Интернет Университет Суперкомпьютерных технологий

Учебный курс

Введение в параллельные алгоритмы

Лекция 2

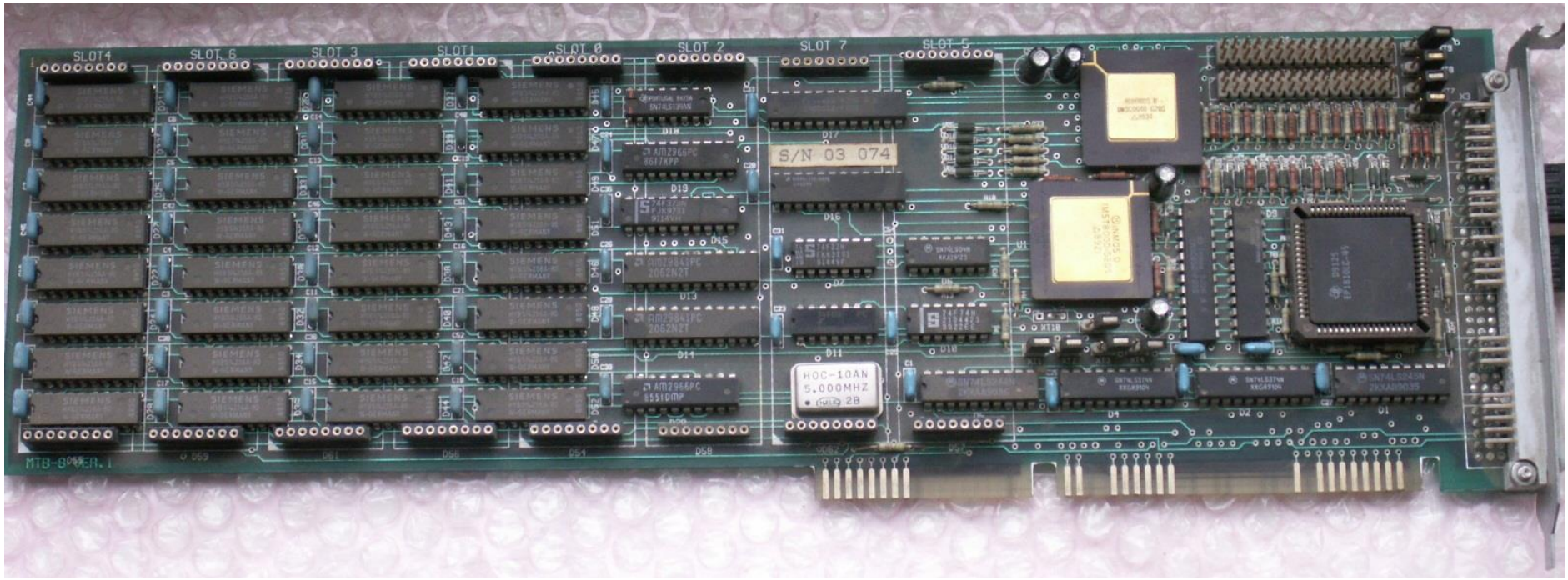
Основные понятия

Якобовский М.В., проф., д.ф.-м.н.
Институт прикладной математики
им. М.В.Келдыша РАН, Москва

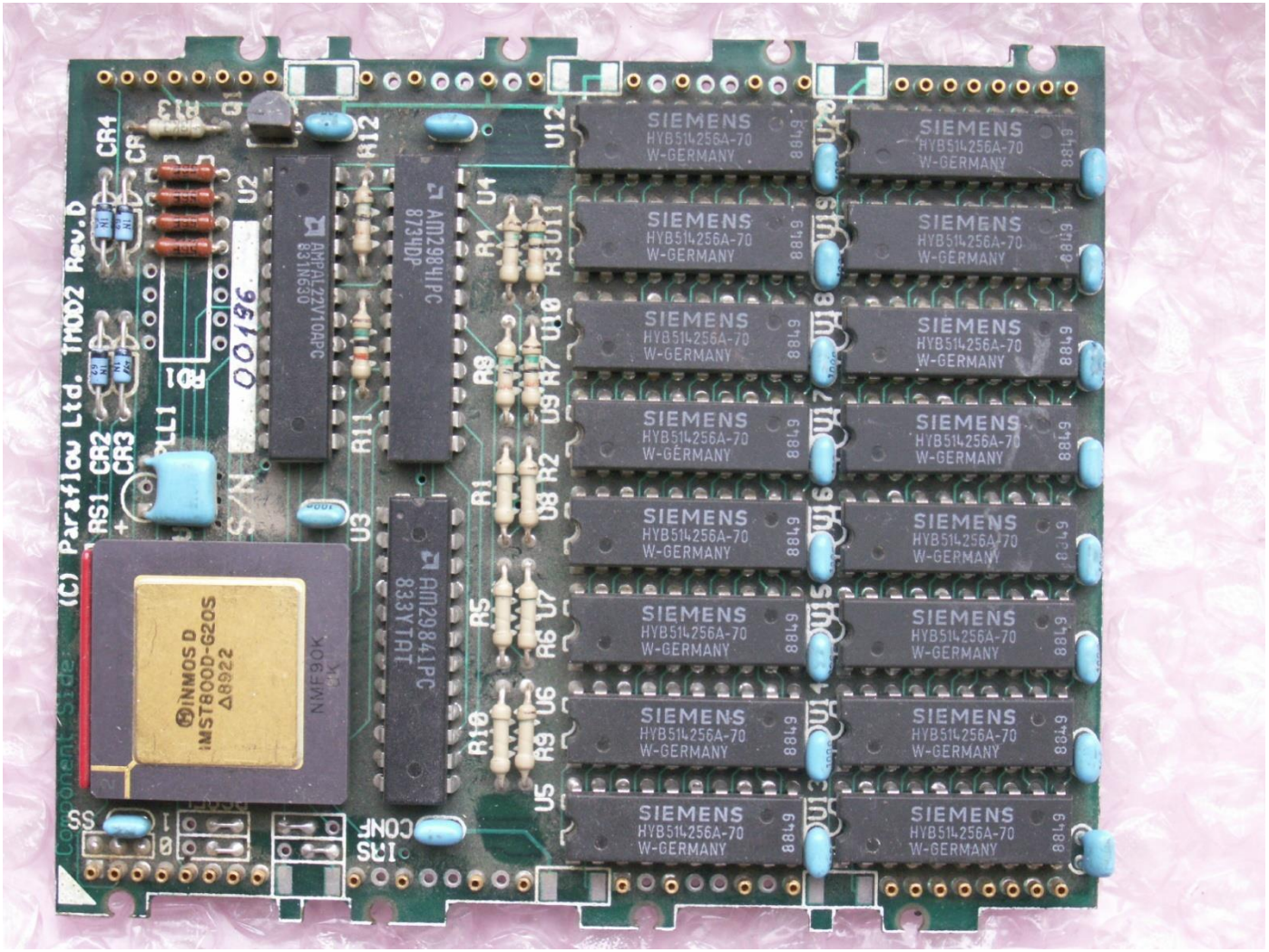
Содержание лекции

- Многопроцессорные системы
  - с распределенной памятью
  - с общей памятью
  - Гибридные
- Модели выполнения программ
- Свойства канала передачи данных
- Методы взаимодействия процессов
  - Методы передачи данных
  - Семафоры
- Ускорение и эффективность параллельных алгоритмов
- Пример алгоритма с низкой эффективностью, но высоким быстродействием

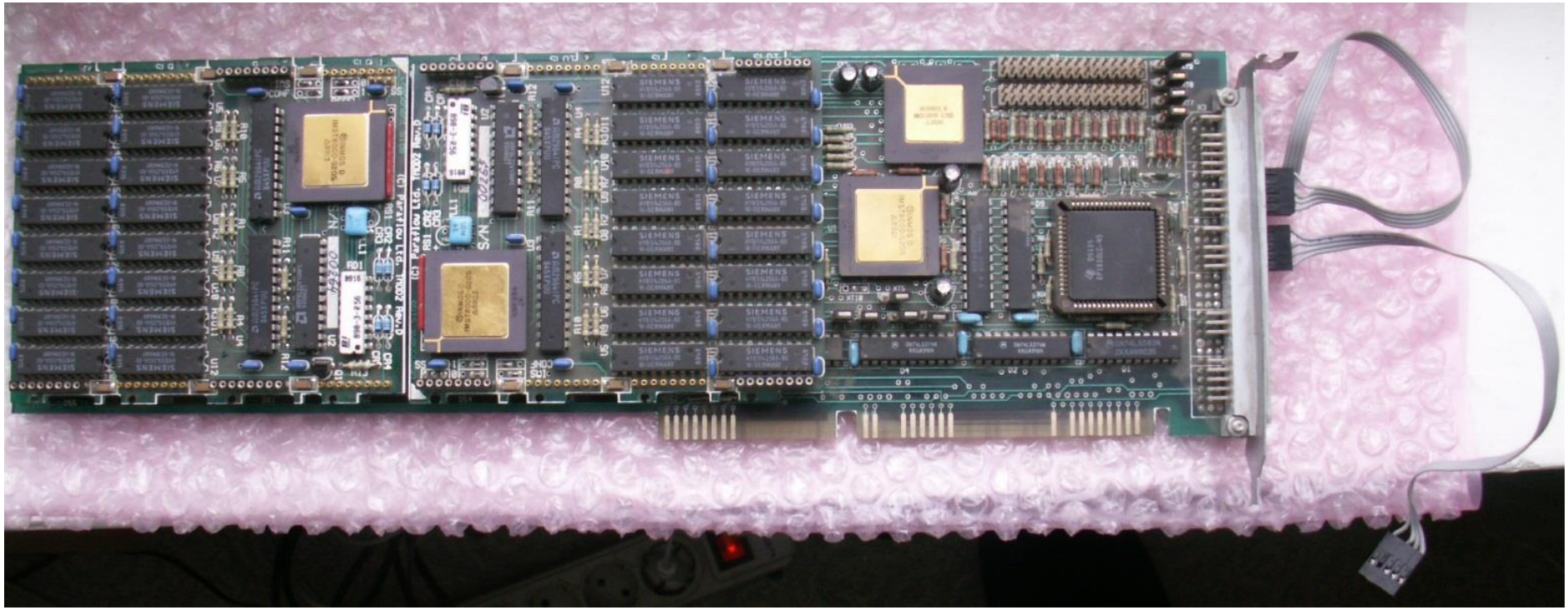
Транспьютерная материнская плата МТБ-8



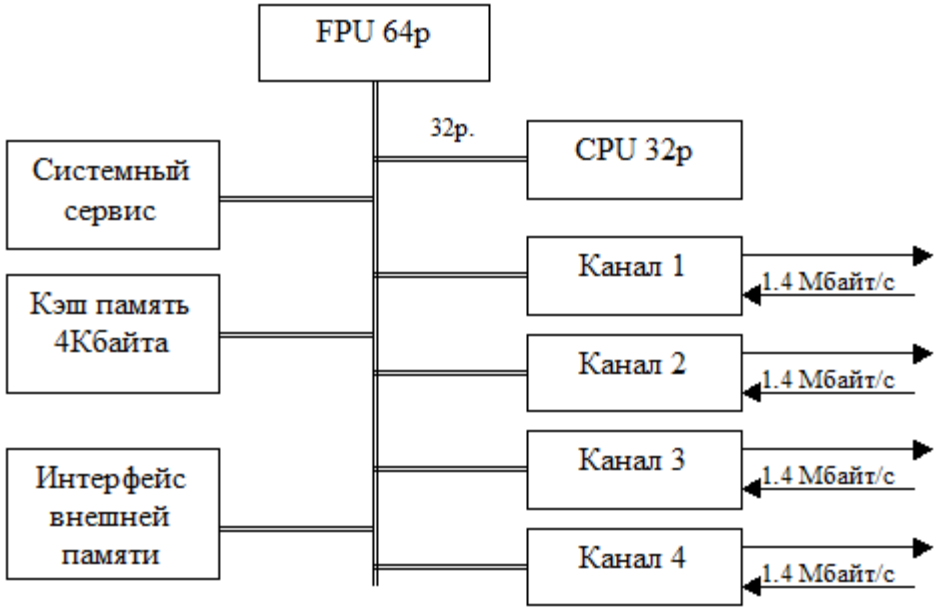
Транспьютер и оперативная память



Три транспьютера на плате МТБ-8

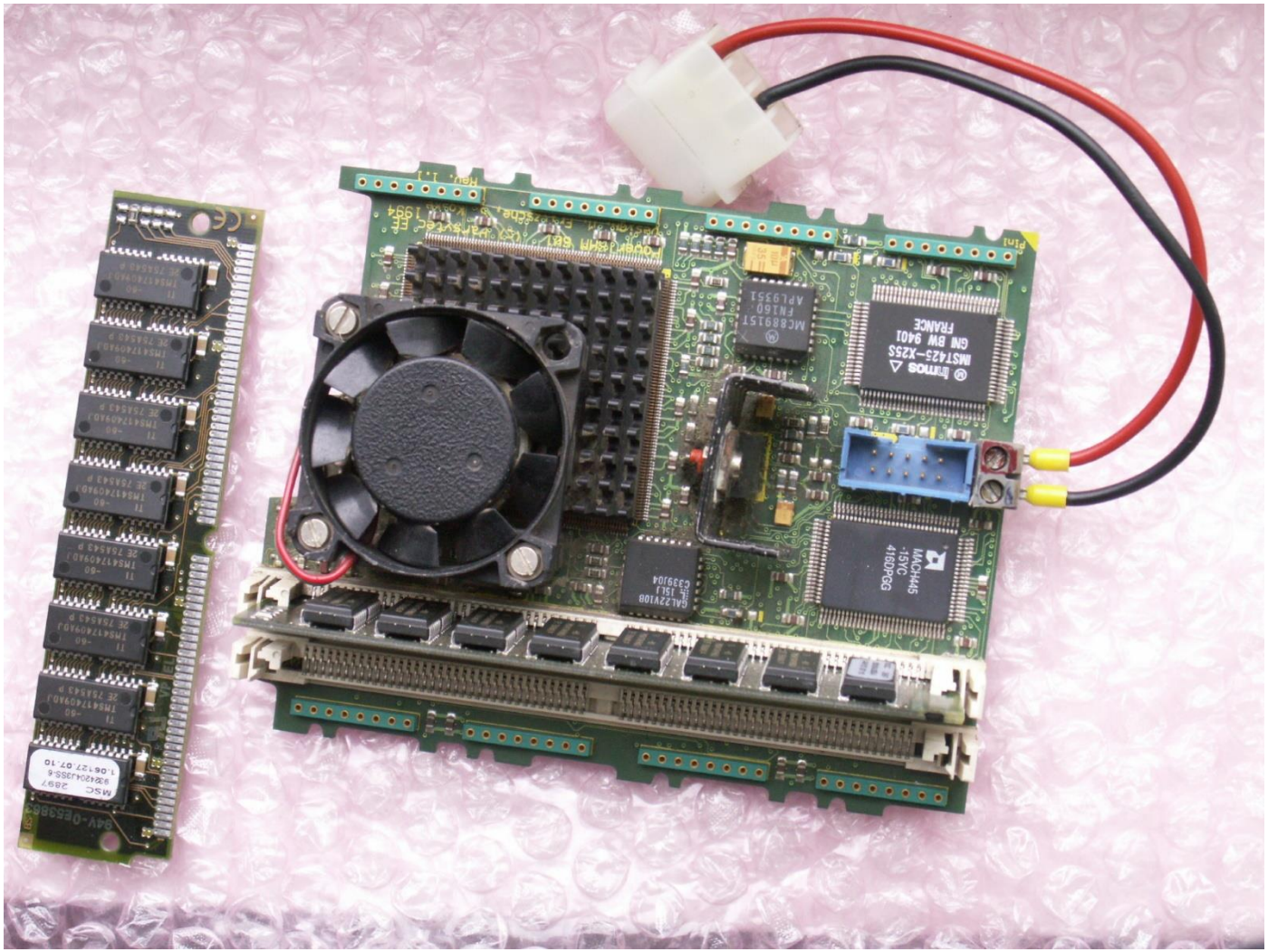


Структура транспьютера Т-800

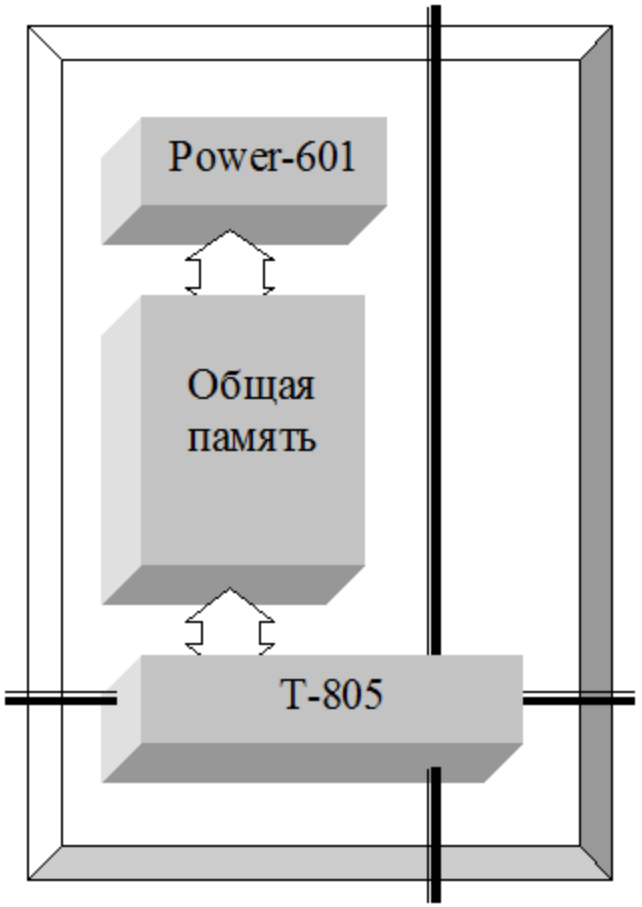


Структура транспьютера Т-800

Узел с общей памятью – транспьютер и процессор общего назначения

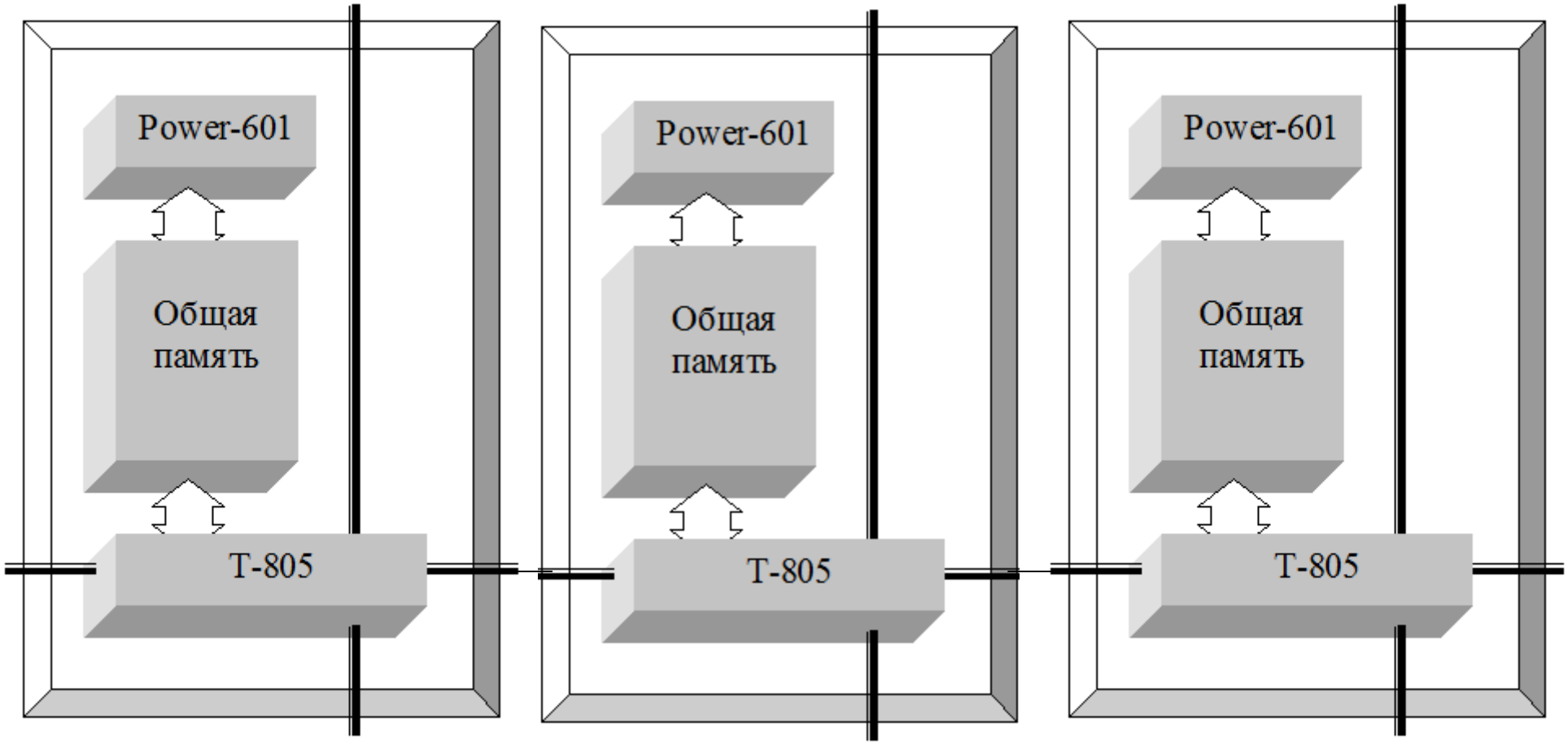


Узел PowerXplorer



Структура узла PowerXplorer

Гибридная система



Вопрос

Зачем нужны транспьютерные линки?

a. Для ввода-вывода данных
b. Для подвода питания к транспьютерам
c. Для передачи данных между транспьютерами

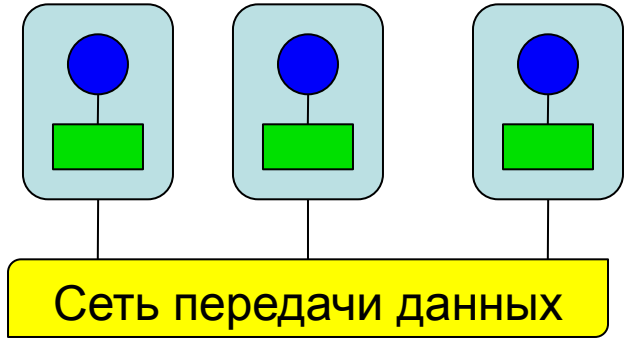
Вопрос

Можно ли на основе транспьютеров делать системы с общей памятью?

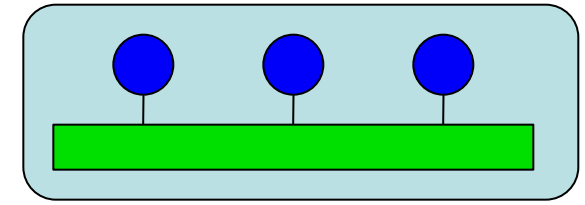
a. Можно делать
b. Нельзя делать
c. Только такие системы и можно делать

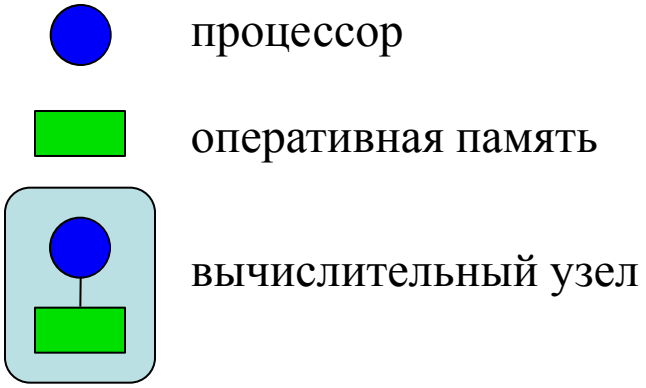
Уточнение круга рассматриваемых систем

- Системы на основе объединенных сетью типовых вычислительных узлов – системы с распределенной оперативной памятью



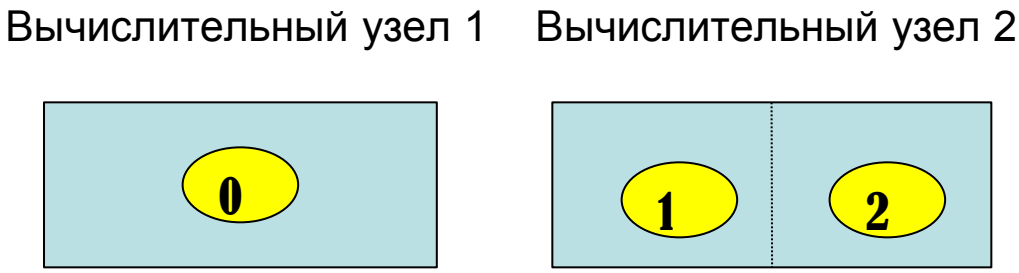
- Системы с доступом всех процессоров к общей оперативной памяти





Модель выполнения программы на распределенной памяти

- При запуске указывается число требуемых процессоров Np и название программы
- На выделенных для расчета узлах запускается Np копий программы
  - Например, на двух узлах запущены три копии программы. Копия программы с номером 1 не имеет непосредственного доступа к оперативной памяти копий 0 и 2:



- В каждой копии программы известны значения двух переменных
  - Np – одинаковое во всех копиях – число копий
  - rank из диапазона [0 … Np-1] – уникальный номер копии
- Любые две копии программы могут непосредственно обмениваться данными с помощью функций передачи сообщений Send/Recv

Синхронные обмены данными

A=3

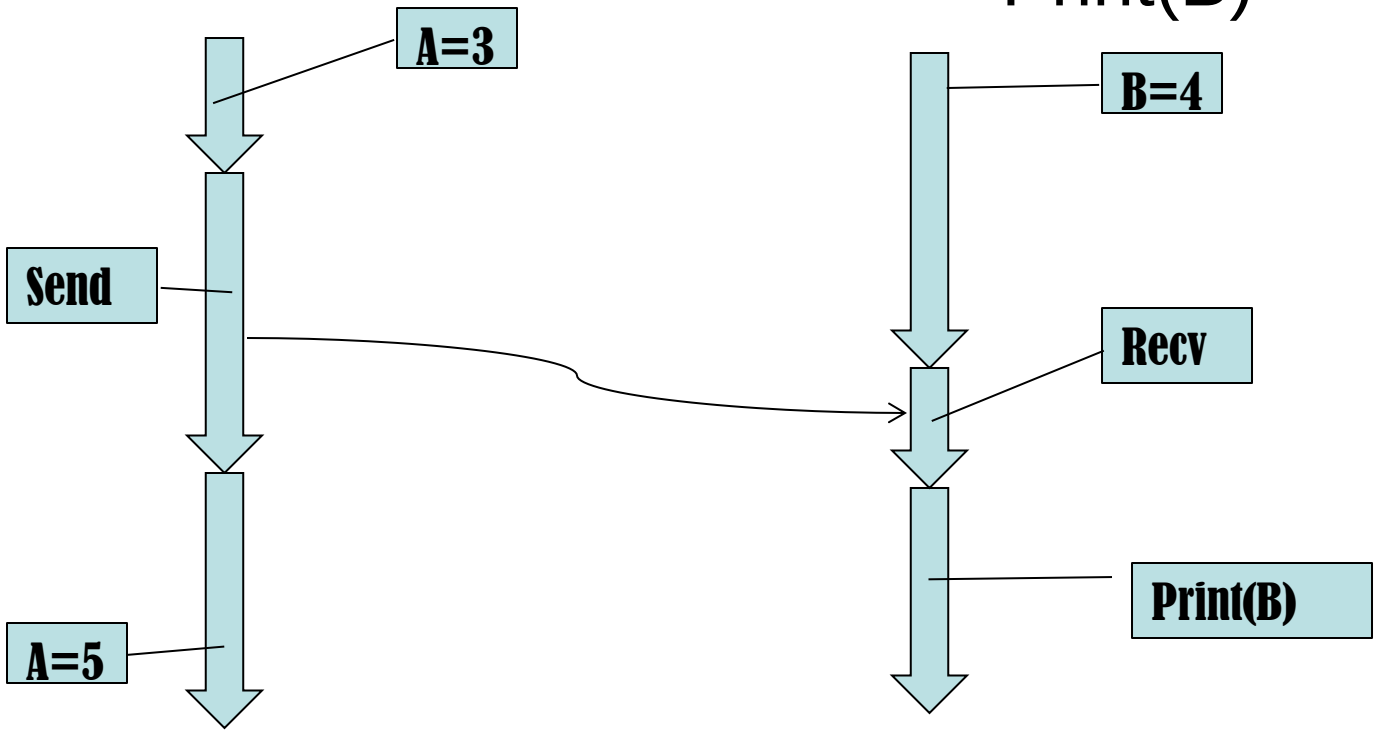
Send(&A)

A=5

B=4

Recv(&B)

Print(B)

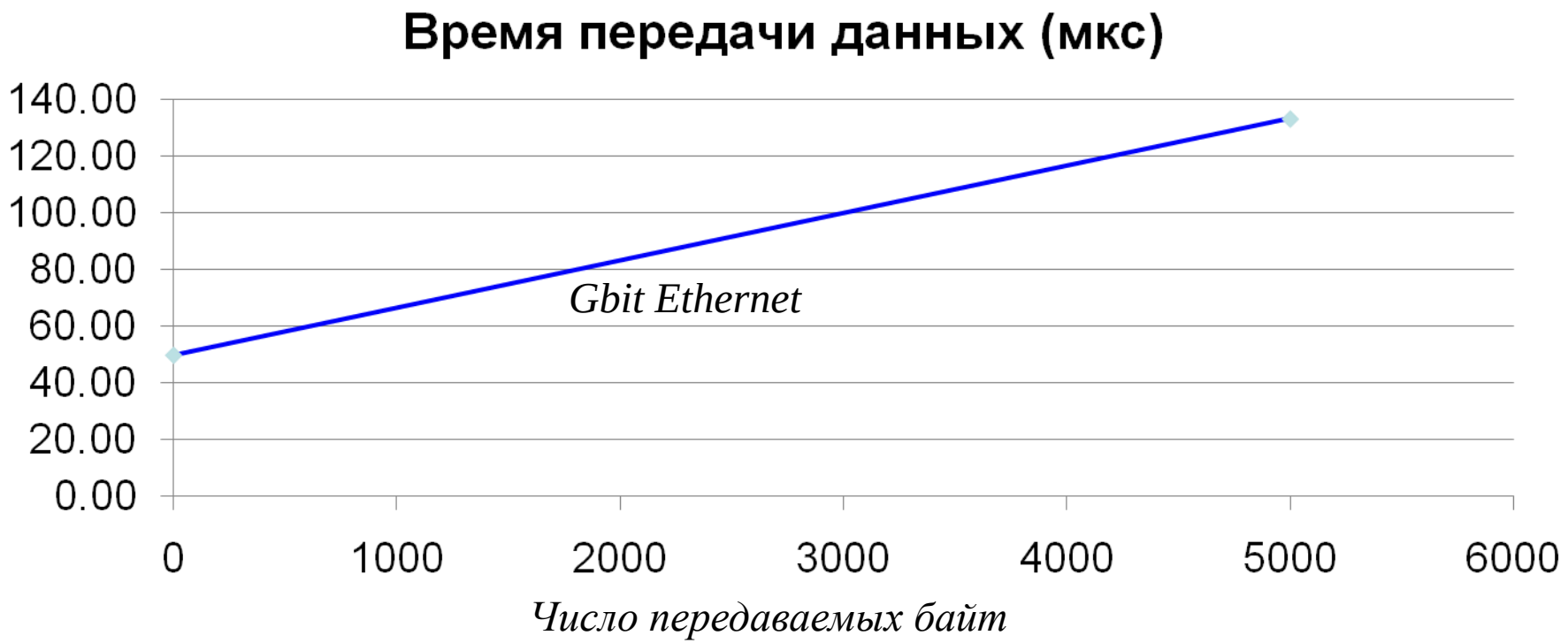


Результат

3

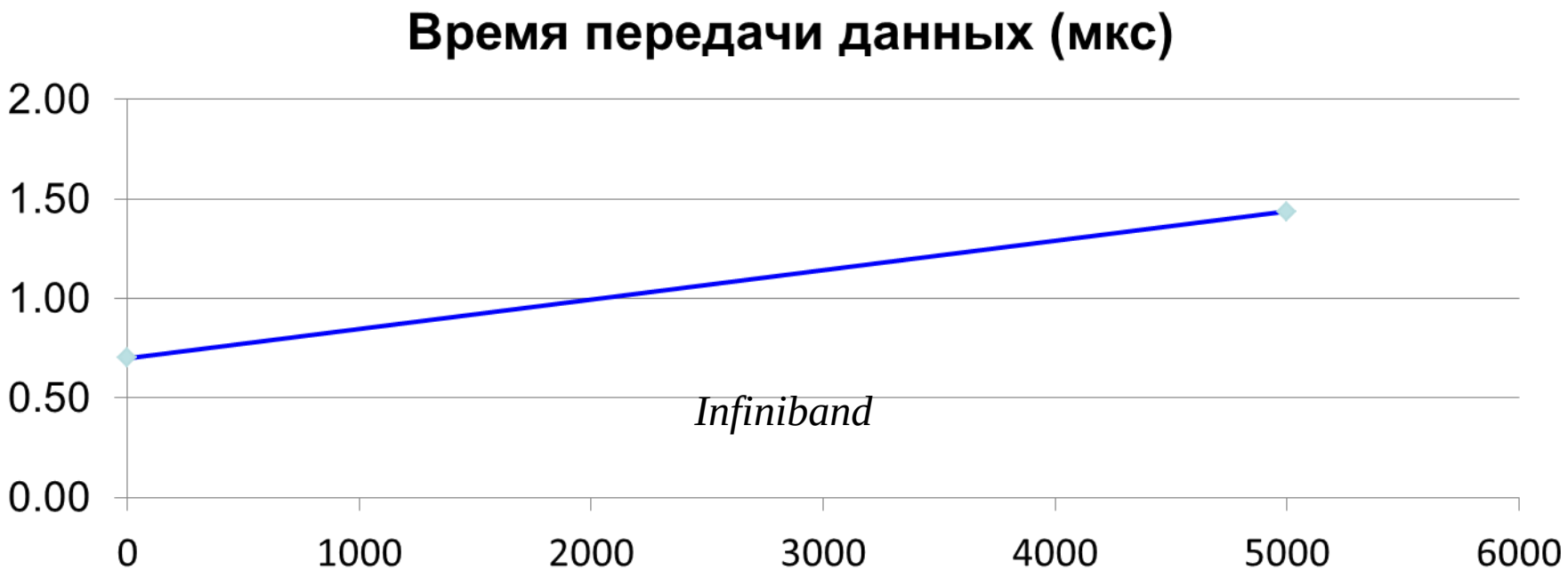
Свойства канала передачи данных

$$T(n) = n * T_{\text{передачи байта}} + T_{\text{латентности}}$$



Свойства канала передачи данных

$$T(n) = n * T_{\text{передачи байта}} + T_{\text{латентности}}$$



Число передаваемых байт

6.8 Гбайт/с

За 0.7 мкс передаётся 5 Кбайт

Вопрос

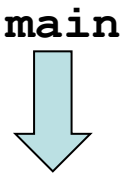
При каком способе передачи массива чисел будет затрачено меньше времени?

a. Передача каждого числа отдельным сообщением
b. Объединение всех чисел в единый массив и передача массива одним сообщением
c. Не имеет значения

Модель выполнения программы на общей памяти

- Работа начинается с запуска одного экземпляра программы
- При необходимости программа порождает новые процессы, каждый из которых:
  - Обладает собственными локальными переменными
  - Имеет доступ к глобальным переменным

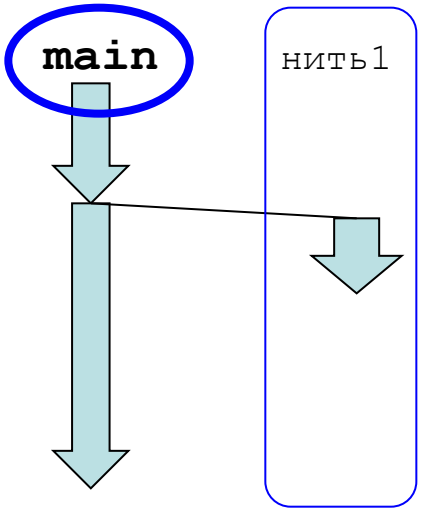
```
int a_global;
main()
{
int b1_local;
Запуск нити(fun())
}
fun()
{
int b2_local;
Запуск нити(…)
}
```



Модель выполнения программы на общей памяти

- Работа начинается с запуска одного экземпляра программы
- При необходимости программа порождает новые процессы, каждый из которых:
  - Обладает собственными локальными переменными
  - Имеет доступ к глобальным переменным

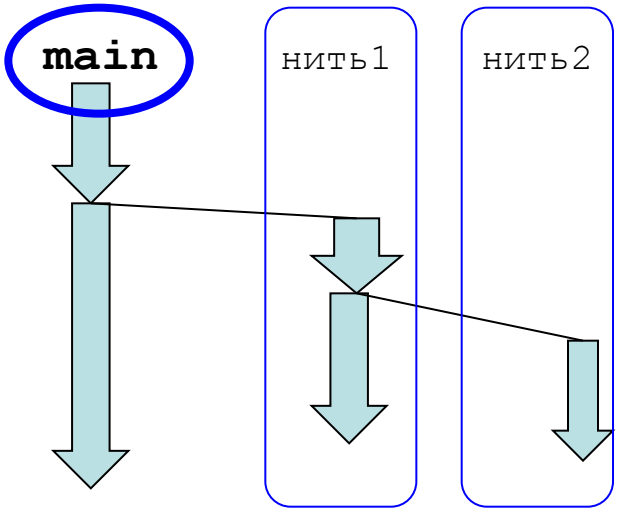
```
int a_global;
main()
{
int b1_local;
Запуск нити(fun())
}
fun()
{
int b2_local;
Запуск нити(…)
}
```



Модель выполнения программы на общей памяти

- Работа начинается с запуска одного экземпляра программы
- При необходимости программа порождает новые процессы, каждый из которых:
  - Обладает собственными локальными переменными
  - Имеет доступ к глобальным переменным

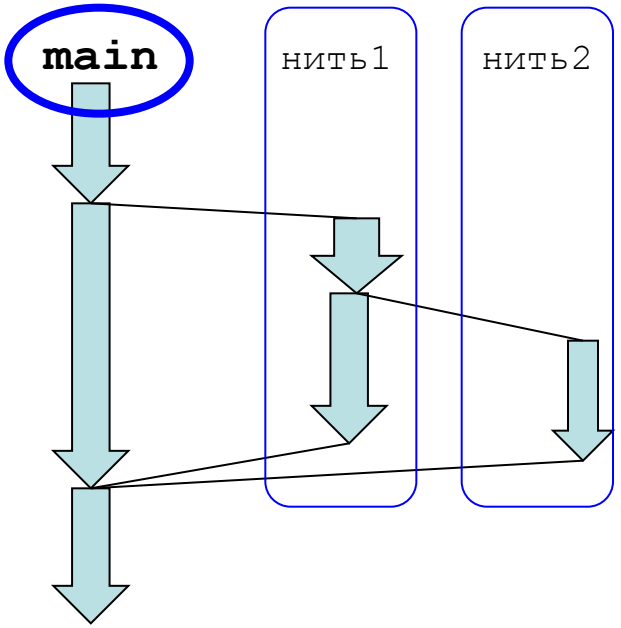
```
int a_global;
main()
{
int b1_local;
Запуск нити(fun())
}
fun()
{
int b2_local;
Запуск нити(…)
}
```



Модель выполнения программы на общей памяти

- Работа начинается с запуска одного экземпляра программы
- При необходимости программа порождает новые процессы, каждый из которых:
  - Обладает собственными локальными переменными
  - Имеет доступ к глобальным переменным

```
int a_global;
main()
{
int b1_local;
Запуск нити(fun())
}
fun()
{
int b2_local;
Запуск нити(…)
}
```



Что будет напечатано?

```
int a=1;
```

```
Нить1
{
   a=a+2
}
```

```
Нить2
{
   a=a+3
}
```

```
Нить3
{
   print(a)
}
```

a=?

Что будет напечатано?

```
int a=1;
```

```
Нить1
{
   a=a+2
}
```

```
Нить2
{
   a=a+3
}
```

```
Нить3
{
   print(a)
}
```

6

Что будет напечатано?

```
int a=1;
```

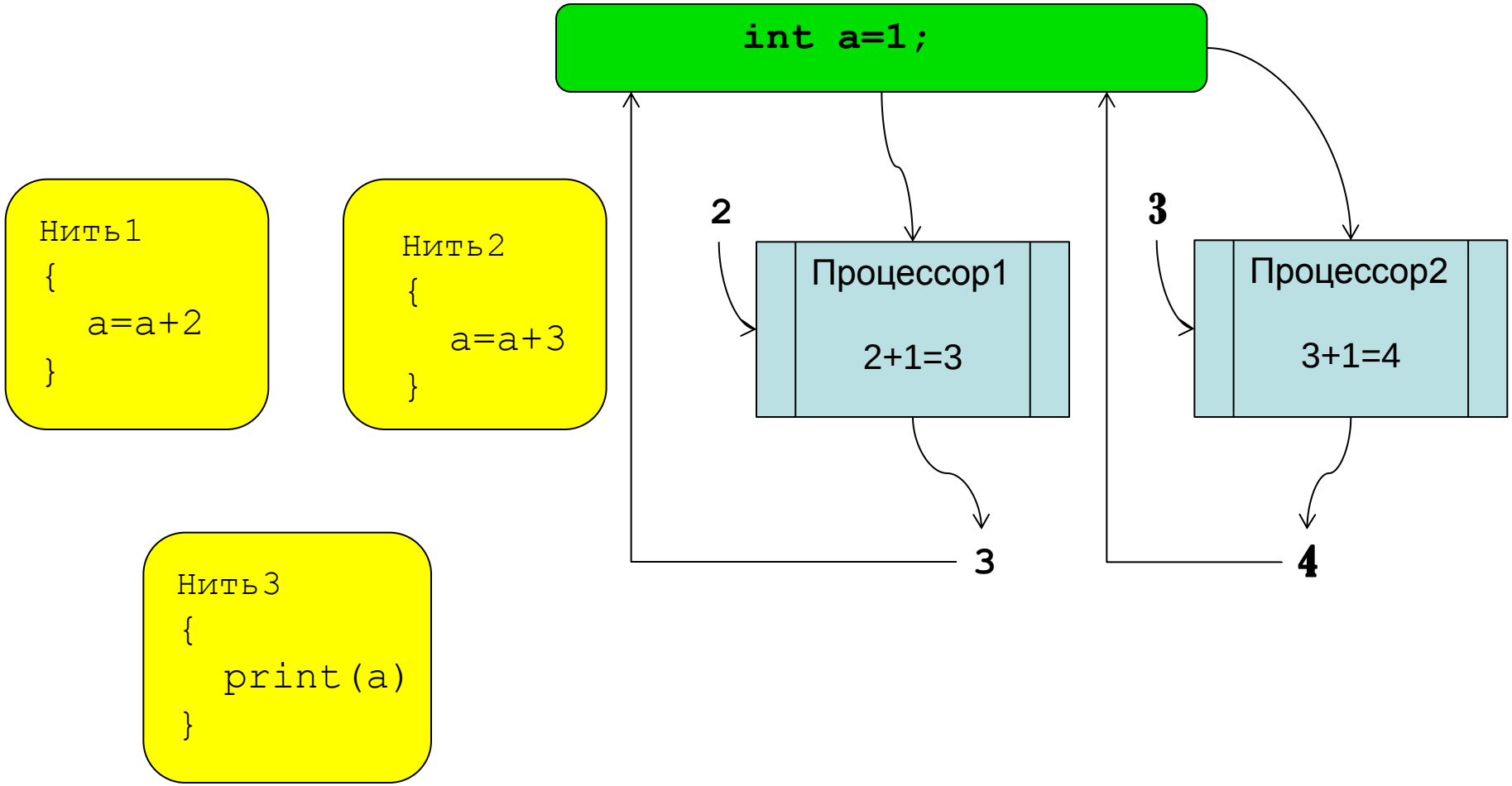
```
Нить1
{
   a=a+2
}
```

```
Нить3
{
   print(a)
}
```

```
Нить2
{
   a=a+3
}
```

3

Что будет напечатано?



6 ? 4 ? 3

Что будет напечатано?

```
int a=1;
```

```
Нить1
{
   a=a+2
}
```

```
Нить2
{
   a=a+3
}
```

```
Нить3
{
   print(a)
}
```

6 ? 4 ? 3 ? ___

Семафор

- Целочисленная неотрицательная переменная
- Инициализация и две атомарные операции

- Операция V(S)
  - Увеличивает значение S на 1

- Операция P(S)
  - Если S положительно, то уменьшает S на 1
  - Иначе ждет, пока S не станет больше 0

Языки программирования. Редактор Ф.Женюи. Перевод с англ. В.П.Кузнецова. Под ред. В.М.Курочкина. М:."Мир", 1972 Э. Дейкстра. Взаимодействие последовательных процессов. http://khpi-iip.mipk.kharkiv.edu/library/extent/dijkstra/ewd123/index.html

Что будет напечатано?

```
int a=0;
Sem S=1, S1=0, S2=0;
```

```
Нить1
{
   P(S)
   a=a+2
   V(S)
   V(S1)
}
```

```
Нить2
{
   P(S)
   a=a+3
   V(S)
   V(S2)
}
```

```
Нить3
{
 P(S1)
 P(S2)
 print(a)
}
```

6

Контакты

Якобовский М.В.

д.ф.-м.н., проф.,

зав. сектором

«Программного обеспечения многопроцессорных систем и вычислительных сетей»

Института прикладной математики им. М.В.Келдыша Российской академии наук

mail: lira@imamod.ru

web: http://lira.imamod.ru