The University of Western Ontario

Faculty of Computer Science

CS 2208 – Computer Organisation and Architecture

Assignment 05

by

Mohammad Ali Sarfraz: 250860782

Lab Section: 006, Studio instructor: Prof. Mahmoud El-Sakka

Date Submitted: April 4, 2018

CODE

```
    AREA power, CODE, READWRITE
        ENTRY

X_value EQU 0x02        ;Declare the value of X needed in the program computation
N_value EQU 0x0A        ;Declare the value of N needed in the program computation

;-------------------------------------------------------------------------------------------------------------------------
MAIN
     LDR r13,=stack     ;Initialize the stack by using the reserved space declared in the memory
     STMFD r13!,{r0-r5} ;Initialize the registers needed by the function to compute the answer onto the stack
     STR fp,[r13,#-4]!  ;Store the frame pointer onto the stack
     MOV fp,r13         ;Allign frame pointer to the current frame
     LDR r1,=X_value    ;Load the address of where the value of X is stored
     LDR r0,=N_value    ;Load the address of where the value of N is stored

     STMFD r13!,{r0,r1} ;Initialize X and N for the current stack frame
     STMFD r13!,{r0,r1} ;Initialize X and N as variables to be used on the stack
     SUB r13,r13,#4     ;Leave space in stack for the computed value
     BL POWER           ;Jump to subroutine and store the return address on the link register

     LDR r2,[fp,#-20]   ;Load the value returned from the subroutine into r2
     STR r2,result      ;Then save this value into the local variable called 'result' in memory
     MOV r13,fp         ;Clear off the stack frame
     ADD r13,r13,#4     ;Then clear the frame pointer off the stack
     LDMFD r13!,{r0-r5} ;Reassign the values previously stored in the registers before the function was called

DONE B DONE             ;Branch indefinitely to indicate the program is done

;-------------------------------------------------------------------------------------------------------------------------
;First we define the calling of the POWER subroutine

POWER
     STMFD r13!,{fp,r14} ;Store the return address and the frame pointer of the calling frame into the current frame
     MOV fp,r13          ;Update the frame pointer to the current stack frame
     LDR r0,[fp,#12]     ;Load the value of N from the calling frame
     LDR r1,[fp,#16]     ;Load the value of X from the calling frame
     STMFD r13!,{r0,r1}  ;Initialize N and X as local variables for the current stack frame
     CMP r0,#0x00        ;Check to see if the value of N is 0
     BEQ RETURN          ;If it is then branch to the return operation
     TST r0,#0x01        ;If it isnt then check if the value of N is odd
     ASREQ r0,r0,#1      ;If it is even then divide N by 2
     SUBNE r0,r0,#1      ;Else if it is odd then decrement N by 1

     STMFD r13!,{r0,r1}  ;Update the parameters for the next frame of calculations
     SUB r13,r13,#4      ;Leave space in the current stack frame for the returned computed value
     BL POWER            ;Recursively call the function with updated values for X and N

;-------------------------------------------------------------------------------------------------------------------------
;Next we define the returning operation for the POWER subroutine

RETURN
     LDR r0,[fp,#-8]     ;Load the current stack frame's value of N
     CMP r0,#0x00        ;Check to see if this value is equal to 0
     MOVEQ r4,#1         ;If it is equal to 0 then the function should return value of 1 (x^0 = 1)
     BEQ JUMP            ;Jump over the next set of instructions if it is

     LDR r2,[fp,#-20]    ;Else if N is not equal to 0 then load the return value from the calling function
     TST r0,#0x01        ;Check to see if the current value of N is odd
     LDRNE r3,[fp,#-12]  ;If it is an odd number then load the current stack frame's value of X
     MULNE r4,r3,r2      ;If the above statements are true then the answer is the value returned times X
     MULEQ r4,r2,r2      ;Else if it is an even number then the answer is the square of the value returned
JUMP
     STR r4,[fp,#8]      ;Store the returned value in the calling function
     MOV r13,fp          ;Clear off the current stack frame
     LDR fp,[fp]         ;Reset the frame pointer to the previous frame
     LDR r5,[r13,#4]!    ;Clear the frame pointer off the stack and also load the return address for this frame
     ADD r13,r13,#4      ;Completely clear off the stack frame by clearing off the return address
     MOV r15,r5          ;Branch back to the main function

;-------------------------------------------------------------------------------------------------------------------------
    AREA Variables, DATA, READWRITE

Storage SPACE 0x01*(24+20+28*10-8)  ;Allocate space for the stack = 4*6 for the registers + 20 for the main function's stack frame
                                    ;+ 28 for each of the 'n' stack frames - 8 for last stack frame
stack   DCD 0x00                    ;Initialize the top of the stack

result  SPACE 0x04                  ;Space to store result

        END
```

Stack Frame Structure

Legend:

Stack frame from previous call

Call Frame

Main Stack Frame

** The stack grows upwards when a new item is placed onto it, i.e the top of the stack is the lowest memory address and the bottom of the stack is the highest memory address.

Value of N
Value of X
Frame Pointer
Return Address
Space for Answer
Argument N
Argument X
Value of N
Value of X
Frame Pointer
Return Address
Space for Answer
Argument N
Argument X
Local Variable N
Local Variable X
Secondary Frame Pointer
Registers r0 – r5

Number of Stack Frames

We use the general rule that when n is an odd number then we make a function call with n – 1, and when n is an even number then we make a function call with n/2. Using this guideline, we can see that the number of stack frames needed for n = 1 to n = 12 are as follows;

Value of N	Number of Stack Frames
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