

Solutions to Tutorial 3

3.1 Stack Manipulation Operations

- (1) `SP = 0xFEC`
- (2) `STR R1, [SP, #-4]!` Or `STMFD SP!, {R1}`
- (3) `LDMFD SP!, {R6-R9}`
- (4) `LDR R0, [SP, #8]`

3.2 Modular Programming – Subroutine Call and Parameter Passing

- (1) **NumX** and **NumY** are passed by value. Values of memory variables are pushed to the stack.
Ans is passed using reference. Address of the memory variable pushed onto the stack.

(2) After (b1) `PC = 0x014` `SP = 0xFFFFFFFF8`

After (c1) `PC = 0x040` `SP = 0xFFFFFFFF0`

After (s6) `PC = 0x028` `SP = 0xFFFFFFFF0`

→ (3) `R0=Value of R3 (can be 0)`

(4) `ADD SP, SP, #12`

→ (5) Replace `B1 MySub` with `MOV LR, PC` followed by `B MySub`.

(6) Suggested solutions:

MySub	<code>STMFD SP!, {R0-R3}</code>	; (s1) Save registers R0,R1,R2,R3
	<code>LDR R0, [SP, #24]</code>	; (s2) Retrieve NumX from stack
	<code>LDR R1, [SP, #20]</code>	; (s3) Retrieve NumY from stack
	<code>MOV R2, #0</code>	; Complete the segment of code to compute the
Loop	<code>ADD R2, R2, R1</code>	; value of NumX*NumY using successive addition
	<code>SUBS R0, R0, #1</code>	; decrement NumY till reach zero
	<code>BNE Loop</code>	; loop back till NumX added NumY times
	<code>LDR R3, [SP, #16]</code>	; (s4a) Move the result directly to ...
	<code>STR R2, [R3]</code>	; (s4b) ... the memory variable Ans
	<code>LDMFD SP!, {R0-R3}</code>	; (s5) Restore saved registers
	<code>MOV PC, LR</code>	; (s6) Return to calling program