# Homework 3

Benny Chen

September 28, 2022

## Question 1

Translate function foo() in the following C code to RISC-V assembly code. Assume function bar() has already been implemented. The constraints/tips are:

1. Allocate register s1 to sum, and register s2 to i.
2. There are no load or store instructions in the loop. If we want to preserve values across function calls, place the value in a saved register before the loop. For example, we keep variable i in register s2.
3. Identify the registers that are changed in function foo() but should be preserved. Note that the callee, bar(), may change any temporary and argument registers
4. Save registers at the beginning of the function and restore them before the exit.

Your code should follow the flow of the C code. Write concise comments. Clearly mark instructions for saving registers, loop, function calls, restoring register, etc.

```
// prototype of bar
// the first argument is an address of an integer
int  bar(int a[], int i);

int foo(int d[], int n)
{
    int sum = 0;
    for (int i = 0; i < n; i += 1) {
        sum += bar(&d[i], n - i);    // &d[i] means d[i]'s address
    }
    return sum;
}
```

**Answer:**

```
foo:
        addi    sp,sp,-20 #Allocate space
        sw      s1,0(sp)
        sw      s2,4(sp)
        sw      s3,8(sp)
        sw      s4,12(sp)
        sw      ra,16(sp)

        addi    s1,s1,0 #s1 = sum = 0
        addi    s2,s2,0 #s2 = i = 0
        addi    s3,s3,0 #s3 = d address = ?
        addi    s4,s4,100 #s4 = n = ?

loop:
        slli    a0,s2,2 #offset of i
        add     a0,a0,s3 #&d[i]
        sub     a1,s4,s2 #n-i
        jal     ra,bar #bar(&d[i],n-i)

        add     s1,s1,a0 #sum += output of bar(&d[i],n-i)

        addi    s2,s2,1 #i+=1
        blt     s2,s4,loop #if i < n

return:
        addi    a0,s1,0 #foo returns sum so into a0

        #restore all
        lw      s1,0(sp)
        lw      s2,4(sp)
        lw      s3,8(sp)
        lw      s4,12(sp)
        lw      ra,16(sp)

        addi    sp,sp,20

        jr ra
```

# Question 2

Translate function msort() in the following C code to RISC-V assembly code. Assume merge() and copy() are already implemented. The array passed to msort() has at most 256 elements. Your code should follow the flow of the C code. Write concise comments. Clearly mark instructions for saving registers, function calls, restoring register, and so on. To make the code easier to read, we change sp twice at the beginning of the function: once for saving registers and

once for allocating memory for array c. The function should have only one exit. There is only one return instruction. Another reminder: callees may change any temporary and argument registers.

## Question 3

## Question 4