

Hands On Session o Gem5 Simulator

- **Step 1 : First of all how to access HPC cluster ?**

Open terminal and run this command:

`ssh -X riscv-sim@10.0.0.153`

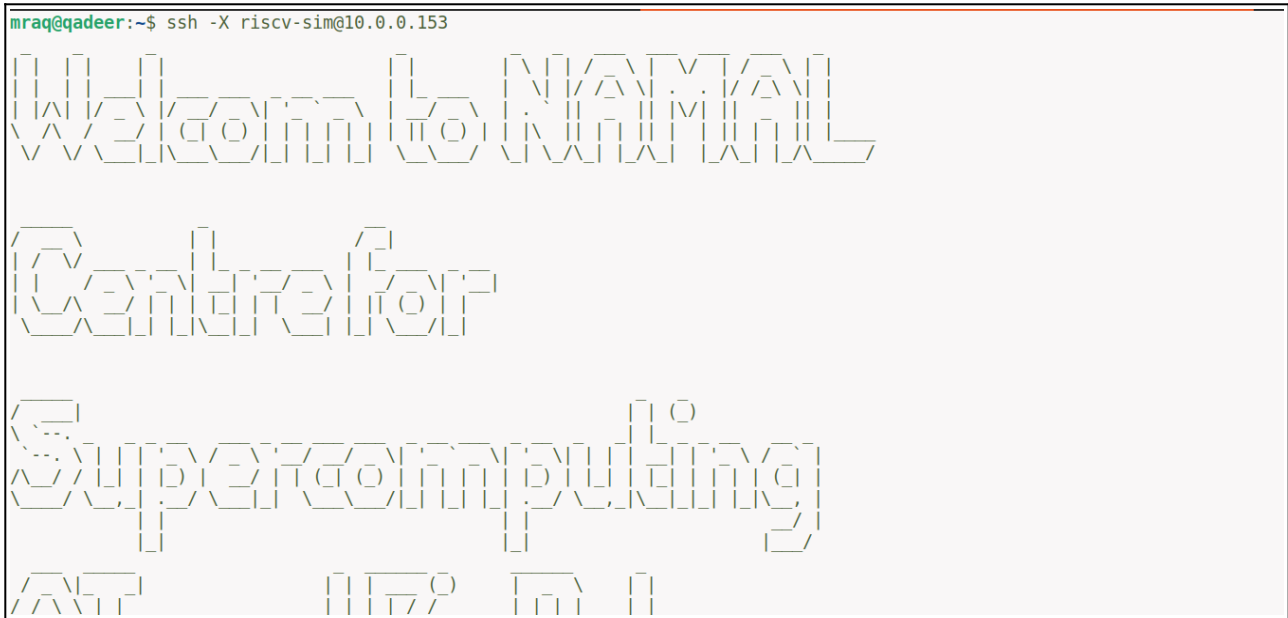


Figure 1: Interface of HPC Cluster

- **Step 2: Run these commands in the terminal:**
`cd demo_simulators/`
- **Step 3: Create directory of your name:**
`mkdir qadeer`
- **Step 4: Write C code of hello world**
`nano helloworld.c`
- **Step 5: Create riscv elf format binary using gcc**
`riscv64-unknown-elf-gcc -o hello helloworld.c`

Summery of the commands above is below:

```
riscv-sim@ucerd-hpc:~$ cd demo_simulators/
riscv-sim@ucerd-hpc:~/demo_simulators$ mkdir qadeer
riscv-sim@ucerd-hpc:~/demo_simulators$ cd qadeer
riscv-sim@ucerd-hpc:~/demo_simulators/qadeer$ nano helloworld.c
riscv-sim@ucerd-hpc:~/demo_simulators/qadeer$ riscv64-unknown-elf-gc
riscv64-unknown-elf-gcc      riscv64-unknown-elf-gcc-nm      riscv64-unknown-elf-gcov-dump
riscv64-unknown-elf-gcc-13.2.0  riscv64-unknown-elf-gcc-ranlib  riscv64-unknown-elf-gcov-tool
riscv64-unknown-elf-gcc-ar      riscv64-unknown-elf-gcov
riscv-sim@ucerd-hpc:~/demo_simulators/qadeer$ riscv64-unknown-elf-gcc -o hello helloworld.c
riscv-sim@ucerd-hpc:~/demo_simulators/qadeer$
```

Figure 2: Summary of commands

- **Step 6 :** Create a python script. Its also provided in *demo simulators* directory with complete explanation. You can copy it
cp scriptgem5.py yourdirectory
- **Step 7 :** Run this command and set a path to your binary
nano script.py
- **Step 8:** Run it on gem5
gem5.opt -d output_directory script.py
- **Step 9:** Go to output_directory and see the results

Summary of commands is shown below

```
riscv-sim@ucerd-hpc:~/demo_simulators$ gem5.opt -d output_directory script.py
gem5 Simulator System.  https://www.gem5.org
gem5 is copyrighted software; use the --copyright option for details.

gem5 version 23.0.0.1
gem5 compiled Nov 10 2023 15:00:21
gem5 started Aug 10 2024 13:52:43
gem5 executing on ucerd-hpc, pid 4944
command line: gem5.opt -d output_directory script.py
```

```
riscv-sim@ucerd-hpc:~/demo_simulators$ ls
binary binary_search binary_search.c m5out output_directory qadeer script.py test
riscv-sim@ucerd-hpc:~/demo_simulators$ cd output_directory/
riscv-sim@ucerd-hpc:~/demo_simulators/output_directory$ ls
config.dot config.dot.pdf config.dot.svg config.ini config.json stats.txt
riscv-sim@ucerd-hpc:~/demo_simulators/output_directory$
```

Figure 1 : Summary of commands

- **Step 10 :**Open the config.pdf file using
xdg-open config.dot.pdf
- **Step 11:** Open the stats.txt file using *cat stats.txt*

Tasks

Task 1: Configure two rv64 based system using O3CPU and TimingSimpleCPU having same set of configurations and workload. Measure their performance and observe which one has higher performance? **Hint:** measure execution time

Script With explanation:

```
import os
import m5
from m5.objects import *

# Create the system object that will represent the entire simulated computer.
system = System()

# Create a clock domain for the system with a clock speed of 1 GHz.
system.clk_domain = SrcClockDomain()
system.clk_domain.clock = "1GHz"
system.clk_domain.voltage_domain = VoltageDomain()

# Set the memory mode to "timing," which models the time it takes for memory operations.
system.mem_mode = "timing"

# Define the memory range to be 512MB.
system.mem_ranges = [AddrRange("512MB")]

# Instantiate a RISC-V TimingSimpleCPU, a simple CPU model for RISC-V architecture.
system.cpu = RiscvTimingSimpleCPU()

# Create a system-wide crossbar (membus) to connect various components.
system.membus = SystemXBar()

# Connect the CPU's instruction and data cache ports to the system bus.
system.cpu.icache_port = system.membus.cpu_side_ports
system.cpu.dcache_port = system.membus.cpu_side_ports

# Create an interrupt controller for the CPU, necessary for handling interrupts.
system.cpu.createInterruptController()

# Instantiate a memory controller and assign it DDR3 memory with a specific configuration.
system.mem_ctrl = MemCtrl()
system.mem_ctrl.dram = DDR3_1600_8x8()
```

```
system.mem_ctrl.dram.range = system.mem_ranges[0]  
system.mem_ctrl.port = system.membus.mem_side
```