NUST CHIP DESIGN CENTRE

Digital Design Verification

Lab Manual # 16 – Calling Convention in RISC-V Assembly

Name	*Khalil Rehman*
Instructor	*Sir Umer*
Date	*20th July 2025*

1. Tasks:

Task # 1

```
163 pow:
164     # BEGIN PROLOGUE
165     addi sp, sp, -4
166     sw s0, 0(sp)
167     # END PROLOGUE
168     li s0, 1
```

```
188 inc_arr:
189     # BEGIN PROLOGUE
190     addi sp, sp, -16
191     sw ra, 12(sp)
192     sw s0, 8(sp)
193     sw s1, 4(sp)
194     # END PROLOGUE
195     mv s0, a0 # Copy start of array to saved register
196     mv s1, a1 # Copy length of array to saved register
197     li t0, 0 # Initialize counter to 0
```

```
228 helper_fn:
229     # BEGIN PROLOGUE
230     addi sp, sp, -4
231     sw s0, 0(sp)
232     # END PROLOGUE
233     lw t1, 0(a0)
234     addi s0, t1, 1
235     sw s0, 0(a0)
236     # BEGIN EPILOGUE
237     lw s0, 0(sp)
238     addi sp, sp, 4
239     # END EPILOGUE
240     jr ra
241
```

Execution:

Copy! Download! Clear!

```
Tests passed.
```

Question/Answers

1. No, next test is a label, not a function. It doesn't have a prologue/epilogue and doesn't use jal to call it.
2. Both functions modified saved registers (s0, s1) without preserving their values, violating the calling convention.
3. No, labels are local jump targets, not function calls. Calling convention only applies to function calls using jal/jalr.
4. inc_arr calls helper_fn using jal, which overwrites ra. Other functions (pow, helper_fn) don't call any functions, so they don't need to save ra.
5. The CC checker only checks functions marked with .globl. helper_fn isn't globally visible, so its violations weren't reported.

Task # 2

Error 1: Stack Pointer Not Adjusted

```
388     # FIX 1: Allocate stack space and save registers
389     addi sp, sp, -8   # space for 2 words (s0 and ra)
390     sw s0, 4(sp)      # Save s0 on stack
391     sw ra, 0(sp)      # Save return address on stack
392
```

Error 2: Return Address Not Saved

```
413 # FIX 2: Restore registers and deallocate stack
414     lw ra, 0(sp)  # Restore return address
415     lw s0, 4(sp)  # Restore s0
```

Error 3: Stack Not Balanced

```
416     addi sp, sp, 8 # Deallocate stack space
```

Execution:

Copy! Download!

```
1024
Exited with error code 0
```

Task # 3

A temporary register (t0) used without saving and restoring it across the recursive function call. According to RISC-V calling conventions, temporary registers (t0-t6) are not preserved across function calls. When ex3 calls itself recursively, the value in t0 gets overwritten, leading to incorrect results

```
450 ex3:
451
452     # Save return address and temporary registers
453     addi sp, sp, -8     # Allocate space on stack
454     sw ra, 0(sp)        # Save return address
455     sw t0, 4(sp)        # Save t0 register
456
```

```
483 ex3_end:
484  # Restore saved registers and return address
485     lw t0, 4(sp)        # Restore t0
486     lw ra, 0(sp)        # Restore return address
487     addi sp, sp, 8      # Deallocate stack space
```

Execution:

```
1024
Exited with error code 0
```

Conclusion:

In RISC-V programming, strictly following calling conventions is essential to avoid errors like infinite loops and incorrect results. Always save and restore registers (e.g., ra, s0-s11) that are modified across function calls and manage the stack properly by allocating and deallocating space. This ensures reliable execution, especially in recursive functions where register preservation is critical.