


Exercise Sheet 2: Basic RISC-V Assembly Language Programming

Before starting call

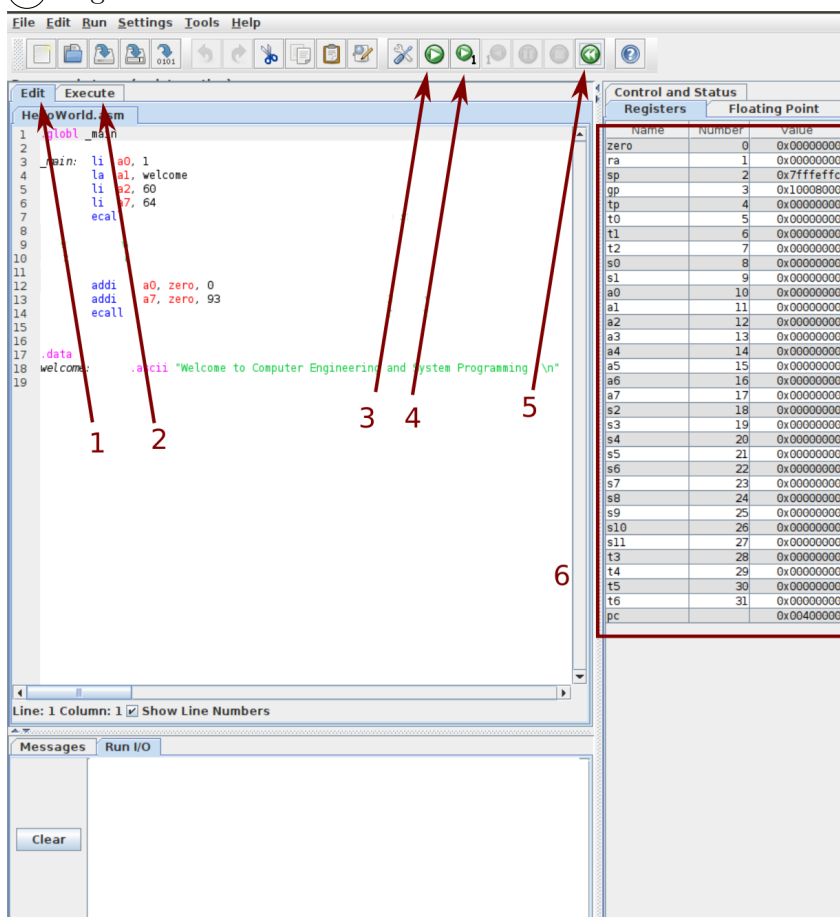
`cesp_setup` 

Material for this exercise is in `$HOME/cesp_course/exercises/02_basic_assembler`

Task 1. Run your first RISC V program.

These are the control elements of RARS that you will need for your first program:

- ① Edit tab
- ② Execute tab
- ③ Run button
- ④ Run1 button
- ⑤ Reset button
- ⑥ Register values



1. Open Rars instruction level simulator from the terminal:
`rars`

2. Open `hello_world.asm`
3. Select Run→Assemble from the context menu.
4. Click Run1 button multiple times until a message appears in the Run I/O window ... Congrats ... You just execute your first RISC-V program.
5. Modify the program to print **Hello World**, then select Run→Assemble, press the Run button.
6. If the Run I/O window displays the **Hello World** message you are done!


Task 2. Unit Test of Simple Addition

1. Open RARS

rars 

2. Open the file `simple_addition.asm` and try to understand what it does.
3. Throughout the course we will strongly use unit tests.
A unit test executes a program and compares its results against predefined values.
The tool `cesp_utest` will help you here. It executes the content in `.json` files.

4. Familiarize yourself with the unit test file

`$HOME/cesp_course/exercises/01_get_tools_running/hello_world.json` 

What it does:

- Execution of asm file:

`java -jar /home/cesp/cesp_course/tools/rars/rars.jar a0 a2 a7 hello_world.asm`




This is the same thing you did before by pressing the *Run* button. The parameters `a0 a2 a7` specify which register values to print after the execution.

- Comparison of output of program against the value specified in the `expected_result` field of the `.json` file.

5. Create a unit test `simple_addition.json` file that checks for the correct result.

6. Run

`cesp_utest` 

Once your unit test succeeds you are done.

Function Calls

In mathematics, the Fibonacci numbers are a sequence where each number is the sum of the two preceding numbers (starting with two ones):

$$1, 1, 2, 3, 5, 8, 13, 21, 34, \dots \quad (1)$$

Task 3. Iterative implementation of Fibonacci Sequence

Implement a function in RISC-V assembly language that calculates the Fibonacci numbers in an iterative fashion and run it in the RARS simulator. Follow the steps in Fib01.asm from the course website that helps you to implement the functionality defined by the following C code.

Please use comments to describe the functionality of your assembler code.

```
1 void fib(int* result, int n){
2     if (n > 0)
3         result[0]=1;
4     if (n > 1)
5         result[1]=1;
6     for(int j=2; j<n; j++){
7         result[j] = result[j-2] + result[j-1];
8     }
9 }
10 int main(){
11     int fibonacci[20];
12     fib(fibonacci, 20);
13     for(int i=0; i<20; i++){
14         printf("%d\n", fibonacci[i]);
15     }
16 }
```

Use the file fib01.json to verify your result using the `cesp_utest` command.

Task 4. Recursive implementation of Fibonacci Sequence

Implement a function in RISC-V Assembler that calculates the Fibonacci sequence in a recursive fashion and run it in the RARS simulator. Follow the steps in fib02.asm that help you to implement the functionality defined by the following C code.

Please use comments to describe the functionality of your Assembler code.

Hint: Use the Stack.

```
1 int fib(int n) {
2     if (n <= 1)
3         return n;
4     return fib(n-1) + fib(n-2);
5 }
6 int main() {
7     printf(" %d ", fib(20));
8 }
```

Use the file `fib02.json` to verify your result using the `cesp_utest` command.