1. INTRODUCTION TO TASM

Introduction:

The aim of this experiment is to introduce the student to assembly language programming and the use of the tools that he will need throughout the lab experiments. This first experiment let the student use the Dos Debugger and the Microsoft Turbo Assembler (TASM). Editing, Assembling, Linking, Execute up can be done using TASM software

Objectives:

1. Introduction to Microsoft Turbo Assembler (TASM)
2. General structure of an assembly language program
3. Use of the Dos Debugger program

Overview:

In general, programming of microprocessor usually takes several iterations before the right sequence of machine code instruction is written. The process, however is facilitated using a special program called an "Assembler". The Assembler allows the user to write alphanumeric instructions. The Assembler, in turn, generates the desired machine instructions from the assembly language instructions.

Assembly language programming consists of following steps:

	STEP	PRODUCES
1	Editing	Source file
2	Assembling	Object file
3	Linking	Executable file
4	Executing	Results

Table1.1: Assembly Language Programming Phases

Assembling the program:

The assembler is used to convert the assembly language instructions to machine code. It is used immediately after writing the Assembly language program. The assembler starts by checking the syntax or validity of the structure of each instruction in the source file .if any errors are found, the assemblers displays a report on these errors along with brief explanation of their nature. However If the program does contain any errors ,the assembler produces an object file that has the same name as the original file but with the "obj" extension

Linking the program:

The Linker is used convert the object file to an executable file. The executable file is the final set of machine code instructions that can directly be executed by the microprocessor. It is the different than the object file in the sense that it is self-contained and re-locatable. An object file may represent one segment of a long program. This segment can not operate by itself, and must be integrated with other object files representing the rest of the program ,in order to produce the final self-contained executable file

Executing the program

The executable contains the machine language code .it can be loaded in the RAM and executed by the microprocessor simply by typing, from the DOS prompt ,the name of the file followed by the carriage Return Key (Enter Key). If the program produces an output on the screen or sequence of control signals to control a piece of hard ware, the effect should be noticed almost immediately. However, if the program manipulates data in memory, nothing would seem to have happened as a result of executing the program.

Procedure to enter a program using TASM software

Start

↓

Run

↓

Type **CMD**

↓

Ok

Display shows

↓

C :\> **D:**

(Change to **D** drive because TASM is in **D** drive)

↓

Press **ENTER**

↓

D :\> **CD TASM**

↓

Press **ENTER**

↓

D: \TASM> **EDIT FILENAME.ASM**

Example edit sarada**.**asm

↓

Press **ENTER**

↓

Then the display shows editor

↓

Type the asm program

↓

Then the save the program (Use Alt+F keys to appear the option window)

↓

Exit from editor Using Alt+F keys

↓

Then Display shows D: \TASM>

↓

Enter the name **TASM FILENAME.ASM**

Example

↓

D: \TASM> **TASM sarada.asm**

Then Display shows Errors,(0)Warnings(0)

If there is errors correct them

↓

Enter the name **Tlink FILENAME.OBJ**

Example

↓

D: \TASM> **TLINK sarada.obj**

↓

Then the display shows

Turbo Link Version 3.0

↓

Enter the name **TD FILENAME.EXE**

Example

↓

D: \TASM> **TD sarada.exe**

↓

Then the display shows

Program has no symbol table

Choose **OK**

↓

RUN the Program using F9 Key or Select the RUN Option

↓

See the data in Registers

↓

See the data in Data segment Using Alt+F -→View-→Dump

Procedure to enter the data into memory location.

Sample program

```
Assume cs: code, ds: data
Data segment
Input equ  3000h
Result equ 4000h
Data ends
Code segment
Start:Mov ax,data
      Mov  ds,ax
      Mov si,input
      Mov al,[si]
      Inc si
      Mov bl,si
      Add al,bl
     Mov [di],al
     Int 03h
Code ends
End start
```

For the above sample program we have to enter the data into memory locations.

For that the procedure is given below

Step1: Type the sample program by using the above procedure.

Follow the steps up to Enter the name **TD FILENAME.EXE**

Example ↓

D: \TASM**> TD SARADA.EXE**

Then the (display shows **the program has no symbol table)**

Step2: Choose **OK** & using single step execution move specified address to index registers (Press **F8** key for single step execution)

Step3: Display shows **C:windows\system32\cmd.td filename.exe**

Select the view option using right arrow and choose the dump

Step4: Right click on the address to enter the address position

Step5: Enter the specified address and give data in data location

Step6: Run the program using F9 Key or Select the RUN Option location

Step7: See the result in specified memory

	Address	**data**
Ex:	4000	08h

2. MULTI-BYTE ADDTITON

AIM: To develop an ALP for the addition of two multi byte numbers

APPARATUS/SOFTWARE:

1.8086 Micro processor kit/TASM -1
2.RPS(+5V) -1

ALGORITHM:

1. Initialize the memory addresses for count , inputs and result in data segment.
2. Initialize DS register with base address of data segment.
3. Initialize AX register with 0000h.
4. Load CX register with count at memory location 2000h.
5. Load the address of inputs, output in SI(input 1) ,BP(input 2) and DI (output) registers respectively.
6. Load AL and BL registers with contents of SI and BP respectively.
7. Perform Addition operation between AL and BL registers.
8. Store the content of AL register into the memory location pointed by DI.
9. Increment the registers SI,BP,DL and decrement the register CX respectively.
10. Repeat from step 6 until count becomes zero.
11. End of the Program.

PROGRAM BEFORE EXECUTION:

```
ASSUME CS:CODE,DS:DATA

DATA SEGMENT
   COUNT   EQU 2000H
   INPUT1  EQU 2050H
   INPUT2  EQU 2080H
   RESULT EQU 3000H
 DATA ENDS
 CODE SEGMENT
    START: MOV AX,DATA
           MOV DS, AX
           MOV AX, 0000H
           MOV SI, COUNT
           MOV CX, [SI]
           MOV SI,INPUT1
           MOV BP,INPUT2
           MOV DI,RESULT
   AGAIN:  MOV AL,[SI]
           MOV BL,[BP]
           ADC AL,BL
           MOV [DI],AL
           INC SI
           INC BP
           INC DI
           DEC CX
           JNZ AGAIN
           INT 03H
  CODE ENDS
  END START
```

PROGRAM AFTER EXECUTION:

ADDRESS	OPCODE	MNEMONIC	OPERAND	COMMENTS

RESULTS:

REGISTER CONTENTS:

AX:

BP :

IP :

MEMORY CONTENTS:

Name	Memory Address	Value
INPUT1	2050	
INPUT2	2080	
RESULT	3000	

CONCLUSION:

3. FACTORIAL OF A GIVEN 8-BIT NUMBER

AIM: To develop an ALP for the factorial of a given number

APPARATUS/SOFTWARE:

1.8086 Microprocessor kit/TASM-1

2.RPS(+5v) -1

ALGORITHM:

1.Initialize the addresses for input and result as 3000h,4000h respectively.

2.Load DS register with the base address of data segment.

3.Load AX register with the data 0001h.

4.Load SI ,DI registers with input, result addresses respectively.

5.Load the input at SI location into CX register.

6.Check whether the input is zero.

7.If input is zero, goto step 8. Otherwise calculate the factorial using recursive formula until input becomes 1.

8.Store the content of AX register into the memory location pointed by DI.

9.End of the program

PROGRAM BEFORE EXECUTION:

```
ASSUME DS: DATA,CS:CODE
DATA SEGMENT
        INPUT EQU 3000H
        RESULT EQU 4000H
DATA ENDS
CODE SEGMENT
START:  MOV AX, DATA
          MOV DS, AX
          MOV AX, 0001H
          MOV DI, RESULT
          MOV SI, INPUT
```

```
        MOV CX, [SI]
        CMP CX, 0000H
        JZ XX
AGAIN:  MUL CX
        DEC CX
        JNZ AGAIN
   XX:  MOV [DI],AX
        INT 03H
        CODE ENDS
        END START
```

PROGRAM AFTER EXECUTION:

ADDRESS	OPCODE	MNEMONIC	OPERAND	COMMENTS

RESULT:

REGISTER CONTENTS:

AX = Z=
SI = P=
DI =
DS =
IP =

MEMORY CONTENTS:

Name	Memory address	Value
Input		
Result		

CONCLUSION:

4. SORTING OF NUMBERS IN ASCENDING ORDER

AIM: To Develop an ALP for Sorting in Ascending order using Bubble Sort Algorithm

APPARATUS/SOFTWARE:

1.8086Microprocessor kit/TASM -1

2. RPS(+5v) -1

ALGORITHM:

1. Initialize the addresses for count,input series as 2000h and 3000h respectively in Data segment.
2..Load DS register with base address of Data segment
3. Load DLwith count at 2000h memory location and decrement DL by one.
4. Copy CL register with data from DL register.
5. Load SI register with the starting address of input series.
6. Compare first number at SI location which is in al with number at SI+1 location
7. If borrow is obtained, conclude the first number is the smallest and go for Next iteration. Otherwise exchange the smallest number at SI+1 location With number at SI location before going to the next iteration
8. Increment SI , decrement CL and repeat from step 6 until CL becomes zero.
9. Decrement DL and repeat from step 4 until DL becomes zero.
10. End of the program

PROGRAM BEFORE EXECUTION:

```
ASSUME DS:DATA,CS:CODE
DATA SEGMENT
        COUNT          EQU 2000H
        INPUTSERIES  EQU 3000H
DATA ENDS
CODE SEGMENT
START:MOV AX,DATA
          MOV DS,AX
          MOV SI,COUNT
          MOV DL,[SI]
          DEC DL
NEXT:  MOV CL,DL
           MOV SI,INPUTSERIES
AGAIN: MOV AL,[SI]
            MOV AL,[SI+1]
            JB XX
            XCHG AL,[SI+1]
            MOV [SI],AL
 XX:      INC SI
            DEC CL
            JNZ AGAIN
            DEC DL
            JNZ NEXT
           INT 03H
CODE ENDS
END START
```

PROGRAM AFTER EXECUTION:

ADDRESS	OPCODE	MNEMONIC	OPERAND	COMMENTS

RESULT:

REGISTER CONTENTS :

AX : C=

SI : Z=

IP : P=

MEMORY CONTENTS:

Name	Memory Address	Value
count		
Input series		
Output		

CONCLUSION:

5. STRING DATA TRANSFER

AIM: To develop an alp to perform string data transfer.

APPARATUS/SOFTWARE:

1.8086 micro processor kit/TASM (TURBO ASSEMBLER) -1

2.RPS(+5v) - 1

ALGORITHM:

1: Assign source string, count to locations 2000h,3000h in data segment and destination string to 4000h extra segments.

2: Load DS and ES registers with base addresses of data and extra segment respectively..

3: Load SI and DI registers with offset addresses of source and destination string Respectively.

4: Load CL with count and execute MOVSB instruction for count number of times using the prefix REP.

5. End of the program

PROGRAM BEFORE EXECUTION:

```
        ASSUME CS:CODE, DS;DATA, ES:EXTRA
        DATA SEGMENT
        SOURCE STR EQU 3000H
        COUNT EQU 2000H
        DATA ENDS
        EXTRA SEGMENT
        DESTINATION STR EQU 4000H
        EXTRA ENDS
        CODE SEGMENT
START:  MOV AX,DATA
        MOV DS,AX
        MOV AX,EXTRA
        MOV ES,AX
        MOV SI,COUNT
```

```
MOV CL,[SI]
MOV SI,SOURCE STR
MOV DI,DESTINATION STR
CLD
REP MOVSB
INT 03H
CODE ENDS
END START
```

PROGRAM AFTER EXECUTION:

ADDRESS	OPCODE	MNEMONIC	OPERAND	COMMENTS

RESULT:

MEMORY CONTENTS:

INPUTS:

NAME	MEMORY LOCATION	VALUE
DEST STR		
SOURCE STR		

OUTPUTS:

NAME	MEMORY LOCATION	VALUE
DEST STR		

REGISTER CONTENTS:

AX:

SI:

DI:

IP:

FLAGS: C=

Z=

P=

CONCLUSION: .

6. COMPARISION OF TWO STRINGS

AIM: To develop an Assembly language program for string comparison.

APPARATUS/SOFTWARE:

1.8086 micro processor kit/TASM (TURBO ASSEMBLER)…….1

2.RPS(+5v) …….1

ALGORITHM:

1: Assign source string, count to locations 2000h,3000h in data segment and destination string to 4000h extra segments.

2: Load DS and ES registers with base addresses of data and extra segment respectively..

3: Load SI and DI registers with offset addresses of source and destination string Respectively.

4: Load CL with count and execute CMPSB instruction for count number of times using the prefix REPE.

5.End of the program

PROGRAM BEFORE EXECUTION:

```
        ASSUME CS:CODE, DS;DATA, ES:EXTRA
        DATA SEGMENT
        SOURCE STR EQU 3000H
        COUNT EQU 2000H
        DATA ENDS
        EXTRA SEGMENT
        DESTINATION STR EQU 4000H
        EXTRA ENDS
        CODE SEGMENT
START:  MOV AX,DATA
        MOV DS,AX
        MOV AX,EXTRA
        MOV ES,AX
```

```
        MOV SI,COUNT
        MOV CL,[SI]
        MOV SI,SOURCE STR
        MOV DI,DESTINATION STR
REPE    CMPSB
  XX:   INT 03H
        CODE ENDS
        END START
```

PROGRAM AFTER EXECUTION:

ADDRESS	OPCODE	MNEMONIC	OPERAND	COMMENTS

RESULT:

INPUTS:

NAME	MEMORY LOCATION	VALUE
COUNT		
SOURCE STR		
DEST STR		

OUTPUTS:

NAME	MEMORY LOCATION	VALUE
CX		

REGISTER CONTENTS:

AX:

BX:

CX:

SI:

DI:

IP:

FLAGS: C=

S=

a=

P =

CONCLUSION:

7. CONVERSION OF ASCII TO PACKED BCD NUMBER

AIM: To develop an ALP conversion on ASCII to BCD.

APPARATUS/SOFTWARE:

1.8086 Micro processor kit/TASM -1

2.RPS(+5v) -1

ALGORITHM:

1. Initialize the addresses for input and output as 2000h, 3000h respectively in data segment and assign count to value 04h.

2.Load DS register with base address of data segment

3.Load CX register with count equal to 04h.

4.Load SI,DI with input, output memory addresses respectively.

5.Load AL with input at SI location

6.Increment SI by one.

7.Load AH with input at SI location.

8.Mask the unwanted bits by performing AND operation of AX with 0F0FH.

9.Shift left the contents of AL by count number of times using SHL operation.

10.Perform Addition between AH and AL and load the value in AH.

11.Store the contents of AH into memory location pointed by DI.

12.End of the program.

PROGRAM BEFORE EXECUTION:

```
ASSUME CS:CODE,DS:DATA
DATA SEGMENT
        COUNT  EQU   04H
        INPUT     EQU  2000H
        OUTPUT EQU  3000H
DATA ENDS
```

```
CODE SEGMENT
START: MOV AX,DATA
       MOV DS,AX
       MOV  CX,COUNT
       MOV SI,INPUT
       MOV AL,[SI]
       INC SI
       MOV AH,[SI]
      AND AX,0F0FH
      SHL AL,CL
      ADD AH,AL
      MOV DI,OUTPUT
       MOV [DI],AX
       INT 03H
CODE ENDS
END START
```

PROGRAM AFTER EXECUTION:

ADDRESS	OPCODE	MNEMONIC	OPERAND	COMMENTS

RESULT:

REGISTER CONTENTS

AX

CX

SI

DI

IP

FLAG P=

MEMORY CONTENTS:

INPUT	**MEMORY ADDRESS**	**VALUE**
INPUT		

OUTPUT:

OUTPUT	**MEMORY ADDRESS**	**VALUE**
OUTPUT		

CONCLUSION:

8. CONVERSION OF PACKED BCD TO ASCII NUMBER

AIM: To write an assembly language program to convert Packed BCD into ASCII.

APPARATUS/SOFTWARE:

1. Soft ware used: 8086 microprocessor kit. ---1
2. RPS(+5) - 1

ALGORITHM:

1: Initialize the addresses for input and result as 2000h & 3000h respectively.

2: Load DS register with base address of data segment

3: Load SI, DI registers with memory addresses of input,result respectively.

4: load AL with input at memory location pointed by SI.

5: Increment SI

6: Load AH with input at memory location pointed by SI.

7: Perform Logical -OR operation between AX and 3030h

8: Store the result in AX at memory location pointed by DI,

9: End of the program.

PROGRAM BEFORE EXECUTION:

```
ASSUME CS:CODE,DS:DATA
DATA SEGMENT
        NUMBER EQU 2000H
        OUTPUT  EQU 3000H
 DATA ENDS
CODE SEGMENT
         START:
                MOV AX,DATA
                MOV DS,AX
                MOV SI,NUMBER
                MOV AL,[SI]
                INC SI
```

```
MOV AH,[SI]
OR AX,3030H
MOV DI,RESULT
MOV [DI],AX
INT 03H
CODE ENDS
END START
```

PROGRAM AFTER EXECUTION:

ADDRESS	OPCODE	MNEMONIC	OPERAND	COMMENTS

RESULT:

REGISTER CONTENT:

AX:

SI:

DI:

FLAGS: P=

MEMORY CONTENT:

INPUT:

NAME	ADDRESS	VALUE
INPUT		

OUTPUT:

NAME	ADDRESS	VALUE
RESULT		

CONCLUSION:

9. TO COUNT POSITIVE AND NEGATIVE NUMBERS IN A GIVEN ARRAY

AIM: To develop ALP for counting the positive and negative numbers in the given set of 8-bit numbers.

APPARATUS/SOFTWARE:

1. 8086 Microprocessor kit/TASM(Turbo Assembler) - - 1
2. RPS (+5v) - - 1

ALGORITHM:

1: Start the program

2: Allocate memory locations for count, input and output.

3: Load DS register with base address of data segment.

4: Load CX register with count form 2000H memory location.

5: Initialize BL and BH registers with 00H.

6: Load AX register with input form 3000H memory location.

7: Perform ROL (Rotate left) operation on AX register for one time to check the nature of MSB.

8: If carry is obtained from MSB, conclude number as negative and increment the negative counter. Otherwise increment positive counter.

9: Repeat from step 7 for each input taken in AX until count becomes zero.

10: Once count is zero, store the count of negative and positive Numbers (available in BX register) into the memory location 4000H.

11: End of the program.

PROGRAM BEFORE EXECUTION:

```
ASSUME : CODE, DS: DATA
DATA SEGMENT
        COUNT EQU 2000H
```

```
        INPUT   EQU 3000H
        OUTPUT EQU 4000H
    DATA ENDS
   CODE SEGMENT

START: MOV AX,DATA
       MOV DS,AX
       MOV SI,COUNT
       MOV CX, [SI]
       MOV BL, 00H
       MOV BH, 00H
      MOV SI, INPUT
NEXT:  MOV AX, [SI]
       ROL AX,01H
       JC XX
       JNC BL
       JMP YY
  XX:  INC BH
  YY:  ADD SI, O2H
       DEC CX
       JNZ NEXT
       MOV SI, OUTPUT
       MOV [SI], BX
       INT 03H
       CODE ENDS
       END START
```

PROGRAM AFTER EXECUTION:

ADDRESS	OPCODE	MNEMONIC	OPERAND	COMMENTS

RESULT:

REGISTERS:

AX:
BX:
SI :
IP:

FLAGS: Z:
P:

MEMORY CONTENTS:

NAME	MEMORY ADDRESS	VALUE
COUNT		
INPUT		
OUTPUT		

CONCLUSION:

10. TO COUNT EVEN AND ODD NUMBERS IN A GIVEN SERIES

AIM: To develop ALP for counting the even and odd numbers in the given set of 8-bit numbers.

APPARATUS/SOFTWARE:

3. 8086 Microprocessor kit/TASM(Turbo Assembler) - - 1
4. RPS (+5v) - - 1

ALGORITHM:

1: Start the program
2: Allocate memory locations for count, input and output.
3: Load DS register with base address of data segment.
4: Load CX register with count form 2000H memory location.
5: Initialize BL and BH registers with 00H.
6: Load AX register with input form 3000H memory location.
7: Perform ROR(Rotate Right) operation on AX register for one time to check the nature of LSB.
8: If carry is obtained from LSB, conclude number as negative and increment the negative counter. Otherwise increment positive counter.
9: Repeat from step 7 for each input taken in AX until count becomes zero.
10: Once count is zero, store the count of negative and positive Numbers (available in BX register) into the memory location 4000H.
11: End of the program.

PROGRAM BEFORE EXECUTION:

```
ASSUME : CODE, DS: DATA
DATA SEGMENT
        COUNT EQU 2000H
        INPUT   EQU 3000H
        OUTPUT EQU 4000H
```

```
DATA ENDS
CODE SEGMENT

        START: MOV AX,DATA
               MOV DS,AX
               MOV SI,COUNT
               MOV CX, [SI]
               MOV BL, 00H
               MOV BH, 00H
               MOV SI, INPUT
         NEXT: MOV AX, [SI]
               ROR AX,01H
               JC XX
               JNC BL
               JMP YY
           XX: INC BH
           YY: ADD SI, O2H
               DEC CX
               JNZ NEXT
               MOV SI, OUTPUT
               MOV [SI], BX
               INT 03H
  CODE ENDS
  END START
```

PROGRAM AFTER EXECUTION:

ADDRESS	OPCODE	MNEMONIC	OPERAND	COMMENTS

RESULT:

REGISTERS:

AX: FLAGS: Z=

BX: P=

SI :

IP:

MEMORY CONTENTS:

Name	Memory address	Value
COUNT		
INPUT		
OUTPUT		

CONCLUSION:

11. COUNT NUMBER OF 0'S AND 1'S IN A MULTI BYTE NUMBER

AIM: To develop an alp to count number of one's and zero's in a given multi-byte number.

APPARATUS/SOFTWARE:

1.8086 microprocessor kit/TASM (TURBO ASSEMBLER) ..1

2.RPS(+5v) ..1

ALGORITHM:

1: Initialize the count, input, result with address 2000H,2050H,3000H respectively.

2: Load ds register with base address of data segment.

3.Load DL with count present at memory location pointed by SI

4.Initialize the register BX with 0000h to hold zeros and ones count finally.

5.Load the input byte pointed by SI into AL register..

6.Load CL register with 08h to act as inner counter.

7.Perform shift right operation on al register by 01H.

8.If carry exists, increment ones counter. Otherwise increment zeros counter before going to next input byte.

9.Increment SI and repeat from step 7 until count becomes zero.

10.Decrement DL and repeat from step 5 until it becomes zero.

11.End of the program

PROGRAM BEFORE EXECUTION:

```
ASSUME CS:CODE,DS:DATA

DATA SEGMENT
        COUNT EQU 2000H
        INPUT  EQU 2050H
        RESULT EQU 3000H
DATA ENDS
```

```
CODE SEGMENT

        START: MOV AX,DATA
               MOV DS,AX
               MOV AX,0000H
               MOV BL,00H
               MOV BH,00H
               MOV SI,COUNT
               MOV DL,[SI]
               MOV SI,INPUT
        NEXT : MOV AL,[SI]
               MOV CL,08H
       AGAIN:SHR AL,1
               JC XX
               INC BL
              JMP YY
         XX: INC BH
         YY: DEC CL
              JNZ AGAIN
              INC SI
             DEC DL
             JNZ NEXT
             MOV SI,RESULT
             MOV [SI],BX
             INT 03H
CODE ENDS
END START
```

PROGRAM AFTER EXECUTION:

ADDRESS	OPCODE	MNEMONIC	OPERAND	COMMENTS

RESULT:

REGISTER CONTENTS:

AX:

BX:

SI:

IP:

MEMORY CONTENTS:

Name	Memory address	Value
Count		
Input		
Result		

CONCLUSION:

12. SUM OF N 8-BIT BINARY NUMBERS

AIM: To develop an ALP in 8086 to perform sum of n 8-bit binary numbers.

APPARATUS/SOFTWARE REQUIRED:

1. 8086 microprocessor kit/TASM -- 1
2. RPS(+5V) -- 1

ALGORITHM:

1. Start the program.
2. Initialise the variables COUNT,INPUT,OUTPUT to memory locations in data segment.
3. Load the base addresses of Data segment into the register DS.
4. Load the count value to the register CL.
5. Load the Input to the register BL.
6. Add the registers AL, BL along with carry.
7. Increment SI by 1 and Decrement CL by 1.
8. If CL is not zero go to step 5 or else go to step 9.
9. If there is a carry resulted go to step 11 or else go to step 10.
10. Increment AH by 1.
11. The final result in the register AX is brought to the variable OUTPUT using the register SI.
12. End of the program.

PROGRAM BEFORE EXECUTION:

```
ASSUME DS:DATA,CS:CODE

DATA SEGMENT
        COUNT   EQU 2000H
        INPUT   EQU 3000H
        OUTPUT EQU 4000H
DATA ENDS
```

```
CODE SEGMENT
START:  MOV AX,DATA
        MOV DS,AX
        MOV AX,0000H
        MOV SI,COUNT
        MOV CL,[SI]
        MOV SI,INPUT
AGAIN:  MOV BL,[SI]
        ADC AL,BL
        INC SI
        DEC CL
        JNZ AGAIN
        JNC XX
        INC AH
XX:     MOV DI,OUTPUT
        MOV [DI],AX
        INT 03H
CODE ENDS
END START
```

PROGRAM AFTER EXECUTION:

ADDRESS	OPCODE	MNEMONIC	OPERAND	COMMENTS

RESULT:

MEMORY CONTENTS:

INPUTS:

NAME	MEMORY ADDRESS	VALUE
COUNT		
INPUT		

OUTPUT:

NAME	MEMORY ADDRESS	VALUE
OUTPUT		

REGISTER CONTENTS:

AX =

BX =

SI =

DI =

IP =

FLAG CONTENTS:

Z=

P=

CONCLUSION:

13. TO FIND THE LARGEST NUMBER IN THE GIVEN ARRAY

AIM: To develop an ALP in 8086 to find the largest number in the given series of numbers.

APPARATUS /SOFTWARE REQUIRED:

1. 8086 microprocessor kit/ TASM --1
2. RPS (+5V) --1

ALGORITHM:

1. Start the Program.
2. Initialize the variables COUNT, NUM, RESULT in memory and equate them to 2000H, 2050H and 3000H respectively in Data segment.
3. Load the base address of Data segment into the register DS.
4. Load the COUNT value into the register DL by using SI.
5. Now decrease the value of DL by 1.
6. Load the register SI with value of NUM i.e, 2050H.
7. Now load the value from the 2050H location into the register AL.
8. Perform the comparison operation between the value in AL and the value from the next memory location of SI.
9. If a BARROW has resulted from the above operation then goto step 12 otherwise goto next step.
10. Now exchange the values of AL, value in the next memory location of SI to get the correct sequence of numbers.
11. Load the value of AL into the memory location given by the value in the register SI.
12. Increase the value of SI by 1 and decrease the value of DL by 1.
13. If the value of count is NOT ZERO then goto step 7 otherwise goto next step.
14. Now the final result available in the register BL is brought into the variable RESULT by using the register SI.
15. End of the Program.

PROGRAM BEFORE EXECUTION:

```
ASSUME CS: CODE, DS: DATA

    DATA SEGMENT
        COUNT EQU 2000H
        NUM EQU 2050H
        RESULT EQU 3000H
    DATA ENDS

    CODE SEGMENT
    START:   MOV AX, DATA
             MOV DS, AX
             MOV SI, COUNT
             MOV DL, [SI]
             DEC DL
             MOV SI, NUM
          YY: MOV AL, [SI]
             CMP AL, [SI+1]
             JB ZZ
             XCHG AL, [SI+1]
             MOV [SI], AL
          ZZ: INC SI
             DEC DL
             JNZ YY
             MOV BL, [SI]
             MOV SI, RESULT
             MOV [SI], BL
             INT 03H
    CODE ENDS
    END START
```

PROGRAM AFTER EXECUTION:

ADDRESS	OPCODE	MNEMONIC	OPERAND	COMMENTS

RESULT:

MEMORY CONTENTS

INPUTS

Name	Memory address	Value
COUNT		
NUM		

OUTPUTS

Name	Memory address	Value
RESULT		

REGISTER CONTENTS

AX =

BX =

SI =

IP =

FLAG CONTENTS

Z =

P =

CONCLUSION:

14. SUM OF N-FACTORIALS

AIM: To write an ALP in 8086 to perform sum of n-factorial numbers and by assuming memory locations.

APPARATUS/SOFTWARE: 1.8086 Microprocessor kit / TASM --1

2.RPS(+5V) -- 1

ALGORITHM:

01: Initialize the addresses for count, input, result as 3000h,3050h,4000h respectively.

02: Initialize DS register with base address of data segment.

03: Initialize base pointer with 0000h to store the final result.

04: Load CL register with required count stored in memory location of 3000h.

05: Initialize SI,DI with starting addresses of input, result respectively.

06: Initialize AX register with 0001h.

07: Load BL register with required data from memory location pointed by SI.

08: Compare BX and 0000h.If zero flag is SET then go to step 12.Other wise go to next step.

09: Multiply AX register with BX.

10: Decrement BX.

11: If BX is not equal to zero, then go to step 09,otherwise go to next step.

12: Perform addition operation on the registers BP and AX.

13: Increment SI register and Decrement CX .

14: If CX is not equal to zero, then go to step 06,otherwise go to next step.

15: Store the content of BP register in the memory location pointed by DI.

16: Stop.

PROGRAM BEFORE EXECUTION:

```
ASSUME CS:CODE,DS:DATA
 DATA SEGMENT
                COUNT EQU 3000H
                 INPUT EQU 3050H
                RESULT EQU 4000H
 DATA ENDS
```

```
CODE SEGMENT
        START:MOV AX,DATA
              MOV DS,AX
              MOV BP,0000H
              MOV SI,COUNT
              MOV DI,RESULT
              MOV CL,[SI]
              MOV SI,INPUT
        NEXT: MOV AX,0001H
              MOV BL,[SI]
              CMP BX,0000H
              JZ LAST
       AGAIN:MUL BX
             DEC BX
             JNZ AGAIN
        LAST: ADC BP,AX
              INC SI
              DEC CX
              JNZ NEXT
             MOV [DI],BP
             INT 03H
CODE ENDS
END START
```

PROGRAM AFTER EXECUTION:

ADDRESS	OPCODE	MNEMONIC	OPERAND	COMMENTS

RESULT:

REGISTERS:

AX:

SI :

DI:

BP:

FLAGS: Z=

P=

MEMORY CONTENTS:

Name	Memory location	Value
count		
input		
output		

CONCLUSION:

15. STEPPER MOTOR MODULE INTERFACING USING INTEL 8255

AIM: Write an ALP in 8086 to rotate the stepper motor for two rotations in clockwise directions and one rotations in anti-clockwise direction repeatively (or) continuously. By interfacing stepper motor control module to 8086 microprocessor through Intel8255 .

APPARATUS:

1. 8086 Microprocessor kit -1
2. Keyboard -1
3. Fixed power supply (D.C) 5V,1.5A -1
4. Fixed power supply (D.C) 12V,1.5A -1
5. Stepper motor module -1
6. Stepper motor 12V -1

SPECIFICATIONS:

1) Permanent magnet D.C. stepping motors two phase bifillar wound.

2) Step angle :$1.8^0 \pm 5\%$ non cumulative .

3) step/revolution :200 .

CIRCUIT DIAGRAM:

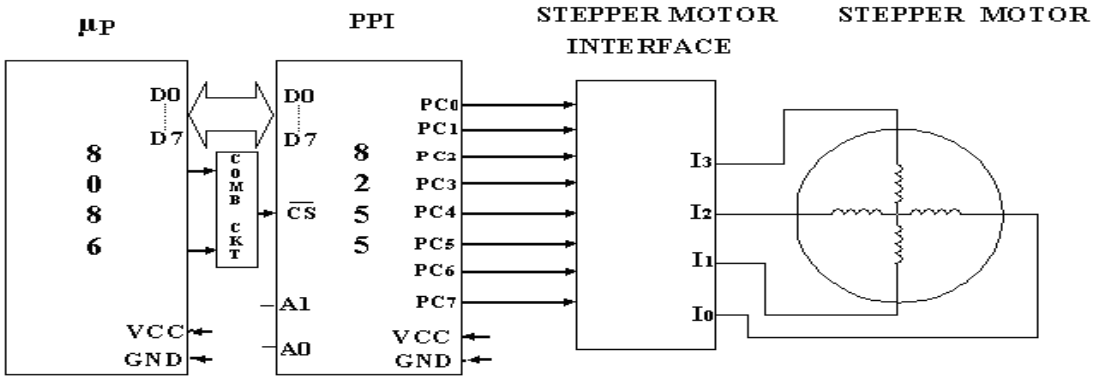


Fig: Interfacing stepper motor module to8086 microprocessor

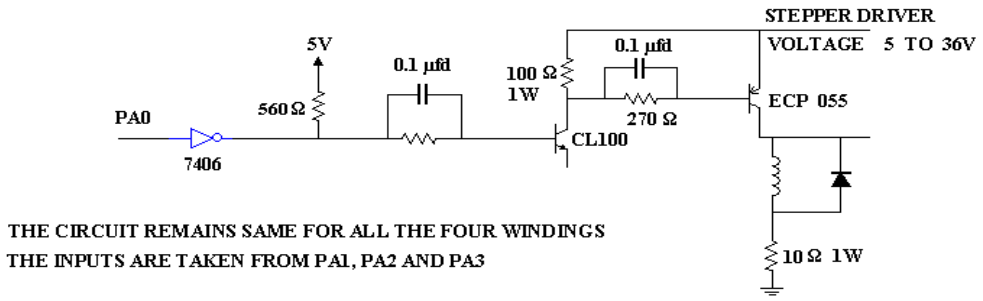


Fig: Internal diagram of stepper motor control module

PROGRAM REQUIREMENTS:

Intel 8255 Port declaration: Port C :output

Control word :80H

Command words: To activate the windings

Clock wise direction	Anti clock wise direction
77 h	EE h
BB h	DD h
DD h	BB h
EE h	77 h

ALGORITHM:

Step 1: write the control word in the control register.
Step 2: locate the forward revolution count value in the count register.
Step 3: activate the armature 1 of the stepper motor.
Step 4: call the delay sub program.
Step 5: activate the armature2 of the stepper motor.
Step 6: call the delay sub program.
Step 7: activate the armature 3 of a stepper motor
Step 8: call the delay sub program.
Step 9: activate the armature 4 of a stepper motor
Step 10: call the delay sub program.
Step 11: repeat the step 3 to step 10 until count value equal to zero.
Step 12: activate the armature 4 of the stepper motor.
Step 13: call the delay sub program.
Step 14: activate the armature 3 of a stepper motor.
Step 15: call the delay sub program.
Step 16: activate the armature2 of the stepper motor.
Step 17: call the delay sub program.
Step 18: activate the armature 1 of the stepper motor.
Step 19: call the delay sub program.

Step 20: repeat the step 3 to step 10 until count value equal to zero.

Step 21:repeat the step 1 to step 20.
Note: count may vary in clockwise and anticlockwise direction

ASSEMBLY LANGUAGE PROGRAM BEFORE EXECUTION:

Label	Mnemonic	Operand	Comment
	MOV	DX,0FFC6	:select the CW address
	MOV	AL,80	;initialize with control word
	OUT	DX,AL	;locate control word in CW register
back:	MOV	CL,64	;initializing with count value
Go	MOV	DX,0FFC4	;choose the port c address
	MOV	AL,77	;initialize AL with 77H
	OUT	DX,AL	;send AL data to port C
	CALL	delay	;call the delay subprogram
	MOV	AL,BB	;initialize AL with BBH
	OUT	DX,AL	;send AL data to port C
	CALL	delay	;call the delay subprogram
	MOV	AL,DD	;initialize AL with DDH
	OUT	DX,AL	;send AL data to port C
	CALL	delay	;call the delay subprogram
	MOV	AL,EE	;initialize AL with EEH
	OUT	DX,AL	;send AL data to port C
	CALL	delay	;call the delay subprogram
	DEC	CL	;decrement the counter
	JNE	go	;if not equal go to go label
	MOV	CH,32	:initialize the counter value.
xyz	MOV	DX,0FFC4	;choose the port c
	MOV	AL,EE	;initialize AL with EEH
	OUT	DX,AL	;send AL data to port C
	CALL	delay	;call the delay subprogram
	MOV	AL,DD	;initialize AL with 77H
	OUT	DX,AL	;send AL data to port C
	CALL	delay	;call the delay subprogram
	MOV	AL,BB	;initialize AL with 77H
	OUT	DX,AL	;send AL data to port C
	CALL	delay	;call the delay subprogram
	MOV	AL,77	;initialize AL with 77H
	OUT	DX,AL	;send AL data to port C
	CALL	delay	;call the delay subprogram
	DEC	CH	;decrement the counter
	JNE	xyz	;if not equal goto xyz label
	JMP	back	;jump to back label

DELAY PROGRAM:

Lable	Mnemonics	Operand	Comments
	MOV	BX,FFFF	;initialize the count value.
abc:	DEC	BX	;decrement the register
	JNE	abc	;if it is not equal to zero goto abc label
	RET		;return to the main program

EXPECTED RESULTS:

When windings are excited in proper manner, stepper motor may rotate two times in clock wise and one time in anti clock wise direction this process repeat.

ASSEMBLY LANGUAGE PROGRAM AFTER EXECUTION:

ADDRESS	OPCODE	MNEMONIC	OPERAND	COMMENTS

DELAY PROGRAM:

Address	Opcode	Mnemonic	Operand

RESULTS: No of Rotations of stepper motor

CONCLUSION:

16. ANALOG TO DIGITAL CONVERTER INTERFACING USING INTEL 8255

AIM: Write an ALP in 8086 to convert Analog information into digital by interfacing ADC (Analog to digital converter) module to 8086 microprocessor through Intel 8255.

APPARATUS:

1. 8086microprocessor kit
2. ADC interfacing module
3. Power supply.+5V
4. Key board

SPECIFICATIONS: 1. Voltage specifications: +5V, GND

2. ADC 0809 IC specifications

a).Resolution 8 bit
b).Single supplu +5v DC
c).Output power 15mW
d).Conversion time 100 μs
e).Total unadjusted error ±½ LSB and ±1 LSB
f).Input channels-8
g). Interface type: Parallel

CIRCUIT DIAGRAM:

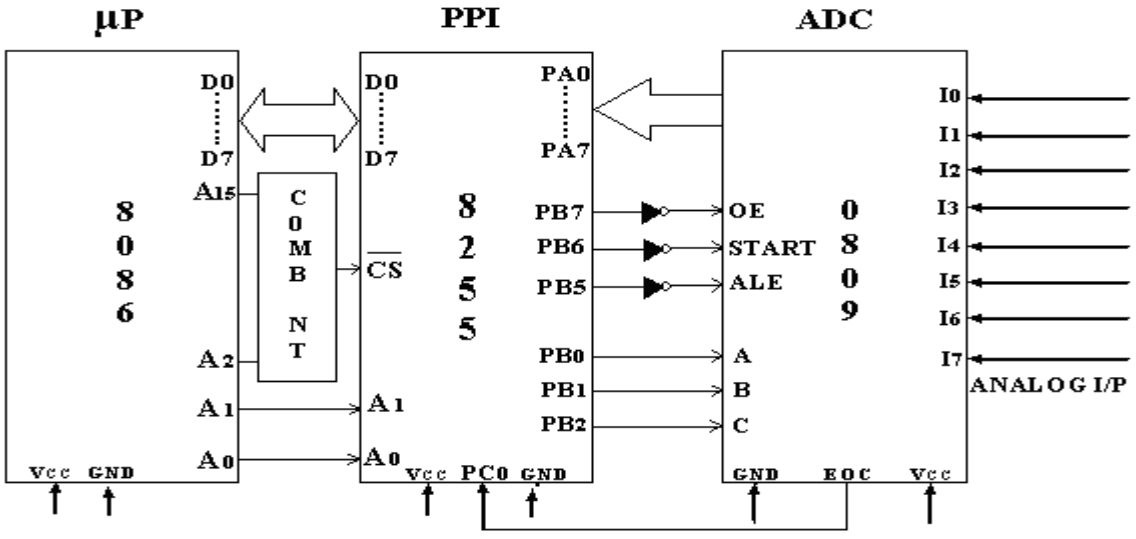


Fig: Interfacing ADC module to 8086 microprocessor through Intel 8255

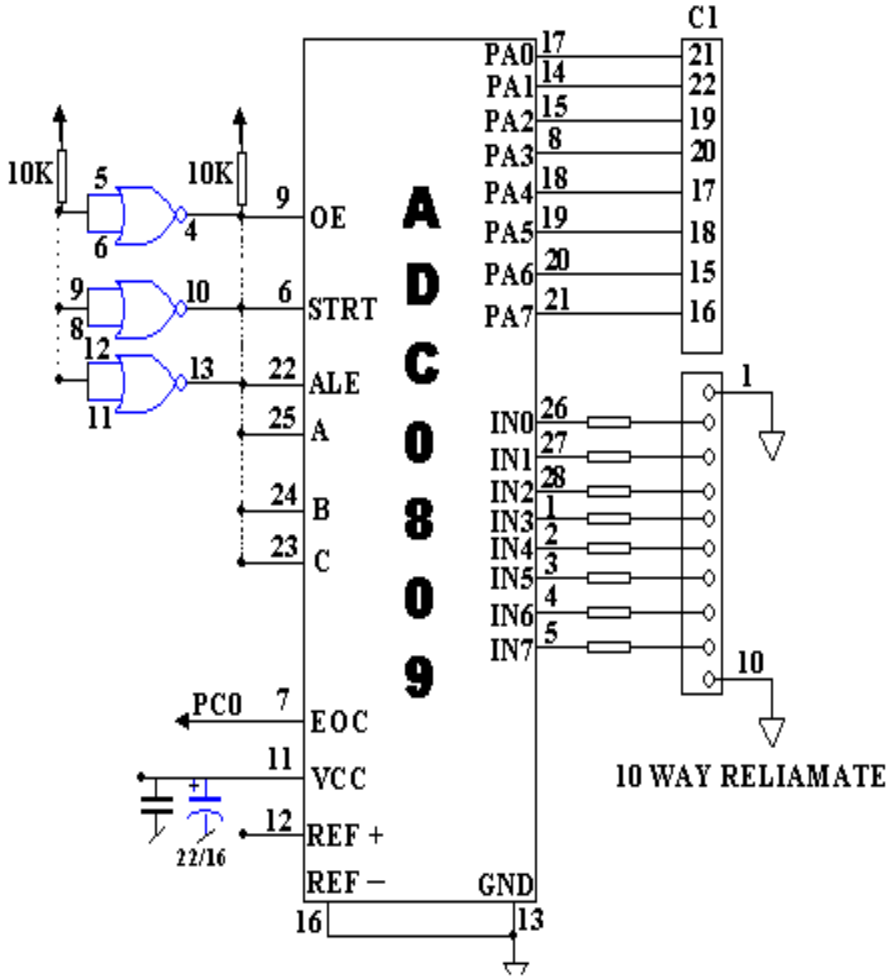


Fig: Internal diagram of ADC module

ALGORITHM:

Step 1:Load the control word in to CWR of 8255 to make the PortA , PortC as input
Port B as an output port
Step 2:Send the dummy word to clear the A/D output
Step 3: Send a soft ware pulse to start of conversion(SOC) &ALE
Step 4:Read EOC signal if EOC =1,conversion is over, other wise read EOC until get
EOC=1
Step 5:Enable the output buffers.

Step 6: Read digital data from port A

Step 7: End of the program

PROGRAM REQUIREMENTS:

Port declaration:

Input ports : Port A, Port C

Outputport :portB

Control word format:

D7	D6	D5	D4	D3	D2	D1	D0	
1	0	0	1	0	0	0	1	=91H

Pulse:

PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
↓	↓	↓					
OE	START	ALE	X	X	C	B	A

Dummy Word:

PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0		
0	1	1	0	0	1	1	1	=67	
1	1	1	0	0	1	1	1	=E7	pulse to ALE&SOC
1	0	0	0	0	1	1	1	=87	
1	1	1	0	0	1	1	1	=E7	

CIRCUIT DESCRIPTION:

IN_0-IN_7 are the Analog inputs to the ADC . These inputs are fed through the 16-way Relimate connector on the bourd. Pin1&pin10 on the ground points & 2-9 are the input channel

ASSEMBLY LANGUAGE PROGRAM BEFORE EXECUTION:

Label	Mnemonics	Operand	Comments
	MOV	DX,0FFC6	Load DX with 0013(CWR address
	MOV	AL,91	Load control word
	OUT	DX,AL	Sends AL to DX
	MOV	DX,0FFC2	Load port B address into DX
	MOV	AL,67	Send a pulse for SOC&ALE
	OUT	DX,AL	
	MOV	AL,E7	
	OUT	DX,AL	
	MOV	AL,87	
	OUT	DX,AL	
	MOV	AL,E7	
	OUT	DX,AL	
Back	MOV	DX,0FFC4	Load port C address into DX
	IN	AL, DX	Read EOC through PC0
	AND	AL,01	AND_{ed} with 01 & AL:working for Pc_0 bit
	CMP	AL,01	Compare AL with 01
	JNZ	BACK	If conversion is not completed go back Other wise proceeds
	MOV	DX,0FFC2	Load Port B address into DX
	MOV	AL,67	Send a word to enable the output buffer
	OUT	DX,AL	
	MOV	DX,0FFC0	Load Port A address into DX
	IN	AL,DX	Read digital data from Port A
	MOV	[2000],AL	Load digital data into 2000H locations
	INT	3	End of the program

EXPECTED RESULTS:

Input	Output	
(Analog)	Address	data
0V	2000:	
1V	2000:	
2V	2000:	
3V	2000:	
4V	2000:	
5V	2000:	

ASSEMBLY LANGUAGE PROGRAM AFTER EXECUTION:

ADDRESS	OPCODE	MNEMONIC	OPERAND	COMMENTS

RESULTS:

1) i/p = 0v o/p =
2) i/p = 4v o/p =
3 i/p = 5V o/p =

CONCLUSION:

17.DIGITAL INPUT DIGITAL OUTPUT MODULE INTERFACING USING INTEL 8255

AIM: Write an ALP in 8086 to implement the following by interfacing DIDO (Digital Input and Digital Output) module with 8086 microprocessor using 8255

i) f(A,B,C)=∑(0,1,2,3,4,5,6)
ii) 4:1 MULTIPLEXER
iii) 3 to8 decoder
iv) 2's complement

APPARATUS:

1. 8086Microprocessor kit
2. DIDO module
3. 5V DC power supply
4. key board

SPECIFICATIONS: Interfacing kit specifications

Vcc +5v
IC7404 Vcc +5v
Current max 12mA

CIRCUIT DIAGRAM

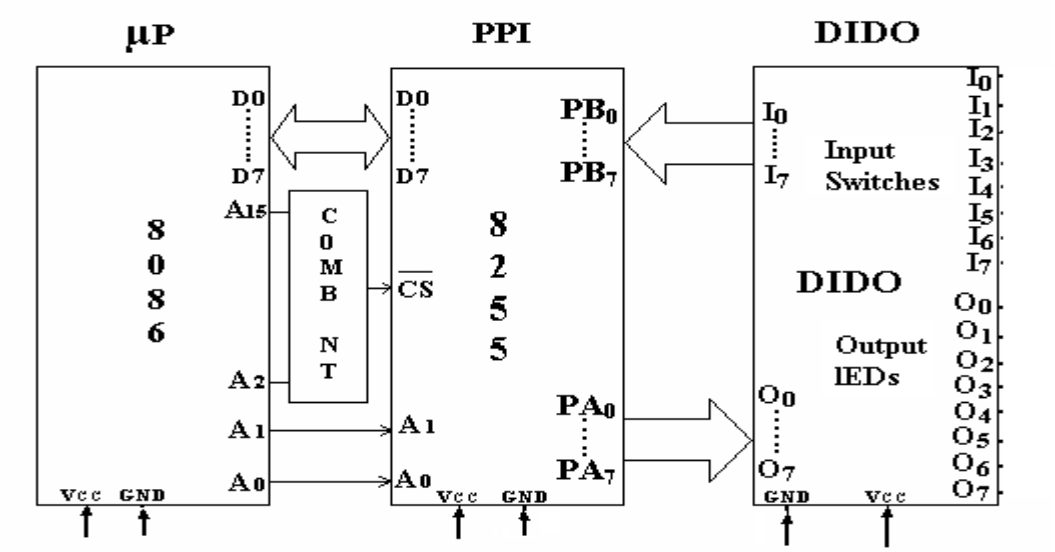


Fig: Interfacing DIDO module to 8086 microprocessor

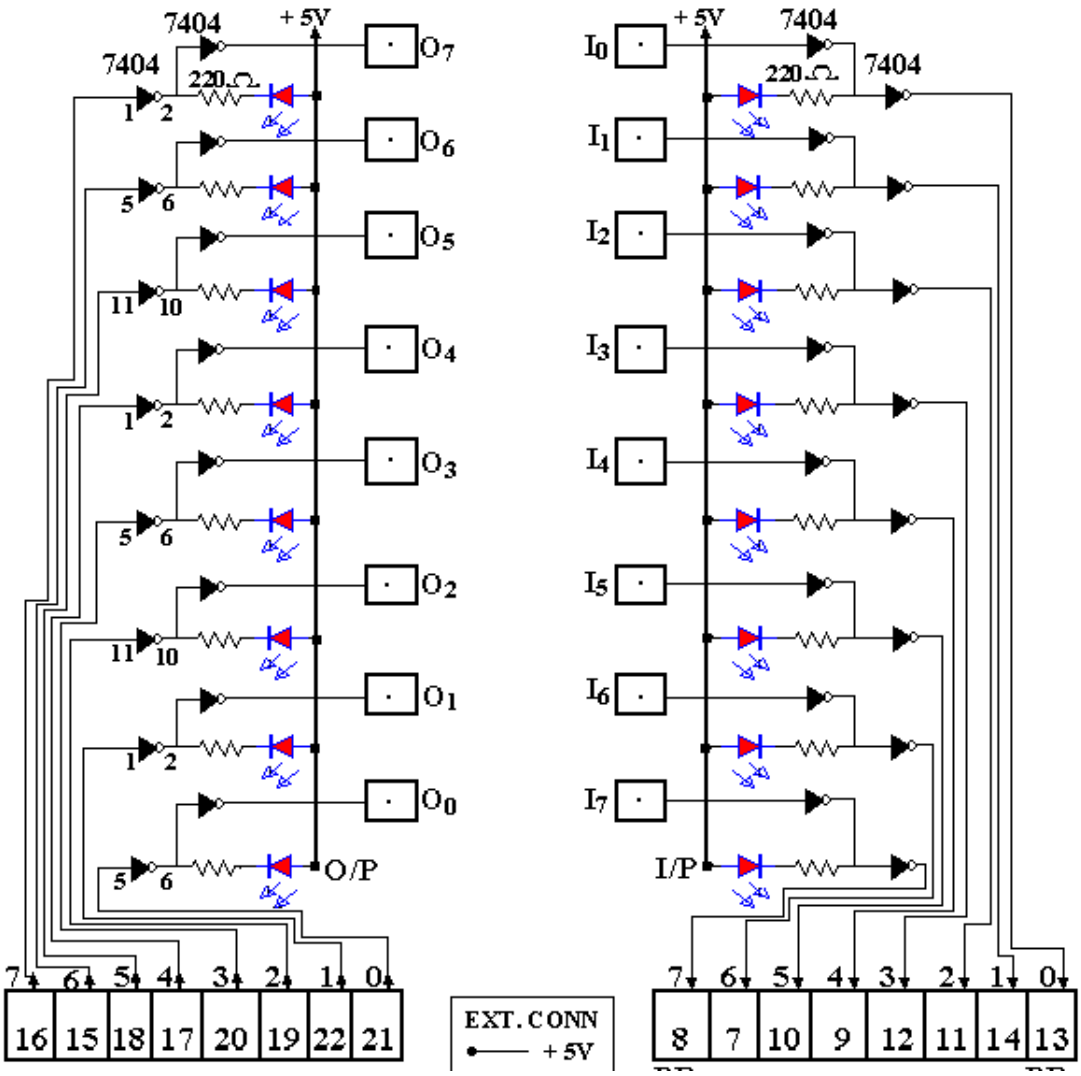


Fig: Internal diagram of DIDO (Digital input Digital output module)

CIRCUIT DESCRIPTION:

The system consists of 8 input SPDT switches which give logic 0 or1 signal to input lines Of port B (bit0-7) the output port A (bit 0-7) is buffered by open collector inverter, the 7406.The output LED's are connected to output buffer

ALGORITHM:

Step 1.Send control word (to make Port B as input port and Port A as output port) to CWR of 8255.

Step 2.Read data byte from port B.

Step 3.To get the desired output Mask the corresponding bits in data byte except required one by using AND operation.

Step 4.Store the result in accumulator.

Step 5.Compare accumulator with 03H and 07H.

Step 6.To get the desired output for each input go to step 3.

PROGRAM REQUIREMENTS:

Port declaration:

Intel 8255: Input port : Port B

Out put port: Port A:

Control word register:

D7	D6	D5	D4	D3	D2	D1
1	0	0	0	0	1	0

= 82H

ASSEMBLY LANGUAGE PROGRAM BEFORE EXECUTION:

Label	Mnemonic	Operand	Comments
	MOV	DX,0FFC6	Load the address of control word register (CWR) in to DX
	MOV	AL,82H	Initialization of control word
	OUT	DX,AL	Load the control word in to CWR
UP:	MOV	DX,0FFC2	Load address of Port B into DX
	IN	AL,DX	Read data from Port B
	AND	AL,07	Mask the bits in data byte by using AND operation
	CMP	AL,03	Compare result with 03h
	JE	DOWN	If it is equal jump down
	CMP	AL,07	Again compare result with 07h
	JE	DOWN	If it equal jump to down
	MOV	AL,01	If it is not equal load AL with o1
	MOV	DX,0FFC0	Load address of port A in to DX register
	OUT	DX,AL	Send 01H to Port A
	JMP	UP	For each input condition repeat this procedure
DOWN	MOV	AL,00	Load accumulator with 00
	MOV	DX,0FFC0	Load address of Port A in to DX register
	OUT	DX,AL	Write data into Port A
	JMP	UP	For each condition repeat this procedure go up again

EXPECTED RESULTS:

Inputs			Output
A	B	C	Y
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

ASSEMBLY LANGUAGE PROGRAM AFTER EXECUTION:

Address	Opcode	Mnemonic	Operand	Comments

RESULTS:

Input (Switch position)	**Out put** LED ON
1 0 0	
1 0 1	
1 1 0	
0 0 0	
0 0 1	
0 1 0	

1= switch ON
0= switch OFF

INSTRUCTIONS:

1. Check the polarities of D.C Chord (+5V) before switch ON the power supply to the module as well as to the 8086 microprocessor kit
2. Reset the Microprocessor Kit while connecting the bus between microprocessor and Interfacing Kit
3. Be sure about the direction of the cable
4. Verify all the connections and then execute the program
5. Change the switch positions of I0 to I7 on the interface card and observe change on the output LEDs 0o -07

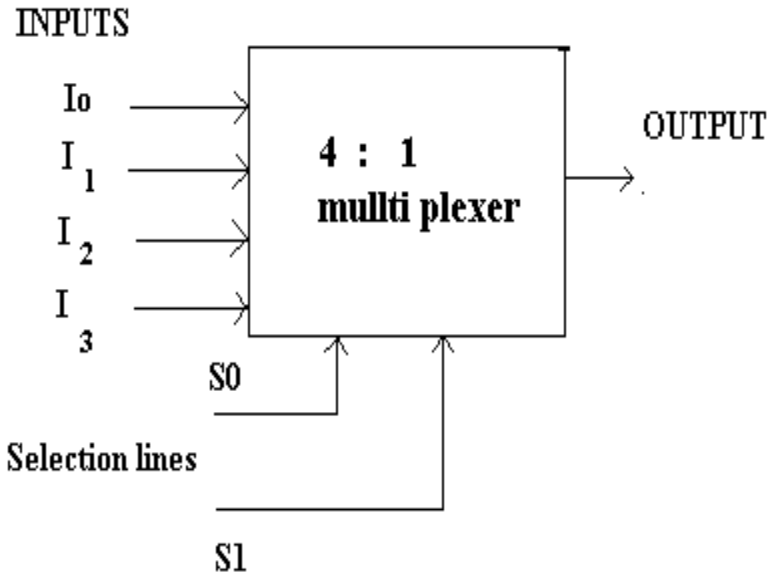


Fig: 4:1 Multiplexer

ALGORITHM:

Step 1:- Send control word (to make portB as input port and portA asoutput port)to CWR of Intel 8255

Step 2:- Initialize count as 04

Step 3:-Read the input by using . I5&I4.

Step 4:- Input data ANDed with 30H, then get I5&I4 values

Step 5:- By manipulating the data get the multiplexer output

Step 6:- Now send the value to the portA as output

Step 7:-In this way,Repeat the steps 2to5,get the Mux output for any input combination

.

PROGRAM REQUIREMENTS:

Intel 8255: Port declaration:
Input port: Port B

Out Put Port: Port A:

Control word register:

D7	D6	D5	D4	D3	D2	D1
1	0	0	0	0	1	0

= 82H

ASSEMBLY LANGUAGE PROGRAM BEFORE EXECUTION:

lable	Mnemonic	Operand	Comments
	MOV	DX,0FFC6	Load address of CWR(Control word register) Into DX register
	MOV	AL,82	Send control word to AL
	OUT	DX,AL	Control word to CWR of 8255
Back	MOV	CL,04	Initialize the count
	MOV	DX,0FFC2	Load address of port B into DX
	IN	AL,DX	Read data from port A
	MOV	BL,AL	Input data load into BL
	AND	AL,30	Input data ANDed with 30h to get desired select lines
	ROR	AL,CL	Rotate input data to the right by count
	MOV	CL,AL	After rotation the value stored at CL
	MOV	AL,BL	Present input data load into CL
	ROR	AL,CL	AL rotated right by CL times
	AND	AL,01	AL value ANDed with 01h
	MOV	DX,0FFC0	Load address of port A into DX
	OUT	DX,AL	Send AL data to portA
	JMP	BACK	If required for another input conditions go to back

EXPECTED RESULTS:

Select lines		inputs				output
S1	S0	I3	I2	I1	I0	(X)
0	0					I0
0	1					I1
1	0					I2
1	1					I3

ASSEMBLY LANGUAGE PROGRAM AFTER EXECUTION:

Address	Opcode	Mnemonic	Operand	Comments

RESULTS:

Select lines		inputs				output
S1	S0	I3	I2	I1	I0	O0
0	0	X	X	X	1	
0	0	X	X	X	0	
0	1	X	X	1	X	
0	1	X	X	0	X	
1	0	X	1	X	X	
1	0	X	0	X	X	
1	1	1	X	X	X	
1	1	0	X	X	X	

Note: outputs in the form of LEDs : '1' means LED -ON
'0' means LED-OFF

CONCLUSION:

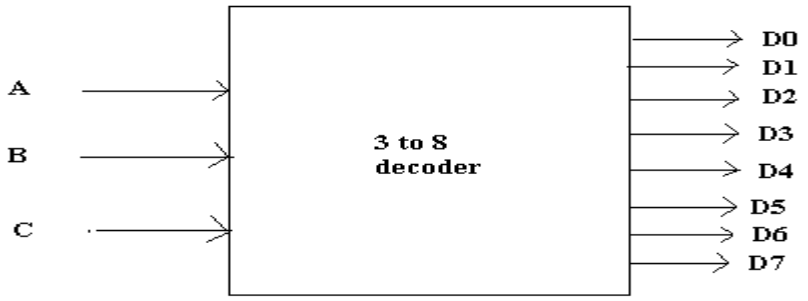


Fig: Block diagram of 3 to8 decoder

ALGORITHM:

Step 1. Send control word (to make port B as input port and port A as output port)to CWR of Intel 8255
Step 2.Initialize port A as output port
Step 3 .Read data byte from port B
Step 4.to get the desired output masking remaining all bit except required one by using AND operation
Step 5.Load accumulator with 01h
Step 6. Rotate accumulator by CL times
Step 7.To get the desired output for each input repeat the above procedure
Step 8.Stop

ASSEMBLY LANGUAGE PROGRAM BEFORE EXECUTION:

LABEL	Mnemonic	operand	Comments
	MOV	AL,82	Load ALwith 82
	MOV	DX,0FFC6	Load address of CWR in to DX register
	OUT	DX,AL	Send control word to CWR(control word register of 8255)
UP	MOV	DX,0FFC2	Load address of port into DX
	IN	AL,DX	read data from port B
	AND	AL,07	to get desired output,mask remaining bits
	MOV	CL,AL	load CL regiester with accumilator value
	MOV	AL,01	load accumilator with 01
	ROL	AL,CL	rotate accumilator with cl times
	MOV	DX,0FFC0	initalise port a as out put port
	OUT	DX,AL	write data into port A
	JMP	UP	for every desired output do the step

EXPECTED RESULTS:

INPUT			OUTPUT							
A	B	C	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

ASSEMBLY LANGUAGE PROGRAM AFTER EXECUTION:

ADDRESS	OPCODE	MNEMONIC	OPERAND	COMMENTS

RESULTS:

Input			Output							
I1	I2	I0	O7	O6	O5	O4	O3	O2	O1	O0
(Switch status)			(LED statuas)							
0	0	0								
0	0	1								
0	1	0								
0	1	1								
1	0	0								
1	0	1								
1	1	0								
1	1	1								

CONCLUSION:

ALGORITHM:

Step 1:- Load DX register with CWR address 0013h
Step 2:- Load the accumulator with control word 82h(specifies portA:o/p port B :i/p)
Step 3:- Load the control word into CWR (control word register)
Step 4:- Load the DX register with 0011(address of port B)
Step 5:- Read input data from port B
Step 6:- Perform NOT operation on AL
Step 7:- Add the content of AL with 01,get the 2's complement of the number
Step 8: - Repeat the procedure for different input from step 5 to step7

ASSEMBLY LANGUAGE PROGRAM BEFORE EXECUTION:

Lable	Mnemonic	Operand	Comments
	MOV	DX,0FFC6	Load the address of CWR in DX
	MOV	AL,82	Load control word into AL
	OUT	DX,AL	Send control word to CWR
START	MOV	DX,0FFC2	Load the address of Port B into DX
	IN	AL,DX	Read input data
	NOT	AL	NOT with i/p data
	ADD	AL,01	Add 01h to the contents AL
	MOV	DX,0FFC0	Copy address of port A into DX
	OUT	DX,AL	Send the results to Port A
	JMP	START	Again start the program for new data

EXPECTED RESULTS:

Input	Output
0 0 0 1	1 1 1 1
0 0 1 0	1 1 1 0
0 0 1 1	1 1 0 1

ASSEMBLY LANGUAGE PROGRAM AFTER EXECUTION:

Address	Opcode	Mnemonic	Operand	Comments

RESULTS:

Input
(switch status)

Output
(LED status)

I7	I6	I5	I4	I3	I2	I1	I0		O7	O6	O5	O4	O3	O2	O1	O0
0	0	0	0	0	0	0	1									
0	0	0	0	0	0	1	1									
1	0	1	1	1	1	0	0									

CONCLUSION:

18. DIGITAL TO ANALOG CONVERTER INTERFACING USING INTEL 8255

AIM : Write an ALP in 8086 to generate the following wave forms by interfacing DAC(Digital to Analog converter module) to 8086 microprocessor through Intel 8255.

i) Square wave form

ii) Triangular wave form

APPARATUS:

1) 8086 microprocessor kit
2) DAC interfacing module
3) Power supply +5v dc
4) Key board

SPECIFICATIONS:

1. Voltage specifications: +12v,-12v&GND from PS-III
2. Port A& port B are connected to chanel1`&chanel 2
3. Reference voltage =8v

CIRCUIT DIAGRAM:

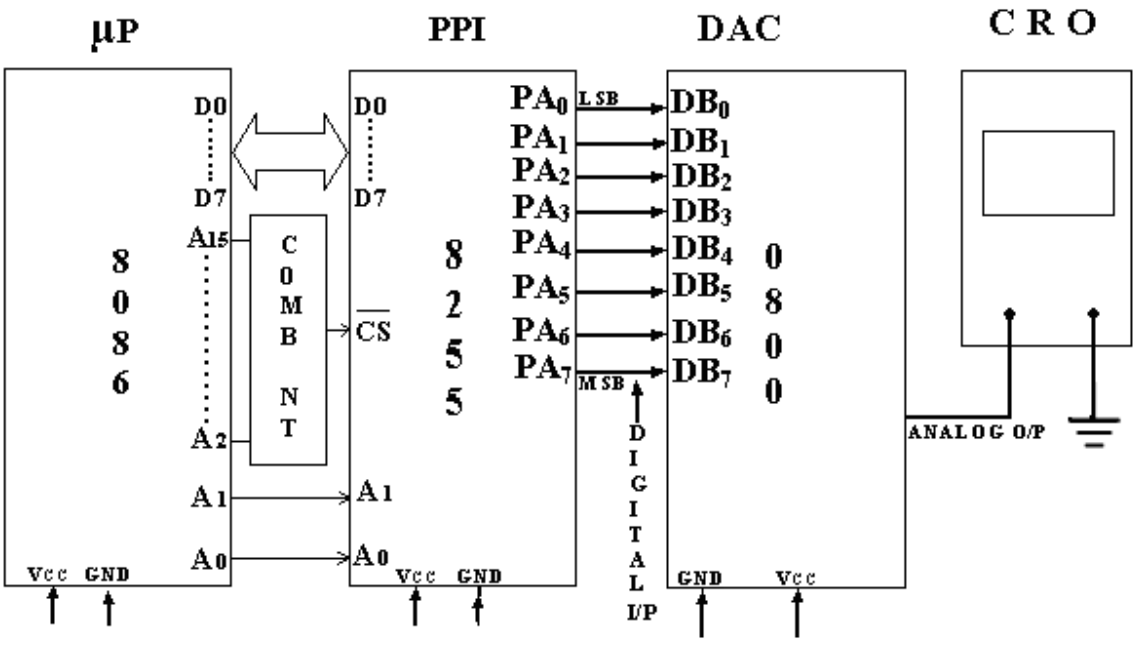


Fig: Interfacing DAC module to 8086 microprocessor

CIRCUIT DESCRIPTION:

Port A and Port B are connected to channel 1 and channel 2 respectively. A reference voltage of 8V is generated using IC 723 and is given to Vref points of the DAC 0800. The standard output voltage will be 7.98 when ff outputted and will be 0V when 00 is outputted. The output of DAC0800 is fed to the operational amplifier to get the final output as out and Y out

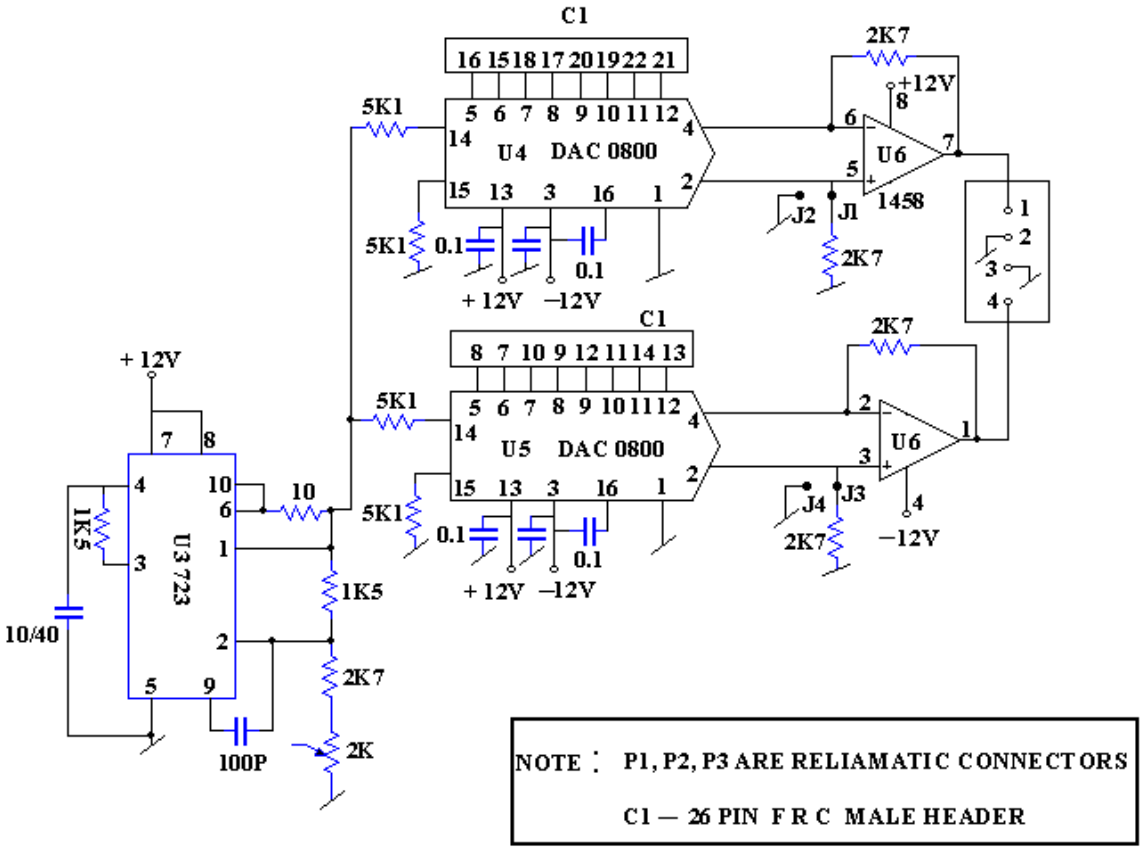


Fig: Internal diagram of DAC module

ALGORITHM:

To generate rectangular wave form with 50% duty cycle

Step1: write control word in the control register(to make all ports O/P ports)
Step2: Initialize the AL register with 00h equivalent digital data
Step3: send digital data to the DAC as input
Step4: call the delay sub program (ON time)
Step5: initialize AL register with ff h value, send the digital data to the DAC as input
Step6: call the delay subprogram three times(off time)
Step7: repeat the step2 to step6 to get continuous wave form

PROGRAM REQUIREMENTS:-

Intel 8255: Port declaration:

Port A, Port B, Port C: O/P Ports

Control word format:

D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	0	0	0	0	0

=80H

ASSEMBLY LANGUAGE PROGRAM BEFORE EXECUTION:

Lable	Mnemonic	Operand	Comments
	MOV	AL,80	Load AL with 80
	OUT	0FFC6,AL	Send AL data to CWR of 8255
UP	MOV	AL,00	Load AL with 00
	OUT	0FFC0,AL	Send AL data to port A
	CALL	1025	call the delay program
	MOV	AL,FF	Load AL with FFh
	OUT	0FFC0,AL	Send AL data to given port
	JMP	UP	Jump to up

Delay program

	MOV	CX,0400	Load CX with given value
UP	DEC	CX	Decrement CX value by 1
	JNZ	UP	Jump to up if no zero
	RET		

EXPECTED RESULTS:

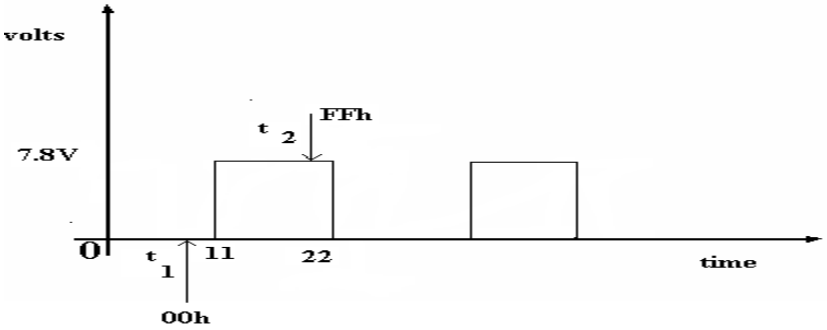


Fig : square wave

t1 = off time =11ms
t2= ON time=11 ms

% duty cycle

$$= \frac{\text{ON time}}{\text{ON time+ OFF time}}$$

$$\frac{11\text{ms}}{(11+11)\text{ms}} = 1/2$$

F=1/t

= 1/22ms

=45 Hz

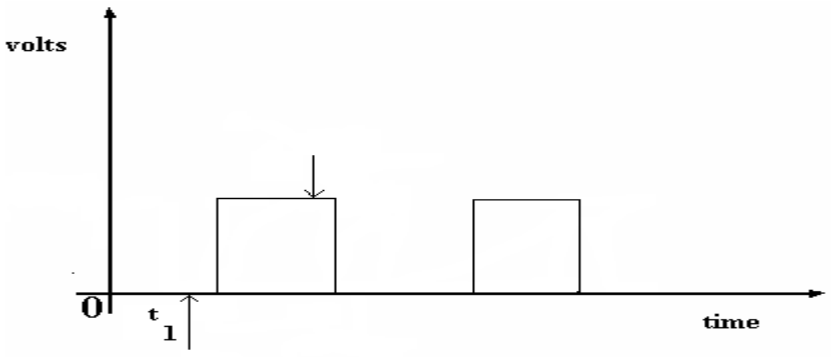
ASSEMBLY LANGUAGE PROGRAM AFTER EXECUTION:

ADDRESS	OPCODE	MNEMONIC	OPERAND	COMMENTS

DELAY PROGRAM

ADDRESS	OPCODE	MNEMONIC	OPERAND	COMMENTS

RESULTS :



250 Hz Wave form generated

ALGORITHM : To generate triangular wave form

Step1: Initialize the control register with control word
Step2: Place the 00 in the AL register
Step3: Send the register (AL) data to input of DAC module
Step4: Increment the register data by one
Step5: Compare the increment data with the FF if it is not matches go to the step2
Step6: Send FF digital data to the DAC i\p via port A
Step7: Decrement digital data by one send to portA
Step8: Repeat the step7 until data reaches to 00h
Step9: if equal then repeat the steps 2 to 8 to get

PROGRAM REQUIREMENTS:

Port declaration:

Port A, Port B, Port C: O/P Ports

Control word format:

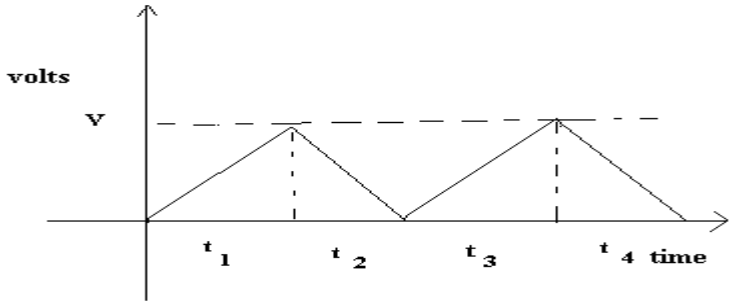
D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	0	0	0	0	0

=80H

ASSEMBLY LANGUAGE PROGRAM BEFORE EXECUTION:

Lable	Mnemonic	Operand	Comments
	MOV	AL,80	Load AL with 80(control word specifies all ports out put ports)
	OUT	0FFC6,AL	send AL data to the CWR of 8255
Back	MOV	AL,00	Clear AL register
up	OUT	0FFC0,AL	send AL data to given Port A
	INC	AL	Increment AL by 1
	CMP	AL,FF	Compare AL data with FF
	JNZ	up	Jump if no zero
	OUT	0FFC0,AL	send AL data to given port
up1	DEC	AL	Clear AL register
	OUT	0FFC0,AL	send AL data to given port
	CMP	AL,00	Compare AL data with 00
	JNZ	up1	Jump if no zero
	JMP	Back	jump

EXPECTED RESULTS:



t1 =t3 **t=t1 +t2**

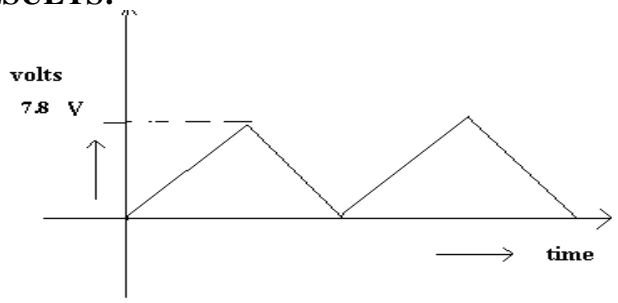
t2=t4

Fig: Triangular Wave Form

ASSEMBLY LANGUAGE PROGRAM AFTER EXECUTION:

Address	Opcode	Mnemonic	Operand	Comments

RESULTS:



t1=5.5ms
t2=5.5ms
t =t1+t2=11ms
f = 1/11ms
=90HZ

Fig: Triangular Wave Form

19. MULTIPLICATION OF TWO 8 BIT NUMBERS

AIM: Write an ALP to perform multiplication of two 8 bit numbers using Repetitive addition by 8051 micro controller

APPARATUS:

1. 8051 microcontroller kit
2. computer
3. Power supply +5v dc
4. Key board

ALGORITHM:

Step 1: Start the program
Step 2: clear the accumulator by sending 00 to register A
Step 3: R0 register is loaded with address of 10h
Step 4: R1 register is assigned with data of 05h
Step 5: Accumulator is added with 03h
Step 6: decrements the value of R1 and check whether it is zero or not.
Step 7: If it is zero go next step, if not goo to specified location.
Step 8: move the result of program to address of R0.

ASSEMBLY LANGUAGE PROGRAM BEFORE EXECUTION:

Label	Mnemonic	Operand	Comments
	MOV	A, ,#00	Clear accumulator
	MOV	R0,#10	Load R0 with value 10h
	MOV	R1,#05	Load R1 with value 05h
UP	ADD	A,#03	Add 03 to accumulator
	DJNZ	R1,UP	If JNZ =1 go to up, else go for next address
	MOV	@R0,A	Copy result to R0 location
	LCALL	0003	End of the program

ASSEMBLY LANGUAGE PROGRAM AFTER EXECUTION:

Address	Opcode	Mnemonic	Operand	Comments

RESULTS:

Acc R0 R1 R2

CONCLUSION:

20. COMPLEMENT THE NUMBER

AIM: Write an ALP to compliment the given 8 bit number using 8051 micro controller

APPARATUS:

1. 8051 microcontroller kit
2. computer
3. Power supply +5v dc
4. Key board

ALGORITHM:

Step1: Start the program
Step 2: clear the accumulator by sending 00 to register A
Step 3: R0 register is loaded with address of 23
Step 4: R1 register is assigned with data of 70h
Step 4: R2 register is assigned with data of 10h
Step 5: compliment the accumulator
Step 6: decrements the value of R2 and check whether it is zero or not.
Step 7: If it is zero go next step, if not goo to specified location.
Step 8: decrements the value of R1 and check whether it is zero or not.
Step 9: If it is zero go next step, if not goo to specified location.
Step 10: move the result of program to address of R0.

ASSEMBLY LANGUAGE PROGRAM BEFORE EXECUTION:

Label	Mnemonic	Operand	Comments
	MOV	A ,#00	Clear accumulator
	MOV	R0 ,#23	Load address of 23 with R0
	MOV	R1,#10	Load value of R1 with 10
Up2	MOV	R2,#70	Load value of R2 with 70
Up1	CPL	A	Compliment A
	DJNZ	R2,UP1	Decrement R2, if JNZ=1 go to up1
	DJNZ	R1,UP2	Decrement R1,if JNZ=1 go to up2
	MOV	@R0,A	Move result to R0
	LCALL	0003	End of the program

ASSEMBLY LANGUAGE PROGRAM AFTER EXECUTION:

Address	Opcode	Mnemonic	Operand	Comments

RESULTS:

Acc R0 R1 R2

CONCLUSION: