ASSIGNMENT_2

Command: lscpu

```
[karnamrh@adhara:21]> lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                8
On-line CPU(s) list:   0-7
Thread(s) per core:    2
Core(s) per socket:    4
Socket(s):             1
NUMA node(s):          1
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 94
Model name:            Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz
Stepping:              3
CPU MHz:               800.000
BogoMIPS:              6816.06
Virtualization:        VT-x
L1d cache:             32K
L1i cache:             32K
L2 cache:              256K
L3 cache:              8192K
NUMA node0 CPU(s):     0-7
[karnamrh@adhara:22]>
```

Test Case 1:

1. Array size = 100M, T=2, index for zero =50M+1

```
[karnamrh@adhara:21]> g++ -O3 MTFindMin.c -lpthread -o MTFindMin
[karnamrh@adhara:22]> MTFindMin 100000000 2 50000001
Sequential search completed in 25 ms. Min = 0
Threaded FindMin with parent waiting for all children completed in 31 ms. Min = 0
Threaded FindMin with parent continually checking on children completed in 0 ms. Min = 0
Threaded FindMin with parent waiting on a semaphore completed in 0 ms. Min = 0
[karnamrh@adhara:23]>
```

Sequential Search	Parent waiting for all children	Parent Continuously checking on all children	Parent waiting on semaphore
25ms	31ms	0ms	0ms

2. Array size = 100M, T=4, index for zero =75M+1

```
[karnamrh@adhara:23]> MTFindMin 100000000 4 75000001
Sequential search completed in 25 ms. Min = 0
Threaded FindMin with parent waiting for all children completed in 17 ms. Min = 0
Threaded FindMin with parent continually checking on children completed in 0 ms. Min = 0
Threaded FindMin with parent waiting on a semaphore completed in 0 ms. Min = 0
[karnamrh@adhara:24]>
```

Sequential Search	Parent waiting for all children	Parent Continuously checking on all children	Parent waiting on semaphore
25ms	17ms	0ms	0ms

3. Array size = 100M, T=8, index for zero =88M

```
[karnamrh@adhara:30]> MTFindMin 100000000 8 88000000
Sequential search completed in 25 ms. Min = 0
Threaded FindMin with parent waiting for all children completed in 15 ms. Min = 0
Threaded FindMin with parent continually checking on children completed in 0 ms. Min = 0
Threaded FindMin with parent waiting on a semaphore completed in 1 ms. Min = 0
[karnamrh@adhara:31]>
```

Sequential Search	Parent waiting for all children	Parent Continuously checking on all children	Parent waiting on semaphore
25ms	15ms	0ms	1ms

4. Array size = 100M, T=2, index for zero =-1 (no zero)

```
[karnamrh@adhara:48]> MTFindMin 100000000 2 -1
Sequential search completed in 25 ms. Min = 1
Threaded FindMin with parent waiting for all children completed in 30 ms. Min = 1
Threaded FindMin with parent continually checking on children completed in 31 ms. Min = 1
Threaded FindMin with parent waiting on a semaphore completed in 31 ms. Min = 1
```

Sequential Search	Parent waiting for all children	Parent Continuously checking on all children	Parent waiting on semaphore
25ms	30ms	31ms	31ms

5. Array size = 100M, T=4, index for zero =-1 (no zero)

```
[karnamrh@adhara:49]> MTFindMin 100000000 4 -1
Sequential search completed in 25 ms. Min = 1
Threaded FindMin with parent waiting for all children completed in 16 ms. Min = 1
Threaded FindMin with parent continually checking on children completed in 24 ms. Min = 1
Threaded FindMin with parent waiting on a semaphore completed in 17 ms. Min = 1
[karnamrh@adhara:50]>
```

Sequential Search	Parent waiting for all children	Parent Continuously checking on all children	Parent waiting on semaphore
25ms	16ms	24ms	17ms

6. Array size = 100M, T=8, index for zero =-1 (no zero)

```
[karnamrh@adhara:54]> MTFindMin 100000000 8 -1
Sequential search completed in 25 ms. Min = 1
Threaded FindMin with parent waiting for all children completed in 16 ms. Min = 1
Threaded FindMin with parent continually checking on children completed in 23 ms. Min = 1
Threaded FindMin with parent waiting on a semaphore completed in 15 ms. Min = 1
[karnamrh@adhara:55]>
```

Sequential Search	Parent waiting for all children	Parent Continuously checking on all children	Parent waiting on semaphore
25ms	16ms	23ms	15ms

Conclusion:

Sequential Search:(Worst Case)
It takes same time for all the test cases and takes maximum time to compute the minimum.

Parent waiting for all children:
performance increases when the number of threads increased. But even if one of the child is done finding zero, parent still waits for other children to complete the task.

Parent continuously checking on all children:
Performance is good when there is zero in array because it terminates all the children if one of the children finds zero. But performance becomes bad when there is no zero because parent wastes CPU cycles while checking the child thread continuously.

Parent Waiting on Semaphore:(Best Case)
Parent has been put on waiting state till the threads find the minimum value. From the first three cases we can see that this method took almost 0ms to find zero. But test case 4 proves that there is no use even if there is max cores but min threads.