Design Assignment
on
Microwave Oven

For the partial fulfillment of the course

CS F241 - Microprocessor Programming & Interfacing

BITS Pilani, Pilani Campus

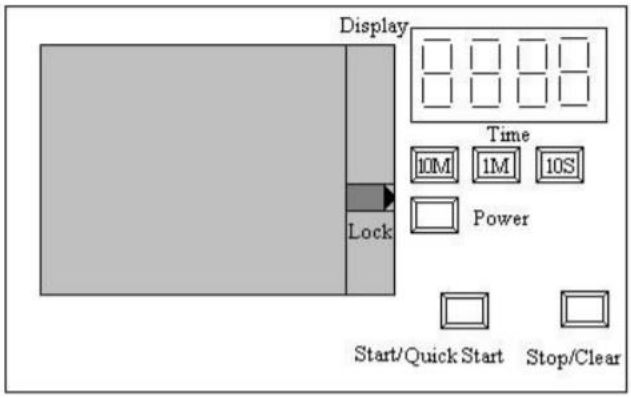
Designed By:

Shivankit Gaind - 2015A7PS076P
Rohit Ghivdonde - 2015A7PS077P
Naveen Venkat - 2015A7PS078P
Anshul Jain - 2015A7PS079P

Problem Statement

System to be Designed : Microwave Oven (P6)

Description: A Simple Microwave Oven without grill.
User Interface: Is shown in the following Figure



- User can cook at 3 different Power levels: 90%, 60%, 30%
- Press of a Power Button decrements the power level by 30 %
- 1 Press - 90%; 2 Presses – 60% ; 3 Presses – 30%; 4 Presses – 90 %;
- 4 Presses – Brings the power level back to 90 %
- The Default power level is 90%
- Power Level is varied by controlling the amount of time for which the microwave is turned on.
- Time of cooking is broken up into 10 sec slots, if power is 60% then for 6 seconds the microwave is on and rest of the 4 seconds the microwave is off.
- Time is set as multiples of 10 Mins, 1 Min, 10 Secs. For e.g. if the cooking time is 12 Minutes and 40 secs- the 10 Minutes button has to be pressed once, 1 Minute Button has to be pressed Twice and 10 seconds button has to be pressed four times.
- Once Time has been set Power cannot be modified.
- When user is setting power level or Time, the value being pressed should be displayed, and when user presses the Start button, the cooking process begins and the time left for cooking to complete is displayed.
- Once the cooking begins the door gets locked and should open only when cooking process is terminated.
- User can terminate cooking anytime by pressing the STOP button. When Stop button is pressed once cooking is aborted, timer is stopped, not cleared; cooking can be resumed by pressing Start.
- When stop is pressed twice, cooking is aborted and timer is also cleared.
- When cooking time elapses, a buzzer is sounded; pressing the Stop Button stops the buzzer.
- A Quick Start mode is available where timer or power need not be set, just Start button needs to be pressed, the default power value is taken and time is set as 30 secs, for every press of the start button time is incremented by 30 seconds.

Assumptions

- There is mechanism already in place whereby door will get locked if PC7 of 8255A is high and unlocked if PC7 is low.
- The heating element of microwave oven is already available which amplifies the current sent to it by 8253.
- The time required for loading the latched values into counters of 8253 after giving the gate trigger has been taken as negligible in comparison to total time.
- Maximum time for cooking user can set is 9999 seconds.
- Code is stored - 0000h
- Time Display format - MM-SS
- Power Display format - PPPP
- The door will automatically get locked once the user presses start/quick start and will open when the process gets completed.
- Multiple keys cannot be pressed simultaneously
- A clock frequency of 2MHz is available to be given to TIMER 2.

List of Important Components Used

Chip Number	Chip	No. of Chips used	Use
8086	Microprocessor	1	CPU
2732	ROM	2	Read Only Memory
74LS245	8 - BIT latch	3	To latch Address Bus
74541	8 - BIT Buffer	1	Unidirectional Buffer
8255	Programmable Peripheral Interface	2	Connected to various I/O devices
8253	Clock timer	2	To produce the stable frequency clock for 8086
8284A	Clock generator	1	Generate CLK for 8086 and 8254. Crystal Frequency - 5 MHz
74HC138	3:8 Decoder	3	For selecting between the various components like ROM, RAM, TIMER1
6116	RAM - 2K	2	Random Access Memory

Other Components Used

1. **Buzzer** - To Indicate the end of cooking of time
2. **NOR** Gates - To allow or disallow Input from Push buttons
3. **74LS245** - Octal Bidirectional buffer(2)
4. **Resistors** ·
5. **7 Segment Display(DL707)** - To Display Time and Power(5)(active low)
6. **AND** Gates(7408)
7. LOGIC **NOT**(7404)
8. LOGIC **OR**(7432)
9. **Nand gate**
10. **Push** Buttons - To input the power, time, start and stop signals from the user.
11. **VCC, Ground, LED's**

Hardware Specifications

Ports of 8255A	Address	Mode	Function of port
Port A	5000h	-	-
Port B	5002h	-	-
Port C	5004h	Mode 0 - Output	Signals sent to buzzer, lock, select lines of 7 segment display
Control Register	5006h	-	Programming of PPI

Ports of 8255B	Address	Mode	Function of Port
Port A	6000h	Mode 0 - Output	Output for 7 segment display
Port B	6002h	Mode 1 - Input	Takes input from the 6 buttons and the timer
Port C	6004h	-	-
Control Register	6006h	-	Programming of PPI

8253 - TIMER 1	Address	Mode	Count Loaded/Function
Counter A	2000h	Mode 3	20,000 Decimal
Counter B	2002h	Mode 3	1000 Decimal
Counter C	2004h	Mode 1	Count Loaded as per Power
Control Register	2006h	-	Used for Programming Timer

8253 - TIMER 2	Address	Mode	Count Loaded/Function
Counter A	3000h	Mode 3	20,000 Decimal
Counter B	3002h	Mode 3	100 Decimal
Counter C	3004h	-	-
Control Register	3006h	-	Used for Programming Timer

MEMORY MAPPING

RAM : 01000H - 01FFFH

RAM (even)	A19	A18	A17	A16	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
From 01000h	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
To 01FFEh	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0

RAM (odd)	A19	A18	A17	A16	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
From 01001h	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1
To 01FFFh	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1

ROM: 00000H - 00FFFH

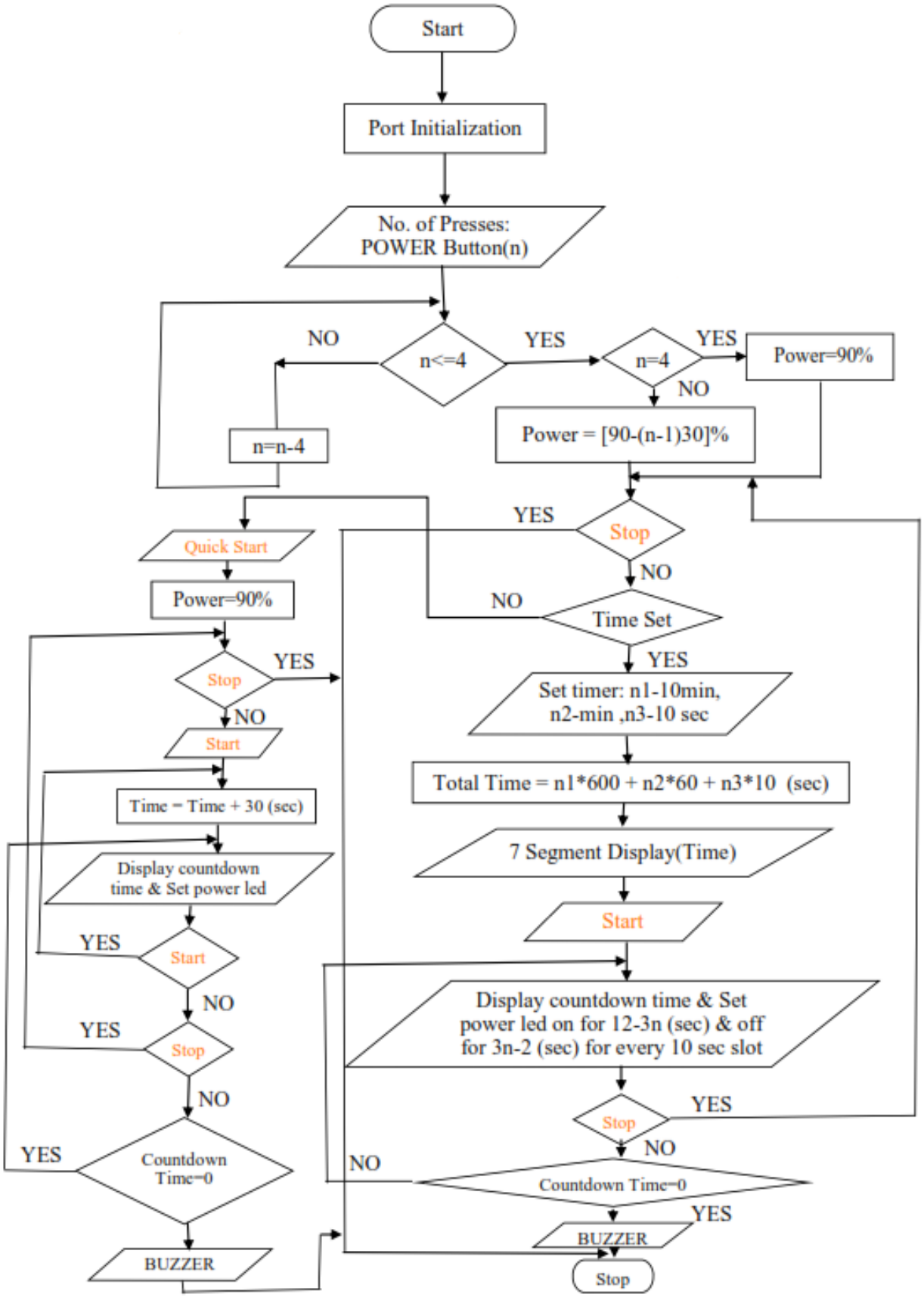
ROM (even)	A19	A18	A17	A16	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
From 00000h	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
To 00FFEh	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0

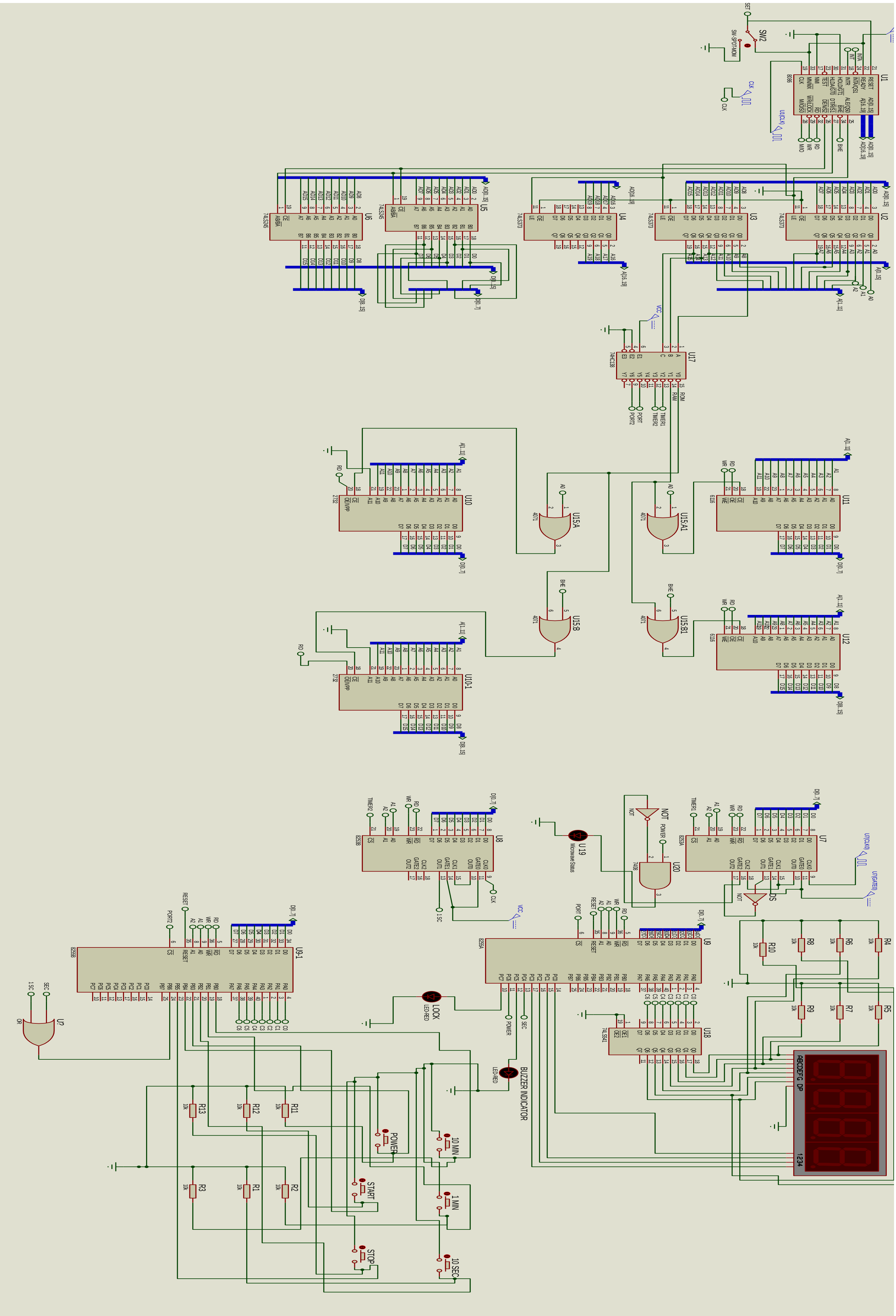
ROM (odd)	A19	A18	A17	A16	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
From 00001h	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
To 00FFFh	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1

Interrupt Vector Numbers Mapping

Vector Number	Associated With
30H	10 MIN Button
31H	1 MIN Button
32H	10 SEC Button
33H	POWER Button
34H	START Button
35H	STOP Button
36H	TIMER

Flowchart


Start
Port Initialization
No. of Presses:
POWER Button(n)
n<=4
NO
YES
n=4
YES
Power=90%
NO
n=n-4
Power = [90-(n-1)30]%
Quick Start
Power=90%
Stop
YES
NO
Time Set
NO
YES
Stop
YES
NO
Start
Set timer: n1-10min,
n2-min ,n3-10 sec
Time = Time + 30 (sec)
Total Time = n1*600 + n2*60 + n3*10 (sec)
Display countdown
time & Set power led
7 Segment Display(Time)
Start
YES
Start
NO
YES
Stop
NO
Display countdown time & Set
power led on for 12-3n (sec) & off
for 3n-2 (sec) for every 10 sec slot
Stop
YES
NO
YES
Countdown
Time=0
NO
Countdown Time=0
YES
BUZZER
BUZZER
Stop



CODE

```
#make_bin#
#LOAD_SEGMENT=0500h#    ; setting loading address, .bin file will be loaded to this address:
#LOAD_OFFSET=0000h#
sec equ 1010h                ;Stores the total time left for the countdown
power equ 1012h              ;Stores the number of power presses
start equ 1013h              ;Stores the number of start button presses
stop equ 1014h               ;Stores the number of stop button presses
disp equ 1015h               ;Next 4 bytes will be stored for 4-digit BCD count
display_table equ 1019h      ;Display table for displaying digits on the 7-segement display
inp equ 1029h                ;Stores the input from the port - to check which button is
time_loop equ 1030h



#0000=0500h#         ; setting entry point

#DS=1000h#          ; set segment registers
#ES=1000h#

#SS=1000h#          ; setting stack
#SP=FFFEh#

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


db 1024 dup(0) ;Initially reserving first 1K of memory for IVT


;Storing IP and CS values for different interrupts in the Interrupt Vector Table

mov ax, 0
mov es, ax
mov al, 30h          ; set general registers
mov bl, 4h           ; multiply 30h by 4, store result in ax:
mul bl
mov bx, ax
mov si, offset [A10MIN]      ; copy offset into interrupt vector:
mov es:[bx], si
add bx, 2
mov ax, 0000         ; copy segment into interrupt vector:
mov es:[bx], ax
```

```
mov ax, 0
mov es, ax
mov al, 31h          ; calculate vector address for interrupt 31h:
mov bl, 4h           ; multiply 31h by 4, store result in ax:
mul bl
mov bx, ax
mov si, offset [A1MIN]   ; copy offset into interrupt vector:
mov es:[bx], si
add bx, 2
mov ax, 0000           ; copy segment into interrupt vector:
mov es:[bx], ax


mov ax, 0
mov es, ax
mov al, 32h        ; calculate vector address for interrupt 32h:
mov bl, 4h         ; multiply 32h by 4, store result in ax:
mul bl
mov bx, ax
mov si, offset [A10SEC]    ; copy offset into interrupt vector:
mov es:[bx], si
add bx, 2
mov ax, 0000         ; copy segment into interrupt vector:
mov es:[bx], ax


mov ax, 0
mov es, ax
mov al, 33h          ; calculate vector address for interrupt 33h:
mov bl, 4h           ; multiply 33h by 4, store result in ax:
mul bl
mov bx, ax
mov si, offset [POW]    ; copy offset into interrupt vector:
mov es:[bx], si
add bx, 2
mov ax, 0000        ; copy segment into interrupt vector:
mov es:[bx], ax


mov ax, 0
mov es, ax
mov al, 34h       ; calculate vector address for interrupt 34h:
mov bl, 4h        ; multiply 34h by 4, store result in ax:
mul bl
mov bx, ax
mov si, offset [STRT]    ; copy offset into interrupt vector:
mov es:[bx], si
add bx, 2
```

```
mov ax, 0000        ; copy segment into interrupt vector:
mov es:[bx], ax


mov ax, 0
mov es, ax
mov al, 35h      ; calculate vector address for interrupt 35h:
mov bl, 4h       ; multiply 35h by 4, store result in ax:
mul bl
mov bx, ax
mov si, offset [STP]     ; copy offset into interrupt vector:
mov es:[bx], si
add bx, 2
mov ax, 0000      ; copy segment into interrupt vector:
mov es:[bx], ax


mov ax, 0
mov es, ax
mov al, 36h       ; calculate vector address for interrupt 36h:
mov bl, 4h        ; multiply 36h by 4, store result in ax:
mul bl
mov bx, ax
mov si, offset [TIMER]    ; copy offset into interrupt vector:
mov es:[bx], si
add bx, 2
mov ax, 0000      ; copy segment into interrupt vector:
mov es:[bx], ax

;IVT SETUP OVER

;Display Table for 7 - segement display

mov si,display_table
mov byte ptr [si],3fh
inc si
mov byte ptr [si],06h
inc si
mov byte ptr [si],5bh
inc si
mov byte ptr [si],4fh
inc si
mov byte ptr [si],66h
inc si
mov byte ptr [si],6dh
inc si
mov byte ptr [si],7dh
inc si
mov byte ptr [si],07h
inc si
```

```
mov byte ptr [si],7fh
inc si
mov byte ptr [si],67h
inc si


mov ax,10000000b     ;All ports are input ports
mov dx,5006h
out dx,al
mov ax,00001010b     ;Buzzer Indicator OFF
out dx,al
mov ax,00001100b     ;Power input can be taken
out dx,al
mov ax,00001001b     ;Countdown can't be started
out dx,al

;PC0 - PC3 set 1 -- which means all the 4 seven segement displays are disabled
mov ax,00000001b
out dx,al
mov ax,00000011b
out dx,al
mov ax,00000101b
out dx,al
mov ax,00000111b
out dx,al


;Initializing 8255(2)
mov ax,10000010b  ;Port B is taken as input
mov dx,6006h
out dx,al

;Initializing the values at the following addresses:

mov si,sec
mov word ptr [si],0 ;Total no. of seconds loaded initially
mov si,power
mov byte ptr [si],-1 ;No. of times Power is pressed
mov si, start
mov byte ptr [si],0 ;No. of times start is pressed
mov si,stop
mov byte ptr [si],00 ;No. of times stop is pressed



mov ax,00110110b      ;TIMER2 COUNTER 0 Control Word
mov dx,3006h
out dx,al

mov ax,20h                        ;Loading count into TIMER2 COUNTER 0
```

```
mov dx,3000h
out dx,al
mov ax,4eh
mov dx,3000h
out dx,al

mov ax,01010110b        ;TIMER2 COUNTER 1 Control Word
mov dx,3006h
out dx,al

mov ax,100              ;Loading count into TIMER2 COUNTER 1
mov dx,3002h
out dx,al


mov ax,00110110b        ;TIMER1 COUNTER 0 Control Word
mov dx,2006h
out dx,al

mov ax,20h              ;Loading count into TIMER1 COUNTER 0
mov dx,2000h
out dx,al
mov ax,4Eh
mov dx,2000h
out dx,al


mov ax,01110110b        ;TIMER1 COUNTER 1 Control Word
mov dx,2006h
out dx,al

mov ax,     0E8h        ;Loading count into TIMER1 COUNTER 1
mov dx,2002h
out dx,al
mov ax,03h
mov dx,2002h
out dx,al



mov ax,10010010b        ;TIMER1 COUNTER 2 Control Word
mov dx,2006h
out dx,al

;Count will be loaded to TIMER1 COUNTER 2 Later



lp: call poll
```

```
mov si,start
mov al,[si]
cmp al,0
je n
call display
n: jmp lp



A10MIN:

mov ax,600                  ;Increment the time by 600 seconds i.e 10 minutes
mov si,sec
add [si],ax

mov di,time_loop
mov ax, 50
mov [di],ax

mov si,start
mov al,[si]
cmp al,0
jne     dontdisplay1
timedisplay1:               ;This is executed only if Start button is not even pressed once
        call display
        dec word ptr [time_loop]
jnz timedisplay1

mov al,00h                  ;Dispaly Clear
mov dx,6000h
out dx,al

dontdisplay1: nop
iret



A1MIN:

mov ax,60                   ;Increment the time by 60 seconds i.e 1 minute
mov si,sec
add [si],ax
mov cx, 60

mov di,time_loop
mov ax, 50
mov [di],ax

mov si,start
```

```
mov al,[si]
cmp al,0
jne	dontdisplay2
timedisplay2:			;This is executed only if Start button is not even pressed once
	call display
	dec word ptr [time_loop]
jnz timedisplay2

mov al,00h			;Dispaly Clear
mov dx,6000h
out dx,al

dontdisplay2: nop
iret


A10SEC:

mov ax,10			;Increment the time by 10 seconds
mov si,sec
add [si],ax
mov cx, 60

mov di,time_loop
mov ax, 50
mov [di],ax

mov si,start
mov al,[si]
cmp al,0
jne	dontdisplay3
timedisplay3:			;This is executed only if Start button is not even pressed once
	call display
	dec word ptr [time_loop]
jnz timedisplay3

mov al,00h			;Dispaly Clear
mov dx,6000h
out dx,al

dontdisplay3: nop
iret



POW:
mov si,power			;Setting up the power
mov al,[si]
add al,1
cmp al,03			;If 4 presses are there, then the power is again set back to 90%
```

```
jne x2
mov al,00
x2: mov [si],al

mov di,time_loop
mov ax, 50
mov [di],ax

mov si,start
mov al,[si]
cmp al,0

jnz dontdisplay4

powerdisplay:                    ;This is executed only if Start button is not even pressed once
        call display_power
        dec word ptr [time_loop]
jnz powerdisplay

mov al,00h                       ;Dispaly Clear
mov dx,6000h
out dx,al

dontdisplay4: nop
iret



STRT: mov si,power
mov al,[si]

mov di,start
mov byte ptr [di],1
xor ah,ah


mov bl, 3
mul bl
mov cl,09h
sub cl,al


mov si,sec
mov ax,[si]

add ax, 30                       ;Incrementing counter by 30 seconds everytime Start is pressed
mov [si],ax

mov si,stop                      ;Number of stops pressed is made equal to zero
mov al,0
```

```
mov byte ptr [si],al


mov ax,00001000b    ;Input fron Timer enabled
mov dx,5006h
out dx,al
mov dx,2004h        ;Count loaded into Timer1 Counter 2
mov ax,cx
out dx,al

mov ax,00001101b    ;Power input can't be taken once start is pressed
mov dx,5006h
out dx,al
mov ax,00001111b    ;Set up Lock
out dx,al
iret



STP:  mov ax,00001001b      ;Input from Timer blocked
mov dx,5006h
out dx,al
mov ax,00001100b            ;Power input can be taken
out dx,al
mov ax,00001010b            ;Buzzer LED Offss
out dx,al
mov si,stop
inc byte ptr [si]
mov al,[si]
cmp al,2                    ;If Stop is Pressed twice, then only execute all these steps
jl x1
mov di,sec
mov word ptr [di],0
mov byte ptr [si],0
mov byte ptr [start],0

mov al,00h                  ;Dispaly Clear
mov dx,6000h
out dx,al

x1: mov dx,5006h
mov ax,00001110b            ;Opening the lock
out dx,al
iret



TIMER:mov si,sec
dec word ptr [si]       ;Decrement the count
mov ax,[si]             ;Check the count
```

```
cmp ax,0
jg dontexecute          ;If count is still greater than zero, then no need to execute the given instructions

mov ax,00001011b    ;If countdown is over, buzzer is on
mov dx,5006h
out dx,al
mov ax,00001001b    ;No further countdown
out dx,al
mov ax,00001100b    ;Power input enabled
out dx,al
mov ax,00001110b    ;Lock opened
out dx,al
dontexecute: nop
iret



proc display near
mov si,disp
mov di,sec
mov ax, [di]            ;Current number of seconds is stored in AX now

mov bx,10

xor dx,dx
div bx
mov [si],dl             ;Remainder is in DL as remainder is not greater than 9, so no need to consider
                         DH - Hence we extract the last digit

xor dx,dx               ;Digit at Tens place
div bx
mov [si+1],dl

xor dx,dx               ;Digit at hundredth place
div bx
mov [si+2],dl

xor dx,dx               ;Digit at Thousandth place
div bx
mov [si+3],dl


mov di,display_table

;Display Last Digit
mov al,[si]
mov bl,al
xor bh,bh
mov ax,[di+bx]                  ;So the displacement is equal to the digit we have
```

```
mov dx,6000h
out dx,al

mov ax, 00000001b          ;Previous display disabled
mov dx,5006h
out dx,al
mov ax, 00000110b          ;Last digit shown
out dx,al
call delay



mov al,[si+1]
mov bl,al
mov ax,[di+bx]
mov dx,6000h
out dx,al


mov ax, 00000111b          ;Previous display disabled
mov dx,5006h
out dx,al
mov ax, 00000100b          ;Second Last digit shown
out dx,al
call delay



mov al,[si+2]
mov bl,al
mov ax,[di+bx]
mov dx,6000h
out dx,al

mov ax, 00000101b          ;Previous display disabled
mov dx,5006h
out dx,al
mov ax, 00000010b          ;Digit at Hundredth place shown
out dx,al
call delay



mov al,[si+3]
mov bl,al
```

```
mov ax,[di+bx]
mov dx,6000h
out dx,al
mov ax, 00000011b    ;Previous display disabled
mov dx,5006h
out dx,al
mov ax, 00000000b    ;Digit at thousandth place shown
out dx,al
call delay


mov ax, 00000001b    ;Finally clearing the previous display also
out dx,al
ret
endp

poll proc near
mov dx,6002h
in al,dx


;Checking which of the inputs is 1 which means the corresponding switch is pressed

;The mapping to switches is:
; PB0 - 10 MIN BUTTON
; PB1 - 1 MIN BUTTON
; PB2 - 10 SEC BUTTON
; PB3 - POWER BUTTON
; PB4 - START BUTTON
; PB5 - STOP BUTTON
; PB6 - TIMER INTERRUPT HANDLING


;Checking which input is receiving interrupt

mov [inp],al
cmp al,0ffh
je pl7


mov al,[inp]
and al,01h
jnz pl1
int 30h
call delayss

pl1: mov al,[inp]
and al,02h
jnz pl2
int 31h
```

```
call delayss

pl2:mov al,[inp]
and al,04h
jnz pl3
int 32h
call delayss

pl3:mov al,[inp]
and al,08h
jnz pl4
int 33h
call delayss

pl4:mov al,[inp]
and al,10h
jnz pl5
int 34h
call delayss

pl5:mov al,[inp]
and al,20h
jnz pl6
int 35h
call delayss

pl6:mov al,[inp]
and al,40h
jnz pl7
int 36h
call delay1s

pl7: ret
endp

;POLLING ENDS


proc display_power near
mov si,disp
mov di,power
mov al,[di]

        ;Loading Power value according to the number of power presses
        cmp al,00
        jnz j1
        mov ax,90
        jmp j0

j1: cmp al,01
```

```
        jnz j2
        mov ax,60
        jmp j0


j2: cmp al,02
        jnz j0
        mov ax,30

j0: nop



mov bx,10

xor dx,dx
div bx
mov [si],dl ;Remainder is in DL as remainder is not greater than 9, so no need to consider DH - Hence we extract the last digit

xor dx,dx          ;Digit at Tens place
div bx
mov [si+1],dl

xor dx,dx          ;Digit at hundredth place
div bx
mov [si+2],dl

xor dx,dx         ;Digit at Thousandth place
div bx
mov [si+3],dl


mov di,display_table

;Display Last Digit
mov al,[si]
mov bl,al
xor bh,bh
mov ax,[di+bx]             ;So the displacement is equal to the digit we have
mov dx,6000h
out dx,al

mov ax, 00000001b    ;Previous display disabled
mov dx,5006h
out dx,al
mov ax, 00000110b     ;Last digit shown
out dx,al
call delay
```

```
mov al,[si+1]
mov bl,al
mov ax,[di+bx]
mov dx,6000h
out dx,al


mov ax, 00000111b    ;Previous display disabled
mov dx,5006h
out dx,al
mov ax, 00000100b    ;Second Last digit shown
out dx,al
call delay



mov al,[si+2]
mov bl,al
mov ax,[di+bx]
mov dx,6000h
out dx,al

mov ax, 00000101b     ;Previous display disabled
mov dx,5006h
out dx,al
mov ax, 00000010b    ;Digit at Hundredth place shown
out dx,al
call delay



mov al,[si+3]
mov bl,al
mov ax,[di+bx]
mov dx,6000h
out dx,al
mov ax, 00000011b    ;Previous display disabled
mov dx,5006h
out dx,al
mov ax, 00000000b    ;Digit at thousandth place shown
out dx,al
call delay
```

```
ret
endp


;DELAY PROCEDURES

delay proc near
mov cx, 2000
l5:    dec cx
loop l5
ret
endp

delayss proc near
mov cx, 60000
l6:    dec cx
loop l6
ret
endp


delay1s proc near
mov ax, 4
delay2:
dec ax
call delayss
cmp ax,0
jnz delay2
ret
endp
```