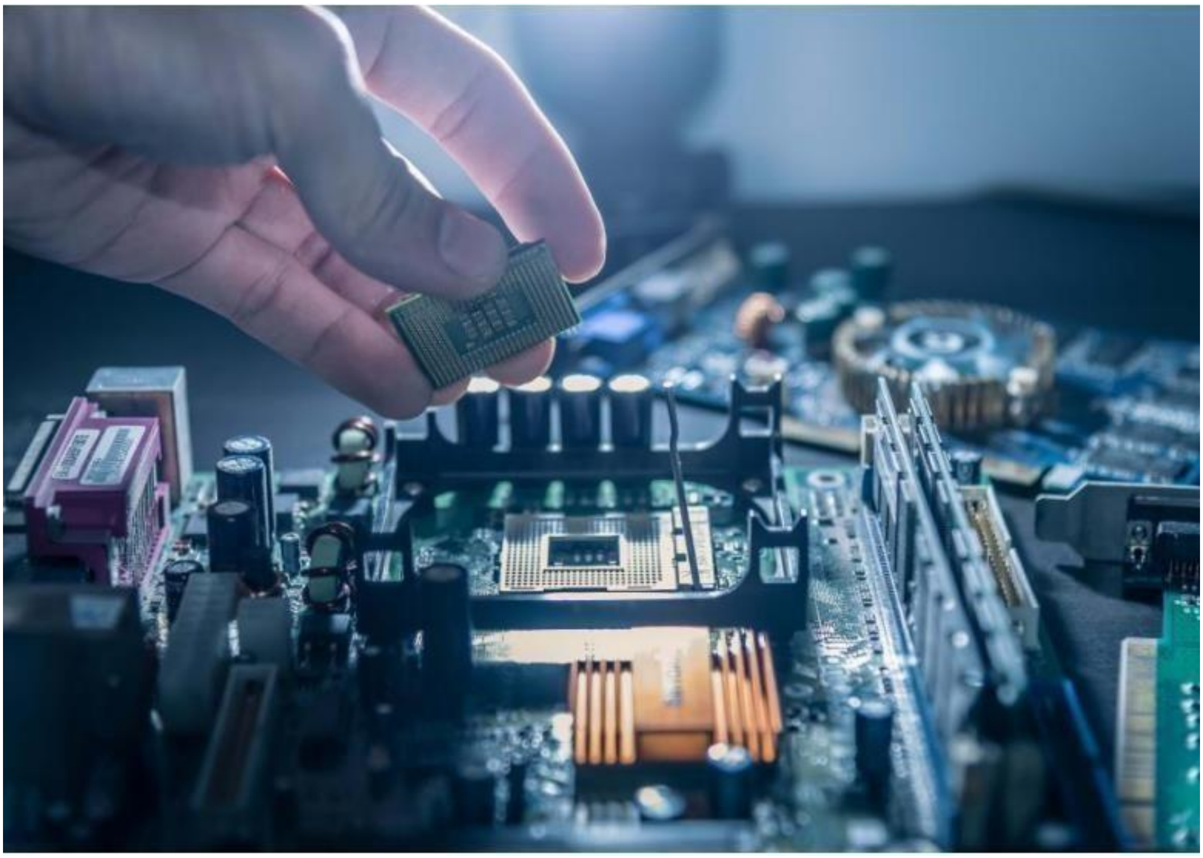
EMBEDDED SYSTEMS LAB

EXPERIMENT 3: Real Time Clock with Stop Watch

CHANAKYA DUPPATLA: 21EC30017

RAHUL SAMINENI: 21EC30045



EMBEDDED SYSTEMS LABORATORY(EC39302)

DATE: 13/03/24

PART– A

Objective: Connect one 7-segment display [common-cathode] to 8255 port (J3/J7) through 74245 and resistor.

(Use individual current limiting resistor for every segment.)

Display 0 to 9 continually with 7 -segment display with 1 second digit display time.

PSEUDO CODE:

```
Initialize program:
  Set the origin address to 8100H

Main:
  Load the address of the port to control the lights into DPTR
  Load the value 10000000B into accumulator A
  Write the value in accumulator A to the port
  Call the LIGHTS subroutine

LIGHTS subroutine:
  Load the address of the port to control the lights into DPTR
  Write the values to control the lights with proper delays:
    00111111B
    00000110B
    01011011B
    01001111B
    01100110B
    01101101B
    01111101B
    00000111B
    01111111B
    01100111B
  Jump back to LIGHTS to repeat the pattern

DELAY subroutine:
  Set up delay using nested loops:
    Loop R1 from 255 down to 0:
      Loop R3 from 120 down to 0:
        Loop R2 from 7 down to 0
  Return from subroutine
```

CODE:

```
 ORG 8100H
MOV DPTR,#0E803H
MOV A,#10000000B
MOVX @DPTR,A
LIGHTS:
MOV DPTR,#0E800H
MOV A,#00111111B
MOVX @DPTR,A
ACALL DELAY
MOV A,#00000110B
MOVX @DPTR,A
ACALL DELAY
MOV A,#01011011B
MOVX @DPTR,A
ACALL DELAY
MOV A,#01001111B
MOVX @DPTR,A
ACALL DELAY


MOV A,#01100110B
MOVX @DPTR,A
ACALL DELAY
MOV A,#01101101B
MOVX @DPTR,A
ACALL DELAY
MOV A,#01111101B
MOVX @DPTR,A
ACALL DELAY

MOV A,#00000111B
MOVX @DPTR,A
ACALL DELAY
MOV A,#01111111B
MOVX @DPTR,A
ACALL DELAY
MOV A,#01100111B
MOVX @DPTR,A
ACALL DELAY

SJMP LIGHTS
DELAY:
MOV R1,#255
DELAY1:
MOV R3,#120
DELAY2:
MOV R2,#7
DELAY3:
DJNZ R2,DELAY3
DJNZ R3,DELAY2
DJNZ R1,DELAY1
```

```
RET
END
```

CODE EXPLANATION:

1. ORG 8100H:This directive sets the origin address of the program to 8100H.

2. MOV DPTR,#0E803H: This instruction loads the address of the data memory location where the control signal for the lights is stored into the data pointer register (DPTR).

3. MOV A,#10000000B: This instruction loads the accumulator register A with the binary value 10000000B, which likely corresponds to a specific control signal for the lights.

4. MOVX @DPTR,A: This instruction writes the content of the accumulator A to the memory location pointed to by DPTR, which is the location specified in the previous instruction.

5. LIGHTS: This label marks the beginning of a loop that controls the lights.

6. MOV DPTR,#0E800H: This instruction loads the address of the data memory location where the status/control of the lights is stored into DPTR.

7. MOV A,#00111111B:This instruction loads the accumulator A with the binary value 00111111B, which likely represents a control signal to turn on specific leds on seven segment display.

8. MOVX @DPTR,A: This instruction writes the content of the accumulator A to the memory location pointed to by DPTR, turning on the corresponding lights.

9. ACALL DELAY: This instruction calls a subroutine named DELAY to introduce a delay.

10. MOV A,#00000110B: This instruction loads the accumulator A with the binary value 00000110B, representing another control signal to turn on different lights.

11. MOVX @DPTR,A: This instruction writes the content of the accumulator A to the memory location pointed to by DPTR, controlling the lights accordingly.

12. ACALL DELAY:This instruction again calls the DELAY subroutine to introduce a delay.

13. [Repeat similar operations for other control signals and calls to DELAY subroutine]

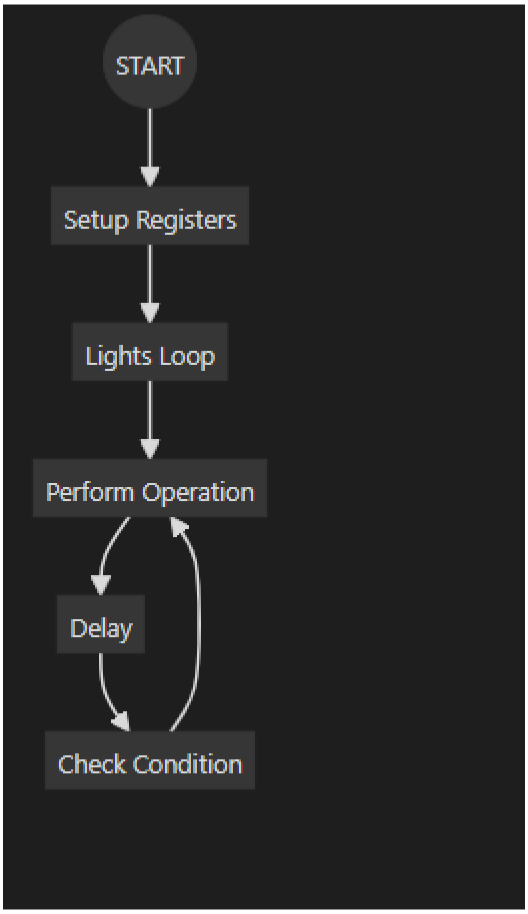
14. SJMP LIGHTS: This instruction jumps back to the LIGHTS label, restarting the loop to continue controlling the lights.

15. DELAY:This subroutine introduces a delay in the program execution.

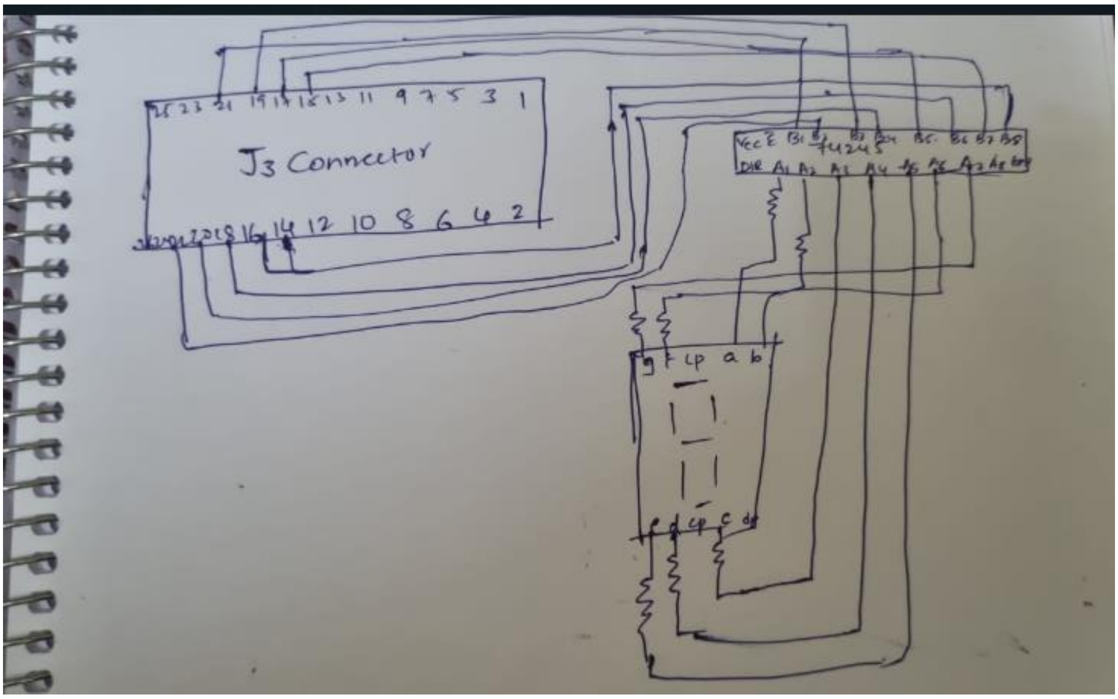
16. RET: This instruction returns from the subroutine.

17. END: This directive marks the end of the program.

FLOWCHART:



CIRCUIT DIAGRAM:



PART-B:

OBJECTIVE: Count 00 to 06, 15 to 23, 29, 47 to 54 continually with two 7-segment displays with 0.5 second digit display time.

(Segments of two displays are to be connected in parallel and driven by one 74245 and one port of 8255) and common points (common cathode pin) of each display are to be driven by another port of 8255 through another 74245)

PSEUDO CODE:

```
Initialize program:
   Set the origin address to 8100H

Main:
   Load the address of the port to control the display into DPTR
   Load the value 10000000B into accumulator A
   Write the value in accumulator A to the port
   Call the DISPLAY subroutine

DISPLAY subroutine:
   Loop through each digit:
      Call the corresponding LOOP subroutine for each digit to display it
   Jump back to DISPLAY to repeat the pattern

LOOP subroutines:
   Loop through each segment of the digit:
      Set up delay
      Display the segment
   Repeat for each segment
   Repeat for each digit
   Return from subroutine

DELAY4 subroutine:
   Set up delay
   Return from subroutine

DELAY subroutine:
   Set up delay
   Return from subroutine

DISPLAY_X subroutines:
   Set up the display for each digit
   Return from subroutine

C_X subroutines:
   Set up the control signals for the display
   Return from subroutine

End program
```

CODE:

```
ORG 8100H

MOV DPTR,#0E803H
MOV A,#10000000B
MOVX @DPTR,A
DISPLAY:

ACALL LOOP_00
ACALL LOOP_01
ACALL LOOP_02
ACALL LOOP_03
ACALL LOOP_04
ACALL LOOP_05
ACALL LOOP_06
ACALL LOOP_15
ACALL LOOP_16
ACALL LOOP_17
ACALL LOOP_18
ACALL LOOP_19
ACALL LOOP_20
ACALL LOOP_21
ACALL LOOP_22
ACALL LOOP_23
ACALL LOOP_29
ACALL LOOP_47
ACALL LOOP_48
ACALL LOOP_49
ACALL LOOP_50
ACALL LOOP_51
ACALL LOOP_52
ACALL LOOP_53
ACALL LOOP_54

SJMP DISPLAY

LOOP_00:MOV R4,#25
LOOP_001:MOV R5,#50
LOOP_002:MOV R6,#10
DISPLAY_00:
ACALL CP_10
ACALL DISP_0
ACALL DELAY4
ACALL CP_01
ACALL DISP_0
ACALL DELAY4
DJNZ R6,DISPLAY_00
DJNZ R5,LOOP_002
DJNZ R4,LOOP_001
RET
LOOP_01:MOV R4,#25
LOOP_011:MOV R5,#50
LOOP_012:MOV R6,#10
DISPLAY_01:
```

```
ACALL DISP_0
ACALL CP_10
ACALL DELAY4
ACALL CP_11
ACALL DISP_1
ACALL CP_01
ACALL DELAY4
ACALL CP_11
DJNZ R6,DISPLAY_01
DJNZ R5,LOOP_012
DJNZ R4,LOOP_011
RET
LOOP_02:MOV R4,#25
LOOP_021:MOV R5,#50
LOOP_022:MOV R6,#10
DISPLAY_02:
ACALL DISP_0
ACALL CP_10
ACALL DELAY4
ACALL CP_11
ACALL DISP_2
ACALL CP_01
ACALL CP_11
DJNZ R6,DISPLAY_02
DJNZ R5,LOOP_022
DJNZ R4,LOOP_021
RET
LOOP_03:MOV R4,#25
LOOP_031:MOV R5,#50
LOOP_032:MOV R6,#10
DISPLAY_03:
ACALL DISP_0
ACALL CP_10
ACALL DELAY4
ACALL CP_11
ACALL DISP_3
ACALL CP_01
ACALL DELAY4
ACALL CP_11
DJNZ R6,DISPLAY_03
DJNZ R5,LOOP_032
DJNZ R4,LOOP_031
RET
LOOP_04:MOV R4,#25
LOOP_041:MOV R5,#50
LOOP_042:MOV R6,#10
DISPLAY_04:
ACALL DISP_0
ACALL CP_10
ACALL DELAY4
ACALL CP_11
ACALL DISP_4
ACALL CP_01
ACALL DELAY4
```

```
ACALL CP_11
DJNZ R6,DISPLAY_04
DJNZ R5,LOOP_042
DJNZ R4,LOOP_041
RET
LOOP_05:MOV R4,#25
LOOP_051:MOV R5,#50
LOOP_052:MOV R6,#10
DISPLAY_05:
ACALL DISP_0
ACALL CP_10
ACALL DELAY4
ACALL CP_11
ACALL DISP_5
ACALL CP_01
ACALL DELAY4
ACALL CP_11
DJNZ R6,DISPLAY_05
DJNZ R5,LOOP_052
DJNZ R4,LOOP_051
RET
LOOP_06:MOV R4,#25
LOOP_061:MOV R5,#50
LOOP_062:MOV R6,#10
DISPLAY_06:
ACALL DISP_0
ACALL CP_10
ACALL DELAY4
ACALL CP_11
ACALL DISP_6
ACALL CP_01
ACALL DELAY4
ACALL CP_11
DJNZ R6,DISPLAY_06
DJNZ R5,LOOP_062
DJNZ R4,LOOP_061
RET
LOOP_15:MOV R4,#25
LOOP_151:MOV R5,#50
LOOP_152:MOV R6,#10
DISPLAY_15:
ACALL DISP_1
ACALL CP_10
ACALL DELAY4
ACALL CP_11
ACALL DISP_5
ACALL CP_01
ACALL DELAY4
ACALL CP_11
DJNZ R6,DISPLAY_15
DJNZ R5,LOOP_152
DJNZ R4,LOOP_151
RET
LOOP_16:MOV R4,#25
```

```
LOOP_161:MOV R5,#50
LOOP_162:MOV R6,#10
DISPLAY_16:
ACALL DISP_1
ACALL CP_10
ACALL DELAY4
ACALL CP_11
ACALL DISP_6
ACALL CP_01
ACALL DELAY4
ACALL CP_11
DJNZ R6,DISPLAY_16
DJNZ R5,LOOP_162
DJNZ R4,LOOP_161
RET
LOOP_17:MOV R4,#25
LOOP_171:MOV R5,#50
LOOP_172:MOV R6,#10
DISPLAY_17:
ACALL DISP_1
ACALL CP_10
ACALL DELAY4
ACALL CP_11
ACALL DISP_7
ACALL CP_01
ACALL DELAY4
ACALL CP_11
DJNZ R6,DISPLAY_17
DJNZ R5,LOOP_172
DJNZ R4,LOOP_171
RET
LOOP_18:MOV R4,#25
LOOP_181:MOV R5,#50
LOOP_182:MOV R6,#10
DISPLAY_18:
ACALL DISP_1
ACALL CP_10
ACALL DELAY4
ACALL CP_11
ACALL DISP_8
ACALL CP_01
ACALL DELAY4
ACALL CP_11
DJNZ R6,DISPLAY_18
DJNZ R5,LOOP_182
DJNZ R4,LOOP_181
RET
LOOP_19:MOV R4,#25
LOOP_191:MOV R5,#50
LOOP_192:MOV R6,#10
DISPLAY_19:
ACALL DISP_1
ACALL CP_10
ACALL DELAY4
```

```
ACALL CP_11
ACALL DISP_9
ACALL CP_01
ACALL DELAY4
ACALL CP_11
DJNZ R6,DISPLAY_19
DJNZ R5,LOOP_192
DJNZ R4,LOOP_191
RET
LOOP_20:MOV R4,#25
LOOP_201:MOV R5,#50
LOOP_202:MOV R6,#10
DISPLAY_20:
ACALL DISP_2
ACALL CP_10
ACALL DELAY4
ACALL CP_11
ACALL DISP_0
ACALL CP_01
ACALL DELAY4
ACALL CP_11
DJNZ R6,DISPLAY_20
DJNZ R5,LOOP_202
DJNZ R4,LOOP_201
RET
LOOP_21:MOV R4,#25
LOOP_211:MOV R5,#50
LOOP_212:MOV R6,#10
DISPLAY_21:
ACALL DISP_2
ACALL CP_10
ACALL DELAY4
ACALL CP_11
ACALL DISP_1
ACALL CP_01
ACALL DELAY4
ACALL CP_11
DJNZ R6,DISPLAY_21
DJNZ R5,LOOP_212
DJNZ R4,LOOP_211
RET
LOOP_22:MOV R4,#25
LOOP_221:MOV R5,#50
LOOP_222:MOV R6,#10
DISPLAY_22:
ACALL DISP_2
ACALL CP_10
ACALL DELAY4
ACALL CP_11
ACALL DISP_2
ACALL CP_01
ACALL DELAY4
ACALL CP_11
DJNZ R6,DISPLAY_22
```

```
DJNZ R5,LOOP_222
DJNZ R4,LOOP_221
RET
LOOP_23:MOV R4,#25
LOOP_231:MOV R5,#50
LOOP_232:MOV R6,#10
DISPLAY_23:
ACALL DISP_2
ACALL CP_10
ACALL DELAY4
ACALL CP_11
ACALL DISP_3
ACALL CP_01
ACALL DELAY4
ACALL CP_11
DJNZ R6,DISPLAY_23
DJNZ R5,LOOP_232
DJNZ R4,LOOP_231
RET
LOOP_29:MOV R4,#25
LOOP_291:MOV R5,#50
LOOP_292:MOV R6,#10
DISPLAY_29:
ACALL DISP_2
ACALL CP_10
ACALL DELAY4
ACALL CP_11
ACALL DISP_9
ACALL CP_01
ACALL DELAY4
ACALL CP_11
DJNZ R6,DISPLAY_29
DJNZ R5,LOOP_292
DJNZ R4,LOOP_291
RET
LOOP_47:MOV R4,#25
LOOP_471:MOV R5,#50
LOOP_472:MOV R6,#10
DISPLAY_47:
ACALL DISP_4
ACALL CP_10
ACALL DELAY4
ACALL CP_11
ACALL DISP_7
ACALL CP_01
ACALL DELAY4
ACALL CP_11
DJNZ R6,DISPLAY_47
DJNZ R5,LOOP_472
DJNZ R4,LOOP_471
RET
LOOP_48:MOV R4,#25
LOOP_481:MOV R5,#50
LOOP_482:MOV R6,#10
```

```
DISPLAY_48:
ACALL DISP_4
ACALL CP_10
ACALL DELAY4
ACALL CP_11
ACALL DISP_8
ACALL CP_01
ACALL DELAY4
ACALL CP_11
DJNZ R6,DISPLAY_48
DJNZ R5,LOOP_482
DJNZ R4,LOOP_481
RET
LOOP_49:MOV R4,#25
LOOP_491:MOV R5,#50
LOOP_492:MOV R6,#10
DISPLAY_49:
ACALL DISP_4
ACALL CP_10
ACALL DELAY4
ACALL CP_11
ACALL DISP_9
ACALL CP_01
ACALL DELAY4
ACALL CP_11
DJNZ R6,DISPLAY_49
DJNZ R5,LOOP_492
DJNZ R4,LOOP_491
RET
LOOP_50:MOV R4,#25
LOOP_501:MOV R5,#50
LOOP_502:MOV R6,#10
DISPLAY_50:
ACALL DISP_5
ACALL CP_10
ACALL DELAY4
ACALL CP_11
ACALL DISP_0
ACALL CP_01
ACALL DELAY4
ACALL CP_11
DJNZ R6,DISPLAY_50
DJNZ R5,LOOP_502
DJNZ R4,LOOP_501
RET
LOOP_51:MOV R4,#25
LOOP_511:MOV R5,#50
LOOP_512:MOV R6,#10
DISPLAY_51:
ACALL DISP_5
ACALL CP_10
ACALL DELAY4
ACALL CP_11
ACALL DISP_1
```

```
ACALL CP_01
ACALL DELAY4
ACALL CP_11
DJNZ R6,DISPLAY_51
DJNZ R5,LOOP_512
DJNZ R4,LOOP_511
RET
LOOP_52:MOV R4,#25
LOOP_521:MOV R5,#50
LOOP_522:MOV R6,#10
DISPLAY_52:
ACALL DISP_5
ACALL CP_10
ACALL DELAY4
ACALL CP_11
ACALL DISP_2
ACALL CP_01
ACALL DELAY4
ACALL CP_11
DJNZ R6,DISPLAY_52
DJNZ R5,LOOP_522
DJNZ R4,LOOP_521
RET
LOOP_53:MOV R4,#25
LOOP_531:MOV R5,#50
LOOP_532:MOV R6,#10
DISPLAY_53:
ACALL DISP_5
ACALL CP_10
ACALL DELAY4
ACALL CP_11
ACALL DISP_3
ACALL CP_01
ACALL DELAY4
ACALL CP_11
DJNZ R6,DISPLAY_53
DJNZ R5,LOOP_532
DJNZ R4,LOOP_531
RET
LOOP_54:MOV R4,#25
LOOP_541:MOV R5,#50
LOOP_542:MOV R6,#10
DISPLAY_54:
ACALL DISP_5
ACALL CP_10
ACALL DELAY4
ACALL CP_11
ACALL DISP_4
ACALL CP_01
ACALL DELAY4
ACALL CP_11
DJNZ R6,DISPLAY_54
DJNZ R5,LOOP_542
DJNZ R4,LOOP_541
```

```
RET
DELAY4:MOV R7,#5
DELAY5:
DJNZ R7,DELAY5
RET
DELAY:MOV R1,#255
DELAY1:MOV R2,#255
DELAY2:MOV R3,#10
DELAY3:DJNZ R3,DELAY3
DJNZ R2,DELAY2
DJNZ R1,DELAY1
RET

DISP_0:
MOV DPTR,#0E800H
MOV A, #00111111B
MOVX @DPTR, A
RET

DISP_1:
MOV DPTR,#0E800H
MOV A, #00000110B
MOVX @DPTR, A
RET

DISP_2:
MOV DPTR,#0E800H
MOV A, #01011011B
MOVX @DPTR, A
RET

DISP_3:
MOV DPTR,#0E800H
MOV A, #01001111B
MOVX @DPTR, A
RET

DISP_4:
MOV DPTR,#0E800H
MOV A, #01100110B
MOVX @DPTR, A
RET

DISP_5:
MOV DPTR,#0E800H
MOV A, #01101101B
MOVX @DPTR, A
RET

DISP_6:
MOV DPTR,#0E800H
MOV A, #01111101B
MOVX @DPTR, A
RET
```

```
DISP_7:
MOV DPTR,#0E800H
MOV A, #00000111B
MOVX @DPTR, A
RET

DISP_8:
MOV DPTR,#0E800H
MOV A, #01111111B
MOVX @DPTR, A
RET

DISP_9:
MOV DPTR,#0E800H
MOV A, #01101111B
MOVX @DPTR, A
RET

CP_01:
MOV DPTR,#0E801H
MOV A, #00000001B
MOVX @DPTR, A
RET

CP_10:
MOV DPTR,#0E801H
MOV A, #00000010B
MOVX @DPTR, A
RET
CP_11:
MOV DPTR,#0E801H
MOV A,#00000011
MOVX @DPTR,A
RET

END
```

CODE EXPLANATION:

1. Initialization:
   - The program starts at memory address 8100H.
   - The data pointer (DPTR) is loaded with the address of the display port (0E803H).
   - The accumulator (A) is loaded with the binary value 10000000B, which likely configures the display port for output.

2. Main Loop (DISPLAY):
   - The main loop repeatedly calls subroutines to display different patterns on the display.
   - Each call to a subroutine displays a specific pattern on the display for a short duration.

3. Subroutines (LOOP_X):
   - There are several subroutines named LOOP_00 to LOOP_54, each responsible for displaying a specific digit or pattern on the display.
   - Each subroutine loops through a sequence of display operations to show the desired pattern.
   - Inside each loop, specific segments of the display are activated to represent the digit or pattern.
   - After displaying the pattern, the subroutine returns to the main loop.

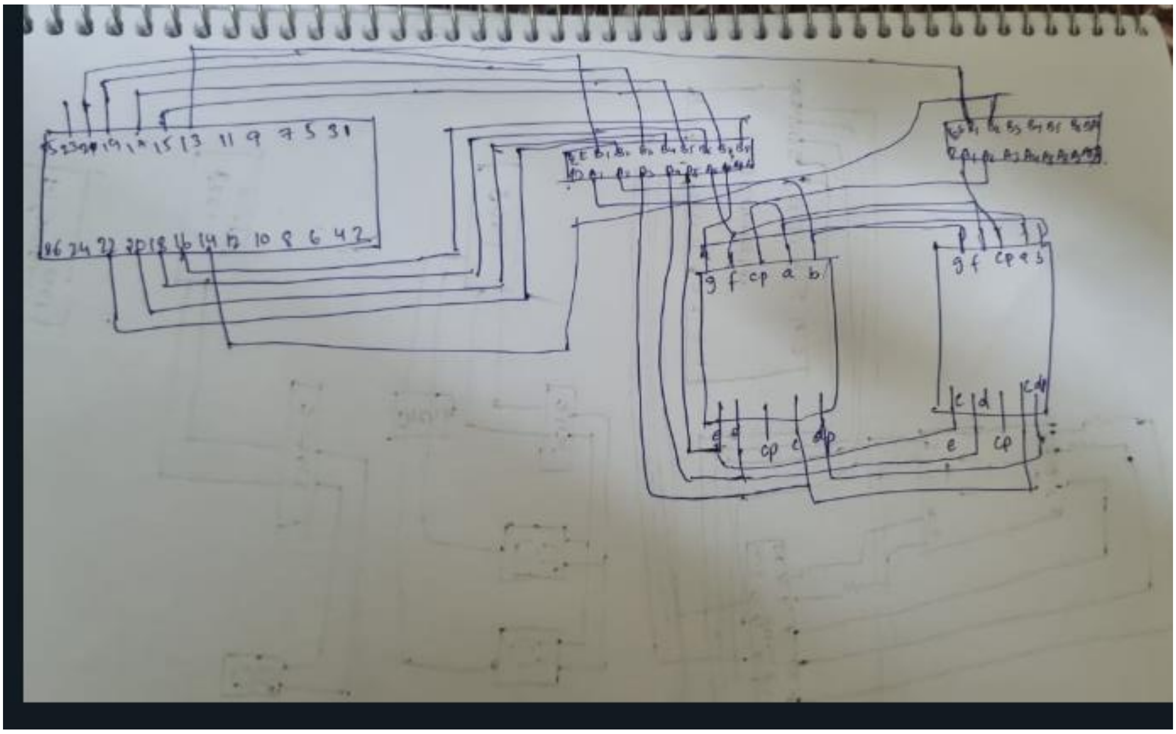
4. Delay Subroutines (DELAY4 and DELAY):
   - These subroutines introduce delays in the program execution to control the timing of the display operations.
   - DELAY4 introduces a shorter delay, likely used for timing between individual segments or transitions.
   - DELAY introduces a longer delay, likely used for timing between complete display updates.

5. Display Control Subroutines (DISP_X and CP_X):
   - DISP_X subroutines set up the display to show a specific digit or pattern.
   - CP_X subroutines set control signals for the display, such as enabling specific segments or transitions.

6. End of Program:
   - The program ends with the END directive.

FLOWCHART:

CIRCUIT DIAGRAM:



PART-C:

Objective: Design a digital clock using common-cathode 7-segment display modules and a mode switch. The clock normally displays the time in hh-mm-ss format. It updates time automatically using the timer interrupt of the microcontroller. On pressing the mode switch, the display changes to stopwatch mode in hh-mm-ss format. In stopwatch mode there are two more buttons – start and stop. The start button starts the stopwatch resetting it to zero, stop button stops the stopwatch. It should be noted that in the stopwatch mode, both normal clock and stopwatch clock get updated with timer interrupt. This ensures that the normal time also gets updated during the run of stopwatch. On pressing the mode button once more, the display returns to show clock time.

Pseudo code:

```
INITIALIZATION:
  Set the interrupt service routine (ISR) address to 0FFF0H
  Set the origin (start address) to 8100H
  Enable external interrupts (INT0 and Timer 0 interrupt)
  Initialize external memory locations from 9000H to 9009H with specific values
  Initialize external memory location 0E803H with a specific value (possibly for LCD control)
  Configure Timer 0 in Mode 1 (16-bit timer)
  Initialize Timer 0 with an initial count of 0
  Start Timer 0
  Set R0 register to 0DH (used as a countdown timer)
  Initialize external RAM locations from 9100H to 9105H with specific values (presumably for digit display)

MAIN LOOP:
  LOOP:
    CALL COMMONPOINT subroutine with a specific value to control common point of display
    Read value from external RAM location 9100H and store in R1
    CALL DISPLAY subroutine to display the value in R1 on the display
    CALL COMMONPOINT subroutine with a specific value to control common point of display
    Repeat the above steps for external RAM locations 9101H to 9105H
    JUMP to LOOP

SUBROUTINES:
  DISPLAY:
    Move the value in R1 to the external RAM location 0E800H (possibly for LCD data)
    Return

  COMMONPOINT:
    Move the value in R2 to the external RAM location 0E801H (possibly for LCD control).
    Return

INTERRUPT SERVICE ROUTINE (ISR):
  Stop Timer 0
  Clear Timer 0 overflow flag
  IF R0 is not zero:
    JUMP to Shortdec
  ELSE:
```

```
  Reset R0 to 0DH
  Read value from external RAM location 9100H
  IF value is not 9:
    Increment value and store it back to 9100H
    JUMP to Shortdec
  ELSE:
    JUMP to CHECK1

CHECK1:
  Read value from external RAM location 9101H
  IF value is not 5:
    Reset value in 9100H to 0
    Increment value in 9101H and store it back
    JUMP to Shortdec
  ELSE:
    JUMP to CHECK2

CHECK2:
  Read value from external RAM location 9102H
  IF value is not 9:
    JUMP to INC2Short
  ELSE:
    Read value from 9103H
    IF value is not 5:
      JUMP to INCDISP3
    ELSE:
      Read value from 9104H
      IF value is not 3:
        JUMP to INCDISP4
      ELSE:
        Read value from 9105H
        IF value is not 2:
          JUMP to INCDISP5
        ELSE:
          Reset all values from 9100H to 9105H to 0
          JUMP to DECREASE

INC2Short:
  JUMP to INC2

Shortdec:
  JUMP to DECREASE

INCDISP5:
  Reset values from 9100H to 9104H to 0
  Increment value in 9105H and store it back
  JUMP to DECREASE

INCDISP4:
  Reset values from 9100H to 9103H to 0
  Increment value in 9104H and store it back
```

JUMP to DECREASE

INCDISP3:
Reset values from 9100H to 9102H to 0
Increment value in 9103H and store it back
JUMP to DECREASE

INC2:
Reset values from 9100H to 9101H to 0
Increment value in 9102H and store it back
JUMP to DECREASE

DECREASE:
Decrement R0 (countdown timer)
Reset Timer 0 with an initial count of 0
Start Timer 0
Return from ISR

END

CODE:

```
ORG 0FFF0H
LJMP ISR

ORG 8100H


MOV IE, #82H

MOV DPTR, #9000H
MOV A, #00111111B
MOVX @DPTR, A

MOV DPTR, #9001H
MOV A, #00000110B
MOVX @DPTR, A

MOV DPTR, #9002H
MOV A, #01011011B
MOVX @DPTR, A

MOV DPTR, #9003H
MOV A, #01001111B
MOVX @DPTR, A

MOV DPTR, #9004H
MOV A, #01100110B
MOVX @DPTR, A

MOV DPTR, #9005H
MOV A, #01101101B
MOVX @DPTR, A
```

```
MOV DPTR, #9006H
MOV A, #01111101B
MOVX @DPTR, A

MOV DPTR, #9007H
MOV A, #00000111B
MOVX @DPTR, A

MOV DPTR, #9008H
MOV A, #01111111B
MOVX @DPTR, A

MOV DPTR, #9009H
MOV A, #01101111B
MOVX @DPTR, A


MOV DPTR, #0E803H
MOV A, #10000001B
MOVX @DPTR, A



MOV TMOD, #01H
MOV TL0, #00H
MOV TH0, #00H
SETB TR0

MOV R0, #0DH

MOV DPTR, #9100H
MOV A, #00H
MOVX @DPTR, A

MOV DPTR, #9101H
MOV A, #5
MOVX @DPTR, A

MOV DPTR, #9102H
MOV A, #9
MOVX @DPTR, A

MOV DPTR, #9103H
MOV A, #5
MOVX @DPTR, A

MOV DPTR, #9104H
MOV A, #3
MOVX @DPTR, A

MOV DPTR, #9105H
MOV A, #2
MOVX @DPTR, A
```

```
NORMAL_TIME :
MOV R2, #11111111B
ACALL COMMONPOINT

MOV DPTR, #9100H
MOVX A, @DPTR
MOV R1, A
ACALL DISPLAY
MOV R2, #11111110B
ACALL COMMONPOINT

MOV R2, #11111111B
ACALL COMMONPOINT

MOV DPTR, #9101H
MOVX A, @DPTR
MOV R1, A
ACALL DISPLAY
MOV R2, #11111101B
ACALL COMMONPOINT

MOV R2, #11111111B
ACALL COMMONPOINT

MOV DPTR, #9102H
MOVX A, @DPTR
MOV R1, A
ACALL DISPLAY
MOV R2, #11111011B
ACALL COMMONPOINT

MOV R2, #11111111B
ACALL COMMONPOINT

MOV DPTR, #9103H
MOVX A, @DPTR
MOV R1, A
ACALL DISPLAY
MOV R2, #11110111B
ACALL COMMONPOINT

MOV R2, #11111111B
ACALL COMMONPOINT

MOV DPTR, #9104H
MOVX A, @DPTR
MOV R1, A
ACALL DISPLAY
MOV R2, #11101111B
ACALL COMMONPOINT

MOV R2, #11111111B
ACALL COMMONPOINT
```

```
MOV DPTR, #9105H
MOVX A, @DPTR
MOV R1, A
ACALL DISPLAY
MOV R2, #11011111B
ACALL COMMONPOINT



SJMP NORMAL_TIME



DISPLAY :
MOV DPH, #90H
MOV DPL, R1
MOVX A, @DPTR
MOV DPTR, #0E800H
MOVX @DPTR, A
RET

COMMONPOINT :
MOV DPTR, #0E801H
MOV A, R2
MOVX @DPTR, A
RET
ISR :
CLR TR0
CLR TF0

CJNE R0, #00H, Shortdec
MOV R0, #0DH

MOV DPTR, #9100H
MOVX A, @DPTR
CJNE A, #09H, INCDISP0
SJMP CHECK1
INCDISP0 :
MOV DPTR, #9100H
MOVX A, @DPTR
INC A
MOVX @DPTR, A
SJMP Shortdec

CHECK1 :
MOV DPTR, #9101H
MOVX A, @DPTR
CJNE A, #05H, INCDISP1
SJMP CHECK2

INCDISP1 :
MOV A, #00H
MOV DPTR, #9100H
MOVX @DPTR, A
```

```
MOV DPTR, #9101H
MOVX A, @DPTR
INC A
MOVX @DPTR, A
SJMP Shortdec

CHECK2 :

MOV DPTR, #9102H
MOVX A, @DPTR
CJNE A, #09H, INC2Short

MOV DPTR, #9103H
MOVX A, @DPTR
CJNE A, #05H, INCDISP3

MOV DPTR, #9104H
MOVX A, @DPTR
CJNE A, #03H, INCDISP4

MOV DPTR, #9105H
MOVX A, @DPTR
CJNE A, #02H, INCDISP5

MOV A, #00H
MOV DPTR, #9100H
MOVX @DPTR, A
MOV DPTR, #9101H
MOVX @DPTR, A
MOV DPTR, #9102H
MOVX @DPTR, A
MOV DPTR, #9103H
MOVX @DPTR, A
MOV DPTR, #9104H
MOVX @DPTR, A
MOV DPTR, #9105H
MOVX @DPTR, A

SJMP DECREASE

INC2Short :
SJMP INC2
Shortdec :
SJMP DECREASE

INCDISP5 :
MOV A, #00H
MOV DPTR, #9100H
MOVX @DPTR, A
MOV DPTR, #9101H
MOVX @DPTR, A
MOV DPTR, #9102H
MOVX @DPTR, A
```

```
MOV DPTR, #9103H
MOVX @DPTR, A
MOV DPTR, #9104H
MOVX @DPTR, A

MOV DPTR, #9105H
MOVX A, @DPTR
INC A
MOVX @DPTR, A
SJMP DECREASE


INCDISP4 :
MOV A, #00H
MOV DPTR, #9100H
MOVX @DPTR, A
MOV DPTR, #9101H
MOVX @DPTR, A
MOV DPTR, #9102H
MOVX @DPTR, A
MOV DPTR, #9103H
MOVX @DPTR, A

MOV DPTR, #9104H
MOVX A, @DPTR
INC A
MOVX @DPTR, A
SJMP DECREASE

INCDISP3 :
MOV A, #00H
MOV DPTR, #9100H
MOVX @DPTR, A
MOV DPTR, #9101H
MOVX @DPTR, A
MOV DPTR, #9102H
MOVX @DPTR, A

MOV DPTR, #9103H
MOVX A, @DPTR
INC A
MOVX @DPTR, A
SJMP DECREASE
INC2 :
MOV A, #00H
MOV DPTR, #9100H
MOVX @DPTR, A
MOV DPTR, #9101H
MOVX @DPTR, A
MOV DPTR, #9102H
MOVX A, @DPTR
INC A
MOVX @DPTR, A
SJMP DECREASE
```

```
DECREASE :
DEC R0
MOV TL0, #00H
MOV TH0, #00H
SETB TR0

RETI

END
```

CODE EXPLANATION:

1. Initialization:
   - Set the interrupt service routine (ISR) address to 0FFF0H.
   - Set the origin (start address) to 8100H.
   - Enable external interrupts (INT0) and Timer 0 interrupt.
   - Initialize external RAM locations from 9000H to 9009H with specific values.
   - Initialize external RAM location 0E803H with a specific value (possibly for LCD control).
   - Configure Timer 0 in Mode 1 (16-bit timer) and initialize it with an initial count of 0.
   - Start Timer 0.
   - Set the R0 register to 0DH (used as a countdown timer).
   - Initialize external RAM locations from 9100H to 9105H with specific values (presumably for digit display).

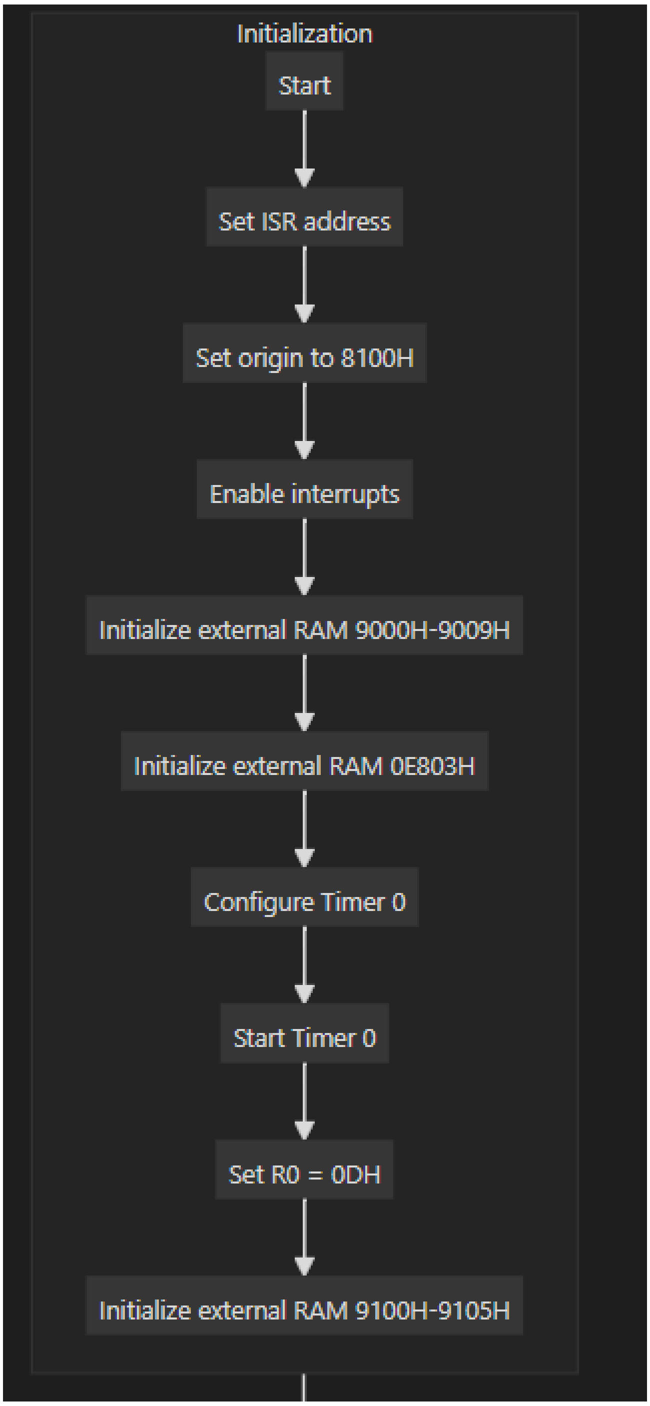
2. Main Loop:
   - Continuously display the values stored in external RAM locations 9100H to 9105H on a display device (possibly an LCD) by calling the `DISPLAY` and `COMMONPOINT` subroutines with specific values.

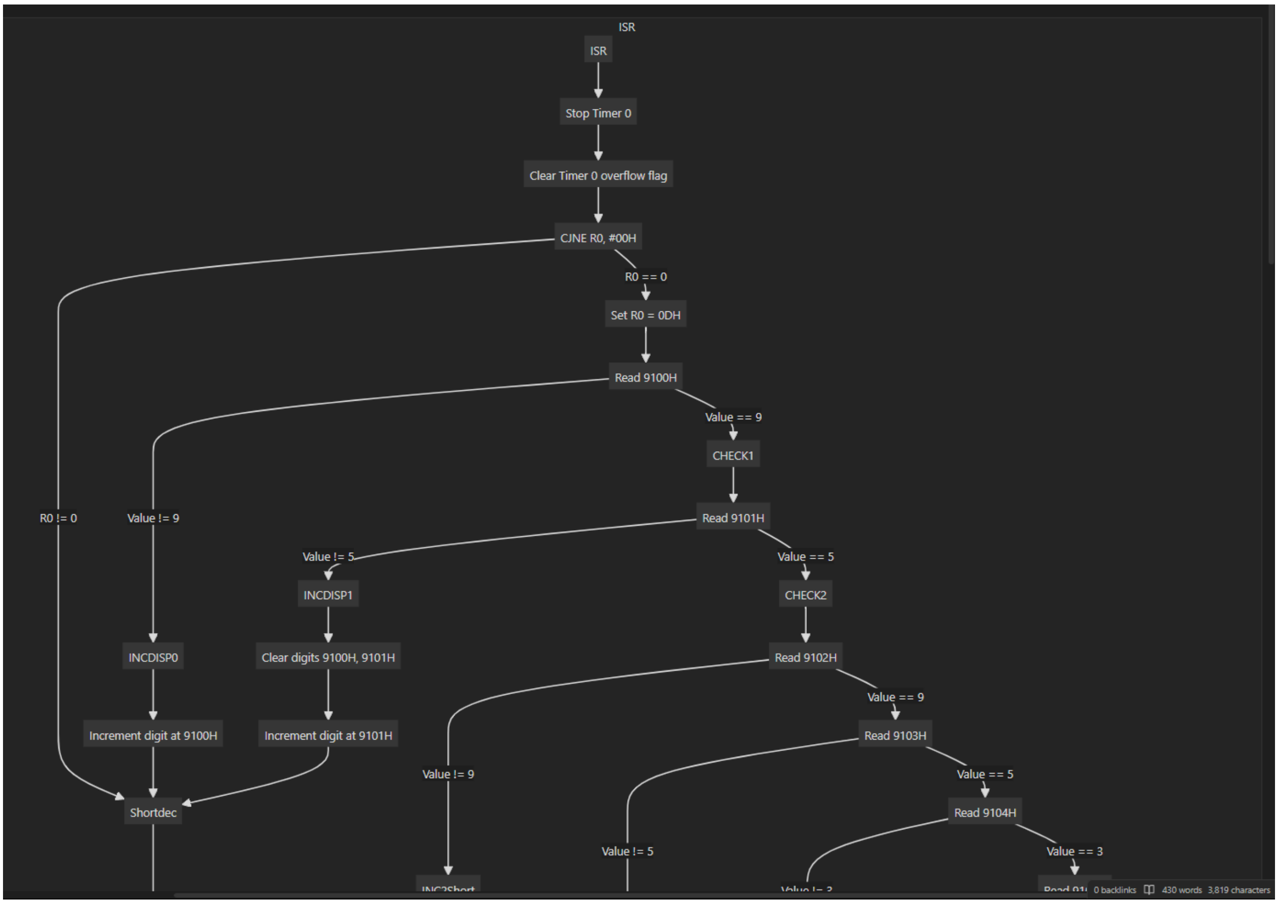
3. Subroutines:
   - `DISPLAY`: Move the value stored in the external RAM location pointed to by `DPTR` (set using the value in R1) to the external RAM location 0E800H, possibly for LCD data.
   - `COMMONPOINT`: Move the value stored in R2 to the external RAM location 0E801H, possibly for LCD control (e.g., setting the common point for digit display).

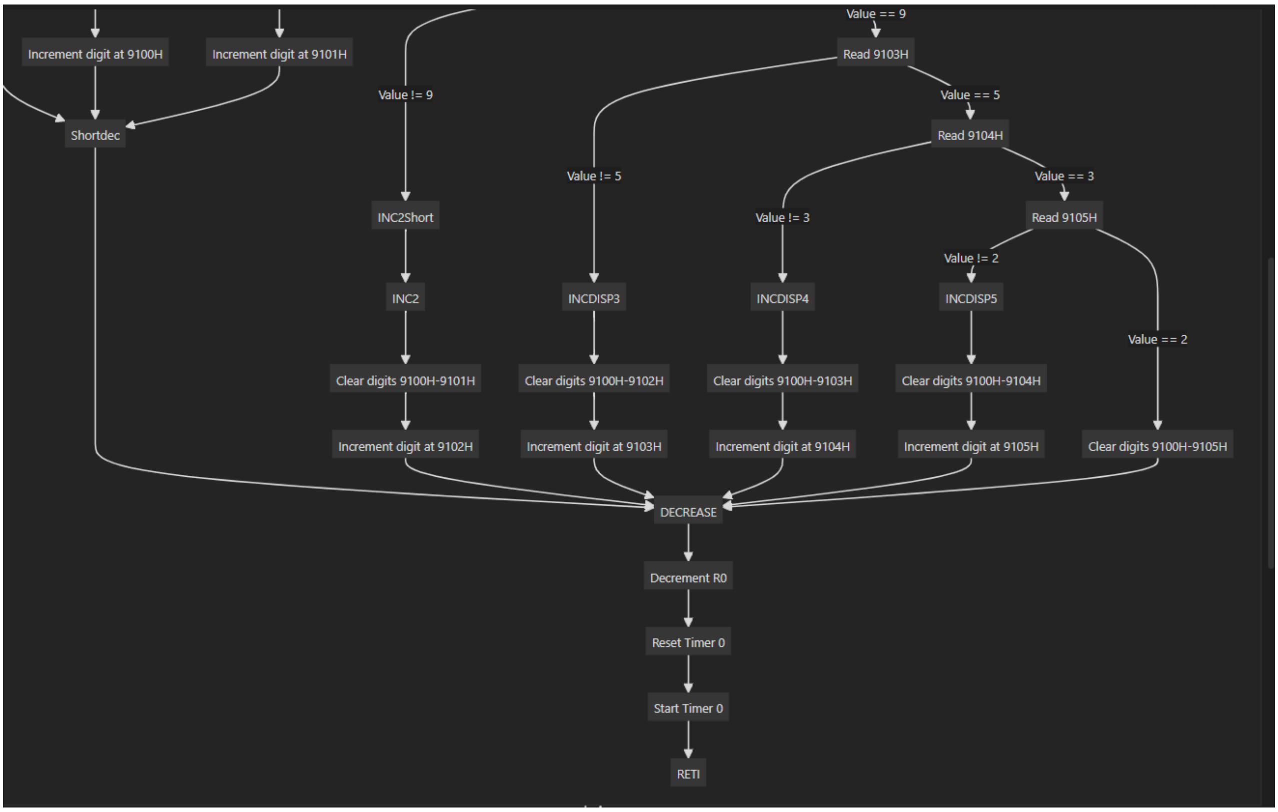
4. Interrupt Service Routine (ISR):
   - Stop Timer 0 and clear the Timer 0 overflow flag.
   - Check and update the values stored in the external RAM locations (9100H to 9105H) based on specific conditions, effectively implementing a countdown timer or clock functionality.
   - Decrement the value in R0 (countdown timer).
   - Reset Timer 0 with an initial count of 0.
   - Start Timer 0 again.
   - Return from the ISR.

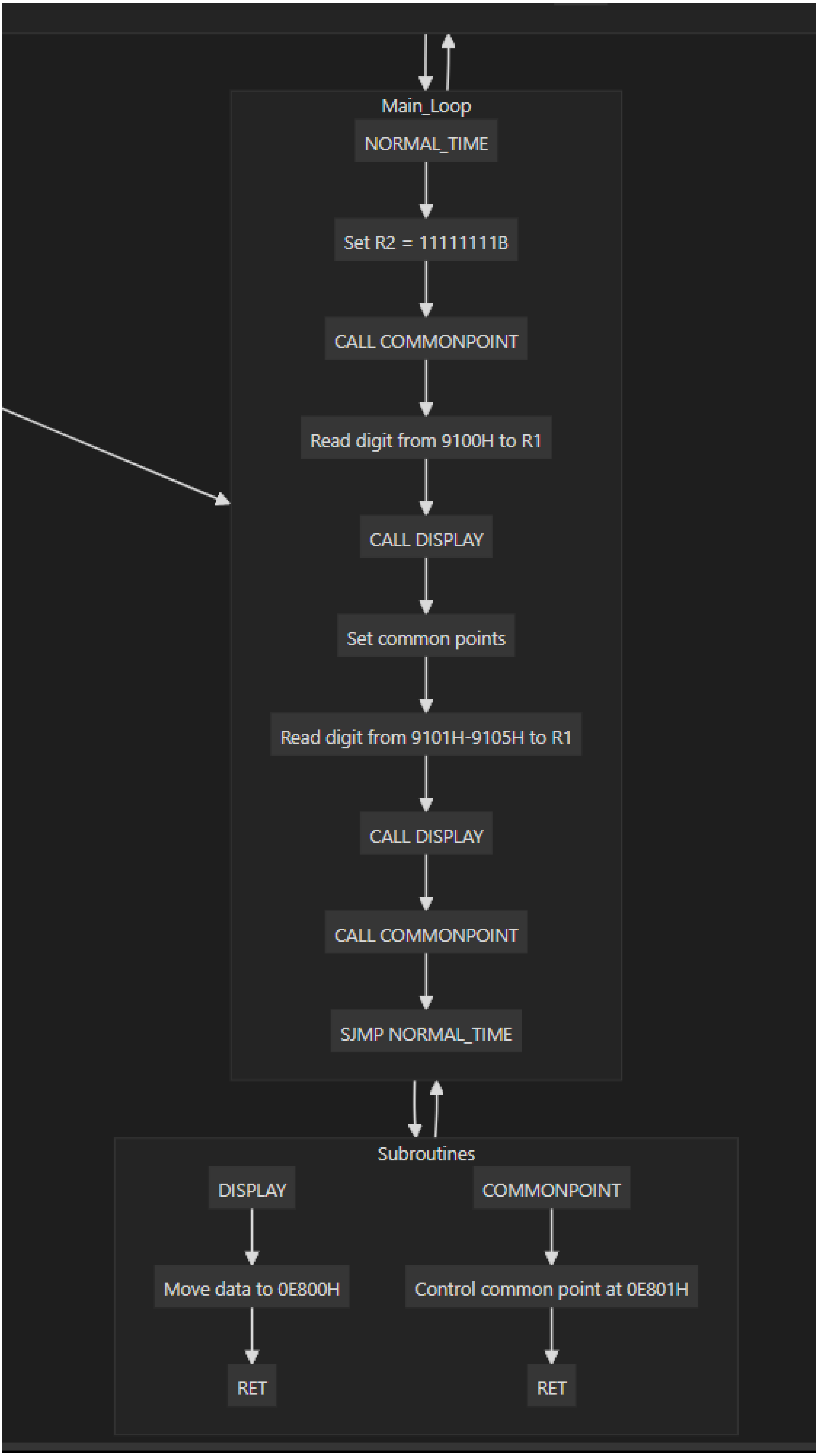
5. Functionality:
   - The program appears to be implementing a digital clock or timer display using an external memory and a display device (possibly an LCD).
   - The main loop continuously displays the values stored in the external RAM locations (9100H to 9105H) on the display.
   - The interrupt service routine (ISR) is triggered periodically by Timer 0 and updates the values in the external RAM based on specific conditions, effectively implementing a countdown timer or clock functionality.

Flow Chart:

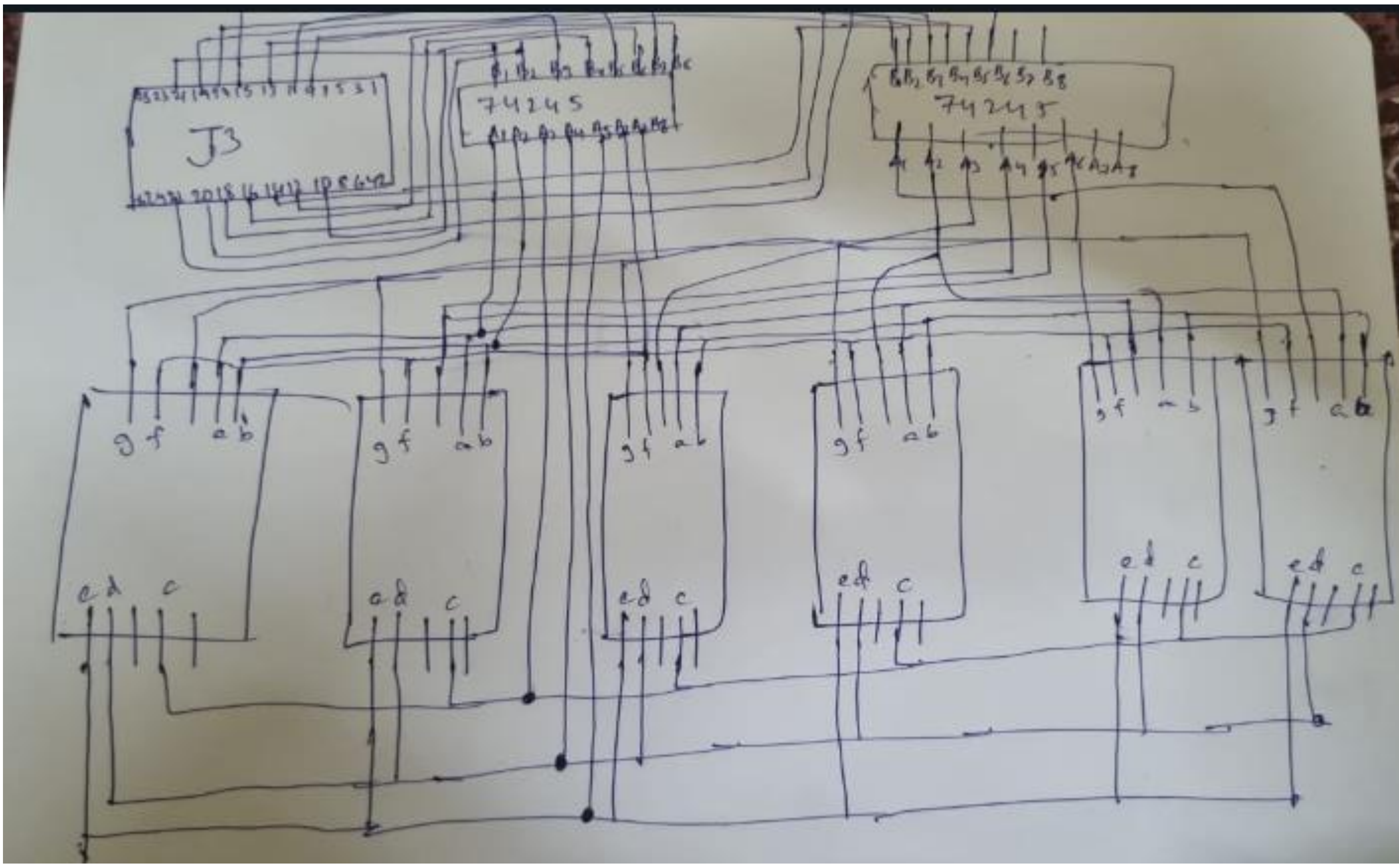

Initialization
Start
Set ISR address
Set origin to 8100H
Enable interrupts
Initialize external RAM 9000H-9009H
Initialize external RAM 0E803H
Configure Timer 0
Start Timer 0
Set R0 = 0DH
Initialize external RAM 9100H-9105H


ISR
ISR
Stop Timer 0
Clear Timer 0 overflow flag
CJNE R0, #00H
R0 == 0
Set R0 = 0DH
Read 9100H
Value == 9
CHECK1
Read 9101H
R0 != 0
Value != 9
Value != 5
Value == 5
INCDISP1
CHECK2
INCDISP0
Clear digits 9100H, 9101H
Read 9102H
Value == 9
Increment digit at 9100H
Increment digit at 9101H
Read 9103H
Value != 9
Value == 5
Shortdec
Read 9104H
Value != 5
Value == 3


Value == 9
Increment digit at 9100H
Increment digit at 9101H
Read 9103H
Value != 9
Value == 5
Shortdec
Read 9104H
Value != 5
Value == 3
INC2Short
Value != 3
Read 9105H
Value != 2
INC2
INCDISP3
INCDISP4
INCDISP5
Value == 2
Clear digits 9100H-9101H
Clear digits 9100H-9102H
Clear digits 9100H-9103H
Clear digits 9100H-9104H
Increment digit at 9102H
Increment digit at 9103H
Increment digit at 9104H
Increment digit at 9105H
Clear digits 9100H-9105H
DECREASE
Decrement R0
Reset Timer 0
Start Timer 0
RETI


Main_Loop
NORMAL_TIME
Set R2 = 11111111B
CALL COMMONPOINT
Read digit from 9100H to R1
CALL DISPLAY
Set common points
Read digit from 9101H-9105H to R1
CALL DISPLAY
CALL COMMONPOINT
SJMP NORMAL_TIME
Subroutines
DISPLAY
Move data to 0E800H
RET
COMMONPOINT
Control common point at 0E801H
RET

CIRCUIT DIAGRAM:



Discussion:

J3 Interface:
J3 is a physical connector on the microcontroller or development board that provides access to specific input/output (I/O) pins.
It could be a header connector with multiple pins, each of which can be individually controlled by the microcontroller.
Control Signal:
The control signal sent through J3 is used to manipulate the segments of the 7-segment display.
By toggling specific pins on J3 high or low, the microcontroller can turn on or off individual segments of the 7-segment display.
The control signal is generated by the microcontroller based on the desired pattern or number to be displayed on the 7-segment display.
LED Segments:
The internal of a seven-segment display is its light-emitting diodes (LEDs), which are arranged in the shape of a figure-eight or "8" pattern.
There are seven LED segments, each representing a specific part of the overall digit. These segments are labeled as 'a', 'b', 'c', 'd', 'e', 'f', and 'g', with an additional eighth segment for the decimal point if required.

The 8051 microcontroller's timer/counter module is a vital component for managing timing-related tasks in various applications. It offers two primary modes: Timer mode for time measurement and Counter mode for event counting. Through registers like TCON and TMOD, developers can configure and control the timers' operation modes and settings.

Interrupts triggered by timer overflows allow precise timing control and event handling. Programmers typically set up the timer by configuring the mode, loading initial values, and starting or stopping it as needed. Timers find extensive use in tasks requiring accurate timing, such as real-time clocks, PWM for motor control, and event counting in industrial automation.

The timer's accuracy depends on the system's clock source, with external crystal oscillators offering higher precision. Overall, the timer/counter module in the 8051 microcontroller plays a crucial role in time-sensitive applications, providing precise timing control essential for various embedded systems and electronic devices.

Interrupts in the 8051 microcontroller serve to swiftly respond to external events or conditions without constant polling. They come from various sources like external hardware, timer/counters overflow, or serial communication. Prioritized

interrupts enable preemptive handling based on priority, while non-prioritized ones are handled sequentially.
When an interrupt occurs, the microcontroller transfers control to a dedicated Interrupt Service Routine (ISR), saving and restoring the program state automatically. Interrupts are managed via an Interrupt Vector Table, where addresses of ISRs are stored. They can be enabled/disabled using control registers like IE and IP. After servicing, the microcontroller sends an acknowledgment signal to the source. Nested interrupts allow handling higher-priority events without delay. Overall, interrupts facilitate efficient event-driven programming, enabling the microcontroller to multitask effectively and respond promptly to external events.

THE END