CS 3410 Lab 6

Spring 2026

Cornell Bowers C·IS
Computer Science

Agenda

1. RISC-V Calling Convention
2. Writing Assembly Functions
3. Assignment 6 Tips

RISC-V Calling Convention

RISC-V Calling Convention Review

- Arguments go in registers `a0-a7`
- Return values go in `a0-a1`
- Return *address* goes in `ra`
- Callee-saved registers are: `s0-s11`
- Stack pointer is saved in `sp`

Prologue:	`Epilogue:`
1. Adjust stack pointer	1. Restore saved registers
2. Save return address (`ra`)	2. Restore return address (`ra`)
3. Save callee-saved registers	3. Restore stack pointer
	4. Return to the caller using `ret`

RISC-V Calling Convention Review

- Arguments go in registers `a0-a7`
- Return values go in `a0-a1`
- Return *address* goes in `ra`
- Callee-saved registers are: `s0-s11`
- Stack pointer is saved in `sp`

Prologue:	`Epilogue:`
1. Adjust stack pointer (decrement) 2. Save return address (`ra`) (on the stack) 3. Save callee-saved registers (on the stack)	1. Restore saved registers 2. Restore return address (`ra`) 3. Restore stack pointer 4. Return to the caller using `ret`

(pseudo-instruction to jump to `ra`)

Calling convention for a leaf-function

Objective: Add 1 to the argument and return the result

```
int addOne(int i) {
  return i + 1;
}
```

Prologue:

1. Adjust stack pointer
2. Save return address (`ra`)
3. Save callee-saved registers

`Epilogue:`

1. Restore saved registers
2. Restore return address (`ra`)
3. Restore stack pointer
4. Return to the caller using `ret`

Add One (Solution)

```
addOne:
    # Prologue.
    addi sp, sp, -8  # Push the stack frame.
    sd   ra, 0(sp)   # Save return address.

    # Body.
    addi a0, a0, 1

    # Epilogue.
    ld   ra, 0(sp)   # Restore return address.
    addi sp, sp, 8   # Pop the stack frame.
    ret
```

Worksheet

Recursive Sum

Objective: Add all integers from 1 through *n*

```
int sum(int n) {
  if (n == 0)
    return n;
  return n + sum(n - 1);
}
```

Write the assembly in `recsum.s`!

Prologue:

1. Adjust stack pointer
2. Save return address (`ra`)
3. Save callee-saved registers

`Epilogue:`

1. Restore saved registers
2. Restore return address (`ra`)
3. Restore stack pointer
4. Return to the caller using `ret`

Don't forget to run with c!

Cornell Bowers CIS
Computer Science

Recursive Sum

Objective: Add all integers from 1 through *n*

```
int sum(int n) {
  if (n == 0)
    return n;
  return n + sum(n - 1);
}
```

Prologue:

1. Adjust stack pointer
2. Save return address (`ra`)
3. Save callee-saved registers

`Epilogue:`

1. Restore saved registers
2. Restore return address (`ra`)
3. Restore stack pointer
4. Return to the caller using `ret`

Write the assembly in `recsum.s`!

Don’t forget to run with c!

Is this function tail-recursive? How does the usage of the stack change as `n` grows?

Assignment 6 Tips

Assignment Overview

Objective: Implement the Fibonacci sequence using **four** different approaches

1. Recursive Fibonacci
   - Straightforward recursive approach
1. Memoized Fibonacci
   - Optimized version that avoids redundant calculations
1. Tail-Recursive Fibonacci
   - Tail-recursive version to reduce recursion overhead
1. Tail-Call Optimized Fibonacci
   - Fully optimized version that eliminates recursion entirely

Assignment Tips

- Ensure recursive calls **always** respect the calling convention!
- Ensure you allocate enough space on the stack to save the *ra* and *callee-saved registers*!
- Read the A6 instructions, especially if you get stuck!
- Review the RISC-V Calling Convention!
- Use the 3410 RISC-V Interpreter!
- Be organized and make comments! Your assembly code can grow to be very complex!

Good Luck!