# Introduction

Companion slides for

The Art of Multiprocessor Programming

by Maurice Herlihy & Nir Shavit

(Abridged version. Original at http://booksite.elsevier.com/9780123705914/?ISBN=9780123705914 )

## Procedure for Thread *i*

```
Counter counter = new Counter(1);

void primePrint {
  long j = 0;
  while (j < 10^10) {
    j = counter.getAndIncrement();
    if (isPrime(j))
      print(j);
  }
}
```

Shared counter object

## Procedure for Thread *i*

```
Counter counter = new Counter(1);

void primePrint {
  long j = 0;
  while (j < 10^10) {
    j = counter.getAndIncrement();
    if (isPrime(j))
      print(j);
  }
}
```

Stop when every value taken

## Procedure for Thread *i*

```
Counter counter = new Counter(1);

void primePrint {
  long j = 0;
  while (j < 10^10) {
    j = counter.getAndIncrement();
    if (isPrime(j))
      print(j);
  }
}
```

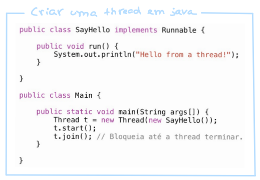
Increment & return each new value

## Counter Implementation

```
public class Counter {
  private long value;

  public long getAndIncrement() {
    return value++;
  }
}
```

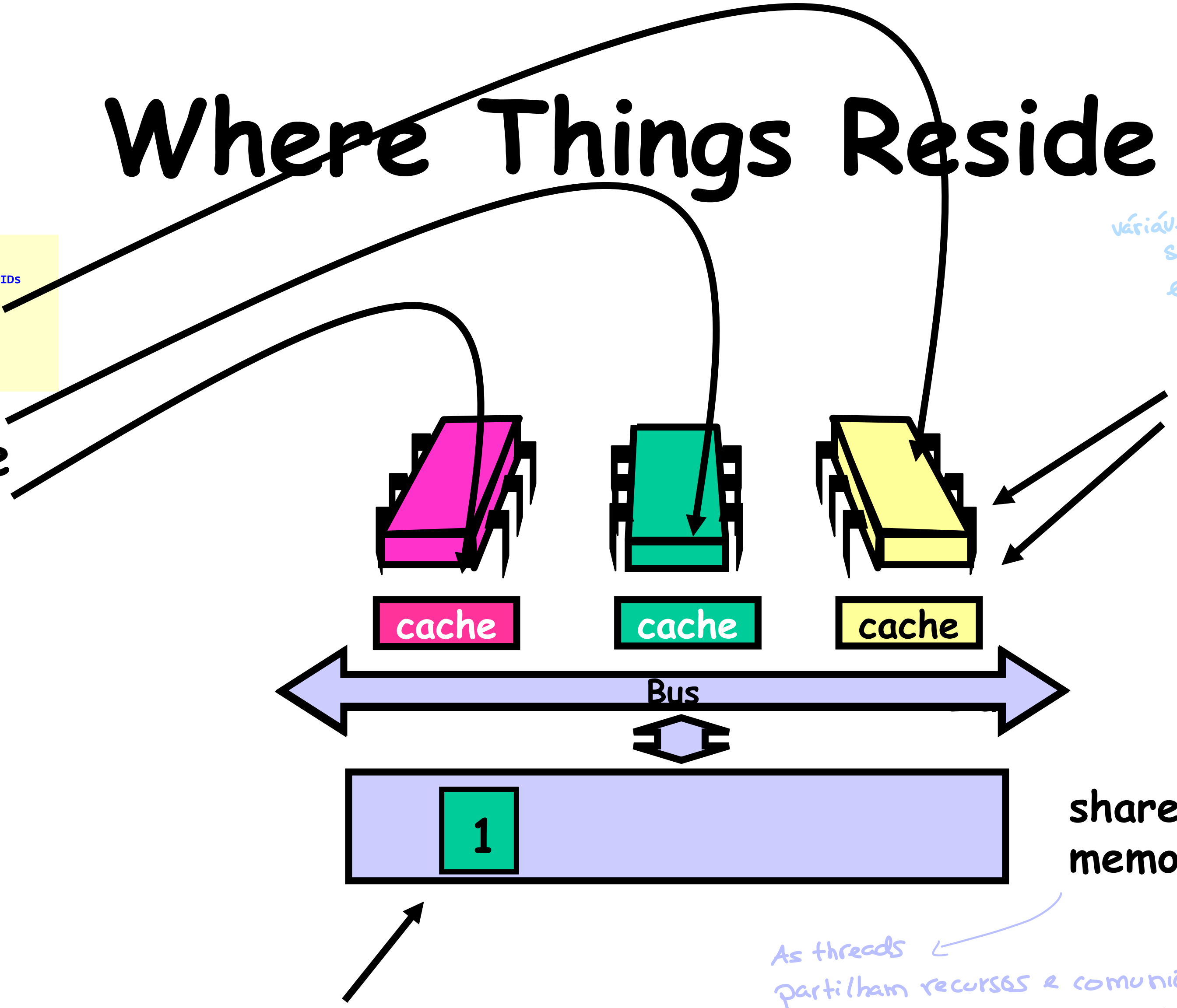
# Where Things Reside

Criar uma thread em java



```
public class SayHello implements Runnable {

    public void run() {
        System.out.println("Hello from a thread!");
    }

}

public class Main {

    public static void main(String args[]) {
        Thread t = new Thread(new SayHello());
        t.start();
        t.join(); // Bloqueia até a thread terminar.
    }

}
```

```
void primePrint {
  int i =
ThreadID.get(); // IDs
in {0..9}
  for (j = i*10⁹+1,
j<(i+1)*10⁹; j++) {
    if (isPrime(j))
      print(j);
  }
}
```



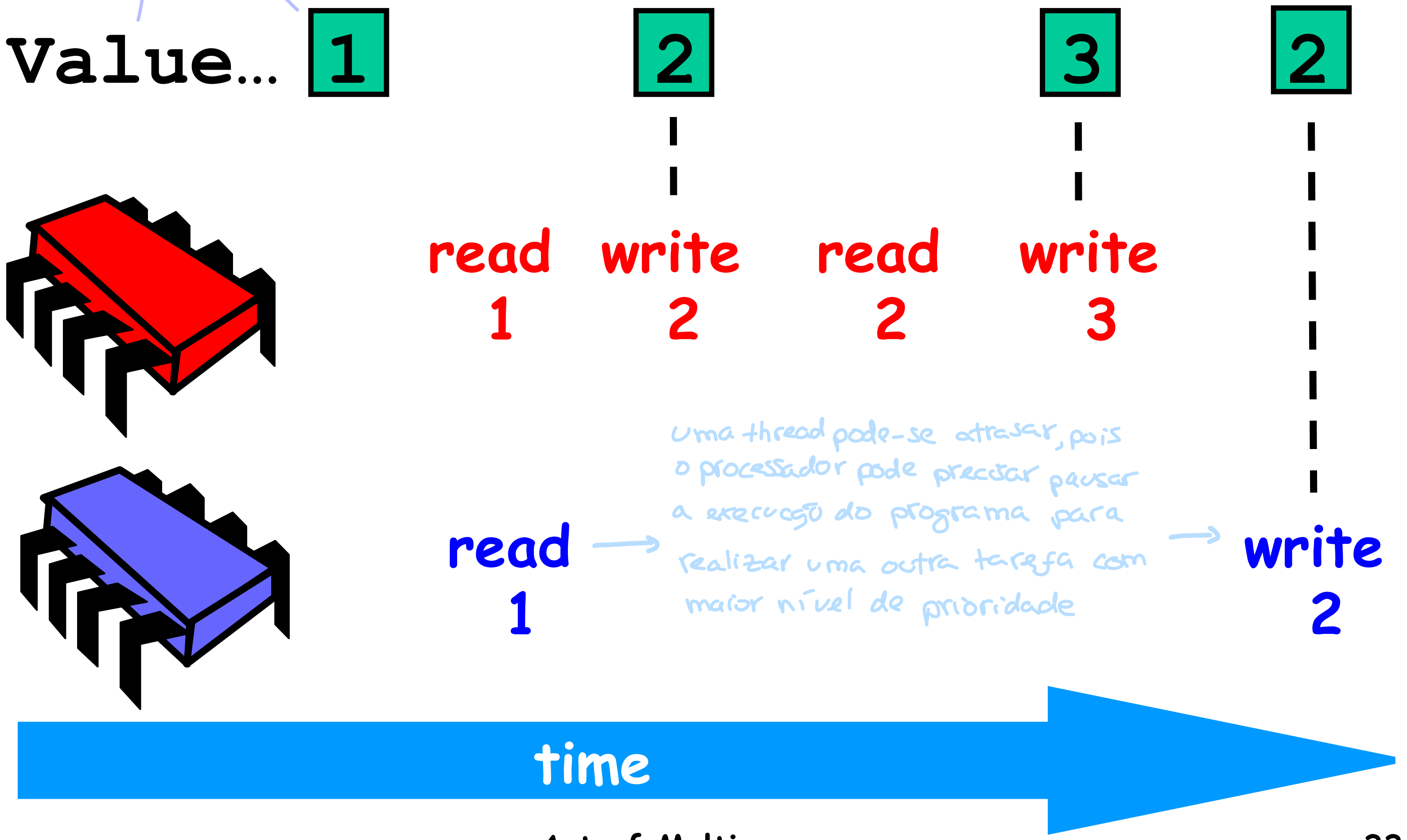
variáveis locais do código são guardadas em cache

As threads partilham recursos e comunicam entre si através da memória partilhada

# Not so good…

valor que está na memória partilhada

escreveu por cima do 3



Uma thread pode-se atrasar, pois o processador pode precisar pausar a execução do programa para realizar uma outra tarefa com maior nível de prioridade

## Challenge

```
public class Counter {
  private long value;

  public long getAndIncrement() {
    temp  = value;
    value = temp + 1;
    return temp;
  }
}
```

Make these steps *atomic* (indivisible)