Лекция 5.3: Многопоточность / JMM, volatile.

Ваша успеваемость

Java Core	14.34%
1. Основы Java	88.07%
2. Базовые алгоритмы	24.13%
3. Исключения	31.21%
4. Ввод/вывод	0%
5. Многопоточность	0%
6. Коллекции	0%
7. ООП: Синтаксис	0%
8. ООП: Шаблоны	0%
9. Продвинутые возможности	0%
10. Java 8	0%

Демонстрация некорректного кода

5. branch prediction, speculative execution, hyperthreading, конвейерные архитектуры, ???
including pipelining, branch prediction, executing multiple instructions in the same clock cycle(s), and even reordering the instruction stream for out-of-order execution
MMX, SSE
A superscalar CPU architecture implements a form of parallelism called instruction level parallelism within a single processor. It therefore allows faster CPU throughput than would otherwise be possible at a given clock rate. A superscalar processor executes more than one instruction during a clock cycle by simultaneously dispatching multiple instructions to redundant functional units on the processor.

Instruction-level parallelism (ILP)

ILP = http://en.wikipedia.org/wiki/Instruction_level_parallelism
- Instruction pipelining = http://en.wikipedia.org/wiki/Instruction_pipelining
- **A superscalar CPU architecture** implements a form of parallelism called instruction level parallelism within a single processor. It therefore allows faster CPU throughput than would otherwise be possible at a given clock rate. A superscalar processor executes more than one instruction during a clock cycle by simultaneously dispatching multiple instructions to redundant functional units on the processor.
- Out-of-order execution = http://en.wikipedia.org/wiki/Out-of-order_execution
- Register renaming = http://en.wikipedia.org/wiki/Register_renaming
- Speculative execution = http://en.wikipedia.org/wiki/Speculative_execution
- Branch prediction = http://en.wikipedia.org/wiki/Branch_prediction
6. cache coherency, cache coherency protocol, MESI, MOESI, MESIF, ???

по введению в Java Memory Model
JSR 133 (Java Memory Model) FAQ

http://www.cs.umd.edu/~pugh/java/memoryModel/jsr-133-faq.html

или перевод

http://habrahabr.ru/company/golovachcourses/blog/221133/

Одновременный счетчик

```
public class ConcurrentCounter {
    public static final int N = 1000_000_000;
    public static int counter = 0;

    public static void main(String[] args) throws InterruptedException {
        Thread t0 = new Thread(new Runnable() {
            public void run() {
                for (int k = 0; k < N; k++) counter++;
            }
        });
        t0.start();
        Thread t1 = new Thread(new Runnable() {
            public void run() {
                for (int k = 0; k < N; k++) counter++;
            }
        });
        t1.start();

        t0.join();
        t1.join();

        System.out.println(counter);
    }
}
>>
```

Флаг остановки

Цена корректности

volatile
synchronized
AtomicXXX

Happens-before формализм (visibility, not ordering)

volatile
synchronized
AtomicXXX

Тест

Для прохождения теста по теме Thread.JMM.SyncVolatile (уровень сложности теста: Hard) нажмите “Start Quiz”

Тесты по всей лекции

Тест, состоящий из случайных вопросов тестов этой лекции

Some issues occured with quiz randomizer :(

Видео

Набор декабрь 2013
Набор июль 2013
Набор апрель 2013
Набор февраль 2013
Набор октябрь 2012

Лабораторные

thread.async_call

aaa

```
import java.util.NoSuchElementException;
import java.util.concurrent.Callable;
import java.util.concurrent.atomic.AtomicInteger;

public class AsyncUtils {
    public <T> Ticket<T> asyncCall(final Callable<T> method) {


    }
}
class Ticket<T> {
    private static final AtomicInteger lastId = new AtomicInteger(0);
    private final int id = lastId.incrementAndGet();
    public boolean equals(Object other) {
        if (this == other) return true;
        if (other == null || getClass() != other.getClass()) {
            return false;
        }
        return id == ((Ticket) other).id;
    }
    public int hashCode() {
        return id;
    }
}
```

Литература

- Руслан Черемин, “Cache-coherency: Basics, MSI”
- Руслан Черемин, “Cache-coherency: от MSI к MESI и далее к звездам (MESIF, MOESI)”
- Jeremy Manson, “Causality and the Java Memory Mode”
- “Causality Test Cases”