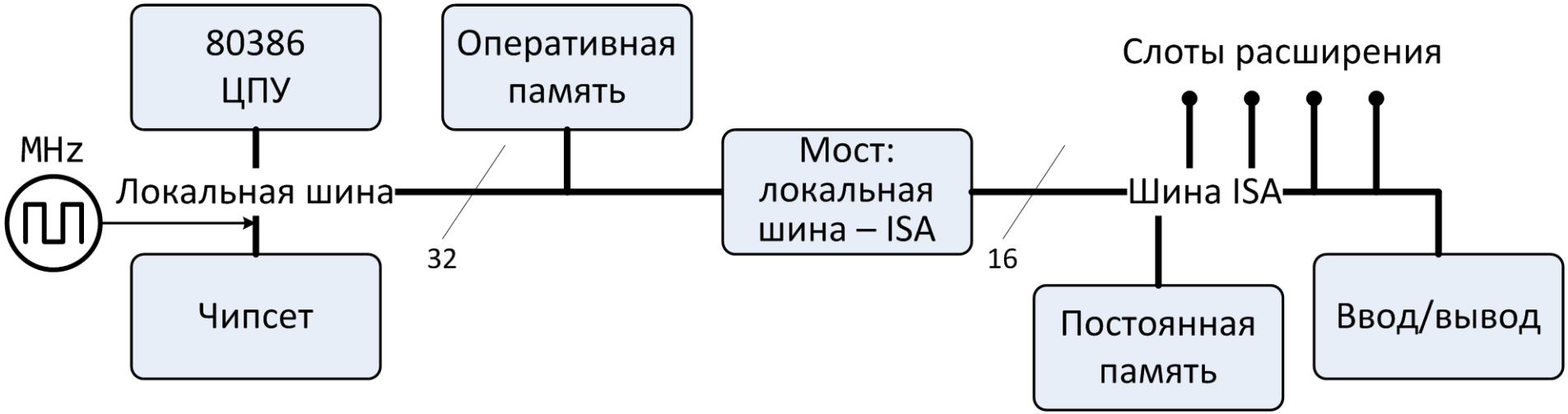
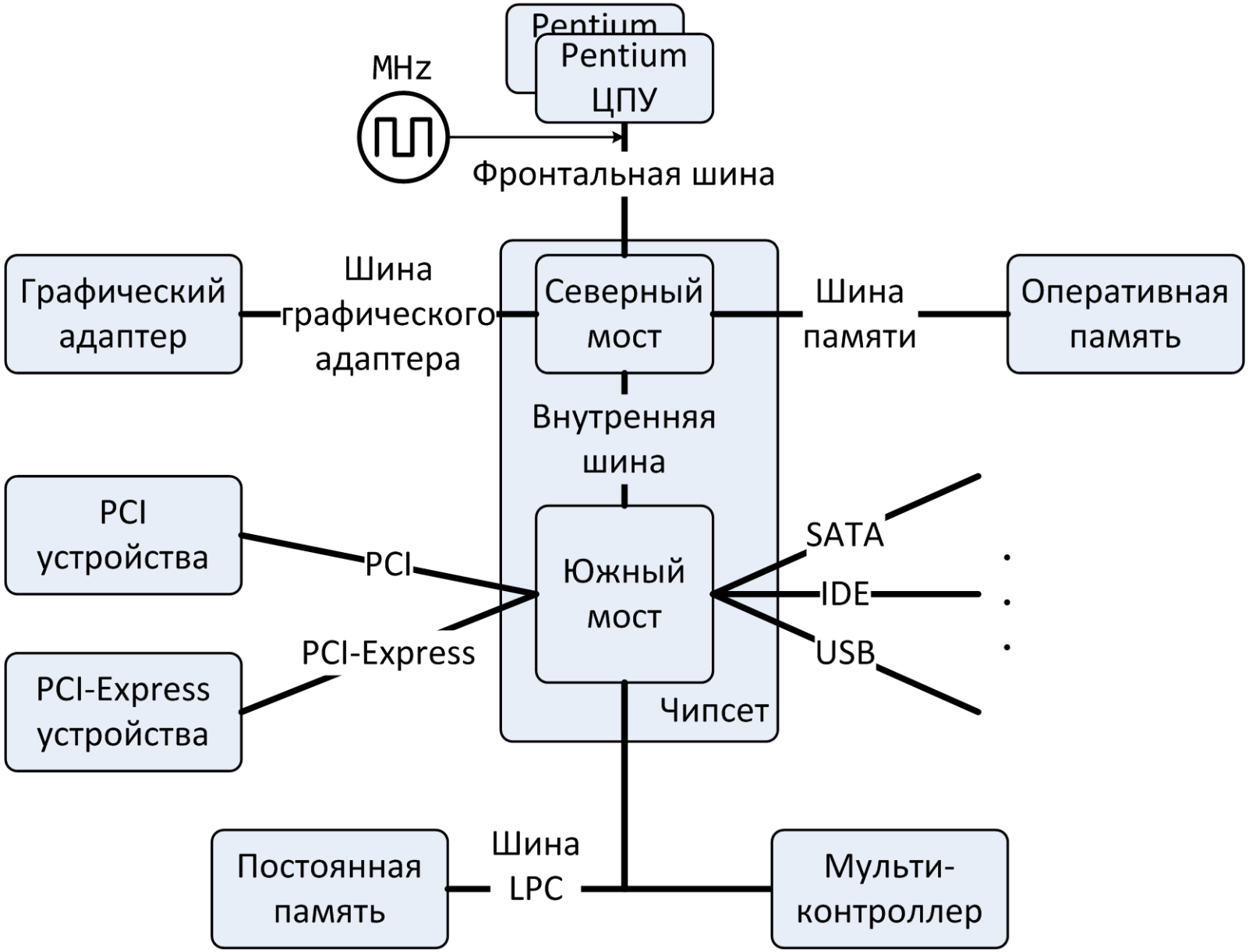
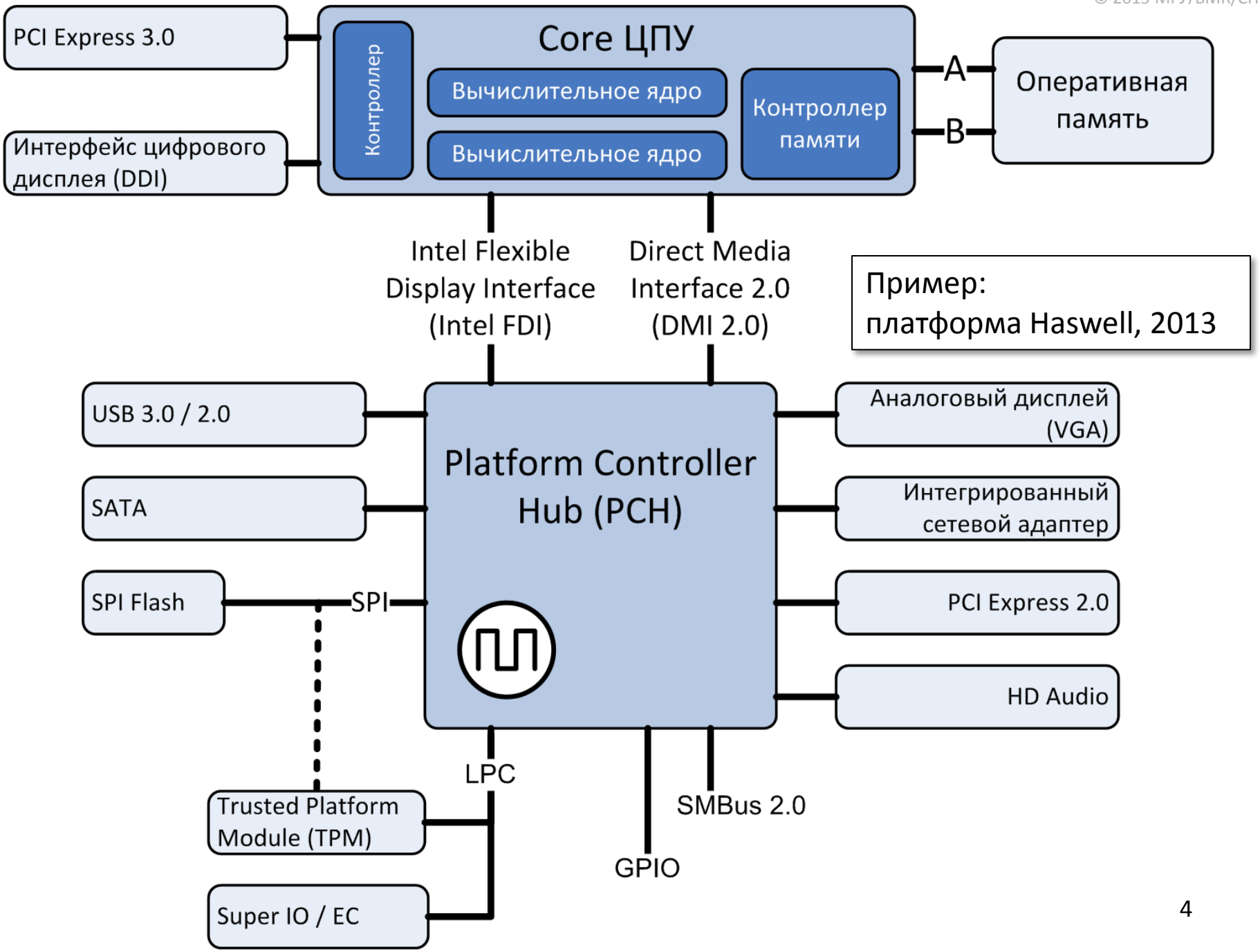
Лекция 0x14

15 апреля

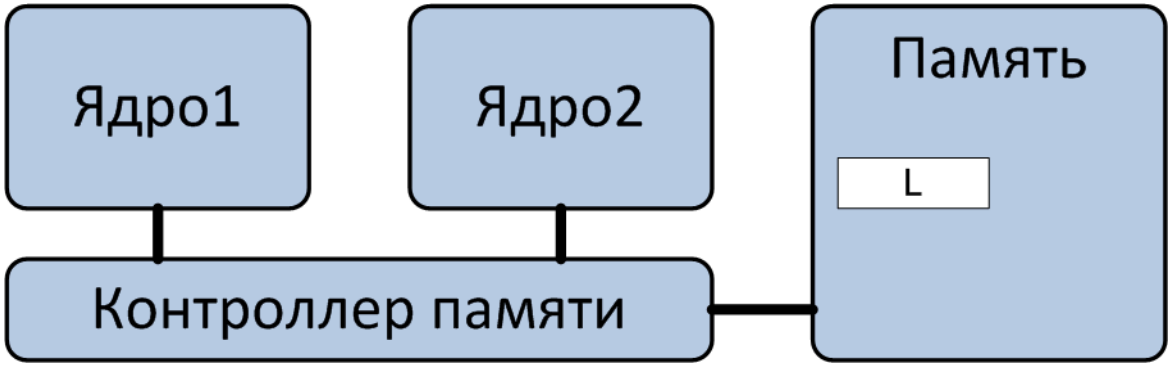






Синхронизация обращений к памяти

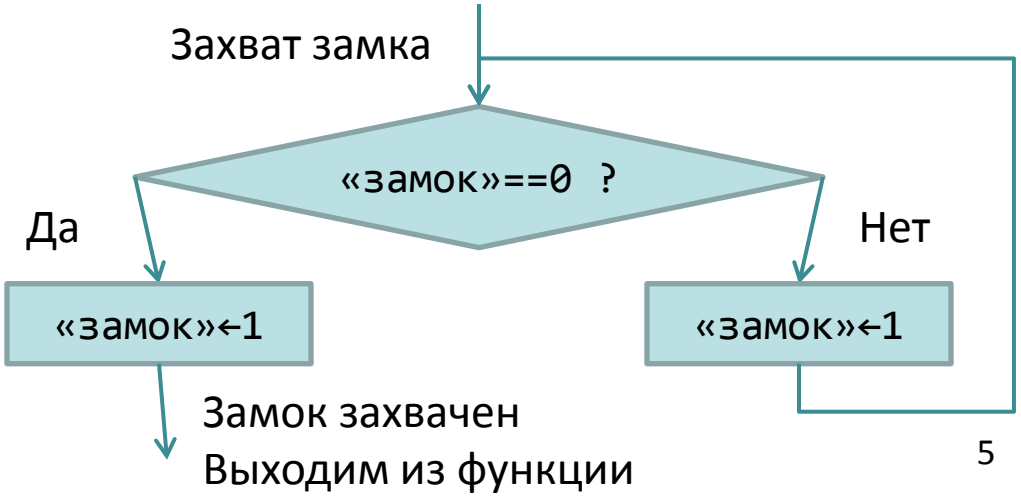
- Захват блокировки
- Работа с общими (разделяемыми) переменными
- Освобождение блокировки



Замок – статическая переменная L (младший разряд)
- 0 – доступ открыт
- 1 – доступ закрыт

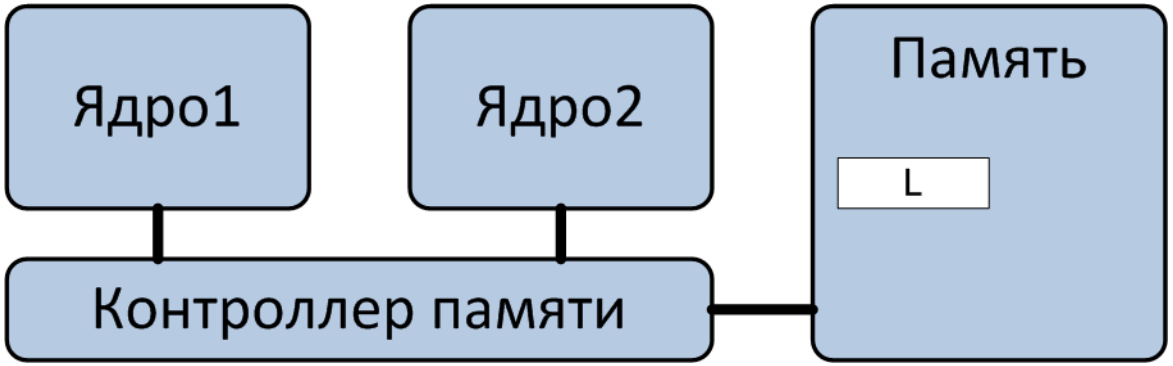
```
acquireLock:
 .retry:
   bts byte [L], 0
   jc .retry
   ret

releaseLock:
   btr byte [L], 0
   ret
```



Синхронизация обращений к памяти

- Захват блокировки
- Работа с общими (разделяемыми) переменными
- Освобождение блокировки



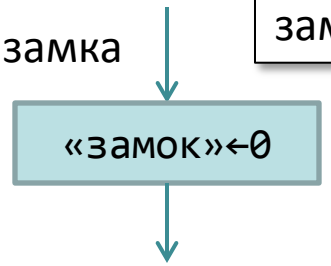
Замок – статическая переменная L (младший разряд)
- 0 – доступ открыт
- 1 – доступ закрыт

Начальное состояние замка – 0

```
acquireLock:
 .retry:
   bts byte [L], 0
   jc .retry
   ret

releaseLock:
   btr byte [L], 0
   ret
```

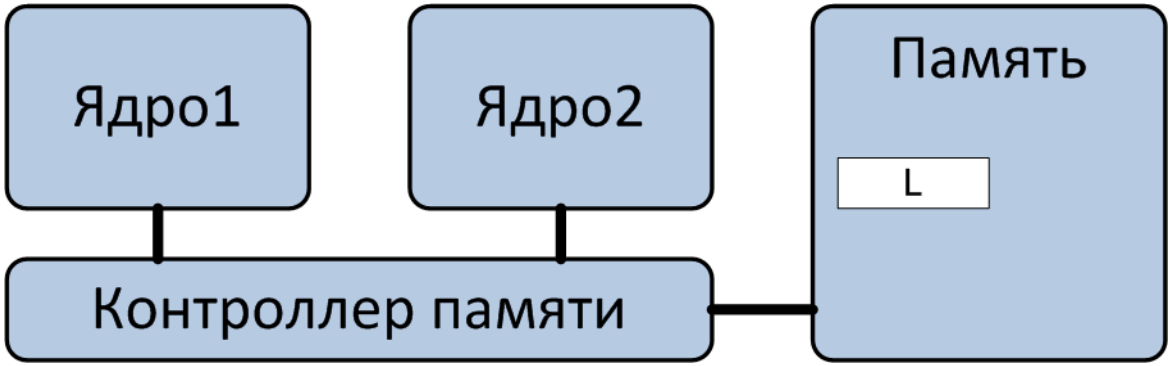
Освобождение замка



Замок освобожден
Выходим из функции

Синхронизация обращений к памяти

- Захват блокировки
- Работа с общими (разделяемыми) переменными
- Освобождение блокировки



Оба ядра попытались захватить замок «***почти***» одновременно

```
acquireLock:
 .retry:
   bts byte [L], 0
   jc .retry
   ret

releaseLock:
   btr byte [L], 0
   ret
```

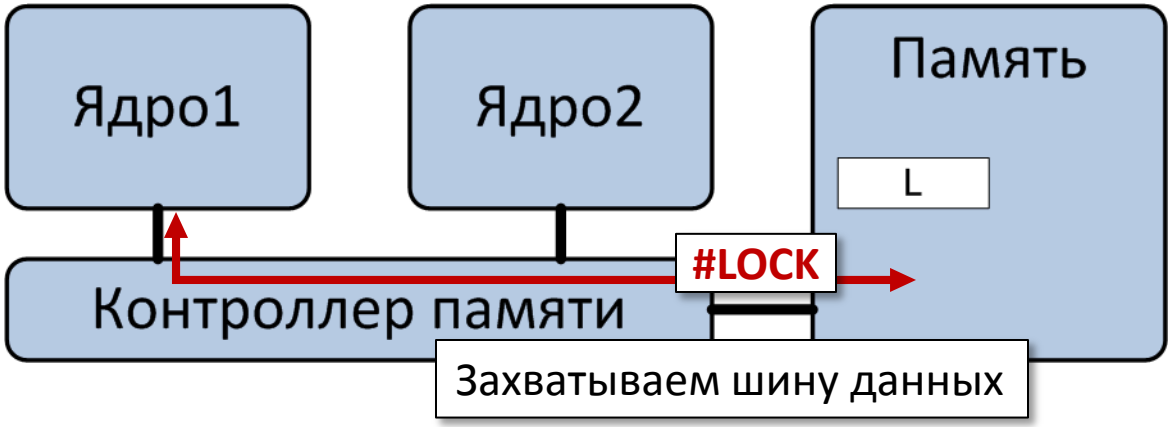
Может возникнуть ***Состояние Гонки***

Ядро1		Ядро2	
ЦПУ ← память	BTS		
CF ← ЦПУ:L|0		ЦПУ ← память	BTS
память:L ← 1		CF ← ЦПУ:L|0	
		память:L ← 1	

Время

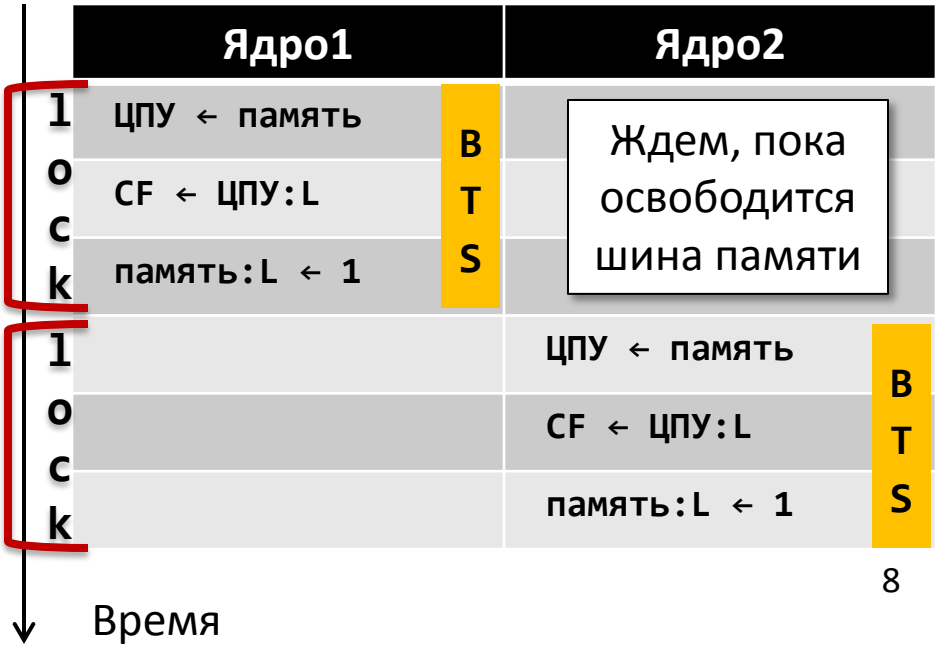
Синхронизация обращений к памяти

- Захват блокировки
- Работа с общими (разделяемыми) переменными
- Освобождение блокировки




```
acquireLock:
 .retry:
   lock bts byte [L], 0
   jc .retry
   ret

releaseLock:
   lock btr byte [L], 0
   ret
```



Синхронизация обращений к памяти

- Явно указываемый префикс `lock`
  - Применим только к некоторым командам ADD, AND, BTC, BTR, BTS, CMPXCHG, …
    - Первый операнд команды – память
- Все время выполнения команды процессор удерживает шину памяти, посылая на нее сигнал #LOCK
- Постоянный захват шины негативно сказывается на производительности

```
acquireLock:
   lock bts byte [L], 0
   jc .retry
   ret
 .retry:
   pause
   lock bts byte [L], 0
   jc .retry
   ret
```

В современных процессорах команда `pause` используется как подсказка, что выполнение находится в цикле активного ожидания (busy wait)

ОС Linux использует для реализации активного ожидания не `bts/btr`, а гораздо более быструю команду `cmpxchg`
http://heather.cs.ucdavis.edu/~matloff/50/PLN/lock.pdf (на английском)

Примеры шин (1/3)

- Фронтальная шина
  - HyperTransport (HT), апрель 2001, AMD. Открытый стандарт - HyperTransport Technology
    - 2 – 32 разряда, двунаправленная
    - 200 – 2600 MHz, DDR
  - QuickPath Interconnect (QPI), ноябрь 2008, Intel
    - 20 линий, двунаправленная, 4 такта = 64 бита
    - 2.4, 2.93, 3.2 GHz, DDR
  - Соединение точка-точка.
    - Гарантированные физические каналы: один отправитель – один приемник.
    - Не требуется арбитраж.

Примеры шин (2/3)

- Общая шина ввода/вывода для подключения периферийных устройств Peripheral component interconnect (PCI), 1992, Intel, открытый стандарт.
  - PCI 1.0 / 2.0
    - Топология - общая шина, децентрализованный арбитраж
    - 32 линии, общие для адресов и данных
      - Передача данных транзакциями, возможна приостановка
    - Частота 33 MHz
  - Расширения
    - PCI 64, PCI 66, PCI 64/66, PCI-X (266 и 533 МГц)
  - PCI Express (PCI-E), июль 2002, Intel
    - Топология – звезда.