EE447 PRELIMINARY WORK #1

Simulation:

1 and 2)

1) Pseudo Code

```
adr ← first_address
coef ← 10^9
digit ← 0x30
counter ← 10
for (counter >0) {
    while (num>0) {
        num = num - coef
        digit = digit + 1
    }
    num = num + coef
    digit = digit - 1
    str digit → [adr]
    if (digit = 0 & adr = first_address)
        adr = adr
    else
        adr = adr+1
    digit = 0x30
    coef = coef/10
    counter = counter - 1
}
if (adr = first_address)
    adr = adr+1
0x040d → [adr] ; end
adr = first_address
Done
```

2) We pass the memory address where our input is stored and load the number into R4.

Figure 1. The corresponding pseudo-code for part 1 and answer of part 2.

I have written a subroutine, CONVRT, that converts an m-digit decimal number represented by n bits (n < 32) in register R4 into such a format that the ASCII codes of the digits of its decimal equivalent would listed in the memory starting from the location address of which is stored in register R5. When printed using OutStr, the printed number is to contain no leading 0s, that is, exactly m digits should be printed for an m-digit decimal number.

I have written a program that, in an infinite loop, waits for a user prompt (any key to be pressed) and prints the decimal equivalent of the number stored in 4 bytes starting from the memory location NUM. I noted that I may define NUM by using proper assembly directives. In this part, I am expected to use the subroutine you are written in Part-1.

Startup.s | OutStr.s | OutChar.s | InChar.s | CONVRT.s | prelim1.s

```
FIRST       EQU         0x20000000
            AREA        CONVRT, READONLY, CODE
            THUMB
            EXPORT      CONVRT

__CONVRT
            push    {r0-r4}
            ldr     r5,=FIRST           ;initial adress
            mov     r0,#0x30            ;ascii value for 0
            mov32   r1,#1000000000
            mov     r2,#10
            mov     r3,#0xa             ;max number of digits
check       subs    r4,r1
            addpl   r0,#1               ;increase the digit value if r4-r1 is positive
            bpl     check
            add     r4,r1               ;make r4 positive again
            str     r0,[r5]             ;store the digit in memory
            cmp     r0,#0x30            ;check if most significant digits are zero
            cmpeq   r5,#FIRST           ;check if most significant digits are zero

            addne   r5,#1               ;increase memory address by 1 byte
            mov     r0,#0x30            ;reset the digit
            udiv    r1,r2               ;prepare r1 for a lower decimal place
            subs    r3,#1               ;decrease digit counter
            bne     check
            cmp     r5,#FIRST
            addeq   r5,#1               ;if input is zero make sure that it is written in memory
            mov     r0,#0x040d
            str     r0,[r5]             ;put 'new line' and 'end of transmission' characters at the end
            ldr     r5,=FIRST           ;reset memory address
            POP     {R0-R4}
            bx      lr

            align
            end
```

Figure 2. Convert.s

Startup.s | OutStr.s | OutChar.s | InChar.s | CONVRT.s | prelim1.s | InNum.s

```
            AREA        InNum, READONLY, CODE
            THUMB
            EXPORT      InNum
            EXTERN      InChar

__InNum     push    {r1-r3}
            mov     r0,#0       ;input
            mov     r1,#0       ;digit counter
            mov     r2,#1       ;coefficient
            mov     r3,#10
loop        push    {lr}
            bl      InChar
            pop     {lr}
            cmp     r5,#0x20
            pushne  {r5}        ;push ascii value of the input characters untill 'space' character
            addne   r1,#1       ;count the number of digits
            bne     loop
dec2hex     pop     {r5}
            sub     r5,#0x30    ;ascii to decimal conversion
            mul     r5,r2       ;multiply with the coefficient of the digit
            add     r0,r5       ;sum value in hex
            mul     r2,r3       ;change the coefficient for higher decimal place
            subs    r1,#1
            bne     dec2hex
            pop     {r1-r3}
            bx      lr
            end
```

Figure 3. InNum.s

HEX 567

DEC 1383

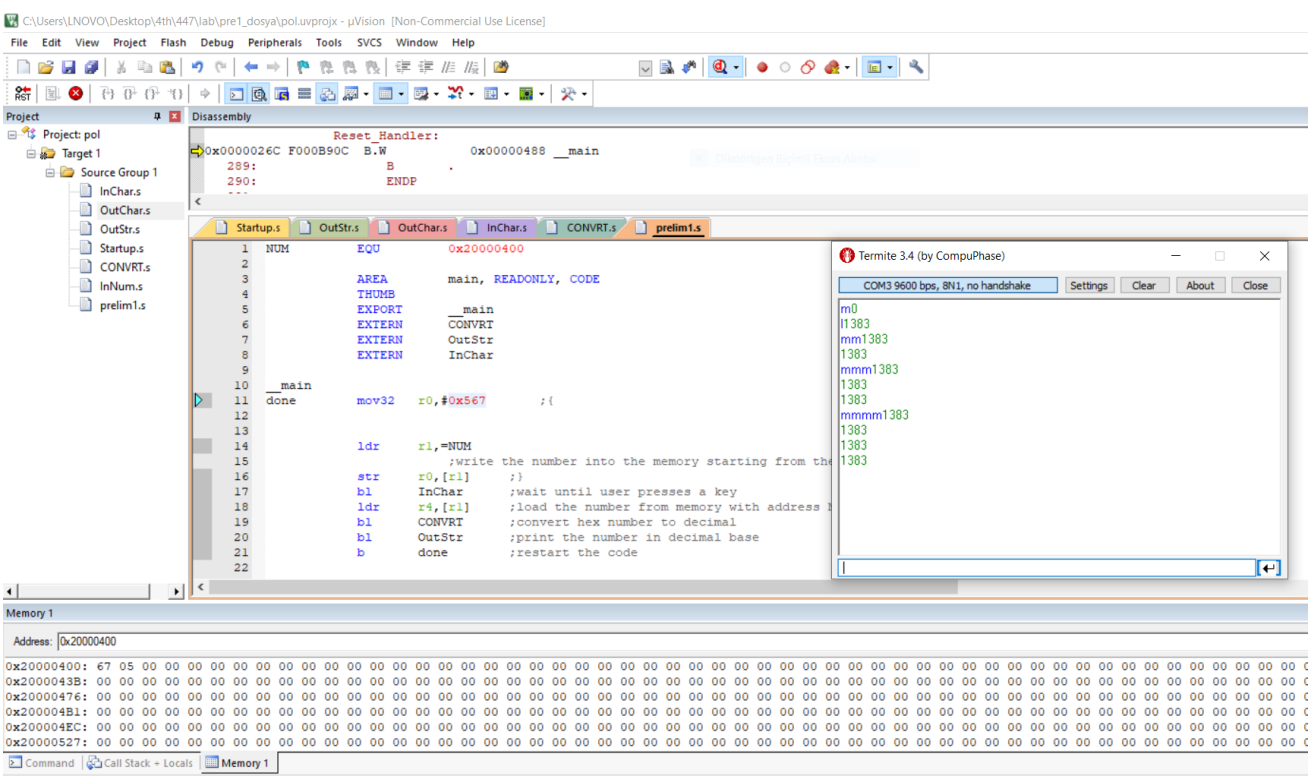


Figure 4. Results for 1ST and 2ND Part

3)

I have written a program for decimal number guessing using binary search method. The number is an integer in the range (0, 2n), i.e. 0 < number < 2n, where n < 32 and n is determined by a user-input. Then, the guessing phase is to be handled through a simple interface where the processor outputs its current guess in decimal base and calculate the next according to the user inputs, D standing for down, U standing for up, or C standing for correct. To fulfill the requirements given above, I included the subroutine CONVRT from the Part-1 in my main program as well as a new subroutine UPBND that updates the search boundaries after each guess. I draw a flowchart of the main algorithm leaving the subroutine parts as black boxes.

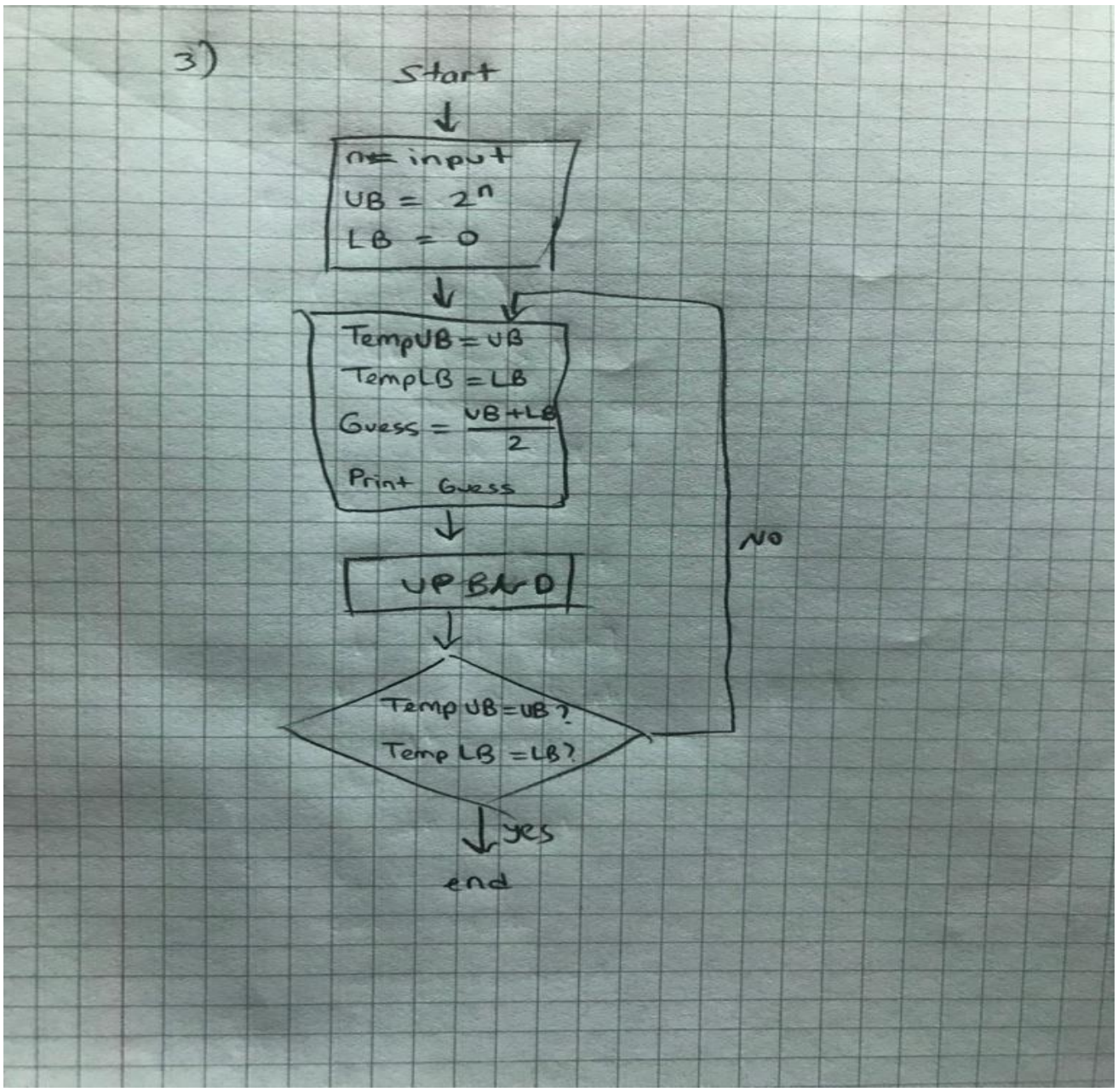
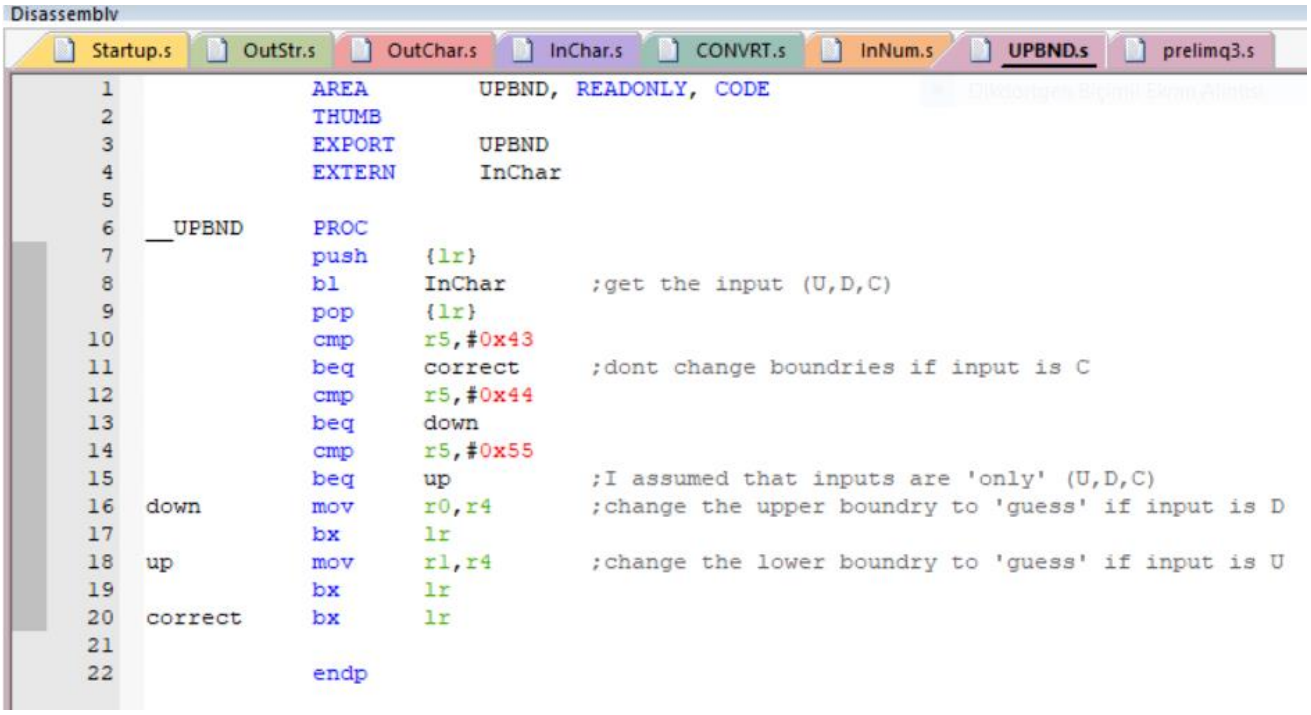


Figure 5. The corresponding pseudo-code for part 3

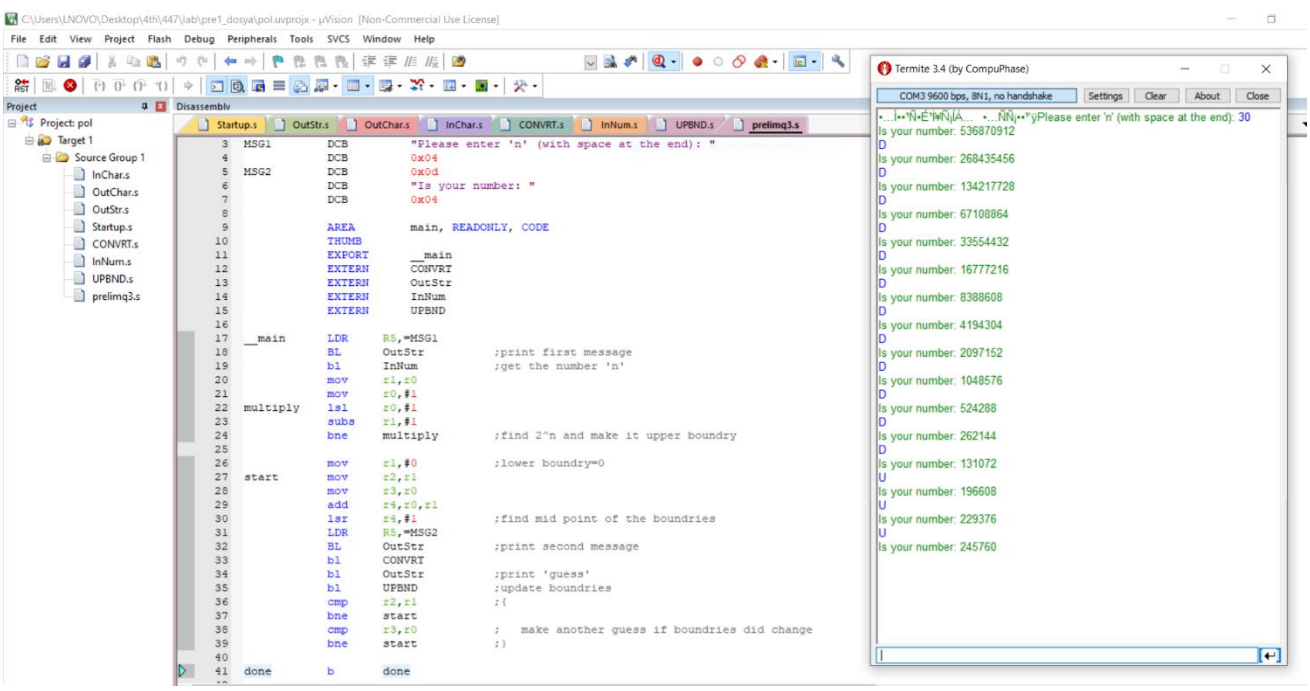


```
                AREA        UPBND, READONLY, CODE
                THUMB
                EXPORT      UPBND
                EXTERN      InChar

__UPBND         PROC
                push    {lr}
                bl      InChar      ;get the input (U,D,C)
                pop     {lr}
                cmp     r5,#0x43
                beq     correct     ;dont change boundries if input is C
                cmp     r5,#0x44
                beq     down
                cmp     r5,#0x55
                beq     up          ;I assumed that inputs are 'only' (U,D,C)
down            mov     r0,r4       ;change the upper boundry to 'guess' if input is D
                bx      lr
up              mov     r1,r4       ;change the lower boundry to 'guess' if input is U
                bx      lr
correct         bx      lr

                endp
```

Figure 6. UPBND.s



```
MSG1            DCB         "Please enter 'n' (with space at the end): "
                DCB         0x04
MSG2            DCB         0x0d
                DCB         "Is your number: "
                DCB         0x04

                AREA        main, READONLY, CODE
                THUMB
                EXPORT      __main
                EXTERN      CONVRT
                EXTERN      OutStr
                EXTERN      InNum
                EXTERN      UPBND

__main          LDR     R5,=MSG1
                BL      OutStr          ;print first message
                bl      InNum           ;get the number 'n'
                mov     r1,r0
                mov     r0,#1
multiply        lsl     r0,#1
                subs    r1,#1
                bne     multiply        ;find 2^n and make it upper boundry

                mov     r1,#0           ;lower boundry=0
start           mov     r2,r1
                mov     r3,r0
                add     r4,r0,r1
                lsr     r4,#1           ;find mid point of the boundries
                LDR     R5,=MSG2
                BL      OutStr          ;print second message
                bl      CONVRT
                bl      OutStr          ;print 'guess'
                bl      UPBND           ;update boundries
                cmp     r2,r1           ;{
                bne     start
                cmp     r3,r0           ;   make another guess if boundries did change
                bne     start           ;}

done            b       done
```

```
Please enter 'n' (with space at the end): 30
Is your number: 536870912
D
Is your number: 268435456
D
Is your number: 134217728
D
Is your number: 67108864
D
Is your number: 33554432
D
Is your number: 16777216
D
Is your number: 8388608
D
Is your number: 4194304
D
Is your number: 2097152
D
Is your number: 1048576
D
Is your number: 524288
D
Is your number: 262144
D
Is your number: 131072
U
Is your number: 196608
U
Is your number: 229376
U
Is your number: 245760
```

Figure 7. The code and result for 3rd part

4.

I have written a recursive program that computes first N elements of the mFibonacci Sequence which is a modified version of Fibonacci Sequence. The number N is to be an integer in the range (0,16) and is determined by a user-input.Then, the computation phase starts and the program computes the first N terms of a modified Fibonacci Sequence. After the computation finishes the sequence of numbers is going to be printed.

Disassemblv

OutStr.s | CONVRT.s | q4.s | Fibonacci.s

```
stradding   EQU     0x20000200
            AREA    Fibonacci, READONLY, CODE
            THUMB
            EXPORT  Fibonacci
            EXTERN  CONVRT
__Fibonacci
            push    {r1-r4}
            mov     r1,r5
            LDR     R2,=stradding
re          cmp     r1,#0x2
            BEQ     exit
            push    {lr}
            sub     r1,#1
            BL      re
            pop     {lr}
            LDR     R3,[R2,#-8]
            LDR     R4,[R2,#-4]
            LSL     R4,#1
            ADD     R1,R3,R4
            STR     R1,[R2],#4
            MOV     PC,LR
exit        MOV     R3,#0x0
            STR     r3,[r2],#0x4
            MOV     R3,#0x1
            STR     r3,[r2],#0x4
            MOV     pc,lr
            END
```

Figure 8. Fibonacci.s

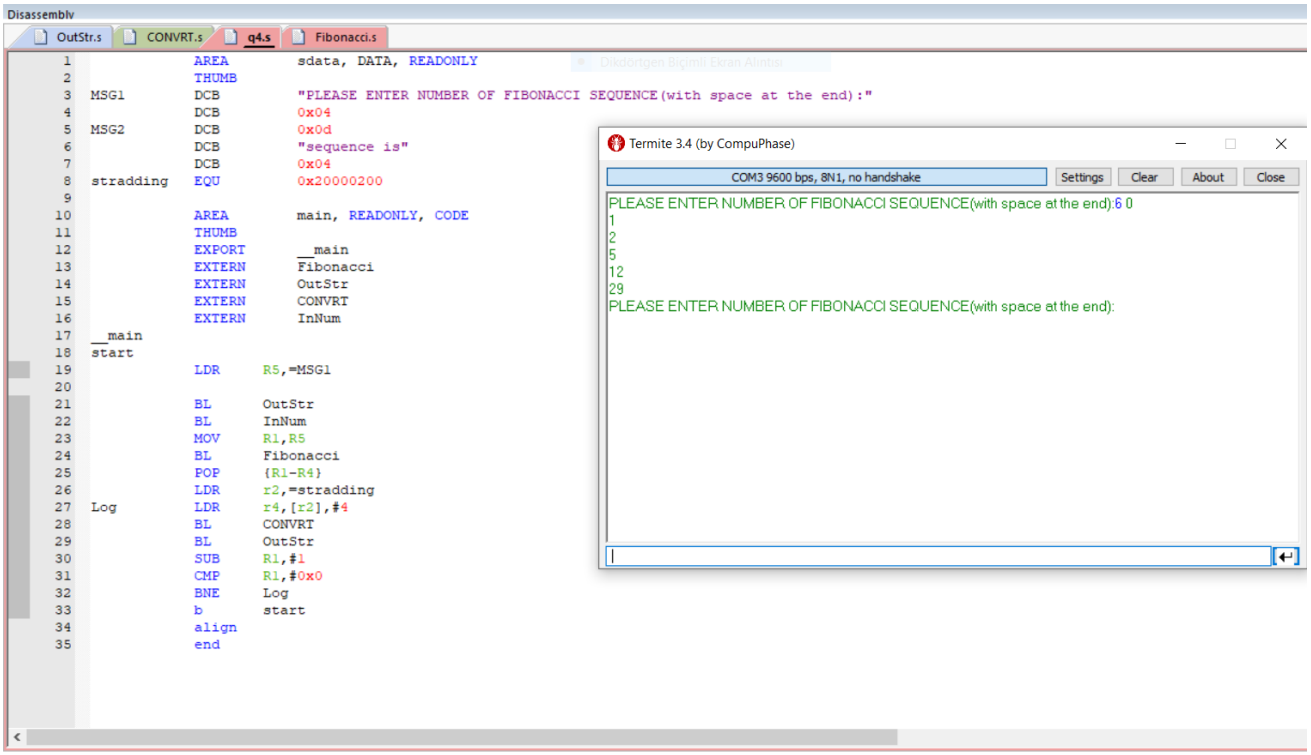


Figure 9. The code and result for 4th part