University of Central Florida

Department of Computer Science

CDA 5106: Spring 2022

Machine Problem 1: Cache Design, Memory Hierarchy Design

by

Ashish Jain

Honor Pledge: "I have neither given nor received unauthorized aid on this test or assignment."

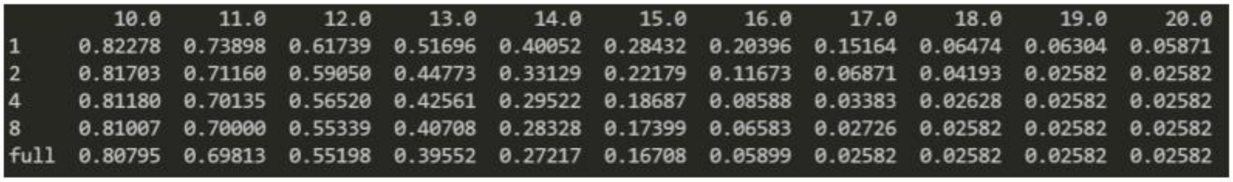
Student’s electronic signature: _______Ashish Jain_____________________

(sign by typing your name)

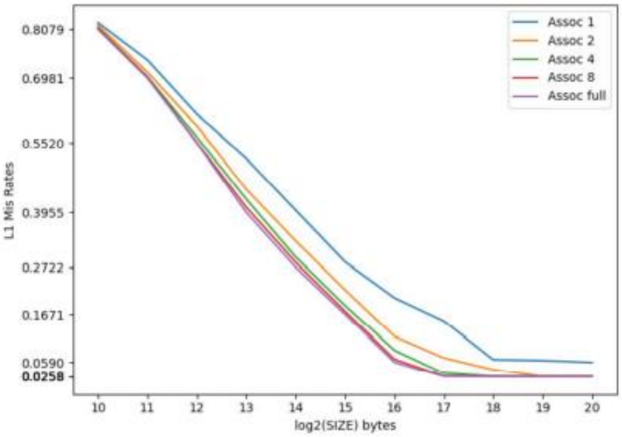
Graph1: L1 Cache Exploration on Miss Rate

Rows: Associativity

Column: Log2(Size)



```
        10.0     11.0     12.0     13.0     14.0     15.0     16.0     17.0     18.0     19.0     20.0
1     0.82278  0.73898  0.61739  0.51696  0.40052  0.28432  0.20396  0.15164  0.06474  0.06304  0.05871
2     0.81703  0.71160  0.59050  0.44773  0.33129  0.22179  0.11673  0.06871  0.04193  0.02582  0.02582
4     0.81180  0.70135  0.56520  0.42561  0.29522  0.18687  0.08588  0.03383  0.02628  0.02582  0.02582
8     0.81007  0.70000  0.55339  0.40708  0.28328  0.17399  0.06583  0.02726  0.02582  0.02582  0.02582
full  0.80795  0.69813  0.55198  0.39552  0.27217  0.16708  0.05899  0.02582  0.02582  0.02582  0.02582
```

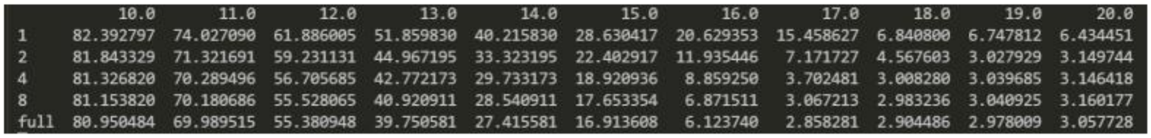


Discussion:

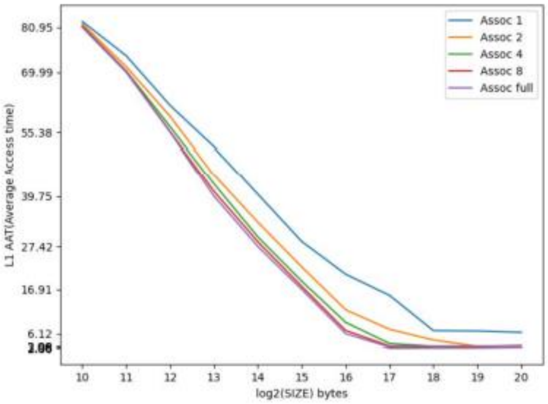
1. **Discuss trends in the graph. For a given associativity, how does increasing cache size affect miss rate? For a given cache size, what is the effect of increasing associativity?**
   For a given associativity as you increase the cache size the Miss Rate decreases. For a given cache size as you increase the associativity the Miss Rate decreas**es.**
2. **Estimate the compulsory miss rate from the graph**
   Compulsory Miss Rate is the rate when we bring blocks from the memory. Thus for our table, the larger cache size and full associativity will give us the compulsory miss rate, as it will take minimum number of blocks from memory. Compulsory Miss Rate= 0.0258**2**
3. **For each associativity, estimate the conflict miss rate from the graph**
   Conflict miss rate is due smaller cache associativity. Hence, we take average of differences with the fully associativity. Direct mapped cache = 0.078904 ,Associativity 2 = 0.031259, Associativity 4 = 0.011689, Associativity 8 = 0.003933, Fully associative cache = 0

Graph2: L1 Cache Exploration on AAT

Rows = ASSOC, Columns = Log2(SIZE)



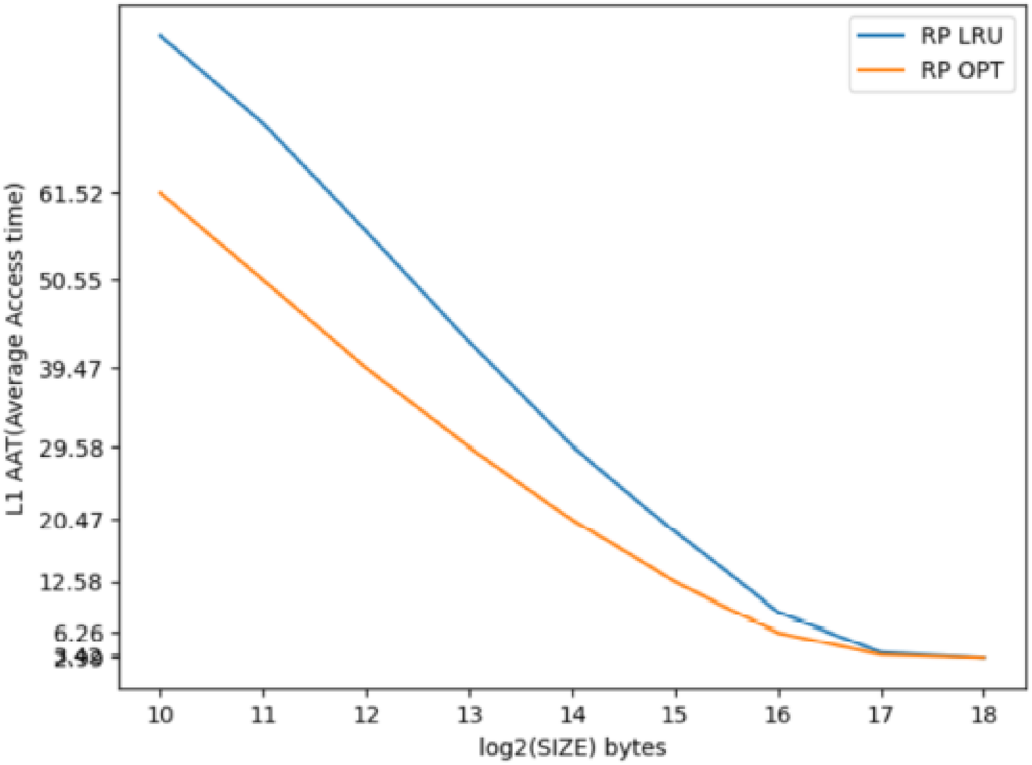
```
          10.0       11.0       12.0       13.0       14.0       15.0       16.0       17.0      18.0      19.0      20.0
1     82.392797  74.027090  61.886005  51.859830  40.215830  28.630417  20.629353  15.458627  6.840800  6.747812  6.434451
2     81.843329  71.321691  59.231131  44.967195  33.323195  22.402917  11.935446   7.171727  4.567603  3.027929  3.149744
4     81.326820  70.289496  56.705685  42.772173  29.733173  18.920936   8.859250   3.702481  3.008280  3.039685  3.146418
8     81.153820  70.180686  55.528065  40.920911  28.540911  17.653354   6.871511   3.067213  2.983236  3.040925  3.160177
full  80.950484  69.989515  55.380948  39.750581  27.415581  16.913608   6.123740   2.858281  2.904486  2.978009  3.057728
```



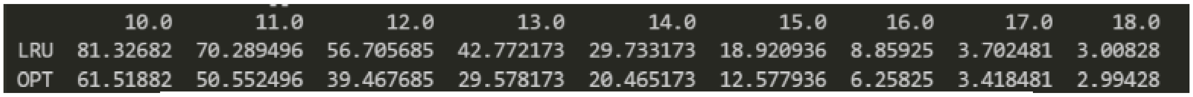
Discussion:

1. **For a memory hierarchy with only an L1 cache and BLOCKSIZE = 32, which configuration yields the best (i.e., lowest) AAT?**
   The best AAT of 2.858281 is given by Cache Size of 2^17 with fully associativity

Graph3: Replacement Policy Study



Rows = ASSOC, Columns = Log2(SIZE)

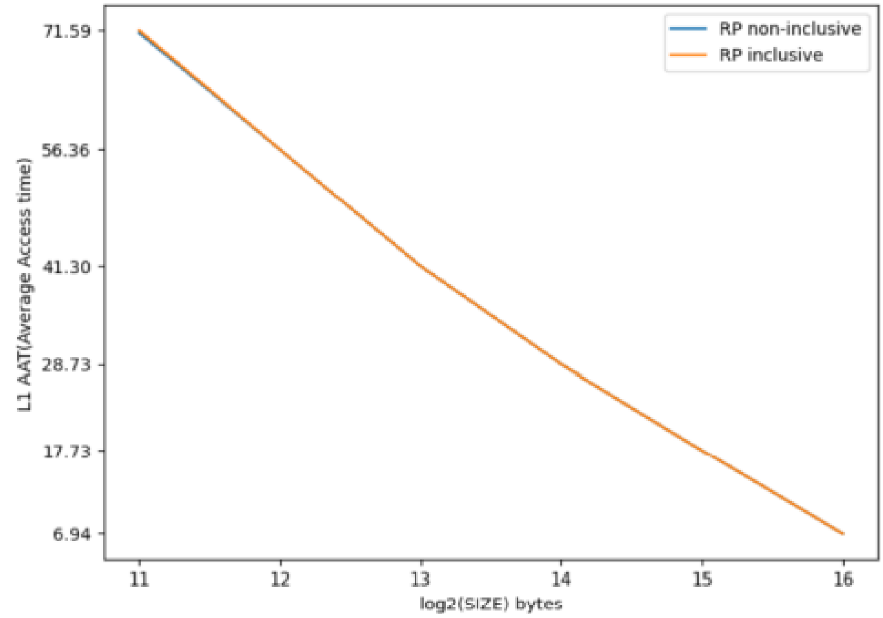


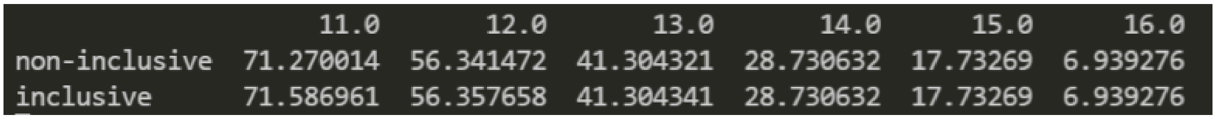
```
          10.0       11.0       12.0       13.0       14.0       15.0      16.0      17.0     18.0
LRU  81.32682  70.289496  56.705685  42.772173  29.733173  18.920936  8.85925  3.702481  3.00828
OPT  61.51882  50.552496  39.467685  29.578173  20.465173  12.577936  6.25825  3.418481  2.99428
```

Discussion:

1. **Discuss trends in the graph. Which replacement policy yields the best AAT?**
   The AAT of Optimal replacement policy is comparatively less than the AAT of LRU. I cannot comment on the AAT of PLRU as I haven’t implemented that. Hence, best replacement policy is Optimal.

Graph 4 Inclusion Property Study





```
                   11.0       12.0       13.0       14.0      15.0      16.0
non-inclusive  71.270014  56.341472  41.304321  28.730632  17.73269  6.939276
inclusive      71.586961  56.357658  41.304341  28.730632  17.73269  6.939276
```

Discussion:

1. **Discuss trends in the graph. Which inclusion property yields a better AAT?**
   At least for the gcc_trace file, I did not see much difference in the AATs of non-inclusive and inclusive properties. But as observed, for initial cache sizes, the AAT for inclusive are slightly higher than that of non-inclusive. Hence, we can conclude that in general non-inclusive yields a better AAT.