(1.a) Different processes do not share global memory, while different threads in a process all share that processes memory. The advantage processes have over threads are processes can have multiple threads and not share memory with other processes. This might be a better choice to handle intense tasks. While the advantages threads have over processes are that they are considered to be lightweight, not being power intensive, which will be much better for smaller tasks. Threads also are time efficient compared to processes.

(1.b)

1. pthread_create() - creates a new thread, or adds a new thread to an existing process
2. pthread_mutex_unlock() - releases the mutex or unlocks mutex if it is previously locked and part of a thread.
3. pthread_cond_signal() - unblocks at least one of the threads that are blocked on the specified condition variable cond (if any threads are blocked on cond).

(1.c)

```
#define NUM_SLICES 100
#define H 10.0 / NUM_SLICES

pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;

double answer;

double f(double x) {}

double trapezoid(double a, double h) { return h * (f(a) + f(a + h)) / 2.0;
}

void *IntegratePart(void *i)
{
    double curr_x, area;
    pthread_mutex_lock(&mutex);

    curr_x = H * i;
    area = trapezoid(curr_x, H);
    answer = answer + area;
```

```
    pthread_mutex_unlock(&mutex);
    pthread_exit(NULL);
}

int main(int argc, const char *argv[])
{
    answer = 0.0;
    pthread_t thread[NUM_SLICES];
    pthread_mutex_init(&mutex, NULL);
    long rc, t;
    for (t = 0; t < NUM_SLICES; t++)
    {
        rc = pthread_create(&thread[t], NULL, IntegratePart, (void *)t);
        if (rc)
        {
            printf("ERROR");
            exit(-1);
        }
    }
    for (t = 0; t < NUM_SLICES; t++)
    {
        pthread_join(thread[t], NULL);
    }
    printf("%f\n", answer);
    return 0;
}
```

(1.d) Two threads thread-1 and thread-2 are running concurrently, where global is a global variable, while local is local to each thread, each of which does the following: local = global; local = local +1; global = local. Assuming that global is initialised to 0 initially, the execution could result in global having a final value of 1 by storing global into the local value in thread-1, where local = 0; then increasing it, local =1; then before storing it back to global, meanwhile thread-2 also stores global into local, so local = 0; then increasing it, local =1; and it will be stored back as 1 regardless of whatever thread is stored into global.

(3.a)
The lifetime of a typical programme running on a typical computer is somewhat like the age of the programme and it shows how long the programme would run without breaking or crashing. The aspect of its behaviour that has the most influence on the design of both hardware and operating system software is the part where the CPU has to wait for slow I/O. This creates the need to improve the CPU to use maximum potential of the processing power, which resulted in the creation of pipeline processors that are capable of fetching, decoding, and executing instructions spontaneously at any given instant.

(3.b)
Two hardware features that have been provided to facilitate the implementation of modern operating systems:

1. The NV storage medium
2. Control data flow hardware and Data Buffer Optimiser

(3.c)

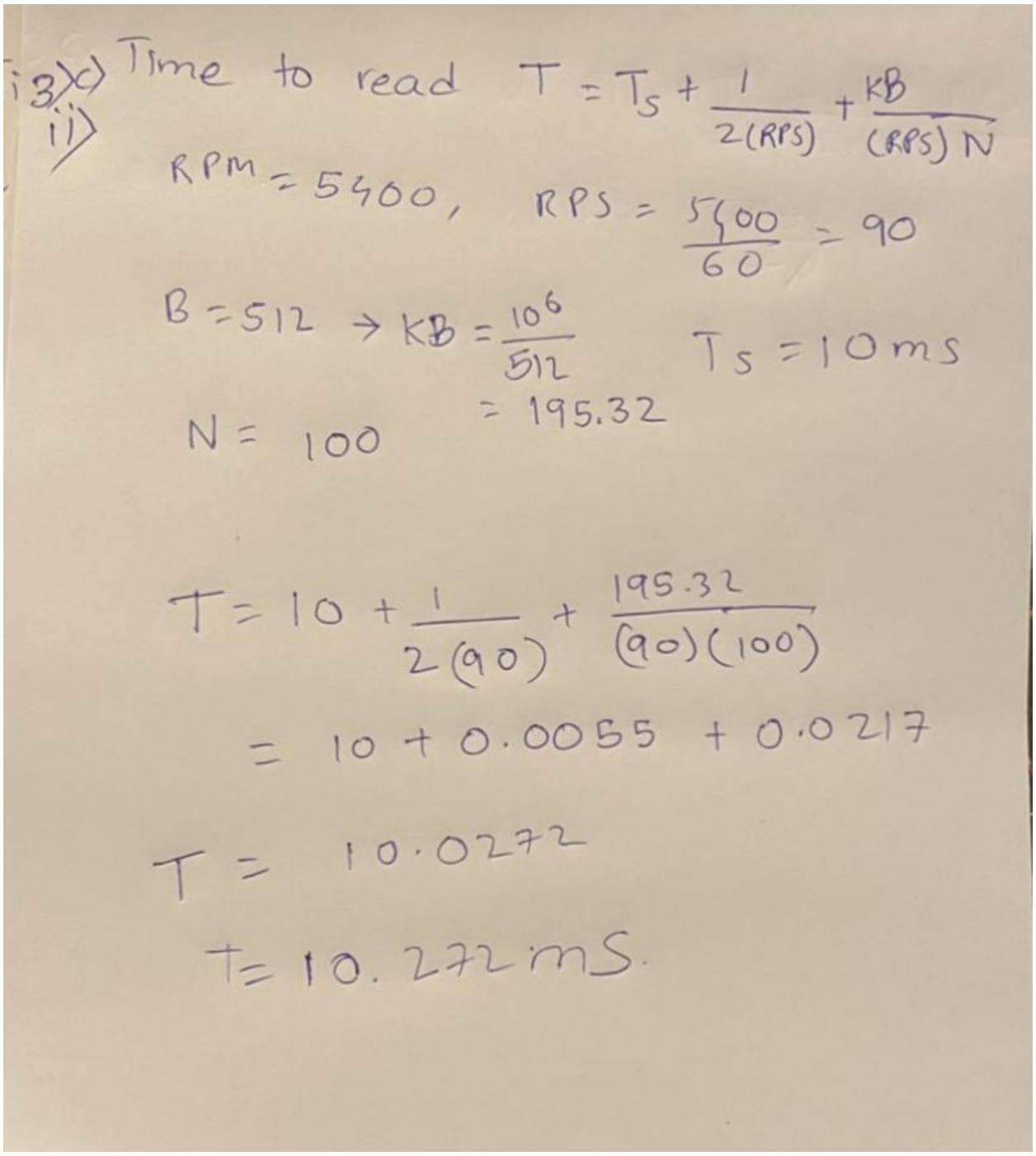
(i)

“Track” - A group of “Sectors”.

“Sector” - basic units of data transfer broken down from “Tracks” usually sized around 512 bytes.

“Seek Time” - time the delay to reach the “Track”

(ii)



3) c) Time to read $T = T_s + \frac{1}{2(RPS)} + \frac{KB}{(RPS)N}$

ii)

$RPM = 5400, \quad RPS = \frac{5400}{60} = 90$

$B = 512 \rightarrow KB = \frac{10^6}{512} = 195.32$

$T_s = 10\,ms$

$N = 100$

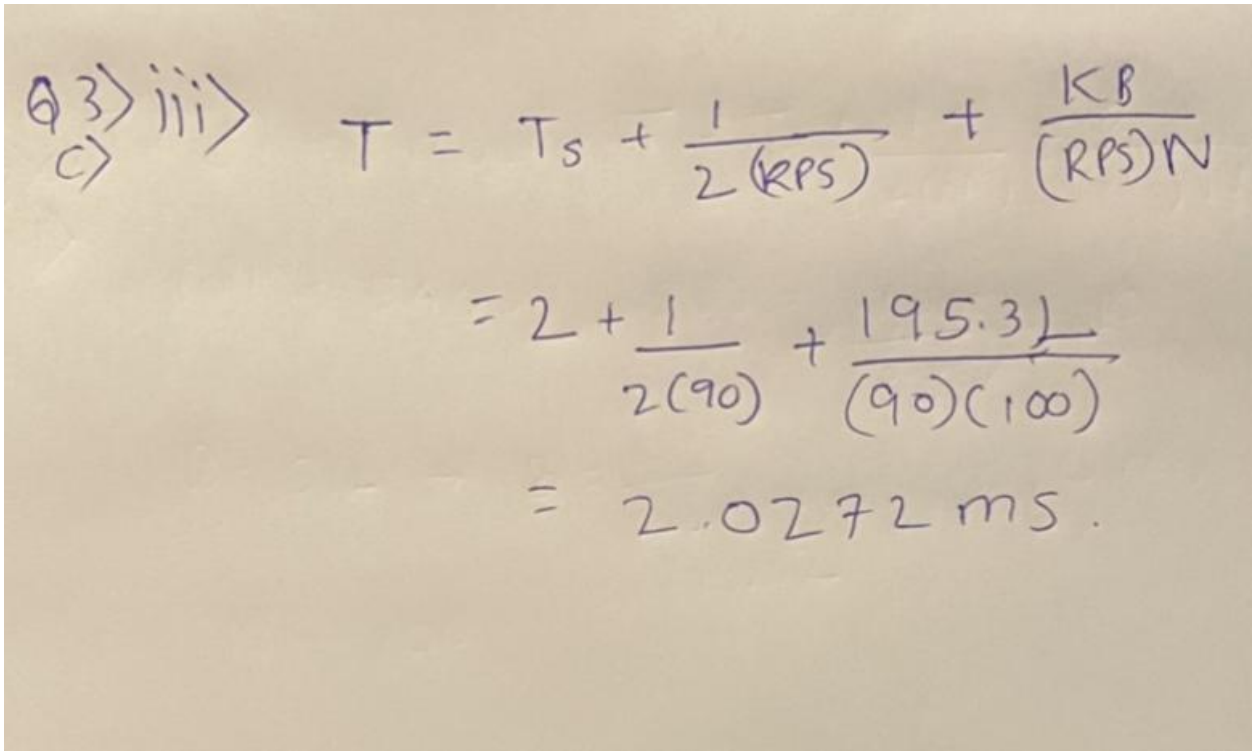
$$T = 10 + \frac{1}{2(90)} + \frac{195.32}{(90)(100)}$$

$$= 10 + 0.0055 + 0.0217$$

$$T = 10.0272$$

$$T = 10.272\,ms.$$

(iii)



(3.d) the File-system concepts of “Inode”, “Filename”, and the distinction between “Open” and “Closed” files.

Inode - represents a file and contains related information

FileName - name given to a file by the user, usually being in the form of a String mapping to the Inode.

the distinction between “Open” and “Closed” files is that an opened file will have a file descriptor active, while a closed file has it’s deleted as it’s closed