CS2208 Assignment 5

“Somewhere Over The Program Stack"

Robert Simoes, 205744623

APRIL 3rd 2018

INTRODUCTION

It’s long been understood through George Lucas’ magnum opus Star Wars trilogy (and there more disappointing sister episodes), that the wise mentoring character Yoda once said:

“Do or do not, there is no try”

Although Yoda was not inherently a programmer (Jedi Masters have more “important” and likely boring things to do probably), this adage is very relevant in Computer Science. One should commit oneself to their code and craft, win or lose.

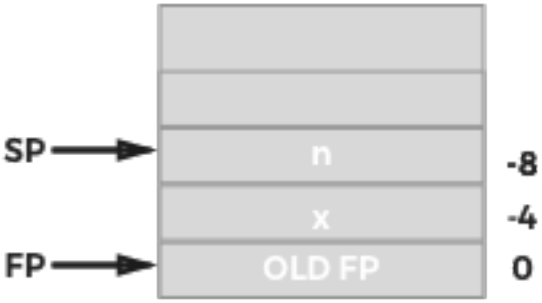
QUESTION 1

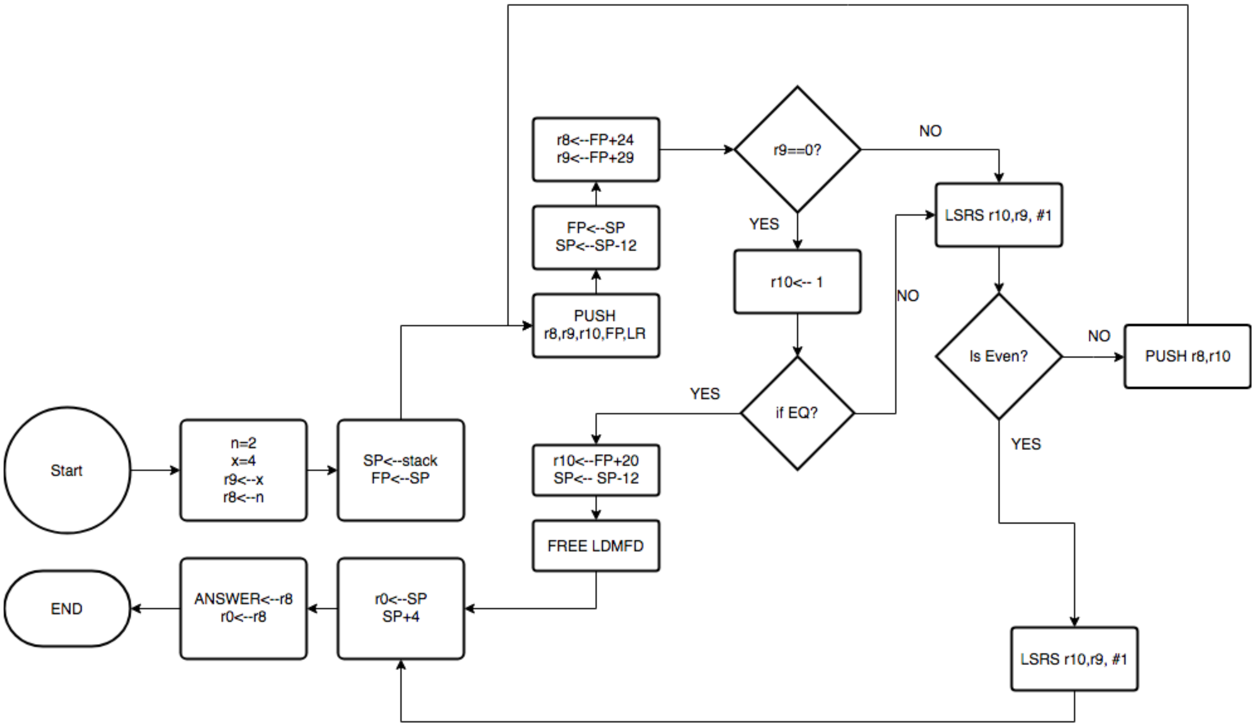
This assignment uses recursion to calculate the power of a value x and n programmed into the ARM assembly code. At current it is not entirely functional, with issues in the memory map and how jumping and recursion is accomplished. This is likely an issue with some of the stack calculations I made, and I continue to get memory and access violations.

I've constructed the partial flow chart map below of the logic I attempted to use.

Note: Please ensure you activate the memory map via Debug menu option and Memory Map *while* in debug mode. Set to Read, Write, Execute for the presented range.

n=0, 1 stack frame
n=1, 2 stack frame
n=2, 3 stack frame
n=3, 4 stack frame
n=4, 4 stack frame
n=5, 5 stack frame
n=6, 5 stack frame
n=7, 6 stack frame
n=8, 5 stack frame,
n=9, 6 stack frame
n=10, 6 stack frame
n=11, 7 stack frame
n=12, 8 stack frame





```
                AREA POWER, CODE, READONLY
            ENTRY
n               EQU 2                                  ; Define n=2
x               EQU 4                            ; Define x=4

MAIN
                MOV r9, #x                  ; [r9] <-- x
                MOV r8, #n                             ; [r8] <-- n

                ADR SP, STACK               ; Let SP= STACK
                MOV FP, SP      ; Prepare framepointer for stack
parameters


                STMFD SP!, {r9,
r8}                                    ; PUSH r9, r8
                SUB SP, #4                  ; Move call frame down

                BL REC_POWER

                LDR r0, [SP], #4                                    ; r0
<-- [SP]
                ADD SP,
#8                                               ; Reclaim stack space
                ADR r8,
ANSWER                                           ; r8 <-- &ANSWER
                STR r0,
[r8]                                        ; Save r0 to r8

ENDMAIN         B ENDMAIN                   ; Ending loop

REC_POWER
                STMFD SP!, {r8-r10,FP,LR}   ; PUSH r8,r9,r10, FP, LR
                MOV FP, SP             ; Let FP = SP
                SUB SP, #12
; Alloc stack
                LDR r8, [FP, #24]                ; r8 <-- x
                LDR r9, [FP, #28]           ; r9 <-- n
                CMP r9, #0             ; Base Case: n=0
                MOVEQ r10, #1               ; Add 1 and finish
                BEQ EXIT_FUNCTION                ; Exit

                LSRS r10,r9,#1              ; Bitwise AND
                BCC NOT_EVEN    ; n%2 ==0? Check Carry, ODD?
                SUB r1, #1      ; ODD -> CONVERT EVEN

                STMFD FP, {r8,r9} ; Prepare next recursive call
                BL REC_POWER                ; Resurge
                LDR r9,[FP,#-12]       ; Load R9 <-- FP-12
                MUL r10,r8,r9
                B EXIT_FUNCTION

NOT_EVEN
                STMFD FP,{r8,r10}      ; PUSH X,n --> STACK
                BL REC_POWER                ; Recurse
                LDR r9,[FP,#-12]      ; Load r1<-- FP-12
                MUL r10,r9,r9    ; r10 <-- (r9)^2

EXIT_FUNCTION
                STR r10, [FP,#20]     ; return value PUSH --> r2
                ADD SP, #12           ; Clear FP in Stack
                LDMFD SP!, {r8-r10, FP,PC}       ; Set registers and
return

                AREA POWER, DATA, READWRITE
ANSWER          DCD 0x00
                SPACE 0xFF       ; Requires space for stack pointer
STACK      DCD 0x00                   ; Use FD model, initial stack
position
                END
```