Microprocessors and Interfacing(EEE/INSTR/CS F241)

Project- Question no.22

Batch

Cash Register

Submitted by

Ronit Dutta 2011B3A8492G

Sajidur Rahman 2011B5A7496G

Prithu Bhatnagar 2011B5A8380G

Nikhil 2011B5A8419G

Specificationsof the problem:

1. A stand alond Cash Register with unputs from the keyboard and output on a LCD with a Battery backup of 36 hours
2. The register has a locking system which is noy unlocked before operation results in a alarm
3. After the lock is open, the LCD is turned on and it displays “System Ready”.
4. The user has to then press the Mode button on the keyboard. The LCD then displays “Select Mode”.
5. The user can operate in any of the two modes Transaction/ Program. Transaction is the normal function and in the Program Mode, user is allowed to add new items and their cost.
6. Every item has an item code and a cost associated with it.
7. If the user presses the Trans key the system enters into transaction mode. The LCD displays “Enter Transaction Mode Y/N ?”.
8. User then has to press Y to confirm. If user presses N it goes back to Mode Select display.
9. In the Transaction mode user is expected to enter the item code and the quantity. Item code has to be entered using the Item No. key followed by the item code. The item code can be entered with the help of the numeric keys 0-9. At the end of the item code the user has to press the Enter key. The item code will be then displayed on the LCD.
10. User can press Backspace key to change the value of last key press or he can press Cancel to delete the whole entry.
11. After the item code is displayed, user has to enter the quantity by pressing Quantity key followed by quantity of the item (using the numeric keys) a person wishes to buy and the Enter key.
12. Automatically the total cost of the item will be displayed on the LCD.
13. The user can continue entering all the items and finally press Total to display the total cost.
14. In the Program mode user can add new items or delete an item. If the cost of an item is to be updated it has to first deleted and re-added to the item list in memory.
15. When you add a new item you have enter the item number by using the Item no. key and the cost using the Cost key. After the cost has been keyed in the user must press Enter.
16. The inter-active display will confirm your entry before storing it in the memory.
17. If an item is to be deleted it is done using the Del Item key. Then user is required to press the Item No key followed by the item code and then press Enter.
18. The inter-active display will confirm your entry before deleting it from the memory.

Problem Assumptions:

1. The Locking of the register is done by using a switch
2. Delay for the debounce check for the keyboard is 20ms
3. Maximum of256 items can be stored
4. Each iten code must be a three digit code eg: for 1 user must enter 001

Components used:

1)Microprocessor-8086

2) IC's used:-

(i)74Ls373-2

(ii)74Ls245-2

(iii)74ALS00

(iv)74ALS02

(v)74ALS32

(vi)74ALS138

(vii)74ALS244

(viii)74ALS245

(ix)74ALS373

(x)74ALS447

xi)7447

xii)7805

xiii)8253A

XIV)8255A

XV)8259

XVI)74ALS02

2)Memory:-

(i)2716-

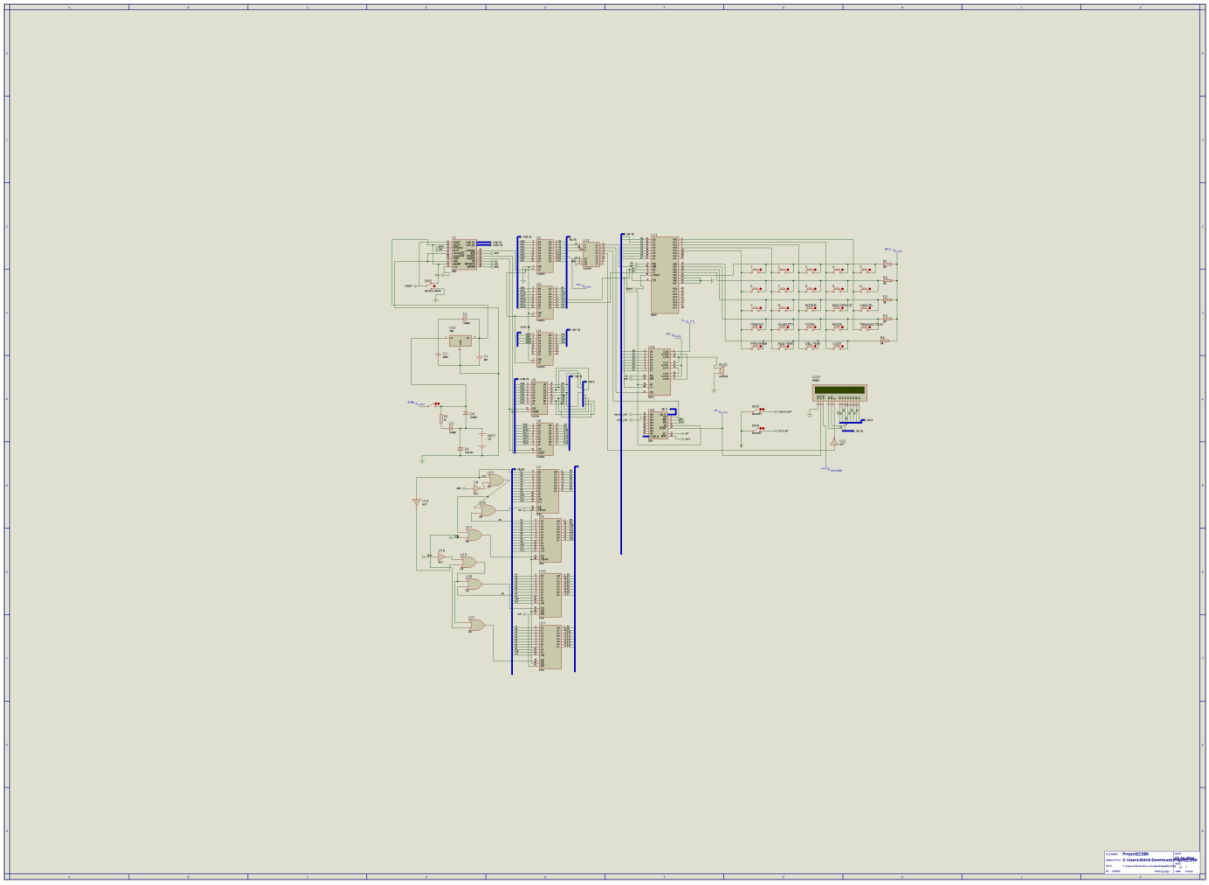
(ii)2732

(iii)6116

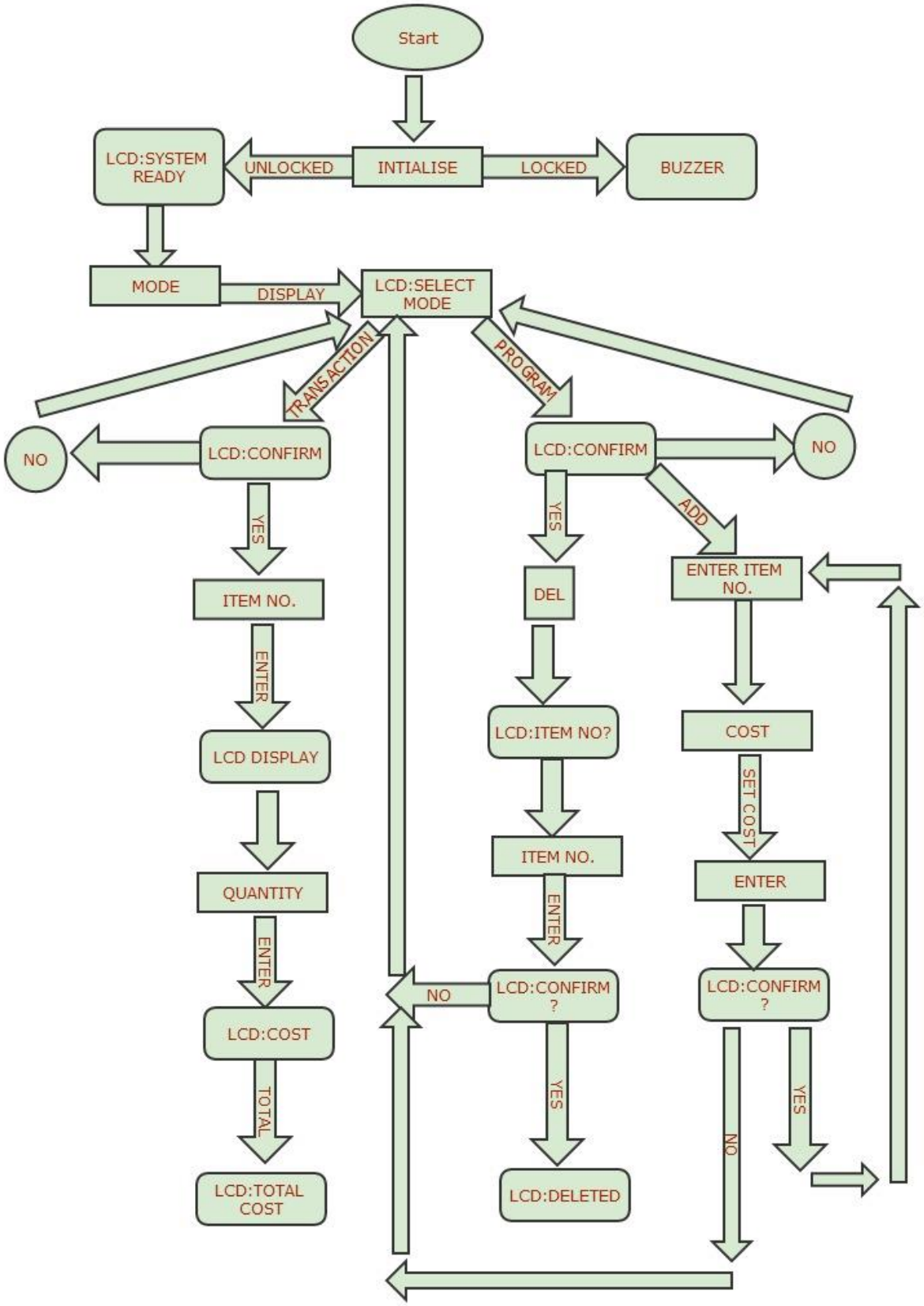
Other components:- Battery(12volts),Buzzer,capacitor,LM020L(LCD),Not gate,Or gate,Resistors

Switches:- SW-SPDT,SW-SPDT-MOM,SW-SPST,SW-SPST-MOM,

CIRCUIT DIAGRAM



FLOWCHART


Start
INTIALISE
UNLOCKED
LOCKED
LCD:SYSTEM READY
BUZZER
MODE
DISPLAY
LCD:SELECT MODE
TRANSACTION
PROGRAM
NO
LCD:CONFIRM
YES
ITEM NO.
ENTER
LCD DISPLAY
QUANTITY
ENTER
LCD:COST
TOTAL
LCD:TOTAL COST
LCD:CONFIRM
NO
YES
DEL
LCD:ITEM NO?
ITEM NO.
ENTER
LCD:CONFIRM ?
NO
YES
LCD:DELETED
ADD
ENTER ITEM NO.
COST
SET COST
ENTER
LCD:CONFIRM ?
NO
YES

ASSEMBLY LANGGUAGE CODE

```
.model tiny

#make_bin#

#LOAD_SEGMENT=0FFFFh#

#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#


.data


porta equ 00h   ;8255(1)

portb equ 02h

portc equ 04h

creg  equ 06h

cnt0  equ 08h ;8254

cnt1  equ 0Ah

cnt2  equ 0Ch
```

```
cregt equ 0Eh
add1  equ 10h ;8259(1)
add2  equ 12h ;8259- two addresses
ireg  equ 18h ;LCD - 3 addresses.
streg  equ 1Ah
dreg  equ 1Ch


MODE        EQU   1D1DH
TRANS       EQU         1E1DH
PROGRAM     EQU         0F1EH
YES         EQU         0F1BH
NO          EQU         171BH
ZERO        EQU         1E17H
ONE         EQU         0F0FH
TWO         EQU         170FH
THREE       EQU         1B0FH
FOUR        EQU         1D0FH
FIVE        EQU         1E0FH
SIX         EQU         0F17H
SEVEN       EQU         1717H
EIGHT       EQU         1B17H
NINE        EQU         1D17H
ENT         EQU         1B1BH
BACKSPACE   EQU         1D1BH
CANCEL      EQU         1E1BH
ITMNO       EQU         0F1DH
```

```
QUANTITY    EQU         171DH
TOTAL       EQU         1B1DH
ADDITM          EQU         171EH
DELITEM         EQU         1B1EH
COST        EQU         1D1EH



TABLE_P     DW 256 DUP(?)

;MESSAGES
MODE2       DB      'ENTER MODE',00h
READY       DB      'SYSTEM READY',00h
CONF        DB      'CONFIRM Y/N ?',00h
ITEMNO      DB      'ITEM NO. ?',00h
NOITEM      DB      'NO ITEM FOUND',00h
QUANTITY2   DB      'ENTER QUANTITY',00h
INVALID     DB      'INVALID KEY',00h
NOSLOT      DB      'NO SLOT',00h
ENT_COST    DB      'ENTER COST',00h
ITEM_SAVED  DB      'ITEM SAVED',00h
NOSLOT2     DB      'NO ITEM IN SLOT',00h
ITEM_DEL    DB      'ITEM DELETED',00h

SUBTOT      DW      '0',00h
TOTAL2 DW       '0',00h
```

```
ITMNO2          DW      '0',00h
QUANT           DW      '0',00h


.code
.startup
;initialise
; add your code here
        jmp     st12


db     509 dup(0)


;IVT entry for 80H


  dw     unlock_isr
 dw     0000
dw                  lock_isr
dw                  0000
 db     508 dup(0)


;main program


st12:      cli
; intialize ds, es,ss to start of RAM
        mov     ax,0200h
        mov     ds,ax
        mov     es,ax
```

```
        mov     ss,ax
        mov     sp,0FFFEH
;intialise portb as input &portc as output
        mov     al,10000011b
                out             creg,al


mov al,00010011b
          out add1,al
          mov al,80h
          out add2,al
          mov al,03h
          out add2,al
          mov al,0FCh
          out add2,al
          sti

;lock

b0:       mov  al,00h
          out  porta,al
b1:       in   al,portb
          and  al,1fh
          cmp  al,1fh
          jnz  b1
          mov  cx,20
```

```
        call delay
         mov  al,00h
        out  porta,al
b2:     in   al,portb
        and  al,1fh
        cmp  al,1fh
        jz   b2
        mov  cx,20
        call delay
      mov  al,00h
        out  porta,al
        in   al,portb
        and  al,1fh
        cmp  al,1fh
        jz   b2

buzzer: mov al,00110110b
        out cregt,al
        mov al,01110110b
        out cregt,al
        mov al,10110110b
        out cregt,al
        mov al,02h
        out cnt0,al
        mov      al,00h
        out cnt0,al
        mov al,0Fh
```

```
        out cnt1,al
        mov al,00h
        out cnt1,al
        mov al,60h
        out cnt2,al
        mov al,0EAh
        out cnt2,al



;lock code ends

X1:
LEA         SI,READY
CALL        LCD
call        KEYBOARD
CMP         AX,MODE
JNZ         X1

X2:
LEA         SI,MODE2
CALL        LCD
CALL        KEYBOARD
CMP         AX,TRANS
JZ          TRANSAC
CMP         AX,PROGRAM
```

```
JZ          PROG



TRANSAC:
LEA         SI,CONF
CALL        LCD
CALL        KEYBOARD
CMP         AX,NO
JZ          X2
CMP         AX,YES
JZ          ITEM
JNZ         TRANSAC

ITEM:
LEA         SI,ITEMNO
CALL        LCD
CALL        KEYBOARD

MOV         DI,00H
X4:
CMP         DI,0100H
JZ          NOITEM2
MOV         SI,AX
MOV         CX,00H
CMP         CX,TABLE_P[SI]
```

```
INC         DI
JNZ         X3
JZ          X4

NOITEM2:
LEA         SI,NOITEM
CALL        LCD
JMP         ITEM


X3:
DEC         DI
LEA         SI,QUANTITY2
CALL        LCD
CALL        KEYBOARD
MOV         QUANT,AX
MOV         CX,TABLE_P[DI]
MUL         CX
ADD         SUBTOT,CX
MOV         CX,0
CALL        KEYBOARD
CMP         AX,TOTAL
JMP         TOTAL_PRICE
CMP         AX,ADDITM
JMP         ITEM
LEA         SI,INVALID
CALL        LCD
```

```
TOTAL_PRICE:
MOV     BX,SUBTOT
MOV     CX,TOTAL2
ADD     CX,BX
MOV     TOTAL2,CX
LEA     SI,TOTAL2
CALL    LCD
CALL    KEYBOARD
CMP     AX,MODE
JMP     X2

PROG:
LEA     SI,CONF
CALL    LCD
CALL    KEYBOARD
CMP     AX,NO
JZ      X2
CMP     AX,YES
JZ      PROG_2
```

```
JNZ             PROG

PROG_2:
CALL            KEYBOARD
CMP             AX,ITMNO
JMP             PROG_ITEM
CMP             AX,DELITEM
JMP             DEL_ITEM
LEA             SI,INVALID
CALL            LCD
JMP             PROG

PROG_ITEM:

LEA             SI,ITEMNO
CALL            LCD
CALL            KEYBOARD

MOV             DI,00H

SLOT:
CMP             DI,0100H
JZ              NO_SLOT
MOV             SI,AX
MOV             CX,0000H
CMP             CX,TABLE_P[SI]
INC             DI
```

```
JNZ         SET_COST
JZ          SLOT

SET_COST:
LEA         SI,ENT_COST
CALL        LCD
CALL        KEYBOARD
MOV         CX,AX
LEA         SI,CONF
CALL        LCD
CALL        KEYBOARD
CMP         AX,YES
JNZ         X2
DEC         DI
MOV         TABLE_P[DI],CX
LEA         SI,ITEM_SAVED
CALL        LCD
JMP         X2

NO_SLOT:
LEA         SI,NOSLOT
```

```
CALL    LCD
JMP             PROG_ITEM

NO_SLOT2:
LEA             SI,NOSLOT2
CALL            LCD
JMP             PROG_ITEM


DEL_ITEM:

LEA             SI,ITEMNO
CALL            LCD
CALL            KEYBOARD

MOV             DI,00H

DEL_SLOT:
CMP             DI,0100H
JZ              NO_SLOT2
MOV             BX,TABLE_P[SI]
MOV             CX,0000H
CMP             CX,BX
INC             DI
JNZ             DL_ITEM
JZ              DEL_SLOT
```

```
DL_ITEM:
LEA             SI,CONF
CALL            LCD
CALL            KEYBOARD
CMP             AX,YES
JNZ             X2
DEC             DI
MOV             TABLE_P[DI],0
LEA             SI,ITEM_DEL
CALL            LCD
JMP             X2



KEYBOARD PROC NEAR


k0:     mov  al,00h
        out  porta,al
k1:     in   al,portb
    and  al,1fh
        cmp  al,1fh
        jnz  k1
        mov  cx,20
        call delay
    mov  al,00h
        out  porta,al
```

```
k2:     in   al,portb
        and  al,1fh
        cmp  al,1fh
        jz   k2
        mov  cx,20
        call delay
        mov  al,00h
        out  porta,al
        in   al,portb
        and  al,1fh
        cmp  al,1fh
        jz   k2

        mov  al,0fh
        mov  bl,al
        out  portb,al
        in   al,porta
        and  al,1fh
        cmp  al,1fh
        jnz  k3
        mov  al,17h
        mov  bl,al
        out  portb,al
        in   al,porta
        and  al,1fh
        cmp  al,1fh
        jnz  k3
```

```
        mov  al,1bh
        mov  bl,al
        out  portb,al
        in   al,porta
        and  al,1fh
        cmp  al,1fh
        jnz  k3
        mov  al,1dh
        mov  bl,al
        out  portb,al
        in   al,porta
        and  al,1fh
        cmp  al,1fh
        jnz  k3
        mov  al,1eh
        mov  bl,al
        out  portb,al
        in   al,porta
        and  al,1fh
        cmp  al,1fh
        jnz  k3
        k3:
        mov  ah,bl
        ret
KEYBOARD ENDP
```

```
LCD proc near
        mov  al,10000011b
        out  creg,al
        call allclr
        call ret_home
        call string
        ret
        LCD endp

allclr proc near

         mov ah,00000001b
out1 :
        push ax
   push dx
        call busy
        mov  al,ah
        mov  dx,ireg
        out  dx,al
        pop  dx
        pop  ax
        ret
        allclr endp
```

```
busy proc near
        push dx
        push ax
        mov  dx,streg

        busy1:
        in   al,streg
        and  al,10000000b
        jnz  busy1
        pop  ax
        pop  dx
        ret
busy endp

charout proc near
        push dx
        push ax
        call busy
        mov  al,ah
        mov  dx,dreg
        out  dx,al
        pop  ax
        pop  dx
        ret
charout endp

string proc near
```

```
chk:
        mov ah,[si]
        cmp ah,00h
        je  string1
        call busy
   call charout
        inc si
        jmp chk
string1: ret
string endp

ret_home proc near
        mov ah,11000000b
out1 :  push ax
      push dx
        call busy
        mov  al,ah
        mov  dx,ireg
        out  dx,al
        pop  dx
        pop  ax
        ret
        ret_home endp

unlock_isr :
        call allclr
        call ret_home
```

```
        jmp  X1
        iret

lock_isr:
        mov al,00001000b
out1 :  push ax
      push dx
        call busy
        mov  al,ah
        mov  dx,ireg
        out  dx,al
        pop  dx
        pop  ax
        jmp  b0
        iret

delay proc near
        mov     ah,86h
        mov     cx,00h
        mov     dx,4e20h
        int     15h
        ret
delay endp

END
```