RISC-V Tools and ISA Simulator Tutorial

Example Program

- Make sure you have done all the steps in "*environment_setup.pptx*".

- In "ee3450_pa1/example", there is an example program as your reference.

- The program sums up the first two elements in an array, and stores the value to the third element.

Compile RISC-V Code

- We provide Makefiles for each part to simplify the building process. You can generate binary files using **make** command.

```
$ make
```

- To delete the compiled files:

```
$ make clean
```

Compile RISC-V Code

- The make process will generate some files: **`example.riscv`, `example.riscv.dump`, `crt.o`, `syscalls.o`, `etc.`**

- If you want to understand the detailed compiling process, you can refer to the Makefile or RISC-V tools and RISC-V toolchain github repository.

Using RISC-V ISA simulator

- The RISC-V ISA simulator is called **spike**. To run simulation:

```
$ spike example.riscv
```

- Since we don’t print anything in the program, the program just ends silently. To see what each instruction do and registers and memory values, we can use the debug mode:

```
$ spike –d example.riscv
```

Using RISC-V ISA simulator

- Now spike is waiting for commands. To see the content of a0, we can use the command:

```
: reg 0 a0
0x0000000000000000
```

- The 0 after **`reg`** mean core 0 (spike supports multicore simulation).

- We can get the register content is zero.

Using RISC-V ISA simulator

- We can refer to "*example.riscv.dump*" to see the address of instructions.

```
. . .
0000000080001746 <main>:
    80001746: 1161                  addi   sp,sp,-8
    80001748: e006                  sd     ra,0(sp)
    8000174a: 00000517              auipc  a0,0x0
    . . .
    80001762: 0121                  addi   sp,sp,8
    80001764: 8082                  ret
. . .
```

Using RISC-V ISA simulator

- We can also check the .data part to see the address of array.

```
. . .
0000000080001938 <array>:
    80001938: 0001                  nop
    8000193a: 0000                  unimp
    8000193c: 0000                  unimp
    8000193e: 0000                  unimp
    80001940: 0002                  c.slli64      zero
. . .
```

Using RISC-V ISA simulator

- If you want to jump to an instruction to see the result after execution, you can use **until** command:

```
: until pc 0 80001764
```

- The 0 after **`pc`** means core 0.

- This will take you to the end of the main function (see page 7).

Using RISC-V ISA simulator

- To check the content of memory, you can use **mem** command:

```
: mem 80001938
0x0000000000000001
: mem 80001940
0x0000000000000002
: mem 80001948
0x0000000000000003
```

- Note that the address and value are both in hexadecimal.

Using RISC-V ISA simulator

- To end the simulation from the debug prompt:

```
: q
```

- If you are interested in how spike works or complete debug guild, please refer to the github repository of spike.