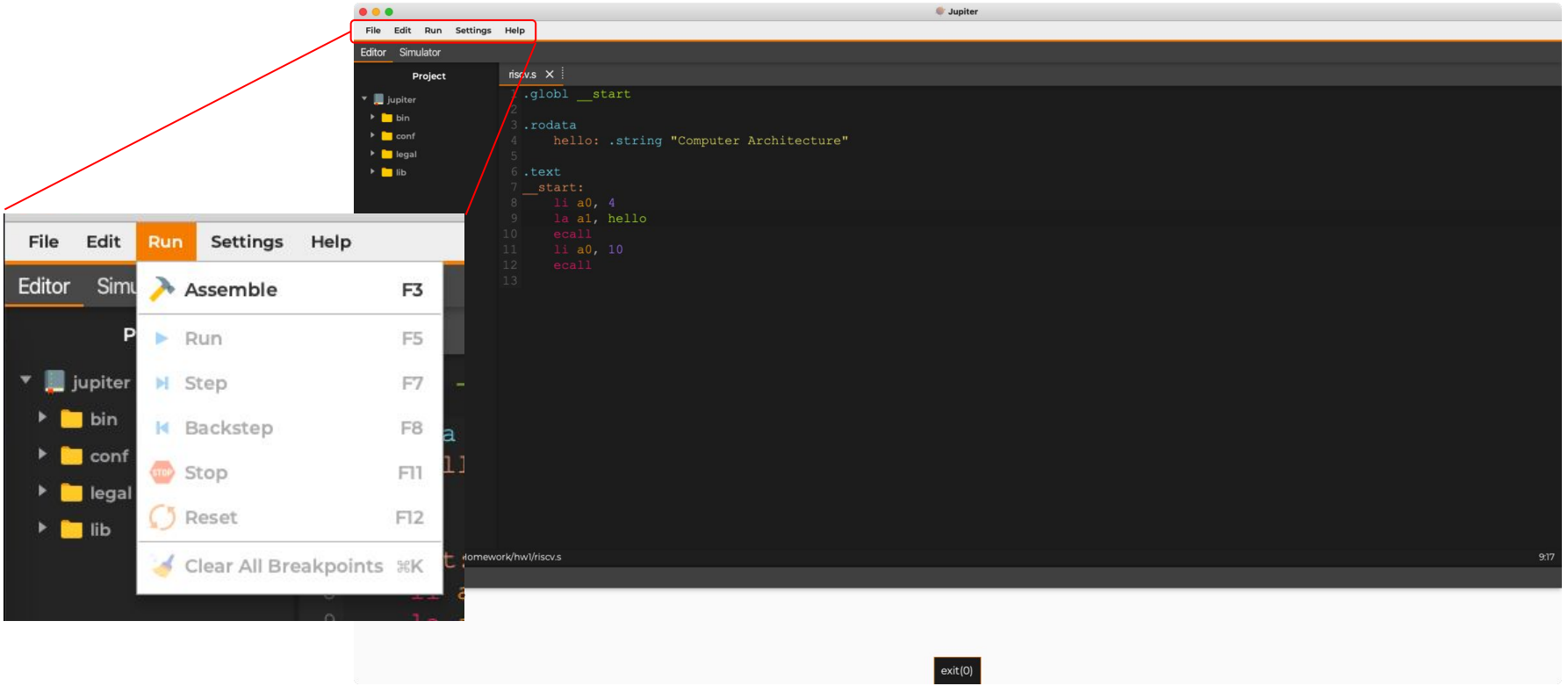
CA2022 Fall HW1

RISC-V Assembly Code

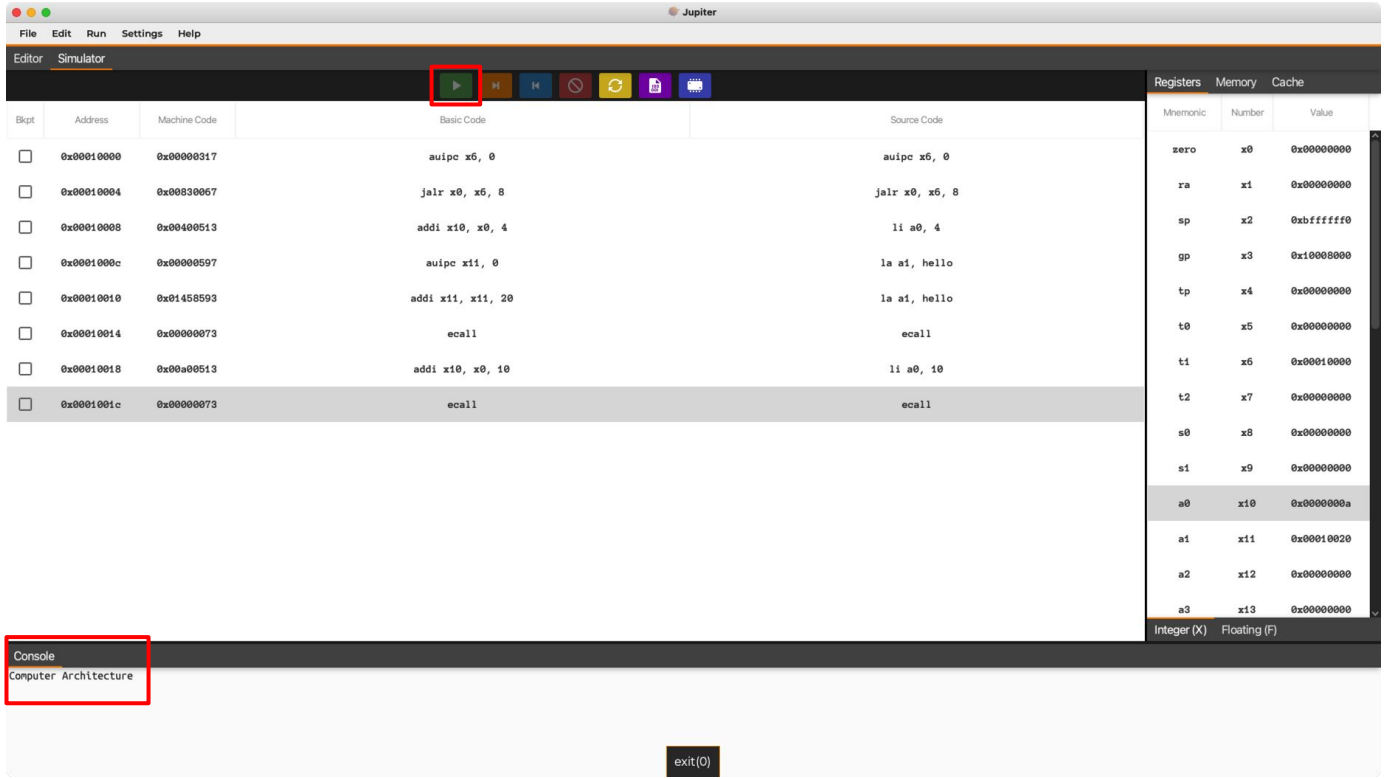
Description

- In this homework, you are going to use Jupiter RISC-V simulator to develop a simple calculator.
- After finishing this homework, you will be familiar with the usage of Jupiter RISC-V simulator, register definition, and some basic operations in RV32I Base Integer Instruction Set.

Jupiter GUI


Jupiter
File Edit Run Settings Help
Editor Simulator
Project
jupiter
bin
conf
legal
lib
riscv.s
.globl __start
.rodata
hello: .string "Computer Architecture"
.text
__start:
li a0, 4
la a1, hello
ecall
li a0, 10
ecall
homework/hw1/riscv.s
9:17
exit(0)
File Edit Run Settings Help
Assemble F3
Run F5
Step F7
Backstep F8
Stop F11
Reset F12
Clear All Breakpoints ⌘K

Jupiter GUI


Jupiter
File Edit Run Settings Help
Editor Simulator
Bkpt Address Machine Code Basic Code Source Code
0x00010000 0x00000317 auipc x6, 0 auipc x6, 0
0x00010004 0x00830067 jalr x0, x6, 8 jalr x0, x6, 8
0x00010008 0x00400513 addi x10, x0, 4 li a0, 4
0x0001000c 0x00000597 auipc x11, 0 la a1, hello
0x00010010 0x01458593 addi x11, x11, 20 la a1, hello
0x00010014 0x00000073 ecall ecall
0x00010018 0x00a00513 addi x10, x0, 10 li a0, 10
0x0001001c 0x00000073 ecall ecall
Registers Memory Cache
Mnemonic Number Value
zero x0 0x00000000
ra x1 0x00000000
sp x2 0xbffffff0
gp x3 0x10008000
tp x4 0x00000000
t0 x5 0x00000000
t1 x6 0x00010000
t2 x7 0x00000000
s0 x8 0x00000000
s1 x9 0x00000000
a0 x10 0x0000000a
a1 x11 0x00010020
a2 x12 0x00000000
a3 x13 0x00000000
Integer (X) Floating (F)
Console
Computer Architecture
exit(0)

Jupiter CLI

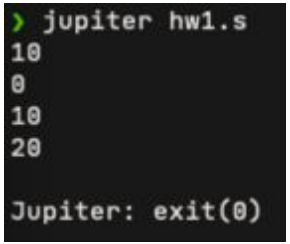
```
❯ jupiter riscv.s
Computer Architecture

Jupiter: exit(0)
```

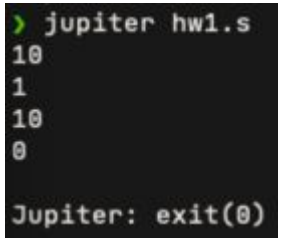
TODO

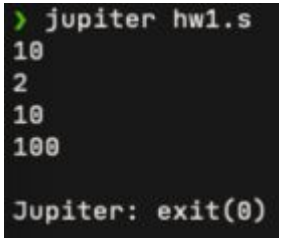
- You are going to develop a simple calculator, which supports seven operations.
- Addition(0), subtraction(1), multiplication(2), integer division(3), minimum(4), power(5), and factorial(6).

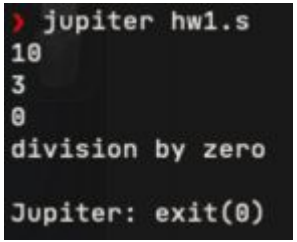
Sample I/O

- Input file contains 3 lines, operand A, operation op, operand B, respectively. ($0 \leq A, B \leq 1024$, $op \in \{0, 1, 2, 3, 4, 5, 6\}$)
- Your program should output the correct result (A op B).









Sample Code

- In the sample code, you don’t need to do I/O operations by yourself. A, op, B will be stored at register s0, s1, s2 registers.
- You need to store the result to register s3.

```
22 ###################################
23 #  TODO: Develop your calculator  #
24 #                                 #
25 ###################################
```

Sample Code

- If op=3 and B=0, just jump to **`division_by_zero_except`**

```
38 division_by_zero_except:
39     li a0, 4
40     la a1, division_by_zero
41     ecall
42     jal zero, exit
```

Grading Policy

- Total 100%
  - For operations +, -, x, / and min, each has 4 test cases, 3 points per test case.
  - For operations ^ and !, each has 5 test cases, 4 points per test case.
- We will judge the correctness of your program by running the following command:

```
$ jupiter [student_id]_hw1.s < input_file
```

- Don’t worry about overflow and underflow.
- No need to handle 0^0.
- 10 points off per day for late submission.
- You will get 0 point for plagiarism.

Submission

- Due date: 10/4 23:59 (Tuesday)
- Please rename your program [student_id]_hw1.s and upload it to NTU COOL.
    - For example, if your student id is b12345678, your program file name should be b12345678_hw1.s.