CPEG655

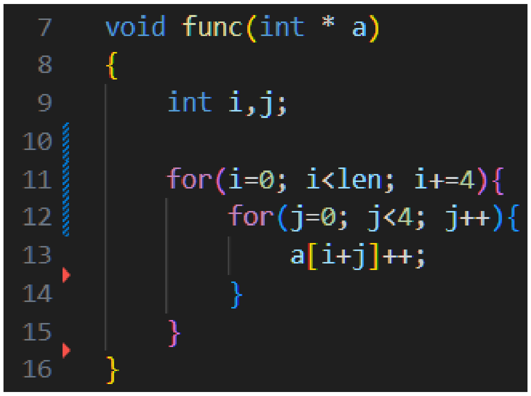
Lab 01

Michael Hutts

Problem 1

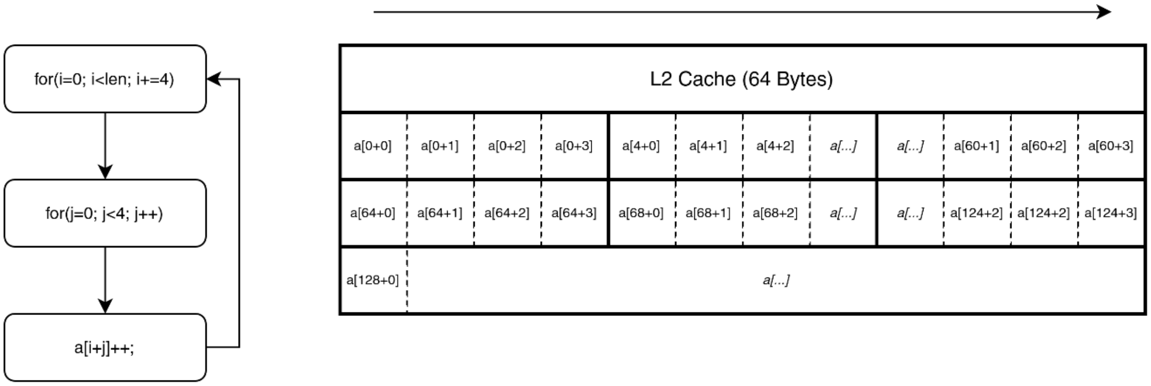
De-Optimizing mem1.c

Original Code:



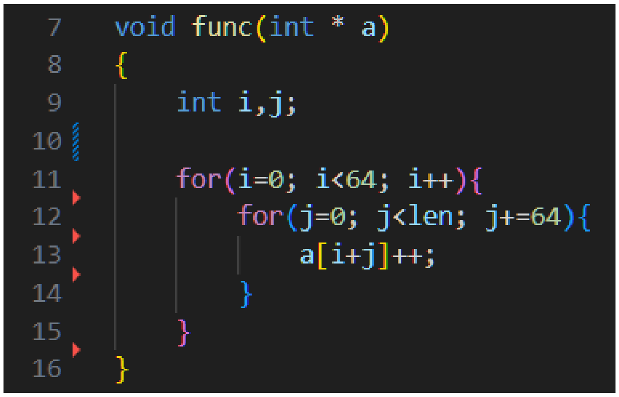
```
void func(int * a)
{
    int i,j;

    for(i=0; i<len; i+=4){
        for(j=0; j<4; j++){
            a[i+j]++;
        }
    }
}
```

The original version of mem1.c is a simple for loop that accesses the memory sequentially in chunks of 4.



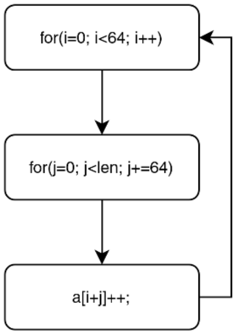
a[0+0]	a[0+1]	a[0+2]	a[0+3]	a[4+0]	a[4+1]	a[4+2]	a[...]	a[...]	a[60+1]	a[60+2]	a[60+3]				
a[64+0]	a[64+1]	a[64+2]	a[64+3]	a[68+0]	a[68+1]	a[68+2]	a[...]	a[...]	a[124+2]	a[124+2]	a[124+3]				
a[128+0]	a[...]														

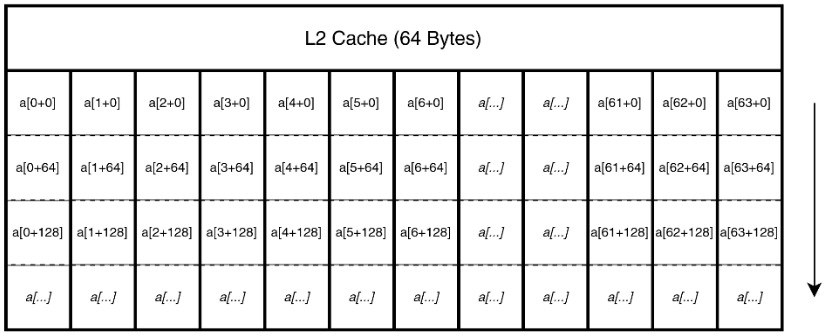
De-Optimized Code:



```c
void func(int * a)
{
    int i,j;

    for(i=0; i<64; i++){
        for(j=0; j<len; j+=64){
            a[i+j]++;
        }
    }
}
```

The de-optimized version is set up to access a new cache-line every iteration. No additional memory accesses are added (as demonstrated in a later comparison), but the L2 cache misses and TLB misses have increased dramatically.

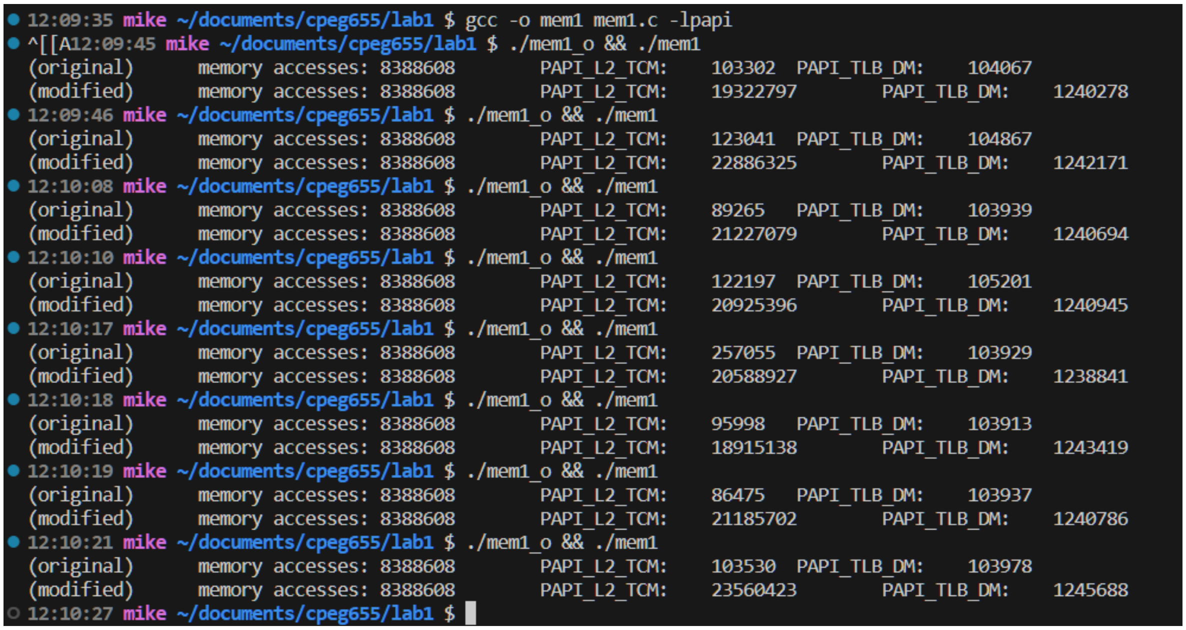




L2 Cache (64 Bytes)

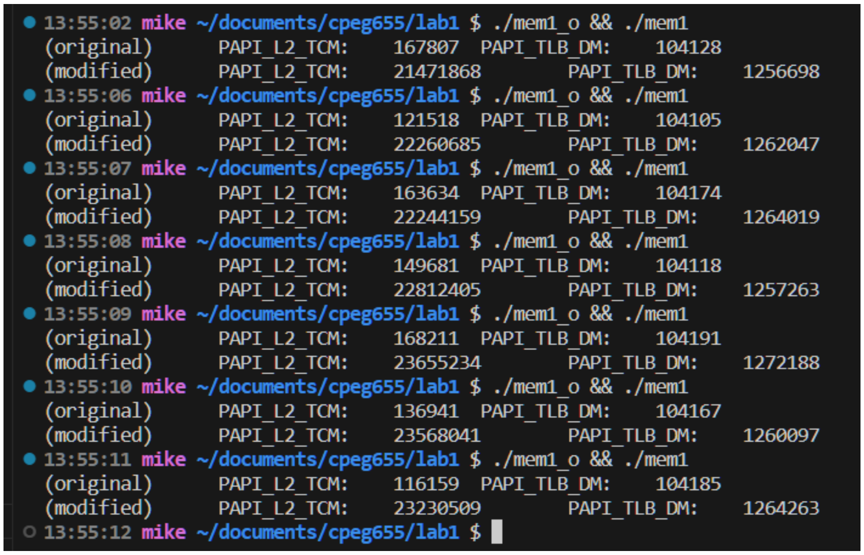
a[0+0]	a[1+0]	a[2+0]	a[3+0]	a[4+0]	a[5+0]	a[6+0]	*a[...]*	*a[...]*	a[61+0]	a[62+0]	a[63+0]
a[0+64]	a[1+64]	a[2+64]	a[3+64]	a[4+64]	a[5+64]	a[6+64]	*a[...]*	*a[...]*	a[61+64]	a[62+64]	a[63+64]
a[0+128]	a[1+128]	a[2+128]	a[3+128]	a[4+128]	a[5+128]	a[6+128]	*a[...]*	*a[...]*	a[61+128]	a[62+128]	a[63+128]
a[...]	*a[...]*	*a[...]*	*a[...]*	*a[...]*	*a[...]*	*a[...]*	*a[...]*	*a[...]*	*a[...]*	*a[...]*	*a[...]*

Performance Comparison:



```
● 12:09:35 mike ~/documents/cpeg655/lab1 $ gcc -o mem1 mem1.c -lpapi
● ^[[A12:09:45 mike ~/documents/cpeg655/lab1 $ ./mem1_o && ./mem1
  (original)      memory accesses: 8388608        PAPI_L2_TCM:    103302  PAPI_TLB_DM:    104067
  (modified)      memory accesses: 8388608        PAPI_L2_TCM:    19322797        PAPI_TLB_DM:    1240278
● 12:09:46 mike ~/documents/cpeg655/lab1 $ ./mem1_o && ./mem1
  (original)      memory accesses: 8388608        PAPI_L2_TCM:    123041  PAPI_TLB_DM:    104867
  (modified)      memory accesses: 8388608        PAPI_L2_TCM:    22886325        PAPI_TLB_DM:    1242171
● 12:10:08 mike ~/documents/cpeg655/lab1 $ ./mem1_o && ./mem1
  (original)      memory accesses: 8388608        PAPI_L2_TCM:    89265   PAPI_TLB_DM:    103939
  (modified)      memory accesses: 8388608        PAPI_L2_TCM:    21227079        PAPI_TLB_DM:    1240694
● 12:10:10 mike ~/documents/cpeg655/lab1 $ ./mem1_o && ./mem1
  (original)      memory accesses: 8388608        PAPI_L2_TCM:    122197  PAPI_TLB_DM:    105201
  (modified)      memory accesses: 8388608        PAPI_L2_TCM:    20925396        PAPI_TLB_DM:    1240945
● 12:10:17 mike ~/documents/cpeg655/lab1 $ ./mem1_o && ./mem1
  (original)      memory accesses: 8388608        PAPI_L2_TCM:    257055  PAPI_TLB_DM:    103929
  (modified)      memory accesses: 8388608        PAPI_L2_TCM:    20588927        PAPI_TLB_DM:    1238841
● 12:10:18 mike ~/documents/cpeg655/lab1 $ ./mem1_o && ./mem1
  (original)      memory accesses: 8388608        PAPI_L2_TCM:    95998   PAPI_TLB_DM:    103913
  (modified)      memory accesses: 8388608        PAPI_L2_TCM:    18915138        PAPI_TLB_DM:    1243419
● 12:10:19 mike ~/documents/cpeg655/lab1 $ ./mem1_o && ./mem1
  (original)      memory accesses: 8388608        PAPI_L2_TCM:    86475   PAPI_TLB_DM:    103937
  (modified)      memory accesses: 8388608        PAPI_L2_TCM:    21185702        PAPI_TLB_DM:    1240786
● 12:10:21 mike ~/documents/cpeg655/lab1 $ ./mem1_o && ./mem1
  (original)      memory accesses: 8388608        PAPI_L2_TCM:    103530  PAPI_TLB_DM:    103978
  (modified)      memory accesses: 8388608        PAPI_L2_TCM:    23560423        PAPI_TLB_DM:    1245688
○ 12:10:27 mike ~/documents/cpeg655/lab1 $ █
```

Even with the additional noise of background processes, the difference in L2 and TLB performance is dramatic enough for the de-optimization to be clear.

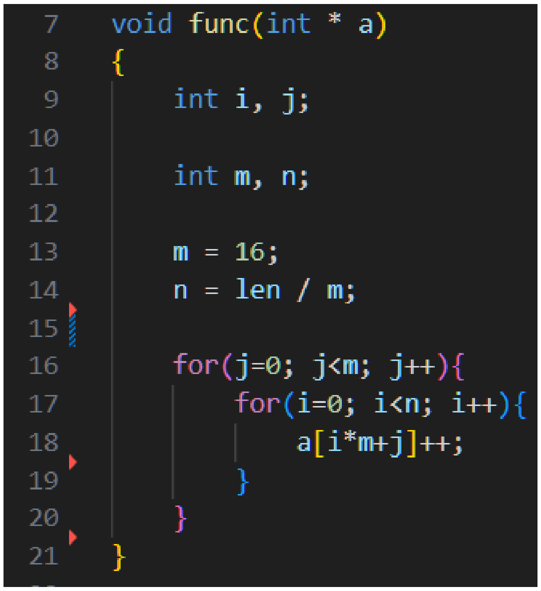


```
● 13:55:02 mike ~/documents/cpeg655/lab1 $ ./mem1_o && ./mem1
  (original)      PAPI_L2_TCM:    167807  PAPI_TLB_DM:    104128
  (modified)      PAPI_L2_TCM:    21471868        PAPI_TLB_DM:    1256698
● 13:55:06 mike ~/documents/cpeg655/lab1 $ ./mem1_o && ./mem1
  (original)      PAPI_L2_TCM:    121518  PAPI_TLB_DM:    104105
  (modified)      PAPI_L2_TCM:    22260685        PAPI_TLB_DM:    1262047
● 13:55:07 mike ~/documents/cpeg655/lab1 $ ./mem1_o && ./mem1
  (original)      PAPI_L2_TCM:    163634  PAPI_TLB_DM:    104174
  (modified)      PAPI_L2_TCM:    22244159        PAPI_TLB_DM:    1264019
● 13:55:08 mike ~/documents/cpeg655/lab1 $ ./mem1_o && ./mem1
  (original)      PAPI_L2_TCM:    149681  PAPI_TLB_DM:    104118
  (modified)      PAPI_L2_TCM:    22812405        PAPI_TLB_DM:    1257263
● 13:55:09 mike ~/documents/cpeg655/lab1 $ ./mem1_o && ./mem1
  (original)      PAPI_L2_TCM:    168211  PAPI_TLB_DM:    104191
  (modified)      PAPI_L2_TCM:    23655234        PAPI_TLB_DM:    1272188
● 13:55:10 mike ~/documents/cpeg655/lab1 $ ./mem1_o && ./mem1
  (original)      PAPI_L2_TCM:    136941  PAPI_TLB_DM:    104167
  (modified)      PAPI_L2_TCM:    23568041        PAPI_TLB_DM:    1260097
● 13:55:11 mike ~/documents/cpeg655/lab1 $ ./mem1_o && ./mem1
  (original)      PAPI_L2_TCM:    116159  PAPI_TLB_DM:    104185
  (modified)      PAPI_L2_TCM:    23230509        PAPI_TLB_DM:    1264263
○ 13:55:12 mike ~/documents/cpeg655/lab1 $ █
```

Note that memory accesses is a simple counter of every time a[] is accessed to confirm that no difference is a result of additional accesses. Its inclusion does not affect the change in performance as shown above.

De-Optimizing mem2.c

Original Code:



```
void func(int * a)
{
    int i, j;

    int m, n;

    m = 16;
    n = len / m;

    for(j=0; j<m; j++){
        for(i=0; i<n; i++){
            a[i*m+j]++;
        }
    }
}
```

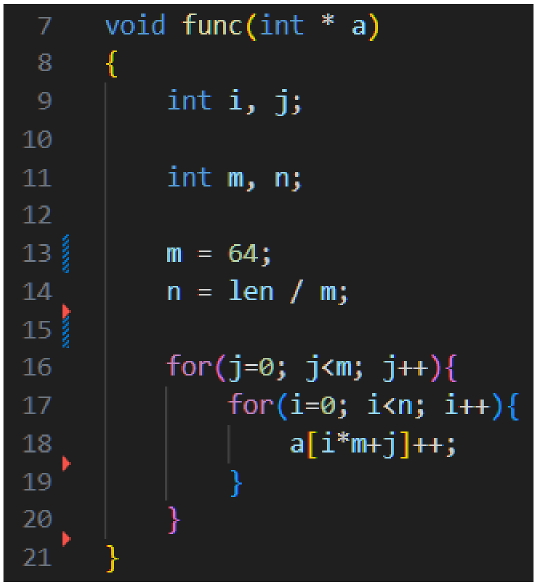
The original version of mem2.c skips through array in blocks of 16 (m). If we reuse the model from mem1.c, we can see that this method is already not ideal, but in a similar means to mem1.c, it can be de-optimized further to utilize the cache. The current access pattern goes as follows:

(n = 524288)

a[i*m+j]
a[0*16+0]
a[1*16+0]
a[…]
a[n*16+0]
a[0*16+1]
a[1*16+1]
a[…]
a[n*16+1]
…

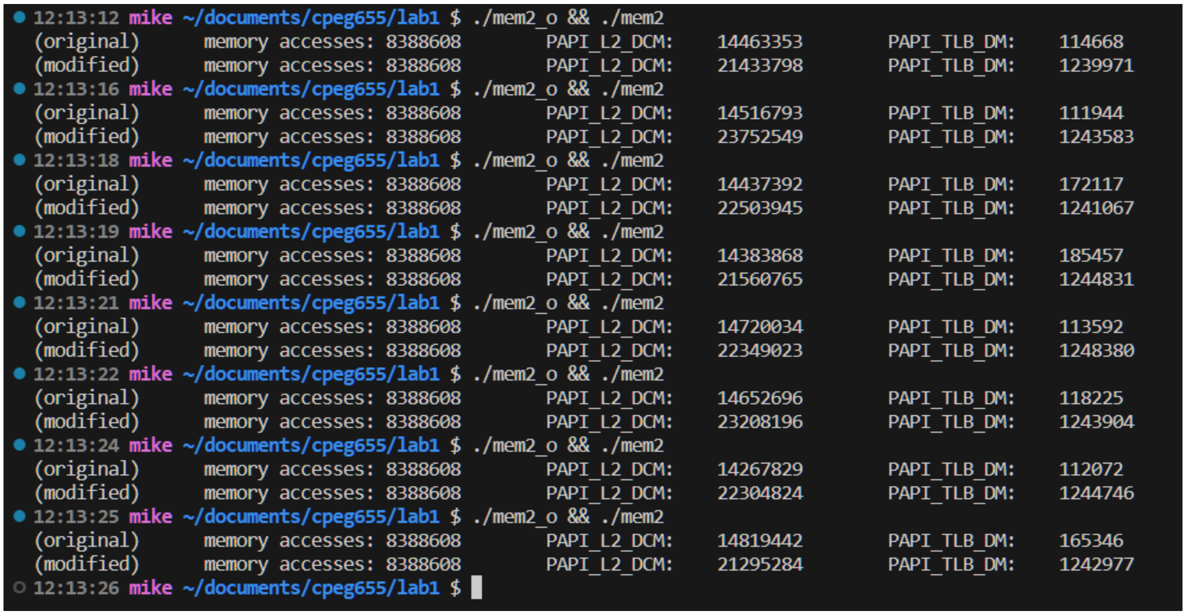
Using our previous understanding of how memory access can be de-optimized, we can make this worse by changing m, or in other words the size of the jumps we make so cache lines can never be used more than once before the next line is used and with the size of the overall cache the line will be overwritten without ever utilizing the cached line.

De-Optimized Code:



```
void func(int * a)
{
    int i, j;

    int m, n;

    m = 64;
    n = len / m;

    for(j=0; j<m; j++){
        for(i=0; i<n; i++){
            a[i*m+j]++;
        }
    }
}
```

Performance comparison:



```
● 12:13:12 mike ~/documents/cpeg655/lab1 $ ./mem2_o && ./mem2
  (original)     memory accesses: 8388608       PAPI_L2_DCM:    14463353        PAPI_TLB_DM:    114668
  (modified)     memory accesses: 8388608       PAPI_L2_DCM:    21433798        PAPI_TLB_DM:    1239971
● 12:13:16 mike ~/documents/cpeg655/lab1 $ ./mem2_o && ./mem2
  (original)     memory accesses: 8388608       PAPI_L2_DCM:    14516793        PAPI_TLB_DM:    111944
  (modified)     memory accesses: 8388608       PAPI_L2_DCM:    23752549        PAPI_TLB_DM:    1243583
● 12:13:18 mike ~/documents/cpeg655/lab1 $ ./mem2_o && ./mem2
  (original)     memory accesses: 8388608       PAPI_L2_DCM:    14437392        PAPI_TLB_DM:    172117
  (modified)     memory accesses: 8388608       PAPI_L2_DCM:    22503945        PAPI_TLB_DM:    1241067
● 12:13:19 mike ~/documents/cpeg655/lab1 $ ./mem2_o && ./mem2
  (original)     memory accesses: 8388608       PAPI_L2_DCM:    14383868        PAPI_TLB_DM:    185457
  (modified)     memory accesses: 8388608       PAPI_L2_DCM:    21560765        PAPI_TLB_DM:    1244831
● 12:13:21 mike ~/documents/cpeg655/lab1 $ ./mem2_o && ./mem2
  (original)     memory accesses: 8388608       PAPI_L2_DCM:    14720034        PAPI_TLB_DM:    113592
  (modified)     memory accesses: 8388608       PAPI_L2_DCM:    22349023        PAPI_TLB_DM:    1248380
● 12:13:22 mike ~/documents/cpeg655/lab1 $ ./mem2_o && ./mem2
  (original)     memory accesses: 8388608       PAPI_L2_DCM:    14652696        PAPI_TLB_DM:    118225
  (modified)     memory accesses: 8388608       PAPI_L2_DCM:    23208196        PAPI_TLB_DM:    1243904
● 12:13:24 mike ~/documents/cpeg655/lab1 $ ./mem2_o && ./mem2
  (original)     memory accesses: 8388608       PAPI_L2_DCM:    14267829        PAPI_TLB_DM:    112072
  (modified)     memory accesses: 8388608       PAPI_L2_DCM:    22304824        PAPI_TLB_DM:    1244746
● 12:13:25 mike ~/documents/cpeg655/lab1 $ ./mem2_o && ./mem2
  (original)     memory accesses: 8388608       PAPI_L2_DCM:    14819442        PAPI_TLB_DM:    165346
  (modified)     memory accesses: 8388608       PAPI_L2_DCM:    21295284        PAPI_TLB_DM:    1242977
○ 12:13:26 mike ~/documents/cpeg655/lab1 $ █
```

Similar to mem1.c, we see a notable jump in L2 cache misses and TLB misses. In this case, the results are a bit more consistent, with approximately 35% more L2 cache misses, and a dramatic 160~% increase in TLB misses.