DEPARTMENT OF ELECTRONICS AND COMMUNICATION
ST JOSEPH ENGINEERING COLLEGE
VAMANJOOR MANGALURU

Program Assignment - 1

Course Title: Embedded System
Course Code: 18EC62
Academic Year: 2022-23
Semester: 6th
Submitted to: Ms. Vijayalaxmi H M
Name: Aaron Dsouza
USN: 4SO20EC003
Section: VI A

1)Write an ALP to check whether the given number is palindrome or not a) byte-wise using REV instructions.

```
   AREA palindrome, CODE, READONLY ENTRY

   ; Define the input number number
        EQU 12321

   ; Define variables
   ptr EQU 0x20000000 ; Starting address of the number rev_ptr EQU
   0x20000004       ; Starting address of the reversed number length
        EQU 5       ; Length of the number (including null character)

   ; Reset registers
   MOV R0, #0
   MOV R1, #0
   MOV R2, #0

   ; Copy the number to a temporary buffer
   LDR R1, =ptr MOV R2,
   #0

copy_loop
   LDRB R3, [R0, R1]
   STRB R3, [R0, R2]
   ADDS R2, #1

   CMP R3, #0
   BNE copy_loop

   ; Reverse the number byte-wise
   LDR R0, =length
   SUBS R2, R2, #2
   LDR R1, =rev_ptr

reverse_loop
   LDRB R3, [R0, R2]
   STRB R3, [R1], #-1
   SUBS R2, R2, #1
   CMP R2, #0
   BGE reverse_loop

   ; Compare the original and reversed numbers
   LDR R0, =length MOV R2, #0

compare_loop
   LDRB R3, [R0, R1]
   LDRB R4, [R0, R2]

   CMP R3, R4
   BNE not_palindrome

   ADDS R1, #1
   ADDS R2, #1
   SUBS R0, R0, #1

   CMP R0, #0
   BNE compare_loop

   ; The number is a palindrome B palindrome

not_palindrome
   ; The number is not a palindrome B not_palindrome

palindrome
   ; Palindrome message
```

```
    LDR R0, =palindrome_message B print_message

not_palindrome
    ; Not palindrome message
    LDR R0, =not_palindrome_message print_message
    ; Print the message
    LDR R1, =0x40004000
    BL print_string

end
    ; Infinite loop B end

    ; String printing subroutine
    print_string STRB R0, [R1] LDRB
    R3, [R0]
        CMP R3, #0
        BNE print_string BX LR

    palindrome_message    DCB "The number is a palindrome.",0
    not_palindrome_message DCB "The number is not a palindrome.",0

    END
```

b) Half word-wise using REV instructions.

```
AREA palindrome, CODE, READONLY


ENTRY
    ; Initialize registers
  LDR R0, =number ; Memory address of the number
   LDR R1, =start ; Start index of the number
```

```
    LDR R2, =end    ; End index of the number
    LDR R3, =1      ; Loop counter
    LDR R4, =0      ; Flag for palindrome check

loop
    CMP R1, R2      ; Compare start and end index
    BHS done      ; If start index >= end index, palindrome check completed
    ; Load half-word values from memory
    LDRH R5, [R0, R1] ; Load half-word from start index
    LDRH R6, [R0, R2] ; Load half-word from end index
    ; Reverse the half-word value REV R6,
    R6

    CMP R5, R6     ; Compare original and reversed half-words
    BNE not_palindrome ; If not equal, number is not a palindrome

   ADD R1, R1, #2 ; Increment start index by 2
         SUB R2, R2, #2     ; Decrement end index by 2
         B loop          ; Repeat the loop

not_palindrome
         MOV R4, #1          ; Set the palindrome flag to 1 (not
palindrome)          B done         ; Palindrome check completed

done
    ; Check the palindrome flag and display result
    CMP R4, #0
    BEQ palindrome_msg ; If palindrome flag is 0, display palindrome message

not_palindrome_msg
    ; Display "Not a Palindrome" message ; Code for
    displaying the message goes here
    ; ...
    ; End program execution B
    end_program

palindrome_msg
    ; Display "Palindrome" message ; Code for displaying
    the message goes here
    ; ...

end_program
    ; End program execution
    B end_program

    ; Data section
number
    ; Define the number to be checked here
    ; ...

start
   DCD 0          ; Initialize start index

end
    ; Initialize end index based on the size of the number
    ; ...

    END
```

2) Write an ALP to generate the Fibonacci series and save the series in the RAM AREA Fibonacci, CODE, READONLY

```
ENTRY
; Initialize registers
LDR R0, =fibonacci_start ; Memory address to store Fibonacci series
MOV R1, #0 ; First Fibonacci number
MOV R2, #1 ; Second Fibonacci number
LDR R3, =fibonacci_length ; Length of the Fibonacci series
MOV R4, #2
generate_fibonacci
; Loop counter
STR R1, [R0], #4 ; Store the Fibonacci number in memory
ADD R4, R4, #1 ; Increment loop counter
CMP R4, R3 ; Compare loop counter with the Fibonacci series length
BHS done
completed
; If loop counter >= Fibonacci series length, generation
ADD R1, R1, R2 ; Calculate the next Fibonacci number: current + previous
MOV R2, R1 ; Move the current Fibonacci number to the previous
number
B generate_fibonacci ; Repeat the loop
done
; End program execution
B done
```

```
; Data section
fibonacci_start
; Define the memory location to store the Fibonacci series
; .space directive can be used to allocate memory, e.g., .space 100
fibonacci_length
; Define the length of the Fibonacci series
; Set the desired length of the series, e.g., .word 10
END
memory
```