CS 61C Fall 2019

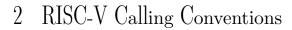
# RISC-V Control Flow Discussion 4: September 23, 2019

#### 1 RISC-V with Arrays and Lists

Comment what each code block does. Each block runs in isolation. Assume that there is an array, int arr[6] = {3, 1, 4, 1, 5, 9}, which starts at memory address 0xBFFFFF00, and a linked list struct (as defined below), struct 11\* 1st, whose first element is located at address 0xABCD0000. Let s0 contain arr's address 0xBFFFFF00, and let s1 contain 1st's address 0xABCD0000. You may assume integers and pointers are 4 bytes and that structs are tightly packed. Assume that 1st's last node's next is a NULL pointer to memory address 0x00000000.

```
struct ll {
         int val;
         struct 11* next;
     }
1.1
         t0, 0(s0)
     lw
        t1, 8(s0)
     lw
     add t2, t0, t1
         t2, 4(s0)
     SW
     loop: beq
                s1, x0, end
1.2
                 t0, 0(s1)
            lw
           addi t0, t0, 1
                 t0, 0(s1)
            SW
           lw
                 s1, 4(s1)
            jal
                x0, loop
      end:
1.3
            add t0, x0, x0
     loop:
            slti t1, t0, 6
            beq t1, x0, end
            slli t2, t0, 2
                 t3, s0, t2
            add
            lw
                  t4, 0(t3)
            sub
                 t4, x0, t4
            SW
                  t4, 0(t3)
            addi t0, t0, 1
             jal x0, loop
      end:
```

2 RISC-V Control Flow



[2.1] How do we pass arguments into functions?

2.2 How are values returned by functions?

2.3 What is sp and how should it be used in the context of RISC-V functions?

2.4 Which values need to saved by the caller, before jumping to a function using jal?

2.5 Which values need to be restored by the callee, before returning from a function?

### 3 More Translating between C and RISC-V

3.1 Translate between the RISC-V code to C. What is this RISC-V function computing? Assume no stack or memory-related issues, and assume no negative inputs.

| С                    | RISC-V               |
|----------------------|----------------------|
| // a0 -> x, a1 -> y, | Func: addi t0 x0 1   |
| // t0 -> result      | Loop: beq a1 x0 Done |
|                      | mul t0 t0 a0         |
|                      | addi a1 a1 -1        |
|                      | jal x0 Loop          |
|                      | Done: add a0 t0 x0   |
|                      | jr ra                |
|                      |                      |
|                      |                      |
|                      |                      |
|                      |                      |
|                      |                      |
|                      |                      |
|                      |                      |
|                      |                      |
|                      |                      |
|                      |                      |
|                      |                      |
|                      |                      |
|                      |                      |
|                      |                      |

## 4 Writing RISC-V Functions

4.1

Write a function sumSquare in RISC-V that, when given an integer n, returns the summation below. If n is not positive, then the function returns 0.

$$n^{2} + (n-1)^{2} + (n-2)^{2} + \ldots + 1^{2}$$

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

First, let's implement the meat of the function: the squaring and summing. We will be abiding by the caller/callee convention, so in what register can we expect the parameter n? What registers should hold square's parameter and return value? In what register should we place the return value of sumSquare?

#### 4 RISC-V Control Flow

4.2 Since sumSquare is the callee, we need to ensure that it is not overriding any registers that the caller may use. Given your implementation above, write a prologue and epilogue to account for the registers you used.