Ch8 ARM Subroutines Quiz

Q1: What is the primary purpose of the Link Register (R14)?
A) To hold temporary data during arithmetic operations
B) To store the return address of a subroutine call
C) To hold the stack pointer value
D) To store interrupt status flags

ANS:

Q2: When the BL (Branch and Link) instruction is executed, what two operations occur?
A) PC is set to target address; LR is incremented by 1
B) LR holds the target address; PC is incremented by 2
C) LR = PC + 4 (return address); PC = target address
D) SP is decremented; PC is set to target address

ANS:

Q3: Which instruction is used to return control to the caller from a subroutine?
A) BL LR
B) BX LR
C) MOV PC, LR
D) JMP LR

ANS:

Q4: According to the ARM EABI, how are the first four 32-bit arguments passed to a subroutine?
A) In registers R0-R3
B) On the stack
C) In registers R4-R7
D) In registers R0-R3 for first two, on stack for the rest

ANS:

Q5: How is a 64-bit argument (such as a long long) passed to a subroutine?
A) In a single 64-bit register
B) Split across R0 and R2
C) In two consecutive 32-bit registers (e.g., R0:R1 or R2:R3)
D) Always passed on the stack

ANS:

Q6: When a subroutine requires more than four arguments, where are the extra arguments passed?
A) In memory-mapped registers
B) On the stack by the caller
C) In the program counter
D) In the link register

ANS:

Q7: Where is a 32-bit return value placed by a subroutine?
A) In register R0
B) In register R7
C) On the stack
D) In the link register

ANS:

Q8: What stack convention does ARM Cortex-M use?
A) Full ascending stack (stack grows toward high memory)
B) Empty descending stack (stack grows toward low memory)
C) Full descending stack (SP points to last item pushed; stack grows toward low memory)
D) Empty ascending stack

ANS:

Q9: PUSH {register_list} is equivalent to which instruction?
A) STMIA SP!, {register_list}
B) STMDB SP!, {register_list}
C) LDMIA SP!, {register_list}
D) LDMDB SP!, {register_list}

ANS:

Q10: POP {register_list} is equivalent to which instruction?
A) LDMIA SP!, {register_list}
B) STMIA SP!, {register_list}
C) LDMDB SP!, {register_list}
D) STMDB SP!, {register_list}

ANS:

Q11: When you execute PUSH {R1, R2, R3}, in what memory address order are the registers stored in memory?
A) R1, R2, R3 (from lowest to highest address)
B) R1, R2, R3 (lowest-numbered reg at lowest address; R1 pushed last)
C) R3, R2, R1 (from lowest to highest address)
D) No specific order; implementation-dependent

ANS:

Q12: On a PUSH operation in Cortex-M (full descending stack), when is SP decremented?
A) After storing each register
B) Only once at the end
C) Before storing the first register
D) SP is incremented, not decremented

ANS:

Q13: Which statement correctly describes caller-saved and callee-saved registers?
A) Caller-saved registers (R0-R3, R12) are preserved by the caller; callee-saved (R4-R11) by the callee
B) All registers are callee-saved

C) Caller-saved registers must be preserved by the callee
D) Only R0-R3 are ever used

ANS:

Q14: If a subroutine modifies register R4-R11, what must the subroutine do?
A) The caller is responsible for saving them
B) The subroutine must save them (PUSH) and restore them (POP)
C) They don't need to be preserved
D) Save only odd-numbered registers

ANS:

Q15: Why must a subroutine save the Link Register (LR) when calling another subroutine?
A) The ARM architecture requires it
B) The inner BL instruction overwrites LR with a new return address, losing the original return path
C) To improve processor performance
D) It's optional and rarely needed

ANS:

Q16: What defines a recursive function?
A) A function that calls itself on smaller sub-problems to solve a larger problem, with a base case
B) A function that calls multiple other functions
C) A function that loops indefinitely
D) A function with no parameters

ANS:

Q17: In a recursive factorial function, what is the base case?
A) factorial(n) = n × factorial(n-1)
B) factorial(0) = n
C) factorial(n) = 1 if $n \leq 1$
D) factorial(n) = n!7

ANS:

Q18: In a descending stack (used by Cortex-M), toward which direction does the stack grow?
A) Toward high memory addresses
B) Toward low memory addresses
C) Horizontally across the address bus
D) In a circular pattern

ANS:

Q19: After executing PUSH {R1, R2}, what is the memory layout (assuming R1=0x11111111, R2=0x22222222)?
A) [SP]: 0x11111111, [SP+4]: 0x22222222
B) [SP]: 0x22222222, [SP+4]: 0x11111111
C) [SP]: 0x22222222, [SP-4]: 0x11111111
D) Both registers stored at the same address

ANS:

Q20: According to the ARM EABI, what must be true about the Stack Pointer (SP/R13) after a subroutine returns?
A) SP can be any value
B) SP must have the same value as before the subroutine was called
C) SP is incremented by 4
D) SP is reset to 0

ANS: