CS61C Spring 2026 — RISC-V Calling Convention, Discussion 5

1 RISC-V Calling Convention

1.1 Consider the following blocks of code:

```
main:
  # Prologue
  # Saves ra

  # Code omitted
  addi s0 x0 5
  # Breakpoint 1
  jal ra foo
  # Breakpoint 3
  mul a0 a0 s0
  # Code omitted

  # Epilogue
  # Restores ra
  j exit
```

```
foo:
  # Prologue
  # Saves s0

  # Code Omitted
  addi s0 x0 4
  # Breakpoint 2

  # Epilogue
  # Restores s0
  jr ra
```

a) Does `main` always behave as expected, as long as `foo` follows calling convention?

b) What does `s0` store at breakpoint 1? Breakpoint 2? Breakpoint 3?

c) Now suppose that `foo` didn't have a prologue or epilogue. What would `s0` store at each of the breakpoints? Would this cause errors in our code?

In part (c) above, we see one way how not following calling convention could make our code misbehave. Other things to watch out for are: assuming that `a` or `t` registers will be the same after calling a function, and forgetting to save `ra` before calling a function.

1.2 Function `myfunc` takes in two arguments: `a0`, `a1`. The return value is stored in `a0`. In `myfunc`, `generate_random` is called. It takes in 0 arguments and stores its return value in `a0`.

```
myfunc:
# Prologue (omitted)

addi t0 x0 1
slli t1 t0 2
add t1 a0 t1
add s0 a1 x0

jal generate_random

add t1 t1 a0
add a0 t1 s0

# Epilogue (omitted)
ret
```

a) Which registers, if any, need to be saved on the stack in the prologue?

b) Which registers, if any, need to be saved on the stack before calling `generate_random`?

c) Which registers, if any, need to be restored from the stack in the epilogue before returning?

2 Recursive Calling Convention

Write a function **`sum_squares`** in RISC-V that, when given an integer **`n`** and a constant **`m`**, returns the summation below. If **`n`** is not positive, then the function returns m.

$$m + n^2 + (n-1)^2 + (n-2)^2 + ... + 1^2$$

To implement this, we will use a tail-recursive algorithm that uses the **`a1`** register to help with recursion.

`sum_squares`: Return the value $m + n^2 + (n-1)^2 + ... + 1^2$		
Arguments	`a0`	A 32-bit number n. You may assume $n \leq 10000$.
	`a1`	A 32-bit number m.
Return value	`a0`	$m + n^2 + (n-1)^2 + (n-2)^2 + ... + 1^2$. If $n \leq 0$, return m

For this problem, you are given a RISC-V function called **`square`** that takes in a single integer and returns its square.

`square`: Squares a number		
Arguments	`a0`	n
Return value	`a0`	n^2

2.1 Since this a recursive function, let's implement the base case of our recursion:

```
sum_squares:
________________ zero_case

# To be implemented in the next question

zero_case:
________________
jr ra
```

2.2 Next, implement the recursive logic. *Hint: if you let* $m' = m + n^2$, *then*

$$m + n^2 + (n-1)^2 + ... + 1^2 = m' + (n-1)^2 + ... + 1^2$$

```
sum_squares:
# Handle zero case (previous question)
_________________ zero_case

mv t0 a0
jal ra _________________

add  a1 a0 a1
addi a0 t0 -1

jal ra _________________
jr ra

zero_case:
# Handle zero case (previous question)
jr ra
```

$\boxed{2.3}$ Now, think about calling convention from the caller perspective. After the call to `square`, what is in `a0` and `a1`? Which one of the registers will cause a calling convention violation?

$\boxed{2.4}$ What about the recursive call? What will be in `a0` and `a1` after the call to `sum_squares`?

2.5 Now, go back and fix the calling convention issues you identified. Note that not all blank lines may be used. There may also be another caller saved register that you need to save as well!

```
sum_squares:
# Handle zero case (previous question)
mv t0 a0
________________________
________________________
________________________
________________________

# (previous question)
jal ra _________________
________________________
________________________
________________________
________________________

add  a1 a0 a1
addi a0 t0 -1

________________________
________________________
________________________
# (previous question)
jal ra _________________
________________________
________________________
________________________

jr ra
zero_case:
# Handle zero case (previous question)
jr ra
```

2.6 Now, from a callee perspective, do we have to save any registers in the prologue and epilogue? If yes, what registers do we have to save, and where do we place the prologue and epilogue? If no, briefly explain why.