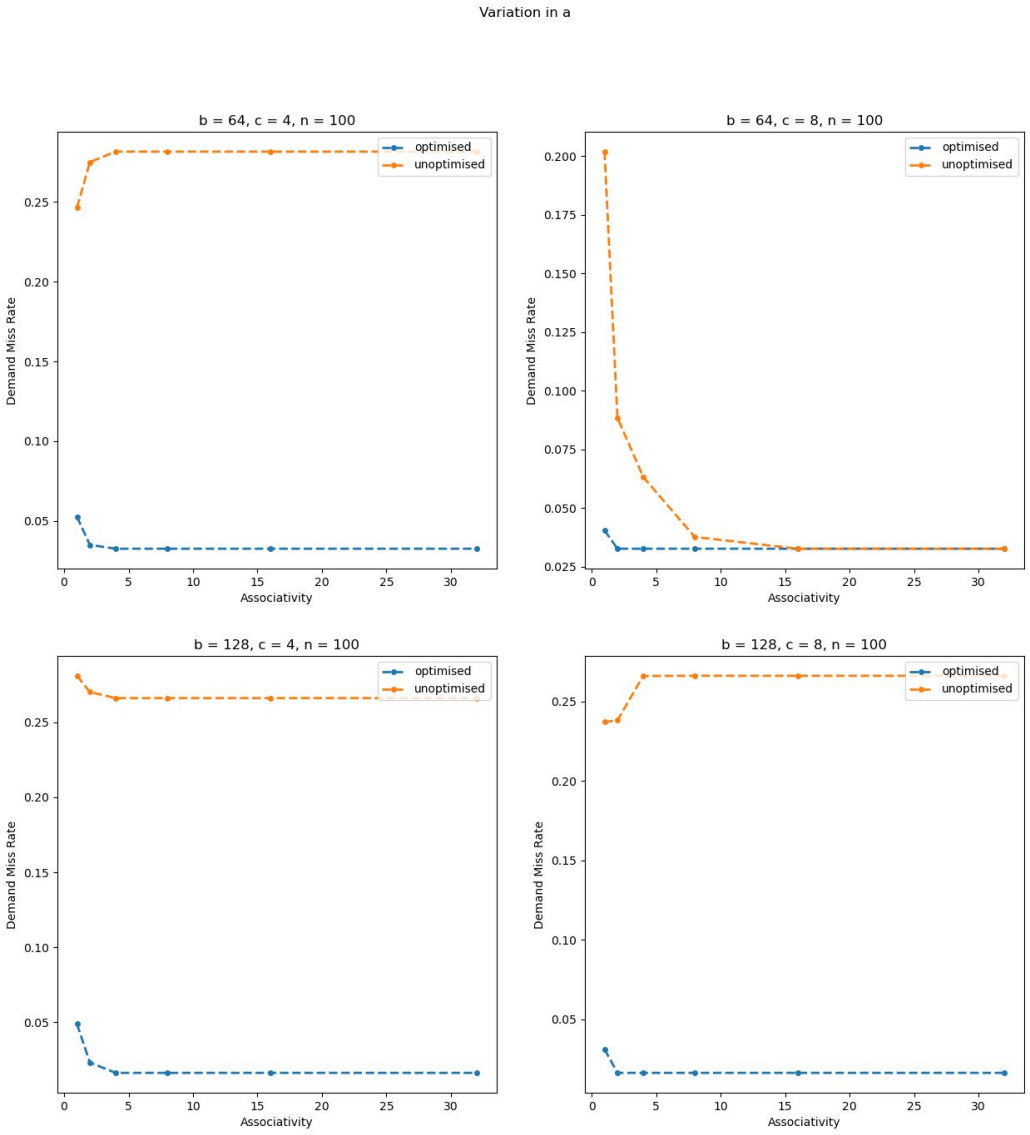
Computer Architecture Lab (CS 341)

Assignment 9: Cache Simulator

Lab Assignment 5

Bhaskar Gupta - 180050022
Niraj Mahajan - 180050069
Shivam Goel - 180050098
Tathagat Verma - 180050111

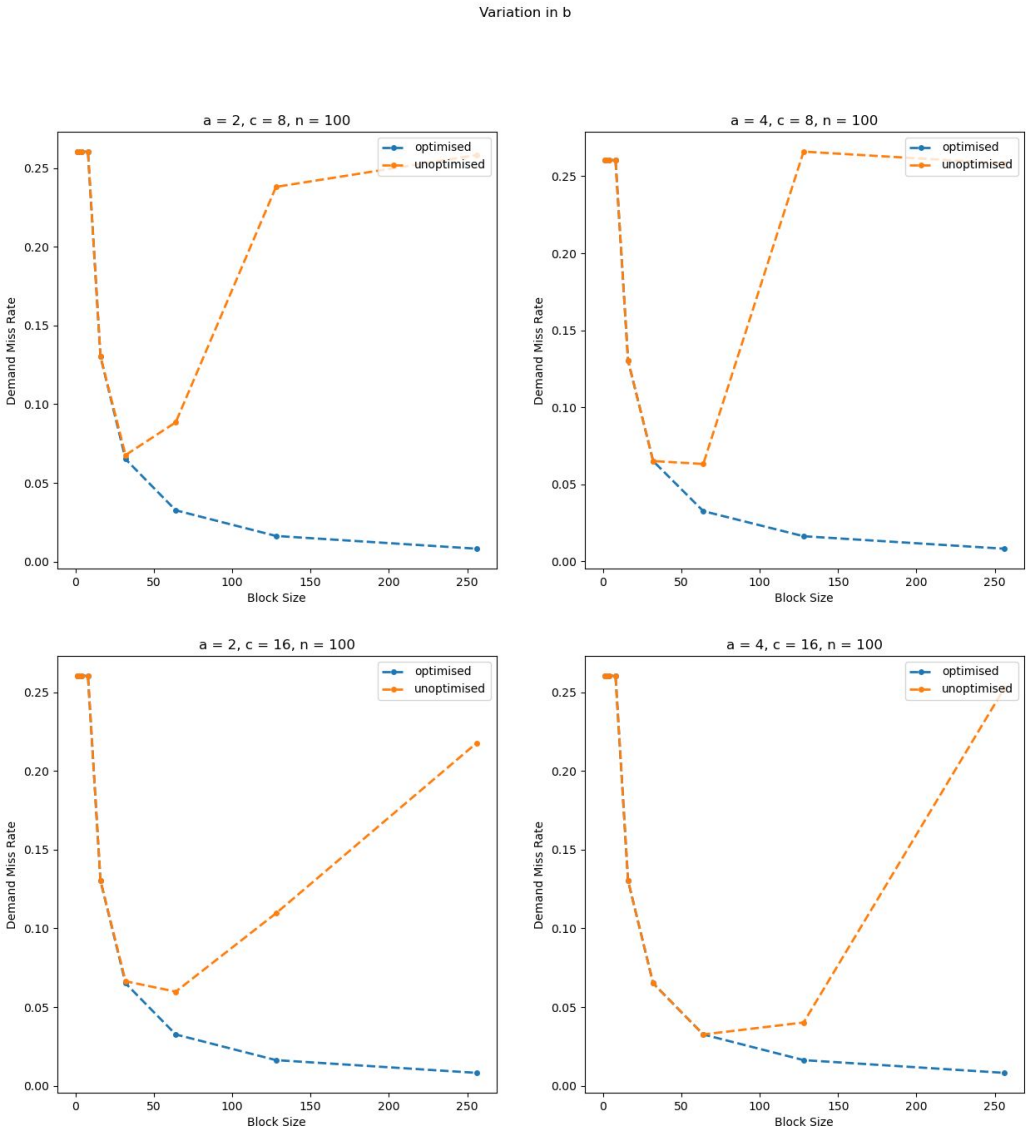
1. Variation in Associativity



Variation in Associativity, b = 64B, c = 4KB, n = 100							
Opt	a	Total Fetches	Total Misses	Miss Rate	Compulsory miss	Conflict miss	Capacity miss
No	1	4030000	993685	0.2465719603	3751	28649	961285
No	2	4030000	1108133	0.2749709677	3751	3956	1100426
No	4	4030000	1134950	0.2816253102	3751	0	1131199
No	8	4030000	1134950	0.2816253102	3751	0	1131199
No	16	4030000	1134950	0.2816253102	3751	0	1131199
No	32	4030000	1134950	0.2816253102	3751	0	1131199
Yes	1	4030000	212137	0.05263945409	3751	80785	127601
Yes	2	4030000	140869	0.03495508685	3751	9517	127601
Yes	4	4030000	131352	0.03259354839	3751	0	127601
Yes	8	4030000	131352	0.03259354839	3751	0	127601
Yes	16	4030000	131352	0.03259354839	3751	0	127601
Yes	32	4030000	131352	0.03259354839	3751	0	127601

Explanation:
(Note: The ‘Yes’ marked rows in column 1 refer to the optimised code, while the ‘No’ marked rows refer to the unoptimised ones.)
As the associativity increases, the miss rate will decrease, but this decrement is not significant for larger block sizes. This is significant in the plots as the miss rate quickly plateaus down. Increasing the associativity implies increasing the number of rows/lines in the cache to which lines in the memory can be mapped to. This additional flexibility in storing memory lines in the cache (and also while evicting them) leads to a decrement in the miss rate. The data in the table above is for block size = 64B, and cache size = 4KB. The tables for the other three cases (as shown in the plot also follow a similar trend, hence not included in the report to avoid unnecessary cluttering). As we can see, the conflict misses gradually decrease as the associativity increases. This is because an increase in associativity reduces the restrictions on the cache to store the memory blocks in specific cache lines, and when we increase the associativity beyond a certain limit, the cache resembles a FA cache, and the conflict misses are reduced to 0.

2. Variation in Block Size

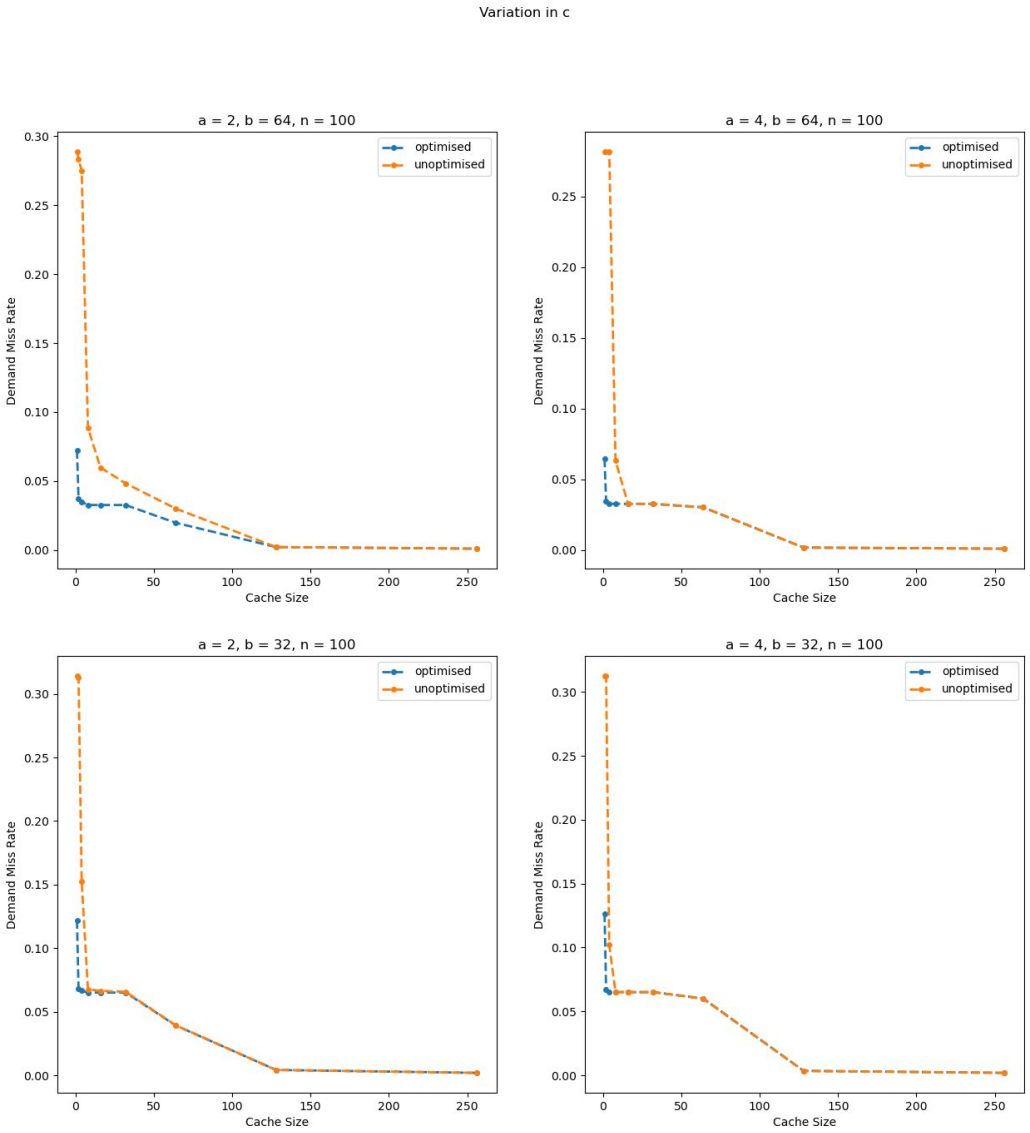


Variation in Block size, a = 2, c = 8, n = 100							
Opt	b	Total Fetches	Total Misses	Miss Rate	Compulsory miss	Conflict miss	Capacity miss
No	1	16120000	4199968	0.2605439206	120000	172	4079796
No	2	8060000	2099984	0.2605439206	60000	86	2039898
No	4	4030000	1049992	0.2605439206	30000	43	1019949
No	8	4030000	1049992	0.2605439206	30000	2	1019990

No	16	4030000	524996	0.1302719603	15000	1	509995
No	32	4030000	272982	0.06773746898	7500	10484	254998
No	64	4030000	356511	0.08846426799	3751	225110	127650
No	128	4030000	959366	0.2380560794	1876	0	957490
No	256	4030000	1041146	0.2583488834	938	0	1040208
Yes	1	16120000	4200000	0.2605459057	120000	0	4080000
Yes	2	8060000	2100000	0.2605459057	60000	0	2040000
Yes	4	4030000	1050000	0.2605459057	30000	0	1020000
Yes	8	4030000	1050000	0.2605459057	30000	0	1020000
Yes	16	4030000	525000	0.1302729529	15000	0	510000
Yes	32	4030000	262500	0.06513647643	7500	0	255000
Yes	64	4030000	131352	0.03259354839	3751	0	127601
Yes	128	4030000	65727	0.01630942928	1876	0	63851
Yes	256	4030000	32864	0.00815483871	938	0	31926

Explanation:
(Note: The ‘Yes’ marked rows in column 1 refer to the optimised code, while the ‘No’ marked rows refer to the unoptimised ones.)
Increasing the block size to an extent decreases the miss rate as this exploits the spatial locality of the data.
Increasing the block size beyond a certain limit leads to unnecessary data being loaded in the cache, or cache pollution as mentioned in the slides. This is evident by the sudden and steep rise in the miss rates of the unoptimised code for matrix multiplication, where the memory(or cache) accesses are in an irregular order. But in the optimised case, since we are accessing the cache in a row-wise manner, there is no cache pollution, and hence the miss rate decreases. But, of course, we must prefer not to increase the block size to this extent as one cannot assume the program to always be able to be rewritten in a way that allows sequential access to the cache. There is also an increase in the number of conflict misses (obviously in case of the unoptimised code, since the optimised has row-wise access), as having larger block sizes implies more blocks in memory being mapped to the same lin in the cache. The tables for the other three cases (as shown in the plot also follow a similar trend, hence not included in the report to avoid unnecessary cluttering).

3. Variation in Cache Size

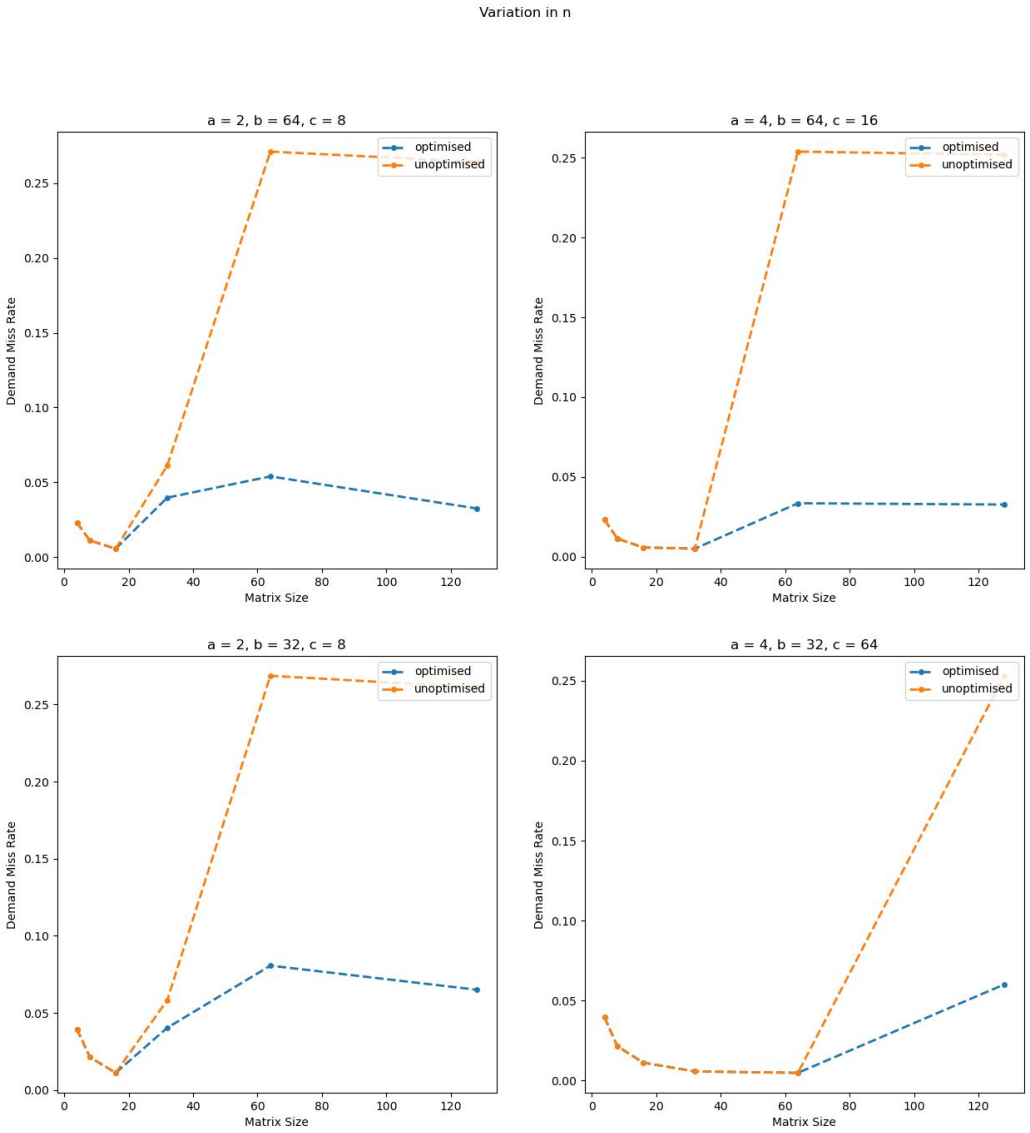


Variation in Cache size, a = 2, b = 32, n = 100							
Opt	c	Total Fetches	Total Misses	Miss Rate	Compulsory miss	Conflict miss	Capacity miss
No	1	4030000	1266804	0.3143434243	7500	6804	1252500
No	2	4030000	1261820	0.3131066998	7500	1820	1252500
No	4	4030000	613724	0.1522888337	7500	351224	255000

No	8	4030000	272982	0.06773746898	7500	10484	254998
No	16	4030000	267708	0.06642878412	7500	5240	254968
No	32	4030000	264910	0.06573449132	7500	2623	254787
No	64	4030000	158328	0.03928734491	7500	1279	149549
No	128	4030000	16795	0.004167493797	7500	2899	6396
No	256	4030000	7500	0.001861042184	7500	0	0
Yes	1	4030000	491759	0.1220245658	7500	9720	474539
Yes	2	4030000	273265	0.06780769231	7500	10765	255000
Yes	4	4030000	267842	0.06646203474	7500	5342	255000
Yes	8	4030000	262500	0.06513647643	7500	0	255000
Yes	16	4030000	262500	0.06513647643	7500	0	255000
Yes	32	4030000	262500	0.06513647643	7500	0	255000
Yes	64	4030000	158126	0.03923722084	7500	0	150626
Yes	128	4030000	16843	0.004179404467	7500	2722	6621
Yes	256	4030000	7500	0.001861042184	7500	0	0

Explanation:
Increasing the cache size keeping a,b constant will simply increase the number of lines in the cache. Hence the miss rate has to be non increasing for an increase in the cache size. Furthermore, after a certain limit, the entire memory gets mapped to the cache, and hence all the misses that occur are compulsory/cold misses. The tables for the other three cases (as shown in the plot also follow a similar trend, hence not included in the report to avoid unnecessary cluttering).
(Note: The ‘Yes’ marked rows in column 1 refer to the optimised code, while the ‘No’ marked rows refer to the unoptimised ones.)

4. Variation in Matrix Size



Variation in Matrix size, a = 2, b = 32B, c = 8KB							
Opt	m	Total Fetches	Total Misses	Miss Rate	Compulsory miss	Conflict miss	Capacity miss
No	4	304	12	0.03947368421	12	0	0
No	8	2240	48	0.02142857143	48	0	0
No	16	17152	192	0.01119402985	192	0	0

No	32	134144	7822	0.05831047233	768	5414	1640
No	64	1060864	285000	0.2686489503	3072	214350	67578
No	128	8437760	2203392	0.2611347087	12288	1658625	532479
Yes	4	304	12	0.03947368421	12	0	0
Yes	8	2240	48	0.02142857143	48	0	0
Yes	16	17152	192	0.01119402985	192	0	0
Yes	32	134144	5421	0.04041179628	768	3020	1633
Yes	64	1060864	85616	0.08070402992	3072	14960	67584
Yes	128	4030000	262500	0.06513647643	7500	0	255000

Explanation:
(Note: The ‘Yes’ marked rows in column 1 refer to the optimised code, while the ‘No’ marked rows refer to the unoptimised ones.)
It makes sense for the miss rate to increase as the matrix dimension increases. This is because the number of operations to be performed in an $A_{mn}*B_{np}$ operation are O(mnp), which is $O(n^3)$ in our case. But again there is a visible difference in the optimised and the unoptimised case because of the cache access order as mentioned in the previous section. The tables for the other three cases (as shown in the plot also follow a similar trend, hence not included in the report to avoid unnecessary cluttering).

Bhaskar Gupta(180050022)
Niraj Mahajan (180050069)
Shivam Goel(180050098)
Tathagat Verma(180050111)

Code

```
from matplotlib import pyplot as plt
import subprocess


def form_command(a,b,c,n, optimised = True):
  if optimised:
    return "../d4-7/dineroIV -l1-usize {}K -l1-ubsize {} -l1-uassoc {} -l1-uwalloc a -l1-uwback a -l1-uccc  -informat d < ../trace_files/optimised/nopt{}.din".format(c,b,a,n)
  else:
    return "../d4-7/dineroIV -l1-usize {}K -l1-ubsize {} -l1-uassoc {} -l1-uwalloc a -l1-uwback a -l1-uccc  -informat d < ../trace_files/unoptimised/n{}.din".format(c,b,a,n)



# variation in associativity
print('
    ================================================================================
    ')
print('Variation in Associativity')
print('
    ================================================================================
    ')
sampler = [1, 2, 4, 8, 16, 32]
fig, ax = plt.subplots(2,2, figsize = (15,15))
fig.suptitle('Variation in a')
for i, (b,c,n) in enumerate([(64,4,100),(64,8,100),(128,4,100)
    ,(128,8,100)]):
  ans_opt = []
  ans_unopt = []
  f = open('../csvs/variationa/{}_{}_{}.csv'.format(b,c,n), 'w')
  f.write('Opt,a,b,c,n,Total Fetches,Total Misses,Miss Rate,Compulsory miss, Conflict miss, Capacity miss\n')
  f.flush()
  for a in sampler:
    print("a = ",a,", b = ",b,", c = ",c,", n = ",n)
    bashCommand = form_command(a,b,c,n, optimised = True)
    output = subprocess.check_output(['bash','-c', bashCommand]).decode('utf8')
    ans_opt.append(float(output.split("\n")[35].split()[3]))
    total_fetches = output.split("\n")[31].split()[2]
    total_misses = output.split("\n")[34].split()[2]
    miss_rate = str(float(total_misses)/float(total_fetches))
    miss_compolsory = output.split("\n")[36].split()[2]
    miss_capacity = output.split("\n")[37].split()[2]
    miss_conflict = output.split("\n")[38].split()[2]
    f.write('{},{},{},{},{},{},{},{},{},{},{}\n'.format('Yes',a,b,c,n,total_fetches,total_misses,miss_rate,miss_compolsory,miss_conflict,miss_capacity))

```

```
    bashCommand = form_command(a,b,c,n, optimised = False)
    output = subprocess.check_output(['bash','-c', bashCommand]).decode('utf8')
    ans_unopt.append(float(output.split("\n")[35].split()[3]))
    total_fetches = output.split("\n")[31].split()[2]
    total_misses = output.split("\n")[34].split()[2]
    miss_rate = str(float(total_misses)/float(total_fetches))
    miss_compolsory = output.split("\n")[36].split()[2]
    miss_capacity = output.split("\n")[37].split()[2]
    miss_conflict = output.split("\n")[38].split()[2]
    f.write('{},{},{},{},{},{},{},{},{},{},{}\n'.format('No',a,b,c,n,total_fetches,total_misses,miss_rate,miss_compolsory,miss_conflict,miss_capacity))
  f.flush()
  f.close()
  ax[i//2,i%2].plot(sampler, ans_opt, linestyle='dashed', linewidth = 2, marker='o', markersize=4,label='optimised')
  ax[i//2,i%2].plot(sampler, ans_unopt, linestyle='dashed', linewidth = 2, marker='o', markersize=4,label='unoptimised')
  ax[i//2,i%2].set_xlabel('Associativity')
  ax[i//2,i%2].set_ylabel('Demand Miss Rate')
  ax[i//2,i%2].set_title('b = {}, c = {}, n = {}'.format(b,c,n))
  ax[i//2,i%2].legend(loc = 'upper right')
plt.savefig('../plots/varya.png')
print('\n\n\n')

# variation in block size
print('
    ================================================================================
    ')
print('Variation in Block Size')
print('
    ================================================================================
    ')
sampler = [1, 2, 4, 8, 16, 32, 64, 128, 256]
fig, ax = plt.subplots(2,2,figsize = (15,15))
fig.suptitle('Variation in b')
for i, (a,c,n) in enumerate([(2,8,100),(4,8,100),(2,16,100),(4,16,100)]):
  ans_opt = []
  ans_unopt = []
  f = open('../csvs/variationb/{}_{}_{}.csv'.format(a,c,n), 'w')
  f.write('Opt,a,b,c,n,Total Fetches,Total Misses,Miss Rate,Compulsory miss, Conflict miss, Capacity miss\n')
  f.flush()
  for b in sampler:
    print("a = ",a,", b = ",b,", c = ",c,", n = ",n)
    bashCommand = form_command(a,b,c,n, optimised = True)
    output = subprocess.check_output(['bash','-c', bashCommand]).decode('utf8')
    ans_opt.append(float(output.split("\n")[35].split()[3]))
    total_fetches = output.split("\n")[31].split()[2]
    total_misses = output.split("\n")[34].split()[2]
    miss_rate = str(float(total_misses)/float(total_fetches))
    miss_compolsory = output.split("\n")[36].split()[2]
    miss_capacity = output.split("\n")[37].split()[2]
```

```
83      miss_conflict = output.split("\n")[38].split()[2]
84      f.write('{},{},{},{},{},{},{},{},{},{},{}\n'.format('Yes',a,b,c,n,
        total_fetches,total_misses,miss_rate,miss_compolsory,miss_conflict,
        miss_capacity))
85
86      bashCommand = form_command(a,b,c,n, optimised = False)
87      output = subprocess.check_output(['bash','-c', bashCommand]).decode
        ('utf8')
88      ans_unopt.append(float(output.split("\n")[35].split()[3]))
89      total_fetches = output.split("\n")[31].split()[2]
90      total_misses = output.split("\n")[34].split()[2]
91      miss_rate = str(float(total_misses)/float(total_fetches))
92      miss_compolsory = output.split("\n")[36].split()[2]
93      miss_capacity = output.split("\n")[37].split()[2]
94      miss_conflict = output.split("\n")[38].split()[2]
95      f.write('{},{},{},{},{},{},{},{},{},{},{}\n'.format('No',a,b,c,n,
        total_fetches,total_misses,miss_rate,miss_compolsory,miss_conflict,
        miss_capacity))
96    f.flush()
97    f.close()
98    ax[i//2,i%2].plot(sampler, ans_opt, linestyle='dashed', linewidth =
       2, marker='o', markersize=4,label='optimised')
99    ax[i//2,i%2].plot(sampler, ans_unopt, linestyle='dashed', linewidth =
        2, marker='o', markersize=4,label='unoptimised')
100   ax[i//2,i%2].set_xlabel('Block Size')
101   ax[i//2,i%2].set_ylabel('Demand Miss Rate')
102   ax[i//2,i%2].set_title('a = {}, c = {}, n = {}'.format(a,c,n))
103   ax[i//2,i%2].legend(loc = 'upper right')
104 plt.savefig('../plots/varyb.png')
105 print('\n\n\n')
106
107 # variation in cache size
108 print('
      ==========================================================================
      ')
109 print('Variation in Cache Size')
110 print('
      ==========================================================================
      ')
111 sampler = [1, 2, 4, 8, 16, 32, 64, 128, 256]
112 fig, ax = plt.subplots(2,2,figsize = (15,15))
113 fig.suptitle('Variation in c')
114 for i, (a,b,n) in enumerate([(2,64,100),(4,64,100),(2,32,100)
      ,(4,32,100)]):
115   ans_opt = []
116   ans_unopt = []
117   f = open('../csvs/variationc/{}_{}_{}.csv'.format(a,b,n), 'w')
118   f.write('Opt,a,b,c,n,Total Fetches,Total Misses,Miss Rate,Compulsory
       miss, Conflict miss, Capacity miss\n')
119   f.flush()
120   for c in sampler:
121     print("a = ",a,", b = ",b,", c = ",c,", n = ",n)
122     bashCommand = form_command(a,b,c,n, optimised = True)
123     output = subprocess.check_output(['bash','-c', bashCommand]).decode
      ('utf8')
124     ans_opt.append(float(output.split("\n")[35].split()[3]))
```

```
    total_fetches = output.split("\n")[31].split()[2]
    total_misses = output.split("\n")[34].split()[2]
    miss_rate = str(float(total_misses)/float(total_fetches))
    miss_compolsory = output.split("\n")[36].split()[2]
    miss_capacity = output.split("\n")[37].split()[2]
    miss_conflict = output.split("\n")[38].split()[2]
    f.write('{},{},{},{},{},{},{},{},{},{},{}\n'.format('Yes',a,b,c,n,
   total_fetches,total_misses,miss_rate,miss_compolsory,miss_conflict,
   miss_capacity))

    bashCommand = form_command(a,b,c,n, optimised = False)
    output = subprocess.check_output(['bash','-c', bashCommand]).decode
   ('utf8')
    ans_unopt.append(float(output.split("\n")[35].split()[3]))
    total_fetches = output.split("\n")[31].split()[2]
    total_misses = output.split("\n")[34].split()[2]
    miss_rate = str(float(total_misses)/float(total_fetches))
    miss_compolsory = output.split("\n")[36].split()[2]
    miss_capacity = output.split("\n")[37].split()[2]
    miss_conflict = output.split("\n")[38].split()[2]
    f.write('{},{},{},{},{},{},{},{},{},{},{}\n'.format('No',a,b,c,n,
   total_fetches,total_misses,miss_rate,miss_compolsory,miss_conflict,
   miss_capacity))
  f.flush()
  f.close()
  ax[i//2,i%2].plot(sampler, ans_opt, linestyle='dashed', linewidth =
   2, marker='o', markersize=4,label='optimised')
  ax[i//2,i%2].plot(sampler, ans_unopt, linestyle='dashed', linewidth =
    2, marker='o', markersize=4,label='unoptimised')
  ax[i//2,i%2].set_xlabel('Cache Size')
  ax[i//2,i%2].set_ylabel('Demand Miss Rate')
  ax[i//2,i%2].set_title('a = {}, b = {}, n = {}'.format(a,b,n))
  ax[i//2,i%2].legend(loc = 'upper right')
plt.savefig('../plots/varyc.png')
print('\n\n\n')

# variation in matrix size
print('
   ==========================================================================
   ')
print('Variation in Matrix Size')
print('
   ==========================================================================
   ')
sampler = [4, 8, 16, 32, 64, 128]
fig, ax = plt.subplots(2,2,figsize = (15,15))
fig.suptitle('Variation in n')
for i, (a,b,c) in enumerate([(2,64,8),(4,64,16),(2,32,8),(4,32,64)]):
  ans_opt = []
  ans_unopt = []
  f = open('../csvs/variationn/{}_{}_{}.csv'.format(a,b,c), 'w')
  f.write('Opt,a,b,c,n,Total Fetches,Total Misses,Miss Rate,Compulsory
   miss, Conflict miss, Capacity miss\n')
  f.flush()
  for n in sampler:
    print("a = ",a,", b = ",b,", c = ",c,", n = ",n)
```

```
169     bashCommand = form_command(a,b,c,n, optimised = True)
170     output = subprocess.check_output(['bash','-c', bashCommand]).decode
    ('utf8')
171     ans_opt.append(float(output.split("\n")[35].split()[3]))
172     total_fetches = output.split("\n")[31].split()[2]
173     total_misses = output.split("\n")[34].split()[2]
174     miss_rate = str(float(total_misses)/float(total_fetches))
175     miss_compolsory = output.split("\n")[36].split()[2]
176     miss_capacity = output.split("\n")[37].split()[2]
177     miss_conflict = output.split("\n")[38].split()[2]
178     f.write('{},{},{},{},{},{},{},{},{},{},{}\n'.format('Yes',a,b,c,n,
    total_fetches,total_misses,miss_rate,miss_compolsory,miss_conflict,
    miss_capacity))
179 
180     bashCommand = form_command(a,b,c,n, optimised = False)
181     output = subprocess.check_output(['bash','-c', bashCommand]).decode
    ('utf8')
182     ans_unopt.append(float(output.split("\n")[35].split()[3]))
183     total_fetches = output.split("\n")[31].split()[2]
184     total_misses = output.split("\n")[34].split()[2]
185     miss_rate = str(float(total_misses)/float(total_fetches))
186     miss_compolsory = output.split("\n")[36].split()[2]
187     miss_capacity = output.split("\n")[37].split()[2]
188     miss_conflict = output.split("\n")[38].split()[2]
189     f.write('{},{},{},{},{},{},{},{},{},{},{}\n'.format('No',a,b,c,n,
    total_fetches,total_misses,miss_rate,miss_compolsory,miss_conflict,
    miss_capacity))
190   f.flush()
191   f.close()
192   ax[i//2,i%2].plot(sampler, ans_opt, linestyle='dashed', linewidth =
     2, marker='o', markersize=4,label='optimised')
193   ax[i//2,i%2].plot(sampler, ans_unopt, linestyle='dashed', linewidth =
      2, marker='o', markersize=4,label='unoptimised')
194   ax[i//2,i%2].set_xlabel('Matrix Size')
195   ax[i//2,i%2].set_ylabel('Demand Miss Rate')
196   ax[i//2,i%2].set_title('a = {}, b = {}, c = {}'.format(a,b,c))
197   ax[i//2,i%2].legend(loc = 'upper right')
198 plt.savefig('../plots/varyn.png')
199 print('\n\n\n')
```