METHODIST

COLLEGE OF ENGINEERING AND TECHNOLOGY

Estd:2008

(Affiliated to Osmania University & Approved by AICTE, New Delhi)

LABORATORY MANUAL

PROGRAMMING FOR PROBLEM SOLVING LABORATORY

BE AI & DS III Semester (Autonomous): 2022-23

NAME: _____

ROLL NO:_____

BRANCH:_____ **SEM:**_____

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERNG

Empower youth- Architects of Future World

VISION

To produce ethical, socially conscious and innovative professionals who would contribute to sustainable technological development of the society.

MISSION

To impart quality engineering education with latest technological developments and interdisciplinary skills to make students succeed in professional practice.

To encourage research culture among faculty and students by establishing state of art laboratories and exposing them to modern industrial and organizational practices.

To inculcate humane qualities like environmental consciousness, leadership, social values, professional ethics and engage in independent and lifelong learning for sustainable contribution to the society.

METHODIST

Estd:2008 COLLEGE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT
OF
COMPUTER SCIENCE AND ENGINEERING
(AI & DS)

LABORATORY MANUAL
COMPUTER ORGANIZATION AND MICROPROCESSORS LAB

Prepared
By
Dr. Diana Moses,
Associate Professor,
Dept. of CSE.

METHODIST

COLLEGE OF ENGINEERING AND TECHNOLOGY

Estd:2008

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VISION & MISSION

VISION

To become a leader in providing Computer Science & Engineering education with emphasis on knowledge and innovation.

MISSION

- To offer flexible programs of study with collaborations to suit industry needs.
- To provide quality education and training through novel pedagogical practices.
- To expedite high performance of excellence in teaching, research and innovations.
- To impart moral, ethical values and education with social responsibility.

METHODIST

COLLEGE OF ENGINEERING AND TECHNOLOGY

Estd:2008

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PROGRAM EDUCATIONAL OBJECTIVES

After 3-5 years of graduation, the graduates will be able to

PEO1: Apply technical concepts, Analyze, Synthesize data to Design and create novel products and solutions for the real life problems.

PEO2: Apply the knowledge of Computer Science Engineering to pursue higher education with due consideration to environment and society.

PEO3: Promote collaborative learning and spirit of team work through multidisciplinary projects

PEO4: Engage in life-long learning and develop entrepreneurial skills.

METHODIST

COLLEGE OF ENGINEERING AND TECHNOLOGY

Estd:2008

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PROGRAM OUTCOMES

Engineering graduates will be able to:

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

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

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

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

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

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

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

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

PO9: Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10: Communication: Communicate effectively on complex engineering activities with the Engineering community and with society at large, such as, being able to

comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

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

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

PROGRAM SPECIFIC OUTCOMES

At the end of 4 years, Computer Science and Engineering graduates at MCET willbe able to:

PSO1: Apply the knowledge of Computer Science and Engineering in various domains like networking and data mining to manage projects in multidisciplinary environments.

PSO2: Develop software applications with open-ended programming environments.

PSO3: Design and develop solutions by following standard software engineering principles and implement by using suitable programming languages and platforms

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Course Code	Course Title						Core / Elective
1PC353AD	**Computer Organization and Microprocessors Lab**						**CORE**
Prerequisite	Contact Hours Per Week				CIE	SEE	Credits
	L	T	D	P			
–	–	–	–	2	25	50	2

Course Objectives:

- To learn programming the 8086 microprocessor
- To practice Assembly language programming.
- To interface 8086 with 8279, 8255 peripherals.

Course Outcomes:

- Evaluate the performance of different addressing modes in 8086.
- Implement programs for mathematical operations on 8-bit, 16-bit, 32-bit numbers.
- Implement assemble language programs for handing array and string operations.
- Perform Searching and sorting using assembly language programs
- Interfacing 8086 with 8279, 8255 peripherals.
- Build a traffic signal control system.

I. Introduction to 8086 and MASM Assembler

II. List of Experiments

1. Demonstration of data transfer instructions
2. 8-bit addition
3. 8-bit subtraction

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

4. 8-bit multiplication
5. 8-bit division
6. Square of 8-bit number
7. 16-bit addition
8. 16-bit subtraction
9. 16-bit multiplication
10. 16-bit division
11. Greatest of 2 numbers
12. Sum of array elements
13. Increment all elements in an array

14. Largest in an array
15. Count of all odd and even numbers in an array
16. Count of all positive and negative numbers in an array
17. 32-bit addition
18. 32-bit subtraction
19. String Operations
20. Searching for an element in an array
21. Sorting of array in ascending order
22. Sorting of array in descending order
23. Factorial of a number
24. Arithmetic expression evaluation
25. Block transfer of data

Estd:2008

METHODIST

COLLEGE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Course Outcomes (CO's):

SUBJECT NAME: Computer Organization and Microprocessors

LabCODE: 1PC353AD SEMESTER: III

CO No.	Course Outcome	Taxonomy Level
PC353AD.1	Evaluate the performance of different addressing modes in8086.	Evaluating
PC353AD.2	Implement programs for mathematical operations on 8-bit,16-bit, 32-bit numbers.	Applying
PC353AD.3	Implement assemble language programs for handing array and string operations.	Applying
PC353AD.4	Perform Searching and sorting using assembly languageprograms	Applying
PC353AD.5	Interfacing 8086 with 8279, 8255 peripherals.	Applying
PC353AD.6	Build a traffic signal control system.	Creating

GENERAL LABORATORY INSTRUCTIONS

1. Students are advised to come to the laboratory at least 5 minutes before (to starting time), those who come after 5 minutes will not be allowed into the lab.
2. Plan your task properly much before to the commencement, come prepared to the lab with the program / experiment details.
3. Student should enter into the laboratory with:
    a. Laboratory observation notes with all the details (Problem statement, Aim, Algorithm, Procedure, Program, Expected Output, etc.,) filled in for the lab session.
    b. Laboratory Record updated up to the last session experiments.
    c. Formal dress code and Identity card.
4. Sign in the laboratory login register, write the TIME-IN, and occupy the computer system allotted to you by the faculty.
5. Execute your task in the laboratory, and record the results / output in the lab observation note book, and get certified by the concerned faculty.
6. All the students should be polite and cooperative with the laboratory staff, mustmaintain the discipline and decency in the laboratory.
7. Computer labs are established with sophisticated and high end branded systems,which should be utilized properly.
8. Students / Faculty must keep their mobile phones in SWITCHED OFF mode during the lab sessions. Misuse of the equipment, misbehaviours with the staff and systems etc., will attract severe punishment.
9. Students must take the permission of the faculty in case of any urgency to go out. If anybody found loitering outside the lab / class without permission during working hours will be treated seriously and punished appropriately.
10. Students should SHUT DOWN the computer system before he/she leaves the lab after completing the task (experiment) in all aspects. He/she must ensure the system / seat is kept properly.

Head of the Department **Principal**

CODE OF CONDUCT FOR THE LABORATORY

- All students must observe the dress code while in the laboratory
- Footwear is NOT allowed
- Foods, drinks and smoking are NOT allowed
- All bags must be left at the indicated place
- The lab timetable must be strictly followed
- Be PUNCTUAL for your laboratory session
- All programs must be completed within the given time
- Noise must be kept to a minimum
- Workspace must be kept clean and tidy at all time
- All students are liable for any damage to system due to their own negligence
- Students are strictly PROHIBITED from taking out any items from the laboratory
- Report immediately to the lab programmer if any damages to equipment

BEFORE LEAVING LAB:

- Arrange all the equipment and chairs properly.
- Turn off / shut down the systems before leaving.
- Please check the laboratory notice board regularly for updates.

Lab In – charge

METHODIST

COLLEGE OF ENGINEERING AND TECHNOLOGY

Estd:2008

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

LIST OF EXPERIMENTS

S. No.	Name of the Experiment	Date of Experiment	Date of Submission	Page No	Faculty Signature
1.	Demonstration of data transfer instructions				
2.	8-bit addition				
3.	8-bit subtraction				
4.	8-bit multiplication				
5.	8-bit division				
6.	Square of 8-bit number				
7.	16-bit addition				
8.	16-bit subtraction				
9.	16-bit multiplication				
10.	16-bit division				
11.	Greatest of 2 numbers				
12.	Sum of array elements				
13.	Increment all elements in an array				
14.	Largest in an array				
15.	Count of all odd and even numbers in an array				
16.	Count of all positive and				

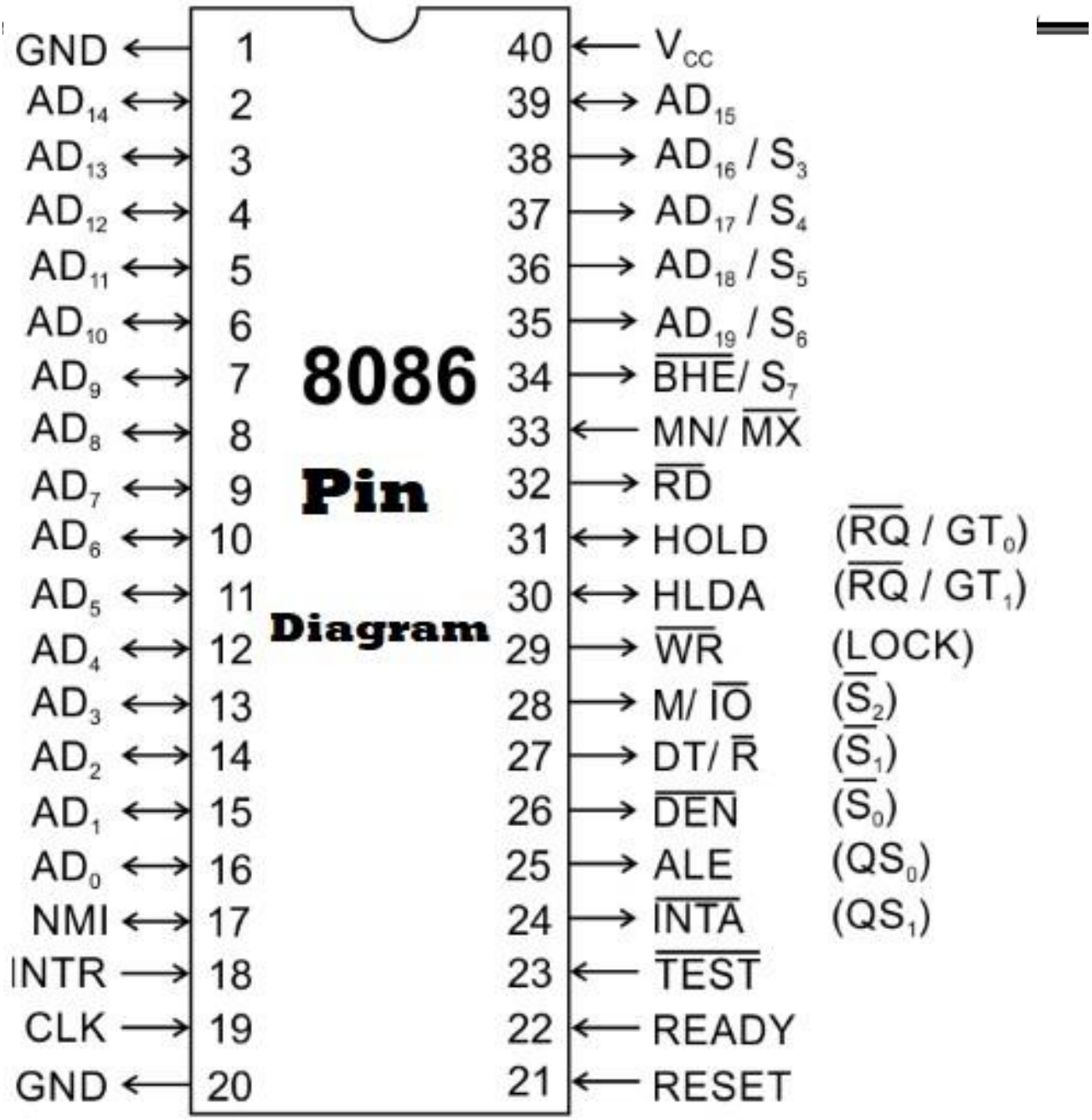
	negative numbers in an array				
17.	32-bit addition				
18.	32-bit subtraction				
19.	String Operations				
20.	Searching for an element in an array				
21.	Sorting of array in ascending order				
22.	Block transfer of data				
23.	Interfacing 8086 with 8279				
24.	Interfacing 8086 with 8255				
25.	Build a Traffic Signal Controller				

ADDITIONAL EXPERIMENTS

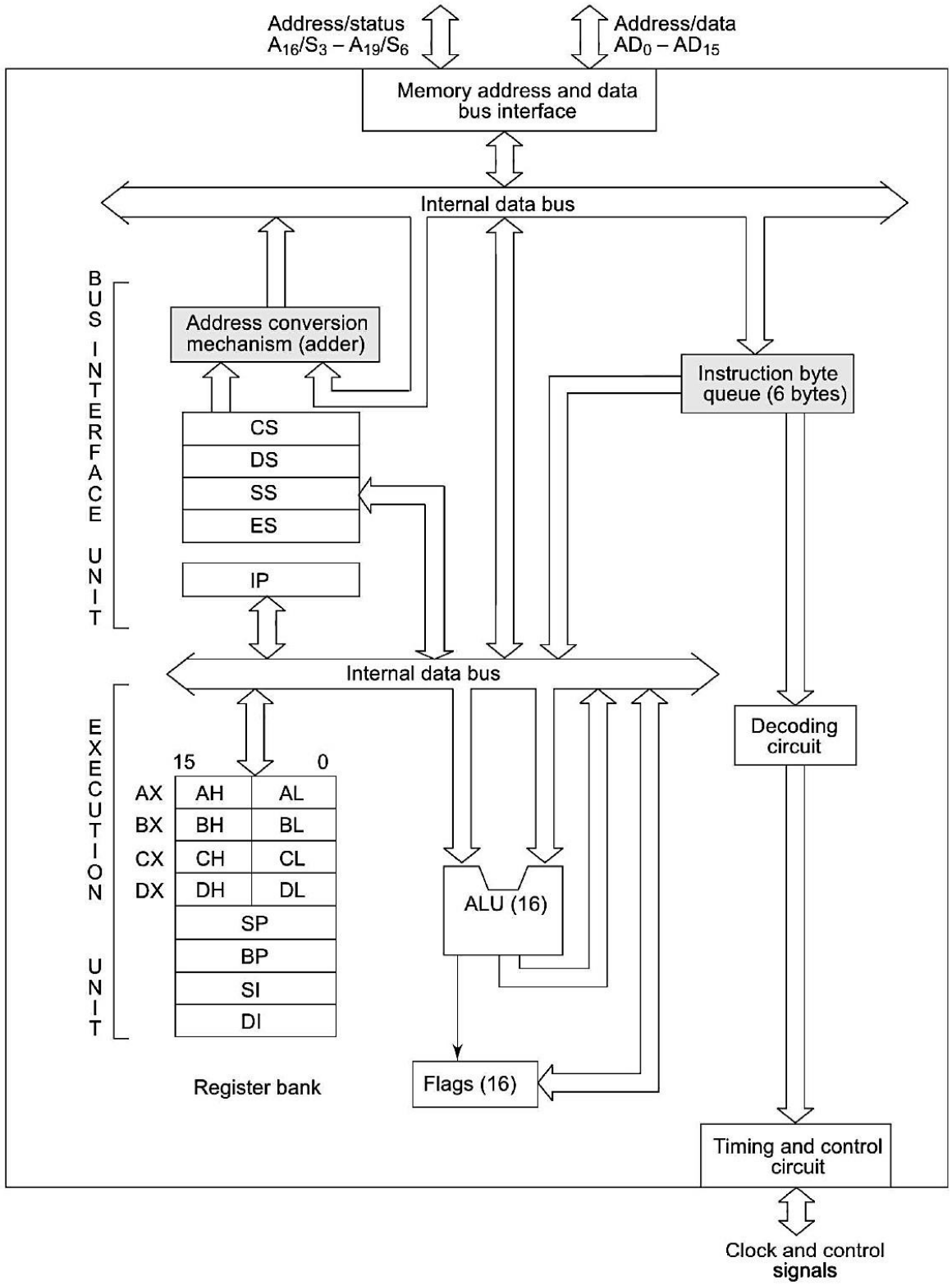
S. No.	Name of the Experiment	Date of Experiment	Date of Submission	Page No	Faculty Signature
1.	Factorial of a number				
2.	Arithmetic expression evaluation				
3.	Sorting of array in descending order				

Introduction to 8086 and MASM Assembler

8086 Pinout



8086 Architecture


Address/status
$A_{16}/S_3 - A_{19}/S_6$
Address/data
$AD_0 - AD_{15}$
Memory address and data bus interface
Internal data bus
BUS INTERFACE UNIT
Address conversion mechanism (adder)
Instruction byte queue (6 bytes)
CS
DS
SS
ES
IP
Internal data bus
EXECUTION UNIT
Decoding circuit
15
0
AX
BX
CX
DX
AH
AL
BH
BL
CH
CL
DH
DL
SP
BP
SI
DI
ALU (16)
Register bank
Flags (16)
Timing and control circuit
Clock and control signals

8086Addressing Modes

To perform any operation, we have to give the corresponding instructions to the microprocessor. In each instruction, programmer has to specify 3 things:

- Operation to be performed.
- Address of source of data.
- Address of destination of result.

The method by which the address of source of data or the address of destination of result is given in the instruction is called **Addressing Modes.** The term addressing mode refers to theway in which the operand of the instruction is specified.

Intel 8086 uses the following addressing modes:

1. Immediate
2. Register
3. Direct
4. Register Indirect
5. Indexed
6. Register relative
7. Base indexed
8. Relative base indexed
9. Intrasegment direct
10. Intrasegment indirect
11. Intersegment direct
12. Intersegment indirect

	Addressing mode	Effective address of the operand	Example
1.	Immediate	NA	MOV AX, 05H ADD BX, 10H
2.	Register	NA	MOV AX, BX SUB BX, CX
3.	Direct	10H * [DS] + Offset given instruction	MOV DX, [0500H] ADD AX, [0900H]
4.	Register Indirect	10H * [DS] + [Reg]	MOV BX, [CX]
5.	Indexed	10H * [DS] + [Index]	MOV AX, [SI] MOX [DI], CX
6.	Register relative	10H * [DS] + [Reg] + Disp	MOV AX, 50H[BX] SUB BX, 100H[DX]

7.	Base indexed	10H * [DS] + [Base] + [Index]	MOV AX, [BP][SI] MOV [BP][DI], CX
8.	Relative base indexed	10H * [DS] + [Base] + [Index] + Disp	MOV 1000H[BX][DI], AX ADD AX, 50H[BP][SI]
9.	Intrasegment direct	10H * [CS] + [IP] + Offset given instr	JMP 5000H
10.	Intrasegment indirect	10H * [CS] + [IP] +[Reg]	JMP [BP]
11.	Intersegment direct	EA = 10H * Base + Off given in Instr	JMP 7000H:2000H
12.	Intersegment indirect	EA = 10H * [ES] + M[Reg]	JMP [2000H

ASSEMBLY LANGUAGE PROGRAMMING USING MASM SOFTWARE

This software is a simulator that is used to write programs for 8086, Pentium processors etc and execute them on a computer without the use of an 8086 kit. The programs are written using assembly language in editor then compiled. The compiler converts assembly language statements into machine language statements and checks for errors. Then the compiled program can be executed.

There are different softwares developed by different companies for assembly language programming. They are

- **MASM** - Microsoft Company.
- **TASM** - Bore Land Company.

EXECUTION OF ASSEMBLY LANGUAGE PROGRAMMING IN MASM SOFTWARE:

Assembly language programming has 4 steps.

1. Entering Program
2. Compile Program
3. Linking a Program
4. Debugging a Program

PROCEDURE:

1. Entering

Program:-Start

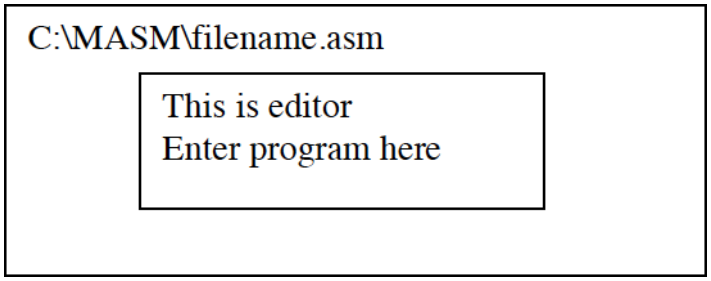
Menu ↩

Run ↩

Cmd ↩

C:\cd MASM ↩

C:\ MASM> edit filename.asm ↩



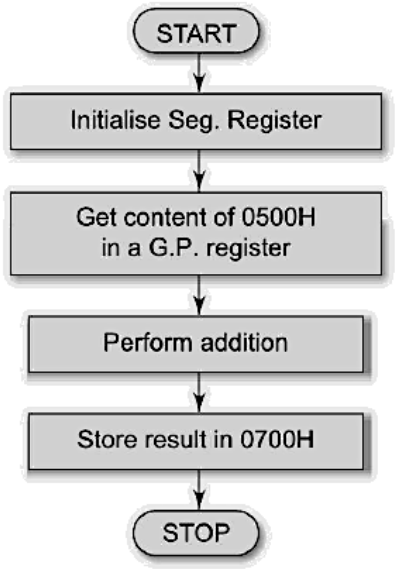
After entering program save & exit (ALT F & Press S or ALT F
&Press X)C:\MASM>
2. Compile the Program:-
C:\MASM> MASM
filename.asm↩
Microsoft @macro assembler version
5.10Copy rights reserved© Microsoft
Corp 1981 All rights
reservedObject filename
[OBJ];
List filename [NUL,
LIST]; Cross Reference
[NUL, CRF]; Press enter
the screen shows c>

3. Linking a
Program:-c> link
filename.obj ↩
Microsoft @ overlay linker version
3.64Copy rights reserved©
Microsoft corp. 1983-88. All rights
reserved
Object module
[.OBJ];Run file
[.EXE];
List [NUL MAP];

4. Debug a Program:- C> debug filename.exe
- (Screen shows only dash)
-t
‘t’ for trace the program execution by single stepping starting from the address SEG.OFFSET. ‘q’ for Quit from Debug & return to DOS.

1. Demonstration of data transfer instructions

Write a program to add a data byte located at offset 0500H in 2000H segment to another data byte available at 0600H in the same segment and store the result at 0700H in the same segment.




```
MOV AX, 2000H ;
MOV DS, AX ;
MOV AX, [0500H] ;
ADD AX, [0600H] ;
MOV [0700H], AX ;
HLT ;
```

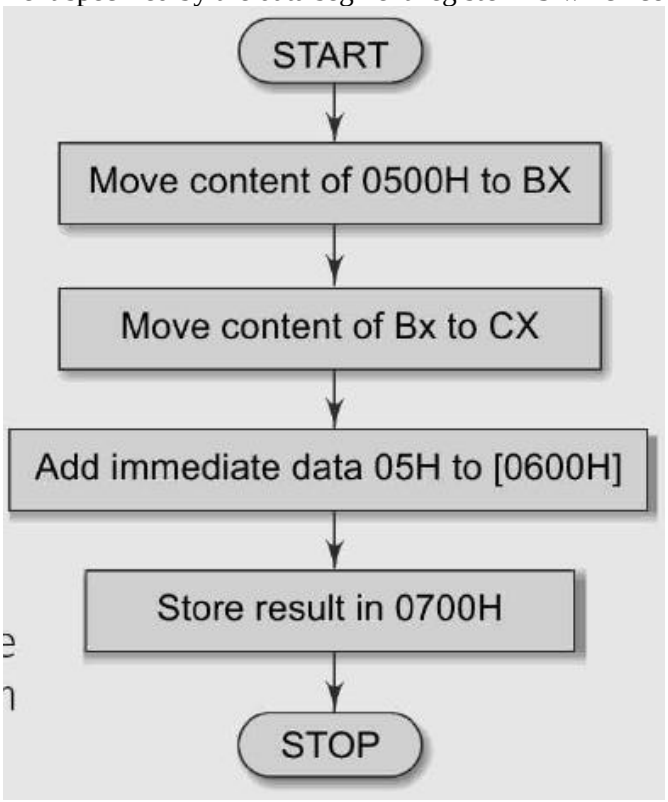
2. Write a program to demonstrate data transfer group of instructions and trace the execution

```
ASSUME CS:CODE,DS:DATA
DATA SEGMENT
        ABC DW 1101H
        DEF DW 0025H
        PQR DW 0011H
DATA ENDS
CODE SEGMENT
START:
        MOV AX,DATA
        MOV DS,AX
        MOV AX,ABC
        MOV BX,DEF
        MOV CX,PQR
        XCHG AX,BX
        PUSH AX
        PUSH BX
        PUSH CX
        POP CX
        POP BX
        POP AX
        MOV AX,ABC
        INC AX
        MOV ABC,AX
        MOV BX,DEF
        DEC BX
        MOV DEF,BX
        INT 3
CODE ENDS
END START
```

3. Write a program to demonstrate data transfer group of instructions and trace the execution

```
ASSUME CS:CODE,DS:DATA
DATA SEGMENT
        ABC DW 0000H
        DEF DW 1111H
DATA ENDS
CODE SEGMENT
START:
        MOV AX,DATA
        MOV DS,AX
        MOV AX,ABC
        MOV BX,ABC
        MOV CX,ABC
        MOV DX, ABC
        MOV AX, DEF
        XCHG AX,BX
        XCHG BX,CX
        XCHG CX,DX
        XCHG CX,DX
        XCHG BX,CX
        XCHG AX,BX
        INT 3
CODE ENDS
END START
```

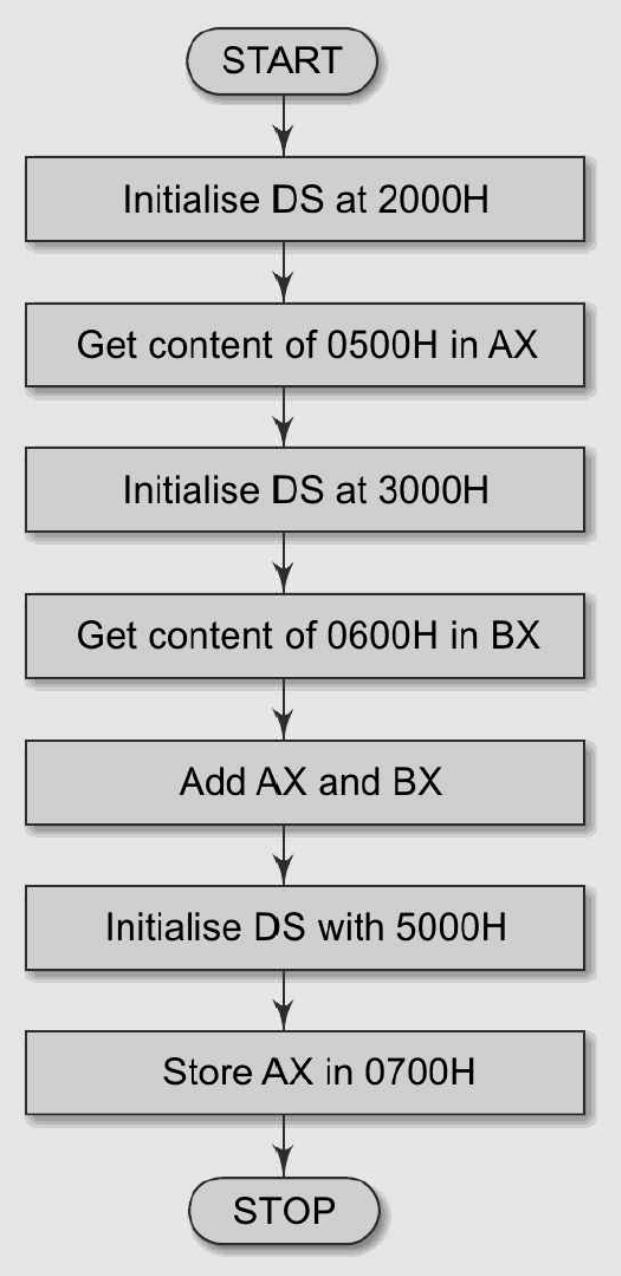
4. Write a program to move the contents of the memory location 0500H to register BX and to CX. Add immediate byte 05H to the data residing in Memory location whose address is computed using DS = 2000H. Store the result of addition in 0700H in the same segment Assume the data is located in the segment specified by the data segment specified by the data segment register DS which contains 2000H




```
MOV AX, 2000H
MOV DS, AX              ; Initialize Data Segment Register
MOV BX, [0500H]  ; Get content of 0500H to BX
MOV CX, BX              ; Copy the same content to CX
ADD [0600H], 05H ; Add 05h immediate data to memory location offset by 0600h
MOV DX, [0600H]  ; store the result in DX
MOV [0700H], DX  ; Store result in 0700H
HLT                     ; Stop
```

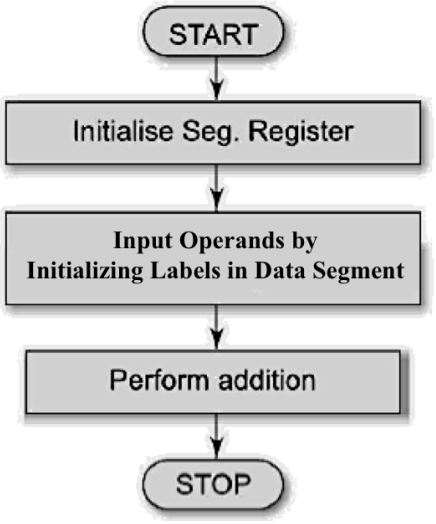
5. 8-bit addition
6. 8-bit subtraction

Subtract the contents of memory location 2000H:0500H from contents of 3000H:0600H and store the result in 5000H:0700H




```
MOV CX, 2000H       ; Initialize DS to 2000H
MOV DS, CX          ;
MOV AX, [0500H]     ; Get one operand in AX
MOV CX, 3000H       ; Initialize DS to 3000H
MOV DS, CX          ;
MOV BX, [0600H]     ; Get other operand in BX
SUB BX, AX          ; BX - AX
MOV CX, 5000H       ; Initialize DS to 5000H
MOV DS, CX          ;
MOV [0700H], BX     ; Store result in data to memory location offset by 0700h
HLT                 ;
```

7. Write a program to subtract two 8 bit numbers




```
ASSUME CS:CODE,DS:DATA
DATA SEGMENT
    NO1 DB 08H
    NO2 DB 06H
DATA ENDS
CODE SEGMENT
START:
    MOV AX,DATA
    MOV DS,AX
    MOV AL,NO1
    MOV BL,NO2
    SUB AL,BL
    INT 3H
CODE ENDS
END START
```

8. 8-bit multiplication

```
ASSUME CS:CODE,DS:DATA
DATA SEGMENT
        NO1 DB 08H
        NO2 DB 06H
DATA ENDS
CODE SEGMENT
START:
        MOV AX,DATA
        MOV DS,AX
        MOV AL,NO1
        MOV BL,NO2
        MUL BL
        INT 3H
CODE ENDS
END START
```

9. 8-bit division

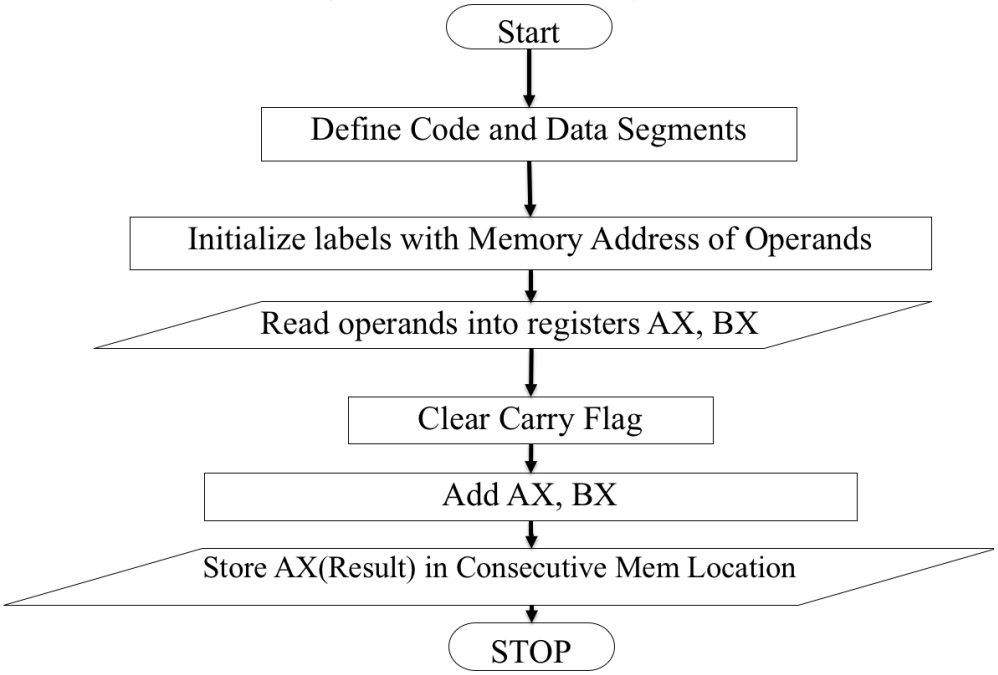
```
ASSUME CS:CODE,DS:DATA
DATA SEGMENT
    NO1 DB 08H
    NO2 DB 06H
DATA ENDS
CODE SEGMENT
START:
    MOV AX,DATA
    MOV DS,AX
    MOV AL,NO1
    MOV BL,NO2
    DIV BL
    INT 3H
CODE ENDS
END START
```

10. Square of 8-bit number

Write a program to find the square of an 8 bit number placed in 0500H and store the result in consecutive memory location.

```
ASSUME CS:CODE,DS:DATA
DATA SEGMENT
   OPR1 DW 0500H
   INZ DW 0000H
DATA ENDS
CODE SEGMENT
START:
   MOV AX,DATA
   MOV DS,AX
   MOV AX, INZ
   MOV BX, OPR1
   MOV AL, [BX]
   MUL AL
   INC BX
   MOV [BX], AL
CODE ENDS
END START
```

11. 16-bit addition




```
ASSUME CS: CODE, DS: DATA
DATA SEGMENT
    OPR1 DW 0500H
    OPR2 DW 0502H
    SUM DW 0504H
DATA ENDS
CODE SEGMENT
START:
    MOV AX, DATA
    MOV DS, AX
    MOV BX, OPR1
    MOV AX, [BX]
    CLC
    MOV. BX, OPR2
    ADD AX, [BX]
    MOV BX, SUM
    MOV [BX], AX
INT 03H
CODE ENDS
END START
```

12. 16-bit subtraction

```
ASSUME CS: CODE, DS: DATA
DATA SEGMENT
    OPR1 DW 0500H
    OPR2 DW 0502H
    SUM DW 0504H
DATA ENDS
CODE SEGMENT
START:
    MOV AX, DATA
    MOV DS, AX
    MOV BX, OPR1
    MOV AX, [BX]
    CLC
    MOV. BX, OPR2
    SUB AX, [BX]
    MOV BX, SUM
    MOV [BX], AX
INT 03H
CODE ENDS
END START
```

13. 16-bit multiplication

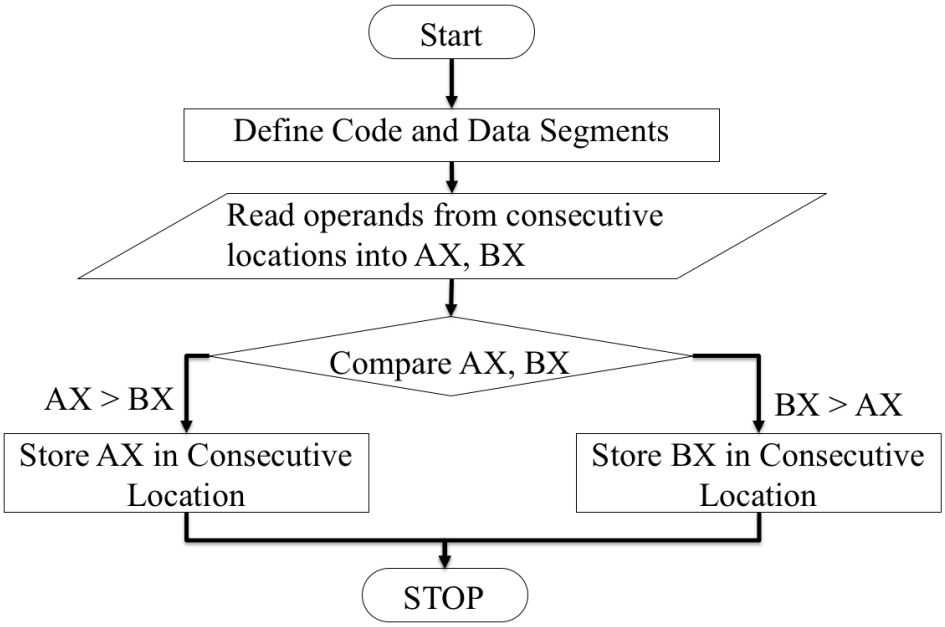
```
ASSUME CS: CODE, DS: DATA
DATA SEGMENT
    OPR1 DW 0500H
    OPR2 DW 0502H
    SUM DW 0504H
DATA ENDS
CODE SEGMENT
START:
    MOV AX, DATA
    MOV DS, AX
    MOV BX, OPR1
    MOV AX, [BX]
    CLC
    MOV. BX, OPR2
    MUL [BX]
    MOV BX, SUM
    MOV [BX], AX
INT 03H
CODE ENDS
END START
```

14. 16-bit division

```
ASSUME CS: CODE, DS: DATA
DATA SEGMENT
    OPR1 DW 0500H
    OPR2 DW 0502H
    SUM DW 0504H
DATA ENDS
CODE SEGMENT
START:
    MOV AX, DATA
    MOV DS, AX
    MOV BX, OPR1
    MOV AX, [BX]
    CLC
    MOV. BX, OPR2
    DIV [BX]
    MOV BX, SUM
    MOV [BX], AX
INT 03H
CODE ENDS
END START
```

15. Greatest of 2 numbers

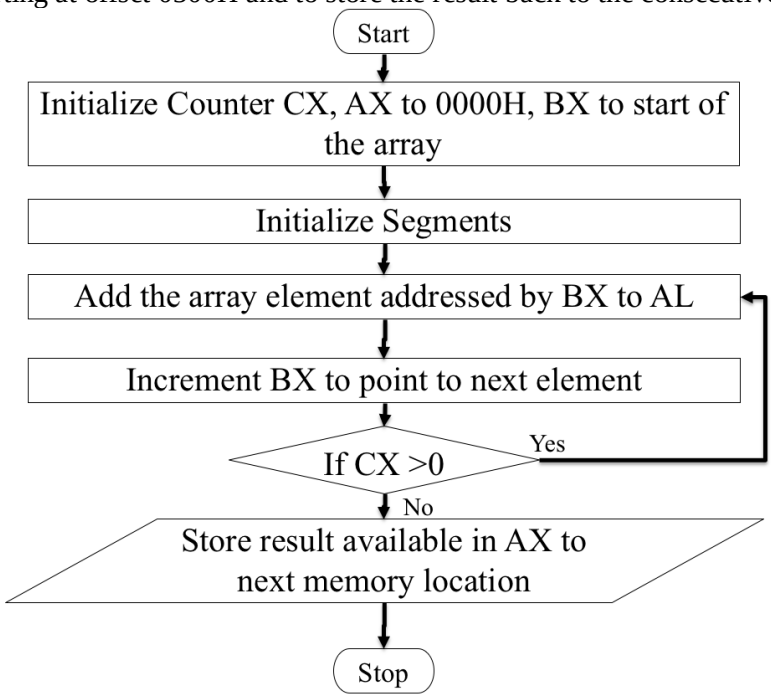
Write a program to read two numbers placed consecutively in Data segment at 0700H. Compare and find out the greater number. Place the greater number in the following memory location.




```
ASSUME CS: CODE, DS: DATA
DATA SEGMENT
   OPR1 DW 0700H
   OPR2 DW 0701H
   RES DW 0702H
DATA ENDS
CODE SEGMENT
ORG 0200H
START:
   MOV AX, DATA
   MOV DS, AX
   MOV BX, OPR1
   MOV AX,[BX]
   MOV CX, OPR2
   MOV BX,[CX]
   CMP AX, BX
   JG GREATER
   MOV CX, RES
   MOV [CX], BX
   JMP DONE
GREATER      : MOV CX, RES
         MOV [CX], AX
DONE  : INT 21
CODE ENDS
END START
```

16. Sum of array elements

Write a procedure to add every element in an array of ten 8-bit numbers placed in the data segment starting at offset 0500H and to store the result back to the consecutive memory location



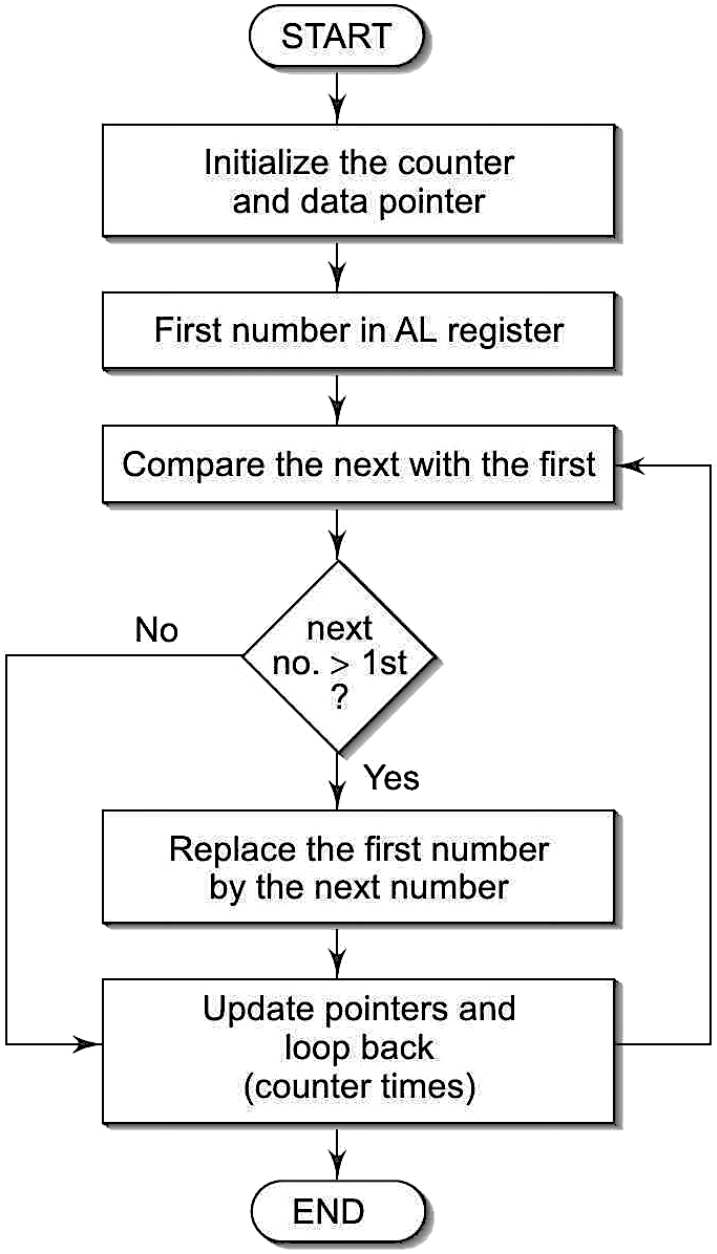

```
ASSUME CS: CODE, DS: DATA
DATA SEGMENT
    CNT DB 0AH
    ARR1 DW 0500H
    INZ DW 0000H
DATA ENDS
CODE SEGMENT
ORG 0200H
START:
    MOV AX, DATA
    MOV DS, AX
    MOV CX, CNT
    MOV AX, INZ
    MOV BX, ARR1
NEXT: ADD AL, [BX]
    INC BX
    LOOP NEXT
    MOV [BX], AL
CODE ENDS
END START
```

17. Increment all elements in an array

Write a procedure to increment every element in an array of 10 elements placed in the data segment starting at offset 0500H.

```
ASSUME CS: CODE, DS: DATA
DATA SEGMENT
    CNT DW 0005H
    ARR1 DW 0500H
DATA ENDS
CODE SEGMENT
ORG 0200H
START:
    MOV AX, DATA
    MOV DS, AX
    MOV CX, CNT
    MOV BX, ARR1
NEXT: MOV AX, [BX]
    INC AX
    MOV [BX], AX
    INC BX
    INC BX
    LOOP NEXT
CODE ENDS
END START
```

18. Largest in an array




```
ASSUME CS:CODE,DS:DATA
DATA SEGMENT
    CNT DW 000FH
    SRCADD DW 0500H
DATA ENDS
CODE SEGMENT
START:
    MOV AX,DATA
    MOV DS,AX
    MOV CX, CNT
    MOV BX, SRCADD
    MOV AL, [BX]
BACK: INC BX
    CMP AL, [BX]
```

```
    JNC NEXT
    MOV AL, [BX]
NEXT: LOOP BACK
    INC BX
    MOV [BX], AL
    INT 03H
CODE ENDS
END START
```

19. Count of all odd and even numbers in an array

Write a program to count the number of odd and even numbers in an an array of ten 8 bit number placed from 0500H.

```
ASSUME CS:CODE,DS:DATA
DATA SEGMENT
   ARRADD DW 0500H
   CNT DW 000AH
DATA ENDS
CODE SEGMENT
START:
   MOV AX,DATA
   MOV DS,AX
   MOV CX, CNT
   XOR BX, BX
   XOR DX, DX
   MOV SI, ARRADD
NEXT: MOV AX, [SI]
   ROR AX, O1H
   JC ODD
   INC BX
   JMP SKIP
ODD:  INC DX
SKIP:  INC SI
   LOOP NEXT
   INT 03H
CODE ENDS
END START
```

20. Count of all positive and negative numbers in an array

Write a program to find if an 8 bit number placed in 0500H is positive or negative. If the number is positive store 0 in the consecutive memory location else store 1.

```
ASSUME CS:CODE,DS:DATA
DATA SEGMENT
   OPR1 DW 0500H
   POS DB 00H
   NEG DB 01H
DATA ENDS
CODE SEGMENT
START:
   MOV AX,DATA
   MOV DS,AX
   MOV AX, INZ
   MOV BX, OPR1
   MOV AL, [BX]
   ROL AL, 1
   JC NEXT
   INC BX
   MOV [BX], POS
   JMP DONE
NEXT:INC BX
   MOV [BX], NEG
```

```
DONE: INT 03H
CODE ENDS
END START
```

21. 32-bit addition

Write a program to find the sum of two 32 bit numbers stored sequentially in the memory locations starting at offset 0500H in the segment

```
ASSUME CS:CODE,DS:DATA
DATA SEGMENT
    OPR1 DW 0500H
    OPR2 DW 0504H
    RES DW 0508H
    INZ DW 0000H
DATA ENDS
CODE SEGMENT
START:
    MOV AX,DATA
    MOV DS,AX
    MOV CX, INZ
    MOV BX, OPR1
    INC BX
    INC BX
    MOV AX, [BX]
    MOV BX, OPR2
    INC BX
    INC BX
    MOV DX, [BX]
    ADD AX, DX
    MOV BX, RES
    INC BX
    INC BX
    MOV [BX], AX
    MOV BX, OPR1
    MOV AX, [BX]
    MOV BX, OPR2
    MOV DX, [BX]
    ADC AX, DX
    MOV BX, RES
    MOV [BX], AX
    INT 03H
CODE ENDS
END START
```

22. 32-bit subtraction

```
ASSUME CS:CODE,DS:DATA
DATA SEGMENT
    OPR1 DW 0500H
    OPR2 DW 0504H
    RES DW 0508H
    INZ DW 0000H
DATA ENDS
CODE SEGMENT
START:
    MOV AX,DATA
    MOV DS,AX
    MOV CX, INZ
    MOV BX, OPR1
    INC BX
    INC BX
    MOV AX, [BX]
    MOV BX, OPR2
    INC BX
    INC BX
    MOV DX, [BX]
    SUB AX, DX
    MOV BX, RES
    INC BX
    INC BX
    MOV [BX], AX
    MOV BX, OPR1
    MOV AX, [BX]
    MOV BX, OPR2
    MOV DX, [BX]
    SBB AX, DX
    MOV BX, RES
    MOV [BX], AX
    INT 03H
CODE ENDS
END START
```

23. String Operations

Write a program to move a string of 4 bytes from data segment to extra segment.

```
ASSUME CS: CODE, DS: DATA, ES: EXTRA
DATA SEGMENT
        MOV CX, 04H
DATA ENDS
CODE SEGMENT
ORG 0200H
START:
        MOV AX, DATA
        MOV DS, AX
        MOV AX, EXTRA
        MOV ES, AX
        MOV SI, OFFSET SOURCE
        MOV DI, OFFSET DESTINATION
        CLD
        REP MOVSB
CODE ENDS
END START
```

24. Write a program for 10 word string comparison and store 1 in AX register if the strings are equal, else store 0

```
ASSUME CS: CODE, DS: DATA, ES: EXTRA
DATA SEGMENT
        MOV CX, 0AH
DATA ENDS
CODE SEGMENT
ORG 0200H
START:
        MOV AX, DATA
        MOV DS, AX
        MOV AX, EXTRA
        MOV ES, AX
        MOV SI, OFFSET STRING1
        MOV DI, OFFSET STRING2
        CLD
        REPE CMPSW
        CMP CX, 00H
        JZ EQUAL
        MOV AX, 00H
        JMP EOP
EQUAL:      MOV AX, 01H
EOP : INT 21
CODE ENDS
END START
```

25. Write a program to search for 1 word in a 10 word string and store the position of substring in AX register if found, else store 0

```
ASSUME CS: CODE, DS: DATA, ES: EXTRA
DATA SEGMENT
        MOV CX, 0AH
        MOV DX, 0AH
DATA ENDS
CODE SEGMENT
ORG 0200H
START:
        MOV AX, DATA
        MOV DS, AX
        MOV AX, EXTRA
        MOV ES, AX
        MOV SI, OFFSET SUB
        MOV DI, OFFSET STRING2
        MOV AX, [SI]
        CLD
        REPNE SCASW
        CMP CX, 00H
        JZ NOTFOUND
        SUB DX, CX
        MOV AX, DX
        JMP EOP
NOTFOUND :      MOV AX, 00H
EOP : INT 21
CODE ENDS
END START
```

26. Write a program to demonstrate data transfer on strings. Move a byte string 16 bytes long from offset 0200H to 0300H.

```
ASSUME CS:CODE,DS:DATA, ES: EXTRA
DATA SEGMENT
   CNT DW 0010
   SRC DW 0200H
   DEST DW 0300H
DATA ENDS
CODE SEGMENT
START:
   MOV AX,DATA
   MOV DS,AX
   MOV AX,EXTRA
   MOV DS,AX
   MOV CX, CNT
   MOV SI, SRC
   MOV DI, DEST
   MOV AX,ABC

NEXT: MOV AL, [SI]
   MOV [DI], AL
   INC SI
   INC DI
   LOOP NEXT
CODE ENDS
END START
```

```
ASSUME CS:CODE,DS:DATA, ES: EXTRA
DATA SEGMENT
   CNT DW 0010
   SRC DW 0200H
   DEST DW 0300H
DATA ENDS
CODE SEGMENT
START:
   MOV AX,DATA
   MOV DS,AX
   MOV AX,EXTRA
   MOV DS,AX
   MOV CX, CNT
   MOV SI, SRC
   MOV DI, DEST
   MOV AX,ABC
   CLD
   REP MOVSB
CODE ENDS
END START
```

27. Searching for an element in an array

Write a program to search for a given element in an array of ten 8 bit numbers placed from 0500H.

```
ASSUME CS:CODE,DS:DATA
DATA SEGMENT
   ARRADD DW 0500H
   CNT DW 000AH
   NO1 DB 02H
   NF DB 00H
   FND DB 01H
DATA ENDS
CODE SEGMENT
START:
   MOV AX,DATA
   MOV DS,AX
   MOV SI, ARRADD
        MOV CX, CNT
        MOV BL, NO1
NEXT:        MOV AL, [SI]
        CMP AL, BL
        JNE NOTFOUND
        MOV DX, FND
        JMP TOEND
NOTFOUND:  MOV DX, NF
        INC SI
        LOOP NEXT
TOEND:       INT 03H
CODE ENDS
END START
```

28. Sorting of array in ascending order

Write a program to sort an array of ten 8 bit number placed from 0500H.

```
ASSUME CS:CODE,DS:DATA
DATA SEGMENT
   ARRADD DW 0500H
   CNT DW 000AH
DATA ENDS
CODE SEGMENT
START:
   MOV AX,DATA
   MOV DS,AX
   MOV DX, CNT
   DEC DX
OUTER:MOV CX, DX
   MOV SI, ARRADD
INNER:MOV AL, [SI]
   CMP AL, [SI+1]
   JL NEXT
   XCHG AL, [SI+1]
NEXT: INC SI
   LOOP INNER
   DEC DX
   JNZ OUTER
   INT 03H
CODE ENDS
END START
```

Sorting of array in descending order

29. Factorial of a number

Write a program to find the factorial of an 8 bit number placed in 0500H and store the result in consecutive memory location.

```
ASSUME CS:CODE,DS:DATA
DATA SEGMENT
    OPR1 DW 0500H
    RES DW 0501H
    ABC DW 0000H
DATA ENDS
CODE SEGMENT
START:
    MOV AX,DATA
    MOV DS,AX
    MOV CX, INZ
    MOV BX, OPR1
    MOV CX, [BX]
    MOV AX,CX
    DEC CX
NEXT: MUL CX
    LOOP NEXT
    MOV BX, RES
    MOV [BX], AX
    INT 03H
CODE ENDS
END START
```

Arithmetic expression evaluation

30. Block transfer of data

Write a program to find the transfer a block of twenty 8-bit numbers placed from 0500H to 0600H.

```
ASSUME CS:CODE,DS:DATA
DATA SEGMENT
   OPR1 DW 0500H
   OPR2 DW 0600H
   CNT DW 0014H
DATA ENDS
CODE SEGMENT
START:
   MOV AX,DATA
   MOV DS,AX
   MOV CX, CNT
   MOV SI, OPR1
   MOV DI, OPR2
NEXT: MOV AX, [SI]
   MOV [DI], AX
   INC SI
   INC DI
   LOOP NEXT
   INT 03H
CODE ENDS
END START
```

COM LAB VIVA QUESTIONS

1.What is a Microprocessor?

Microprocessor is a CPU fabricated on a single chip, program-controlled device, which fetches the instructions from memory, decodes and executes the instructions.

2. What is Instruction Set?

It is the set of the instructions that the Microprocessor can execute.

3. What is Bandwidth ?

The number of bits processed by the processor in a single instruction.

4. What is Clock Speed ?

Clock speed is measured in the MHz and it determines that how many instructions a processor can processed. The speed of the microprocessor is measured in the MHz or GHz.

5. What are the features of Intel 8086 ?

Features:

- Released by Intel in 1978
- Produced from 1978 to 1990s
- A 16-bit microprocessor chip.
- Max. CPU clock rate:5 MHz to 10 MHz
- Instruction set: x86-16
- Package: 40 pin DIP
- 16-bit Arithmetic Logic Unit
- 16-bit data bus (8088 has 8-bit data bus)
- 20-bit address bus - 220 = 1,048,576 = 1 meg
- The address refers to a byte in memory.
- In the 8088, these bytes come in on the 8-bit data bus. In the 8086, bytes at even addresses come in on the low half of the data bus (bits 0-7) and bytes at odd addresses come in on the upper half of the data bus (bits 8-15).
- The 8086 can read a 16-bit word at an even address in one operation and at an odd address in two operations. The 8088 needs two operations in either case.
- The least significant byte of a word on an 8086 family microprocessor is at the lower address.

6. What is Logical Address:?

• A memory address on the 8086 consists of two numbers, usually written in hexadecimal and separated by a colon, representing the segment and the offset. This combination of segment and offset is referred to as a logical address

• Logical address=segment: offset

7. What is The Effective Address:

• In general, memory accesses take the form of the following example:

• Mov ax, [baseReg + indexReg + constant]

• This example copies a word sized value into the register AX.

• Combined, the three parameters in brackets determine what is called the effective address, which is simply the offset referenced by the instruction

8. What is Physical Address?

Physical memory address pointed by SEGMENT:OFFSET pair is calculated as:

Physical address = (<Segment Addr> * 10) + <Offset Addr>

9.What are the flags in 8086?

In 8086 Carry flag, Parity flag, Auxiliary carry flag, Zero flag, Overflow flag, Trace flag, Interrupt flag, Direction flag, and Sign flag.

10.Why crystal is a preferred clock source?

Because of high stability, large Q (Quality Factor) & the frequency that doesn't drift with aging. Crystal is used as a clock source most of the times.

11.What is Tri-state logic?

Three Logic Levels are used and they are High, Low, High impedance state. The high and low are normal logic levels & high impedance state is electrical open circuit conditions. Tri-state logic has a third line called enable line.

12.What happens when HLT instruction is executed in processor?

The Micro Processor enters into Halt-State and the buses are tri-stated.

13.What is Program counter?

Program counter holds the address of either the first byte of the next instruction to be fetched for execution or the address of the next byte of a multi byte instruction, which has not been completely fetched. In both the cases it gets incremented automatically one by one as the instruction bytes get fetched. Also Program register keeps the address of the next instruction.

14.What is 1st / 2nd / 3rd / 4th generation processor?

The processor made of PMOS / NMOS / HMOS / HCMOS technology is called 1st / 2nd / 3rd / 4th generation processor, and it is made up of 4 / 8 / 16 / 32 bits.

15.Name the processor lines of two major manufacturers?

High-end: Intel - Pentium (II, III, 4), AMD - Athlon. Low-end: Intel - Celeron, AMD - Duron. 64-

bit: Intel -Itanium 2, AMD - Opteron.

16.How many bit combinations are there in a byte?

Byte contains 8 combinations of bits.

17.Have you studied buses? What types?

There are three types of buses.

Address bus: This is used to carry the Address to the memory to fetch either Instruction or Data.

Data bus : This is used to carry the Data from the memory.

Control bus : This is used to carry the Control signals like RD/WR, Select etc.

18.What is the Maximum clock frequency in 8086?

5 Mhz is the Maximum clock frequency in 8086.

19.What is meant by Maskable interrupts?

An interrupt that can be turned off by the programmer is known as Maskable interrupt.

20.What is Non-Maskable interrupts?

An interrupt which can be never be turned off (ie. disabled) is known as Non-Maskable interrupt

21.What are the different functional units in 8086?

Bus Interface Unit and Execution unit, are the two different functional units in 8086.

22.What are the various segment registers in 8086?

Code, Data, Stack, Extra Segment registers in 8086.

23.What does EU do?

Execution Unit receives program instruction codes and data from BIU, executes these instructions and store the result in general registers.

24.Which Stack is used in 8086? k is used in 8086?

FIFO (First In First Out) stack is used in 8086.In this type of Stack the first stored information is retrieved first.

25.What are the flags in 8086?

In 8086 Carry flag, Parity flag, Auxiliary carry flag, Zero flag, Overflow flag, Trace flag, Interrupt flag, Direction flag, and Sign flag.

26.What is SIM and RIM instructions?

SIM is Set Interrupt Mask. Used to mask the hardware interrupts.

RIM is Read Interrupt Mask. Used to check whether the interrupt is Masked or not.

27.What are the different types of Addressing Modes?

A:- There are 12 different types of Addressing Modes. They are:-

<1> Immediate:-The Immediate data is a part of instruction, and appears in the form of successive bytes.

<2> Direct:-A 16-bit memory address(offset) is directly specified in the instruction as a part of it.

<3> Register:-Data is stored in a register and it is referred using the particular register (except IP).

<4> Register Indirect:-The address of the memory location which contains data or operand is determined in an indirect way.

<5> Indexed:-offset of the operand is stored in one of the index registers.

<6> Register Relative:-The data is available at an effective address formed by adding an 8-bit or 16-bit displacement with the content of any one of the registers BX,BP,SI and DI in the default (either DS or ES) segment.

<7> Based Indexed:-The effective address of the data is formed, in this addressing mode, by adding content of a base register to the content of an index register.

<8> Relative Based Indexed:- The effective address is formed by adding an 8 or 16-bit displacement with the sum of contents of any one of the base registers and any one of the index registers, in the default segment.

<9> Intrasegment Direct Mode:-In this mode, the address to which the control is to bve transferred lies in the segment in which the control transfer instruction lies and appears directly in the instruction as an immediate displacement value.

<10> Intrasegment Indirect Mode:-In this mode, the displacement to which the control is to be transferred, is in the same segment in which the control transfer instruction lies, but it is passed to the instruction indirectly.

<11> Intersegment Direct:-In this mode,the address to which the control is to be transferred is in a different segment.

<12> Intersegment Indirect:-In this mode,the address to which the control is to be transferred lies in a different segment and it is passed to the instruction indirectly sequentially.

28.What are the General Data Registers & their uses?

A:- The Registers AX,BX,CX,DX are the general Purpose 16-bit registers.AX register as 16-bit accumulator. BX register is used as an offset Storage.CX register is used as default or implied counter. Dx register is used as an implicit operand or destination in case of a few instructions.

29.What are Segment Registers & their uses?

A:-There are 4 Segment Registers Code Segment(CS),Data Segment(DS),Extra Segment(ES) & Stack Segment(SS) registers.CS is used for addressing memory locationin code.DS is used to points the data. ES refers to a segment which is essentially in another data segment.SS is used fopr

addressing stack segment of memory.

30.What are Flag registers?

A:-Divided into 2 parts:-Condition code or status flags and machine control flags.

S-Sign Flag:-Is to set when the result of any computation is negative.

Z-Zero Flag:-Is to set if the result of the computation or comparision performed by the previous instruction is zero.

C-Carry Flag:-Is set when there is carry out of MSB in case of addition or a borrow in case of subtraction.

T-Trap Flag:-Is set,the processor enters the single step execution mode.

I-Interrupt Flag:-Is set,the maskable interrupts are recognised by the CPU.

D-Direction Flag:-Is set for autoincrementing or autodecrementing mode in string manipulation instructions.

AC-Auxiliary Carry Flag:-Is set if there is a carry from the lowest nibble during addition or borrow for the lowest nibble.

O-Overflow Flag:-Is setif the result of a signed operation is large enough to be accommodated in a destination register.

31.What does the 8086 Architecture contain?

A:-The complete architecture of 8086 can be divided into 2types :-Bus Interface Unit(BIU) & Execution Unit.

The BIU contains the circuit for physical address calculations and a precoding instruction byte queue & it makes the bus signals available for external interfacing of the devices.

The EU contains the register set of 8086 except segment registers and IP.It has a 16-bit ALU,able to perform arithmetic and Logic operations.

32) What are Data Copy/Transfer Instructions?

A:- Mov

Push

Pop

Xchg

In

Out

Xlat

Lea

Lds/Les

Lahf

Sahf

Pushf

Popf

33. What are Machine Control Instructions?

A:- Nop

Hlt

Wait

Lock

34) What are Flag Manipulation Instructions?

A:- Cld

Std

Cli

Sti

35) What are String Instructions?

A:- Rep

MovSB/MovSW

Cmps

Scas

Lods

Stos

36) What are different parts for 8086 architecture?

A:- The complete architecture of 8086 can be divided into 2types :-Bus Interface Unit(BIU) & Execution Unit.

The BIU contains the circuit for physical address calculations and a precoding instruction byte queue & it makes the bus signals available for external interfacing of the devices.

The EU contains the register set of 8086 except segment registers and IP.It has a 16-bit ALU,able to perform arithmetic and Logic operations.

37.What is an Interrupts

Def:- An interrupt operation suspends execution of a program so that the system can take special action.The interrupt routine executes and normally returns control to the interrupted procedure, which then resumes execution.BIOS handles Int 00H-1FH, whereas DOS handles INT 20H-3FH.

38. What is an Opcode?

A:-The part of the instruction that specifies the operation to be performed is called the Operation code or Op code.

39.What is an Operand?

A:-The data on which the operation is to be performed is called as an Operand.

40.Explain the difference between a JMP and CALL instruction?

A:-A JMP instruction permantely changes the program counter.

A CALL instruction leaves information on the stack so that the original program execution sequence can be resumed.

41.What is meant by Polling?

A:- Polling or device Polling is a process which idenfies the device that has interrupted the microprocessor.

42.What is meant by Interrupt?

A:-Interrupt is an external signal that causes a microprocessor to jump to a specific subroutine.

43.What is an Instruction?

A:-An instruction is a binary pattern enetered through an input device to command the microprocessor to perform that specific function.

44.What is Microcontroller and Microcomputer?

A:- Microcontroller is a device that includes microprocessor:memory and I/O signal lines on a single chip,fabricated using VLSI technology.

Microcomputer is a computer that is designed using microprocessor as its CPU.It includes microprocessor,memory and I/O.

45.What is Assembler?

A:-The assembler translates the assembly language program text which is given as input to the assembler to their binary equivalents known as object code.

The time required to translate the assembly code to object code is called access time.The assembler checks for syntax errors&displays them before giving the object code.

46.Define Variable?

A:-A Variable is an identifier that is associated with the first byte of data item.

47.Explain Dup?

A:-The DUP directive can be used to initialize several location & to assign values to these locations.

48.Define Pipelining?

A:-In 8086,to speedup the execution program,the instructions fetching and execution of instructions are overlapped each other.this is known as Pipelining.

49.What is the use of HLDA?

A:-HLDA is the acknowledgment signal for HOLD. It indicates whether the HOLD signal is received or not. HOLD and HLDA are used as the control signals for DMA operations.

50. Explain about "LEA"?

A:-LEA(Load Effective Address) is used for initializing a register with an offset address.

A common use for LEA is to intialize an offset in BX, DI or SI for indexing an address in memory.

An equivalent operation to LEA is MOV with the OFFSET operator, which generates slightly shorter machine code.

51. Difference between "Shift" and "Rotate".

A:-Shift and Rotate commands are used to convert a number to another form where some bits are shifted or rotated.

A rotate instruction is a closed loop instruction. That is, the data moved out at one end is put back in at the other end.

The shift instruction loses the data that is moved out of the last bit locations.

Basic difference between shift and rotate is shift command makes "fall of " bits at the end of the register.

Where rotate command makes "wrap around" at the end of the register.

52. Explain about .MODEL SMALL?

A:- .MODEL directive:-This simplified segment directive creates default segments and the required

ASSUME and GROUP statements.

Its format is .MODEL memory-model. The following are the memory models

Tiny:-Code and data in one segment, for .COM programs.

Small:-Code in one segment (<=64K), data in one segment (<=64K). It generates 16-bit offset addresses.

Medium:-Any number of code segments, data in one segment (<=64K).

Compact:-Code in one segment (<=64K), any number of data segments. It generates 32-bit addresses, which require more time for execution.

Large:-Code and data both in any number of segments, no array >64K.

Huge:-Code and data both in any number of segments, arrays may be >64K.

Flat:-Defines one area upto 4 gigabytes for both code and data. It is unsegmented. The program uses 32- bit addressing and runs under Windows in protected mode.

53.Difference between JMP and JNC?

A:-JMP is Unconditional Branch.

JNC is Conditional Branch.

54.List the String Manipulation Commands?

A:-REP=Repeat.

MOVS=Move Byte/Word

CMPS=Compare Byte/Word

SCAS=Scan Byte/Word

LODS=Load byte/Wd to AL/AX

STOS=Store Byte/Wd from AL/A

55.What are the 4 Segments?

A:-Code Segment Register {CS}

Data Segment Register {DS}

Extra Segment Register {ES}

Stack Segment Register{SS}

56.What is the main use of ready pin?

A:-READY is used by the microprocessor to check whether a peripheral is ready to accept or transfer data.

A peripheral may be a LCD display or analog to digital converter or any other.

These peripherals are connected to microprocessor using the READY pin.

If READY is high then the periphery is ready for data transfer. If not the microprocessor waits until READY goes high.

57.Explain about Direction Flag?

A:-This is used by string manipulation instructions.

If this flag bit is 0 , the string is processed beginning from the lowest to the highest address, i.e.,. Autoincrement mode.

Otherwise, the string is processed from the highest towards the lowest address, i.e.,. Autodecrementing mode.

58.What are the basic units of a microprocessor ?

The basic units or blocks of a microprocessor are ALU, an array of registers and control unit.

59.what is Software and Hardware?

The Software is a set of instructions or commands needed for

performing a specific task by a programmable device or a computing machine.

The Hardware refers to the components or devices used to form

computing machine in which the software can be run and tested. Without software the Hardware is an idle machine.

60.What is assembly language?

The language in which the mnemonics (short -hand form of instructions) are used to write a program is called assembly language. The manufacturers of microprocessor give the mnemonics.

61.What are machine language and assembly language programs?

The software developed using 1's and 0's are called machine language, programs. The software developed using mnemonics are called assembly language programs.

62. What is the drawback in machine language and assembly language, programs?

The machine language and assembly language programs are machine dependent. The programs developed using these languages for a particular machine cannot be directly run on another machine.

63. Define bit, byte and word.

A digit of the binary number or code is called bit. Also, the bit is the fundamental storage unit of computer memory.

The 8-bit (8-digit) binary number or code is called byte and 16-bit binary number or code is called word. (Some microprocessor manufactures refer the basic data size

operated by the processor as word).

64. What is a bus?

Bus is a group of conducting lines that carries data, address and control signals.

65. Why data bus is bi-directional?

The microprocessor has to fetch (read) the data from memory or input device for processing and after processing, it has to store (write) the data to memory or output device.

Hence the data bus is bi-directional.

66. Why address bus is unidirectional?

The address is an identification number used by the microprocessor to identify or access a memory location or I / O device. It is an output signal from the processor. Hence the address bus is unidirectional.

67. What is the function of microprocessor in a system?

The microprocessor is the master in the system, which controls all the activity of the system. It issues address and control signals and fetches the instruction and data from memory. Then it executes the instruction to take appropriate action.

68. What are the modes in which 8086 can operate?

The 8086 can operate in two modes and they are minimum (or uniprocessor) mode and maximum (or multiprocessor) mode.

69. What is the data and address size in 8086?

The 8086 can operate on either 8-bit or 16-bit data. The 8086 uses 20 bit address to access memory and 16-bit address to access 1/0 devices.