CA 2024 Spring HW1

RISC-V Assembly Code

Agenda

- Assignment Introduction
  - Description
  - Sample I/O
  - Sample Code
- Grading Policy
- Submission
- Jupiter Installation

Assignment Introduction

- 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

RISC-V Assembler & Runtime Simulator

Description

- 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. ($0 \le A, B \le 1024$, op $\in \{0, 1, 2, 3, 4, 5, 6\}$)
- Your program should output the correct result (A op B).

```
❯ ./jupiter hw1.s
10
0
10
20

Jupiter: exit(0)
```

Add

```
❯ ./jupiter hw1.s
10
3
0
division by zero

Jupiter: exit(0)
```

Division

```
❯ ./jupiter hw1.s
2
5
5
32

Jupiter: exit(0)
```

Power

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.

```
.text
__start:
    # Read first operand
    li a0, 5
    ecall
    mv s0, a0
    # Read operation
    li a0, 5
    ecall
    mv s1, a0
    # Read second operand
    li a0, 5
    ecall
    mv s2, a0
```

```
output:
    # Output the result
    li a0, 1
    mv a1, s3
    ecall

exit:
    # Exit program(necessary)
    li a0, 10
    ecall
```

Jupiter Ecalls

Sample Code

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

```
division_by_zero_except:
    li a0, 4
    la a1, division_by_zero
    ecall
    jal zero, exit
```

Sample Code

- You may finish operation implementations.

```
21 ##################################
22 #  TODO: Develop your calculator  #
23 #                                 #
24 ##################################
```

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 your program by running the following command:

```
$ jupiter [student_id].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: 10/02 23:59 (Wednesday)
- Please rename your program [student_id].s and upload it to NTU COOL.
  - For example, if your student id is b12345678, your program file name should be b12345678.s.

Jupiter Installation

- Please go to jupiter release page to download the latest version (v3.1) according to your operating system.

Jupiter v3.1 Latest

Change Log

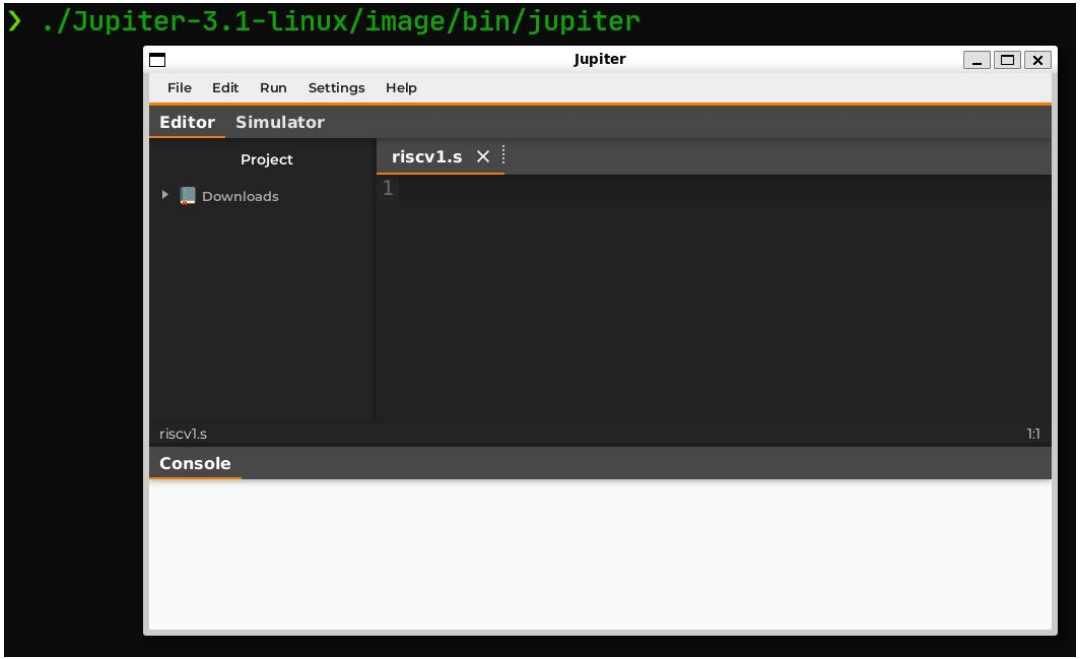
- Fix typo in Assembler.java, use data() instead of bss()
- Fix align method in LinkedProgram.java
- Remove explicit allocation with zeros in allocateBytesFromHeap for performance
- Add `-cache` option in CLI mode and Enable Cache Simulation setting in GUI mode
- Add deb package source files

Assets 6

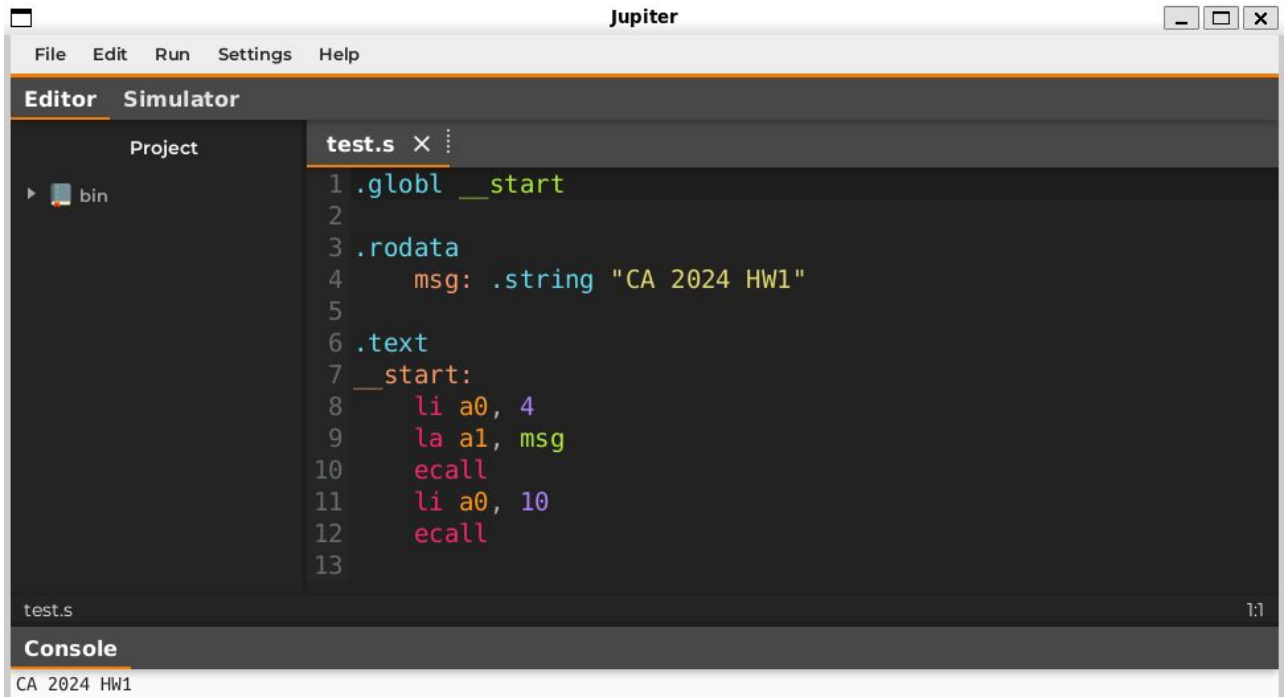
Jupiter-3.1-linux.zip	40 MB	Sep 4, 2019
Jupiter-3.1-mac.zip	35.7 MB	Sep 4, 2019
Jupiter-3.1-win.zip	37.6 MB	Sep 4, 2019
jupiter_3.1_amd64.deb	87.3 KB	Sep 4, 2019
Source code (zip)		Sep 4, 2019
Source code (tar.gz)		Sep 4, 2019

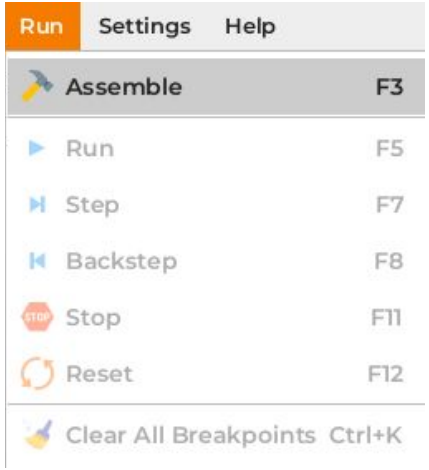
Jupiter GUI

- Extract the downloaded file, execute image/bin/jupiter.

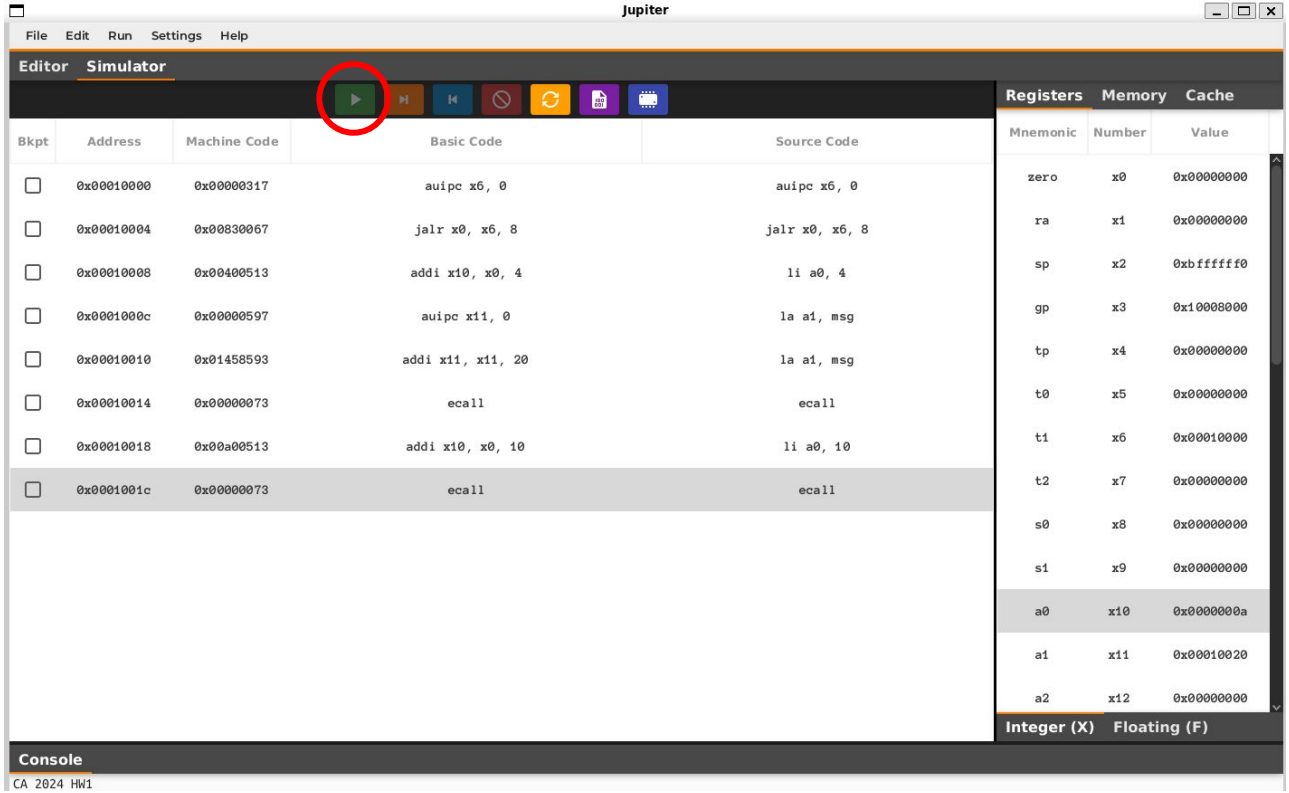


Jupiter GUI


Jupiter
File
Edit
Run
Settings
Help
Editor
Simulator
Project
bin
test.s
.globl __start
.rodata
msg: .string "CA 2024 HW1"
.text
__start:
li a0, 4
la a1, msg
ecall
li a0, 10
ecall
test.s
1:1
Console
CA 2024 HW1


Run
Settings
Help
Assemble F3
Run F5
Step F7
Backstep F8
Stop F11
Reset F12
Clear All Breakpoints Ctrl+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, msg
0x00010010 0x01458593 addi x11, x11, 20 la a1, msg
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
Integer (X) Floating (F)
Console
CA 2024 HW1

Jupiter CLI

- Extract the downloaded file, execute image/bin/jupiter with your code.

```
❯ ./jupiter hw1.s
10
0
10
20

Jupiter: exit(0)
```

Add

```
❯ ./jupiter hw1.s
10
3
0
division by zero

Jupiter: exit(0)
```

Division

```
❯ ./jupiter hw1.s
2
5
5
32

Jupiter: exit(0)
```

Power

Assignment Definition

Requirement

The calculator should support the following operations:

$+,\ -,\ \times,\ /,\ \min, \wedge,\ !$

Input format:	Output format:
A	[Result]
$operator$	
B	

$0 \le A, B \le 1024, op \in \{0, 1, 2, 3, 4, 5, 6\}$

If $op = 0$, calculate $A + B$ and output the result.

If $op = 1$, calculate $A - B$ and output the result.

If $op = 2$, calculate $A \times B$ and output the result.

If $op = 3$, calculate $A\ /\ B$ and output the result. (Quotient)

If $op = 4$, calculate minimum (A, B) and output the result.

If $op = 5$, calculate A^B and output the result.

If $op = 6$, calculate $A!$ and output the result. (In this case, $B = 0$)

If division by zero occurs, the program should print "$division\ by\ zero$".

(Don't worry about overflow or underflow.)

Note : No need to handle 0^0.

Input

Every input file has three lines. The first line contains a non-negative integer A, the second line contains a non-negative integer op, the third line contains a non-negative integer B, corresponding to the first operand, the operator, and the second operand.

Output

The output should contain only one integer that is the result of the input equation.