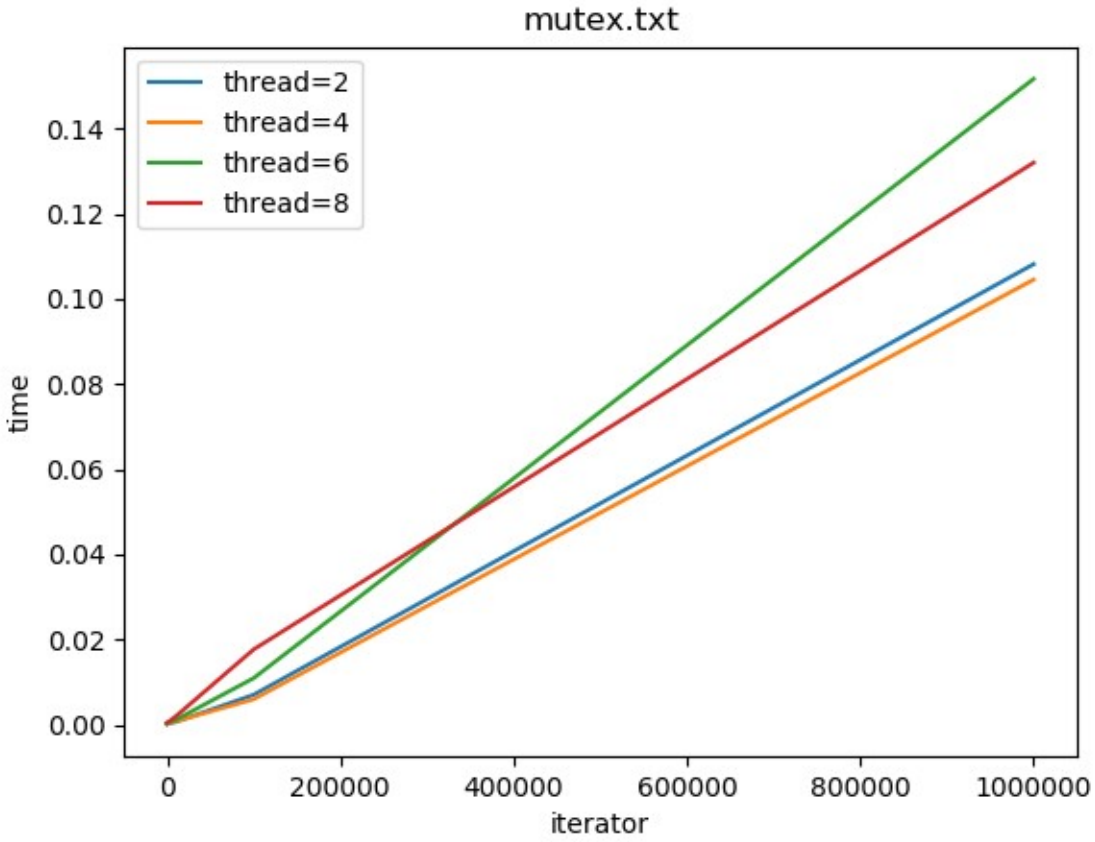
Hardware Specification

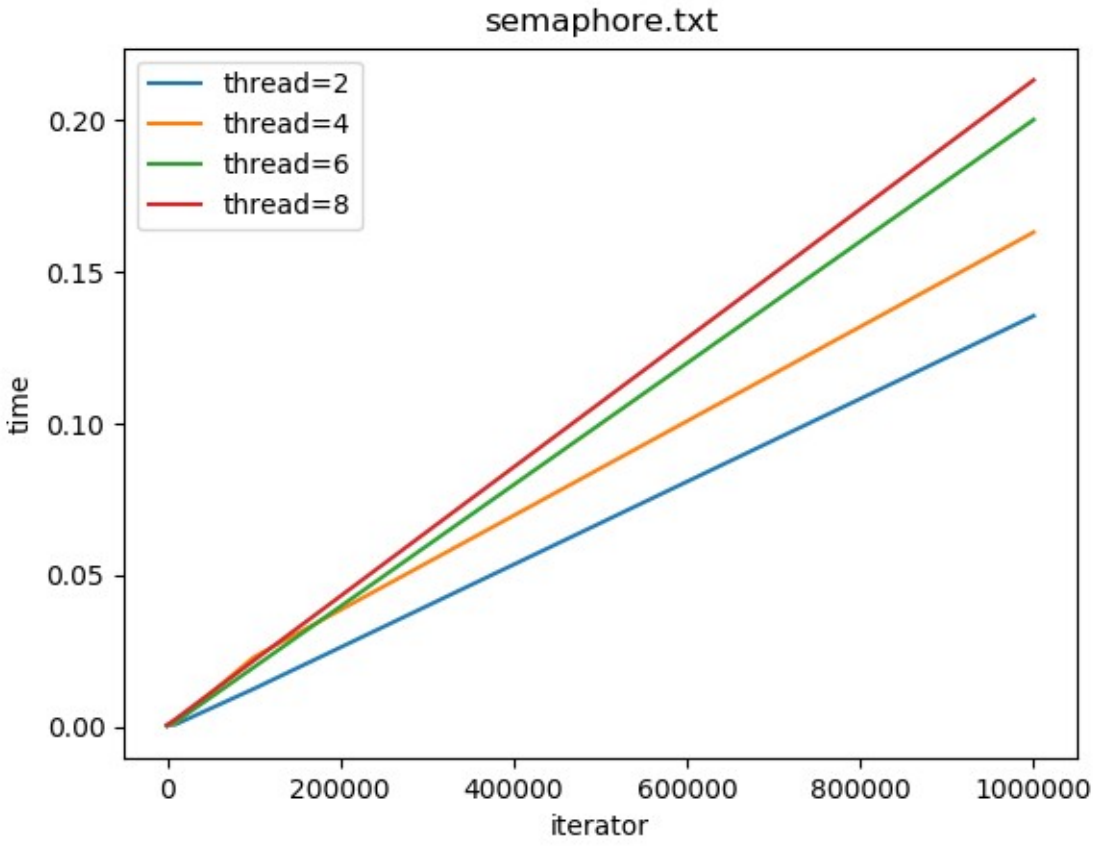
Architecture: x86_64
CPU op-mode(s): 32-bit, 64-bit
Byte Order: Little Endian
CPU(s): 2
On-line CPU(s) list: 0,1
Thread(s) per core: 1
Core(s) per socket: 2
Socket(s): 1
NUMA node(s): 1
Vendor ID: AuthenticAMD
CPU family: 21
Model: 112
Model name: AMD A6-9220 RADEON R4, 5 COMPUTE CORES 2C+3G
Stepping: 0
CPU MHz: 1960.710
CPU max MHz: 2500.0000
CPU min MHz: 1300.0000
BogoMIPS: 4990.57
Virtualization: AMD-V
L1d cache: 32K
L1i cache: 64K
L2 cache: 1024K
NUMA node0 CPU(s): 0,1
Flags: fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse sse2 ht syscall nx mmxext fxsr_opt pdpe1gb rdtscp lm constant_tsc rep_good acc_power nopl nonstop_tsc cpuid extd_apicid aperfmperf pni pclmulqdq monitor ssse3 fma cx16 sse4_1 sse4_2 movbe popcnt aes xsave avx f16c rdrand lahf_lm cmp_legacy svm extapic cr8_legacy abm sse4a misalignsse 3dnowprefetch osvw ibs xop skinit wdt lwp fma4 tce nodeid_msr tbm perfctr_core perfctr_nb bpext ptsc mwaitx cpb hw_pstate ssbd ibpb vmmcall fsgsbase bmi1 avx2 smep bmi2 xsaveopt arat npt lbrv svm_lock nrip_save tsc_scale vmcb_clean flushbyasid decodeassists pausefilter pfthreshold avic v_vmsave_vmload vgif overflow_recov

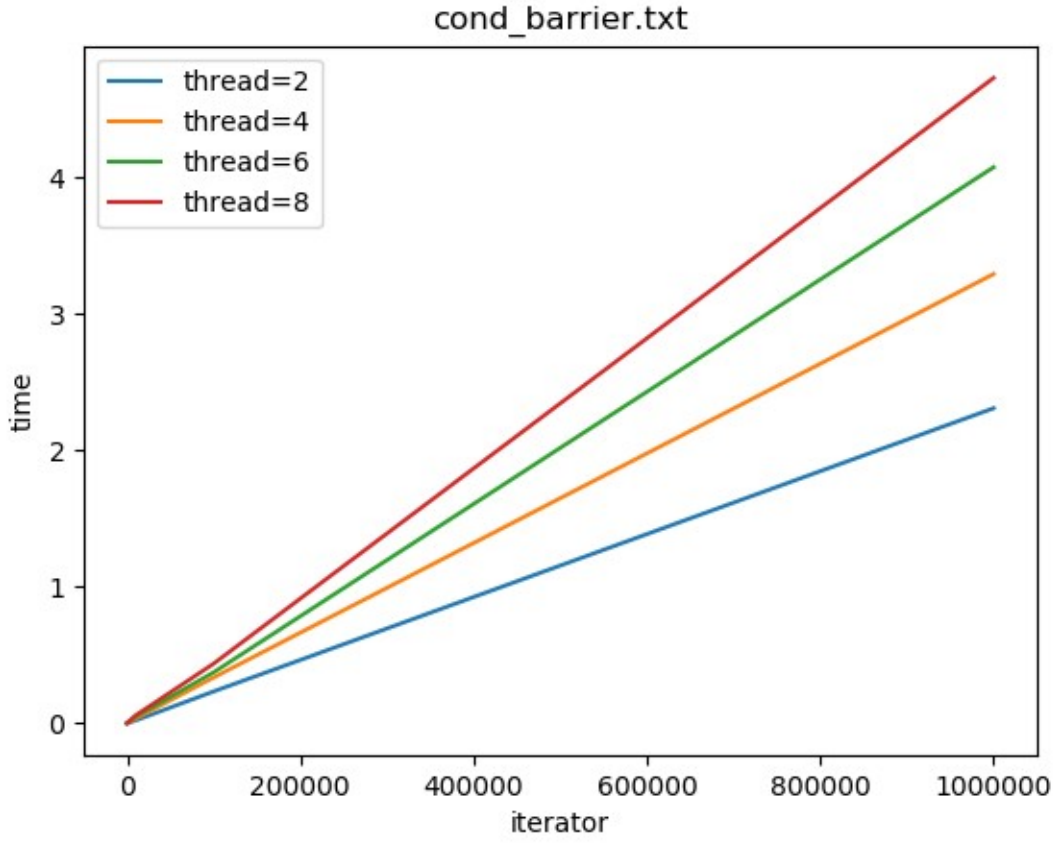
Comparision different synchronisation mechanism by graph

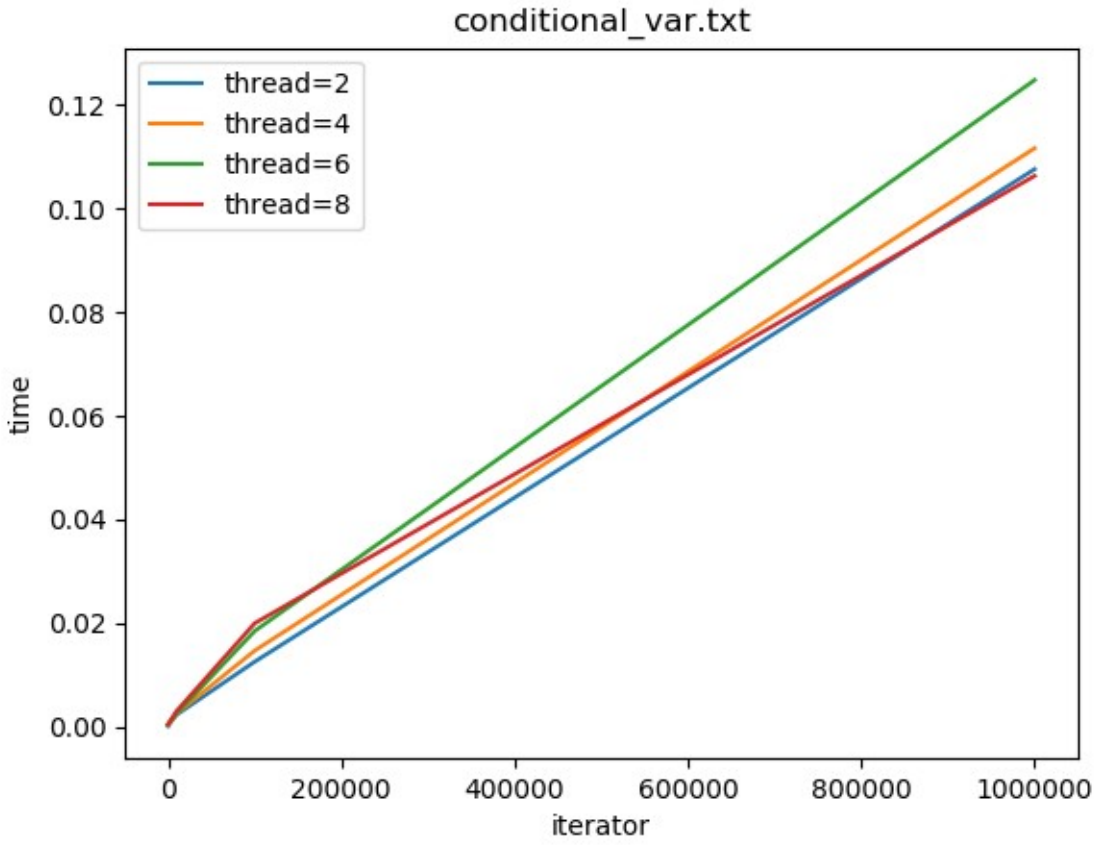
To compare the performance of each technique is used to compute the summation of sereies s(n)=4*(1-(-1)^n/(2*n+1)).
For a given number of thread and iteration running time is computed as average running time after running the program for 5 time.

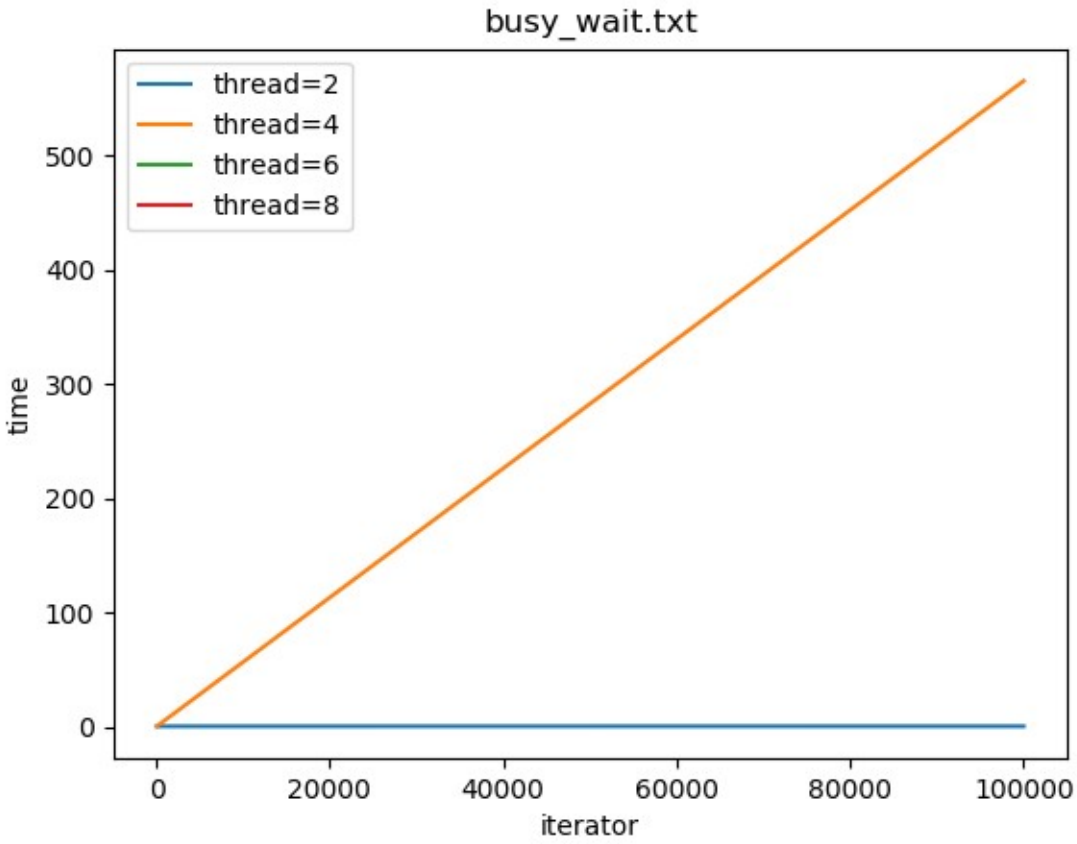
Graph of iteration vs average time taken is drawn for threads number(2,4,6,8) and observations are made.


mutex.txt
thread=2
thread=4
thread=6
thread=8
time
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0
200000
400000
600000
800000
1000000
iterator


semaphore.txt
thread=2
thread=4
thread=6
thread=8
time
0.00
0.05
0.10
0.15
0.20
0
200000
400000
600000
800000
1000000
iterator


cond_barrier.txt
thread=2
thread=4
thread=6
thread=8
time
iterator
0
1
2
3
4
0
200000
400000
600000
800000
1000000


conditional_var.txt
thread=2
thread=4
thread=6
thread=8
time
iterator
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0
200000
400000
600000
800000
1000000



Observation:
Since my pc has 2 cores with 1 thread in each core it perforformes best when number of threads is 2
Mutex and semaphore perfoms better than other techniques in all aspects(from the graph)

Busy_wait performs the worst among all the techniques implemented in the experiment.

The experiment data for each technique is available in same folder for references.