CS61C Discussion 3 – RISC-V

1 Powerful RISC-V Functions

1. Write a function `double` in RISC-V that, when given an integer x, returns $2x$.

2. Write a function `power` in RISC-V that takes in two numbers x and n, and returns x^n. You may assume that $n \geq 0$ and that multiplication will always result in a 32-bit number.

2 RISC-V with Arrays and Lists

Comment each snippet with what the snippet does. Assume that there is an array, `int arr[6] = {3, 1, 4, 1, 5, 9}`, which is starts at memory address 0xBFFFFF00, and a linked list struct (as defined below), `struct ll* lst;`, whose first element is located at address 0xABCD0000. `s0` then contains `arr`'s address, 0xBFFFFF00, and `s1` contains `lst`'s address, 0xABCD0000. You may assume integers and pointers are 4 bytes and that structs are tightly packed.

```
struct ll {
    int val;
    struct ll* next;
}
```

1.
```
lw  t0, 0(s0)
lw  t1, 8(s0)
add t2, t0, t1
sw  t2, 4(s0)
```

2.
```
        add  t0, x0, x0
loop:   slti t1, t0, 6
        beq  t1, x0, end
        slli t2, t0, 2
        add  t3, s0, t2
        lw   t4, 0(t3)
        sub  t4, x0, t4
        sw   t4, 0(t3)
        addi t0, t0, 1
        jal  x0, loop
 end:
```

3.
```
loop: beq  s1, x0, end
      lw   t0, 0(s1)
      addi t0, t0, 1
      sw   t0, 0(s1)
      lw   s1, 4(s1)
      jal  x0, loop
 end:
```

3 Translating between C and RISC-V

Translate between the C and RISC-V code. You may want to use the RISC-V Green Card as a reference. We show you how the different variables map to registers – you don't have to worry about the stack or any memory-related issues.

C	RISC-V

`// Nth_Fibonacci(n):`
`// s0 -> n, s1 -> fib`
`// t0 -> i, t1 -> j`
`// Assume fib, i, j are already these values`
`int fib = 1, i = 1, j = 1;`
`if (n==0)      return 0;`
`else if (n==1) return 1;`
`n -= 2;`
`while (n != 0) {`
`  fib = i + j;`
`  j = i;`
`  i = fib;`
`  n--;`
`}`
`return fib;`	

4 RISC-V Calling Conventions

1. How do we pass arguments into functions?

2. How are values returned by functions?

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

4. Which values need to saved before using `jal`?

5. Which values need to be restored before using `jr` to return from a function?

5 Writing RISC-V Functions

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.

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

For this problem, you are given a RISC-V function called `square` that takes in an integer and returns its square. Implement `sumSquare` using `square` as a subroutine.