NYU Tandon School of Engineering Fall 2023, ECE 6913

**Homework Assignment 7** *Instructor: Azeez Bhavnagarwala,* email: [ajb20@nyu.edu](mailto:ajb20@nyu.edu) *Course Assistant Office Hour Schedule*

On Zoom: 9:30AM – 11AM Monday, Tuesday, Wednesday & Thursday(Focus on Project A)

1. Arushi Arora, [aa10350@nyu.edu](mailto:aa10350@nyu.edu)
2. Prashanth Rebala, [pr2359@nyu.edu](mailto:pr2359@nyu.edu)
3. Moin Khan, [mk8793@nyu.edu](mailto:mk8793@nyu.edu)
4. Raj Ghodasara, [g4357@nyu.edu](mailto:g4357@nyu.edu)

**Homework Assignment 7** [released Thursday November 9th 2022] [due Wednesday December 6th by 11:59PM]

You *are allowed* to discuss HW assignments with anyone. You are *not allowed* to share your solutions with other colleagues in the class. Please feel free to reach out to the Course Assistants or the Instructor during office hours or by appointment if you need any help with the HW.

Please enter your responses in this Word document after you download it from NYU Classes.

*Please use the Brightspace portal to upload your completed HW.*

* + Please use the online 32-bit RISC V simulator:

[**https://www.kvakil.me/venus/**](https://www.kvakil.me/venus/)

or

[**https://www.cs.cornell.edu/courses/cs3410/2019sp/riscv/interpreter/**](https://www.cs.cornell.edu/courses/cs3410/2019sp/riscv/interpreter/)

* + Please write the RISC V code, run it online to test/debug, demonstrate it works, include your code in the PDF you upload – as text not as an image
  + Your code is graded for (1) validity (it works) (2) size (fewer lines, higher grades) (3) discussion explaining choices you made and why
  + You cannot use/copy parts of or all of anyone else’s code

1. Write a RISC V program using instructions in the RISC V ISA to calculate the sum of the squares of all odd numbers between 0 and

+N where N is an integer < 100

.text # recursive implementation of factorial

.globl \_\_start

fact: # arg: n in a0, returns n! in a1

addi sp, sp, -8

sw ra, 0(sp)

li t0, 2

blt a0, t0, ret\_one # 0! and 1! == 1

addi a0, a0, -1

jal fact # call fact (n-1)

# a1 <- fact(n-1)

lw t0, 4(sp) # t0 <- n

mul a1, t0, a1 # a1 <- n \* fact(n-1)

j done

ret\_one:

li a1, 1

done:

lw ra, 0(sp) # restore return address from stack

addi sp, sp, 8 # free our stack frame

jr ra # and return

\_\_start:

li a0, 5 # compute 5!

jal fact

li a0, 1 # print it

ecall

li a0, 17

ecall # and exit

1. Write a RISC V program using instructions in the RISC V ISA to calculate the factorial of any positive integer N < 10

# RISC-V Assembly Program to Calculate the Factorial of N

# N = user-defined (N < 10)

.data

N: .word 5 # Change the value of N as needed (N < 10)

factorial: .word 1 # Variable to store the factorial result

.text

la a1, N # Load the address of N into a1

lw a1, 0(a1) # Load the value of N into a1

li t0, 1 # Initialize t0 to 1 (factorial)

li t1, 1 # Initialize t1 to 1 (loop counter)

loop:

bge a1, t1, end\_loop # If a1 >= t1, exit the loop

mul t0, t0, t1 # Multiply the current factorial by the loop counter

addi t1, t1, 1 # Increment the loop counter

j loop

end\_loop:

la a2, factorial # Load the address of factorial into a2

sw t0, 0(a2) # Store the final factorial result in the factorial variable

# Exit program (you can add additional code or syscalls here if needed)

li a7, 10 # System call number for program exit

ecall

1. Write a RISC V program using instructions in the RISC V ISA to calculate the sum of all prime numbers less than a given integer N where N < 100

.data

N: .word 50

sum: .word 0

.text

main:

li t0, 2

la t1, N

lw t2, 0(t1)

la t1, sum

loop:

beq t0, t2, exit

mv a0, t0

jal is\_prime

not\_prime\_ret:

beqz a1, not\_prime

lw t3, 0(t1)

add t3, t3, t0

sw t3, 0(t1)

not\_prime:

addi t0, t0, 1

j loop

exit:

lw a0, 0(t1)

li a7, 93

ecall

is\_prime:

li t4, 2

divide\_loop:

rem t5, a0, t4

beqz t5, not\_prime\_in\_func

blt t4, a0, cont

j end

not\_prime\_in\_func:

li a1, 0

j end

cont:

addi t4, t4, 1

j divide\_loop

end:

li a1, 1

jr ra

1. Write a RISC V program that calculates the sum of N terms in a geometric series where a = 1 and r = -3

# RISC-V Assembly Program to Calculate the Sum of N Terms in a Geometric Series

# a = 1 (initial term), r = -3 (common ratio), N = user-defined

.data

N: .word 10 # Change the value of N as needed

a: .word 1 # Initial term

r: .word -3 # Common ratio

sum: .word 0 # Variable to store the sum

.text

la a1, N # Load the address of N into a1

lw a1, 0(a1) # Load the value of N into a1

la a2, a # Load the address of a into a2

lw a2, 0(a2) # Load the initial term a into a2

la a3, r # Load the address of r into a3

lw a3, 0(a3) # Load the common ratio r into a3

li t0, 0 # Initialize t0 to 0 (sum)

loop:

beqz a1, end # If N is zero, exit the loop

mul t1, t0, a2 # Multiply the current sum by the current term

add t0, t1, t0 # Add the result to the sum

mul a2, a2, a3 # Multiply the current term by the common ratio

addi a1, a1, -1 # Decrement N

j loop

end:

la a4, sum # Load the address of sum into a4

sw t0, 0(a4) # Store the final sum in the sum variable

# Exit program (you can add additional code or syscalls here if needed)

li a7, 10 # System call number for program exit

ecall

1. Write a RISC V program that calculates the sum of N terms in a arithmetic series where a0 = 1 and d = 3

# RISC-V Assembly Program to Calculate the Sum of N Terms in an Arithmetic Series

# a0 = 1 (initial term), d = 3 (common difference), N = user-defined

.data

N: .word 10 # Change the value of N as needed

a0: .word 1 # Initial term

d: .word 3 # Common difference

sum: .word 0 # Variable to store the sum

.text

la a1, N # Load the address of N into a1

lw a1, 0(a1) # Load the value of N into a1

la a2, a0 # Load the address of a0 into a2

lw a2, 0(a2) # Load the initial term a0 into a2

la a3, d # Load the address of d into a3

lw a3, 0(a3) # Load the common difference d into a3

li t0, 0 # Initialize t0 to 0 (sum)

loop:

beqz a1, end # If N is zero, exit the loop

add t0, t0, a2 # Add the current term to the sum

add a2, a2, a3 # Update the current term by adding the common difference

addi a1, a1, -1 # Decrement N

j loop

end:

la a4, sum # Load the address of sum into a4

sw t0, 0(a4) # Store the final sum in the sum variable

# Exit program (you can add additional code or syscalls here if needed)

li a7, 10 # System call number for program exit

ecall