A
Laboratory Manual for

Computer Organization & Architecture

(3140707)

B.E. (Semester-4)
Computer Engineering

Vishwakarma Government Engineering College

Directorate of Technical Education, Gandhinagar, Gujarat

Vishwakarma Government Engineering College

Department of Computer Engineering

CERTIFICATE

This is to certify that Mr./Ms. ____________Thakar Atri Kamleshkumar_______________ Enrollment No. _220170107141__ of ***B.E. Semester - 4*** *from* ***Computer Engineering Department*** *of this Institute (GTU Code:* ***017****) has satisfactorily completed the Practical / Tutorial work for the subject* ***Computer Organization & Architecture (3140707)*** *for the academic year 2023-24.*

Place: ____________

Date: ____________

Signature of Course Faculty **Head of the Department**

Preface

Main motto of any laboratory/practical/field work is for enhancing required skills as well as creating ability amongst students to solve real time problem by developing relevant competencies in psychomotor domain. By keeping in view, GTU has designed competency focused outcome-based curriculum for engineering degree programs where sufficient weightage is given to practical work. It shows importance of enhancement of skills amongst the students and it pays attention to utilize every second of time allotted for practical amongst students, instructors and faculty members to achieve relevant outcomes by performing the experiments rather than having merely study type experiments. It is must for effective implementation of competency focused outcome-based curriculum that every practical is keenly designed to serve as a tool to develop and enhance relevant competency required by the various industry among every student. These psychomotor skills are very difficult to develop through traditional chalk and board content delivery method in the classroom. Accordingly, this lab manual is designed to focus on the industry defined relevant outcomes, rather than old practice of conducting practical to prove concept and theory.

By using this lab manual students can go through the relevant theory and procedure in advance before the actual performance which creates an interest and students can have basic idea prior to performance. This in turn enhances pre-determined outcomes amongst students. Each experiment in this manual begins with competency, industry relevant skills, course outcomes as well as practical outcomes (objectives). The students will also achieve safety and necessary precautions to be taken while performing practical.

This manual also provides guidelines to faculty members to facilitate student centric lab activities through each experiment by arranging and managing necessary resources in order that the students follow the procedures with required safety and necessary precautions to achieve the outcomes. It also gives an idea that how students will be assessed by providing rubrics.

Computer Architecture is concerned with the way hardware components are connected together to form a computer system. It acts as the interface between hardware and software. Computer Architecture helps us to understand the functionalities of a system. A programmer can view architecture in terms of instructions, addressing modes and registers. While designing a computer system architecture is considered first. Computer Architecture deals with high-level design issues. Architecture involves Logic (Instruction sets, Addressing modes, Data types, Cache optimization).

Utmost care has been taken while preparing this lab manual however always there is chances of improvement. Therefore, we welcome constructive suggestions for improvement and removal of errors if any.

DTE's Vision

- To provide globally competitive technical education.
- Remove geographical imbalances and inconsistencies.
- Develop student friendly resources with a special focus on girls' education and support to weaker sections.
- Develop programs relevant to industry and create a vibrant pool of technical professionals.

Institute's Vision

- To create an ecosystem for proliferation of socially responsible and technically sound engineers, innovators and entrepreneurs.

Institute's Mission

- To develop state-of-the-art laboratories and well-equipped academic infrastructure.
- To motivate faculty and staff for qualification up-gradation, and enhancement of subject knowledge.
- To promote research, innovation and real life problem solving skills.
- To strengthen linkages with industries, academic and research organizations.
- To reinforce concern for sustainability, natural resource conservation and social responsibility.

Department's Vision

- To create an environment for providing value based education in Computer Engineering through innovation, team work and ethical practices.

Department's Mission

- To produce computer engineering graduates according to the needs of industry, government, society and scientific community.
- To develop state of the art computing facilities and academic infrastructure.

- To develop partnership with industries, government agencies and R & D organizations for knowledge sharing and overall development of faculties and students.
- To solve industrial, governance and societal issues by applying computing techniques.
- To create environment for research and entrepreneurship.

Programme Outcomes (POs)

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **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.
3. **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.
4. **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.
5. **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.
6. **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.
7. **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.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **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.

11. **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.
12. **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 Specific Outcomes (PSOs)

- Sound knowledge of fundamentals of computer science and engineering including software and hardware.
- Develop the software using sound software engineering principles having web based/mobile based interface.
- Use various tools and technology supporting modern software frameworks for solving problems having large volume of data in the domain of data science and machine learning.

Program Educational Objectives (PEOs)

- Possess technical competence in solving real life problems related to Computing.
- Acquire good analysis, design, development, implementation and testing skills to formulate simple computing solutions to the business and societal needs.
- Provide requisite skills to pursue entrepreneurship, higher studies, research, and development and imbibe high degree of professionalism in the fields of computing.
- Embrace life-long learning and remain continuously employable.
- Work and excel in a highly competence supportive, multicultural and professional environment which abiding to the legal and ethical responsibilities.

Practical – Course Outcome matrix

Course Outcomes (COs)

CO_3140707.1	Identify and explain the basic structure and functional units of a digital computer.
CO_3140707.2	Write assembly language programs and identify the role and working of various functional units of a computer for executing instructions.
CO_3140707.3	Design processing unit using the concepts of ALU and control logic design.
CO_3140707.4	Design circuits for interfacing memory and I/O with processor.
CO_3140707.5	Comprehend the features and performance parameters of different types of computer architectures.

Sr. No.	**Objective(s) of Experiment**	**CO 1**	**CO 2**	**CO 3**	**CO 4**	**CO 5**
1.	Study the basic structure and functional units of digital computer.	√				
2.	Write the working of 8085 simulator GNUsim8085.	√	√			
3.	Study basic architecture of 8085 along with small introduction.	√		√		

4.	Write following 8085 assembly language programs. a. To add two 8-bit numbers and store result in memory location. b. To add five 8-bit numbers and store sum and carry in memory location. c. Perform addition of two 8-bit numbers and store result after using DAA instruction. d. Perform 16-bit subtraction. e. Perform multiplication of two 8-bit numbers using loop. f. Find minimum and maximum from five 8-bit numbers stored in memory location. g. To count the no. of 1's in given 8-bit value.		√			
5.	Write following 8085 assembly language programs. a. To add two arrays of five 8-bit values and store result in third array. b. To store sum of unpacked BCD numbers from packed BCD numbers. c. To store an ASCII value of each Hex digits. d. To set an MSB of an 8-bit values if it has even no. of 1's. e. To reverse an array of ten 8-bit values using subroutine. f. Find factorial of given 8-bit value. g. To perform division of 16-bit number by 8-bit number.		√			
6.	Design ALU using Logisim.			√		
7.	Implement Booth's multiplication algorithm.			√		
8.	Study interconnection structures of components of multiprocessor system.			√	√	√
9.	Design circuits for interfacing memory and I/O with processor.				√	
10.	Study about different types of computer architectures and comprehend the features and performance parameters.					√

Industry Relevant Skills

The following industry relevant competencies are expected to be developed in the student by undertaking the practical work of this laboratory.

1. Will be able to demonstrate various features of microprocessor, memory and I/O devices through microprocessor kit.
2. Will be able to design assembly language programs using appropriate 8085 instructions based on size and functions for conditional statements, branching, looping, subroutine
3. Will be able to design a given interfacing system using concepts of memory and I/O interfacing

Guidelines for Faculty members

1. Teacher should provide the guideline with demonstration of practical to the students with all features.
2. Teacher shall explain basic concepts/theory related to the experiment to the students before starting of each practical
3. Involve all the students in performance of each experiment.
4. Teacher is expected to share the skills and competencies to be developed in the students and ensure that the respective skills and competencies are developed in the students after the completion of the experimentation.
5. Teachers should give opportunity to students for hands-on experience after the demonstration.
6. Teacher may provide additional knowledge and skills to the students even though not covered in the manual but are expected from the students by concerned industry.
7. Give practical assignment and assess the performance of students based on task assigned to check whether it is as per the instructions or not.
8. Teacher is expected to refer complete curriculum of the course and follow the guidelines for implementation.

Instructions for Students

1. Students are expected to carefully listen to all the theory classes delivered by the faculty members and understand the COs, content of the course, teaching and examination scheme, skill set to be developed etc.
2. Students will have to perform experiments on 8085 microprocessor kit as well as on 8085 simulator.
3. Students should develop assembly language programs and execute all the programs on 8085 simulator. Students have to show output of each program in their practical file.
4. Students are instructed to submit practical list as per given sample list shown on next page.
5. Student should develop a habit of submitting the experimentation work as per the schedule and s/he should be well prepared for the same.

Common Safety Instructions

Students are expected to

1) Switch on the PC carefully (not to use wet hands)
2) Shutdown the PC properly at the end of your Lab
3) Carefully handle 8085 microprocessor kit
4) Carefully handle the peripherals (Mouse, Keyboard, Network cable etc)
5) Use Laptop in lab after getting permission from Teacher

Index

(Progressive Assessment Sheet)

Sr. No.	Objective(s) of Experiment	Page No.	Date of performance	Date of submission	Assessment Marks	Sign. of Teacher with date	Remarks
1.	Study the basic structure and functional units of digital computer.						
2.	Write the working of 8085 simulator GNUsim8085.						
3.	Study basic architecture of 8085 along with small introduction.						
4.	Write following 8085 assembly language programs. a. To add two 8-bit numbers and store result in memory location. b. To add five 8-bit numbers and store sum and carry in memory location. c. Perform addition of two 8-bit numbers and store result after using DAA instruction. d. Perform 16-bit subtraction. e. Perform multiplication of two 8-bit numbers using loop. f. Find minimum and maximum from five 8-bit numbers stored in memory location. g. To count the no. of 1's in given 8-bit value.						
5.	Write following 8085 assembly language programs. a. To add two arrays of five 8-bit values and store result in third array. b. To store sum of unpacked BCD numbers from packed BCD numbers. c. To store an ASCII value of each Hex digits. d. To set an MSB of an 8-bit values if it has even no. of 1's. e. To reverse an array of ten 8-bit values using subroutine. f. Find factorial of given 8-bit value. g. To perform division of 16-bit number by 8-bit number.						
6.	Design ALU using Logisim.						

7.	Implement Booth's multiplication algorithm.						
8.	Study interconnection structures of components of multiprocessor system.						
9.	Design circuits for interfacing memory and I/O with processor.						
10.	Study about different types of computer architectures and comprehend the features and performance parameters.						
Total							

Experiment No: 1

Date: ________________

Study the basic structure and functional units of digital computer.

Competency and Practical Skills: Not required

Relevant CO: CO1

Objectives:

a) To understand the structure and functional units of digital computer.
b) To understand the various input and output devices.

Equipment/Instruments: Not required

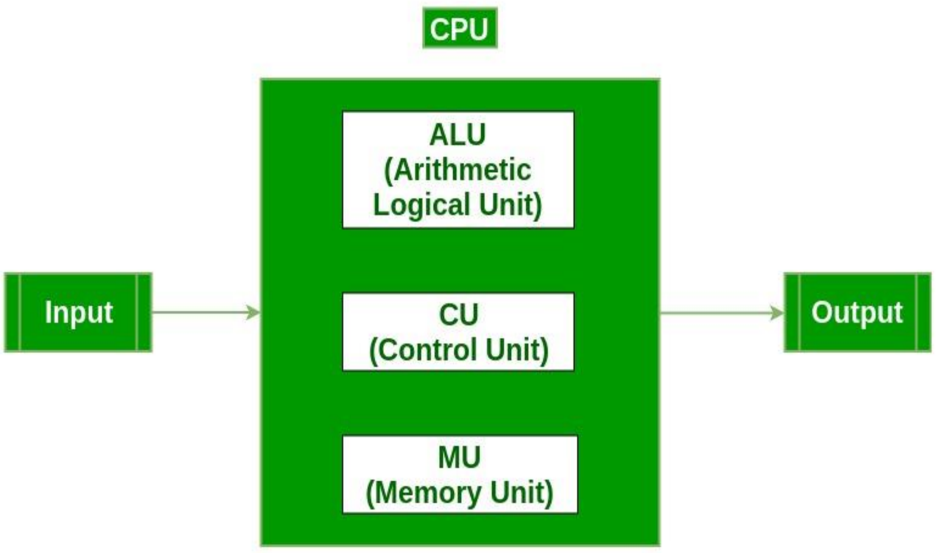
Explanation:

A computer is a combination of hardware and software resources which integrate together and provides various functionalities to the user. Hardware are the physical components of a computer like the processor, memory devices, monitor, keyboard etc. while software is the set of programs or instructions that are required by the hardware resources to function properly.

There are a few basic components that aid the working-cycle of a computer i.e. the Input-Process-Output Cycle and these are called as the functional components of a computer. It needs certain input, processes that input and produces the desired output. The input unit takes the input, the central processing unit does the processing of data and the output unit produces the output. The memory unit holds the data and instructions during the processing.

Digital Computer: A digital computer can be defined as a programmable machine which reads the binary data passed as instructions, processes this binary data, and displays a calculated digital output. Therefore, digital computers are those that work on the digital data.

Details of Functional Components of a Digital Computer



Input Unit:

The input unit consists of input devices that are attached to the computer. These devices take input and convert it into binary language that the computer understands. Some of the common input devices are keyboard, mouse, joystick, scanner etc.

Central Processing Unit (CPU):

Once the information is entered into the computer by the input device, the processor processes it. The CPU is called the brain of the computer because it is the control center of the computer. It first fetches instructions from memory and then interprets them so as to know what is to be done. If required, data is fetched from memory or input device. Thereafter CPU executes or performs the required computation and then either stores the output or displays on the output device. The CPU has three main components which are responsible for different functions – Arithmetic Logic Unit (ALU), Control Unit (CU) and Memory registers

Arithmetic and Logic Unit (ALU):

The ALU, as its name suggests performs mathematical calculations and takes logical decisions. Arithmetic calculations include addition, subtraction, multiplication and division. Logical decisions involve comparison of two data items to see which one is larger or smaller or equal.

Control Unit:

The Control unit coordinates and controls the data flow in and out of CPU and also controls all the operations of ALU, memory registers and also input/output units. It is also responsible for carrying out all the instructions stored in the program. It decodes the fetched instruction, interprets it and sends control signals to input/output devices until the required operation is done properly by ALU and memory.

Memory Registers:

A register is a temporary unit of memory in the CPU. These are used to store the data which is directly used by the processor. Registers can be of different sizes(16 bit, 32 bit, 64 bit and so on) and each register inside the CPU has a specific function like storing data, storing an instruction, storing address of a location in memory etc. The user registers can be used by an assembly language programmer for storing operands, intermediate results etc. Accumulator (ACC) is the main register in the ALU and contains one of the operands of an operation to be performed in the ALU.

Memory:

Memory attached to the CPU is used for storage of data and instructions and is called internal memory. The internal memory is divided into many storage locations, each of which can store data or instructions. Each memory location is of the same size and has an address. With the help of the address, the computer can read any memory location easily without having to search the entire memory. When a program is executed, its data is copied to the internal memory and is stored in the memory till the end of the execution. The internal memory is also called the Primary memory or Main memory. This memory is also called as RAM, i.e. Random Access Memory. The time of access of data is independent of its location in memory; therefore this memory is also called Random Access memory (RAM).

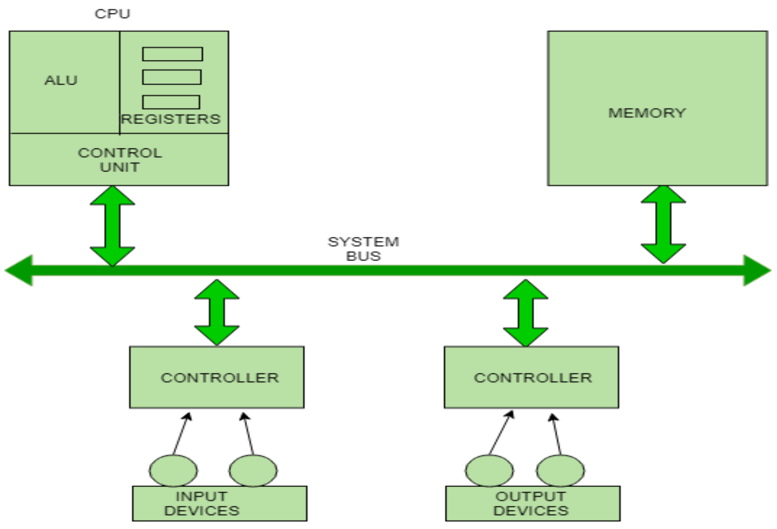
Output Unit:

The output unit consists of output devices that are attached with the computer. It converts the binary data coming from CPU to human understandable form. The common output devices are monitor, printer, plotter etc.

Interconnection between Functional Components

A computer consists of input unit that takes input, a CPU that processes the input and an output unit that produces output. All these devices communicate with each other through a common bus. A bus is a transmission path, made of a set of conducting wires over which data or information in the form of electric signals is passed from one component to another in a computer. The bus can be of three types – Address bus, Data bus and Control Bus.

Following figure shows the connection of various functional components:



The address bus carries the address location of the data or instruction. The data bus carries data from one component to another and the control bus carries the control signals. The system bus is the common communication path that carries signals to/from CPU, main memory and input/output devices. The input/output devices communicate with the system bus through the controller circuit which helps in managing various input/output devices attached to the computer.

Conclusion:

Quiz:

1) List all the components of digital computer.

Ans. A digital computer has the following components in it:

- Input device: Keyboard, Mouse, Mic, etc…
- Arithmetic and Logic Unit
- Control Unit
- Memory Unit
- Primary memory: RAM, ROM, PROM, EPROM, etc…
- Secondary memory: HDD, SSD, Pen drive, SD card, etc…
- Output device: Monitor, Printer, Speaker, etc…

2) What is memory unit?

Ans. A memory unit is a unit in a digital computer that stores data in it. Ex: RAM is a volatile memory unit that loads OS and other necessary programs when PC is on while ROM is a non-volatile memory unit that has some basic firmware like BIOS, etc…

HDD, SSD, etc… don’t come under memory unit as they are **storage** devices.

3) List the bus names with their functionality.

Ans. Buses are communication pathways that allow data to be transferred between different components of a computer. Each bus has a specific function and is responsible for transmitting different types of information. Here are some common bus names and their functionalities:

1. Address Bus:

Functionality: Carries memory addresses from the CPU to memory and other devices. It specifies the location in memory for read or write operations.

2. Data Bus:

Functionality: Transfers data between the CPU, memory, and other peripherals. It carries the actual data being processed.

3. Control Bus:

Functionality: Manages the control signals that coordinate and synchronize various operations within the computer. It includes signals for reading, writing, and interrupt requests.

4. System Bus:

Functionality: Refers collectively to the combination of the address bus, data bus, and control bus. It provides a communication path for all major components of the computer.

5. Front Side Bus (FSB):

Functionality: Connects the CPU to the Northbridge chip on the motherboard, facilitating communication between the CPU, memory, and other components.

6. Back Side Bus (BSB):

Functionality: Connects the CPU to the Level 2 (L2) cache. It allows the CPU to quickly access and retrieve data stored in the cache.

7. Expansion Bus:

Functionality: Connects peripheral devices, such as expansion cards (graphics cards, network cards, etc.), to the motherboard. Common examples include the PCI and PCIe buses.

8. Memory Bus:

Functionality: Connects the CPU to the computer's main memory (RAM). It is responsible for data transfer between the CPU and RAM.

9. Internal Bus:

Functionality: Handles communication within the internal components of the CPU, including the ALU (Arithmetic Logic Unit) and registers.

10. ISA (Industry Standard Architecture) Bus:

Functionality: An older bus standard used for connecting peripheral devices to the motherboard. It has been largely replaced by newer standards like PCI.

11. PCI (Peripheral Component Interconnect) Bus:

Functionality: A standard for connecting various hardware devices, including expansion cards, to the motherboard.

12. PCI Express (PCIe):

Functionality: An advanced version of PCI that provides high-speed data transfer between the motherboard and expansion cards. It is commonly used for graphics cards and other high-performance peripherals.

These buses collectively facilitate communication and data transfer, allowing different components of the computer to work together seamlessly.

Suggested Reference:

1. M. Morris Mano, "Computer System Architecture", Pearson Education
2. R.S.Gaonkar, "Microprocessor Architecture, Programming and Applications with 8085A", Penram International

Rubric wise marks obtained:

Knowledge of subject (2)		**Programming Skill**		**Team work (2)**		**Communication Skill (2)**		**Ethics (2)**	
Good (2)	Average (1)	Good (2)	Average (1)	Good (2)	Satisfactory (1)	Good (2)	Satisfactory (1)	Good (2)	Average (1)

Rubrics	**1**	**2**	**3**	**4**	**5**	**Total**
Marks						

Experiment No: 2

Date: __________________

Write the working of 8085 Simulator GNUsim8085.

Competency and Practical Skills: Not required

Relevant CO: CO1, CO2

Objectives:

a) To understand various components available in GNUsim8085.
b) To understand how to write an 8085 assembly language program.
c) To understand working of 8085 Simulator GNUsim8085.

Equipment/Instruments: Personal Computer, GNUsim8085

Explanation:

Introduction to GNU Simulator 8085

8085 simulator is software on which instructions are executed by writing the programs in assembly language.

GNUSim8085 is an 8085 microprocessor simulator with following features.

- A simple editor component with syntax highlighting.
- A keypad to input assembly language instructions with appropriate arguments.
- Easy view of register contents.
- Easy view of flag contents.
- Hexadecimal <--> Decimal converter.
- View of stack, memory and I/O contents.
- Support for breakpoints for programming debugging.
- Stepwise program execution.
- One click conversion of assembly program to opcode listing.
- Printing support (known not to work well on Windows).
- UI translated in various languages.

Writing a program in assembly language:-

Format of the instruction is as follows:-

Label	**Operation**	**Operands**	**Comments**
Its optional	Necessary	Necessary	Its optional

A basic assembly program consists of 4 parts.

1. Labels

2. Operations :- these operations can be specified as
    Machine operations (mnemonics):- used to define operations in the form of opcode as mention in the instruction set of microprocessor 8085.
    Pseudo operations (like preprocessor in C):- these are assembly directives.
3. Operands
4. Comments

In addition, you have **constants** in an assembly program. Unless otherwise specified, a constant which is always numeric is in decimal form. If appended with a character h it is assumed to be in hexadecimal form. If a hex constant starts with an alpha-char don't forget to include the number 0 in the beginning, since that will help the assembler to differentiate between a label and a constant.

Labels:-

When given to any particular instruction/data in a program, takes the address of that instruction or data as its value. But it has different meaning when given to EQU directive. Then it takes the operand of EQU as its value. Labels must always be placed in the first column and must be followed by an instruction (no empty line). Labels must be followed by a : (colon), to differentiate it from other tokens.

Operations:-

As mentioned above the operations can be specified in two ways that are **mnemonics** and **pseudo operation**.

Pseudo operations can be defined by using following directives:-

There are only 3 directives currently available in our assembly language.

1. DB - define byte (8 bits)
2. DS - define size (no. of bytes)
3. EQU - like minimalistic #define in C

DB is used to define space for an array of values specified by comma separated list. And the label (if given to the beginning of DB) is assigned the address of the first data item.

DS is used to define the specified number of bytes to be assigned and initialize them to zero. To access each byte you can use the + or -operator along with label.

EQU behaves similar to #define in C. But it is simple. It can be used to give names only to numeric constants. Nesting of EQU is not allowed. You can use EQU only in operands for pseudo ops and mnemonics.

Operands:-

Operands are specified according to the user. The register set specified in the architecture of 8085 (A, B, C, D, E, H and L) are used to access and store data. These registers are specified as operand. In case of accessing data or storing data in the memory ‘m’ is specified as an operand and the address of this memory location is taken from the HL pair (data in HL pair).

Lab requirements (details of H/W & S/W to be used)

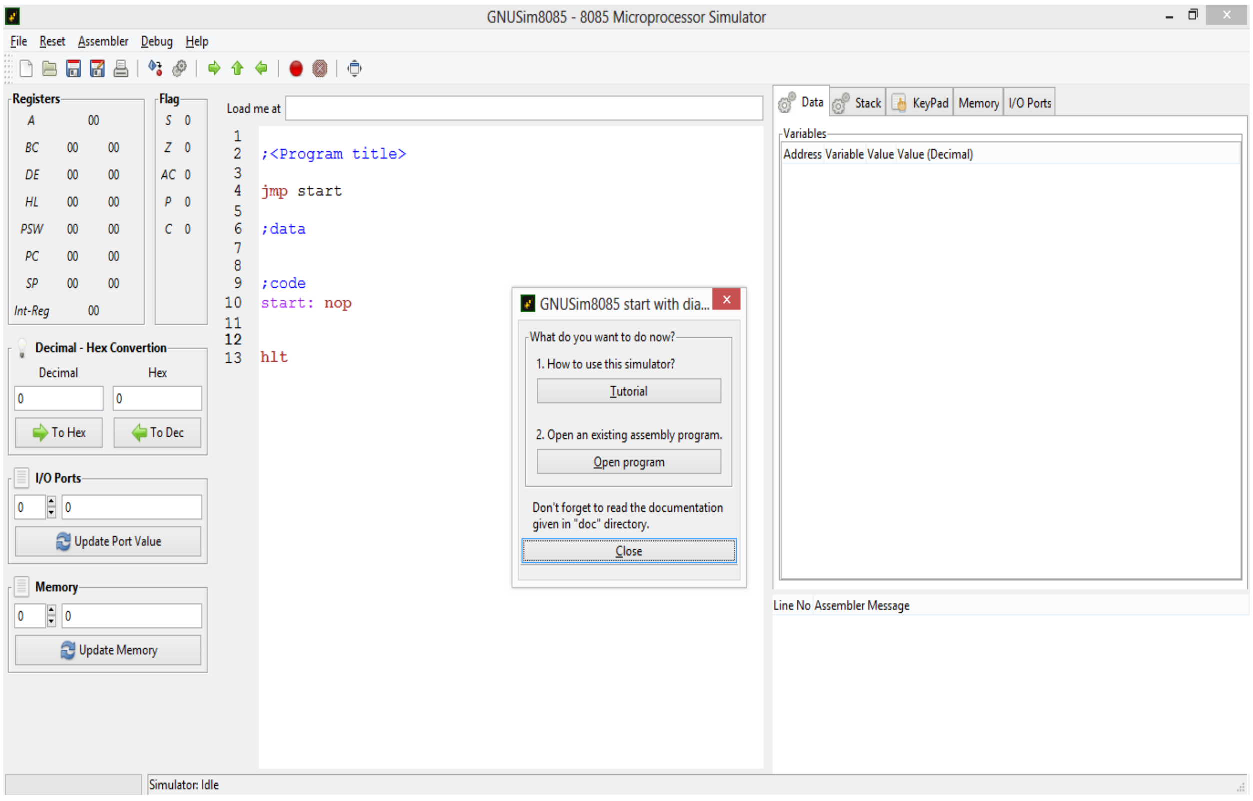
GNU Sim 8085 is an open source and is platform independent.

Software requirements: GNU Sim 8085

Operating System: Windows 7 or higher

Hardware requirements: P-IV C2D 2.9 GHZ, 320 GB HDD/2 GB RAM, Cabinet/1.44 FDD

How to start with GNU Simulator 8085


GNUSim8085 - 8085 Microprocessor Simulator
File Reset Assembler Debug Help
Registers
Flag
Load me at
Decimal - Hex Convertion
To Hex
To Dec
I/O Ports
Update Port Value
Memory
Update Memory
Data Stack KeyPad Memory I/O Ports
Variables
Address Variable Value Value (Decimal)
Line No Assembler Message
Simulator: Idle
GNUSim8085 start with dia...
Tutorial
Open program
Close

```
;<Program title>

jmp start

;data


;code
start: nop


hlt
```

What do you want to do now?

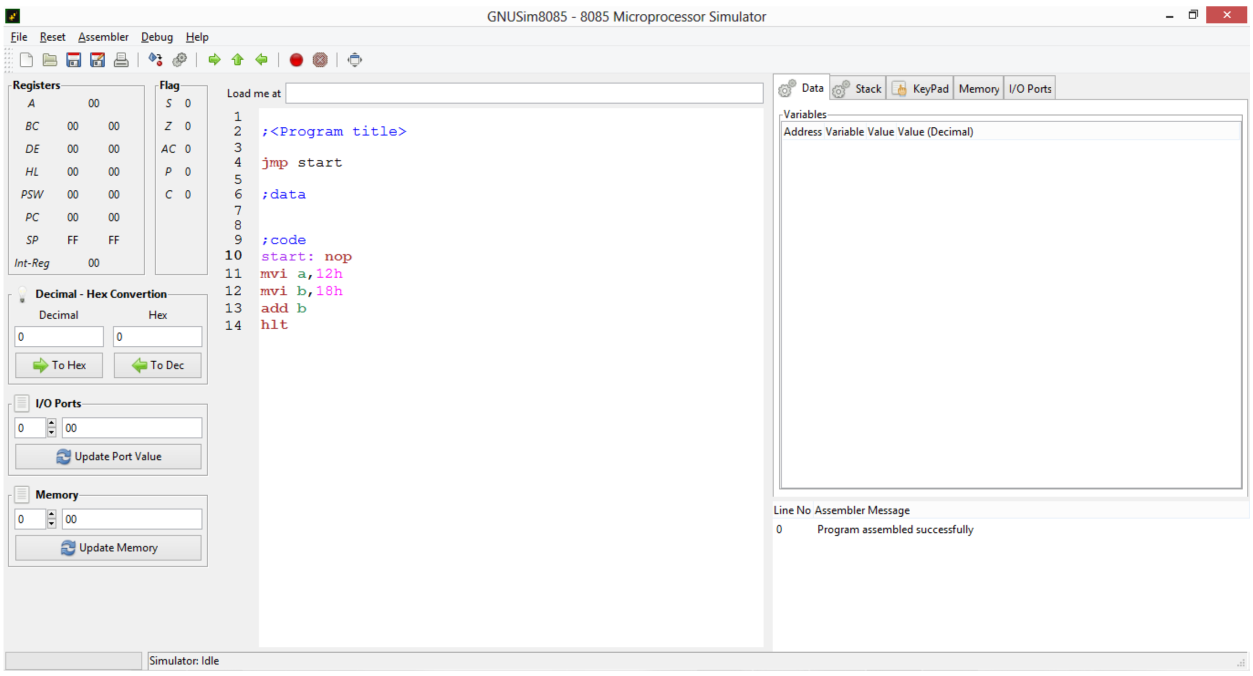
1. How to use this simulator?

2. Open an existing assembly program.

Don't forget to read the documentation given in "doc" directory.

Picture 1

Step 1: Open GNU Sim 8085 above window will open. Now click on close button highlighted in the above screen shot.

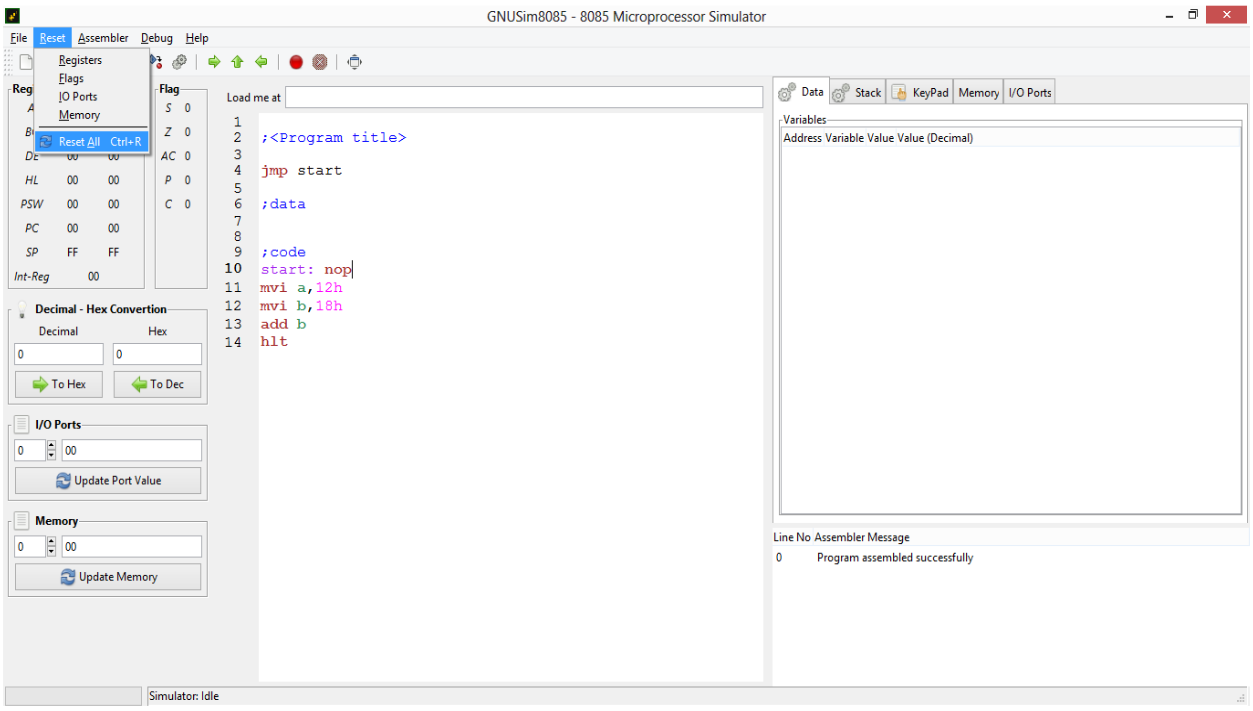

GNUSim8085 - 8085 Microprocessor Simulator
File Reset Assembler Debug Help
Registers
Flag
Load me at
Decimal - Hex Convertion
To Hex
To Dec
I/O Ports
Update Port Value
Memory
Update Memory
Data Stack KeyPad Memory I/O Ports
Variables
Address Variable Value Value (Decimal)
Line No Assembler Message
Simulator: Idle

```
;<Program title>

jmp start

;data


;code
start: nop
mvi a,12h
mvi b,18h
add b
hlt
```

```
0       Program assembled successfully
```

Picture 2

Step 2: Start writing the code after start: nop in load me at 10 that is at load me at 11.



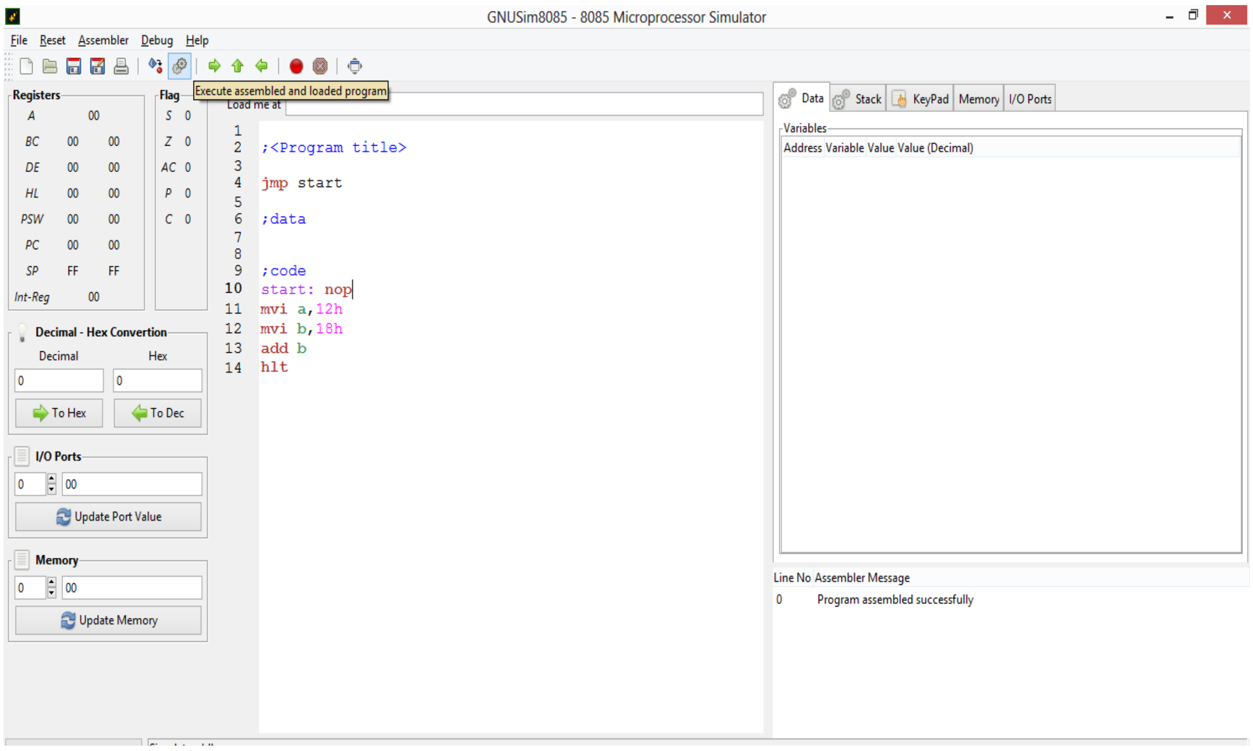
```
;<Program title>

jmp start

;data


;code
start: nop
mvi a,12h
mvi b,18h
add b
hlt
```

Address Variable Value Value (Decimal)

Line No Assembler Message
0 Program assembled successfully

Picture 3

Step 3: Click on reset and reset all the registers by clicking on reset all.


GNUSim8085 - 8085 Microprocessor Simulator
File Reset Assembler Debug Help
Execute assembled and loaded program
Load me at
Data Stack KeyPad Memory I/O Ports
Variables
Decimal - Hex Convertion
To Hex
To Dec
I/O Ports
Update Port Value
Memory
Update Memory

Registers		
A	00	
BC	00	00
DE	00	00
HL	00	00
PSW	00	00
PC	00	00
SP	FF	FF
Int-Reg	00	

Flag	
S	0
Z	0
AC	0
P	0
C	0

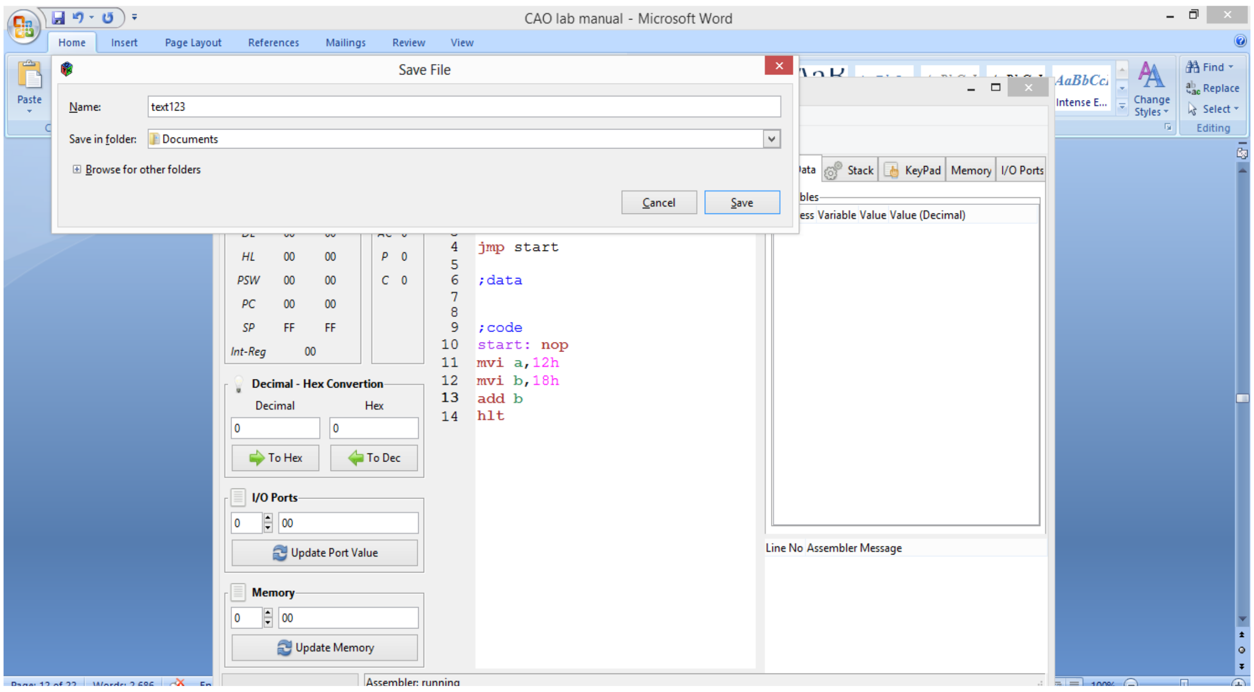
```
;<Program title>

jmp start

;data


;code
start: nop
mvi a,12h
mvi b,18h
add b
hlt
```

Address Variable Value Value (Decimal)

Line No Assembler Message
0 Program assembled successfully

Picture 4

Step 4: click on the highlighted button to execute the code



Save File

Name: text123

Save in folder: Documents

Browse for other folders

Cancel Save

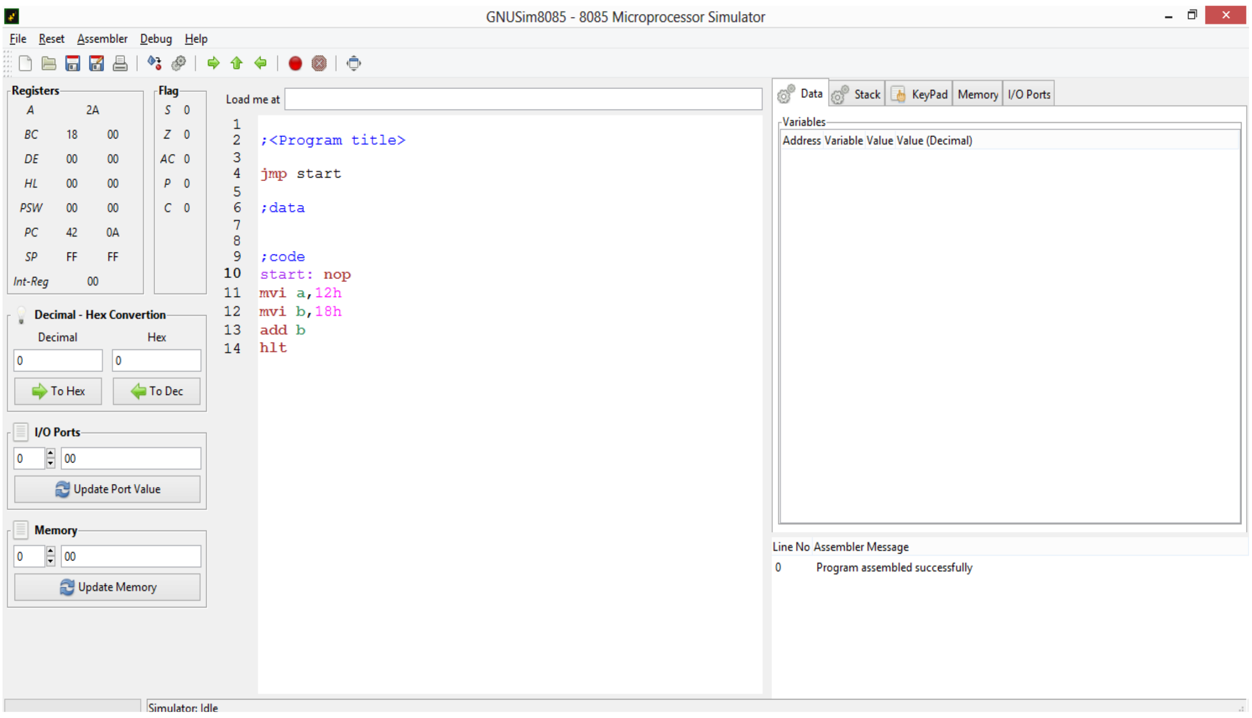
HL	00	00
PSW	00	00
PC	00	00
SP	FF	FF
Int-Reg	00	

```
jmp start

;data


;code
start: nop
mvi a,12h
mvi b,18h
add b
hlt
```

Assembler: running

Picture 5

Step 5: after you execute the code mention the name your program by writing the name in the name section as mentioned in the screen shot in picture 5 and the drive where you want to save it. After that click on save.



GNUSim8085 - 8085 Microprocessor Simulator

Registers		
A	2A	
BC	18	00
DE	00	00
HL	00	00
PSW	00	00
PC	42	0A
SP	FF	FF
Int-Reg	00	

Flag	
S	0
Z	0
AC	0
P	0
C	0

```
;<Program title>

jmp start

;data


;code
start: nop
mvi a,12h
mvi b,18h
add b
hlt
```

Line No	Assembler Message
0	Program assembled successfully

Simulator: Idle

Picture 6

Step 6: after this you will see the result of the instructions in the respective registers as seen in the above picture 6.

Conclusion:

Quiz:

1) Explain the format of the instruction.
2) How we can set the value at memory location and I/O port?
3) List any five instruction of 8085microprocessor

Suggested Reference:

1. M. Morris Mano, “Computer System Architecture”, Pearson Education
2. R.S.Gaonkar, “Microprocessor Architecture, Programming and Applications with 8085A”, Penram International
3.

Rubric wise marks obtained:

Knowledge of subject (2)		**Programming Skill**		**Team work (2)**		**Communication Skill (2)**		**Ethics (2)**	
Good (2)	Average (1)	Good (2)	Average (1)	Good (2)	Satisfactory (1)	Good (2)	Satisfactory (1)	Good (2)	Average (1)

Rubrics	**1**	**2**	**3**	**4**	**5**	**Total**
Marks						

Experiment No: 3

Date: ________________

Study basic architecture of 8085 along with small introduction.

Competency and Practical Skills: Not required

Relevant CO: CO1, CO3

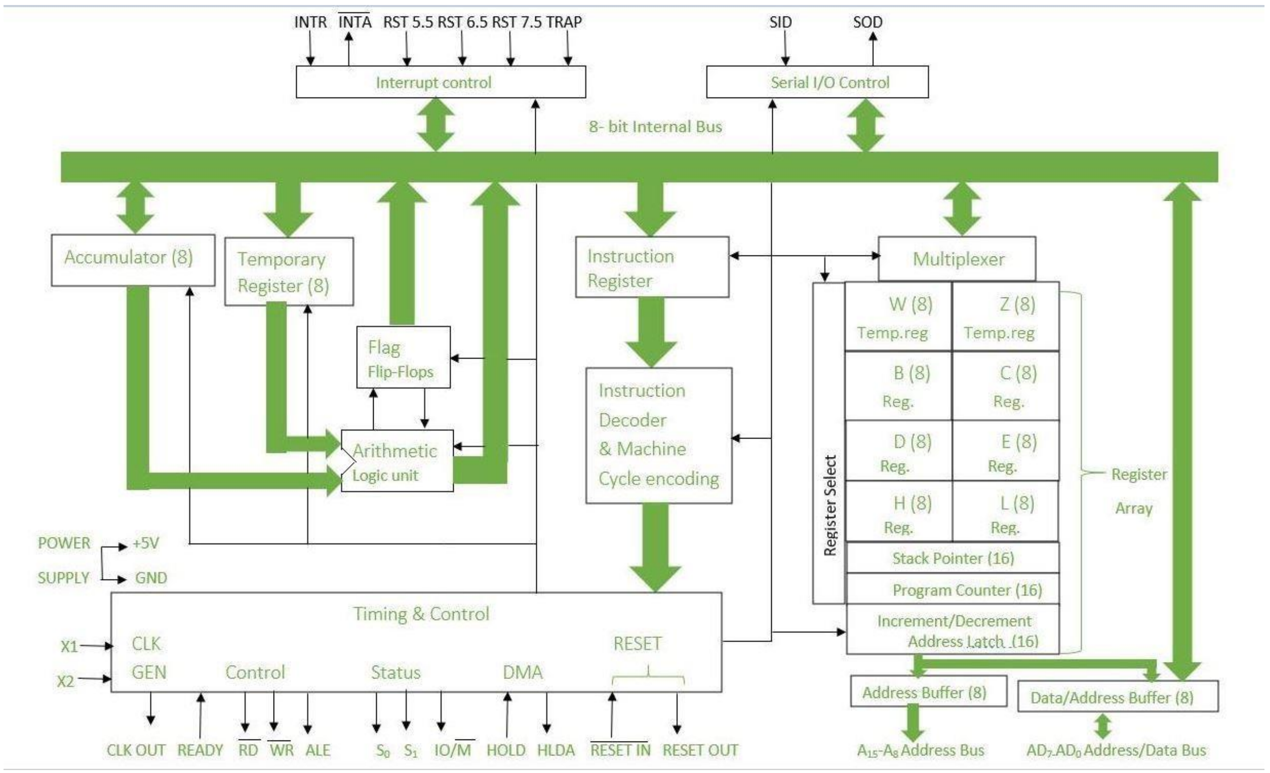
Objectives:

a) Get the knowledge about various components of 8085 microprocessor.
b) To understand the importance of components.

Equipment/Instruments: Not required

Explanation:

The architecture of the 8085 microprocessor consists of several key components, including the accumulator, registers, program counter, stack pointer, instruction register, flags register, data bus, address bus, and control bus.



8085 is an 8-bit, general-purpose microprocessor. It consists of the following functional units:

Arithmetic and Logic Unit (ALU):

It is used to perform mathematical operations like addition, multiplication, subtraction, division, decrement, increment, etc. Different operations are carried out in ALU: Logical operations, Bit-Shifting Operations, and Arithmetic Operations.

Flag Register:

It is an 8-bit register that stores either 0 or 1 depending upon which value is stored in the accumulator. Flag Register contains 8-bit out of which 5-bits are important and the rest of 3-bits are "don't Care conditions". The flag register is a dynamic register because after each operation to check whether the result is zero, positive or negative, whether there is any overflow occurred or not, or for comparison of two 8-bit numbers carry flag is checked. So for numerous operations to check the contents of the accumulator and from that contents if we want to check the behavior of given result then we can use Flag register to verify and check. So we can say that the flag register is a status register and it is used to check the status of the current operation which is being carried out by ALU.

Different Flags are: Carry Flag, Parity Flag, Auxiliary Carry Flag, Zero Flag, Sign Flag

Accumulator:

Accumulator is used to perform I/O, arithmetic, and logical operations. It is connected to ALU and the internal data bus. The accumulator is the heart of the microprocessor because for all arithmetic operations Accumulator's 8-bit pin will always there connected with ALU and in most-off times all the operations carried by different instructions will be stored in the accumulator after operation performance.

General Purpose Registers:

There are six general-purpose registers. These registers can hold 8-bit values. These 8-bit registers are B, C, D, E, H, L. These registers work as 16-bit registers when they work in pairs like B-C, D-E, and H-L. Here registers W and Z are reserved registers. We can't use these registers in arithmetic operations. It is reserved for microprocessors for internal operations like swapping two 16-bit numbers. We know that to swap two numbers we need a third variable hence here W-Z register pair works as temporary registers and we can swap two 16-bit numbers using this pair.

Program Counter:

Program Counter holds the address value of the memory to the next instruction that is to be executed. It is a 16-bit register.

Stack Pointer:

It works like a stack. In stack, the content of the register is stored that is later used in the program. It is a 16-bit special register. The stack pointer is part of memory but it is part of Stack operations, unlike random memory access. Stack pointer works in a continuous and contiguous part of the memory. Whereas Program Counter (PC) works in random memory locations. This pointer is very useful in stack-related operations like PUSH, POP, and nested CALL requests initiated by Microprocessor. It reserves the address of the most recent stack entry.

Temporary Register:

It is an 8-bit register that holds data values during arithmetic and logical operations.

Instruction register and decoder:

It is an 8-bit register that holds the instruction code that is being decoded. The instruction is fetched from the memory.

Timing and control unit:

The timing and control unit comes under the CPU section, and it controls the flow of data from the CPU to other devices. It is also used to control the operations performed by the microprocessor and the devices connected to it. There are certain timing and control signals like Control signals, DMA Signals, RESET signals and Status signals.

Data bus, Address bus, Control bus:

The **data bus** is an 8-bit bus that is used to transfer data between the microprocessor and memory or other devices. The data bus is bidirectional, which means that it can be used to read data from memory or write data to memory.

The **address bus** is a 16-bit bus that is used to address memory and other devices. The address bus is used to select the memory location or device that the microprocessor wants to access.

The **control bus** is a set of signals that controls the operations of the microprocessor, including the read and write operations. The control bus includes signals such as the read signal, write signal, interrupt signal, and reset signal. The read signal is used to read data from memory or other devices, the write signal is used to write data to memory or other devices, the interrupt signal is used to signal the microprocessor that an interrupt has occurred, and the reset signal is used to reset the microprocessor to its initial state.

Conclusion:

Quiz:

1) List different registers of 8085 microprocessor with their importance.

2) Explain flags of 8085 microprocessor?

3) What are the data bus, address bus and control bus?

Suggested Reference:

1. R.S.Gaonkar, "Microprocessor Architecture, Programming and Applications with 8085A", Penram International
2.

Rubric wise marks obtained:

Knowledge of subject (2)		**Programming Skill**		**Team work (2)**		**Communication Skill (2)**		**Ethics (2)**	
Good (2)	Average (1)	Good (2)	Average (1)	Good (2)	Satisfactory (1)	Good (2)	Satisfactory (1)	Good (2)	Average (1)

Rubrics	**1**	**2**	**3**	**4**	**5**	**Total**
Marks						

Experiment No: 4

Date: ___________________

Write following 8085 assembly language programs with output.

a. To add two 8-bit numbers and store result in memory location.
b. To add five 8-bit numbers and store sum and carry in memory location.
c. Perform addition of two 8-bit numbers and store result after using DAA instruction.
d. Perform 16-bit subtraction.
e. Perform multiplication of two 8-bit numbers using loop.
f. Find minimum and maximum from five 8-bit numbers stored in memory location.
g. To count the no. of 1's in given 8-bit value.

Competency and Practical Skills: understanding of working of registers, instruction set of 8085

Relevant CO: CO2

Objectives:

a) To understand working of the 8085 instruction set.
b) To understand the behavior of the flags and how to use instructions.

Equipment/Instruments: Personal Computer, GNUsim8085

Explanation:

a. To add two 8-bit numbers and store result in memory location.

MVI A, 24H	; Store 24H in A register
MVI B, 75H	; Store 75H in B register
ADD B	; Add register B contents to A and store the sum in A register
STA 0002H	; Store the sum from A register on memory location 0002H
HLT	; Stop the execution

LDA 2050H	; Read 8-bit value from memory location 2050H in A register
MOV B, A	; Copy value of A register in B
LDA 2051H	; Read 8-bit value from memory location 2051H in A register
ADD B	; Add register B contents to A and store the sum in A register
STA 2052H	; Store the sum from A register on memory location 2052H
HLT	; Stop the execution

LXI H, 2050H	; Load 16-Bit address in HL register pair (location of first value)
MOV A, M	; Copy value from location stored in HL register pair to A
INX H	; Increment address in HL register pair (location of second value)
ADD M	; Add second value from M to A and store the sum in A register
INX H	; Increment address in HL register pair (location of result to be stored)

MOV M, A	; Store the value from A register to M (location point by HL register pair)
HLT	; Stop the execution

Output:

b. To add five 8-bit numbers and store sum and carry in memory location.

JMP START	; Jump on start label
DATA: DB 01H, 05H, 08H, 99H, 12H	; Five 8-bit values stored named as DATA
COUNT: DB 05H	; Count value is stored in COUNT
SUM: DB 00H	; Initial value of SUM is 00H
CARRY: DB 00H	; Initial value of CARRY is 00H
START: LDA COUNT	; Load COUNT value in A register
MOV C, A	; Copy COUNT value in C register from A
MVI A, 00H	; Clear value of A register (to store sum)
MVI B, 00H	; Clear value of B register (to store carry)
LXI H, DATA	; Load 16-Bit address of DATA in HL register pair
LOOP: ADD M	; Add first value to A register
JNC NEXT	;Check if Carry flag is not set, take jump on label NEXT
INR B	; if Carry flag is set increment B register value
NEXT: INX H	;Increment address in HL register pair (for other values)
DCR C	; Decrement C register value (to decrement count value)
JNZ LOOP	; if Zero flag is not set, take jump on label LOOP
STA SUM	; Store sum from A register in SUM
MOV A, B	; Copy B register value to A (to store carry in CARRY)
STA CARRY	; Store carry from A register to CARRY
HLT	; Stop the execution

Output:

c. Perform addition of two 8-bit numbers and store result after using DAA instruction.

```
MVI A,  05H     ; Store 8-bit value in A register
MVI B,  07H     ; Store 8-bit value in B register
ADD B           ; Add value of B register to A,   sum is in A register ( 05H + 07H = 0CH )
DAA             ; Decimal Adjust Accumulator (convert value higher than 09 to BCD)
STA 2050H       ; Store the result on the location 2050H
HLT             ; Stop the execution
```

Output:

d. Perform 16-bit subtraction.

```
LHLD 2050       ; Load H-L pair with value of address 2050H (in L) and 2051H (in H)
XCHG            ; Exchange HL register pair value with DE register pair (D<->H,   E<->L)
LHLD 2052       ; Load H-L pair with value of address 2052H (in L) and 2053H (in H)
MOV A,   E      ; Copy value of E (Lower byte of first 16-bit value) register to A
SUB L           ; Subtract value of L (Lower byte of second 16-bit value) register from A
STA 2054        ; Store the result (lower byte subtraction) on location 2054H
MOV A,   D      ; Copy value of D (Higher byte of first 16-bit value) register to A
SBB H           ; Subtract value of H (Higher byte of second 16-bit value) register from A
STA 2055        ; Store the result (lower byte subtraction) on location 2055H
HLT             ; Stop the execution
```

Output:

e. Perform multiplication of two 8-bit numbers using loop.

```
MVI A,  00H     ; Clear the A register value
MVI B,  03H     ; Store 8-bit value in B register
MVI C,  05H     ; Store 8-bit value in C register
LOOP: ADD C     ; Add content of C register to A (B times)
DCR B           ; Decrement the value of B register
JNZ LOOP        ; Check if Zero flag is not set,   take a jump on label LOOP
STA 2050H       ; Store the result of multiplication from A register to location 2050H
HLT              ; Stop the execution
```

Output:

f. Find minimum and maximum from five 8-bit numbers stored in memory location.

JMP START	; Jump on start label
DATA: DB 56H, 32H, 59H, 01H, 23H	; Five 8-bit values stored named as DATA
MAX: DB 00H	; Initial value of MAX is 00H
MIN: DB 00H	; Initial value of MIN is 00H
START: MVI C, 05H	; Initial value of C register is 05H (work as counter)
LXI H, DATA	; Load 16-Bit address of DATA in HL register pair
MOV B, M	; Copy first value in B register from M (B will store Max value)
MOV D, M	; Copy first value in D register from M (D will store Min value)
LOOP: MOV A, M	; Copy first value in A register
CMP B	; Compare the value of A and B register
JC MINI	; Check if Carry flag is set, take a jump on label MINI
MOV B, A	; if Carry flag is not set, copy the value of A register to B
MINI: CMP D	; Compare the value of A and D register
JNC SKIP	; if Carry flag is not set, take a jump on label SKIP
MOV D, A	; if Carry flag is set, copy the value of A register to D
SKIP: INX H	; Increment address in HL register pair (for other values)
DCR C	; Decrement C register value (to decrement count value)
JNZ LOOP	; if Zero flag is not set, take jump on label LOOP
MOV A, B	; Store B register value in A (Max value)
STA MAX	; Store value from A register to MAX
MOV A, D	; Store D register value in A (Min value)
STA MIN	; Store value from A register to MIN
HLT	; Stop the execution

Output:

g. To count the no. of 1's in given 8-bit value.

MVI A, 67H	; Load 8-bit value in A register

MVI B, 00H	; Clear the B register value (to count no. of 1's)
MVI C, 08H	; Set the value of C register to 08H (to rotate 8 times)
LOOP: RLC	; Rotate A left
JNC NEXT	; Check if Carry flag is not set, take jump on label NEXT
INR B	; if Carry flag is set increment B register value
NEXT: DCR C	; Decrement the value of C register
JNZ LOOP	; Check if Zero flag is not set, take a jump on label LOOP
MOV A, B	; Copy the count of 1's from B register to A
STA 2050H	; Store the result (count of 1's) of from A register to location 2050H
HLT	; Stop the execution

Output:

Conclusion:

Quiz:

1) Explain difference of LHLD and LXI instruction.

2) What is the purpose of the DAA instruction?

3) Explain JNC and JNZ instruction.

Suggested Reference:

1. R.S.Gaonkar, "Microprocessor Architecture, Programming and Applications with 8085A", Penram International

Rubric wise marks obtained:

Knowledge of subject (2)		**Programming Skill**		**Team work (2)**		**Communication Skill (2)**		**Ethics (2)**	
Good (2)	Average (1)	Good (2)	Average (1)	Good (2)	Satisfactory (1)	Good (2)	Satisfactory (1)	Good (2)	Average (1)

Rubrics	**1**	**2**	**3**	**4**	**5**	**Total**
Marks						

Experiment No: 5

Date: ___________________

Write following 8085 assembly language programs with output.

a. To add two arrays of five 8-bit values and store result in third array.
b. To store sum of unpacked BCD numbers from packed BCD numbers.
c. To store an ASCII value of each Hex digits.
d. To set an MSB of an 8-bit values if it has even no. of 1's.
e. To reverse an array of ten 8-bit values using subroutine.
f. Find factorial of given 8-bit value.
g. To perform division of 16-bit number by 8-bit number.

Competency and Practical Skills: understanding of working of registers, instruction set of 8085

Relevant CO: CO2

Objectives:

a) To understand working of the 8085 instruction set.
b) To understand the behavior of the flags and how to use flag instructions.

Equipment/Instruments: Personal Computer, GNUsim8085

Explanation:

a. To add two arrays of five 8-bit values and store result in third array.

JMP START	; Jump on start
DATA1: DB 01, 02, 03, 04, 05	; First Array
DATA2: DB 06, 07, 08, 09, 10	; Second Array
SUM: DS 5	; Third array to store sum
COUNT: DB 05	; Counter value
START: LXI B, DATA1	; Load address of DATA1 in BC reg. pair
LXI H, DATA2	; Load address of DATA2 in HL reg. pair
LXI D, SUM	; Load address of SUM in DE reg. pair
LOOP: LDAX B	; Start of LOOP; Read values from DATA1
ADD M	; Add values from DATA2
STAX D	; Store sum in SUM
INX B	; Increment location of DATA1
INX H	; Increment location of DATA2
INX D	; Increment location of SUM
LDA COUNT	; Load count in A register
DCR A	; Decrement count value in A register
STA COUNT	; Store count from A register
JNZ LOOP	; Check Zero flag, if it is not set go back to LOOP

HLT	; Stop the execution

Output:

b. To store sum of unpacked BCD numbers from packed BCD numbers.

JMP START	; Jump on start
PACKBCD: DB 26H	; Packed BCD number
SUM: DB 00H	; To store sum of digits of Packed BCD number
START: LDA PACKBCD	; Load Packed BCD number to A register
MOV B, A	; Copy number to B register
ANI 0FH	; Perform AND operation with 0Fh to clear first four bits
MOV C, A	; Move last hex digit from A to C register
MOV A, B	; Move Packed BCD number to A register
ANI 0F0H	; Perform AND operation with 0F0h to clear last four bits
RLC	; Rotate A register value 1-bit left
RLC	; Rotate A register value 1-bit left
RLC	; Rotate A register value 1-bit left
RLC	; Rotate A register value 1-bit left (First hex digit is in last position)
ADD C	; Add another digit from C register
STA SUM	; Store the sum on SUM
HLT	; Stop the execution

Output:

c. To store an ASCII value of each Hex digits.

JMP START	; Jump on start
DATA: DB 0AH	; Hexadecimal Digit
ASCII: DB 00H	; To store Ascii value

START: LDA DATA	; Load Hexadecimal Digit
CPI 0AH	; Compare Hex digit with 0Ah (to check Number of Leter)
JC NEXT	; If carry flag is set take jump on next
ADI 07H	; Add 07h to the value in A register
NEXT: ADI 30H	; Add 30H to the value in A register
STA ASCII	; Store the Ascii value on ASCII
HLT	; Stop the execution

Output:

d. To set an MSB of an 8-bit values if it has even no. of 1's.

JMP START	; Jump on start
NUM: DB 06H	; Store 8-bit number in NUM
RESULT: DB 00H	; To store result
START: LDA NUM	; Load NUM in A register
MOV D, A	; Copy NUM from A register to D
MVI B, 00H	; Clear B register value to store count of 1's
MVI C, 07H	; Store 07 in C register to rotate value 7 times right
LOOP: RRC	; Rotate A register value right
JNC NEXT	; If carry flag is not set, take a jump on NEXT
INR B	; If carry flag is set, increment value of B register
NEXT: DCR C	; Decrement counter value in C register
JNZ LOOP	; If Zero flag is not set, take a jump on LOOP
MOV A, B	; Move count of 1's to A register
RRC	; Rotate right to check LSB of count of 1's
JC NEXT1	; if Carry flag is set, take jump on NEXT1 (no. of 1's are odd)
MOV A, D	; Restore NUM in A register
ORI 80H	; Perform OR operation with 80H(10000000) to set MSB in NUM
NEXT1: STA RESULT	; Store the result
HLT	; Stop the execution

Output:

e. To reverse an array of ten 8-bit values using subroutine.

JMP START	; Jump on start
DATA: DB 01, 02,03, 04, 05,06, 07, 08, 09, 10	; Store ten 8-bit values with DATA
START: LXI H, DATA	; Load first address of DATA in HL reg pair
LXI D, DATA+9	; Load last address of DATA in DE reg pair
MVI C, 05H	; Set C register value to 05 for counter
LOOP: CALL SWAP	; Call fun to swap values of address in HL and DE reg. pair
INX H	; Go to next location of DATA in forward
DCX D	; Decrement address for location of DATA in backward
DCR C	; Decrement counter
JNZ LOOP	; if Zero flag is not set, take a jump on LOOP
HLT	; Stop the execution
SWAP: LDAX D	; Subroutine Load values from the location in DE reg. pair
MOV B, A	; Move value to B register
MOV A, M	; Move value to A from the location in HL reg. pair
STAX D	; Store the value on the location in DE reg. pair
MOV M, B	; Store the value on the location in HL reg. pair
RET	; Return from the subroutine

Output:

f. Find factorial of given 8-bit value.

JMP START	; Jump on start
NUM: DB 05H	; Store 8-bit number in NUM
ANS: DB 00H	; To store the factorial
START: LDA NUM	; Load NUM to A register
MOV B, A	; Move value to B from A reg.
MVI D, 01H	; Set value of D reg. to 01
FACT: CALL MUL	; Call subroutine
DCR B	; Decrement B register value

JNZ FACT	; If Zero flag is not jump on FACT
MOV A, D	; Copy value of D reg. to A
STA ANS	; Store the result in ANS
HLT	; Stop the execution
MUL: MOV E, B	; Subroutine Move value of B reg to A
XRA A	; Clear value of A reg
ML: ADD D	; Add value of D reg
DCR E	; Decrement value of E reg.
JNZ ML	; If Zero flag is not set, take jump on ML
MOV D, A	; Store sum in D reg.
RET	; Return from Subroutine

Output:

g. To perform division of 16-bit number by 8-bit number.

LXI H, 8000H	; Point 8000H address
MOV A, M	; Store the lower order byte
INX H	; Increase the HL pair to point next loc
MOV B, M	; Store the higher order byte
INX H	; Increase the HL pair to point next loc
MOV C, M	; Load the denominator
INR B	; Increase B register
LXI H, 0000H	; Store 0000Hinto HL pair
LOOP: SUB C	; Subtract C from acc
JC SKIP	; Jump to SKIP when CY = 1
INCR: INX H	; Increase quotient part
JMP LOOP	; Jump to LOOP
SKIP: DCR B	; Decrease B
JZ STORE	; Jump to STORE when Z = 1
JMP INCR	; Jump to INCR
STORE: ADD C	; Add C with Acc
XCHG	; swap DE and HL pair contents
LXI H, 8050H	; Load the destination address
MOV M, E	; Store the lower order quotient
INX H	; Increase HL pair
MOV M, D	; Store the higher order quotient
INX H	; Increase HL pair
MOV M, A	; Store the remainder

HLT	; Stop the execution

Output:

Conclusion:

Quiz:

1) Explain XCHG instruction.

2) What is the purpose of the DAA instruction?

3) Explain JNC and JNZ instruction.

Suggested Reference:

1. R.S.Gaonkar, “Microprocessor Architecture, Programming and Applications with 8085A”, Penram International

Rubric wise marks obtained:

Knowledge of subject (2)		**Programming Skill**		**Team work (2)**		**Communication Skill (2)**		**Ethics (2)**	
Good (2)	Average (1)	Good (2)	Average (1)	Good (2)	Satisfactory (1)	Good (2)	Satisfactory (1)	Good (2)	Average (1)

Rubrics	**1**	**2**	**3**	**4**	**5**	**Total**
Marks						

Experiment No: 6

Date: ________________

Design ALU using Logisim.

Competency and Practical Skills: working of ALU, arithmetic and logical operations, concepts of Digital fundamentals for logic gates, adders, etc.

Relevant CO: CO3

Objectives:

a) To understand the arithmetic and logical operations.
b) To understand the necessary connections from inputs to the required components.

Equipment/Instruments: Logisim

Explanation:

Conclusion:

Quiz:

1) Perform 4-bit binary addition/subtraction operation.

2) Perform 4-bit AND operation.

3) Perform 4-bit OR operation.

Suggested Reference:

1. M. Morris Mano, “Computer System Architecture”, Pearson Education
2. M. Morris Mano, “Digital Logic and Computer Design”, Pearson Education

Rubric wise marks obtained:

Knowledge of subject (2)		**Programming Skill**		**Team work (2)**		**Communication Skill (2)**		**Ethics (2)**	
Good (2)	Average (1)	Good (2)	Average (1)	Good (2)	Satisfactory (1)	Good (2)	Satisfactory (1)	Good (2)	Average (1)

Rubrics	**1**	**2**	**3**	**4**	**5**	**Total**
Marks						

Experiment No: 7

Date: ___________________

Implement Booth's multiplication algorithm.

Competency and Practical Skills: Knowledge of any programming language

Relevant CO: CO3

Objectives:

a) To understand the working of Booth's algorithm.
b) To understand the necessary connections from inputs to the required components.

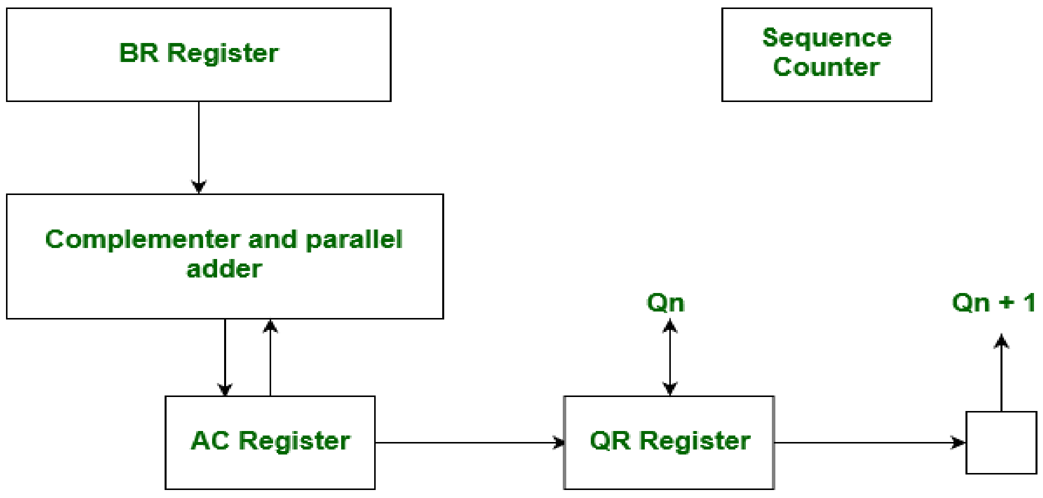
Equipment/Instruments: C or any Programming language

Explanation:

Booth algorithm gives a procedure for multiplying binary integers in signed 2's complement representation in efficient way, i.e., less number of additions/subtractions required. It operates on the fact that strings of 0's in the multiplier require no addition but just shifting and a string of 1's in the multiplier from bit weight 2^k to weight 2^m can be treated as $2^{(k+1)}$ to 2^m. As in all multiplication schemes, booth algorithm requires examination of the multiplier bits and shifting of the partial product. Prior to the shifting, the multiplicand may be added to the partial product, subtracted from the partial product, or left unchanged according to following rules:

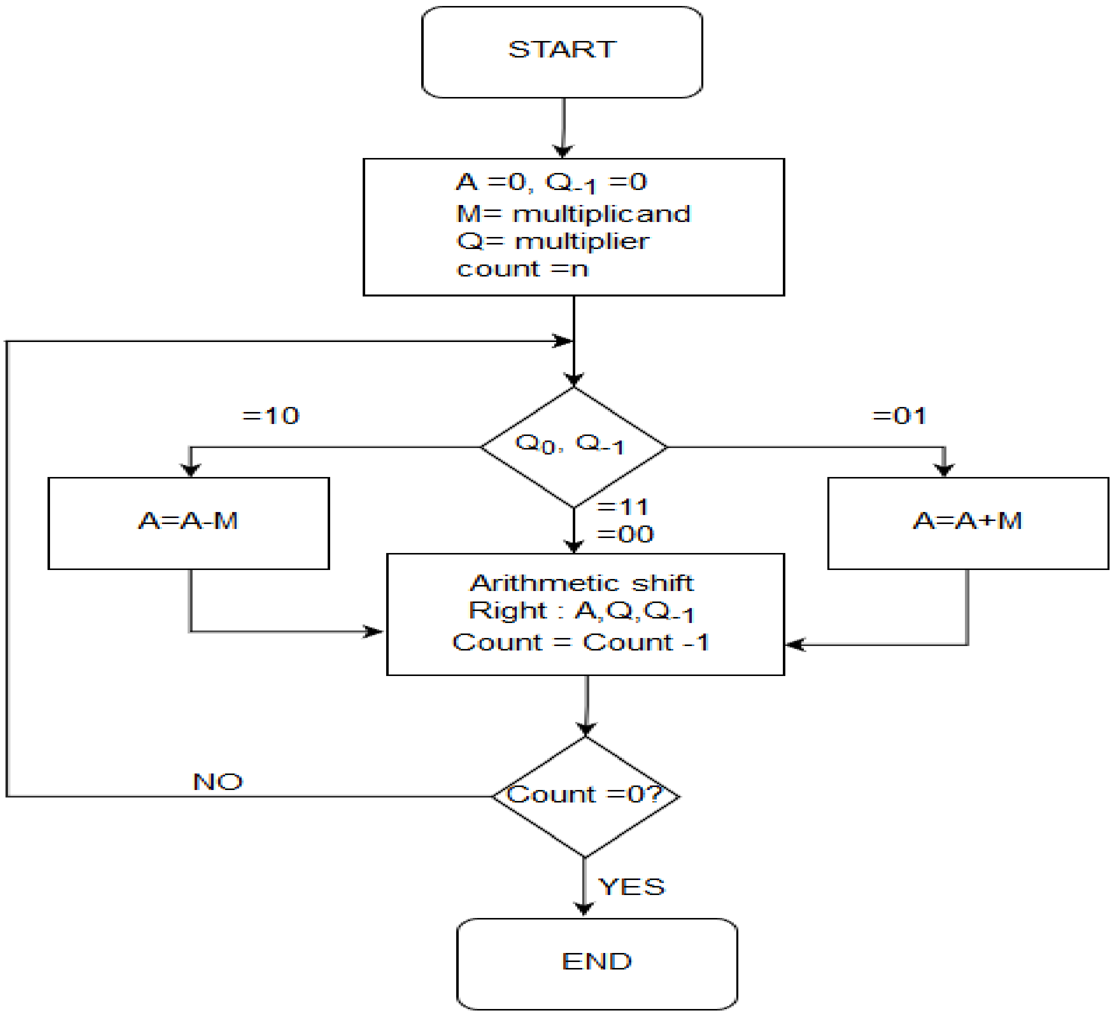
1. The multiplicand is subtracted from the partial product upon encountering the first least significant 1 in a string of 1's in the multiplier
2. The multiplicand is added to the partial product upon encountering the first 0 (provided that there was a previous '1') in a string of 0's in the multiplier.
3. The partial product does not change when the multiplier bit is identical to the previous multiplier bit.

Hardware Implementation of Booths Algorithm – The hardware implementation of the booth algorithm requires the register configuration shown in the figure below.



Booth's Algorithm Flowchart:

We name the register as A, B and Q, AC, BR and QR respectively. Qn designates the least significant bit of multiplier in the register QR. An extra flip-flop Qn+1is appended to QR to facilitate a double inspection of the multiplier. The flowchart for the booth algorithm is shown below.



AC and the appended bit Qn+1 are initially cleared to 0 and the sequence SC is set to a number n equal to the number of bits in the multiplier.

The two bits of the multiplier in Qn and Qn+1are inspected.

a. If the two bits are equal to 10, it means that the first 1 in a string has been encountered. This requires subtraction of the multiplicand from the partial product in AC.
b. If the 2 bits are equal to 01, it means that the first 0 in a string of 0's has been encountered. This requires the addition of the multiplicand to the partial product in AC.
c. When the two bits are equal, the partial product does not change. An overflow cannot occur because the addition and subtraction of the multiplicand follow each other. As a consequence, the 2 numbers that are added always have an opposite signs, a condition that excludes an overflow.

The next step is to shift right the partial product and the multiplier (including Qn+1). This is an arithmetic shift right (ashr) operation which AC and QR to the right and leaves the sign bit in AC unchanged.

The sequence counter is decremented and the computational loop is repeated n times.

Product of negative numbers is important, while multiplying negative numbers we need to find 2’s complement of the number to change its sign, because it’s easier to add instead of performing binary subtraction. Product of two negative numbers is demonstrated below along with 2’s complement.

Example – A numerical example of booth’s algorithm is shown below for n = 4. It shows the step by step multiplication of -5 and -7.

BR = -5 = 1011,

BR' = 0100, <-- 1's Complement (change the values 0 to 1 and 1 to 0)

BR'+1 = 0101 <-- 2's Complement (add 1 to the Binary value obtained after 1's complement)

QR = -7 = 1001 <-- 2's Complement of 0111 (7 = 0111 in Binary)

The explanation of first step is as follows: Qn+1

AC = 0000, QR = 1001, Qn+1 = 0, SC = 4

Qn Qn+1 = 10

So, we do AC + (BR)'+1, which gives AC = 0101

On right shifting AC and QR, we get

AC = 0010, QR = 1100 and Qn+1 = 1

OPERATION	AC	QR	Qn+1	SC
	0	1001	0	4
AC + BR’ + 1	0101	1001	0	
ASHR	0010	1100	1	3
AC + BR	1101	1100	1	
ASHR	1110	1110	0	2
ASHR	1111	0111	0	1
AC + BR’ + 1	0100	0111	0	
ASHR	0010	0011	1	0

Conclusion:

Quiz:

1) Perform multiplication of -4 with 6 using Booth's multiplication algorithm with explanation of operations.

Suggested Reference:

1. M. Morris Mano, “Computer System Architecture”, Pearson Education
2. M. Morris Mano, “Digital Logic and Computer Design”, Pearson Education

Rubric wise marks obtained:

Knowledge of subject (2)		**Programming Skill**		**Team work (2)**		**Communication Skill (2)**		**Ethics (2)**	
Good (2)	Average (1)	Good (2)	Average (1)	Good (2)	Satisfactory (1)	Good (2)	Satisfactory (1)	Good (2)	Average (1)

Rubrics	**1**	**2**	**3**	**4**	**5**	**Total**
Marks						

Experiment No: 8

Date: ___________________

Study interconnection structures of components of multiprocessor system.

Competency and Practical Skills: Not required

Relevant CO: CO3, CO4, CO5

Objectives:

a) To understand the structure and functional units of digital computer.
b) To understand the various input and output devices.

Equipment/Instruments: Not required

Explanation:

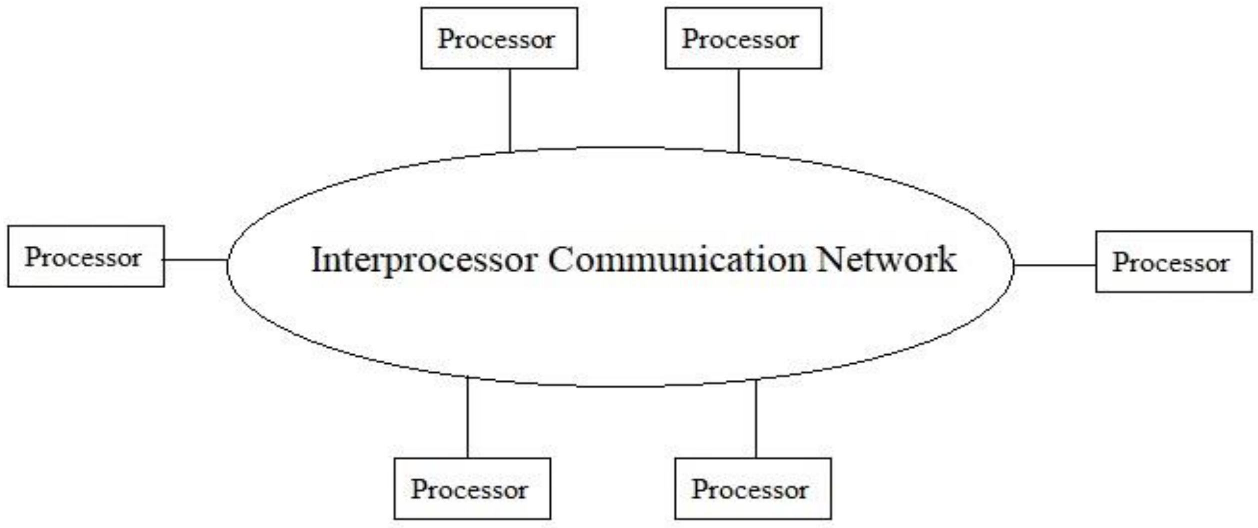


Figure: Basic Multiprocessor Architecure

The components that form a multiprocessor system are CPUs, IOPs connected to input output devices, and a memory unit.

The interconnection between the components can have different physical configurations, depending on the number of transfer paths that are available

- Between the processors and memory in a shared memory system
- Among the processing elements in a loosely coupled system

There are several physical forms available for establishing an interconnection network.

- Time-shared common bus
- Multiport memory
- Crossbar switch

- Multistage switching network
- Hypercube system

Time Shared Common Bus

A common-bus multiprocessor system consists of a number of processors connected through a common path to a memory unit. Part of the local memory may be designed as a cache memory attached to the CPU. A more economical implementation of a dual bus structure is depicted in Fig. below.

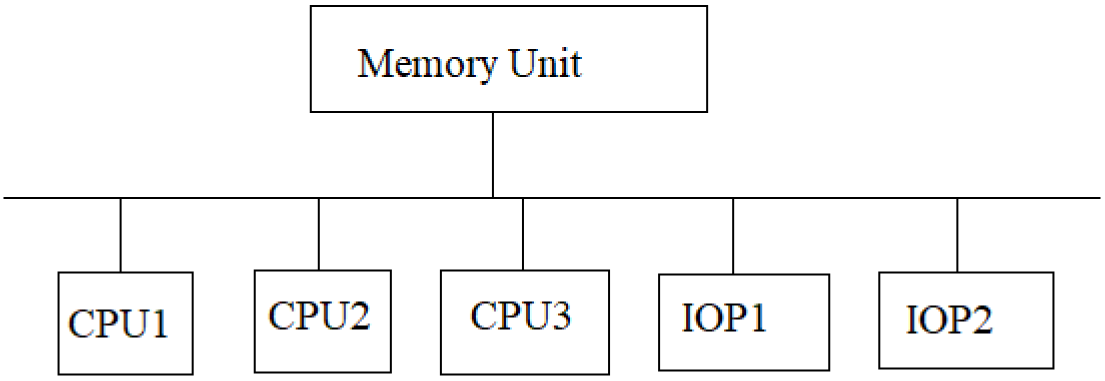


Fig. Time Shared Common Bus Organization

Disadvantage:

- Only one processor can communicate with the memory or another processor at any given time.
- As a consequence, the total overall transfer rate within the system is limited by the speed of the single path.

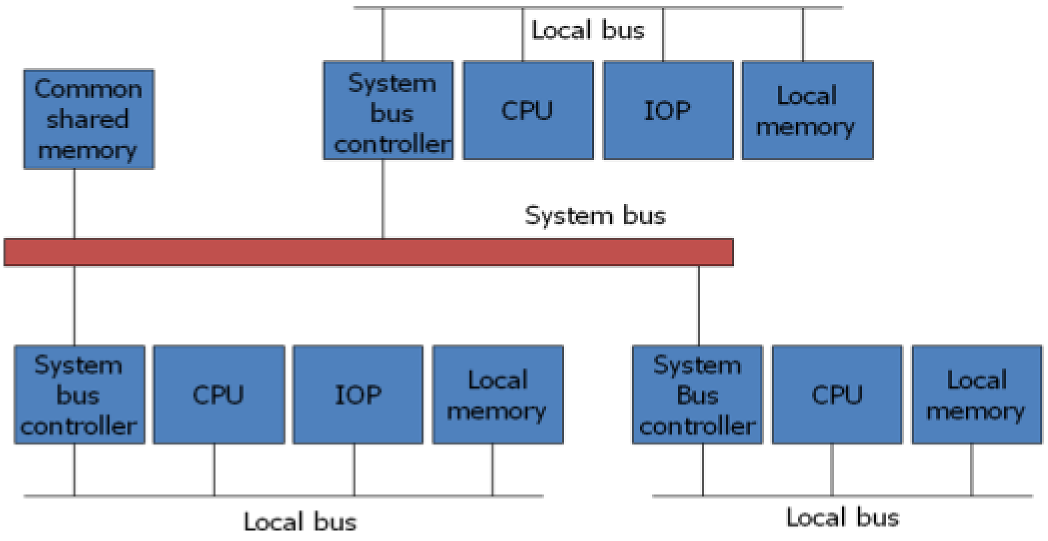


Fig: System bus structure for multiprocessors

Multiport Memory

A multiport memory system employs separate buses between each memory module and each CPU. The module must have internal control logic to determine which port will have access to memory at any given time. Memory access conflicts are resolved by assigning fixed priorities to each memory port.

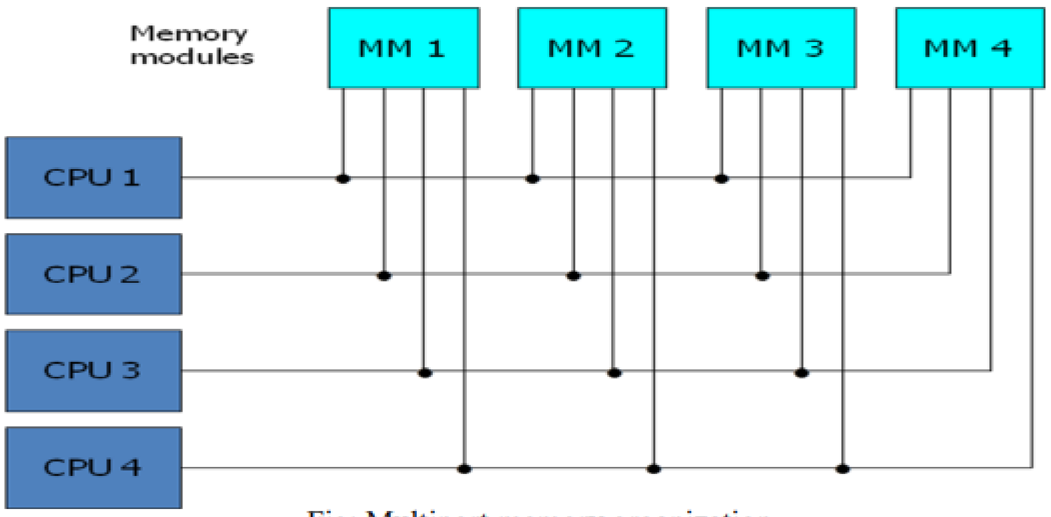


Fig: Multiport memory organization

Advantage:

- The high transfer rate can be achieved because of the multiple paths.

Disadvantage:

- It requires expensive memory control logic and a large number of cables and connections.

Crossbar Switch

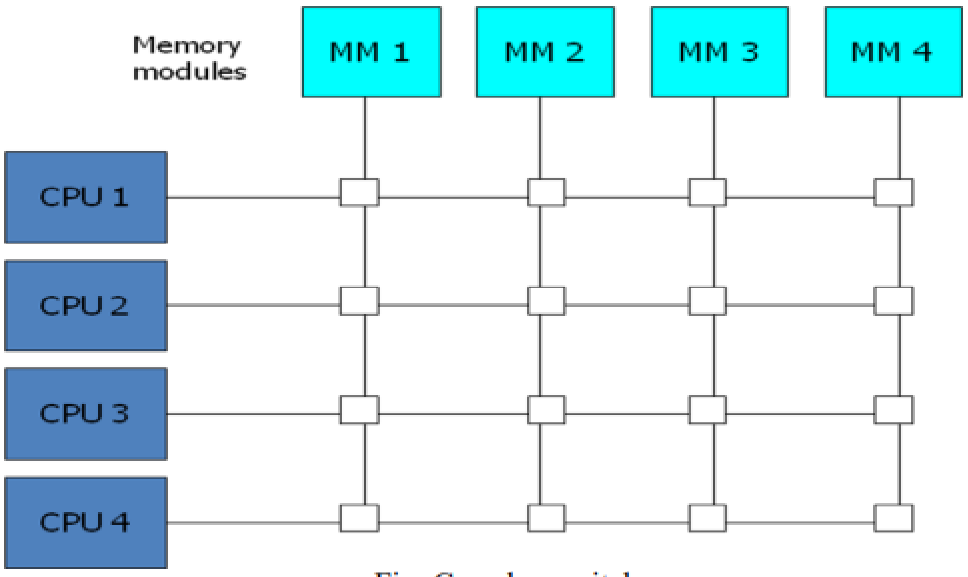


Fig: Crossbar switch

It consists of a number of cross-points that are placed at intersections between processor buses and memory module paths. The small square in each cross-point is a switch that determines the path from a processor to a memory module.

Advantage:

- Supports simultaneous transfers from all memory modules.

Disadvantage:

- The hardware required to implement the switch can become quite large and complex.

Below fig. shows the functional design of a crossbar switch connected to one memory module.

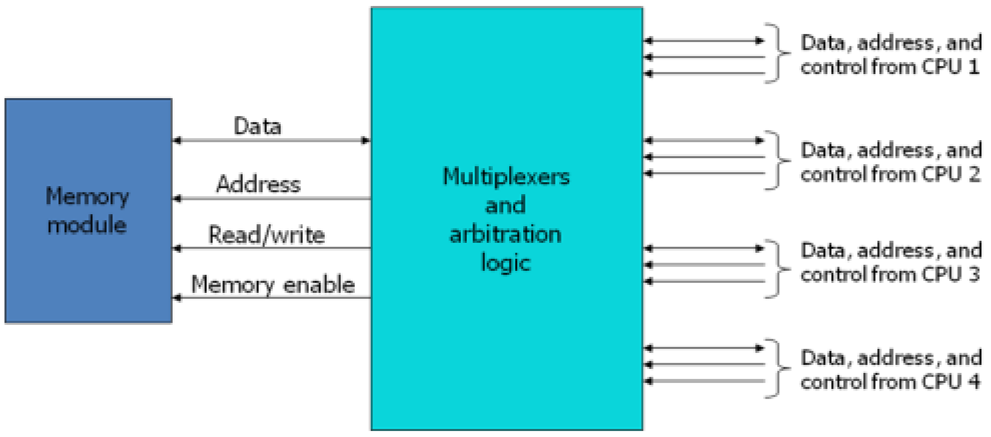
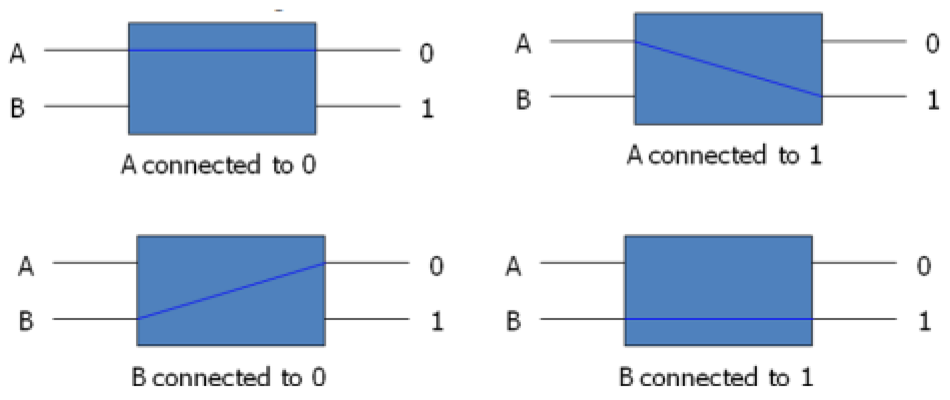

Memory module
Data
Address
Read/write
Memory enable
Multiplexers and arbitration logic
Data, address, and control from CPU 1
Data, address, and control from CPU 2
Data, address, and control from CPU 3
Data, address, and control from CPU 4

Fig: Block diagram of crossbar switch

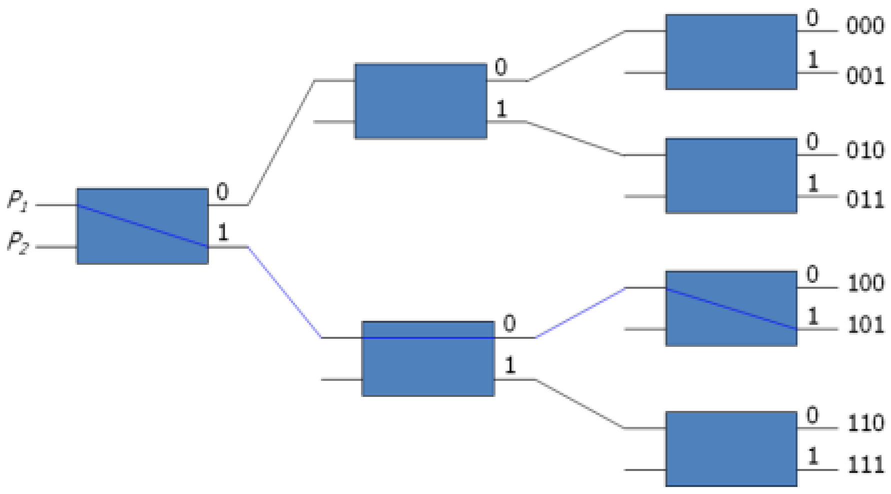
Multistage Switching Network

The basic component of a multistage network is a two-input, two-output interchange switch as shown in Fig. below.

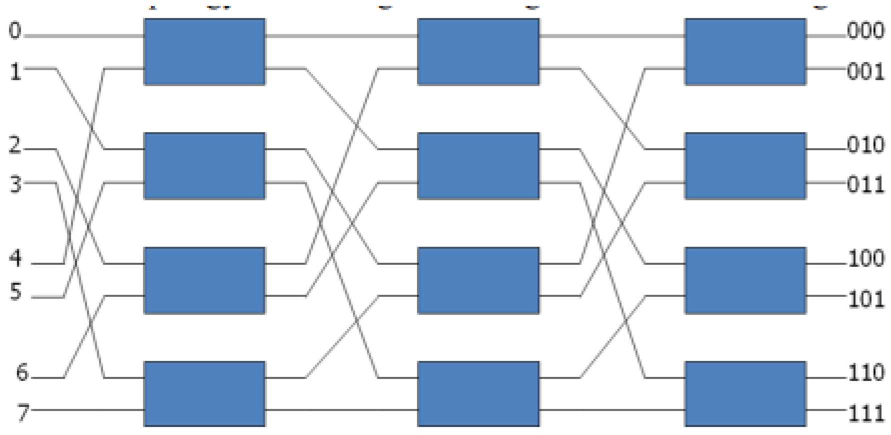


Using the 2x2 switch as a building block, it is possible to build a multistage network to control the communication between a number of sources and destinations. To see how this is done, consider the binary tree shown in Fig. below.

Certain request patterns cannot be satisfied simultaneously. i.e., if P1 → 000~011, then P2 → 100~111.



One such topology is the omega switching network shown in Fig. below



Some request patterns cannot be connected simultaneously. i.e., any two sources cannot be connected simultaneously to destination 000 and 001. In a tightly coupled multiprocessor system, the source is a processor and the destination is a memory module. Set up the path → transfer the address into memory → transfer the data. In a loosely coupled multiprocessor system, both the source and destination are processing elements.

Hypercube System

The hypercube or binary n-cube multiprocessor structure is a loosely coupled system composed of $N=2^n$ processors interconnected in an n-dimensional binary cube.

- Each processor forms a node of the cube, in effect it contains not only a CPU but also local memory and I/O interface.
- Each processor address differs from that of each of its n neighbors by exactly one bit position.

Fig. below shows the hypercube structure for n=1, 2, and 3.

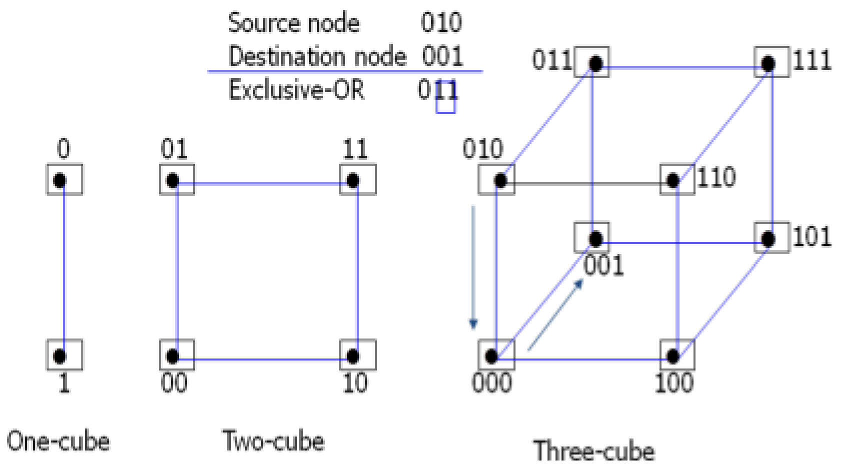


Fig: Hypercube structures for n=1,2,3

Routing messages through an n-cube structure may take from one to n links from a source node to a destination node.

- A routing procedure can be developed by computing the exclusive-OR of the source node address with the destination node address.
- The message is then sent along any one of the axes that the resulting binary value will have 1 bits corresponding to the axes on which the two nodes differ.

A representative of the hypercube architecture is the Intel iPSC computer complex.

- It consists of 128(n=7) microcomputers, each node consists of a CPU, a floating point processor, local memory, and serial communication interface units.

Conclusion:

Quiz:

1) List characteristics of Multiprocessor system.

2) What is cache coherence?

3) Differentiate tightly coupled system with loosely coupled system.

Suggested Reference:

1. M. Morris Mano, "Computer System Architecture", Pearson Education
2. R.S.Gaonkar, "Microprocessor Architecture, Programming and Applications with 8085A", Penram International

Rubric wise marks obtained:

Knowledge of subject (2)		**Programming Skill**		**Team work (2)**		**Communication Skill (2)**		**Ethics (2)**	
Good (2)	Average (1)	Good (2)	Average (1)	Good (2)	Satisfactory (1)	Good (2)	Satisfactory (1)	Good (2)	Average (1)

Rubrics	**1**	**2**	**3**	**4**	**5**	**Total**
Marks						

Experiment No: 9

Date: ___________________

Design circuits for interfacing memory and I/O with processor.

Competency and Practical Skills: Understanding and analyzing

Relevant CO: CO4

Objectives:

a) To understand concepts of memory and I/O device Interface.
b) To design and analyze interface of memory and I/O device with basic computer unit/ 8085 microporcessor.

Equipment/Instruments: Simulator tool

Theory:

// Write theory about interface of memory and I/O device with basic computer unit here…

- **Memory Interfacing:**
  The interfacing process includes matching the memory requirements with the microprocessor signals. Therefore, the interfacing circuit should be designed in such a way that it matches the memory signal requirements with the microprocessor's signals.

- **I/O Interfacing:**
  Keyboard and displays are used as communication channel with outside world. Therefore, it is necessary that we interface keyboard and displays with the microprocessor. This is called I/O interfacing.

Example of interfacing (circuit diagram):

// Draw interfacing diagram here

Observations:

// Write your observation here

Conclusion:

// Write conclusion here

Quiz:

1) Why interfacing is required?

2) Which are the techniques for interfacing?

3) What is synchronization in interfacing?

Suggested Reference:

1. M. Morris Mano, “Computer System Architecture”, Pearson Education

Rubric wise marks obtained:

Knowledge of subject (2)		**Programming Skill**		**Team work (2)**		**Communication Skill (2)**		**Ethics (2)**	
Good (2)	Average (1)	Good (2)	Average (1)	Good (2)	Satisfactory (1)	Good (2)	Satisfactory (1)	Good (2)	Average (1)

Rubrics	**1**	**2**	**3**	**4**	**5**	**Total**
Marks						

Experiment No: 10

Date: ___________________

Study about different types of computer architectures and comprehend the features and performance parameters.

Competency and Practical Skills: Understanding and analyzing

Relevant CO: CO5

Objectives:

a) To understand different types of computer architecture.
b) To analyze architecture based on features and performance parameters.

Equipment/Instruments: Not required

Explanation:

// Write theory about different types of computer architecture here

- RISC (REDUCED INSTRUCTION SET COMPUTER)
- CISC (COMPLEX INSTRUCTION SET COMPUTER)

Observations:

// Write your observation here

Conclusion:

// Write conclusion here

Quiz:

1) What is difference between RISC and CISC architecture?

2) What are the advantages of CISC architecture?

3) Example of RISC based processor?

Suggested Reference:

2. M. Morris Mano, "Computer System Architecture", Pearson Education

Rubric wise marks obtained:

Knowledge of subject (2)		**Programming Skill**		**Team work (2)**		**Communication Skill (2)**		**Ethics (2)**	
Good (2)	Average (1)	Good (2)	Average (1)	Good (2)	Satisfactory (1)	Good (2)	Satisfactory (1)	Good (2)	Average (1)

Rubrics	**1**	**2**	**3**	**4**	**5**	**Total**
Marks						