Assembly programming using RISCV-ISA simulator tool chain

1. Testing the RISCV-Tool chain

All the lab machines have been installed with the RISC-V tools. To check if the installation is correct, please run the following code to check the spike RISCV ISA simulator.

1) mkdir riscv-tests
2) cd riscv-tests
3) echo -e '#include <stdio.h>\n int main(void) { printf("Hello world!\\n"); return 0; }' > hello.c
4) riscv64-unknown-elf-gcc -o hello hello.c
5) spike pk hello

Alternately you may install the riscv-tool chain on your laptops by following the steps at the link https://github.com/riscv/riscv-tools.

2. Getting started with RISC-V assembly programs

a. Create a file called link.ld in the current working directory with the following contents.

```
OUTPUT_ARCH( "riscv" )
 SECTIONS
    {
     . = 0x200;
     .text : { *(.text) }
     .data ALIGN(0x1000) : { *(.data) }
     .bss : { *(.bss) }
     _end = .;
    }
```

b. All your RISC-V asm programs that you code in this tutorial will have the following code at the beginning in the your_file.s

```
.globl _start;
_start:

    # Put your own code here


_exit:
```

c. To compile please use **:** riscv32-unknown-elf-gcc -nostdlib -nostartfiles -march=rv32i -T./link.ld your_file.s

d. To execute please use: spike pk a.out

3. Programming questions

Once the hello world program is working, the goal of this tutorial is to write the below programs using RISCV ISA in assembly language.

a) Compute the sum of "n" natural numbers assuming register x5 has the value n. Let the result of the sum be in register x6. Test your assembly program for different values of n (i) 1 (ii) 1024 (iii) 2048 and (iv) 4096 .

   Rename these programs as sum_1.s, sum_1024.s, sum_2048.s, and sum_4096.s respectively before submission.

b) The Fibonacci numbers are the numbers in the following integer sequence, called the Fibonacci sequence, and characterized by the fact that every number after the first two is the sum of the two preceding ones: 1,1,2,3,5,8,13,21,34,55,89 …

   Assume an integer "n" is in register x5 and compute the last integer in a sequence of n Fibonacci numbers. Return the last integer in register x6. Test your assembly program for different values of n (i) 1 (ii) 2 (iii) 10 (iv) 15 and (v) 20

   Rename these programs as fib_1.s,fib_2.s, fib_10.s, fib_15.s and fib_20.s respectively before submission.

c) Compute the greatest common divisor (gcd) of two integers, assuming the two integers are in registers x5 and x6. Put the result of the computation in x7. Test your assembly program for different values of x5 and x6 for the following: (i) 0,5 (ii) 5,0 (iii) 3, 5 (iv) 256,2048 (v) 4095, 15

   Rename these programs as gcd_0_5.s, gcd_5_0.s, gcd_3_5.s, gcd_256_2048.s and gcd_4095_15.s respectively before submission.

4. Moodle submission

Tar zip the folder with all the appropriately named assembly programs and submit the zip file with your roll number as the zip file name.

i. **Challenge question** Currently the program after executing prints the contents of all the registers and the statement "An illegal instruction was executed!". One easy way of overcoming this statement is by looping back to the last instruction, by adding
exit: jal exit
to the end of the assembly program.

Other than looping back to the last instruction any other interesting ways of overcoming the above illegal instruction message are welcome. However, the program does execute and change the register values despite this message and you can check the contents of the registers for your program correctness.