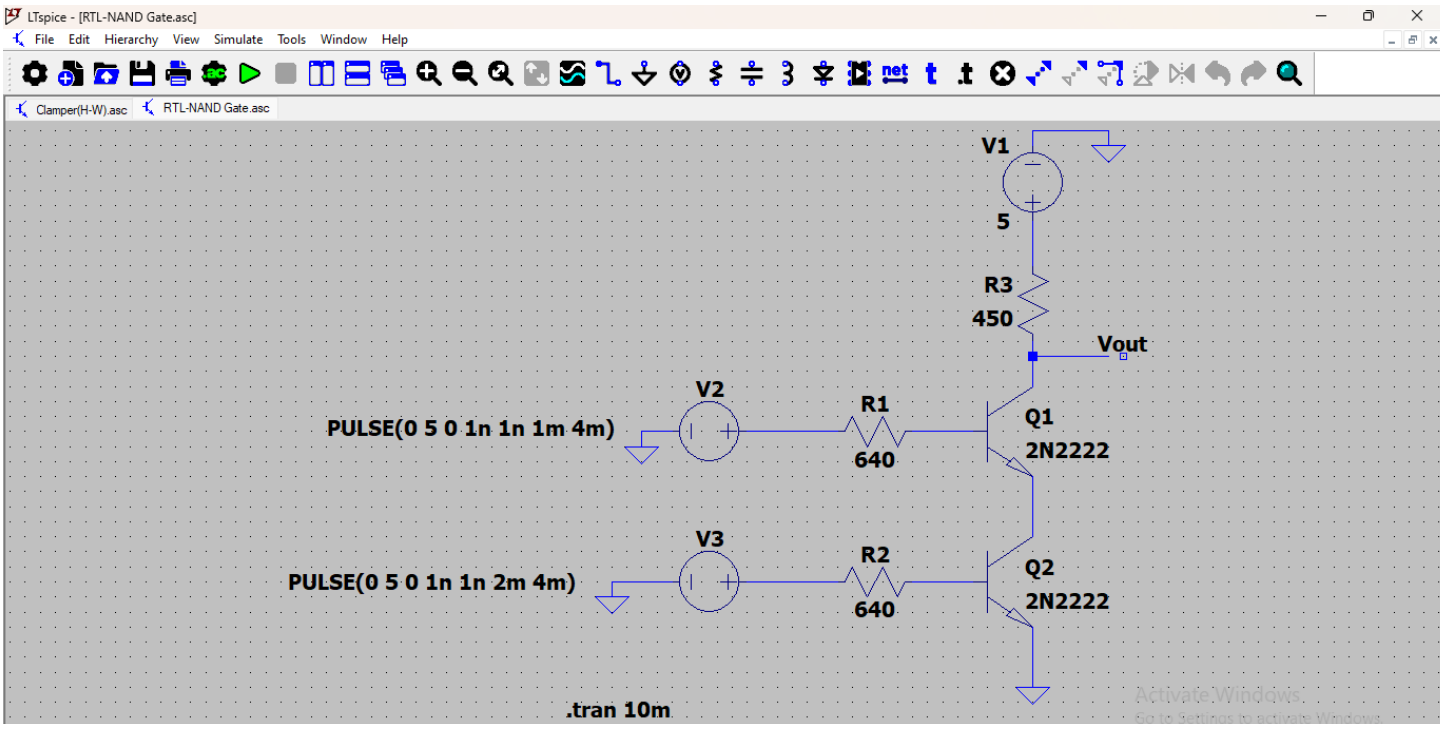
Percentage	Evaluation Item
10%	Attendance and Participation
30%	Semi-Final Exam
20%	Quizzes
30%	Final Practical Exam
10%	Practical Project

9. References and Sources:

- “Digital Integrated Circuits Analysis and Design” by: John E. Ayers.2004
- 1“ .Analysis and Design of Digital Integrated Circuits” by: David A. Hodges. 1988
- 2.Lab Manual , LTSPICE Design Tool.

10. Attachments:

- Weekly work plan
- Instructions for using LTSpice



Description of the Object Oriented Programming Laboratory

1. General Information:	
Laboratory Name:	Object Oriented Programming (Lab 312)
Associated Course Name:	Object Oriented Programming
Department:	Computer Engineering
Weekly Lab Hours:	3 Hours
Number of Weeks in the Semester:	15 Weeks
Academic Level:	Second Year
Lab Supervisor:	Asst. Prof. Dr. Turkan Ahmed Khaleel & Sahar Khalid Ahmed +Hiba Dhiya Ali

2. General Description of the Laboratory:

The Object Oriented Programming (OOP) Lab aims to provide students with practical experience in designing and implementing software using object-oriented principles in C++. The lab supports the theoretical part of the course and emphasizes modular design, code reuse, and scalability. Students will work on hands-on tasks involving encapsulation, inheritance, polymorphism, and abstraction to solve programming problems effectively.

3. Laboratory Objectives:

- Reinforce understanding of object-oriented programming concepts through practical application.
- Enable students to implement C++ programs using OOP principles such as classes, objects, encapsulation, inheritance, and polymorphism.
- Develop students’ ability to design maintainable and scalable software.
- Train students to apply abstract classes and interfaces in real-world scenarios.

4. Learning Outcomes:

By the end of the lab, students should be able to:

- Analyze and solve programming problems using OOP in C++.
- Design and implement object-oriented programs with a focus on reusability and maintainability.

- Apply encapsulation, inheritance, and polymorphism in class hierarchies.
- Implement abstract classes and method overloading.
- Handle exceptions and perform code debugging.
- Collaborate effectively in team-based software projects.

5. Weekly Experiment Schedule:

Week	Experiment Title	Tools / Software Used	Main Objective
1	Introduction and Programming Review	C++ / Code::Blocks	Setting up the environment and reviewing basics
2	Objects and Classes	C++ / Code::Blocks	Understanding object structures
3	Data Abstraction	C++ / Code::Blocks	Separating interface from implementation
4	Encapsulation and Hiding	C++ / Code::Blocks	Securing class data
5	Constructors and Destructors	C++ / Code::Blocks	Managing object lifecycle
6	Class Methods	C++ / Code::Blocks	Defining and calling methods
7	Method Overloading	C++ / Code::Blocks	Creating multiple versions of methods
8	Inheritance	C++ / Code::Blocks	Creating class hierarchies
9	Polymorphism	C++ / Code::Blocks	Allowing flexible and dynamic behavior
10	Abstract Classes	—	Defining generic templates for subclasses
11	Abstract Methods	C++ / Code::Blocks	Forcing subclass implementation
12	Exception Handling	C++ / Code::Blocks	Handling runtime errors
13	Project Presentation / Support	C++ / Code::Blocks	Student assessment and help

14	Review and Final Preparation	—	General review and final project assistance

6. Tools and Equipment Used:

- Lab Computers
- C++ Programming Software (Code::Blocks)

7. Safety Guidelines:

- Ensure power is off when connecting hardware
- Do not touch power sockets or networking hardware without supervisor approval
- Maintain silence and organize cables to avoid accidents

8. Evaluation Method:

Percentage	Evaluation Item
Assessment Component	5%
Weekly Lab Reports	5%
Quizzes	10%
Final Practical Exam	10%

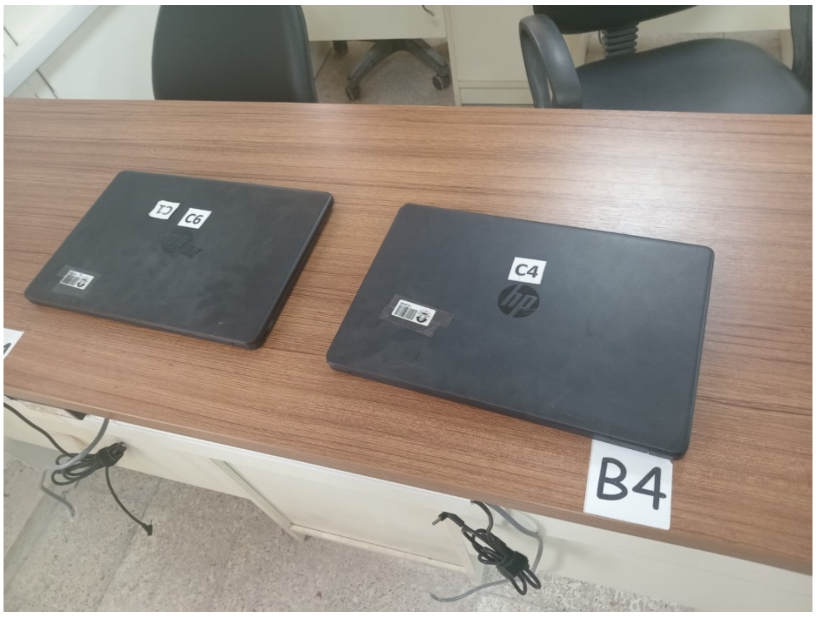
9. References and Sources:

Object-Oriented Programming in C++, Fourth Edition, by , Robert Lafore , Waite Group,Sams Publishing,2002.

C++ programming an object oriented approach, by Admin , 2022 .

10. Attachments:

Photos


C1
C6
C4
B4

Description of the Data Structures Laboratory

1. General Information:	
Laboratory Name:	Data Structures Laboratory (Lab 312)
Associated Course Name:	Data Structures
Department:	Computer Engineering
Weekly Lab Hours:	3 Hours
Number of Weeks in the Semester:	15 Weeks
Academic Level:	Second Year
Lab Supervisor:	Asst. Prof. Dr. Turkan Ahmed Khaleel + Asst. Lecturer Heba Diaa Al-Nu'ma

2. General Description of the Laboratory:

The Data Structures Lab aims to reinforce the theoretical concepts covered in the Data Structures course through practical applications and programming. The lab provides an interactive environment where students can gain hands-on experience in designing, implementing, and analyzing various data structures using the C++ language and the Code::Blocks development environment.

3. Laboratory Objectives:

- Reinforce theoretical understanding through practical application
- Enable students to connect theoretical data structure concepts with practical C++ implementation
- Develop programming skills using different data structures
- Train students to write programs using arrays, linked lists, stacks, queues, trees, graphs, and hashing
- Enable students to analyze and select appropriate data structures for given problems
- Encourage analytical thinking and problem decomposition into solvable components using proper data structures
- Enable students to combine multiple data structures in a comprehensive, practical program (useful for graduation projects or real-world applications)

4. Learning Outcomes:

By the end of the lab, students should be able to:

- Understand the basic concepts of data structures
- Define and implement linear and non-linear structures (such as arrays, stacks, queues, linked lists, trees, and graphs)
- Write C++ programs to implement data structures
- Design and implement appropriate structures to solve specific programming problems
- Apply object-oriented programming (OOP) principles
- Use classes and objects to organize code and encapsulate data
- Apply recursion to solve classical problems
- Implement efficient searching and sorting algorithms
- Execute and analyze algorithms like Binary Search, Bubble Sort, Selection Sort, etc.
- Analyze the performance and time complexity of data structures and algorithms
- Represent complex relationships using trees and graphs
- Build and manipulate binary trees (insertion, deletion, traversal)
- Represent graphs using adjacency matrices or adjacency lists
- Understand and apply hashing techniques

5. Weekly Experiment Schedule:

Week	Experiment Title	Tools / Software Used	Main Objective
1	Implementing Programs: Arrays, Functions, Structures, and Classes	C++ / Code::Blocks	Review basic programming concepts using C++
2	Recursion in Programming: Factorial and Classical Problems	C++ / Code::Blocks	Understand recursion and apply it to problem solving
3	Stacks (ADT): Implementation using Arrays and Structures	C++ / Code::Blocks	Create a Stack using arrays and apply push/pop operations
4	Stacks: Implementation using Linked Lists	C++ / Code::Blocks	Design a dynamic Stack using linked lists
5	Queues (ADT):	C++ / Code::Blocks	Implement a Queue

	Implementation using Arrays and Structures		using arrays and analyze its behavior
6	Queues: Implementation using Linked Lists	C++ / Code::Blocks	Design a dynamic Queue using linked lists
7	Circular Queues: Implementation using Arrays and Structures	C++ / Code::Blocks	Implement a Circular Queue and solve overflow issues
8	Circular Queues: Implementation using Linked Lists	C++ / Code::Blocks	Design a Circular Queue using linked lists
9	Tree Traversals: Binary Tree Applications	C++ / Code::Blocks	Implement Inorder, Preorder, and Postorder traversal algorithms
10	Midterm Exam	—	Practical assessment of skills covered in the first half
11	Graph Theory	C++ / Code::Blocks	Represent graphs and apply BFS / DFS algorithms
12	Hashing Techniques	C++ / Code::Blocks	Implement basic hashing and understand collision handling
13	Sorting and Searching Techniques	C++ / Code::Blocks	Apply algorithms like Bubble, Selection, Binary Search
14	Final Exam	—	Comprehensive assessment of the full course content

6. Tools and Equipment Used:

- Lab Computers
- C++ Programming Software (Code::Blocks)

7. Safety Guidelines:

- Ensure power is off when connecting hardware
- Do not touch power sockets or networking hardware without supervisor approval
- Maintain silence and organize cables to avoid accidents
- Use simulators for testing before using real hardware (if applicable)

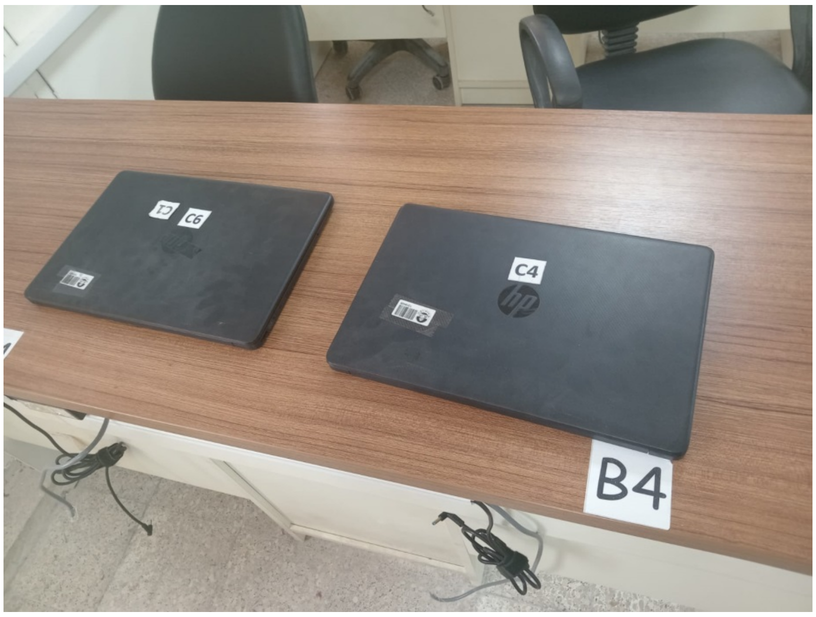
8. Evaluation Method:

Percentage	Evaluation Item
Assessment Component	5%
Weekly Lab Reports	5%
Quizzes	10%
Final Practical Exam	10%

9. References and Sources:

- Data Structures Using C++ (Second Edition) by D.S. Malik, 2012
- Data Structures and Algorithms in C++ (4th Edition) by Mark A. Weiss, 2014

10.Attachments:


C6
C4
B4