Project Report - Pranjal Walia (IMT2019062)

Cache Size = 256kb or (2^{18}) bytes , Block size = 4 bytes

- Block size = 4 bytes implies 2 bit offset for both caches.

  Direct Mapped Cache:

  a. Index bits required to address $\Rightarrow 2^{18}/4 \Rightarrow 2^{16}$ locations $\Rightarrow 16$ bits
  b. Offset $\Rightarrow 2$ bits
  c. Tag $\Rightarrow 32 - 16 - 2 \Rightarrow 14$ bits

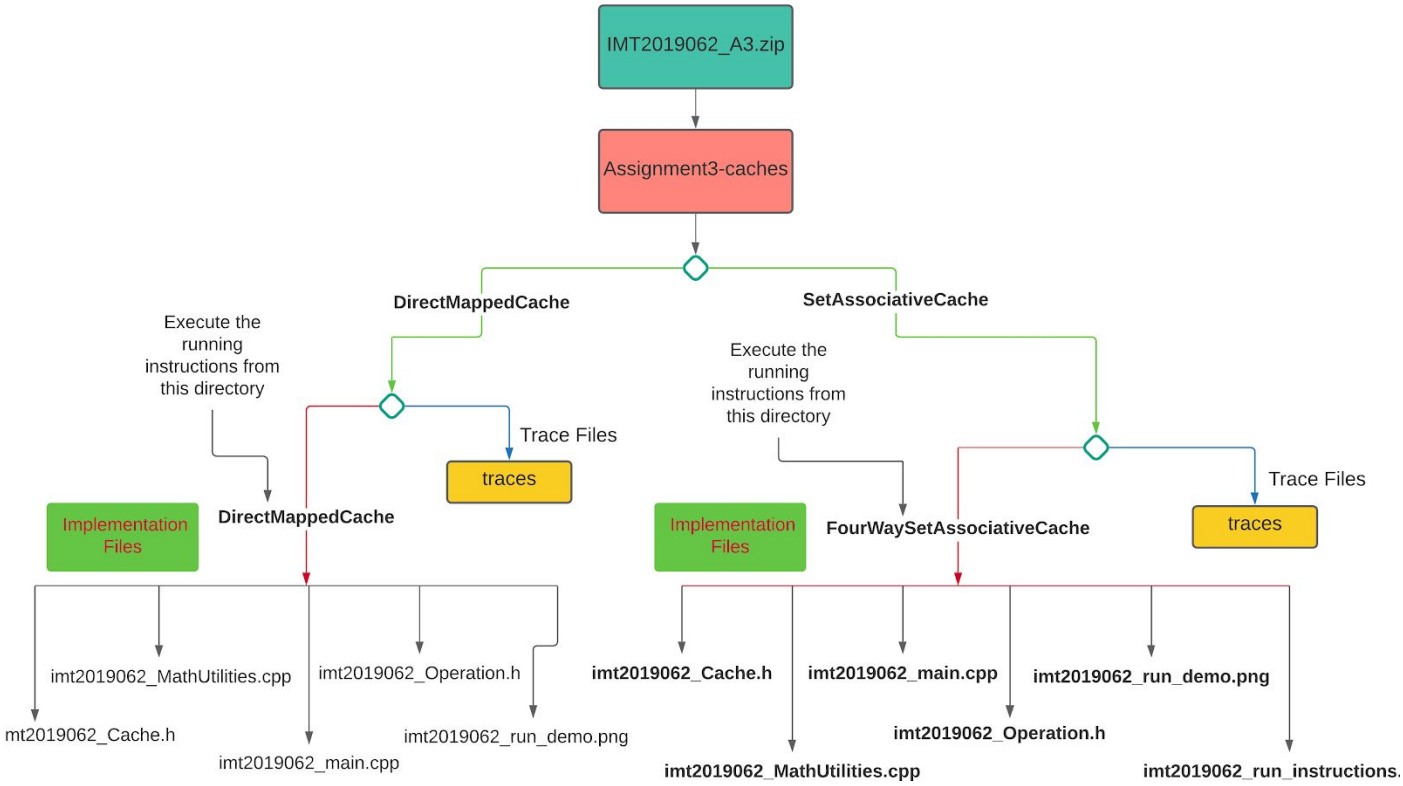
  Four Way Set Associative Cache:

  a. Index bits required to address $\Rightarrow 2^{18}/(4 * ways) \Rightarrow 2^{14}$ locations $\Rightarrow 14$ bits
  b. Offset $\Rightarrow 2$ bits
  c. Tag $\Rightarrow 32 - 14 - 2 \Rightarrow 16$ bits

Final Cache Parameters:

1. **Direct Mapped Cache:**
   - **Tag:** 14 bits
   - **Index:** 16 bits
   - **OffSet:** 2 bits
2. **Four Way Set Associative Cache:**
   - **Tag:** 16 bits
   - **Index:** 14 bits
   - **OffSet:** 2 bits

File Structure:



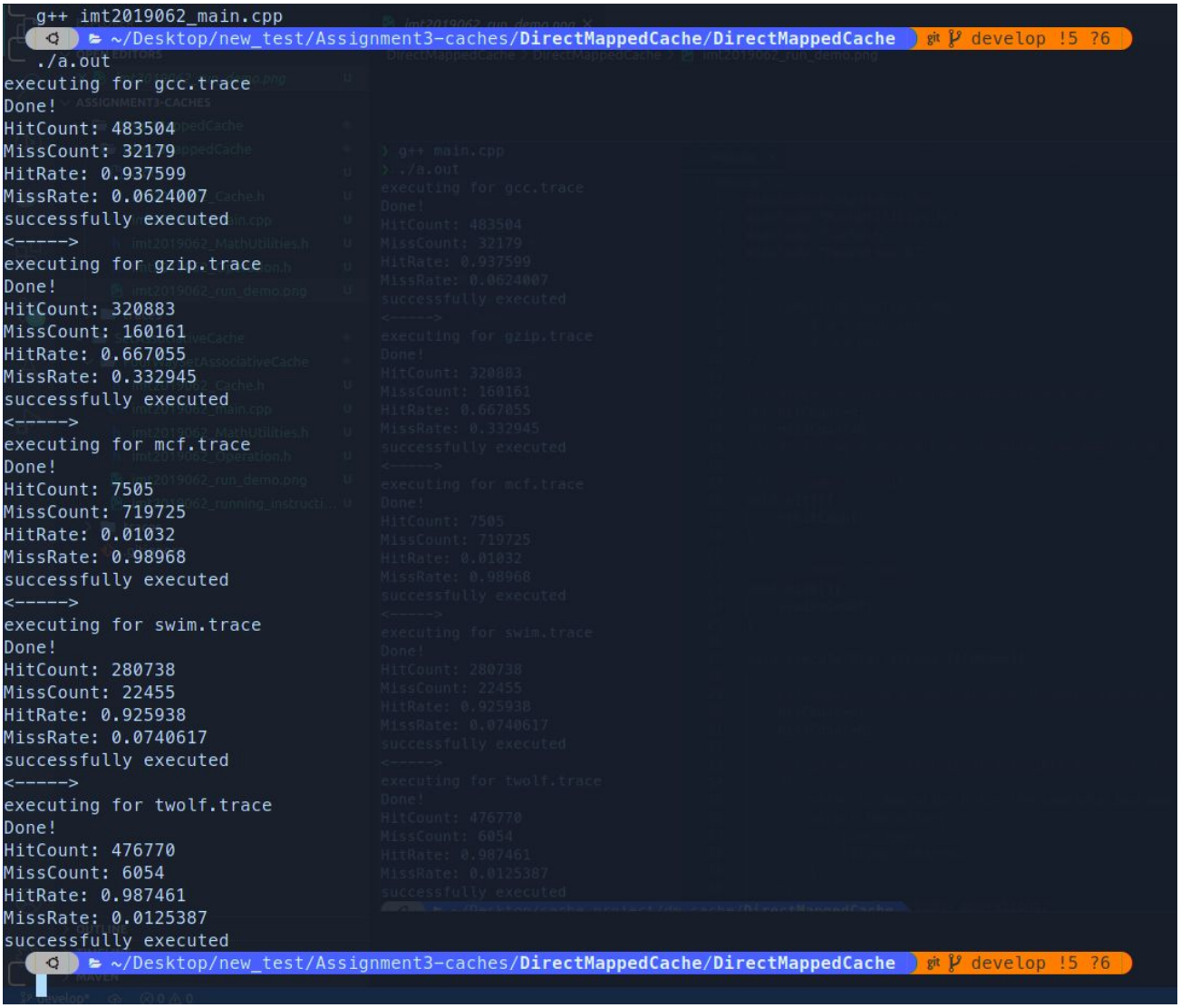
File Structure as from inside the main directory (Assignment3-caches) as illustrated from the flow chart:

1. ./DirectMappedCache/DirectMappedCache/
    a. Contains all the code for the implementation of the direct mapped cache and relevant screenshots of the execution.
    b. Files Contained:
        i. imt2019062_Cache.h
        ii. imt2019062_Operation.h
        iii. imt2019062_MathUtilities.h
        iv. imt2019062_main.cpp
        v. imt2019062_run_demo.png
2. ./DirectMappedCache/traces/

    a. Contans the trace files that are to be used during the execution of the direct mapped cache.
3. ./SetAssociativeCache/FourWaySetAssociativeCache/
    a. Contains all the code for the implementation of the four way set associative cache.
    b. Files Contained:
        i. imt2019062_Cache.h
        ii. imt2019062_Operation.h
        iii. imt2019062_MathUtilities.h
        iv. imt2019062_main.cpp
        v. imt2019062_run_demo.png
        vi. imt2019062_run_instruction.png
4. ./SetAssociativeCache/traces/
    a. Contains all the trace files that are to be used during the execution of the four way set associative cache.

Running Instructions:

1. Direct Mapped Cache:
    a. Make sure not to change any file paths as the program utilities the current set of file paths to access the trace files provided.
    b. Open the terminal
    c. Go to the directory “Assignment3-caches/DirectMappedCache/DirectMappedCache”
    d. Execute the following commands and the following execution appears:
        i. $ g++ imt2019062_main.cpp
        ii. $ ./a.out



2. Four way set associative cache
    a. Make sure not to change any file paths as the program utilities the current set of file paths to access the trace files provided.
    b. Open the terminal
    c. Go to the directory

       “Assignment3-caches/SetAssociativeCache/FourWaySetAssociativeCache”
    d. Execute the following commands:
        i. $ g++ imt2019062_main.cpp
        ii. $ ./a.out

```
~/De/new_test/Assignment3-caches/SetAssociativeCache/FourWaySetAssociativeCache  develop !6 ?6
g++ imt2019062_main.cpp
~/De/new_test/Assignment3-caches/SetAssociativeCache/FourWaySetAssociativeCache  develop !6 ?6
./a.out
executing gcc.trace ...
Done!
ways: 4
HitCount: 483871
MissCount: 31812
HitRate: 0.938311
MissRate: 0.0616891
successfully executed !
<-------->
```

e. The following execution will appear, indicating correct execution of the program.

```
pranjal@BroadSword:~/Desktop/Assignment3-caches/SetAssociativeCache/FourWaySetAssociativeCache$ ./a.out
executing gcc.trace ...
Done!
ways: 4
HitCount: 483871
MissCount: 31812
HitRate: 0.938311
MissRate: 0.0616891
successfully executed !
<-------->
executing gzip.trace ...
Done!
ways: 4
HitCount: 320883
MissCount: 160161
HitRate: 0.667055
MissRate: 0.332945
successfully executed !
<-------->
executing mcf.trace ...
Done!
ways: 4
HitCount: 7508
MissCount: 719722
HitRate: 0.0103241
MissRate: 0.989676
successfully executed !
<-------->
executing swim.trace ...
Done!
ways: 4
HitCount: 280825
MissCount: 22368
HitRate: 0.926225
MissRate: 0.0737748
successfully executed !
<-------->
executing twolf.trace ...
Done!
ways: 4
HitCount: 476844
MissCount: 5980
HitRate: 0.987615
MissRate: 0.0123855
```

Hit/Miss rates of the **Direct Mapped Cache**:

Sample Name	**Hits**	**Misses**	**Hit Rate**	**Miss Rate**
gcc.trace	483504	32179	0.937599	0.0624007
gzip.trace	320883	160161	0.667055	0.332945
mcf.trace	7505	719725	0.01032	0.98968
swim.trace	280738	22455	0.925938	0.0740617
twolf.trace	476770	6054	0.987461	0.0125387

Hit/Miss rates of the **Four Way Set Associative Cache**:

Sample Name	**Hits**	**Misses**	**Hit Rate**	**Miss Rate**
gcc.trace	483871	31812	0.938311	0.0616891
gzip.trace	320883	160161	0.667055	0.332945
mcf.trace	7508	719722	0.0103241	0.989676
swim.trace	280825	22368	0.926225	0.0737748
twolf.trace	476844	5980	0.987615	0.0123855

Observations:

1. In general the hit rate of set associative caches is greater than the same for direct mapped caches.
2. The hit and miss rates for any kind of cache implementation is largely influenced by the data that is going to be processed and importantly the order in which it is processed.

3. In the current experiment, the increase and relative increase of hit rates for the Four Way Set Associative Cache in comparison to the DirectMappedCache is as follows for the following data set:

FIleName	**Increase in Hits**	**Relative Increase in Hits**	**Relative Increase (%)**
gcc.trace	367	0.000759042	0.0759042
gzip.trace	0	0.00	0.00
mcf.trace	3	0.000399574	0.0399574
swim.trace	87	0.000309802	0.0309802
twolf.trace	74	0.000155187	0.0155187

4. The improvement in hit rate produced by the set associative cache can be large or very minute as compared to the direct mapped cache, again depending on the input data.
5. Multiple instances of a direct mapped cache can be used without any modification to code to implement a N-way set associative cache (this is limited only to code implementation)
6. The hit rate in set associative caches is also influenced by the replacement policy employed, the current version of the four-way set associative cache implements the

popular LRU (Least recently used) policy for replacement of data in the cache, other policies might include:

a. Random Replacement
b. First/Last Only
c. FIFO(first-in-first-out)
d. LIFO (last-in-first-out)

Code Review:

Following three files are **common** to both the Direct Mapped Cache and Four Way Set Associative Cache.

1. imt2019062_Cache.h:

   a. Contains the fields required to instantiate a direct mapped cache object with size (number of rows) as an attribute, initialises the required validate and tag fields according to the size.

   b. In case of a four way set associative cache, four instances of the cache object (size corresponding to the index bits in set associative cache) are created to store the validate and tag fields.

2. imt2019062_Operation.h
   a. Contains a struct to store an entire input line from the trace files, we will only be using the address field from this for our computations.
   b. Each line of input makes a new Operation object and this is pushed onto a vector of such objects, thereby parsing and storing the entire input trace file.

3. imt2019062_MathUtilities.h

    a. Contains the utility functions required for converting hexadecimal to binary and vice versa.

- The cache implementations defer in imt2019062_main.cpp as it contains the core of the implementation.
- The file calls the function execute() on all the trace files, this does the required computation for each trace file and resets the cache for running the next trace file.
- It prints the hits, misses, hit rate and miss rate in every execution of a trace file.