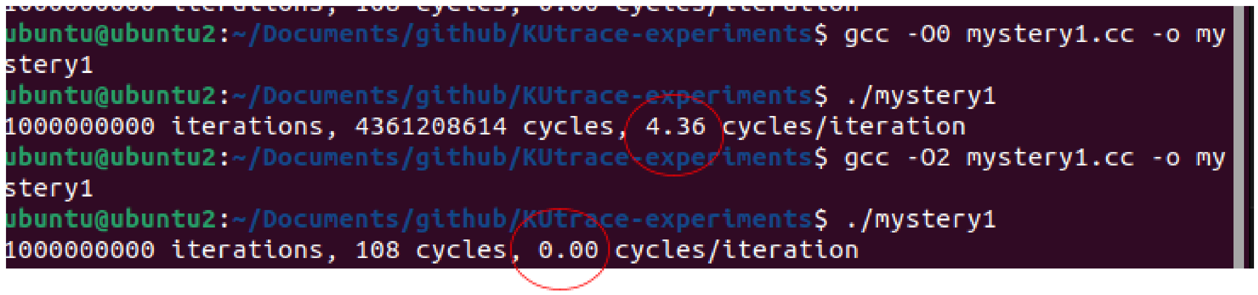
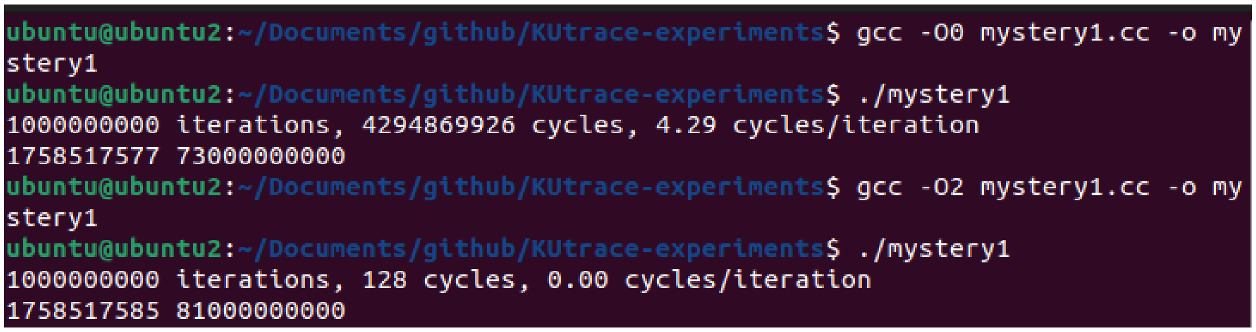
# Understanding Software Dynamics Ch 2 Lab Report

2.2) The optimized one's number of cycles is O(1 million) less than the unoptimized.
0.00 cycles per iteration means that the loop is removed. The gcc optimizer constant-folded all the billion increments of incr into just a constant result pre-calculated at compile time.



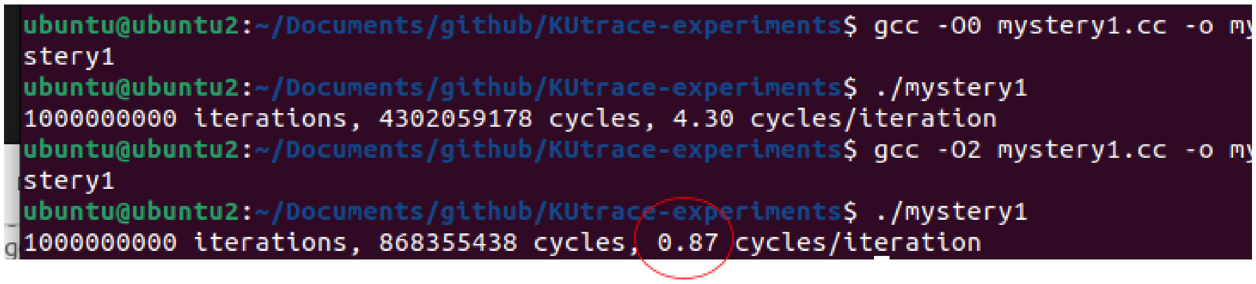
```
ubuntu@ubuntu2:~/Documents/github/KUtrace-experiments$ gcc -O0 mystery1.cc -o my
stery1
ubuntu@ubuntu2:~/Documents/github/KUtrace-experiments$ ./mystery1
1000000000 iterations, 4361208614 cycles, 4.36 cycles/iteration
ubuntu@ubuntu2:~/Documents/github/KUtrace-experiments$ gcc -O2 mystery1.cc -o my
stery1
ubuntu@ubuntu2:~/Documents/github/KUtrace-experiments$ ./mystery1
1000000000 iterations, 108 cycles, 0.00 cycles/iteration
```

2.3) The compiler does not remove the dead code variable sum now.



```
ubuntu@ubuntu2:~/Documents/github/KUtrace-experiments$ gcc -O0 mystery1.cc -o my
stery1
ubuntu@ubuntu2:~/Documents/github/KUtrace-experiments$ ./mystery1
1000000000 iterations, 4294869926 cycles, 4.29 cycles/iteration
1758517577 73000000000
ubuntu@ubuntu2:~/Documents/github/KUtrace-experiments$ gcc -O2 mystery1.cc -o my
stery1
ubuntu@ubuntu2:~/Documents/github/KUtrace-experiments$ ./mystery1
1000000000 iterations, 128 cycles, 0.00 cycles/iteration
1758517585 81000000000
```

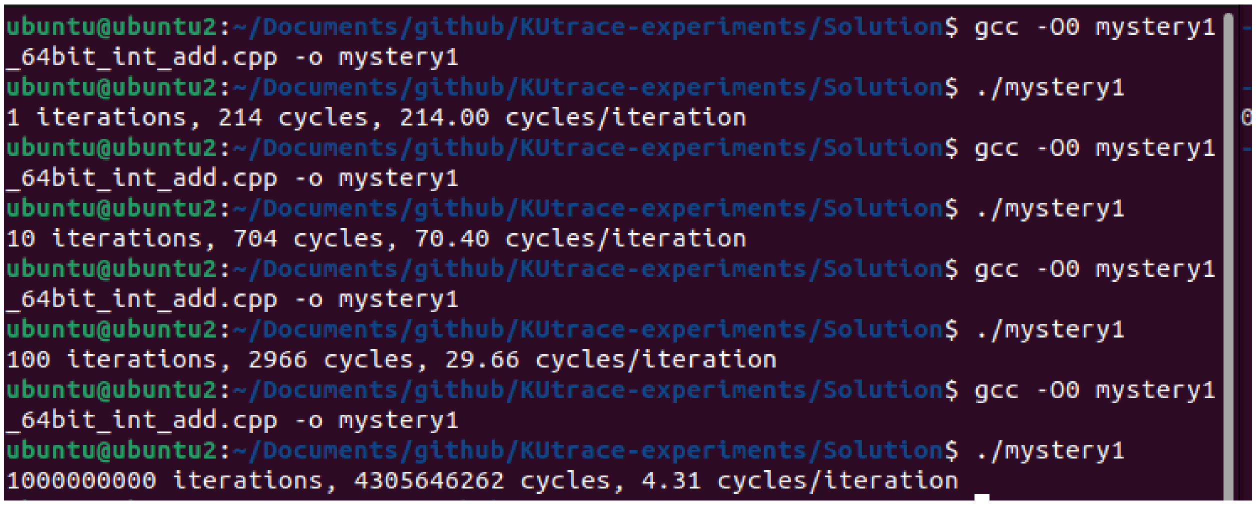
2.4) Volatile keyword is used to indicate that a variable may change in ways that the compiler cannot predict. So it prevents compiler optimization on loop unrolling. So the optimized magnitude of cycles and cycles/iteration is now close to 1/5 of the unoptimized.



```
ubuntu@ubuntu2:~/Documents/github/KUtrace-experiments$ gcc -O0 mystery1.cc -o my
stery1
ubuntu@ubuntu2:~/Documents/github/KUtrace-experiments$ ./mystery1
1000000000 iterations, 4302059178 cycles, 4.30 cycles/iteration
ubuntu@ubuntu2:~/Documents/github/KUtrace-experiments$ gcc -O2 mystery1.cc -o my
stery1
ubuntu@ubuntu2:~/Documents/github/KUtrace-experiments$ ./mystery1
1000000000 iterations, 868355438 cycles, 0.87 cycles/iteration
```

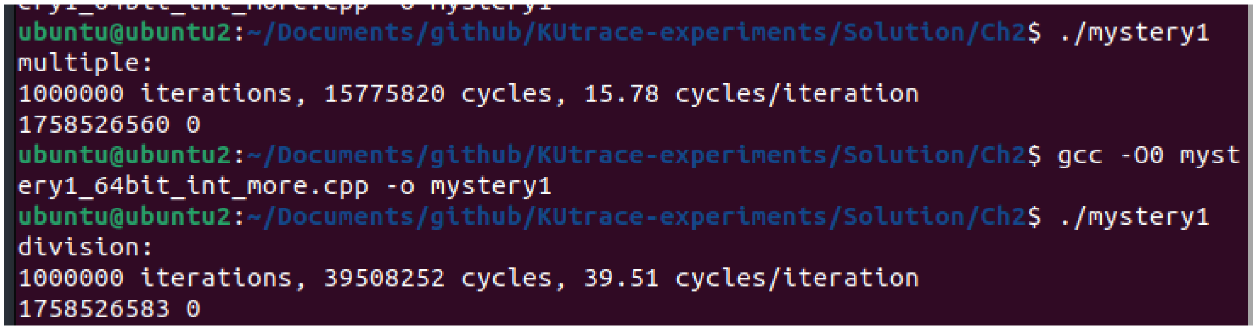
2.5) mystery1_64bit_int_add.cpp

2.6) The numbers in large iterations are not useful because the CPU may do caching (or instruction pipelining) which leads to measuring the issue time instead.

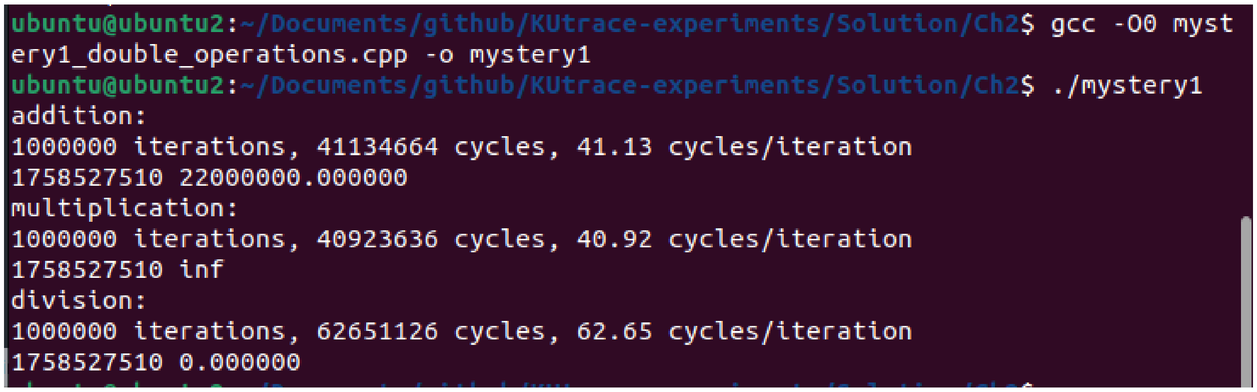


```
ubuntu@ubuntu2:~/Documents/github/KUtrace-experiments/Solution$ gcc -O0 mystery1
_64bit_int_add.cpp -o mystery1
ubuntu@ubuntu2:~/Documents/github/KUtrace-experiments/Solution$ ./mystery1
1 iterations, 214 cycles, 214.00 cycles/iteration
ubuntu@ubuntu2:~/Documents/github/KUtrace-experiments/Solution$ gcc -O0 mystery1
_64bit_int_add.cpp -o mystery1
ubuntu@ubuntu2:~/Documents/github/KUtrace-experiments/Solution$ ./mystery1
10 iterations, 704 cycles, 70.40 cycles/iteration
ubuntu@ubuntu2:~/Documents/github/KUtrace-experiments/Solution$ gcc -O0 mystery1
_64bit_int_add.cpp -o mystery1
ubuntu@ubuntu2:~/Documents/github/KUtrace-experiments/Solution$ ./mystery1
100 iterations, 2966 cycles, 29.66 cycles/iteration
ubuntu@ubuntu2:~/Documents/github/KUtrace-experiments/Solution$ gcc -O0 mystery1
_64bit_int_add.cpp -o mystery1
ubuntu@ubuntu2:~/Documents/github/KUtrace-experiments/Solution$ ./mystery1
1000000000 iterations, 4305646262 cycles, 4.31 cycles/iteration
```

2.8) mystery1_64bit_int_more.cpp, mystery1_double_operations.cpp
(For keeping data values away from extremes of overflow and underflow, kIterations should be around 10.)

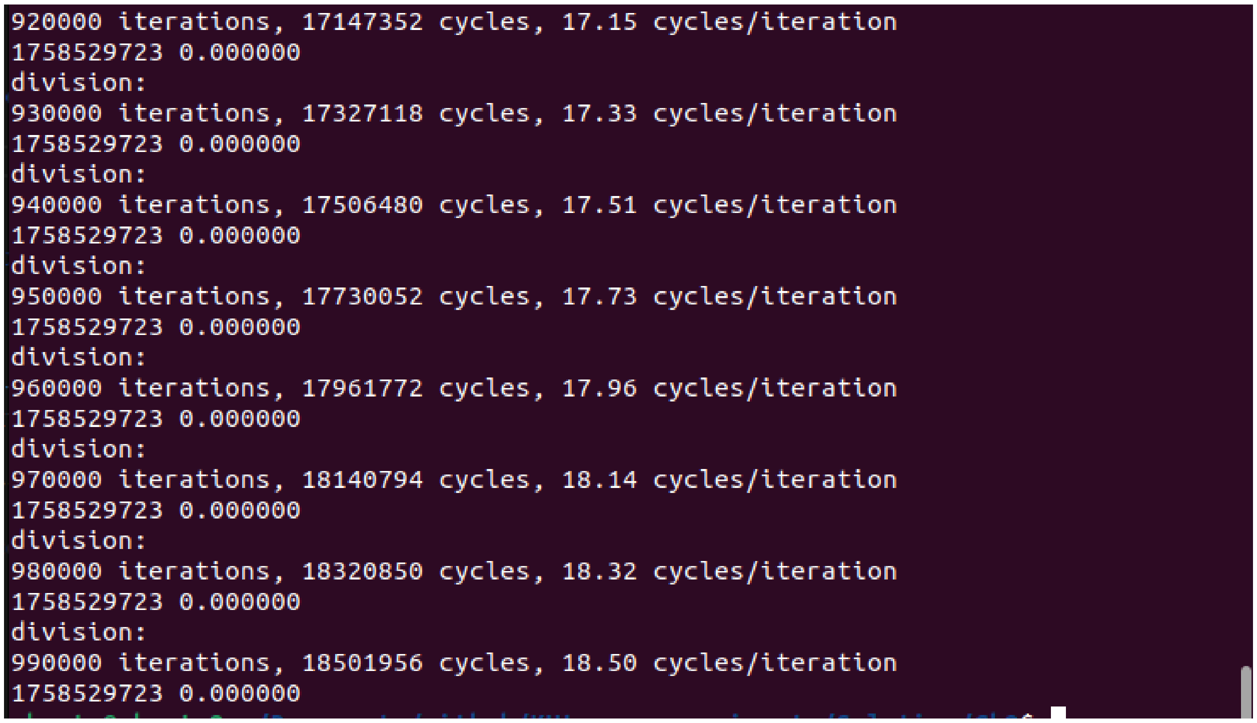


```
ubuntu@ubuntu2:~/Documents/github/KUtrace-experiments/Solution/Ch2$ ./mystery1
multiple:
1000000 iterations, 15775820 cycles, 15.78 cycles/iteration
1758526560 0
ubuntu@ubuntu2:~/Documents/github/KUtrace-experiments/Solution/Ch2$ gcc -O0 myst
ery1_64bit_int_more.cpp -o mystery1
ubuntu@ubuntu2:~/Documents/github/KUtrace-experiments/Solution/Ch2$ ./mystery1
division:
1000000 iterations, 39508252 cycles, 39.51 cycles/iteration
1758526583 0
```



```
ubuntu@ubuntu2:~/Documents/github/KUtrace-experiments/Solution/Ch2$ gcc -O0 myst
ery1_double_operations.cpp -o mystery1
ubuntu@ubuntu2:~/Documents/github/KUtrace-experiments/Solution/Ch2$ ./mystery1
addition:
1000000 iterations, 41134664 cycles, 41.13 cycles/iteration
1758527510 22000000.000000
multiplication:
1000000 iterations, 40923636 cycles, 40.92 cycles/iteration
1758527510 inf
division:
1000000 iterations, 62651126 cycles, 62.65 cycles/iteration
1758527510 0.000000
```

2.9) mystery1_double_operations.cpp
There is no sudden change in latency.



```
920000 iterations, 17147352 cycles, 17.15 cycles/iteration
1758529723 0.000000
division:
930000 iterations, 17327118 cycles, 17.33 cycles/iteration
1758529723 0.000000
division:
940000 iterations, 17506480 cycles, 17.51 cycles/iteration
1758529723 0.000000
division:
950000 iterations, 17730052 cycles, 17.73 cycles/iteration
1758529723 0.000000
division:
960000 iterations, 17961772 cycles, 17.96 cycles/iteration
1758529723 0.000000
division:
970000 iterations, 18140794 cycles, 18.14 cycles/iteration
1758529723 0.000000
division:
980000 iterations, 18320850 cycles, 18.32 cycles/iteration
1758529723 0.000000
division:
990000 iterations, 18501956 cycles, 18.50 cycles/iteration
1758529723 0.000000
```