\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Requirements:**

To compile your code for POWER ISA, you will need a cross-compiler (gcc-powerpc64-linux-gnu) installed on your PC. We recommend using any one of the following options.

**Option 1 –**

If you have Ubuntu 18.04 (or 20.04) installed on your PC, install the cross compiler using the command below  
 sudo apt install gcc-powerpc64-linux-gnu

**Option 2 (For windows users only) –**

Install Windows subsystem for Linux <https://docs.microsoft.com/en-us/windows/wsl/install> . Launch WSL and install the cross compiler.

sudo apt install gcc-powerpc64-linux-gnu

**Option 3 –**

If you do not have access to Ubuntu 18.04 (or 20.04) machine, install VirtualBox <https://www.virtualbox.org/wiki/Downloads>. Download the appropriate platform package for your host OS. Follow the instructions on <https://ubuntu.com/tutorials/how-to-run-ubuntu-desktop-on-a-virtual-machine-using-virtualbox#1-overview> to start Ubuntu VM. Install the cross compiler using the command below  
 sudo apt install gcc-powerpc64-linux-gnu

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Compilation**

On the Ubuntu terminal, compile the provided matmul.c (Exercises→Code→CH1→matmul.c) using the command below:

powerpc64-linux-gnu-gcc -O0 -ggdb3 -std=c99 -static matmul.c -o matmul.out

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Adding caches to your configuration script**

Refer to the section on “Adding caches to the simple config file” from <https://www.gem5.org/documentation/learning_gem5/part1/cache_config/>

The above example shows how to add a two-level cache hierarchy in the configuration script.

The process of adding L3 and L4 caches is similar to the example above.

Apply the provided multilevel.patch to gem5 and build gem5.

*git clone https://github.com/gem5/gem5 -b stable*

*cd gem5*

*#reset the the commit on top of which the patch for L3 and L4 ill be applied*

*git reset --hard c890e6b*

*#copy the provided patch to ~/gem5\_workspace/gem5*

*#apply the patch, ignore whitespace warnings*

*git apply multilevel.patch*

*#build gem5 using scons*

(optional)Update L3 parameters in caches.py, (or your own copy of caches.py) as needed  
  
*class L3Cache(Cache):*

*size = '4096kB'*

*# size = '8192kB'*

*# size = '131072kB'*

*assoc = 8*

*tag\_latency = 20*

*data\_latency = 20*

*response\_latency = 20*

*mshrs = 20*

*tgts\_per\_mshr = 12*

*SimpleOpts.add\_option('--l3\_size', help="L3 cache size. Default: %s" % size)*

*def \_\_init\_\_(self, opts=None):*

*super(L3Cache, self).\_\_init\_\_()*

*if not opts or not opts.l3\_size:*

*return*

*self.size = opts.l3\_size*

*def connectCPUSideBus(self, bus):*

*self.cpu\_side = bus.mem\_side\_ports*

*def connectMemSideBus(self, bus):*

*self.mem\_side = bus.cpu\_side\_ports*

After updating the caches.py you can add L3 cache in your configuration script as shown below.

*# Create a simple CPU*

*system.cpu = TimingSimpleCPU()*

*# Create an L1 instruction cache and L1 data cache*

*system.cpu.icache = L1ICache()*

*system.cpu.dcache = L1DCache()*

*# Connect the instruction and data caches to the CPU*

*system.cpu.icache.connectCPU(system.cpu)*

*system.cpu.dcache.connectCPU(system.cpu)*

*# Create a memory bus, a coherent crossbar, in this case*

*system.l2bus = L2XBar()*

*system.l3bus = L3XBar()*

*# Hook the CPU ports up to the l2bus*

*system.cpu.icache.connectBus(system.l2bus)*

*system.cpu.dcache.connectBus(system.l2bus)*

*# Create an L2 cache and connect it to the l2bus*

*system.l2cache = L2Cache()*

*system.l2cache.connectCPUSideBus(system.l2bus)*

*#system.l2cache.connectCPUSideBus(system.l3bus)*

*#system.cpu.addTwoLevelCacheHierarchy(*

*#Create an L3 cache and connect it to the l2bus*

*system.l3cache = L3Cache()*

*system.l3cache.connectCPUSideBus(system.l3bus)*

*# Create a memory bus*

*system.membus = SystemXBar()*

*# Connect the L2 cache to the membus*

*system.l2cache.connectMemSideBus(system.l3bus)*

*# Connect the L3 cache to the membus*

*system.l3cache.connectMemSideBus(system.membus)*

Similarly, L4 cache can be added by making appropriate additions (Effectively the same process as adding L3 cache, just replace L3 with L4)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_