計算機結構
Computer Architecture
HW3: Gem5 on memory hierarchy

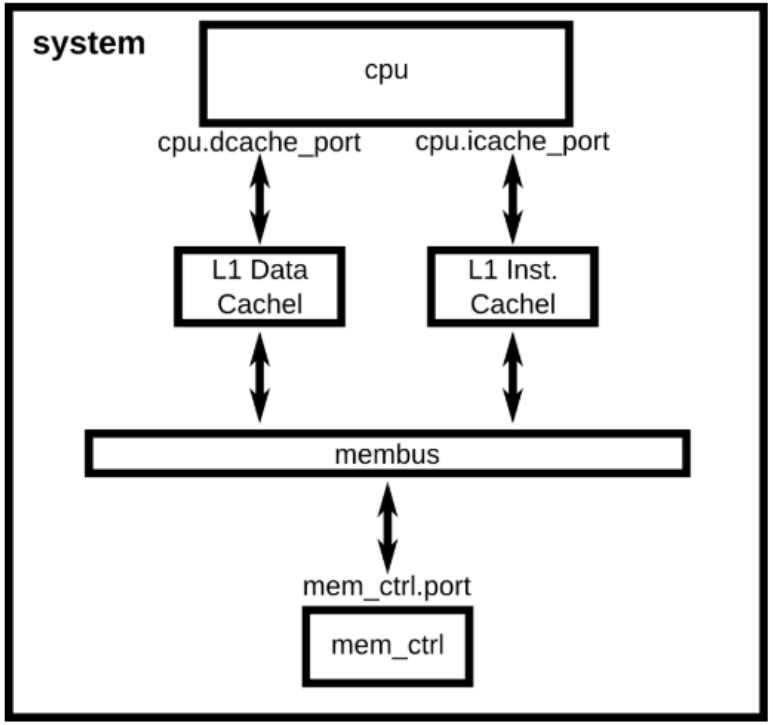
Name: Kuan-Heng Liu
Advisor: Prof An-Yeu Wu
Date: 2025/05/05

HW3 Overview

The HW3 consists of the following sections:

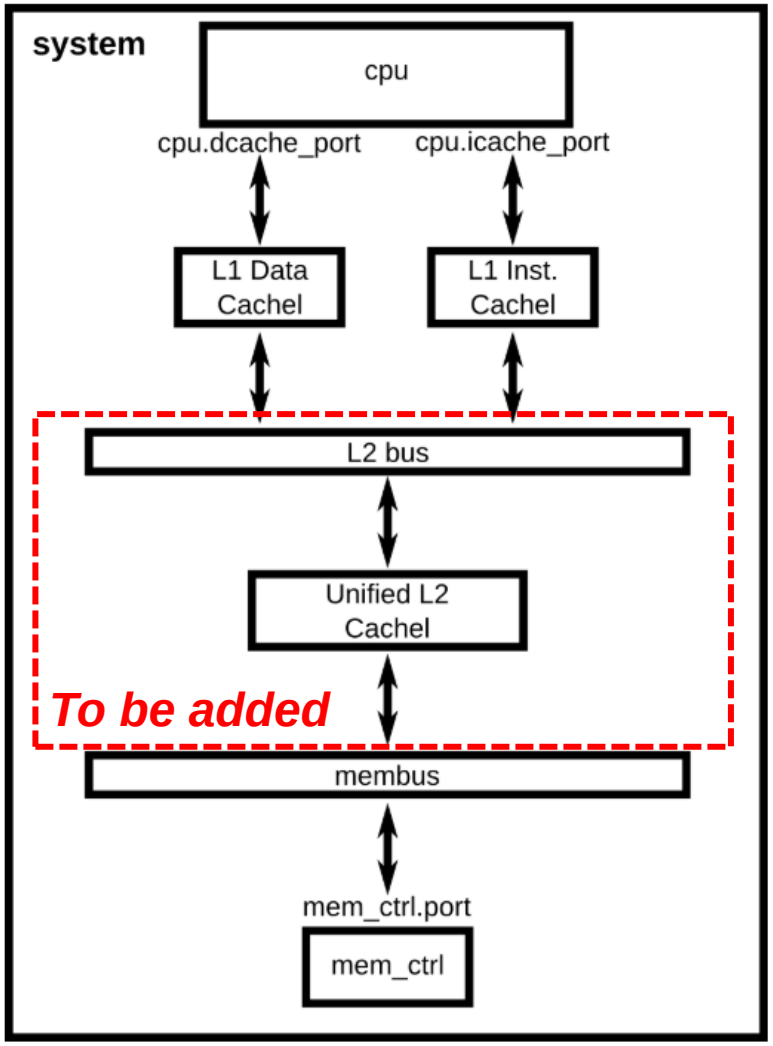
- Part1
  - Implement L2 cache in gem5 config
  - Draw and analyze the program results based on the cache config
- Part2
  - Find the optimal config settings for this program
  - Explain why this config achieves optimal performance

HW3 Purpose



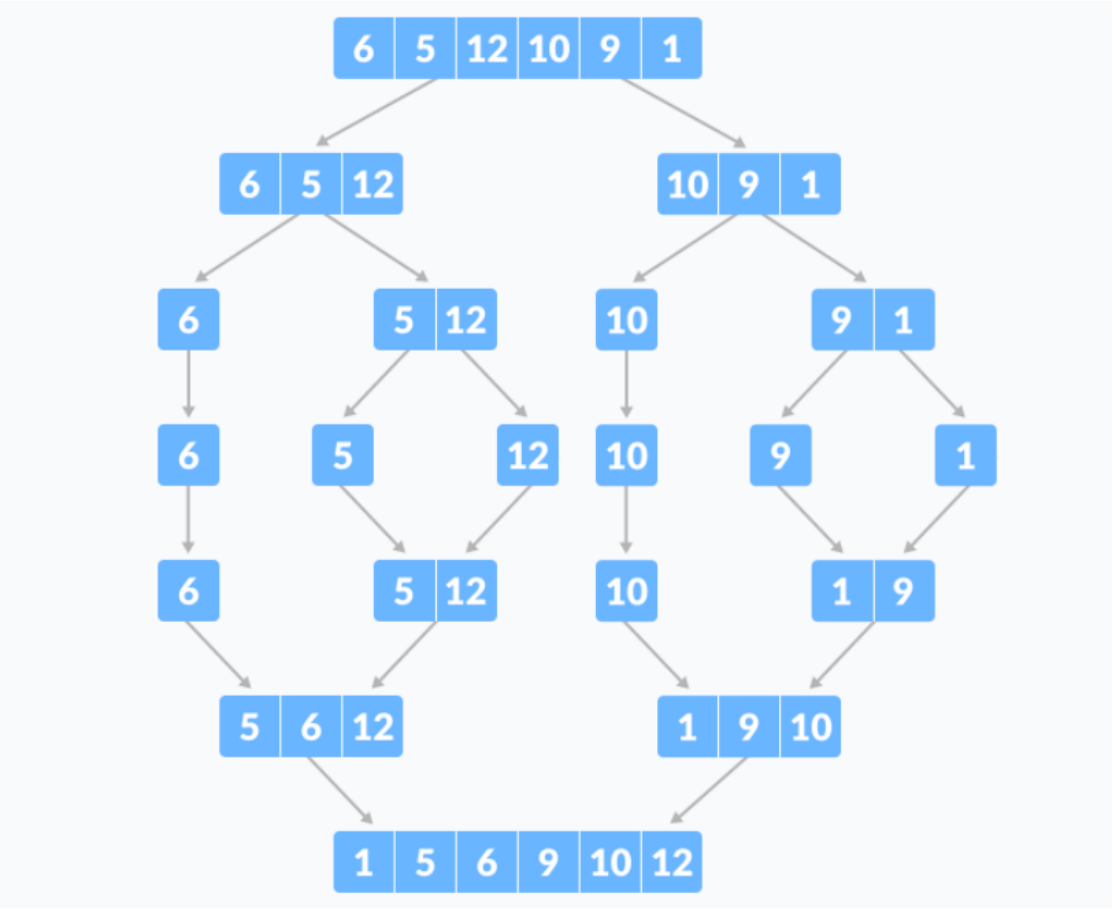
- Learn how to add new components to gem5 config
- Can analyze and find the optimal config based on the application

HW3 Purpose



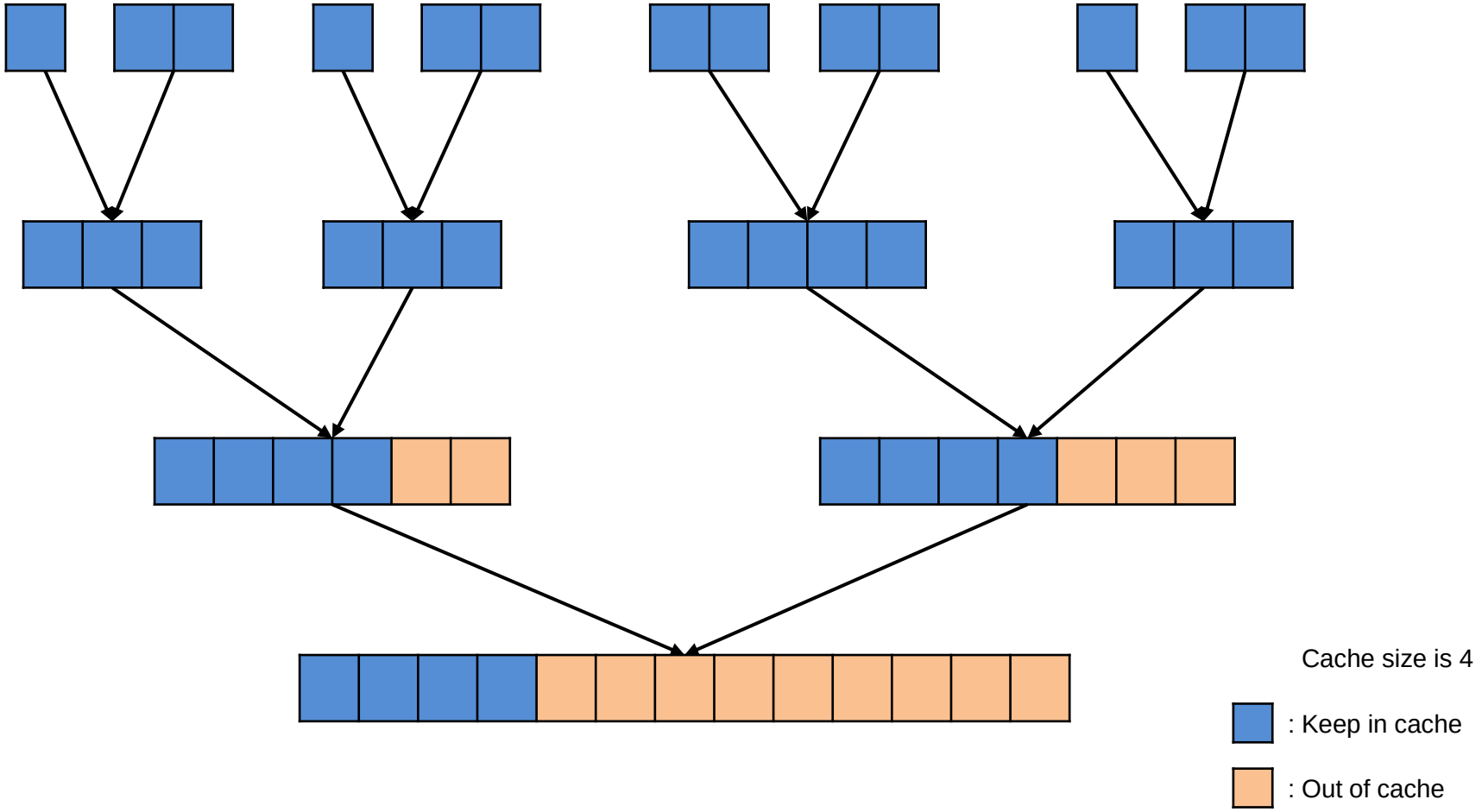
- Learn how to add new components to gem5 config
- Can analyze and find the optimal config based on the application

Application: Merge Sorting



- Merge Sorting is a algorithm that follows the divide-and-conquer approach
- Time Complexity: O(N logN)

Application: Merge Sorting



PART1

Part1 – Environment Setting

- Place the following file in the your docker container
- /workspace/hw3/
  - merge_sort.cpp
  - profiler.py
  - profiler_plot.py
  - Makefile
  - simple-riscv-mod-config.py

Part1 – Compile Program

- cd workspace/hw3/
- Compile merge_sort.cpp

```
root@64cf7f5a775c:/workspace# make g++
/opt/riscv/bin/riscv64-unknown-linux-gnu-g++ -march=rv64gcv -mabi=lp64d merge_sort.cpp -o merge_sort -static
C++ code compiled to merge_sort
```

make g++

Part1 – Implement L2 Cache

- You have to **add L2 Cache in simple-riscv-mod-config.py**
  - Support argument input for L2 cache config
  - Gem5 script tutorial: CMU material
  - **Modification of cache latency is prohibited !**

```
# ------------------- L2 Cache Arguments --------------------
parser.add_argument("--l2_size",  type=str, default="64kB",
                    help="L2 cache size")
parser.add_argument("--l2_assoc", type=int, default=8,
                    help="L2 cache associativity")
                                                cache latency
parser.add_argument("--l2_tag_latency", type=int, default=20,
                    help="L2 cache tag latency (cycles)")
parser.add_argument("--l2_data_latency", type=int, default=20,
                    help="L2 cache data latency (cycles)")
parser.add_argument("--l2_response_latency", type=int, default=20,
                    help="L2 cache response latency (cycles)")
```

- make gem5 \
GEM5_ARGS=--isa_type 32 --l1i_size 1kB --l1i_assoc 2 --l1d_size 1kB --l1d_assoc 2 --l2_size 16kB --l2_assoc 4"

Part1 – Implement L2 Cache

- You have to **add L2 Cache in simple-riscv-mod-config.py**
  - Support argument input for L2 cache config
  - Gem5 script tutorial: CMU material
- Should be able to pass in L2 cache config & run **make gem5** successfully

```
Original array:
403 623 817 57 210 808 842 410 329 301 46 8 889 470 320 469 625 243 630 883 638 157 235 102 866 665 788 108 63 597 695 818 572 864 876 783 672 70 545 2 37
1 943 362 612 414 34 433 39 629 64 274 268 221 509 722 87 175 511 547 590 460 242 761 32 106 637 167 131 707 64 485 430 8 847 42 774 234 828 813 863 892 4
39 131 465 949 206 905 476 717 452 66 177 47 179 561 153 816 81 636 875 145 121 305 505 321 700 279 555 880 444 418 772 884 902 589 833 108 494 661 177 94
7 727 706 994 259 619 499 75

Sorted array:
2 8 8 32 34 39 42 46 47 57 63 64 64 66 70 75 81 87 102 106 108 108 121 131 131 145 153 157 167 175 177 177 179 206 210 221 234 235 242 243 259 268 274 279
 301 305 320 321 329 362 371 403 410 414 418 430 433 439 444 452 460 465 469 470 476 485 494 499 505 509 511 545 547 555 561 572 589 590 597 612 619 623 6
25 629 630 636 637 638 661 665 672 695 700 706 707 717 722 727 761 772 774 783 788 808 813 816 817 818 828 833 842 847 863 864 866 875 876 880 883 884 889
 892 902 905 943 947 949 994

sorted numbers: 128
Exiting @ tick 4424434000 because exiting with last active thread context
Emulated merge sort  on gem5 with arguments: --isa type 64 --l1i size 1kB --l1i assoc 2 --l1d size 1kB --l1d assoc 2 --l2 size 16kB --l2 assoc 4
```

➢ make gem5 \
GEM5_ARGS=--isa_type 32 --l1i_size 1kB --l1i_assoc 2 --l1d_size 1kB --l1d_assoc 2 --l2_size 16kB --l2_assoc 4"

Part1 – Draw and Analyze the Program

- Profile the execution details of the program
- **Clear all data in the CSV** files and store the current gem5 execution results by **make profile**

```
Program summary
---------------------------------------
simulated time              | 0.001845 s
simulated tick              | 1,845,419,000 ticks
total Inst.                 | 378,943 instructions
total cycle                 | 1,845,419 cycles
CPI                         | 4.866355
IPC                         | 0.205493
Int-Inst. count             | 376,012 instructions
Load-Inst. count            | 80,753 instructions
Store-Inst. count           | 40,782 instructions
Vector-Inst. count          | 0 instructions
```

```
L2-Cache summary
---------------------------------------
$L2 hit count               | 22,516 counts
$L2 miss count              | 1,408 counts
$L2 access count            | 23,924 counts
$L2 miss rate               | 5.89% miss rate
L2 assoc                    | 4
L2 size                     | 16384
```

```
L1-Instruction-Cache summary
---------------------------------------
$L1-I hit count             | 442,355 counts
$L1-I miss count            | 17,213 counts
$L1-I access count          | 459,568 counts
$L1-I miss rate             | 3.75% miss rate
L1-I assoc                  | 2
L1-I size                   | 1024

L1-Data-Cache summary
---------------------------------------
$L1-D hit count             | 114,792 counts
$L1-D miss count            | 6,705 counts
$L1-D access count          | 121,497 counts
$L1-D miss rate             | 5.52% miss rate
L1-D assoc                  | 2
L1-D size                   | 1024
```

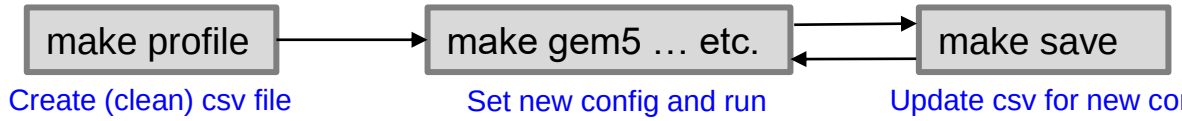
make profile

Part1 – Draw and Analyze the Program

- Add the results of new config to a CSV file by **make save**
- Draw a graph based on the data in the csv file and analyze the impact of each config on execution time
- Visualization is required for the following config
  - L1-I cache size
  - L1-D cache size
  - L2 cache size
  - L1-I associativity
  - L1-D associativity
  - L2 associativity

Range
- **Cache Size: 1KB ~ 128KB**
- **Associativity: 1 ~ 8**
- **Unit: ms**

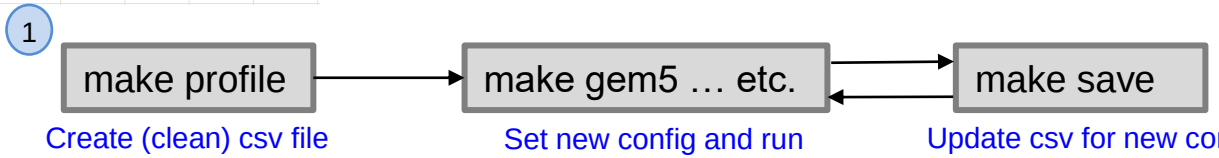
 # Note: power of 2



Part1 – Draw and Analyze the Program

- Add the results of new config to a CSV file by **make save**
- Draw a graph based on the data in the csv file and analyze the impact of each config on execution time

Program summary	
simulated	1.845
simulated	1.8E+09
total Inst.	378943
total cycle	1845419
CPI	4.86636
IPC	0.20549
Int-Inst. cc	376012
Load-Inst.	80753
Store-Inst.	40782
Vector-Ins	0
L1-Instruction-Cache summary	
$L1-I hit c	442355
$L1-I miss	17213
$L1-I acce	459568
$L1-I miss	0.03746
L1-I assoc	2
L1-I size	1024
L1-Data-Cache summary	
$L1-D hit	114792
$L1-D mi:	6705
$L1-D acc	121497
$L1-D mi:	0.05519
L1-D asso	2
L1-D size	1024



Part1 – Draw and Analyze the Program

- Add the results of new config to a CSV file by **make save**
- Draw a graph based on the data in the csv file and analyze the impact of each config on execution time

Program summary	
simulated	1.845
simulated	1.8E+09
total Inst.	378943
total cycle	1845419
CPI	4.86636
IPC	0.20549
Int-Inst. cc	376012
Load-Inst.	80753
Store-Inst.	40782
Vector-Ins	0
L1-Instruction-Cache summary	
$L1-I hit c	442355
$L1-I miss	17213
$L1-I acce	459568
$L1-I miss	0.03746
L1-I assoc	2
L1-I size	1024
L1-Data-Cache summary	
$L1-D hit	114792
$L1-D mi:	6705
$L1-D acc	121497
$L1-D mi:	0.05519
L1-D asso	2
L1-D size	1024

make gem5 … etc.
+
make save

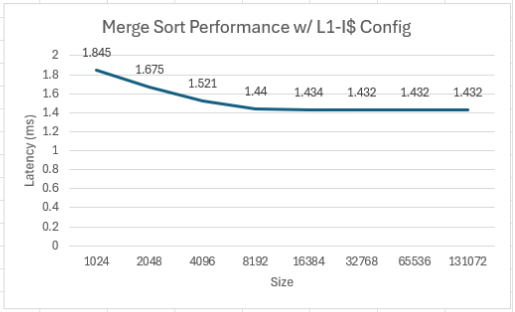
Program summary		
simulated	1.845	1.675
simulated	1.8E+09	1.7E+09
total Inst.	378943	378943
total cycle	1845419	1674680
CPI	4.86636	4.41612
IPC	0.20549	0.22644
Int-Inst. cc	376012	376012
Load-Inst.	80753	80753
Store-Inst.	40782	40782
Vector-Ins	0	0
L1-Instruction-Cache summary		
$L1-I hit c	442355	449467
$L1-I miss	17213	10101
$L1-I acce	459568	459568
$L1-I miss	0.03746	0.02198
L1-I assoc	2	2
L1-I size	1024	2048
L1-Data-Cache summary		
$L1-D hit	114792	114792
$L1-D mi:	6705	6705
$L1-D acc	121497	121497
$L1-D mi:	0.05519	0.05519
L1-D asso	2	2
L1-D size	1024	1024

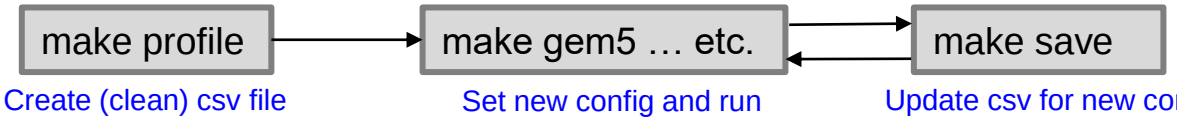
1. make profile — Create (clean) csv file
2. make gem5 … etc. — Set new config and run
3. make save — Update csv for new config

Part1 – Draw and Analyze the Program

- Add the results of new config to a CSV file by **make save**
- Draw a graph based on the data in the csv file and analyze the impact of each config on execution time

Program summary								
simulated	1.845	1.675	1.521	1.44	1.434	1.432	1.432	1.432
simulated	1.8E+09	1.7E+09	1.5E+09	1.4E+09	1.4E+09	1.4E+09	1.4E+09	1.4E+09
total Inst.	378943	378943	378943	378943	378943	378943	378943	378943
total cycle	1845419	1674680	1521221	1440443	1434421	1431641	1431593	1431593
CPI	4.86636	4.41612	4.01145	3.79844	3.78256	3.77523	3.7751	3.7751
IPC	0.20549	0.22644	0.24929	0.26327	0.26437	0.26489	0.26489	0.26489
Int-Inst. cc	376012	376012	376012	376012	376012	376012	376012	376012
Load-Inst.	80753	80753	80753	80753	80753	80753	80753	80753
Store-Inst.	40782	40782	40782	40782	40782	40782	40782	40782
Vector-Ins	0	0	0	0	0	0	0	0
L1-Instruction-Cache summary								
$L1-I hit c	442355	449467	455822	459080	459215	459245	459253	459253
$L1-I miss	17213	10101	3746	488	353	323	315	315
$L1-I acce	459568	459568	459568	459568	459568	459568	459568	459568
$L1-I miss	0.03746	0.02198	0.00815	0.00106	0.00077	0.0007	0.00069	0.00069
L1-I assoc	2	2	2	2	2	2	2	2
L1-I size	1024	2048	4096	8192	16384	32768	65536	131072
L1-Data-Cache summary								
$L1-D hit	114792	114792	114792	114792	114792	114792	114792	114792
$L1-D mi	6705	6705	6705	6705	6705	6705	6705	6705
$L1-D acc	121497	121497	121497	121497	121497	121497	121497	121497
$L1-D mi	0.05519	0.05519	0.05519	0.05519	0.05519	0.05519	0.05519	0.05519
L1-D asso	2	2	2	2	2	2	2	2
L1-D size	1024	1024	1024	1024	1024	1024	1024	1024





PART2

Part2 – Find the Optimal Config

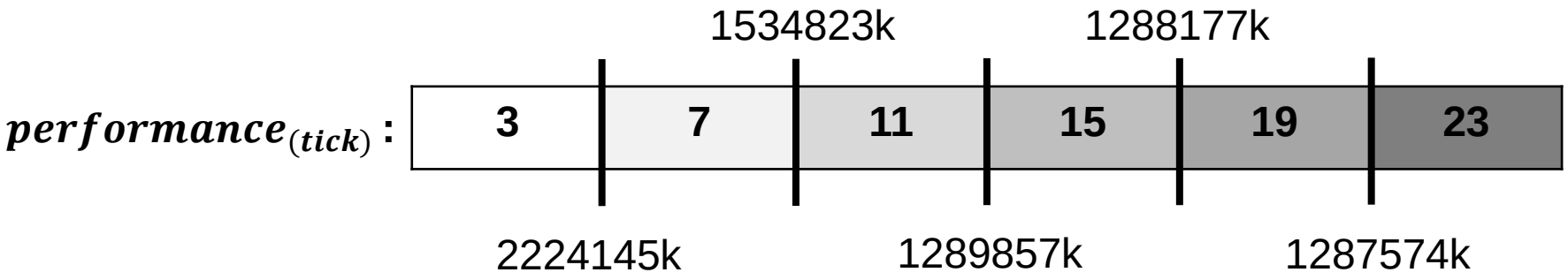
- Find **the optimal config settings** for this program
- **Cache size range is constrained** (refer to p.13)
- Minimize execution time
- Save your optimal config in **gem5_args.conf**

```
Original array:
403 623 817 57 210 808 842 410 329 301 46 8 889 470 320 469 625 243 630 883 638 157 235 102 866 665 788 108 63 597 695 818 572 864 876 783 672 70 545 2 37
1 943 362 612 414 34 433 39 629 64 274 268 221 509 722 87 175 511 547 590 460 242 761 32 106 637 167 131 707 64 485 430 8 847 42 774 234 828 813 863 892 4
39 131 465 949 206 905 476 717 452 66 177 47 179 561 153 816 81 636 875 145 121 305 505 321 700 279 555 880 444 418 772 884 902 589 833 108 494 661 177 94
7 727 706 994 259 619 499 75

Sorted array:
2 8 8 32 34 39 42 46 47 57 63 64 64 66 70 75 81 87 102 106 108 108 121 131 131 145 153 157 167 175 177 177 179 206 210 221 234 235 242 243 259 268 274 279
 301 305 320 321 329 362 371 403 410 414 418 430 433 439 444 452 460 465 469 470 476 485 494 499 505 509 511 545 547 555 561 572 589 590 597 612 619 623 6
25 629 630 636 637 638 661 665 672 695 700 706 707 717 722 727 761 772 774 783 788 808 813 816 817 818 828 833 842 847 863 864 866 875 876 880 883 884 889
 892 902 905 943 947 949 994

sorted numbers: 128
Exiting @ tick 4424434000 because exiting with last active thread context
Emulated merge sort  on gem5 with arguments: --isa type 64 --l1i size 1kB --l1i assoc 2 --l1d size 1kB --l1d assoc 2 --l2 size 16kB --l2 assoc 4
```

➢ make gem5 \
GEM5_ARGS=--isa_type 32 --l1i_size 1kB --l1i_assoc 2 --l1d_size 1kB --l1d_assoc 2 --l2_size 16kB --l2_assoc 4"

Part2 – Find the Optimal Config

- Find **the optimal config settings** for this program
- **Cache size range is constrained** (refer to p.13)
- Minimize execution time
- Save your optimal config in **gem5_args.conf**

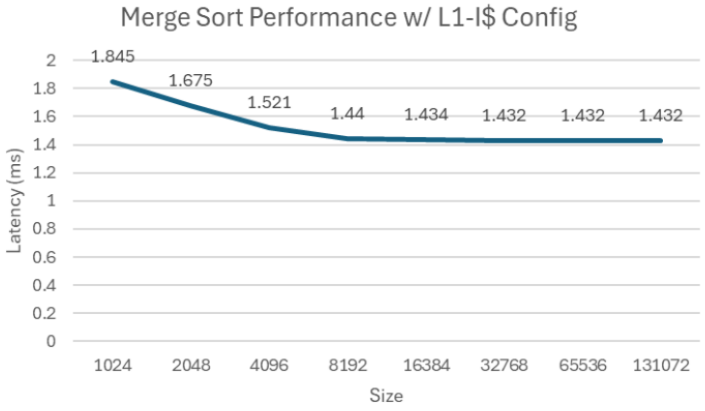


$$Score = \boldsymbol{performance}_{(tick)} - log_2\left(\frac{L1\ I\$\ size}{1KB}\right) - log_2\left(\frac{L1\ D\$\ size}{1KB}\right) - log_2\left(\frac{L2\$\ size}{1KB}\right)$$

- make gem5 \
GEM5_ARGS=--isa_type 32 --l1i_size 1kB --l1i_assoc 2 --l1d_size 1kB --l1d_assoc 2 --l2_size 16kB --l2_assoc 4"

HW3 Report Format

- Use provided **report template** to write your report
- Part1:
  - Implement L2 cache in gem5 config
  - Plot the performance trend based on each config



Range
- **Cache Size: 1KB ~ 128KB**
- **Associativity: 1 ~ 8**
- **Unit: ms**

 # Note: power of 2

- Part2: your settings and execution time
  - Find the optimal config settings for this program
  - Explain why this config achieves optimal performance

HW3 Submission

- **Deadline: 5/18 (Sun.) 23:59**
- Upload <student_id>_hw3.zip and Report.pdf to NTU COOL

(e.g. bxxxxxxxx_hw3.zip)

- <student_id>_hw3.zip
  - <student_id>_hw3/
    - simple-riscv-mod-config.py
    - gem5_args.conf
- Submit to NTU COOL

- Wrong file name or format would get **10%** penalty each

HW3 Grading Policy

- Part1
  - Implement L2 cache in gem5 config (30%)
  - Draw and analyze the program results based on the cache config (40%)

- Part2
  - Find the optimal config settings for this program (10%)
  - Explain why this config achieves optimal performance (20%)

HW3 Grading Policy

- Evaluate score based on report
  - Part1(30%+40%)+Part2(10%+20%) =100%
- If the result of your **code does not match the report**, your score for that question will be **reduced by 50%.**
- **-10%** for any wrong file name or format for submission
- **No grade**
  - **Late submission**
  - **plagiarism**
- If your have any problem on hw3, ask question through NTUCOOL 討論區 or send email to **john@access.ee.ntu.edu.tw** with the subject starting with '[Computer Architecture]'