Department of Electrical and Computer Engineering
Rutgers, The State University of New Jersey

Computer Architecture and Assembly Lab
Spring 2021

Lab 4
RISC-V Assembly

Instructions

Please complete all the exercises below. Use the Venus RISC-V simulator to test your code. Note that the simulator is 32-bit, and we do not consider overflow here.
Be sure to comment on your code for clarity.
Once complete, upload your source code and PDF lab report on Sakai.

Exercises

1. [20 pts] Write a function in RISC-V that computes the following:
   - When given a positive integer **x** as an input, return **10x** using only **add** instructions.
   - When given a negative integer **y** as an input, compute the opposite value (i.e. **-y**) *without* using **mul** instructions.

Have a **main()** function check the value of the input, perform the appropriate computations, and print the output in the console. Note: you need to provide your own inputs and show screenshots of the outputs based on the given inputs. (Each part worth 10 pts.)

Department of Electrical and Computer Engineering
Rutgers, The State University of New Jersey

2. [30 pts] Write a function **exp()** in RISC-V that, when given a positive integer **x**, returns $\mathbf{x^5+6x^3+3x+4}$. Your code should include the following:

- Write a **main()** function to call the **exp** function and print the output in the console. [5 pts]
- Function **exp()** will compute and return the expression. Function **exp** must call the following additional functions: [10 pts]
    - Call a **power()** function to calculate $\mathbf{x^n}$. You may assume **n** will be positive. [10 pts]
    - Call a **times()** function to calculate **n*x**. [5 pts]

Note: you need to provide your own inputs and show screenshots of the outputs based on the given inputs.

Department of Electrical and Computer Engineering
Rutgers, The State University of New Jersey

3. [50 pts] Write a function **reorder** in RISC-V that, when given an array {3, 7, 45, 66, 80, 1}, returns the array in **reverse order** (i.e. the first number is now the last). Your code should include the following:

- Write a **main()** function to perform the following tasks:
  - Call an **input** function to write the given array values into a certain continuous memory starting from **0(0x0fffffe8)**. Your **input** function should work for any size of the array.
  - Call the **reorder** function to reorder the array.
    - Your function **reorder()** must call a **swap()** function to perform the necessary operations to reorder two elements of the array.
  - Call an **output()** function to print the reordered array values in the console. Your **output()** function should work for any array size.

Note: you need to show screenshots of the outputs based on the given inputs.
(There are 5 functions in total, and each function worth 10pts.)