INDIAN INSTITUTE OF TECHNOLOGY

KHARAGPUR

DEPARTMENT OF ELECTRONICS AND ELECTRICAL COMMUNICATION

EC49001

MICROCONTROLLER SYSTEMS LABORATORY

ASSIGNMENT 1

SHIHKAR MOHAN

18EC10054

E & ECE

(B. Tech)

2021 – 2022

Questions

1. Sort a set of 16 numbers stored at RAM locations 30 to 3F. Before running the program, edit those locations to input the numbers. Show the sorted output.
2. Check if the bytes stored at locations 30H to 3FH constitute a palindrome or not. If the bytes make a palindrome, store 01H at location 40H, else store 0FFH at 40H.
3. Convert the number stored at location 30H into BCD format. Store the digits at locations 40H and 41H.

Objectives

1. The question asks us to sort a sequence stored from 30H to 3FH. We use the Bubble Sort algorithm to accomplish this in $O(N^2)$.
2. The question asks us to check if the sequence stored in 30H to 3FH is a palindrome or not. Our objective here is to keep checking the equality of elements equidistant from the ends, which performs the palindrome check in $O(N)$.
3. The question asks us to convert the HEX number stored at 30H to BCD format where the hundreds digit is stored at 40H and the remaining two digits are stored in 41H.

Description

1. The bubble sort algorithm swaps adjacent elements based on one being greater than the other, and as a result in one pass form 1 to 16 (30H to 3FH) the greatest element is "bubbled" down to the 16th position. Repeating the pass now bubbles down the second largest element to the 15th position (since comparing with 16th position will not result in a swap). Since we perform N comparisons each pass, and we perform N passes, the overall time complexity turns out to be $O(N^2)$.

2. For this part, our implementation simply maintains two markers at either ends and we keep comparing as we move inwards. Since we make $N/2$ comparisons, the time complexity is $O(N)$. We keep 01 at 40H and the moment we find the values at the two pointers unequal, we move FF to 40H and break.

3. We divide by 10 and store the remainder in R0, then divide by 10 again to obtain the 100s digit, which is stored to 41H and the remainder (10s digit) stored in B. We multiply 16 to B and add the remainder to obtain the last two digits' decimal representation written in HEX.

Code

Part 1

```
START:
MOV 30H,#16H; Load data as required
MOV 31H,#7H
MOV 32H,#15H
MOV 33H,#2H
MOV 34H,#10H
MOV 35H,#11H
MOV 36H,#12H
MOV 37H,#9H
MOV 38H,#5H
MOV 39H,#1H
MOV 3AH,#8H
MOV 3BH,#6H
MOV 3CH,#3H
MOV 3DH,#14H
MOV 3EH,#4H
MOV 3FH,#13H

MOV R3,#0FH; Size of our array

OUT_LOOP:; Control for the outer loop
MOV B,R3
MOV R4,B; Copy the value in R3 to R4
MOV R0,#30H; Initialise the iterator

IN_LOOP:
MOV B,@R0; Store @R0 to B
INC R0; Increment R0
MOV A,@R0; Store @R0 to A
CJNE A,B,NEXT; Carry is 1 if A>B
NEXT: JNC NO_SWP; If carry is 0, jump to NO_SWP and skip the swap
MOV @R0,B; Following lines swap the two elements
DEC R0
```

```
MOV @R0,A
INC R0

NO_SWP:; Skip swap if not needed
DJNZ R4,IN_LOOP; Perform the comparison N times, making a single pass
DJNZ R3,OUT_LOOP; Perform the pass N times
END
```

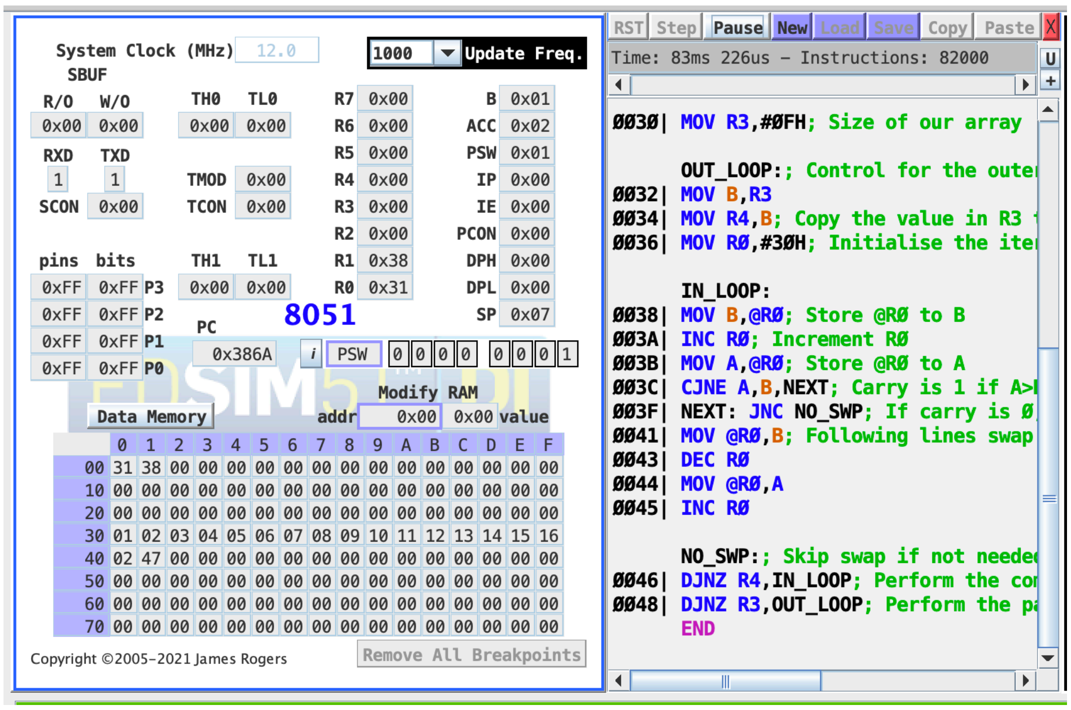
Part 2

```
MOV 30H,#1H; Load data into RAM as necessary
MOV 31H,#2H
MOV 32H,#3H
MOV 33H,#4H
MOV 34H,#5H
MOV 35H,#6H
MOV 36H,#7H
MOV 37H,#8H
MOV 38H,#8H
MOV 39H,#7H
MOV 3AH,#6H
MOV 3BH,#5H
MOV 3CH,#4H
MOV 3DH,#3H
MOV 3EH,#2H
MOV 3FH,#1H

MOV R1,#30H; Left pointer
MOV R0,#3FH; Right pointer
MOV A,#0; Store 0 to Accumulator
MOV R2,#8; Store the amount of traversing needed
; half the length of sequence
MOV 40H,#01; Default is 0, change to FF if parity breaks

CHECK:; loop for checking parity

MOV A,@R1; Move @R1 to A
MOV B,@R0; Move @R0 to B
SUBB A,B; Subtract A and B
JNZ NEXT; If the above is nonzero, go to NEXT
INC R1; Increase R1 for the next iteration
DEC R0; Decrease R0 for the next iteration

DJNZ R2,CHECK; Perform CHECK R2 times
MOV 40H,#01H; If performed CHECK R2 times, move 01 to 40H
JMP EXIT; Jump to the EXIT

NEXT: MOV 40H,#0FFH; A and B found unequal, change 40H to FF and exit
EXIT:; End the program
END
```

Part 3

1. MOV 30H,#247; Store the number in 247
2. MOV A,30H; Move the number to accumulator
3. MOV B,#10; Store 10 in B
4. DIV AB; Divide A by 10
5. MOV R0,B; Store the remainder in R0
6. MOV B,#10; Store 10 in B
7. DIV AB; Divide A by 10 again
8. MOV 40H,A; Store the 100s digit in A
9. MOV A,B; Shift the 10s digit to A
10. MOV B,#16; Set B to 16
11. MUL AB; Multiply 16 to the 10s digit
12. ADD A,R0; Add the initial remainder to A
13. MOV 41H,A; Move this to 41H
14. END

Screenshots



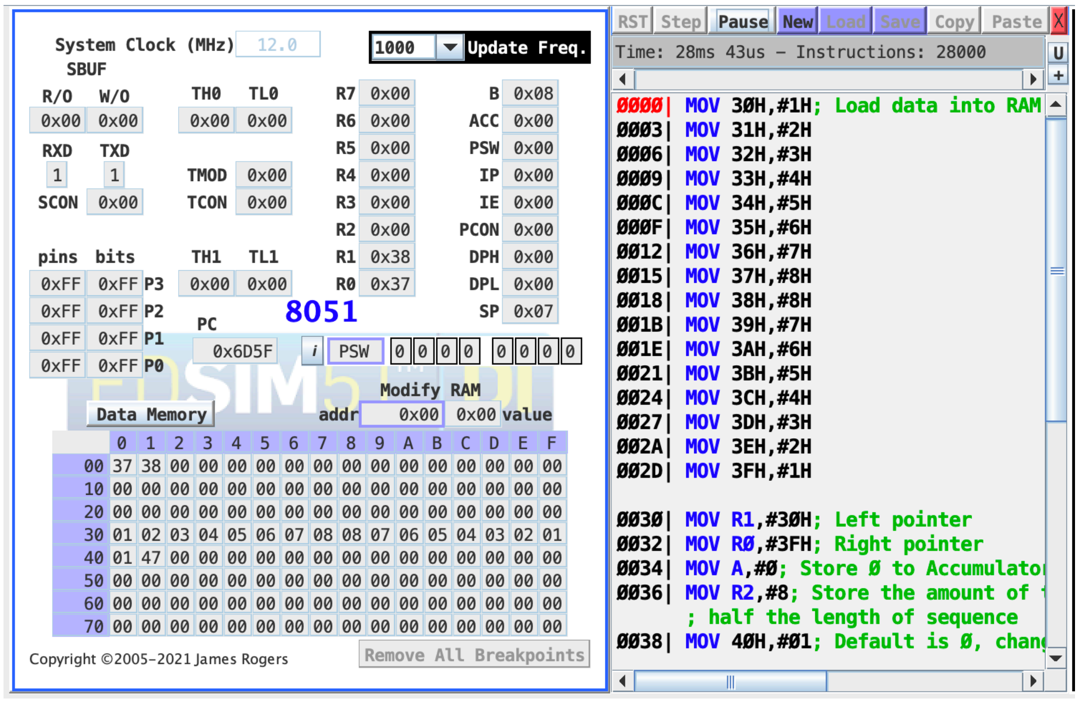
```
System Clock (MHz) 12.0          1000  Update Freq.
SBUF
R/O   W/O      TH0   TL0     R7 0x00      B 0x01
0x00  0x00     0x00  0x00    R6 0x00    ACC 0x02
RXD   TXD                    R5 0x00    PSW 0x01
 1     1       TMOD  0x00    R4 0x00     IP 0x00
SCON  0x00     TCON  0x00    R3 0x00     IE 0x00
                             R2 0x00   PCON 0x00
pins  bits     TH1   TL1     R1 0x38    DPH 0x00
0xFF  0xFF P3  0x00  0x00    R0 0x31    DPL 0x00
0xFF  0xFF P2                            SP 0x07
0xFF  0xFF P1   PC
0xFF  0xFF P0   0x386A     PSW 0 0 0 0  0 0 0 1

                           Modify RAM
                    addr 0x00  0x00 value
```

```
    0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
00 31 38 00 00 00 00 00 00 00 00 00 00 00 00 00 00
10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
30 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16
40 02 47 00 00 00 00 00 00 00 00 00 00 00 00 00 00
50 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
60 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
70 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Copyright ©2005-2021 James Rogers

```
Time: 83ms 226us - Instructions: 82000

0030| MOV R3,#0FH; Size of our array

      OUT_LOOP:; Control for the outer
0032| MOV B,R3
0034| MOV R4,B; Copy the value in R3 t
0036| MOV R0,#30H; Initialise the iter

      IN_LOOP:
0038| MOV B,@R0; Store @R0 to B
003A| INC R0; Increment R0
003B| MOV A,@R0; Store @R0 to A
003C| CJNE A,B,NEXT; Carry is 1 if A>
003F| NEXT: JNC NO_SWP; If carry is 0
0041| MOV @R0,B; Following lines swap
0043| DEC R0
0044| MOV @R0,A
0045| INC R0

      NO_SWP:; Skip swap if not neede
0046| DJNZ R4,IN_LOOP; Perform the co
0048| DJNZ R3,OUT_LOOP; Perform the pa
      END
```

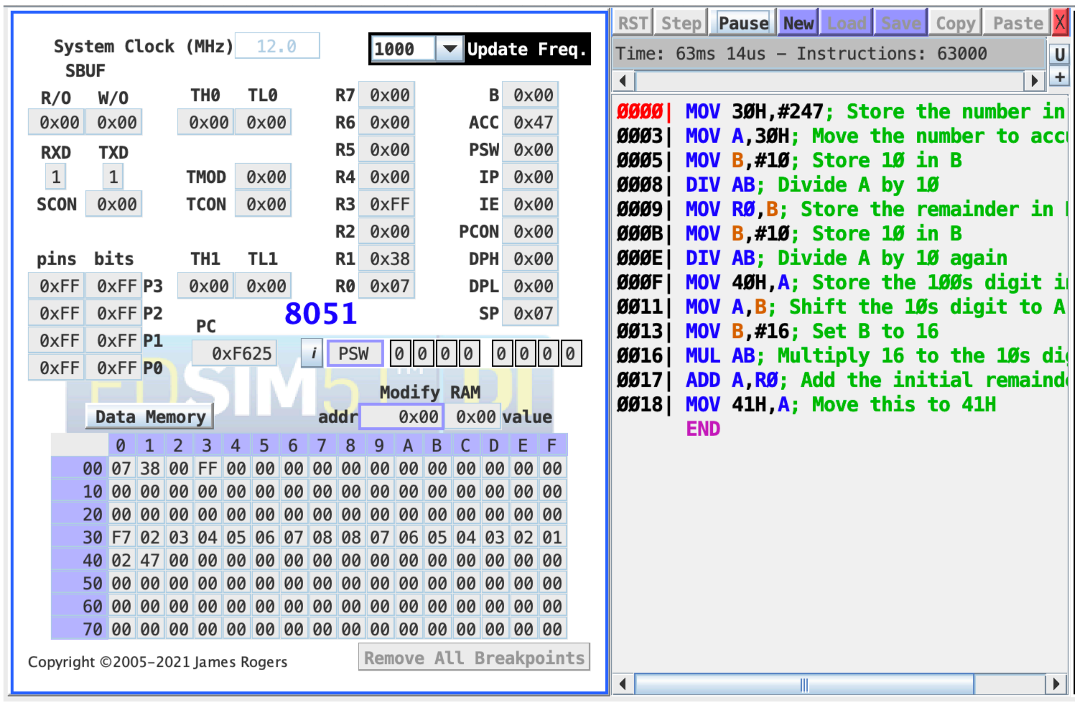
Fig. 1: Corresponding to Part 1. We can observe the sorted sequence from 30H to 3FH.



```
System Clock (MHz) 12.0        1000  Update Freq.
SBUF
R/O   W/O      TH0   TL0     R7 0x00        B 0x08
0x00  0x00     0x00  0x00    R6 0x00      ACC 0x00
RXD   TXD                    R5 0x00      PSW 0x00
 1     1       TMOD  0x00    R4 0x00       IP 0x00
SCON  0x00     TCON  0x00    R3 0x00       IE 0x00
                             R2 0x00     PCON 0x00
pins  bits     TH1   TL1     R1 0x38      DPH 0x00
0xFF  0xFF P3  0x00  0x00    R0 0x37      DPL 0x00
0xFF  0xFF P2              8051            SP 0x07
0xFF  0xFF P1    PC
0xFF  0xFF P0    0x6D5F   i  PSW  0 0 0 0  0 0 0 0
                                Modify RAM
   Data Memory               addr  0x00  0x00 value
     0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
  00 37 38 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  30 01 02 03 04 05 06 07 08 08 07 06 05 04 03 02 01
  40 01 47 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  50 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  60 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  70 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Copyright ©2005-2021 James Rogers     Remove All Breakpoints
```

```
RST Step Pause New Load Save Copy Paste X
Time: 28ms 43us - Instructions: 28000
0000| MOV 30H,#1H; Load data into RAM
0003| MOV 31H,#2H
0006| MOV 32H,#3H
0009| MOV 33H,#4H
000C| MOV 34H,#5H
000F| MOV 35H,#6H
0012| MOV 36H,#7H
0015| MOV 37H,#8H
0018| MOV 38H,#8H
001B| MOV 39H,#7H
001E| MOV 3AH,#6H
0021| MOV 3BH,#5H
0024| MOV 3CH,#4H
0027| MOV 3DH,#3H
002A| MOV 3EH,#2H
002D| MOV 3FH,#1H

0030| MOV R1,#30H; Left pointer
0032| MOV R0,#3FH; Right pointer
0034| MOV A,#0; Store 0 to Accumulato
0036| MOV R2,#8; Store the amount of 
      ; half the length of sequence
0038| MOV 40H,#01; Default is 0, chan
```

Fig. 2: Screenshot for Part 2. We can see the palindrome from 30H to 3FH and the result in 40H.



```
System Clock (MHz) 12.0        1000  Update Freq.
SBUF
R/O   W/O      TH0   TL0     R7 0x00        B 0x00
0x00  0x00     0x00  0x00    R6 0x00      ACC 0x47
RXD   TXD                    R5 0x00      PSW 0x00
 1     1       TMOD  0x00    R4 0x00       IP 0x00
SCON  0x00     TCON  0x00    R3 0xFF       IE 0x00
                             R2 0x00     PCON 0x00
pins  bits     TH1   TL1     R1 0x38      DPH 0x00
0xFF  0xFF P3  0x00  0x00    R0 0x07      DPL 0x00
0xFF  0xFF P2              8051            SP 0x07
0xFF  0xFF P1    PC
0xFF  0xFF P0    0xF625   i  PSW  0 0 0 0  0 0 0 0
                                Modify RAM
   Data Memory               addr  0x00  0x00 value
     0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
  00 07 38 00 FF 00 00 00 00 00 00 00 00 00 00 00 00
  10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  30 F7 02 03 04 05 06 07 08 08 07 06 05 04 03 02 01
  40 02 47 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  50 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  60 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  70 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Copyright ©2005-2021 James Rogers     Remove All Breakpoints
```

```
RST Step Pause New Load Save Copy Paste X
Time: 63ms 14us - Instructions: 63000
0000| MOV 30H,#247; Store the number in
0003| MOV A,30H; Move the number to acc
0005| MOV B,#10; Store 10 in B
0008| DIV AB; Divide A by 10
0009| MOV R0,B; Store the remainder in 
000B| MOV B,#10; Store 10 in B
000E| DIV AB; Divide A by 10 again
000F| MOV 40H,A; Store the 100s digit i
0011| MOV A,B; Shift the 10s digit to A
0013| MOV B,#16; Set B to 16
0016| MUL AB; Multiply 16 to the 10s di
0017| ADD A,R0; Add the initial remaind
0018| MOV 41H,A; Move this to 41H
      END
```

Fig. 3: Screenshot for Part 3. Line 0 shows 247 is the number to be represented, 40H and 41H display the representation.

Results

1. An array sorted in ascending order is obtained.
2. 40H acts as a flag and tells us if we've found a palindrome. The value at 40H is 01H if a palindrome is found and FFH if it isn't.
3. The required HEX number was displayed as BCD on 40H and 41H.

Discussion

1. There are multiple sort algorithms available for our task. Due to the small number of inputs at hand, $O(N^2)$ sorting algorithms such as Bubble Sort, Insertion Sort and Selection Sort work well here. We use Bubble sort due to its ease of implementation but it's possible for other algorithms to offer a better constant for the time complexity. We have $O(Nlog(N))$ time complexity sorting algorithms as well, but we don't use them here because of implementational difficulty as well as larger overheads which dominate the time taken for small values of N such as 16.
   We employ the `CJNE` instruction with `JNC` to decide whether a swap needs to happen or not and use the `DJNZ` instruction to perform the `INNER_LOOP` and `OUTER_LOOP` both 16 times.
2. Our algorithm works only for the even case, but it can very easily be modified for the odd case as well by pre-calculating how many iterations of checking are needed. In most cases, less than $N/2$ are needed since our algorithm breaks at the first loss of equality. We use the `SUBB` and `JNZ` instruction to check equality of A and B where the two values from the maintained pointers are kept. `DJNZ` instruction is used to loop.
3. Here we display a BCD number on a space which supports only HEX representation. We do this by separating the digits of the 3 digit number abc. We store a directly in 40H since $a < 10$ implies it's the same in BCD and HEX. Now since we need to display bc in HEX, we store y in 41H where y is bc in HEX, which means $y = 16b + c$.

Summary

In this assignment, we explore basic control flow functionality in 8081 Microcontroller Assembly Language. We perform sorting in $O(N^2)$, palindrome check in $O(N)$ and BCD to HEX conversion in $O(1)$.