# EE447

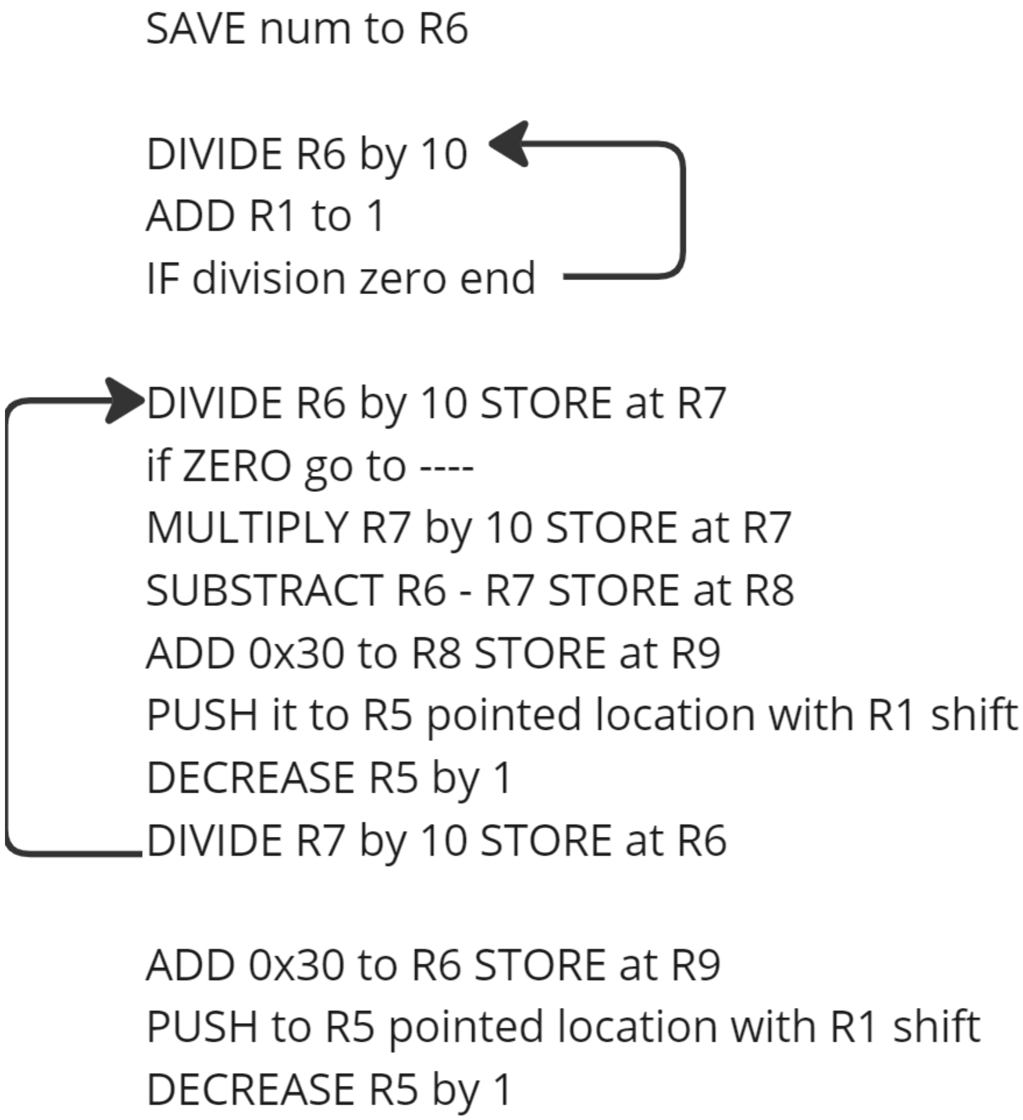
## LAB 1

## Preliminary Work

Işık Emir Altunkol - 2442408
Özgür Gülsuna - 2307668

## Part 1

### PSEUDO CODE



```
SAVE num to R6

DIVIDE R6 by 10
ADD R1 to 1
IF division zero end

DIVIDE R6 by 10 STORE at R7
if ZERO go to ----
MULTIPLY R7 by 10 STORE at R7
SUBSTRACT R6 - R7 STORE at R8
ADD 0x30 to R8 STORE at R9
PUSH it to R5 pointed location with R1 shift
DECREASE R5 by 1
DIVIDE R7 by 10 STORE at R6

ADD 0x30 to R6 STORE at R9
PUSH to R5 pointed location with R1 shift
DECREASE R5 by 1
```

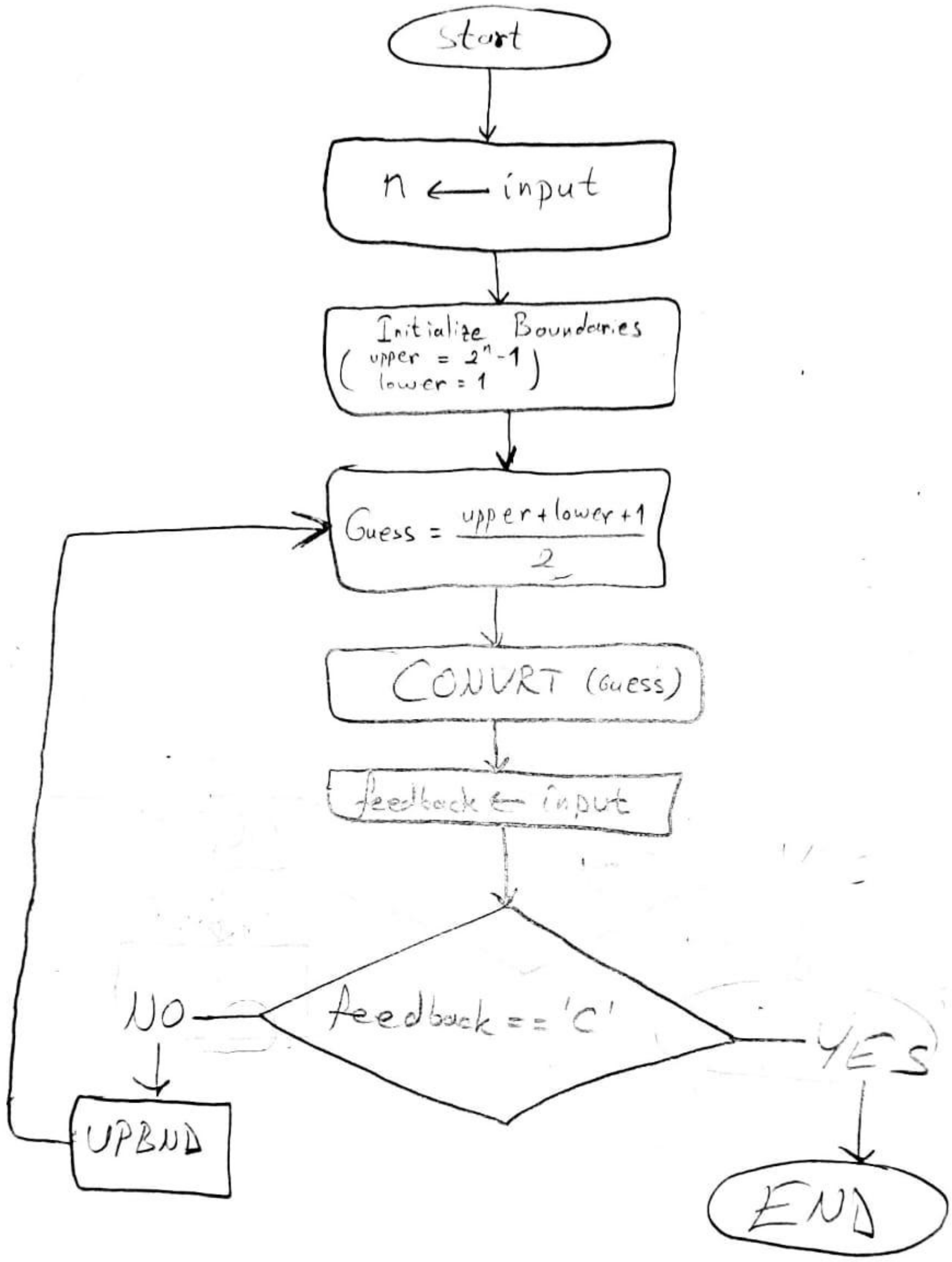
```
;***************************************************************

;***************************************************************
;SYMBOL     DIRECTIVE   VALUE           COMMENT
;***************************************************************
; Program section
;***************************************************************
;LABEL      DIRECTIVE   VALUE           COMMENT
            AREA        main, READONLY, CODE
            THUMB
            EXPORT      __main
            EXTERN      CONVRT
__main      PROC
            LDR         R4, =4294967295
            LDR         R5, =0x20000000
            BL          CONVRT

done        B           done
            ENDP
;***************************************************************
; End of the program section
;***************************************************************
;LABEL      DIRECTIVE   VALUE           COMMENT
            ALIGN
            END
```

```
;***************************************************************
;MODULUS
;***************************************************************
;SYMBOL     DIRECTIVE   VALUE           COMMENT
EOL         EQU         0x04
;***************************************************************
; Program section
;***************************************************************
;LABEL      DIRECTIVE   VALUE           COMMENT
            AREA        main, READONLY, CODE
            THUMB
            EXPORT      CONVRT
            EXTERN      OutStr

CONVRT      PROC
            PUSH        {R0-R9}         ; store current register conditions
            PUSH        {LR}            ; store current LR
            MOV         R2, #10         ; constant for division by 10
            MOV         R1, #0          ; R1 init, stores digit number
            MOV         R6, R4          ; store R4 at R6 to be processed

digit       UDIV        R7, R6, R2      ; divide the number by ten
            ADD         R1, #1          ; increment digit counter
            SUBS        R7, R7, #0      ; just subtract 0 to check zero(Z) flag
            BEQ         pre             ; if the digit is the last one the divison result in zero, prepare
            for next
            MOV         R6, R7          ; continue with one less digit
            B           digit           ;

pre         MOV         R6, R4          ; store R4 at R6 to be processed
            LDR         R10, =EOL       ; End Of Line EOL number load
            STRB        R10, [R5, R1]   ; store the ASCII at location pointed by [R5] and shifter by digit
            count
            BFC         R10, #0, #32    ; Clear R10 to be safe
            B           loop1           ; loop1

loop1       UDIV        R7, R6, R2      ; divide the number by ten
            SUBS        R7, R7, #0      ; just subtract 0 to check zero(Z) flag
            BEQ         last            ; if the digit is the last one the divison result in zero, loop2
            MUL         R7, R7, R2      ; again mutliply by 10 to restore the right amount of digits
            SUB         R8, R6, R7      ; store the last digit at R7
            ADD         R9, R8, #0x30   ; add this to convert single decimal number in ASCII representaiton
            SUB         R5, R5, #1      ; decrease the address pointer
            STRB        R9, [R5, R1]    ; store the ASCII at location pointed by [R5]
            UDIV        R6, R7, R2      ; move to the next digit
            B           loop1           ; loop1

last        ADD         R9, R6, #0x30   ; add this to convert single decimal number in ASCII representaiton
            SUB         R5, R5, #1      ; increase the address pointer
            STRB        R9, [R5, R1]    ; store the ASCII at location pointed by [R5] and shifter by digit
            count
            B           print           ;

print       ADD         R5, R5, R1      ; Shift R5 back to start to set it to starting adress for OutStr
            MOV         R0, R5          ; load register
            BL          OutStr          ; Print
            B           exit            ;

exit        POP         {LR}            ; pop the link register
            POP         {R0-R9}         ; pop registers back
            BX          LR

            ENDP
;***************************************************************
; End of the program section
;***************************************************************
;LABEL      DIRECTIVE   VALUE           COMMENT
            ALIGN
            END
```

# Part 2

```
;**************************************************************

;**************************************************************
;SYMBOL      DIRECTIVE   VALUE           COMMENT
NUM          EQU         0x20000000
;**************************************************************
; Program section
;**************************************************************
;LABEL       DIRECTIVE   VALUE           COMMENT
             AREA        main, READONLY, CODE
             THUMB
             EXPORT      __main
             EXTERN      CONVRT
             EXTERN      InChar

__main       PROC
             LDR         R5, =NUM
             LDR         R4, =447
goto         BL          InChar          ; wait for first input
             CMP         R0, #0x0a
             BEQ         goto
             BL          CONVRT

done         B           done
             ENDP
;**************************************************************
; End of the program section
;**************************************************************
;LABEL       DIRECTIVE   VALUE           COMMENT
             ALIGN
             END
```

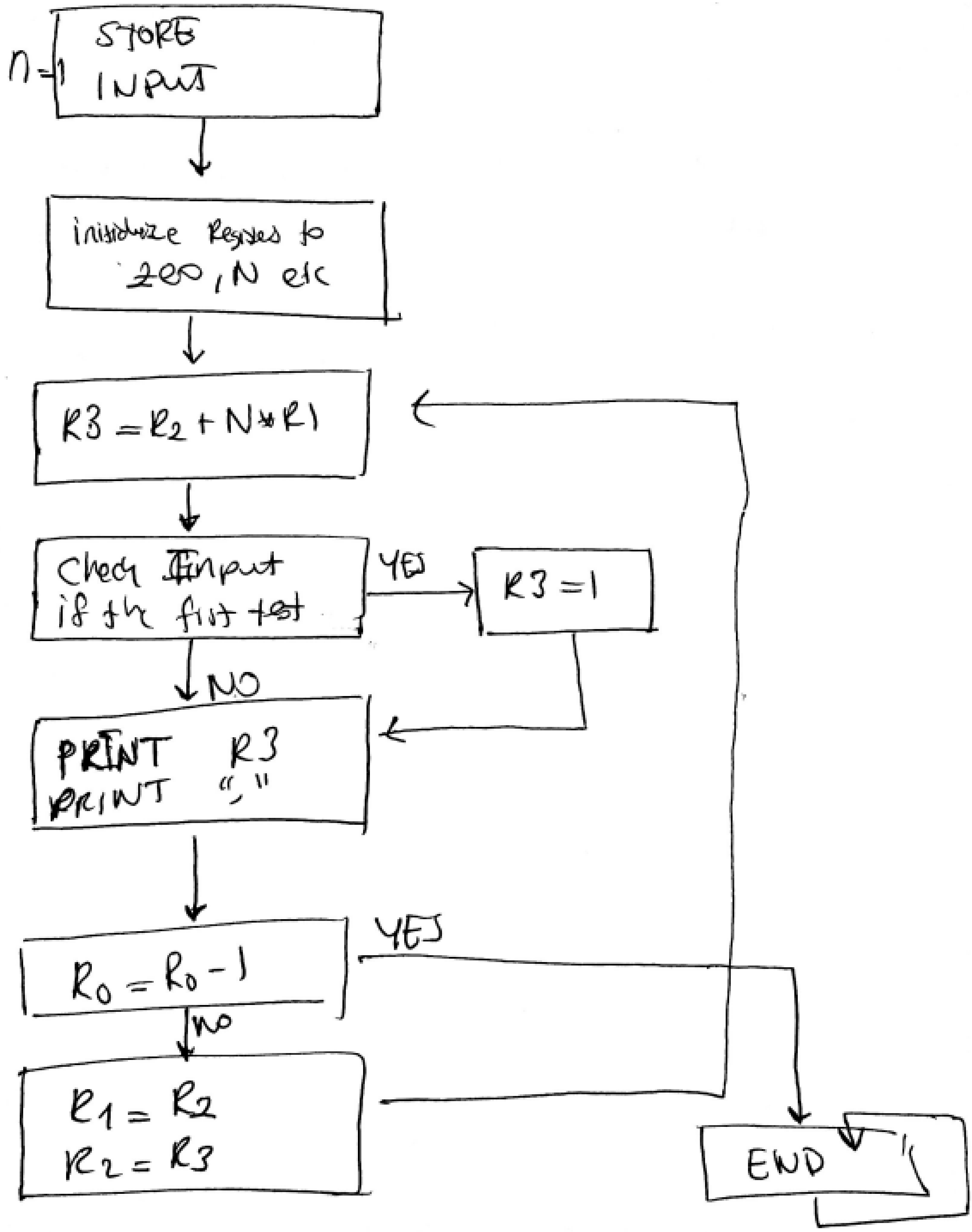
# Part 3


Start
n ← input
Initialize Boundaries
( upper = 2^n-1
lower = 1 )
Guess = upper+lower+1 / 2
CONVRT (Guess)
feedback ← input
feedback == 'C'
NO
YES
UPBND
END

```
; EQU Directives
; These directives do not allocate memory
;***************************************************************
;LABEL      DIRECTIVE   VALUE       COMMENT
STR_ADDR    EQU         0x20000400
;***************************************************************
; Directives - This Data Section is part of the code
; It is in the read only section  so values cannot be changed.
;***************************************************************
;LABEL      DIRECTIVE   VALUE       COMMENT
            AREA        sdata, DATA, READONLY
            THUMB
;***************************************************************
; Program section
;***************************************************************
;LABEL      DIRECTIVE   VALUE       COMMENT
            AREA        main, READONLY, CODE
            THUMB
            EXTERN      CONVRT      ; Reference external subroutine
            EXTERN      InChar      ; Reference external subroutine
;           EXTERN      UPBND       ; Reference external subroutine
            EXPORT      __main      ; Make available

__main
start       MOV         R0,#0
            BL          InChar
            SUB         R2, R0, #0x30
;           MOV         R0,#0
;           PUSH        {LR}
            BL          InChar
;           POP         {LR}
            MOV         R3, #10
            SUB         R0, R0, #0x30
            MUL         R2, R2, R3
            ADD         R2, R2, R0  ; R2 = n

            ; From now on, R3 is the lower boundary,
            ; R5 is the upper boundary

            MOV         R3, #1          ; Initialize the boundaries
            MOV         R5, #1
            LSL         R5, R2
            SUB         R5, R5, #1      ; The value is smaller than 2^n

guess       ADD         R4, R3, R5      ; Guess value is stored in R4 = (upper+lower+1)/2
            ADD         R4, #1
            LSR         R4, #1
            PUSH        {R5}            ; Upper boundary is stored in stack
            LDR         R5, =STR_ADDR
            BL          CONVRT          ; Writes the decimal digit sequence representing R4 to the address
            at R5
;           MOV         R0, R5
            POP         {R5}            ; Retrieve the upper boundary value from stack
;           BL          OutStr          ; Prints the value stored in adress R0
;           BL          InChar          ; Takes a single byte input from the user, stores it in R0
run         BL          InChar          ;
            CMP         R0, #0x0a       ;
            BEQ         run
            CMP         R0, #0x43       ; If input is 'C', terminate the program
            BEQ         idle
            BL          UPBND           ; Changes lower boundary R3 and upper boundary R5 according to the
            input stored in R0

; ---------------------------------------------------

UPBND       CMP         R0, #0x44       ; If the input is 'D', update upper boundary
            BNE         not_D
            SUB         R5, R4, #1      ; upper = guess - 1
            BL          guess           ; Guess again
not_D       CMP         R0, #0x55       ; If the input is 'U', update lower boundary
```

```
            BNE         idle            ; If the input is neither of them, there is an erroneaus input,
            terminate the program
            MOV         R3, R4          ; lower = guess
            BL          guess           ; Guess again

; ---------------------------------------------------

idle        B           idle            ; Idle loop



;*****************************************************************
; End of the program  section
;*****************************************************************
;LABEL      DIRECTIVE       VALUE                           COMMENT
            ALIGN
            END
```

# Part 4


n=1
STORE
INPUT
initialize Registers to
zero, N etc
$R_3 = R_2 + N*R_1$
Check ~~I~~input
if the first test
YES
$R_3 = 1$
NO
PRINT R3
PRINT ","
YES
$R_0 = R_0 - 1$
no
$R_1 = R_2$
$R_2 = R_3$
END

```
;****************************************************************
;SYMBOL     DIRECTIVE   VALUE           COMMENT
NUM         EQU         0x20000000      ; start adress
MLT         EQU         2               ; multiply by
;****************************************************************
; Program section
;****************************************************************
;LABEL      DIRECTIVE   VALUE           COMMENT
            AREA        main, READONLY, CODE
            THUMB
            EXPORT      __main
            EXTERN      CONVRT
            EXTERN      OutChar
            EXTERN      InChar          ; Reference external

__main      PROC
            LDR         R0, =2          ; input (0-16)
            LDR         R1, =0          ; F_{n-2}
            LDR         R2, =0          ; F_{n-1}
            LDR         R3, =0          ; F_{n}
            LDR         R4, =0          ; used in the convrt
            LDR         R5, =NUM        ; used in the convrt as pointer
            LDR         R6, =MLT        ; R4 to store the multiplication value, for flexibility
;----------------------------------------------------------------------
            PUSH        {R8, R9}        ; store R8, R9 to be used later
            MOV         R0, #0          ; clear R0
            BL          InChar          ; wait for first input
            SUB         R8, R0, #0x30   ; convert ASCII to decimal
            BL          InChar          ; save second input
            MOV         R9, #10         ; set R9 to 10
            SUB         R0, R0, #0x30   ; convert ASCII to decimal
            MUL         R8, R8, R9      ; multiply by 10 to create tens digit
            ADD         R8, R8, R0      ; add it to ones digit
            MOV         R0, R8          ; set it to R0, which is the input
            POP         {R8, R9}        ; restore R8, R9
;----------------------------------------------------------------------
            MOV         R7, R0          ; handle n<= case


recursion   MUL         R1, R6          ; 2*F_{n-2}
            ADD         R3, R2, R1      ; F_{n} = F_{n-1} + 2*F_{n-2}
            CMP         R0, R7          ; compare if its the first number
            BEQ         once

cont        MOV         R4, R3          ; set the output for convrt
            BL          CONVRT          ; convrt subroutine
            MOV         R1, R2          ; recurse move
            MOV         R2, R3          ; recurse move
            SUBS        R0, R0, #1      ; decrement the counter
            BEQ         done            ;
            PUSH        {R0}            ; store R0 to use later
            LDR         R0, =0x2C       ; the hex value for ASCII comma
            BL          OutChar         ; print comma
            POP         {R0}            ; bring back R0
            B           recursion

once        LDR         R3, =1          ; set output to 1 for n=1;
            B           cont

done        B           done
            ENDP
;****************************************************************
; End of the program section
;****************************************************************
;LABEL      DIRECTIVE   VALUE           COMMENT
            ALIGN
            END
```