Microprocessor programming and Interfacing Design Assignment

IC TESTER [P6]

Group 101

Ayush Garg	2017A7PS0193P
Shivangi Singh	2017A7PS0213P
Patel Abhishek Bipinkumar	2017A7PS0214P
Jhaveri Ayush Rajesh	2017A7PS0215P

CS F241: MICROPROCESSOR PROGRAMMING AND INTERFACING

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, Pilani
RAJASTHAN-333031

24 APRIL, 2019

Problem Statement (P6)

Design a Microprocessor based tester to test the logical functioning of the following chips:
(i) 7408
(ii) 7486
(iii)7432

The IC to be tested will be inserted in a 14 pin ZIF socket. The IC number is to be entered via a keyboard. The results of the test along with the IC number are to be displayed on LCD as "74xy PASS" or "74xy FAIL". Design the necessary hardware and write the necessary ALP for implementing the above-mentioned task.

Assumptions:

The following are the assumptions made regarding the system:-

- The user presses the appropriate button.
- User enters a 4-digit number as the IC number.
- The 8086 Chip is already programmed with the specified code from an external source.
- The User inserts the IC into the ZIF Socket before entering the IC Number on the Keypad
- There is a jump instruction at the address that the processor generates to the place where our code is stored (i.e. to FF000H).

- We have connected the ICs individually to the 8086 through ZIF socket.

- GND Pins of the 8086 are grounded and VCC & MIN/MAX pins are connected to +5V.

System Description

The basic function of the digital IC tester is to test a digital IC for correct logical functioning as described in the truth table. It can be used with digital ICs having 14 pins. Since it is programmable, any number of ICs can be tested within the constraint of the memory available. This model applies the necessary signals to the inputs of the IC, monitoring the outputs at each stage and comparing them with the outputs in the truth table. Any discrepancy in the functioning of the IC results in a fail indication, displays the faulty and good gates on the LCD. The testing procedure is accomplished using input provided by the Hex Keypad.

This model examines the most common used digital IC's used in digital circuits which are 7408, 7486 and 7432 and assembly program is also provided along with the model.

- The system is based on an 8086 microprocessor using an 8255 for Input/Output interfacing.
- A total of 2, 4k ROM and 2, 2k RAM memory are interfaced to a 8086 microprocessor.
- The memory chips are interfaced by memory mapped interfacing while all other devices are interfaced to the microprocessor by I/0 mapped interfacing.
- 16 push buttons are used to form a 4x4 matrix keypad with keys 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, BACKSPACE, RESET, A, B, C, D.
- Output is in the form of text messages displayed on the LCD.
- 3 ICs are connected to 8086 through ZIF socket individually.

List of Components

Chip Number	Quantity	Chip	Purpose
8086	1	Microprocessor	Central Processing Unit
6132	2	RAM	Read write memory to house data and stack segment
2732	2	EPROM	Read only erasable programmable memory to house the code
8255	2	Programmable Peripheral Interface	Interfacing the LCD and Keyboard
74LS245	2	8 bit bidirectional buffer	
74LS273	3	8 bit Latches	
74154	1	Decoder	
LM020L	1	LCD	Display
74LS138	1	Decoder	CS LOGIC
214-3339-00-0602J	1	ZIF Socket	For attaching IC

Memory Mapping

STARTING ADDRESS - 00000H
Number of 2732 chips - 2 ROM-4K
We divide 4K ROM in - 2K even bank and 2K odd bank.
Number of 6116 chips - 2
RAM -2K
We divide 2K RAM in -1K even bank and 1K odd bank
ROM1E - 00000H,00002H, ... , 00FFEH
ROM1O - 00001H,00003H, ... , 00FFFH
ROM2E - 01000H,01002H, ... , 01FFEH
ROM2O - 01001H,01003H, ... , 01FFFH
RAM1E - 02000H,02002H, ... , 027FEH
RAM1O - 02001H,02003H, ... , 027FFH
RAM2E - 02800H,02802H, ... , 02FFEH
RAM2O - 02801H,02803H, ... , 02FFFH

Input/Output Organization

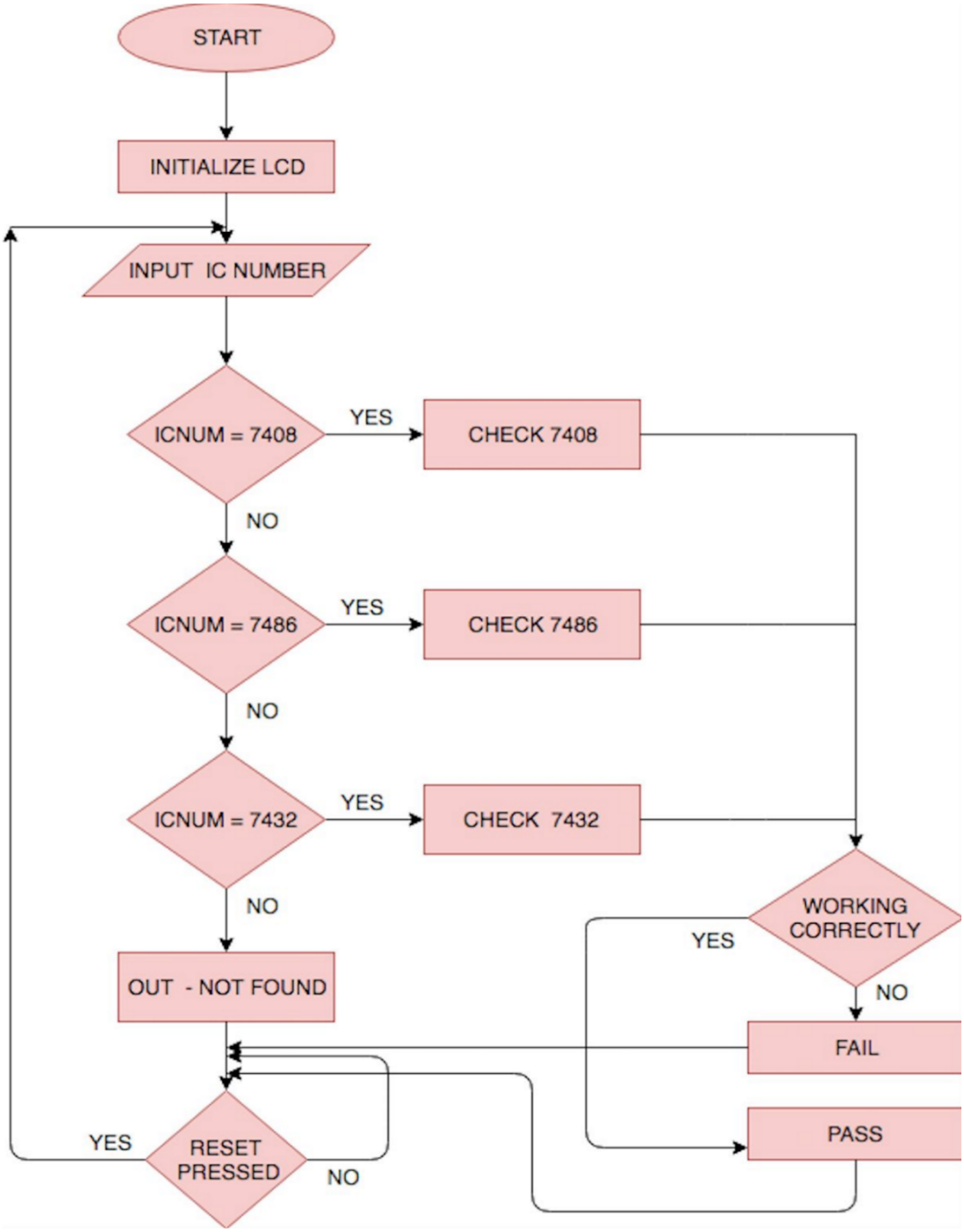
8255(1)

Port	Port Address	Mode	Input/Output
A	10h	0	Output
B	12h	0	Output
Lower C	14h	0	-
Upper C	14h	0	-
Control Register	16h		

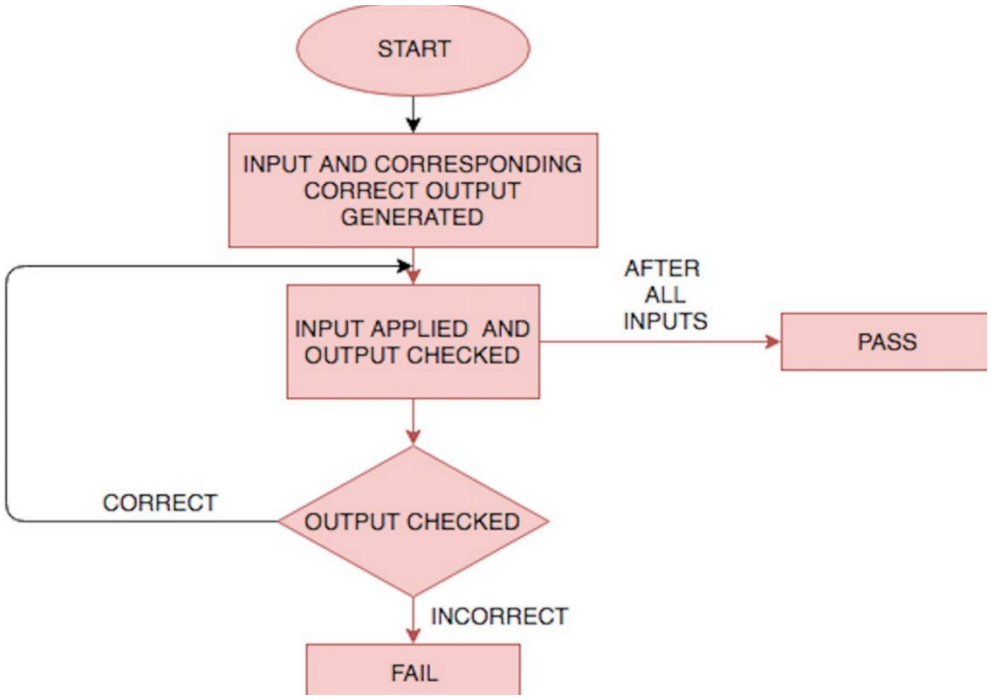
8255(2)

Port	Port Address	Mode	Input/Output
A	20h	0	Output
B	22h	0	Input
Lower C	24h	0	Output
Upper C	24h	0	Input
Control Register	26h		

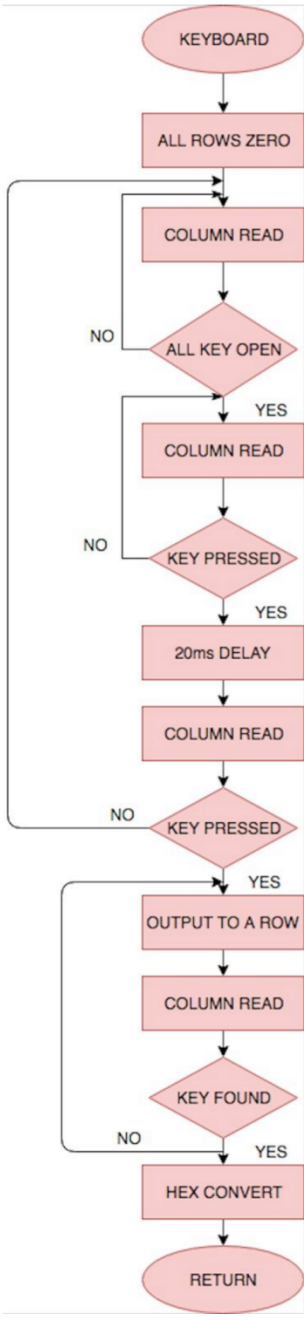
FLOWCHART


START
INITIALIZE LCD
INPUT IC NUMBER
ICNUM = 7408
YES
CHECK 7408
NO
ICNUM = 7486
YES
CHECK 7486
NO
ICNUM = 7432
YES
CHECK 7432
NO
OUT - NOT FOUND
WORKING CORRECTLY
YES
NO
FAIL
PASS
YES
RESET PRESSED
NO

IC TESTING ALGORITHM



KEYBOARD TESTING ALGORITHM



Code

```
.MODEL TINY
.DATA

    TEXT_FAIL DB " FAIL"
    TEXT_PASS DB " PASS"
    TEXT_NOT_FOUND DB " NOT FOUND"
     ;message displayed when incorrect IC number is entered
    TEXT_IC_REQUEST DB "IC NUM-  " ;input message(asks user to enter an IC number)

    NUM_7408 DB "7408" ; IC numbers of the ICs that we need to check
    NUM_7432 DB "7432"
    NUM_7486 DB "7486"

    KEYPAD_KEYS DB "0123456789SRABCD" ;list of keypad keys
    IC_NUM 4 DUP('$') ;stores the IC number entered by the user

    ;values used for checking 'correctness' of ics in74xx-input values  out74xx-corresponding output
    values
    IN_7408  DB 00H,55H,0AAH,0FFH
    OUT_7408 DB 00H,00H,00H,0FH
    IN_7486  DB 00H,55H,0AAH,0FFH
    OUT_7486 DB 00H,0FH,0FH,00H
    IN_7432  DB 00H,55H,0AAH,0FFH
    OUT_7432 DB 00H,0FH,0FH,0FH

    ;values used for interfacing push-button matrix with 8086
    ;display table
    DISPLAY_TABLE DW 0H,  01H, 2H,  3H, 4H, 5H
    DW 6H,7H,8H,9H,0AH,0BH,
    DW 0CH,0DH,0EH,0FH
    ; kb table
    KEYBOARD_TABLE DB 0EEh,0EDH,0EBH,0E7H,0DEH,0DDH
    DB 0DBH,0D7H,0BEH,0BDH,0BBH,0B7H
```

```
        DB 07EH,07DH,07BH,077H

        IC_STATUS DB 00H
        ;00H-IC currently not found(i.e. IC num not in 7408,7486,7432) ,01H- correct IC num entered
        LOOP_COUNT DW 00H    ;keeps loop count
        CHAR_COUNT DW 0000H    ;keeps count of number of characters to be printed by WRITE_MEM
        procedure



        ;assigning port addresses 8255-1 and 8255-2
        PORTA equ 10h
        PORTB equ 12h
        PORTC equ 14h
        command_address equ 16h
        PORT2A equ 20h
        PORT2B equ 22h
        PORT2C equ 24h
        command_address2 equ 26h


.CODE
.STARTUP
        mov al,010001010B
        out command_address2,al
        ;initialises 8255-2: PORTA and lower PORTC as output and PORTB and upper PORTC as input
        CALL LCD_START   ;initialises LCD unit
        S1: ;Start label
        MOV LOOP_COUNT,0
        MOV IC_STATUS,00H
        CALL CLS         ;clear LCD
        LEA DI,TEXT_IC_REQUEST   ;di stores address of message to be printed by WRITE_MEM
        MOV CHAR_COUNT,9       ;no of characters to be printed
        CALL WRITE_MEM     ;prints input message to LCD
        S2:
        MOV CHAR_COUNT,01H
        CALL KEYBOARD       ;reads single keypush
        LEA DI,KEYPAD_KEYS
```

```
ADD DI,AX         ;DI stores address pointing to the key pressed
MOV SI,LOOP_COUNT
MOV AL,[DI]       ;reads key pushed into al
CMP AL,"R"        ;if key pushed==R(RESET) ->go to start of program
JE S1
CMP AL,"S"
 ;else if key pushed==S(BACKSPACE)->backspace if LOOP_COUNT>0{i.e. characters
enetered>0}
JNE  S2_1
CMP LOOP_COUNT,00H
JE S2
CALL BACKSPACE    ;backspace procedure
DEC LOOP_COUNT    ;dec number of chaacters{digits} entered
JMP S2
S2_1:
MOV IC_NUM[SI],AL   ;stores the key pressed
CALL WRITE_MEM_APP  ;appends the key pressed onto the current LCD string
INC LOOP_COUNT
CMP LOOP_COUNT,04H  ;loop exit condition
JZ S3
JMP S2
S3:
LEA DI,IC_NUM
MOV CHAR_COUNT,04H
CALL WRITE_MEM   ;writes ic number entered by user onto LCD
LEA BX,NUM_7408             ;7408
CALL IC_CMP  ;compares IC_NUM with 4 digit number whose address is stored in BX.Makes
IC_STATUS=01H if the IC numbers match
CMP IC_STATUS,01H
JNE IC2         ;if IC_STATUS==1 then check if IC is good or bad
CALL CHECK_7408
JMP S4
IC2:
LEA BX,NUM_7486
CALL IC_CMP       ;similar as for 7408
CMP IC_STATUS,01H
JNE IC3
```

```
        CALL CHECK_7486
        JMP S4
        IC3:
        LEA BX,NUM_7432
        CALL IC_CMP        ;similar as for 7408
        CMP IC_STATUS,01H
        JNE NO_IC
        CALL CHECK_7432
        JMP S4
        NO_IC:
        LEA DI,TEXT_NOT_FOUND
        MOV CHAR_COUNT,10          ;if no ic found then writes IC_NUM "not found" on LCD
        CALL WRITE_MEM_APP
        S4:
        CALL KEYBOARD
        LEA DI,KEYPAD_KEYS
        ADD DI,AX          ;loops until user pushes R{RESET}. R resets the IC checker.Goes back to start
        of program
        MOV AL,[DI]
        CMP AL,"R"
        JE S1
        JMP S4
        S5:

.EXIT



; Procedure used

;lcd(liquid crystal device) used in the project to display user input and display status of IC number entered
by user[good,bad,not found].
;Initially it shows the output as "IC NUM -  "

LCD_START PROC NEAR
        MOV AL, 38H ;initialize LCD for 2 lines & 5*7 matrix
        CALL LCD_WRITE ;write the command to LCD
```

```
        CALL DELAY ;wait before issuing the next command
        CALL DELAY ;this command needs lots of delay
        CALL DELAY
        MOV AL, 0EH ;send command for LCD on, cursor on
        CALL LCD_WRITE
        CALL DELAY
        MOV AL, 01  ;clear LCD
        CALL LCD_WRITE
        CALL DELAY
        MOV AL, 06  ;command for shifting cursor right
        CALL LCD_WRITE
        CALL DELAY
        RET
LCD_START ENDP


LCD_WRITE PROC ;this procedure writes commands to LCD
        MOV DX, PORTA
        OUT DX, AL  ;send the code to Port A
        MOV DX, PORTB
        MOV AL, 00000100B  ;RS=0,R/W=0,E=1 for H-To-L pulse
        OUT DX, AL
        NOP
        NOP
        MOV AL, 00000000B  ;RS=0,R/W=0,E=0 for H-To-L pulse
        OUT DX, AL
        RET
LCD_WRITE ENDP


WRITE_MEM PROC NEAR
        CALL CLS
        ;MOV CL,4
        ;MOV CH,0
        XY:MOV AL, [DI]
        CALL DATWRIT ;issue it to LCD
        CALL DELAY  ;wait before issuing the next character
```

```
        CALL DELAY  ;wait before issuing the next character
        INC DI
        DEC CHAR_COUNT
        JNZ XY
        RET
WRITE_MEM ENDP

WRITE_MEM_APP PROC NEAR
 ; SIMILAR AS WRITE_MEM expect that it does not clear the current LCD state
        ;MOV CL,4
        ;MOV CH,0
        X10:MOV AL, [DI]
        CALL DATWRIT  ;issue it to LCD
        CALL DELAY  ;wait before issuing the next character
        CALL DELAY ;wait before issuing the next character
        INC DI
        DEC CHAR_COUNT
        JNZ X10
        RET
WRITE_MEM_APP ENDP



; Procedure for reading key pressed from the keyboard
KEYBOARD PROC NEAR
        PUSHF
        PUSH BX
        PUSH CX
        PUSH DX   ; SAVING THE REGISTERS USED
        MOV AL,0FFH
        OUT PORT2C,AL
        X0: MOV AL,00H
        OUT PORT2C,AL
        X1: IN AL, PORT2C
        AND AL,0F0H
        CMP AL,0F0H
        JNZ X1
```

```
CALL D20MS ;key debounce check
MOV AL,00H
OUT PORT2C ,AL ;provide column values as output through lower port C
X2:IN AL,PORT2C
AND AL,0F0H
CMP AL,0F0H
JZ X2
CALL D20MS              ;key debounce check
MOV AL,00H
OUT PORT2C ,AL      ;provide column values as output through lower port C
IN AL,  PORT2C
AND AL,0F0H
CMP AL,0F0H
JZ X2               ;key debounce check
MOV AL, 0EH  ;column 0
MOV BL,AL
OUT PORT2C,AL
IN  AL, PORT2C
AND AL,0F0H
CMP AL,0F0H
JNZ X3
MOV AL, 0DH ;column 1
MOV BL,AL
OUT PORT2C ,AL
IN  AL, PORT2C
AND AL,0F0H
CMP AL,0F0H
JNZ X3
MOV AL, 0BH ;column 2
MOV BL,AL
OUT PORT2C,AL
IN  AL, PORT2C
AND AL,0F0H
CMP AL,0F0H
JNZ X3
MOV AL, 07H  ;column 3
MOV BL,AL
```

```
        OUT PORT2C,AL
        IN  AL, PORT2C
        AND AL,0F0H
        CMP AL,0F0H
        JZ  X2
        X3: OR AL,BL
        MOV CX,0FH
        MOV DI,00H
        X4: CMP AL,KEYBOARD_TABLE[DI]    ;Comparing with expected AL values
        JZ  X5
        INC DI
        LOOP X4
        X5:      MOV AX,DI ;move the signal for the key pressed to AX
        POP DX
        POP CX
        POP BX
        POPF
        RET
KEYBOARD   ENDP



CLS PROC
        MOV AL, 01  ;clear LCD
        CALL LCD_WRITE
        CALL DELAY
        CALL DELAY
        RET
CLS ENDP



DATWRIT PROC
        PUSH DX  ;save DX
        MOV DX,PORTA  ;DX=port A address
        OUT DX, AL ;issue the char to LCD
```

```
        MOV AL, 00000101B  ;RS=1, R/W=0, E=1 for H-to-L pulse
        MOV DX, PORTB ;port B address
        OUT DX, AL  ;make enable high
        MOV AL, 00000001B  ;RS=1,R/W=0 and E=0 for H-to-L pulse
        OUT DX, AL
        POP DX
        RET
DATWRIT ENDP  ;writing on the lcd ends

BACKSPACE PROC NEAR  ;backspace
        PUSH DX ;save registers
        PUSH AX
        MOV DX,PORTA  ;DX=port A address
        MOV AL,00010000B ;used for shifting cursor to one space right
        OUT DX,AL       ;issue to LCD
        MOV DX,PORTB     ;DX=PORTB address
        MOV AL,00000100B  ;RS=0, R/W=0, E=1 for H-to-L pulse
        OUT DX,AL       ;issue to LCD
        MOV AL,00000000B ;RS=0, R/W=0, E=0 for H-to-L pulse
        OUT DX,AL       ;issue to LCD
        CALL DELAY      ;wait before issuing next command
        CALL DELAY
        MOV AL,' '
        CALL DATWRIT    ;overwrite " "
        CALL DELAY
        CALL DELAY      ;wait before issuing next command
        MOV DX,PORTA     ;code to shift one space right .Same as above.Used for setting back cursor at
proper place after writing " "
        MOV AL,00010000B
        OUT DX,AL
        MOV DX,PORTB
        MOV AL,00000100B
        OUT DX,AL
        MOV AL,00000000B
        OUT DX,AL
        POP AX         ;retrieve registers
        POP DX
```

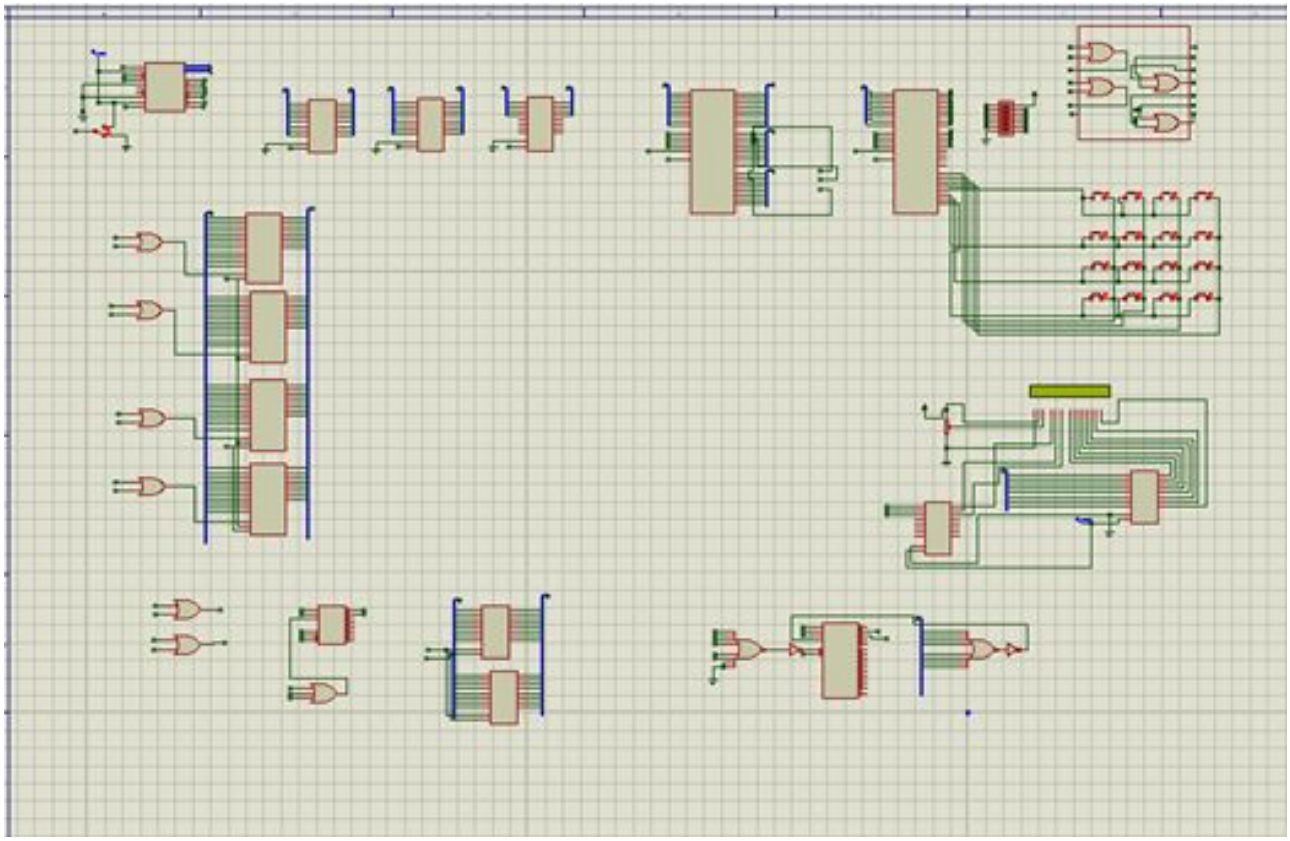
```
        RET
BACKSPACE ENDP



CHECK_7408 PROC NEAR  ;checks if 7408 is good or bad[pass or fail]
        MOV DI,00H
        SEND_INP7408:
        MOV AL,IN_7408[DI]
        OUT PORT2A,AL ;output address
        IN AL,PORT2B ;input address
        AND AL,0FH
        CMP AL,OUT_7408[DI]
        JE NXT_7408
        CALL FAIL
        JMP EP_7408
        NXT_7408:
        CMP DI,03H
        JE PASS_7408
        INC DI
        JMP SEND_INP7408
        PASS_7408:
        CALL PASS
        EP_7408:
        RET
CHECK_7408 ENDP

CHECK_7432 PROC NEAR      ;checks if 7432 is good or bad[pass or fail]
        MOV DI,00H
        SEND_INP7432:
        MOV AL,IN_7432[DI]
        OUT PORT2A,AL ;output address
        IN AL,PORT2B ;input address
        AND AL,0FH
        CMP AL,OUT_7432[DI]
        JE NXT_7432
        CALL FAIL
```

```
        JMP EP_7432
        NXT_7432:
        CMP DI,03H
        JE PASS_7432
        INC DI
        JMP SEND_INP7432
        PASS_7432:
        CALL PASS
        EP_7432:
        RET
CHECK_7432 ENDP


CHECK_7486 PROC NEAR     ;checks if 7486 is good or bad[pass or fail]
        MOV DI,00H
        SEND_INP7486:
        MOV AL,IN_7486[DI]
        OUT PORT2A,AL  ;output address
        IN AL,PORT2B   ;input address
        AND AL,0FH
        CMP AL,OUT_7486[DI]
        JE NXT_7486
        CALL FAIL
        JMP EP_7486
        NXT_7486:
        CMP DI,03H
        JE PASS_7486
        INC DI
        JMP SEND_INP7486
        PASS_7486:
        CALL PASS
        EP_7486:
        RET
CHECK_7486 ENDP


IC_CMP PROC NEAR  ;compares 4-digit{ASCII values} chip numbers in IC_NUM and [BX] for equality
```

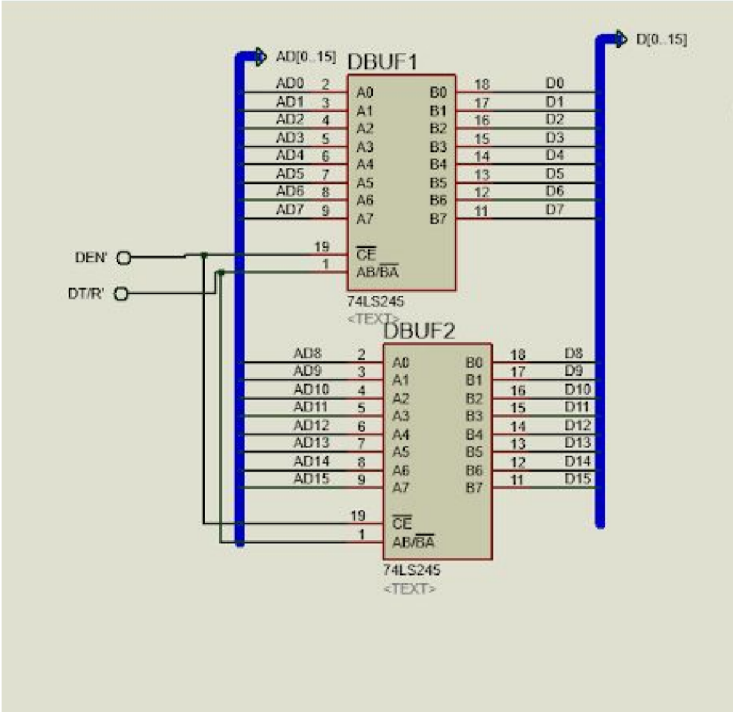
```
                ;If equal then IC_STATUS=01H
    MOV SI,0000H
    CMP_NUM:
    MOV AL,IC_NUM[SI]
    CMP AL,BX[SI]
    JE NXT_NUM
    JMP EP_CMP_IC
    NXT_NUM:
    CMP SI,03H
    JE PASS_CMP_IC
    INC SI
    JMP CMP_NUM
    PASS_CMP_IC:
    MOV IC_STATUS,01H
    EP_CMP_IC:
    RET
IC_CMP ENDP

FAIL PROC NEAR    ;appends fail to LCD output
        PUSHF
        PUSH DI
        MOV CHAR_COUNT,05
        LEA DI,TEXT_FAIL
        CALL WRITE_MEM_APP
        POP DI
        POPF
        RET
FAIL ENDP

PASS PROC NEAR    ;appends pass to LCD output
        PUSHF
        PUSH DI
        MOV CHAR_COUNT,05
        LEA DI,TEXT_PASS
        CALL WRITE_MEM_APP
        POP DI
        POPF
```

```
        RET
PASS ENDP

;delay in the circuit here the delay of 20 millisecond is produced
DELAY PROC
        MOV CX, 1325  ;1325*15.085 usec = 20 msec
        W1:
        NOP
        NOP
        NOP
        NOP
        NOP
        LOOP W1
        RET
DELAY ENDP



D20MS:mov cx,2220 ; delay generated will be approx 0.45 secs
xn:loop xn
ret
END
```

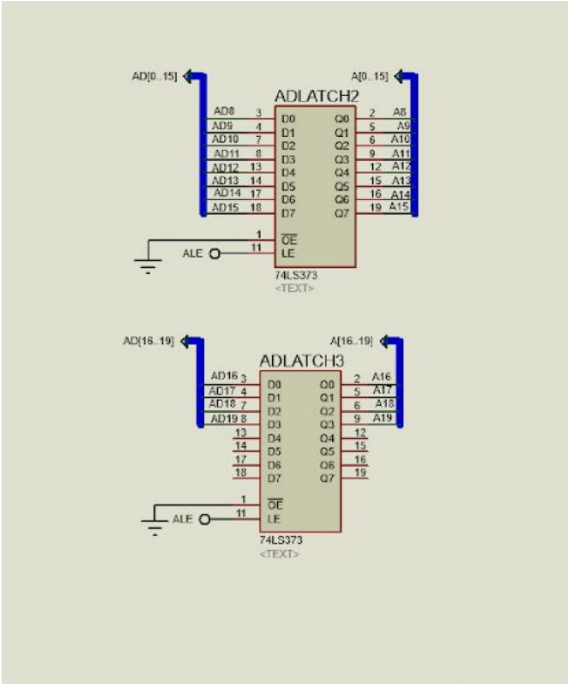
CIRCUIT DIAGRAMS



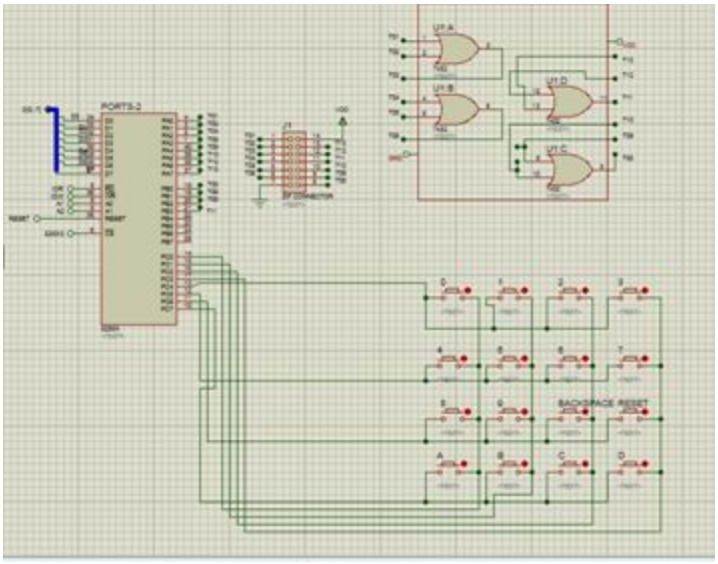
Overall view of schematic for 7432 testing



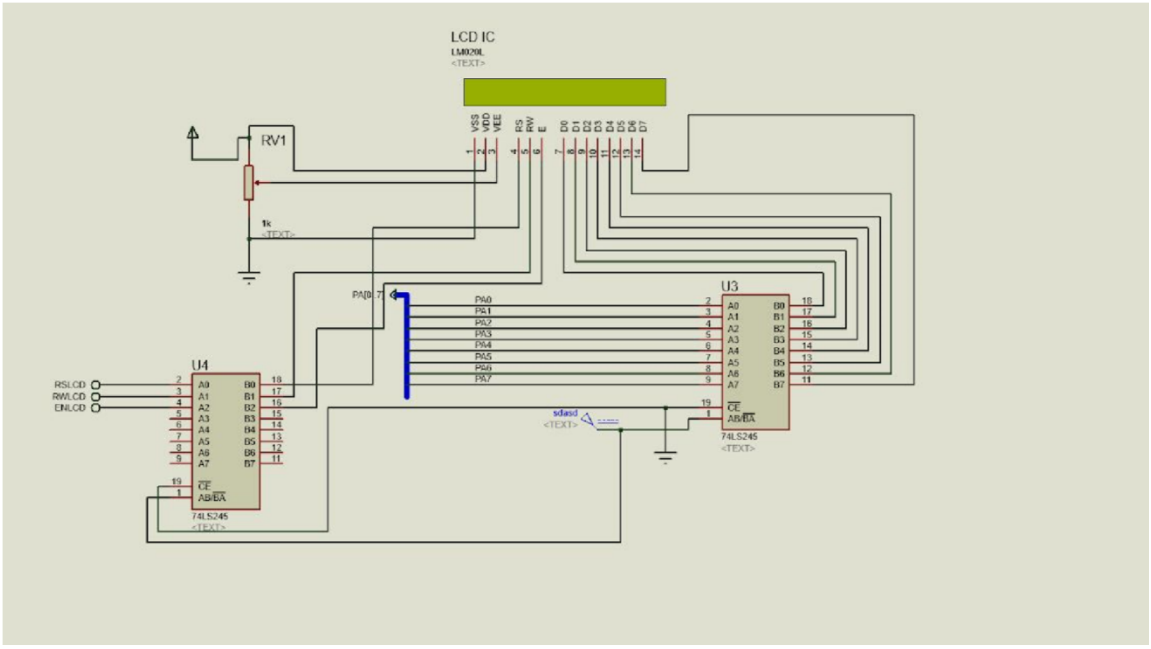
Delay buffers



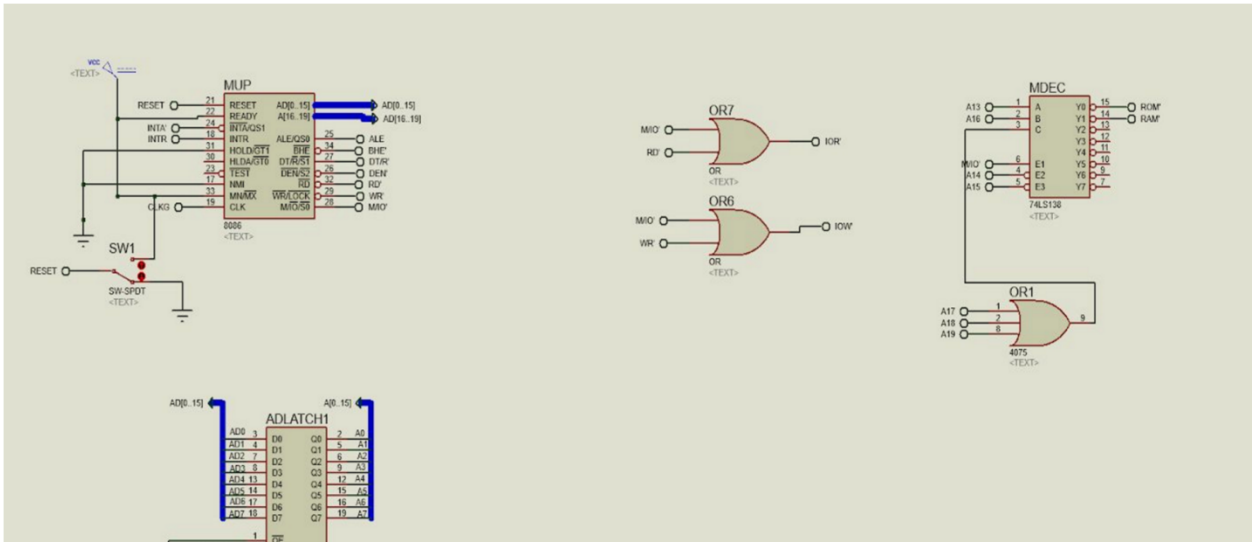
Latches



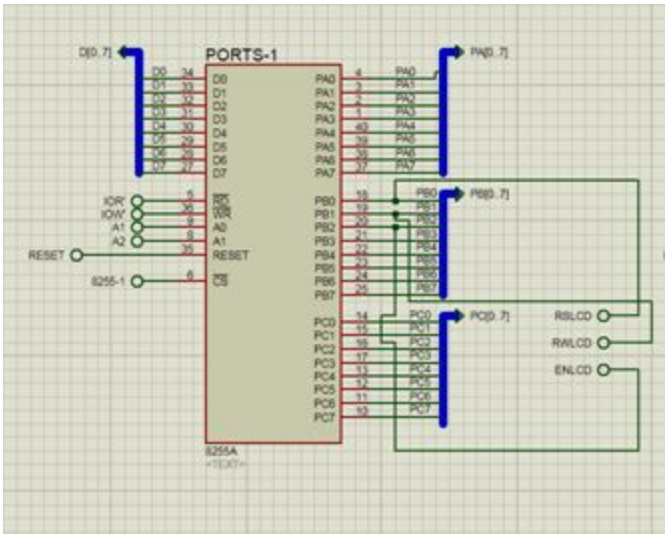
Keyboard with 8255(2) and ZIF Socket



LCD Interfacing



8086 and decoder for CS logic for memory selection



8255(1)