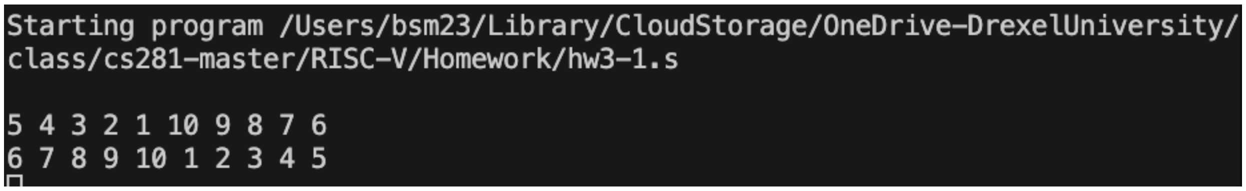
CS281 – Homework #3

The goal of this homework is to gain some additional hands-on experience writing RISC-V programs correctly. For this assignment I will be providing you with a short scaffold. Your objective is to write a program (filling out the functions below) to reverse the array A and then print out the result. I would recommend implementing it in the Venus emulator we have been using for labs to make sure things are operating correctly. The output should look something like this:



```
Starting program /Users/bsm23/Library/CloudStorage/OneDrive-DrexelUniversity/
class/cs281-master/RISC-V/Homework/hw3-1.s

5 4 3 2 1 10 9 8 7 6
6 7 8 9 10 1 2 3 4 5
```

Here is your scaffold – to make things a little easier to grade, just submit a single .s file in blackboard with your solution. Also, please remove any scaffolded comments that I provide below to help you along, and insert meaningful comments to document your code.

```
.data

    A : .word 5, 4, 3, 2, 1, 10, 9, 8, 7, 6
    newline: .asciiz "\n"
    space: .asciiz " "

    .equ sz_A, 10


.text
    .globl  main      # assembly directive that makes the symbol main
                      # global and this is where execution starts

main:
    #its a good practice to setup the arguments prior to making
    #a function call, sometimes this is not necessary but it is
    #a good habit to get into.

    la a0, A        # s0 = &A[0]
    li a1, sz_A     # s1 = sz_A
    jal print_array

    la a0, A        # s0 = &A[0]
    li a1, sz_A     # s1 = sz_A
    jal reverse

    la a0, A        # s0 = &A[0]
    li a1, sz_A     # s1 = sz_A
    jal print_array
    ret

# This is the reverse function...
#       Reverse (a0 = &A[0], a1 = sz_A)
# After the call to reverse, the array A will be reversed
reverse:
    #make sure you setup a stack frame, this is a non leaf function
    #thus you will need to at the minimum save ra, you can add
    #more registers to save if you need to

    addi sp, sp #FINISH THIS LINE – should be the first line of code

    #setup an manage a loop to reverse the array.  Use the loop
    #to iterate over the array, however use a helper function called
    #swap to handle the actual swapping of elements in the reversal
    #process.  Make sure you you properly clean up the stack before
    #returning...

    #dont forget to use the swap function aka jal swap in your solution

    #restore the ra register, do appropriate cleanup, restore the
    #stack and return
    lw ra,          #FINISH THIS LINE
    addi sp, sp     #FINISH THIS LINE
    jr ra           #Go back to main


# This is the swap function...
#      swap (a0 = &A[0], a1 = i, a2 = j)
# After the call to swap, the elements at A[i] and A[j] will be swapped
swap:
   #Implement this function, you should be able to do this as a leaf function

   jr ra    #return to reverse

   ret

# This is the print_array function...
#       print_arrray(a0 = &A[0], a1 = sz_A)
# This function should print the array. Use a space
# between printing out each element and print a newline
# at the end (after the elements have printed).
print_array:
    #Implement this function, you should be able to do this
    #as a leaf function.  See lab2 for examples of how
    #to print out strings and integers
    jr ra     #return to main
```