PARTH CHOPRA
2016ME10829.

HW-4

SPIN LOCKING.

As we enabled sti() just after acquiring lock.
80104461 8012 0f3 80105965 8015 58af 80100183
80101485 8010 4ff 80130774 8010 56b2 0.
I couldn't look up kernel.asm.

(II) It didn't panic

(III) release()

here exchange instruction acts as a fence between for: lk → cpu = 0; instruction. hence prevents from and. so concurrent access of lock internal structure as if this lk→cpu=0 instruction after releasing lock can result in race conditions.

Uniprocessor locking

```
lock(L){
  cli;
  while(L==0) continue;
  L=0;
  sti;
}
```

```
unlock(L){
  L=1;
}
```

This implementation will not be able to work in a uniprocessor because as soon as a process which has acquired lock context switches out another process tries to acquire the same lock it will spin indefinitely However will not never be able to acquire the lock, classic dead lock.

```
lock(L){
int acquired=0;
while(!acquired){
  cli();
  if(L==1){
    acquired=1;
    L=0;
  }
  sti();
}
}
```

```
unlock(L){
  L=1;
}
```

Yes, this implementation of lock can work on a uniprocessor. However incase after acquiring of lock if interupt handler is called req the same lock it would result in dead lock.

SLEEP & WAKEUP

wakeup only wakes up processes that are already waiting, wakeup marks all process waiting for the channel as RUNNABLE. Yes, channel is just a number, if all all callers of sleep and wakeup follow convention everything is fine. So, even though they are on same channel all processes to sleeping would wakeup however as they are rechecking condition in loop, only right one will wake rest would go back to sleep. Sleeping can also be implemented on different chan

Say there are multiple consumer sleeping at chanel one producer calls wakeup(q) all the consumer thread will become RUNNABLE. However, only one consumer thread would consume others would simply sleep again without consuming anything. Even if some unrelated part of code accidentally wakes consumer it will check condition & sleep again.

XV6 file system.

NOTE b→sector is incorrect
replaced with b→blockno

echo > a.

iallocc) calls log write 34 (Block No) to allocate inode
iupdate() calls log write 34 to modify disk inode
writei() calls log write 59 write data to inode

echo x > a . Block No.

balloc calls log write 58
bzero calls log write 571
writei calls log write 571
iupdate calls log write 34.
writei calls log write 571
iupdate calls log write 34.

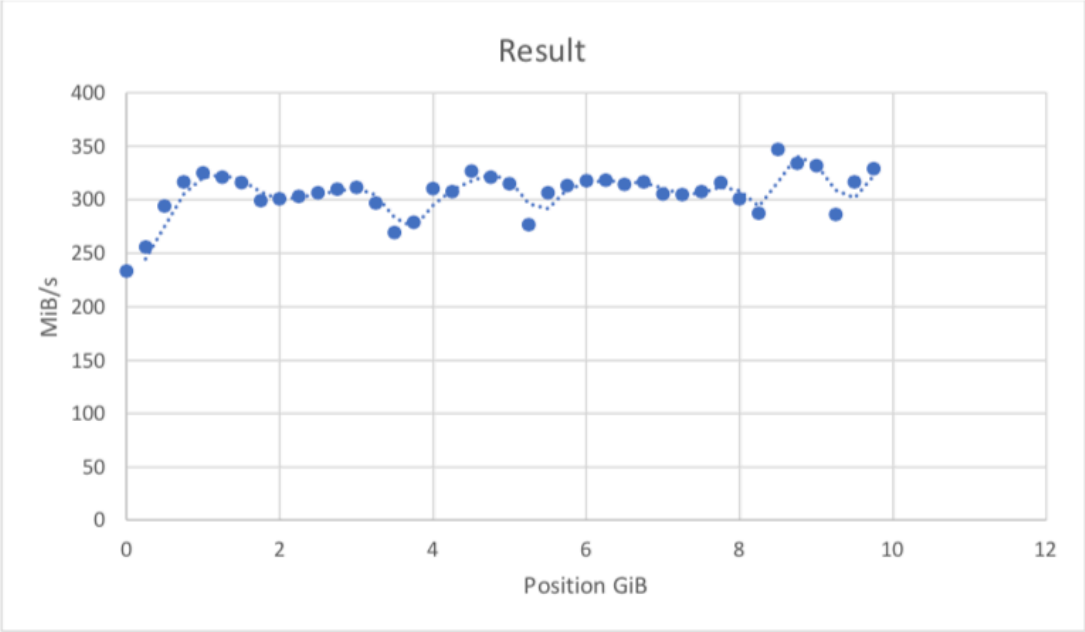
rm a

writei calls log write 59.
iupdate calls log write 34
bfree calls log write 58.
iupdate calls log write 34.
iupdate calls log write 34.

ZCAV

I run ubuntu as a VM on my MacBook Air (13-inch, Early 2015): APPLE SSD SM0128G
I have partitioned 10GB space on my disk for ubuntu
Here are the results

0.00 233.35 1.097
0.25 256.02 1.000
0.50 294.28 0.870
0.75 316.80 0.808
1.00 325.21 0.787
1.25 321.36 0.797
1.50 315.89 0.810
1.75 299.42 0.855
2.00 300.73 0.851
2.25 303.42 0.844
2.50 306.70 0.835
2.75 309.76 0.826
3.00 311.53 0.822
3.25 297.23 0.861
3.50 269.23 0.951
3.75 279.12 0.917
4.00 310.62 0.824
4.25 307.92 0.831
4.50 326.69 0.784
4.75 321.02 0.797
5.00 314.96 0.813
5.25 276.85 0.925
5.50 306.66 0.835
5.75 313.11 0.818
6.00 317.74 0.806
6.25 318.14 0.805
6.50 314.52 0.814
6.75 316.64 0.808
7.00 305.25 0.839
7.25 304.68 0.840
7.50 307.94 0.831
7.75 316.26 0.809
8.00 300.69 0.851
8.25 287.19 0.891
8.50 347.27 0.737
8.75 333.97 0.767
9.00 332.15 0.771
9.25 286.49 0.894
9.50 316.73 0.808
9.75 328.94 0.778



Read 10 gigs in 33 seconds, 304 megabytes per second.

Now for USB drive
There was some issue with xfat so couldn't use it on VM moreover zcav can't be used on macOS so tried a similar application.