Group Number: 95
Problem Number: 10

Electronic Voting Machine

By

Lavanya Soni	2017A7PS0151P
Sanjeev Singla	2017A7PS0152P
Panchumarthi Pranav	2017A7PS0153P
Abhishek Sharma	2017A7PS0150P

An assignment submitted in partial fulfilment of the course

CS F241 – Microprocessors Programming and Interfacing

BIRLA INSTITUTE OF TECHNOLOGY &SCIENCE
PILANI, RAJASTHAN-333031

April 24,2019

Problem Statement

P10

Design a microprocessor Voting Machine which has provision for 4 candidates. It should keep the count of total votes polled and the count of votes polled for each candidate. Before being put in use, it should check if all memory location allotted to candidates, and the total count are empty. If not, it should clear these as well as the display. To put it in use, it needs to be enabled by 4 polling agents and the Presiding officer. If anyone is missing it should not be enabled. After 8 hours (9 a.m. to 5 p.m.) it should stop taking input. There has to be a provision that the Presiding officer by pressing a code ['CA'] can lock it in between & then can restart it by pressing [FØ]. For retrieving the count of each candidate provision should be there. The count of each candidate has to be send to a remote device using a serial interface. (Do not take into consideration the design of the remote device and its programming)

HARDWAREDEVICES

1. 8086 16-bit Microprocessor
2. 2 X 2732 ROM(8KB)
3. 1 X 6116 SRAM(2KB)
4. 74LS373 Latch with TRI-STATE Outputs
5. 74LS245 Bi-Directional Buffers
6. 2 X 82C55A CHMOS PROGRAMMABLE PERIPHERAL INTERFACE
7. 1 X 8253 PROGRAMMABLE INTERVALTIMER
8. 64 X Push-switches
9. 3 X 74LS138 3-8 Line Decoder
10. 4 X OR (2 input) Gates
11. 4 X NOT Gates
12. LED-RED
13. LCD Display – LM032L
14. 11 X Resistor

HARDWARE SPECIFICATIONS

1. Input is given by user from keypad.
2. Based on this data, the system specifications are as follows:

ITEM	ADDRESS	REMARK
MEMORY		
RAM	00000H-007FFH	1 x 2KB
ROM	00800H-01FFFH	2 x 4KB
8255(#1)		USED TO INTERFACE KEYPAD
PA	00H	Connected to Rows of KeyPad (Input)
PB	02H	Connected to Columns of KeyPad (Output)
PC	04H	PC0 connected to LED (Output)
Creg	06H	
8255(#2)		USED TO INTERFACE LCD
PA	10H	Connected to LCD
PB	12H	
PC	14H	Connected to LCD
Creg	16H	
8253		Programmable Peripheral Interface
Cnt1	30H	
Cnt2	32H	
Cnt3	34H	
Creg	36H	

ASSUMPTIONS:

1) None of the parties get more than 2^{16} votes.

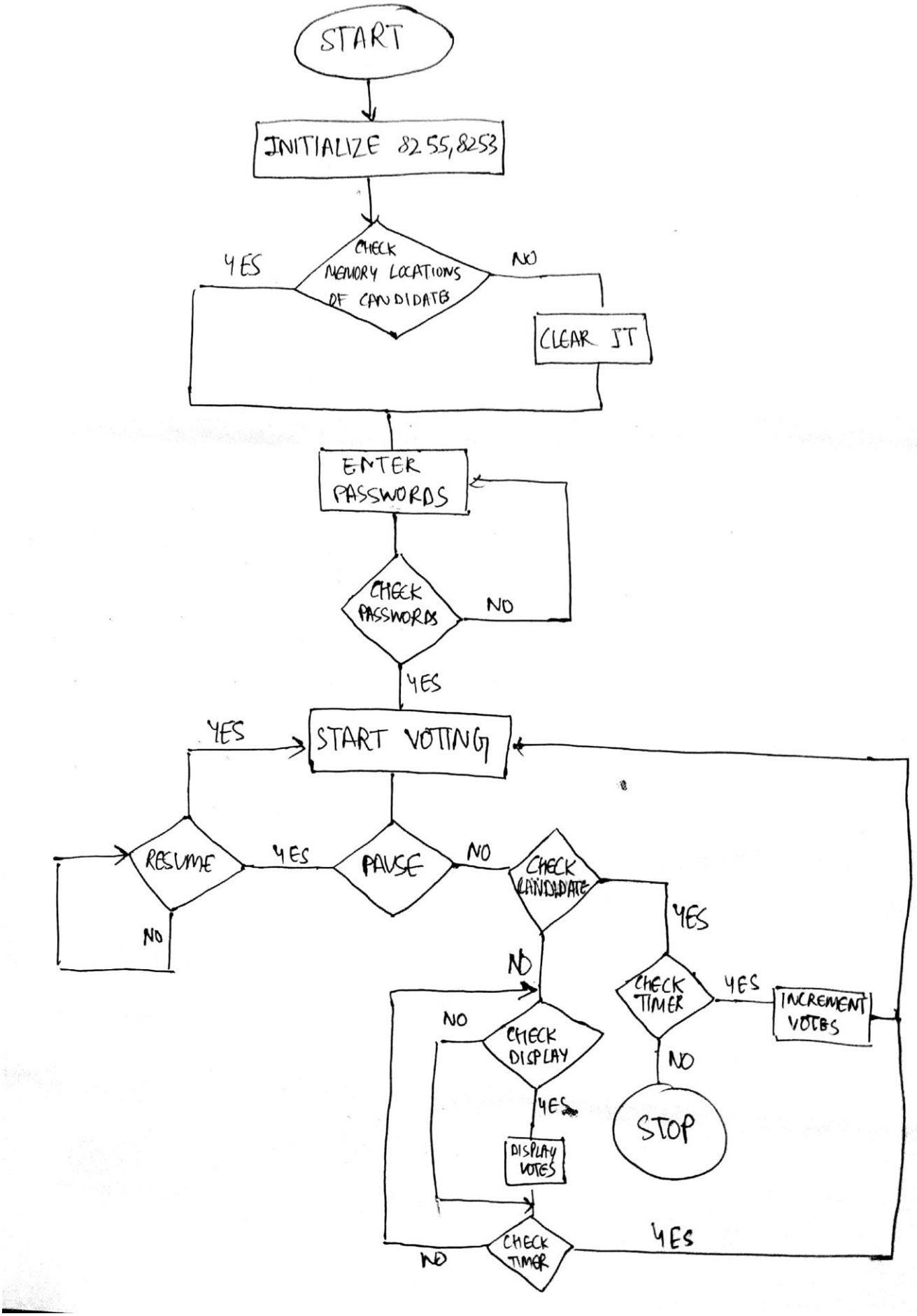
2) We assume that ALP begins at 9A.M. There is no provision to check the actual time.

3) RAM is battery powered and as such gets a continuous power supply even if the external supply is switched off.

4) Voting for a candidate will be done by pressing the respective switch and will be confirmed on the LCD.

5) After every vote, the number of votes received by the respective candidate will be sent serially to a remote device.

FLOWCHART


START
INITIALIZE 8255,8253
CHECK MEMORY LOCATIONS OF CANDIDATES
YES
NO
CLEAR IT
ENTER PASSWORDS
CHECK PASSWORDS
NO
YES
START VOTING
YES
RESUME
YES
PAUSE
NO
CHECK CANDIDATE
YES
NO
NO
CHECK TIMER
YES
INCREMENT VOTES
NO
CHECK DISPLAY
NO
YES
STOP
DISPLAY VOTES
CHECK TIMER
YES
NO

CODE:

```
#make_bin#

#LOAD_SEGMENT=FFFFh#
#LOAD_OFFSET=0000h#

#CS=0000h#
#IP=0000h#

#DS=0000h#
#ES=0000h#

#SS=0000h#
#SP=0FFFEh#

#AX=0000h#
#BX=0000h#
#CX=0000h#
#DX=0000h#
#SI=0000h#
#DI=0000h#
#BP=0000h#




; add your code here
        jmp     st1
        db      125 dup(0)
        ;db     509 dup(0)

;IVT entry for 80H

        dw ISR_INT2
        dw 0000h

                db      892 dup(0)

        ;db     508 dup(0)
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; IVT has to be put here
        ;vector no. 32

portA1                  equ     00h                             ;8255(For KeyPad)
portB1                  equ     02h
portC1                  equ     04h
CW1                     equ     06h

portA2                  equ     10h                             ;8255(For LCD)
portB2                  equ     12h
portC2                  equ     14h
CW2                     equ     16h

cntReg0                 equ     30h                             ;8253(Timer)
cntReg1                 equ     32h
cntReg2         equ     34h
cReg            equ     36h


TABLE_K         DW      0FEFEH,0FEFDH,0FEFBH,0FEF7H,0FEEFH,0FEDFH,0FEBFH,0FE7FH
                DW      0FDFEH,0FDFDH,0FDFBH,0FDF7H,0FDEFH,0FDDFH,0FDBFH,0FD7FH
                DW      0FBFEH,0FBFDH,0FBFBH,0FBF7H,0FBEFH,0FBDFH,0FBBFH,0FB7FH
                DW      0F7FEH,0F7FDH,0F7FBH,0F7F7H,0F7EFH,0F7DFH,0F7BFH,0F77FH
```

```
        DW      0EFFEH,0EFFDH,0EFFBH,0EFF7H,0EFEFH,0EFDFH,0EFBFH,0EF7FH
        DW      0DFFEH,0DFFDH,0DFFBH,0DFF7H,0DFEFH,0DFDFH,0DFBFH,0DF7FH
        DW      0BFFEH,0BFFDH,0BFFBH,0BFF7H,0BFEFH,0BFDFH,0BFBFH,0BF7FH
        DW      7FFEH,7FFDH,7FFBH,7FF7H,7FEFH,7FDFH,7FBFH,7F7FH


DATA_K      DW      0F001H,0F002H,0F003H,0F004H,0F005H,0F006H,0F007H,0F008H
        DW      0E001H,0E002H,0E003H,0E004H,0E005H,0E006H,0E007H,0E008H
        DW      0030H,0031H,0032H,0033H,0034H,0035H,0036H,0037H
        DW      0038H,0039H,000AH,000BH,0041H,0042H,0043H,0044H             ;;10 11->
PAUSE RESUME
        DW      0045H,0046H,0047H,0048H,0049H,004AH,004BH,004CH
                        DW              004DH,004EH,004FH,0050H,0051H,0052H,0053H,0054H
        DW      0055H,0056H,0057H,0058H,0059H,005AH,0021H,0040H
        DW      0023H,0024H,0025H,002CH,002AH,002DH,002FH,003EH



;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; STORES VOTES TALLY
c1votes dw ?
c2votes dw ?
c3votes dw ?
c4votes dw ?
c5votes dw ?
c6votes dw ?
c7votes dw ?
c8votes dw ?


c1key       dw      0F001h
c2key       dw      0F002h
c3key       dw      0F003h
c4key       dw      0F004h
c5key       dw      0F005h
c6key       dw      0F006h
c7key       dw      0F007h
c8key       dw      0F008h
cakey dw    000AH
temp   dw  ?


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; STORES PASSWORDS
passwd1 db   'AAAAA'
passwd2 db   'BBBBB'
passwd3 db   'CCCCC'
passwd4 db   'DDDDD'
passadm db   'EEEEE'
locker  db  'FFFFF'
unlocker db 'GGGGG'


;main program

st1:    sti
; intialize ds, es,ss to start of RAM
      mov     ax,0000h
      mov     ds,ax
              mov     ax,0200h
      mov     es,ax
      mov     ss,ax
      mov     sp,0FFEH        ;Initializing SP

      mov AL,10010000b                  ;To set Port A in inp mode for rows and Port B in out mode for
columns and port c lower as o/p for alarm led
```

```
    out CW1,AL                              ;To initialize 8255 at 00H for Keypad
        mov AL,10000000b            ;To set the Port A, B and C in outp mode
    out CW2,AL                              ;To initialize 8255 at 10H for LCD



;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; MAIN FUNCTION
                jmp START

ALARM:
        call raise_alarm
START:
                mov     dx,0000h                                        ;To initialize memory
locations for count storage with 0000h
                mov     c1votes,dx
                mov             c2votes,dx
                mov             c3votes,dx
                mov             c4votes,dx
                mov             c5votes,dx
                mov             c6votes,dx
                mov             c7votes,dx
                mov             c8votes,dx

                ;call clear_lcd
                call initialise_lcd
                call clear_lcd

    m1:         call disp_c1
                call disp_passwd
                LEA     SI,passwd1              ;To check for candidate 1
                CALL    CHECKPASS
                cmp bp,0h
                jz m1
                call disp_correct

    m2:         call disp_c2
                call disp_passwd
                LEA     SI,passwd2              ;To check for candidate 2
                CALL    CHECKPASS
                cmp bp,0h
                jz m2
                call disp_correct

    m3:         call disp_c3
                call disp_passwd
                LEA     SI,passwd3              ;To check for candidate 3
                CALL    CHECKPASS
                cmp bp,0h
                jz m3
                call disp_correct

    m4:         call disp_c4
                call disp_passwd
                LEA     SI,passwd4              ;To check for candidate 4
                CALL    CHECKPASS
                cmp bp,0h
                jz m4
                call disp_correct

    m9:         call disp_admin
                call disp_passwd
                LEA     SI,passadm              ;To check for officer
                CALL    CHECKPASS
                cmp bp,0h
                jz m9
```

```
			call disp_correct



			mov	AL,00110100b		;Setting counter0 into mode 2
			out		cReg,AL
			;mov al,'/'
			;call write_lcd

			mov	AL,0c4h				;Setting count(2500) in counter0 ???CHANGE
COUNT VALUE TO 2500 ---
			out		cntReg0,AL
			mov	AL,09h
			out		cntReg0,AL
			;mov al,'/'
			;call write_lcd

			mov	AL,01110100b		;Setting counter0 into mode 2
			out		cReg,AL
			;mov al,'/'
			;call write_lcd

			mov	AL,0e8h					;Setting count(2500) in counter0
???CHANGE COUNT VALUE TO 2500 ---
			out		cntReg1,AL
			mov	AL,03h
			out		cntReg1,AL
			;mov al,'/'
			;call write_lcd

			mov	AL,10110100b		;Setting counter0 into mode 2
			out		cReg,AL
			;mov al,'/'
			;call write_lcd

			mov	AL,0A0h					;Setting count(2500) in counter0
???CHANGE COUNT VALUE TO 2500 ---
			out		cntReg2,AL
			mov	AL,8ch
			out		cntReg2,AL
			;mov al,'/'
			;call write_lcd

			call disp_start_voting

	fetchVote:
			;mov al,'/'
			;call write_lcd
			CALL	GetVotingData
			jmp		fetchVote


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; ISR when 'CA' is pressed

isr_1:		;mov al,00h					;checking for press of 'ca'
			;out portB1,al
			;in al,portA1
			;cmp al,0ffh
			;jz y3						;if no key press detected return from
isr
			;call delay
			;mov al,00h
			;out portB1,al
			;in al,portA1
			;cmp al,0ffh
```

```
                ;jz y3                          ;if no key press detected return from
isr

                call disp_ent_pass              ;ask for password
                lea si,locker                   ;locker is the address of password to lock
                call CHECKPASS
                cmp bp,0000h
                jz y4                           ;if wrong password output incorrect
and jump to iret i.e., y4
                ;---CODE ABOVE IS NOT REQUIRED!

                call disp_locked                ;if correct password output then system locked
                jmp y2
                ;actually,if we want, we can compare the i/p key with 'ca', but not needed as intr
occurs only when 'ca' is pressed

    y5:         ;call disp_incorrect  no need as this is already displayed at the end of CHECKPASS
procedure
                ;call raise_alarm
                call disp_locked

    y2:         call getKeyData

                cmp bl,000bh                    ;give the row of 'f$' as second
parameter
                jnz y2

                call disp_ent_pass              ;ask for password
                lea si,unlocker                 ;unlocker is the address of password to
unlock
                call CHECKPASS
                cmp bp,0000h
                jz y5                           ;if wrong password output display
incorrect, jump to y5

                call disp_unlocked              ;if correct output unlocked and get out of isr
                jmp y3

    y4:         ;call disp_incorrect  no need as this is already displayed at the end of CHECKPASS
procedure
                ;call raise_alarm
                call disp_not_locked

    y3:         jmp final

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; FUNCTION TO GET THE DATA FROM THE KEY PRESSED
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


getKeyData proc near

;;;;;;;;;;;;;;;;; REMEMBER TO INITIALIZE THE PORT B WITH ALL ZEROS;;;;;;;;;;;;;;;;;;;;;;;;;;;

                ;??? OUTPUT 0 TO ALL COLUMNS---
                MOV AL,00H
                OUT portB1,al
key_release1:
                in al,portA1
                cmp al,0FFh
                jnz key_release1

keypad_check:

            ;???TO CHECK FOR KEY PRESS FIRST OUTPUT 00 TO ALL COLS--
            ;mov al,'3'
```

```
;call write_lcd
MOV AL,00H
OUT portB1,al
in      al,portA1                       ;To check the key press
cmp     al,0FFh
jz keypad_check                                 ; ???SHOULD BE : jz keypad_check---

;call delay2                            ;To set the delay to counter the debouncing
;???TO CHECK FOR KEY PRESS FIRST OUTPUT 00 TO ALL COLS---
MOV AL,00H
OUT portB1,al

in      al,portA1
cmp     al,0FFh
jz      keypad_check                    ;???SHOULD BE - jz keypad_check---


mov     al,0FEh                         ;To check the cloumn 0 press
mov     bl,al
out     portB1,al
in      al,portA1
cmp     al,0FFh
jnz     getData

mov     al,0FDh                         ;To check the cloumn 1 press
mov     bl,al
out     portB1,al
in      al,portA1
cmp     al,0FFh
jnz     getData

mov     al,0FBh                         ;To check the cloumn 2 press
mov     bl,al
out     portB1,al
in      al,portA1
cmp     al,0FFh
jnz     getData

mov     al,0F7h                         ;To check the cloumn 3 press
mov     bl,al
out     portB1,al
in      al,portA1
cmp     al,0FFh
jnz     getData

mov     al,0EFh                         ;To check the cloumn 4 press
mov     bl,al
out     portB1,al
in      al,portA1
cmp     al,0FFh
jnz     getData

mov     al,0DFh                         ;To check the cloumn 6 press    ???SHOULD BE 0DFH--
-
mov     bl,al
out     portB1,al
in      al,portA1
cmp     al,0FFh
jnz     getData

mov     al,0BFh                         ;To check the cloumn 7 press    ???SHOULD BE 0BFH---
mov     bl,al
out     portB1,al
in      al,portA1
cmp     al,0FFh
```

```
                jnz     getData

                mov     al,7Fh                          ;To check the cloumn 8 press    ???SHOULD BE 07FH---
                mov     bl,al
                out     portB1,al
                in      al,portA1
                cmp     al,0FFh
                jnz     getData
                jmp     keypad_check

getData:
                mov     bh,al                           ;The BX register now contains HEX code for the
keyPress
                mov al,bl
                mov     CX,64d                          ;The total number of entries in the Table ???ADDED 'd'
TO INDICATE DECIMAL---
                mov     DI,00h
x4:
                cmp     BX,TABLE_K[DI]
                jz      x5
                inc     DI
                inc     DI
                loop x4
x5:
                mov bx,DI
                mov si,offset DATA_K
                mov al,[si+bx]
                mov ah,[si+bx+1]
                mov    bx,ax                            ;BX NOW HAS THE 16BIT DATA KEY VALUE

                RET
getKeyData endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; FUNCTION FOR THE DELAY
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
delay           proc    near
                        push    cx
                        mov     cx,900d
dl1:            nop             ;3 CLK CYCLES

                        loop    dl1     ;18 CYCLES IF JMP IS TAKEN, 5 OTHER WISE
                        pop     cx      ;by now 3*900+18*899+5=18887 cycles done. DELAY GENERATED
BASED ON CLK(10mhz)=1.8887ms
                        ret
delay           endp

delay2                  proc    near
                        push    cx
                        mov     cx,3000d
dl2:            nop             ;3 CLK CYCLES
                        nop             ;3 CLK CYCLES
                        loop    dl2     ;18 CYCLES IF JMP IS TAKEN, 5 OTHER WISE
                        pop     cx      ;by now 6*3000+18*2999+5= 18000+53982+5cycles done. DELAY
GENERATED BASED ON CLK(10mhz)=1.8887ms
                        ret
delay2                  endp

delay3                  proc    near
                        push cx
                                mov cx,20
        looper:         call delay
                                call delay
                                call delay
                                loop looper
                        pop cx
```

```
                        ret
delay3                  endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; POLLED VOTES AT THE END
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; parameters: offest for the
CHECKPASS   proc    near
                        mov temp,si
                        mov             bp,0003h        ;3 chances for the input ???3H OR 3d (DECIMAL)

candidate_auth:
        ;push cx
                        mov si,temp
                        call clear_lcd
                        mov             dx,0005h                ;5 characters in the password
                passwordinp:

                        push si     ; getKeyData will use si,cx hence push onto stack
                        call getKeyData
                        mov al,'*'
                        call write_lcd
                        pop si

                        cmp     bl,ds:[SI]
                        jnz     wronginp1
                        inc     si
                        dec     dx
                        cmp     dx,0h
                        jnz     passwordinp
                        ;pop cx
                        ret
                wronginp1:

                        CALL disp_incorrect                             ; CALL LCD TO DISPLAY
'INCORRECT'
                        ;pop cx
                        dec     bp
                        cmp     bp,0h
                        jnz             candidate_auth                  ;CALL RAISE ALARM (Raise
the LED and restart the system). This is done at the end of taking i/p from each candidate
                        call raise_alarm
                        ret
CHECKPASS  endp
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; GETTING VOTES FROM KEYPAD

GetVotingData proc    near
                                CALL getKeyData
                                ;mov ax,'h'
                                ;call write_lcd
                                cmp  bx,c1key
                                jnz  candidate2
                                call disp_voted
                                inc  c1votes                            ;???compare c1votes 270f
                                mov ax,c1votes
                                add al,30h
                                ;call write_lcd
                                jmp      final
                candidate2:
                                cmp  bx,c2key
                                jnz  candidate3
                                call disp_voted
                                inc  c2votes
                                mov ax,c2votes
```

```
                        add al,30h
                        ;call write_lcd
                        jmp     final
        candidate3:
                        cmp  bx,c3key
                        jnz  candidate4
                        call disp_voted
                        inc  c3votes
                        mov ax,c3votes
                        add al,30h
                        ;call write_lcd
                        jmp     final
        candidate4:
                        cmp  bx,c4key
                        jnz  display1
                        call disp_voted
                        inc  c4votes
                        mov ax,c4votes
                        add al,30h
                        jmp final
                        ;call write_lcd

        display1:
                        cmp  bx,c5key
                        jnz  display2
                        MOV dx,c1votes
                        call clear_lcd
                        call convertToBCD
                        call disp_vote_cnt
                        jmp     final
        display2:
                        cmp  bx,c6key
                        jnz  display3
                        MOV dx,c2votes
                        call clear_lcd
                        call convertToBCD
                        call disp_vote_cnt
                        jmp     final

        display3:
                        cmp  bx,c7key
                        jnz  display4
                        MOV dx,c3votes
                        call clear_lcd
                        call convertToBCD
                        call disp_vote_cnt
                        jmp     final
        display4:
                        cmp  bx,c8key
                        jnz  ca
                        MOV dx,c4votes
                        call clear_lcd
                        call convertToBCD
                        call disp_vote_cnt
                        jmp     final

        ca:
                        cmp  bx,cakey
                        jnz  final
                        jmp isr_1
                        ;jmp     final
        final:
                        ret
GetVotingData endp
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; LCD functions
initialise_lcd  proc near

                                mov al,10000000b            ;Initialising 8255(2)
                                out CW2,al                  ;portA2 is o/p and is connected to data lines
d0-d7 and pc7-pc5 are o/p and connected to rs,r/w and e resp.

                                ;STEP 1
                                mov al,00110000b            ;PAGE 40 STEP NO.2
                                out portA2,al               ;function setting over
                                mov al, 00100000b           ;set E=0;setting to 8bit operation and 1 line
i.e.,function setting           ;pc7,pc6-0,p5=1
                                out portC2,al
                                mov al, 00000000b           ;set E=1;
                                out portC2,al
                                call delay

                                ;STEP2
                                mov al,00001110b
                                out portA2,al               ;display switched on
                                mov al,00100000b            ;switching on display           ;E=0
                                out portC2,al
                                mov al, 00000000b           ;set E=1;
                                out portC2,al
                                call delay

                                ;STEP3
                                mov al,00000110b            ;setting address to increment by 1 and diplay
shifting enabled i.e., entry mode set
                                out portA2,al               ;??? SHOULDNT IT BE 0000 0110B ?---
                                mov al,00100000b            ;E=0
                                out portC2,al
                                mov al, 00000000b           ;set E=1;
                                out portC2,al
                                call delay
                                ret
initialise_lcd  endp

write_lcd               proc near                           ;before calling this, al must be loaded with
the ascii value
                                out portA2,al               ;data written to lcd
                                mov al,10100000b            ;setting to write mode          ;pc7-1,pc6-
0,p5=0
                                out portC2,al
                                mov al, 10000000b           ;set E=1;
                                out portC2,al
                                push cx
                                mov cx,10
                d:              call delay
                                call delay
                                call delay
                                loop d
                                pop cx
                                ret
write_lcd               endp

clear_lcd               proc near
                                mov al,00000001b
                                out portA2,al               ;lcd cleared
                                mov al,00100000b            ;clearing display of lcd        ;pc7,pc6-
0,p5=0
                                out portC2,al
                                mov al, 00000000b           ;set E=1;
```

```
                        out portC2,al
                        call delay
                        ret
clear_lcd               endp

disp_locked   proc near
                        call clear_lcd          ;???CALL CLEAR LCD
                        call delay
                        call delay
                        mov al,'L'
                        call write_lcd
                        mov al,'o'              ;??? PREVIOUSLY '0'---
                        call write_lcd
                        mov al,'c'
                        call write_lcd
                        mov al,'k'
                        call write_lcd
                        mov al,'e'
                        call write_lcd
                        mov al,'d'
                        call write_lcd
                        ret
disp_locked   endp

disp_unlocked         proc near
                        call clear_lcd          ;CALL CLEAR_LCD
                        call delay
                        call delay
                        mov al,'U'
                        call write_lcd
                        mov al,'n'
                        call write_lcd
                        mov al,'l'
                        call write_lcd
                        mov al,'o'                      ;??? PREVIOUSLY '0'---
                        call write_lcd
                        mov al,'c'
                        call write_lcd
                        mov al,'k'
                        call write_lcd
                        mov al,'e'
                        call write_lcd
                        mov al,'d'
                        call write_lcd
                        call delay3
                        call clear_lcd
                        ret
disp_unlocked         endp

disp_correct  proc near
                        call clear_lcd          ;??? CALL CLEAR_LCD
                        call delay
                        call delay
                        mov al,'C'
                        call write_lcd
                        mov al,'o'
                        call write_lcd
                        mov al,'r'
                        call write_lcd
                        mov al,'r'
                        call write_lcd
                        mov al,'e'
                        call write_lcd
                        mov al,'c'
                        call write_lcd
```

```
                        mov al,'t'
                        call write_lcd
                        call delay3
                        ret
disp_correct   endp

disp_incorrect          proc near
                        call clear_lcd              ;CALL CLEAR LCD
                        call delay
                        call delay
                        mov al,'I'
                        call write_lcd
                        mov al,'n'
                        call write_lcd
                        mov al,'c'
                        call write_lcd
                        mov al,'o'
                        call write_lcd
                        mov al,'r'
                        call write_lcd
                        mov al,'r'
                        call write_lcd
                        mov al,'e'
                        call write_lcd
                        mov al,'c'
                        call write_lcd
                        mov al,'t'
                        call write_lcd
                        call delay3
                        ret
disp_incorrect          endp

disp_ent_passproc near
                        call clear_lcd              ;???CALL
                        call delay
                        call delay
                        mov al,'E'
                        call write_lcd
                        mov al,'n'
                        call write_lcd
                        mov al,'t'
                        call write_lcd
                        mov al,'e'
                        call write_lcd
                        mov al,'r'
                        call write_lcd
                        mov al,' '
                        call write_lcd
                        mov al,'P'
                        call write_lcd
                        mov al,'a'
                        call write_lcd
                        mov al,'s'
                        call write_lcd
                        mov al,'s'
                        call write_lcd
                        mov al,'w'
                        call write_lcd
                        mov al,'o'
                        call write_lcd
                        mov al,'r'
                        call write_lcd
                        mov al,'d'
                        call write_lcd
                        call delay3
```

```
                        ret
disp_ent_passendp

disp_c1                 proc near
                        call clear_lcd          ;???CALL
                        call delay
                        call delay
                        mov al,'P'
                        call write_lcd
                        mov al,'1'
                        call write_lcd
                        ret
disp_c1                 endp

disp_c2                 proc near
                        call clear_lcd          ;???CALL
                        call delay
                        call delay
                        mov al,'P'
                        call write_lcd
                        mov al,'2'
                        call write_lcd
                        ret
disp_c2                 endp

disp_c3                 proc near
                        call clear_lcd          ;???CALL
                        call delay
                        call delay
                        mov al,'P'
                        call write_lcd
                        mov al,'3'
                        call write_lcd
                        ret
disp_c3                 endp

disp_c4                 proc near
                        call clear_lcd          ;???CALL
                        call delay
                        call delay
                        mov al,'P'
                        call write_lcd
                        mov al,'4'
                        call write_lcd
                        ret
disp_c4                 endp

disp_c5                 proc near
                        call clear_lcd          ;???CALL
                        call delay
                        call delay
                        mov al,'P'
                        call write_lcd
                        mov al,'5'
                        call write_lcd
                        ret
disp_c5                 endp

disp_c6                 proc near
                        call clear_lcd          ;???CALL
                        call delay
                        call delay
                        mov al,'P'
                        call write_lcd
                        mov al,'6'
```

```
                        call write_lcd
                        ret
disp_c6                 endp

disp_c7                 proc near
                        call clear_lcd          ;???CALL
                        call delay
                        call delay
                        mov al,'P'
                        call write_lcd
                        mov al,'7'
                        call write_lcd
                        ret
disp_c7                 endp

disp_c8                 proc near
                        call clear_lcd          ;???CALL
                        call delay
                        call delay
                        mov al,'P'
                        call write_lcd
                        mov al,'8'
                        call write_lcd
                        ret
disp_c8                 endp

disp_admin          proc near
                        call clear_lcd          ;???CALL
                        call delay
                        call delay
                        mov al,'A'
                        call write_lcd
                        mov al,'d'
                        call write_lcd
                        mov al,'m'
                        call write_lcd
                        mov al,'i'
                        call write_lcd
                        mov al,'n'
                        call write_lcd
                        ret
disp_admin          endp

disp_voted          proc near
                        call clear_lcd          ;???CALL
                        call delay
                        call delay
                        mov al,'V'
                        call write_lcd
                        mov al,'o'
                        call write_lcd
                        mov al,'t'
                        call write_lcd
                        mov al,'e'
                        call write_lcd
                        mov al,'d'
                        call write_lcd
                        call clear_lcd
                        ret
disp_voted      endp

disp_candidate      proc near
                        call clear_lcd          ;???CALL
                        call delay
                        call delay
```

```
                        mov al,'C'
                        call write_lcd
                        mov al,'A'
                        call write_lcd
                        mov al,'N'
                        call write_lcd
                        mov al,'D'
                        call write_lcd
                        mov al,'I'
                        call write_lcd
                        mov al,'D'
                        call write_lcd
                        mov al,'A'
                        call write_lcd
                        mov al,'T'
                        call write_lcd
                        mov al,'E'
                        call write_lcd
                        mov al,' '
                        call write_lcd
                        ret
disp_candidate      endp

disp_v1                 proc near
                        mov al,'1'
                        call write_lcd
                        mov al,':'
                        call write_lcd
                        mov al,' '
                        call write_lcd
                        ret
disp_v1                 endp

disp_v2                 proc near
                        mov al,'2'
                        call write_lcd
                        mov al,':'
                        call write_lcd
                        mov al,' '
                        call write_lcd
                        ret
disp_v2                 endp

disp_v3                 proc near
                        mov al,'3'
                        call write_lcd
                        mov al,':'
                        call write_lcd
                        mov al,' '
                        call write_lcd
                        ret
disp_v3                 endp

disp_v4                 proc near
                        mov al,'4'
                        call write_lcd
                        mov al,':'
                        call write_lcd
                        mov al,' '
                        call write_lcd
                        ret
disp_v4                 endp

disp_v5                 proc near
                        mov al,'5'
```

```
                        call write_lcd
                        mov al,':'
                        call write_lcd
                        mov al,' '
                        call write_lcd
                        ret
disp_v5                 endp

disp_v6                 proc near
                        mov al,'6'
                        call write_lcd
                        mov al,':'
                        call write_lcd
                        mov al,' '
                        call write_lcd
                        ret
disp_v6                 endp

disp_v7                 proc near
                        mov al,'7'
                        call write_lcd
                        mov al,':'
                        call write_lcd
                        mov al,' '
                        call write_lcd
                        ret
disp_v7                 endp

disp_v8                 proc near
                        mov al,'8'
                        call write_lcd
                        mov al,':'
                        call write_lcd
                        mov al,' '
                        call write_lcd
                        ret
disp_v8                 endp


disp_passwd         proc near
                        ;call clear_lcd             ;???CALL
                        mov al,' '
                        call write_lcd
                        mov al,'P'
                        call write_lcd
                        mov al,'a'
                        call write_lcd
                        mov al,'s'
                        call write_lcd
                        mov al,'s'
                        call write_lcd
                        mov al,'w'
                        call write_lcd
                        mov al,'o'
                        call write_lcd
                        mov al,'r'
                        call write_lcd
                        mov al,'d'
                        call write_lcd
                        call delay3
                        ret
disp_passwd         endp

disp_start_voting proc near
                        call clear_lcd
```

```
                        call delay
                        call delay
                        mov al,'S'
                        call write_lcd
                        mov al,'t'
                        call write_lcd
                        mov al,'a'
                        call write_lcd
                        mov al,'r'
                        call write_lcd
                        mov al,'t'
                        call write_lcd
                        mov al,' '
                        call write_lcd
                        mov al,'V'
                        call write_lcd
                        mov al,'o'
                        call write_lcd
                        mov al,'t'
                        call write_lcd
                        mov al,'i'
                        call write_lcd
                        mov al,'n'
                        call write_lcd
                        mov al,'g'
                        call write_lcd
                        call delay3
                        call clear_lcd
                        ret
disp_start_voting endp

disp_not_locked     proc near
                                call clear_lcd
                                call delay
                                call delay
                                mov al,'N'
                                call write_lcd
                                mov al,'o'
                                call write_lcd
                                mov al,'t'
                                call write_lcd
                                mov al,' '
                                call write_lcd
                                mov al,'L'
                                call write_lcd
                                mov al,'o'
                                call write_lcd
                                mov al,'c'
                                call write_lcd
                                mov al,'k'
                                call write_lcd
                                mov al,'e'
                                call write_lcd
                                mov al,'d'
                                call write_lcd
                                call delay3
                                ret
disp_not_locked     endp

disp_alarm          proc near
                        call clear_lcd              ;???CALL
                        call delay
                        call delay
                        mov al,'A'
                        call write_lcd
```

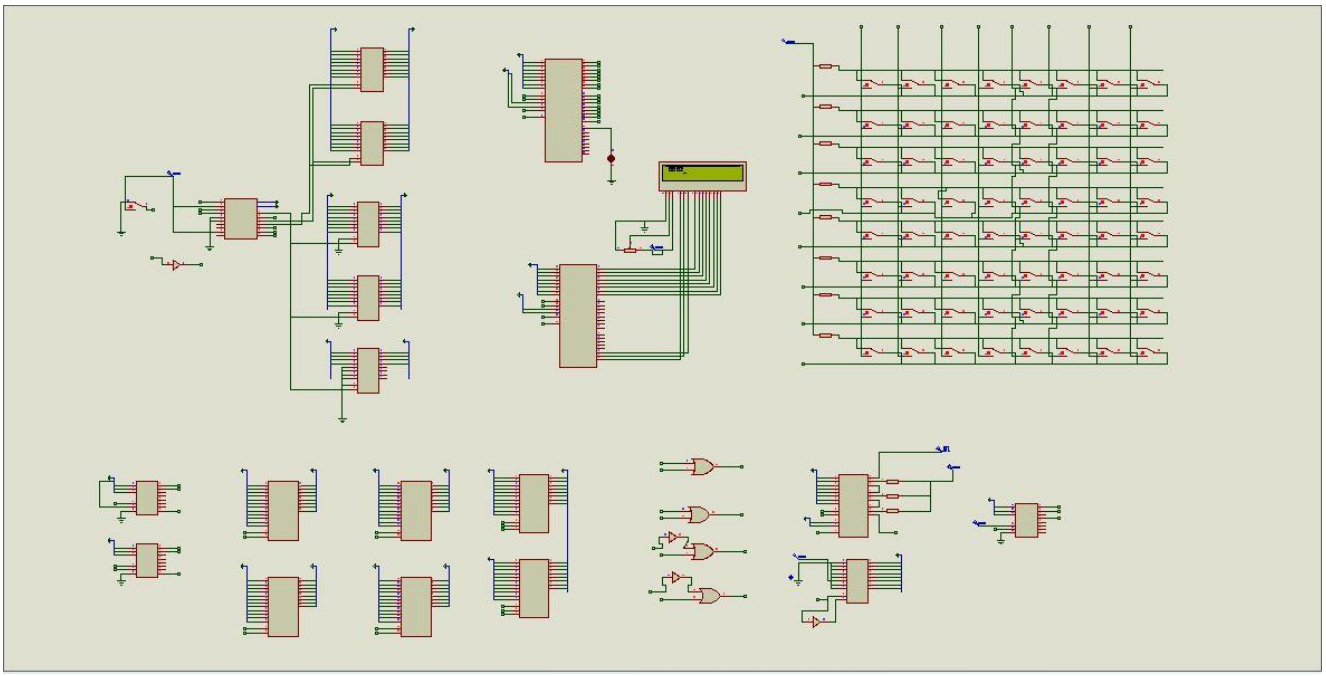
```
                        mov al,'L'
                        call write_lcd
                        mov al,'A'
                        call write_lcd
                        mov al,'R'
                        call write_lcd
                        mov al,'M'
                        call write_lcd
                        mov al,'!'
                        call write_lcd
                        push cx
                        mov cx,10
    loo2:       call delay
                        call delay
                        call delay
                        loop loo2
                        pop cx
                        ret
disp_alarm          endp

raise_alarm         proc near
        CALL disp_alarm
                        mov al,00000001b  ;bsr mode of portc to raise the alarm led
                        out portC1,al
                        push cx
                        mov cx,30
    loo:        call delay
                        call delay
                        call delay
                        loop loo
                        pop cx
                        mov al,00000000b ;bsr mode of portc to lower the alarm led
                        out portC1,al
                        ret
raise_alarm         endp

ISR_INT2:
    ;mov al,'L'
    ;call write_lcd
    keepPolling:
    CALL getKeyData
    MOV DI,00H
    MOV CX,8D                                                                   ;???is it for
skipping first 8 keys only used to vote
POSITION:
    INC DI
    INC DI
    LOOP POSITION
    MOV DX,DATA_K[DI]
    CMP BX,DX
    JNZ CANDIDATE_2
    call disp_candidate
    call disp_v1
    MOV dx,c1votes
    call convertToBCD
    call disp_vote_cnt
    ;CALL disp_votes
    JMP keepPolling

CANDIDATE_2:
    INC DI
    INC DI
    MOV DX,DATA_K[DI]
    CMP BX,DX
    JNZ CANDIDATE_3
```

```
        call disp_candidate
        call disp_v2
        MOV dx,c2votes
        call convertToBCD
        call disp_vote_cnt
        ;CALL disp_votes
        JMP keepPolling

CANDIDATE_3:
        INC DI
        INC DI
        MOV DX,DATA_K[DI]
        CMP BX,DX
        JNZ CANDIDATE_4
        call disp_candidate
        call disp_v3
        MOV dx,c3votes
        call convertToBCD
        call disp_vote_cnt
        ;CALL disp_votes
        JMP keepPolling

CANDIDATE_4:
        INC DI
        INC DI
        MOV DX,DATA_K[DI]
        CMP BX,DX
        JNZ keepPolling
        call disp_candidate
        call disp_v4
        MOV dx,c4votes
        call convertToBCD
        call disp_vote_cnt
        ;CALL disp_votes
        JMP keepPolling




        IRET


convertToBCDproc    near
                        mov             AX,0000H
                        mov             CX,0

                d1:

                        inc     cx

                        SHL     AX,1
                        SHL      DX,1                   ;Hex Data to be converted to BCD form is in
DX
                        jnc          d2
                        inc       AX
                        d2:

                        cmp     cx,16
                        jz           finish

                        mov     bl,ah               ;BIN-4
                        and     bl,0F0h
                        rol     bl,1
                        rol     bl,1
                        rol     bl,1
```

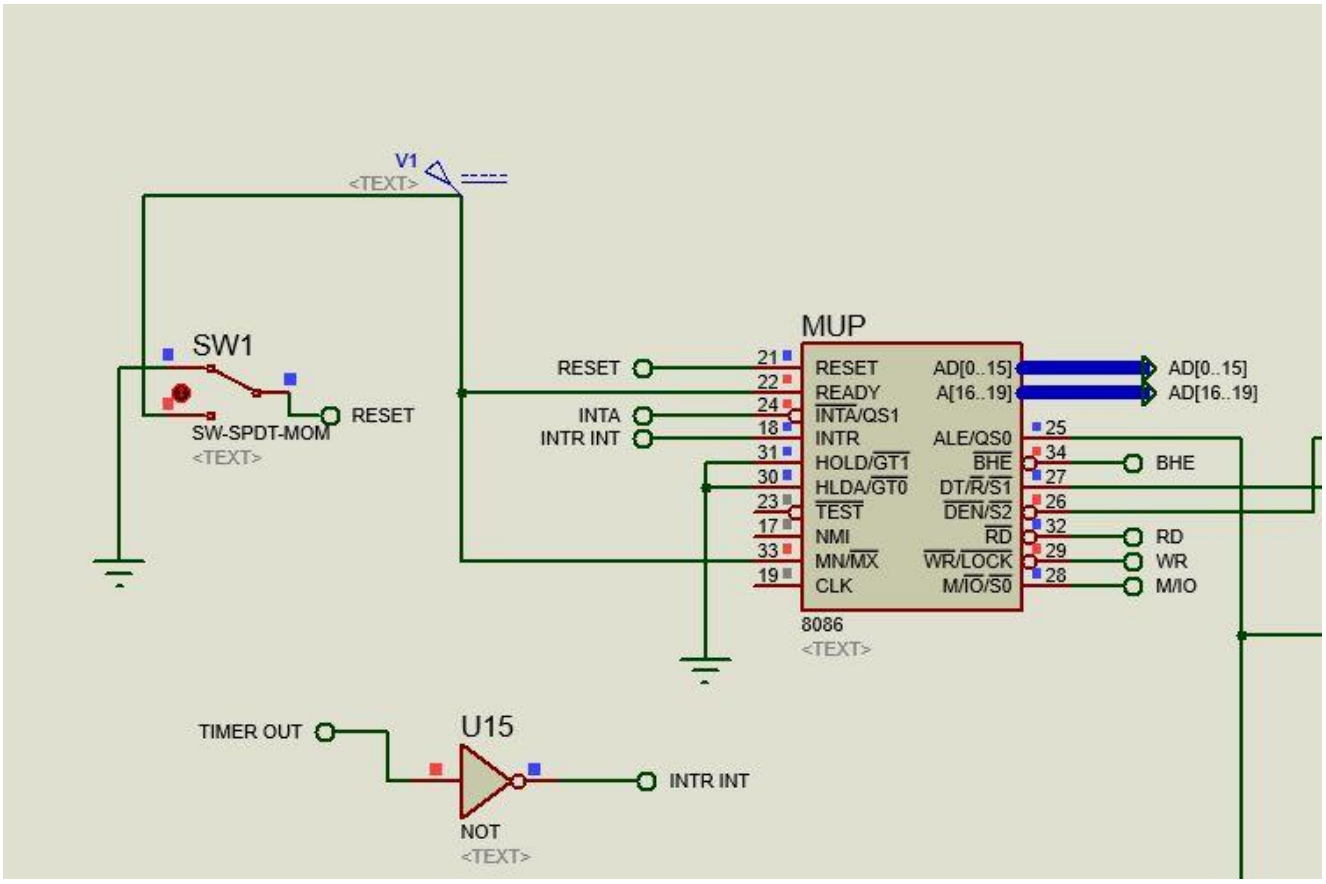
```
                        rol    bl,1
                        rol    bl,1
                        cmp    bl,5
                        jb    d5
                        add    AX,0030h
                d5:
                        mov        bl,al          ;BIN-1
                        and    bl,0Fh
                        cmp    bl,5
                        jb    d6
                        add    AX,0003h

                d6:

                        cmp   cx,16
                        ja finish
                        jmp d1
                        finish:
                        ret
convertToBCD        endp

disp_vote_cnt       proc near
                        mov    cx,AX
                        mov    bh,ah
                        and    bh,0F0h
                        rol    bh,1
                        rol    bh,1
                        rol    bh,1
                        rol    bh,1
                        mov    AX,cx

                        mov    cx,AX
                        add     bh,30h
                        mov    al,' '
                        call   write_lcd
                        mov    al,bh            ; To display one character
                        call   write_lcd
                        mov    AX,cx

                        mov    cx,AX
                        mov    bh,ah
                        and    bh,0Fh
                        mov    AX,cx

                        mov    cx,AX
                        add     bh,30h
                        mov      al,bh          ; To display one character
                        call   write_lcd
                        mov    AX,cx

                        mov    cx,AX
                        mov    bh,al
                        and    bh,0F0h
                        rol    bh,1
                        rol    bh,1
                        rol    bh,1
                        rol    bh,1
                        mov    AX,cx

                        mov    cx,AX
                        add     bh,30h
                        mov      al,bh          ; To display one character
                        call   write_lcd
                        mov    AX,cx
```

```
                        mov    cx,AX
                        mov    bh,al
                        and    bh,0Fh
                        mov    AX,cx

                        mov    cx,AX
                        add    bh,30h
                        mov     al,bh           ; To display one character
                        call   write_lcd
                        mov    AX,cx
                        push cx
                        mov cx,30
    loo3:           call delay
                        call delay
                        call delay
                        loop loo
                        pop cx
                        call clear_lcd
                        ret
disp_vote_cnt       endp
```

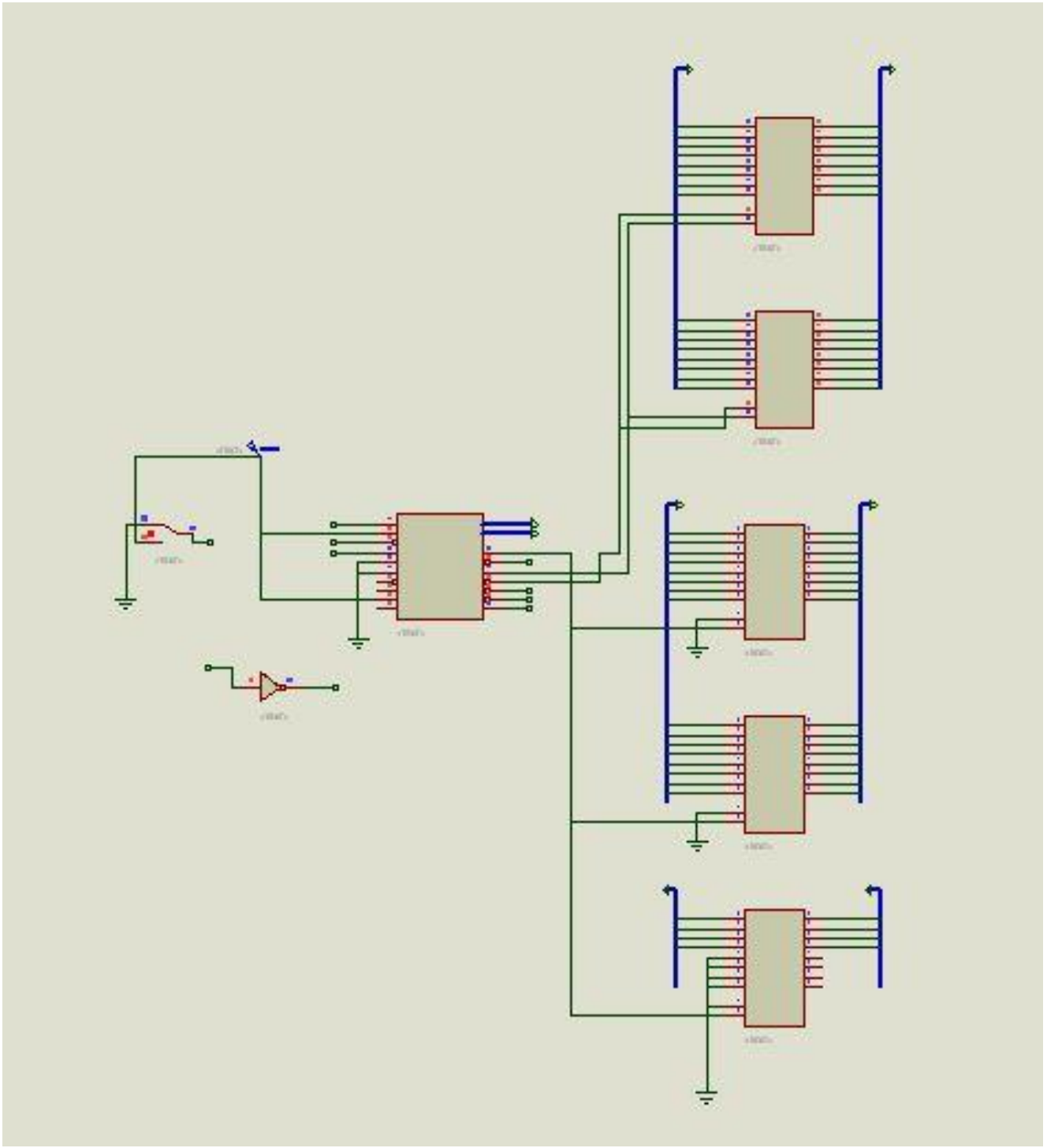
CIRCUIT DESIGN



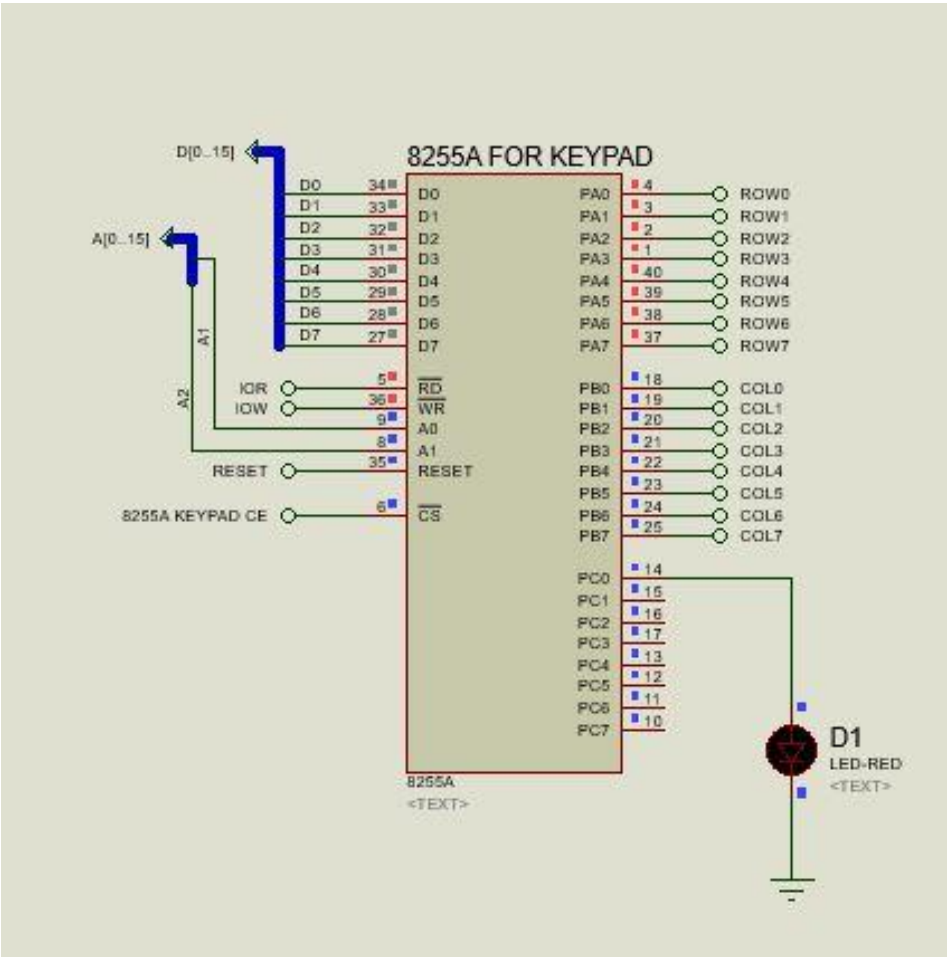
FULL Circuit

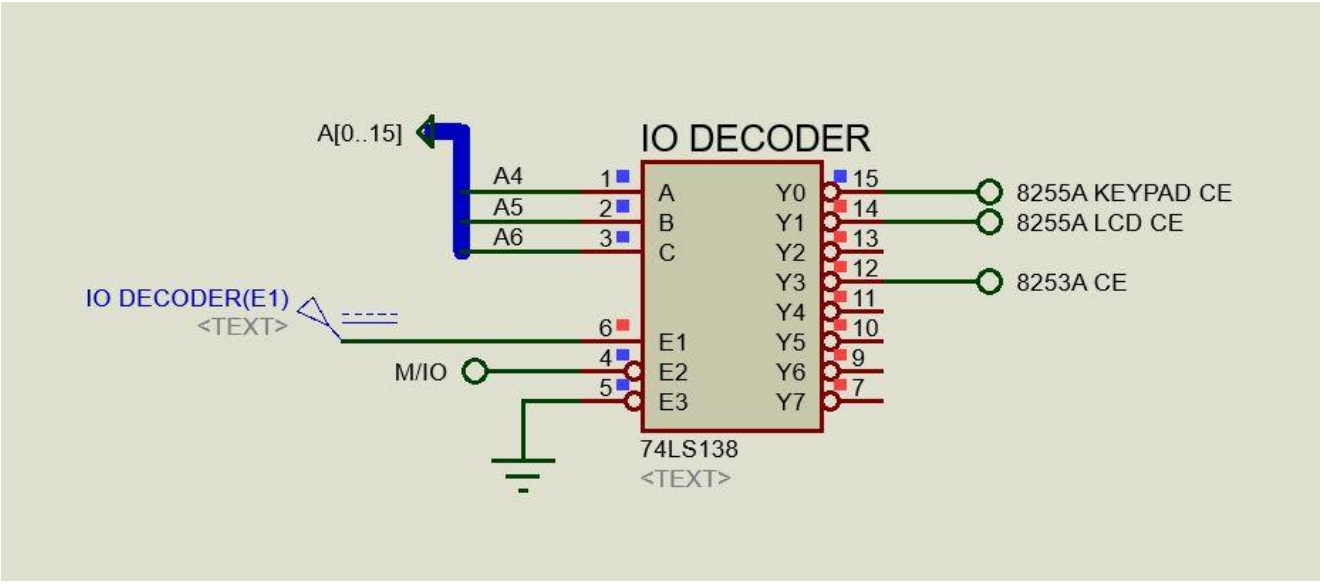


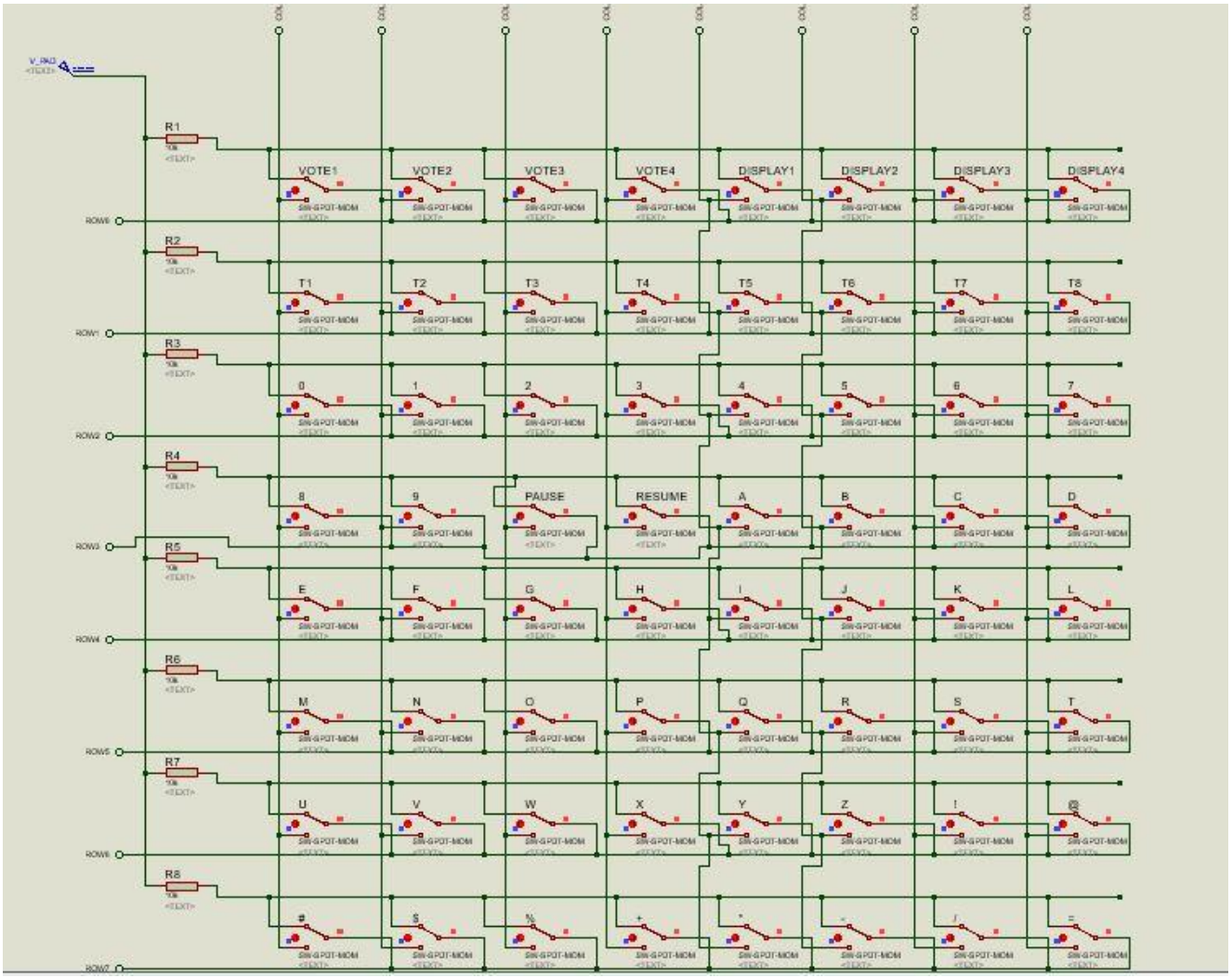
Main 8086



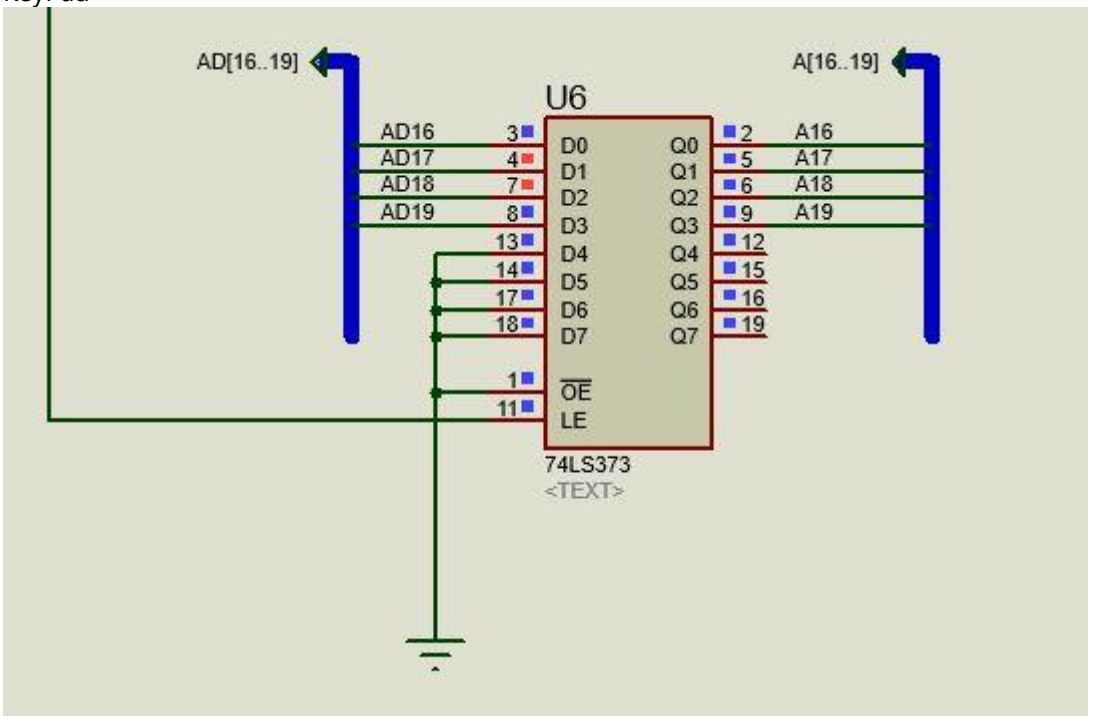
Main 8086 Interfacing


8255A FOR KEYPAD
D[0..15]
A[0..15]
A1
A2
IOR
IOW
RESET
8255A KEYPAD CE
D0
D1
D2
D3
D4
D5
D6
D7
RD
WR
A0
A1
RESET
CS
PA0
PA1
PA2
PA3
PA4
PA5
PA6
PA7
PB0
PB1
PB2
PB3
PB4
PB5
PB6
PB7
PC0
PC1
PC2
PC3
PC4
PC5
PC6
PC7
ROW0
ROW1
ROW2
ROW3
ROW4
ROW5
ROW6
ROW7
COL0
COL1
COL2
COL3
COL4
COL5
COL6
COL7
8255A
<TEXT>
D1
LED-RED
<TEXT>

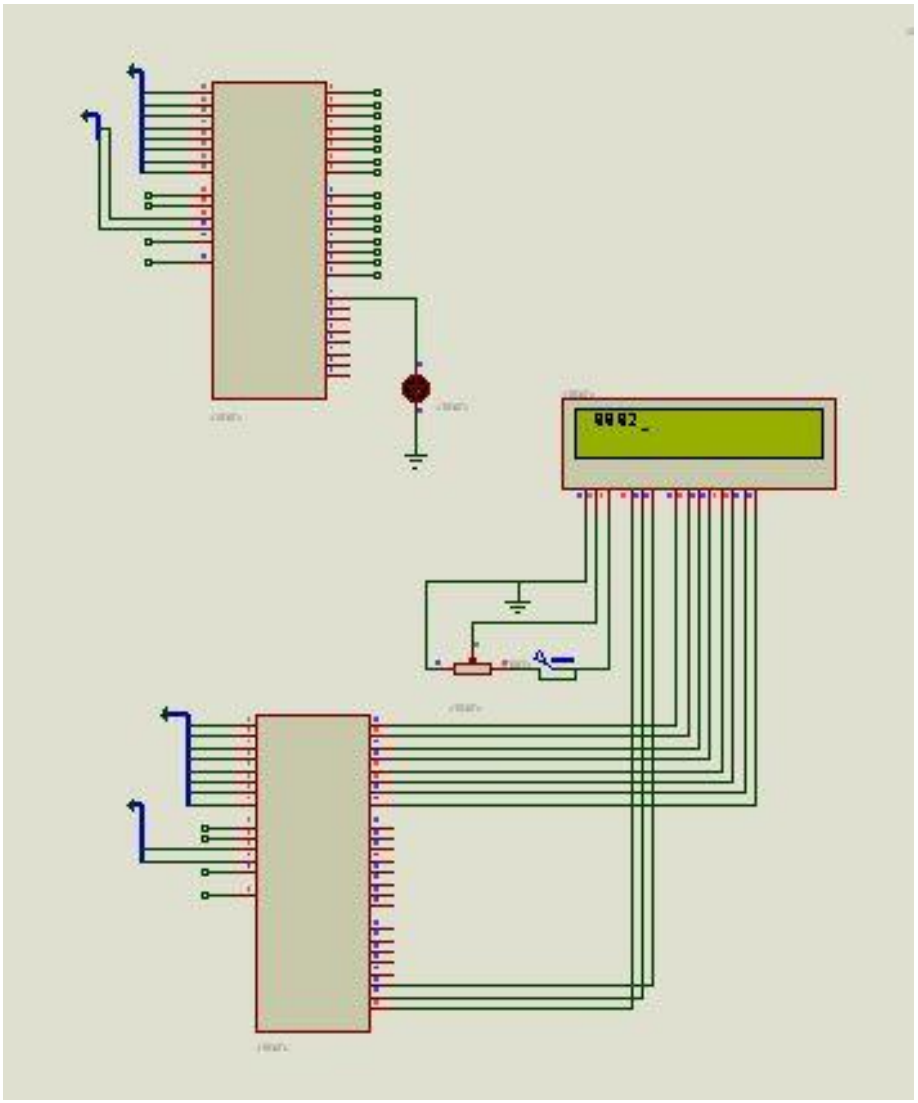

A[0..15]
IO DECODER
A4
A5
A6
IO DECODER(E1)
<TEXT>
M/IO
A
B
C
E1
E2
E3
Y0
Y1
Y2
Y3
Y4
Y5
Y6
Y7
8255A KEYPAD CE
8255A LCD CE
8253A CE
74LS138
<TEXT>



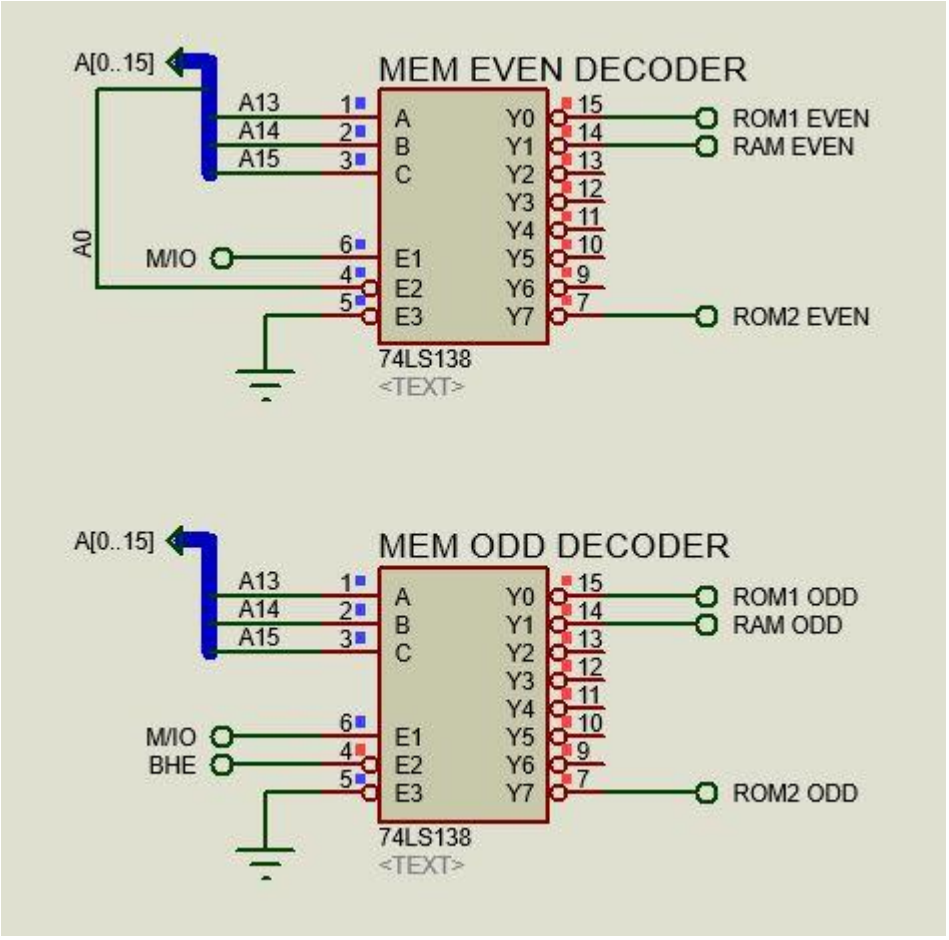
KeyPad

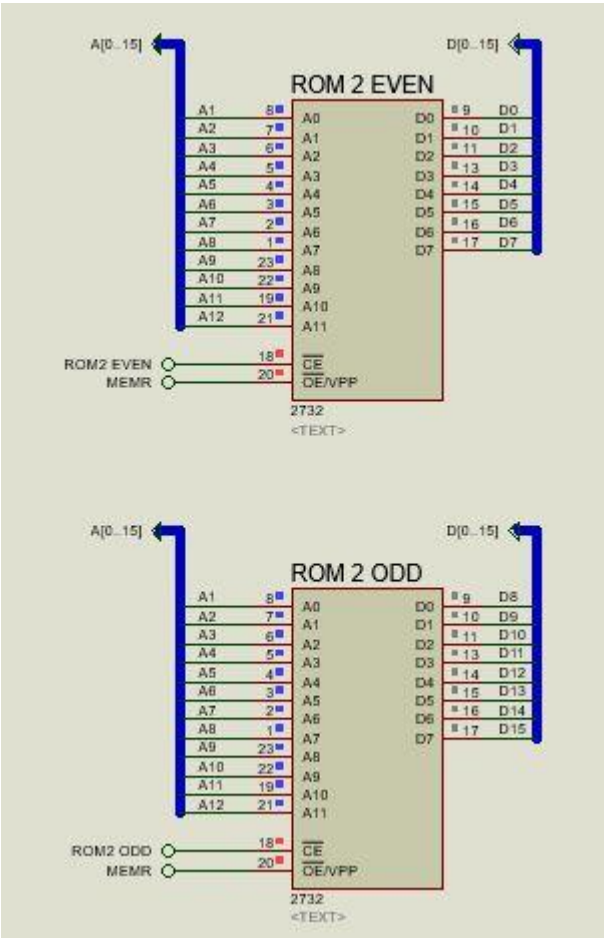


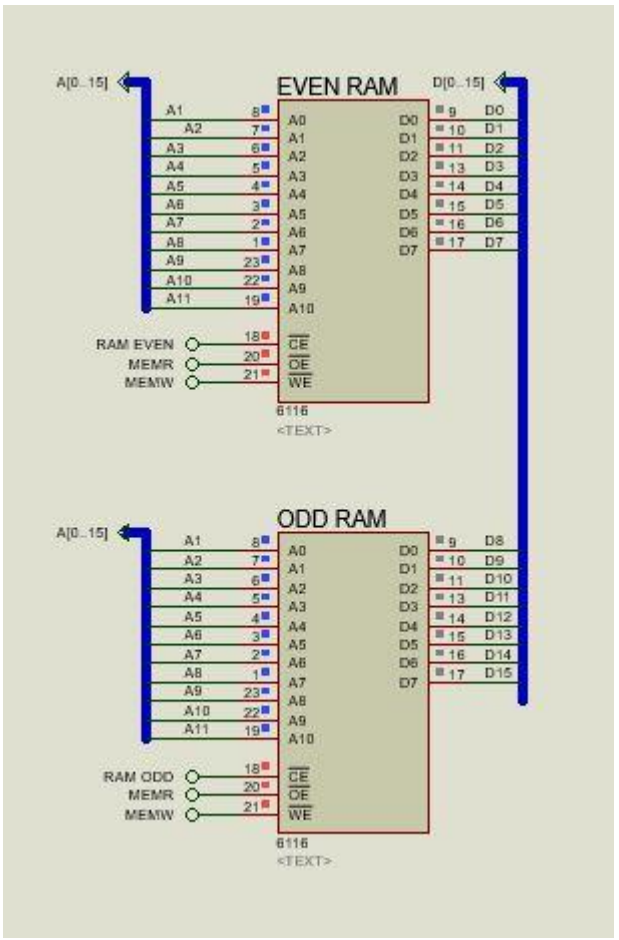
U6(Main 8086)

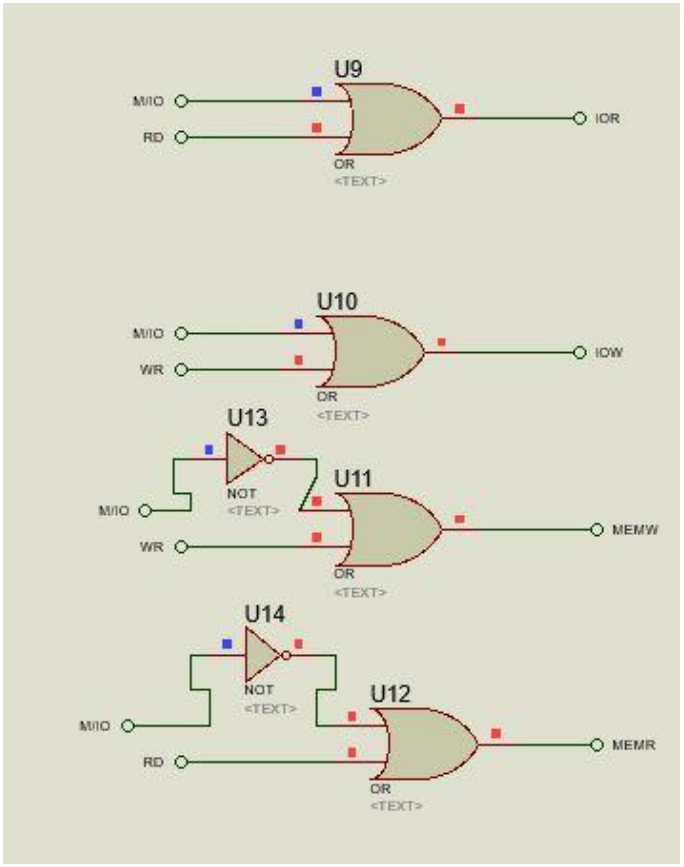


LCD Display

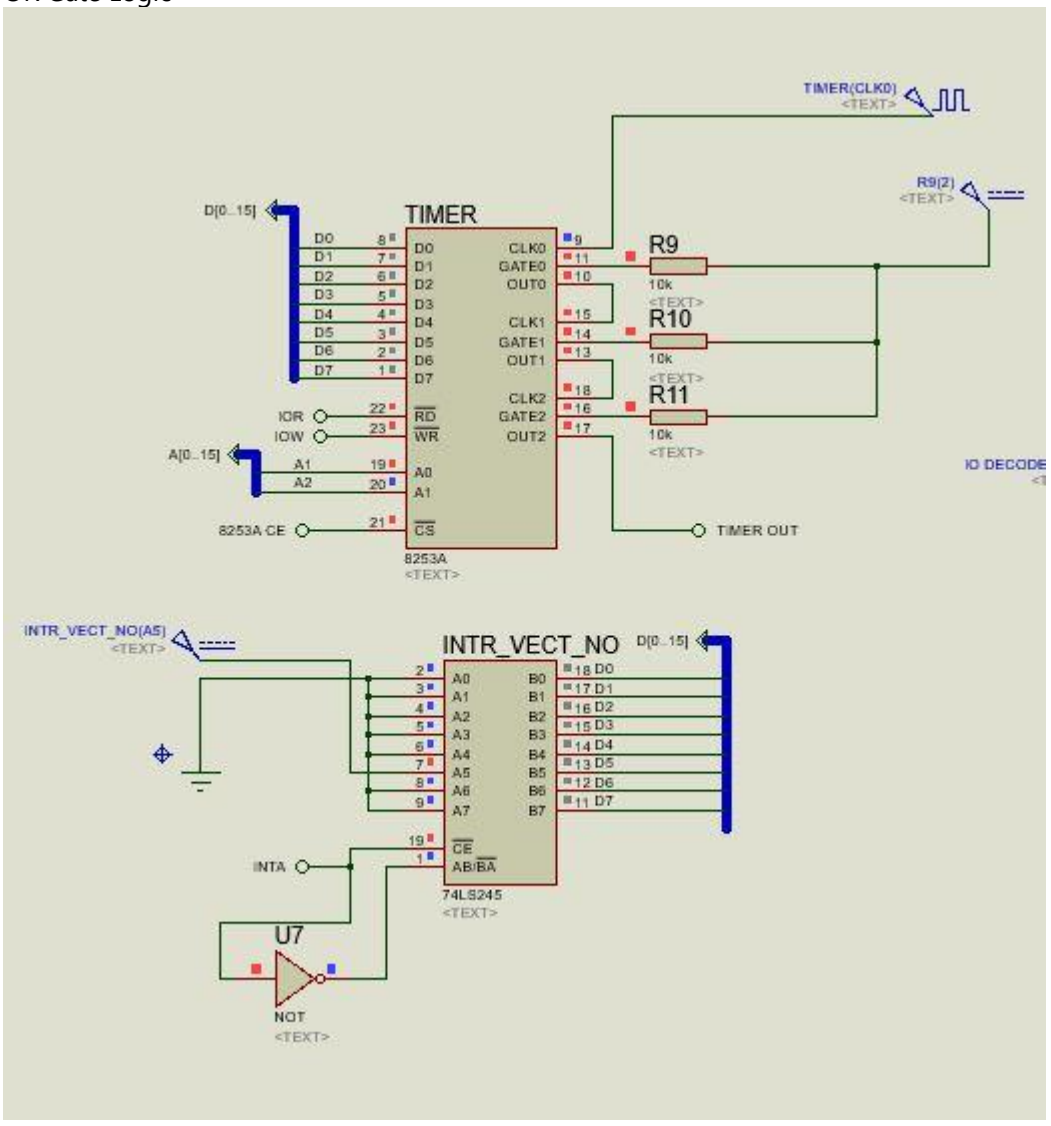

MEM EVEN DECODER
A[0..15]
A13
A14
A15
A0
M/IO
A
B
C
E1
E2
E3
Y0
Y1
Y2
Y3
Y4
Y5
Y6
Y7
ROM1 EVEN
RAM EVEN
ROM2 EVEN
74LS138
<TEXT>
MEM ODD DECODER
A[0..15]
A13
A14
A15
M/IO
BHE
A
B
C
E1
E2
E3
Y0
Y1
Y2
Y3
Y4
Y5
Y6
Y7
ROM1 ODD
RAM ODD
ROM2 ODD
74LS138
<TEXT>

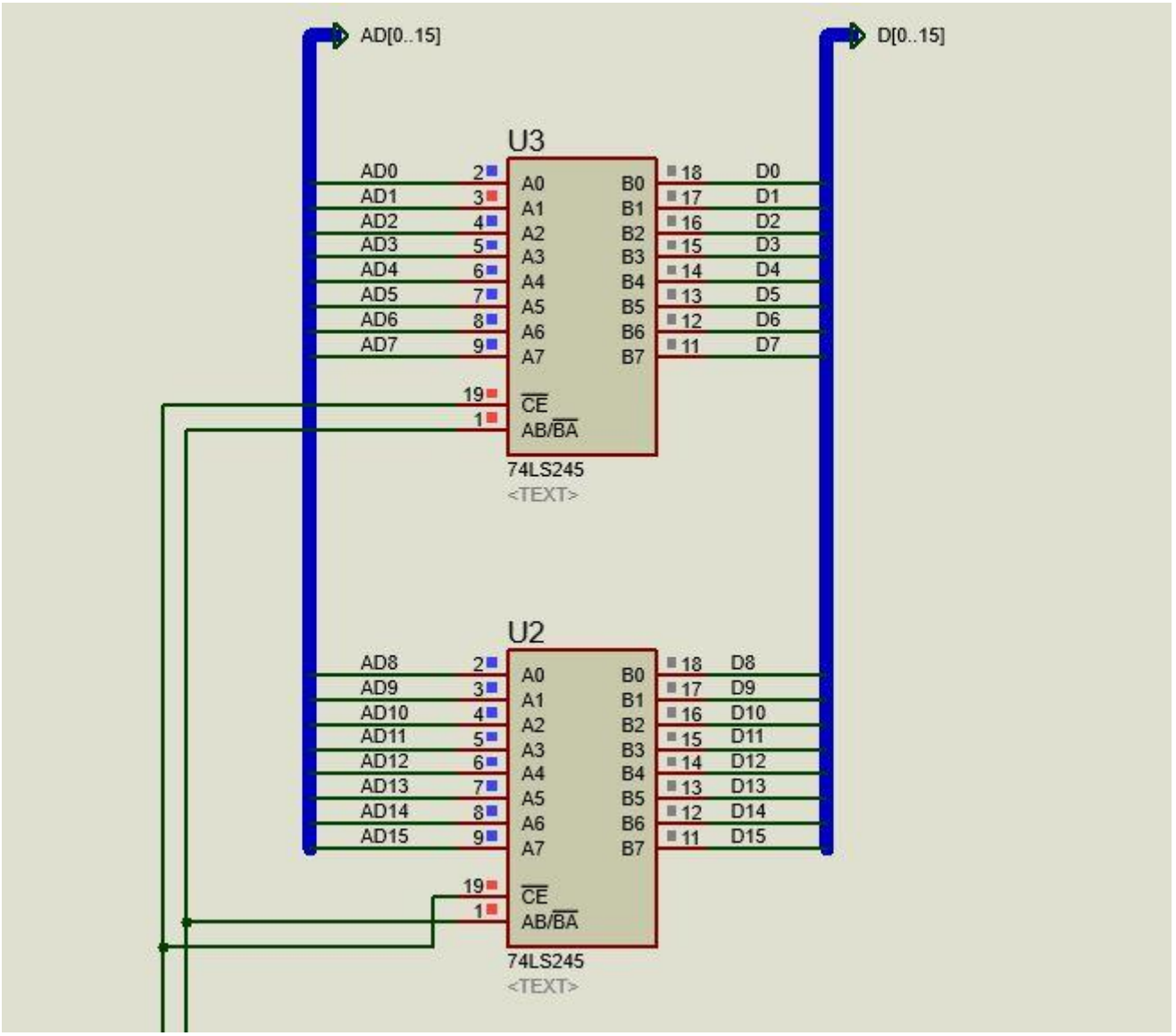

A[0..15]
D[0..15]
ROM 2 EVEN
ROM2 EVEN
MEMR
CE
OE/VPP
2732
<TEXT>
A[0..15]
D[0..15]
ROM 2 ODD
ROM2 ODD
MEMR
CE
OE/VPP
2732
<TEXT>


A[0..15]
D[0..15]
EVEN RAM
RAM EVEN
MEMR
MEMW
CE
OE
WE
6116
<TEXT>
A[0..15]
ODD RAM
RAM ODD
MEMR
MEMW
CE
OE
WE
6116
<TEXT>

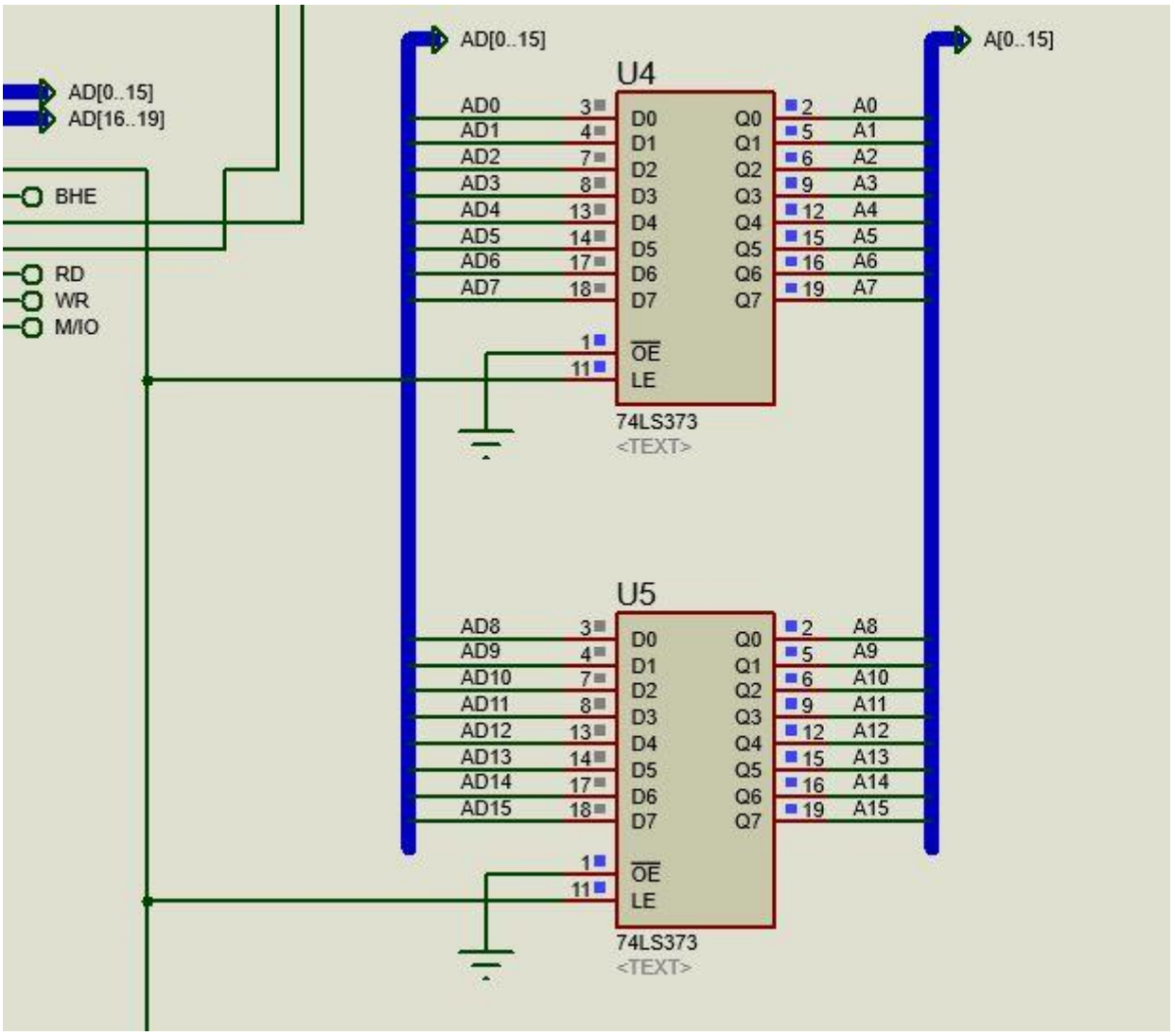


OR Gate Logic





U3 and U2(Main 8086)



U4 and U5(Main 8086)

Thank You