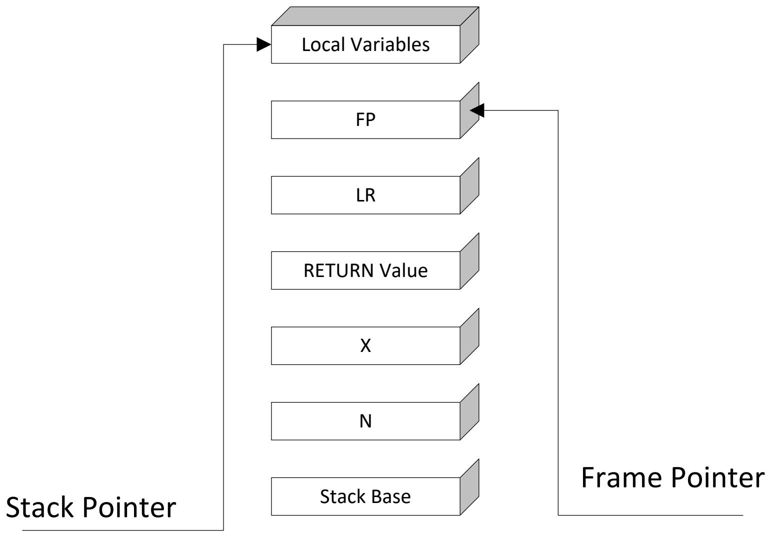
```
                AREA q_one, CODE, READONLY
                ENTRY

;-----------Constant Alias--------------
x               EQU 2                           ; the value of x
n               EQU 12                          ; the value of n
zero        EQU 0                           ; the value 0 as zero
one             EQU 1                           ; the value of 1 as one

;-----------Main Program----------------
                ADR sp, stack
                MOV r1, #x                      ; put the value of x in register 1
                MOV r2, #n                      ; put the value of n in register 2
                STMFD sp!, {r1-r2}          ; store the parameters and the result

                BL power                        ; call the power function
                LDR r0, [sp], #4        ; load the returned value into r0
                LDMFD sp!, {r1-r2}          ; load the parameters.
done        B   done                        ; endless loop to finish program

;-----------Power Function--------------
power       STMFD sp!, {fp, lr}     ; store frame pointer and link register in stack

                MOV fp, sp                      ; frame pointer at the bottom
of the frame
                SUB sp, sp, #4              ; create the stack frame
                LDR r2, [fp, #12]           ; load the value of n into r2
            ;---if n == 0 ------
                CMP r2, #zero               ; compare n with 0
                MOVEQ r2, #one                              ; if n = 0, make return value 1

                BEQ return                                  ; branch to return
            ;--- if n & 1 odd---
                TST r2, #one                        ; test if the last bit is 1 (odd number)
                LDR r1, [fp, #8]            ; get the x value from parameters
                BEQ case3                                   ; if even, go to case3
                SUBNE r2, r2, #one                  ; subtract 1 from n if it is odd
                STMFDNE sp!, {r1-r2}        ; store the parameters and the result
                BLNE power                                  ; call power again
                LDR r0, [sp], #4            ; get the value returned
                LDMFD sp!, {r1-r2}                  ; load the parameters.
                MUL r2, r1, r0                      ; multiply returned value by x
                B return                                    ; branches to return
            ;--- else if even---
case3       ASR r2, #one                    ; if even, shift n right by 1 (divide by 2)
                STMFD sp!, {r1-r2}                  ; store the parameters of current frame
                BL power                                    ; make a recursive call
                LDR r0, [sp], #4            ; load the returned value into r0
                LDMFD sp!, {r1-r2}                  ; load the parameters of current call
                MOV r2, r0                                  ; make a copy of the returned
value
                MUL r2, r0, r2                      ; multiply the returned value (y) by (y)
                B return                                    ; branch to return


            ;--- return portion-
return      LDR r0, [sp]                    ; get return value
                ADD sp, sp, #4                      ; clean up stack frame
                LDMFD sp!, {fp, lr}                 ; restore frame pointer and link register
from stack
```

```
                ADD r0, r0, r2                  ; add previous return value to current
                STR r0, [sp, #-4]!              ; Stores the return value on the stack
above the parameters.
                BX lr                               ; return to place function was
called from
;---------------------------------------


;-----------Data Area-------------------
                AREA q_one, DATA, READWRITE
                SPACE 0x1                           ; used to fix a padding problem
                SPACE 0x8F                          ; space allocated for stack

stack       DCD 0x00                        ; stack
                END
```



This is a full descending stack.

Size of n: stack frames needed. {0:1} {1:2} {2:3} {3:4} {4:4} {5:5} {6:5} {7:6} {8:5} {9:6} {10:6} {11:7} {12:6}