**Experiment number:**  1

**Experiment name:** Familiarization with the MDA-8086 Microprocessor Trainer and EMU8086 Microprocessor Emulator

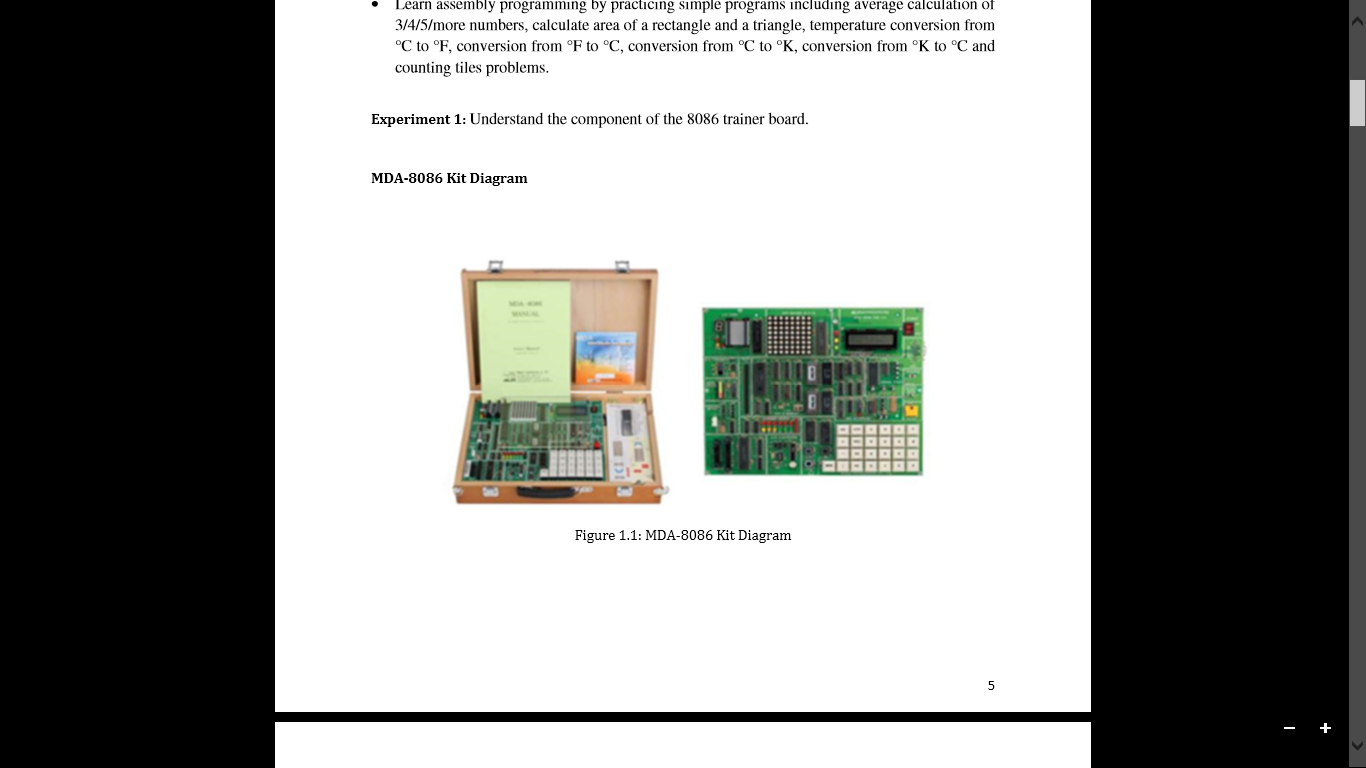
**Objective:** To Understand the component of 8086 trainer board.

• To learn 8086 16-bit Intel Microprocessor, its register and assembly level programming.

**Introduction**: MDA-8086 has high performance 64K-byte monitor program. It is designed for easy function. After power is on, the monitor program begins to work. In addition to all the key function the monitor has a memory checking routine.

Emu8086 is a software emulation of Intel's 8086 microprocessor, and I/O Emulation Kit is a software emulation of a group of hardware devices that can be controlled by Emu8086 virtual central processing unit (CPU).

**Figure:**



**Experiment number:**  2

**Experiment name:** Add, Subtraction, multiplication, Division

**Objective:** To implement assembly language by practicing simple programs like addition, multiplication, subtraction and division

**Code and Output:**

|  |  |  |
| --- | --- | --- |
| Name | Code | Output |
| Add | MOV ax,30  mov bx,20  add ax,bx |  |
| Sub | MOV ax,30  mov bx,20  sub ax,bx |  |
| Mul | MOV ax,30  mov bx,20  mul bx |  |
| div | mov ax,30  mov bx,20  div bx |  |

**Experiment number:**  3

**Experiment name:** Experiment with jmp and loop

**Objective:** To implement the instructions jmp and loop in assembly language

**Code and Output:**

|  |  |  |
| --- | --- | --- |
| jmp | MOV AX, 00 ; Initializing AX to 0  MOV BX, 00 ; Initializing BX to 0  MOV CX, 01 ; Initializing CX to 1  L20:  ADD AX, 01 ; Increment AX  ADD BX, AX ; Add AX to BX  SHL CX, 1 ; shift left CX, this in turn doubles the CX value  JMP L20 ; repeats the statements |  |
| loop | MOV CX, 4 ; Set loop counter to 4  MOV BX, 0 ; Initialize BX (sum) to 0  MOV AX, 1 ; Initialize AX to 1 (value to add)  LOOP\_START: ; Start of the loop  ADD BX, AX ; Add AX to BX (accumulate sum)  JMP CHECK ; Unconditionally jump to the CHECK label  CHECK: ; Label to check loop condition  LOOP LOOP\_START ; Decrement CX and jump to LOOP\_START if CX != 0  HLT ; Halt the program |  |

**Experiment number:**  4

**Experiment name:** Experiment with shift and rotate instructions

**Objective:** To implement shift left and right

To implement Rotate left and rotate right

**Code and Output:**

|  |  |  |
| --- | --- | --- |
| Shift left | mov ax, 20  shl ax,1 |  |
| Shift right | mov ax, 20  shr ax,1 |  |
| Rotate left | mov al, 255  rol al,1 |  |
| Rotate right | mov al, 1  ror al,1 |  |

**Experiment number:**  5

**Experiment name:** Implementing a simple program using assembly language in 8086

**Objective:** To learn to program in assembly language

**Code and Output:**

|  |  |
| --- | --- |
|  |  |
| mov ax,30  mov bx,20  add ax,bx  mov cx,ax  mov ax,5  mov bx,2  mul bx  mov bx,ax  mov ax,cx  div bx |  |

**Experiment number:**  6

**Experiment name:** Determine temperature, area and factorial

**Objective:** To calculate Temperature (Fahrenheit to Celsius)

To calculate Area (Area of Trapezium) using assembly language

To calculate factorial using assembly language

**Code and Output:**

|  |  |  |
| --- | --- | --- |
| Name | Code | Output |
| Temperature  (Fahrenheit to Celsius) | MOV ax,50  mov bx,32  sub ax,bx  mov bx,5  mul bx  mov bx, 9  div bx |  |
| Area of trapezium | MOV ax,5  mov bx,7  add ax,bx  mov bx,2  div bx  mov bx, 3  mul bx |  |
| Factrial | MOV ax,1  mov cx,5  l1: mul cx  loop l1  mov bx,ax  mov ax,1  mov cx,3  l2: mul cx  loop l2  sub bx,ax |  |

**Experiment number:**  7

**Experiment name:** Experiment with Seven (7)-Segments Display

**Objective:** To display 0-9 in Seven-segment display using assembly language

**Code and Output:**

|  |  |  |
| --- | --- | --- |
|  | START: ; Control register turn on  MOV AL, 80H  OUT 1FH, AL  SSD: ; Display digits 0-9 sequentially  ; Display 0  MOV AL, 0C0H  OUT 19H, AL  MOV CX, 0FFFFH  L0: LOOP L0  ; Display 1  MOV AL, 0F9H  OUT 19H, AL  MOV CX, 0FFFFH  L1: LOOP L1  ; Display 2  MOV AL, 0A4H  OUT 19H, AL  MOV CX, 0FFFFH  L2: LOOP L2  ; Display 3  MOV AL, 0B0H  OUT 19H, AL  MOV CX, 0FFFFH  L3: LOOP L3  ; Display 4  MOV AL, 099H  OUT 19H, AL  MOV CX, 0FFFFH  L4: LOOP L4  ; Display 5  MOV AL, 092H  OUT 19H, AL  MOV CX, 0FFFFH  L5: LOOP L5  ; Display 6  MOV AL, 082H  OUT 19H, AL  MOV CX, 0FFFFH  L6: LOOP L6  ; Display 7  MOV AL, 0F8H  OUT 19H, AL  MOV CX, 0FFFFH  L7: LOOP L7  ; Display 8  MOV AL, 080H  OUT 19H, AL  MOV CX, 0FFFFH  L8: LOOP L8  ; Display 9  MOV AL, 090H  OUT 19H, AL  MOV CX, 0FFFFH  L9: LOOP L9  ; Repeat the sequence  JMP SSD  S ENDS  END START |  |

**Experiment number:**  8

**Experiment name:** Experiment with LED Connection Program

**Objective:** To write an assembly code to glow R1, G, Y and R2 in LED Display respectively.

**Code and Output:**

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  | L SEGMENT PARA PUBLIC 'CODE'  ASSUME CS:L  ORG 1000H  START: ; Start of the program  ; Turn on the control register  MOV AL, 80H  OUT 1FH, AL  ; Turn off all LEDs initially  MOV AL, 0FFH  OUT 19H, AL  LED: ; Start of the LED control loop  ; R1 LED turn on  MOV AL, 01H ; Turn on R1 LED (bit 0)  OUT 1BH, AL ; Send the value to port 1BH  MOV CX, 0FFFFH ; Set delay  LR1: LOOP LR1 ; Delay loop  ; G LED turn on  MOV AL, 02H ; Turn on G LED (bit 1)  OUT 1BH, AL ; Send the value to port 1BH  MOV CX, 0FFFFH ; Set delay  LG: LOOP LG ; Delay loop  ; Y LED turn on  MOV AL, 04H ; Turn on Y LED (bit 2)  OUT 1BH, AL ; Send the value to port 1BH  MOV CX, 0FFFFH ; Set delay  LY: LOOP LY ; Delay loop  ; R2 LED turn on  MOV AL, 08H ; Turn on R2 LED (bit 3)  OUT 1BH, AL ; Send the value to port 1BH  MOV CX, 0FFFFH ; Set delay  LR2: LOOP LR2 ; Delay loop  ; Repeat the sequence  JMP LED ; Jump back to LED label  L ENDS  END START |  |