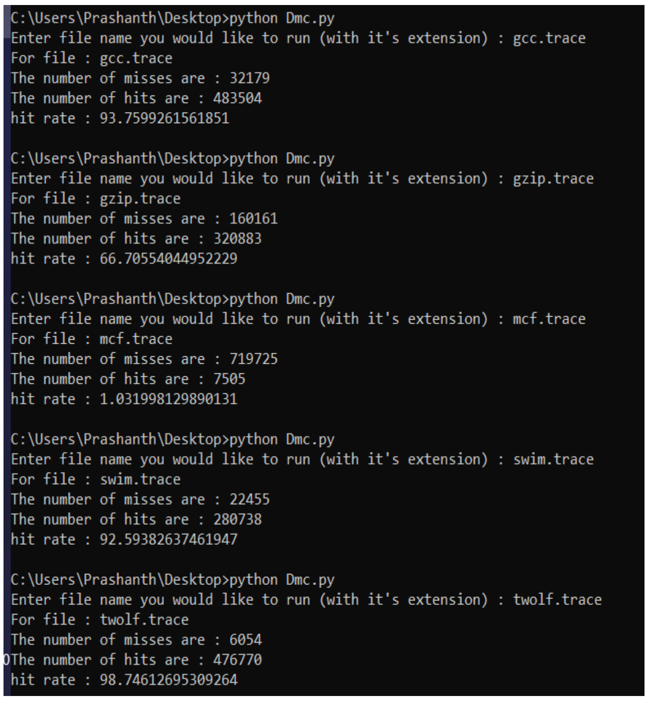
CA Assignment 3

Code file : IMT2019507_Dmc.py

For the Direct mapped cache, the outputs for the given input trace files are,

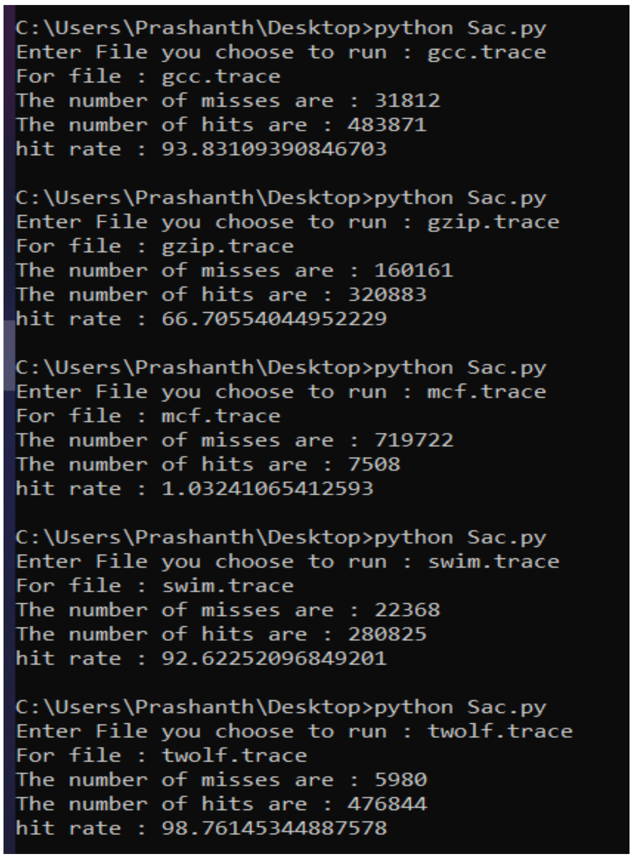


```
C:\Users\Prashanth\Desktop>python Dmc.py
Enter file name you would like to run (with it's extension) : gcc.trace
For file : gcc.trace
The number of misses are : 32179
The number of hits are : 483504
hit rate : 93.7599261561851

C:\Users\Prashanth\Desktop>python Dmc.py
Enter file name you would like to run (with it's extension) : gzip.trace
For file : gzip.trace
The number of misses are : 160161
The number of hits are : 320883
hit rate : 66.70554044952229

C:\Users\Prashanth\Desktop>python Dmc.py
Enter file name you would like to run (with it's extension) : mcf.trace
For file : mcf.trace
The number of misses are : 719725
The number of hits are : 7505
hit rate : 1.031998129890131

C:\Users\Prashanth\Desktop>python Dmc.py
Enter file name you would like to run (with it's extension) : swim.trace
For file : swim.trace
The number of misses are : 22455
The number of hits are : 280738
hit rate : 92.59382637461947

C:\Users\Prashanth\Desktop>python Dmc.py
Enter file name you would like to run (with it's extension) : twolf.trace
For file : twolf.trace
The number of misses are : 6054
The number of hits are : 476770
hit rate : 98.74612695309264
```

Code file : IMT2019507_Sac.py

For 4-way Set Associative cache, the outputs for the given input trace files are,

(Least recently used-LRU algorithm is used in this 4- way set associative cache)



```
C:\Users\Prashanth\Desktop>python Sac.py
Enter File you choose to run : gcc.trace
For file : gcc.trace
The number of misses are : 31812
The number of hits are : 483871
hit rate : 93.83109390846703

C:\Users\Prashanth\Desktop>python Sac.py
Enter File you choose to run : gzip.trace
For file : gzip.trace
The number of misses are : 160161
The number of hits are : 320883
hit rate : 66.70554044952229

C:\Users\Prashanth\Desktop>python Sac.py
Enter File you choose to run : mcf.trace
For file : mcf.trace
The number of misses are : 719722
The number of hits are : 7508
hit rate : 1.03241065412593

C:\Users\Prashanth\Desktop>python Sac.py
Enter File you choose to run : swim.trace
For file : swim.trace
The number of misses are : 22368
The number of hits are : 280825
hit rate : 92.62252096849201

C:\Users\Prashanth\Desktop>python Sac.py
Enter File you choose to run : twolf.trace
For file : twolf.trace
The number of misses are : 5980
The number of hits are : 476844
hit rate : 98.76145344887578
```

Comparing the obtained cache results

	Direct Mapped Cache	4-Way Set Associative Cache
gcc.trace	Hits: 483504 Misses:32179 Hit Rate: 93.75 %	Hits: 483871 Misses:31812 Hit Rate: 93.83 %
gzip.trace	Hits: 320883 Misses:160161 Hit Rate: 66.70 %	Hits: 320883 Misses:160161 Hit Rate: 66.70 %
mlf.trace	Hits: 7505 Misses:719725 Hit Rate: 1.03 %	Hits: 7508 Misses:719722 Hit Rate: 1.03 %
swim.trace	Hits: 280738 Misses:22455 Hit Rate: 92.59 %	Hits: 280825 Misses:22368 Hit Rate: 92.62 %
twolf.trace	Hits: 476770 Misses:6054 Hit Rate: 98.74 %	Hits: 476844 Misses:5980 Hit Rate: 98.76 %

From the above table we can observe that,

The hit rates of 4-way set associative cache are slightly higher than those of Direct mapped cache.

Using Direct mapped cache or Set associative cache doesn't guarantee us that the hit rate will be greater than or around 90 percent.

It completely depends on the input files whether the hit rates will be high or not. (Like twolf.trace file produces a very high hit rate of 98.74% in both the caches while mlf.trace produces a very low hit rate of 1.03% in both the caches)

But we can see that in some cases, both Direct mapped cache and set associative cache produce the same hit rates, like in the case of gzip.trace where the hit rate is 66.70% irrespective of the cache type used. So, we can't guarantee that using a cache will produce better hit rates for each and every file.

Direct mapped caches give out results and process faster than Set associative caches in general because Direct mapped caches only have to look at one memory location for a tag comparison, whereas in Set associative caches it has to look at an array of addresses for tag comparison… also it has a replacement policy(that's LRU in our case) which further increases the time taken to produce output.

Using different algorithms in set associative cache like LRU(Least Recently Used), FIFO(First in, First out) , LFU(Least Frequently Used) or Random doesn't make a lot of difference in terms of hit rates generally. But LRU produces the most effective results in general.

So, we can finally say that the set associative cache performs equally good if not better than the direct mapped cache depending on the input file.