F241 Microprocessors & Interfacing
Project Report

Group 7
Batch Weighing Machine

2020-21

Group Members :

ID	Name
2019A7PS0043G	Nishant Vikas Sarangdhar
2019A7PS0086G	Shubham Rajnish Gandhi
2019AAPS0315G	Suhaas Mahajan
2019A3PS0373G	Saksham Shivshankar Kamath
2019AAPS0489G	Rishabh Garg
2019A8PS0509G	Shivang Thalpiyal

Project done under the guidance of Dr. K.R. Anupama at Birla Institute of Technology & Sciences, Goa

TABLE OF CONTENTS

PROBLEM STATEMENT

A microprocessor based system is to be designed as a batch weighing machine.

- The system is interfaced to three load cells by means of an 8 bit A/D converter.
- The conditioned output of the load cells is given by the equation:
    - *Vout = 0.025 x weight (kg)*
- The system monitors the output of the load cells and finds out the total weight by taking the average of the three values that are sensed by each load cell.
- This value is displayed on a seven-segment display.
- When this value exceeds 99 kgs, an output port, which is connected to a relay, is switched on to sound an alarm.
- Once the objects are placed on the load cell user presses a switch labelled **weigh**, this starts the weighing process
- There is also an alarm off switch to turn off the alarm

USER REQUIREMENTS AND TECHNICAL SPECIFICATIONS

A microprocessor based system is to be designed as a batch weighing machine.

- The system is interfaced to three load cells by means of an 8 bit A/D converter.
- The conditioned output of the load cells is given by the equation: Vout = 0.025 x weight (Kgs.)
- The system monitors the output of the load cells and finds out the total weight by taking the average of the three values that are sensed by each load cell.
- This value is displayed on a seven-segment display.
- When this value exceeds 99 kgs, an output port, which is connected to a relay, is switched on to sound an alarm.
- Once the objects are placed on the load cell user presses a switch labelled weigh.

ASSUMPTIONS AND JUSTIFICATIONS

Assumptions

1. User always presses the alarm off switch, before moving on to the next measurement, if the alarm is ringing due to previous measurements
2. If the weight exceeds 99kgs, **99** is displayed on the 7-segment display & buzzer goes off.
3. At a time only data from one load cell is being processed
4. It can measure with a least count of 1kg.

Justification

1. We display 99 as it is the maximum possible weight as per the problem statement
2. The ADC can process one analog input at a time and convert it into digital output one at a time only. Our design is an optimum design. Thus, we have used only 1 ADC.
3. We are making use of BSR functionality of port C, hence we are able to accommodate the Input output devices in 1 8255.

COMPONENTS USED WITH JUSTIFICATION WHEREVER REQUIRED

DEVICE	NOS.	USE
8255 - PPI	1	Used for interfacing with I/O devices like load cell and 7-segment display
8259 -Interrupt Controller	1	Used for managing the control of various interrupts
6116 - RAM	2	Used for external storage
2716 - ROM	4	Stores the code, IVT
8284- Clock generator	1	Used for generating the clock pulse taking the input from 15MHz Crystal
8086	1	Microprocessor used to perform calculations (average of 3 weights)
8254- Programmable Interval Timer	1	Used to generate 1MHz clock pulse using 5MHz pulse. (reason :- ADC requires 1MHz input)
ADC-0808N	1	Converts analog input of load sensor to digital output processed by 8086.
74LS138- 3x8 Decoder.	2	For memory and I/O decoding.
74LS373- Latch	3	To latch the address given on the AD bus.
74LS245- Buffer	2	To boost up the bus signals.
TIP120- Darlington configuration	1	To amplify the current from 8255 port. Amplified current is required by relay.
G2RG- Relay	1	To connect or disconnect the buzzer from the rest of the circuit.

12V Buzzer	1	It rings when the average weight exceeds 99kg.
74LS47 decoder	2	BCD to seven segment decoder.
TDSG 5150- 7 segment display	2	The average weight is displayed on it if it's less than 99 kg.
M74HC32- Quad 2 input OR gates	3	Used for generating various chip select and control signals.
Load Sensors	3	***Vout = 0.025 x weight(kg)***
LS244- octal buffer	1	Used for generating control signals.

ADDRESS MAP

Memory Map

DEVICE	START ADDR.	END ADDR.
ROM - 1	00000 h	00FFF h
RAM	01000 h	01FFF h
ROM - 2	FF000 h	FFFFF h

I/O Map

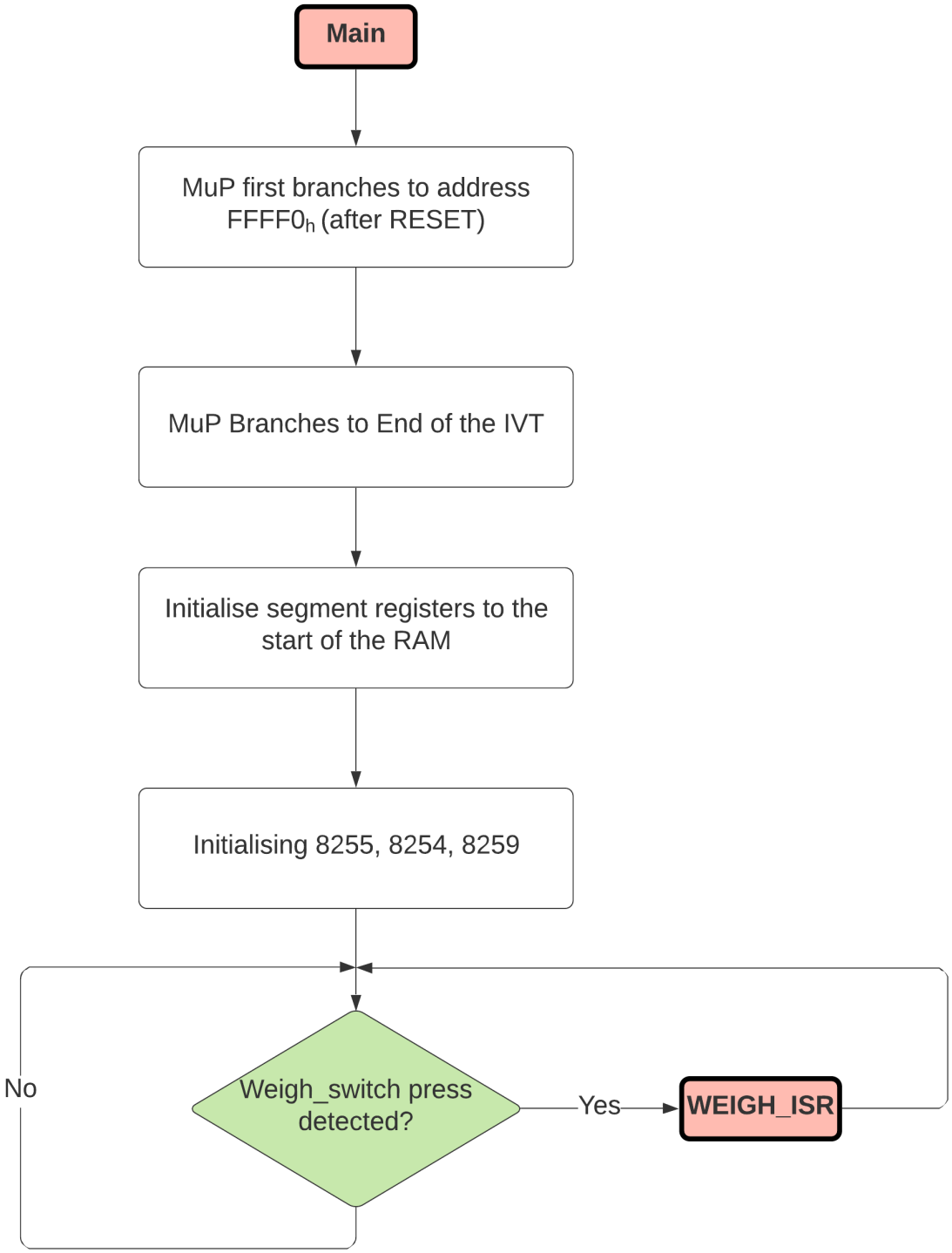
DEVICE	START ADDR.	END ADDR.
8255 - PPI	00 h	06 h
8254 - Interval Timer	20 h	26 h
8259 - Interrupt Controller	30 h	32 h

DESIGN

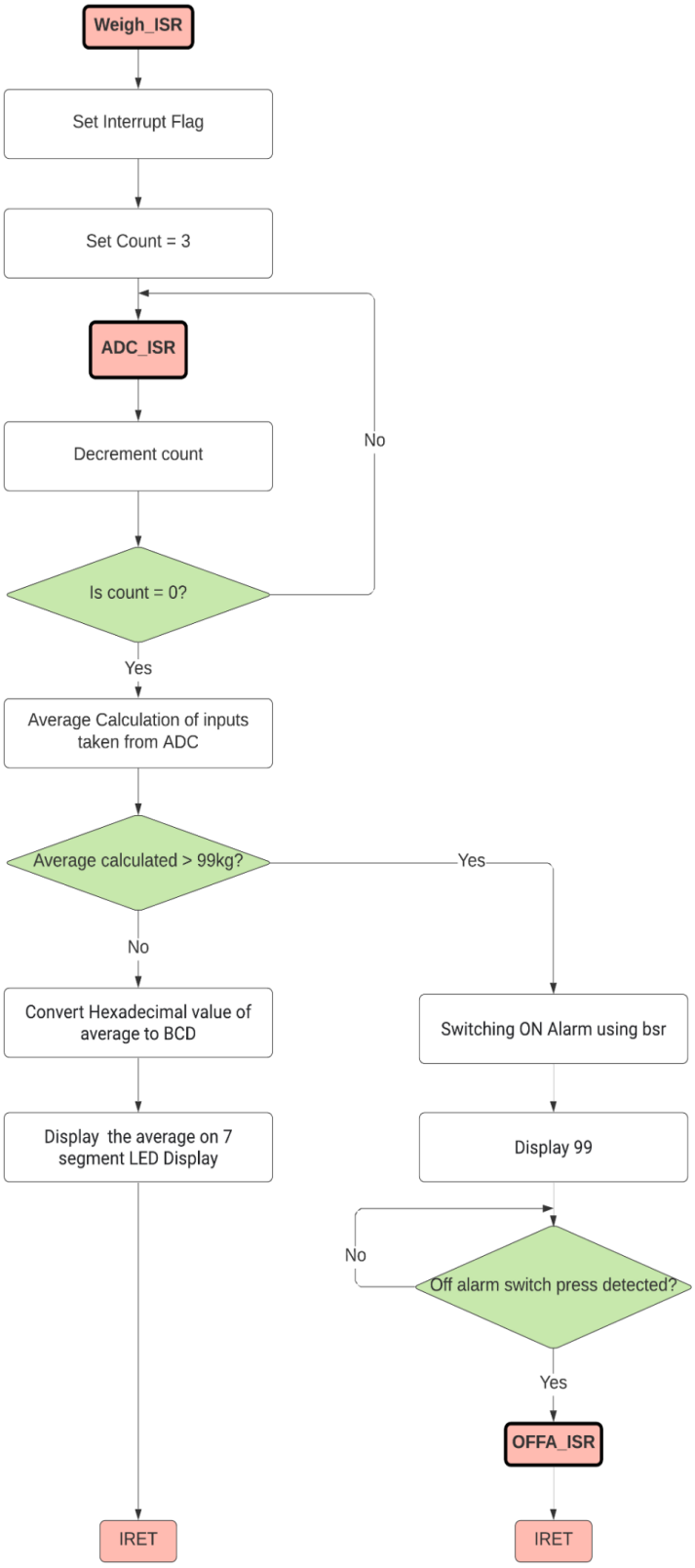
Complete design shown with proper labelling (ON-PAPER DESIGN attached as both PPT and PDF files. Please check zip folder)

FLOWCHARTS

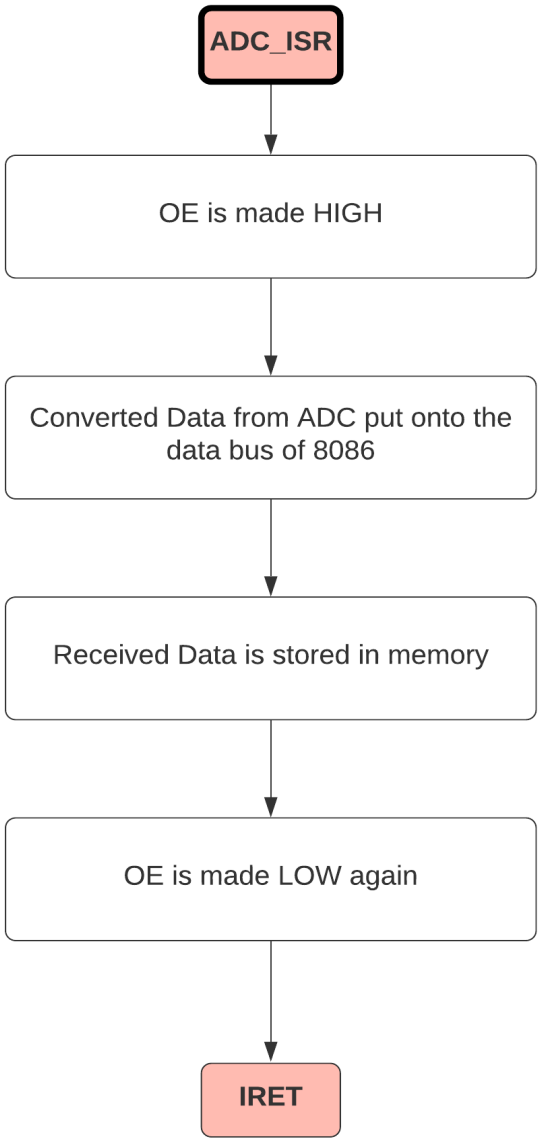
Main Program



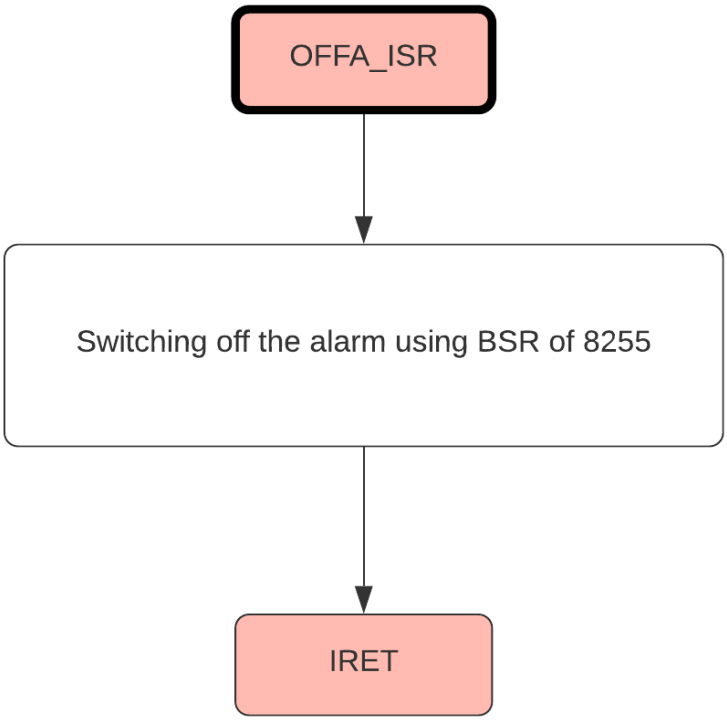
WEIGH_ISR


Weigh_ISR
Set Interrupt Flag
Set Count = 3
ADC_ISR
Decrement count
No
Is count = 0?
Yes
Average Calculation of inputs taken from ADC
Average calculated > 99kg?
Yes
No
Convert Hexadecimal value of average to BCD
Switching ON Alarm using bsr
Display the average on 7 segment LED Display
Display 99
No
Off alarm switch press detected?
Yes
OFFA_ISR
IRET
IRET

ADC_ISR


ADC_ISR
OE is made HIGH
Converted Data from ADC put onto the data bus of 8086
Received Data is stored in memory
OE is made LOW again
IRET

OFFA_ISR



VARIATIONS IN PROTEUS IMPLEMENTATION WITH JUSTIFICATION

- 8284 and 8254 are not available. The clock of 8086 is directly provided as 5 MHz.
- Instead of 8254, 8253 is used, the input clock fed in is 2.5MHz. This generates 500KHz , instead of 1MHz which is fed into ADC.
- We are using two 8255's in Proteus as 8259 is not available. The first: 8255A is used for taking in the values from ADC. The second : 8255B has connections to 7 segment displays. WEIGH_SWITCH gets connected to NMI of 8086. PC0 has EOC of ADC.
- Load sensors are not available, hence variable DC voltage sources are used in place of load cells.
- 2716 is not available in proteus hence we used 2732.

CODE ACCORDING TO ON-PAPER DESIGN

```
#make_bin#

; set loading address, .bin file will be loaded to
  this address:
#LOAD_SEGMENT=0000h#
#LOAD_OFFSET=0000h#

; set entry point:
#CS=0000h#       ; same as loading segment
#IP=0000h#       ; same as loading offset

; set segment registers
#DS=0000h#       ; same as loading segment
#ES=0000h#       ; same as loading segment

; set stack
#SS=0000h#       ; same as loading segment
#SP=FFFEh#       ; set to top of loading segment

; set general registers
#AX=0000h#
#BX=0000h#
#CX=0000h#
#DX=0000h#
#SI=0000h#
#DI=0000h#
#BP=0000h#
```

```
;Jump to start of code
        JMP ST
        NOP

;NOP is added so that we get proper intervals of 4
  bytes, jmp st1 will be 3 bytes and nop will be 1
  byte

        DW 114 DUP (0) ; int 01h to 39h unused  //
  39h * 2 = 114d

;Interrupt for off alarm at vector 40h

        DW OFFA_ISR
        DW 0000H

;Interrupt for weigh switch at vector 41h

        DW WEIGH_ISR
        DW 0000H

;Interrupt for eoc at vector 42h keeping cs as 0000
  only

        DW ADC_ISR
        DW 0000H
```

```
; int 43h to FFh are unused
; FFH-43H+1H = BDH = 189D, 189D * 2D = 378D
        DW 378 DUP (0)

;Defining some Labels

        ;8255
        PORTA1 EQU 00H
        PORTB1 EQU 02H
        PORTC1 EQU 04H
        CREG1  EQU 06H

        ;8254
        CNT3   EQU 20H
        CREG3  EQU 26H

        ;8259
        A82591 EQU 30H
        A82592 EQU 32H

;Main Program

ST:     CLI ; Clear interrupt flag so that no
  interrupts are recieved during initialization

;Intialize ds, es,ss to start of RAM
        MOV       AX, 1000H
        MOV       DS,AX
        MOV       ES,AX
```

```
        MOV         SS,AX
        MOV         SI,AX ; used for storing weights
  measured temporarily

;Initializing ports

        ;For 8255 Port A & B are inputs Port C is
  output

        MOV AL, 10010010B       ;8255
        OUT CREG1,AL


        ;8254 - counter 0, read write lsb and msb,
  mode 3, binary
        MOV AL,00110110B
        OUT CREG3,AL                ;8254 Control
  Word
        MOV AL,05H
        OUT CNT3,AL                 ;Giving count of 5
  to counter0 of 8254
        MOV AL, 00H
        OUT CNT3, AL

        ;8259 initialization
        MOV AL, 00010011B       ;8259 ICW1
        OUT A82591,AL
        MOV AL, 01000000B       ;8259 ICW2, starting
  interrupt vector is 40H
```

```
        OUT A82592,AL
        MOV AL, 00000001B       ;8259 ICW4, since we
  dont have a slave, ICW3 is skipped
        OUT A82592,AL
        MOV AL, 11111000B       ;8259 OCW1 only
  IR0,IR1,IR2 are enabled
        OUT A82592,AL

 ;Setting 8255 to i/o mode

        MOV AL, 10010010B       ;8255
        OUT CREG1,AL

        MOV AL, 00H      ; display 00 by default
        OUT PORTA1,AL

        STI ;Set interrupt flag to enable receiving
  interrupts

INF1:   JMP INF1          ; Waiting for user to press
  the weigh switch for measurement

WEIGH_ISR:

        STI

; Taking inputs from adc using 8255
; We have to take 3 values
```

```
        MOV CX,0003H
        MOV DH,00H
        MOV DI,0000H

; Ports need to be reconfigured

wloop:  MOV AL, DH
        OUT PORTC1, AL


; Making ale 1 from BSR mode

        MOV AL, 00001011B
        OUT CREG1, AL
        NOP
; Making soc 1 from BSR mode

        MOV AL, 00001001B
        OUT CREG1, AL

; Make ale 0 using bsr mode, since ALE must be
  active only for 1 clock cycle
        MOV AL, 00001010B
        OUT CREG1, AL
        NOP

; Make soc 0 using bsr mode, since SOC must be
  active only for 1 clock cycle
        MOV AL, 00001000B
```

```
        OUT CREG1, AL

;Wait for EOC to be received from the ADC
        HLT


; Looping

        INC DH ; for taking in next analog input
  from load cell
        INC DI
        LOOP wloop

; Average calculation

        MOV SI,0000H
        CLD
        MOV AH,00H
        MOV BX,0000H
        MOV CX,0003H

LLOOP:  LODSB
        ADD BX,AX

        LOOP LLOOP

        MOV AX,BX
        MOV BL,3
        DIV BL
```

```
; Compare with 99

        ;IF WEIGHT<=99KG then it is valid
        CMP AL,99
        JBE VALI

;Setting 8255 to i/o mode

        MOV AL, 10010010B       ;8255
        OUT CREG1,AL

        ;If weight > 99
        MOV AL, 99H      ; If weight exceeds 99kgs,
  we display 99
        OUT PORTA1,AL

; Switching ON Alarm using bsr

        MOV AL,00001111B
        OUT CREG1,AL

        HLT ; Waiting for user to turn off alarm
  manually, this is part of assumption

        JMP INVALI

; convert from hex to bcd
```

```
VALI:   MOV         BL,AL
        MOV         AL,0
HTB:    ADD         AL,01
        DAA
        DEC         BL
        JNZ         HTB

; Setting 8255 to input/output mode

        MOV AL, 10010010B       ;8255
        OUT CREG1,AL

; display bcd

        OUT PORTA1,AL

INVALI:
        ;ending interrupt using Non-Specific EOI
        MOV AL, 00100000b
        OUT A82591,AL

        IRET



OFFA_ISR:

; Making alarm off using bsr
```

```
        MOV AL,00001110B
        OUT CREG1,AL

        ;ending interrupt using Non-Specific EOI
        MOV AL, 00100000b
        OUT A82591,AL
        IRET


ADC_ISR:

; Making OE 1 using bsr for 8255

        MOV AL,00001101B
        OUT CREG1,AL

; Taking Inputs from Port B
; Setting 8255 to i/o mode

        MOV AL, 10010010B       ;8255
        OUT CREG1,AL

        IN AL, PORTB1
        MOV [DI], AL

        MOV AL,00001100B; Making OE low again
        OUT CREG1,AL

        ;ending interrupt using Non-Specific EOI
```

```
        MOV AL, 00100000b
        OUT A82591,AL

        IRET
```

LIST OF ATTACHMENTS

1. Complete Hardware Design
2. Manuals (all manual texts are clickable links)
    a. 74LS47
    b. 74LS373
    c. 8086
    d. 8255A
    e. 8259A
    f. 8284b
    g. ADC0808
    h. Buzzer
    i. Quad 2 input OR Gate
    j. Relay G2RG
3. Proteus File -
4. EMU8086 ASM files -
5. Binary file after assembly -