1)

    a)

```
            #lock(lk)
    lock:   LI      $t0, 1          #Load the value of 1
            LL      $t1, 0($a0)     #Load the 'lock'
            SC      $t0, 0($a0)     #Lock the 'lock'
            beq     $t0, $0, lock   #If the lock fails, try again

            #shvar = max(shvar, x)
            LW      $t1, 0($a1)     #get shvar
            SLT     $t0, $t1, $a2   #if value of shvar < x, $t0 = 0, else $t0=1
            BEQ     $t0, $0, smax   #If shvar is max
            MOVE    $a1, $a2        #shvar = x
            J       unlock

    smax:   MOVE    $a1, $a1        #shvar = shvar

            #unlock(lk)
    unlock: LI      $t0, 0          #Set the lock to open
            SW      $t0, 0($a0)     #Open the lock again
```

    b)

Suppose two processes are running on two different processors. One of the processors will access the
lock section first, upon doing this, the first program will lock the lock with ll/sc. This memory address will
now be locked. It will continue on to the critical section. During this time, the second process will hit the lock.
The second program is unable to unlock the lock, because the first program has not unlocked the lock at the memory address
$a0. The second program will continue to loop until the lock is cleared, that is, the first program finishes the critical
section, and executes the final call to unlock the memory address. After doing this, the second process is now able to
access the critical section.

If the two programs try to access the 'lock' subroutine at EXACTLY the same time, the hardware will handle it, and one
processor will be given access to the lock depending on which processor the hardware biases.

2)

    a)

Smallest Positive is $1.000000000 \times 10 ^ (011\ 110)$, Largest Positive is $1.11111111 \times 10 ^ (111\ 110)$

    b)

```
  1 100001 111111110 = -1.776 x 10^1  = -17.7600
+ 0 011111 101010101 =  1.525 x 10^-1 =   0.1525
                                        + -------
                                         -17.6053

                          then: -1.76053 x 10^-1
```

3)

    a)   1/CPI = Instructions per Cycle   1/R = Clock Cycle Time(s)

        So, Then Instructions per cycle * Cycle Time = (instructions/cycles) * (Cycles/s) = Instructions/seconds, so for p0:
           1 / (CPI * R)

    b) Using Above:
       p1:

```
    1 / (1.5 * (3.0*10^9)) = 2.2222222e-10

p3:
    1 / (2.2 * (4.0*10^9)) = 1.1363636e-10

So, Processor p1 is faster, with 2.2222222e-10 instructions per second
```

c)
```
1 / CPI = Instructions per Cycle = IPC
R = cycles/second
T = Time
----------
Instructions / Cycle  *  Cycles / Second  * Seconds/1 = Instructions
=
(IPC * R) * T
=
T *((1/CPI)*R) = Instructions
    AND
T * R = Cycles
```

d)
```
p1:
    (10) * (3.0*10^9) = 30000000000 cycles
    (10) * ((1/(1.5))*(3.0*10^9)) = 20000000000 instructions
p2:
    (10) * (2.5*10^9) = 25000000000 cycles
    (10) * ((1/1.0)) * (2.5*10^9) = 25000000000 instructions

p3:
    (10) * (4.0*10^9) = 40000000000 cycles
    (10) * ((1/(2.2))*(4.0*10^9)) = 18181818181.8 instructions
```

e)
```
Time = (Num Instructions * CPI * CycleTime)
     = (NumInstructions * CPI) / (Clock rate)
so...

T * (1-a/100) = Instructions * CPI * (1+(b/100)) / Rate

Rate = Instructions * CPI * (1+(b/100)) / T * (1-a/100)
so...

Rate*((1+(b/100)) /(1-a/100)) = New Rate
```

f)
```
((1+b/100)/(1-a/100)) = 1.2 / 0.7 = 1.71

p1:
    3.0 * 1.71 = 5.13 GHz

p2:
    2.5 * 1.71 = 4.27 GHz

p3:
    4.0 * 1.71 = 6.84 GHz
```

assignment5.txt