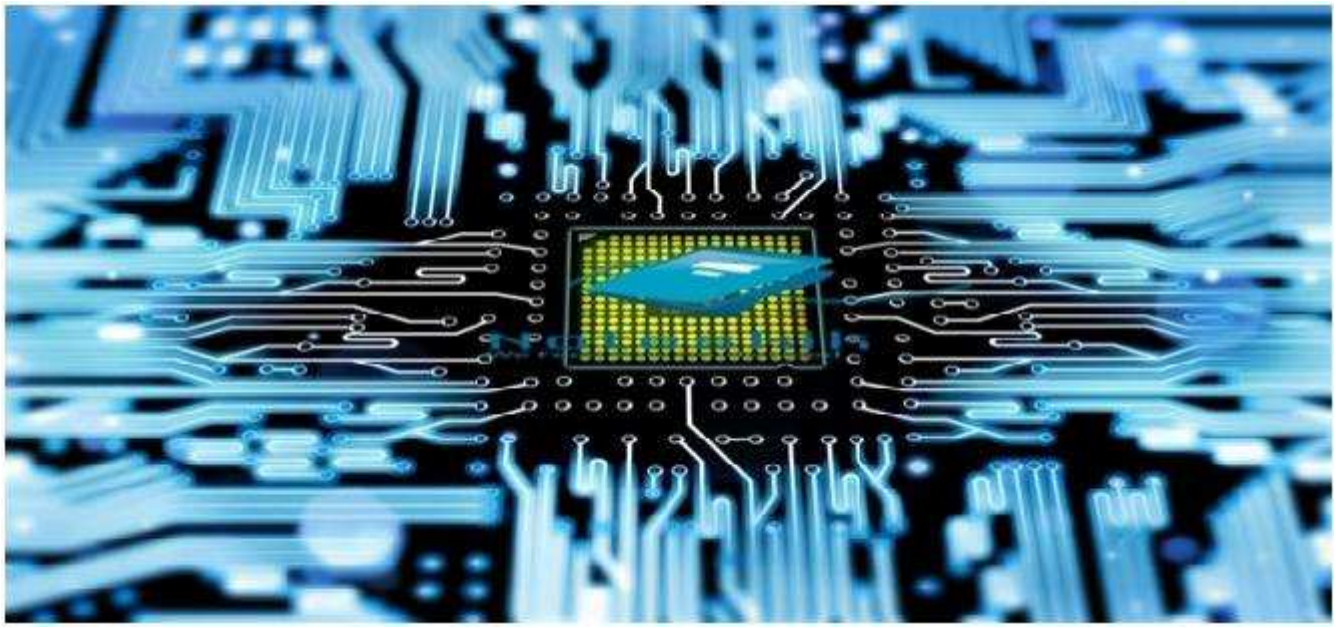
MUFFAKHAM JAH COLLEGE OF ENGINEERING AND TECHNOLOGY

Banjara Hills, Hyderabad, Telangana

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Micro Processor Laboratory Manual



Academic Year 2016-2017

Table of Contents

I Contents

II Programs

Part I

Contents

1. Vision of the Institution

To be part of universal human quest for development and progress by contributing high calibre, ethical and socially responsible engineers who meet the global challenge of building modern society in harmony with nature.

2. Mission of the Institution

- To attain excellence in imparting technical education from undergraduate through doctorate levels by adopting coherent and judiciously coordinated curricular and co-curricular programs.
- To foster partnership with industry and government agencies through collaborative research and consultancy.
- To nurture and strengthen auxiliary soft skills for overall development and improved employability in a multi-cultural work space.
- To develop scientific temper and spirit of enquiry in order to harness the latent innovative talents.
- To develop constructive attitude in students towards the task of nation building and empower them to become future leaders
- To nourish the entrepreneurial instincts of the students and hone their business acumen.
- To involve the students and the faculty in solving local community problems through economical and sustainable solutions.

3. Department Vision

To contribute competent computer science professionals to the global talent pool to meet the constantly evolving societal needs.

4. Department Mission

Mentoring students towards a successful professional career in a global environment through quality education and soft skills in order to meet the evolving societal needs.

5. Programme Education Objectives

1. Graduates will demonstrate technical skills and leadership in their chosen fields of employment by solving real time problems using current techniques and tools.

2. Graduates will communicate effectively as individuals or team members and be successful in the local and global cross cultural working environment.

3. Graduates will demonstrate lifelong learning through continuing education and professional development.

4. Graduates will be successful in providing viable and sustainable solutions within societal, professional, environmental and ethical contexts

6. Programme Outcomes

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. **Lifelong learning:** Recognize the need for, and have the preparation and ability to engage in independent and lifelong learning in the broadest context of technological change.

7. Programme Specific Outcomes

The graduates will be able to:

PSO1: Demonstrate understanding of the principles and working of the hardware and software aspects of computer systems.

PSO2: Use professional engineering practices, strategies and tactics for the development, operation and maintenance of software

PSO3: Provide effective and efficient real time solutions using acquired knowledge in various domains.

8. Introduction To Microprocessor And Interfacing Laboratory

The Intel 8085 ("eighty-eighty-five") is an 8-bit microprocessor introduced by Intel in 1977. The 8085 is a conventional von Neumann design based on the Intel 8080.

It is designed by using nmos technology. The "5" in the model number came from the fact that the 8085 requires only a +5-Volt (V) power supply rather than requiring the +5 V, 5 V and +12 V supplies the 8080 needed.

It has 8 bit data bus and 16 bit address bus. It can work up to 5 MHz frequency. It has 40 pins in its chip. Lower order address bus is multiplexed with data bus to minimize the chip size.

8051 is an 8-bit family of microcontroller developed by Intel in the year 1981. This is one of the most popular family of microcontroller being used all across the world.

This microcontroller was also referred as system on a chip because it has 128 bytes of RAM, 4Kbytes of ROM, 2 Timers, 1 Serial port, and four ports on a single chip.

The CPU can work for only 8bits of data at a time because 8051 is an 8bit processor. In case the data is larger than 8 bits then it has to be broken into parts so that the CPU can process conveniently.

Most manufacturers have put 4Kbytes of ROM even though the quantity of ROM can be exceeded up to 64 K bytes.

Laboratory Objective

Upon successful completion of this Lab the student will be able to:

- Apply the programming techniques in developing the assembly language programs.
- Familiarize the architecture of 8086 processor, assembling language programming and interfacing with various modules.
- The student can also understand of 8051 Microcontroller concepts, architecture, programming and application of Microcontrollers.
- These course can be prerequisite for the future course Embedded Systems.
- Microprocessors are designed for Personal Computers and General purpose Computers. Microcontrollers are designed for Embedded Systems.

Overview Of 8085 Microprocessor

The architectural overview of 8085 is depicted in diagram 1

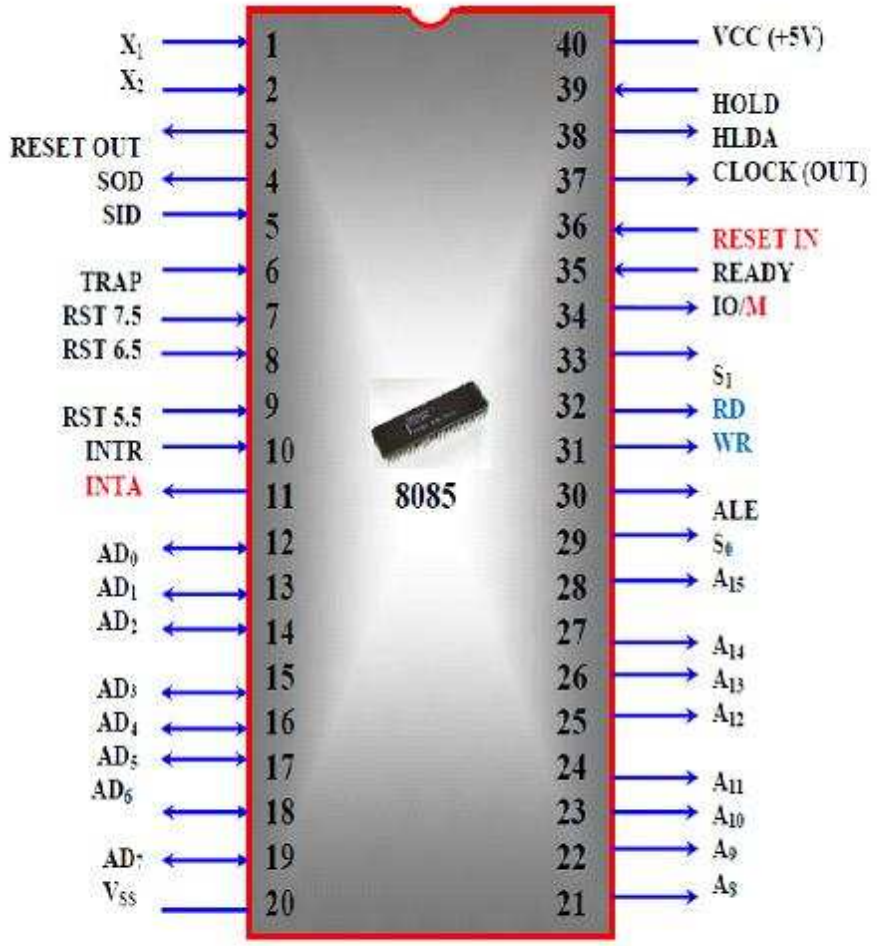


Diagram 1: Architecture of 8085

PIN DESCRIPTION

- **Address Bus**
  1. The pins Ao A15 denote the address bus.
  2. They are used for most significant bit
- **Address / Data Bus**
  1. AD0 AD7 constitutes the address / Data bus
  2. These pins are used for least significant bit

- **ALE : (Address Latch Enable)**
  1. The signal goes high during the first clock cycle and enables the lower order address bits.
- **IO / M**
  1. This distinguishes whether the address is for memory or input.
  2. When this pins go high, the address is for an I/O device.
- **S0 S1**
  1. S0 and S1 are status signal which provides different status and functions.
- **RD**
  1. This is an active low signal
  2. This signal is used to control READ operation of the microprocessor.
- **WR**
  1. WR is also an active low signal
  2. Controls the write operation of the microprocessor.
- **HOLD**
  1. This indicates if any other device is requesting the use of address and data bus.
- **HLDA**
  1. HLDA is the acknowledgement signal for HOLD
  2. It indicates whether the hold signal is received or not.
- **INTR**
  1. INTE is an interrupt request signal
  2. IT can be enabled or disabled by using software
- **INTA**
  1. Whenever the microprocessor receives interrupt signal
  2. It has to be acknowledged.

- **RST 5.5, 6.5, 7.5**
  1. These are nothing but the restart interrupts.
  2. They insert an internal restart junction automatically.
- **TRAP**
  1. Trap is the only non-maskable interrupt
  2. It cannot be enabled (or) disabled using program.
- **RESET IN**
  1. This pin resets the program counter to 0 to 1 and results interrupt enable and HLDA flip flops.
- **X1, X2**
  1. These are the terminals which are connected to external oscillator to produce the necessary and suitable clock operation.
- **SID**
  1. This pin provides serial input data
- **SOD**
  1. This pin provides serial output data.
- **VCC and VSS**
  1. VCC is +5V supply pin
  2. VSS is ground pin

Features Of 8085 Microprocessor

1. **General purpose register**

   It is an 8 bit register i.e.**B, C, D, E, H, and L**. The combination of **8** bit register is known as register pair, which can hold **16** bit data. The HL pair is used to act as memory pointer is accessible to program.

2. **Accumulator**

   It is an **8** bit register which hold one of the data to be processed by **ALU** and stored the result of the operation.

3. **Program counter (PC)**

   It is a **16-bit** pointer which maintains the address of a byte entered to line stack.

4. **Stack pointer (Sp)**

   It is a **16** bit special purpose register which is used to hold line memory address for line next instruction to be executed.

5. **Arithmetic and logical unit**

   It carries out arithmetic and logical operation by 8 bit address it uses the accumulator content as input the ALU result is stored back into accumulator.

6. **Temporary register**

   It is an 8 bit register associated with ALU hold data, entering an operation,used by the microprocessor and not accessible to programs.

7. **Flags**

   Flag register is a group of fire, individual flip flops line content of line flag register will change after execution of arithmetic and logic operation. The line states flags are

   - Carry flag (C)
   - Parity flag (P)
   - Zero flag (Z)
   - Auxiliary carry flag (AC)
   - Sign flag (S)

8. **Timing and control unit**

   Synchronous all microprocessor, operation with the clock and generator and control signal from it necessary to communicate between controller and peripherals.

9. **Instruction register and decoder**

   Instruction is fetched from line memory and stored in line instruction register decoder the stored information.

10. **Register Array**

    These are used to store 8 bit data during execution of some instruction.

Overview Of 8051 Microcontroller

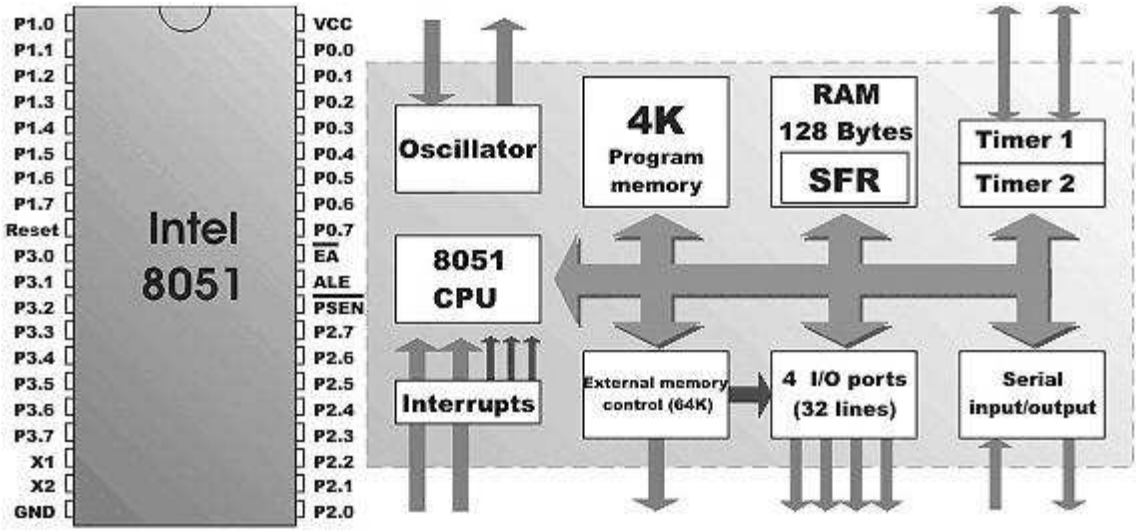
The architectural overview of 8051 is depicted in diagram 2.

8051 Microcontroller us a 40 PIN Integrated Circuit. Above is the figure of Pins of IC of 8051 micro-controller.

Explanation for each PIN is given below:

Pins 1 to 8(Port 1):

The Pins 1.0 to 1.7 are 8 Pins of port 1. Each of them can be configured as input or output pin.



Pin 9:

It is used to Reset Microcontroller 8051. A positive pulse is given on this Pin to reset Microcontroller.

Pin 10 to 17(Port 3):

These Pins are similar to Pins of Port 1. These Pins can be used as universal Input or output. These are dual function Pins. Function of each Pin is given as:

Pin 10:

It is Serial Asynchronous Communication Input or Serial Asynchronous Communication Output.

Pin 11:

Serial Asynchronous Communication Output or Serial Synchronous Communication Output.

Pin 12:

Interrupt 0 input.

Pin 13:

Interrupt 1 input.

Pin 14:

Counter 0 clock input.

Pin 15:

Counter 1 clock input.

Pin 16:

Writing Signal for writing content on external RAM.

Pin 17:

Reading Signal to read contents of external RAM.

Pin 18 and 19:

These are input output PINS for oscillator. An internal oscillator is connected to Micro controller through these PINS.

Pin 20:

Pin 20 is grounded.

Pin 21 to 28 (Port 2):

These Pins can be configured as Input Output Pins. But this is only possible in case when we don't use any external memory. If we use external memory then these pins will work as high order address bus (A8 to A15).

Pin 29:

If we uses an external ROM then it should has a logic 0 which indicates Micro controller to read data from memory.

Pin 30:

This Pin is used for ALE that is Address Latch Enable. If we uses multiple memory chips then this pin is used to distinguish between them. This Pin also gives program pulse input during programming of **EPROM.**

Pin 31:

If we have to use multiple memories then by applying logic 1 to this pin instructs Micro controller to read data from both memories first internal and afterwards external.

Pin 32 to 39(Port 0):

Similar to port 2 and 3, these pins can be used as input output pins when we don't use any external memory. When ALE or Pin 30 is at 1 then this port is used as data bus, when ALE pin at 0, then this port is used as lower order address bus(A0 to A7).

Features of 8051 Microcontroller

- 4 KB on chip program memory.
- 128 bytes on chip data memory(RAM)
- 32 bank reg + 16 bit addressable reg + 80 general purpose reg
- 4 reg banks.
- 128 user defined software flags.
- 8-bit data bus
- 16-bit address bus
- 16 bit timers (usually 2, but may have more, or less).
- 3 internal and 2 external interrupts.
- Bit as well as byte addressable RAM area of 16 bytes.
- Four 8-bit ports, (short models have two 8-bit ports).
- 16-bit program counter and data pointer.
- 1 Microsecond instruction cycle with 12 MHz Crystal.

Specifications

- CPU : 8085 Operated At 3.072 Mhz.
- Memory : Three 28-Pin JEDEC Sockets Offer 64K Bytes Of Memory As Follows:
- 16K Bytes Of Firmware In One 27128
- 4K/8K/16K Expansion Through 2732/2764/6264/27128
- 32KB Of Static RAM Using One 62256 With Battery Backup.
- Firmware: Serial And Keyboard Monitors.
- Centronics Printer Interface Driver Software.
- EPROM Programming Software.
- Audio Tape Interface Driver Software

ICs Used :

- 8085 - 8 bit MP
- 8253 - Programmable Internal Timer

- 8255 - Programmable Peripheral Interface
- 8279 - Programmable Key Boards / Display Interface
- 8251 - Programmable Communication Interface
- 2764 - 8 Kv Vv Eprom
- 6264 - 8k Static Prom
- 7414 - Hex Inverter
- 7432 - Quad 21/P Or Gate
- 7409 - Quad 21/P And Gate
- 7400 - Nand Gate
- 7404 - Dual D-Ff
- 74373 - Octal D Latch
- 74139 - Dual 2 To 4 Line Decoder
- 74138 - 3 To 8 Line Decoder

Power Supply :

- +5V,(+,- 0.1V), 3A
- +12V,(+,-1.0V),250mA
- -12V,(+,-1.0V), 100mA
- 30V,(2.0V), 100mA

To Enter Program into Trainer Kit

```
 Press RESET key
 Press Exam Mem Key
 Enter the address (16 bit) and digit in hex
 Press NEXT key
 Enter the data
 Again press NEXT
After  HLT instruction its Hex code
 Press NEXT
```

How To Executive Program

```
  Press RESET
  Press GO
  Enter the address location in which line program was executed
  Press Execute key
```

Part II

Programs

Program 1

Understanding Networking Commands

Problem Definition

To write an assembly language for adding and subtracting two 8 bit numbers by using micro processor kit.

Problem Description

Adding two numbers (Hex) which are stored in two registers. The result which is stored in Accumulator after Addition is stored in desired Memory Location.

Subtracting two numbers (Hex) which are stored in two registers. The result which is stored in Accumulator after Subraction is stored in desired Memory Location.

Algorithm:

Addition

```
Step 1 : Start the microprocessor
Step 2 : Load the first 8 bit data into the accumulator
Step 3 : Copy the contents of accumulator into the register B
Step 4 : Load the second 8 bit data into the accumulator.
Step 5 : Add the 2 - 8 bit data .
Step 6 : Store the added result in accumulator
Step 7 : Store the result in desired memory location.
Step 8: Stop the program execution.
```

Subtraction

```
Step 1 : Start the microprocessor
Step 2 : Load the first 8 bit data into the accumulator
Step 3 : Copy the contents of accumulator into the register B
Step 4 : Load the second 8 bit data into the accumulator.
Step 5 : Add the 2 - 8 bit data .
Step 6 : Store the Subtraction result in accumulator
Step 7 : Store the result in desired memory location.
Step 8: Stop the program execution.
```

Program Validation

Addition

Input

```
REGA 10H
REG-B 20H
```

Output

```
REGA 30H
Memory 30H
```

Subtraction

Input

```
REG A 20H
REG-B 10H
```

Output

```
REG A 10H
Memory 10H
```

Conclusion

The assembly language program addition of two 8 bit numbers was executed successfully by using 8085 micro processing kit.

The assembly language program subtraction of two 8 bit numbers was executed successfully by using 8085 micro processing kit.

Program 2

MULTIPLICATION AND DIVISION OF TWO 8-BIT NUMBERS

Problem Definition

To write an assembly language program for Multiplication and Division of two 8 bit numbers.

Problem Description

Multiplication of two 8-bit numbers using repeated addition method.

Division of two 8-bit numbers using repeated Subtraction method.

Multiplication

Algorithm

```
Step 1: Start the microprocessor.
Step 2: Get the 1st 8 bit numbers
Step 3: Move the 1st 8it number to register B
Step 4: Get the 2nd 8 bit number
Step 5: Move the 2nd 8 bit number to register C
Step 6: Initialize the accumulator as zero
Step 7: Initialize the carry as zero
Step 8:Add both register B value as accumulator
Step 9: Jump on if no carry
Step 10: Increment carry by 1 if there is
Step 11: Decrement the 2nd value and repeat from step 8,
         till the 2ndvalue becomes zero.
Step 12: Store the multiplied value in accumulator
Step 13: Move the carry value to accumulator
Step 14: Store the carry value in accumulator
```

Division

Algorithm

```
Step 1:  Start the microprocessor
Step 2:  Initialise  the Quotient as zero
Step 3:  Load the 1st 8-bit data
Step 4:  Copy the contents of accumulator into register B
Step 5:  Load the 2nd 8 bit data
Step 6:  Compare both the values
Step 7:  Jump if divisor is greater than dividend
Step 8:  Subtract the dividend value by divisor value
```

```
Step 9:  Increment Quotient
Step 10:  Jump to step 7, till the dividend becomes zero
Step 11:  Store the result (Quotient) value in accumulator
Step 12:  Move the remainder value to accumulator
Step 13:  Store the result in accumulator
Step 14:  Stop the program execution
```

Program Validation

Multiplication

Input

```
Input  Address Value
8080 04H
8081 02H
```

Output

```
Output Address Value
8500 08H
8501 00H
```

Program Validation

Division

Input

```
Input  Address Value
8080 09H
8081 02H
```

Output

```
Output Address Value
8500 04H
8501 01H
```

Conclusion

The assembly language program for multiplication of two 8 bit numbers was executed using 8085 Micro processing kit.

The assembly language program for division of two 8 bit numbers was executed using 8085 Micro processing kit.

Program 3

BLOCK TRANSFER

Problem Definition

To write an assembly language program for Block data transfer between memory locations using 8085 Microprocessor kit.

Program Description

Storing finite numbers(bytes) at consecutive memory locations. Transfer these numbers from source to destination memory locations.

Algorithm

```
Step 1 :  Start the microprocessor
Step 2:  Load the 16-bit address in HL  register pair
Step 3:  Load the 16-bit address in DE  register pair
Step 4:  Initialize a counter with number of bytes to transfer
Step-5:  Transfer the byte stored in source memory location
         to Accumulator
Step-6:  Transfer the byte from Accumulator to Destination
         memory address.
Step-7 : Increment the address of both memory locations
Step-8:  Decrement the counter and check for zero
Step-9:  Repeat from step-5 if not zero
Step-10 :Stop the program execution
```

Program Validation

Input

```
Input  Address Value
8080 09H
8081 02H
8082 03H
8083 05H
8084 06H
```

Output

```
Output Address Value
8500 09H
8501 02H
8502 03H
8503 05H
8504 06H
```

Conclusion

The assembly language program for Block Transfer was executed using 8085 Microprocessor kit.

Program 4

SUM OF FINITE NUMBERS

Problem Definition

To write an Assembly language to find sum of finite set of numbers.

Program Description

A set of numbers(bytes) are stored in consecutive memory locations. the numbers are added to each other and result with carry are stored and displayed.

Algorithm

```
Step 1:  Start the microprocessor
Step 2:  Load the number of values in series in accumulator and move it
             to register C and load the starting address of array
Step 3 :  Initialize the value of A as 00
Step 4:  Move the value of A to B register
Step 5:  Add the content of accumulator with the data pointed by HL pair
Step 6:  If there exists a carry, increment B by 1, if not continue
Step 7:  Increment the pointer to next data
Step 8:  Decrement the value of C by 1, which is used as counter
Step 9:  If C is equal to zero, go to step 10 if not go to step 5.
Step 10 :  Store the value of A to memory, it shows the result
Step 11:  Move the content of  B to A
Step 12: Store the value of A to memory
Step 13:  Stop the program
```

Program Validation

Input

InputAddress	Value
8080	01H
8081	02H
8082	03H
8083	04H
8084	05H

Output

OutputAddress	Value
8500	0FH
8501	00H

Conclusion:

The assembly language program for Sum of finite numbers was executed using 8085 Microprocessor kit.

Program 5

ADDING POSITIVE NUMBERS BY REJECTING NEGATIVE NUMBERS

Problem Definition

To write an assembly language program for adding positive numbers by Rejecting negative numbers in a given set of numbers and display the sum of positive numbers and if there is carry display FFH.

Program Description

A set of numbers are stored in consecutive storage locations. Each number is retrieved from Memory one by one and checked for negative (D7=1) number. If the number is positive(D=0) then it is added to its previous number, and if the number is negative the number is rejected for addition. The next number is retrieved and checked again and the process continues till the last number .

Algorithm

```
Step 1 : Start the Microprocessor
Step 2 : Set the counter, clear sum, set up memory pointer
Step 3 : Get the data byte
Step 4 : Check for  sign
Step 5: If sign is negative goto Step 10
Step 6: Restore byte
Step 7:  Calculate Sum=Old sum + data byte
Step 8 : Check if Sum>FFH  Goto  Step 14
Step 9 : Save sum
Step 10 : Calculate Pointer=pointer+1,  Counter=Counter+1
Step 11: If  counter is  Not Zero goto Step 3
Step 12 : Output Sum
Step 13 : Stop the Program
Step 14: Output FFH
Step 15: Stop the Program
```

Program Validation

Input

Input Address	Value
8080	28H
8081	D8H
8082	C2H
8083	21H
8084	24H
8085	30H
8086	2FH
8087	19H
8088	F2H
8089	9FH

Output

OutputAddress	Value
8500	E5H

Conclusion

The assembly language program for Sum of Positive numbers rejecting negative numbers was Executed using 8085 Microprocessor kit.

Program 6

LARGEST AND SMALLEST OF GIVEN N NUMBERS

Problem Definition

To write an Assembly Language Program to find Largest and Smallest of given N numbers.

Program Description

A set of numbers(bytes) are stored in consecutive memory locations. Each number is retrieved and compared with the first number if it is greater than the retrieved number if yes, the that number is replaced with first position and process is repeated till the counter becomes zero.

For finding the smallest among given set of numbers, the smallest number is placed at first position and compared with all other numbers.

Algorithm

Largest

```
Step 1: Start the Microprocessor
Step 2:  Initialize the Memory Pointer
Step 3: Set the counter
Step 4: Transfer the value into Accumulator
Step 5: Increment the pointer by one
Step 6: Compare the values.
Step 7: Check for  No carry if  true  Goto Step 5
Step 8:  Save the Largest value
Step 9:  Display the largest number
Step 10: Stop the program
```

Smallest

```
Step 1: Start the Microprocessor
Step 2:  Initialize the Memory Pointer
Step 3: Set the counter
Step 4: Transfer the value into Accumulator
Step 5: Increment the pointer by one
Step 6: Compare the values.
Step 7: Check for  carry if  true  Goto Step 5
Step 8:  Save the Smallest value
Step 9:  Display the Smallest number
Step 10: Stop the program
```

Program Validation

Input

```
InputAddress Value
8080         10H
8081         80H
8082         44H
8083         21H
8084         24H
```

Output

```
OutputAddress Value
8500           80H
```

Smallest

Input

```
InputAddress Value
8080           30H
8081           80H
8082           44H
8083           21H
8084           24H
```

Output

```
OutputAddress Value
8500          21H
```

Conclusion

The assembly language program to find Largest and Smallest numbers in a given set of numbers was Executed using 8085 Microprocessor kit.

Program 7

FACTORIAL OF GIVEN NUMBER

Problem Definition

To write an Assembly Language Program to find Factorial of given number.

Program Description

The Factorial of a given number is calculated using Factorial subroutine and for the values between 0 to 8.

Algorithm

```
Step 1:  Initialize the stack pointer
Step 2 :  Get the number in accumulator
Step 3 :  Check for if the number is greater than 1. If no
          store the result otherwise go to next step.
Step  4:  Load the counter and initialize result
Step  5:  Now factorial program in sub-routine is called.
Step  6:  In factorial, initialize HL RP with 0.
              Move the count value to B Add HL content  with  Rp.
              Decrement count (for multiplication)
Step  7:  Exchange content of  Rp (DE) with HL.
Step  8:  Decrement counter (for factorial) till zero flag is set.
Step  9:  Store the result.
Step 10: Stop the program Executed.
```

Program Validation

Input

InputAddress	Value
8500	04H

Output

OutputAddress	Value
8550	18H

Conclusion

The assembly language program to find Factorial of a given number was Executed using 8085 Microprocessor kit.

Program 8

FIBBONACCI SERIES

Problem Definition

Write an Assembly Language Program to generate Fibbonacci series.

Program Description

The program calculates the Fibbonacci series of given number. The initial value of the number is loaded in a register and the number is added to its previous number to generate fibbonacci number.

Algorithm

```
Step 1 : Start the microprocessor
Step 2 : Load the length of series in the accumulator and
         decrement it by 2
Step 3 : Move the value to register D
Step 4 : Load the starting value of data value address
Step 5 : Intialise the 1stnumber as 00
Step 6 : Move the pointer to 2Nd  data and intialise them as 01
Step 7 : Move the pointer to next position for next data
Step 8 : Initialise B as 00 and C as 01 for calculations
Step 9 : Copy the contents of B to accumulator
Step 10 : Add the content of C register to accumulator
Step 11 : Move the content C to B and A to C
Step 12 : Now store the result to memory pointed by HL pair
Step 13 : Move the pointer to next pointer
Step 14 : Decrement 0 by 1 for counter
Step 15 : If D is not zero, go to step 9
Step 16 : if D is zero, end the program
```

Program validation

Input

InputAddress	Value
8080	05H

Output

Output Address	Value
8500	00H
8501	01H
8502	01H
8503	02H
8504	05H

Conclusion

The assembly language program to generate Fibbonacci numbers was executed using 8085 Micro processor kit.

Program 9

16-BIT ADDITION AND SUBRACTION

Problem Definition

To Write an Assembly Language Program for 16-bit Addition, 16-bit Subtraction using 8085 Microprocessor kit.

Program Description

The program adds two 16-bit numbers which are stored in two register pairs and the result is placed in destination register pair.

The program Subtracts two 16-bit numbers which are stored in two register pairs and the result is placed in destination register pair.

Addition

Algorithm

```
Step 1 :  Start the microprocessor
Step 2 :  Get the 1St 8 bit in C register (LSB) and 2Nd 8 bit in H
           register (MSB) of 16 bit number.
Step 3 :  Save the 1St 16 bit in DE register pair
Step 4 :  Similarly get the 2Nd 16 bit number and store it in
          HL register pair.
Step 5 :  Get the lower byte of 1St number into L register
Step 6 :  Add it with lower byte of 2Nd number
Step 7 :  Store the result in L register
Step 8 :  Get the higher byte of 1St number into accumulator
Step 9 :  Add it with higher byte of 2Nd number and carry of the
          lower bit addition.
Step 10 : Store the result in H register
Step 11 : Store 16 bit addition value in HL register pair
Step 12 : Stop program
```

Subtraction

Algorithm

```
Step 1 : Start the microprocessor
Step 2 : Get the 1St 8 bit in C register (LSB) and 2Nd 8 bit in H
        register (MSB) of 16 bit number.
Step 3 : Save the 1St 16 bit in DE register pair
Step 4 : Similarly get the 2Nd 16 bit number and store it
         in HL register pair.
Step 5 : Get the lower byte of 1St number into L register
Step 6 : Add it with lower byte of 2Nd number
Step 7 : Store the result in L register
Step 8 : Get the higher byte of 1St number into accumulator
Step 9 : Add it with higher byte of 2Nd number and carry of the lower
          bit addition.
Step 10 : Store the result in H register
Step 11 : Store 16 bit  value after subtraction  in HL register pair
Step 12 : Stop program
```

Program validation
Input

```
InputAddress Value
8080         07 H
8081         08H
8082         05H
8083         06H
```

Output

```
OutputAddress Value
8500          02H
8501          02H
8502          00H
```

Conclusion
The assembly language program for Addition and Subtraction of two 16-bit numbers was executed using 8085 Microprocessor kit.

Program 10

16-BIT MULTIPLICATION AND DIVISION

Problem Definition

Write an ALP for 16-bit Multiplication, Division.

Program Description

The 16-bit Multiplication is performed by repeated addition of two 16-bit numbers .The result ie product is stored in two 16-bit registers and finally the result is stored in desired memory locations.

The 16-bit Division is performed by repeated subtraction method. The result quotient and remainder are stored in two separate 16-bit registers.

Multiplication

Algorithm

```
Step 1 : Start the microprocessor
Step 2 : Load the 1st data in HL register pair
Step 3 : Move content of HL pair to stack pointer
Step 4 : Load the 2nd data in HL and move it to DE
Step 5 : Make HL pair as 00 and 00
Step 6 : Add HL pair and SP
Step 7 : Check for carry condition, if carry is present increment
          it by one
         else move to next step.

Step 8 :  Decrement DE register
Step 9 : Then move E to A and perform OR operation with a and D
Step 10 : The value of operation is zero, then store the value else go
           to step 3
Step 11 : Stop the program
```

Division

Algorithm

```
Step 1 :  Start the microprocessor
Step 2 :  Initialise BC as 0000 for Quotient
Step 3 :  Load the divisor in HL pair and save it in DE register pair
Step 4 :  Load the dividend in HL pair
Step 5 :  Move the value of a to register E
Step 6 :  Subtract the content of accumulator with E register
Step 7:  Move the content A to C & H to A
Step 8 :  Subtract with borrow, the content of A with D
Step 9 :  Move the value of a to H
Step 10 :  If cy = 1, go to step 12, otherwise next step
Step 11 :  Increment B register & jump to step 4
Step 12 :  Add both contents of DC and HL
Step 13 :  Store the remainder in memory
Step 14 :  Move the content of C to L & B to H
Step 15 :  Store the Quotient in memory
Step 16 :  Stop the program
```

Program validation

Multiplication

Input

InputAddress	Value
4200	04
4201	07
4202	02
4203	01

Output

OutputAddress	Value
4204	08
4205	12
4206	01
4207	00

Division

Program validation

Input

InputAddress	Value
4200	04
4201	00
4202	02
4203	00

Output

OutputAddress	Value
4204	02
4205	00
4206	FE
4207	FF

Conclusion

The Assembly language program for 16 bit Multiplication, 16 bit Division using 8085 Microprocessor kit was executed successfully.

Program 11

BINARY TO BCD AND BCD TO BINARY

Problem Definition

To Write an Assembly Language Program for Binary to BCD and BCD to Binary using 8085 Microprocessor kit.

Program Description

Clear register to account for hundreds and tens load the binary data in accumulator. Compare A with 64 if carry is 1. Subtract 64 from (64+1) A register. Increment E register.

St Compare the register A with 0A, if carry is true, Save the units, tens and hundreds in memory otherwise Subtract (0AH) from A register. Increment D register, Combine the units and tens to from 8 bit result.

Binary to BCD

Algorithm

```
Step 1 :  Start the microprocessor
Step 2 :  Clear D and E register to account for hundreds and tens
              load the binary data in accumulator
Step 3 :  Compare A with 64 if cy = 01, go step C otherwise next step
Step 4 :  Subtract 64 from (64+1) A register
Step 5 :  Increment E register
Step 6 :  Compare the register A with 0A, if cy=1, go to step 11,
               otherwise next step
Step 7 :  Subtract (0AH) from A register
Step 8 :  Increment D register
Step 9 :  Go to step 7
Step 10 : Combine the units and tens to from 8 bit result
Step 11 : Save the units, tens and hundreds in memory
Step 12 : Stop the program execution
```

BCD to Binary

Algorithm

```
Step 1 : Start the microprocessor
Step 2 : Get the BCD data in accumulator and save it in register E
Step 3 : Mark the lower nibble of BCD data in accumulator
Step 4 : Rotate upper nibble to lower nibble and save it in register B
Step 5 : Clear the accumulator
Step 6 : Move 0AH to C register
Step 7 : Add A and B register
Step 8 : Decrement C register. If zf = 0, go to step 7
Step 9 : Save the product in B
Step 10 : Get the BCD data in accumulator from E register and
          mark the upper nibble
Step 11 : Add the units (A-ug) to product (B-ug)
Step 12 : Store the binary value in memory
Step 13 : End the program
```

Program validation

Binary to BCD

Input

```
InputAddress Value
8400          54
```

Output

```
Output Address Value
8500           84
8501           00
```

BCD to Binary

Input

```
InputAddress Value
8400          68
```

Output

```
OutputAddress Value
8500            44
```

Conclusion

Thus the binary to BCD conversion was executed successfully using 8085 Microprocessor kit.

The BCD to binary Conversion was executed successfully using 8085 Microprocessor kit.

Program 12

ROLLING DISPLAY AND FLASHING DISPLAY

Problem Definition

To Write an Assembly Language Program for Rolling Display and Flashing Display.

Program Description

- Get the control words in accumulator and output the control words through 8 bit port address .
- Load HL register pair with memory address and transfer memory content to C register .
- Increment HL pair with one and transfer the particular bit pattern through 8 bit port address Call subroutine delay .
- Check If the count value in C is not equal to zero then Load HL pair with value.
- Load DE register pair by memory address .
- Decrement DE register pair by one
- If DE is not equal to zero, decrement DE else main program.
- Rolling Display

Algorithm

```
Step 1 : Get the control words in accumulator and output
         the control words through 8 bit port address
Step 2 : Load HL register pair with memory address and transfer memory
         content to C register
Step 3 : Increment HL pair with one and transfer the particular
          bit pattern through 8 bit port address
Step 4 : Call subroutine delay at step 6
Step 5 : If the count value in C is not equal to zero then go to step 3
         else go to step 2
Step 6 : Load DE register pair by memory address
Step 7 : Decrement DE register pair by one
Step 8 : If DE is not equal to zero, go to step 7 else main program.
```

Flashing Display

Algorithm

```
Step 1 : Get the control words in accumulator and output words
         through 8 bit address
Step 2 : Load HL register pair with memory address
Step 3 : Get the count value in C register
Step 4 : Increment the register pair by one and display the character
         and call for delay.
Step 5 : Clear the display and call delay routine to step 7
Step 6 : Go to step 7
Step 7 : Load DE register pair with memory address
Step 8 : Decrement DE pair with memory address
Step 9 : If the content is not equal to zero, go to step 8
Step 10 : Return to main program
```

Program Validation

Rolling Display

Input

InputAddress	Value
5000	06
5001	98
5002	68
5003	7A
5004	C8
5005	1A
5006	2C

Output

```
Welcome

Flashing Display
```

Input

Input Address	Value
5000	05
5001	68
5002	68
5003	68
5004	FD
5005	88

Output

```
CSE  B
```

Conclusion

Thus, an assembly language program to obtain rolling display of a particular value written using 8085 microprocessor kit is executed successfully.

Thus, an assembly language program to obtain flashing display of a particular data was written using 8085 microprocessor kit is executed successfully.

Program 13

LED INTERFACE

Problem Definition

Write an Assembly Language Program for LED Interface.

Program Description

The program scrolls any string which is given as input to the program. The input is stored in the microprocessor and the LED is interfaced with the kit. The Control word of the Led is initialized.

The string is displayed by sending the data to the port with which the LED interface is associated.

The delay subroutine is used to create delay between the display.

Algorithm

```
Step 1 : Start the Microprocessor.

Step 2 :  Initialize the  control word.

Step 3:  Load the String in desired memory location.

Step 4:  Set  the counters for  string as no of words, characters ,bits.

Step 5 : Display each character one by one at the port.

Step 6 : Call Delay subroutine.

Step 7 : Repeat Step 4  to continue the scroll.
```

Program Validation

Input

InputAddress	Value	Value	Value	Value	String
8000	FF	FF	FF	FF	Blank
8001	86	C7	86	C6	ELEC
8002	87	DE	C0	BF	TRO-
8003	92	91	92	BF	SYS-
8004	87	86	C8	92	TEMS

Output

```
ELEC TRO- SYS-TEMS
```

Conclusion

The program for interfacing the Microprocessor with LED is successfully executed using Microprocessor kit.

Program 14

STEPPER MOTOR

Problem Definition

To write an Assembly Language Program to make the Stepper Motor run in forward and reverse direction.

Program Description

The register pair HL is loaded with a value and register B is set as counter for number of rotations. The value which is transferred to Accumulator is set to control word for stepper motor rotation. Now Delay subprogram is called in between. The memory pointer is incremented. When the value of B becomes zero , the program is restarted.

Algorithm

```
Step 1 :  Load the HL pair with value from table
Step 2 :  Move it to B register for setting the counter
Step 3 :  Move the memory value to accumulator and display it by
              control word
Step 4 :  Load DE register pair with FFFF for starting delay subroutine
Step 5 :  Run the delay loop control D-register becomes zero.
Step 6 :  Increment H address for next value from table
Step 7 :  Jump on no zero
Step 8 :  When B = 0, go to start and restart the program
```

Program Validation

Input

Input Address	Value
811A	0A
811B	06
811C	05
811D	09

Output

Stepper motor rotates in Forward Direction.

Input

Output Address	Value
811A	09
811B	05
811C	06
811D	0A

Output

Stepper motor rotates in Reverse Direction

Conclusion

Thus,an assembly language program to control of stepper motor was written using 8085 microprocessor kit.

Program 15

GENERATE WAVEFORMS USING DAC

DAC can be used to generate some of the following waveforms

- Square Wave Generator
- Saw Tooth Wave Generator
- Triangular Wave

Square Wave Generator

Problem Definition

To write a program and to generate Square wave using DAC.

Program Description

The Square wave is generated by initialization of Accumulator with zero and send to control word of port . Then call delay subroutine. Now the Accumulator is loaded with FFH and the control word is initialized is with the value. Again delay is called. The process is again restarted.

Algorithm

```
Step 1 : Intialise A as 00 and take data pointer to port C
Step 2 : Call delay
Step 3 : Move FF to A and take port C
Step 4 : Call delay
Step 5 : Go to step 1
```

Delay Subtroutine

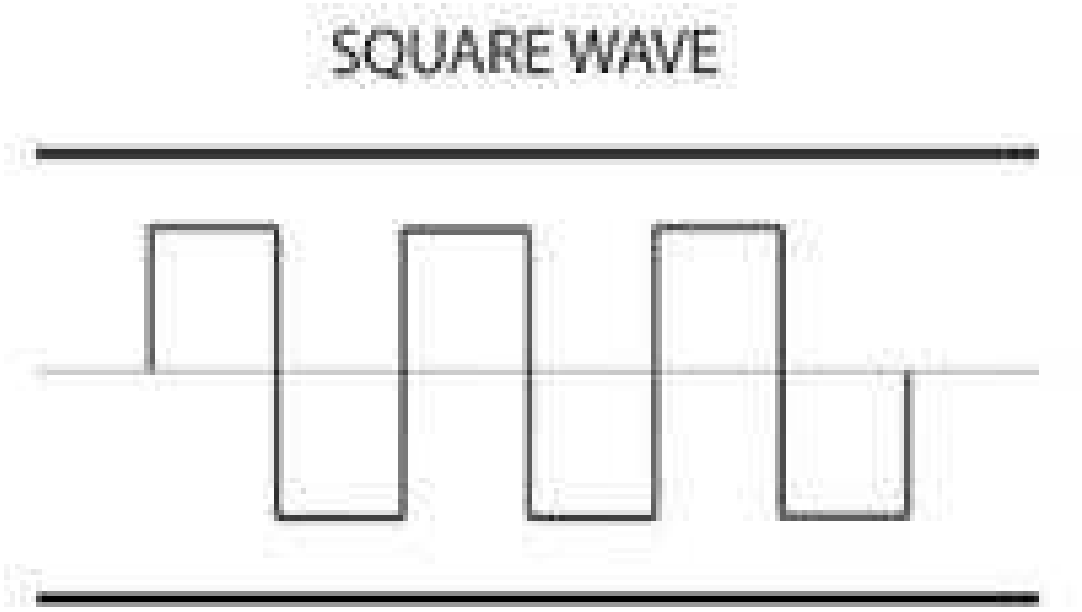
```
Step 1 : Counter 1 = 05
Step 2 : Counter 2 = FF
Step 3 : Decrement counter 2
Step 4 : Check if c= 0, if no jump to step 3
Step 5 : Decrement counter 1
Step 6 : Check if B = 0, if no jump to step 2
Step 7 : Return to main program
```

Program Validation

Input

`Use the above algorithm as input.`

Output



Conclusion

Thus square wave was generated using 8085 microprocessor kit is successfully generated.

Saw Tooth Wave Generator

Program Definition

To write a program and to generate Saw Tooth Waveform using DAC.

Program Description

The Square wave is generated by initialization of Accumulator with zero and send to control word of port . Now the Accumulator is incremented by one and the control word is initialized is with the value. The process is again restarted.

Algorithm

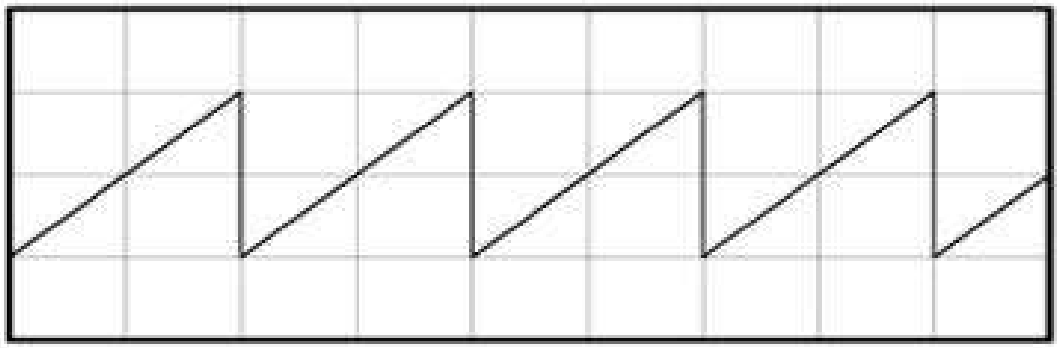
```
Step 1 : Intialise accumulator with 00
Step 2 : Output current address specified
Step 3 : Increment accumulator by one
Step 4 : Jump to step one
```

Program Validation

Input

```
Use the above algorithm as input.
```

Output



Conclusion

Thus the saw tooth wave was generated using 8085 microprocessor kit.

Triangular Wave Generator

Program Definition

To write a Assembly Language program and to generate Traingular Waveform using DAC.

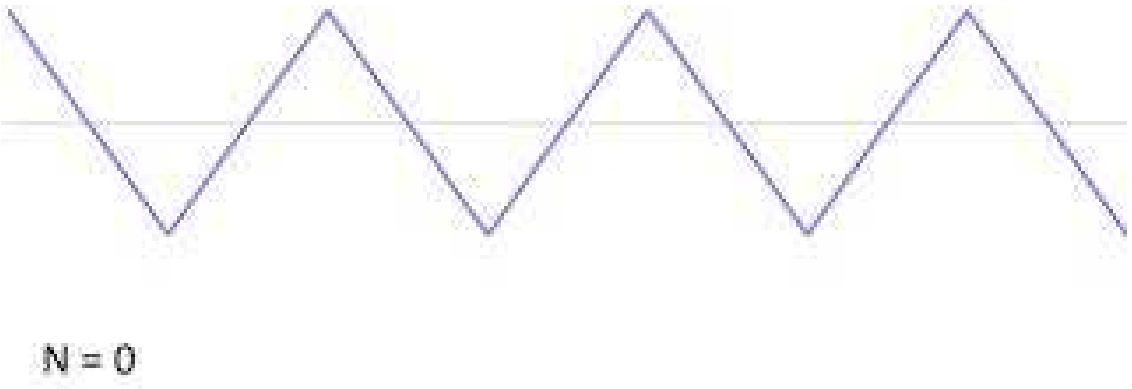
Program Description

To generate a triangular waveform the accumulator is initialized with zero then incremented by one till it reaches a max value. Now the value of accumulator is decremented till it reaches min value. T he process is repeated.

Algorithm

```
Step 1 : Move content of C to A where L is intialised to 00
Step 2 : Output content of C8
Step 3 : Increment L till zf = 0
Step 4 : Intialise L register with FF
Step 5 : Move content of L to accumulator and output to port
Step 6 : Decrement L if not equal to zero jump else go to next step
Step 7 : Jump on next step
```

Program Validation
Input
Use the above algorithm as input.
Output



Conclusion
Thus the triangular wave was generated using 8085 microprocessor kit.

Program 16

ARITHMETIC OPERATIONS USING 8051

Program Definition

To write Assembly Language Program for the Arithmetic operations using 8051 Microcontroller.

Algorithm

Addition / Subtraction

```
Step 1 : Move 1st data to memory
Step 2 : Add or subtract 1st  data with 2nd data
Step 3 : Initialize data pointer.
Step 4 : Move result to memory pointed by DPTR.
```

Program Validation

Addition

Input Address	Value
811A	02
811B	03

Output Address	Value
800A	05

Subtraction

Input Address	Value
811A	05
811B	03

Output Address	Value
800A	02

Conclusion

Thus 8-bit addition, subtraction is performed using 8051.

Algorithm

Multiplication / Division

```
Step 1 : Get 1st data and 2nd data to memory
Step 2 : Multiply or divide 1st data with 2nd data
Step 3 : Initialize data pointer.
Step 4 : Move result to memory pointed by DPTR (first port)
Step 5 : Increment DPTR
Step 6 : Move 2nd part of result to register A
Step 7 : Move result to 2nd memory location pointer by DPTR
```

Program Validation

Multiplication

```
Input Address Value
8500          04
8501          02


OutputAddress Value
8700            08
```

Division

```
InputAddress Value
8500         04
8501         02



OutputAddress Value
8700          02
```

Conclusion

Thus 8bit multiplication & division is successfully executed using 8051 Microcontroller kit.

Program 17

Block Transfer

Program Definition

To Write an Assembly Language Program for 8051 for block transfer using Microcontroller kit.

Program Description

Two memory locations are initialized which are used as source location and Destination location. The counter is set for the number of bytes to be transferred. The transfer of the data one by one is done from source to destination till all the data bytes are transferred and counter becomes zero. The output is checked at destination Memory location.

Algorithm

1. Start the Microcontroller.
2. Set the pointers for the two Memory locations.
3. Set the counter for the number of bytes to be transferred.
4. Transfer the data from source location to Accumulator.
5. Transfer the data from Accumulator to Destination location.
6. Decrement the counter by one
7. Repeat step 4 till the counter becomes zero
8. Stop the program Execution

Program Validation

Input

Input Address	Value
30	01
31	02
32	03
33	04
34	05

Output

Output Address	Value
35	01
36	02
37	03
38	04
39	05

Conclusion

The Assembly language program to perform block transfer is successfully executed using 8051 Microcontroller Kit.

Program 18

Sum of BCD Numbers

Program Definition

Write an ALP for 8051 to find sum of n BCD Numbers using 8051 Microcontroller kit.

Program Description

The number of bytes to be added are stored at desired memory location. The counter is set for n numbers. Initialize Accumulator for sum and another register for carry. Read values from the memory and transfer to accumulator after adding to previous value. Increment the carry if generated. Repeat the addition till the counter reaches zero .Convert the sum generated to decimal value. Store the final result in Accumulator.

Algorithm

```
Step 1 :  Start the Microcontroller
Step 2:  Set counter to N, Accumulator to 0, carry register to 0.
Step 3 :  Initialize the pointer to location where the values are stored.
Step 4 :  Add the values.
Step 5 :  Convert to decimal the result in accumulator.
Step 6 :  Increment the carry if carry is generated.
Step 7 :  Increment the pointer
Step 8 :  Decrement the counter and check for zero.
Step 9 :  Repeat Step 4.
Step 10:  Stop  program .
```

Program Validation
Input

```
Input Address Value
30             01
31             02
32             03
33             04
34             05
```

Output

```
Output Reg Value
   A         0F
```

Conclusion

The Assembly language program to add BCD nos is successfully executed using 8051 Microcontroller Kit.

Program 19

16-BIT ADDITION USING 8051

Program Definition

To Write an Assembly Language Program for 8051 for 16-bit Addition using Microcontoller kit.

Program Description

The two 16-bit values to be added are divided into two bytes each. The lower order byte of the addend and augend are added with each other and result which is in accumulator is transferred to a register. Now the High order byte of both the values are added with carry. Transfer the sum to register. The sum is available in two registers.

Algorithm

```
Step 1 :  Start the Microcontroller
Step 2 :  Transfer the lower byte of 1st number into accumulator.
Step 3 :  Add the lower byte of 2nd number with the accumulator.
Step 4 :  Save the result in accumulator to a register(lower byte).
Step 5 :   Transfer the higher byte of 1st number into accumulator.
Step 6 :  Add with carry the higher byte of 2nd number with the
           accumulator .
Step 7 :  Save the high byte sum in register.
Step 8 :  Stop the Execution
```

Program Validation

Input

```
Use the above steps to generate input.
```

Output

```
Output Reg Value
     R6     03
     R7    04
```

Conclusion

The Assembly language program to add two 16-bit nos is successfully executed using 8051 Microcontroller Kit.

Program 20

PACKED BCD TO ASCII CONVERSION

Program Definition

Write an Assembly Language Program for 8051 for Packed Bcd To ASCII Conversion using Microcontroller kit.

Program Description

- Transfer the number to be converted to accumulator.
- Save the value in a register1.
- And the value with 0FH and or with 30H.
- Save the result into a register2.
- Now transfer the number from register1 to accumulator.
- And the result with FOH and rotate the value in accumulator four times to right.
- Or the contents with 30H .
- Save the result in register2.

Algorithm

```
Step 1 :  Start the Microcontroller.
Step 2 :  Load the Accumulator with the value.
Step 3 :  Save the value in Register2.
Step 4 :  And with 0FH
Step 5 :  Or with 30H
Step 6 :  Save the result in  Register6.
Step 7 :  Transfer the value from register2 to accumulator.
Step 8 :  And with F0H
Step 9 :  rotate the Acc contents to right by 4 bits.
Step 10 :  Or the accumulator with 30H
Step 11:  Store the result in register2.
Step 12:  Stop the program  Execution.
```

Program Validation

Input

```
Use the above steps to generate input.
```

Output

```
Output Register Value
     R6         39
     R2         32
```

Conclusion

The Assembly language program to convert packed BCD to ASCII is successfully executed using 8051 Microcontroller Kit

OU CURRICULUM

MICROPROCESSOR LABORATORY

CS 282

Instruction	3	Periods per week
Duration of University Examination	3	Hours
University Examination	50	Marks
Sessional	25	Marks

Course objectives:

Write simple assembly language program using 8085 instruction set
Write programs to interface various peripheral devices with 8085.
Design simple applications using 8051 Micro controller.

PART A: 8085 PROGRAMMING USING MICROPROCESSOR TRAINER KIT

1. Simple programming examples using 8085 instruction set.
   To understand the use of various instructions and addressing modes.

2. Interfacing and programming of 8255. (E.g. traffic light controller).

3. Interfacing and programming of 8254.

4. Interfacing and programming of 8279.

PART B: 8051 PROGRAMMING

1. Simple Programming examples using 8051 Micro Controller.

2. A/D and D/A converter interface.

3. Stepper motor interface.

4. Display