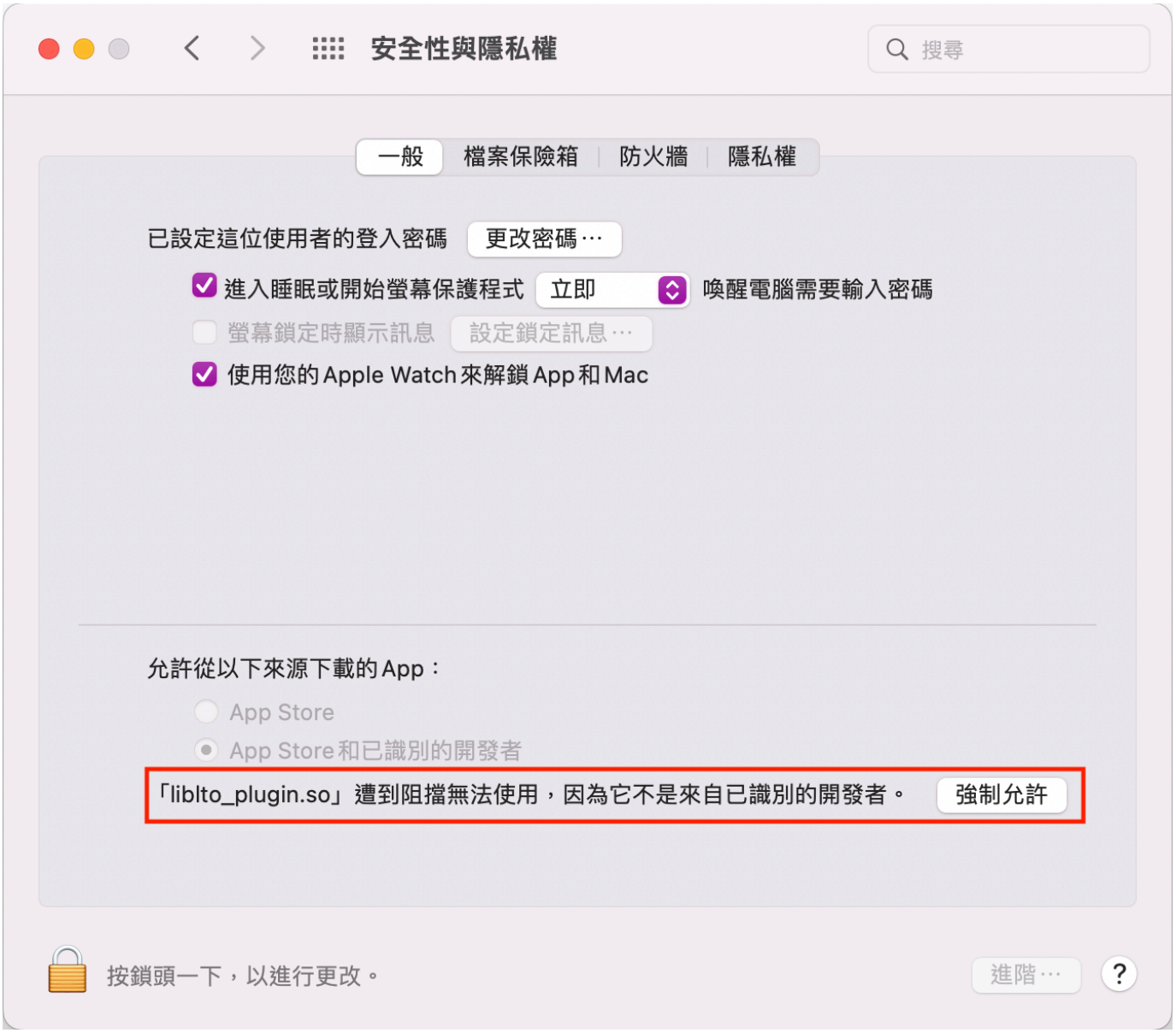
For MacOS

Please use HackMD to view the file.

1. Download Ripes

1. Download the release version of Ripes from Ripes Release
    - For example, `Ripes-v2.2.4-mac-x86_64.zip` is the current version for Mac platform.
2. Extract the file. There is one binary executable for Mac. If you want to execute the program, simply double-click it.
3. Note: The latest versions of macOS implemented a strict security mechanism. If you encounter the following error when launching a program. Go to the System Settings Panel and allow its execution. You may see this issue when launching the Ripes simulator (e.g., `Ripes-v2.2.4-mac-x86_64`). Several programs might also trigger the error when running the toolchain, e.g., `riscv64-unknown-elf-gcc`, `as`, `ld`, `cc1`, `collect2`, `liblto_plugin.so`, etc. It is totally fine if you do not encounter the error. Move on and good luck.


macOS 無法驗證「libIto_plugin.so」的開發者。您確定要打開它嗎？
若您打開此 App 將會覆蓋系統安全性，這可能使您的電腦和個人資訊暴露於惡意軟體，其可能會損害您的 Mac 或危害您的隱私權。
Safari 在今天下午 4:54 從 static.dev.sifive.com 下載此檔案。
打開
丟到垃圾桶
取消



2. Install Toolchain

1. Please download the SiFive RISC-V toolchain from SiFive github.
    - For example, `riscv64-unknown-elf-toolchain-10.2.0-2020.12.8-x86_64-apple-darwin.tar.gz` is the current version for MacOS.
2. Open your terminal and use **tar** command to extract the file to a folder.
    - For example :

    ```
    tar zxvf riscv64-unknown-elf-toolchain-10.2.0-2020.12.8-x86_64-apple-darwin.tar.gz
    ```

3. Locate the folder that contains **riscv64-unknown-elf-gcc**. Set this folder as `$RV64_GCC_PATH` and add it to the search path (see the following step).

- For example :
  Assume the folder is `~/HW5/riscv64-unknown-elf-toolchain-10.2.0-2020.12.8-x86_64-apple-darwin/bin`.
  You can use the following commands to set the variable and add it to the search path. You may also add them to your login shell.

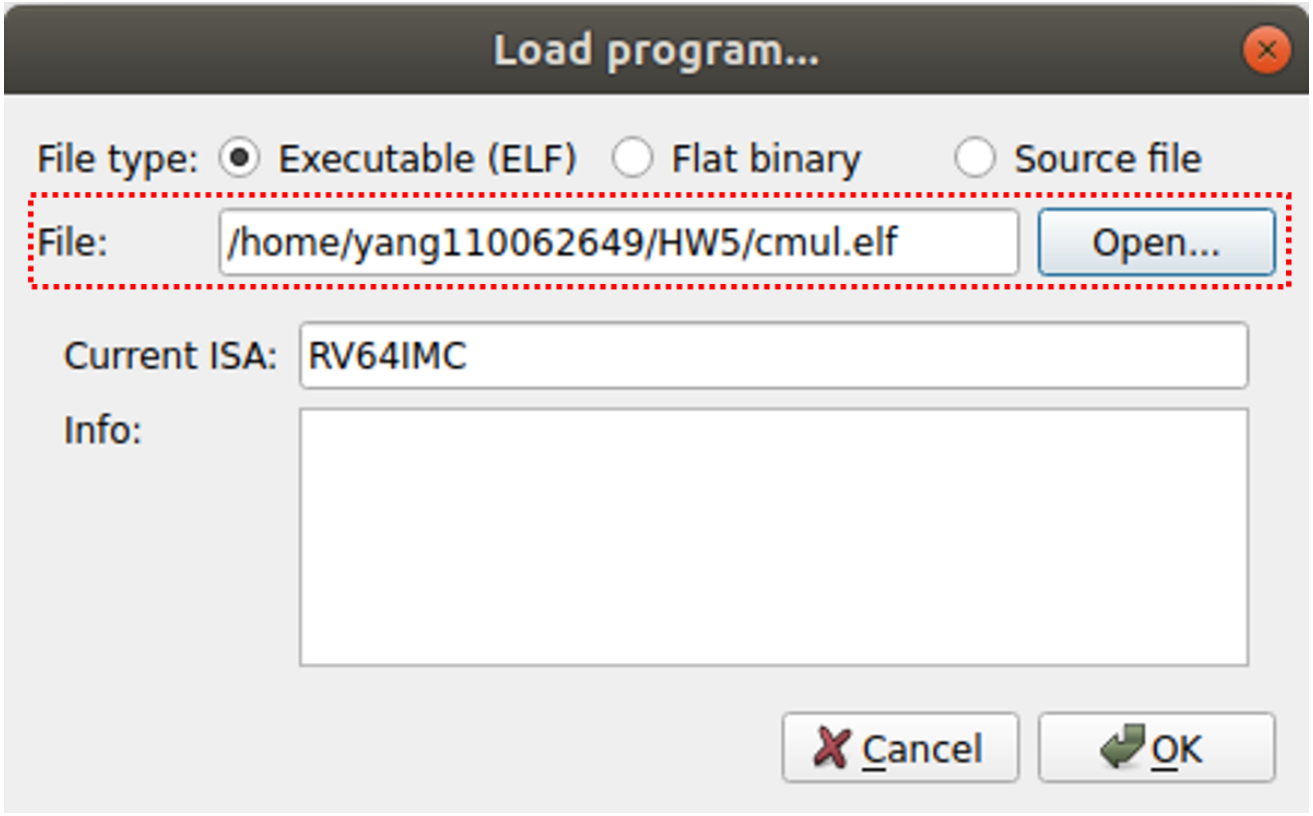
```
 RV64_GCC_PATH=~/HW5/riscv64-unknown-elf-toolchain-10.2.0-2020.12.8-x86_64-apple-darwin/bin
 export PATH=$PATH:$RV64_GCC_PATH
```

3. Test Setup

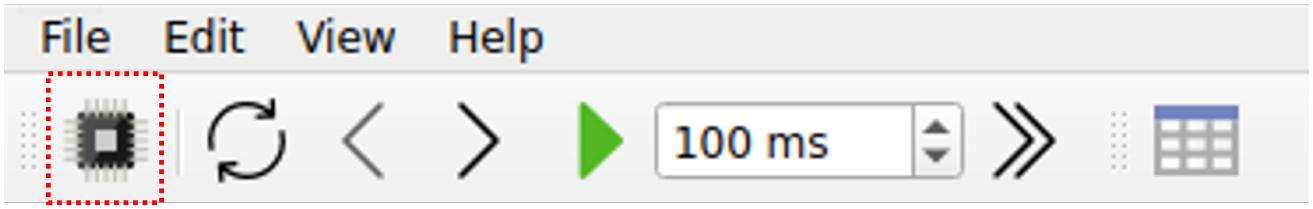
1. Download cmul.S from EECLASS for HW5 and open a terminal. Generate an rv64im executable by the RISC-V compiler.
   - For example :

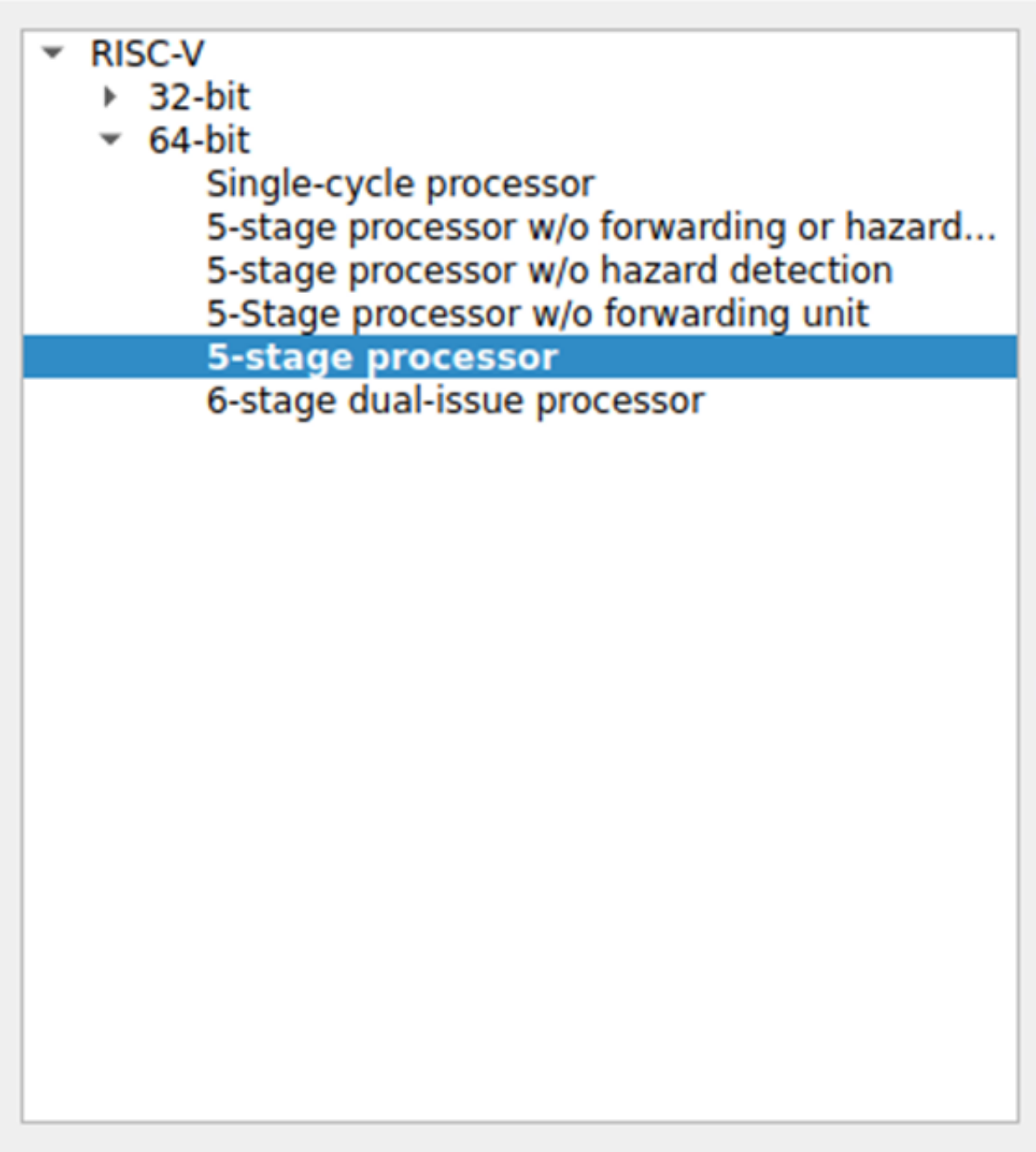
```
 $RV64_GCC_PATH/riscv64-unknown-elf-gcc -march=rv64im -mabi=lp64 -s -static -nostdlib -o cmul.elf cmul.S
```

     Parameter Reference:
     - "-march=rv64im" to use an 64-bit ISA version with integer ("i") and multiply ("m") supports.
     - "-mabi=lp64" to specify the language data model. In this setup, long ("l") and pointer are all 64 bits.
     - "-s" to strip symbols from binary.
     - "-static" to link statically to produce a complete executable.
     - "-nostdlib" do no use stdlib.
     - "-o" specify output name.
2. Start Ripes GUI by double clicking the program. Then, select `File > Load Program`.
   Use Open to search for cmul.elf.
   - For example :



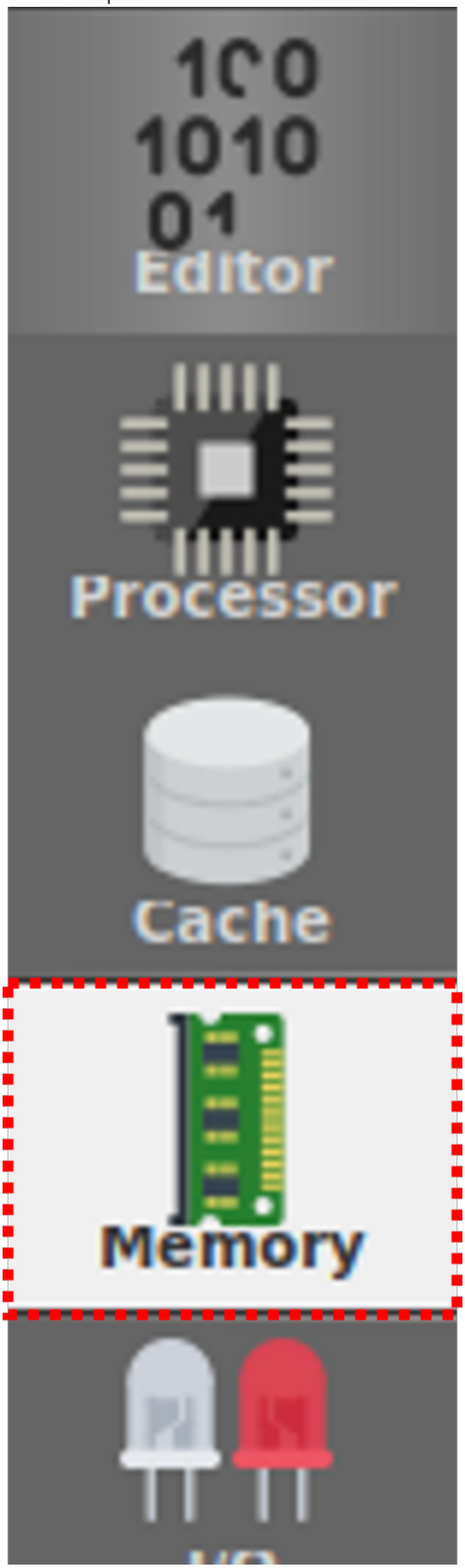
- If you found that current ISA in `Load Program` is not RV64, please click on `processor selection button` at the top-left and then select the `RISC-V > 64bit > 5-stage processor` with extension `M&C`

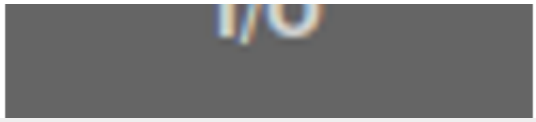




3. Select `Memory Tab`. In `Memory Tab`, we check the `data segment base memory address` to calculate the `global pointer (x3)`.

- For example :





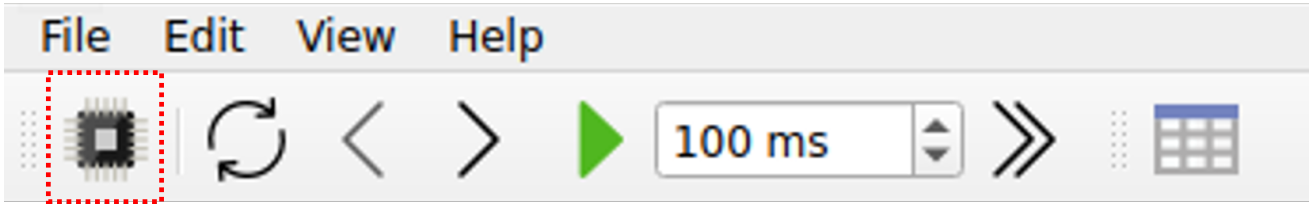
Memory map

Name	Size	Range
.shstrtab	48	0x0000000000000000 - 0x0000000000000030
.text	120	0x00000000000100b0 - 0x0000000000010128
.data	35	0x0000000000011130 - 0x0000000000011153
.sdata	0	0x0000000000011153 - 0x0000000000011153

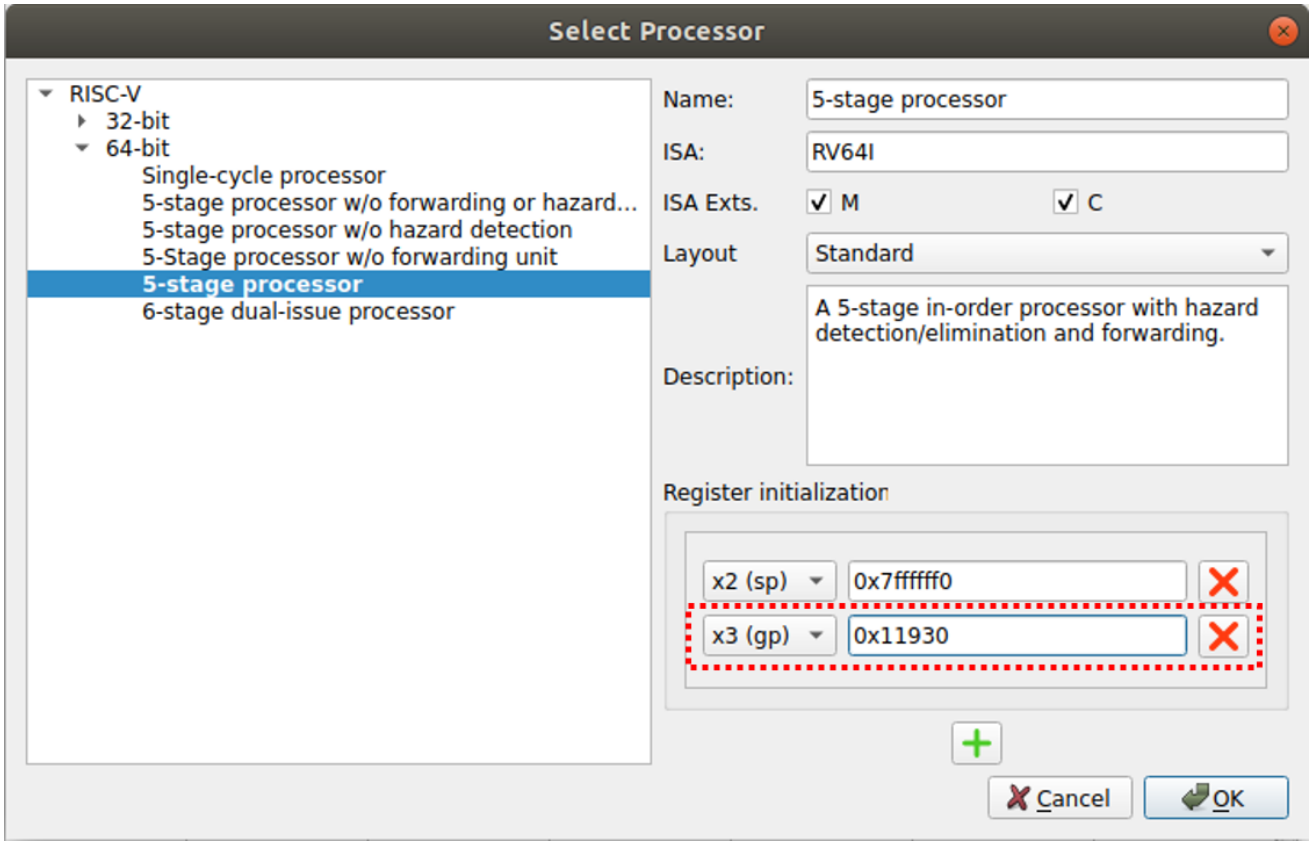
Here we find that the base is `0x11130`. Since the assembler assumes the `global pointer (x3)` to be set at the `base + 0x800`, we will use `0x11130+0x800=0x11930` to set up `global pointer`.

4. Set up the `global pointer` :

   - **Method 1** :
     Click on `processor selection button` at the top-left.

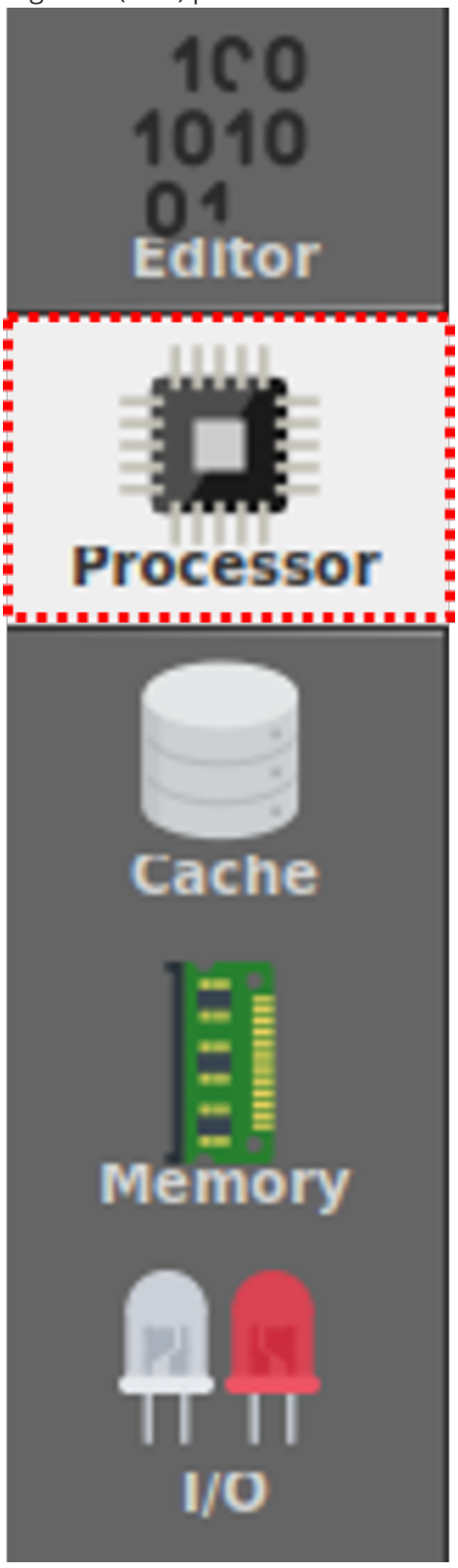


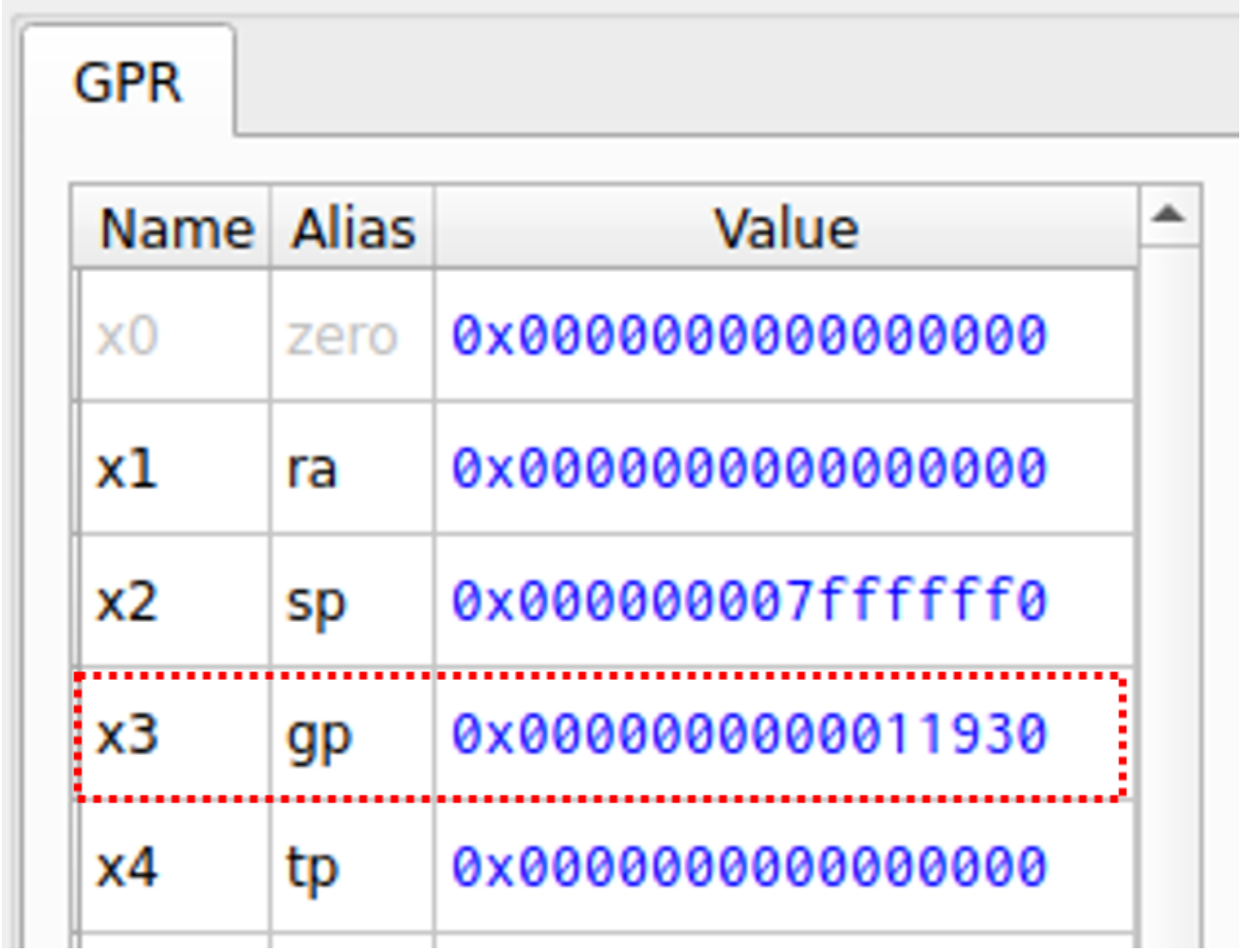
Then, set up the `global pointer` as 0x11930.



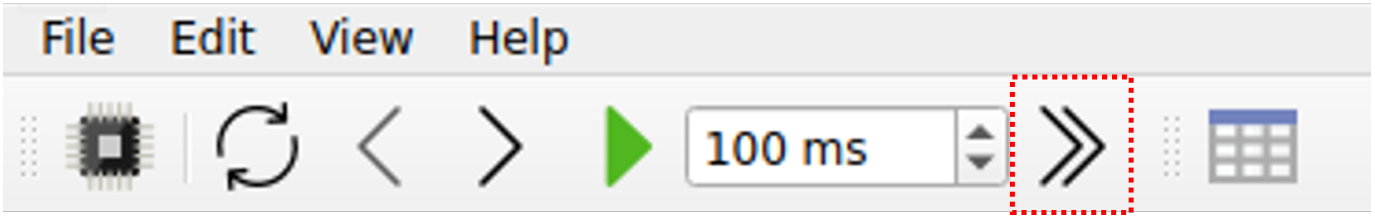
After we set the `global pointer` for the processor, **we need to reload the cmul.elf again (do step 2 again).**

- **Method 2** :
  Click on `processor Tab` and modify the global pointer to 0x11930 directly on general purpose registers (GPR) panel.

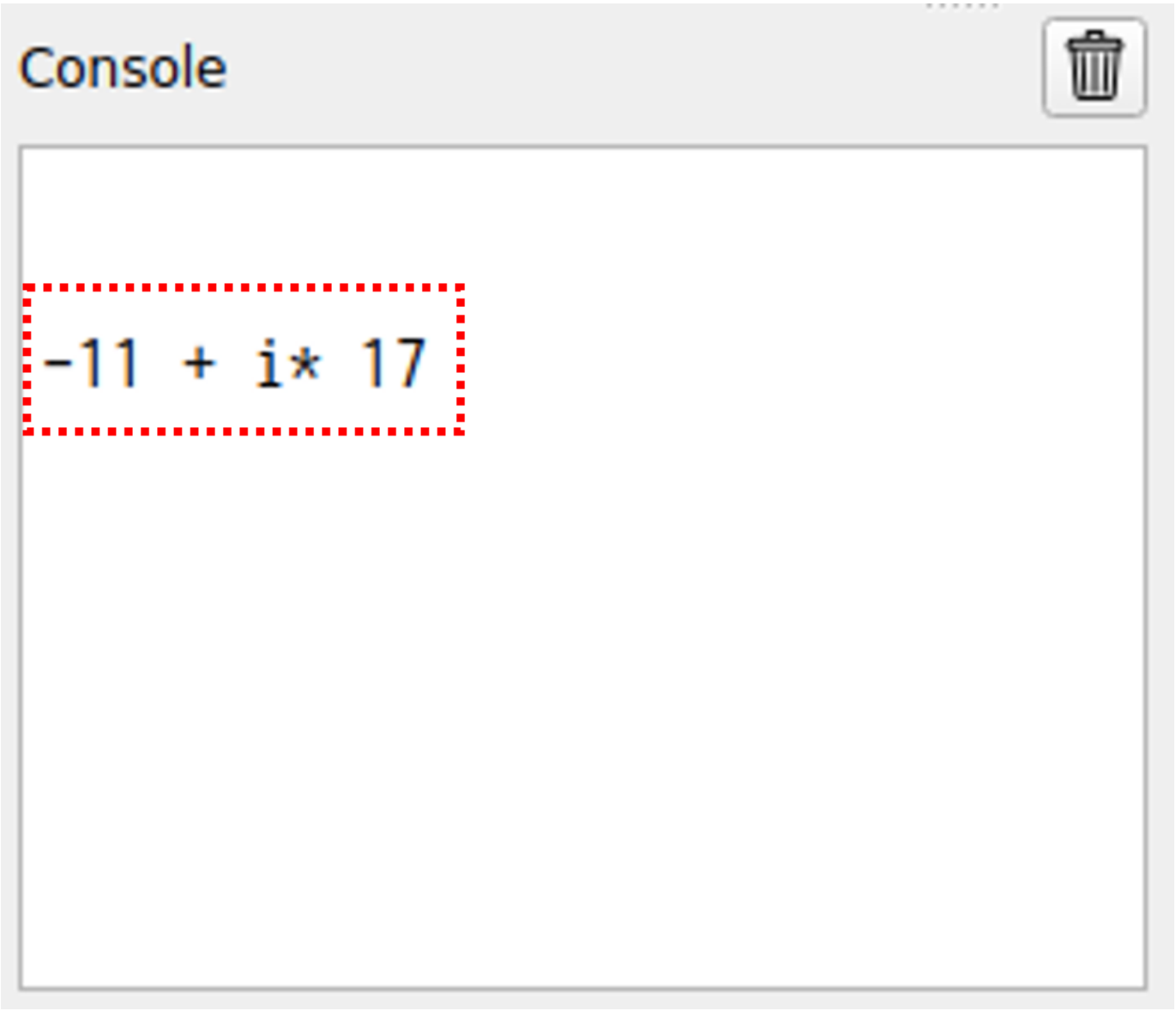




5. Click `>>` on the top-right to run and simulate the binary in Ripes without GUI updates.



We should see the program finishes and prints `11 + i* 17` on the console.



- Note: We need to set the global pointer for each new elf program load (since the data segment changes according to the text segment).