Computer Architecture (CSE511/ ECE 511)
Assignment 1

1. Create a configuration script to simulate the system given in the figure below using two different CPU models: **RiscvTimingSimpleCPU()** and **RiscvO3CPU()**. Run a benchmark application, " qsort_small from MiBench benchmark suite [1]" in system emulation mode. Unless otherwise specified, the configuration of caches is as given; L1 Data Cache (size = 16kB, Associativity = 2, latencies = 2); L1 Inst Cache (size = 16kB, Associativity = 2, latencies = 2); L2 Cache (size = 256kB, Associativity = 4, latencies = 10).

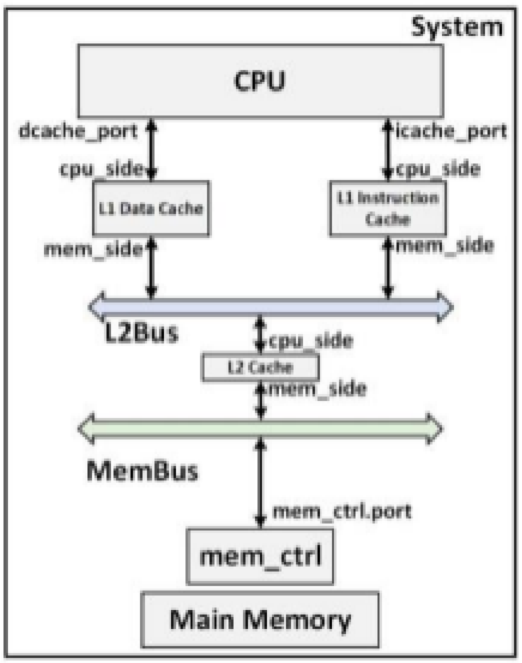


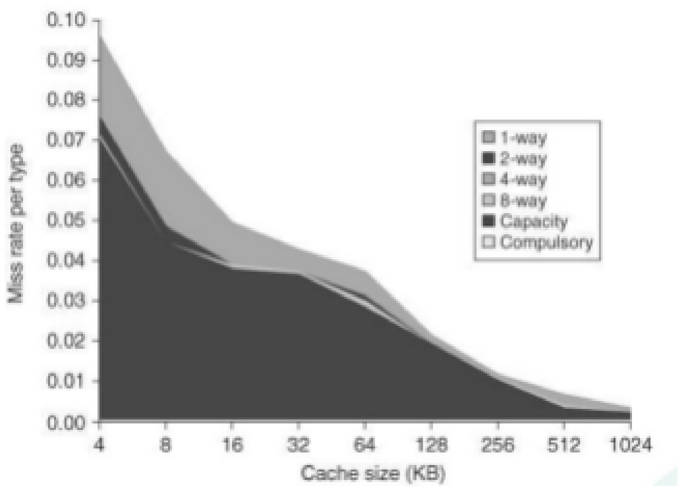
Fig. 1: A system architecture with a processor, two cache levels, and main memory. The memory hierarchies are connected by buses.

Note:

1. For this assignment, since you are using RISC-V ISA, you need to build gem5 in RISC-V ISA using "**python3 `which scons` build/RISCV/gem5.opt -j$(nproc)**"
2. For running gem5 " **build/RISCV/gem5.opt configs/tutorial/two_level.py**". The highlighted area may be different from your configs file.
3. The binary must be compiled using targeted RISC-V architecture.
4. You do not need to give parameters for **RiscvO3CPU(),** use only the gem5 default, if you wish you can give different parameters or branch predictors.

2. Perform experimental simulations by changing L2 cache configurations on the above system with the "qsort_small" benchmark to achieve a similar plot in Fig.2 below. For your convenience, we have provided a few sets of configurations in Table 1. You can extend the set of configurations to have a better understanding.

[1] https://github.com/embecosm/mibench/tree/master/automotive/qsort



Plot from Hennessy and Patterson Ed. 5 Image Copyright © 2011, Elsevier Inc. All rights Reserved.

Fig.2. Variation in Cache Miss Rate with different cache sizes and associativity.

L2 Configurations (size_associativity)	L2 Configurations(size_associativity)
64kB_2	1024kB_8
64kB_4	1024kB_2
256kB_2	32kB_4
256kB_4	64kB_8

Table 1. Few sets of configurations for L2 Cache.

Deliverables:

1. **The assignment requires you to make observations on the L2 cache miss rate statistics report generated by each configuration for each CPU (RiscvTimingSimpleCPU()** and **RiscvO3CPU()) as you change the cache configurations.**
2. **Submit the following:**
   A. **Configuration script for both systems given in question 1.**
   B. **A separate graph (as shown in Figure 2) with the cache configurations you have considered and simulated on gem5 for both CPU models. (list all the configurations in a table). And compare the two graphs.**
   C. **Report the observed simulation Ticks for each CPU models.**
   D. **Explain your observations from the experiment performed in question 2.**

Note: 2.B, 2.C and 2.D should be uploaded in a single pdf. The python script should be kept separately in the same folder. All these should be zipped with folder name <your_name_roll_number_SA1> and uploaded in the classroom. **Do not forget to "TURN IN" your assignment.**

[1] https://github.com/embecosm/mibench/tree/master/automotive/qsort

Steps to Install the RISC-V Toolchain (this is used to generate the binary)
Link for the toolchain:https://github.com/riscv-collab/riscv-gnu-toolchain

1. **Install Prerequisites:**

You'll need some dependencies before you can build the RISC-V toolchain. Run the following commands:

```
sudo apt-get update
sudo apt-get install autoconf automake autotools-dev curl python3
python3-pip libmpc-dev libmpfr-dev libgmp-dev gawk build-essential bison
flex texinfo gperf libtool patchutils bc zlib1g-dev libexpat-dev
ninja-build git cmake libglib2.0-dev libslirp-dev
```

2. **Clone the RISC-V GNU Toolchain Repository:**

Clone the toolchain repository from GitHub:

```
git clone https://github.com/riscv/riscv-gnu-toolchain
cd riscv-gnu-toolchain
```

3. **Build the Toolchain:**

You can build the toolchain for the RISC-V architecture with the following command:

```
./configure --prefix=/opt/riscv
make
```

This will take some time to compile. The binaries will be installed in /opt/riscv/bin.

4. **Add the Toolchain to PATH:**

Once the toolchain is built, you should add the directory to your PATH so that you can use the riscv64-unknown-elf-gcc command from anywhere:

```
export PATH=/opt/riscv/bin:$PATH
```

You can also add this line to your '.bashrc' or '.bash_profile' to make this change permanent:

```
echo 'export PATH=/opt/riscv/bin:$PATH' >> ~/.bashrc
source ~/.bashrc
```

5. **Compile Your Program:**

Now, you should be able to compile your RISC-V program:

```
riscv64-unknown-elf-gcc -o qsort_small.elf qsort_small.c
```

If you're using Docker, you can install the toolchain directly inside the container or use an existing Docker image with the RISC-V toolchain pre-installed.