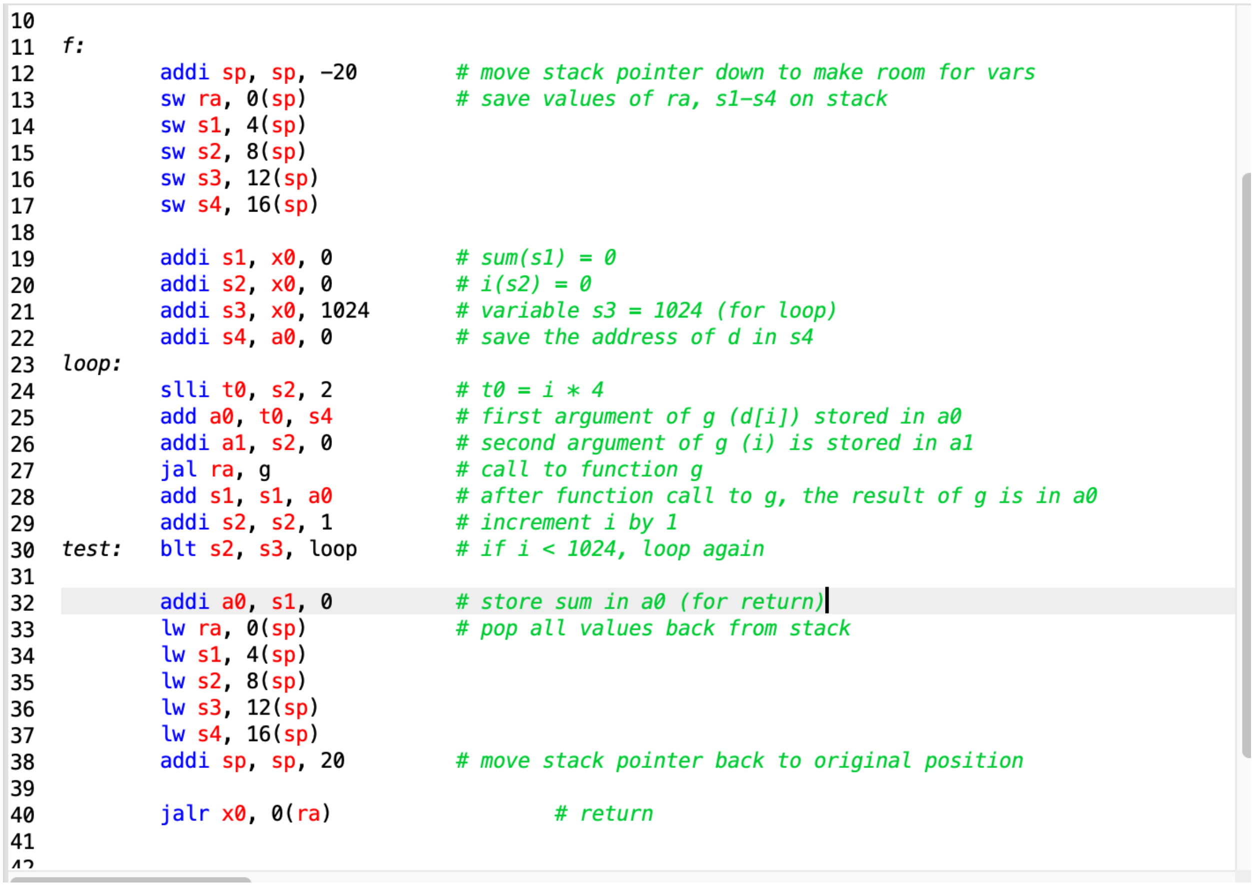
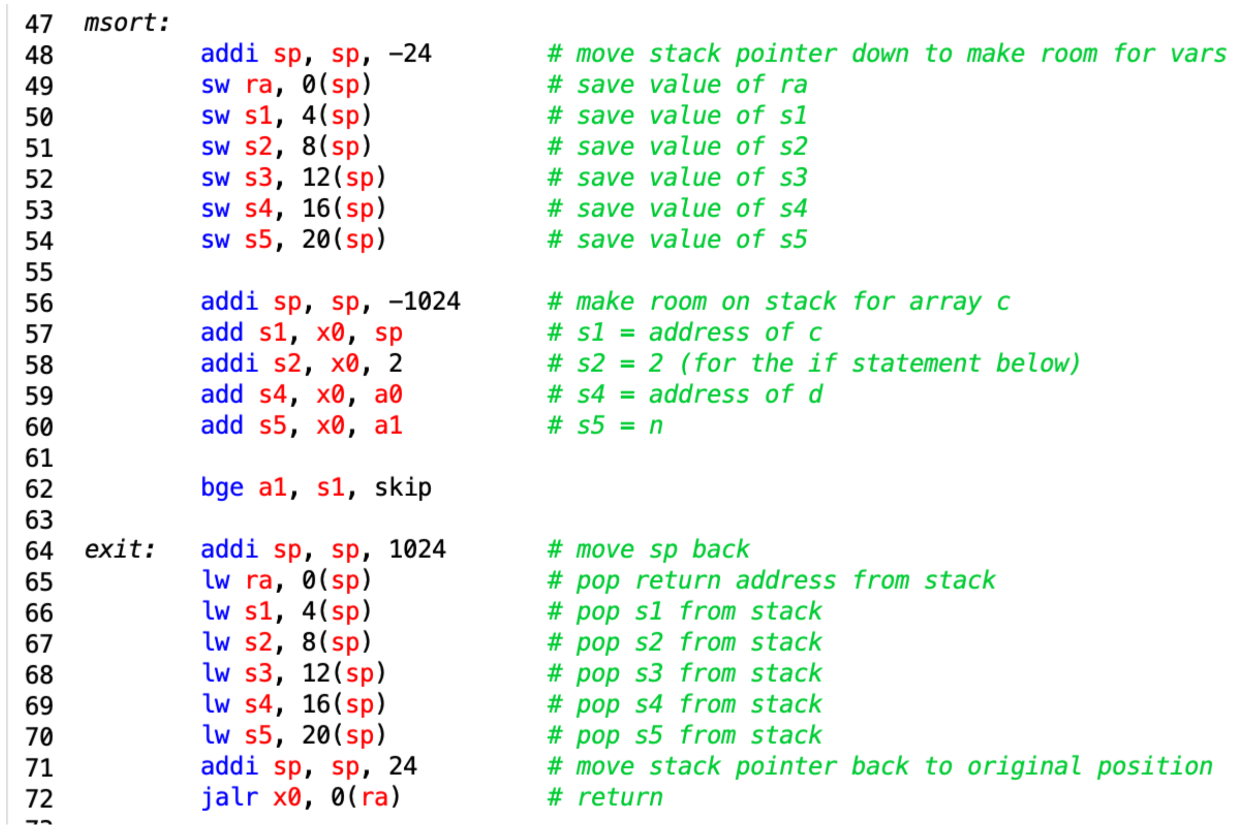
Jonathan Ameri
2/21/22

#5:

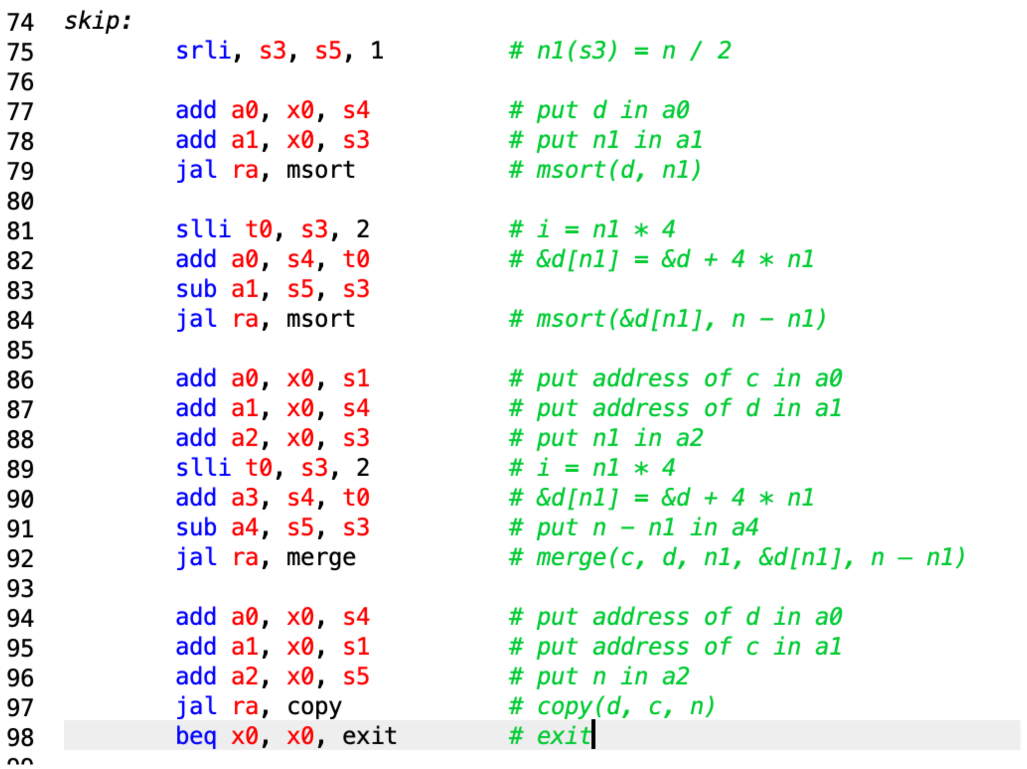


```

f:
        addi sp, sp, -20        # move stack pointer down to make room for vars
        sw ra, 0(sp)            # save values of ra, s1-s4 on stack
        sw s1, 4(sp)
        sw s2, 8(sp)
        sw s3, 12(sp)
        sw s4, 16(sp)

        addi s1, x0, 0          # sum(s1) = 0
        addi s2, x0, 0          # i(s2) = 0
        addi s3, x0, 1024       # variable s3 = 1024 (for loop)
        addi s4, a0, 0          # save the address of d in s4
loop:
        slli t0, s2, 2          # t0 = i * 4
        add a0, t0, s4          # first argument of g (d[i]) stored in a0
        addi a1, s2, 0          # second argument of g (i) is stored in a1
        jal ra, g               # call to function g
        add s1, s1, a0          # after function call to g, the result of g is in a0
        addi s2, s2, 1          # increment i by 1
test:   blt s2, s3, loop        # if i < 1024, loop again

        addi a0, s1, 0          # store sum in a0 (for return)
        lw ra, 0(sp)            # pop all values back from stack
        lw s1, 4(sp)
        lw s2, 8(sp)
        lw s3, 12(sp)
        lw s4, 16(sp)
        addi sp, sp, 20         # move stack pointer back to original position

        jalr x0, 0(ra)                  # return

```

#6:



```
msort:
        addi sp, sp, -24        # move stack pointer down to make room for vars
        sw ra, 0(sp)            # save value of ra
        sw s1, 4(sp)            # save value of s1
        sw s2, 8(sp)            # save value of s2
        sw s3, 12(sp)           # save value of s3
        sw s4, 16(sp)           # save value of s4
        sw s5, 20(sp)           # save value of s5

        addi sp, sp, -1024      # make room on stack for array c
        add s1, x0, sp          # s1 = address of c
        addi s2, x0, 2          # s2 = 2 (for the if statement below)
        add s4, x0, a0          # s4 = address of d
        add s5, x0, a1          # s5 = n

        bge a1, s1, skip

exit:   addi sp, sp, 1024       # move sp back
        lw ra, 0(sp)            # pop return address from stack
        lw s1, 4(sp)            # pop s1 from stack
        lw s2, 8(sp)            # pop s2 from stack
        lw s3, 12(sp)           # pop s3 from stack
        lw s4, 16(sp)           # pop s4 from stack
        lw s5, 20(sp)           # pop s5 from stack
        addi sp, sp, 24         # move stack pointer back to original position
        jalr x0, 0(ra)          # return
```



```
skip:
        srli, s3, s5, 1         # n1(s3) = n / 2

        add a0, x0, s4          # put d in a0
        add a1, x0, s3          # put n1 in a1
        jal ra, msort           # msort(d, n1)

        slli t0, s3, 2          # i = n1 * 4
        add a0, s4, t0          # &d[n1] = &d + 4 * n1
        sub a1, s5, s3
        jal ra, msort           # msort(&d[n1], n - n1)

        add a0, x0, s1          # put address of c in a0
        add a1, x0, s4          # put address of d in a1
        add a2, x0, s3          # put n1 in a2
        slli t0, s3, 2          # i = n1 * 4
        add a3, s4, t0          # &d[n1] = &d + 4 * n1
        sub a4, s5, s3          # put n - n1 in a4
        jal ra, merge           # merge(c, d, n1, &d[n1], n - n1)

        add a0, x0, s4          # put address of d in a0
        add a1, x0, s1          # put address of c in a1
        add a2, x0, s5          # put n in a2
        jal ra, copy            # copy(d, c, n)
        beq x0, x0, exit        # exit
```