

CA2020 – HW2

RISC-V Assembly Code – A simple calculator

Description

- In this homework, you have to use a RISC-V simulator to develop a simple calculator program

Jupiter

- An open source RISC-V assembler and runtime simulator (32bit) [\[GitHub Repo\]](#)
- Jupiter is available on the CSIE Workstation, both CLI and GUI are supported
- To use the GUI interface, you can install Jupiter on your PC or use X11 to run the GUI of Jupiter installed on CSIE workstation



Jupiter tutorial – CLI

```
d08922025@linux1 [~] jupiter -h

  j u p i t e r

RISC-V Assembler & Runtime Simulator

usage: jupiter [options] <files>

[General Options]
-h, --help           show Jupiter help message and exit
-v, --version        show Jupiter version
-l, --license        show Jupiter license

[Simulator Options]
-b, --bare           bare machine (no pseudo-instructions)
-s, --self           enable self-modifying code
-e, --extract        assembler warnings are consider errors
-g, --debug          start debugger
    --start <label>  set global start label (default: __start)
    --hist <size>    history size for debugging

[Cache Options]
-c, --cache          enable cache simulation
    --assoc <assoc>  cache associativity as a power of 2 (default: 1)
    --block-size <size> cache block size as a power of 2 (default: 16)
    --num-blocks <num> number of cache blocks as a power of 2 (default: 4)
    --policy <policy> cache block replace policy (LRU|FIFO|RAND) (default: LRU)

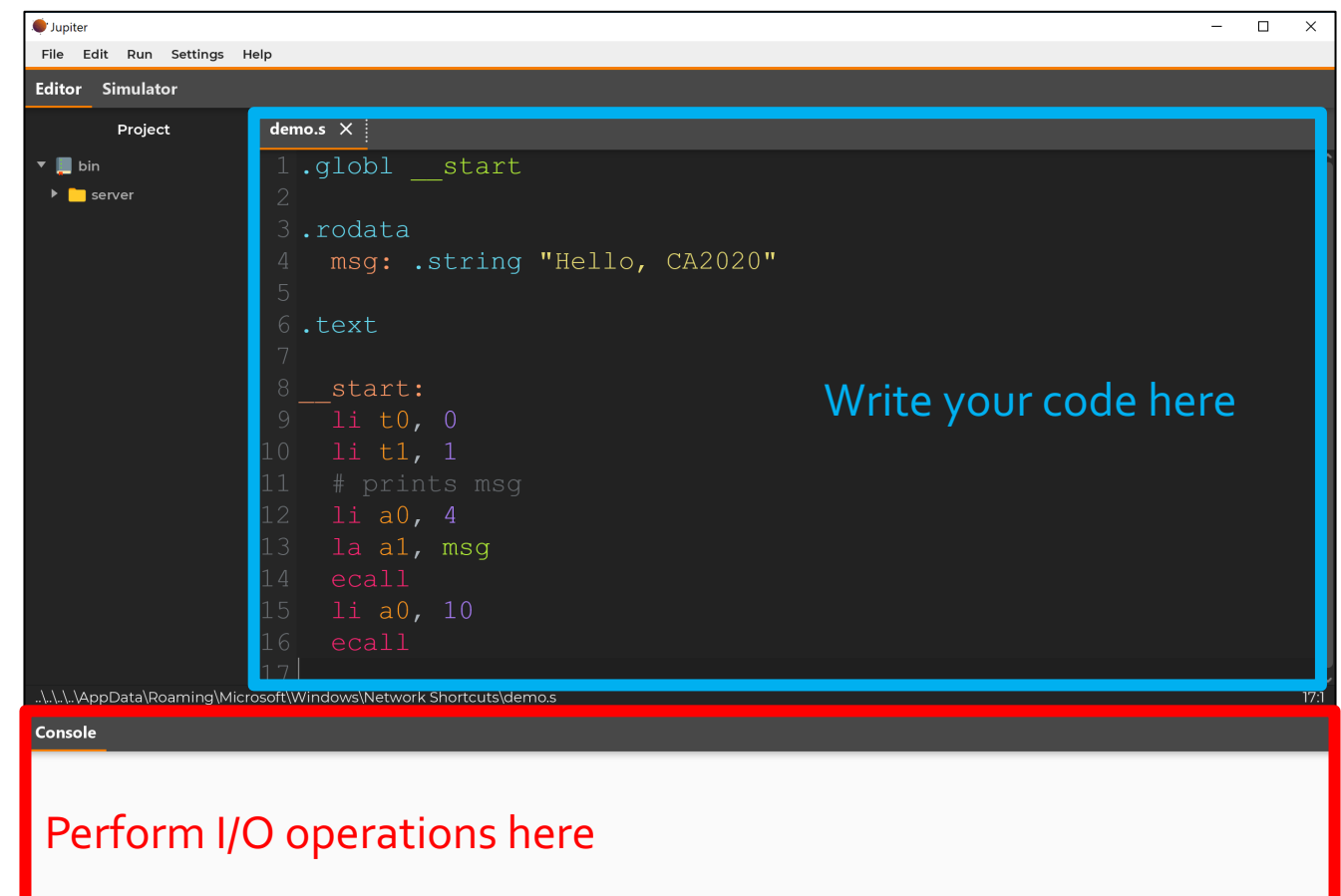
[Dump Options]
--dump-code <file>  dump generated machine code to a file
--dump-data <file>  dump static data to a file

Please report issues at https://github.com/andrescv/Jupiter/issues
```

```
d08922025@linux1 [~] jupiter demo.s
Hello, CA2020

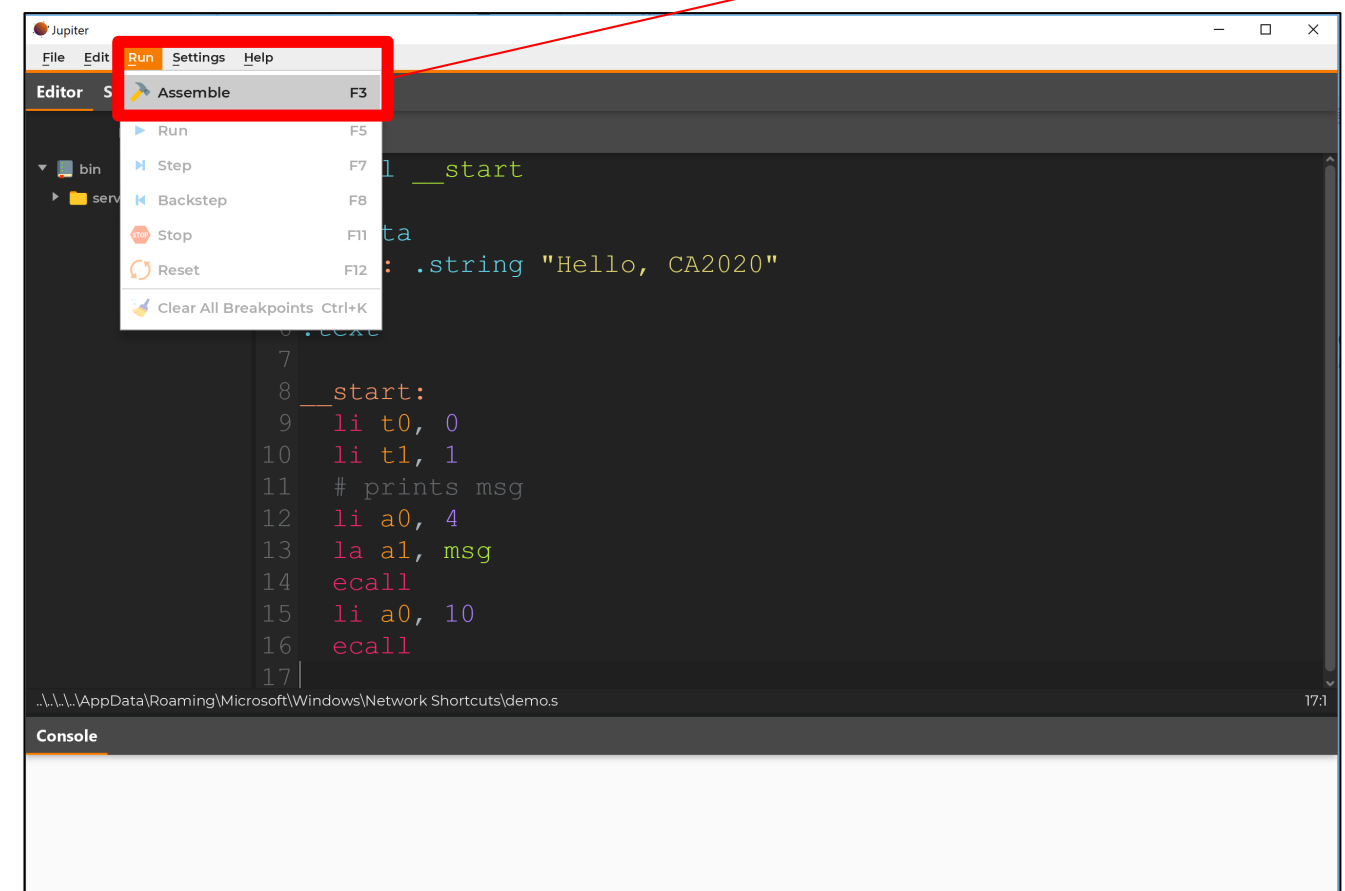
Jupiter: exit(0)
```

Jupiter tutorial - GUI



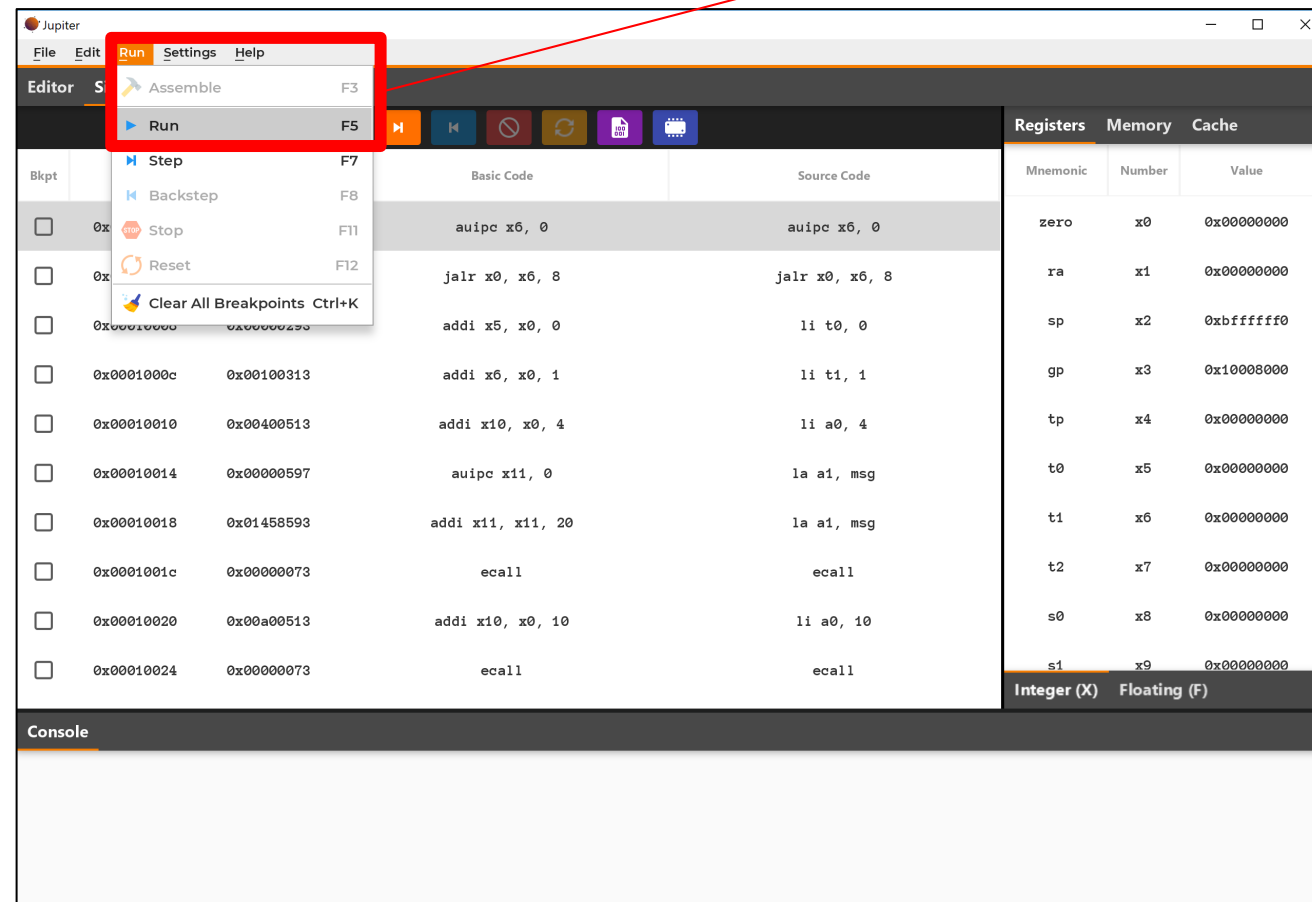
Jupyter tutorial - GUI

Click Run > Assemble

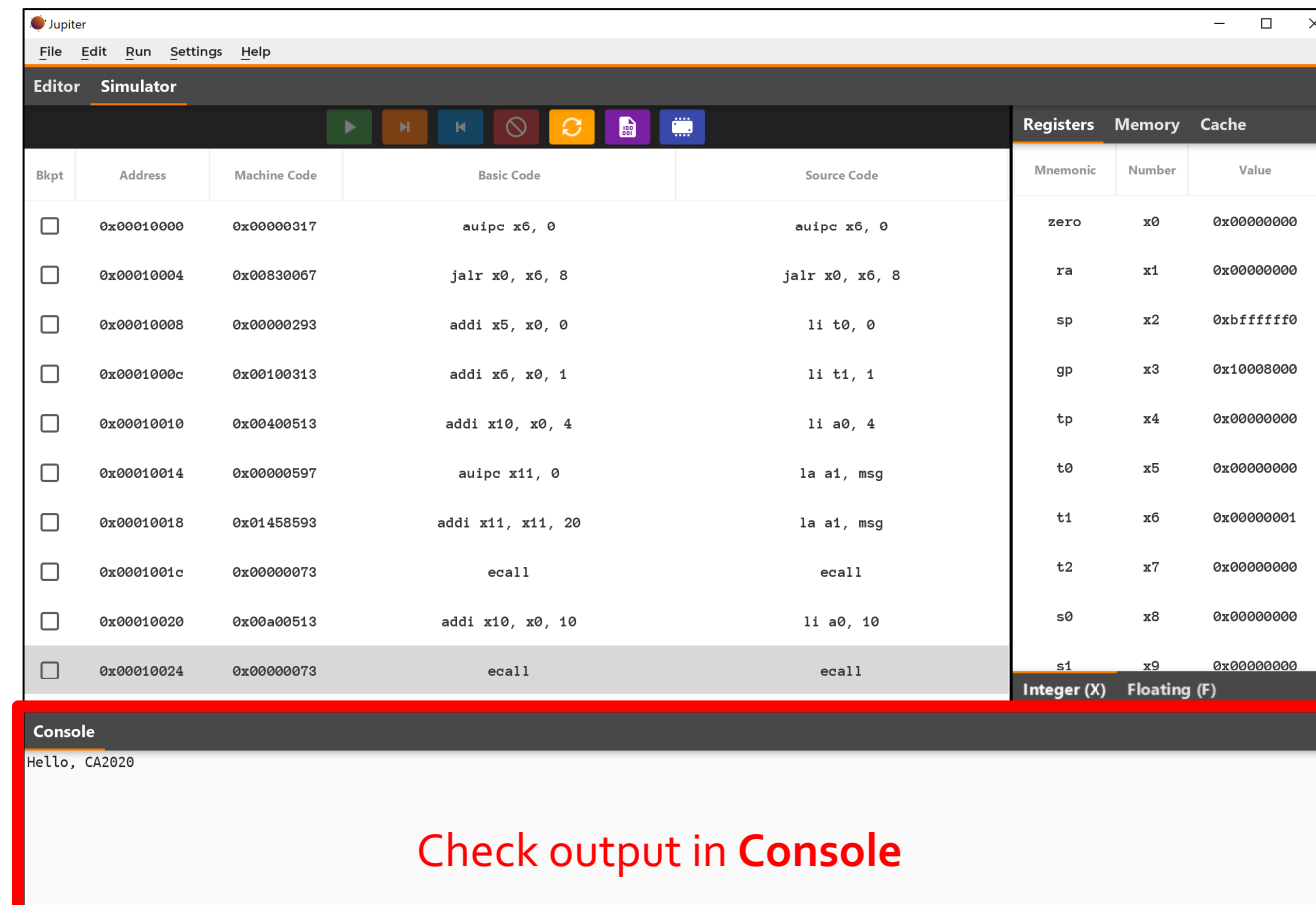


Jupiter tutorial - GUI

Click **Run** > **Run** to execute your code



Jupiter tutorial - GUI



The screenshot displays the Jupiter GUI interface. The top menu bar includes File, Edit, Run, Settings, and Help. Below the menu is a toolbar with icons for running, stepping through code, and saving. The main window is divided into two panes: the Editor and the Simulator. The Editor pane shows a table of assembly instructions with columns for Bkpt, Address, Machine Code, Basic Code, and Source Code. The Simulator pane shows a table of registers with columns for Mnemonic, Number, and Value. The Console pane at the bottom shows the output of the program.

Bkpt	Address	Machine Code	Basic Code	Source Code
<input type="checkbox"/>	0x00010000	0x00000317	auipc x6, 0	auipc x6, 0
<input type="checkbox"/>	0x00010004	0x00830067	jair x0, x6, 8	jair x0, x6, 8
<input type="checkbox"/>	0x00010008	0x00000293	addi x5, x0, 0	li t0, 0
<input type="checkbox"/>	0x0001000c	0x00100313	addi x6, x0, 1	li t1, 1
<input type="checkbox"/>	0x00010010	0x00400513	addi x10, x0, 4	li a0, 4
<input type="checkbox"/>	0x00010014	0x00000597	auipc x11, 0	la a1, msg
<input type="checkbox"/>	0x00010018	0x01458593	addi x11, x11, 20	la a1, msg
<input type="checkbox"/>	0x0001001c	0x00000073	ecall	ecall
<input type="checkbox"/>	0x00010020	0x00a00513	addi x10, x0, 10	li a0, 10
<input type="checkbox"/>	0x00010024	0x00000073	ecall	ecall

Mnemonic	Number	Value
zero	x0	0x00000000
ra	x1	0x00000000
sp	x2	0xbfffffff0
gp	x3	0x10008000
tp	x4	0x00000000
t0	x5	0x00000000
t1	x6	0x00000001
t2	x7	0x00000000
s0	x8	0x00000000
s1	x9	0x00000000

Console

Hello, CA2020

Check output in Console

Jupiter tutorial - debug

Set breakpoint

The Jupiter IDE interface is shown with the **Simulator** tab active. The main window displays a list of instructions with columns for **Bkpt**, **Address**, **Machine Code**, **Basic Code**, and **Source Code**. A breakpoint is set at address `0x00010008` (indicated by a checked checkbox). The **Registers** panel on the right shows the current state of registers, with a context menu open for the **zero** register showing display mode options: **Hex Display Mode** (selected), **Decimal Display Mode**, and **Unsigned Display Mode**.

Bkpt	Address	Machine Code	Basic Code	Source Code
<input type="checkbox"/>	0x00010000	0x00000317	<code>auipc x6, 0</code>	<code>auipc x6, 0</code>
<input type="checkbox"/>	0x00010004	0x00830067	<code>jair x0, x6, 8</code>	<code>jair x0, x6, 8</code>
<input checked="" type="checkbox"/>	0x00010008	0x00000293	<code>addi x5, x0, 0</code>	<code>li t0, 0</code>
<input type="checkbox"/>	0x0001000c	0x00100313	<code>addi x6, x0, 1</code>	<code>li t1, 1</code>
<input type="checkbox"/>	0x00010010	0x00400513	<code>addi x10, x0, 4</code>	<code>li a0, 4</code>
<input type="checkbox"/>	0x00010014	0x00000597	<code>auipc x11, 0</code>	<code>la a1, msg</code>
<input checked="" type="checkbox"/>	0x00010018	0x01458593	<code>addi x11, x11, 20</code>	<code>la a1, msg</code>
<input type="checkbox"/>	0x0001001c	0x00000073	<code>ecall</code>	<code>ecall</code>
<input type="checkbox"/>	0x00010020	0x00a00513	<code>addi x10, x0, 10</code>	<code>li a0, 10</code>
<input type="checkbox"/>	0x00010024	0x00000073	<code>ecall</code>	<code>ecall</code>

Mnemonic	Number	Value
zero	x0	0x00000000
ra	x1	0x00000000
tp	x4	0x00000000
t0	x5	0x00000000
t1	x6	0x00000001
t2	x7	0x00000000
s0	x8	0x00000000
s1	x9	0x00000000

Console: Hello, CA2020

Check registers
(right click to change display mode)

TODO: A simple calculator

- In this homework, you are asked to implement a simple calculator.
- The calculator should be able to output the correct results of $A \text{ op } B$
 - $-1024 \leq A, B \leq 1024; A, B, op \in \mathbb{Z}$
 - 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 $A \div B$ and output the result (integer division)
 - If $op \notin \{0, 1, 2, 3\}$, you should print "*op not supported*"
 - If divided by zero, you should print "*divided by zero*"
- Input format
 - Input file will contain 3 lines. Line1: A , Line2: op , Line3: B

Sample code

- Operations related to I/O have been implemented in the sample code. *A*, *op*, *B* will be stored in register *s0*, *s1*, *s2* respectively. You have to store the result to *s3* and jump to `result`. Please **DON'T** modify the code outside the TODO block.

```
24 #####
25 # TODO block: Add your code here
26 #
27 #####
```

- To print the “*zero division*” or “*op not supported*” message, just jump to `zero_div` or `op_not_supported`. You don't have to implement by yourself.

```
28 zero_div:
29     li a0, 4
30     la a1, zero_div_msg
31     ecall
32
33     j return
```

```
45 op_not_supported:
46     li a0, 4
47     la a1, op_not_supported_msg
48     ecall
49
50     j return
```

Sample output

```
d08922025@linux1 [~] jupiter hw2.s
3
0
-4
-1

Jupiter: exit(0)
d08922025@linux1 [~] jupiter hw2.s
3
1
-4
7

Jupiter: exit(0)
d08922025@linux1 [~] jupiter hw2.s
3
2
-4
-12

Jupiter: exit(0)
```

```
d08922025@linux1 [~] jupiter hw2.s
3
3
-4
0

Jupiter: exit(0)
d08922025@linux1 [~] jupiter hw2.s
3
3
0
divided by zero

Jupiter: exit(0)
d08922025@linux1 [~] jupiter hw2.s
3
5
-4
op not supported

Jupiter: exit(0)
```

Scoring

- We will judge the correctness of your code by running
`$ jupiter hw2.s < input_file` on CSIE workstation
- Don't worry about overflow/underflow, it won't happen in this homework
- 100 pts for calculator (20 testcase, 5 pts per testcase)
- 10 pts off per day for late submission
- You will get zero pts for plagiarism

Submission

- Due date: 2020/10/21 Wed. 14:20
- Please compress your homework into a *.zip file and upload to [NTUCOOL](#).
- After unzipping, the folder should have the following structure:
 - do8922025_hw2 (lowercase)
 - readme.txt (Write down what platform you use. Linux, windows, MacOS ...)
 - hw2.s (Remember to change the filename)