Ding (Nick) Yu
Dyu97
250838916
Assignment 5

Source Code:

```
int power(int x, unsigned int n)
{
  int y;

  if (n == 0)
    return 1;

  if (n & 1)
    return x * power(x, n - 1);
  else
  { y = power(x, n >> 1);
    return y * y;
  }
```

```
1           AREA question1, CODE, READWRITE
2           ENTRY
3
4           LDR sp, =stack1             ;Initializing the stack pointer, the main() part of the function
5           BL power                    ;Calling the power function
6 loop      B loop                      ;Program is finished
7
8 power     STMIB sp, {r0-r12}          ;Store the original registers into the stack
9           LDR sp, =stack2             ;Initializing the stack pointer to where the values are being passed
10          LDMIB sp!, {r1-r2}          ;Load the value of x and n into respective registers
11          LDR r12, =n                 ;Set up the terminal condition of the stack to the bottom of the stack
12          MOV r3, #OddC               ;Used to check if n is odd (n & 1), indicated as OddCondition
13          MOV r4, #EvenC              ;Used to check if n is even, the else part
14
15 back     MOV r0, #1                  ;Base case, return 1
16          CMP r2, #0                  ;Check if n is zero - the terminal condition for the recursive function
17          BEQ terminal                ; if it is start popping and storing value
18
19          ANDS r3, r2, #Odd           ;Check if n is odd
20          BEQ elsec                   ;If not go to the else condition
21          SUBNE r2, r2, #1            ;If it is odd, n - 1
22          STR r3, [sp, #4]!           ;Remembers the odd condition, used to return x*power(x, n-1), then updates the relative position of the stack pointer to be used 1for the next call
23          B back                      ;Recursively calls back to the power function
24
25 elsec    MOV r2, r2, LSR #1          ;divides n by 2 using logical shift n >> 2
26          STR r4, [sp, #4]!           ;Remembers the even condition used to return y*y, then updates the relative position of the stack pointer to be used for the next call
27          B back                      ;Recursively calls back to the power function
28
```

```
29 terminal  STR r0, [sp, #4]            ;Once the terminal condition is reached, stores the base case on the stack
30 popc      CMP sp, r12                 ;Used to determine if the stack is fully returned
31           BEQ done                    ;Once the stack is fully returned, it goes to the end to store the result and load back the registers
32           LDR r5, [sp]                ;Use another pointer to check for the conditions remembered on the stack
33           ANDS r5, #Odd               ;When popping the stack, check the condition previously remembered, if it is one, it is the odd case, otherwise it is even
34           LDRNE r6, [sp,#4]           ;If it is odd, load back the value to calculate the first condition
35           BEQ condition2              ;If it is not odd, go to the second condition
36           MUL r6, r1, r6              ;x*power(x,n-1), performes the calculation and use a temporary register to store the value
37           STR r6, [sp], #-4           ;Return and stores it on the stack to be used for the next recursive call
38           B popc                      ;Pops the next item on the stack by going back
39 condition2 LDR r6, [sp,#4]            ;When it is not odd, use the item being popped
40           MOV r7, r6                  ;Use another temporary register to store the item, used for y = power(x, n>>1)
41           MUL r6, r7, r6              ;calculates y * y
42           STR r6, [sp], #-4           ;Return and stores it on the stack to be used for the next recursive call
43           B popc                      ;Pops the next item on the stack by going back
44
45
46
47 done      LDR sp, =stack1             ;Once the calculation is over, return to the stack where the value of the registers were previously stored
48           LDMIB sp, {r0-r12}          ;Load them back into their original registers
49           MOV pc, lr                  ;Link back to the main function
50
51           AREA A5Q1, DATA, READWRITE
52 OddC      EQU 1                       ;Used to remember the odd condition
53 EvenC     EQU 2                       ;Used to remember the even condition
54 Odd       EQU 0x1                     ;Used to check if n or the condition is odd
55 stack1    DCD 0x00                    ;First stack to store the original values of the register
56           SPACE 0x40                  ;Space created to store the values
57 stack2    DCD 0x00                    ;Used to pass down the parameters
58 x         DCD 2                       ;parameter x
59 n         DCD 9                       ;parameter n
60 result    DCD 0x00                    ;The beginning of the calculation stack, also where the returned result will be stored
61           END
```

Ding (Nick) Yu
Dyu97
250838916
Assignment 5

Structure of the stack frame: