|  |  |  |
| --- | --- | --- |
| 1. | True/False | |
|  | Q: | T/F: All shared memory architectures allow for uniform access times from any processor. |
|  |  |  |
|  | A: | False, NUMAs exists, in which case some processors may have to take extra “hops” in order to access certain memory segments. |
|  |  |  |
|  | Ref: | Page 231 |

|  |  |  |
| --- | --- | --- |
| 2. | Multiple Choice | |
|  | Q: | Which of the following is not shared among threads with the same parent process:   1. Files 2. Heap 3. Stack 4. Interrupt routines |
|  |  |  |
|  | A: | C. Every thread gets its own runtime stack and instruction pointer/program counter. Other constructs like the heap belong to the parent process. |
|  |  |  |
|  | Ref: | Page 235 |

|  |  |  |
| --- | --- | --- |
| 3. | Fill in the Blank | |
|  | Q: | Two kinds of memory consistency/critical section protection tools are \_\_\_\_\_\_\_\_\_\_\_ and \_\_\_\_\_\_\_\_\_\_\_\_\_\_. |
|  |  |  |
|  | A: | Mutex locks, semaphore locks, monitors, etc. are all acceptable answers. |
|  |  |  |
|  | Ref: | Pages 240-247 |

|  |  |  |
| --- | --- | --- |
| 4. | Short Answer/Code | |
|  | Q: | Briefly explain why caching data complicates maintaining memory consistency. |
|  |  |  |
|  | A: | Caches are local to only one processor (maybe more, depending on architecture), and cache memory is not seen by all processors. Because of this, cached copies of data can become outdated when another process writes to the original memory, or when a process writes to its cache and fails to update the shared memory before another process has a chance to “see” this change. This adds complexity, but it is manageable. |
|  |  |  |
|  | Ref: | Pages 258-271 |