Lishan Huang
260777962
CS2208
Assignment 5

How many stack frames are needed to calculate xn , when n = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, and 12?
In hex,34 ,34 ,50 ,62, 62, 88 ,88 ,A4, A4, A4, A4 C0,C0

| |
|---|
| LR |
| PC |
| Odd push 1 even push 0 |
| The caller to push the parameter the remain n that need to be calculated |
| The caller to allocate memory inside the stack for the return recursion value |
| The caller to push the original parameter n on the stack |
| |

```
;Lishan Huang
;250777962
;cs2208 assignment4
                AREA power, CODE, READONLY
x               EQU 2                           ; initialize variable x for base
n               EQU 7                           ; initialize variable n for the exponent
Next    EQU 4           ; initialize variable next for the step the need to move to next block
RemainN EQU -0x1C         ; initialize the position need to move to store the remained exponent
need to be calculated
returnN EQU -0x14         ; initialize the position need to move to store the exponent already be
calculated
                ENTRY
                ADR     sp, stack               ; load the address of stack
                MOV     r0, #x          ; store  the value in x into r0
                MOV     r1, #n          ; store  the value in n into r1
                STR     r1,[sp],#Next           ;store the value of n into the stack
                ADD sp,sp,#Next         ;move the next block of the stack
                BL main                                 ;jump to main
                LDR r2,[sp],#Next         ;Load the register in the second block of stack
                ADD sp,sp,#Next         ;move to the pointer to the next block
                ADR r5,result                   ;store the address of output in r5
                STR r2,[r5]             ;store r2 in the address of r5

loop    B loop                          ;stop here
main    STMEA sp!,{r1,r2,r3,fp,lr}  ;store multiple registers by empty stack and growth
ascending
                MOV fp,sp                       ;store stack pointer in fp
                LDR r2,[fp,#RemainN]                    ;load the remain need to be calculated
                CMP r2,#0                               ;compare exponent and 0
                MOVEQ r2,#1                     ;if exponenet is 0 the store 1 in r2
                STREQ r2,[fp,#returnN]          ;store r2 in the number of n already calculated
                BLEQ back                               ;jump to back
                AND r3,r2,#1                    ;and r2 and r1 and store the result in r3
                CMP r3,#1                               ;compare r3 and #1
                BEQ     odd                                     ;if r3 equal 1 then jump to odd
                BNE     even                            ;else jump to even
odd             SUB r2,#1                               ;substract 1 from r2
                STR r2,[sp],#Next               ;store r2 to the next block of the stack
                ADD sp,#Next                    ;move to the next block
                BL main                 ;jump to main
                LDR r2,[sp],#Next               ;load the next position in stack into r2
                ADD sp,#Next                    ;move to next block
                MUL r1,r2,r0            ;multiple r2 and r0 then store in r1
```

```
                STR r1,[fp,#returnN]        ;store r1 in the return value position
                BL back                                     ;jump to back
even    LSR r2,r2,#1                        ;divide r2 by 2 and store in r2
                STR r2,[sp],#Next                   ;store r2 in the next block
                ADD sp,#Next                        ;move to pointer to next block
                BL main                                 ;jump to main
                LDR r2,[sp],#Next             ;load the content in the next block of the stack
into r2
                ADD sp,#Next                        ;move the pointer to next block

                MUL r1,r2,r2                        ;multiple r2 by r2 and store in r1
                STR r1,[fp,#returnN]        ;store r1 in the position of return value
                BL back                            ;jump to back
back    MOV sp,fp                           ;store fp to sp
                LDMEA sp!,{r1,r2,r3,fp,pc}  ;Load Multiple registers by empty  stack and
growth ascending
stack   DCD  0x00                                   ;stack start
                SPACE 0xA4;space to store the stack
result  DCD  0x00                           ;result
                END
```