Ch8 ARM Subroutines Quiz ANS

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: B
The Link Register (R14) stores the return address when a subroutine is called. When BL executes, it saves PC + 4 in LR, allowing the subroutine to return via BX LR.

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: C
BL performs: (1) LR = PC + 4 (save return address), (2) PC = target_address (branch to subroutine).

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

ANS: B
BX LR sets PC = LR, returning control to the caller. Equivalent to POP {PC}.

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: A
ARM EABI specifies: Argument 1→R0, Argument 2→R1, Argument 3→R2, Argument 4→R3.

Q5: According to the ARM EABI, how is a 64-bit argument (such as a long int) 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: C
64-bit values use two consecutive registers: R0:R1 (first argument) or R2:R3 (second argument).

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: B
When more than 4 arguments exist, extra arguments are passed on the stack by the caller, who is also responsible for cleaning up.

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: A
32-bit return values are placed in R0. 64-bit values use R0:R1, and 128-bit values use R0-R3.

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: C
ARM Cortex-M uses full descending stack: SP points to the last item pushed (full), and stack grows toward low memory addresses (descending).

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: B
PUSH is equivalent to STMDB SP! (Store Multiple, Decrement Before). DB means decrement SP first, then store.

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: A
POP is equivalent to LDMIA SP! (Load Multiple, Increment After). Load first, then increment SP.

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

ANS: B

Registers are sorted automatically by hardware. R1 (lowest number) is stored at the lowest memory address but pushed last. Final memory: [R1] [R2] [R3] (lowest to highest).

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: C
In full descending stack, SP is decremented BEFORE storing (pre-decrement). For PUSH {r4, r5}, SP is decremented by 8, then stores occur.

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: A
Caller-saved (R0-R3, R12): Caller preserves if needed. Callee-saved (R4-R11): Callee must preserve if modified.

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: B
If a subroutine modifies R4-R11 (callee-saved), it must save them at entry (PUSH) and restore at exit (POP).

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: B
When foo() calls bar(), the BL bar instruction overwrites LR with bar's return address, destroying foo's return address. Solution: foo must save LR before the nested call.

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: A
A recursive function calls itself on smaller sub-problems with a base case to terminate recursion.

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 ≤ 1
D) factorial(n) = n!7

ANS: C
Factorial base case: if (n ≤ 1) return 1. Prevents infinite recursion.

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: B
Descending stack grows toward low memory addresses. SP decreases as items are pushed.

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: A
R1 is at lower address of the new SP, R2 at higher address of SP+4.

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: B
ARM EABI requires SP to have the same value after subroutine return as before the call. Maintains stack balance and prevents corruption.