Topic V14

Register Calling Conventions
for Functions Inside Loops

Thinking about Data Flow

Reading: (Section 2.8)

Register Calling Convention Around Function Calls

What is wrong with the following RISC-V assembly code?

```
foo:
        …
        mv    t1, s1            # a1 ← s
        jal   ra, bar           # t ← bar(n)
        add   a0, a0, t1        # t + s
        jalr  zero, ra, 0       # return
```

The value in t1 is unknown after a function call

How About a Function Call Inside a Loop?

What is wrong with the following RISC-V assembly code?

```
foo:
        add   t1, zero, zero   # sum ← 0
next:
        beq   a0, zero, done   # if n == 0
        addi  a0, a0, -1       # n ← n – 1
        jal   ra, bar          # t ← bar(n)
        add   t1, t1, a0       # sum ← sum + t
        jal   zero, next
done:
        add   a0, zero, t1     # a0 ← sum
        jalr  zero, ra, 0
```

The value in t1 is unknown after a call to bar

How About a Function Call Inside a Loop?

Assume that i is a loop index.

What is wrong with the following RISC-V assembly code?

```
foo:
      add   s1, zero, zero   # sum ← 0
next:
      beq   a0, zero, done   # if i == 0
      addi  a0, a0, -1       # i ← i – 1
      jal   ra, bar          # t ← bar(n)
      add   s1, s1, a0       # sum ← sum + t
      jal   zero, next
done:
      add   a0, zero, s1     # a0 ← sum
      jalr  zero, ra, 0
```

The value in a0 has changed after a call to bar

How About a Function Call Inside a Loop?

What is still missing in this RISC-V assembly code?

Must save/restore s registers and ra

```
foo:
        add   s0, zero, a0      # s0 ← i
        add   s1, zero, zero    # sum ← 0
next:
        beq   s0, zero, done    # if i == 0
        addi  s0, s0, -1        # i ← i – 1
        add   a0, zero, s0      # a0 ← n
        jal   ra, bar           # t ← bar(n)
        add   s1, s1, a0        # sum ← sum + t
        jal   zero, next
done:
        add   a0, zero, s1      # a0 ← sum
        jalr  zero, ra, 0
```

Must Save/Restore Registers

```
foo:
        add     sp, sp, -12
        sw      ra, 8(sp)
        sw      s0, 4(sp)
        sw      s1, 0(sp)
        add     s0, zero, a0          # s0 ← n
        add     s1, zero, zero        # sum ← 0
next:
        beq     s0, zero, done        # if n == 0
        addi    s0, s0, -1            # n ← n – 1
        add     a0, zero, s0          # a0 ← n
        jal     ra, bar               # t ← bar(n)
        add     s1, s1, a0            # sum ← sum + t
        jal     zero, next
done:
        add     a0, zero, s1          # a0 ← sum
        lw      ra, 8(sp)
        lw      s0, 4(sp)
        lw      s1, 0(sp)
        add     sp, sp, 12
        jalr    zero, ra, 0
```