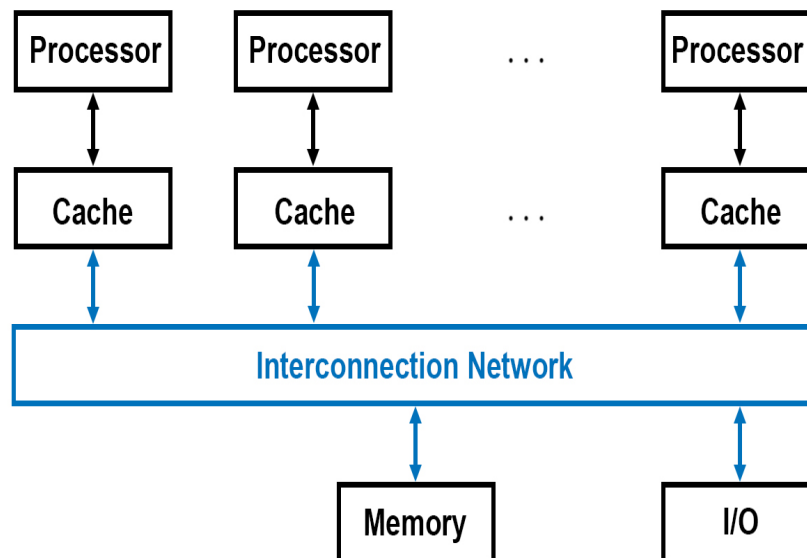


Help

The video and the CYU problems that follow assume a multiprocessor without an L2 cache as shown below:



Show Discussion

 New Post

CACHE COHERENCE

Help

2:03 / 2:03	1.0x			
-------------	------	--	--	--

[Download transcript](#) .txt[Show Discussion](#)[New Post](#)

1. CHECK YOUR UNDERSTANDING (1/1 point)

Which one of the following scenarios would produce a stale copy of block A in the cache of CPU 2?

- ☐ CPU 1 reads block A from memory then CPU 2 reads block A from memory.
- ☐ CPU 2 reads block A from memory then CPU 1 reads block A from memory.
- ☒ CPU 1 reads block A from memory, CPU 2 reads block A from memory, then CPU 1 writes to block A in its cache.
- ☐ CPU 2 reads block A from memory, CPU 2 writes to block A in its cache, then CPU 1 reads block A from memory.

EXPLANATION

In the first two scenarios, both CPUs have read the same block from memory so they are the same.

In the third scenario, after both CPUs read the data, CPU 1 writes the data, creating a stale copy of the block in the cache

of CPU 2.

In the fourth scenario, CPU 2 writes the block and then CPU 1 obtains a stale copy.

Help

Final Check

Save

Hide Answer

You have used 1 of 2 submissions

Show Discussion

 New Post

MESI SOLUTION

2:30 / 2:30

1.0x

Download transcript .txt

Show Discussion

 New Post

2. CHECK YOUR UNDERSTANDING (1/1 point)

Help

Consider the application of the MESI cache coherence protocol when CPU 1 and CPU 2 both have a copy of block A in the *Shared* state. CPU 1 writes to the block after which CPU 2 reads it. Which of the following actions will be taken to ensure that CPU 2 does not end up reading a stale copy of the data?

- ☒ Before CPU 1 writes to the block, it will first send an *Invalidate A* message to CPU 2. ✓
- ☐ Before CPU 1 writes to the block, it will first erase the stale copy in memory.
- ☐ Before CPU 2 reads the block, it will first send an *Invalidate A* message to CPU 1.
- ☐ Before CPU 2 reads the block, it will change its state to *Invalid*.

EXPLANATION

According to the MESI protocol, CPU 1 must invalidate other copies before writing to a block that is marked as *Shared*. When CPU 2 reads the block, because its state is *Invalid*, it will miss, and CPU 1 will provide the block, since it has the most up to date copy.

Final Check

Save

Hide Answer

You have used 1 of 2 submissions





Show Discussion

 New Post

EdX offers interactive online classes and MOOCs from the world's best universities. Online courses from MITx, HarvardX, BerkeleyX, UTx and many other universities. Topics include biology, business, chemistry, computer science, economics, finance, electronics, engineering, food and nutrition, history, humanities, law, literature, math, medicine, music, philosophy, physics, science, statistics and more. EdX is a non-profit online initiative created by founding partners Harvard and MIT.

© 2015 edX Inc.

EdX, Open edX, and the edX and Open edX logos are registered trademarks or trademarks of edX Inc.

[Terms of Service and Honor Code](#)**About edX**[About](#)[News](#)[Contact](#)[FAQ](#)[edX Blog](#)[Donate to edX](#)[Jobs at edX](#)**Follow Us** Facebook Twitter LinkedIn Google+ Tumblr Meetup Reddit Youtube

Help