MICROPROCESSORS AND MICROCONTROLLERS LAB

EC262/R22

LAB MANUAL

PREPARED BY

Mrs. R.LOGANYA, Assistant Professor in E.C.E

Mrs.S.BHARGAVI, Assistant Professor in E.C.E

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

BAPATLA WOMEN'S ENGINEERING COLLEGE::BAPATLA

(Affiliated to Acharya Nagarjuna University, Guntur)

Bapatla, Pin:522101

COLLEGE VISION:

BWEC has been established with forethought to organize the growing needs of higher technical education for WOMEN, to provide them a competitive edge and a unique niche in the society. We at BWEC are determined to create a conducive environment with updated technological inputs and latest information to facilitate the students with a pleasant feel in acquiring knowledge and skills required to make them globally competitive technocrats.

COLLEGE MISSION:

Our mission is to impart the quality education on par with global standards to the students from all over India, and particularly those from local and rural areas. We continuously try to maintain high standards to make them technologically competent and ethically strong individuals who shall be able to improve the quality of life and economy of our country

Department of Electronics and Communication Engineering

VISION

To produce highly professional and socially responsible women engineers through world class quality technical education and research to meet industry and Society needs in the field of Electronics and Communication Engineering.

MISSION

M1- To inculcate quality education for women through innovative learning practices with the use of modern tools.

M2- To impart professional ethics and moral values through co-curricular and extra-curricular activities as individual and team member.

M3- Empowering graduates through industrial trainings/on-job training in collaboration with industries with-a-view to life-long learning.

Programme Educational Objectives (PEOs):

PEO1: To develop and implementation of Electronics and Communication systems by imparting technical knowledge, professional leadership and communication skills as an individual and team in a multi-disciplinary stream.

PEO2: To train Graduates in solving complex problems, analyzing, designing and imparting research-based knowledge by using modern scientific tools to develop least cost-effective and innovative products.

PEO3: To produce women graduates through life-long learning, who ensure ethical and moral behaviour.

Program Specific Outcomes (PSOs):

PSO1: To understand and analyze basic concepts of electronics and communication in various areas like signal processing, VLSI, embedded systems, communications, digital and analog devices.

PSO2: Ability to design and develop computing systems using concepts of mathematics, electronics engineering and other related inter disciplines to asses and resolve societal, environmental, health safety and cultural issues with professional ethics.

Programme Outcomes(POs):

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

PO2- Problem analysis: Identify, formulate, review research literature, and analyse complex engineering problems related to Electronics & Communication Engineering andreaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences

PO3- Design /development of solutions: Design solutions for complex engineering problems related to Electronics & Communication Engineering 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 knowledgeand research methods including design of experiments, analysis and interpretation of data, andsynthesisoftheinformationtoprovidevalidconclusions.

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

PO6 - 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 Electronic s& Communication Engineering professional engineering practice.

PO7- Environmentandsustainability: Understand the impact of the Electronics &Communication Engineering professional engineering solutions in societal and environmental contexts ,and demonstrate he 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 withthe engineering community and with society at large, such as, being able to comprehend andwrite effective reports and design documentation, make effective presentations, and give andreceive clearinstructions.

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 lifelong learning in the broadest context of technological change.

Laboratory Rules

General Rules in Laboratories:

1. You are expected to arrive on time and not depart before the end of a your laboratory.
2. You must not enter a lab unless you have permission from a technician or lecturer.
3. You are expected to comply with instructions, written or oral, that the laboratory Instructor gives you during the laboratory session.
4. You should behave in an orderly fashion always in the lab.
5. You must not stand on the stools or benches in the laboratory.
6. Keep the work bench tidy and do not place coats and bags on the benches.
7. You must ensure that at the end of the laboratory session all equipment used is stored away where you found it.
8. You must put all rubbish such as paper outside in the corridor bins. Broken components should be returned to the lab technician for safe disposal.
9. You must not remove test equipment, test leads or power cables from any lab without permission.
10. Eating and drinking in the laboratories are forbidden.

11. The use of mobile phones during laboratory sessions is forbidden.

Specific Safety Rules for Laboratories:

1. You must not damage or tamper with the equipment or leads.
2. You should inspect laboratory equipment for visible damage before using it. If there isa problem with a piece of equipment, report the technician or lecturer. DONOT returnittoequipmenttoastoragearea.
3. You should not work on circuits where the supply voltage exceeds 40 volts without very specific approval from your lab supervisor. If you need to work on such circuits, you should contact your supervisor for approval and instruction on how to do this safely before commencing the work.
4. Always use an appropriate stand for holding your soldering iron.
5. Turn off your soldering iron if it is un likely to be used for more than 10 Minutes.
6. Never leave a hot soldering iron unattended.
7. Never touch a soldering iron element or bitunless theironhasbeendisconnectedfromthe mains and has had adequate time to cool down.
8. Neverstrip insulationfromawirewithyourteethoraknife,alwaysuseanappropriatewire strippingtool.
9. Shield wire with your hands when cutting it with a pliers to prevent bits of wire flyingaboutthebench.

CourseObjectives:

The course should enable the students to:

1:Introduce the programming and interfacing techniques of 8086 microprocessor.

2:Analyze the basic concepts and programming of 8051microcontroller

Course Outcomes:

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

CO No.	Course Outcome Statement	Taxonomy Level
C261.1	Develop the assemly language Programs on Data Transfer Instructions, Arithmetic and Logical Instructions, Branch Instructions, Subroutines and Sorting of an Array	Analyze
C261.2	Develop the assemly language Programs using DOS and BIOS Interrupts	Analyze
C261.3	Generate Waveforms by using DAC Interface to 8086 Microprocessor	Analyze
C261.4	Develop the assembly language Programs on Data Transfer Instructions and Arithmetic and Logical Instructions using 8051 Microcontroller.	Analyze

Mapping of course outcomes with PO's And PSO's:

High -3 Medium -2 Low-1, '-':No correlation

PO / CO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO 10	PO 11	PO 12	PSO1	PSO2
C261.1	3	3	2	2	2	-	-	-	1	-	-	1	2	2
C261.2	3	3	2	2	2	-	-	-	1	-	-	1	2	-
C261.3	3	3	2	1	1	-	-	-	1	-	-	1	2	-
C261.4	3	3	2	2	2	-	-	-	1	-	-	1	2	2
C261 Avg.	3	3	2	1.83	1.83	-	-	-	1	-	-	1	2	2

MICROPROCESSORS AND MICROCONTROLLERS LAB EC261/R22

Experiments Based On ALP (8086)

1.Programs on Data Transfer Instructions

2.Programs on Arithmetic and Logical Instructions

3.Programs on Branch Instructions

4. Programs on Subroutines

5. Sorting of an array

6. Programs on interrupts (Software and Hardware)

7. 8086 programs using DOS and BIOS interrupts

Experiments based on Interfacing and Microcontroller (8051)

8. DAC interface—Waveform generations

9. Stepper motor control

10. Key board interface/LCD interface

11. Data transfer between two PCs using RS-232C serial port

12. Programs on Data transfer instructions using 8051 Microcontroller

13. Programs on Arithmetic and Logical instructions using 8051 Microcontroller

14. Applications with Microcontroller 8051

Experiment beyond the syllabus

1. Program for digital clock design using 8086

8086 MICROPROCESSOR

ASSEMBLING AND EXECUTING THE PROGRAM USING TASM ASSEMBLER

Writing an ALP:

- **A**ssembly level programs generally abbreviated as ALP are written in text editor EDIT
- Type EDIT in front of command prompt to open an untitled text file--- EDIT<file name>
- After typing the program save the file with appropriate file name with an extension .ASM

Assembling an ALP:

- To assemble an ALP we needed executable file called TASM.EXE, only if this file is in current working Directory. We can assemble the program, the command is TASM <filename>
- If the program is free from all syntactical errors, this command will give the OBJECT file, in case of errors it list out the number of errors, warnings and kind of error.

NOTE: No object file is created until all errors are rectified

Linking:

- After successful assembling of the program ,we have to link it to get Executable file, the command used is TLINK<filename>
- This command results in <filename.EXE>which can be executed in front of the command prompt

Executing the program:

- Open the program in debugger by the command (note only .exe files can be open) by the command

  TD <filename>

- This will open the program in debugger screen where in you can view the assemble code with the **CS** and **IP** values at the left most side and the machine code. Register content ,memory content is also viewed using **VIEW** option of debugger.
- **Execute** option in the menu can be used to execute the program either in **single step mode(F7)** or

  burst execution mode(F5)

ADDITION OF TWO -16 BIT NUMBERS (SOFT WARE)

```
CODE SEGMENT
ASSUME CS:CODE,DS:DATA
START:MOV AX,DATA
MOV DS,AX
MOV AL,N1
MOV BL,N2
ADD AX,BX
MOV N3,AX
HLT
CODE ENDS
DATA SEGMENT
N1 DW 0202H
N2 DW 0305H
N3 DW ?
DATA ENDS
END START
```

RESULT:

INPUT:

N1=0202

N2=0305

OUTPUT:

N3=0507

AX=0507 BX=0305

ADDITION OF TWO -16 BIT NUMBERS AND CHECK WHETHER THERE IS CARRY OR NOT (SOFT WARE)

```
CODE SEGMENT
ASSUME CS:CODE,DS:DATA
START:MOV AX,DATA
MOV DS,AX
MOV AL,N1
MOV BL,N2
ADD AX,BX
MOV N3,AX
JC  AHEAD
MOV CX,0000H
JMP  L1
AHEAD: MOV CX,0FFFFH
L1:MOV  N4,CX
HLT
CODE ENDS
DATA SEGMENT
N1  DW 0202H
N2  DW 0305H
N3  DW ?
N4  DW  ?
DATA ENDS
END START
```

RESULT:

INPUT:

N1=0202

N2=0305

OUTPUT:

N3=0507H

N4=0000H

AX=0507 BX=0305

SUBTRACTION OF TWO -16 BIT NUMBERS (SOFT WARE)

```
CODE SEGMENT
ASSUME CS:CODE,DS:DATA
START:MOV AX,DATA
MOV DS,AX
MOV AX,N1
MOV BX,N2
SUB AX,BX
MOV N3,AX
JC  AHEAD
MOV CX,0000H
JMP  L1
AHEAD: MOV CX,0FFFFH
L1:MOV  N4,CX
HLT
CODE ENDS
DATA SEGMENT
N1 DW0505H
N2 DW0305H
N3  DW?
```

```
N4  DW  ?
DATA ENDS
END START
```

RESULT:

INPUT:

N1=0505

N2=0305

OUTPUT:

N3=0200H

N4=0000H

AX=0200 BX=0305

SUBTRACTION OF TWO -16 BIT NUMBERS AND CHECK WHETHER IS BORROW OR NOT (SOFT WARE)

```
CODE SEGMENT
ASSUME CS:CODE,DS:DATA
START:MOV AX,DATA
MOV DS,AX
MOV AX,N1
MOV BX,N2
SUB AX,BX
MOV N3,AX
HLT
CODE ENDS
DATA SEGMENT
N1 DW 0505H
```

```
N2 DW 0305H
N3 ?
DATA ENDS
END START
```

RESULT:

INPUT:	**OUTPUT:**
N1=0505	N3=0200
N2=0305	AX=0200 BX=0305

MULTIPLICATION OF TWOUNSIGNED -16 BIT NUMBERS (SOFT WARE)

```
CODE SEGMENT
ASSUME CS:CODE,DS:DATA
START:MOV AX,DATA
MOV DS,AX
XOR DX,DX
MOV AX,N1
MOV BX,N2
MUL BX
MOV N3,AX
MOV N4,DX
HLT
CODE ENDS
DATA SEGMENT
N1 DW 0505H
N2 DW 0404H
N3 DW ?
```

```
N4 DW ?
DATA ENDS
END START
```

RESULT:

INPUT:

N1=0505

N2=0404

OUTPUT:

N3=2814

N4=0014

AX=0200 BX=0404 DX=0014

MULTIPLICATION OF TWO SIGNED -16 BIT NUMBERS (SOFT WARE)

```
CODE SEGMENT
ASSUME CS:CODE,DS:DATA
START:MOV AX,DATA
MOV DS,AX
XOR DX,DX
MOV AX,N1
MOV BX,N2
IMUL  BX
MOV N3,AX
MOV N4,DX
HLT
CODE ENDS
DATA SEGMENT
N1 DW 0505H
```

```
N2 DW -0404H
N3 DW ?
N4 DW ?
DATA ENDS
END START
```

RESULT:

INPUT:

N1=0505

N2=0404

OUTPUT:

N3=2814

N4=0014

AX=0200 BX=0404 DX=0014

DIVISION OF TWO UNSIGNED- 16 BIT NUMBERS (SOFT WARE)

```
CODE SEGMENT
ASSME CS:CODE,DS:DATA
START:MOV  AX,DATA
MOV DS,AX
XOR AX,AX
XOR DX,DX
MOV AX,N1
MOV BX,N2
DIV BX
MOV N3,AX
MOV N4,DX
HLT
```

```
CODE ENDS
DATA SEGMENT
N1 DW 0505H
N2 DW 0404H
N3 DW ?
N4 DW ?
DATA ENDS
END START
```

RESULT:

INPUT:

N1=0505

N2=0404

OUTPUT:

N3=0001 AX=0001 BX=0404

N4=0101 AX=0001 BX=0404

DIVISION OF TWO SIGNED- 16 BIT NUMBERS (SOFT WARE)

```
CODE SEGMENT
ASSUME CS:CODE,DS:DATA
START:MOV  AX,DATA
MOV DS,AX
XOR AX,AX
XOR DX,DX
MOV AX,N1
MOV BX,N2
IDIV BX
MOV N3,AX
```

```
MOV N4,DX
HLT
CODE ENDS
DATA SEGMENT
N1 DW 0505H
N2 DW -0404H
N3 DW ?
N4 DW ?
DATA ENDS
END START
```

RESULT:

INPUT:

N1=0505

N2=0404

OUTPUT:

N3=0001 AX=0001 BX=0404

N4=0101 CX=0000 DX=0101

MULTI BYTE ADDITION:

```
CODE SEGMENT
ASSUME CS:CODE, DS:DATA
START:MOV AX,DATA
MOV DS,AX
LEA SI,ARRAY1
LEA BX,ARRAY2
LEA DI,ARRAY3
MOV CL,COUNT
```

```
BACK: MOV AL,[SI]
      ADC AL,[BX]
      MOV [DI],AL
      INC SI
      INC BX
      INC DI
      DEC CL
      JNZ  BACK
      HLT
      CODE SEGMENT
      COUNT EQU 04H
      ARRAY1 DQ 06060606H
      ARRAY2 DQ 07070707H
      ARRAY3 DQ ?
 DATA ENDS
 END START
```

RESULT:

IN PUT: COUNT 04

ARRAY1 :06060606H

ARRAY2 :07070707H

OUTPUT:ARRAY3 :0D0D0D0D

MULTI BYTE SUBTRACTION:

```
CODE SEGMENT
ASSUME CS:CODE, DS:DATA
        START:MOV AX,DATA
        MOV  DS,AX
        LEA SI,ARRAY1
        LEA BX,ARRAY2
        LEA DI,ARRAY3
        MOV CL,COUNT
   BACK:MOV AL,[SI]
        SBB AL,[BX]
        INC SI
        INC BX
        INC DI
        DEC CL
        JNZ  BACK
        HLT
        CODE ENDS
        DATA SEGMENT
        COUNT EQN 04H
        ARRAY1 DQ 08080808H
        ARRAY2 DQ 07070707H
        ARRAY3 DQ ?
        DATA ENDS
        END START
```

RESULT:

INPUT:COUNT 04H

ARRAY1 : 08080808H

ARRAY2 : 07070707H

OUT PUT:

ARRAY3 : 01010101H

MULTI BYTE MULTIPLICATION:

```
CODE SEGMENT
ASSUME CS:CODE, DS:DATA
START:MOV AX,DATA
MOV DS,AX
MOV DX,0000
LEA SI,ARRAY1
LEA DI,ARRAY2
MOV CL,COUNT
MOV BL,N1
BACK:MOV AL,[SI]
   CBW
   MUL BL
   ADD DX,AX
   MOV DL,AH
   MOV [DI],AL
   INC SI
   INC DI
   INC CL
   JNZ BACK
```

```
    HLT
    CODE ENDS
    DATA SEGMENT
    COUNT EQN 04H
    N1 DB  02H
    ARRAY1 DQ  05050505H
    ARRAY2  DQ  ?
DATA ENDS
END START
```

RESULT:

INPUT:COUNT :04

N1 :02H

ARRAY1 :05050505H

OUTPUT:

ARRAY2 :0A0A0A0A

MULTI BYTE DIVISION:

```
CODE SEGMENT
 ASSUME  CS:CODE, DS:DATA
 START:MOV AX,DATA
 MOV DS,AX
 LEA SI,ARRAY1
 LEA DI,ARRAY2
 MOV DX,0000
 MOV AX,0000
 MOV CL,COUNT
 MOV BL,N
```

```
BACK:MOV AL,[SI+3]
DIV BL
  MOV [DI+3],AL
  DEC SI
  DEC DI
  DEC CL
  JNZ BACK
  HLT
  CODE  ENDS
  DATA SEGMENT
  COUNT EQN  04H
  N1  DB 02H
  ARRAY1  DQ  04040404H
  ARRAY2  DQ  ?
  DATA  ENDS
  END START
```

RESULT:

INPUT:COUNT :04H

N1 :02H

ARRAY1 : 04040404H

OUTPUT:

ARRAY2 :02020202H

ARITHMETIC OPERATIONS VIVA-VOCE QUESTIONS :-

1.Define the word microprocessor ?

2.What are the assembler directives used in assembly language program ?

3.What is the purpose of Movax,data , Movds,ax instruction purpose ?

4.What is the word size of 8086 ALU ?

5.What are the flags used in 8086 Microprocessor ?

6.What is the purpose of CS & IP ?

7.What is meant by multi byte ?

8.What is the difference between ADD & ADC instructions ?

9.What is the difference between SUB & SBB instructions ?

10.What is the default register used in multiplication & division ?

11.What are the flags effected in arithmetic operations ?

12.What is the purpose of DAA instruction?

13.What is the purpose of DAS Instruction ?

14.What is the another instruction used for LEA ?

15.write an example of immediate addressing mode ?

16.what is the operation of the instruction JNZ label ?

17.which data cannot be the operands of instructions INC & DEC ?

18.what is the use of index register ?

19.when OF &CF both will set ?

20.what are the general data registers ?

21.where are the most & least significant word of the result stored ?

22.what is meant by IMUL ?

23.which flags are undefined after IMUL ?

24.what are the implicant operands?

25.When an interrupt is generated ?

26.What does AX & DX contain during DIV instruction ?

27.How the flags are affected after executing AAD ?

28.By which addressing mode the division must be specified ?

29.what is meant by IDIV ?

30.How the flags are affected after IDIV instruction ?

31.Which operation to be done before signed division ?

32.what is meant by trap flag ?

33. How many segments,the 8086 memory is divided ?

34.How many addressing lines 8086 processor is having ?

35.what is the memory size of 8086 processor ?

36. How many data lines 8086 processor is having ?

37.what are extra flags used in 8086 rather than 8085 processor ?

38.what is the need of segmentation in 8086 processor ?

ASCII ADDITION (SOFT WARE) :

```
CODE SEGMENT
ASSUME  CS:CODE, DS:DATA
START: MOV AX,DATA
        MOV DS,AX
        MOV AX,N1
        MOV BX,N2
        ADD CX,DX
        AAA
        MOV N3,AX
        ADD AX,BX
        HLT
        CODE ENDS
        DATA SEGMENT
        N1 DW ‘03’
        N2 DW ‘04’
        N3 DW ?
        DATA ENDS
        END START
```

RESULT:

INPUT:

N1=03

N2=04

OUT PUT:

N3=07

ASCII SUBTRACTION (SOFT WARE):

```
CODE SEGMENT
ASSUME CS:CODE, DS:DATA
START: MOV AX,DATA
       MOV DS,AX
       MOV AX,N1
       MOV BX,N2
       SUB AX,BX
       AAS
       MOV N3,B2
       HLT
       CODE ENDS
       DATA SEGMENT
       N1 DW ‘05’
       N2 DW ‘03’
       N3 DW  ?
       DATA ENDS
       END START
```

RESULT:

INPUT:

N1=05

N2=03

OUT PUT:

N3=02

ASCII MULTIPLICATION (SOFT WARE) :

```
CODE SEGMENT
ASSUME  CS:CODE, DS :DATA
START:  MOV AX,DATA
        MOV DS,AX
        MOV AX,N1
        MOV BX,N2
        AND BX,0030H
        AND AX,0030H
        MUL BX
        AAM
    MOV N3,AX
        HLT
        CODE ENDS
    DATA SEGMENT
            N1 DW '03'
            N2 DW '02'
        N3 DW ?
            DATA ENDS
        END START
```

RESULT:

INPUT:

N1 = 03

N2 = 02

OUT PUT:

N3 = 06

ASCII DIVISION (SOFT WARE):

```
CODE SEGMENT
ASSUME  CS:CODE, DS:DATA
START: MOV AX,DATA
      MOV DX,AX
      MOV AX,N1
      MOV BX,N2
    AND  BX,0030H
    AND  AX,0030H
      DIV BX
          AAD
MOV  N3,AX
      HLT
      CODE ENDS
      DATA SEGMENT
      N1 DW ‘08’
      N2 DW ‘04’
      N3 DW ?
      DATA ENDS
      END START
```

RESULT:

INPUT:

N1 04

N2 02

OUT PUT: N3 02

ASCII-OPERATIONS VIVA-VOCE QUESTIONS :-

1.How many bit character code ASCII is ?

2.What is the purpose of AAA ?

3.What is the purpose of AAS ?

4.What is the difference between BCD & ASCII numbers ?

5.Which type of BCD is used in ASCII ?

6.What is the purpose of AAM ?

7.What is the purpose of AAD ?

8.What are the flags effected in ASCII operation ?

9.What are the default registers used in multiplication and division ?

10.What is meant by AAM?

11.When the sign flag will set ?

12.How the flags are affected after executing AAD ?

13.what is meant by AAA ?

14.what is meant by AAD ?

15.How the accumulator is affected when lower nibble of AL is greater than 09 ?

16.When AC flag is set ?

17.To which category does the instruction ADD ,AAA belongs to?

18.What does w-bit indicates?

19.What is the syntax of direct addressing mode ?

20.What does the instruction DAS indicates ?

21.Which operand cannot be destination operand in SUB instruction ?

CONVERSION OF 'BCD' TO 'ASCII' (SOFTWARE)

```
CODE SEGMENT
ASSUME CS:CODE,DS:DATA
START:MOV AX,DATA
MOV DS,AX
MOV AL,N1
MOV BL,AL
MOV CL,04H
AND AL,0FH
ADD AL,30H
MOV RES1,AL
ROR BL,CL
AND BL,0FH
ADD BL,30H
MOV RES2,BL
HLT
CODE ENDS
DATA SEGMENT
N1 DB 65H
RES1 DB ?
RES2 DB ?
DATA ENDS
END START
```

RESULT:

INPUT:

N1: 65H

OUTPUT: RES1:35HRES2:36 H

CONVERSION OF PACKED BCD TO UNPACKED BCD (SOFT WARE)

```
CODE SEGMENT
ASSUME CS:CODE,DS:DATA
START:MOV AX,DATA
MOV DS,AX
MOV AL,N1
MOV BL,AL
MOV CL,04H
AND AL,0FH
MOV RES1,AL
ROR BL,CL
AND BL,0FH
MOV RES2,BL
HLT
CODE ENDS
DATA SEGMENT
N1 DB 65H
RES1 DB ?
RES2 DB ?
DATA ENDS
END START
```

RESULT:

INPUT:

N1: 65H

OUTPUT:

RES1:05 RES2:06

TO COUNT POSITIVE AND NEGATIVE NUMBERS

```
CODE SEGMENT
ASSUME CS:CODE,DS:DATA
START:MOV AX,DATA
MOV DS,AX
LEA SI,ARRAY
MOV CL,COUNT
XOR  BX,BX
BACK: MOV AL,[SI]
RCL  AL,01
JC  AHEAD
INC  BL
JMP  L1
AHEAD:  INC  BH
L1: INC  SI
DEC  CL
JNZ  BACK
MOV  POSITIVE, BL
MOV NEGATIVE,BH
HLT
CODE  ENDS
DATA  SEGMENT
ARRAY  DB  89H,02H,9AH,34H,5FH
```

```
COUNT   EQU  05H
POSITIVE  DB  ?
NEGATIVE  DB  ?
DATA  ENDS
END  START
```

RESULT: INPUT : ARRAY---- 89H,02H,9AH,34H,5FH

OUTPUT: POSITIVE-----03

NEGATIVE----02

TO COUNT EVEN AND ODD NUMBERS

```
CODE SEGMENT
ASSUME CS:CODE,DS:DATA
START:MOV AX,DATA
MOV DS,AX
LEA SI,ARRAY
MOV CL,COUNT
XOR  BX,BX
BACK: MOV AL,[SI]
RCR  AL,01
JC  AHEAD
INC  BL
JMP  L1
AHEAD:  INC  BH
L1: INC  SI
DEC  CL
JNZ  BACK
```

```
MOV  POSITIVE, BL
MOV NEGATIVE,BH
HLT
CODE  ENDS
DATA  SEGMENT
ARRAY  DB  89H,02H,9AH,34H,5FH
COUNT   EQU  05H
EVENNUM  DB  ?
ODDNUM  DB  ?
DATA  ENDS
END  START
```

RESULT: INPUT : ARRAY---- 89H,02H,9AH,34H,5FH

OUTPUT: EVENNUM-----03

ODDNUM------02

LOGICAL OPERATIONS VIVA-VOCE QUESTIONS :-

1. what are the different types of logical instructions used in 8086 ?

2. what is meant by packed BCD ?

3 . What is meant by unpacked BCD ?

4 .what is the purpose of ROR instruction ?

5 .what is the difference between ROR& RCR instructions ?

6 .what is the logical instruction used for double of the given number ?

7 .what is the difference between arithmetic shift & logical shift instructions ?

8 .what is the logical instruction used to refresh the register ?

9 .what is the difference between TEST& AND instructions ?

10. what are the flags effected in logical operations ?

11 .what does shift & rotate instruction involve ?

12 .what is meant by ROL ?

13 .what is the counter used in case of shift & rotate instruction ?

14. which mode is not allowed in case of shift instruction ?

15. what are the unpacked BCD arithmetic instruction ?

16 which instruction are called as packed BCD arithmetic instruction ?

17. which flags are left unchanged by rotate operation ?

18. To which category does the instruction ROR belongs to ?

19. what are the unpacked BCD arithmetic instruction ?

20. which instruction are called as packed BCD arithmetic instruction ?

21. which flags are left unchanged by rotate operation ?

22. To which category does the instruction ROR belongs to ?

STRING COMPARISION (SOFT WARE)

```
CODE SEGMENT
ASSUME CS:CODE,DS:DATA,ES:DATA
START:MOV AX,DATA
MOV DS,AX
MOV ES,AX
MOV CX,AX
LEA SI,ARRAY1
LEA DI,ARRAY2
MOV CX,STR_LENGTH
CLD
REP CMPSB
JNE  AHEAD
MOV AX,0000H
MOV RES,AX
JMP NEXT
AHEAD:  MOV AX,0FFFFH
        MOV RES,AX
NEXT: HLT
CODE ENDS
DATA SEGMENT
ARRAY1 DB ‘MICRO’
ARRAY2 DB ‘PROCESSOR’
RES DW ?
STR_LENGTH  EQU ($-ARRAY1)
DATA ENDS
```

END START

RESULT:

INPUT:

(1) ARRAY1: 'MICRO'

ARRAY2:'PROCESSOR'

(2)ARRAY1:'HALL'

ARRAY2:'HALL'

OUTPUT:

RES: (1) 0FFFFH

(2)0000H

STRING REVERSE (SOFT WARE)

```
CODE SEGMENT
ASSUME CS:CODE,DS:DATA,ES:DATA
START:MOV AX,DATA
MOV DS,AX
MOV ES,AX
MOV CL,STR_LENGTH
MOV AL,[SI]
ROR AL,CL
XCHG [DI+STR_LENGTH],AL
HLT
CODE ENDS
DATA SEGMENT
```

```
N1 DQ  “INTEL”
STR_LENGTH  EQU ($-N1)
DATA ENDS
END START
```

RESULT:

INPUT:

N1: “INTEL”

OUTPUT:

LETNI

MOVING OF STRINGS

```
CODE SEGMENT
ASSUME CS:CODE,DS:DATA,ES:DATA
START:MOV AX,DATA
MOV DS,AX
MOV ES,AX
MOV CX,AX
LEA SI,ARRAY1
LEA DI,ARRAY2
MOV CX,STR_LENGTH
CLD
REP MOVSB
CODE ENDS
DATA SEGMENT
ARRAY1 DB ‘MICRO’
STR_LENGTH  EQU ($-ARRAY1)
```

```
ARRAY2 DB?
DATA ENDS
END START
```

RESULT:

INPUT ARRAY:MICRO

OUTPUT ARRAY:MICRO

SORTING:

ASCENDING ORDER (SOFT WARE)

```
CODE SEGMENT
ASSUME CS:CODE,DS:DATA
START:MOV AX,DATA
MOV DS,AX
MOV DX,COUNT-1
AGAIN0: MOV CX,DX
        MOV SI,OFFSET ARRAY
AGAIN1:  MOV AL,[SI]
   CMP AL,[SI+1]
        JL  BACK
      XCHG [SI+1],AL
   XCHG [SI],AL
BACK: ADD SI,01
        LOOP AGAIN1
   DEC DX
   JNZ AGAIN0
       HLT
```

```
CODE ENDS
DATA SEGMENT
ARRAY DB 12H,42H,52H,32H
COUNT EQU 04H
DATA  ENDS
END START
```

RESULT:

INPUT:

ARRAY: 12H,42H,52H,32H

OUTPUT:

12H,32H,42H,52H

DESCENDING ORDER (SOFT WARE)

```
CODE SEGMENT
ASSUME CS:CODE,DS:DATA
START:MOV AX,DATA
MOV DS,AX
MOV DX,COUNT-1
AGAIN0: MOV CX,DX
        MOV SI,OFFSET ARRAY
AGAIN1:  MOV AL,[SI]
        CMP AL,[SI+1]
JG  BACK
        XCHG [SI+1],AL
       XCHG [SI],AL
```

```
BACK: ADD SI,01
        LOOP AGAIN1
   DEC DX
   JNZ AGAIN0
       HLT
CODE ENDS
DATA SEGMENT
ARRAY DB 12H,42H,52H,32H
COUNT EQU 04H
DATA  ENDS
END START
```

RESULT:

INPUT:

ARRAY: 12H,42H,52H,32H

OUTPUT:

52H, 42H,32H,12H

STRING OPERATIONS VIVA-VOCE QUESTIONS

1. what is meant by “string” ?
2. what are the assembler directives used in strings ?
3. what is the necessity of ES in strings ?
4. what is the purpose of direction flags in strings ?

5.what is the difference between CMP &CMPS ?

6. what is the difference between MOV & MOVS ?

7. what are the flags effected in strings ?

8. what is the instructions used to check whether the byte is present or not ?

9. what is the purpose of XCHG instruction in Reverse string ?

10. How many number of counters used in sorting & what is the purpose of that counters ?

11. what is the another instructions used rather than ADD SI,01 ?

12. what is the branch instruction used to sort the given aarray in ascending order ?

13. what is the branch instruction used to sort the given array in descending order ?

14. what are the internal instructions used in LOOP instruction?

15. what is the purpose of LOOP instruction?

16.What is the purpose of REP instruction prefix ?

17.what is the purpose of SI & DI registers ?

.

PROGRAM TO USE SOFTWARE AND HARDWARE INTERRUPTS FOR RECEIVING A INPUT FROM KEY BOARD AND DISPLAY IT ON SCREEN

```
DATA SEGMENT
INKEY DB ?
BUF DB 20 DUP (0)
MES DB 10,13,' STUDY OF MICROPROCESSORS IS INTERSTING' 10,13 ' $'
DATA ENDS
CODE SEGMENT
ASSUME CS:CODE,DS:DATA
START: MOV AX,DATA
MOV DS,AX
MOV AH,01H
INT 21H
MOV INKEY ,AL
MOV BUF,10
MOV AH,0AH
LEA DX,BUF
INT 21H
MOV AH,06H
MOV DL,'A'
INT 21H
MOV AH,09H
LEA DX,MES
INT 21H
MOV AH,4CH
INT 21H
CODE ENDS
```

```
END START
```

RESULT:

INPUT :-- A

OUTPUT :-- A

DOS /BIOS PROGRAMMING (SOFTWARE)

```
DATA SEGMENT
Cr EQU 0Ah
Lf EQU 0d H
MESG1 DB Cr,Lf ,'TYPE ANY KEY :$'
MESG2 DB Cr,Lf ,'TYPED KEY IS :$'
CHAR DB ?
DATA ENDS
CODE SEGMENT
ASSUME CS:CODE,DS:DATA
START: MOV AX,DATA
MOV DS,AX
MOV AH,09H
LEA DS,MESG1
INT 21H
MOV AH,01H
INT 21H
MOV CHAR,AL
MOV AH,09H
LEA DS,MESG2
INT 21H
```

```
MOV AH,4CH
HLT
CODE ENDS
END START
```

RESULT:

INPUT :

TYPE ANY KEY : M

OUTPUT:

TYPED KEY IS : M

DOS\BIOS PROGRAMMING VIVA-VOCE QUESTIONS :-

1.What is meant by DOS ?

2.What is the purpose of Cr and Lf ?

3.What is the purpose of INT 21H ?

4.What is the instruction used to read a character ?

5.what is the instruction used to display the input ?

6. what is the instruction used to display the output ?

7. what is the instruction used to return to DOS prompt ?

8. what is the purpose of Mov ah,4ch instruction ?

9.what are the flags effected in DOS Programming ?

10 what is the interrupt we have to use for DOS operation ?

FACTORIAL OF A GIVEN "N" NUMBER USING RECURSIVE PROCEDURE:

```
STACK_SEG  SEGMENT STACK
DW 200 DUP(0)
STACK_TOP LABEL WORD
STACK_SEG ENDS
NUMBER EQU 08
CODE SEGMENT
ASSUME CS:CODE,SS:STACK_SEG
START:MOV AX,STACK_SEG
MOV SS,AX
MOV SP,OFFSET STACK_TOP
SUB SP,0004H
MOV AX,NUMBER
PUSH AX
CALL FACTO
ADD SP,2
POP AX
POP DX
NOP
JMP FORWARD
FACTO PROC NEAR
PUSHF
PUSH AX
PUSH DX
PUSH BP
MOV BP,SP
MOV AX,[BP+10]
```

```
CMP AX,0001H
JNE GO_ON
MOV WORD PTR [BP+12],0001H
MOV WORD PTR [BP+14],0000H
JMP EXIT
GO_ON: SUB SP,0004H
DEC AX
PUSH AX
CALL FACTO
MOV BP,SP
MOV AX,[BP+2]
MUL WORD PTR [BP+16]
MOV [BP+18],AX
MOV [BP+20],DX
ADD SP,0006H
EXIT: POP BP
POP DX
POP AX
POPF
RET
FACTO ENDP
FORWARD: NOP
CODE ENDS
END START
```

RESULT:

INPUT: GIVEN NUMBER : (1)08 (2)04

OUTPUT :(1)9D80H (2)18

MODULAR PROGRAMMING VIVA-VOCE QUESTIONS :-

1 what is meant by a procedure ?

2 what is meant by a recursive procedure ?

3 How we can call the procedure into main program ?

4 what is the difference b\w NEAR &FAR procedure ?

5 what are the changes occur in main program when we call the procedure ?

6 what is the purpose of RET instruction ?

7 what is the difference between RET& IRET instruction ?

8 what is the status of flag register in calling the procedure ?

9 what are the segment registers used in procedure programs ?

10 what is the purpose of INT 03h ?

11 what is the another instruction used other than INT 03 ?

12 what is the purpose of DUP assembler directive ?

13 what is the logic behind of recursive procedure ?

14 what is the purpose of PUSH & POP instructions ?

15.what is the difference between procedure and macro ?

16.How many number of interrupts are used in 8086 processor?

INTERFACING THROUGH 8086

Working procedure for interfacing with 8086 microprocessor trainer kit:

1.Connect the interfacing module of required one with 8086 trainer kit by using 50 core FRC cable.

2. Switch on the trainer kit

3. Type A and then press enter key

4. Enter the starting address of assembler

5. Enter the program step by step and note down the addresses and op-codes simultaneously

6. At the end of program press reset key on trainer kit

7. To run the program give the command is GO starting address of assembler

8. Then display shows Executing.....

9. Then observe the corresponding interfacing module output on the kits

INTERFACING D/A CONVERTER TO 8086 TO GENERATE

SQUARE WAVE , TRIANGULAR WAVE, SAWTOOTH WAVE AND SINE WAVE

GENERATION OF SQUARE WAVE PROGRAM

```
L1:MOV AL,00
OUT C8,AL
CALL DELAY
MOV AL,0FF
OUT C8,AL
CALL DELAY
JMP L1
DELAY: MOV CX,05FF
BACK: LOOP BACK
RET
```

GENERATION OF SAWTOOTHWAVE PROGRAM

```
L1: MOV AL,00
BACK: OUT C8,AL
INC AL
JNZ BACK
JMP L1
```

GENERATION OF SINEWAVE PROGRAM

```
L1: MOV BX,1100
MOV CL,46
BACK:MOV AL,[BX]
OUT C0,AL
INC BX
LOOP BACK
JMP L1
```

1100:7F,8A,95,A0

A4 B5 BF C8

D1 D9 E0 E7

ED F2 F7 FA

FC FE FF FE

FC FA F7 F2

ED E7 E0 D9

D1 C8 BF B5

AA A0 95 8A

7F 74 69 5F

53 49 3F 36

2D 25 1D 17

10 0B 07 04

01 00 01 04
07 0B 10 17

1D 25 2D 36

3F 49 53 5F

69 74

INTERFACE 8251 TO SEND DATA SERIALLY TO ANOTHER SERIAL PORT

Connect Two Serial Ports With Rs-232c Cable.First make the receiver end ready and then execute the program at the transmitter end

TRASMITTER END:

```
ORG 1000
MOV AL,36
OUT CE,AL
MOV AL,10
OUT C8,AL
MOV AL,00
OUT C8,AL
MOV AL,4E
OUT C2,AL
MOV AL,37
OUT C2,AL
LOOP1: IN AL,C2
AND AL,04
JZ LOOP1
MOV AL,41
OUT C0,AL
INT 2
```

***Here 41 is the transmitted data

RECEIVER END:

```
ORG 1200
MOV AL,36
OUT CE,AL
MOV AL,10
OUT C8,AL
```

```
MOV AL,00
OUT C8,AL
MOV AL,4E
OUT C2,AL
MOV AL,37
OUT C2,AL
LOOP1: IN AL,C2
AND AL,02
JZ LOOP1
IN AL,C0
MOV BX,1500
MOV [BX],AL
INT 2
```

***Here 1500H is the destination address

(i) MATRIX KEY BOARD INTERFACE:

```
DATA SEGMENT
PORT A EQU 120H
PORT C EQU 122H
CWRD EQU 123H
ARRAY DB '0123456789.+-*/%ACK=MMMM'
DATA ENDS
CODE SEGMENT
ASSUME CS:CODE,DS:DATA
START:MOV AX,DATA
MOV DS,AX
MOV AL,90
MOV DX,CWRD
OUT DX,AL
REPEAT:MOV DX,PORT C
MOV AL,01
OUT DX,AL
MOV DX,PORT A
IN AL,DX
CMP AL,00
JZ NEXT
JMP FIRSTROW
NEXT:MOV DX,PORT C
MOV AL,02
OUT DX,AL
MOV DX,PORT A
IN AL,DX
```

```
CMP AL,00
JNZ SECONDROW
MOV AL,04
MOV DX,PORT C
OUT DX,AL
MOV DX,PORTA
IN AL,DX
CMP AL,00
JNZ THIRDROW
JMP REPAT
FIRSTROW: CALL DELAY
LEA SI,ARRAY
UP:SHR AL,01
JC DISPLAY
INC SI
JMP UP
JMP DISPLAY
SECONDROW:CALL DELAY
LEA SI,ARRAY+08
UP1:SHR AL,01
JC DISPLAY
JMP UP1
THIRDROW:CALL DELAY
LEA SI,ARRAY+10
UP2:SHR AL,01
JC DISPLAY
INC SI
```

```
JMP UP2
JMP DISPLAY
DISPLAY:MOV DL,[SI]
CMP DL,97
JZ EXIT
MOV AH,02
INT 21H
JMP REPEAT
DELAY:MOV BX,0FFFFH
L1:MOV CX,0FFFH
L2:DEC CX
JNZ L2
DEC BX
JNZ L1
RET
EXIT :MOV AH,4CH
INT 21H
CODE ENDS
END START
```

(ii) SEVEN SEGMENT DISPLAY INTERFACE:

```
DATA SEGMET
PORT A EQU 120H
PORT B EQU 121H
PORT C EQU 122H
CWRD EQU 123H
TABLE DB 8CH,0C7H,86H,89H
DATA ENDS
CODE SEGMENT
ASSUME CS:CODE,DS:DATA
START:MOV AX,DATA
MOV DS,AX
MOV AL,80      ---------  Initialize port b and port c of 8255 as o/p
MOV DX,CWRD
OUT DX,AL
MOV BH,04      --------BH=no.of digits to be displayed
LEA SI,TABLE  -------Starting address of look-up table
NEXTDIGIT:MOV CL,08        -------CL= no. of segments
MOV AL,[SI]
NEXTBIT;ROL AL,01
MOV CH,AL
MOV DX,PORT B
OUT DX,AL
MOV AL,01
MOV DX,PORT C
OUT DX,AL
DEC AL
MOV DX,PORT C
```

```
OUT DX,AL
MOV AL,CH   ----------Get the 7-segment code back into AL
DEC CL
JNZ NEXTBIT
DEC BH
INC SI
JNZ NEXTDIGIT
MOV AH,4CH
INT 21H
CODE ENDS
END START
```

STEPPER MOTOR INTERFACING :

```
Data segment
Port aequ 120h
Port b equ 121h
Port c equ 122h
Cwrdequ 123h
Data ends
Code segment
Assume cs:code,ds:data
Start: movax,data
Movds,ax
Mov al,80
Movdx,cwrd
Oudx,al
Movdx,port a
Mov al,88
Out dx,al
Up:call delay
Rol al,01
Out dx,al
Jmp up
Delay: mov cx,0ffff
Up2:mov bx,0ff
Up1:decbx
Jnz up1
Dec cx
Jnz up2
```

```
Ret
Mov ah,4ch
Int 21h
Code ends
End start
```

INTERFACING THROUGH 8086 VIVA-VOCE QUESTIONS :-

1.what is the need of Interfacing ?

2. what is the purpose of FRC cable ?

3. what is the purpose of RS232-C Cable ?

4.what is the IC chip used for Keyboard display ?

5. what are the IC chips used for D\A and A\D Converters(8-bit &12-bit) ?

6.what is the logic behind of keyboard interfacing ?

7. what is the logic behind of Traffic lights interfacing ?

8. what is the logic behind of D\A Converter interfacing ?

9. what is the logic behind of Stepper motor interfacing ?

10.what is difference between normal motor and stepper motor working ?

MICROCONTROLLER

8051

Working procedure for 8051 microcontroller trainer kit:

1. Switch on the trainer kit
2. Type A and then press enter key
3. Enter the starting address of assembler
4. Enter the program step by step and note down the addresses and op-codes simultaneously
5. At the end of program press reset key on trainer kit
6. To run the program give the command is GO starting address of assembler
7. Then display shows Executing.....
8. Then press reset key and then type R and press enter key , shows the 8051 all registers status
9. To view the particular memory location give the command is SD address then press enter key

PROGRAMS ON DATA TRANSFER INSTRUCTIONS (HARDWARE) :

COPY A BLOCK OF DATA FROM ONE LOCATION TO ANOTHER IN INTERNAL MEMORY

ADDRESS	LABEL	MNEMONICS	OPERANDS	OPCODES
8000		MOV	R0,#35	
		MOV	R1,#60	
		MOV	R3,#06	
	L1:	MOV	A,@R0	
		MOV	@R1,A	
		INC	R0	
		INC	R1	
		DJNZ	R3,L1	
	L2:	SJMP	L2	

RESULT:

I/P: **IR 35 : 00 01 02 03 04 05**

O/P: **IR 60: 00 01 02 03 04 05**

TRANSFER A BYTE OF DATA FROM INTERNAL MEMORY TO EXTERNAL MEMORY

ADDRESS	LABEL	MNEMONICS	OPERANDS	OPCODES
8000		MOV	R0,#6E	
		MOV	DPTR,#9000	
		MOV	A,R0	
		MOVX	@DPTR,A	
	L1:	SJMP	L1	

RESULT:

I/P: R0: 6E

O/P: SD 9000: 6E

GENERATION OF NUMBERS FROM "00 " TO "0F"

ADDRESS	LABEL	MNEMONICS	OPERANDS	OPCODES

```
8000        MOV     A,#00
            MOV     DPTR,#9000
            MOVX    @DPTR,A
            INC     DPTR
            CJNZ    A,#10H
L1:         SJMP    L1
```

RESULT:

O/P: 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

PROGRAMS ON ARITHMETIC AND LOGICAL INSTRUCTIONS:

ADDITION OF TWO NUMBERS :

ADDRESS	LABEL	MNEMONICS	OPERANDS	OPCODE
8000		MOV	DPTR,#9000	90 90 00
8003		MOVX	A,@DPTR	EO
8004		MOV	0F0,A	F5 F0
8006		MOV	DPTR,#9001	90 90 01
8009		MOVX	A,@DPTR	E0
800A		ADD	A,0F0	25 F0
800C		MOV	DPTR,#9002	90 90 02
800F		MOVX	@DPTR,A	F0
8010	L1:	SJMP	L1	8D F3

RESULT:

INPUT:

SD 9000 05

9001 03

OUTPUT:

G 8000

SD 9002 08

SUBTRACTION OF TWO NUMBERS:

ADDRESS	LABEL	MNEMONICS	OPERANDS	OPCODE

8000		MOV	DPTR,#9000	90 90 00
8003		MOVX	A,@DPTR	EO
8004		MOV	0F0,A	F5 F0
8006		MOV	DPTR,#9001	90 90 01
8009		MOVX	A,@DPTR	E0
800A		SUBB	A,0F0	25 F0
800C		MOV	DPTR,#9002	90 90 02
800F		MOVX	@DPTR,A	F0
8010	L1:	SJMP	L1	8D F3

RESULT:

INPUT:

SD 9000 05

9002 03

OUTPUT:

G 8000

SD 9002 02

MULTIPLICATION OF TWO NUMBERS :

ADDRESS	**LABEL**	**MNEMONICS**	**OPERANDS**	**OPCODE**
8000		MOV	DPTR,#9001	90 90 01

8003	MOVX A,@DPTR		E0	
8004	MOV	0F0,A		F5 F0
8006	MOV	DPTR,#9000		90 90 00
8009	MOVX A,@DPTR		E0	
800A	MOV	B,A		F5 0B
800C	MUL	AB		A4
800D	MOV	DPTR,#9002		90 90 02
8010	MOVX @DPTR,A		F0	
8012	MOV	A,0F0		E5 F0
8014	MOVX @DPTR,A		F0	
8015	LJMP	00		02 00 00

RESULT:

INPUT:

SD 9000 02

9001 02

OUTPUT:

G 8000

SD 9002 04

DIVISION OF TWO NUMBERS :

ADDRESS LABEL MNEMONICS OPERANDS OPCODES

8000	MOV	DPTR,#9001	90 90 01
8003	MOVX	A,@DPTR	E0
8004	MOV	0F0,A	F5 F0
8006	MOV	DPTR,#9000	90 90 00
8009	MOVX	A,@DPTR	E0
800A	MOV	B,A	F5 0B
800C	DIV	AB	84
800D	MOV	DPTR,#9002	90 90 02
8010	MOVX	@DPTR,A	F0
8011	INC	DPTR	A3
8012	MOV	A,0F0	E5 F0
8014	MOVX	@DPTR,A	F0
8015	LJMP	00	02 00 00

RESULT:

INPUT:

SD 9000 02

9001 04

OUTPUT:

G 8000

SD 9002 02

9003 00

HEX TO ASCII CONVERSION :

ADDRESS LABELMNEMONICS OPERANDS OPCODES

8000	MOV	DPTR,#9000	90 90 00
8003	MOVX	A,@DPTR	E0
8004	INC	DPTR	A3
8005	MOV	R1,A	F9
8006	CLR	C	C3
8007	SUBB	A,#0A	94 09
8009	JC	AHEAD	40 06
800B	MOV	A,R1	E9
800C	ADD	A,#37	24 37
800E	BACK: MOVX	@DPTR,A	F0
800F	L1:SJMP	L1	80 FE
8011	AHEAD: MOV	A,R1	E9
8012	CLR	C	C3
8013	ADD	A,#30	24 30
8015	LJMP	BACK	02 80 0E

RESULT:

INPUT:

SD 9000(i) 05 (ii) 0A

OUTPUT:

G 8000

SD 9001 (I) 35 (II) 41

CHECKING THE 5TH BIT POSITION:

ADDRESSLABEL MNEMONICS OPERANDS OPCODES

8000		MOV	DPTR,#8900	90 90 00
8003		MOVX	A,@DPTR	E0
8004		INC	DPTR	A3
8005		RLC	A	33
8006		RLC	A	33
8007		RLC	A	33
8008		JC	AHEAD	40 06
800A		MOV	A,#0FF	74 FF
800C		MOVX	@DPTR,A	F0
800D		LJMP	0030	02 00 63
8010	AHEAD:	MOV	A,#00	74 00
8012		MOVX	@DPTR,A	F0
8013		LJMP	00	02 00 00

RESULT:

INPUT:

1)SD 8900 05

2)SD 8900 25

OUTPUT:

G 8000

1)SD 8901 FF

2)SD 8901 00

TO FIND THE LARGEST AMONG "N" NUMBERS:

ADDRESS	LABEL	MNEMONICS	OPERANDS	OPCODES

8000	START:	MOV	DPTR,#8900		90 89 00
8003		MOVX	A,@DPTR		E0
8004		MOV	R2,A		FA
8005		MOV	DPTR,#8901		90 89 01
8008		MOVX	A,@DPTR		E0
8009		DEC	R2		1A
800A		MOV	R1,A		F9
800B	L2:	INC	DPTR	A3	
800C		MOVX	A,@DPTR	E0	
800D		MOV	R3,A		FB
800E		SUBB	A,R1		99
800F		JNC	L1		05 18
8011	BACK1:	DJNZ	R2,L2		0A F8
8013		MOV	A,R1		E9
8014		INC	DPTR		A3
8015		MOVX	@DPTR,A		F0
8016	BACK:	SJMP	BACK		80 FE
8018	L1:	MOV	A,R3		EB
8019		MOV	R1,A		F9
801A		LJMP	BACK1		02 80 11
801C		LJMP	START		02 80 00

RESULT:

INPUT:

SD 8900 05 (COUNT)

8901 01
8902 04

8903 07
8904 06
8905 02

OUTPUT:

G 8000

SD 8906 07

EXCHANGE CONTENT OF ACCUMULATOR WITH R0 REGISTER CONTENT

ADDRESS	**LABEL**	**MNEMONICS**	**OPERANDS**	**OPCODES**
8000		**MOV**	**A,#C5**	
		SWAP	**A**	
		MOV	**R0,#C6**	

	XCH	**A,R0**
L1:	**SJMP**	**L1**

RESULT:

INPUT: A:5C

R0:C6

OUTPUT: A:C6

R0:5C

REVERSE AND XOR OF A GIVEN NUMBER

ADDRESS	**LABEL**	**MNEMONICS**	**OPERANDS**	**OPCODES**
8000		**MOV**	**A,#54**	
		MOV	**R0,A**	
		SWAP	**A**	

```
        XCH     A,R0
L1:     SJMP    L1
```

RESULT:

INPUT : A:54H

R0:54H

OUTPUT: A:54H

R0:45H

TO CLEAR 16 RAM LOCATIONS OF INTERNAL MEMORY

ADDRESS	LABEL	MNEMONICS	OPERANDS	OPCODES
8000		CLR	A	
		MOV	R1,#60H	
		MOV	R7,#16H	

```
L1:  MOV   @R1,A
     INC   R1
     DJNZ  R7,L1
L2:  SJMP  L2
```

RESULT:

OUTPUT: IR 60

ADDRESS	60	61	62	63	64	65	
DATA	00	00	00	00	00	00	

APPLICATIONS WITH MICROCONTROLLER 8051

(I) READING AND WRITING ON PARALLEL PORTS OF 8051 MC

PROGRAM ON BYTE TRANSFERS:

```
MOV A,P0        ---Read A Set Of Switches From P0
MOV P1,A        ---Write Them To Set Of Leds On P1
INC P1    ---Increment Value OfLeds On P1
```

PROGRAM ON BIT MANIPULATION:

PROGRAM: P1 has 4 LEDS to bits 0-3,P3.4(T0), P3.5 (T1) have switches attached, by pressing T0 LED is ON,T1-LED is OFF

```
SETB  T0
SETB  T1
LP0: MOV  P1,#0
LP1: JNB  T0,LP1
MOV  P1,#0F
LP2: JNB  T1,LP2
SJMP  LP0
```

(II) TIMER IN DIFFERENT MODES OF OPERATION OF 8051MC

(A) To generate a delay of 100msec using an external frequency of 25KHz, timer in mode 0 of operation.

```
CLR TCON
MOV TMOD,#04          Initialize T0
MOV TL0,#06    Start  Pre-scalar  At  0000
MOV TH0,#0B2 Load Up count  In TH0
MOV TCON,#10Issue Run Command T0 And Wait For TF0 to set
HERE: JNB TCON,HERE
SETB P0.0
CLR TCON.5
RET
```

(B) To generate a delay of 10 msec using internal machine cycle , cycle clock frequency is at 11.059MHz,Timer1 in mode 1 of operation

```
CLR  TCON.7
MOV  TMOD,#10
MOV  TL1,#2A
MOV  TH1,#0DC
MOV  TCON,#40
HERE: JNB TCON,HERE
SETB  P0.0
CLR  TCON.7
```

(C) To generate a frequency of 100Hz using external clock of 20KHz at P0.0, Timer/Counter in mode 2 of operation

```
MOV  TH0,#9C
CLR  TCON.5
MOV  TMOD,#46
MOV  TL0,#9C
CONTINUE: MOV  TCON,#10
HERE: JNB  TCON.5,HERE
CPL  P0.0
CLR  TCON.5
SJMP  CONTINUE
```

(III) SERIAL COMMUNICATION IMPLEMENTATION IN 8051 MC

(A) PROGRAM: To receive bytes of data serially and put them in P1. Set the baud rate at 4800, 8-bit data and 1-stop bit

```
MOV  TMOD,#20          Timer1 in mode2 i.e in Auto-reload
MOV  TH1,#6            4800 baud rate
MOV  SCON,#50          8-bit data,1-stop bit, REN enabled
SETB  TR1              Start timer1
HERE: JNB R1,HERE      Wait for char to come in
MOV  A,SBUF            Save incoming byte in Accumulator 'A'
MOV  P1,A              Send to port1
CLR  RI                Get ready to receive next byte
SJMP  HERE
```

(B) PROGRAM: To transfer the message "MPMC" serially at 9600 baud rate,8-bit data,1-stop bit, and this continuously.

```
MOV TMOD,#20           Timer1 in mode2
MO TH1,#3              9600 baud rate
MOV SCON,#50           8-bit data,1-stop bit, REN enabled
SETB TR1               Start timer1
AGN:MOV A,"M"          Transfer "M"
ACALL  XMIT
MOV A,"P"              Transfer "P"
ACALL  XMIT
MOV A,"M"              Transfer "M"
ACALL  XMIT
MOV A,"C"              Transfer "C"
ACALL  XMIT
SJMP  AGN              Keep doing it serial data transfer subroutine
XMIT:  MOV SBUF,A      Load SBUF
HERE:  JNB T1,HERE     Wait for last bit to transfer
CLR  TI                Get ready for next byte
RET
```

REGISTERS AND IT'S CORRESPONDING ADDRESS

NAME OF	ADDRESS

REGISTER	
ACC	0E0
B	0F0
DPH	82
DPL	83
P0	80
P1	90
P2	0A0
P3	0B0
TMOD	89
TCON	88
TH0	8C
TL0	8A
TH1	8D
TL1	8B
SCON	98
SBUF	99
PCON	87

8051MICROCONTROLLER PROGRAMMING USING KEIL SOFTWARE PROCEDURE:

- ✓ Create a new folder and give name (ex. My project as follows)

- ✓ Open the keiluvision software (double click on Keil Icon)
- ✓ goto project ---→new uvision project
- ✓ Browse your folder location and give project name and click on Save
- ✓ Select Atmel and expand it
- ✓ select AT89c51 and click on ok and then select NO
- ✓ goto file select NEW
- ✓ goto file select SAVE-→give the file name along with (.c) extension
- ✓ Click on Save and write the code….
- ✓ Goto project window-→expand TARGET-→right click on SOURCE GROUP1-→ADD Files to GROUP SOURCE GROUP1
- ✓ Browse your saved .c file -→add-→close
- ✓ Goto project window-→right click on target-→select options for target
- ✓ Change frequency to 11.0592-→goto OUTPUT tab-→select CREATE HEX FILE-→OK
- ✓ Goto project-→ select BUILD Target
- ✓ Then finally clear the errors (if any). Then run the program file on your folder---you can check the output on registers at which you perform the operations.

ARITHMETIC OPERATIONS USING KEIL(SOFTWARE) :

ADDITION OF TWO 8-BIT NUMBERS

```
MOV A,#13H
MOV B,#23H
ADD A,B
L1:SJMP L1
END
```

RESULT:

INPUT:

A: 13H

B: 23H

OUTPUT:

A:46H B: 23H

SUBTRACTION OF TWO 8-BIT NUMBERS

```
MOV A,#43H
MOV B,#23H
SUBB A,B
L1:SJMP L1
END
```

RESULT:

INPUT:

A: 43H

B: 23H

OUTPUT:

A: 20H B: 23H

MULTIPLICATION OF TWO 8-BIT NUMBERS

```
MOV A,#05H
MOV B,#20H
```

MUL AB

L1:SJMP L1

END

RESULT:

INPUT:

A: 05H

B: 20H

OUTPUT:

A: A0H B:00H

DIVISION OF TWO 8-BIT NUMBERS

MOV A,#40H

MOV B,#04H

DIV AB

L1:SJMP L1

END

RESULT:

INPUT:

A: 40H

B: 04H

OUTPUT:

A: 0AH B:00H

DATA TRANSFER AND LOGICAL OPERATIONS USING KEIL:

TO EXCHANGE THE CONTENTS OF ACCUMULATOR WITH R_O REGISTER CONTENT

MOV A,#05H

MOV R_O,#06H

XCH A, R_O

L1:SJMP L1

END

RESULT:

INPUT:

A: 05H R_O: 06H

OUTPUT: R_O:05H A:06H

TO REVERSE AND EXCHANGE THE GIVEN NUMBER

MOV A,#34H

MOV R_O,A

SWAP A

XCH A, R_O

L1:SJMP L1

END

RESULT:

A: 34H R_O:34

SWAP A : 43H

OUTPUT :

A:34H R_O:43H

TO CLEAR 16-RAM LOCATIONS

CLR A

MOV R_1,#60H

```
MOV R7,#16H
AHEAD:MOV @R1,A
        INC R1
      DJNZ R7,AHEAD
      L1: SJMP  L1
  END
```

RESULT:

INPUT:

R_1: 60H

R_7: 16H

OUTPUT:

Address (H)	60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
Value (H)	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

CHECKING OF 5TH BIT POSITION IN A GIVEN BYTE WHETHER IT IS "1" OR "0"

```
START: MOV A,#05H
       CLR A
       RLC A
       RLC A
       RLC A
       JC L1
       MOV A,#0FH
       MOV R1,A
L2: SJMP  L2
L1:  MOV A,#00H
     MOV R1,A
SJMP  START
END
```

RESULT:

INPUT:

(i) A:05H (ii) A:32H

OUTPUT:

(i) A:0FH (ii) A:00H

MICROCONTROLLER 8051 VIVA-VOCE QUESTIONS :-

1.Define the term Micro controller?

2.what is the difference between microprocessor and micro controller?

3.what is word size of 8051 micro controller?

4.what is the internal memory capacity of 8051 micro controller?

5.what is the external memory capacity of 8051 micro controller?

6.what is meant by DPTR?

7.what is the purpose of DPTR?

8.what is the word size of ALU in 8051 micro controller?

9.what are the different general purpose registers used in 8051 ?

10.How many number of timers used in 8051 ?

11.what are the different flags effected in arithmetic and logical operations in 8051 ?

12.which registers are used as pointers ?

13.what is the difference between conditional jump and unconditional jump ?

14.what are the different memories used in 8051 ?

15.what is the instruction used to access the external memory ?

16.What is the difference between JNZ & DJNZ ?

17.What is meant by Mov DPTR,#8000 and Movx A,@DPTR ?

18.Which register address 0F0 is ?

19. what are the different logical instructions used in 8051 ?

20. what are the different arithmetic instructions used in 8051 ?

BEYOND SYLLABUS EXPERIMENT

2. PROGRAM FOR DIGITAL CLOCK DESIGN USING 8086

AIM: to write program to display current system time

OBJECTIVE:

To write an assembly language program to display current system time.(DOS PROGRAMMING)

TOOLS REQUIRED: : PC installed with TASM

ALGORITHM:

Step1 : get the current system time

Step2 : BX=offset address of the string TIME

Step3 : PUSH AX onto the STACK,PUSH CX onto the STACK

Step4 : set AL=CH , CH=hours

Step5 : set [BX]=hr , [BX] is pointing to hr in the string TIME

Step6 : set AL=CL , CL=minutes

Step7 : set [BX+3]=min , [BX] is pointing to min in the string TIME

Step8 : set AL=DH , DH=seconds

Step9 : set [BX+6]=min , [BX] is pointing to sec

Step10 : POP a value from STACK into CX

Step11 : POP a value from STACK into AX

Step12 : Store the result

Step13 : Stop.

VIVA QUESTIONS:

1. What is the purpose of the 8086 microprocessor in a digital clock?
2. How does the 8086 fetch and execute instructions to display the time?
3. What is the role of the program counter (PC) in the digital clock program?
4. How does the 8086 handle interrupts generated by the clock circuitry?
5. What is the function of the output ports in displaying the time on the digital clock display?
6. How does the 8086 perform arithmetic operations to update the time?
7. What is the significance of the clock signal in the digital clock circuit?
8. How does the 8086 interact with the memory to store and retrieve the time?
9. What is the purpose of the delay loop in the digital clock program?
10. How does the 8086 handle errors or exceptions in the digital clock program?
11. What is the clock frequency used in the digital clock?
12. How many digits are displayed in the digital clock?
13. What type of display is used in the digital clock (e.g., 7-segment, LCD)?
14. How is the time set in the digital clock?
15. What is the power consumption of the digital clock?
16. How is the digital clock reset?
17. What is the accuracy of the digital clock?
18. How does the digital clock handle leap years?
19. Can you explain the timing diagram of the digital clock?
20. How does the digital clock interface with other devices?