

Multi-threaded programming

An Introduction to concurrency in JAVA

CS2810, January - May 2015

Department of Computer Science and Engineering
Indian Institute of Technology Madras

Evolution of Computers

Evolution of Computers



Figure: The 1990s

Evolution of Computers



Figure: The 1990s



Figure: Improving Clock Speed

Multi-core processors

Multi-core processors

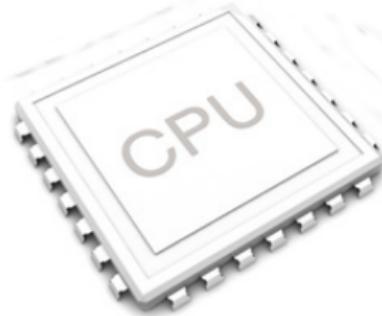


Figure: Single Core at 12 GHz?

Multi-core processors

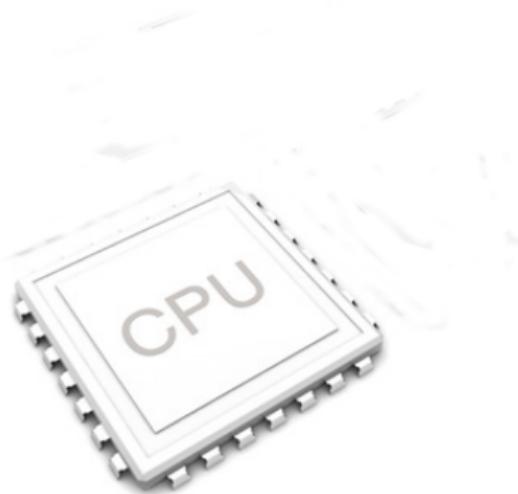


Figure: Single Core at 12 GHz?

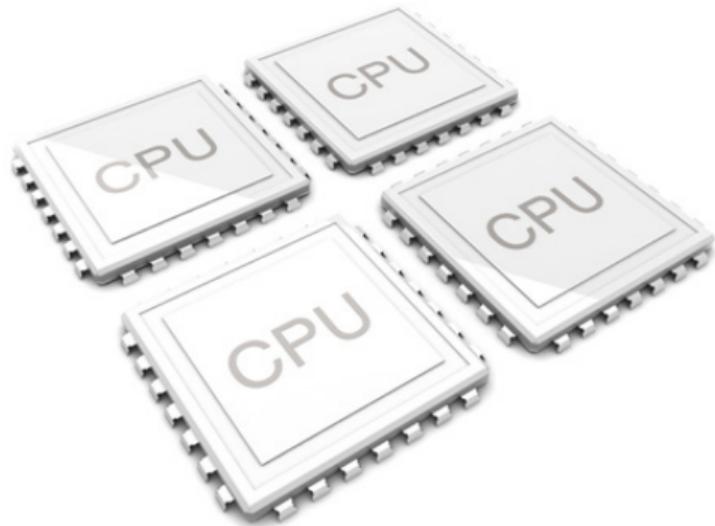


Figure: Multiple Cores: 4 X 3GHz

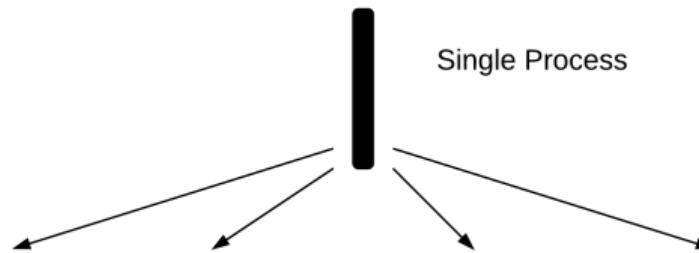
Splitting the process

Single Process

```
for (int i = 25; i < 50; i++)  
    A[i]++;
```

```
for (int i = 75; i < 100; i++)  
    A[i]++;
```

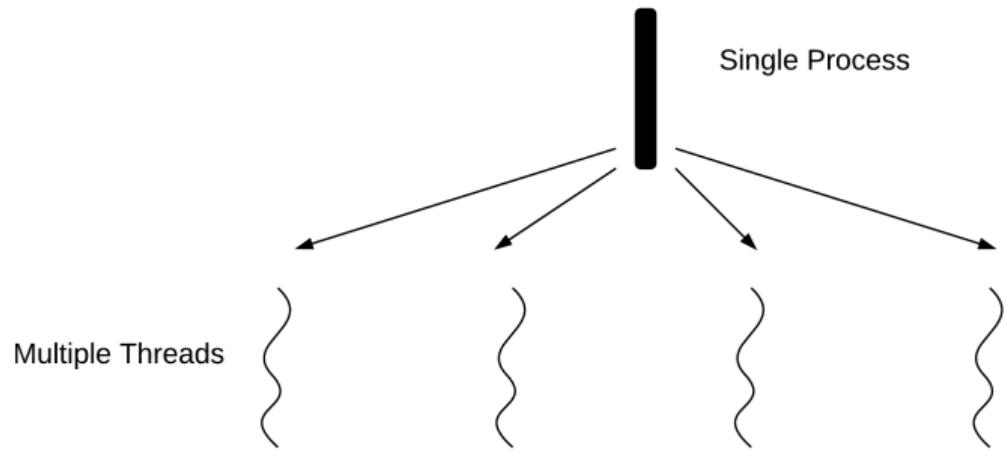
Splitting the process



```
for (int i = 25; i < 50; i++)  
    A[i] ++;
```

```
for (int i = 75; i < 100; i++)  
    A[i] ++;
```

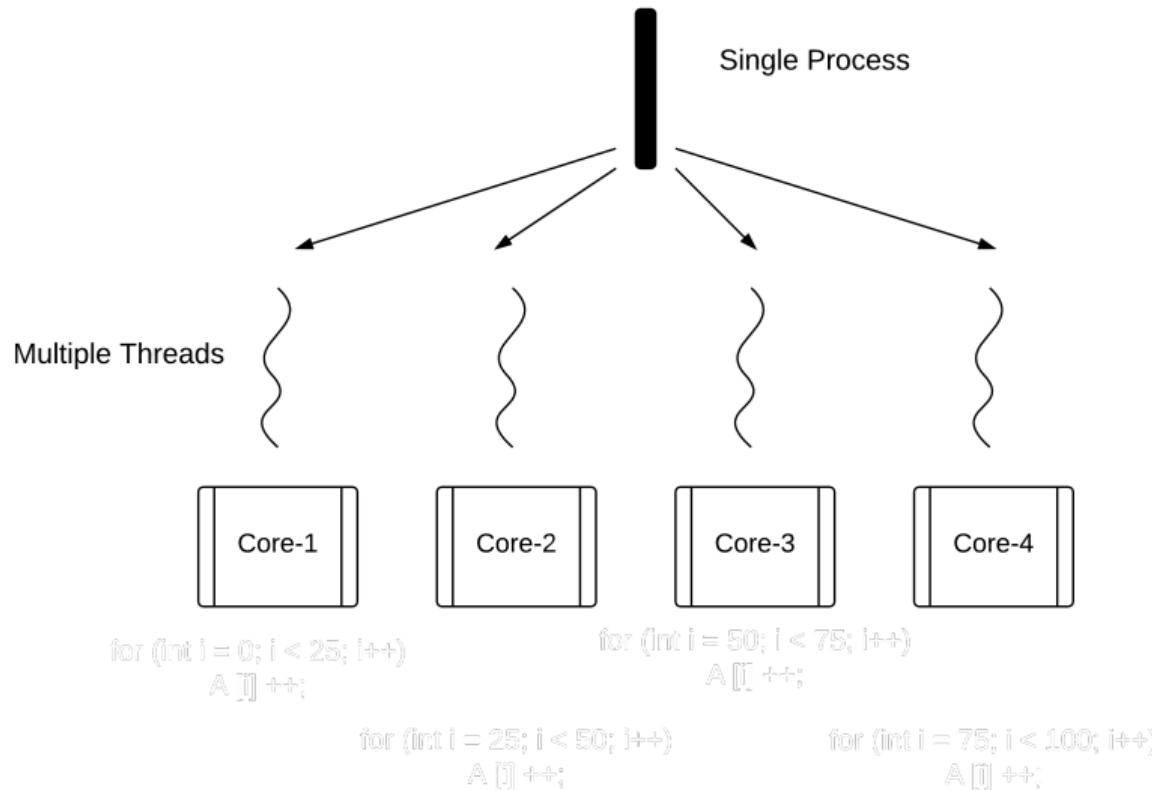
Splitting the process



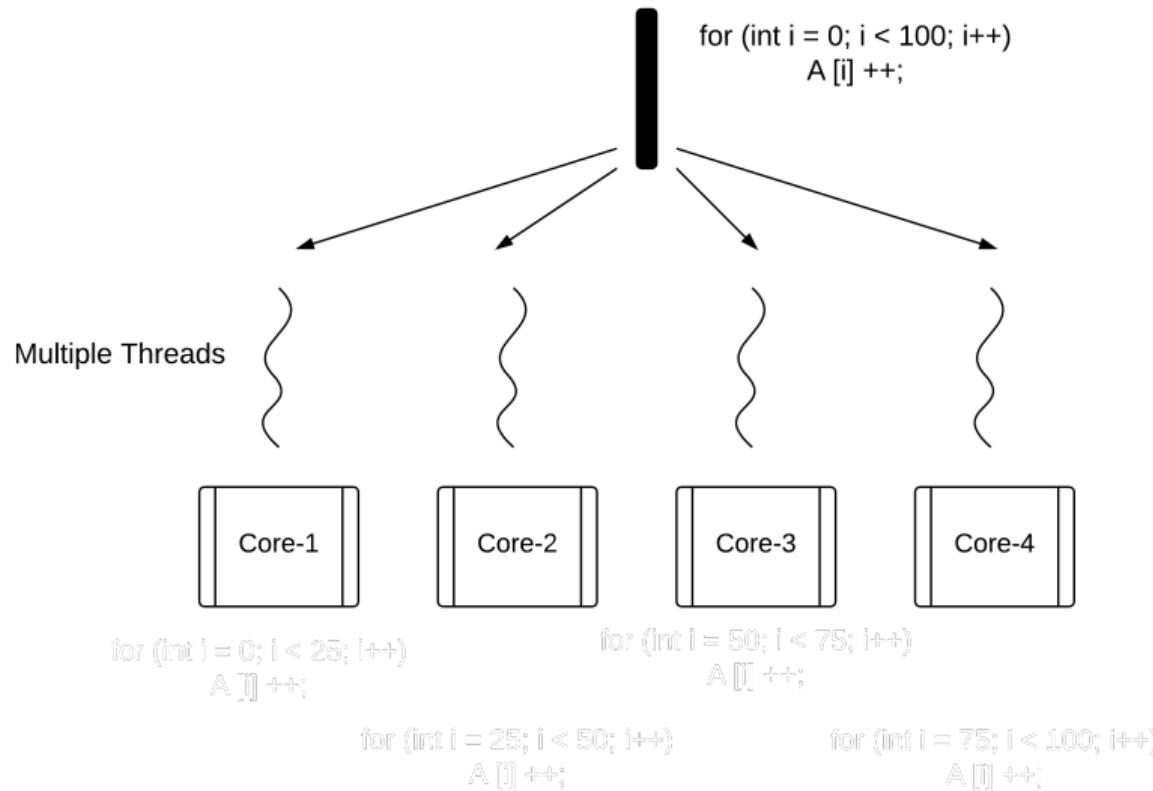
```
for (int i = 25; i < 50; i++)  
    A[i]++;
```

```
for (int i = 75; i < 100; i++)  
    A[i]++;
```

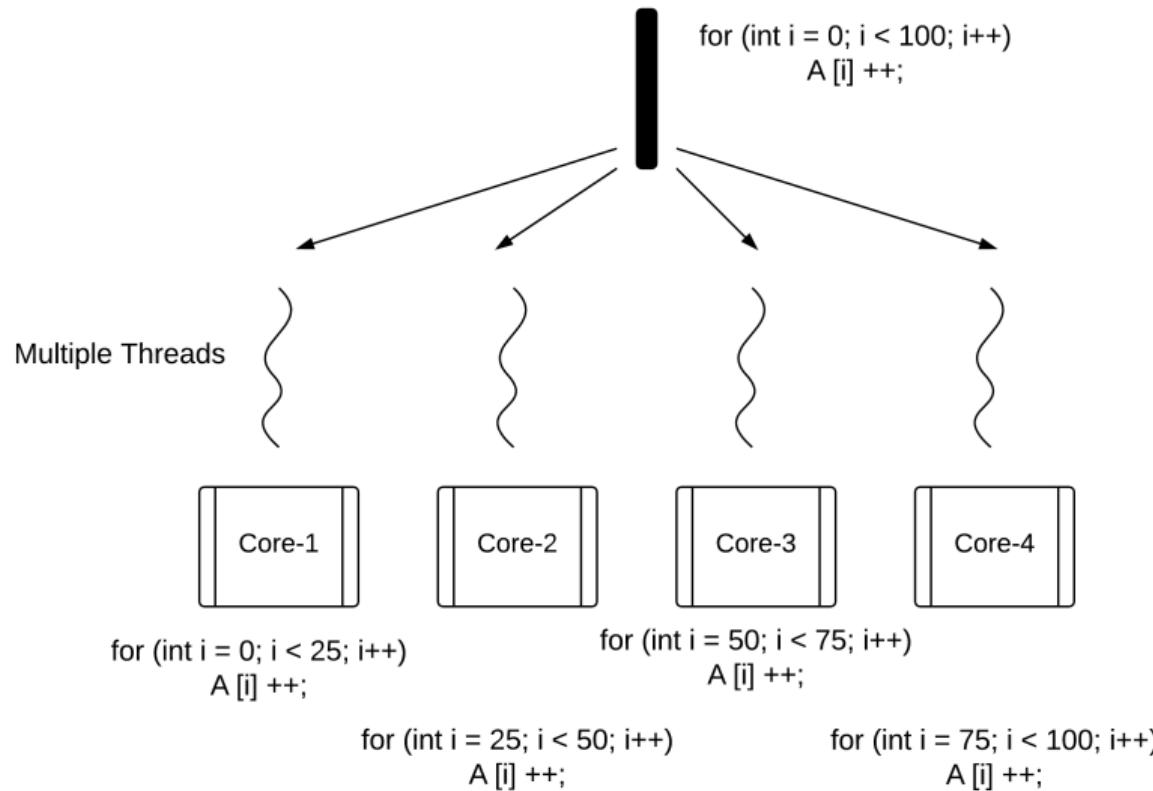
Splitting the process



Splitting the process



Splitting the process



Motivation 1

Motivation 1

- We want to make best use of the multiple cores in our modern processors.

A Stronger Motivation

The notion of blocking

The notion of blocking

- Every line of code has to wait till everything before it is executed.

The notion of blocking

- Every line of code has to wait till everything before it is executed.
- Are there any issues with this?

The notion of blocking

- Every line of code has to wait till everything before it is executed.
- Are there any issues with this?
 - Eg. VLC Player



The notion of blocking

- Every line of code has to wait till everything before it is executed.
- Are there any issues with this?

- Eg. VLC Player
- Video is playing



The notion of blocking

- Every line of code has to wait till everything before it is executed.
- Are there any issues with this?



- Eg. VLC Player
- Video is playing
- Must *still* be able to search playlist

The notion of blocking

- Every line of code has to wait till everything before it is executed.
- Are there any issues with this?

- Eg. VLC Player
- Video is playing <- Thread 1
- Must *still* be able to search playlist <- Thread 2



The notion of blocking

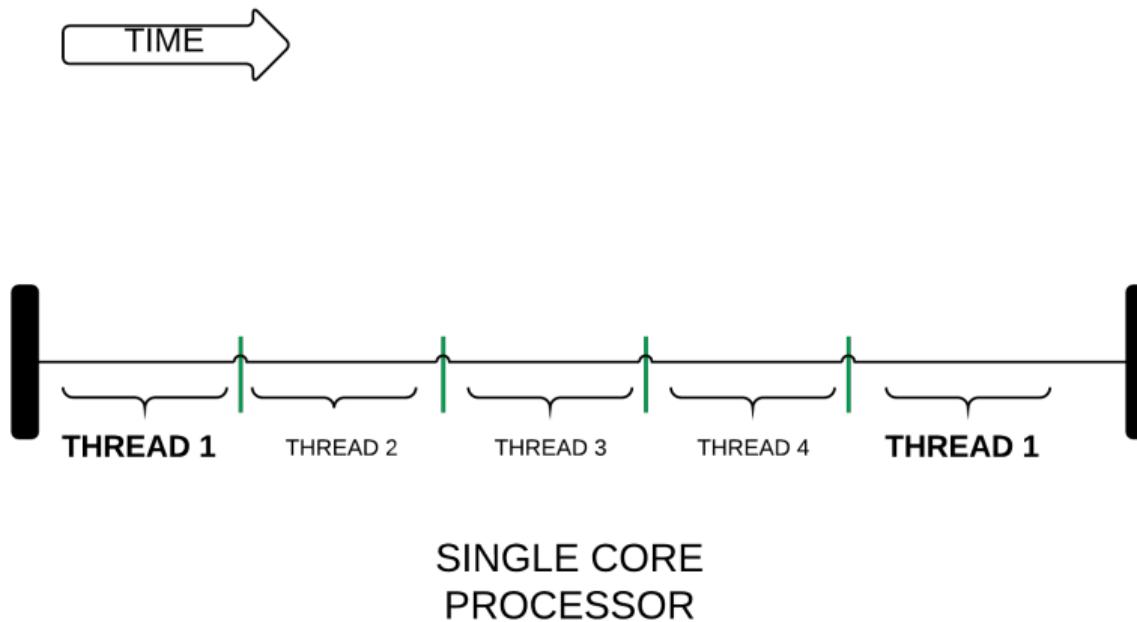
- Every line of code has to wait till everything before it is executed.
- Are there any issues with this?



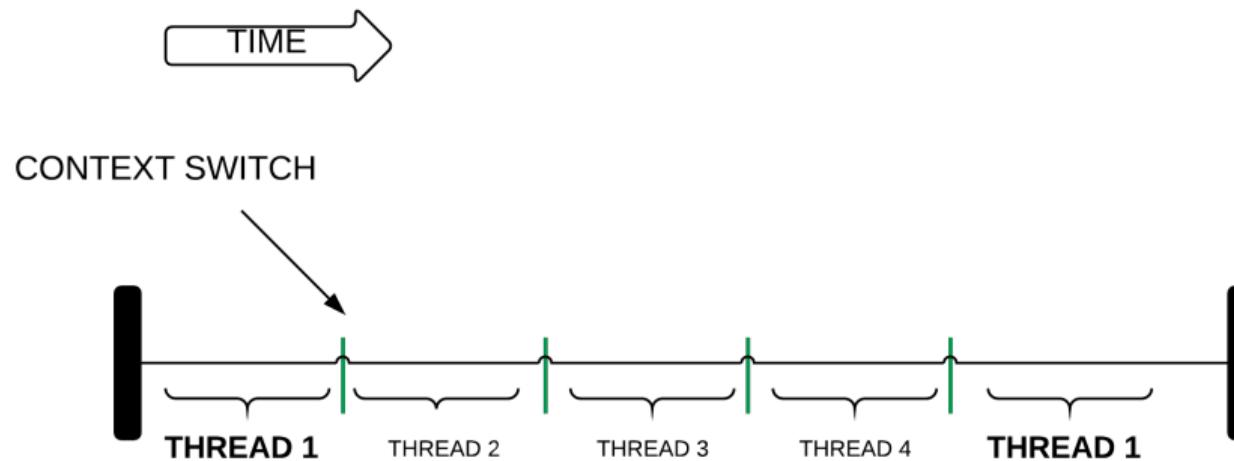
- Eg. VLC Player
- Video is playing <- Thread 1
- Must *still* be able to search playlist <- Thread 2
- This must work even on single-core machines.

Time Slicing

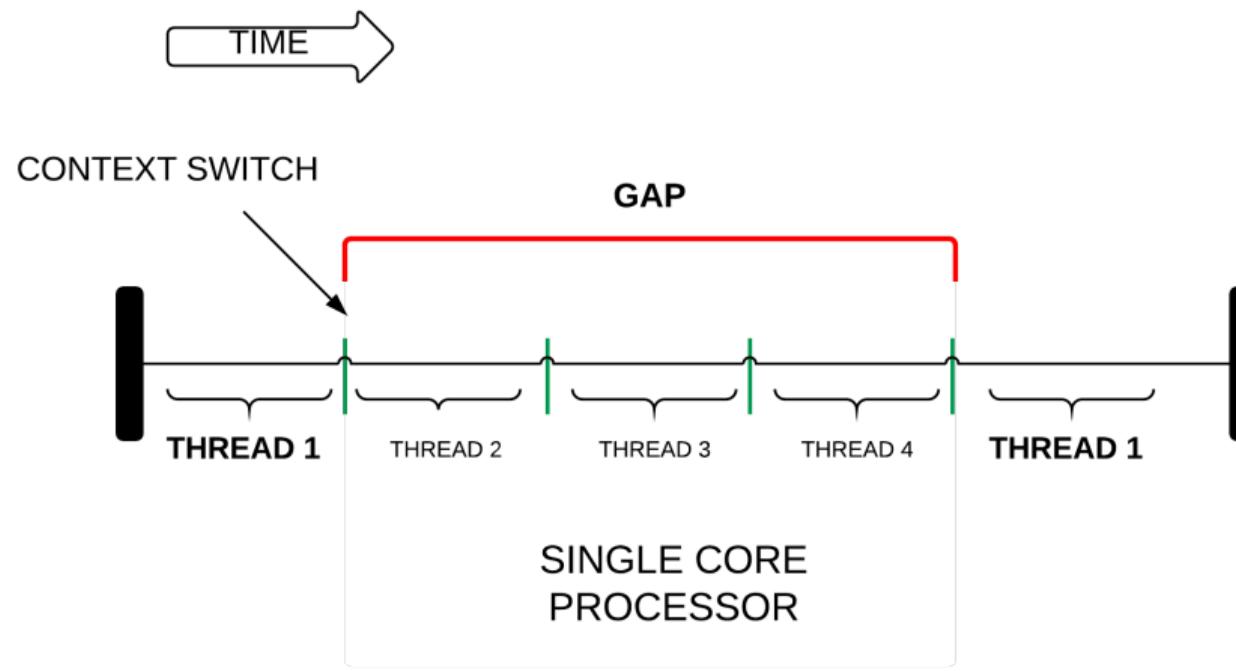
Time Slicing



Time Slicing



Time Slicing



Motivation 1

- We want to make best use of the multiple cores in our modern processors.

Motivation 2

Motivation 1

- We want to make best use of the multiple cores in our modern processors.

Motivation 2

- We need programs to be non-blocking and concurrent.

Defining a Thread

Thread

Thread

- A thread is a light-weight process.

Thread

- A thread is a light-weight process.
- A process can consist of multiple threads.

Thread

- A thread is a light-weight process.
- A process can consist of multiple threads.
- Threads of the same process share the data segment, but have their own stack, code.

How to create and use threads?

Syntax

How to create and use threads?

Syntax

- Language specific.

Syntax

- Language specific.
- JAVA -java.util.concurrent package

How to create and use threads?

Syntax

- Language specific.
- JAVA -java.util.concurrent package
- To create a thread:

How to create and use threads?

Syntax

- Language specific.
- JAVA -`java.util.concurrent` package
- To create a thread:
 - Implement the Runnable interface

How to create and use threads?

Syntax

- Language specific.
- JAVA -`java.util.concurrent` package
- To create a thread:
 - Implement the Runnable interface
 - Override the `run()` function

How to create and use threads?

Syntax

- Language specific.
- JAVA -java.util.concurrent package
- To create a thread:
 - Implement the Runnable interface
 - Override the run() function
 - Create a Thread object out of this class

Syntax

- Language specific.
- JAVA -java.util.concurrent package
- To create a thread:
 - Implement the Runnable interface
 - Override the run() function
 - Create a Thread object out of this class
 - Start the Thread.

Using Threads

Using Threads

```
class MyRunnable implements Runnable
{
    @Override
    public void run ()
    {
        // Code that the thread should execute
    }
}
```

Using Threads

```
class MyRunnable implements Runnable
{
    // Thread's internal variables
    private int param;

    // Constructor
    public MyRunnable (int param)
    {
        this.param = param;
    }

    @Override
    public void run ()
    {
        // Code that the thread should execute
    }
}
```

Using Threads

```
class MyRunnable implements Runnable
{
    // Thread's internal variables
    private int param;

    // Constructor
    public MyRunnable (int param)
    {
        this.param = param;
    }

    @Override
    public void run ()
    {
        // Code that the thread should execute
    }
}
```

```
public static void main (String [] args)
{
    // Initiating Runnable
    MyRunnable mRunnable = new MyRunnable (10);

    // Constructing the Thread
    Thread t = new Thread (mRunnable);

    // Starting the thread
    t.start ();
}
```

```
public class Counter
{
    private int c = 0;

    public void increment () {
        c++;
    }

    public void decrement () {
        c--;
    }

    public int value () {
        return c;
    }
}
```

```
public class Counter
{
    private int c = 0;

    public void increment () {
        c++;
    }

    public void decrement () {
        c--;
    }

    public int value () {
        return c;
    }
}
```

- Incrementing c is three steps

```
public class Counter
{
    private int c = 0;

    public void increment () {
        c++;
    }

    public void decrement () {
        c--;
    }

    public int value () {
        return c;
    }
}
```

- Incrementing c is three steps
 - MOV AX, [memC]
 - INC AX
 - MOV [memC], AX

```
public class Counter
{
    private int c = 0;

    public void increment () {
        c++;
    }

    public void decrement () {
        c--;
    }

    public int value () {
        return c;
    }
}
```

- Incrementing c is three steps
 - MOV AX, [memC]
 - INC AX
 - MOV [memC], AX
- Decrementing c is also three steps

```
public class Counter
{
    private int c = 0;

    public void increment () {
        c++;
    }

    public void decrement () {
        c--;
    }

    public int value () {
        return c;
    }
}
```

- Incrementing c is three steps
 - MOV AX, [memC]
 - INC AX
 - MOV [memC], AX
- Decrementing c is also three steps
 - MOV AX, [memC]
 - DEC AX
 - MOV [memC], AX

```
public class Counter
{
    private int c = 0;

    public void increment () {
        c++;
    }

    public void decrement () {
        c--;
    }

    public int value () {
        return c;
    }
}
```

- Incrementing c is three steps
 - MOV AX, [memC]
 - INC AX
 - MOV [memC], AX
- Decrementing c is also three steps
 - MOV AX, [memC]
 - DEC AX
 - MOV [memC], AX
- Suppose Thread 1 calls increment() and Thread 2 calls decrement ()

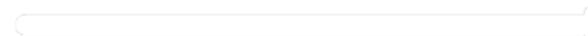
Issue

THREAD 1
Calls increment ()

THREAD 2
Calls decrement ()

MOV AX, [memC]

INC AX



CONTEXT SWITCH

MOV AX, [memC]

DEC AX

MOV [memC], AX



CONTEXT SWITCH

MOV [memC], AX

Issue

THREAD 1

Calls increment ()

THREAD 2

Calls decrement ()

MOV AX, [memC]

INC AX



MOV AX, [memC]

DEC AX

MOV [memC], AX



MOV [memC], AX

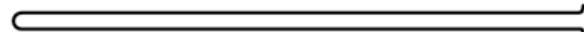
Issue

THREAD 1
Calls increment ()

THREAD 2
Calls decrement ()

MOV AX, [memC]

INC AX



CONTEXT SWITCH

MOV AX, [memC]

DEC AX

MOV [memC], AX



CONTEXT SWITCH

MOV [memC], AX

Now c has the value -1

Issue

THREAD 1
Calls increment ()

THREAD 2
Calls decrement ()

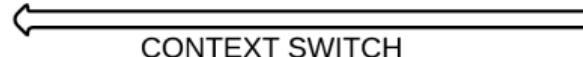
MOV AX, [memC]

INC AX



MOV AX, [memC]

DEC AX



MOV [memC], AX

MOV [memC], AX

Now c has the value 1. But we expected 0

Using Locks

```
public class Counter
{
    private int c = 0;

    private Object lock = new Object();

    public void increment () {
        synchronized (lock) {
            c++;
        }
    }

    public void decrement () {
        synchronized (lock) {
            c--;
        }
    }

    public int value () {
        return c;
    }
}
```

Deadlock

```
public class Counter
{
    private int c = 0;

    private Object lock1 = new Object ();
    private Object lock2 = new Object ();

    public void increment () {
        synchronized (lock1) {
            synchronized (lock2) {
                c++;
            }
        }
    }

    public void decrement () {
        synchronized (lock2) {
            synchronized (lock1) {
                c--;
            }
        }
    }

    public int value () {
        return c;
    }
}
```

Multiplayer games - Dead Reckoning



<https://docs.oracle.com/javase/tutorial/essential/concurrency/>

The Java™ Tutorials

Concurrency

[Processes and Threads](#)

[« Previous](#) • [Trail](#) • [Next »](#)

[Thread Objects](#)

[Defining and Starting
a Thread](#)

[Pausing Execution
with Sleep](#)

[Interrupts
Joins](#)

[The SimpleThreads
Example](#)

[Synchronization](#)

[Thread Interference](#)

[Memory Consistency
Errors](#)

Lesson: Concurrency

Computer users take it for granted that their systems can do more than one thing at a time. They assume that they expect to do more than one thing at a time. For example, that streaming audio application must simultaneously handle keyboard and mouse events, no matter how busy it is reformatting text or updating the display. Software that

The Java platform is designed from the ground up to support concurrent programming, with basic concurrency support introduced by the `java.util.concurrent` package. This lesson introduces the platform's basic concurrency support and summarizes some of the high-level APIs in the `java.util.concurrent` package.

[« Previous](#) • [Trail](#) • [Next »](#)