CS1022 Tutorial #3
Introduction to Subroutines

1 Nested Subroutines

Consider the following ARM Assembly Language program. Explain in detail the execution of the program. Why will the program not work? Suggest how you would fix the problem.

```
1  ; Top level program
2  start
3      BL  sub1          ; call sub1
4
5  stop
6      B   stop
7
8  ; sub1 subroutine
9  sub1
10     ADD R0, R1, R2    ; do something
11     BL  sub2          ; call sub2
12     ADD R3, R4, R5    ; do something
13     BX  lr            ; return from sub1
14
15 ; sub2 subroutine
16 sub2
17     BX  lr            ; return from sub2
```

2 Subroutine Calls and the System Stack

Consider the execution of the following ARM Assembly Language program. Illustrate the state of the system stack after the execution of each instruction that manipulates the stack. Note that the effect of the instructions on the system stack is cumulative.

```
1  start
2      BL       subroutine1
3  stop
4      B        stop
5
6  subroutine1
7      STMFD    sp!, {R1-R2,LR}
8      ADD      R0, R1, R2
9      MOV      R1, R0
10     BL       subroutine2
11     LDMFD    sp!, {R1-R2,PC}
12
13 subroutine2
14     STMFD    sp!, {LR}
15     MUL      R0, R1, R2
16     LDMFD    sp!, {PC}
```

3 Subroutine Interfaces

(a) For each of the following Java/C-like method declarations, design an appropriate ARM Assembly Language interface for a corresponding assembly language subroutine. The interface must include a specification of how each parameter is passed into the subroutine and how any return values are passed back to the calling program.

(i) `void zeroMemory(unsigned int startAddress, unsigned int length)`
(zero a range of addresses in memory)

(ii) `int divide(unsigned int x, unsigned int y)`

(iii) `int factorial(unsigned int x)`

(iv) `int power(int x, unsigned int y)`

(v) `int quadratic(int a, int b, int c, int x)`
(evaluate a quadratic function)

(vi) `void swap(unsigned int startAddress, int i, int j)`
(swap two elements in an array)

(b) Implement each of the above subroutines, taking care to hide any unintended side-effects from a calling program using LDM and STM instructions to save and restore registers on the system stack. Your subroutines must adhere to the interfaces that you defined above. Try to use registers R0–R3 to pass parameters and R4–R12 to store variables that are local to the subroutine.

(c) For each of the subroutines listed above, show how you would invoke (call) the subroutine, assuming the variables to be passed as parameters are initially stored in registers other than R0–R3. Comment your code.

(d) For each of the subroutines listed above, show the state of the system stack immediately after the subroutine saves any registers on the system stack.