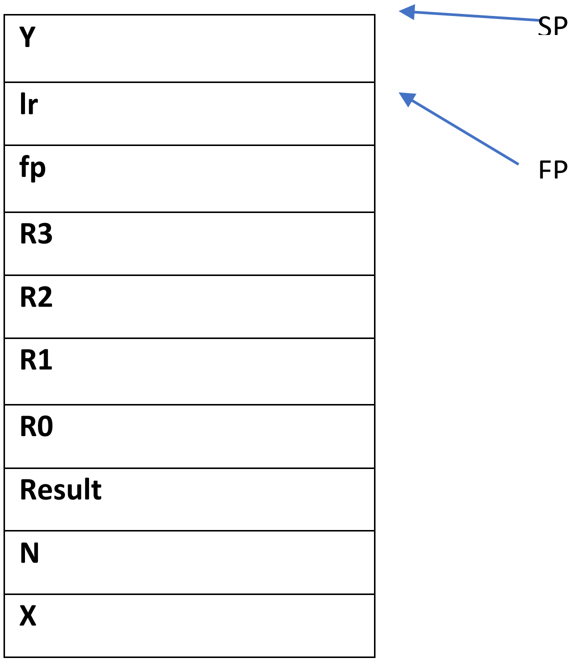
Sam Silverstein
250934990
ssilve29

Report 3 (Drawing the Stack Frame)

STACK FRAME



Y
lr
fp
R3
R2
R1
R0
Result
N
X

This is an instance of a stack frame, this is created from each call of the subroutine, and Result will be empty, until it reaches the base case

In addition, N will either be decremented by 1, or halved by each recursive call

Value of N	# Recursive Calls
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

```
                ENTRY
;---------Main function, where parameters are loaded, stack is created------------------------------
main    ADR             sp, stack               ;point stack pointer sp to the beginning of the stack
                LDR     r0, x                   ;prepare parameter x to be pushed on stack
                LDR     r1, n                   ;Prepare parameter n to be pushed on the stack
                STR     r0,[sp,#-4]!    ;push paramter x on the stack
                STR     r1,[sp,#-4]!    ;Push parameter y on the stack
                SUB     sp,sp,#4                ;increase the stack pointer 4 bytes, to store the result

                BL              Func1

                LDR     r0,[sp],#4              ;load the result in r0, then pop it off the stack
                ADD     sp,sp,#4                ;also remove the parameter from the stack

                ADR     r1,result               ;r1 to store the address of the return variable
                STR     r0,[r1]                 ;store the result in register r0, which is the result variable

                B               loop                    ;endless loop, represents end of program


;-------Recursive function power (labeled Func1)----------------------------
Func1   STMFD   sp!,{r0-r3,fp,lr}               ;to begin function, push registers to be used, along with fp, lr
                MOV     fp,sp                                   ;set up the frame pointer for this call
                LDR     r0,[fp,#32]                             ;load parameter x into r0
                LDR     r1,[fp,#28]                     ;load paramter n into r1

                ;this section represents the base case, where n = 0
                CMP     r1,#0                                   ;test to see if n = 0
                MOVEQ   r2,#1                           ;move #1 into register r2,and now register r2 is the result register
                STREQ   r2,[fp,#24]                     ;store the result register r2 into result section of the stack
                BEQ             return

                ;This section represents the case where n is odd
oddTst  TST     r1,#1                           ;check to see if r1 (n) is odd
                BEQ             evenTst                                 ;if it is even, then just branch to the test for even
                SUBNE   r1,#1                                   ;if it is not equal, then subtract 1 from n, and begin new frame
                STRNE   r0,[sp,#-4]!                    ;push x onto the stack
                STRNE   r1,[sp,#-4]!                    ;push the new n onto the stack
                SUBNE   sp, sp, #4                              ;increase stack pointer 4 bytes to make room for the result
                BLNE    Func1                                   ;branch back to the function
                LDR             r3,[fp,#-12]                    ;retrieve the result from the previous call
                MUL             r2,r3,r0                                ;multiply x by what was previously returned, and store it in the rsult register
                B               return                                  ;branch unconditionally to return

                ;this section represents the case where te test to see if n is even
evenTst LSR     r1,#1                           ;if it is even (by knowing its not odd) we use logical shift right by one to divide it by 2
                STR     r0,[sp,#-4]!                    ;push x onto the stack
                STR     r1,[sp,#-4]!                    ;push the updated n on the stack
                SUB     sp,sp,#4                                ;increase stack pointer 4 bytes to make room for the result variable
                BL      Func1                                   ;branch with link back to the beginnning of fuNC1
                LDR             r3,[fp,#-12]                    ;retrieve the result from the previous call
                MUL             r2,r3,r3                                ;multiply the preious returned result, and store it in register r2
                B               return


                ;this section represents where we restroe the stack frame, by storing the results from the previous call into the appropriate result space in each stack frame
return  STR             r2,[fp,#24]                             ;store r2, which was the result we calculated based on the case, and store it in the space for the result in the
stack
                MOV     sp,fp                                   ;Collaps the space form the function call
                LDMFD   sp!,{r0-r3,fp,pc}               ;load all of the regesters back, and return back to the main function



;------Function is now finished----------------------------------------------------------------

loop    B               loop





;------Declaring Memory-------------------------------------------------------------------------
                AREA power, DATA, READWRITE

x               DCD     0x2
n               DCD     0x4
                SPACE   0xFF
stack   DCD     0x0000
result  DCD     0x00    ;where we will store the final result
```