ORG 0000H ; the starting address is from 0000H

CLR P1.2 ; clear P1.2 to switch off the buzzer

MOV DPTR, #SEG7 ; load DPTR with the address of the 7-segment display lookup table

ANL P2, #0F0H ; switch off all 7-segment displays

SETB P1.0 ; set P1.0 as input pin (button) for the service number

SETB P1.1 ; set P1.1 as input pin (button) for the queue number

MOV SCON, #52H ; Serial mode 1, enable receiver, set transmist interrupt flag

MOV TMOD, #20H ; Timer 1 mode 2

MOV TH1, #0FDH ; 9600bps with SMOD=0 for 11.0592MHz crystal

ANL PCON, #7FH ; SMOD=0

SETB TR1

MOV R4, #00H ; Move 0 to R4

MOV R5, #00H ; Move 0 to R5

MOV R6, #00H ; Move 0 to R6

MOV R7, #00H ; Move 0 to R7

MOV 20H, #30H ; 30H is the ASCII code for zero

MOV 21H, #30H ; 30H is the ASCII code for zero

MOV 22H, #30H ; 30H is the ASCII code for zero

MOV 23H, #30H ; 30H is the ASCII code for zero

MOV 24H, #30H ; 30H is the ASCII code for zero

SWITCH:

JNB P1.0, SERINCREMENT ; when the service button at P1.0 is pressed and released call SERINCREMENT

JNB P1.1, QINCREMENT ; when the queue button at P1.1 is pressed and released call QINCREMENT

JMP START ; Jump to the START subroutine

START:

MOV A, R4 ; move the value of R4 into A

MOVC A, @A+DPTR ; load the data into the accumulator

ANL P2, #0F0H ; switch off all 7-segment displays

MOV P0, A ; send data to display the value

SETB P2.3 ; turn on Digit 3

CALL DELAY ; wait a short while

MOV A, R5 ; move the value of R4 into A

MOVC A, @A+DPTR ; load the data into the accumulator

ANL P2, #0F0H ; switch off all 7-segment displays

MOV P0, A ; send data to display the value

SETB P2.2 ; turn on Digit 2

CALL DELAY ; wait a short while

MOV A, R6 ; move the value of R4 into A

MOVC A, @A+DPTR ; load the data into the accumulator

ANL P2, #0F0H ; switch off all 7-segment displays

MOV P0, A ; send data to display the value

SETB P2.1 ; turn on Digit 1

CALL DELAY ; wait a short while

START1:

MOV A, R7 ; move the value of R4 into A

MOVC A, @A+DPTR ; load the data into the accumulator

ANL P2, #0F0H ; switch off all 7-segment displays

MOV P0, A ; send data to display the value

SETB P2.0 ; turn on Digit 0

CALL DELAY ; wait a short while

AJMP SWITCH ; jump back to SWITCH and repeat

SERINCREMENT:

MOV A, R4 ; move the value in R4 into A

MOV B, R6 ; move the value in R6 into B

CJNE A,B, Service1 ; Compare the values in A and B and jump to Service1 if they are not equal ( it is to make sure the service number is never greater than the queue number)

MOV A, R5 ; move the value in R5 into A

MOV B,R7 ; move the value in R7 into B

CJNE A,B, Service1 ; compare the values in A and B and jump to service1 if they are not equal ( it is to make sure the service number is never greater than the queue number)

AJMP START ; jump back to START

Service1:

INC R4 ; increment the value in R4 from the lookup table

SETB P1.2 ; switch on the buzzer

CALL DELAYS ;wait a short while

CLR P1.2 ; switch off the buzzer

INC 21H ; increment 21H

ACALL LOOP ; call the LOOP subroutine to store the value of 21H

CALL DELAY ; wait a short while

CJNE R4, #0AH, START ; compare if R4 is equal to '0AH' which is 10 and jump to START, once R4 reaches 0AH move R4 to 00H

MOV R4, #00H ; move R4 to 00H

MOV 21H, #30H ; set 21H back to zero

INC R5 ; increment the value in R5 from the lookup table

SETB P1.2 ; switch on the buzzer

CALL DELAYS ;wait a short while

CLR P1.2 ;switch off the buzzer

INC 22H ; increment 22H

ACALL LOOP ; call the LOOP subroutine to store the value of 22H

CALL DELAY ; wait a short while

CJNE R5,#0AH, START ; compare if R5 is equal to '0AH' which is 10 and jump to START, once R5 reaches 0AH move R5 to 00H

MOV R5, #00H ; move R5 to 00H

MOV 22H, #30H ; move 22H back to zero

ACALL LOOP ; call the LOOP subroutine

RET ; return

QINCREMENT:

INC R6 ; increment R6

INC 23H ; increment 23H

ACALL LOOP ; call the LOOP subroutine to store the value of 23H

CALL DELAY ; wait a short while

CJNE R6, #0AH, START ; compare if R6 is equal to '0AH' which is 10 and jump to START, once R6 reaches 0AH move R6 to 00H

MOV R6, #00H ; move 00H to R6

MOV 23H, #30H ; move 23H back to zero

INC R7 ; increment R7

INC 24H ; increment 24H

ACALL LOOP ; call the LOOP subroutine to store the value of 24H

CALL DELAY ; wait a short while

CJNE R7, #0AH, START1 ; compare if R7 is equal to '0AH' which is 10 and jump to START1, once R7 reaches 0AH move R7 to 00H

MOV R7, #00H ; move 00H to R7

MOV 24H, #30H ;move 24H to zero

ACALL LOOP ; call the LOOP subroutine

RET ; return

; subroutine for sending the data to the internet

LOOP:

MOV 20H, 22H

ACALL SEND

MOV 20H, 21H

ACALL SEND

MOV 20H, #'a'

ACALL SEND

MOV 20H, 24H

ACALL SEND

MOV 20H, 23H

ACALL SEND

MOV 20H, #'b'

ACALL SEND

MOV 20H, #0AH ; 0AH is the ASCII code for newline character

ACALL SEND

ACALL DELAYS

RET

SEND:

JNB TI, SEND

CLR TI

MOV SBUF, 20H

RET

; Subroutine to delay for a few seconds

DELAYS:

MOV R3, #20

RPTA: MOV R2, #100

AGN: MOV R1, #100

DJNZ R1, $

DJNZ R2, AGN

DJNZ R3, RPTA

RET

DELAY:

MOV R0, #250

RPT:

NOP

NOP

DJNZ R0, RPT

RET

SEG7: DB 3FH, 06H, 5BH, 4FH, 66H, 6DH, 7DH, 07H, 7FH, 6FH ;7 segment look-up table

END