Computer Organization and Architecture Lab Manual

B.Tech. , Semester -III

Subject Code: BCS-352

Session: 2024-25, ODD Semester

Dronacharya Group of Institutions
Plot No. 27, Knowledge Park-3, Greater Noida, Uttar Pradesh 201308
Affiliated to
Dr. A P J Abdul Kalam Technical University
Lucknow, Uttar Pradesh 226031

Table of Contents

Vision of the Institute

Instilling core human values and facilitating competence to address global challenges by providing Quality Technical Education.

Mission of the Institute

- M1 - Enhancing technical expertise through innovative research and education, fostering creativity and excellence in problem-solving.
- M2 - Cultivating a culture of ethical innovation and user-focused design, ensuring technological progress enhances the well-being of society.
- M3 - Equipping individuals with the technical skills and ethical values to lead and innovate responsibly in an ever- evolving digital landscape.

Vision of the Department

- Promoting technologists by imparting profound knowledge in information technology, all while instilling ethics through specialized technical education.

Mission of the Department

- Delivering comprehensive knowledge in information technology, preparing technologists to excel in a rapidly evolving digital landscape.
- Building a culture of honesty and responsibility in tech, promoting smart and ethical leadership.
- Empowering individuals with specialized technical skills and ethical values to drive positive change and innovation in the tech industry.

Program Outcomes (POs)

PO1: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2: Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3: Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6: The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7: Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of engineering practice.

PO 9: Individual and teamwork: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10: Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11: Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12: Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Program Educational Objectives (PEOs)

- To enable graduates to think logically, pursue lifelong learning and will have the capacity to understand technical issues related to computing systems and to design optimal solutions.
- To enable graduates to develop hardware and software systems by understanding the importance of social, business and environmental needs in the human context.
- To enable graduates to gain employment in organizations and establish themselves as professionals by applying their technical skills to solve real world problems and meet the diversified needs of industry, academia and research.

Program Specific Outcomes (PSOs)

- To adapt to emerging technologies and develop innovative solutions for existing and newer problems.
- To create and apply appropriate techniques IT tools to complex engineering activities with an understanding of the limitations.
- To manage complex IT projects with consideration of the human, financial, ethical and environmental factors.

University Syllabus

1. Implementing HALF ADDER, FULL ADDER using basic logic gates
2. Implementing Binary -to -Gray, Gray -to -Binary code conversions.
3. Implementing 3-8 line DECODER.
4. Implementing 4x1 and 8x1 MULTIPLEXERS.
5. Verify the excitation tables of various FLIP-FLOPS.
6. Design of an 8-bit Input/ Output system with four 8-bit Internal Registers.
7. Design of an 8-bit ARITHMETIC LOGIC UNIT.
8. Design the data path of a computer from its register transfer language description.
9. Design the control unit of a computer using either hardwiring or microprogramming based on its register transfer language description.
10. Implement a simple instruction set computer with a control unit and a data path.

Course Outcomes (COs)

Upon successful completion of the course, the students will be able to

C.1: Design and verify combinational circuits (adder, code converter, decoder,multiplexer) using basic gates.
C.2: Design and verify various flip-flops.
C.3: Design I/O system and ALU.
C.4: Demonstrate a simple instruction set computer .

CO-PO Mapping:

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
C.1	3		3					2	2	2		2
C.2	3		3					2	2	2		2
C.3	3		3					2	2	2		2
C.4	2		3					2	2	2		

CO-PSO Mapping

	PSO1	PSO2	PSO3
C207.1	-	3	-
C207.2	-	3	-
C207.3	-	3	-
C207.4	-	3	-
C207	**-**	**3**	**-**

Course Overview

An important part of the undergraduate curriculum for IT students is coverage of computer organization. Typically, this is accomplished by both a lecture and lab course. The purpose of the lab course is to have the students develop practical design skills. Our computer organization laboratory course starts with traditional logic gate design, then gradually incorporates control unit, up to the point where students are building a simple instruction set computer. The laboratory course is scheduled for a two-hour time slot. There are ten lab experiments.

During the first set of Experiments, students become familiar with basic digital hardware by constructing simple combinational circuits, and learning troubleshooting skills. During the second set of Experiments, students become familiar with arithmetic logic unit and control unit. Finally, students apply their knowledge to the design a simple instruction set computer. Because students are constructing complete computer organization projects, the computer organization lab typically requires more effort than traditional laboratory courses.

List of Experiments mapped with COs

SLno.	*List of Experiments*	*Course Outcome*
1.	Implementing HALF ADDER, FULL ADDER using basic logic gates	CO.1
2.	Implementing Binary -to -Gray, Gray -to -Binary code conversions.	CO.1
3.	Implementing 3-8 line DECODER**.**	CO.1
4.	Implementing 4x1 and 8x1 MULTIPLEXERS.	CO.1
5.	Verify the excitation tables of various FLIP-FLOPS.	CO.2
6.	Design of an 8-bit Input/ Output system with four 8-bit Internal Registers.	CO.2
7.	Design of an 8-bit ARITHMETIC LOGIC UNIT.	CO.3
8.	Design the data path of a computer from its register transfer language description.	CO.4
9.	Design the control unit of a computer using either hardwiring or microprogramming based on its register transfer language description.	CO.4
10.	Implement a simple instruction set computer with a control unit and a data path.	CO.4

DOs and DON'Ts

Dos

1. Login-on with your username and password.
2. Log off the Computer every time when you leave the Lab.
3. Arrange your chair properly when you are leaving the lab.
4. Put your bags in the designated area.
5. Ask permission to print.

DON'Ts

1. Do not share your username and password.
2. Do not remove or disconnect cables or hardware parts.
3. Do not personalize the computer setting.
4. Do not run programs that continue to execute after you log off.
5. Do not download or install any programs, games or music on computer in Lab.
6. Personal Internet use chat room for Instant Messaging (IM) and Sites Strictly Prohibited.
7. No Internet gaming activities allowed.
8. Tea, Coffee, Water & Eatables are not allowed in the Computer Lab.

General Safety Precautions

Precaution (In case of Injury or Electric Shock)

1. To break the victim with live electric source .Use an insulator such as fire wood or plastic to break the contact. Do not touch the victim with bare hands to avoid the risk of electrifying yourself.
2. Unplug the risk of faulty equipment. If main circuit breaker is accessible, turn the circuit off.
3. If the victim is unconscious, start resuscitation immediately, use your hands to press the chest in and out to continue breathing function. Use mouth-to-mouth resuscitation if necessary.
4. Immediately call medical emergency and security. Remember! Time is critical; be best.

Precaution (In case of Fire)

1. Turn the equipment off. If power switch is not immediately accessible, take plug off.
2. If fire continues, try to curb the fire if possible by using the fire extinguisher or by covering it with a heavy cloth if possible isolate the burning equipment from the other surrounding equipment.
3. Sound the fire alarm by activating the nearest alarm switch located in the hallway.
4. Call security and emergency department immediately:

Guidelines to students for report preparation

All students are required to maintain a record of the experiments conducted by them. Guidelines for its preparation are as follows: -

1) All files must contain a title page followed by an index page. ***The files will not be signed by the faculty without an entry in the index page.***

2) Student's Name, Roll number and date of conduction of experiment must be written on all pages.

3) For each experiment, the record must contain the following

(i) Aim/Objective of the experiment

(ii) Equipment's required

(iii) Pre-experiment work (as given by the faculty)

(iv) Observation table

(v) Results/ output

Note:

1. Students must bring their lab record along with them whenever they come for the lab.

2. Students must ensure that their lab record is regularly evaluated.

Lab Assessment Criteria

An estimated 10 lab classes are conducted in a semester for each lab course. These lab classes are assessed continuously. Each lab experiment is evaluated based on 5 assessment criteria as shown in following table. Assessed performance in each experiment is used to compute CO attainment as well as internal marks in the lab course.

Grading Criteria	**Exemplary (4)**	**Competent (3)**	**Needs Improvement (2)**	**Poor (1)**
AC1: **Pre-Lab written work (for last lab class, this may be assessed through viva)**	Complete procedure with underlined concept is properly written	Underlined concept is written but procedure is incomplete	Not able to write concept and procedure	Underlined concept is not clearly understood
AC2: **Program Writing/ Modeling**	Assigned problem is properly analyzed, correct solution designed, appropriate language constructs/ tools are applied, Program/solution written is readable	Assigned problem is properly analyzed, correct solution designed, appropriate language constructs/ tools are applied	Assigned problem is properly analyzed & correct solution designed	Assigned problem is properly analyzed
AC3: **Identification & Removal of errors/ bugs**	Able to identify errors/ bugs and remove them	Able to identify errors/ bugs and remove them with little bit of guidance	Is dependent totally on someone for identification of errors/ bugs and their removal	Unable to understand the reason for errors/ bugs even after they are explicitly pointed out
AC4:Execution & Demonstration	All variants of input /output are tested, Solution is well demonstrated and implemented concept is clearly explained	All variants of input /output are not tested, However, solution is well demonstrated and implemented concept is	Only few variants of input /output are tested, Solution is well demonstrated but implemented concept is not clearly explained	Solution is not well demonstrated and implemented concept is not clearly explained

		clearly explained		
AC5:Lab Record Assessment	All assigned problems are well recorded with objective, design constructs and solution along with Performance analysis using all variants of input and output	More than 70 % of the assigned problems are well recorded with objective, design contracts and solution along with Performance analysis is done with all variants of input and output	Less than 70 % of the assigned problems are well recorded with objective, design contracts and solution along with Performance analysis is done with all variants of input and output	Less than 40 % of the assigned problems are well recorded with objective, design contracts and solution along with Performance analysis is done with all variants of input and output

LAB EXPERIMENTS

LAB EXPERIMENT 1

OBJECTIVE: Design and implementation of Half Adder and Full Adder.

EQUIPMENTS & COMPONENTS REQUIRED:

SL.No.	Equipment's	Specification	Quantity
1	Digital IC Trainer kit	-	1
2	Digital Multimeter		1

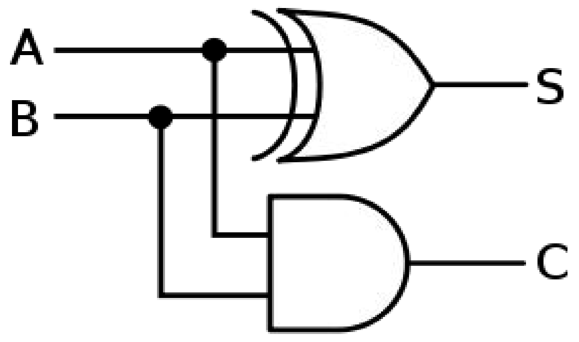
SL.No.	Components	Specification	Quantity
1	Digital ICs	7400, 7402, 7404, 7408, 7432, 7486.	1 each
2	Patch cords	-	6

BRIEF DESCRIPTION:

- To design and implement half adder using logic gates

HALF ADDER

INPUT A	INPUT B	OUTPUTS	
		S	**C**
0	**0**	**0**	**0**
0	**1**	**1**	**0**
1	**0**	**1**	**0**
1	**1**	**0**	**1**

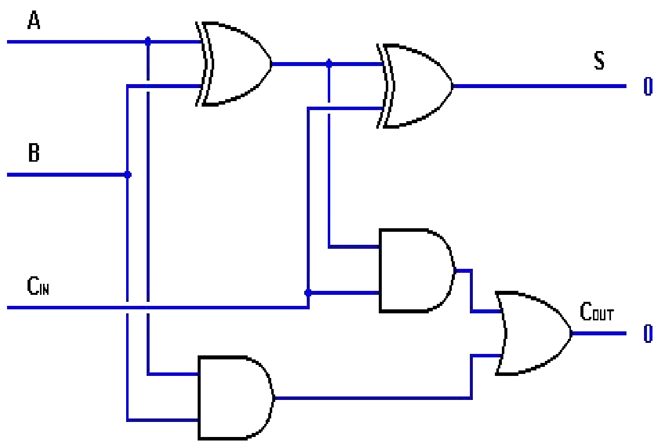


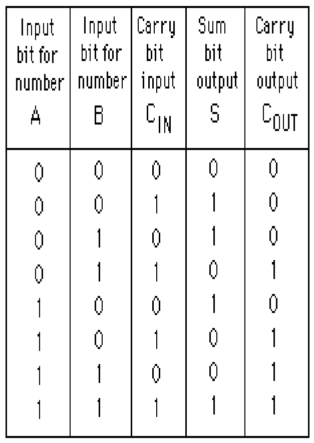
Circuit Diagram

Truth Table

- To design and implement full adder using logic gates.

FULL ADDER





Input bit for number A	Input bit for number B	Carry bit input C_{IN}	Sum bit output S	Carry bit output C_{OUT}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

CIRCUIT DIAGRAM **TRUTH TABLE**

PRE-EXPERIMENT QUESTIONS:-

1. Explain the truth table of half adder.
2. How many Ex-or and or or gate can be used to make a half adder?

PROCEDURE:-

- Identify the pins.
- Connect the circuit as per circuit diagram.
- Obtain outputs with various input combinations.
- Verify it with the Boolean function using truth table

POST-EXPERIMENT QUESTIONS:-

1. What are the applications of half adder?
2. What are the application of full adder?

LAB EXPERIMENT 2

OBJECTIVE: Design and implementation of Binary to Gray, Gray to Binary Code conversions

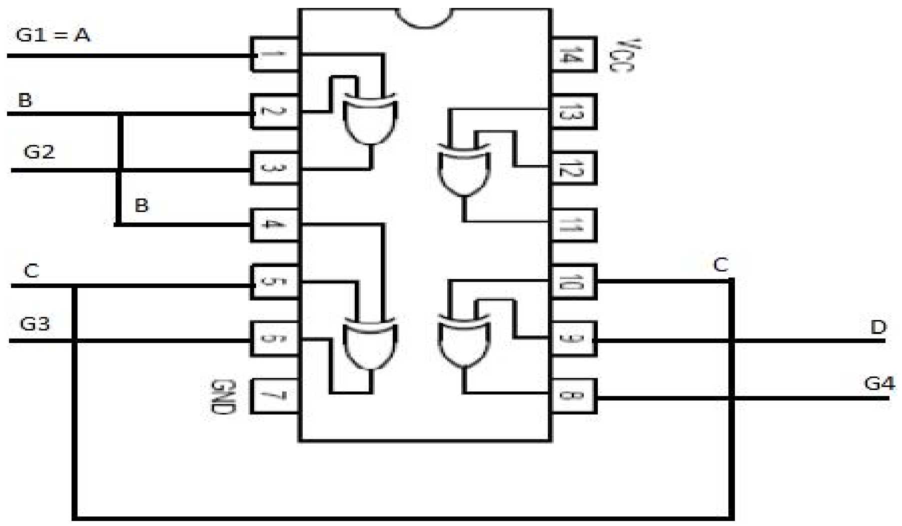
EQUIPMENTS & COMPONENTS REQUIRED:

SL. No.	Equipments	Specification	Quantity
1	Digital IC Trainer kit	-	1
2	Digital Multimeter		1

SL. No.	Components	Specification	Quantity
1	Digital ICs	7400, 7402, 7404, 7408, 7432, 7486.	1 each
2	Patch cords	-	6

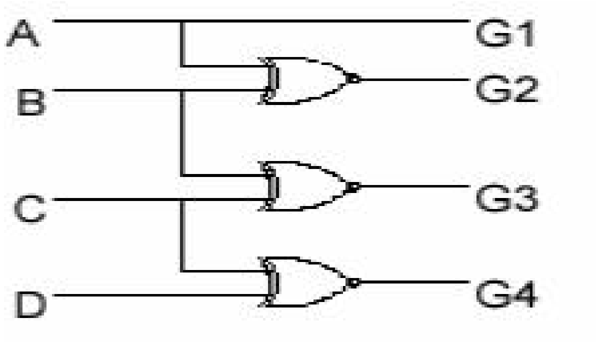
BRIEF DESCRIPTION:

a) To design and implement Binary to Gray Code conversions



Pin diagram of Binary to gray code converter using 7486 IC (ex-or Gate)

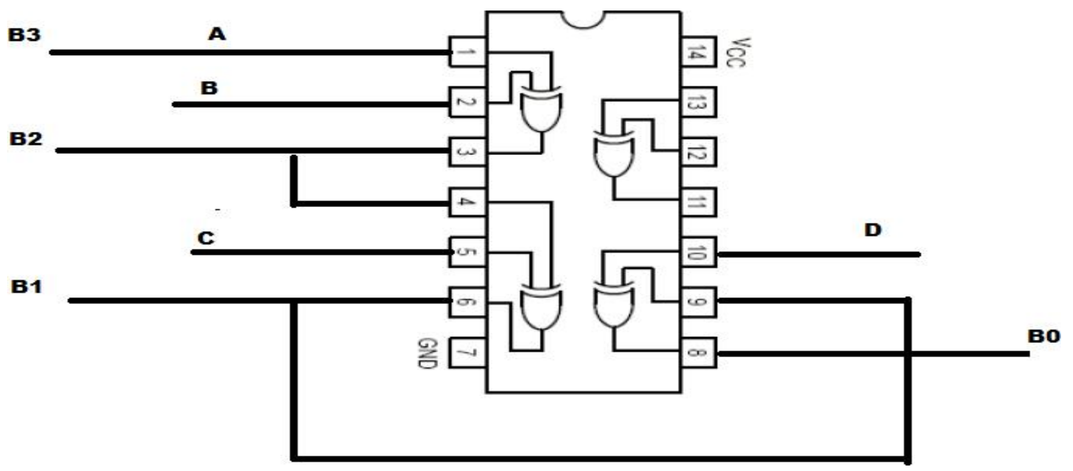
INPUTS				OUTPUTS			
A	B	C	D	G_4	G_3	G_2	G_1
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0



Circuit Diagram of Binary to Gray Code Converter

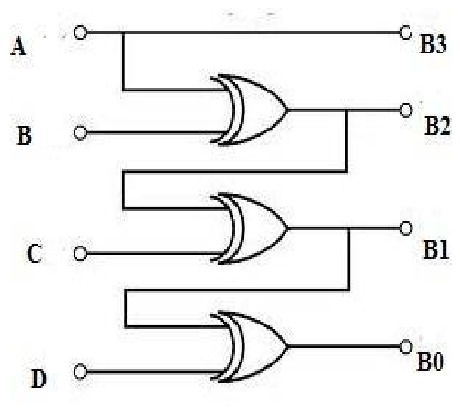
Truth Table

b) To design and implement Binary to Gray Code conversions



Pin diagram of Gray to Binary code converter using 7486 Ic(Exor Gate)

INPUTS				OUTPUTS			
A	B	C	D	B_3	B_2	B_1	B_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	1	0	0	1	0
0	0	1	0	0	0	1	1
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	1
0	1	0	1	0	1	1	0
0	1	0	0	0	1	1	1
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	1
1	1	1	1	1	0	1	0
1	1	1	0	1	0	1	1
1	0	1	0	1	1	0	0
1	0	1	1	1	1	0	1
1	0	0	1	1	1	1	0
1	0	0	0	1	1	1	1



Circuit Diagram for Gray to Binary Code Converter

TRUTH TABLE

PRE-EXPERIMENT QUESTIONS:-

1 What is a code converter.

2 Differentiate between translator and code converter.

PROCEDURE:-

- Collect the components necessary to accomplish this experiment.
- Plug the IC chip into the breadboard.
- Connect the supply voltage and ground lines to the chips. PIN7 = Ground and PIN14 = +5V.
- Make connections as shown in the respective circuit diagram.
- Connect the inputs of the gate to the input switches of the LED.
- Connect the output of the gate to the output LEDs.
- Once all connections have been done, turn on the power switch of the breadboard
- Operate the switches and fill in the truth table (Write "1" if LED is ON and "0" if L1 is OFF Apply the various combination of inputs according to the truth table and observe the condition of Output LEDs.

POST-EXPERIMENT QUESTIONS:-

1. What are the advantages of code converter?
2. What are the properties of gray code?

LAB EXPERIMENT 3

OBJECTIVE: Design and implementation of 2- 4 and 3-8 line decoder.

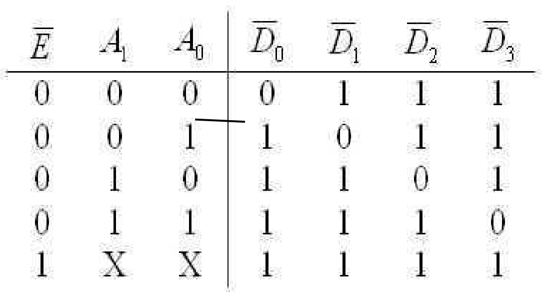
EQUIPMENTS & COMPONENTS REQUIRED:

SL.No.	Equipments	Specification	Quantity
1	Digital IC Trainer kit	-	1
2	Digital Multimeter		1

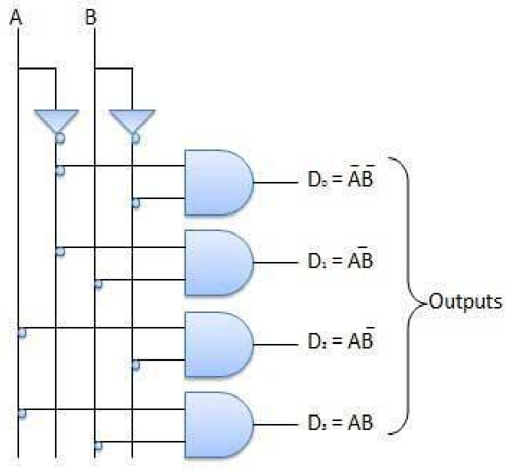
S	Components	Specification	Quantity
1	Digital ICs	7400, 7402, 7404, 7408, 7432, 7486.	1 each
2	Patch cords	-	6

BRIEF DESCRIPTION:

a) 2 to 4Decoder using logic gates:

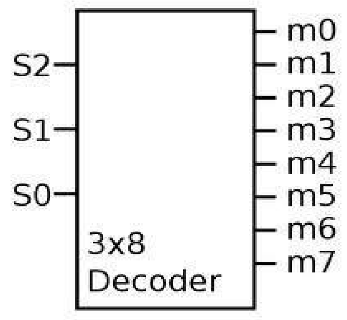


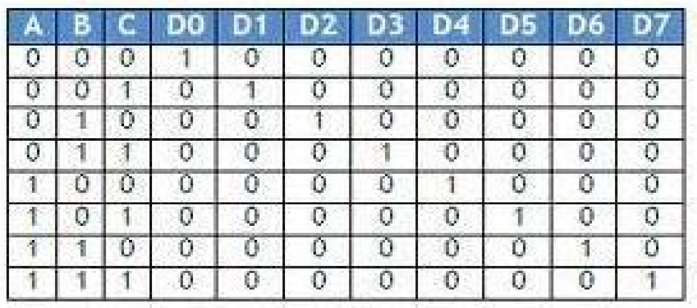
$\overline{E}$	A_1	A_0	$\overline{D_0}$	$\overline{D_1}$	$\overline{D_2}$	$\overline{D_3}$
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0
1	X	X	1	1	1	1



Truth Table **Logic Diagram**

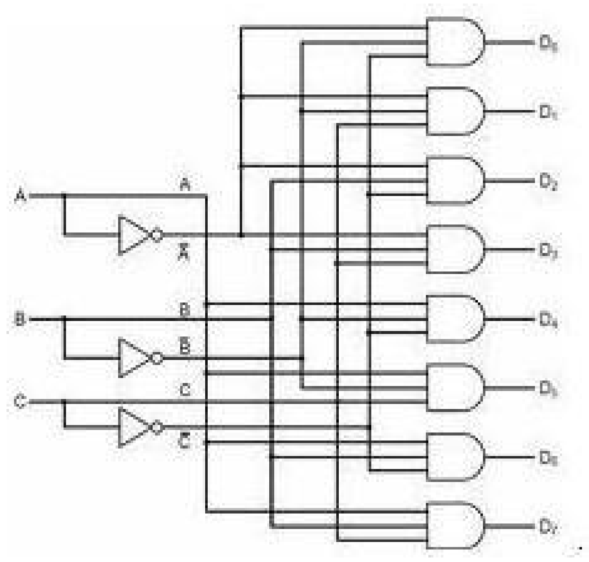
b) 3 to 8 decoder using logic gates:





A	B	C	D0	D1	D2	D3	D4	D5	D6	D7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

SYMBOL **TRUTH TABLE**



LOGIC DIAGRAM OF 3 TO 8 DECODER:

PRE-EXPERIMENT QUESTIONS:

1. Difference between Encoder and Decoder.
2. Explain the need of multiplexer.

PROCEDURE:

- Collect the components necessary to accomplish this experiment.
- Plug the IC chip into the breadboard.
- Connect the supply voltage and ground lines to the chips. PIN7 = Ground
- and PIN14 = +5V.
- Make connections as shown in the respective circuit diagram.
- Connect the inputs of the gate to the input switches of the LED.
- Connect the output of the gate to the output LEDs.
- Once all connections have been done, turn on the power switch of the breadboard

- Operate the switches and fill in the truth table (Write "1" if LED is ON and "0" if L1 is OFF Apply the various combination of inputs according to the truth tab alend observe the condition of Output LEDs.

POST EXPERIMENT QUESTIONS:

1. Design a 5 to 32 decoder using one 2 to 4 and four 3 to 8 decoder ic's.
2. Write a note on BCD to decimal decoder.

LAB EXPERIMENT 4

OBJECTIVE: Implementation Of 4x1 And 8x1 Multiplxer

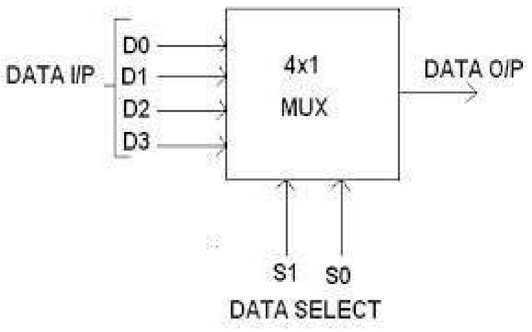
EQUIPMENTS & COMPONENTS REQUIRED:

SL.No.	Equipment's	Specification	Quantity
1	Digital IC Trainer kit	-	1
2	Digital Multimeter		1

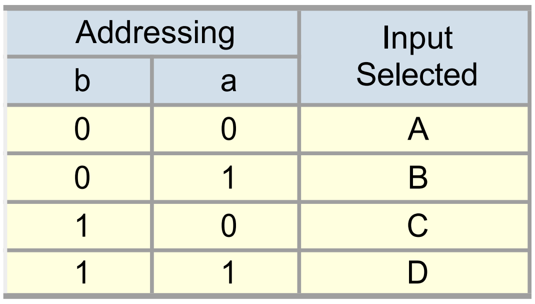
SL.No.	Components	Specification	Quantity
1	Digital ICs	7400, 7402, 7404, 7408, 7432, 7486.	1 each
2	Patch cords	-	6

BRIEF DESCRIPTION:

a) 4 to 1 MULTIPLEXERS:

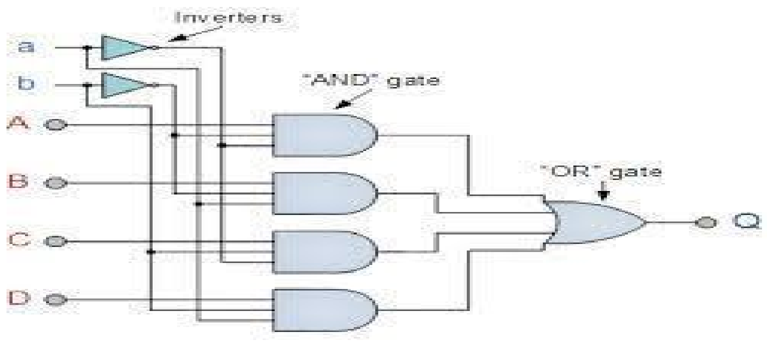


LOGIC DIAGRAM :



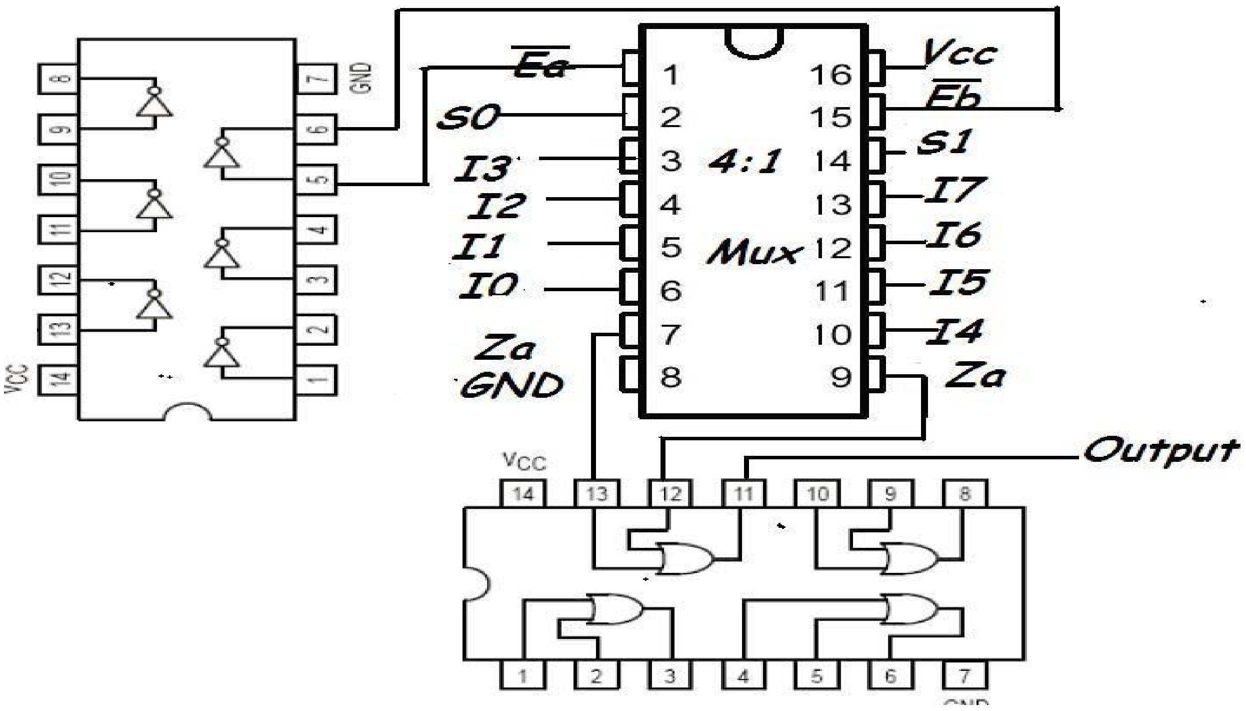
Addressing		Input Selected
b	a	
0	0	A
0	1	B
1	0	C
1	1	D

TRUTH TABLE:

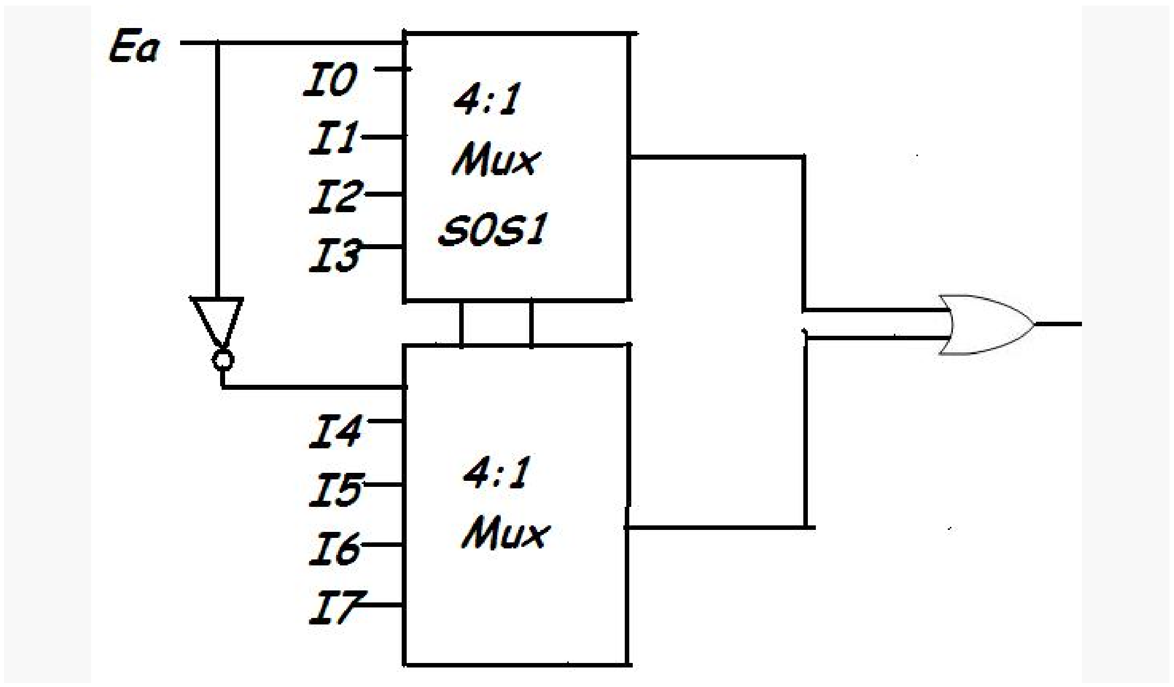


$$Q=\bar{a}\bar{b}A+\bar{a}bB+a\bar{b}C+abD$$

d) 8x1 Multiplexer



PIN DIAGRAM OF 8:1 MUX USING TWO 4:1 MUX



CIRCUIT OF 8:1 MUX USING DUAL 4:1 MUX

Select Lines			Inputs								Output		
E_a	S_0	S_1	I_o	I_1	I_2	I_3	I_4	I_5	I_6	I_7	Z_a	Z_b	Y
0	0	0	0	×	×	×	×	×	×	×	0	×	0
0	0	0	1	×	×	×	×	×	×	×	1	×	1
0	0	1	×	0	×	×	×	×	×	×	0	×	0
0	0	1	×	1	×	×	×	×	×	×	1	×	1
0	1	0	×	×	0	×	×	×	×	×	0	×	0
0	1	0	×	×	1	×	×	×	×	×	1	×	1
0	1	1	×	×	×	0	×	×	×	×	0	×	0
0	1	1	×	×	×	1	×	×	×	×	1	×	1
1	0	0		×	×	×	0	×	×	×	×	0	0
1	0	0	×	×	×	×	1	×	×	×	×	1	1
1	0	1	×	×	×	×	×	0	×	×	×	0	0
1	0	1	×	×	×	×	×	1	×	×	×	1	1
1	1	0	×	×	×	×	×	×	0	×	×	0	0
1	1	0	×	×	×	×	×	×	1	×	×	1	1
1	1	1	×	×	×	×	×	×	×	0	×	0	0
1	1	1	×	×	×	×	×	×	×	1	×	1	1

TRUTH TABLE OF 8:1 MUX USING DUAL 4:1 MUX

PRE EXPERIMENT QUESTIONS:-

1. Difference between Encoder and Decoder.
2. Explain the need of multiplexer.

PROCEDURE:-

- Collect the components necessary to accomplish this experiment.
- Plug the IC chip into the breadboard.
- Connect the supply voltage and ground lines to the chips. PIN7 = Ground and PIN14 = +5V.
- Make connections as shown in the respective circuit diagram.

- Connect the inputs of the gate to the input switches of the LED.
- Connect the output of the gate to the output LEDs.
- Once all connections have been done, turn on the power switch of the breadboard
- Operate the switches and fill in the truth table (Write "1" if LED is ON and "0" if L1 is OFF Apply the various combination of inputs according to the truth table and observe the condition of Output LEDs.

POST EXPERIMENT QUESTION:-

1. Design a 16 to 1 Multiplexer using one 4 to 1 Mux IC’S.

LAB EXPERIMENT 5

OBJECTIVE: Verify the excitation tables of various FLIP-FLOPS.

EQUIPMENTS &COMPONENTS REQUIRED:

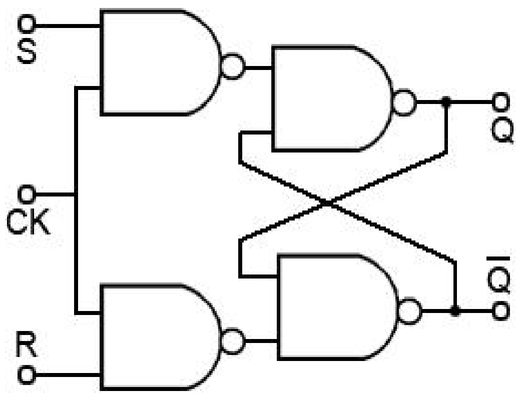
Digital IC trainer kit, IC 7400, IC 7410, IC 7473, IC 7474, IC 7476

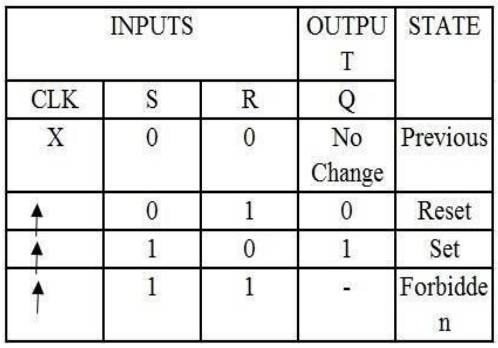
BRIEF DESCRIPTION:

Flip flops are the basic building blocks in any memory systems since its output will remain in its state until it is forced to change it by some means.

S R FLIP FLOP:

S and R stands for set and reset. There are four input combination possible at the inputs. But = =1 is forbidden since the output will be indeterminate.



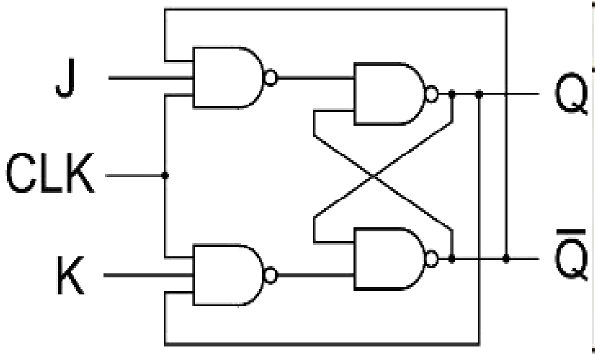


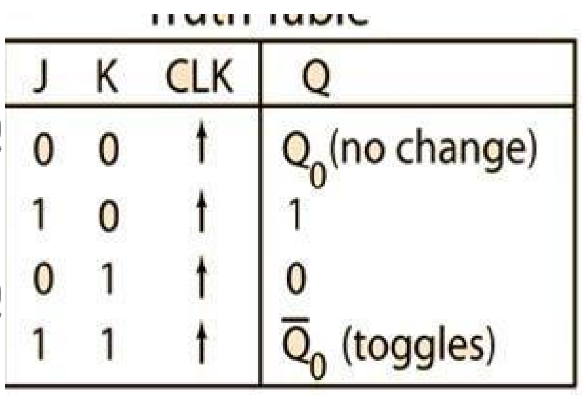
INPUTS			OUTPUT	STATE
CLK	S	R	Q	
X	0	0	No Change	Previous
↑	0	1	0	Reset
↑	1	0	1	Set
↑	1	1	-	Forbidden

LOGIC DIAGRAM OF R S FLIP FLOPS: **TRUTH TABLE:**

J K FLIP FLOP

The indeterminate output state of SR FF when S=R=1 is avoided by converting it to a JK FF. When flip flop is switched on its output state is uncertain. When an initial state is to be assigned two separate inputs called preset and clear are used. They are active low inputs



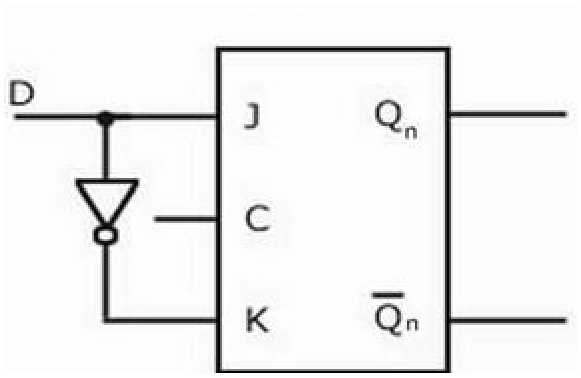


J	K	CLK	Q
0	0	↑	Q_0(no change)
1	0	↑	1
0	1	↑	0
1	1	↑	$\overline{Q}_0$ (toggles)

LOGIC DIAGRAM OF R S FLIP FLOPS **TRUTH TABLE**

D Flip flop

It has only one input called as D input or Data input. The input data is transferred to the output after a clock is applied. D FF can be derived from JK FF by using J input as D input and J is inverted and fed to K input.

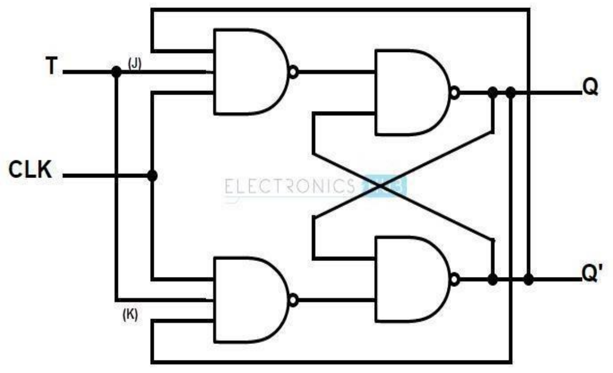


Q	D	Q(t+1)
0	0	0
0	1	1
1	0	0
1	1	1

LOGIC DIAGRAM **TRUTH TABLE**

T Flip Flop

T stands for Toggle. The output toggles when a clock pulse is applied. T FF can be derived from JK FF by shorting J and K input.



T	Q_n	Q_{n+1}
0	0	0
0	1	1
1	0	1
1	1	0

T FLIP FLOP LOGIC DIAGRAM **TRUTH TABLE**

PRE-EXPERIMENT QUESTIONS:-

1. What is propagation delay time?
2. How is JK FF made to toggle?

PROCEDURE:-

- Test all components and IC packages using digital IC tester and multimeter
- Set up FF using Gates and verify their truth tables
- Verify the Truth tables of 7473, 7474, and 7476 ICs

POST-EXPERIMENT QUESTIONS:-

1. How many FF are in 7475 IC?
2. How many FF are required to produce a divide-by-128 device?

LAB EXPERIMENT6

OBJECTIVE: Design And Implementation Of Shift Registers (SISO & SIPO)

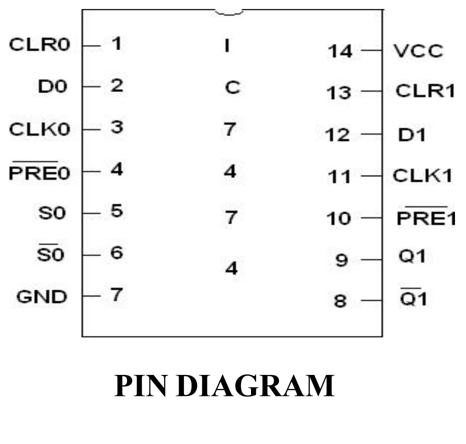
EQUIPMENTS & COMPONENTS REQUIRED:

SL.No.	Equipments	Specification	Quantity
1	Digital IC Trainer kit	-	1
2	Digital Multimeter	-	1

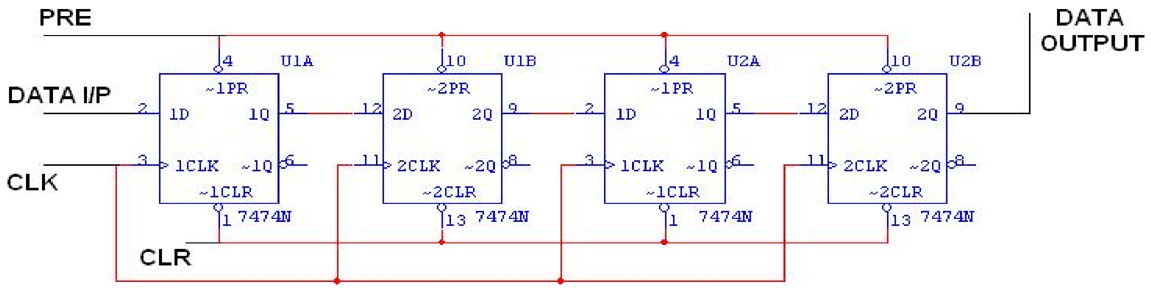
SL.No.	COMPONENT	SPECIFICATION	QTY.
1.	D FLIP FLOP	IC 7474	2
2.	IC TRAINER KIT	-	1
3.	PATCH CORDS	-	15

BRIEF DESCRIPTION:
A register is capable of shifting its binary information in one or both directions is known as shift register. The logical configuration of shift register consist of a D-Flip flop cascaded with output of one flip flop connected to input of next flip flop. All flip flops receive common clock pulses which causes the shift in the output of the flip flop. The simplest possible shift register is one that uses only flip flop. The output of a given flip flop is connected to the input of next flip flop of the register. Each clock pulse shifts the content of register one bit position to right.

(a) Serial in serial out Shift Register
(b) Serial in parallel out Shift Register



PIN DIAGRAM

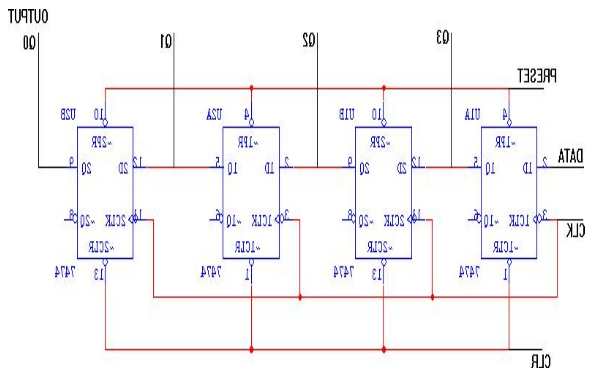


LOGIC DIAGRAM:SERIAL IN SERIAL OUT

CLK	Serial in	Serial out
1	1	0
2	0	0
3	0	0
4	1	1
5	X	0
6	X	0
7	X	1

TRUTH TABLE

SERIAL IN PARALLEL OUT



CLK	DATA	OUTPUT			
		Q_A	Q_B	Q_C	Q_D
1	1	1	0	0	0
2	0	0	1	0	0
3	0	0	0	1	1
4	1	1	0	0	1

LOGIC DIAGRAM **TRUTH TABLE**

PRE-EXPERIMENT QUESTIONS:-

1. State the features of IC 7495.
2. State the features of IC 74195.

PROCEDURE:-

- Connections are given as per circuit diagram.
- Logical inputs are given as per circuit diagram.
- Observe the output and verify the truth table.

POST-EXPERIMENT QUESTIONS:-

1. How can we use shift registers in serial communications? Explain.
2. List the ICs which are used as 8 bit SISO, SIPO, PISO, PIPO modes and as a bidirectional shift register.

LAB EXPERIMENT: 7

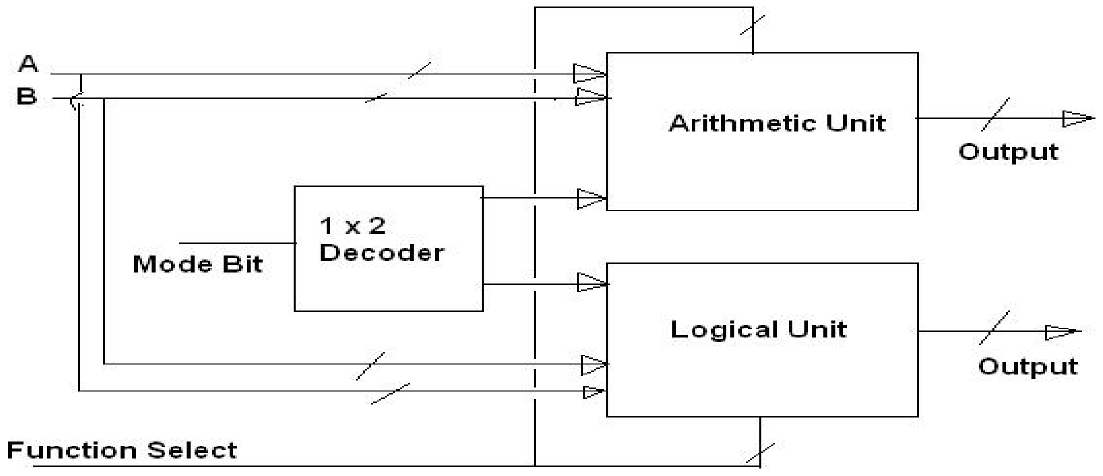
OBJECTIVE: Design and implementation of an 8 bit arithmetic logic unit.

EQUIPMENTS & COMPONENTS REQUIRED:

SL.No.	Equipments	Specification	Quantity
1	Digital IC Trainer kit	-	1
2	Digital Multimeter		1

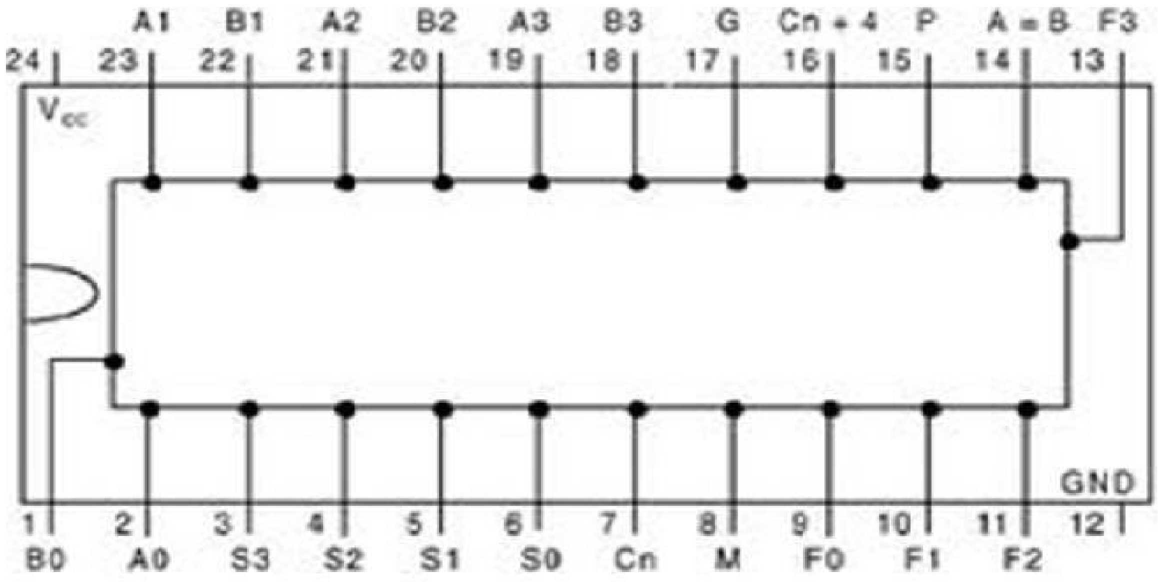
SL.No.	COMPONENT	SPECIFICATION	QTY.
1.	ALU IC	IC 7474	2
2.	PATCH CORDS	-	15

BRIEF DESCRIPTION:
ALU stands for the arithmetic and logical unit and is one of the important unit in almost all the calculating machine these days be it with the hand-held mobile, or computers. All the computational work in the system are carried out by this unit. The typical ALU sizes are :
4-bit ALU : ALU that processes two 4-bit numbers.
8-bit ALU: ALU that processes two 8-bit numbers.
Still in the latest systems ALU sizes are 16, 32, 64-bit etc.Figure-1 shows the block diagram of a typical ALU.



Block Diagram of ALU:

In figure-1, the 1x2 selector on the left is as a mode selector to select one of the two units i.e. either the arithmetic unit or the logical unit. The function select lines are then used to select one of the many functions of arithmetic or the logical type.
MSI package for ALU:- IC 74181 a 4-bit Arithmetic and logical unit:



PIN DIAGRAM OF IC 74181 ALU

SN54/74LS181

FUNCTION TABLE

MODE SELECT INPUTS				ACTIVE LOW INPUTS & OUTPUTS		ACTIVE HIGH INPUTS & OUTPUTS	
S_3	S_2	S_1	S_0	LOGIC (M = H)	ARITHMETIC** (M = L) (C_n = L)	LOGIC (M = H)	ARITHMETIC** (M = L) (C_n = H)
L	L	L	L	A	A minus 1	A	A
L	L	L	H	AB	AB minus 1	A + B	A + B
L	L	H	L	A + B	AB minus 1	AB	A + B
L	L	H	H	Logical 1 minus 1		Logical 0 minus 1	
L	H	L	L	A + B	A plus (A + B)	AB	A plus AB
L	H	L	H	B	AB plus (A + B)	B	(A + B) plus AB
L	H	H	L	A ⊕ B	A minus B minus 1	A ⊕ B	A minus B minus 1
L	H	H	H	A + B	A + B	AB	AB minus 1
H	L	L	L	AB	A plus (A + B)	A + B	A plus AB
H	L	L	H	A ⊕ B	A plus B	A ⊕ B	A plus B
H	L	H	L	B	AB plus (A + B)	B	(A + B) plus AB
H	L	H	H	A + B	A + B	AB	AB minus 1
H	H	L	L	Logical 0	A plus A*	Logical 1	A plus A*
H	H	L	H	AB	AB plus A	A + B	(A + B) plus A
H	H	H	L	AB	AB plus A	A + B	(A + B) Plus A
H	H	H	H	A	A	A	A minus 1

L = LOW Voltage Level
H = HIGH Voltage Level
*Each bit is shifted to the next more significant position
**Arithmetic operations expressed in 2s complement notation

PRE-EXPEIMENT QUESTIONS:-

1. What is Digitizing?

PROCEDURE:-

- Keep the datasheet of IC 74181 ready.
- Insert the IC on the Breadboard.
- Make connections as shown in figure.
- Verify the connections

POST-EXPEIMENT QUESTIONS:-

1. Which Boolean operator combines search terms so that each search result contains all term ?

LAB EXPERIMENT 8

OBJECTIVE: Design a data path of a computer from its register transfer language description.

EQUIPMENTS & COMPONENTS REQUIRED:

SL.No.	Equipment's	Specification	Quantity
1	Digital IC Trainer kit	-	1
2	Digital Multimeter		1

SL.No.	Components	Specification	Quantity
1	Digital ICs	7400, 7402, 7404, 7408, 7432, 7486.	1 each
2	Patch cords	-	6

BRIEF DESCRIPTION:

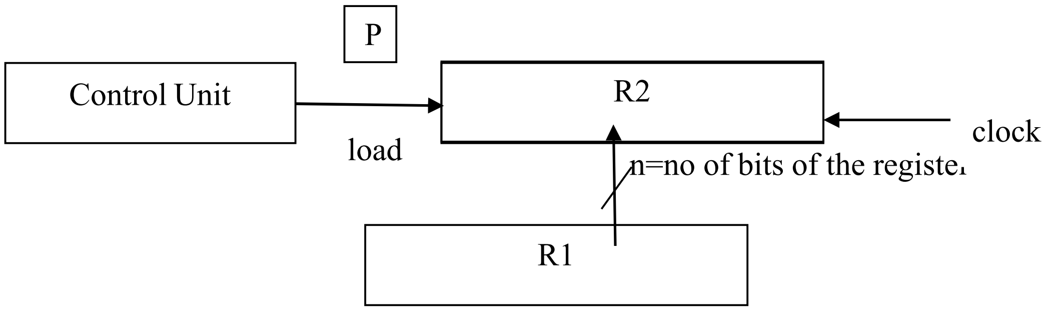
The symbolic notation used to describe the micro-operation transfers among registers is called Register Transfer Language. The term "register transfer" implies the availability of hardware logic circuits that can perform a stated micro-operation and transfer the result of the operation to the same or another register.

A statement that specifies a register transfer implies that circuits are available from the outputs of the source register to the inputs of the destination register and that the destination register has a parallel load capacity. If the transfer is to occur under a predetermined condition i.e.

$$\text{If}(P=1) \text{ then } R2 \leftarrow R1$$

Where P is the control signal generated in the control section. A Control Function is a Boolean variable that is equal to 0 or 1

$$P: R2 \leftarrow R1$$



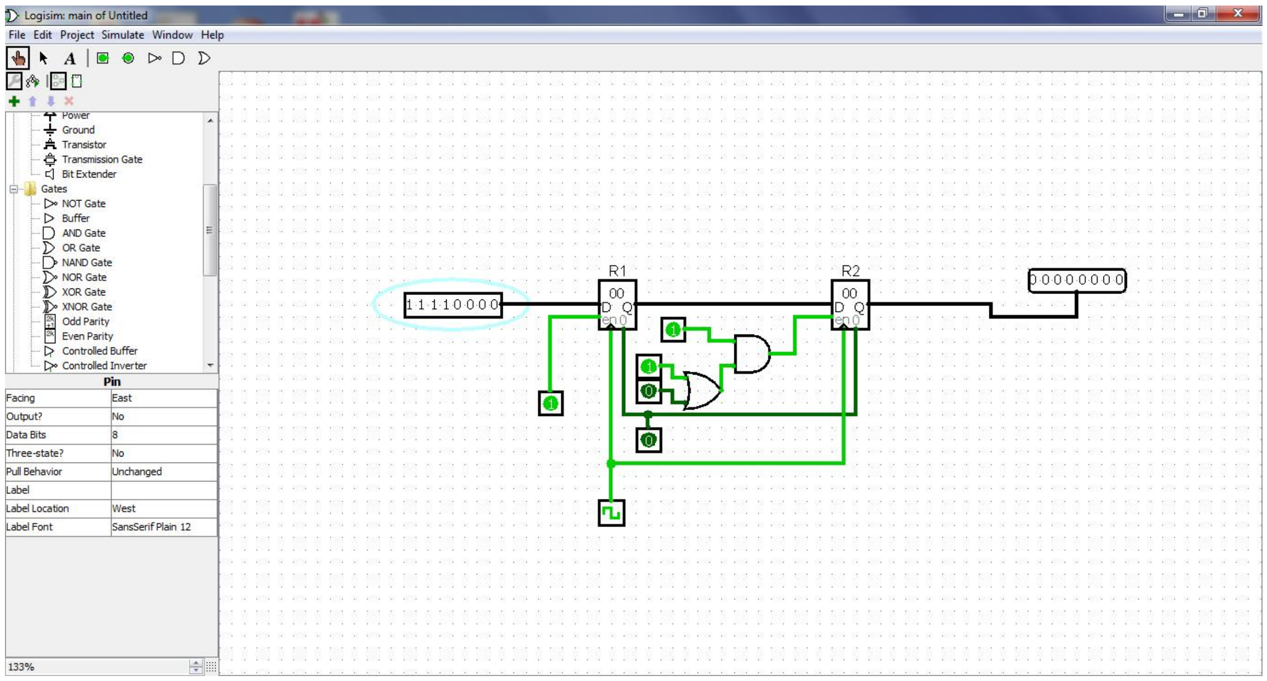
BLOCK DIAGRAM

PRE-EXPERIMENT QUESTIONS:-

1. What is the significance of Data Path?
2. What is Register Transfer Logic?

PROCEDURE:-

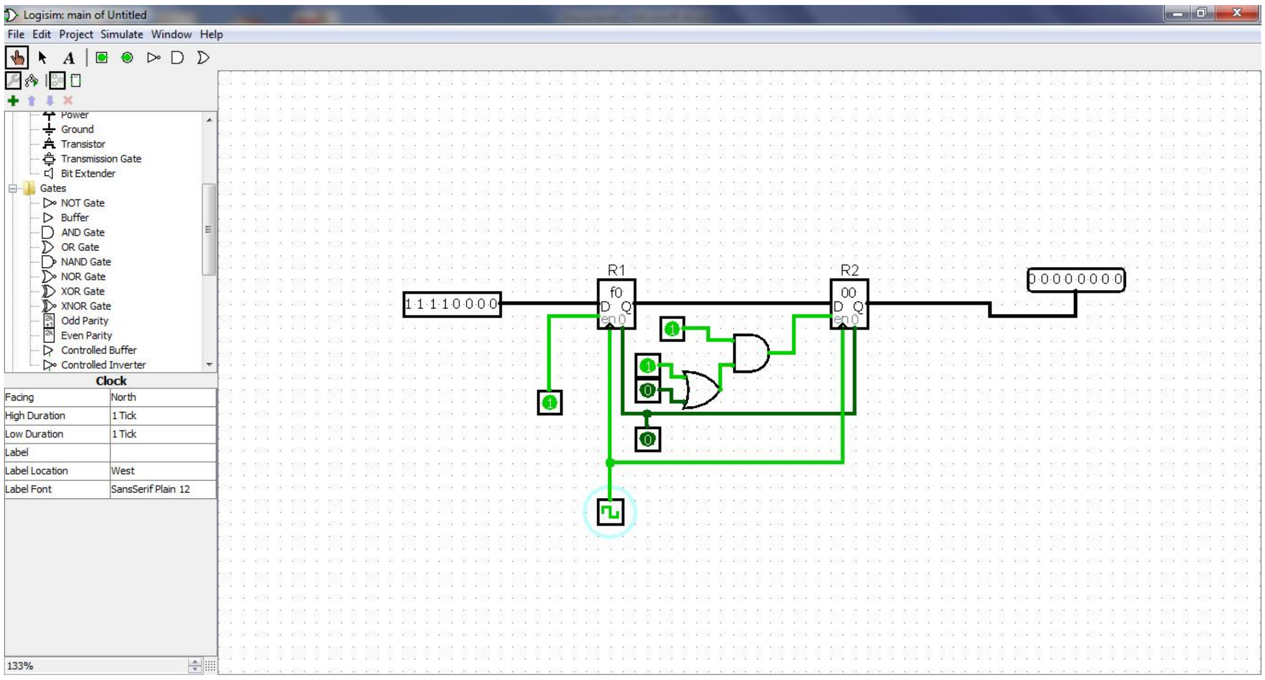
Data bits Ready to be transferred



Pin

Facing	East
Output?	No
Data Bits	8
Three-state?	No
Pull Behavior	Unchanged
Label	
Label Location	West
Label Font	SansSerif Plain 12

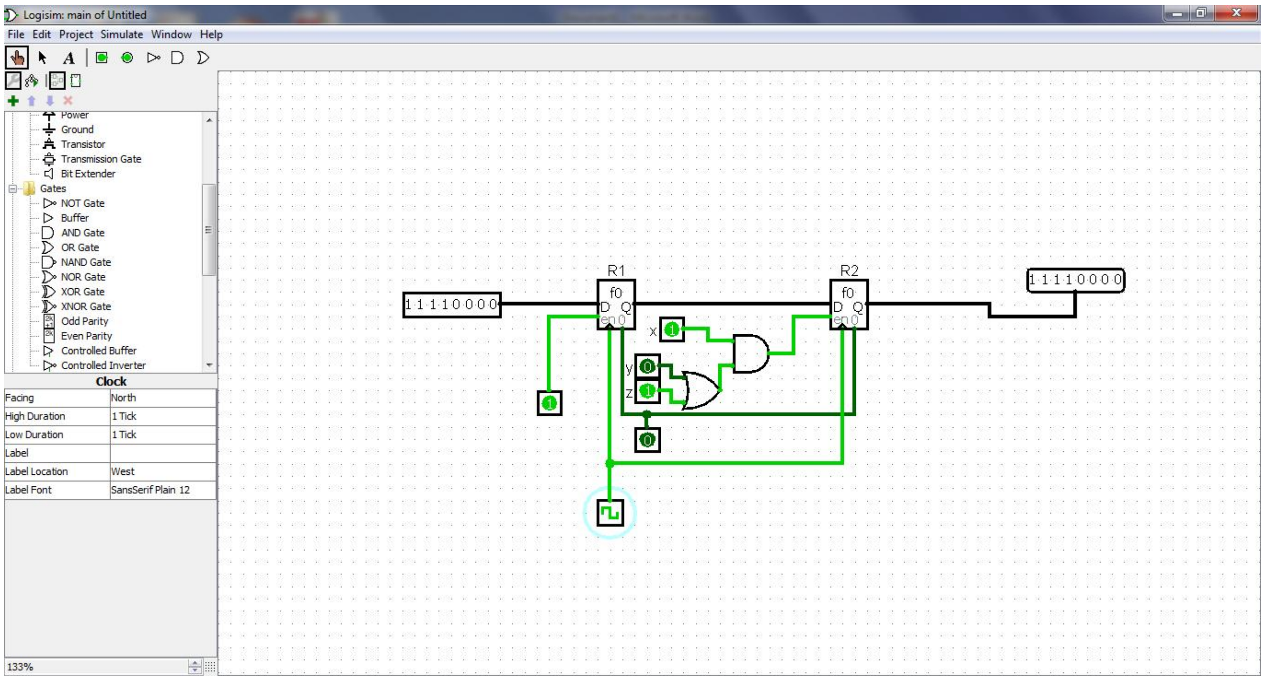
In the first clock tick Register R1 stores the value



Clock

Facing	North
High Duration	1 Tick
Low Duration	1 Tick
Label	
Label Location	West
Label Font	SansSerif Plain 12

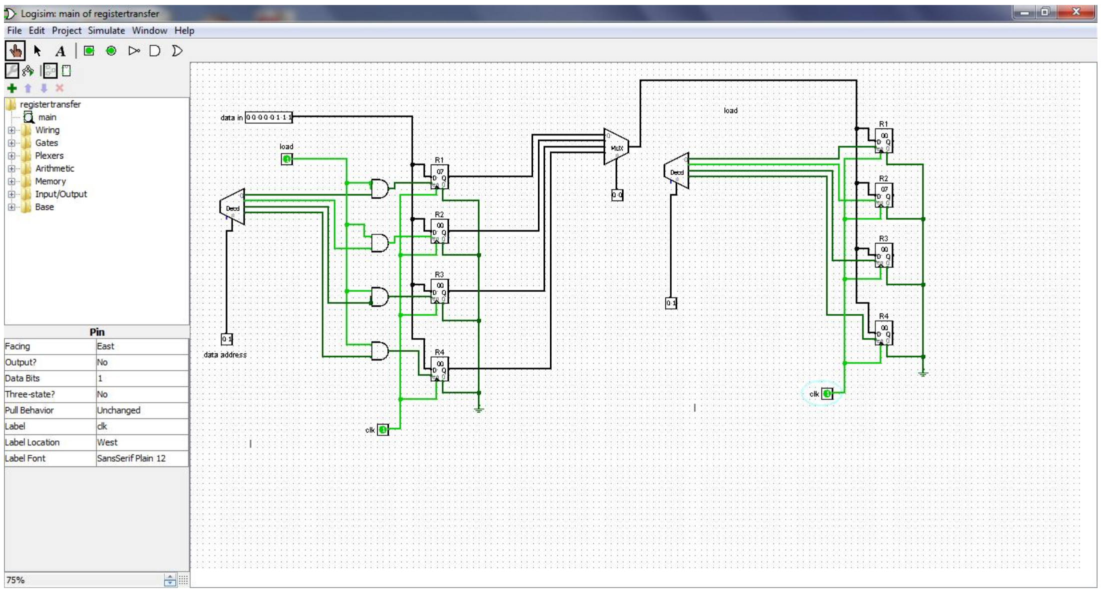
In the next clock pulse the value gets transferred to register R2 if its Enable input is high, i.e. its control function x.(y+z) =1



Clock

Facing	North
High Duration	1 Tick
Low Duration	1 Tick
Label	
Label Location	West
Label Font	SansSerif Plain 12

Register Transfer using BUS



Pin	
Facing	East
Output?	No
Data Bits	1
Three-state?	No
Pull Behavior	Unchanged
Label	clk
Label Location	West
Label Font	SansSerif Plain 12

POST-EXPERIMENT QUESTIONS: -

1. How a data path is designed?
2. Design a data path for $R_3 \leftarrow R_2 + R_1$

LAB EXPERIMENT 9

OBJECTIVE: Design a data path of a computer from its register transfer language description.

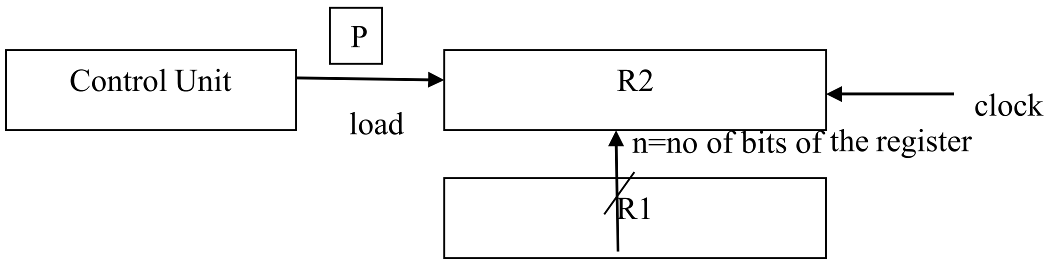
EQUIPMENTS&COMPONENTS REQUIRED:
Registers
AND Gates
OR Gate
Connecting Wires

BREIF DESCRIPTION:
The symbolic notation used to describe the micro-operation transfers among registers is called Register Transfer Language.
The term "register transfer" implies the availability of hardware logic circuits that can perform a stated micro-operation and transfer the result of the operation to the same or another register. A statement that specifies a register transfer implies that circuits are available from the outputs of the source register to the inputs of the destination register and that the destination register has a parallel load capacity.
If the transfer is to occur under a predetermined condition i.e

If(P=1) then R2←R1

Where P is the control signal generated in the control section. A Control Function is a Boolean variable that is equal to 0 or 1

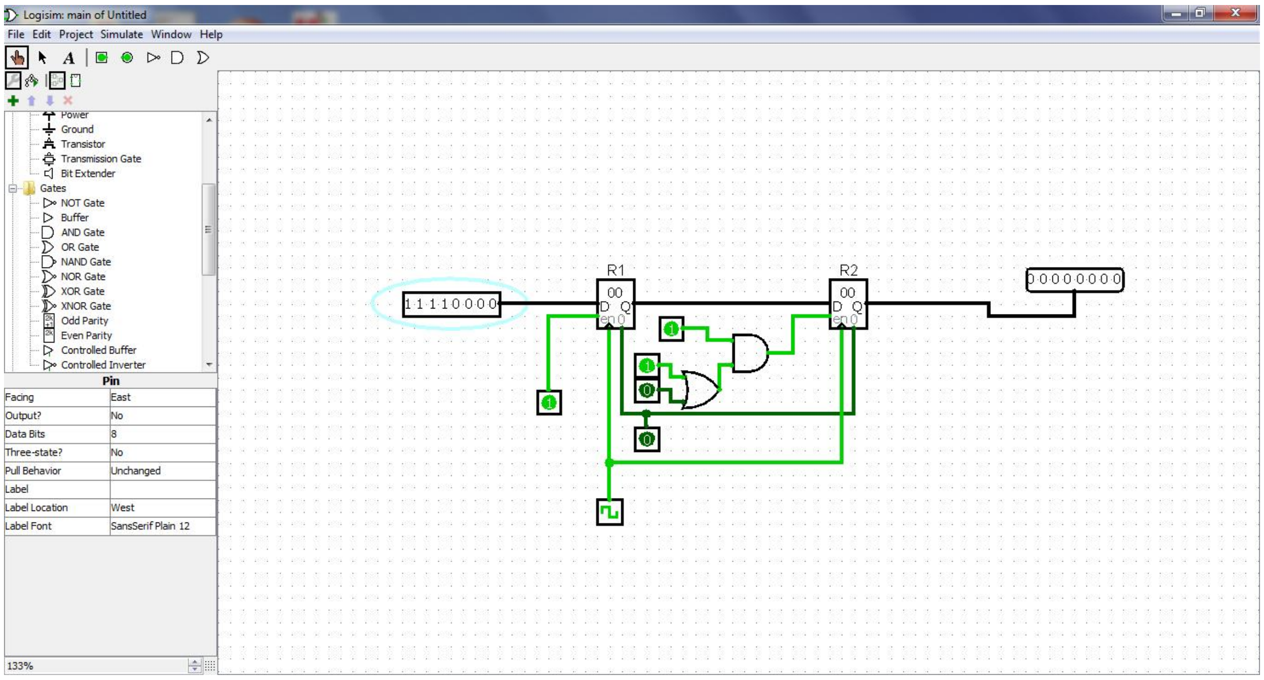
P: R2←R1



BLOCKDIAGRAM

PRE-EXPERIMENT QUESTIONS:-
1. What is RTL?
2. What is the minimum no. of registers needed in the instruction set architecture of the processor to compile a code with 3 operands?

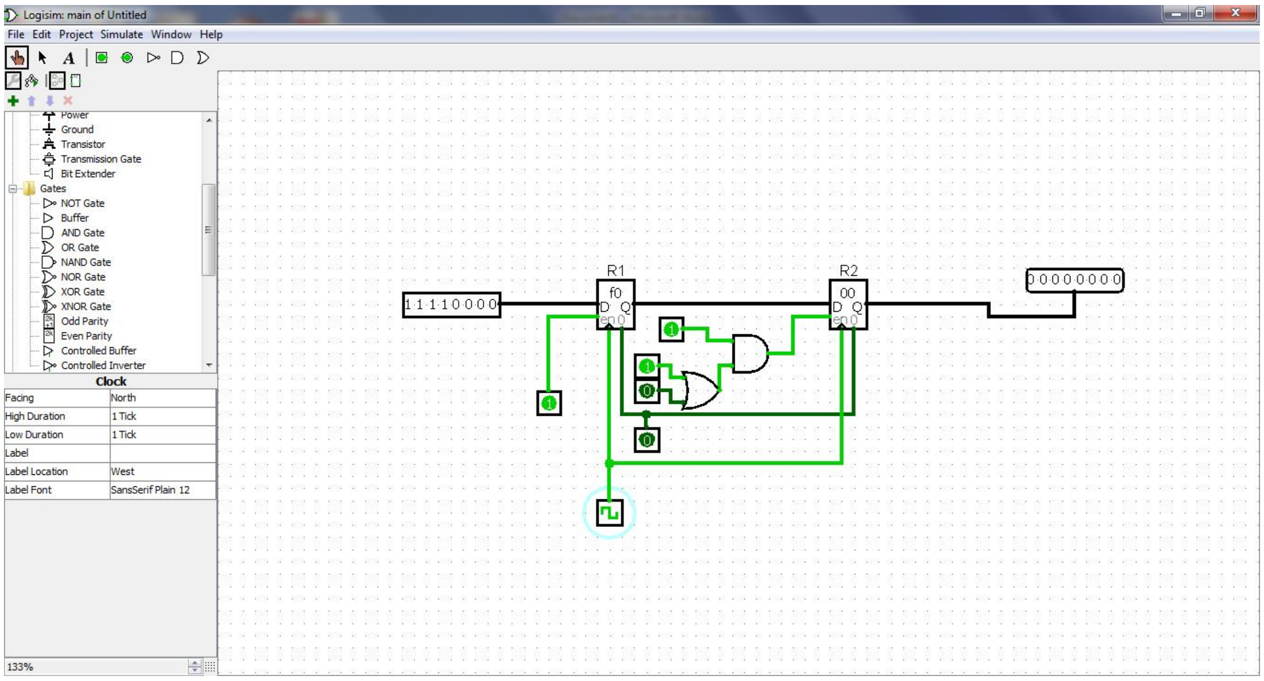
PROCEDURE:-
Data bits Ready to be transfered



Pin

Facing	East
Output?	No
Data Bits	8
Three-state?	No
Pull Behavior	Unchanged
Label	
Label Location	West
Label Font	SansSerif Plain 12

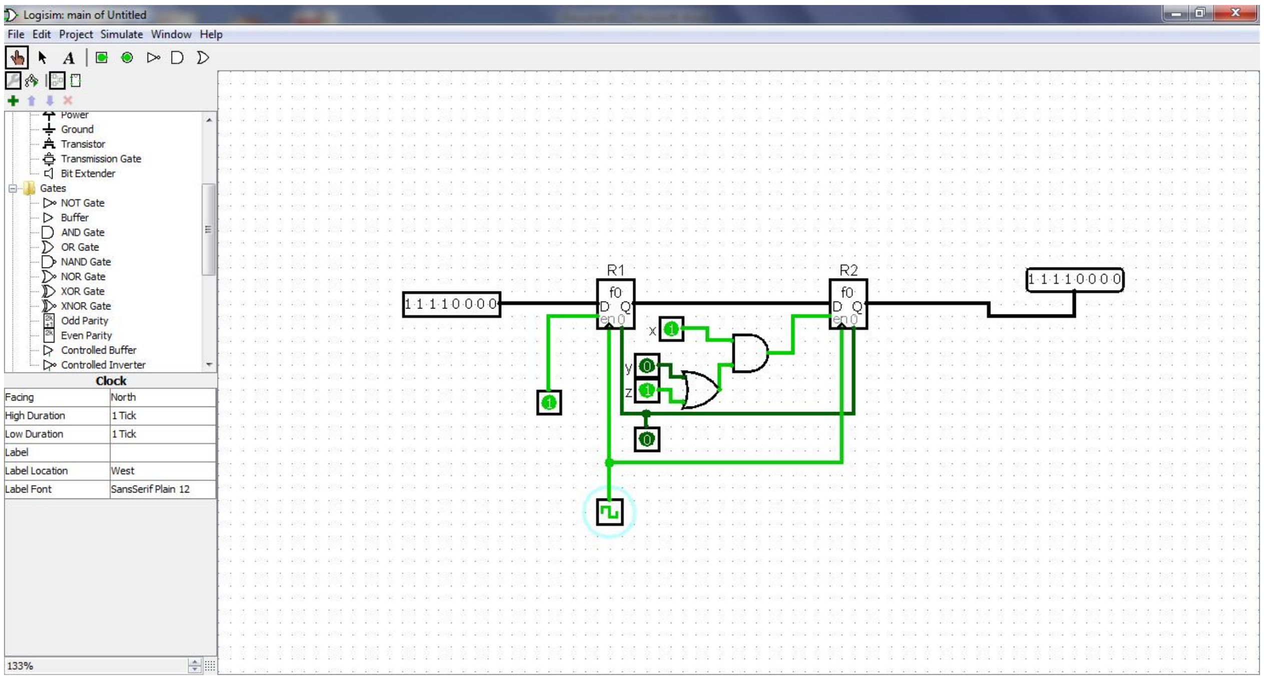
In the first clock tick Register R1 stores the value



Clock

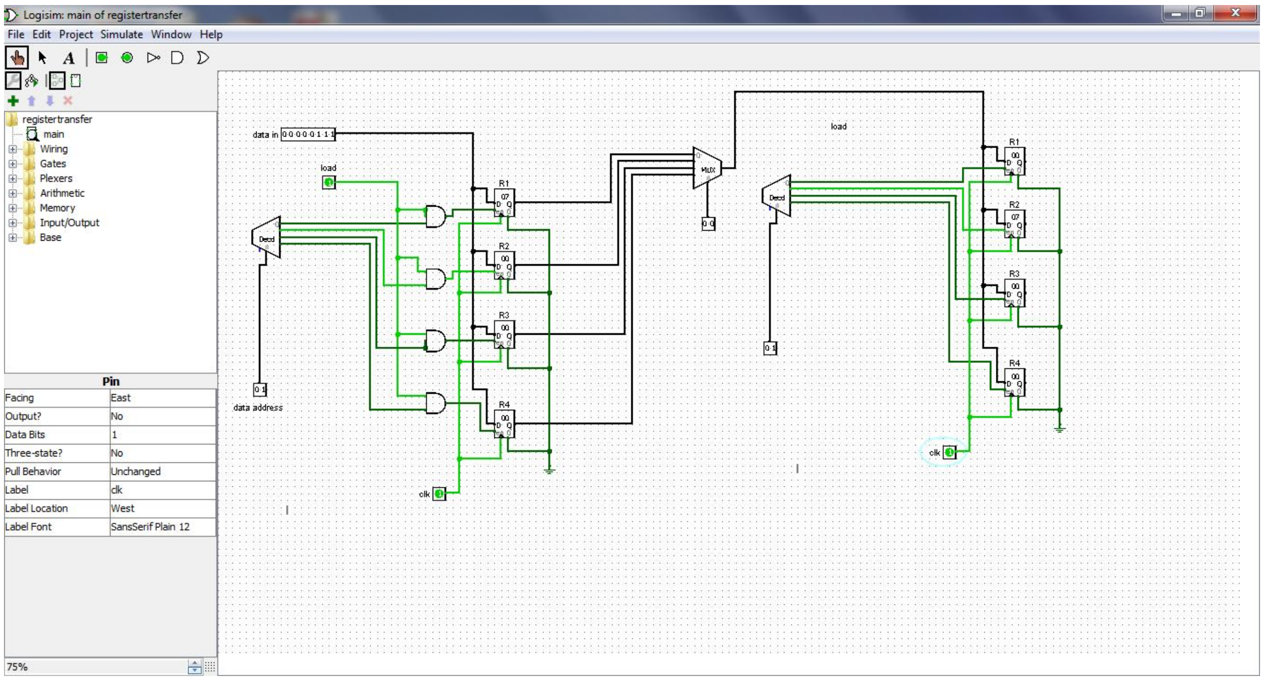
Facing	North
High Duration	1 Tick
Low Duration	1 Tick
Label	
Label Location	West
Label Font	SansSerif Plain 12

In the next clock pulse the value gets transferred to register R2 if its Enable input is high, i.e its control function x.(y+z) =1



Clock

Facing	North
High Duration	1 Tick
Low Duration	1 Tick
Label	
Label Location	West
Label Font	SansSerif Plain 12



Pin

Facing	East
Output?	No
Data Bits	1
Three-state?	No
Pull Behavior	Unchanged
Label	clk
Label Location	West
Label Font	SansSerif Plain 12

REGISTER TRANSFER USING BUS

POST-EXPERIMENT QUESTION:-

1. What is the amount of ROM needed to implement a 4 bit multiplier.

LAB EXPERIMENT 10

OBJECTIVE: Design the control unit of a computer using hardwired based on its register transfer language description.

COMPONENTS REQUIRED:

1. Registers
2. AND Gates
3. OR Gate
4. Memory
5. Decoders and Multiplexers
6. Connecting Wires, etc.

BREIF DESCRIPTION:

The control unit (CU) is a component of a computer's central processing unit (CPU) that directs the operation of the processor. It tells the computer's memory, arithmetic/logic unit and input and output devices on how to respond to a program's instructions.

It directs the operation of the other units by providing timing and control signals. Most computer resources are managed by the CU. It directs the flow of data between the CPU and the other devices. The Control Unit (CU) is digital circuitry contained within the processor that coordinates the sequence of data movements into, out of, and between a processor's many sub-units. The result of these routed data movements through various digital circuits (sub-units) within the processor produces the manipulated data expected by a software instruction (loaded earlier, likely from memory). It controls (conducts) data flow inside the processor and additionally provides several external control signals to the rest of the computer to further direct data and instructions to/from processor external destination's (i.e. memory).

PRE-EXPERIMENT QUESTION:-

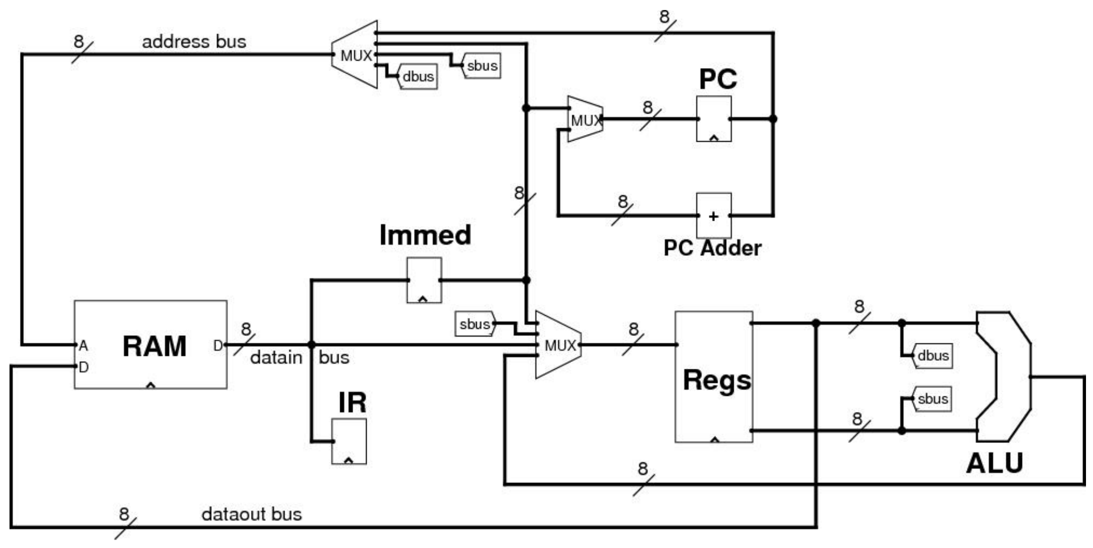
1. Why is register renaming done in pipelined processor.

PROCEDURE:-

The CPU has an 8-bit data bus and an 8-bit address bus, so it can only support 256 bytes of memory to hold both instructions and data.

- Internally, there are four 8-bit registers, R0 to R3, plus an Instruction Register, the Program Counter, and an 8-bit register which holds immediate values.
- The ALU is the same one that we designed last week. It performs the four operations AND, OR, ADD and SUB on two 8-bit values, and supports signed ADDs and SUBs.
- The CPU is a load/store architecture: data has to be brought into registers for manipulation, as the ALU only reads from and writes back to the registers.
- The ALU operations have two operands: one register is a source register, and the second register is both source and destination register, i.e. destination register = destination register OP source register.

- All the jump operations perform absolute jumps; there are no PC-relative branches. There are conditional jumps based on the zeroness or negativity of the destination register, as well as a "jump always" instruction.
- The following diagram shows the datapaths in the CPU:



- The *dbus* and *sbus* labels indicate the lines coming out from the register file which hold the value of the destination and source registers.
- Note the data loop involving the registers and the ALU, whose output can only go back into a register.
- The dataout bus is only connected to the *dbus* line, so the only value which can be written to memory is the destination register.
- Also note that there are only 3 multiplexors:
  - the address bus multiplexor can get a memory address from the PC, the immediate register (for direct addressing), or from the source or destination registers (for register indirect addressing).
  - the PC multiplexor either lets the PC increment, or jump to the value in the immediate register.
  - the multiplexor in front of the registers determines where a register write comes from: the ALU, the immediate register, another register or the data bus.

As we have decided the hardware to be used in our design now we have to decide the instruction set for our computer.

Instruction Set

- Half of the instructions in the instruction set fit into one byte:

op1	op2	Rd	Rs
2	2	2	2

- These instructions are identified by a 0 in the most-significant bit in the instruction, i.e. *op1* = 0X.
- The 4 bits of opcode are split into *op1* and *op2*
- *Rd* is the destination register, and *Rs* is the source register.
- The other half of the instruction set are two-byte instructions. The first byte has the same format as above, and it is followed by an 8-bit constant or immediate value:

op1	op2	Rd	Rs	immediate
2	2	2	2	8

- These two-byte instructions are identified by a 1 in the most-significant bit in the instruction, i.e. *op1* = 1X.
- With 4 operation bits, there are 16 instructions:

op1	**op2**	**Mnemonic**	**Purpose**
00	00	AND Rd, Rs	Rd = Rd AND Rs
00	01	OR Rd, Rs	Rd = Rd OR Rs
00	10	ADD Rd, Rs	Rd = Rd + Rs
00	11	SUB Rd, Rs	Rd = Rd –Rs
01	00	LW Rd, (Rs)	Rd = Mem[Rs]
01	01	SW Rd, (Rs)	Mem[Rs] = Rd
01	10	MOV Rd, Rs	Rd = Rs
01	11	NOP	Do nothing
10	00	JEQ Rd, immed	PC = immed if Rd == 0
10	01	JNE Rd, immed	PC = immed if Rd != 0
10	10	JGT Rd, immed	PC = immed if Rd > 0
10	11	JLT Rd, immed	PC = immed if Rd < 0
11	00	LW Rd, immed	Rd = Mem[immed]
11	01	SW Rd, immed	Mem[immed] = Rd
11	10	LI Rd, immed	Rd = immed
11	11	JMP immed	PC = immed

- Note the regularity of the ALU operations and the jump operations: we can feed the *op2* bits directly into the ALU, and use *op2* to control the branch decision.
- The rest of the instruction set is less regular, which will require special decoding for certain of the 16 instructions.

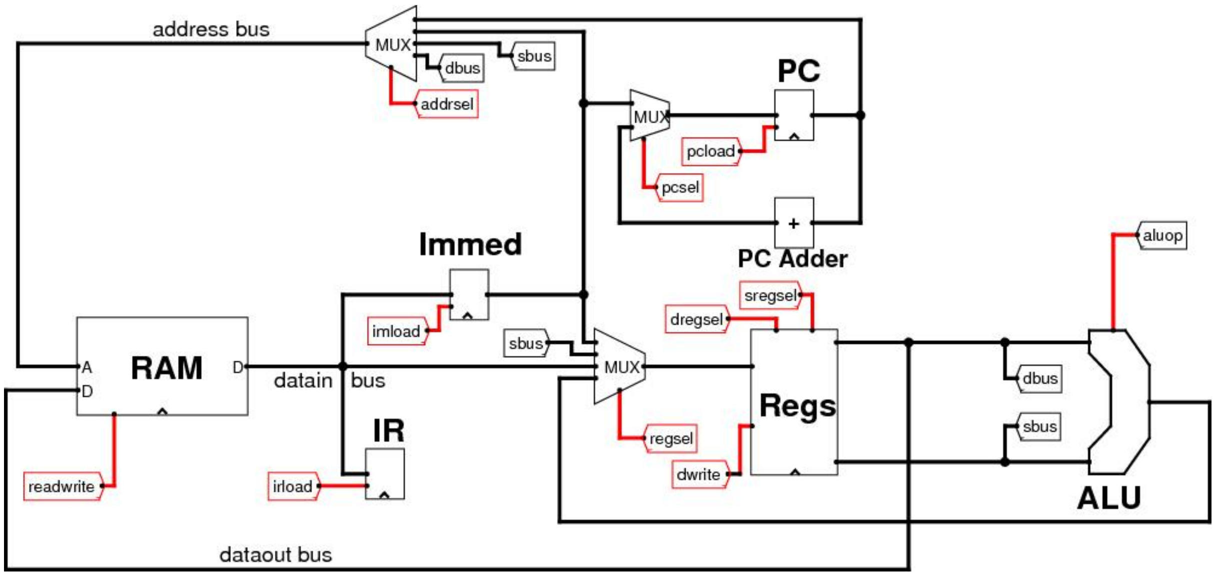
Instruction Phases

- The CPU internally has three phases for the execution of each instruction.

- On phase 0, the instruction is fetched from memory and stored in the Instruction Register.
- On phase 1, if the fetched instruction is a two-byte instruction, the second byte is fetched from memory and stored in the Immediate Register. For one-byte instructions, nothing occurs in phase 1.
- On phase 2, everything else is done as required, which can include:
    - an ALU operation, reading from two registers.
    - a jump decision which updates the PC.
    - a register write.
    - a read from a memory location.
    - a write to a memory location.
- After phase 2, the CPU starts the next instruction in phase 0.

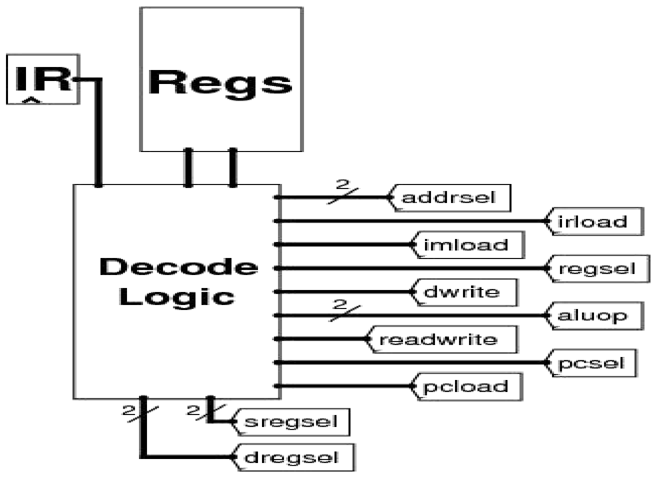
CPU Control Lines

- Below is the main CPU diagram again, this time with the control lines shown.



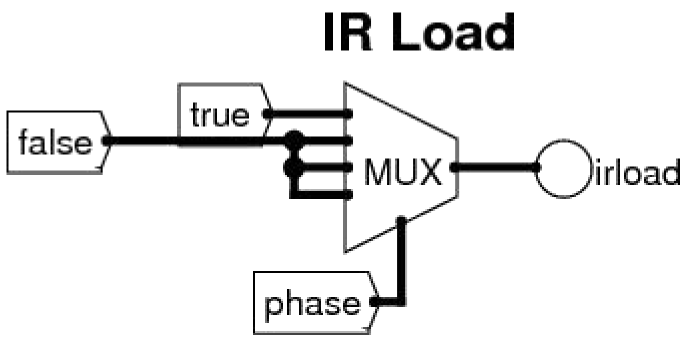
- There are several 1-bit control lines:
    - *pcsel*, increment PC or load the jump value from the Immediate Register.
    - *pcload*, load the PC with a new value, or don't load a new value.
    - *irload*, load the Instruction Register with a new instruction.
    - *imload*, load the Immediate Register with a new value.
    - *readwrite*, read from memory, or write to memory.
    - *dwrite*, write a value back to a register, or don't write a value.
- There are also several 2-bit control lines:
    - *addrsel*, select an address from the PC, the Immediate Register, the source register or the destination register.
    - *regsel*, select a value to write to a register from the Immediate Register, another register, the data bus or from the ALU.
    - *dregsel* and *sregsel*, select two registers whose values are sent to the ALU.
    - *aluop*, which are the *op2* bits that control the operation of the ALU.

- The values for all of these control lines are generated by the Decode Logic, which gets as input the value from the Instruction Register, and the zero & negative lines of the destination register.



Phase Zero

- On phase zero, the PC's value has to be placed on the address bus, so the *addrsel* line must be 0. The *irload* line needs to be 1 so that the IR is loaded from the *datain* bus. Finally, the PC must be incremented in case we need to fetch an immediate value in phase 1.
- All of this can be done using multiplexors which output different values depending on the current phase. Here is the control logic for the *irload* line.



- We only need to load the IR on phase 0, so we can wire true to the 0 input of the *irload* multiplexor, and false to the other inputs. **Note:** input 11 (i.e. decimal 3) to the multiplexor is never used, as we never get to phase 3, but Logisim wants all multiplexor inputs to be valid.
- Another way to look at each phase is the value which needs to be set for each control line, for each instruction.
- For phase zero, these control line values can be set for all instructions:

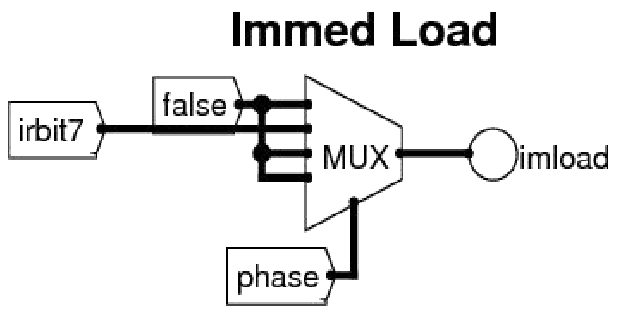
op 1	op 2	instruct	pcsel	pcload	Irload	imload	Rw	dwrite	jumpsel	addrsel	regsel	dreg	sreg	aluop

xx	xx	all	1	1	1	0	0	0	0	0	x	x	x	x

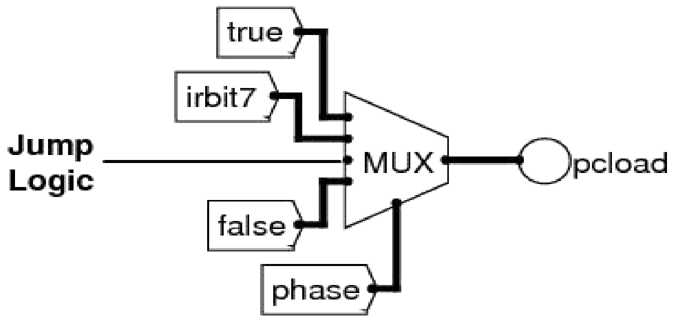
- 'x' stands for any value, i.e. accept any opcode value, output any control line value.

Phase One

- On phase 1, we need to load the Immediate Register with a value from memory if the *irbit7* from the IR is true. The PC's value has to be placed on the address bus, so the *addrsel* line must be 0. The *imload* line needs to be 1 so that the Immediate Register is loaded from the *datain* bus. Finally, the PC must be incremented so that we are ready to fetch the next instruction on the next phase 0.



- The *imload* logic is shown above. It is very similar to the *irload* logic, but this time an enable value is output only on phase 1, and only if the *irbit7* is set.



- Some of the *pcload* logic is shown above. The PC is always incremented at phase 0. It is incremented at phase 1 if *irbit7* is set, i.e. a two-byte instruction. Finally, the PC can be loaded with an immediate value in phase 2 if we are performing a jump instruction and the jump test is true. We will come back to the jump logic later.
- We can tabulate the values of the control lines for phase 1. This time, what is output depends on the top bit of the *op1* value:
-

op1	**op2**	**instruct**	**pcsel**	**pcload**	**Irload**	**imload**	**Rw**	**dwrite**	**jumpsel**	**addrsel**	**regsel**	**dreg**	**sreg**	**aluop**

0x	xx	all	x	0	0	0	0	0	0	0	x	x	x	x
1x	xx	all	1	1	0	1	0	0	0	0	x	x	x	x

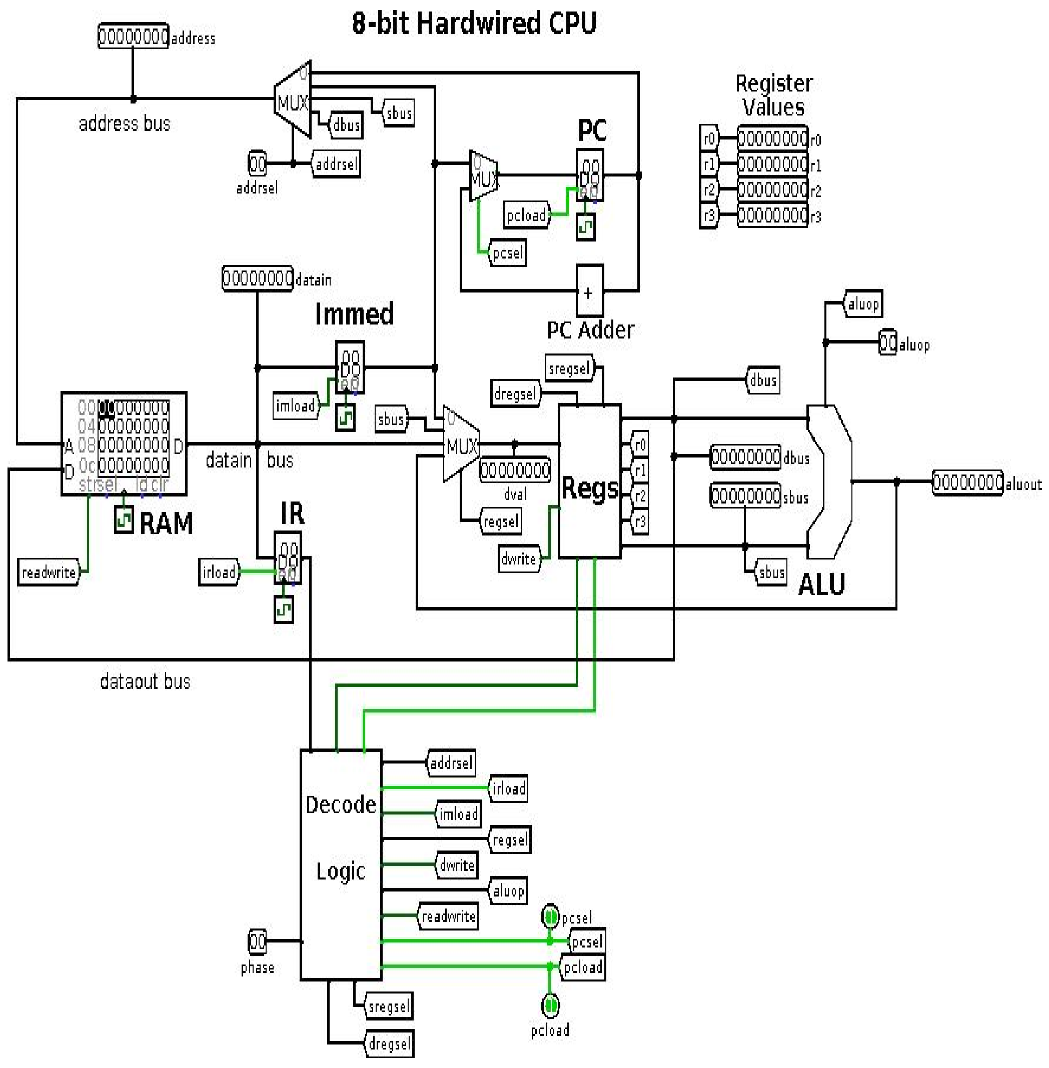
Phase Two

- Here, the values of the control lines depend heavily on what specific instruction we are performing. Here's the table of control line outputs depending on the instruction:

op1	**op2**	**instruct**	**pcsel**	**pcload**	**irload**	**imload**	**rw**	**dwrite**	**addrsel**	**regsel**	**dreg**	**sreg**	**aluop**
00	00	AND Rd, Rs	x	0	0	0	0	1	x	3	Rd	Rs	op2
00	01	OR Rd, Rs	x	0	0	0	0	1	x	3	Rd	Rs	op2
00	10	ADD Rd, Rs	x	0	0	0	0	1	x	3	Rd	Rs	op2
00	11	SUB Rd, Rs	x	0	0	0	0	1	x	3	Rd	Rs	op2
01	00	LW Rd, (Rs)	x	0	0	0	0	1	2	2	Rd	Rs	x
01	01	SW Rd, (Rs)	x	0	0	0	1	0	3	x	Rd	Rs	x
01	10	MOV Rd, Rs	x	0	0	0	0	1	x	1	Rd	Rs	x
01	11	NOP	x	0	0	0	0	0	x	x	x	x	x
10	00	JEQ Rd, immed	0	J	0	0	0	0	x	x	Rd	x	op2
10	01	JNE Rd, immed	0	J	0	0	0	0	x	x	Rd	x	op2
10	10	JGT Rd, immed	0	J	0	0	0	0	x	x	Rd	x	op2
10	11	JLT Rd, immed	0	J	0	0	0	0	x	x	Rd	x	op2
11	00	LW Rd, immed	x	0	0	0	0	1	1	2	Rd	x	x
11	01	SW Rd, immed	x	0	0	0	1	0	1	x	Rd	x	x
11	10	LI Rd, immed	x	0	0	0	0	1	x	0	Rd	x	x
11	11	JMP immed	0	1	0	0	0	0	x	x	x	x	x

- To make the control line logic as simple as possible, a CPU designer is always striving for regularity. However, this is often in conflict with the desired CPU functionality.
- From the table above, the ALU instructions (*op1*=00) and the jump instructions (*op1*=10) are nice and regular. All the *op1*=1x instructions use the Immediate Register, while the *op1*=0x instructions don't.
- We can always tie *dregsel* to *Rd* from the instruction, and the same goes for *sregsel* = *Rs* and *aluop* = *op2*. And *irload* and *imload* are always 0 for phase 2.
- With the remaining control lines, the regularities cease.

Putting this all back together, we now have this device:



POST-EXPERIMENT QUESTION:-

1. Which DMA transfer mode and interrupt handling mechanism will enable the highest I/O bandwidth.