## LIST OF EXPERIMETS

**Part I: 8086 Programs**

1. Data Transfer Operations ( MOV & XCHG )
2. Arithmetical Operations ( ADD,ADC,SUB,SBB,DAA,AAA)
3. Logical Operations ( AND,OR,XOR,Shift, Rotate)
4. Convert Packed BCD to unpacked BCD and vice versa (8 bit and 16 bit BCD numbers)
5. Sorting (Ascending & Descending Order)
6. Square root of a given number
7. Display string using MASM
8. String Comparison

**Part II: 8086 Interfacing**

9. 8255 Programmable Peripheral Interfacing
10. Stepper motor Interfacing
11. DAC Interfacing (Square, Saw tooth, Triangular)
12. Traffic light control

| DATA TRANSFER OPERATIONS | EXPT. NO : 1 |
|---|---|
| | DATE : |

**MOVE:**

```
 CODE SEGMENT
ASSUME CS:CODE
START: XOR AX, AX
        MOV BL, AL
        MOV CL, AL
        MOV SI, 2000H
        MOV CL, [SI]
        MOV SI, 3000H
        MOV DI, 4000H
L1:     MOV AL, [SI]
        MOV [DI], AL
        INC SI
        INC DI
        LOOP L1
        INT 3
CODE ENDS
END START
```

**XCHG:**

```
 CODE SEGMENT
ASSUME CS:CODE
START:  XOR AX, AX
        MOV BL, AL
        MOV CL, AL
        MOV SI, 2000H
        MOV CL, [SI]
        MOV SI, 3000H
        MOV DI, 4000H
```

```
L1:   MOV AL, [SI]
      MOV BL, [DI]
      XCHG AL, BL
      MOV [SI], AL
      MOV [DI], BL
      INC SI
      INC DI
      LOOP L1
      INT 3
CODE ENDS
END START
```

| ARITHEMETIC OPERATIONS | EXPT. NO : 2 |
|---|---|
| | DATE : |

1. AIM :Write 8086 program to implement Multi byte addition.

2. COMPONENTS & TOOLS REQUIRED :

2.1 Microprocessor 8086 Trainer Kit,

2.2 Power supply (5V, 3A)

3. THEORY: Generally 8-bits are called a byte, 16-bits are called a word, 32-bits are called a double word. The data more than 4 bytes is called a multi byte. Here, we are adding two multi bytes which are stored at locations 3000H and 3006H. By using the instruction ADC, we can add byte by byte.

4. ALGORITHM:

Following steps were required to implement Multi byte addition.

1. SI ← 3000H
2. DI ← 3006H
3. BX ← 2050H
4. Clear AX register
5. Clear CX register
6. Move the no.of element to CX register
7. AL ← [SI]
8. AL ← AL + [DI] with carry
9. Store the partial result in [BX]
10. Increment SI, DI and BX
11. Decrement counter
12. If

    counter is not equal to zero,

    then

    jump to step 7

13. If

carry flag is not set,

then

jump to step 15

14. Increment AH register to store the carry
15. Terminate the program.

**a. PROGRAM: ( 8 BIT ADDITION)**

```
CODE SEGMENT
ASSUME CS:CODE
START:      XOR AX, AX
            MOV SI, 3000
            MOV AL, [SI]
            INC SI
            MOV BL, [SI]
            ADD AL, BL
            INC SI
            MOV [SI], AL
            JC   L1
            INC SI
            MOV [SI], AH
            INT 3
L1          INC AH
            INC SI
            MOV [SI], AH
            INT 3
CODE ENDS
END START
```

**.b. PROGRAM: ( 16 BIT ADDITION)**

```
 CODE SEGMENT
ASSUME CS:CODE
START:      XOR AX, AX
            MOV DX, AX
            MOV SI, 3000
            MOV AX, [SI]
            ADD SI, 02
            MOV BX, [SI]
            ADD AX, BX
            ADD SI, 02
            MOV [SI], AX
```

```
                JC    L1
                ADD SI, 02
                MOV [SI], DX
          L1:   INC DX
                ADD SI, 02
                MOV [SI], DX
                INT 3
CODE ENDS
END START
```

**c. PROGRAM: ( Multi byte Addition)**

```
CODE SEGMENT
ASSUME  CS:CODE
START:
                MOV     SI, 3000H
                MOV     DI, 3006H
                MOV     BX, 2050H
                XOR     AX, AX
                MOV     CX, AX
                MOV     CL, 06H
          L1:   MOV     AL, [SI]
                ADC     AL, [DI]
                MOV     [BX], AL
                INC     SI
                INC     DI
                INC     BX
                DEC     CL
                JNZ     L1
                JNC     L2
                INC     AH
          L2:   MOV     [BX], AH
                INT     3
CODE ENDS
END START
```

6. EXPECTED RESULTS:

Example :

| INPUT | OUTPUT |
|---|---|
| [3000] = 56 | [2050] = 12 |
| [3001] = 78 | [2051] = 0F1 |
| [3002] = 9A | [2052] = 0F0 |
| [3003] = 0BC | [2053] = 9A |
| [3004] = 0DE | [2054] = 0CF |
| [3005] = 0F0 | [2055] = 8B |
| | [2056] = 01 (CARRY) |
| [3006] = 0BC | |
| [3007] = 78 | |
| [3008] = 56 | |
| [3009] = 0DE | |
| [300A] = 0F0 | |
| [300B] = 9A | |

**d. PROGRAM: ( 8 BIT SUBTRACTION)**

```
CODE SEGMENT
ASSUME  CS:CODE
START:          XOR AX, AX
                MOV SI, 3000
                MOV AL, [SI]
                INC SI
                MOV BL, [SI]
                SUB AL, BL
                INC SI
                MOV [SI], AL
                JC    L1
                INC SI
                MOV [SI], AH
                INT 3
          L1    INC AH
                INC SI
                MOV [SI], AH
                INT 3
CODE ENDS
END START
```

**e. PROGRAM: ( 16 BIT SUBTRACTION)**

```
CODE SEGMENT
ASSUME  CS:CODE
START:          XOR AX, AX
                MOV DX, AX
                MOV SI, 3000
                MOV AX, [SI]
                ADD SI, 02
                MOV BX, [SI]
                SUB AX, BX
                ADD SI, 02
                MOV [SI], AX
                JC   L1
                ADD SI, 02
                MOV [SI], DX
          L1:   INC DX
                ADD SI, 02
                MOV [SI], DX
                INT 3
CODE ENDS
END START
```

**f. PROGRAM: (Multi byte subtraction)**

```
CODE SEGMENT
ASSUME  CS:CODE
    START:
            MOV     SI, 3000H
            MOV     DI, 3006H
            MOV     BX, 2050H
            XOR     AX, AX
            MOV     CX, AX
            MOV     CL, 06H
       L1:  MOV     AL, [SI]
            SBB     AL, [DI]
            MOV     [BX], AL
            INC     SI
```

```
        INC     DI
        INC     BX
        DEC     CL
        JNZ     L1
        JNC     L2
        INC     AH
   L2:  MOV     [BX], AH
        INT     3
CODE ENDS
END START
```

**g. 16-bit multiplication**

```
CODE SEGMENT
ASSUME  CS:CODE
   START:
            XOR     AX, AX
            MOV     DX, AX
            MOV     AX, [2500H]
            MOV     BX, [2502H]
            MUL     BX
            MOV     [2050H], AX
            MOV     [2052H], DX
            INT     3

CODE ENDS
END START
```

**g. 16-bit division**

```
CODE SEGMENT
ASSUME  CS:CODE
START:
                    XOR AX,AX
                    XOR DX,DX
```

```
        MOV BX,AX
        MOV     AX, [2500H]
        MOV     DX, [2502H]
        MOV     BX, [2504H]
        DIV     BX

        MOV     [2050H], AX
        MOV     [2052H], DX
        INT     3
CODE ENDS
END START
```

| LOGICAL OPERATIONS | EXPT. NO :3 |
|---|---|
| | DATE : |

1. AIM :Write 8086 program to implement Multi byte Subtraction.

2. COMPONENTS & TOOLS REQUIRED:

2.1 Microprocessor 8086 Trainer Kit,

2.2 Power supply (5V, 3A)

3. THEORY: Shift and rotate operations are the bit wise logical operations. If we want to shift one bit only directly we can write SHL Al,01H. If we want to shift more than one bit we have to use the counter CL. The same case is applicable for rotate operations. We can use these instructions when we want to check the carry flag and when we want to check particular bit in the specified register. There are types of shift operations and rotate operations both with carry and with out carry. For ex RCR means Rotate with carry.

4. ALGORITHM:

1. AX ← [2500H]
2. Count will be stored in count register
3. Rotate left to the contents of accumulator
4. Store the result
5. Terminate the program

**ROTATE LEFT:**

```
CODE SEGMENT
ASSUME  CS:CODE
START:
                MOV     AX, [2500H]
                MOV     CL, 04
                ROL     AX, CL
                MOV     [2050H], AX
                INT     3
CODE ENDS
END START
```

EXPECTED RESULTS:

Example :

| INPUT | OUTPUT |
|---|---|
| [2500] = 12 | [2050] = 41 |
| [2501] = 34 | [2051] = 23 |

**ROTATE RIGHT:**

```
CODE SEGMENT
ASSUME  CS:CODE
START:
                    MOV     AX, [2500H]
                    MOV     CL, 04
                    ROR     AX, CL
                    MOV     [2050H], AX
                    INT     3
CODE ENDS
END START
```

**SHIFT LEFT AND SHIFT RIGHT:**

```
CODE SEGMENT
ASSUME  CS:CODE
START:
                    MOV     AX, [2500H]
                    MOV     CL, 04
                    SHL/SHR AX, CL
                    MOV     [2050H], AX
                    INT     3
CODE ENDS
END START
```

| PACKED BCD TO UNPACKED BCD | EXPT. NO : 4 |
|---|---|
| | DATE : |

1. AIM :Write 8086 program to implement Multi byte Subtraction.

2. COMPONENTS & TOOLS REQUIRED:

2.1 Microprocessor 8086 Trainer Kit,

2.2 Power supply (5V, 3A)

3. THEORY:

BCD means Binary Code Decimal. Binary Coded Decimal numbers are the numbers which are decimal numbers from 0 to 9, but they are represented in binary form.

Generally we deal with decimal numbers rather than Hexa decimal numbers. So it is necessary to know how to convert from Packed BCD to unpacked BCD. When we are converting we use masking operations using AND instructions.

4. ALGORITHM:

1. AL ← [2050H]
2. BL ← AL
3. mask lower nibble
4. Roatate BL 4 times
5. Store the partial result in 2060 location
1. mask upper nibble
2. store the result in 2061 location.
3. terminate the program.

5.a. PROGRAM: (Packed to un packed)

| ADDR | OPCODE | LABEL | MNEMONICS | OPERANDS | COMMENTS |
|---|---|---|---|---|---|
| 2000 | A0 50 20 | | MOV | AL, [2050H] | ; Move 2050h contents to AL |
| 2003 | 88 C3 | | MOV | BL, AL | ; Move AL contents to BL |
| 2005 | 80 E3 F0 | | AND | BL, 0F0H | ; AND BL contents with 0F0 and store result in BL |

| | | | | |
|---|---|---|---|---|
| 2008 | B1 04 | MOV | CL, 04H | ; Load count register with 04h |
| 200A | D2 CB | ROR | BL, CL | ; Rotate right BL by 04h times |
| 200C | 88 1E 60 20 | MOV | [2060H], BL | ; Moving contents of BL to 2060H |
| 2010 | 24 0F | AND | AL, 0FH | ; AND AL with 0FH and store result in AL |
| 2012 | A2 61 20 | MOV | [2061H], AL | ; Moving contents of AL to 2061H |
| 2015 | CC | INT | 3 | ; Breaking interrupt |

6. EXPECTED RESULTS:

Example :

| INPUT | OUTPUT |
|---|---|
| [2050] = 59 | [2060] = 05 |
| | [2061] = 09 |

**5.b. PROGRAM: (un packed to packed)**

```
XOR AL, AL
MOV SI, 3000
MOV AL, [SI]
INC SI
MOV BL, [SI]
MOV CL, 04
ROR BL, CL
ADD AL, BL
INC SI
MOV [SI], AL
INT 3
```

RESULT:

| INPUT | OUTPUT |
|---|---|
| [3000] = 09 | [3002] = 59 |
| [3001] = 05 | |

7. CONCLUSION: The Packed BCD to unpacked BCD is performed by using 8086 instructions.

8. PRECAUTIONS:

1. Give connections carefully such as Keyboard, Flat Ribbon cable if any and power supply etc.
2. Switch on the power supply only after giving all the connections.
3. Handle the Trainer kits carefully.

9. VIVA -VOCE QUESTIONS:

1. What are the commands to perform this program on the kit?
2. What numbers the BCD numbers consists of?
3. Give an example for Packed BCD number?
4. Give an example for unpacked BCD number?
5. What is the difference between AND and TEST instruction?
6. What is the procedure to convert from Packed BCD to Unpacked BCD?
7. What is the difference between BCD and Hex number?

| SORTING | EXPT. NO : 5 |
|---|---|
| | DATE : |

1. AIM :Write an ALP to implement sorting .

2. COMPONENTS & TOOLS REQUIRED :
   2.1 Computer system,
   2.2 MASM / TASM SOFTWARE

3. THEORY:

The following instructions are used for the signed and unsigned operations:

| | SIGNED | UNSIGNED |
|---|---|---|
| ASCENDING ORDER | JLE | JBE |
| DESCENDING ORDER | JGE | JAE |

4. PROGRAM: SORTING

```
DATA SEGMENT
      COUNT EQU 2000H
      ADDR EQU 2500H
      DATA ENDS
CODE SEGMENT
 ASSUME CS:CODE,DS:DATA
START:  MOV AX,DATA
       MOV DS,AX
       MOV SI,COUNT
       MOV BL,[SI]
       DEC BL

L3:    MOV CL,BL
```

```
        MOV SI,ADDR

L2:     MOV AL,[SI]
        CMP AL,[SI+1]
        JLE L1
        XCHG [SI+1],AL
        MOV [SI],AL

L1:     INC SI
        LOOP L2
        DEC BL
        JNZ L3
        INT 3
CODE ENDS
END START
```

5. EXPECTED RESULTS:

UNSIGNED NUMBERS:

INPUT: 2500: 05H (NUMBER OF ELEMENTS)

2501: 9A
2502: 08
2503: 93
2504: 34
2505: 01

OUTPUT: 2501: 01
2502: 08
2503: 34
2504: 93
2505: 9A

SIGNED NUMBERS:

INPUT: 2500: 05H (NUMBER OF ELEMENTS)

2501: 9A
2502: E0
2503: FF
2504: 34
2505: 01

OUTPUT: 2501: 9A
2502: E0
2503: FF
2504: 01
2505: 34

6. CONCLUSION:

The given numbers are sorted in Ascending order for both signed and unsigned and stored in the same locations.

7. PRECAUTIONS:

1. Give connections carefully such as Keyboard, Flat Ribbon cable if any and power supply etc.
2. Switch on the power supply only after giving all the connections.
3. Handle the Trainer kits carefully.

8. VIVA-VOCE QUESTIONS:

1. what is the instruction to jump instruction in implementing the sorting of signed numbers in ascending order?
2. what is the instruction to jump instruction in implementing the sorting of unsigned numbers in ascending order?
3. Give names of different sorting methods?
4. What is the difference between XCHG and CMP?
5. What is the difference between CMP and TEST?

| SQUARE ROOT OF A GIVEN NUMBER | EXPT. NO :6 |
|---|---|
| | DATE : |

1. AIM :Write an ALP to determine square root of a given number .

2. COMPONENTS & TOOLS REQUIRED :
   2.1 Computer system,
   2.2 MASM / TASM SOFTWARE

4. PROGRAM:

```
DATA SEGMENT
      NUMBER DB 64H
      RESULT DB ?
DATA ENDS
CODE SEGMENT
 ASSUME CS:CODE,DS:DATA
START:  MOV AX,DATA
       MOV DS,AX
       XOR AX,AX
      XOR CX,CX
      MOV CL,NUMBER
      MOV BL,01H
L2: CMP CL,00H
     JZ L1
     SUB CL,BL
     INC AL
     ADD BL,02H
     JMP L2
L1: MOV RESULT,AL
      INT 21H
CODE ENDS
END START
```

| DISPLAY STRING ON THE MONITOR | EXPT. NO : 7 |
|---|---|
| | DATE : |

1. AIM :Write an ALP to implement Display string on the monitor.

2. COMPONENTS & TOOLS REQUIRED :
   2.1 Computer system,
   2.2 MASM / TASM SOFTWARE

3. PROGRAM:

```
DATA SEGMENT
        MESSAGE DB ' COLLEGE $ '
 DATA ENDS

CODE SEGMENT
        ASSUME CS: CODE, DS: DATA
START:  MOV AX, DATA
        MOV DS, AX                  ;Initialize DS register
        LEA DX, MESSAGE             ;EA of message will move to DX register
        MOV AH, 09H                 ;to display a string
        INT 21H                     ;DOS Interrupt service routine
        MOV AH, 4CH                 ;return to DOS prompt
        INT 21H
CODE ENDS                           ; end of code segment

END START
```

1. EXPECTED RESULTS:

GOOD MORNING should be displayed on the screen.

2. CONCLUSION:

String "Good morning" is displayed on the screen.

6. PRECAUTIONS:

1. Give connections carefully such as Keyboard, Flat Ribbon cable if any and power supply etc.
2. Switch on the power supply only after giving all the connections.
3. Handle the Trainer kits carefully.

7. VIVA QUESTIONS:

a. What the function code '0A' indicates?
b. What the function code '09' indicates?
c. What the function code '01'indicates?
d. In which register we have to move the function codes?
e. What is the interrupt that we have to use for DOS operations?
f. Which register we have to use to store the address of the source?
g. Is there any need to initialize the DS registers?
h. How many segments the 8086 memory is divided?

| STRING COMPARISION | EXPT. NO : 8 |
|---|---|
| | DATE : |

1. AIM :Write an ALP to implement String comparison (Pass word checking) .

2. COMPONENTS & TOOLS REQUIRED :
   2.1 Computer system,
   2.2 MASM / TASM SOFTWARE

3. PROGRAM:

```
DATA SEGMENT
        PASSWORD DB 'LBRCE'
        NEWSTR DB 'LBECE' / 'LBRCE'
        CNT DB 05H
        STR1 DB 'THE GIVEN STRING IS VALID$'
        STR2 DB 'THE GIVEN STRING IS INVALID$'
DATA ENDS

CODE SEGMENT
ASSUME CS: CODE, DS: DATA, ES: DATA
START:  MOV AX, DATA
        MOV DS, AX
        MOV ES, AX
        MOV CL, CNT
        LEA SI, PASSWORD
        LEA DI, NEWSTR
        REP CMPSB
        JNZ L1
        LEA DX, STR1
        JMP L2
L1:     LEA DX, STR2
L2:     MOV AH, 09H
        INT 21H
```

```
        INT 3
CODE ENDS
END START
```

4. EXPECTED RESULTS:

INPUT: LBRCE

LRBCE

OUTPUT: THE GIVEN STRING IS INVALID

INPUT: LBRCE

LBRCE

OUTPUT: THE GIVEN STKRING IS VALID

5. CONCLUSION:

The given two strings are compared and displayed the GIVEN STRING IS EQUAL if they are equal.

6. PRECAUTIONS:

1. Give connections carefully such as Keyboard, Flat Ribbon cable if any and power supply etc.
2. Switch on the power supply only after giving all the connections.
3. Handle the Trainer kits carefully.

7. VIVA-VOCE QUESTIONS:

1. what we store in data segment?
2. What the function code '0A' indicates?
3. What the function code '09' indicates?
4. What the function code '01'indicates?
5. In which register we have to move the function codes?
6. What is the interrupt that we have to use for DOS operations?
7. Which register we have to use to store the address of the source?
8. Is there any need to initialize the DS registers?
9. What is the use of SI and DI registers?

| 8255 PERIPHERAL INTERFACE | EXPT. NO : 9 |
|---|---|
| | DATE : |

1. AIM :Write 8086program to implement 8255 peripheral control interface

2. COMPONENTS & TOOLS REQUIRED :
   2.1 Microprocessor 8086 Trainer Kit,
   2.2 Power supply (5V, 3A)
   2.3. 8255 trainer chip

3. THEORY:

4.PROGRAM:

1) All ports are output ports

```
MOV DX, 0FFE6H
MOV AL,80H
OUT DX,AL
INT 3
```

2) All ports are input ports

```
MOV DX, 0FFE6H
MOV AL, 9BH
OUT DX,AL
INT 3
```

5. EXPECTED RESULTS:

1. CONCLUSION:

7. PRECAUTIONS:

1. Give connections carefully such as Keyboard, Flat Ribbon cable if any and power supply etc.
2. Switch on the power supply only after giving all the connections.

3. Handle the Trainer kits carefully.

8. VIVA-VOCE QUESTIONS:

1. Which registers are used as pointers?
2. On what condition the carry flag is set?
3. What are the commands to perform this program on the kit?
4. What is the difference between conditional and unconditional jump instructions?
5. What are the flags available in 8086?

| STEPPER MOTOR CONTROL | EXPT. NO : 10 |
|---|---|
| | DATE : |

1. AIM :To control the direction of Stepper motor with the help of 8086.

2. COMPONENTS & TOOLS REQUIRED :

2.1 Microprocessor 8086 Trainer Kit,

2.2 Power supply (5V, 3A)

2.3. Stepper motor

3. THEORY:

4.PROGRAM

```
        MOV DX, 0FFE6H
        MOV AL, 80H
        OUT DX, AL
        MOV AL, 88H
        MOV DX, 0FFE0H
L1:     OUT DX, AL
        MOV CX, 0FFFFH
L2:     NOP
        NOP
        NOP
        LOOP L2
        ROR/ROL AL, 1H
        JMP L1
```

6. EXPECTED RESULTS:

3. CONCLUSION:

8. PRECAUTIONS:

1. Give connections carefully such as Keyboard, Flat Ribbon cable if any and power supply etc.

2. Switch on the power supply only after giving all the connections.
3. Handle the Trainer kits carefully.

9. VIVA-VOCE QUESTIONS:

1. Which registers are used as pointers?
2. On what condition the carry flag is set?
3. What are the commands to perform this program on the kit?
4. What is the difference between conditional and unconditional jump instructions?
5. What are the flags available in 8086?

| DAC INTERFACING | EXPT. NO : 11 |
|---|---|
| | DATE : |

1. AIM :Write 8086 program to interface 8255 PPI, generate saw tooth wave, triangular wave and square wave using DAC interfacing.

2. COMPONENTS & TOOLS REQUIRED :

2.1 Microprocessor 8086 Trainer Kit,

2.2 Power supply (5V, 3A)

2.3 DAC interface module

2.4 Power supply (12V)

2.5 Flat ribbon cable

2.6 connector ( 3 or 4 pin)

3. THEORY:

The Dual DAC interface can be used to generate different interesting waveforms using microprocessor. There are two eight-bit digital to analog converters provided, based on DAC 0800. The digital inputs to these DACs are provided through the port A and port B of 8255 used as output ports. The analog output from the DACs are given to operational amplifiers which act as current to voltage converters. Two 10 K ohms pots are provided for the offset balancing of opamps.

The reference voltage needed for the DACs is obtained from a onboard voltage regulator uA723. The voltage generated by this regulator is about 8V The outputs from the DACs vary between 0 to 5V corresponding to values between 00 to FF. Different waveforms can be observed at the opamp outputs depending upon the digital input patterns.

For One Clock = 1 / 8 MHz = 0.125 $\mu$s

To generate 1 ms delay i.e. 1000$\mu$s = 1000 / 0.125 = 8000 Clock Cycles ($C_T$)

| | | | |
|---|---|---|---|
| | MOV CX, N | 4 Cycles | ( $C_0$ ) |
| L1: | NOP | 3 Cycles | |
| | NOP | 3 Cycles | $C_L$ |
| | LOOP L1 | 17 or 5 Cycles | |

$C_T = C_0 + N (C_L) – 12$

$N = C_T – C_0 + 12 / C_L$

= (8000 – 4 + 12) / 23

= 349 = 015D

$C_X$ = 015D

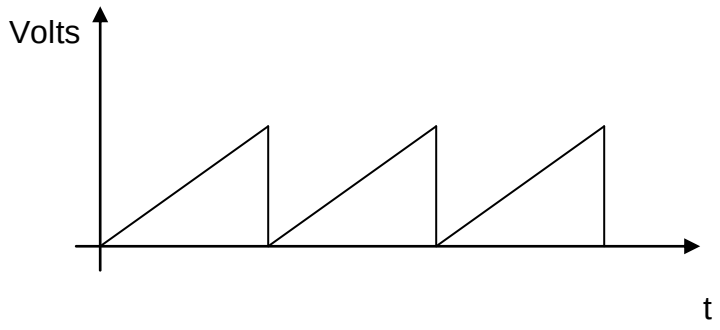
4. PROGRAM:

**1. SAW TOOTH WAVE GENERATION:**

| ADDR | OPCODE | LABEL | MNEMONICS | OPERANDS | COMMENTS |
|---|---|---|---|---|---|
| 2000 | BA E7 FF | | MOV | DX, 0FFE7H | ; Set up 8255 for Mode 0 |
| 2003 | B0 80 | | MOV | AL, 80H | ; Ports A, B |
| 2005 | EE | | OUT | DX, AL | ; |
| 2006 | B0 00 | | MOV | AL, 00H | ; Start with a value of 00h |
| 2008 | BA E1 FF | L1: | MOV | DX, 0FFE1H | ; |
| 200B | EE | | OUT | DX, AL | ; Out $X_{out}$ |
| 200C | FE C0 | | INC | AL | ; Increment AL |
| 200E | EB F8 | | JMP | L1 | ; Repeat forever |

5. EXPECTED RESULTS:

OUTPUT:




Amplitude =

Time period =

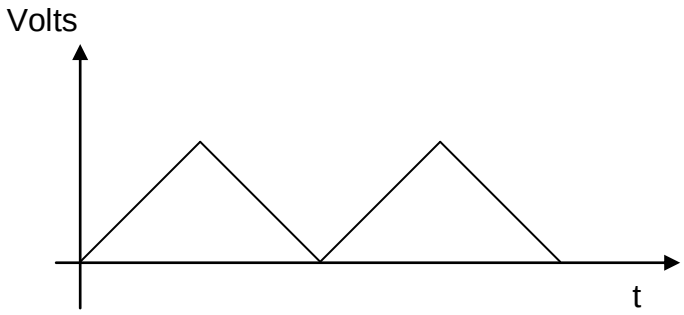
Frequency =

Resolution =

**2. TRIANGULAR WAVE GENERATION:**

| ADDR | OPCODE | LABEL | MNEMONICS | OPERANDS | COMMENTS |
|---|---|---|---|---|---|
| 2000 | BA E7 FF | | MOV | DX, 0FFE7H | |
| 2003 | B0 80 | | MOV | AL, 80H | |
| 2005 | EE | | OUT | DX, AL | |
| 2006 | B0 00 | | MOV | AL, 00H | |
| 2008 | BA E1 FF | | MOV | DX, 0FFE1H | |
| 200B | EE | L1: | OUT | DX, AL | |
| 200C | FE C0 | | INC | AL | |
| 200E | 3C FF | | CMP | AL, FFH | |
| 2010 | 75 F9 | | JNZ | L1 | |
| 2012 | EE | L2: | OUT | DX, AL | |
| 2013 | FE C8 | | DEC | AL | |
| 2015 | 75 FB | | JNZ | L2 | |
| 2017 | EB F2 | | JMP | L1 | |

OUTPUT:




Amplitude =

Time period =

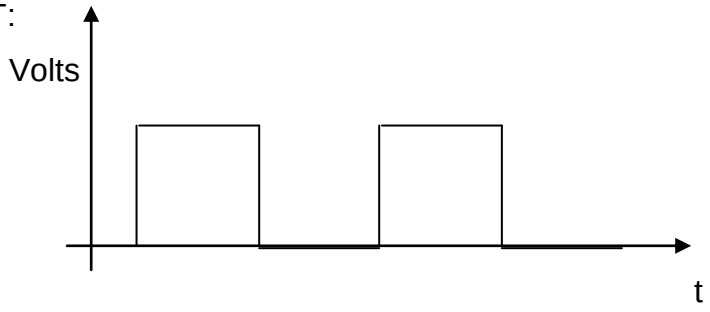
Frequency =

Resolution =

**3. SQUARE WAVE GENERATION:**

| ADDR | OPCODE | LABEL | MNEMONICS | OPERANDS | COMMENTS |
|---|---|---|---|---|---|
| 2000 | BA E7 FF | | MOV | DX, 0FFE7H | |
| 2003 | B0 80 | | MOV | AL, 80H | |
| 2005 | EE | L1: | OUT | DX, AL | |
| 2006 | B0 00 | | MOV | AL, 00H | |
| 2008 | BA E1 FF | | MOV | DX, 0FFE1H | |
| 200B | EE | | OUT | DX, AL | |
| 200C | E8 08 00 | | CALL | DLY | |
| 200F | B0 FF | | MOV | AL, 0FFH | |
| 2011 | EE | | OUT | DX, AL | |
| 2012 | E8 02 00 | | CALL | DLY | |
| 2015 | EB EE | | JMP | L1 | |
| 2017 | B9 5D 01 | DLY: | MOV | CX, 015DH | |
| 201A | 90 | L2: | NOP | | |
| 201B | 90 | | NOP | | |
| 201C | E2 FC | | LOOP | L2 | |
| 201E | C3 | | RET | | |

OUTPUT:




Amplitude =

Time period =

Frequency =

Resolution =

6. CONCLUSION:

The 8255 interfacing is implemented and different waveforms such as square wave, saw tooth and triangular waves are generated.

7. PRECAUTIONS:

1. Give connections carefully such as Keyboard, Flat Ribbon cable if any and power supply etc.
2. Switch on the power supply only after giving all the connections.
3. Handle the Trainer kits carefully.

8. VIVA-VOCE QUESTIONS:

1. Explain OUT and IN instructions?
2. What is the difference between Fixed Port and Variable Port?
3. What are the commands to execute this program?
4. What is Duty Cycle?
5. How can you generate Ramp Waveform?
6. How can you write 2ms delay program?
7. What is the difference between Macro and Procedure?
8. What is the difference between conditional Jump and Unconditional Jump instructions?

| TRAFFIC LIGHT CONTROLLER INTERFACE | EXPT. NO : 12 |
|---|---|
| | DATE : |

1. AIM : Control the traffic light with the help of 8086 processor with 8255 peripheral chip.

2. COMPONENTS & TOOLS REQUIRED :

2.1 Microprocessor 8086 Trainer Kit,

2.2 Power supply (5V, 3A)

2.3. Traffic light controller kit

3. Theory

4. PROGRAM:

TRAFFIC LIGHT CONTROLLER INTERFACE

; The interface is connected over J8 of of trainer

; Traffic system moves from one state to other after a fixed delay

; This program starts at 2000H location

```
        OUTPUT  2500AD

        ORG     2000H
START:  MOV     AL,80H          ; Initialisation of 8255 Mode 0
        MOV     DX,0FFE6H
        OUT     DX,AL           ; All ports as o/p ports
AGAIN:  MOV     SI,2038H        ; Table of port values
NEXTST: MOV     AL,[SI]
        MOV     DX,0FFE0H
        OUT     DX,AL           ; PortA value
        INC     SI
        ADD     DX,2
        MOV     AL,[SI]
        OUT     DX,AL           ; PortB value
        INC     SI
```

```
        ADD     DX,2
        MOV     AL,[SI]
        OUT     DX,AL               ; PortC value
        INC     SI
        CALL    DELAY               ; Calling Delay routine
        CMP     SI,2056H            ; Checking for the end of the data values
        JNZ     NEXTST
        JMP     AGAIN
DELAY:  MOV     CX,0FFH             ; Delay routine
DLY5:   PUSH    CX
        MOV     CX,03FFH
DLY10:  NOP
        LOOP    DLY10
        POP     CX
        LOOP    DLY5
        RET

        ORG     2038H
        PORTVALUES:   DB      10H,81H,7AH           ; State 1
                      DB      44H,44H,0F0H          ; All ambers ON
                      DB      08H,11H,0E5H          ; State 2
                      DB      44H,44H,0F0H          ; All ambers ON
                      DB      81H,10H,0DAH          ; State 3
                      DB      44H,44H,0F0H          ; All ambers ON
                      DB      11H,08H,0B5H          ; State 4
                      DB      44H,44H,0F0H          ; All ambers ON
                      DB      88H,88H,00H           ; State 5
                      DB      44H,44H,0F0H          ; All ambers ON
                      DB      00H                   ; Dummy
```

5. EXPECTED RESULTS:

6. CONCLUSION:

7. PRECAUTIONS:

1. Give connections carefully such as Keyboard, Flat Ribbon cable if any and power supply etc.
2. Switch on the power supply only after giving all the connections.
3. Handle the Trainer kits carefully.

8. VIVA-VOCE QUESTIONS:

2. Which registers are used as pointers?
3. On what condition the carry flag is set?
4. What are the commands to perform this program on the kit?
5. What is the difference between conditional and unconditional jump instructions?
6. What are the flags available in 8086?