```
1           AREA power, CODE, READONLY
2           ENTRY
3   Main    ADR sp, Stack           ;Getting the address of the base of the stack. In this application a full ascending stack will be used
4           MOV r1, #5              ;r0 will represent the base of the exponential function
5           MOV r2, #12             ;r1 will represent the exponent of the exponetial function
6           STMFD sp!, {r1,r2}      ;Store both the base and exponent on the stack to be used as parameters
7           BL Power                ;Call the function power to calculate the base to the power of the exponent specified
8
9           LDR r3, [sp], #4        ;Store the answer returned by the function in the r2
10          STR r3, result          ;Storing the answer in variable Answer
11          LDMFD sp!, {r1,r2}      ;Pop the parameters, we dont need them so we can ignore their values
12  Loop    B Loop                  ;Infinite loop, indicating that the program is finsihed executing
13
14  Power   SUB sp, sp, #4          ;Reserving the location of the return value
15          STMFD sp!, {fp, lr}     ;Store the return address and previous frame pointer on the stack so we can retrive it later
16          MOV fp, sp              ;Move the frame pointer to the bottom of the current activaion frame
17          SUB sp, sp, #4          ;Advance the stack pointer to so it is in its proper location (pointing at the top of the stack)
18          LDR r1, [fp, #12]       ;Get the first parameter (base) passed into the fucntion and store it in r1
19          LDR r2, [fp, #16]       ;Get the second paramter (exponent) passed into the function and store it in r2
20
21          CMP r2, #0              ;Check to see if the base case is reached: when the exponent is equal to 0
22          MOVEQ r2, #1            ;If it is move the number one to register 6 so we can return it
23          BEQ Bottom              ;Then finally the method is done so go to the end of the method
24
25          TST r2, #2_1            ;If we get here that means we have not reached base case. So check if the exponent is odd
26          BEQ Else                ;If the result of the above line is 0, then the value of the exponent is even so go to the else
27          SUB r2, r2, #1          ;Otherwise it is an odd number. So lets subtract one from the current exponent
28          STMFD sp!, {r1,r2}      ;Then place the base and new epxonent value on the stack as paramters for the next method call
29          BL Power                ;And finally call the function the Power with paramters base and exponent
30          ADD sp, sp, #12         ;Pop the parameters from the call
31          LDR r1, [fp, #12]       ;recovering the original value of r1
32          LDR r2, [fp, #16]       ;recovering the original value of r2
33          LDR r3, [fp, #-16]      ;Get the returned value from the call
34          MUL r2, r1, r3          ;Use the returned value and multiply it by the orginal base
35          B Bottom                ;Then we are done the method, so go to the end of the method
36
37  Else    ASR r2, #1              ;Else the number is even, so divide the exponent by 2 by shifting it to the right
38          STMFD sp!, {r1,r2}      ;Then store the the base and new exponent on the stack as paramters
39          BL Power                ;Then finally call the power function with paramters base and exponent
40          ADD sp, sp, #12         ;Pop the paramters from the call
41          LDR r1, [fp, #12]       ;recovering the original value of r1
42          LDR r2, [fp, #16]       ;recovering the original value of r2
43          LDR r3, [fp, #-16]      ;Get the returned value from the call and store it in the local variable y
44          STR r3, [fp, #-4]       ;Update the local variable y in the stack frame
45          MUL r2, r3, r3          ;Multiply y by y
46
47  Bottom  STR r2, [fp, #8]        ;Then store the answer of the previouis calculation on the stack as a return value
48          ADD sp, sp, #4          ;Adjust stack pointer so it is pointing at the stop of the stack
49          LDMFD sp!,{fp,pc}       ;restore frame pointer and return
50
51          space 256               ;making room for the stack
52  Stack   DCD 0x0000              ;start of the stack
53  result  DCD 0x0000              ;variable for storing the answer
54          END                     ;By: Jakob Wanger, Student #: 250950022
```