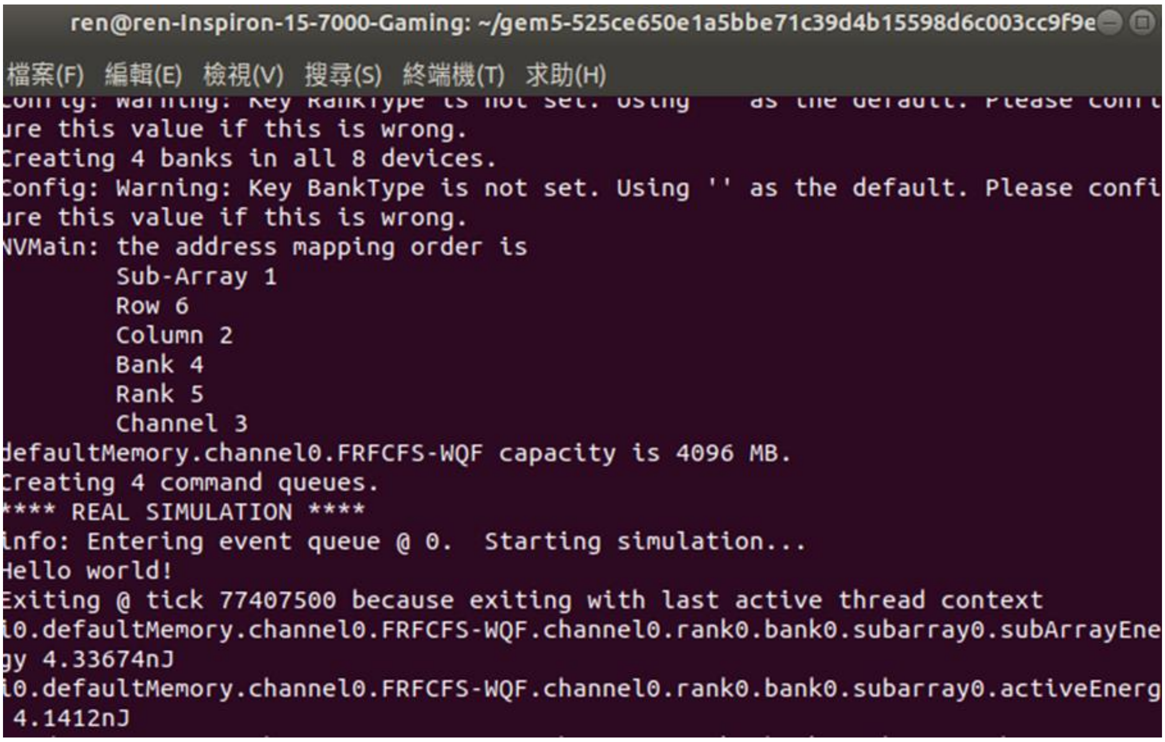
期末 Demo Project 說明

Q1.GEM5 + NVMAIN BUILD-UP (40%)

由於虛擬機的運行比較久,因此這次 project 我是直接灌雙系統來執行

使用 Hello World 來測試是否能 access L2 cache:

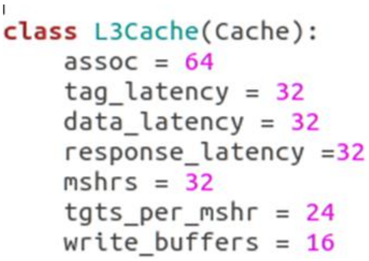

ren@ren-Inspiron-15-7000-Gaming: ~/gem5-525ce650e1a5bbe71c39d4b15598d6c003cc9f9e
檔案(F) 編輯(E) 檢視(V) 搜尋(S) 終端機(T) 求助(H)

```
onfig: Warning: Key RankType is not set. Using '' as the default. Please confi
ure this value if this is wrong.
reating 4 banks in all 8 devices.
onfig: Warning: Key BankType is not set. Using '' as the default. Please confi
ure this value if this is wrong.
VMain: the address mapping order is
        Sub-Array 1
        Row 6
        Column 2
        Bank 4
        Rank 5
        Channel 3
defaultMemory.channel0.FRFCFS-WQF capacity is 4096 MB.
reating 4 command queues.
*** REAL SIMULATION ****
nfo: Entering event queue @ 0.  Starting simulation...
ello world!
xiting @ tick 77407500 because exiting with last active thread context
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.subArrayEne
gy 4.33674nJ
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.activeEnerg
 4.1412nJ
```

Q2.Enable L3 last level cache in GEM5 + NVMAIN (15%)

(1)./config/common/Caches.py :

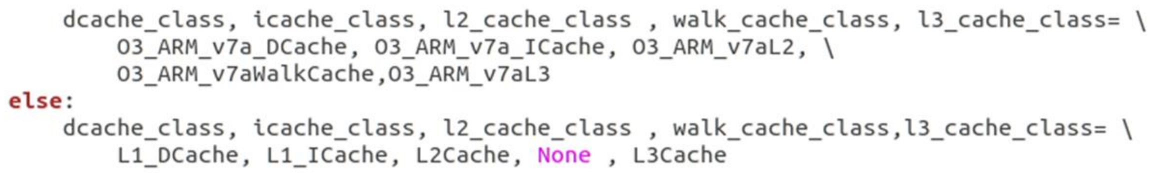
在此處進行新增 L3 Cache Class 的參數(可以照著 L2 的參數進行設定)



```
class L3Cache(Cache):
    assoc = 64
    tag_latency = 32
    data_latency = 32
    response_latency =32
    mshrs = 32
    tgts_per_mshr = 24
    write_buffers = 16
```

(2)./config/common/CacheConfig.py:

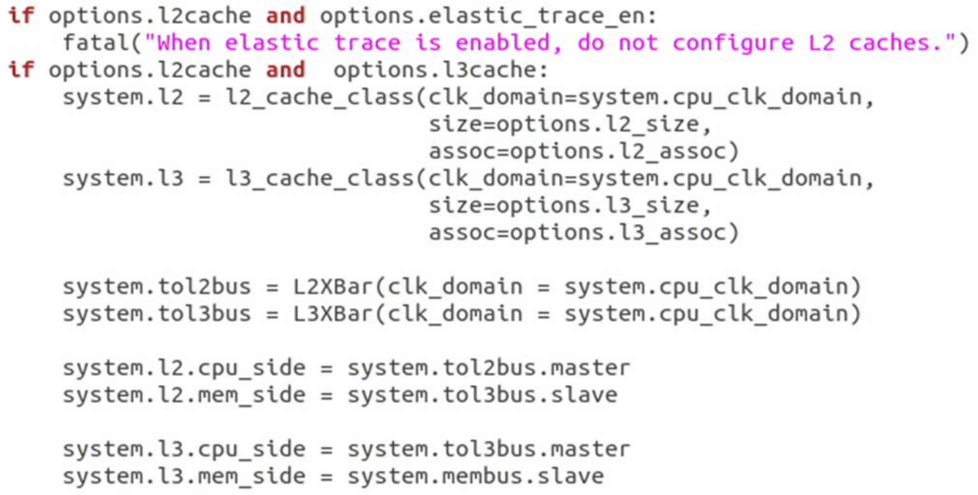
在此處設置 Cache3 的連接(是照著之前的形式設置):



```
    dcache_class, icache_class, l2_cache_class , walk_cache_class, l3_cache_class= \
        O3_ARM_v7a_DCache, O3_ARM_v7a_ICache, O3_ARM_v7aL2, \
        O3_ARM_v7aWalkCache,O3_ARM_v7aL3
else:
    dcache_class, icache_class, l2_cache_class , walk_cache_class,l3_cache_class= \
        L1_DCache, L1_ICache, L2Cache, None , L3Cache
```

(3)./config/common/CacheConfig.py :

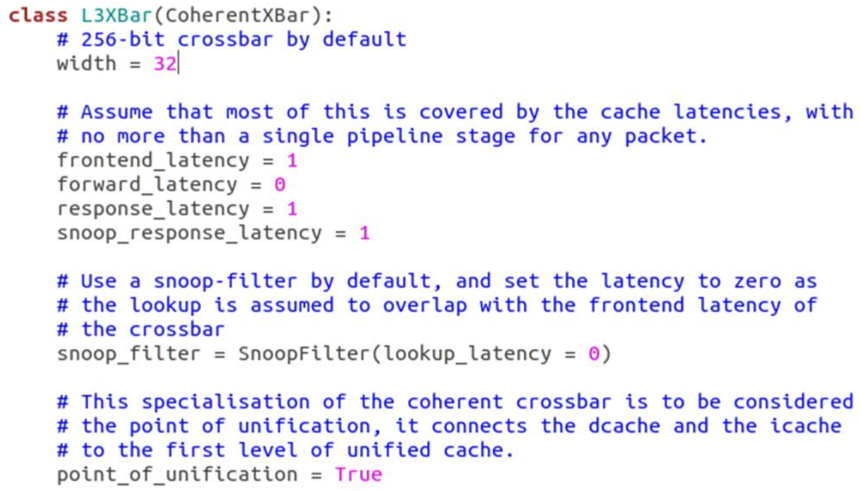
需要在此處設置出先經過 L2 cache 再 access L3 cache 的條件



```
if options.l2cache and options.elastic_trace_en:
    fatal("When elastic trace is enabled, do not configure L2 caches.")
if options.l2cache and  options.l3cache:
    system.l2 = l2_cache_class(clk_domain=system.cpu_clk_domain,
                               size=options.l2_size,
                               assoc=options.l2_assoc)
    system.l3 = l3_cache_class(clk_domain=system.cpu_clk_domain,
                               size=options.l3_size,
                               assoc=options.l3_assoc)

    system.tol2bus = L2XBar(clk_domain = system.cpu_clk_domain)
    system.tol3bus = L3XBar(clk_domain = system.cpu_clk_domain)

    system.l2.cpu_side = system.tol2bus.master
    system.l2.mem_side = system.tol3bus.slave

    system.l3.cpu_side = system.tol3bus.master
    system.l3.mem_side = system.membus.slave
```

(4)./src/mem/XBar.py :

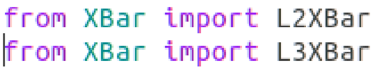
進行設置 L3 XBAR(一樣是複製 L2 的格式)



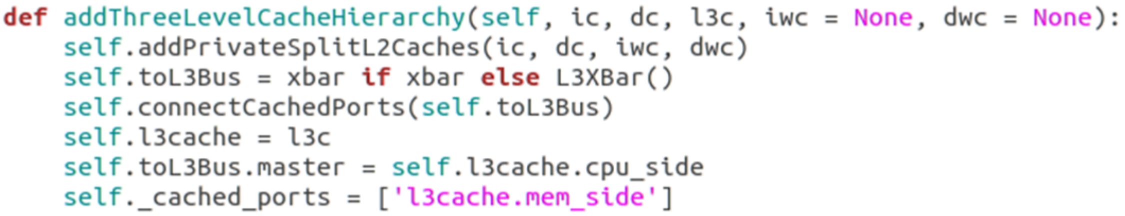
```
class L3XBar(CoherentXBar):
    # 256-bit crossbar by default
    width = 32

    # Assume that most of this is covered by the cache latencies, with
    # no more than a single pipeline stage for any packet.
    frontend_latency = 1
    forward_latency = 0
    response_latency = 1
    snoop_response_latency = 1

    # Use a snoop-filter by default, and set the latency to zero as
    # the lookup is assumed to overlap with the frontend latency of
    # the crossbar
    snoop_filter = SnoopFilter(lookup_latency = 0)

    # This specialisation of the coherent crossbar is to be considered
    # the point of unification, it connects the dcache and the icache
    # to the first level of unified cache.
    point_of_unification = True
```

(5)./src/cpu/BaseCPU.py :

將 L3 導入到 CPU 內,因此需要 import 並且 define L3 cache(照 L2)



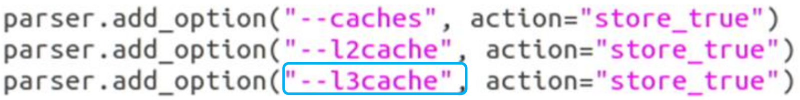
```
from XBar import L2XBar
from XBar import L3XBar
```



```
def addThreeLevelCacheHierarchy(self, ic, dc, l3c, iwc = None, dwc = None):
    self.addPrivateSplitL2Caches(ic, dc, iwc, dwc)
    self.toL3Bus = xbar if xbar else L3XBar()
    self.connectCachedPorts(self.toL3Bus)
    self.l3cache = l3c
    self.toL3Bus.master = self.l3cache.cpu_side
    self._cached_ports = ['l3cache.mem_side']
```

(6)./configs/common/Options.py:

加上能啟用 L3 的程式

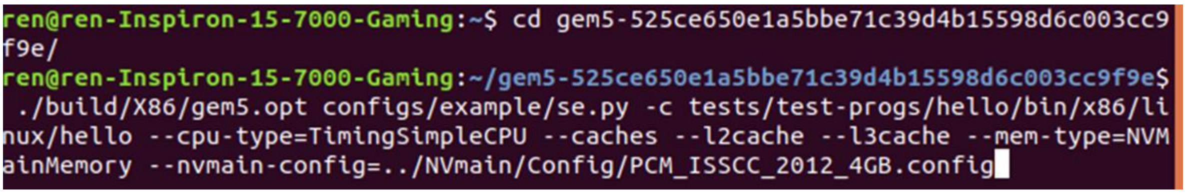


```
parser.add_option("--caches", action="store_true")
parser.add_option("--l2cache", action="store_true")
parser.add_option("--l3cache", action="store_true")
```

(7)由於有動過 GEM5 的內容,因此再次重新編譯

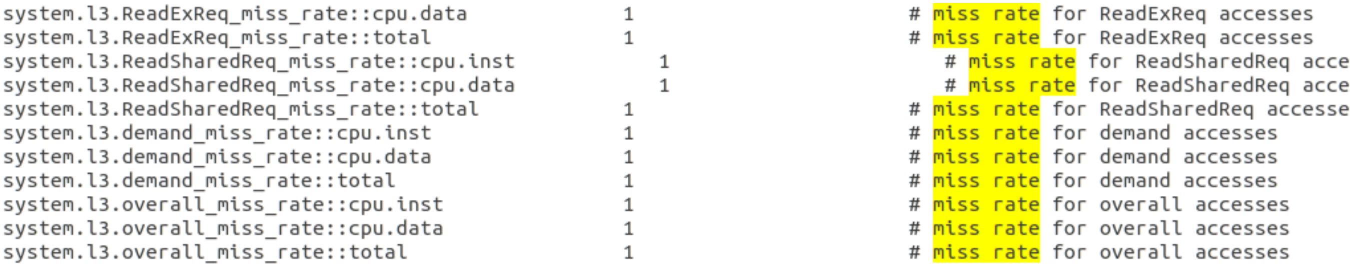
(8)再來透過 ppt 內的 Hello World 來測試是否 L3 Cache 會被 access 到：

(在--l2cache 後面加上 --l3cache)



```
ren@ren-Inspiron-15-7000-Gaming:~$ cd gem5-525ce650e1a5bbe71c39d4b15598d6c003cc9
f9e/
ren@ren-Inspiron-15-7000-Gaming:~/gem5-525ce650e1a5bbe71c39d4b15598d6c003cc9f9e$
 ./build/X86/gem5.opt configs/example/se.py -c tests/test-progs/hello/bin/x86/li
nux/hello --cpu-type=TimingSimpleCPU --caches --l2cache --l3cache --mem-type=NVM
ainMemory --nvmain-config=../NVmain/Config/PCM_ISSCC_2012_4GB.config
```

(9)最後會有 l3 cache 的 data



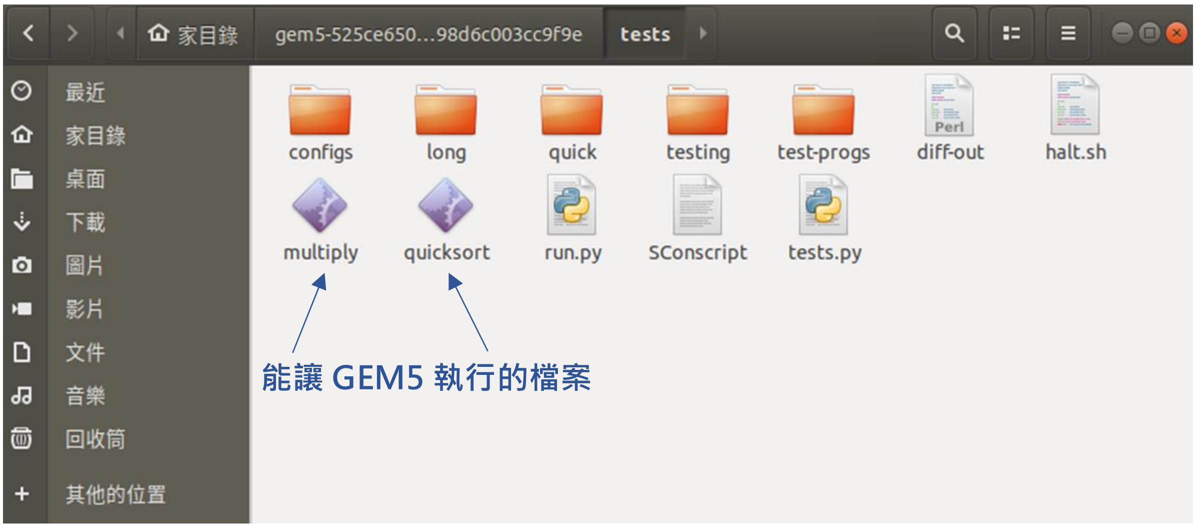
```
system.l3.ReadExReq_miss_rate::cpu.data            1                    # miss rate for ReadExReq accesses
system.l3.ReadExReq_miss_rate::total               1                    # miss rate for ReadExReq accesses
system.l3.ReadSharedReq_miss_rate::cpu.inst           1                    # miss rate for ReadSharedReq acce
system.l3.ReadSharedReq_miss_rate::cpu.data           1                    # miss rate for ReadSharedReq acce
system.l3.ReadSharedReq_miss_rate::total           1                    # miss rate for ReadSharedReq accesse
system.l3.demand_miss_rate::cpu.inst               1                    # miss rate for demand accesses
system.l3.demand_miss_rate::cpu.data               1                    # miss rate for demand accesses
system.l3.demand_miss_rate::total                  1                    # miss rate for demand accesses
system.l3.overall_miss_rate::cpu.inst              1                    # miss rate for overall accesses
system.l3.overall_miss_rate::cpu.data              1                    # miss rate for overall accesses
system.l3.overall_miss_rate::total                 1                    # miss rate for overall accesses
```

Q3.Config last level cache to 2-way and full-way associative cache and test performance (15%)

(1) 首先這次的 benchmark 是 Quicksort.c，因此得先在 ubuntu 內下載 gcc 編譯器並且透過 gcc --static quicksort.c -o quicksort

gcc --static multiply.c -o multiply

獲得能讓 GEM5 執行的檔案，我把它們放在 tests 資料夾內:

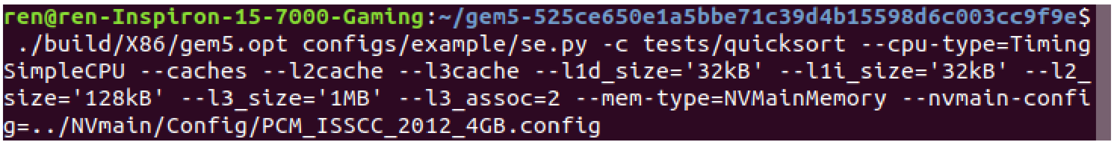


能讓 GEM5 執行的檔案

(2)再來透過指令能直接操作 2-way 以及設置 benchmark 內的 cache size

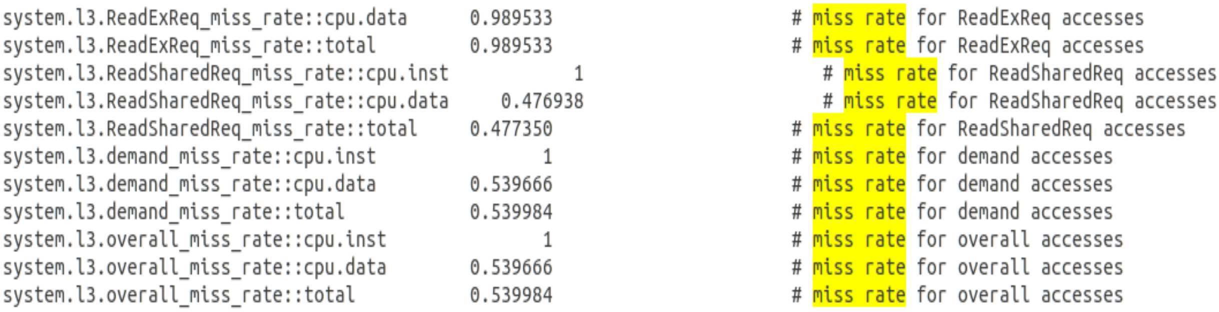
（或是要到 option.py 去改 default 也行）

以下為輸入指令:



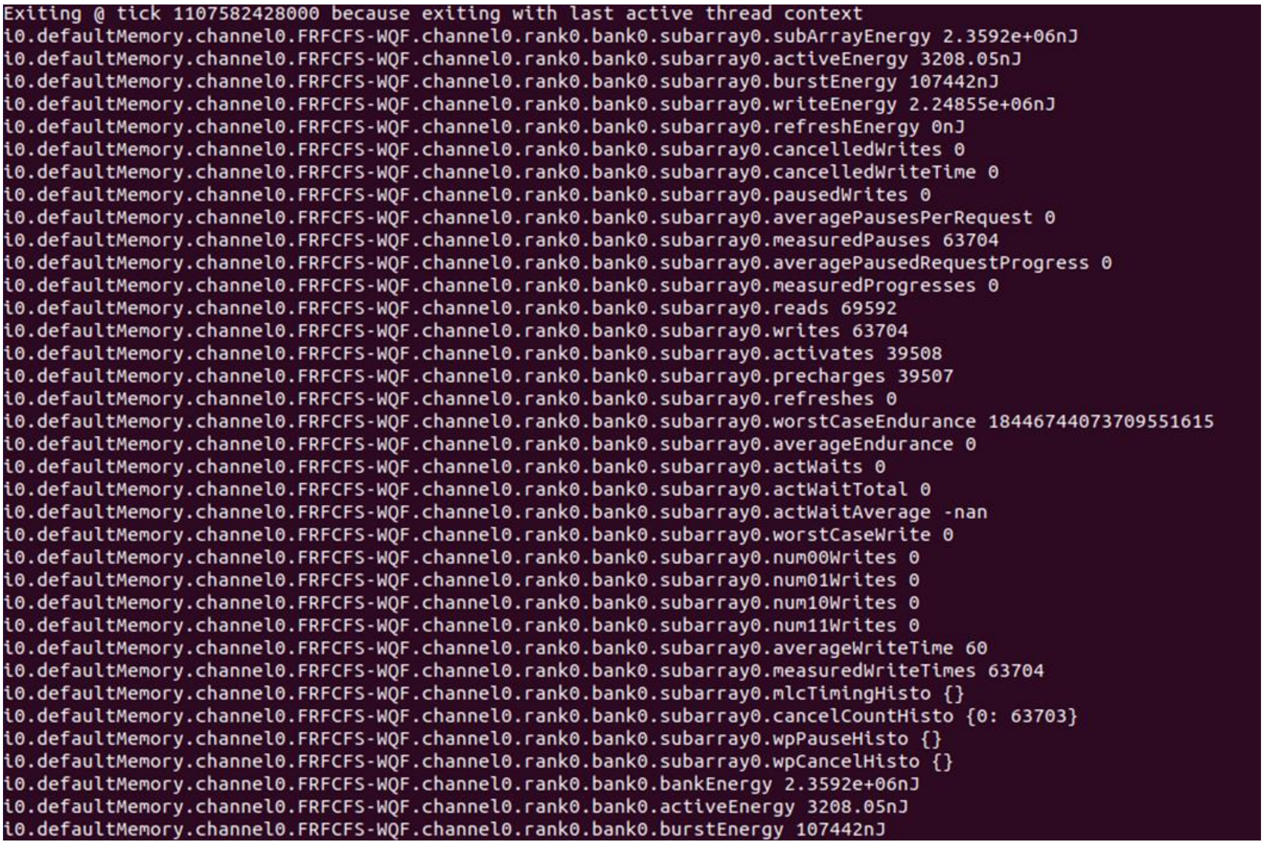
```
ren@ren-Inspiron-15-7000-Gaming:~/gem5-525ce650e1a5bbe71c39d4b15598d6c003cc9f9e$
 ./build/X86/gem5.opt configs/example/se.py -c tests/quicksort --cpu-type=Timing
SimpleCPU --caches --l2cache --l3cache --l1d_size='32kB' --l1i_size='32kB' --l2_
size='128kB' --l3_size='1MB' --l3_assoc=2 --mem-type=NVMainMemory --nvmain-confi
g=../NVmain/Config/PCM_ISSCC_2012_4GB.config
```

以下為在 2-way 下的 L3 miss rate:



```
system.l3.ReadExReq_miss_rate::cpu.data      0.989533                  # miss rate for ReadExReq accesses
system.l3.ReadExReq_miss_rate::total         0.989533                  # miss rate for ReadExReq accesses
system.l3.ReadSharedReq_miss_rate::cpu.inst          1                    # miss rate for ReadSharedReq accesses
system.l3.ReadSharedReq_miss_rate::cpu.data    0.476938                   # miss rate for ReadSharedReq accesses
system.l3.ReadSharedReq_miss_rate::total     0.477350                  # miss rate for ReadSharedReq accesses
system.l3.demand_miss_rate::cpu.inst             1                     # miss rate for demand accesses
system.l3.demand_miss_rate::cpu.data         0.539666                  # miss rate for demand accesses
system.l3.demand_miss_rate::total            0.539984                  # miss rate for demand accesses
system.l3.overall_miss_rate::cpu.inst            1                     # miss rate for overall accesses
system.l3.overall_miss_rate::cpu.data        0.539666                  # miss rate for overall accesses
system.l3.overall_miss_rate::total           0.539984                  # miss rate for overall accesses
```

以下為程式執行時的過程:



```
Exiting @ tick 1107582428000 because exiting with last active thread context
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.subArrayEnergy 2.3592e+06nJ
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.activeEnergy 3208.05nJ
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.burstEnergy 107442nJ
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.writeEnergy 2.24855e+06nJ
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.refreshEnergy 0nJ
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.cancelledWrites 0
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.cancelledWriteTime 0
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.pausedWrites 0
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.averagePausesPerRequest 0
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.measuredPauses 63704
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.averagePausedRequestProgress 0
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.measuredProgresses 0
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.reads 69592
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.writes 63704
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.activates 39508
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.precharges 39507
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.refreshes 0
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.worstCaseEndurance 18446744073709551615
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.averageEndurance 0
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.actWaits 0
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.actWaitTotal 0
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.actWaitAverage -nan
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.worstCaseWrite 0
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.num00Writes 0
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.num01Writes 0
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.num10Writes 0
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.num11Writes 0
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.averageWriteTime 60
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.measuredWriteTimes 63704
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.mlcTimingHisto {}
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.cancelCountHisto {0: 63703}
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.wpPauseHisto {}
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.wpCancelHisto {}
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.bankEnergy 2.3592e+06nJ
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.activeEnergy 3208.05nJ
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.burstEnergy 107442nJ
```

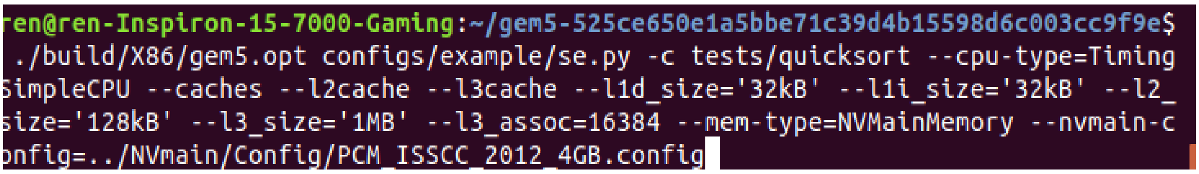
(3)再來透過指令能直接操作 **fully-way** 以及設置 benchmark 內的 cache size

首先先得知道，在 benchmark 中設置 L3 size 為 1MB

又在 GEM5 中 block size 預設的值為 64B

因此 2^20/2^6=2^14=16384

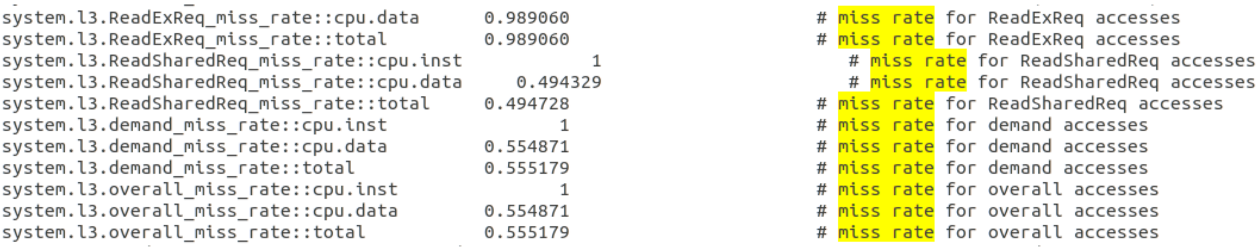
因此指令修改為:



```
ren@ren-Inspiron-15-7000-Gaming:~/gem5-525ce650e1a5bbe71c39d4b15598d6c003cc9f9e$
 ./build/X86/gem5.opt configs/example/se.py -c tests/quicksort --cpu-type=Timing
SimpleCPU --caches --l2cache --l3cache --l1d_size='32kB' --l1i_size='32kB' --l2_
size='128kB' --l3_size='1MB' --l3_assoc=16384 --mem-type=NVMainMemory --nvmain-c
onfig=../NVmain/Config/PCM_ISSCC_2012_4GB.config
```

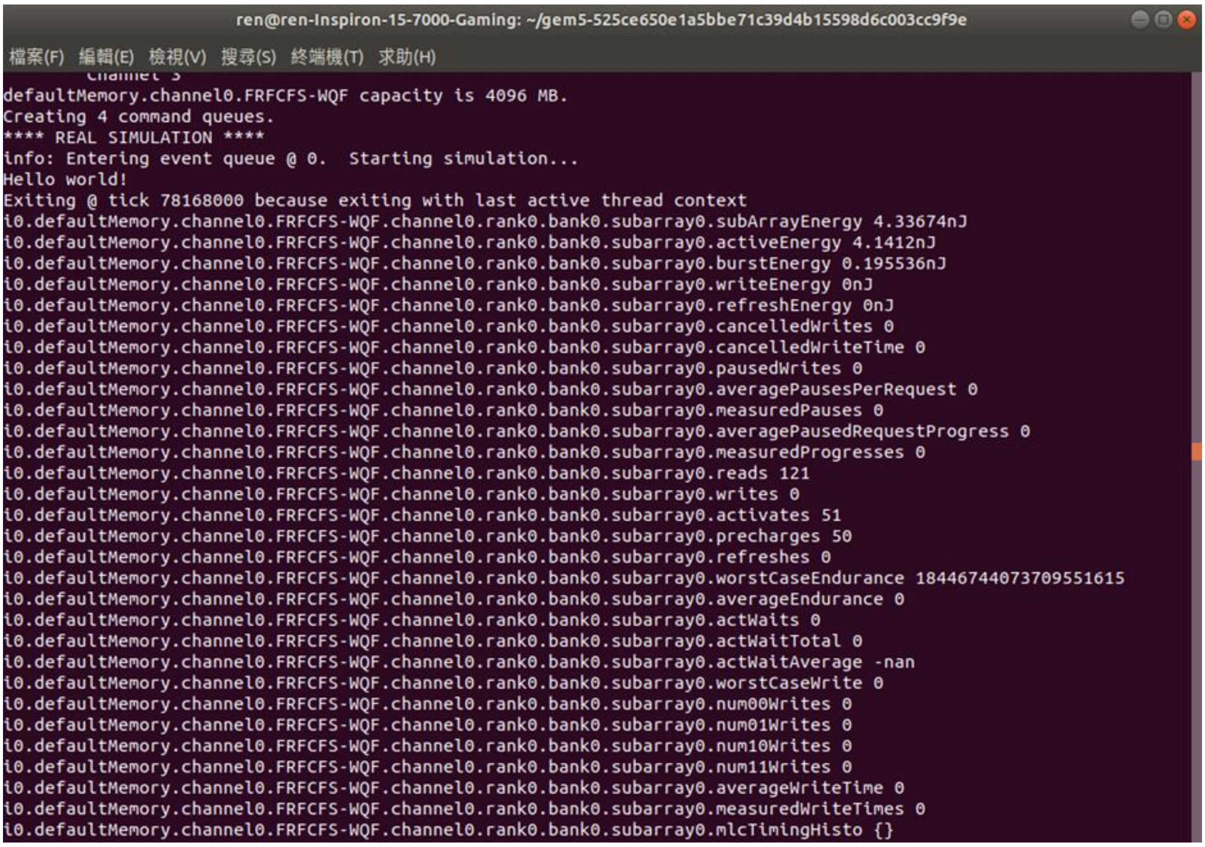
以下為 L3 的 miss rate，我們能發現在 fully 的實作下

Miss rate 反而往上升了:



```
system.l3.ReadExReq_miss_rate::cpu.data       0.989060                        # miss rate for ReadExReq accesses
system.l3.ReadExReq_miss_rate::total          0.989060                        # miss rate for ReadExReq accesses
system.l3.ReadSharedReq_miss_rate::cpu.inst          1                          # miss rate for ReadSharedReq accesses
system.l3.ReadSharedReq_miss_rate::cpu.data     0.494329                        # miss rate for ReadSharedReq accesses
system.l3.ReadSharedReq_miss_rate::total      0.494728                        # miss rate for ReadSharedReq accesses
system.l3.demand_miss_rate::cpu.inst                 1                        # miss rate for demand accesses
system.l3.demand_miss_rate::cpu.data          0.554871                        # miss rate for demand accesses
system.l3.demand_miss_rate::total             0.555179                        # miss rate for demand accesses
system.l3.overall_miss_rate::cpu.inst                1                        # miss rate for overall accesses
system.l3.overall_miss_rate::cpu.data         0.554871                        # miss rate for overall accesses
system.l3.overall_miss_rate::total            0.555179                        # miss rate for overall accesses
```

以下為程式執行過程:



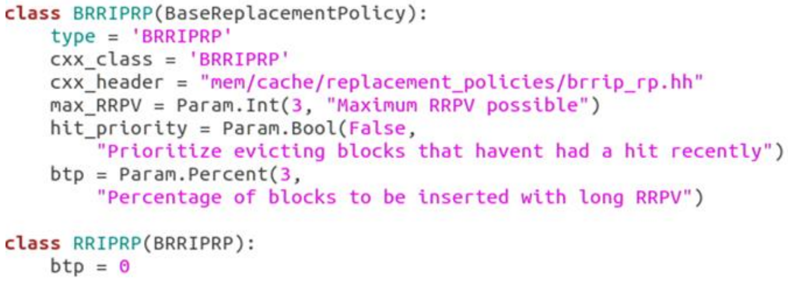
ren@ren-Inspiron-15-7000-Gaming: ~/gem5-525ce650e1a5bbe71c39d4b15598d6c003cc9f9e
檔案(F) 編輯(E) 檢視(V) 搜尋(S) 終端機(T) 求助(H)

```
        Channel 3
defaultMemory.channel0.FRFCFS-WQF capacity is 4096 MB.
Creating 4 command queues.
**** REAL SIMULATION ****
info: Entering event queue @ 0.  Starting simulation...
Hello world!
Exiting @ tick 78168000 because exiting with last active thread context
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.subArrayEnergy 4.33674nJ
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.activeEnergy 4.1412nJ
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.burstEnergy 0.195536nJ
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.writeEnergy 0nJ
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.refreshEnergy 0nJ
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.cancelledWrites 0
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.cancelledWriteTime 0
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.pausedWrites 0
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.averagePausesPerRequest 0
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.measuredPauses 0
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.averagePausedRequestProgress 0
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.measuredProgresses 0
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.reads 121
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.writes 0
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.activates 51
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.precharges 50
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.refreshes 0
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.worstCaseEndurance 18446744073709551615
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.averageEndurance 0
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.actWaits 0
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.actWaitTotal 0
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.actWaitAverage -nan
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.worstCaseWrite 0
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.num00Writes 0
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.num01Writes 0
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.num10Writes 0
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.num11Writes 0
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.averageWriteTime 0
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.measuredWriteTimes 0
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.mlcTimingHisto {}
```

Q4.Modify last level cache policy based on RRIP (15%)

(1) 首先在 ReplacementPolicies.py 能找到很多 GEM5 所提供的 policy

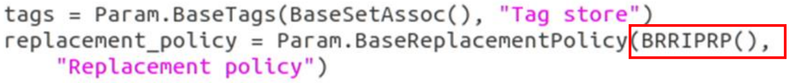
代表已經有內建寫好一些能替代的 policy:



```
class BRRIPRP(BaseReplacementPolicy):
    type = 'BRRIPRP'
    cxx_class = 'BRRIPRP'
    cxx_header = "mem/cache/replacement_policies/brrip_rp.hh"
    max_RRPV = Param.Int(3, "Maximum RRPV possible")
    hit_priority = Param.Bool(False,
        "Prioritize evicting blocks that havent had a hit recently")
    btp = Param.Percent(3,
        "Percentage of blocks to be inserted with long RRPV")

class RRIPRP(BRRIPRP):
    btp = 0
```

(2) 在 build/x86/mem/cache/cache.py 之中改掉原本 LRURP 的 policy 換

成 BRRIPRP:

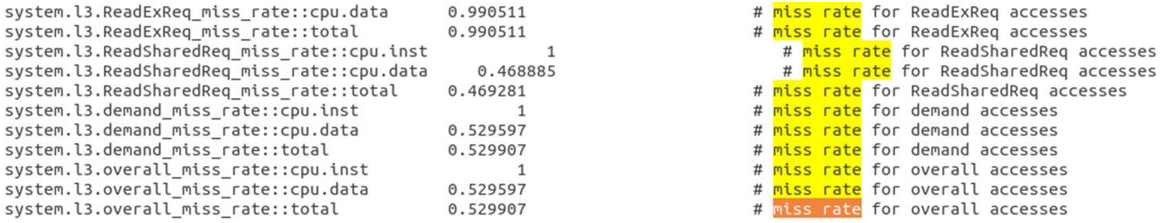


```
tags = Param.BaseTags(BaseSetAssoc(), "Tag store")
replacement_policy = Param.BaseReplacementPolicy(BRRIPRP(),
    "Replacement policy")
```

(3) 再進行一次編譯

(4) 接著，可觀察在 rrip 實作後的結果，會比原本 LRU 時，miss rate 有所

下降:

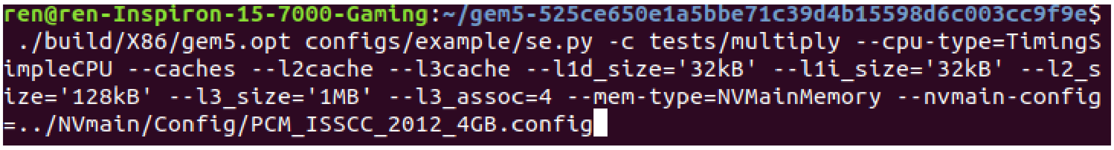


```
system.l3.ReadExReq_miss_rate::cpu.data      0.990511                          # miss rate for ReadExReq accesses
system.l3.ReadExReq_miss_rate::total         0.990511                          # miss rate for ReadExReq accesses
system.l3.ReadSharedReq_miss_rate::cpu.inst             1                        # miss rate for ReadSharedReq accesses
system.l3.ReadSharedReq_miss_rate::cpu.data      0.468885                        # miss rate for ReadSharedReq accesses
system.l3.ReadSharedReq_miss_rate::total     0.469281                          # miss rate for ReadSharedReq accesses
system.l3.demand_miss_rate::cpu.inst                1                          # miss rate for demand accesses
system.l3.demand_miss_rate::cpu.data         0.529597                          # miss rate for demand accesses
system.l3.demand_miss_rate::total            0.529907                          # miss rate for demand accesses
system.l3.overall_miss_rate::cpu.inst               1                          # miss rate for overall accesses
system.l3.overall_miss_rate::cpu.data        0.529597                          # miss rate for overall accesses
system.l3.overall_miss_rate::total           0.529907                          # miss rate for overall accesses
```

(miss rate 有下降約 0.01，跟 2-way inplement by LRU 相比)

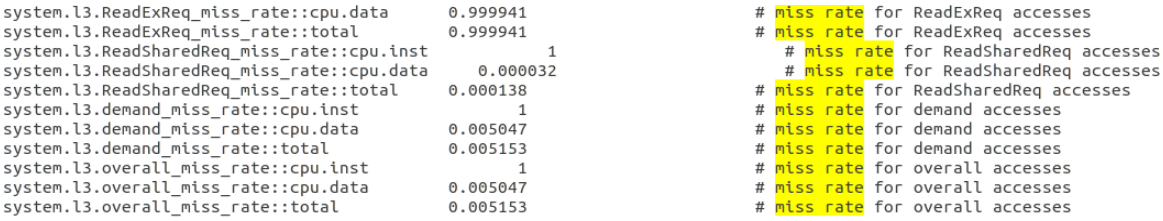
Q5.Test the performance of write back and write through policy based on 4-way associative cache with isscc_pcm(15%)

(1) 由於 default 本身預設就是 write back 了,因此輸入指令如下：



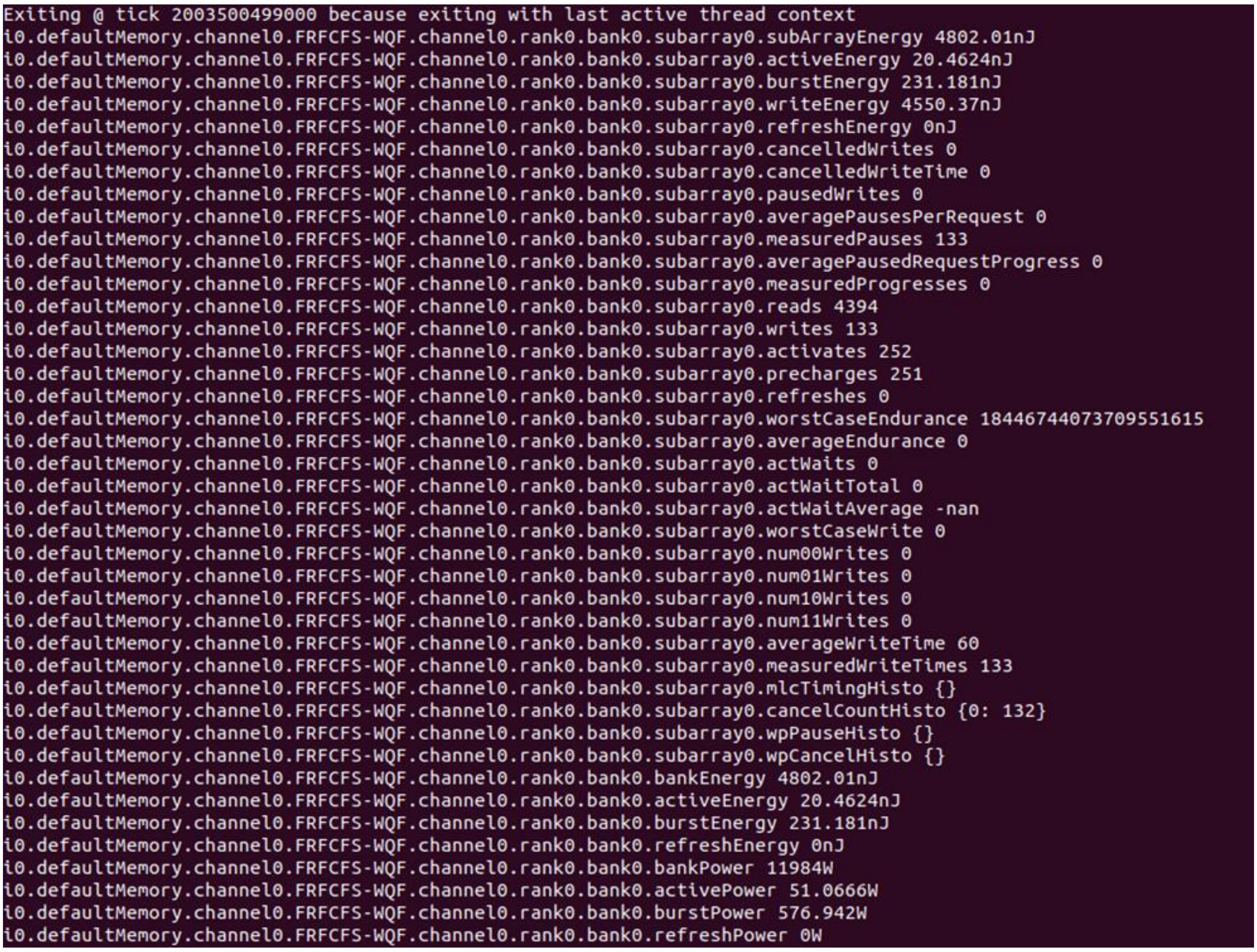
```
ren@ren-Inspiron-15-7000-Gaming:~/gem5-525ce650e1a5bbe71c39d4b15598d6c003cc9f9e$
 ./build/X86/gem5.opt configs/example/se.py -c tests/multiply --cpu-type=TimingS
impleCPU --caches --l2cache --l3cache --l1d_size='32kB' --l1i_size='32kB' --l2_s
ize='128kB' --l3_size='1MB' --l3_assoc=4 --mem-type=NVMainMemory --nvmain-config
=../NVmain/Config/PCM_ISSCC_2012_4GB.config
```

而以下是 L3 的 miss rate:



```
system.l3.ReadExReq_miss_rate::cpu.data      0.999941                          # miss rate for ReadExReq accesses
system.l3.ReadExReq_miss_rate::total         0.999941                          # miss rate for ReadExReq accesses
system.l3.ReadSharedReq_miss_rate::cpu.inst         1                          # miss rate for ReadSharedReq accesses
system.l3.ReadSharedReq_miss_rate::cpu.data     0.000032                       # miss rate for ReadSharedReq accesses
system.l3.ReadSharedReq_miss_rate::total     0.000138                          # miss rate for ReadSharedReq accesses
system.l3.demand_miss_rate::cpu.inst                1                          # miss rate for demand accesses
system.l3.demand_miss_rate::cpu.data         0.005047                          # miss rate for demand accesses
system.l3.demand_miss_rate::total            0.005153                          # miss rate for demand accesses
system.l3.overall_miss_rate::cpu.inst               1                          # miss rate for overall accesses
system.l3.overall_miss_rate::cpu.data        0.005047                          # miss rate for overall accesses
system.l3.overall_miss_rate::total           0.005153                          # miss rate for overall accesses
```

以下是程式執行過程:



```
Exiting @ tick 2003500499000 because exiting with last active thread context
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.subArrayEnergy 4802.01nJ
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.activeEnergy 20.4624nJ
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.burstEnergy 231.181nJ
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.writeEnergy 4550.37nJ
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.refreshEnergy 0nJ
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.cancelledWrites 0
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.cancelledWriteTime 0
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.pausedWrites 0
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.averagePausesPerRequest 0
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.measuredPauses 133
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.averagePausedRequestProgress 0
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.measuredProgresses 0
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.reads 4394
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.writes 133
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.activates 252
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.precharges 251
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.refreshes 0
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.worstCaseEndurance 18446744073709551615
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.averageEndurance 0
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.actWaits 0
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.actWaitTotal 0
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.actWaitAverage -nan
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.worstCaseWrite 0
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.num00Writes 0
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.num01Writes 0
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.num10Writes 0
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.num11Writes 0
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.averageWriteTime 60
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.measuredWriteTimes 133
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.mlcTimingHisto {}
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.cancelCountHisto {0: 132}
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.wpPauseHisto {}
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.subarray0.wpCancelHisto {}
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.bankEnergy 4802.01nJ
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.activeEnergy 20.4624nJ
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.burstEnergy 231.181nJ
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.refreshEnergy 0nJ
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.bankPower 11984W
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.activePower 51.0666W
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.burstPower 576.942W
i0.defaultMemory.channel0.FRFCFS-WQF.channel0.rank0.bank0.refreshPower 0W
```