Code

NOTE: Memory map for “0xC8,0x1E0” should be changed to read and write   
  
 AREA question1, CODE, READONLY ENTRY

ADR sp, regs ; define stack for registers STMED sp!, {r1 - r3} ; save registers r1 - r3 ADR sp, stack ; define stack SUB fp, sp, #4 ; make fp point to stack base MOV r0, #3 ; choose x value STR r0, [sp, #-4]! ; push x on stack MOV r0, #11 ; choose n value STR r0, [sp, #-4]! ; push n on stack BL power ; call power function LDR r0, [sp], #4 ; pop result STR r0, result ; store result in memory ADR sp, regs ; get back stack of saved registers LDMFA sp!, {r1-r3} ; pop saved values into registers  
loop B loop

power STMFD sp!, {fp, lr} ; push frame pointer and link register MOV fp, sp ; make fp point to frame base SUB sp, sp, #8 ; create stack frame (2 words) LDR r1, [fp,#12] ; get x parameter LDR r2, [fp,#8] ; get n parameter CMP r2, #0 ; check if n = 0

; if n = 0 then return 1 ADDEQ sp, sp, #8 ; clean up stack frame LDREQ fp, [sp], #4 ; restore frame pointer to previous frame LDREQ lr, [sp], #4 ; load branched address in lr MOVEQ r0, #1 ; store 1 in register STREQ r0, [sp, #-4]! ; push 1 on top of parameters BXEQ lr ; return

AND r3, r2, #2\_1 ; AND n with 1 to see if it is odd CMP r3, #1 ; check if n is odd

;if n is odd then return x \* power(x, n-1) STREQ r1, [fp, #-4] ; push x on frame SUBEQ r2, r2, #1 ; subtract 1 from n STREQ r2, [fp, #-8] ; push n on frame BLEQ power ; call power(x, n-1) LDREQ r0, [sp], #4 ; pop returned value MULEQ r0, r1, r0 ; x \* power(x, n-1) ADDEQ sp, sp, #8 ; clean up stack frame LDREQ fp, [sp], #4 ; restore frame pointer to previous frame LDREQ lr, [sp], #4 ; load branched address in lr STREQ r0, [sp, #-4]! ; push x \* power(x, n-1) on stack BXEQ lr ; return

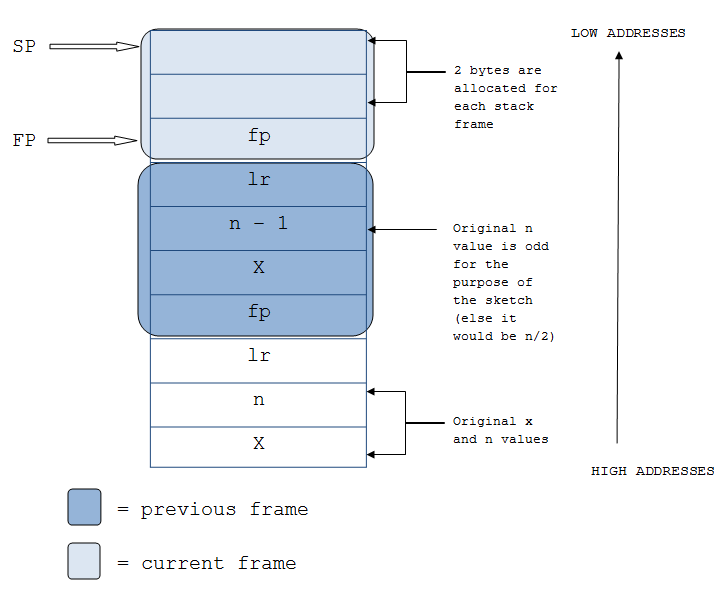
;else STRNE r1, [fp, #-4] ; push x on frame MOVNE r2, r2, ASR#1 ; divide n by 2 STRNE r2, [fp, #-8] ; push n on frame BLNE power ; call power(x, n/2) LDR r0, [sp], #4 ; pop returned value MUL r3, r0, r0 ; power(x, n/2) \* power(x, n/2) ADD sp, sp, #8 ; clean up stack frame LDR fp, [sp], #4 ; restore frame pointer to previous frame LDR lr, [sp], #4 ; load branched address in lr STR r3, [sp, #-4]! ; push power(x, n/2) \* power(x, n/2) on stack BX lr ; return

AREA question1, DATA, READWRITE

SPACE 0xFF ; stack body (FD)   
stack DCD 0x00 ; TOS  
result DCD 0x00 ; final result SPACE 0x10 ; stack for original values of register  
regs DCD 0x00 ; TOS END

Stack Frame Structure/Content

Stack type: full descending



Answer to Question

Assuming that when the parameters are first put on the stack at the beginning, a frame is created (rather than the first frame being made after the function is called), then the following are the number of frames needed for specific n values:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Value of n | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| # of frames | 2 | 3 | 4 | 5 | 5 | 6 | 6 | 7 | 6 | 7 | 7 | 8 | 7 |

Otherwise if we assume that the first frame is created after the function is called for the first time, then the number of frames is one less than it is written in the charts (e.g. if n = 0 then # frames = 1, if n = 1, then # frames = 2, and so on).