BHARATI VIDYAPEETH DEEMED UNIVERSITY

COLLEGE OF ENGINEERING, PUNE

Lab Manual

Microprocessors and Microcontrollers

DEPARTMENT OF COMPUTER ENGINEERING

B.Tech. Computer Sem IV

VISION OF THE INSTITUTE

To be World Class Institute for Social Transformation Through Dynamic Education.

MISSION OF THE INSTITUTE

A. To provide quality technical education with advanced equipment, qualified faculty members, infrastructure to meet needs of profession and society.

B. To provide an environment conducive to innovation, creativity, research, and entrepreneurial leadership.

C. To practice and promote professional ethics, transparency and accountability for social community, economic and environmental conditions.

VISION OF THE DEPARTMENT

To pursue and excel in the Endeavour for creating globally recognized Computer Engineers through Quality education.

MISSION OF THE DEPARTMENT

- To impart engineering knowledge and skills confirming to a dynamic curriculum.
- To develop professional, entrepreneurial & research competencies encompassing continuous intellectual growth.
- To produce qualified graduates exhibiting societal and ethical responsibilities in working environment.

PROGRAM EDUCATIONAL OBJECTIVES

1. Demonstrate technical and professional competencies by applying engineering fundamentals, computing principles and technologies.

2. Learn, Practice, and grow as skilled professionals/ entrepreneur/researchers adapting to the evolving computing landscape.
3. Demonstrate professional attitude, ethics, understanding of social context and interpersonal skills leading to a successful career.

PROGRAM SPECIFIC OUTCOMES

1. To apply fundamental knowledge and technical skills towards solving Engineering problems.
2. To employ expertise and ethical practise through continuing intellectual growth and adapting to the working environment.

PROGRAM OUTCOMES

1. To apply knowledge of computing and mathematics appropriate to the domain.
2. To logically define, analyze and solve real world problems.
3. To apply design principles in developing hardware/software systems of varying complexity that meet the specified needs.
4. To interpret and analyze data for providing solutions to complex engineering problems.
5. To use and practice engineering and IT tools for professional growth.
6. To understand and respond to legal and ethical issues involving the use of technology for societal benefits.
7. To develop societal relevant projects using available resources.
8. To exhibit professional and ethical responsibilities.
9. To work effectively as an individual and a team member within the professional environment.
10. To prepare and present technical documents using effective communication skills.
11. To demonstrate effective leadership skills throughout the project management life cycle.
12. To understand the significance of lifelong learning for professional development.

GENRAL INSTUCTIONS:

- Equipment in the lab is meant for the use of students. Students need to maintain a proper decorum in the computer lab. Students must use the equipment with care.
- Students are required to carry their reference materials, files and records with completed assignment while entering the lab.
- Students are supposed to occupy the systems allotted to them and are not supposed to talk or make noise in the lab.
- All the students should perform the given assignment individually.
- Lab can be used in free time/lunch hours by the students who need to use the systems should take prior permission from the lab in-charge.
- All the Students are instructed to carry their identity cards when entering the lab.
- Lab files need to be submitted on or before date of submission.
- Students are not supposed to use pen drives, compact drives or any other storage devices in the lab.
- For Laboratory related updates and assignments students should refer to the notice board in the Lab.

COURSE NAME: ***MICROPROCESSORS AND MICROCONTROLLERS***

WEEKLY PLAN:

Week No.	Practical/Assignment Name	Problem Definition
1	**Multiplication of two 16-bit numbers**	• To multiply two 16-bit numbers
2	**Array**	• To write program to add elements of array.
3	**Negative and Positive Numbers**	• To write an assembly language programs for counting negative and positive numbers from given array.

4	**Ascending and Descending order**	• To write an assembly language program to arrange given set of numbers in ascending order and descending order.
5	**String Operations**	• To write an assembly language program for implementing following operations 1. to check whether entered string is palindrome or not 2. to compare two strings
6	**Multiplication of two 32-bit numbers**	• To multiply two 32-bit numbers using the instructions of 80386.
7	**8-bit BCD addition and subtraction**	• To write an assembly language program for 8-bit BCD addition and subtraction.
8	**System Timer**	• Implement the assembly language program to display the current time from system.
9	**Keyboard Interfacing**	• Write a assembly language program to interface 8051 microcontroller with keyboard
10	**DC motor interfacing**	• Write a assembly language program to interface 8051 microcontroller with DC motor

EXAMINATION SCHEME

Practical Exam: 25 Marks
Term Work: 25 Marks
Total: 50 Marks
Minimum Marks required: 20 Marks

PROCEDURE OF EVALUATION

Each practical/assignment shall be assessed continuously on the scale of 25 marks. The distribution of marks as follows.

Sr. No	Evaluation Criteria	Marks for each Criteria	Rubrics

1	**Timely Submission**	**07**	➢ Punctuality reflects the work ethics. Students should reflect that work ethics by completing the lab assignments and reports in a timely manner without being reminded or warned.
2	**Presentation**	**06**	➢ Students are expected to write the technical document (lab report) in their own words. The presentation of the contents in the lab report should be complete, unambiguous, clear, and understandable. The report should document approach/algorithm/design and code with proper explanation.
3	**Understanding**	**12**	➢ Correctness and Robustness of the code is expected. The Learners should have an in-depth knowledge of the practical assignment performed.The learner should be able to explain methodology used for designing and developing the program/solution. He/she should clearly understand the purpose of the assignment and its outcome.

LABORATORY USAGE

Students use computers for executing the lab experiments, document the results and to prepare the technical documents for making the lab reports.

OBJECTIVE

The objective of this lab is to give students the knowledge about microprocessor and microcontrollers and to implement various programs using assembly language.

PRACTICAL PRE-REQUISITE

- Digital logic design

SOFTWARE REQUIREMENTS

- Turbo Assembler
- EdSim51 - The 8051 Simulator

COURSE OUTCOMES

1. Describe microprocessor and micro controller architecture.
2. Understand programmer's model of 80386.
3. Understand concepts of segmentation and paging
4. Comprehend hardware and software interaction and integration.
5. Design microcontroller based systems.
6. Write assembly language program using 32 bit registers.

HOW OUTCOMES ARE ASSESSED?

Outcome	Assignment Number	Level	Proficiency evaluated by
Describe microprocessor and micro controller architecture.	**1,2,3,4,5,7,9**	**1,1,1,1,1,1,1**	Performing Practical and reporting results
Understand programmer's model of 80386.	**6**	**1**	Problem definition&Performing Practical and reporting results
Understand concepts of segmentation and paging	**6**	**2**	Performing experiments and reporting results
Comprehend hardware and software interaction and integration.	**8,9,10**	**1,1,1**	Performing experiments and reporting results
Design microcontroller based systems.	**9,10**	**1,1**	Performing experiments and reporting results
Write assembly language program using 32 bit registers.	**6**	**1**	Performing experiments and reporting results

CONTRIBUTION TO PROGRAM OUTCOMES

	Program Outcomes												PSOs	
	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PSO 1	PSO 2
Course outcomes	3	2	2	3	2			3	3	3		2	3	
	3	3	2	3	2	3		3	3	3	1	3	3	3
	3	2	2	3	2		3	3	3	3		3		
	3	3	3	3	3			3	3	3	2	3	3	3
	1		3			3		3	3	3		2		
	2	2	3	2	1		2	3	3	3	3	3	3	3

DESIGN EXPERIENCE GAINED

The students gain moderate design experience by creating assembly language programs along with interfacing.

LEARNING EXPERIENCE GAINED

The students learn both soft skills and technical skills while they are undergoing the practical sessions. The soft skills gained in terms of communication, presentation and behavior. While technical skills they gained in terms of microprocessor programming and microcontroller interfacing.

LIST OF PRACTICAL ASSIGNMENTS:

1. Write a assembly language program for multiplication of two 16-bit numbers.
2. Write a assembly language program to add the elements of array.
3. Write a assembly language program to count number of negative and positive numbers from given array.
4. Write a assembly language program to arrange given set of numbers in ascending order and descending order.
5. Write a assembly language program perform string operations.(to check whether entered string is palindrome or not and to compare two strings).
6. Write a assembly language program(for 80386) for the multiplication of two 32-bit numbers

7. Write a assembly language program for 8-bit BCD addition and subtraction
8. Write a assembly language program to display current time from system
9. Write a assembly language program to interface 8051 microcontroller with keyboard
10. Write a assembly language program to interface 8051 microcontroller with DC motor

Note: All assignments based on Intel processor specific language.

Introduction to 8086 Architecture

- A microprocessor is a computer processor that incorporates the functions of a computer's central processing unit (CPU) on a single integrated circuit (IC), or at most a few integrated circuits.
- The microprocessor is a multipurpose, programmable device that accepts digital data as input, processes it according to instructions stored in its memory, and provides results as output.
- Microprocessors contain both combinational logic and sequential digital logic. Microprocessors operate on numbers and symbols represented in the binary numeral system.
- The integration of a whole CPU onto a single chip or on a few chips greatly reduced the cost of processing power.
- Integrated circuit processors are produced in large numbers by highly automated processes resulting in a low per unit cost.
- Single-chip processors increase reliability as there are many fewer electrical connections to fail. As microprocessor designs get faster, the cost of manufacturing a chip (with smaller components built on a semiconductor chip the same size) generally stays the same.

Features of 8086 Microprocessor:

1. Intel 8086 was launched in 1978.

2. It was the first 16 bit microprocessor.

3. This microprocessor had major improvement over the execution speed of 8085

4.It is available as 40 pin Dual Inline Package (DIP).

5. It is available in three versions:

a.8086 (5 MHz)

b. 8086 - 2 (8 MHz)

c. 8086 - 1 (10 MHz)

6. It consists of 29,000 transistors.

Concepts of architecture of 8086 microprocessor

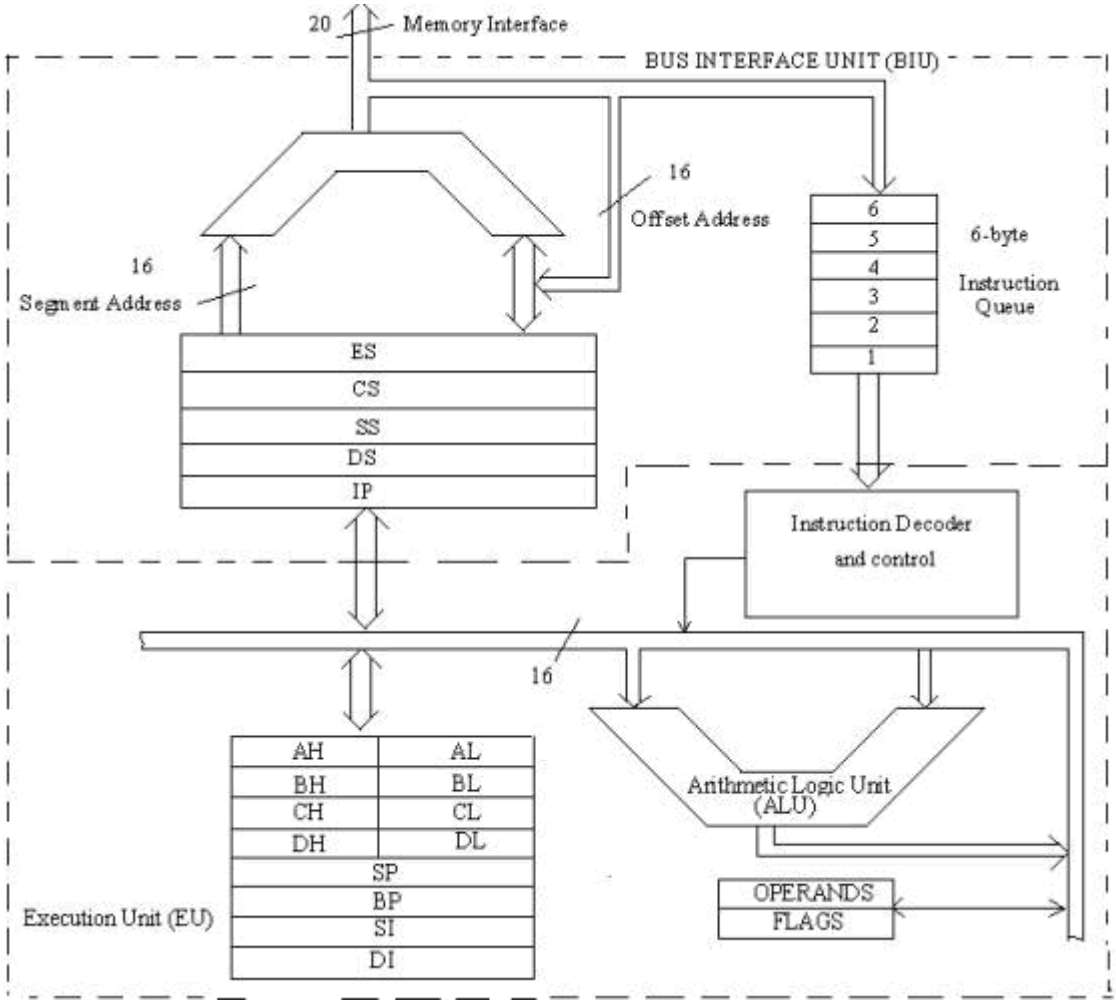


Figure. Architecture of 8086 microprocessor

The 8086 microprocessor is divided into two independent functional units:

1. Bus Interface Unit (BIU)

2. Execution Unit(EU)

Bus Interface Unit (BIU)

The function of BIU is to:

1. Fetch the instruction or data from memory.
2. Write the data to memory.
3. Write the data to the port.
4. Read data from the port

Instruction Queue

1. To increase the execution speed, BIU fetches as many as six instruction bytes ahead to time from memory.
2. All six bytes are then held in first in first out 6 byte register called instruction queue.
3. Then all bytes have to be given to EU one by one.
4. This pre fetching operation of BIU may be in parallel with execution operation of EU, which improves the speed execution of the instruction.

Instruction Pointer(IP)

1. The Instruction Pointer is a register that holds the address of the next instruction to be fetched from memory.
2. This register is responsible for holding the 16 bit offset, of the next code byte within this code segment.
3. The value contained in the IP is referred to as the offset because this value must be "offset" from(added to)the segment base address in code segment to produce the required 20 bit physical address sent out by the BIU.

Execution Unit (EU)

1. The functions of execution unit are:
2. To tell BIU where to fetch the instructions or data from.
3. To decode the instructions.
4. To execute the instructions.

The EU contains the control circuitry to perform various internal operations. A decoder in EU decodes the instruction fetched memory to generate different internal or external control signals required to perform the operation. EU has 16- bit ALU, which can perform arithmetic and logical operations on 8-bit as well as 16 - bit.

Intel 8086 is a 16 bit integer processor. It has 16-bit data bus and 20-bit address bus. The lower 16- bit address lines and 16-bit data lines are multiplexed (AD0-AD15)

Tool to perform the microprocessor based assignment.

Turbo Assembler (**TASM**) is a computer assembler (software for program development) developed by Borland which runs on and produces code for 16- or 32-bit x86 MS-DOS or Microsoft Windows. It can be used with Borland's high-level language compilers, such as Turbo Pascal, Turbo Basic, Turbo C and Turbo C++. The Turbo Assembler package is bundled with the Turbo Linker, and is interoperable with the Turbo Debugger. TASM can assemble Microsoft Macro Assembler (MASM) source using its MASM mode and has an ideal mode with a few enhancements. Object-Oriented programming has been supported since version 3.0. The last version of Turbo Assembler is 5.4, with files dated 1996 and patches up to 2010; it is still supplied with Delphi and C++Builder.

TASM itself is a 16-bit program; it will run on 16- and 32-bit versions of Windows, and produce code for the same versions. There are ways to run 16-bit programs such as TASM on 64-bit Windows (e.g., on a virtual machine), but it will not generate 64-bit Windows code.

Command to execute the assembly language program

1. To compile:

   Tasm programname.asm

2. To link after successful compilation

   Tlink programname.obj

3. To execute after linking

   Programname

Assignment 1

Write an assembly language program for multiplication of two 16-bit numbers

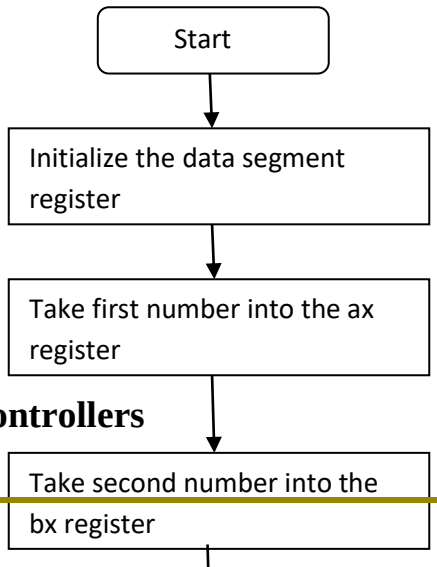
Theory:-

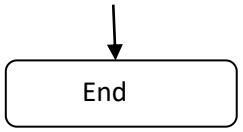
- Consider two 16-bit numbers. First number is present in AX and second number present in BX.
- We have to multiply the numbers from AX and BX.
- We will use “mul” instruction for multiplication

Algorithm

1. Declare the two 16-bit numbers for multiplication
2. Initialize the data segment register
3. Take first number in ax(accumulator)
4. Take second number in bx(base register)
5. Multiply two numbers
6. Result is greater than 16-bit so the result is partitions into two parts MSB of is stored into dx and LSB of result is stored into ax
7. Call the display subroutine to display the result.

Flowchart:





Questions:-

1. Describe the accumulator register.
2. Enlist the features of 8086 microprocessor .
3. Explain the use of base register.
4. Enlist the arithmetic instructions in assembly language
5. Explain the use int 21h and enlist any two functions of int 21h.

Assignment 2

Write an assembly language program to add the elements of array

THEORY:

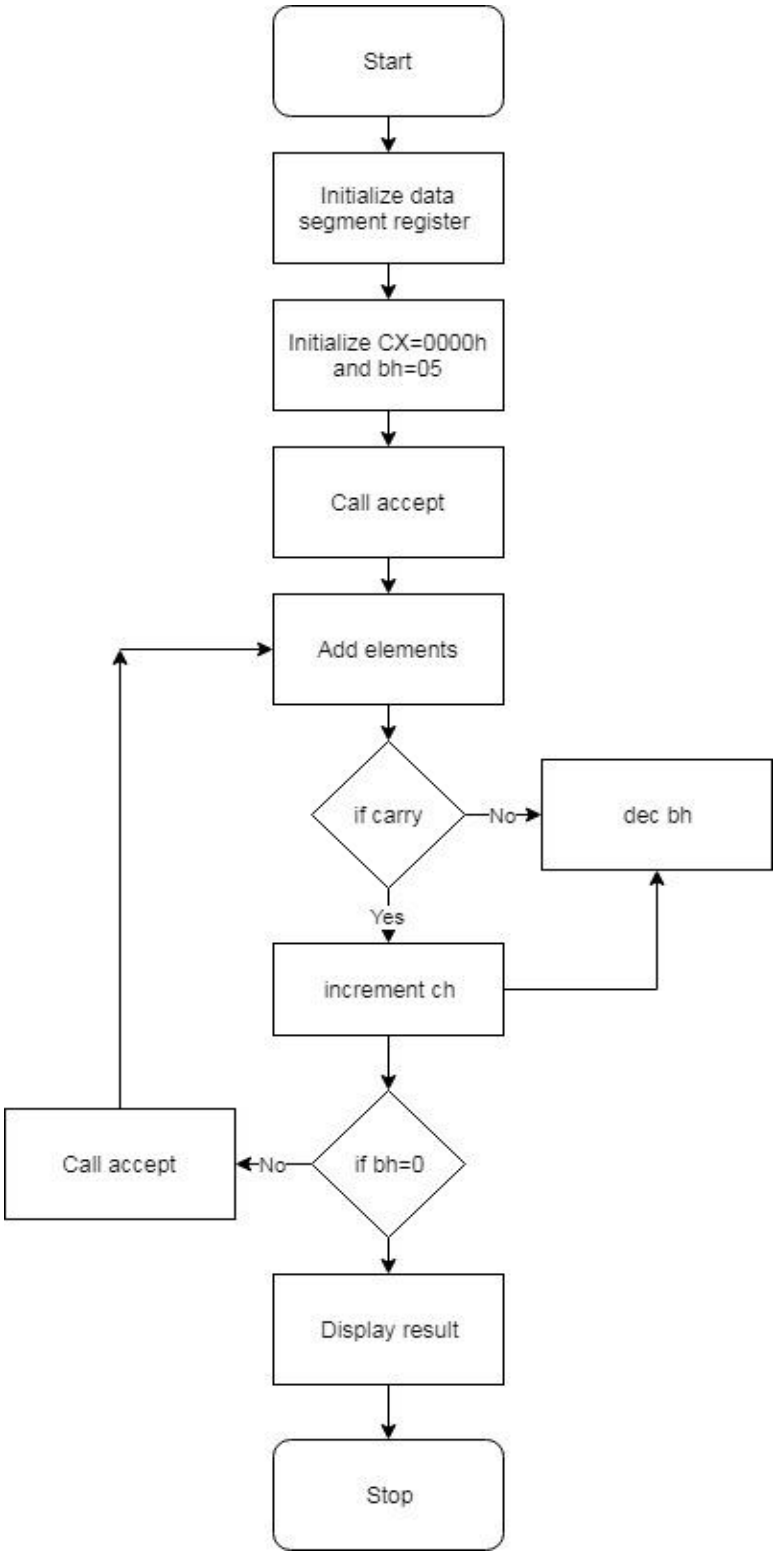
Array is the set of elements. We have to add five elements in the array.

1. Initialize data segment register.
2. Initialize CX=0000h and bh=05
3. Call accept
4. Add data
5. If carry will generate then increment ch
6. If there is no carry then decrement bh
7. Call display
8. stop

Algorithm:

1. Initialize data segment register
2. Accept array of five 8-bit numbers using accept procedure.
3. Add the elements of array
4. Call display procedure.

Flowchart


Start
Initialize data segment register
Initialize CX=0000h and bh=05
Call accept
Add elements
if carry
No
dec bh
Yes
increment ch
Call accept
No
if bh=0
Display result
Stop

Questions:-

1. Enlist various assembler directives.
2. Explain accept procedure.
3. Explain display procedure.
4. Explain following instructions:
   i) jnc
   ii) inc
   iii) mov
5. Explain the use of following function
   i) mov ah, 4ch
   ii) mov ah,01h

Assignment 3

Write an assembly language program to count number of negative and positive numbers from given array.

Theory:

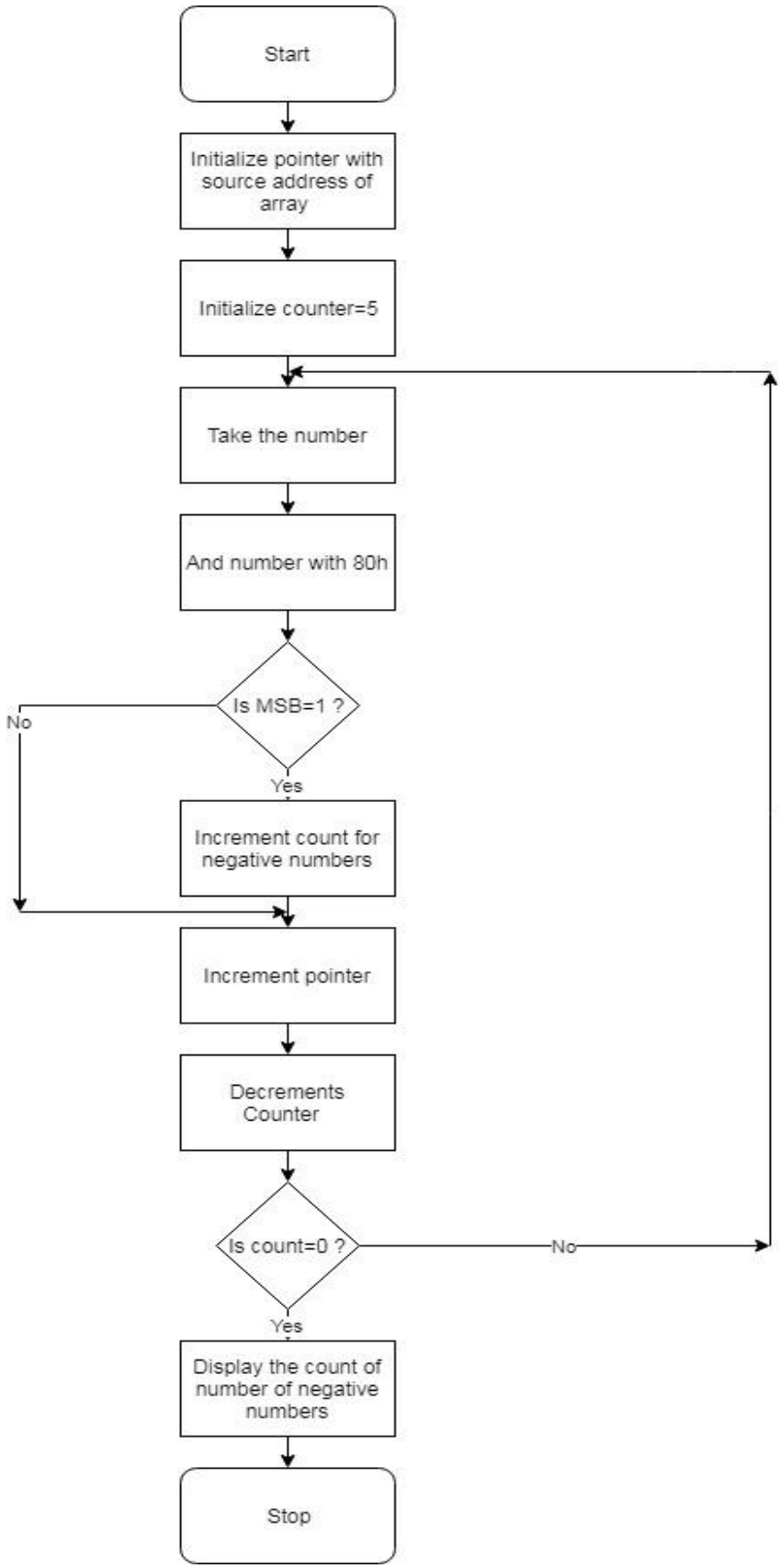
Part A: Write an assembly language program to count number of negative numbers from given array.

We have an array of N numbers. We initialize the count with 5 for eg. Also we initialize a pointer to point the elements in the array. We will check for the MSB. If the MSB of number is 1, the number is negative. Increment count for counting the negative numbers. Increment the pointer to pit to the next element. Check if MSB is 1. Repeat process till all the numbers are scanned. Display count of negative numbers.

And if the MSB of number is 0, then number is positive.

Algorithm:

1. Initialize data segment register
2. Initialize pointer with source address of array
3. Initialize count for number of elements
4. Get the number using accept procedure
5. AND the number with 80H
6. Check if MSB is 1. If yes go to step 7, else go to step 8
7. Increment count for counting negative numbers
8. Increment pointer
9. Decrement count
10. Check for count=0. If yes, go to step 11, else go to step 4
11. Display the count for negative numbers
12. Stop


Start
Initialize pointer with source address of array
Initialize counter=5
Take the number
And number with 80h
Is MSB=1 ?
No
Yes
Increment count for negative numbers
Increment pointer
Decrements Counter
Is count=0 ?
No
Yes
Display the count of number of negative numbers
Stop

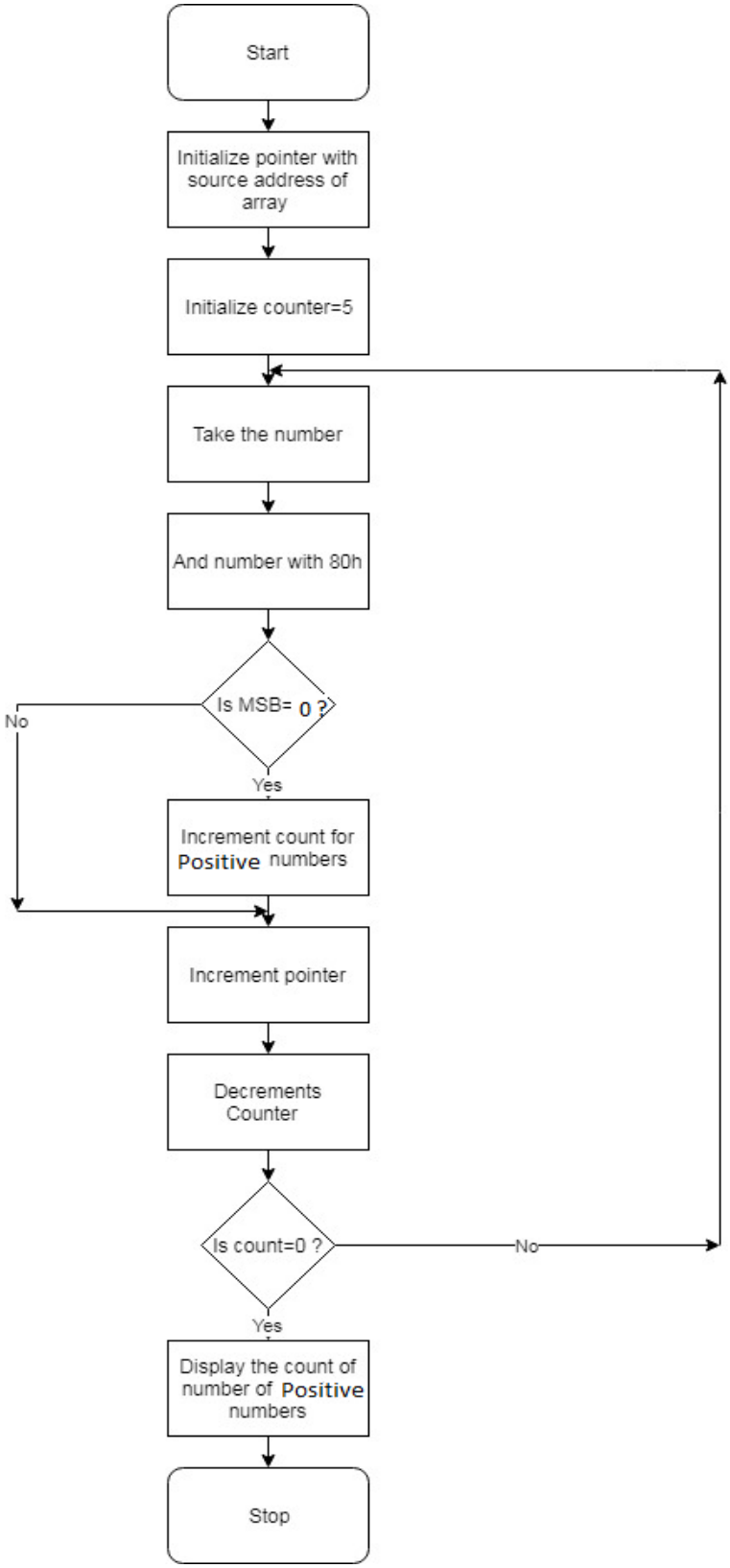
Part A: Write an assembly language program to count number of positive numbers from given array.

Write an assembly language program to count number of positive numbers from given array.

We have an array of N numbers. We initialize the count with 5 for eg. Also we initialize a pointer to point the elements in the array. We will check for the MSB. If the MSB of number is 0, the number is positive. Increment count for counting the negative numbers. Increment the pointer to pit to the next element. Check if MSB is 0. Repeat process till all the numbers are scanned. Display count of positive numbers.

Algorithm:

1. Initialize data segment register
2. Initialize pointer with source address of array
3. Initialize count for number of elements
4. Get the number using accept procedure
5. AND the nu,mber with 80H
6. Check if MSB is 0. If yes go to step 7, else go to step 8
7. Increment count for counting positive numbers
8. Increment pointer
9. Decrement count
10. Check for count=0. If yes, go to step 11, else go to step 4
11. Display the count for positive numbers
12. Stop


Start
Initialize pointer with source address of array
Initialize counter=5
Take the number
And number with 80h
Is MSB= 0 ?
No
Yes
Increment count for Positive numbers
Increment pointer
Decrements Counter
Is count=0 ?
No
Yes
Display the count of number of Positive numbers
Stop

Questions:

1. Explain the concept of positive and negative numbers
2. Describe following instruction
   i. and
   ii. or
   iii. add
3. Explain the counter register.
4. Explain the significance of code and data segment is program.
5. Define the memory model in assembly language.

Write an assembly language program to arrange given set of numbers in ascending order and descending order.

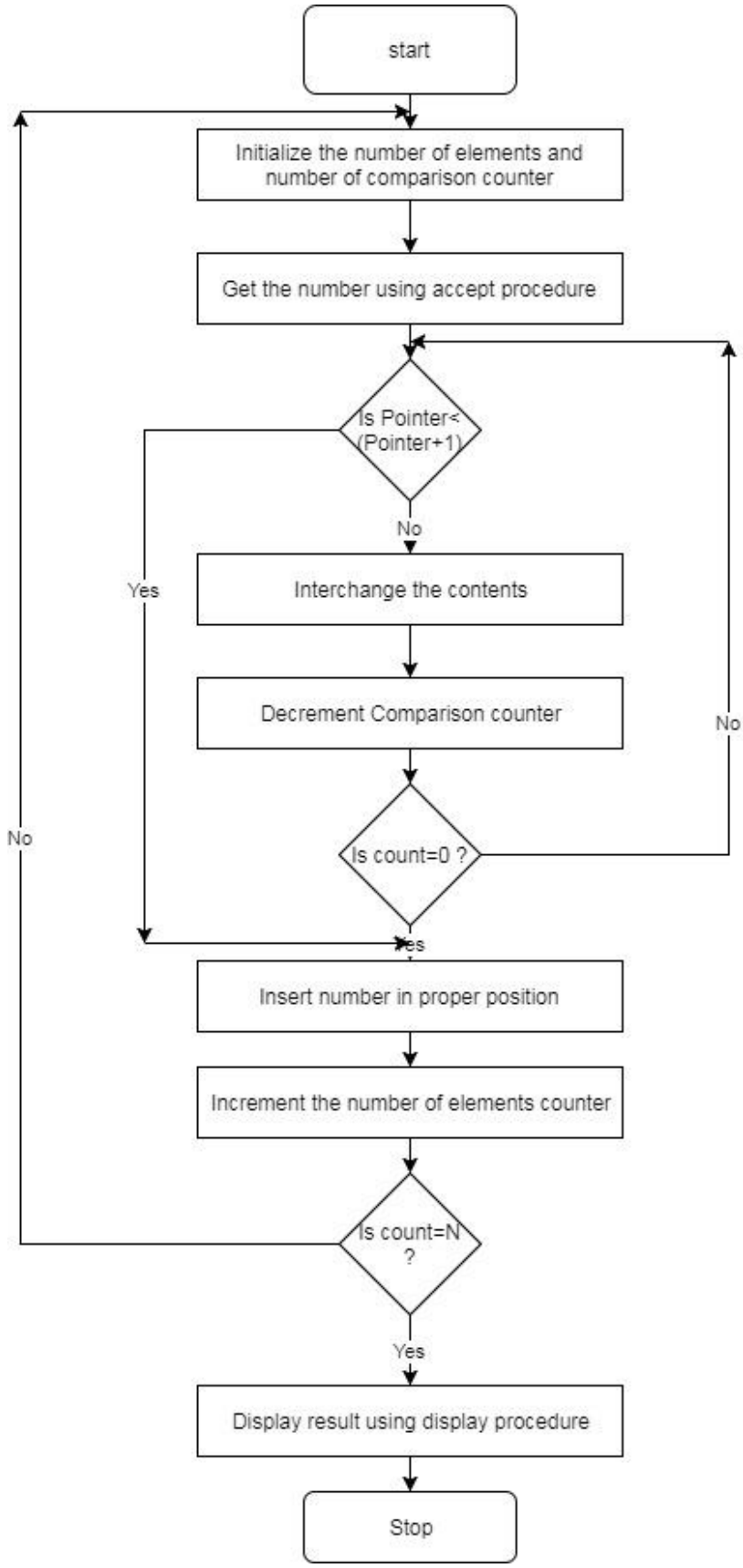
Theory

- Consider that a block of N words is present. Now we have arrange N words in ascending order. Let N=4 for example we will use SI as pointer to point the block of N words.
- Initially in the first iteration we compare first number with the second number. If first number < second number, don't interchange the contents, otherwise if first number > second number swap the contents.
- In the next iteration we will perform the same procedure until all numbers will sort.

Algorithm

1. Initialize the data segment register.
2. Initialize the number of elements counter.
3. Initialize the number of comparison counter.
4. Compare the elements. If first element < second element go to step 8 else go to step 5.
5. Swap the elements
6. Decrement the comparison counter
7. If count = 0 ? if yes go to step 8 else go to step 4
8. Insert number in proper position.
9. Increment the number of elements counter.
10. If count = N ? if yes, go to step 11 else go to step 2
11. Display the result using display procedure.
12. Stop

Note: for descending the same algorithm and flow chart except we have compare as first element > second element.


start
Initialize the number of elements and number of comparison counter
Get the number using accept procedure
Is Pointer< (Pointer+1)
No
Yes
Interchange the contents
Decrement Comparison counter
No
Is count=0 ?
No
Insert number in proper position
Increment the number of elements counter
Is count=N ?
Yes
Display result using display procedure
Stop

Questions:-

1. What is the use sounce index register?
2. Explain “lea” instruction.
3. Explain any five addresing modes.
4. Define procedure in assembly language.
5. Explain following functions of int 21h
   i. mov ah,02h
   ii. mov ah,01h

Write an assembly language program perform string operations.(to check whether entered string is palindrome or not and to compare two strings).

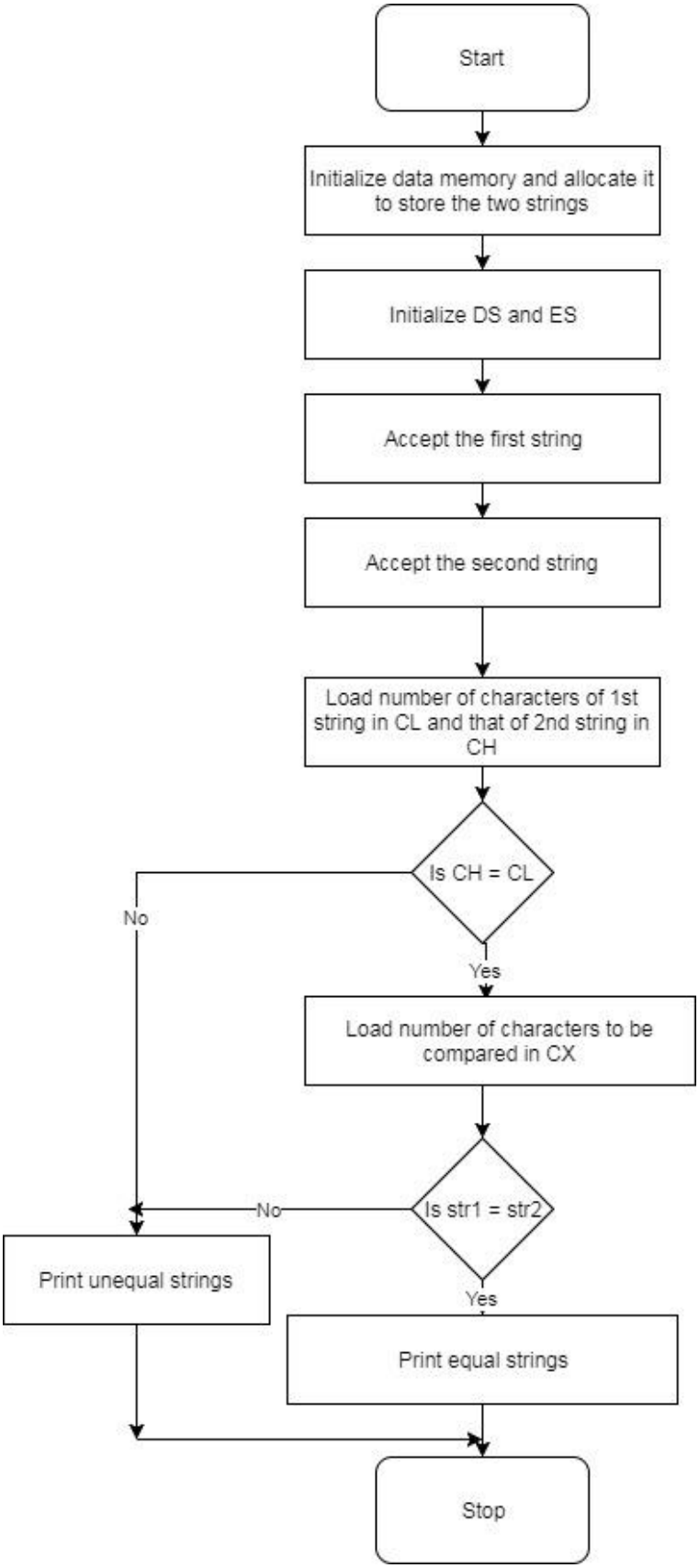
Theory :

Part A: Write an assembly language program to compare two strings.

- We will accept two strings from the user. After accepting two strings, first step in string comparison is to check whether the length of strings is same or not and then we will comapre each character from string.
- The source and destinationj address are initialized in DS:SI and ES:DI registers.
- Using the string instruction REPE CMPSB, two data are compared character by character.
- If all the characters are matching display the message “equal strings” otherwise display “unequal strings”

Algorithm:

1. Initialize the data memory
2. Allocate data memory to save the strings
3. Initialize DS and ES register
4. Accept the first string
5. Accept the second string
6. Load the number of characters of first string in CL
7. Load the number of characters of second string in CH register
8. Compare the lengths of the two strings. If not go to step 13
9. Load number of characters to be compared in CX
10. Compare the strings character by character. If not same go to step 13
11. Print “equal strings” using macro
12. Jump to step 14
13. Print “unequal strings” using macro
14. Stop

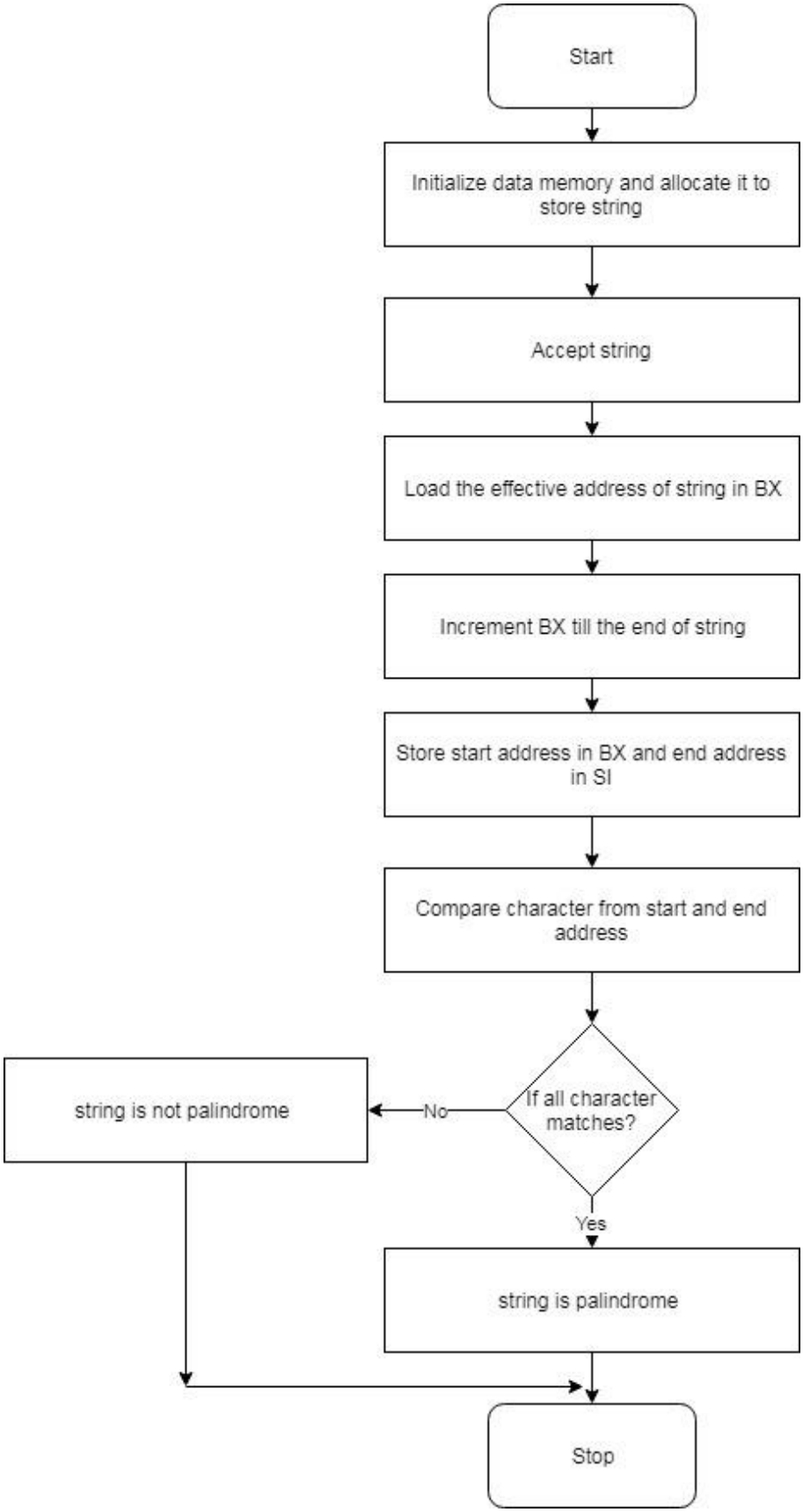

Start
Initialize data memory and allocate it to store the two strings
Initialize DS and ES
Accept the first string
Accept the second string
Load number of characters of 1st string in CL and that of 2nd string in CH
Is CH = CL
No
Yes
Load number of characters to be compared in CX
Is str1 = str2
No
Yes
Print unequal strings
Print equal strings
Stop

Part B: Write an assembly language program to check whether entered string is palindrome or not.

Theory:
A **palindrome** is a word, phrase, number, or other sequence of characters which reads the same backward as forward, such as madam or “taco cat” or racecar. Sentence-length palindromes may be written when allowances are made for adjustments to capital letters, punctuation, and word dividers, such as "A man, a plan, a canal, Panama!", "Was it a car or a cat I saw?" or "No 'x' in Nixon".

Algorithm:

1. Initialize the data memory
2. Allocate data memory to save the strings
3. Initialize DS and ES register
4. Accept the string
5. Load the effective address of string in BX
6. Increment BX till the end of string
7. Store starting address in BX and end address in SI
8. Compare character from start and end address.
9. If all character then string is palindrome otherwise not.
10. stop


Start
Initialize data memory and allocate it to store string
Accept string
Load the effective address of string in BX
Increment BX till the end of string
Store start address in BX and end address in SI
Compare character from start and end address
If all character matches?
No
string is not palindrome
Yes
string is palindrome
Stop

Questions:-

1. Define Palindrome.
2. What is the need to initialize the data segment register.
3. Explain data memory and program memory.
4. Explain the “cld” instruction.
5. Explain the various string instructions.

Assignment No. 6

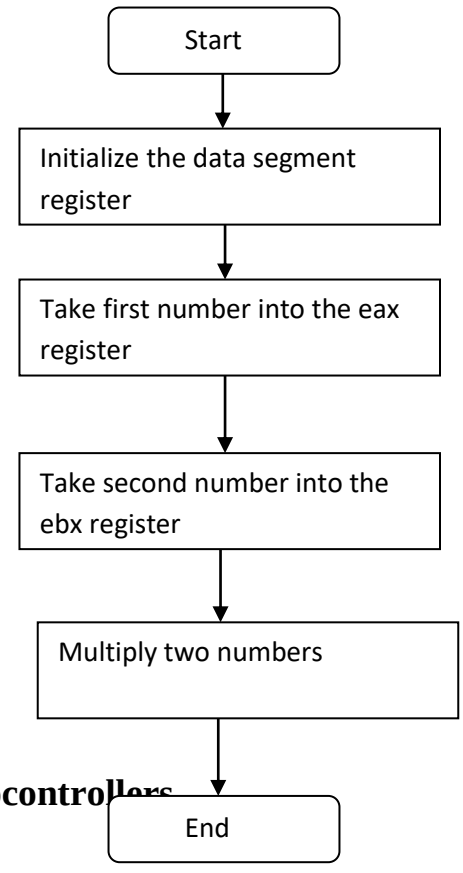
Title: Write and build ALP(80386) for multiplication of two 32-bit numbers.

Theory

The Intel 80386 ("eight-oh-three-eighty-six"), also known as i386 or just 386, is a 32-bit microprocessor introduced in 1985. The first versions had 275,000 transistors and were the CPU of many workstations and high-end personal computers of the time. As the original implementation of the 32-bit extension of the 80286 architecture, the 80386 instruction set, programming model, and binary encodings are still the common denominator for all 32-bit x86 processors, which is termed the i386-architecture, x86, or IA-32, depending on context.

Algorithm

8. Declare the two 16-bit numbers for multiplication
9. Initialize the data segment register
10. Take first number in eax(extended accumulator)
11. Take second number in ebx(extended base register)
12. Multiply two numbers
13. Result is greater than 32-bit so the result is partitions into two parts MSB of is stored into edx and LSB of result is stored into eax
14. Call the display subroutine to display the result



Questions:

1. Enlist the various segment registers in 80386.
2. Explain addressing modes in 80386.
3. Illustrate the use test register in 80386.
4. Justify the need to partition the result of multiplication.
5. Explain segmentation mechanism in 80386.

Assignment No. 7

Write an assembly language program for 8-bit BCD addition and subtraction

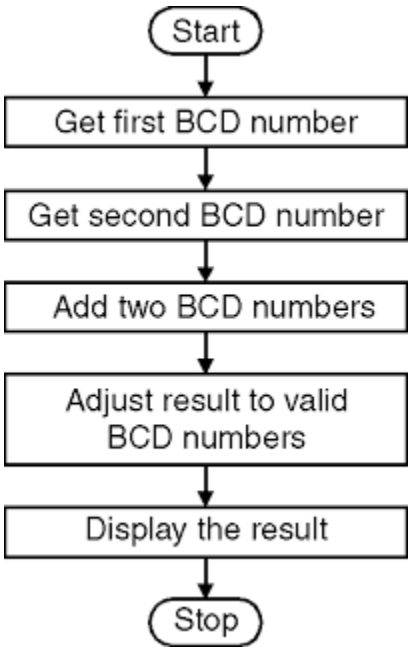
Theory

- Consider that a byte of data is present in the AL register and a second byte of data is present in the BL register. We have to add byte in AL with the byte in BL. Using add instruction, add the contents of 2 registers.
- Result will be stored in the AL register.
- Use DAA instruction that will check if BCD is valid, if it is not valid then 6 is added to give proper BCD result. Display the result.

Algorithm :

- **Step I :** Initialize the data memory.
- **Step II :** Get the first BCD number in AL.
- **Step III :** Get the second BCD number in BL.
- **Step IV :** Add the two BCD numbers.
- **Step V :** Using DAA, adjust result to valid BCD number.
- **Step VI :** Display the result.
- **Step VII :** Stop.

Flowchart



8086 program to subtract 8 bit BCD numbers

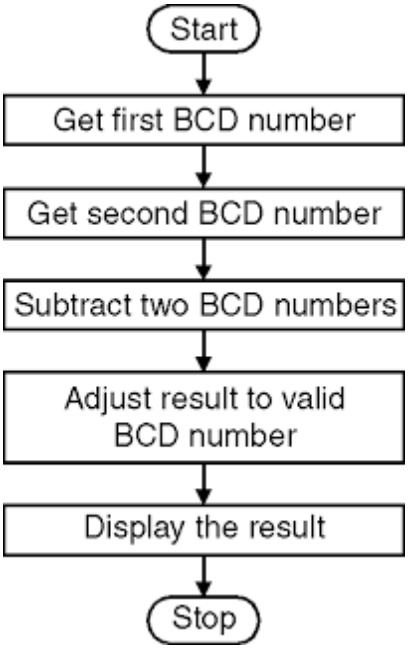
Consider that a byte of data is present in the AL register and a second byte of data is present in the BL register. We have to subtract byte in BL with byte in AL. Using SUB instruction, we will subtract the contents. The result is stored in the AL register. Using DAS instruction we will make sure that the result is valid BCD.

eg. :	AL = 32 BCD		0	0	1	1	0	0	1	0	
	BL = 17 BCD	–									
			0	0	0	1	0	1	1	1	
			0	0	0	1	1	0	1	1	invalid BCD, so
		–									subtract 6.
		DAS					0	1	1	0	
			0	0	0	1	0	1	0	1	= 15 BCD

Invalid BCD, means if digit is greater than 9. If digit is invalid we subtract 6 to make it valid. DAS instruction (decimal adjust after subtraction), automatically adjusts to valid BCD result.

Algorithm

Step I : Initialize the data memory.
Step II : Get the first BCD number in AL.
Step III : Get the second BCD number in BL.
Step IV : Subtract the two BCD numbers.
Step V : Adjust result to valid BCD number.
Step VI : Display the result.
Step VII : Stop.



Questions:-

1. Define BCD numbers.
2. Explain the concept of valid and invalid BCD.
3. Explain DAA instruction
4. Justify the significance of BCD numbers.
5. Explain the concept of valid BCD and Invalid BCD

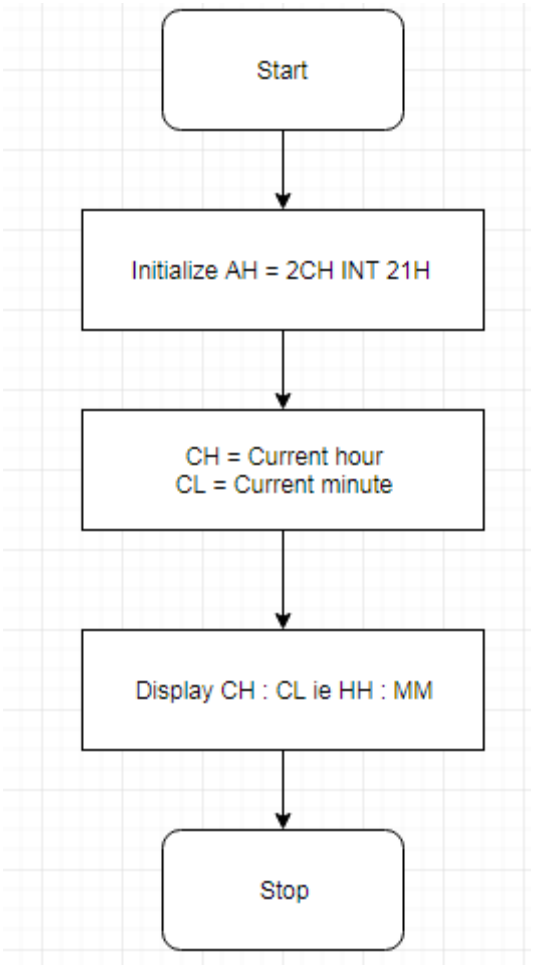
Write an assembly language program to display current time from system

Theory:
The system lines can be read in CX register by using function 2CH. The contents available in the CX register is displayed on the screen using DOS interrupt int 21h, function 02h.

Algorithm

1. Initialise AH register with 2CH and call DOS interrupt.
2. Register CH indicates current hour in 24 hour format (0 through 23) and CL indicates the current minute (0 through 59).
3. Use INT 21H function 02H to display the contents of CX in the format HH:MM.
4. Stop

Flowchart



Questions:

1. Explain various functions to display current time from system.
2. Explain counter register.
3. Explain following function: mov ah, 2ch
4. Explain DAA instruction.

Assignment No. 9

Write an assembly language program to interface 8051 microcontroller with keyboard

Theory:

Introduction to MCS-8051

The Intel MCS-51 (commonly termed 8051) is an internally Harvard architecture, complex instruction set computer (CISC) instruction set, single chip microcontroller (µC) series developed by Intel in 1980 for use in embedded systems. Intel's original versions were popular in the 1980s and early 1990s and enhanced binary compatible derivatives remain popular today.

Intel's original MCS-51 family was developed using N-type metal-oxide-semiconductor (NMOS) technology like its predecessor Intel MCS-48, but later versions, identified by a letter C in their name (e.g., 80C51) used complementary metal–oxide–semiconductor (CMOS) technology and consume less power than their NMOS predecessors. This made them more suitable for battery-powered devices.

Interfacing of 8051 with keypad

This program scans the keypad.

1. While no key is pressed the program scans row0, row1, row2, row3 and back to row0, continuously.
2. When a key is pressed the key number is placed in R0.
3. For this program, the keys are numbered as

11	10	9	Row 3
8	7	6	Row 2
5	4	3	Row 1
2	1	0	Row 0
Col 2	Col 1	Col 0	

4. The pressed key number will be stored in R0. Therefore, R0 is initially cleared.
5. Each key is scanned, and if it is not pressed R0 is incremented. In that way, when the pressed key is found, R0 will contain the key's number.
6. The general purpose flag, F0, is used by the column-scan subroutine to indicate whether or not a pressed key was found in that column.
7. If, after returning from colScan, F0 is set, this means the key was found.

Questions:

1. Explain the concept and need of interfacing.

2. Describe the architecture of 8051 microcontroller
3. Enlist various special function registers in 8051.
4. What is register bank?

Assignment No. 10

Write an assembly language program to interface 8051 microcontroller with DC Motor.

Theory:

DC Motor:

A DC motor is any of a class of rotary electrical machines that converts direct current electrical energy into mechanical energy. The most common types rely on the forces produced by magnetic fields. Nearly all types of DC motors have some internal mechanism, either electromechanical or electronic; to periodically change the direction of current flow in part of the motor.

DC motors were the first type widely used, since they could be powered from existing direct-current lighting power distribution systems. A DC motor's speed can be controlled over a wide range, using either a variable supply voltage or by changing the strength of current in its field windings. Small DC motors are used in tools, toys, and appliances. The universal motor can operate on direct current but is a lightweight motor used for portable power tools and appliances. Larger DC motors are used in propulsion of electric vehicles, elevator and hoists, or in drives for steel rolling mills. The advent of power electronics has made replacement of DC motors with AC motors possible in many applications.

Interfacing of 8051 with DC motor

This program exercises the motor.

1. The motor is rotated in a clockwise direction and the number of revolutions is displayed on Display 0 (the 7-segment display). The display only shows up to nine revolutions and then resets.
2. The motor sensor is connected to P3.5, which is the external clock source for timer 1. Therefore, timer 1 is put into event counting mode. In this way, the timer increments once every motor revolution.
3. The value in timer 1 low byte is moved to A and this value together with the data pointer (DPH and DPL) are used to get the 7-segment code from program memory.
4. The code is then sent to P1 to put the appropriate number on the Display 0.
5. The motor can be changed from clockwise to anti-clockwise by pressing SW0 (on P2.0).
6. The motor direction is stored in F0 (1 for clockwise, 0 for anti-clockwise). The value in F0 is sent to Display 0's decimal point (P1.7). This indicates the motor's direction - if the

decimal point is lit, the motor is rotating anti-clockwise, while if it is not lit the motor is rotating clockwise.

7. The value in F0 is compared with the value of SW0. If they are the same the motor direction does not need to be changed. If they are not the same it means the user has pressed SW0 and the motor direction must be reversed. When this happens the new motor direction is then stored in F0.

Questions:

1. Illustrate the concept of DC Motor
2. Explain the various port registers used while interfacing the of 8051 with DC motor.
3. Explain counter and timer in 8051