CS2208b Assignment 5

Issued on: Tuesday, March 27, 2018

Due by: 11:55 pm on Tuesday, April 3, 2018

Stack Frame

-0x 24	R2 save
-0x 20	R1 save
-0x 16	R0 save
-0x 12	return address
-0x 8	return value
-0x 4	n
Bottom of Stack	x

The above configuration is one stack frame. For each recursive call of the power function, another identical copy of the above configuration will be created on the top of the stack. For example:

-0x 34	(1st recursion) R2 save
-0x 30	(1st recursion) R1 save
-0x 2C	(1st recursion) R0 save
-0x 28	(1st recursion) return address
-0x 24	(1st recursion) return value
-0x 20	(1st recursion) n
-0x 1C	(1st recursion) x
-0x 18	(0th recursion) R2 save
-0x 14	(0th recursion) R1 save
-0x 10	(0th recursion) R0 save
-0x C	(0th recursion) return address
-0x 8	(0th recursion) return value
-0x 4	(0th recursion) n
Bottom of Stack	(0th recursion) x

The stack has 0x200 reserved space.

Documentation

N	Stack Frames required
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

Code

Note: only 44 instructions required

```
                AREA Assignment5,CODE,READONLY
                ENTRY
                ;for safety, stack allocation will be set to 0x200
                ADR r13,Stack               ;set stack pointer to point to the stack

                MOV r0,#3                   ;change this to set x
                STR r0,[r13,#-4]!           ;push x to stack

                MOV r0,#5                   ;change this to set n
                STR r0,[r13,#-4]!           ;push n to stack

                ADR r0, after               ;push return address, with a return slot
                STR r0,[r13,#-8]!

                b power                     ;branch to function

after           LDR r2,[r13],#12            ;pull return into result, also pull twice more (x and n) to clean the stack
                STR r2,result

loop            b loop                      ;infinite loop forever

                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;Function
power                                       ;save registers
                STR r0,[r13,#-4]!           ;push r0 to preserve value
                STR r1,[r13,#-4]!           ;push r1 to preserve value
                STR r2,[r13,#-4]!           ;push r2 to preserve value

                LDR r2,[r13,#24]            ;read x into r2
                LDR r1,[r13,#20]            ;read n into r1



                CMP r1,#0                   ;check n==0
                BEQ ret1                    ;return 1 if n==0


                ANDS r0,r1,#1               ;check n&1

                                            ;{return x*power(x,n-1)
                STRNE r2,[r13,#-4]!         ;push x

                SUBNE r0,r1,#1              ;push n-1
                STRNE r0,[r13,#-4]!

```

```

            ADRNE r0,andl            ;push return address, with a slot for the return value
            STRNE r0,[r13,#-8]!

            BNE power                ;call power recursively

                                     ;else
                                     ;y = power(x,n>>1)
            STR r2,[r13,#-4]!        ;push x

            MOV r1,r1,LSR #1         ;push n>>1
            STR r1,[r13,#-4]!

            ADR r0,els               ;push return address, with a slot for the return value
            STR r0,[r13,#-8]!

            b power                  ;call power recursively


retl        MOV r0,#1                ;store 1 into r0
            b rtn                    ;branch to universal return

andl        LDR r0,[r13],#12         ;pull return into r0
            MUL r0,r2,r0
            b rtn                    ;branch to universal return

els         LDR r1,[r13],#12         ;pull return to r1
            MUL r0,r1,r1             ;y*y
                                     ;proceed to universal return

rtn         STR r0,[r13,#16]         ;store r0 into the return slot in the stack

                                     ;restore registers
            LDR r2,[r13],#4          ;pull r2
            LDR r1,[r13],#4          ;pull r1
            LDR r0,[r13],#4          ;pull r0


            ADD r13,r13,#4           ;pull return address into PC
            LDR PC,[r13,#-4]         ;go to return address

```

```
             ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;Function
             AREA Assignment5,DATA,READWRITE
             SPACE 0x200
Stack        DCD 0x00
result       DCD 0x00
             END
```