LAB-1

① PCB → Printed Circuit Board → (pdf-এ আছে figure টা)

② 8086 μp chip (example → 14.7456 MHz).

③ KBIC → Key Board Interface Controller [under the keypad – not visible]

④ Output device → LCD [Liquid Crystal Display] unit ; LCDIC হিসেবে 16 (char) × 2 lines

⑤ 7SDD → 7-segment display Device.

⑥ CA7SDD → Common Anod type 7-segment Display Device.

IC → 8255 chip : programmable I/O device that acts as interface between PERIPHERAL Devices and μP. for parallel data tranfer.

⑥ LED → Light Emitting Diode . [LED-11 – LED-13 in board
IC → 8255 (chip U29 in pic).]

⑦ RAM → Random Access Read and Write Memory.

Capacity (normally) → RAM ১৭ ২২ টা 32Kbyte.

(32 × 1024 × 8) bits . means 32 × 1024 byte locations inside the RAM.

address of memory → [00000H – 0FFFFH] totally 64Kbyte.

IC → 62256 (2x)

⑧ EPROM → UV Erasable Electrically Programmable Random Access Read Only Memory.

↓ 32KB

↓ range of address: F0000H ~ FFFFH

IC → 27C256 (2x)

↓ They have the Monitor Program of the MDA-8086 trainer.

↓

* provides interface btw user and computer at low level.
* provides a way of Loading a program to primary memory (RAM), test memory, examine registers, more etc cpy memory, run programs.
* provides all [Basic Input Output [BIOS]] functions.

⑨ A/D converter → converts analog signal to digital signal [ADC0804]

⑩ D/A converter → " digital " to analog " [DAC0800]
controls level meter

KeyBoard (26-keys)

MPU control keys + Command keys = [2+7] = 9 keys.

MPU control keys

*RES → MPU (μP) RESET — key pressed : 8086 remains in reset state.
" released : " enters into working "
message is shown on LCD monitor.
jumper (switch-এর বলে) must be set to left position [left position - 2 jmper বাহ্যা মানে kit position]

* MON → Monitor
② NMI → Non Maskable Interrupt [Pin আছে 2টা]
— key pressed : CPU receives external interuppt signal via NMI Pin
↓
* Hardware interrupt (or signal to processor) that prioritizes a certain thread or process.
* Can't be interrupt by masking techniques.

Example → Clicking reset button, turn off power supply.

Command keys:

① * AD (set memory ADdress) : allows to set 20 bit address of mem location.
Format → "Segment : Offset = Seg : Oset"

* Press the key; LCD will show ↓

offset ← Seg. Oset Data → 8 bit mem location pointed by Seg : Oset
segment ← xxxx yyyy zz

* Address is 5 digit hex.
* " obtained by placing 0 to segment's right then add with Oset.
* Example : Seg : Oset = 0000 : 1002.
address : 00000
হবে addition of seg + Oset → 00000 + 1002
= 01002 ans.
* another example : Seg : Oset = F000 : 0000.
address = 0F000h.

② DA [Data entry]: brings cursor to the data field. + enter data [0-F]
③ + (di increment): move to next mem Location address.
④ - (decrement): " " " previous " "
⑤ GO (exits prog).
⑥ REG → checks + examines contents of internal register
⑦ STP → Single step — executes 1 instructions at a time.

Data keys: 0-9 and A-F (hexa keys)

EMULATION KIT:

* software emulation of Intel's 8086 µP: Emu8086.
* A group of hardware devices that can be controlled by Emu8086 virtual central processing unit (CPU).
* Available hardware devices in it: ① Dot Matrix ② 7 Segment Display ③ ASCII LCD display ④ LEDs. ⑤ Push buttons (input) ⑥ Keyboard input. ⑦ Switches input ⑧ Thermometer input ⑨ Pressure Gauge input

STEPS to control µP:

⓪ Emulatore will get the HEX code.
① Press RES - to reset.
② Press DA - to update seg:Oset.
③ Type HEX code.
④ Press + to go to next address.
⑤ After typing (finishing) press STP - to execute
⑥ Press REG to see the result on Display of MDA - Win 8086

Instruction Set :

MOV : copys values from one reg to another
ADD : adds 2 numbers.
SUB : subtracts
MUL : Multiplies
DIV : divides

AX : Accumulator reg
BX : Base reg
CX : Counter "
DX : Data "

INT 3 :
* special 1byte instruction
* Opcode = CCH
* used at breakpoints
* When hit, interrupt handler breaks into debugger.
- replaces original instruction
- lets execute when user is ready

Example : HEX CODE

⊛ converse °C to °K. ; temp = 39°C ; $1°K = 1°C + 273$ formula.

```
MOV AX, 39 ; AX=0027H
MOV BX, 273 ; BX=111H
ADD AX, BX ; AX=0138H
INT 3.
```

suppose AX address 0404
BX " (0404+3) = 0407
result " (0407+3) = 040A
[39] 27H loaded in AX → 0404
[273] 111H " " BX → 0407.
ans (addition of AX+BX) → 040A after STP and REG.
↳ AX = 0138H

⊛ converse °K to °C ; temp = 270°K. $1°C = 1°K - 273$.

```
MOV AX, 270.  ; AX এ 10EH load দিল ; address = 0404
MOV BX, 273.  ; BX এ 111H. "  " ; "  = 0407
SUB AX, BX    ; AX-এ subtract করে মান load হইলো ; address = 040A.
INT 3         [executes AX = -3]
```

⊛ average of 3 numbers ; (2+3+5)/3 = কিছু একটা আসবে

```
MOV AX, 2   ; AX=2 or 2H ⟶ 0404
MOV BX, 3   ; BX=3 or 3H ⟶ 0407
ADD AX, BX  ; AX=(2+3)=5 or 5H ⟶ 040A.
MOV BX, 5   ; BX=5 ⟶ 040C
ADD AX, BX  ; AX=(5+5)=10 or AH ⟶ 040F
MOV BX, 3.  ; BX=3. ⟶ 0411
DIV BL      ; 10/3 ⟶ 0413
```

°F to °K → $K = \frac{(F-32)5}{9} + 273$

(*) Floor size = 20 × 20
Tiles " = 2 × 2
How many tiles needed?

Normal solution ⇒
tiles needed = (20×20)/(2×2) [∵ Floor size / tiles size]

```
MOV AX, 20    ; AX=20 or 14H
MOV BX, 20    ; BX=20 or 14H
MUL BL        ; AX = AX × BX = 400 or 190H.
MOV CX, AX    ; CX = AX = 190H or 400.

MOV AX, 2     ; AX=2
MOV BX, 2     ; BX=2
MUL BL        ; AX=4 or 4H (2×2)
MOV BX, AX    ; BX = AX = 4H
MOV AX, CX    ; AX = CX = 400 or 190H.
DIV BL        ; Divide → AX/BX = 400 ÷ 4 or, 190H/4H = 100 or 064H
INT 3         ; break.
```

(*) Factorial of 5! - 3! মানে (5×4×3×2×1) − (3×2×1) = 114

```
MOV AX, 1
MOV CL, 5     ; CL=5
L1:
MUL CL        ; AX = AX × CL = 1×5.
DEC CL
LOOP L1       ; CL-- so loop-complete - হয়ে AX = 5×4×3×2×1. = 120 or 078
MOV DX, AX    ; DX = AX = 120 or 078H.
MOV AX, 1
MOV CL, 3
L2:           ; AX = 3×2×1 = 6
MUL CL
DEC CL
LOOP L2
MOV BX, AX    ; BX = 6.
MOV AX, DX    ; AX = DX = 120.
SUB AX, BX    ; AX = AX − BX = 120 − 6 = 114 or 72H.

INT 3.
```

✳ $(5!/3!)+4!$

```
MOV AX, 1
MOV CL,5
L1:                ; AX = 5.4.3.2.1 = 120
MUL CL
LOOP L1
MOV DX, AX   ; DX = AX = 120
```

```
MOV AX, 1
MOV CL, 3.
L2:                ; AX = 3.2.1 = 6
MUL CL
LOOP L2
MOV BX,AX .; BX = AX = 6 .
MOV AX,DX   ; AX = DX = 120
DIV BL      ; AX = 120/6 = 20 .
MOV DX,AX   ; DX = AX = 20 .
```

```
MOV AX, 1
MOV CL, 4
L3:                AX = 4.3.2.1 = 24
MUL CL
LOOP L3.
ADD AX,DX.   AX = AX + DX = 24 + 20 = 44 or 2CH .
```

INT 3.

✳ $(2! * 3! * 4!)+4!$

```
MOV AX, 1
MOV CL 2
L1:                AX = 2.1 = 2
MUL CL
LOOP L1
MOV DX, AX ; DX = AX = 2
```

```
MOV AX, 1
MOV CL, 3
L2:                AX = 3.2.1 = 6
MUL CL
LOOP L2
MOV ~~AX, DX~~ BX, AX  ; BX = AX = 6 .
MOV AX, DX  ; AX = DX = 2
MUL BL      ; AX = AX x BX = 2x6 = 12
MOV BX, AX  ; DX = AX = 12
```

```
MOV AX, 1
MOV CL, 4
L3:                AX = 4.3.2.1 = 24
MUL CL
LOOP L3
MOV BX, AX. ; BX = AX = 24
MOV AX, DX_; AX = DX = 12
MUL BL.     ; AX = AX x BX = 12x24
                   = 288
ADD AX, BX  ; AX = AX + BX = 288 + 24
                   = 312
INT 3.
```

⊛ Byte with Byte Division.

```
ORG 100h
:MODEL SMALL
.DATA
   num_1 DB 0F2H
   num_2 DB 4H
.CODE
   MOV BH, num_2
   MOV AL, num_1
   DIV BH ; BH/AL
   RET    ; return
```

⊛ Word by Word করলে

.DATA – তে DW হবে

আর code –এ

```
MOV AX, num_1
DIV num_2 ; AX/num_2.
```

Division–এ : numerator যে call করবে byte–এর জন্য সেইরূপ
denominator " " " word " "

byte–এ AL = quotient
AH = remainder

word–এ DX = remainder
AX = quotient

⊛ ORG := ORiGin → directive not instruction.
defines where the machine is to place in memory.

ORG 100H মানে 1 segment –এ 64 KB max space আছে
and machine code starts from 100H (offset address).
Effective address → CS:100H.

⊛ .model small → you get a programe
মানে CS [code segment] –এ max space 64 KB.
DS [data "] " " " "

.MODE MEDIUM : CS can exceed 64KB
DS = 64KB.

.MODEL COMPACT : DS exceeds > 64KB
CS < 64 KB.

.MODEL LARGE : DS, CS > 64KB. but signle set of DATA ≤ 64KB.

.MODEL HUGE : DS, CS > 64 KB ; array data > 64KB too.

.MODEL TINY : DS, CS = 64KB.

Lab 2 :

8051 MC simulation tool → Keil C51.

* MC is called computer on chip. as it includes µP, with RAM, ROM, parallel and serial ports.
* Applications : Washing Machine, Ved player, oven, robotics.
* 8051 → 8 bit MicroController आहे 8bit Data bus.
  reads, writes, process 8 bit data.
  executes code from embedded MASKED ROM.
* Original MCS-51 family (Intel's) ~~der~~ was developed with
  N-MOS (N type metal Oxide - semiconductor)
  consumes more power

but their predecessors MCS-48 family used Complementary MOS (CMOS) → 89C51
- consumed less power.
- more suitable for battery powered devices.

AT89C52 – Atmel's 8051 family 8bit MC.
- has 8KB of flash programmable erasable read only memory (PEROM)
- 256 bytes of RAM.
- Endurance of 1000 write/Erase cycle.
  आहे erased/programmed to a max 1000 times.

Oscillator : Provides clock to 8051 MC [pins – XTAL2, XTAL1]
decides the speed of MC.
Crystal oscillator → normally 11.0592 MHz frequency.

I/O ports : 4 available P0, P1, P2, P3.
each port is 8 bit and bit addressable [can be set or reset by bit instructions → SETB (High), CLR (Low)]

port 0 → dual work ; lower order address (A0 to A7)
multiplexed with P0.0 to P0.7 is AD0 to AD7
address bus and data bus is demultiplex by ALE signal and latch.

P1 → true I/O port (P1.0 to P1.7) [general I/O only]
P2 → I/O as well as higher order address bus A8 to A15.
P3 → dual func. as I/O ; also has specific function.

for AT89X51 → header : # include <AT89X51.h>

Lab-3

ARDUINO: open source electronics platform

- reads inputs [light on a sensor]
- turns into an output [activates motors]

→ UNO → 14 pins (digital) + 6 pins [analog] = 20
→ MEGA → 54 " " + 16 " " = 70

UNO : D13 → SPI SCK · AX0 → D0
D12 → " MISO TX0 → D1
D11 → " MOSI
D10 → " SS

MEGA ⇒ analog pins digital behaviour ⇒ [oscillator = 16 MHz]

Arduino lang: 3 parts 1. Structure
2. values [variables + constants]
3. functions

Structure : Sketch → loop()
setup()

Further → #define (define)
#include (include)

Control struc → break
continue
do.... while
....

Arithmetic → % remainder
* mul
+ add

Functions:

① digitalRead(pin_no) ; reads and returns high or low

~~② digitalWrite(pin_no) ; writes~~

② digitalWrite(pin, value) ; writes high or low to a pin.

returns nothing.

③ pinMode(pin, mode)

configures pin's behaviour

mode might be input/output.

Basic syntax:

```
void setup(){
    // to run once
}
void loop(){
    // to run repeatedly
}
```

Keypad

4x3 : R(4) C(3) Left to right

4x4 : R(4) C(4)

func → makeKeymap(keys). → initilizes internal keymap to be equal to user defined keymap

syntax: Keypad kpd = Keypad(makeKeymap(keys), rowPins, colPins, rows, cols);

class — Keypad

obj — kpd

initialize — makeKeymap(keys)

defined arduino 4+4 pins — rowPins, colPins

Serial Communication: used for communication btw. Arduino board and other devices.

0,1 pins are used.

Serial Functions:

① Serial.begin(9600) → sets data rate in bits per second (baud)
typical ↳ or more

② Serial.print("String") → prints data (human readable ASCII text)

③ Serial.println("String") → : " " " " " "
followed by carriage return char '\r' and a newline '\n'

setup(){}

Piezo Buzzer : produces sound.

- based on inverse principle of piezo electricity discovered in 1880 by Jacques and Pierre Curie.

Phenomena of generating electricity when MECHANICAL PRESSURE is applied to certain materials and vice versa is true

ELECTRICITY ∝ MECHA PRESSURE

Heart of piezo buzzer (DISC) → made of piezoceramic (piezoelectric mat)
↓
poses piezo electric effect.

in alternating electric field
they streach or compress – fq of signal – causes sound.

Buzzer Functions:

→ tone(): → generates square wave.
syntax: tone(pin, frequency, duration)
ton (" , ") বাজতে থাকবে

→ notone(pin) stops generation.

Buzzer: — pos ~~is~~ pin to mentioned pin
neg " to gnd.

Servo Motor:

Brown ⊖.
Red ⊕
orange signal/PWM.

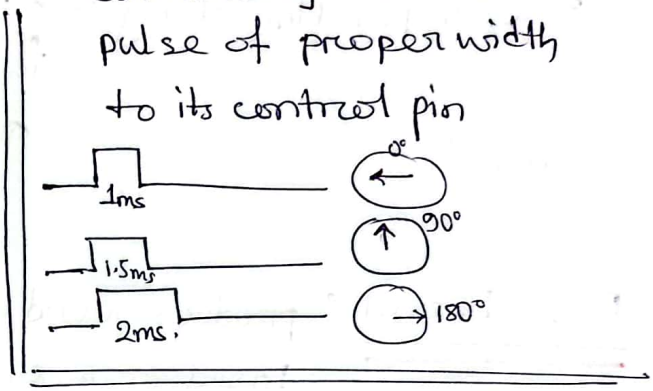
rotation range : 180° (−90° to +90°).

parts : DC Motor
Gear system.
Position sensor
Control circuit

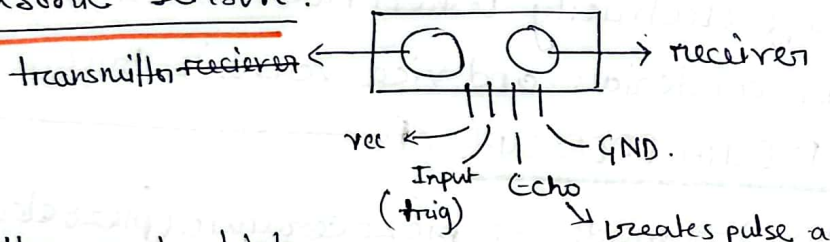
~~de~~

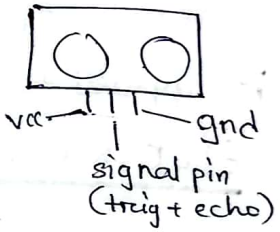
~~PM~~: PWM – Pulse Width Modulation.

Controlled by electrical pulse of proper width to its control pin



Ultrasonic sensor:





ultrasound – high pitched – higher than 20KHz

HC SR-04 – 40KHz converts.

detection range : 2cm – 4m.

Creates pulse as output
Length of pulse = time of transmission

$s = vt$

IR sensor :

3 pins → vcc, gnd, vout.

emits light after sensing obj

Emitter + Detector
↙ LED. ↘ photodiode

Output = intensity of reflection received by photodiode
↳ binary

Infrared Sensor :

- contains active emitter.
- detects exact position of obj
- used :- garage door sys, industrial settings.

PIR Sensor :

- contains no active emitter
- detects motion
- used : security alarms, automatic lighting setups.

3 pins
Vcc
gnd
Vout

has 2 slot — each slot ~~has IR~~ made of mat that is sensitive to IR.

when sensor is idle
both slot detect same amount of IR.

→ detects warm objects one half — causes a positive differential chng btw the 2 halves
→ objat leaves · with reverse reaction — negative differential chng btw the halves.

*** Do not detect or measure heat

detects IR emitted by objects.
" thermal radiation. in infrared range.
purpose of lens → widening sensing area; output : binary.