CS 2208 Assignment 5 report3

Michael Tang-Tran (250 735 158)

This program will recursively calculate the value of x^n with given x and n from the user. The program will use a full descending stack to do the calculations in the program.

A flow chart has been attached in the next page to demonstrate the thinking used in creating the program.

A stack frame diagram and a copy of the power.s file is provided in the document.

Table: Stack frames needed to calculate x^n for size exponent n.

Stack Frames Needed to Calculate X^n	**N**
0	0
1	1
3	2
5	3
6	4
8	5
10	6
11	7
13	8
14	9
15	10
17	11
19	12

Assembly Code:

```
; CS 2208b Assignment 5
; Michael Tang-Tran (250 735 158)
; This program will recursively calculate the value of x^n with given x and n.
```

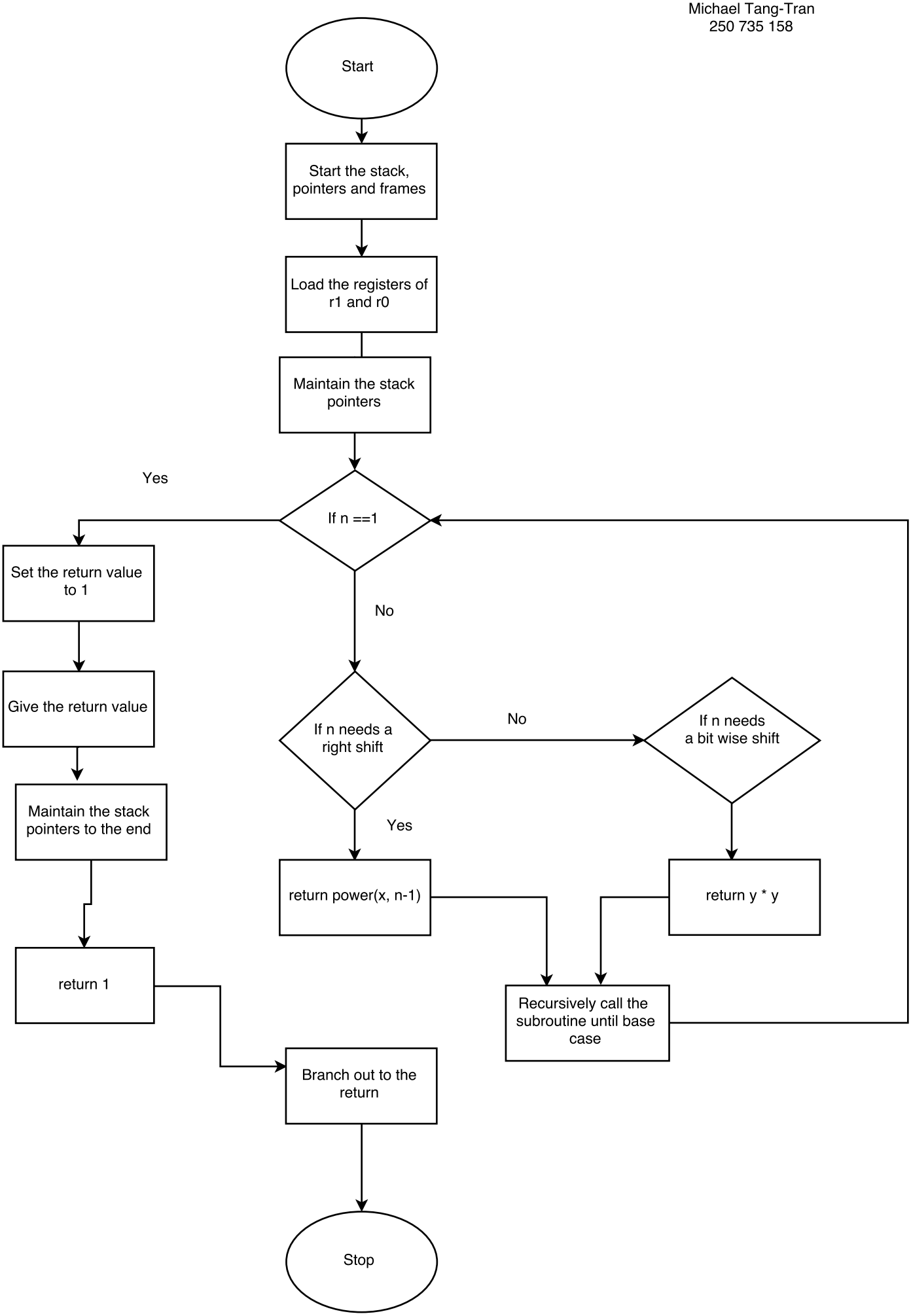
```
;-------------------------------------------------------------
        AREA power, CODE, READONLY
        ENTRY
main    ADR sp, stack                   ; Define the stack
        STR r0, [sp, #-4]!              ; push the parameter on the stack

        SUB sp, sp, #4                  ; reserve a place in the stack for the return
value

        BL pow                                  ; call the pow subroutine

        LDR r0, [sp], #4                ; Load the result into r0 and pop the stack
        ADD sp, sp, #4                          ; remove the parameter from the
stack

        ADR r1, result                  ; get the address of the stack
        STR r0, [r1]                    ; store the final result in the result variable.

Loop    B Loop
;-------------------------------------------------------------
        AREA power, CODE, READONLY

pow     STMFD sp!, {r0, r1, r2, fp, lr}         ; push general registers, as well as fp and lr

        MOV fp, sp                      ; set up fp for the call.
        SUB sp, sp, #4                  ; Create space for the x variable.

        LDR r0, [fp, #0x18]             ; get the parameter from the stack
```

```
        CMP r1, #0                  ; Check if n is 0
        MOVEQ r1, #1                ; prepare return to 1
        STREQ r1, [fp, #0x14]       ; store the returned value in the stack
        BEQ ret                     ; branch to the return section

        SUB r1, r0, #1              ; prepare the new parameter value
        STR r1, [sp, #-4]!          ; push the parameter on the stack.

        SUB sp, sp, #4              ; Prepare a spot for the return value
        BL pow                      ; Call the subroutine recursively.

        LDR r1, [sp], #4            ; Load the result and pop it from the stack
        ADD sp, sp, #4              ; remove the parameter from the stack


        LSR r1,         #1          ; logical shift of r1 by 1
        CMP r0, r1                  ; Compare the value of the function

        MULEQ r1, r0, r1            ; Get the return value for the stack
        MOVEQ r1, r1                ; Prepare the return value for the stack.
        STREQ r1, [FP, #0x14]       ; Store the value onto the stack
        BEQ ret                     ; prepare the return value

        LSREQ r1, #2                ; Prepare a bitwise shift of two.
        MULNE r0, r1, r1            ; Multiply for y * y
        STRNE r1, [fp, #0x14]        ; get the return value ready for the stack
        BEQ ret                     ; branch to the return section

ret     MOV sp, fp                  ; collapse all working spaces for this function call
```

```
        LDMFD sp!, {r0, r1, r2, fp, pc}     ; load all registers and return to the caller.
;-------------------------------------------------------------------------------
        AREA power, DATA, READWRITE
result  DCD 0x00                            ; the final result
        SPACE 0x84                          ; declare space for the stack
stack   DCD 0x00                            ; initial stack position (FD)

        END
```

CS 2208B report 3 flowchart

Michael Tang-Tran
250 735 158



Stack Frame

FD Stack