AREA question1, CODE, READONLY

ENTRY

| base | EQU 2 | | ;storing variable |
|--------------------------------------|-------|-----------------------|---|
| exp | | EQU 2 | ;storing variables |
| | | | |
| | | ADR sp, stack | ;define stack parameter |
| | | MOV r1, #base | ;prepare base parameter |
| | | STR r1, [sp,#-4]! | ;push base parameter onto stack; |
| | | MOV r2, #exp | ;prepare exponent parameter |
| | | STR r2, [sp, #-4]! | ;push exp parameter onto stack |
| | | SUB sp, sp,#4 | ;reserve a place int the stack for return value |
| | | | |
| | | BL Power | ;calls power function |
| | | LDR r2, [sp], #4 | |
| | | ADD sp, sp, #8 | ;move stack pointer down 8 bytes to restore stack |
| | | STR r2, result | ;store the answer in a space reserved for result |
| Loop B Loop | | | |
| | | | |
| ;~~~~~~Power function ~~~~~~~~~~~~~~ | | | |
| | | | |
| Power | STMFD | sp!, {r0,r1,r2,fp,lr} | ;push frame pointer and link register onto stack |
| | | MOV fp,sp | ;make frame pointer point to the bottom |
| | | | |
| ,~~~~ | ~~~~~ | if exponent = 0 | unnanananananan |
| | | CMP r2, #0 | ;check if exponent in r1 is equal to 0 |
| | | MOVEQ r2, #1 | ;move 1 to register 1 if equal |
| | | BLEQ Return | ;call return |

;~~~~~~~~ if exponent & 1 is true~~~~~~~~~~~~~

TST r2, #0x1 ;test if number is odd

BLEQ Even ;if even call Even function

SUB sp,sp, #4 ;if odd create stack frame

SUB r2, r2, #1 ;substract r2 by 1

BL Power ;call Power function

LDR r2, [sp,#0] ;store returned value in r2

MUL r0, r1, r2 ;mutliply and store

result in r0

MOV r2, r0 ;move stored result to r2

ADD sp, sp, #4 ;increment stack pointer

B Return

Return STREQ r2, [fp, #20] ;store result

MOV sp,fp ;make frame pointer point to bottom

LDMFD sp!, {r0,r1,r2,fp,pc} ;restore values

Even ASR r2, #1 ;conduct arithmethic shift right

SUB sp,sp, #4 ;create stack frame

STR r2, [sp, #0]! ;store result to where stack pointer points to

SUB sp,sp, #4 ;if odd create stack frame

BL Power ;call Power function

LDR r2, [sp] ;store value that sp points to in r2

MUL r0, r2, r2 ;mutliply value by itself

MOV r2, r0 ;move result to r2

B Return ;call return function

;~~~~~~~ data ~~~~~~~~

AREA question1, DATA, READWRITE

result DCD 0x6666666

SPACE OxFF ;space reserved for stack

stack DCD 0x000 ;start of stack

END

Question1

N = 0 : need 1 frame

N = 1: need 2 frame

N = 2 : need 3 frame

N = 3: need 4 frame

N = 4 : need 5 frame

N = 5 : need 6 frame

N = 6: need 7 frame

N = 7 : need 8 frame

N = 8 : need 9 frame

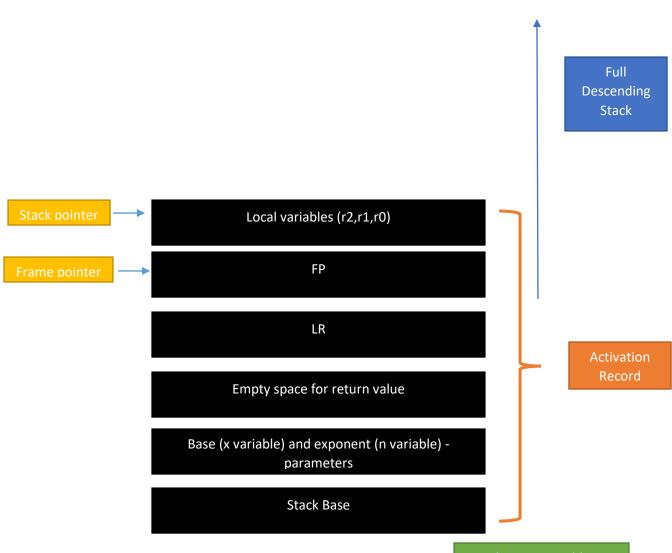
N = 9 : need 10 frame

N = 10 : need 11 frame

N = 11 : need 12 frame

N = 12 : need 13 frame

Low memory address



High memory address