Aleks Murauskas

260718389

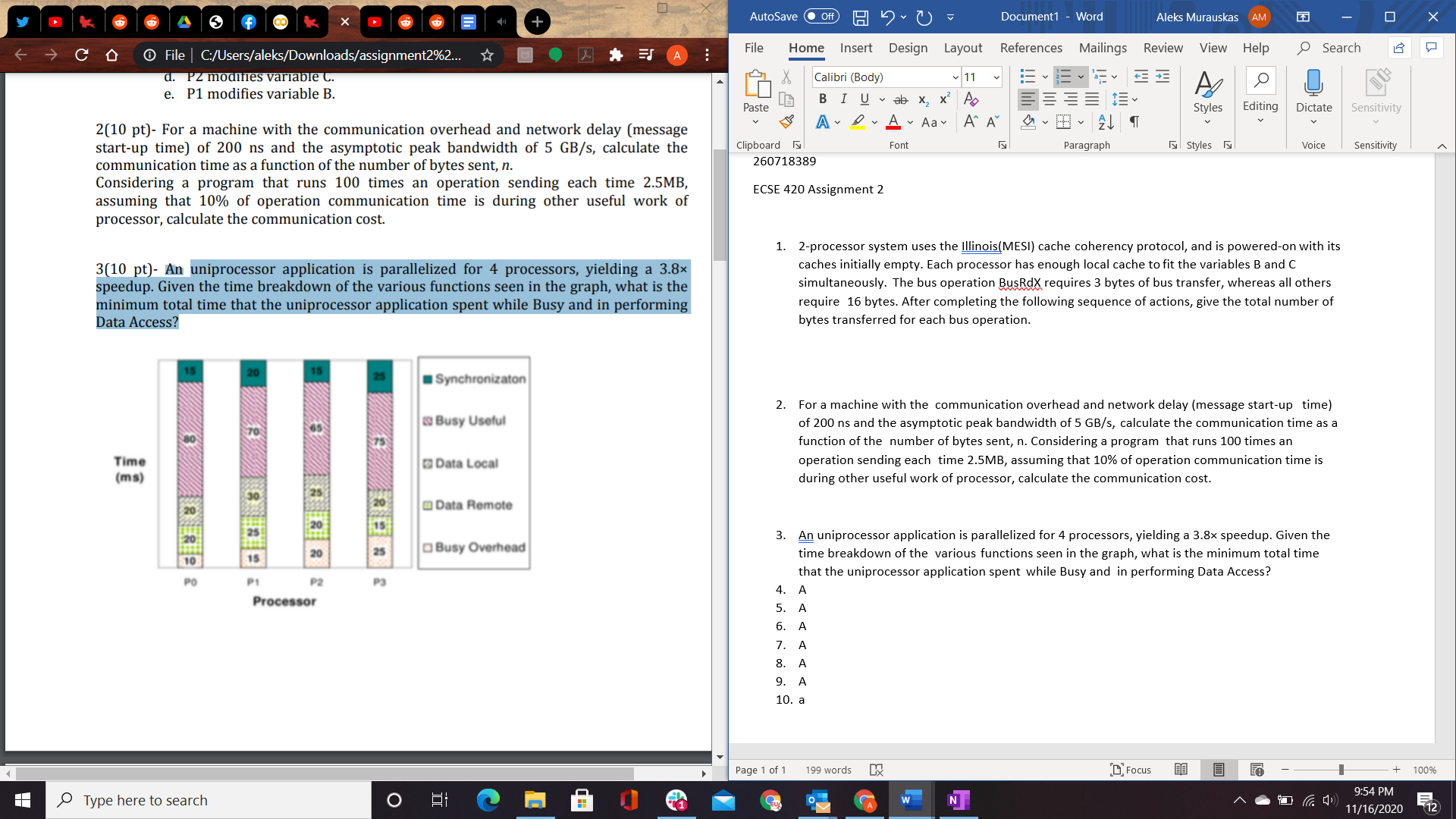
ECSE 420 Assignment 2

1. 2-processor system uses the Illinois(MESI) cache coherency protocol, and is powered-on with its caches initially empty. Each processor has enough local cache to fit the variables B and C simultaneously. The bus operation BusRdX requires 3 bytes of bus transfer, whereas all others require 16 bytes. After completing the following sequence of actions, give the total number of bytes transferred for each bus operation.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | P1 State | P2 State | BUS Action | Bytes for this action | Total Bytes transferred so far |
| P1 loads Variable B | (Holds B) | Empty | BusRd | 16 | 16 |
| P2 loads variable C | Holds B | Holds C | BusRd | 16 | 32 |
| P1 Loads Variable C | Holds BC | Holds C | BusRD | 16 | 48 |
| P2 Modifies Variable C | Holds B C’  C Now invalid | Holds C | BusRDX | 3 | 51 |
| P1 modifies variable B | Holds B C  B updated | Holds C | None | 0 | 51 |

A total number of 51 Bytes

1. For a machine with the communication overhead and network delay (message start-up time) of 200 ns and the asymptotic peak bandwidth of 5 GB/s, calculate the communication time as a function of the number of bytes sent, n. Considering a program that runs 100 times an operation sending each time 2.5MB, assuming that 10% of operation communication time is during other useful work of processor, calculate the communication cost.
2. An uniprocessor application is parallelized for 4 processors, yielding a 3.8× speedup. Given the time breakdown of the various functions seen in the graph, what is the minimum total time that the uniprocessor application spent while Busy and in performing Data Access?



1. A)What is the difference between a write-through cache and a write-back cache? Why does using write-through caches in a shared memory multiprocessor not scale well to a large number of processors?

Write Through: All cache memory writes are written back to main memory at the same time as the cache write

Write Back: A write in Cache memory does not immediately write back into main memory unless another cache line changes and it requires an update.

Write Through scales poorly because every time an update is made in cache the main memory must be updated, resulting in a much larger amount of BUS traffic, which would result in an overload of bus traffic. This could lead to a bottleneck.

B)An alternative to the snooping cache coherence protocol is a directory-based scheme by which a table is maintained that shows where in the system is each cache line. Describe briefly the advantages and disadvantages of each system.

The directory based scheme scales better than snooping cache. As there are fewer interactions in the system as a cache miss doesn’t notify the whole cache-snooping system. As processors increase in numbers, this would eventually cause a bottleneck. Cache snooping systems however require less memory to implement.

1. Express a Test&Set instruction using LL and SC instruction using a pseudo-assembler code.

testset:

ADDI R2, R0, #1

LL R1, lock //Check if free, If not free loop

BNEZ R1, testset

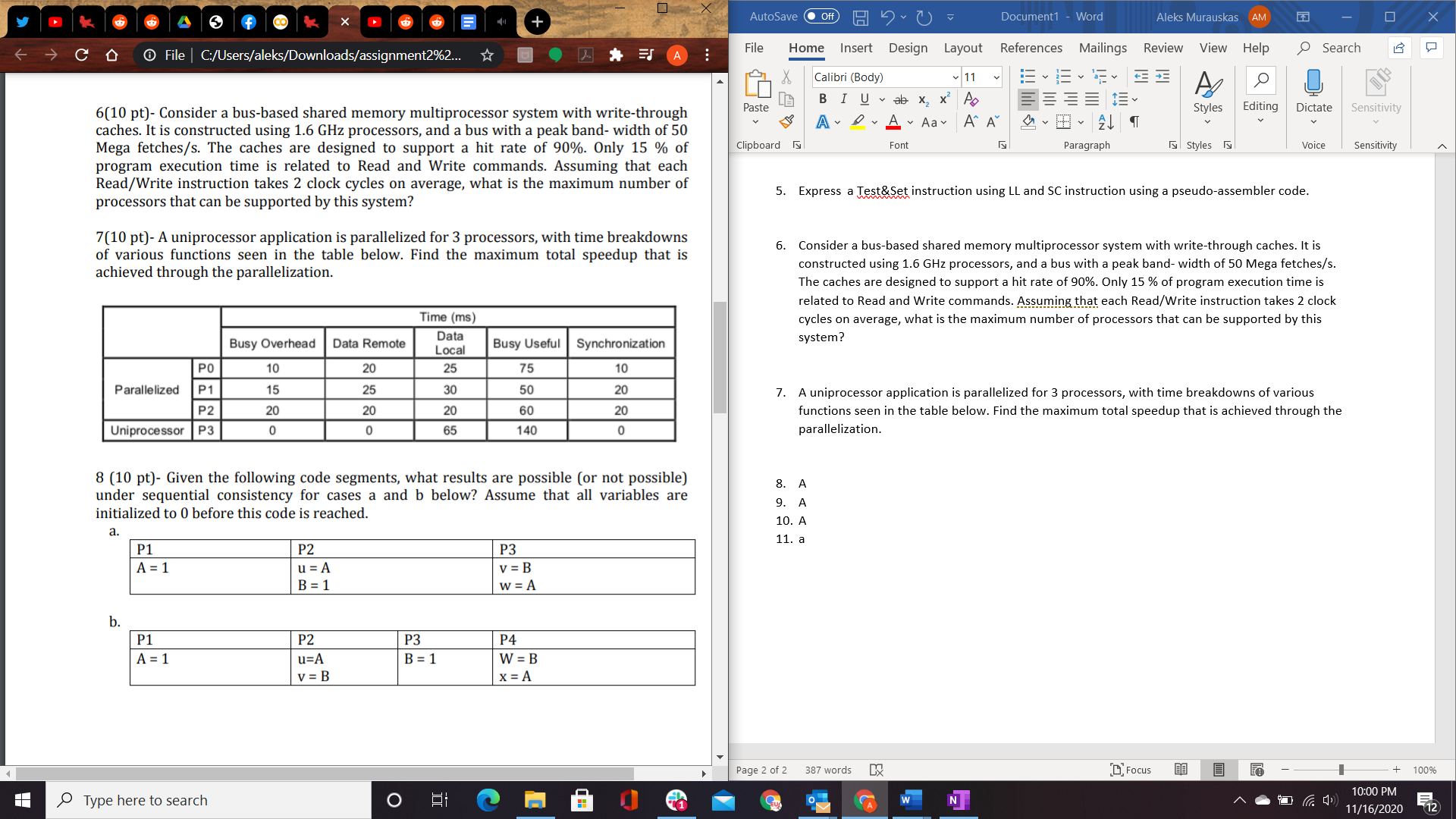
SC R2, location //Check if store succeed, if it failed loop

BEQZ R2, testset

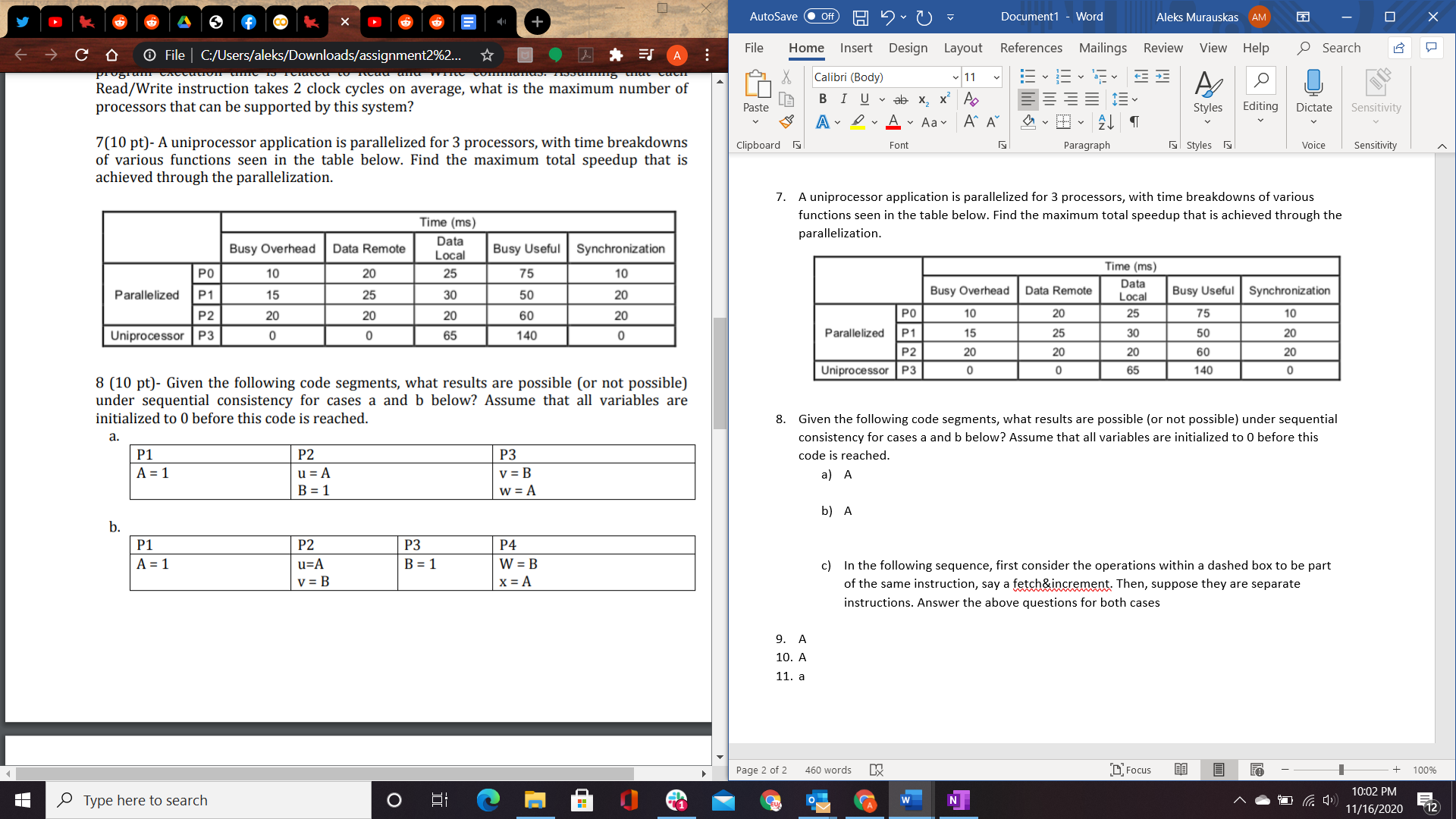
//Section where further instruction is

Unlock: SW lock, #0 //Reset Lock to unlock

1. Consider a bus-based shared memory multiprocessor system with write-through caches. It is constructed using 1.6 GHz processors, and a bus with a peak band- width of 50 Mega fetches/s. The caches are designed to support a hit rate of 90%. Only 15 % of program execution time is related to Read and Write commands. Assuming that each Read/Write instruction takes 2 clock cycles on average, what is the maximum number of processors that can be supported by this system?
2. A uniprocessor application is parallelized for 3 processors, with time breakdowns of various functions seen in the table below. Find the maximum total speedup that is achieved through the parallelization.



1. Given the following code segments, what results are possible (or not possible) under sequential consistency for cases a and b below? Assume that all variables are initialized to 0 before this code is reached.



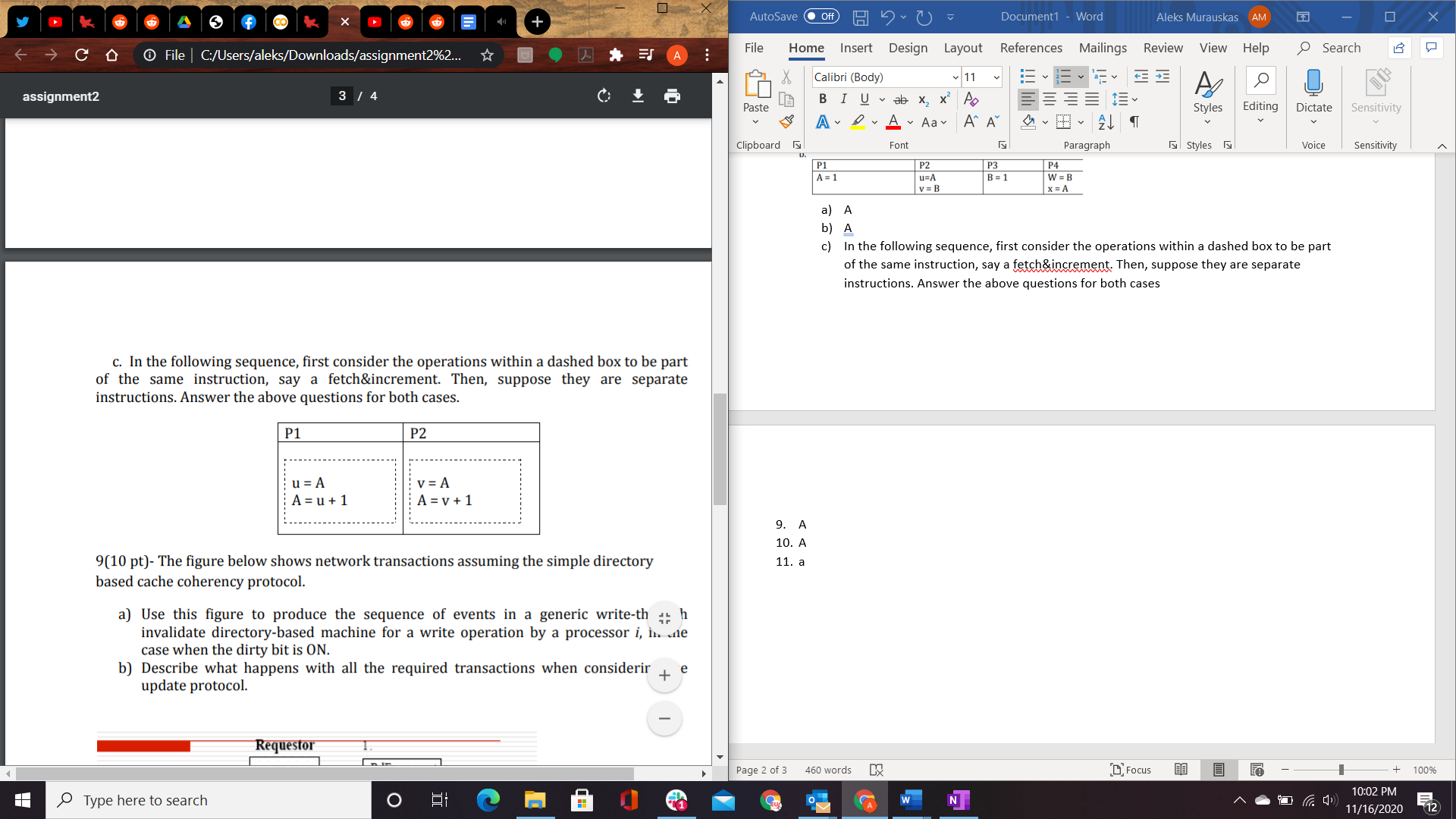
1. a

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| U value | V Value | W Value | Possible? | Sequence |
| 0 | 0 | 0 | Yes |  |
| 0 | 0 | 1 | Yes |  |
| 0 | 1 | 0 | Yes |  |
| 0 | 1 | 1 | Yes |  |
| 1 | 0 | 0 | No | N/A |
| 1 | 0 | 1 | Yes |  |
| 1 | 1 | 0 | No | N/A |
| 1 | 1 | 1 | Yes |  |

1. A

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| U value | V value | W Value | X Value | Possible? | Sequence |
| 0 | 0 | 0 | 0 | Yes |  |
| 0 | 0 | 0 | 1 | Yes |  |
| 0 | 0 | 1 | 0 | Yes |  |
| 0 | 0 | 1 | 1 | Yes |  |
| 0 | 1 | 0 | 0 | Yes |  |
| 0 | 1 | 0 | 1 | Yes |  |
| 0 | 1 | 1 | 0 | Yes |  |
| 0 | 1 | 1 | 1 | Yes |  |
| 1 | 0 | 0 | 0 | Yes |  |
| 1 | 0 | 0 | 1 | Yes |  |
| 1 | 0 | 1 | 0 | No | N/A |
| 1 | 0 | 1 | 1 | Yes |  |
| 1 | 1 | 0 | 0 | Yes |  |
| 1 | 1 | 0 | 1 | Yes |  |
| 1 | 1 | 1 | 0 | Yes |  |
| 1 | 1 | 1 | 1 | Yes |  |

1. In the following sequence, first consider the operations within a dashed box to be part of the same instruction, say a fetch&increment. Then, suppose they are separate instructions. Answer the above questions for both cases.



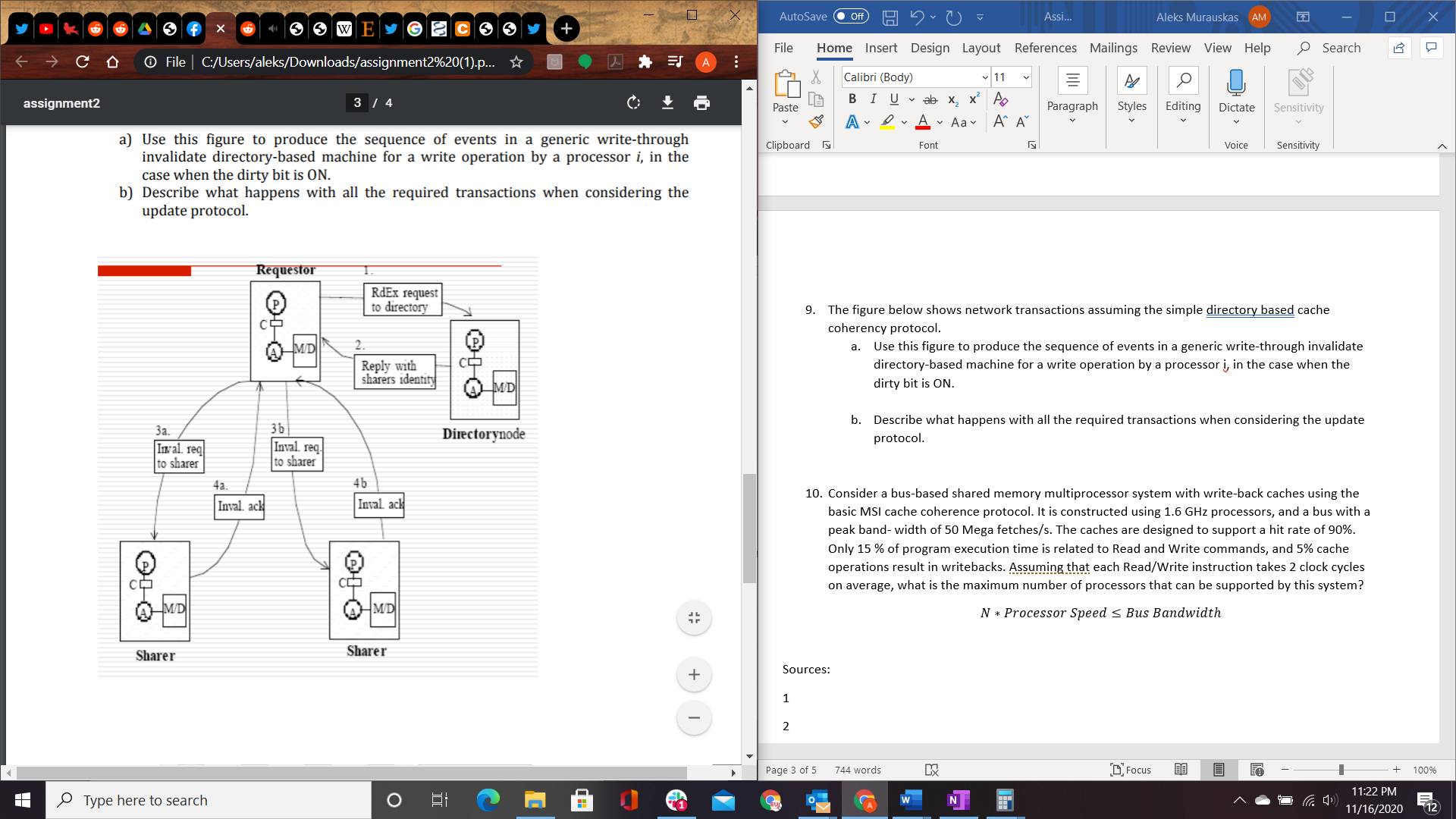
Same Instruction

|  |  |  |  |
| --- | --- | --- | --- |
| U value | V value | Possible? | Sequence |
| 0 | 0 | No | N/A |
| 0 | 1 | Yes |  |
| 1 | 0 | Yes |  |
| 1 | 1 | No | N/A |

Separate Instruction

|  |  |  |  |
| --- | --- | --- | --- |
| U value | V value | Possible? | Sequence |
| 0 | 0 | Yes |  |
| 0 | 1 | Yes |  |
| 1 | 0 | Yes |  |
| 1 | 1 | No |  |

1. The figure below shows network transactions assuming the simple directory based cache coherency protocol.



* 1. Use this figure to produce the sequence of events in a generic write-through invalidate directory-based machine for a write operation by a processor i, in the case when the dirty bit is ON.
* Requestor I sends RdEx request to Directory node
* Directory returns IDs of processors that hold that cache line
* Directory updates to mark the other processors have invalid values in their cache and the requestor is valid
* I sends an Invalid request to all processors that the Directory returned
* The Processors that received the invalid requests Acknowledge the request
* I writes new data to cache
* Due to a Write through, the memory is updated with the new value as well
  1. Describe what happens with all the required transactions when considering the update protocol.
* Requestor I sends RdEx request to Directory node
* Directory returns IDs of processors that hold that cache line
* Directory updates to mark I has a copy of the data
* I writes the new data
* I sends the new data to all processors that share the cache line
* All sharers now have the new data in their caches
* Due to a Write through, the memory is updated with the new value as well

1. Consider a bus-based shared memory multiprocessor system with write-back caches using the basic MSI cache coherence protocol. It is constructed using 1.6 GHz processors, and a bus with a peak band- width of 50 Mega fetches/s. The caches are designed to support a hit rate of 90%. Only 15 % of program execution time is related to Read and Write commands, and 5% cache operations result in writebacks. Assuming that each Read/Write instruction takes 2 clock cycles on average, what is the maximum number of processors that can be supported by this system?