

MULTIPROCESSOR (I)

Assignment Project Exam Help
<https://tutorcs.com>

Lecturer: Hui Annie Guo

h.guo@unsw.edu.au

K17-501F

WeChat: cstutorcs

Lecture overview

- **Topics**

- **Background**

- **Shared memory multiprocessors**

- **Cache coherence**

<https://tutorcs.com>

WeChat: cstutorcs

- **Suggested reading**

- **H&P Chapter 6.5**

Background

- **Many applications require high performance computing systems. For example**
 - **Scientific applications**
 - **Search engines**
 - **Machine learning**
 - **Big data processing**
- **Increasing frequency often helps to increase performance**

Assignment Project Exam Help

<https://tutorcs.com>

WeChat: cstutorcs

Background (cont.)

- **Power consumption is another design issue**
Modern computer chips are mainly made with the CMOS technology
- **With CMOS, the processor (dynamic) power consumption:**

$$\text{Power} = \text{Capacitive load} \times \text{voltage}^2 \times \text{Frequency}$$

- **Reducing load, voltage and frequency can reduce power consumption**
 - **See an example in the next slide**

For example

Suppose a new CPU has 85% of capacitive load of the old CPU. Its voltage and clock frequency are each reduced by 15%. How much is the power consumption reduced?

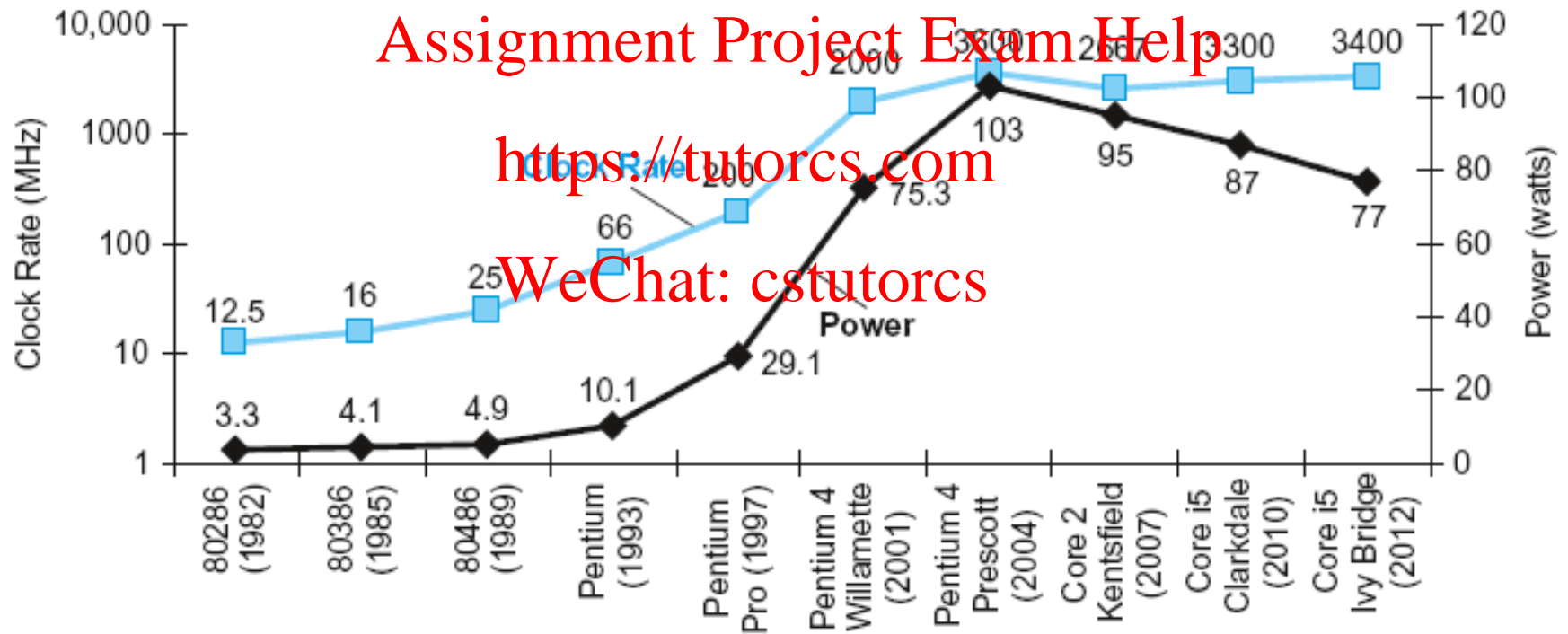
Assignment Project Exam Help

<https://tutorcs.com>

$$\frac{P_{\text{new}}}{P_{\text{old}}} = \frac{C_{\text{old}} \times 0.85 \times (V_{\text{old}} \times 0.85)^2 \times F_{\text{old}} \times 0.85}{C_{\text{old}} \times V_{\text{old}}^2 \times F_{\text{old}}} = 0.85^4 = 0.52$$

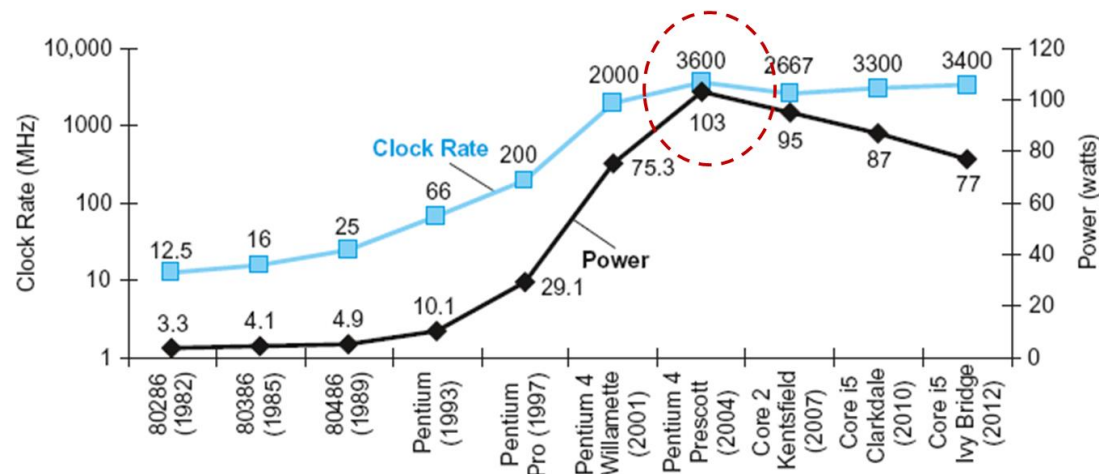
Background (cont.)

- Processor clock rate and power trend



Background (cont.)

- **Frequency cannot be increased arbitrarily**
 - **Power wall exists**
 - **We cannot remove more heat**
- **Increasing frequency becomes more and more technologically hard and costly**
- **How else can we improve performance?**



Multiprocessor

- **A parallel processing hardware system that**
 - **consists of a connection of multiple low-cost processors (cores)**
 - **all processors can perform in parallel, hence achieving high performance**
 - **high throughput for independent tasks.**
 - **is cost effective**
- **One of four processor organization classes**

Assignment Project Exam Help

<https://tutorcs.com>

WeChat: cstutorcs

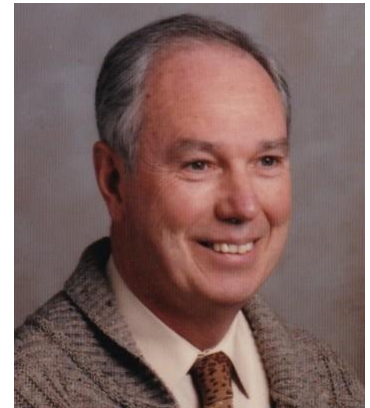
Four processor classes

- **Introduced by Flynn in 1966**
 - **Based on the number of instruction and data streams executed in a “processor”**
 - **Single instruction, single data (SISD)**
 - **Single instruction, multiple data (SIMD)**
 - **Multiple instruction, single data (MISD)**
 - **Multiple instruction, multiple data (MIMD)**

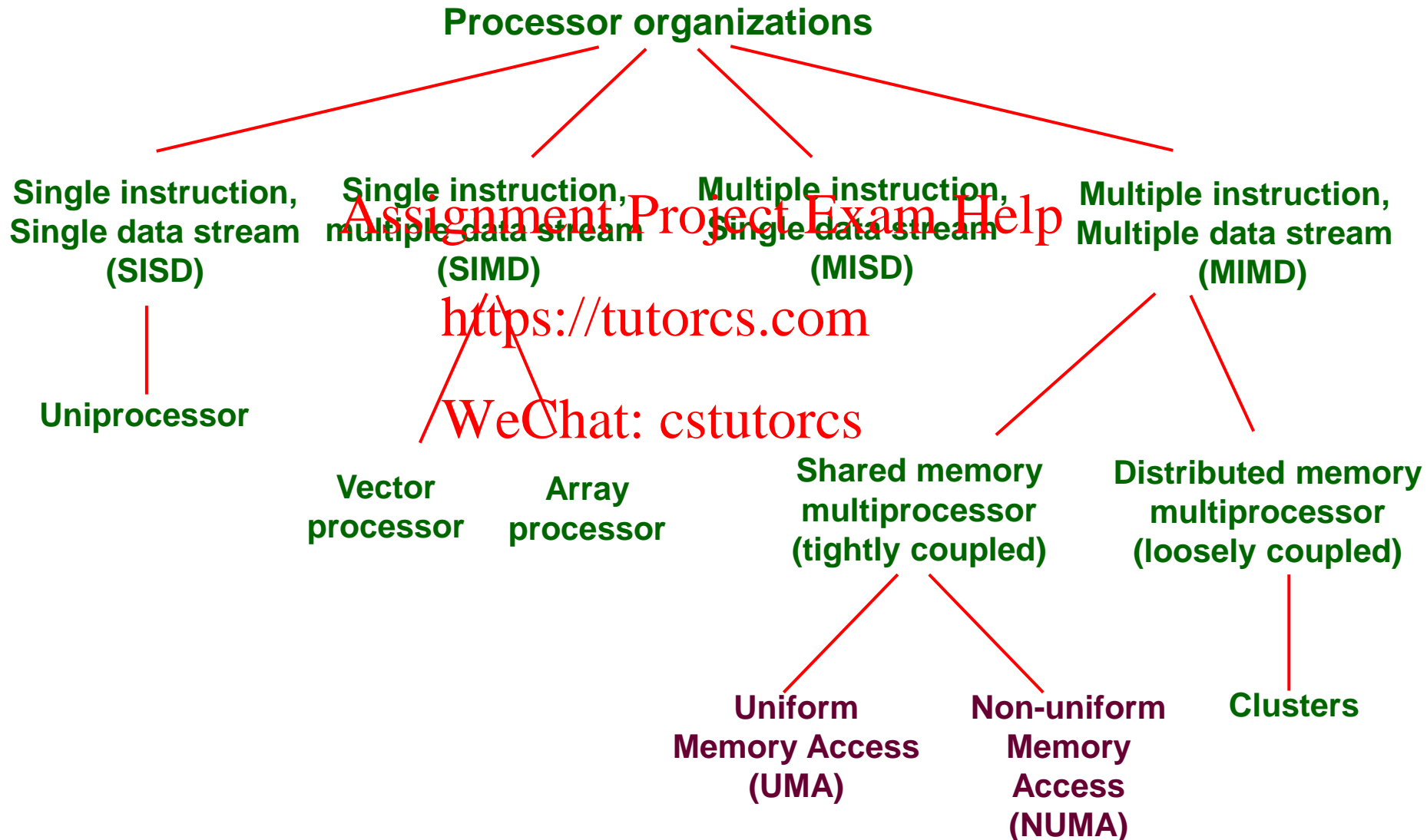
Assignment Project Exam Help

<https://tutorcs.com>

WeChat: cstutorcs



Overview of different processor organizations



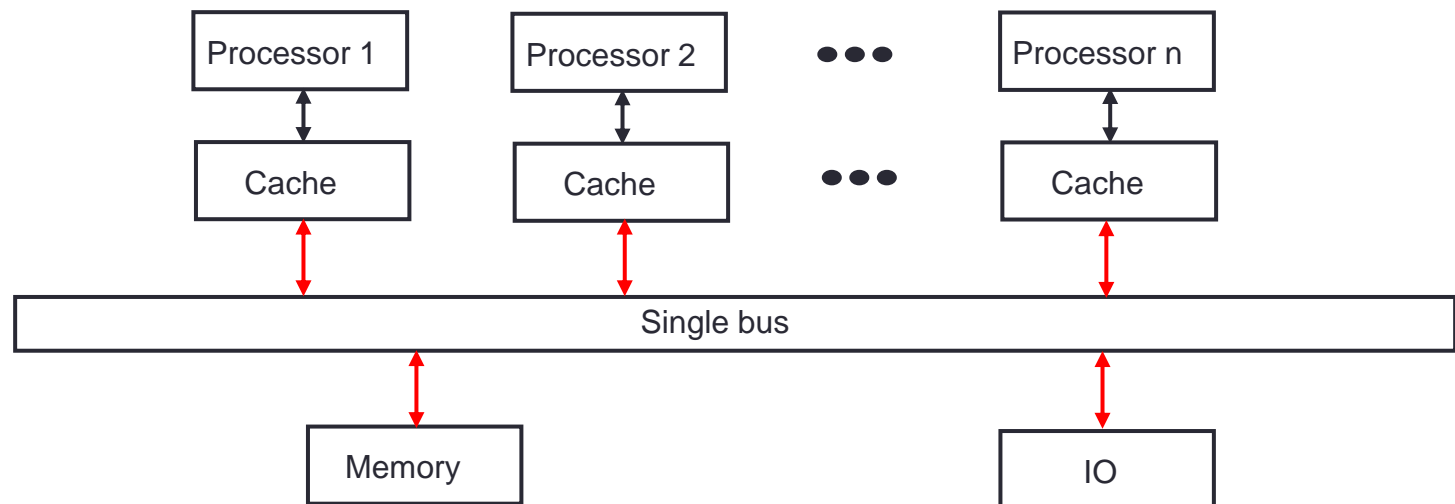
UMA multiprocessor

- **Bus-connected**

- Each time only one processor can use the bus

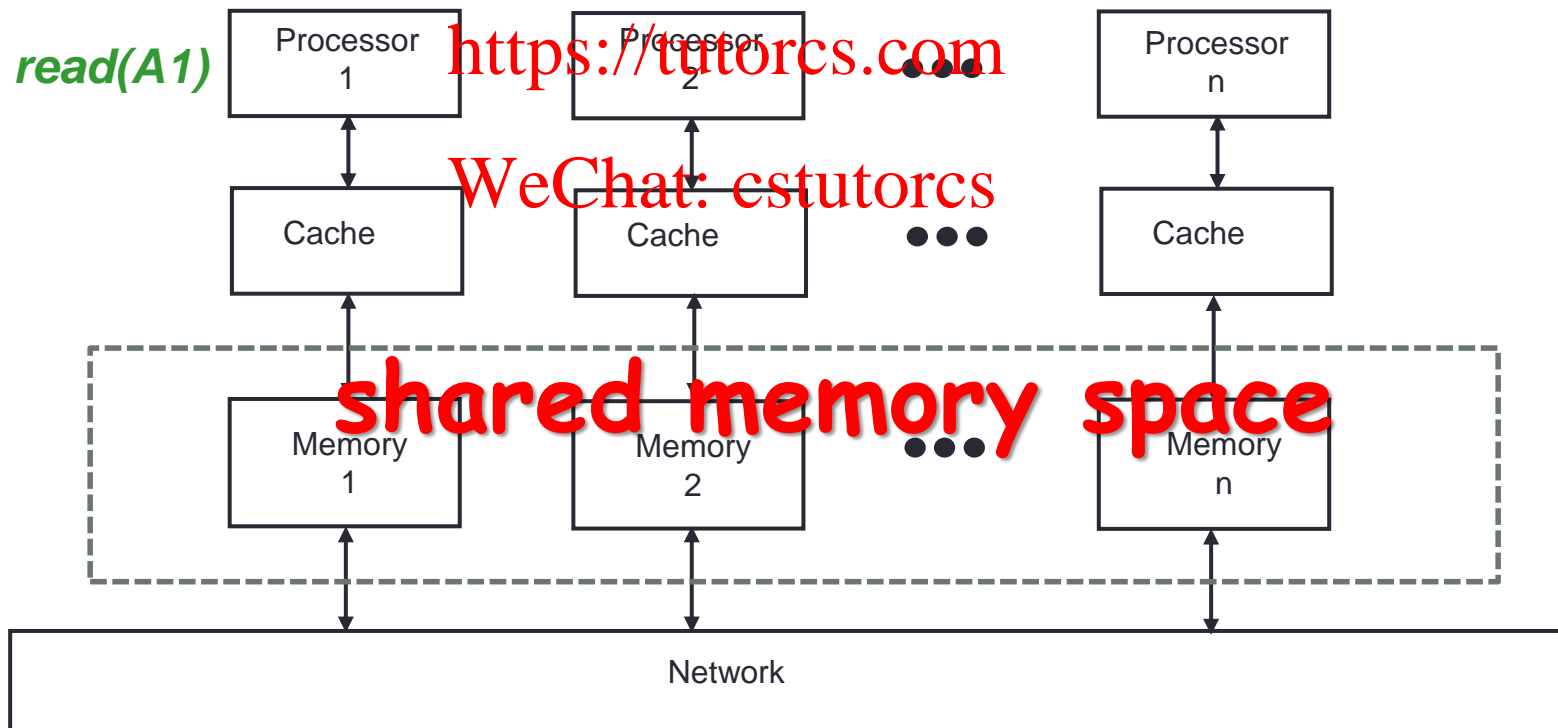
- **One shared physical memory with a uniform access time.**

- All memory locations have similar latencies
 - Data sharing through memory reads/writes



NUMA multiprocessor

- **Network-connected multiprocessors**
 - Processors have faster access to local memory, slower access to “remote” memories located at other processors



Cache coherence

- **Shared-memory multiprocessors have cache coherence problem**
 - **Multiple copies of the same memory data can exist in different caches simultaneously.**
 - **Update in the caches will cause an inconsistent view of memory**

Assignment Project Exam Help

<https://tutorcs.com>

WeChat: cstutorcs

Design solutions for cache coherence

- **Software oriented**

- **Compiler identifies the data items that may cause cache inconsistency and instructs hardware not to cache them**

- **Can be supported by the bus design**

- **Hardware oriented**

- **Snoop-based protocol (UMA)**

- **Write-update**
- **Write-invalidate**

- **Directory-based protocol (NUMA)**

MESI protocol

- **Snoop-based write-invalid protocol**
 - **Snoop:** Every time a data item is transmitted over the bus, it is broadcasted to every processor
- **Using four states for a cache line**
 - **Modified**
 - The line in the cache has been modified and is available only in this cache
 - **Exclusive**
 - The line in the cache is the same as that in main memory and is not present in any other cache
 - **Shared**
 - The line in the cache is the same as that in main memory and also presents in other caches
 - **Invalid**
 - The line in the cache does not contain valid data

How MESI works?

- **Let us consider it under four scenarios**
 - **Read miss**
 - **Read hit** [Assignment Project Exam Help](#)
 - **Write miss**
 - **Write hit** <https://tutorcs.com>
[WeChat: cstutorcs](#)

How MESI works?

read miss

- **Read miss**

- The processor inserts a signal on the bus that alerts all other processors' cache units to snoop the transaction

- **Case 1: If another cache exclusively has a clean copy of the line**

- The cache returns a signal indicating that it shares this line; then transitions the state of its copy from exclusive to shared
- The initiating processor reads the line from main memory and changes the line in its cache from invalid to shared

- **Case 2: If more than one cache has a clean copy of the line**

- Each of them signals that it shares the line.
- The initiating processor reads the line and changes the line in its cache from invalid to shared.

How MESI works?

read miss

- **Read miss (cont.)**

- The processor inserts a signal on the bus that alerts all other processors' cache units to snoop the transaction

- **Case 3: If another cache has a modified copy of the line**

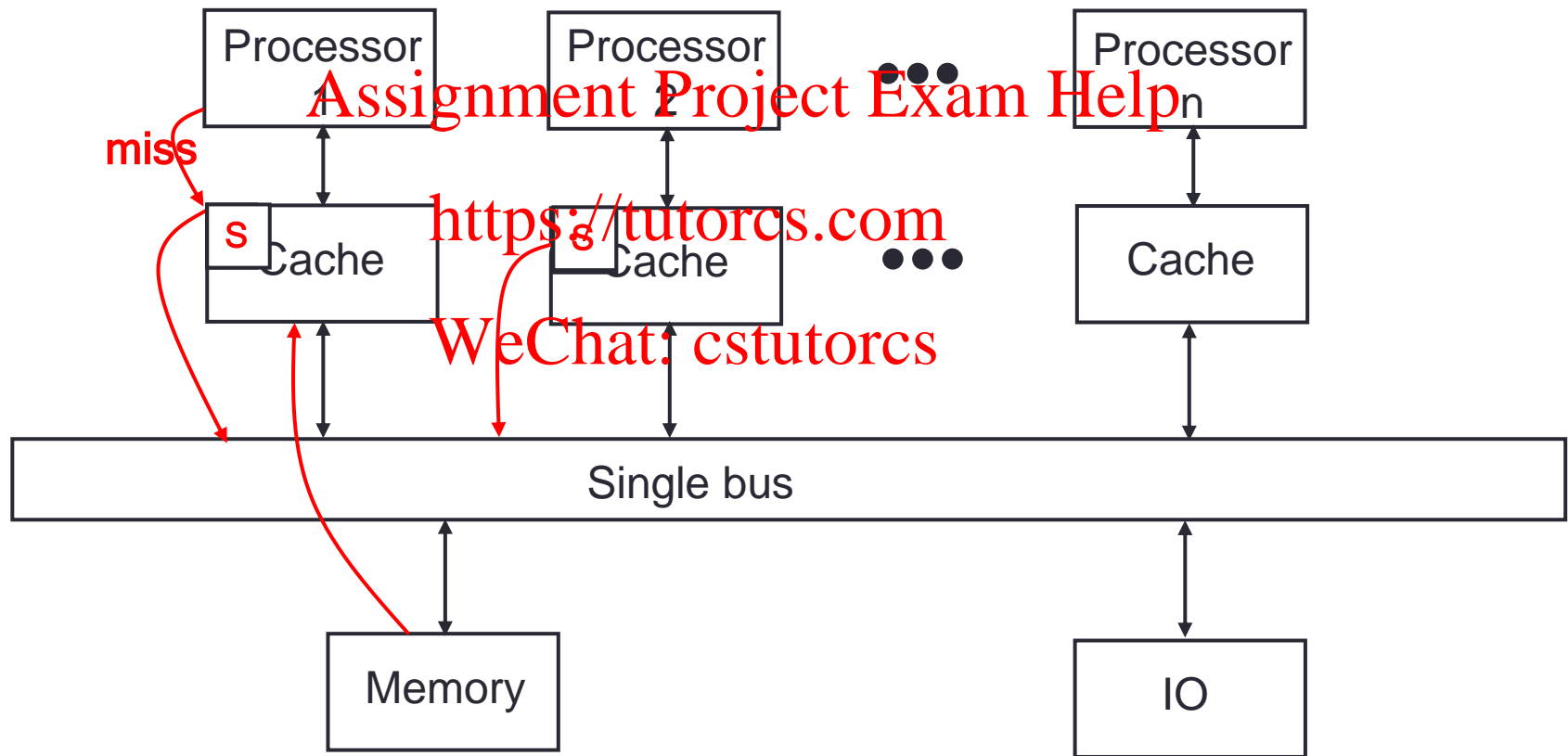
- The cache signals the initiating processor that another processor has a modified copy of the line.
- The initiating processor surrenders the bus and waits.
- The other processor gains access to the bus, writes the modified cache line back to main memory, and changes the state of the cache line to shared
- The corresponding cache reads the line from main memory then changes its line from invalid to shared.

- **See some demonstrations in the next slides**

Read miss - Case 1:

read miss

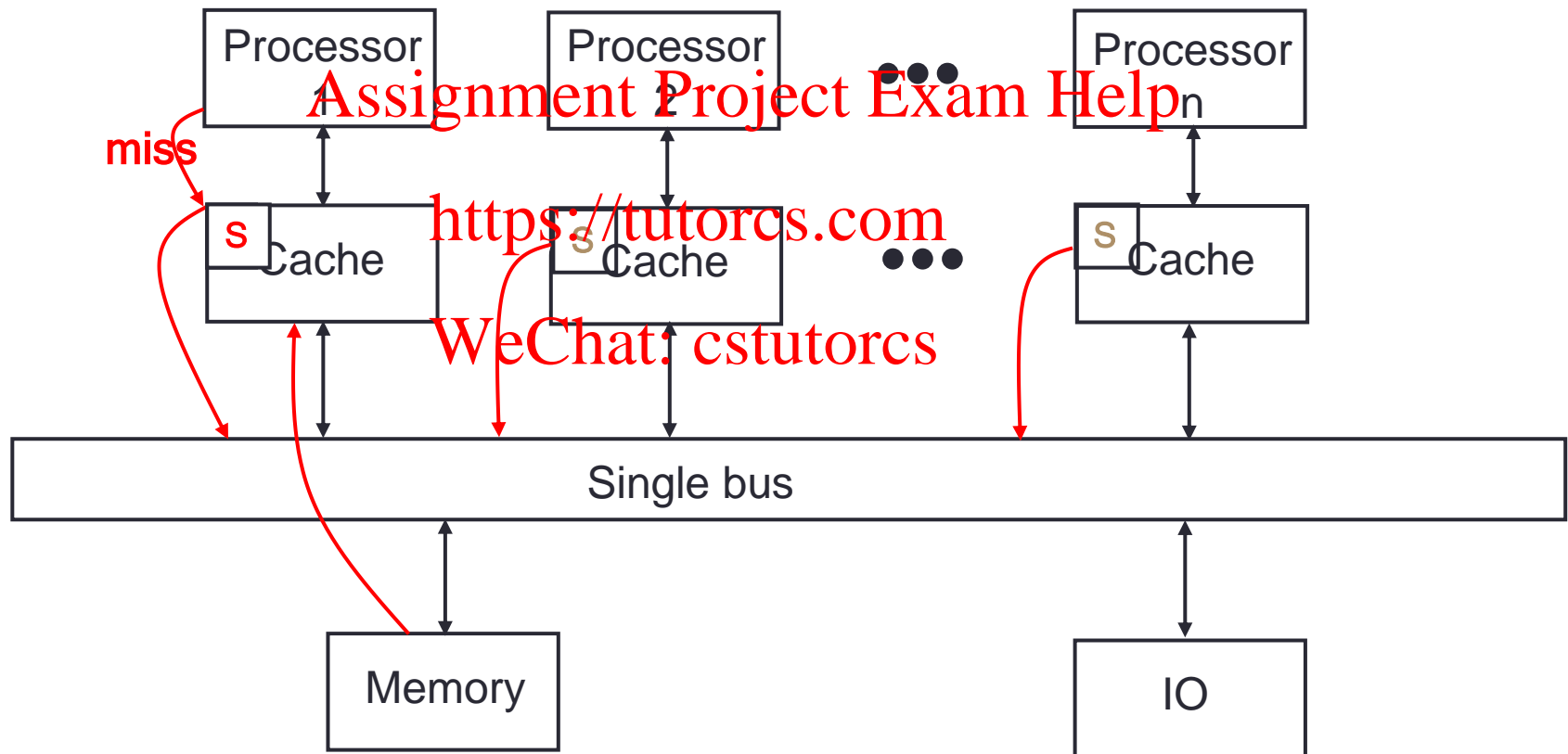
- Processor 1 wants to access data A that has been cached only in processor 2



Read miss - Case 2:

read miss

- Processor 1 wants to access data A that has been cached by more than one processor



Read miss - Case 3:

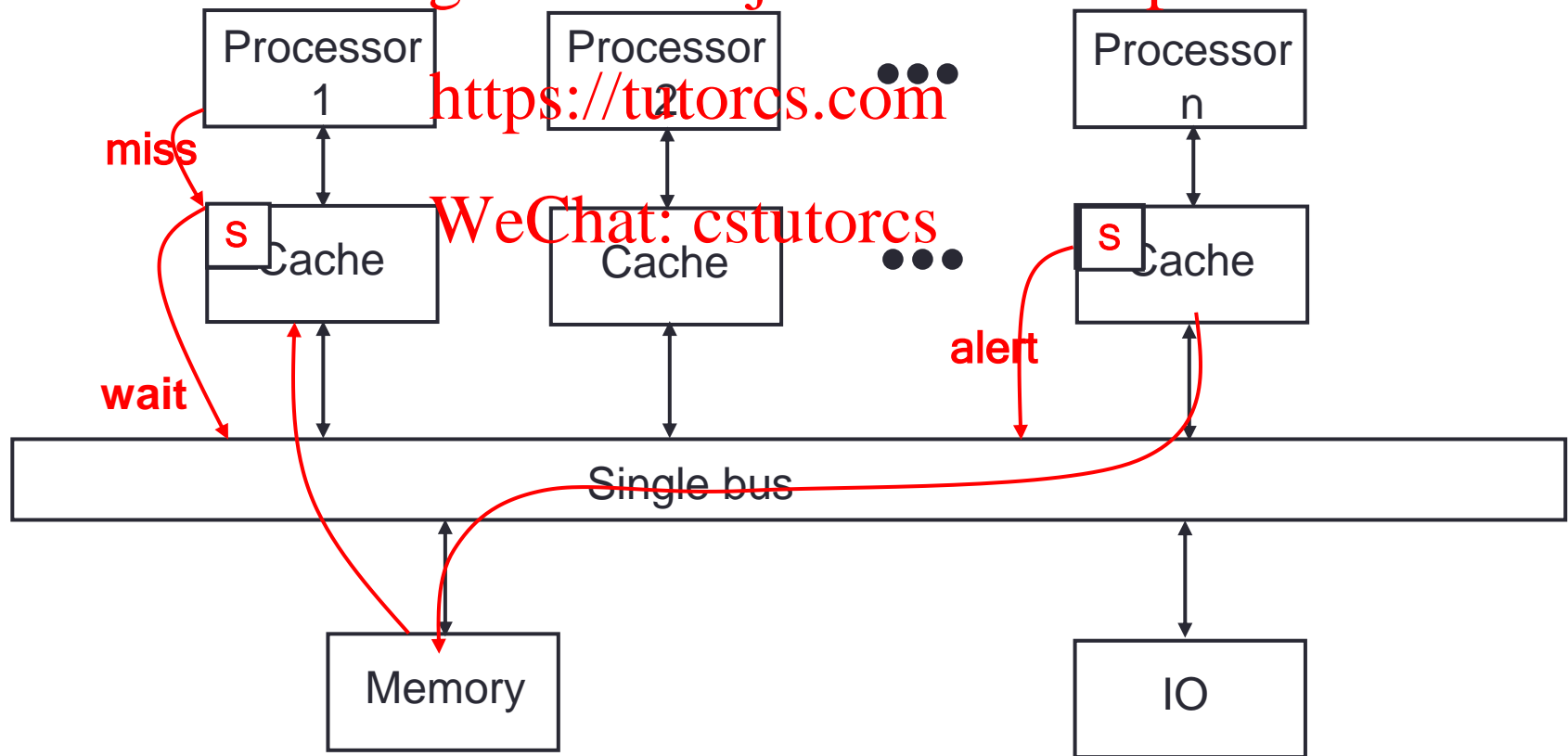
read miss

- Processor 1 wants to access data A that has been cached by one processor and also modified.

Assignment Project Exam Help

<https://tutorcs.com>

WeChat: cstutorcs



How MESI works?

read hit

- **Read hit**

- **The CPU simply reads the required data.**

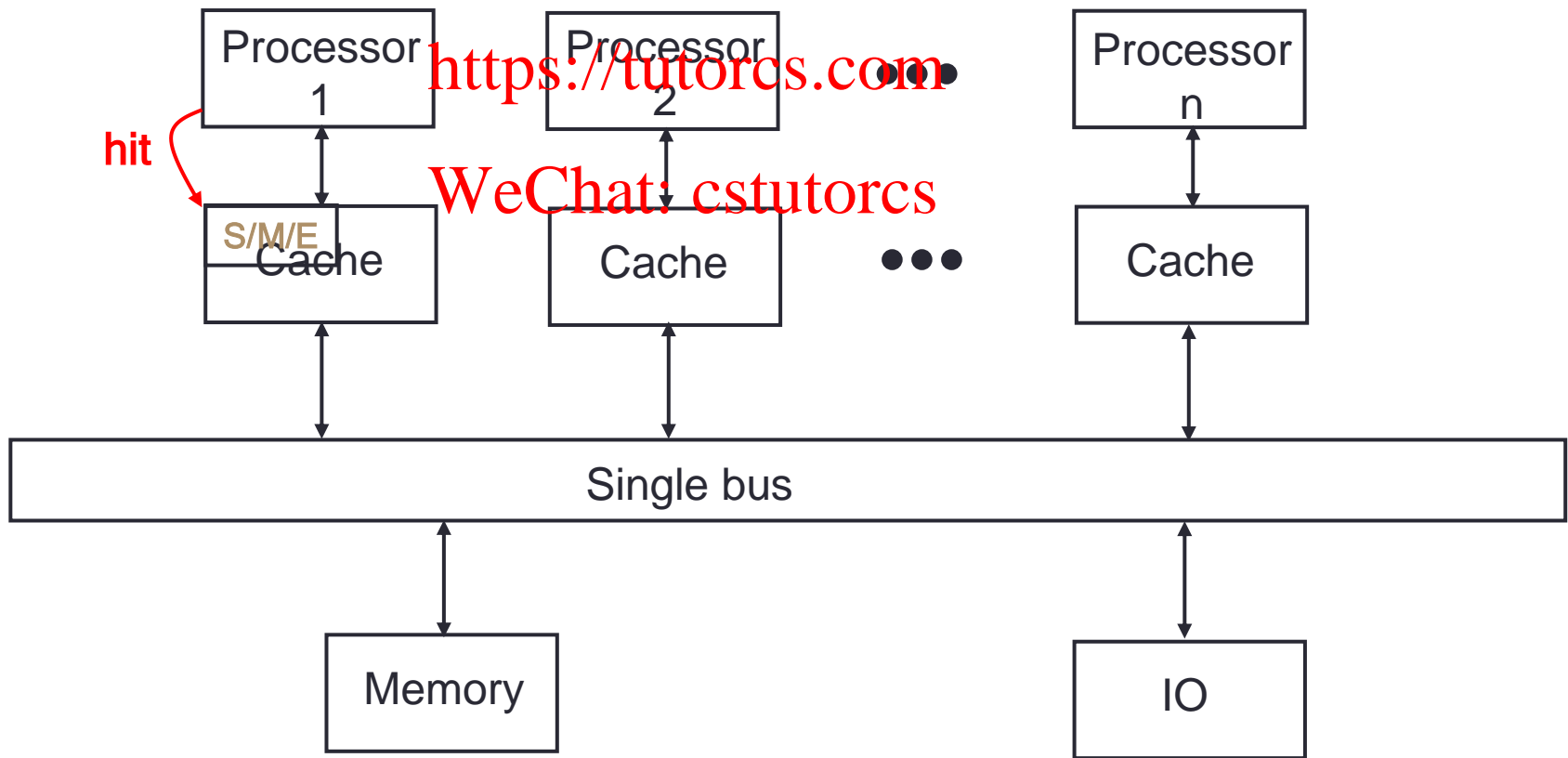
- **There is no state change**

- remaining modified, or shared, or exclusive

Assignment Project Exam Help

<https://tutorcs.com>

WeChat: cstutorcs



How MESI works?

write miss

- **Write miss**

- The processor initiates a memory read to read the line of main memory containing the missing data, then issues a signal on the bus that means **read-with-intent-to-modify (RWITM)**. When the line is loaded, it is immediately marked modified.

- **Case1: some other cache has a modified copy of this line**

- The cache signals the initiating processor that another processor has a modified copy of the line.
- The initiating processor surrenders the bus and waits.
- The other processor gains access to the bus, writes the modified cache line back to main memory, and changes the state of the cache line to invalid
- The initiating processor will again issue a signal to the bus of RWITM and then read the line from main memory, modify the line in the cache, and mark the line as modified.

How MESI works?

write miss

- **Write miss (cont.)**

- The processor initiates a memory read to read the line of main memory containing the missing data, then issues a signal on the bus that means read-with-intent-to-modify (RWITM). When the line is loaded, it is immediately marked modified.
- **Case2: No other cache has a modified copy of the requested line.**
 - If one or more caches have a clean copy of the line, each cache invalids its copy of the line. No signal is returned. The initiating processor reads in the line and modifies it.

Write miss - Case 1:

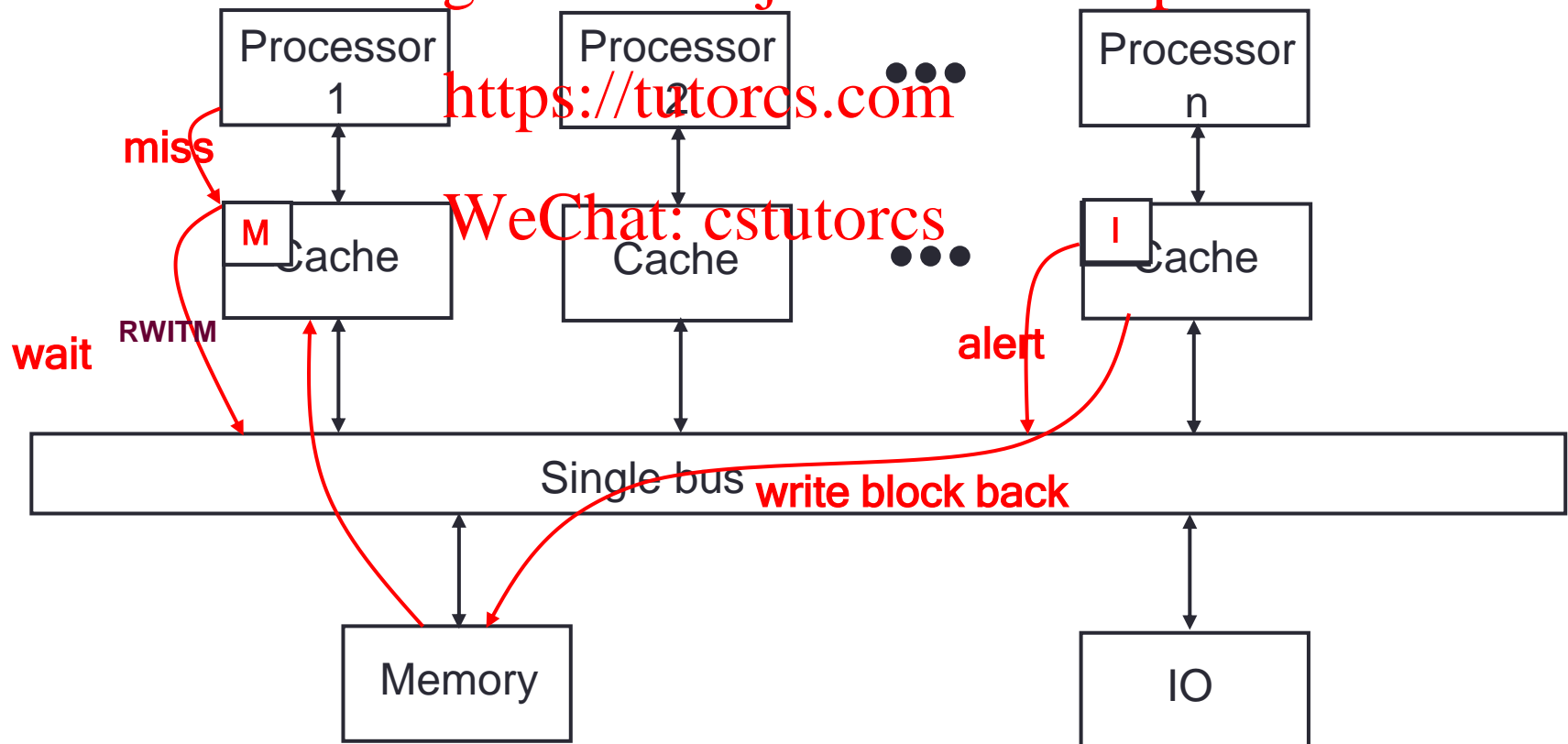
write miss

- Processor 1 wants to write data A that has been cached by processor n and also modified

Assignment Project Exam Help

<https://tutorcs.com>

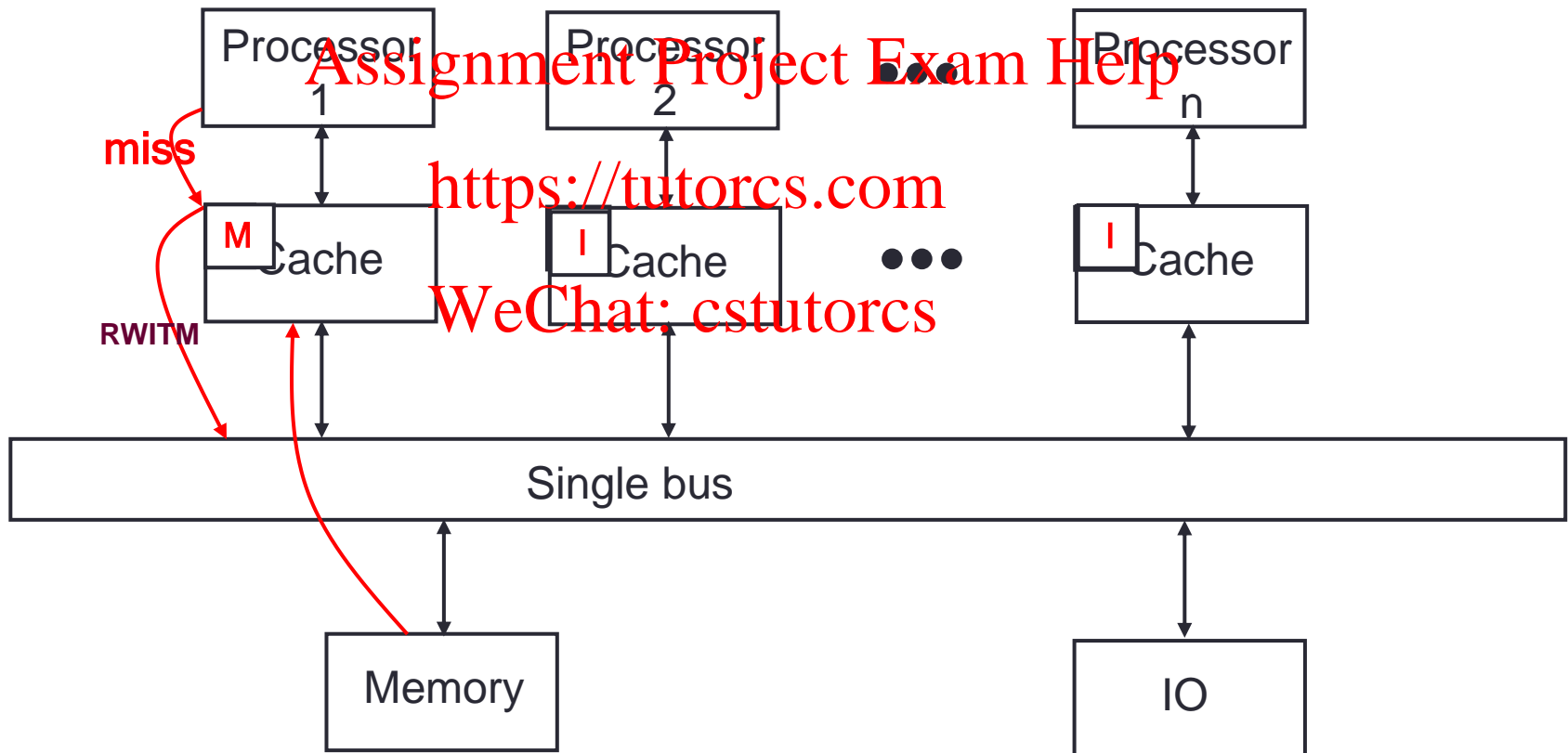
WeChat: cstutorcs



Write miss - Case 2:

write miss

- Processor 1 wants to write data A that has been cached by some other caches.



How MESI works?

write hit

- **Write hit**

- **Based on the current state of the local cache**

- **Case 1 (Shared):**

- The processor signals its write intent on the bus.
- Each processor that has a shared copy of the line in its cache changes the state from shared to invalid.
- The initiating processor performs the update and changes its copy of the line from shared to modified.

- **Case 2 (Exclusive):**

- The processor performs the update and changes its copy of the line from exclusive to modified.

- **Case 3 (Modified):**

- The processor simply performs the update.

Assignment Project Exam Help

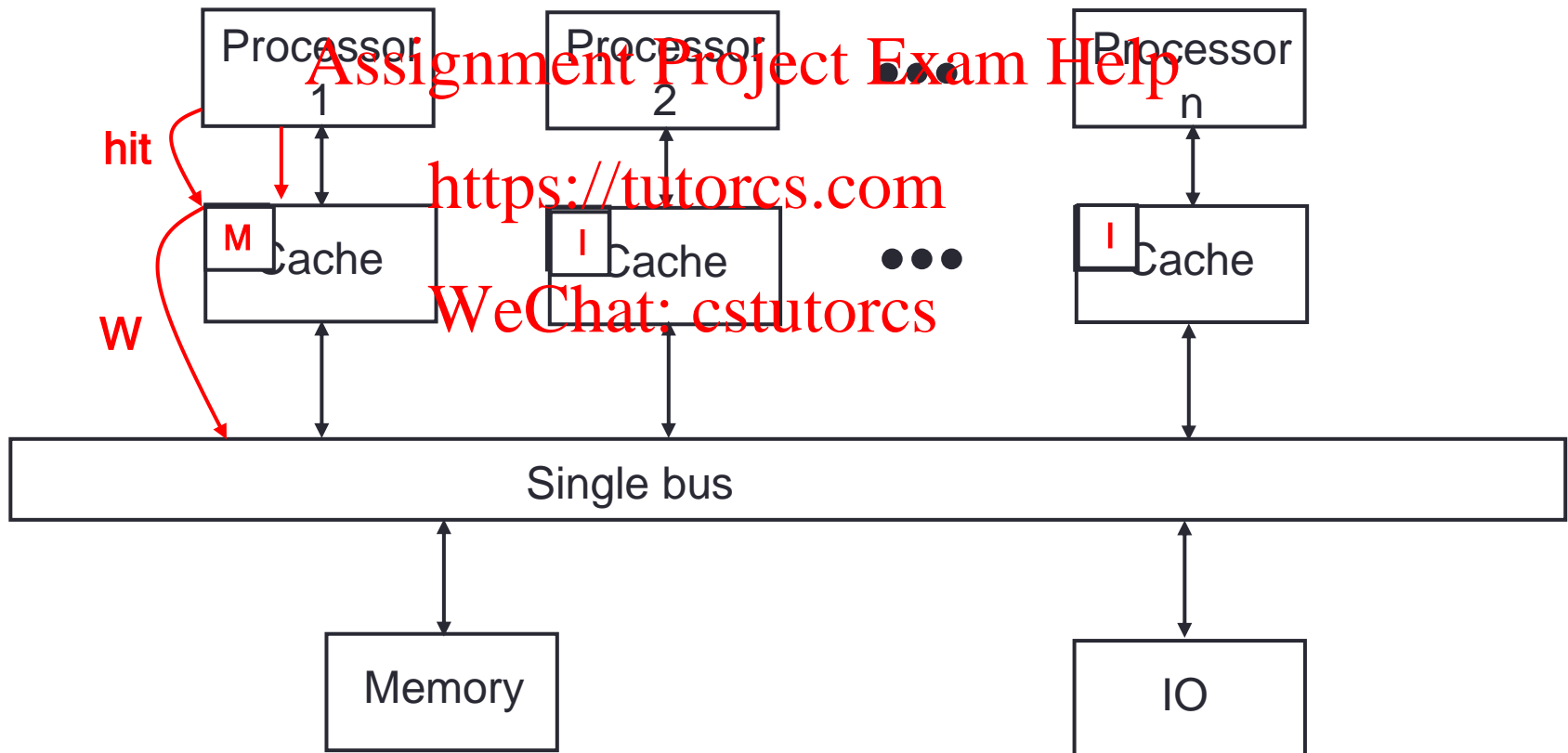
<https://tutorcs.com>

WeChat: cstutorcs

Write hit - Case 1:

write hit

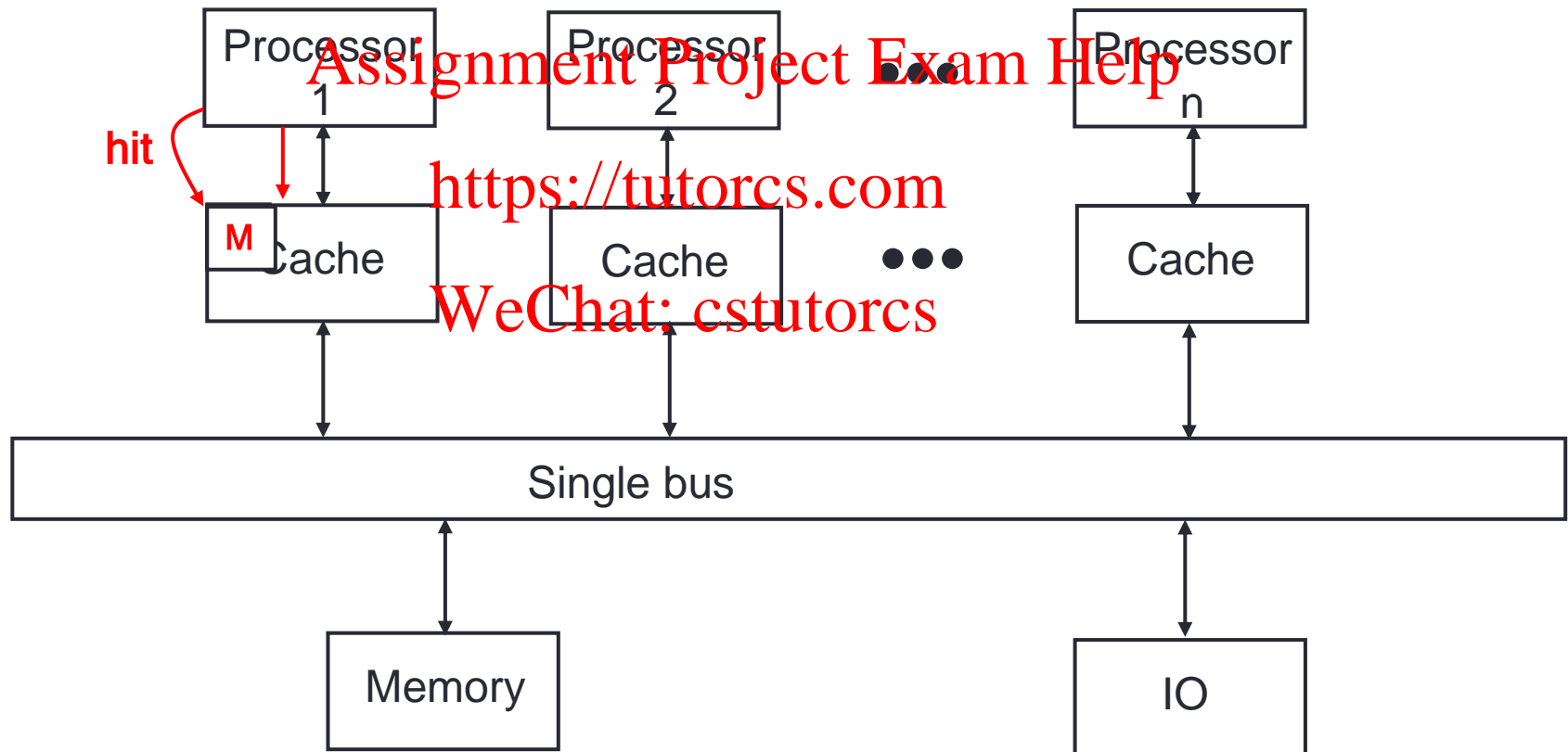
- Processor 1 wants to write data A that has been cached by some other caches.



Write hit - Case 2:

write hit

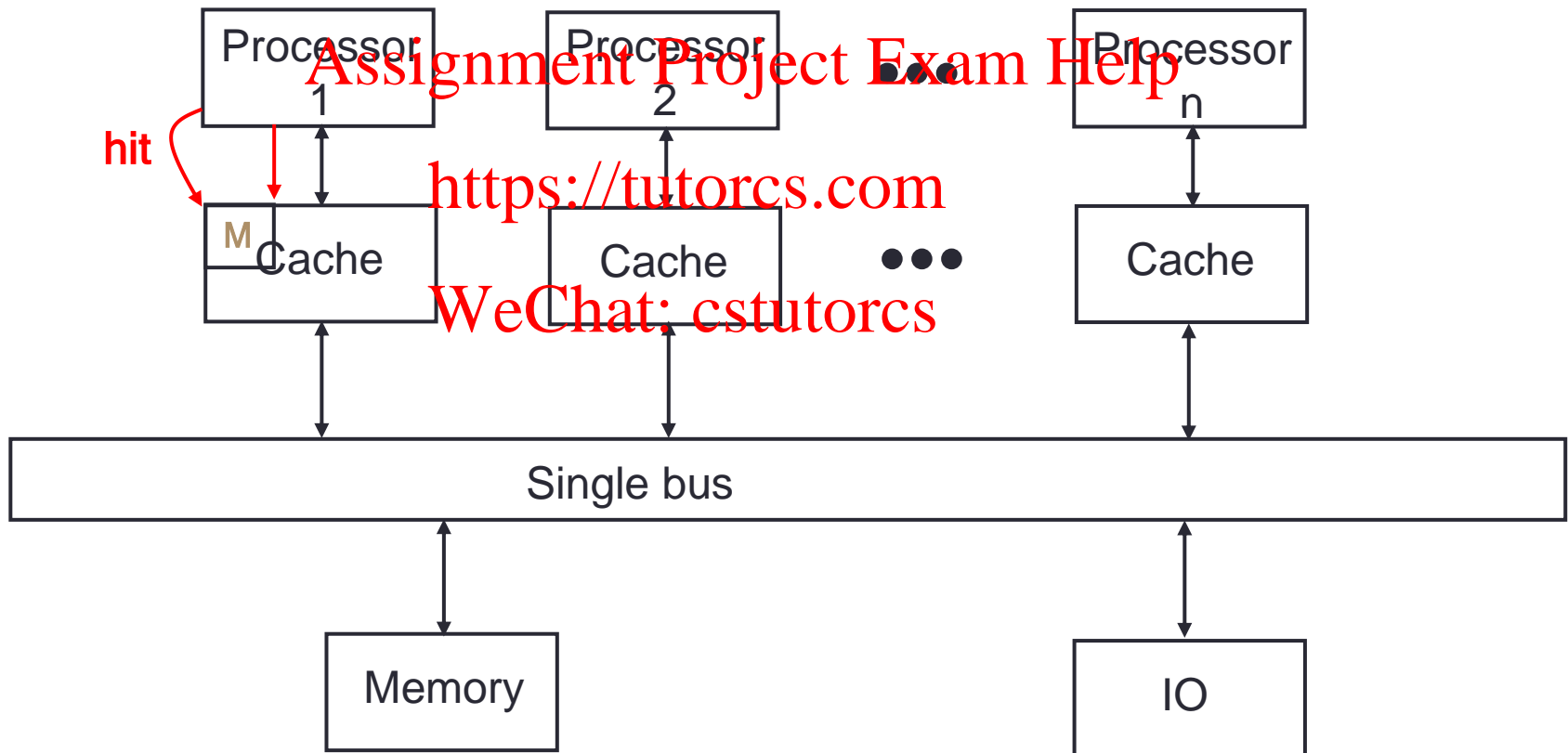
- Processor 1 wants to write data A that has exclusively been cached.



Write hit - Case 3:

write hit

- Processor 1 wants to write data A that has locally been cached and also modified.



Issues with snooping protocol

- **Limited scalability**
 - Due to the bottleneck formed by the bus and shared memory
- **Inefficient distributed approach for tracking the state of the caches**
 - Increasing communication traffic for tracking the state of the caches
 - Asking every one for a cache state of a memory block

Assignment Project Exam Help

<https://tutorcs.com>

WeChat: cstutorcs

Directory based protocol*

- Does not rely on broadcast, instead uses point-to-point messages sent between the processors and memories to keep caches consistent
- Uses a directory to keep tracking of the status of memory blocks
 - The directory holds the information of which caches have a copy of a memory block
 - The participating processors query the directory to retrieve the information about a memory block and then communicate only with the related caches.

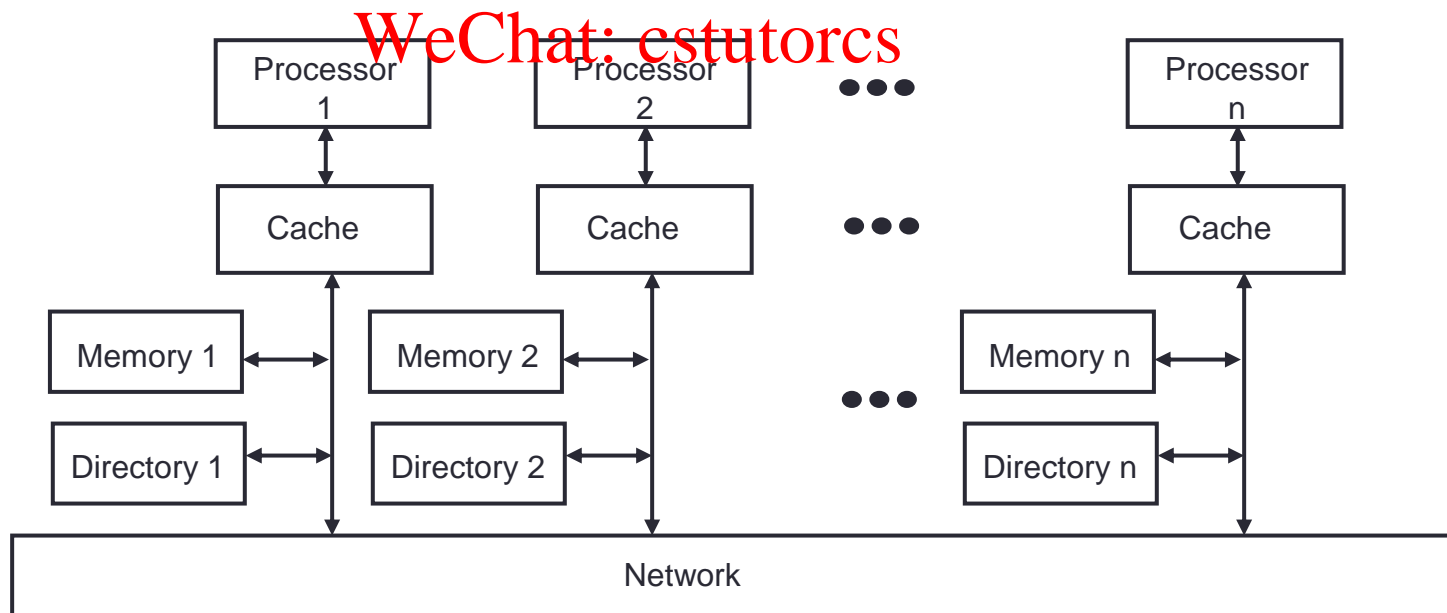
Assignment Project Exam Help

<https://tutorcs.com>

WeChat: cstutorcs

Directory based protocol (cont.)*

- Each processor has its own directory
- An entry in a directory consists of
 - A dirty bit
 - A presence vector
 - 1 bit for each processor, telling which processors have cached copies



Directory-based protocol (cont.)*

- **All misses sent to the block's home directory**
- **The home directory responds the request by**
 - **Finding the cache that holds the latest copy of the block**
 - **Obtaining the copy from the cache**
 - **Sending the copy to the requesting processor**
- **Coherence actions are performed on the related directories**

Directory-based protocol – example*

- Assume the following directory at processor 3 and total 8 processors

Directory	Address	Dirty	Presence
P3	5004	0	1 0 0 0 1 0 0 0
	5008	1	1 0 0 0 0 0 0 0
	5012	0	0 0 0 0 0 1 1 1

processors 1,5 have data

- If P2 requests memory data at address 5008
 - P2 sends the request to P3
 - P3 checks the directory and sees P1 has a modified copy and requests the data from P1 for P2
 - P3 gets data back, updates directory, sends data to P2
 - Dirty: 0 Presence: 1 1 0 0 0 0 0 0