Report for CS223 Assignment

Group 24

230101012 - Annigandla Kameshwara Rao

230101044 - Gunje Manideep Ram

1. **Changes Made to Gem5 Simulator**
    a. **Document all changes made to the gem5 simulator.**

- First, we downloaded the gem5 and scons folders and put them in one directory.
- Then we have installed as instructed in the tutorial.
- Next to build we have changed the paths to our respective folders viz:

- a) BoolVec.hh
- b) DataBlock.hh
- c) Set.hh
- d) TBETable.hh
- e) TimerTable.hh
- f) WriteMask.hh
-
- Hello world program was successfully made run.
- Next up we downloaded the benchmark cpu2017 zip folder.
- Then we followed these steps:
- Move the CPU-2017.py file to the common directory within gem5/configs.
- Update the CPU-2017.py configuration file to reflect the correct paths for the benchmark binaries and data.
- Import the CPU-2017.py file into the se.py script.

b. Indicate the files modified and the purpose of each change.

Key Changes Made:

CacheMemory.hh:

- We have declared some variables in stats struct: vectors and scalars to print in the stats file.
- Then initialized a c++ map to track set access.
- Also, we have declared some functions to get top 5 accesses, per set hits, per set misses, hits and misses between intervals etc.

CacheMemory.cc:

- Used the declared functions and added the stat variables in ADD STATS lines, also initialized the vectors in CacheMemory.init() function with no of sets.
- Functions to calculate the required data were neatly written.
- You can verify them in the uploaded files.

RubySlicc_Types.sm:

- Declared the newly added functions in this file along with the old ones.

MESI_Two_Level-L1cache.sm:

- Called the functions some with parameter address in the required action commands.

MESI_Two_Level-L2cache.sm:

- Called the functions some with parameter address in the required action commands.
- Build the processor again and simulated the benchmarks

c. Provide justifications for the modifications and their expected impact.

- Created functions to track hits and misses only between T1 and T2 with new variables to not disturb the old total misses and total hits using curTick() function.
- Created a function set access which takes address as input and calculates the set, and adds its frequency then sorted the data in a new vector and stored top 5 sets.
- Created new functions to track each set hit and set misses separately and stored them.
- Called the set access functions whenever a cache is accessed by processor – Called in action commands, meanwhile called hits and misses sets functions only when hits and misses occurred respectively

```
  int m_cache_size;
  int m_cache_num_sets;
  int m_cache_num_set_bits;
  int m_cache_assoc;
  int m_start_index_bit;
  bool m_resource_stalls;
  int m_block_size;
  std::map<int,int> myaccess;
```

```
action(uu_profileMiss, "\um", desc="Profile the demand miss") {
  L2cache.profileDemandMiss();
   L2cache.profileMyDemandMiss();
  L2cache.setaccessdata(address);
   L2cache.setaccessdatamisses(address);
}

action(uu_profileHit, "\uh", desc="Profile the demand hit") {
  L2cache.profileDemandHit();
   L2cache.profileMyDemandHit();
  L2cache.setaccessdata(address);
   L2cache.setaccessdatahits(address);
}
```

```
void
CacheMemory::profileDemandHit()
{

    cacheMemoryStats.m_demand_hits++;
}

void
CacheMemory::profileDemandMiss()
{

    cacheMemoryStats.m_demand_misses++;
}

void
CacheMemory::profileMyDemandHit()
{

    if((curTick()>=(6500000LL*500)) && (curTick()<=(7000000LL*500))){

        cacheMemoryStats.my_demand_hits++;
    }
}

void
CacheMemory::profileMyDemandMiss()
{
    if((curTick()>=(6500000LL*500)) && (curTick()<=(7000000LL*500))){
        cacheMemoryStats.my_demand_misses++;
    }
}
```

```
        statistics::Scalar m_demand_hits;
        statistics::Scalar m_demand_misses;

        statistics::Scalar my_demand_hits;
        statistics::Scalar my_demand_misses;

        statistics::Formula m_demand_accesses;

        statistics::Scalar m_prefetch_hits;
        statistics::Scalar m_prefetch_misses;
        statistics::Formula m_prefetch_accesses;

        statistics::Vector m_accessModeType;

        statistics::Vector setAccessCounter;

        statistics::Vector setAccessHits;
        statistics::Vector setAccessMisses;

        statistics::Vector setAccessTop5Number;
        statistics::Vector setAccessTop5Count;

        // statistics::Scalar m_prefetch_misses;

        // statistics::Scalar noofsets;

        // statistics::Scalar mapsize;


    } cacheMemoryStats;

  public:
    // These function increment the number of demand hits/misses by one
    // each time they are called
    void profileDemandHit();
    void profileDemandMiss();
    void profileMyDemandHit();
    void profileMyDemandMiss();
    void profilePrefetchHit();
    void profilePrefetchMiss();
    void setaccessdata(Addr address);
    void setaccessdatahits(Addr address);
    void setaccessdatamisses(Addr address);
};
```

```
structure (CacheMemory, external = "yes") {
  bool cacheAvail(Addr);
  Addr cacheProbe(Addr);
  AbstractCacheEntry getNullEntry();
  AbstractCacheEntry allocate(Addr, AbstractCacheEntry);
  AbstractCacheEntry allocate(Addr, AbstractCacheEntry, bool);
  void allocateVoid(Addr, AbstractCacheEntry);
  void deallocate(Addr);
  AbstractCacheEntry lookup(Addr);
  bool isTagPresent(Addr);
  Cycles getTagLatency();
  Cycles getDataLatency();
  void setMRU(Addr);
  void setMRU(Addr, int);
  void setMRU(AbstractCacheEntry);
  void recordRequestType(CacheRequestType, Addr);
  bool checkResourceAvailable(CacheResourceType, Addr);

  // hardware transactional memory
  void htmCommitTransaction();
  void htmAbortTransaction();

  int getCacheSize();
  int getNumBlocks();
  Addr getAddressAtIdx(int);

  void profileDemandHit();
  void profileDemandMiss();
  void profileMyDemandHit();
  void profileMyDemandMiss();
  void profilePrefetchHit();
  void profilePrefetchMiss();
  void setaccessdata(Addr address);
  void setaccessdatahits(Addr address);
  void setaccessdatamisses(Addr address);
}
```

```
action(uu_profileInstMiss, "\uim", desc="Profile the demand miss") {
  L1Icache.profileDemandMiss();
  L1Icache.profileMyDemandMiss();
  L1Icache.setaccessdata(address);
  L1Icache.setaccessdatamisses(address);

}

action(uu_profileInstHit, "\uih", desc="Profile the demand hit") {
  L1Icache.profileDemandHit();
  L1Icache.profileMyDemandHit();
  L1Icache.setaccessdata(address);
  L1Icache.setaccessdatahits(address);

}

action(uu_profileDataMiss, "\udm", desc="Profile the demand miss") {
  L1Dcache.profileDemandMiss();
  L1Dcache.profileMyDemandMiss();
  L1Dcache.setaccessdata(address);
  L1Dcache.setaccessdatamisses(address);

}

action(uu_profileDataHit, "\udh", desc="Profile the demand hit") {
  L1Dcache.profileDemandHit();
  L1Dcache.profileMyDemandHit();
  L1Dcache.setaccessdata(address);
  L1Dcache.setaccessdatahits(address);

}
```

```
    ADD_STAT(m_demand_hits, "Number of cache demand hits"),
    ADD_STAT(m_demand_misses, "Number of cache demand misses"),

    ADD_STAT(my_demand_hits, "Number of cache demand hits between T1 and T2"),
    ADD_STAT(my_demand_misses, "Number of cache demand misses between T1 and T2"),

    ADD_STAT(m_demand_accesses, "Number of cache demand accesses",
             m_demand_hits + m_demand_misses),
    ADD_STAT(m_prefetch_hits, "Number of cache prefetch hits"),
    ADD_STAT(m_prefetch_misses, "Number of cache prefetch misses"),
    ADD_STAT(m_prefetch_accesses, "Number of cache prefetch accesses",
             m_prefetch_hits + m_prefetch_misses),
    ADD_STAT(m_accessModeType, ""),

    ADD_STAT(setAccessCounter, " Set Frequency"),
    ADD_STAT(setAccessHits, " Set Hit Frequency"),
    ADD_STAT(setAccessMisses, " Set Miss Frequency"),
    ADD_STAT(setAccessTop5Number, " Sets of Top 5 Number"),
    ADD_STAT(setAccessTop5Count, " Sets of Top 5 corresponding Frequency")
```

```
void
CacheMemory::init()
{
    if (m_block_size == 0) {
        m_block_size = RubySystem::getBlockSizeBytes();
    }
    m_cache_num_sets = (m_cache_size / m_cache_assoc) / m_block_size;


    assert(m_cache_num_sets > 1);
    m_cache_num_set_bits = floorLog2(m_cache_num_sets);
    assert(m_cache_num_set_bits > 0);


    cacheMemoryStats.setAccessCounter
        .init(m_cache_num_sets);

    cacheMemoryStats.setAccessHits
        .init(m_cache_num_sets);
    cacheMemoryStats.setAccessMisses
        .init(m_cache_num_sets);


    m_cache.resize(m_cache_num_sets,
                   std::vector<AbstractCacheEntry*>(m_cache_assoc, nullptr));
    replacement_data.resize(m_cache_num_sets,
                            std::vector<ReplData>(m_cache_assoc, nullptr));
    // instantiate all the replacement_data here
    for (int i = 0; i < m_cache_num_sets; i++) {
        for ( int j = 0; j < m_cache_assoc; j++) {
            replacement_data[i][j] =
                              m_replacementPolicy_ptr->instantiateEntry();
        }
    }

}
```

```
void
CacheMemory::setaccessdata(Addr add)
{
    int64_t cacheSet = addressToCacheSet(add);
    cacheMemoryStats.setAccessCounter[cacheSet]++;
    myaccess[cacheSet]++;

    std::vector<std::pair<int, int>> freqVec(myaccess.begin(),myaccess.end());


    std::sort(freqVec.begin(), freqVec.end(), [](auto &a, auto &b) {
        return a.second > b.second;
    });

    for (int i = 0; i < 5 && i < freqVec.size(); ++i) {
        cacheMemoryStats.setAccessTop5Number[i] = freqVec[i].first;
        cacheMemoryStats.setAccessTop5Count[i] = freqVec[i].second;
    }


}


void
CacheMemory::setaccessdatahits(Addr add)
{
    int64_t cacheSet = addressToCacheSet(add);
    cacheMemoryStats.setAccessHits[cacheSet]++;

}

void
CacheMemory::setaccessdatamisses(Addr add)
{
    int64_t cacheSet = addressToCacheSet(add);
    cacheMemoryStats.setAccessMisses[cacheSet]++;

}
```

2. **Experimental Data Collection**
    a. Present results separately for each benchmark.
    b. Use the tables below for structured data representation.

Experiment-2

a)

Parest C4 Parameter Set 1

S.No	Statistics	Count
1	Total L1-D hits	827411
2	Total L1-D miss	23976
3	L1-D hits between T1 and T2	34780
4	L1-D miss between T1 and T2	1369

Parest C4 Parameter Set 2

S.No	Statistics	Count
1	Total L1-D hits	829609
2	Total L1-D miss	21778
3	L1-D hits between T1 and T2	35577
4	L1-D miss between T1 and T2	1326

Sjeng C4 Parameter Set 1

S.No	Statistics	Count
1	Total L1-D hits	9894585
2	Total L1-D miss	1410440
3	L1-D hits between T1 and T2	22512
4	L1-D miss between T1 and T2	3216

Sjeng C4 Parameter Set 2

S.No	Statistics	Count
1	Total L1-D hits	9894585
2	Total L1-D miss	1410440
3	L1-D hits between T1 and T2	22505
4	L1-D miss between T1 and T2	3215

b)

Parest C4 Parameter Set 1

Set Number	L1D Hits
0	8220
1	4842
2	6926
3	3477
4	8175

5	7349
6	5804
7	8347
8	15199
9	3731
10	4494
11	3518
12	4618
13	7412
14	4265
15	16559
16	3515

17	4122
18	12881
19	8814
20	5214
21	4343
22	8835
23	10184
24	15370
25	4851
26	4714
27	4161
28	16947

29	27182
30	13227
31	19472
32	31399
33	21580
34	16901
35	19199
36	13152
37	23653
38	4490
39	31860
40	94126

41	23654
42	40793
43	21600
44	29265
45	14707
46	30749
47	11736
48	20606
49	10177
50	6380
51	10544
52	7057

53	11584
54	11922
55	5615
56	5476
57	8213
58	3877
59	2923
60	3014
61	2917
62	8412
63	3062

Set Number	L1D Misses

0	362
1	373
2	391
3	350
4	433
5	415
6	455
7	411
8	377
9	369
10	351
11	375

12	372
13	410
14	355
15	363
16	361
17	387
18	380
19	362
20	364
21	377
22	365
23	392

24	358
25	360
26	392
27	364
28	393
29	381
30	375
31	383
32	380
33	364
34	370
35	374

36	350
37	346
38	374
39	376
40	408
41	363
42	418
43	391
44	394
45	352
46	362
47	359

48	368
49	371
50	367
51	366
52	362
53	350
54	376
55	357
56	358
57	405
58	391
59	341

60	361
61	333
62	359
63	374

Parest C4 Parameter Set 2

Set Number	L1D Hits
0	2251
1	2311
2	2019
3	1984
4	1708

5	1778
6	1576
7	2649
8	12505
9	1465
10	2282
11	1439
12	2900
13	3910
14	1372
15	14095
16	1497

17	1478
18	1421
19	1490
20	2413
21	2091
22	1255
23	3496
24	10745
25	1679
26	2688
27	2433
28	12127

29	11658
30	1519
31	2026
32	10134
33	1702
34	1141
35	1517
36	1191
37	1357
38	1379
39	1276
40	1377

41	1366
42	24272
43	4410
44	5202
45	3329
46	3224
47	3227
48	3099
49	5179
50	3079
51	3408
52	3585

53	3424
54	3698
55	3460
56	3112
57	3406
58	2285
59	1170
60	1229
61	1099
62	6627
63	1394
64	5996

65	2580
66	4939
67	1510
68	6521
69	5626
70	4297
71	5746
72	2728
73	2309
74	2234
75	2121
76	1753

77	3554
78	2916
79	2488
80	2053
81	2700
82	11497
83	7353
84	2843
85	2290
86	7607
87	6728
88	4650

89	3200
90	2059
91	1759
92	4867
93	15558
94	11740
95	17480
96	21282
97	19906
98	15788
99	17721
100	11986

101	22320
102	3138
103	30615
104	92799
105	22319
106	16574
107	17240
108	24132
109	11413
110	27553
111	8534
112	17533

113	5030
114	3332
115	7163
116	3492
117	8178
118	8245
119	2177
120	2384
121	4849
122	1635
123	1779
124	1818

125	1844
126	1811
127	1699

Set Number	L1D Misses
0	167
1	161
2	175
3	162
4	171
5	166
6	176

7	171
8	174
9	158
10	167
11	167
12	181
13	185
14	157
15	171
16	160
17	164
18	167

19	156
20	158
21	166
22	154
23	158
24	162
25	168
26	183
27	165
28	164
29	161
30	149

31	159
32	163
33	145
34	151
35	154
36	139
37	149
38	165
39	160
40	170
41	155
42	180

43	174
44	151
45	148
46	158
47	153
48	149
49	150
50	159
51	161
52	161
53	152
54	180

55	163
56	147
57	161
58	165
59	140
60	150
61	138
62	143
63	161
64	168
65	163
66	184

67	171
68	208
69	194
70	210
71	192
72	169
73	168
74	162
75	166
76	156
77	173
78	175

79	168
80	166
81	167
82	176
83	177
84	164
85	173
86	184
87	194
88	171
89	164
90	176

91	168
92	182
93	186
94	194
95	190
96	200
97	191
98	191
99	181
100	186
101	173
102	182

103	185
104	188
105	177
106	185
107	167
108	174
109	169
110	176
111	181
112	193
113	189
114	177

115	178
116	181
117	180
118	175
119	172
120	191
121	202
122	183
123	175
124	178
125	169
126	190

127	182

Sjeng C4 Parameter Set 1

Set Number	L1D Hits
0	154325
1	154329
2	154322
3	154330
4	154413
5	154441
6	154474
7	154466
8	154394

9	154406
10	154381
11	154457
12	154399
13	154514
14	154384
15	154361
16	154405
17	154416
18	154384
19	154341
20	154343

21	154343
22	154350
23	154343
24	154343
25	154333
26	154333
27	154355
28	154359
29	154360
30	154392
31	154359
32	154390

33	154379
34	154366
35	154371
36	154377
37	154448
38	154454
39	154456
40	154449
41	154427
42	157534
43	157610
44	154546

45	154562
46	154563
47	156959
48	156043
49	154802
50	154894
51	154721
52	154729
53	154733
54	154686
55	154651
56	154620

57	154620
58	154464
59	154312
60	154316
61	154316
62	154316
63	154316

Set Number	L1D Misses
0	22038
1	22038
2	22038
3	22038

4	22039
5	22040
6	22040
7	22042
8	22040
9	22040
10	22040
11	22039
12	22039
13	22039
14	22038
15	22039

16	22039
17	22040
18	22039
19	22038
20	22038
21	22038
22	22039
23	22039
24	22039
25	22039
26	22037
27	22037

28	22037
29	22038
30	22039
31	22037
32	22038
33	22038
34	22038
35	22037
36	22036
37	22038
38	22038
39	22040

40	22040
41	22037
42	22040
43	22037
44	22037
45	22037
46	22037
47	22038
48	22037
49	22037
50	22038
51	22038

52	22039
53	22039
54	22038
55	22038
56	22037
57	22036
58	22036
59	22036
60	22036
61	22037
62	22036
63	22036

Sjeng C4 Parameter Set 2

Set Number	L1D Hits
0	77201
1	77203
2	77196
3	77204
4	77287
5	77315
6	77348
7	77340
8	77268
9	77276
10	77250

11	77331
12	77255
13	77378
14	77217
15	77217
16	77217
17	77217
18	77219
19	77215
20	77217
21	77217
22	77217

23	77217
24	77217
25	77200
26	77207
27	77229
28	77233
29	77234
30	77233
31	77233
32	77233
33	77222
34	77213

35	77221
36	77227
37	77241
38	77241
39	77242
40	77235
41	77214
42	77196
43	77199
44	77183
45	77187
46	77183

47	79582
48	78709
49	77500
50	77503
51	77220
52	77197
53	77201
54	77154
55	77119
56	77119
57	77119
58	77119

59	77119
60	77119
61	77119
62	77119
63	77119
64	77124
65	77126
66	77126
67	77126
68	77126
69	77126
70	77126

71	77126
72	77126
73	77130
74	77131
75	77126
76	77144
77	77136
78	77167
79	77144
80	77188
81	77199
82	77165

83	77126
84	77126
85	77126
86	77133
87	77126
88	77126
89	77133
90	77126
91	77126
92	77126
93	77126
94	77159

95	77126
96	77157
97	77157
98	77153
99	77150
100	77150
101	77207
102	77213
103	77214
104	77214
105	77213
106	80338

107	80411
108	77363
109	77375
110	77380
111	77377
112	77334
113	77302
114	77391
115	77501
116	77532
117	77532
118	77532

119	77532
120	77501
121	77501
122	77345
123	77193
124	77197
125	77197
126	77197
127	77197

Set Number	L1D Misses
0	11019
1	11020

2	11020
3	11020
4	11021
5	11022
6	11022
7	11023
8	11022
9	11021
10	11021
11	11021
12	11021
13	11021

14	11020
15	11020
16	11020
17	11020
18	11020
19	11020
20	11020
21	11020
22	11020
23	11021
24	11021
25	11020

26	11019
27	11019
28	11019
29	11020
30	11019
31	11019
32	11019
33	11019
34	11019
35	11019
36	11018
37	11019

38	11019
39	11020
40	11020
41	11018
42	11020
43	11017
44	11017
45	11017
46	11017
47	11018
48	11018
49	11018

50	11019
51	11019
52	11019
53	11019
54	11018
55	11017
56	11017
57	11017
58	11017
59	11017
60	11017
61	11017

62	11017
63	11017
64	11019
65	11018
66	11018
67	11018
68	11018
69	11018
70	11018
71	11019
72	11018
73	11019

74	11019
75	11018
76	11018
77	11018
78	11018
79	11019
80	11019
81	11020
82	11019
83	11018
84	11018
85	11018

86	11019
87	11018
88	11018
89	11019
90	11018
91	11018
92	11018
93	11018
94	11020
95	11018
96	11019
97	11019

98	11019
99	11018
100	11018
101	11019
102	11019
103	11020
104	11020
105	11019
106	11020
107	11020
108	11020
109	11020

110	11020
111	11020
112	11019
113	11019
114	11019
115	11019
116	11020
117	11020
118	11020
119	11021
120	11020
121	11019

122	11019
123	11019
124	11019
125	11020
126	11019
127	11019

c. Parest C4 Parameter Set 1 Top 5 Frequency Count

Set number	Hit Count	Miss Count	Total
40	94126	408	94534
42	40793	418	41211
39	31860	376	32236
32	31399	380	31779
46	30749	362	31111

Parest C4 Parameter Set 2 Top 5 Frequency Count

Set number	Hit Count	Miss Count	Total
104	92799	188	92987

103	30615	185	30800
110	27553	176	27729
42	24272	180	24452
108	24132	174	24306

Sjeng C4 Parameter Set 1 Top 5 Frequency Count

Set number	Hit Count	Miss Count	Total
43	157610	22037	179647
42	157534	22040	179574
47	156959	22038	178997
48	156043	22037	178080
50	154894	22038	176932

Sjeng C4 Parameter Set 2 Top 5 Frequency Count

Set number	Hit Count	Miss Count	Total
107	80411	11020	91431
106	80338	11020	91358
47	79582	11018	90600
48	78709	11018	89727
119	77532	11021	88553

3. **Analysis & Conclusion**
    a. Summarize key findings from the experiment.

In sjeng benchmark all sets were equally accessed (approximately) making the program spread equally over the data cache in both the cases of change in size of it by a factor of 2.

Coming to overall hits and misses, the overall hits and misses were same even if we double the size, maybe the capacity misses were not there so that they didn't change. But the hit time has increased because the size increased.

Set 43(C4 1) and Set 107(C4 2) were most frequently accessed in respective parameters.

In parest benchmark different sets were accessed at different frequencies making the program spread some areas over the data cache in both the cases of change in size of it by a factor of 2.

Coming to overall hits and misses, the overall hits were increased, and misses were reduced if we double the size, maybe the capacity misses were reduced. But the hit time has increased because size increased.

Set 40(C4 1) and Set 104(C4 2) were most frequently accessed in respective parameters as much as (2 X difference in first and second most frequently accessed sets).

b. Discuss any observations, challenges encountered, and potential improvements.

Observations:

- Workload Behavior: Different workloads stress cache systems differently — sjeng is more uniform, while parest has hotspots in memory access patterns.
- Cache Size Trade-off: Increasing cache size doesn't always improve performance. While it can reduce capacity misses, it also increases hit latency and may consume more power/area.

- Access Pattern Insights: Frequently accessed sets help identify potential optimization areas like prefetching or replacing MRU policies with more tailored ones.

Challenges Encountered:

- Accurately tracking per-set access frequency required modifications to the gem5 simulator and verifying correctness. It was quite time-consuming.
- Maintaining compatibility with simulator outputs and integrating results into readable statistics.
- Interpreting whether changes in hit/miss counts were due to cache size or access pattern variation.

Potential Improvements:

- Investigate prefetching strategies or set-level priority management for sets that are accessed very frequently.
- Reducing the set associativity to reduce conflict misses.
- Extend analysis to L2 or LLC to understand interaction across cache hierarchy.