ПОТОКОВЫЕ

ВЫЧИСЛИТЕЛЬНЫЕ

СИСТЕМЫ

Введение

Идеология вычислений, управляемых потоком данных, была разработана в 60-х годах Карпом и Миллером.
В начале 70-х годов Деннис, а позже и другие начали разрабатывать компьютерные архитектуры, основанные на вычислительной модели с потоком данных.

Это класс ВС, в которых очередной поток вычислений в соответствии с алгоритмом инициируется не инструкциями программы, а готовностью к обработке необходимых данных.

Эффективность подобных структур во многом определяется эффективным программированием, задачей которого является формулировка задачи в терминах параллельных и независимых операций.

Вычислительная модель потоковой обработки

В ПВМ для описания вычислений используется ориентированный граф - *граф потоков данных* (dataflow graph).

Этот граф состоит из

1. *Узлов* или *вершин,* отображающих операции;
2. *Ребер* или *дуг,* показывающих потоки данных между теми вершинами графа, которые они соединяют.

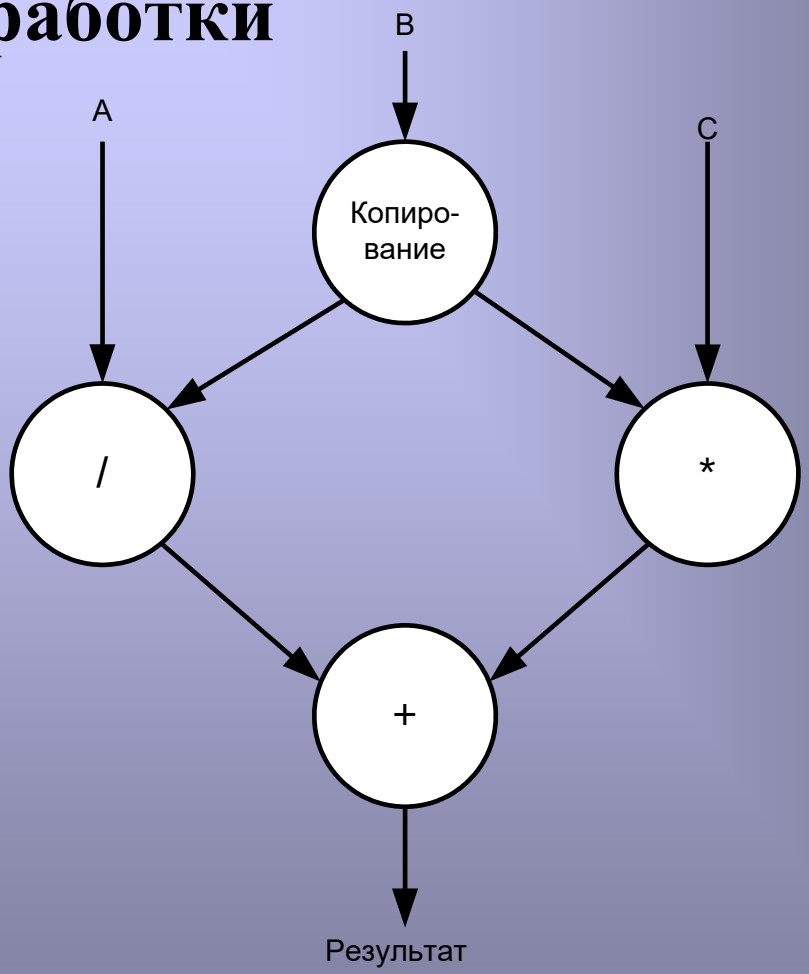


Рисунок 1. Граф потоков данных для выражения A/C+B*C

Движение токенов между узлами

Данные (операнды/результаты), перемещаемые вдоль дуг, содержаться в опознавательных информационных кадрах, маркерах специального формата — ***«токенах»*** (иначе «фишках» или маркерах доступа).

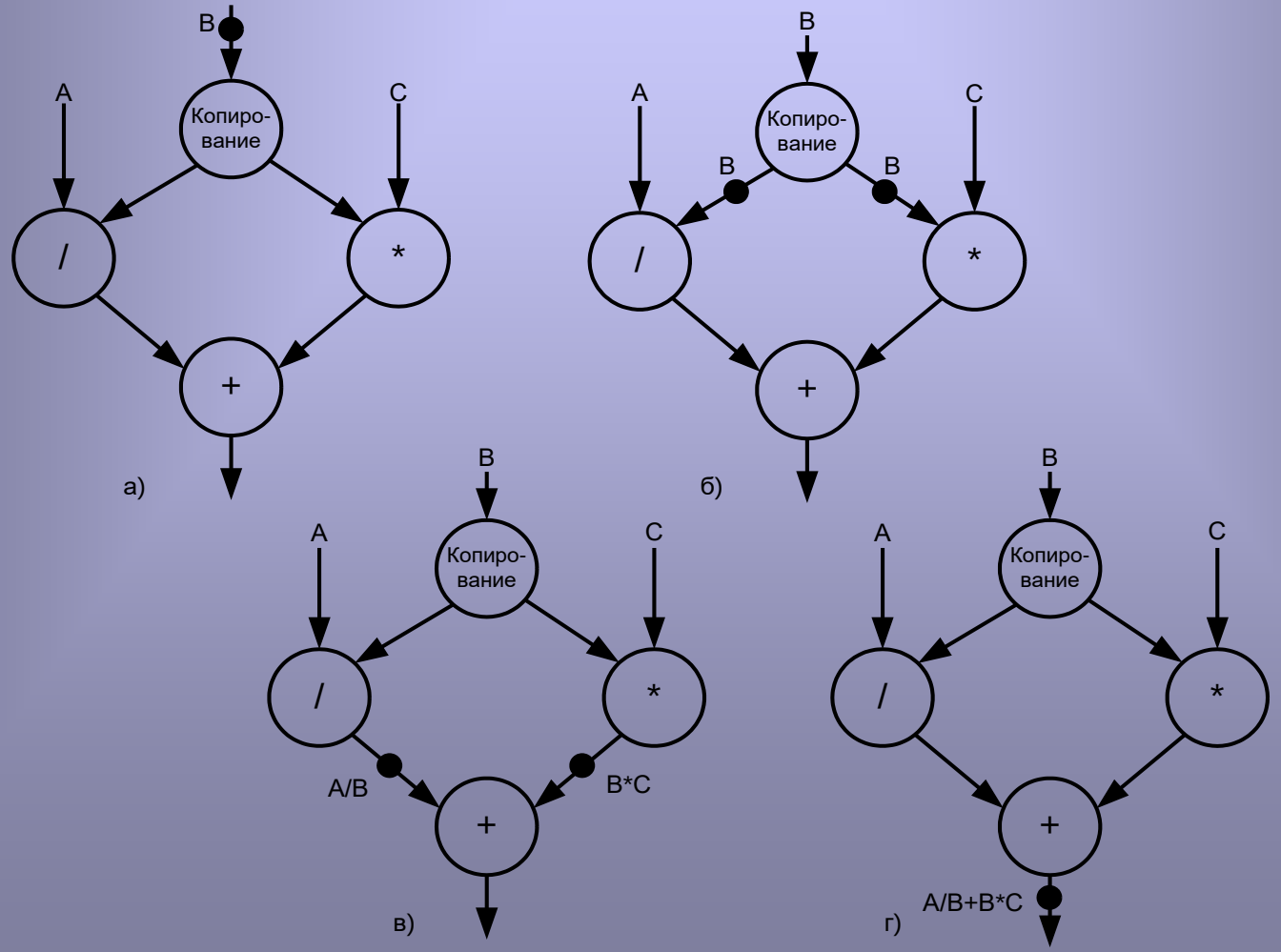


Рисунок 2. Движение маркеров при вычислении A/B+B*C: а – после передачи входных данных; б – после копирования; в – после умножения и деления; г – после суммирования

Примитивы узлов

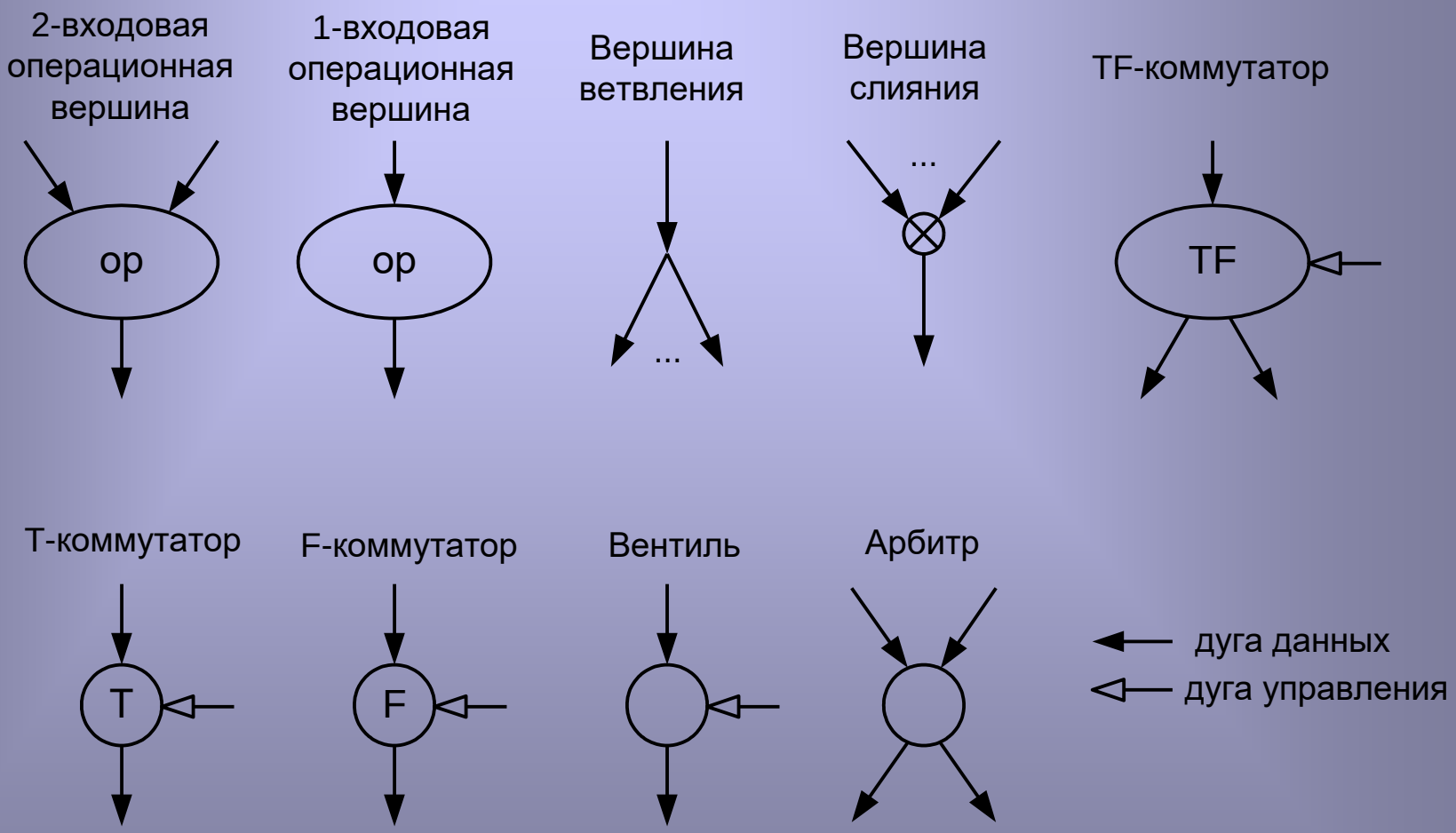
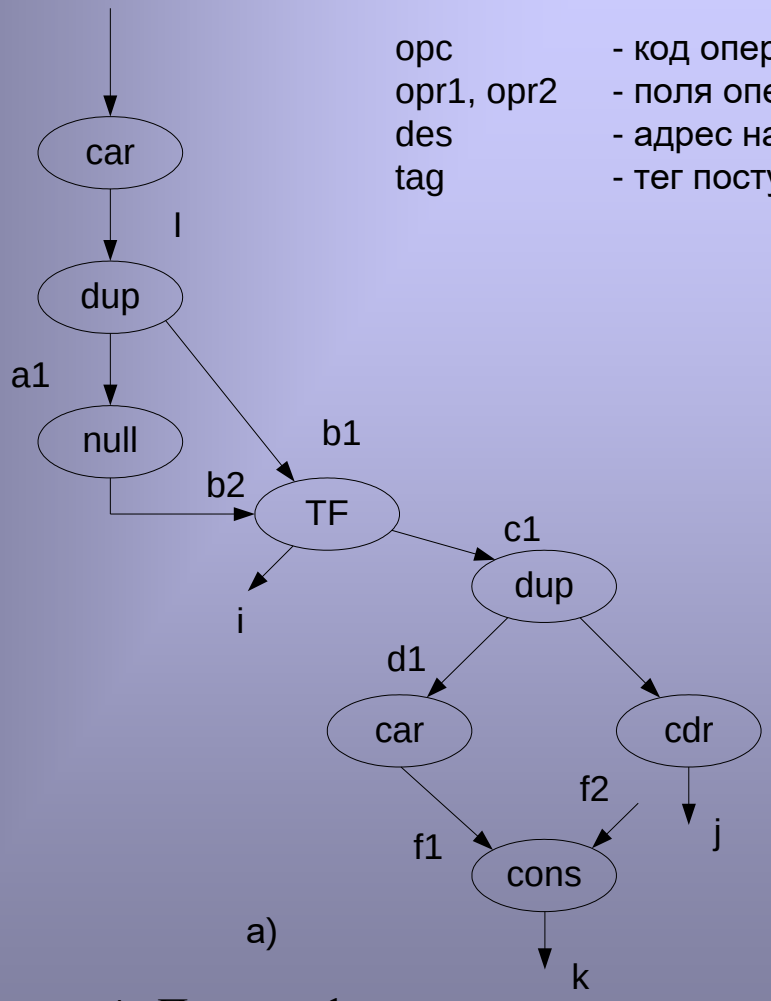


Рисунок 3. Примитивы узлов

Архитектура ПВС

opc - код операции;
opr1, opr2 - поля операндов;
des - адрес назначения;
tag - тег поступления данных (2 бита)



а)

	opc	opr1	opr2	des	tag
	car			l	
l	dup			a1, b1	0 ,1
a	null			b2	0 ,1
b	TF			i, c1	0 ,1
c	dup			d1, e1	0 ,1
d	car			f1	0 ,1
e	cdr			j	0 ,1
f	cons			k	0 ,1

б)

Рисунок 4. Пример формы хранения потоковой программы: а) – потоковый граф; б) – память функционального блока

Структура ПВМ

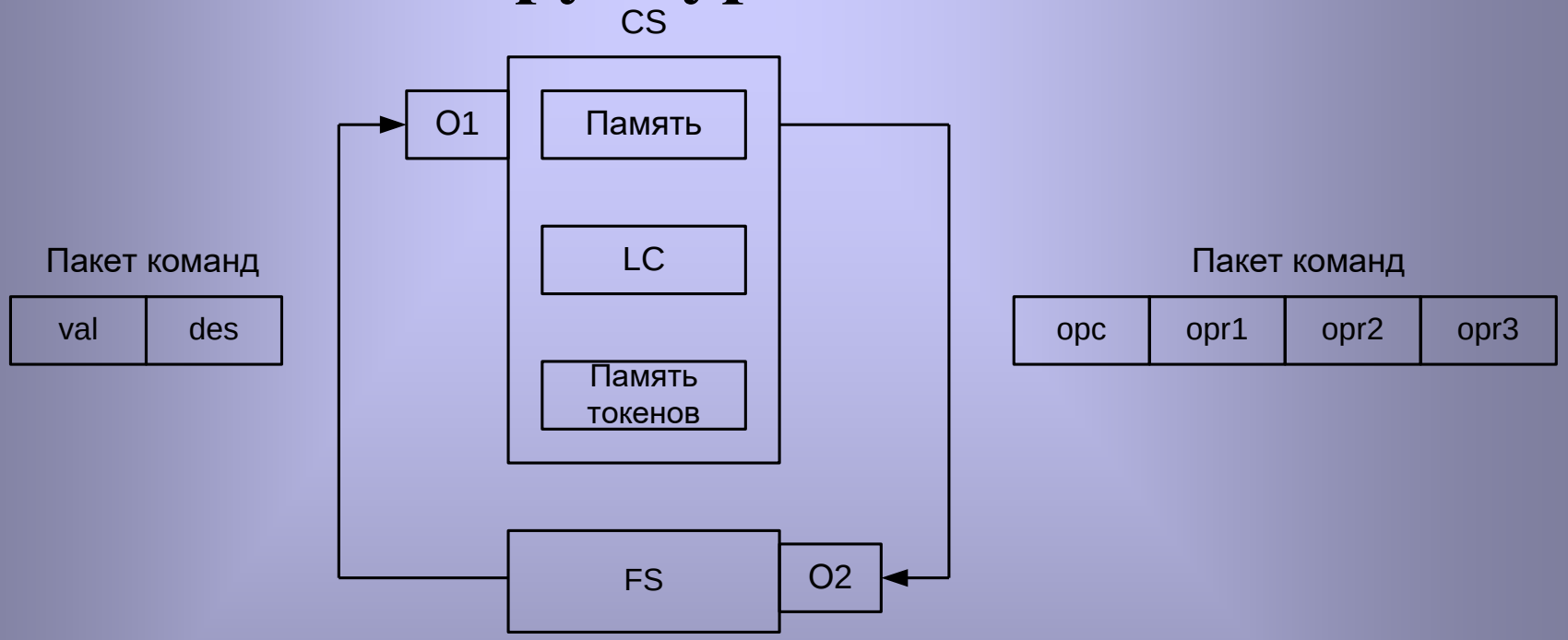


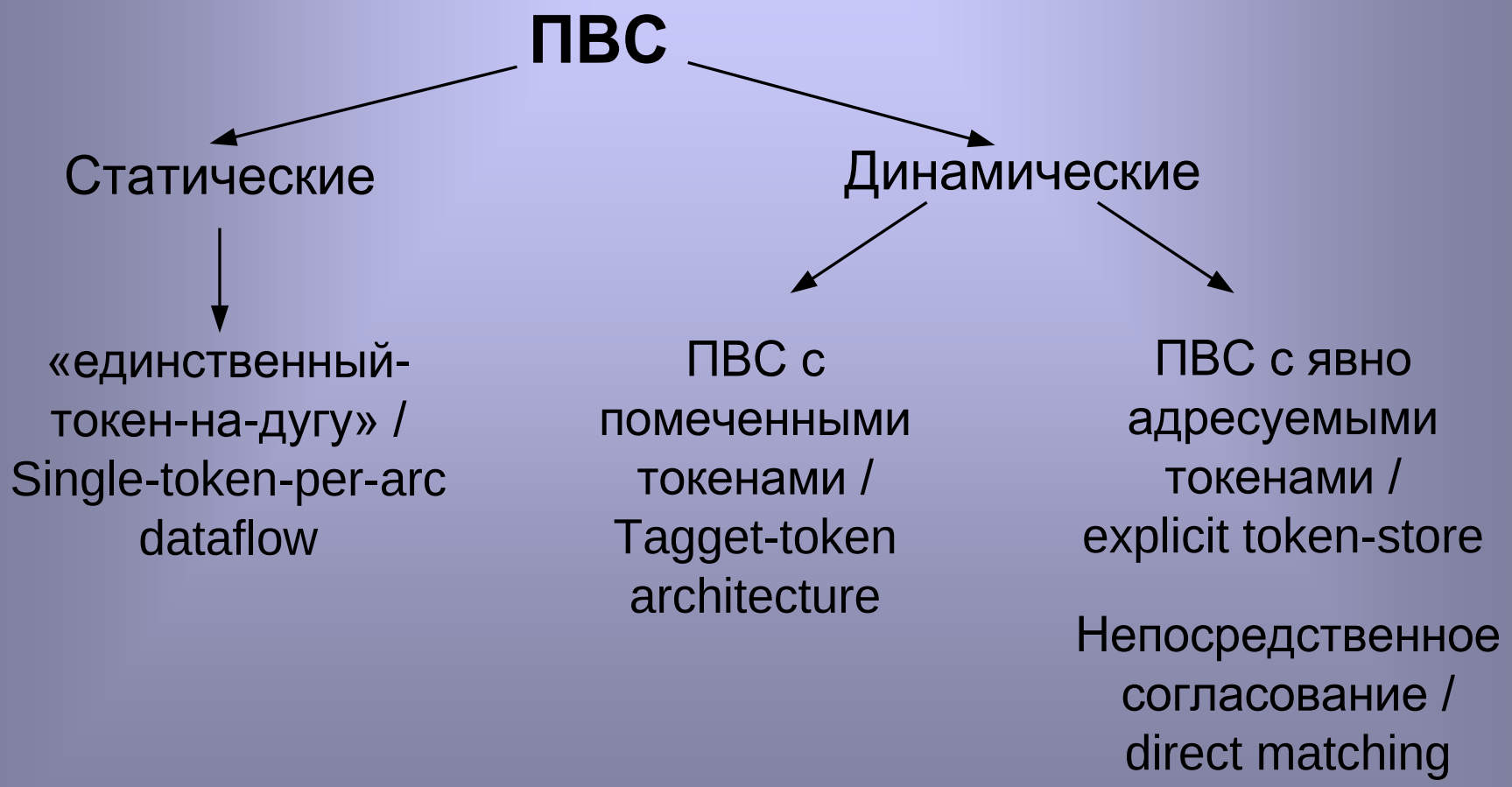
Рисунок 5. Структура потоковой вычислительной машины

CS - блок управления, где хранится потоковый граф, который используется для выборки обрабатываемых команд,

FS - функциональный блок, выполняющий команду, переданную из CS, и возвращающий результат ее выполнения в CS.

LC - блок управления загрузкой при активировании определенной функции загружает из памяти программ код этой функции

Классификация ПВС



Статические ПВС

Правило активации узла: вершина активируется, когда на всех ее входных дугах присутствует по токену и ни на одном из ее выходов токенов нет => В ней допускается присутствие на ребре графа не более чем одного токена.

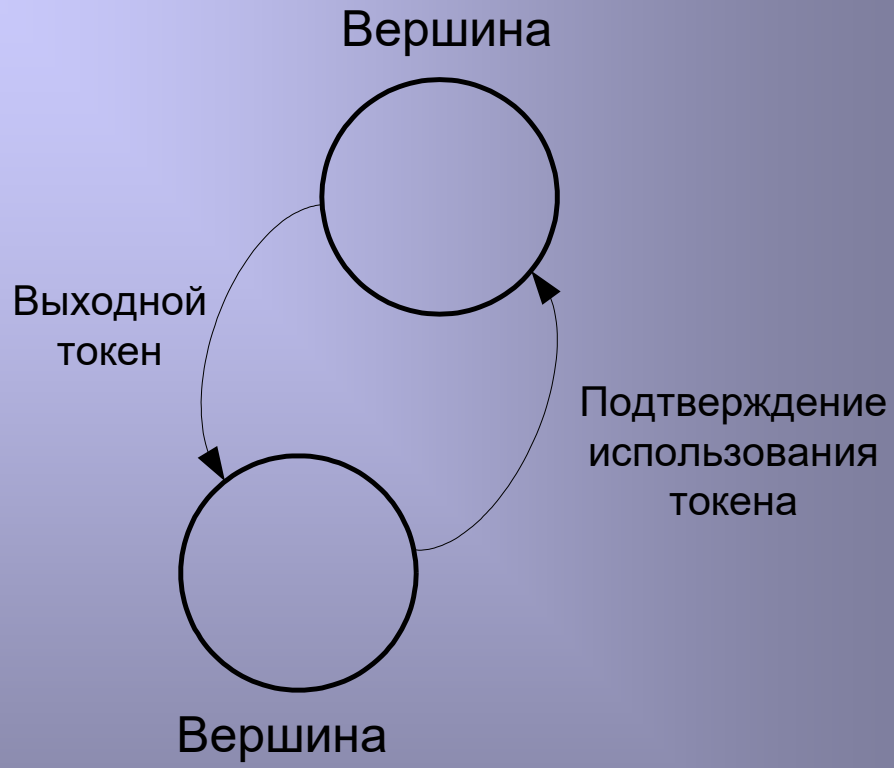


Рисунок 6. Механизм подтверждения с квитированием

Структура процессорного элемента типовой статической ПС

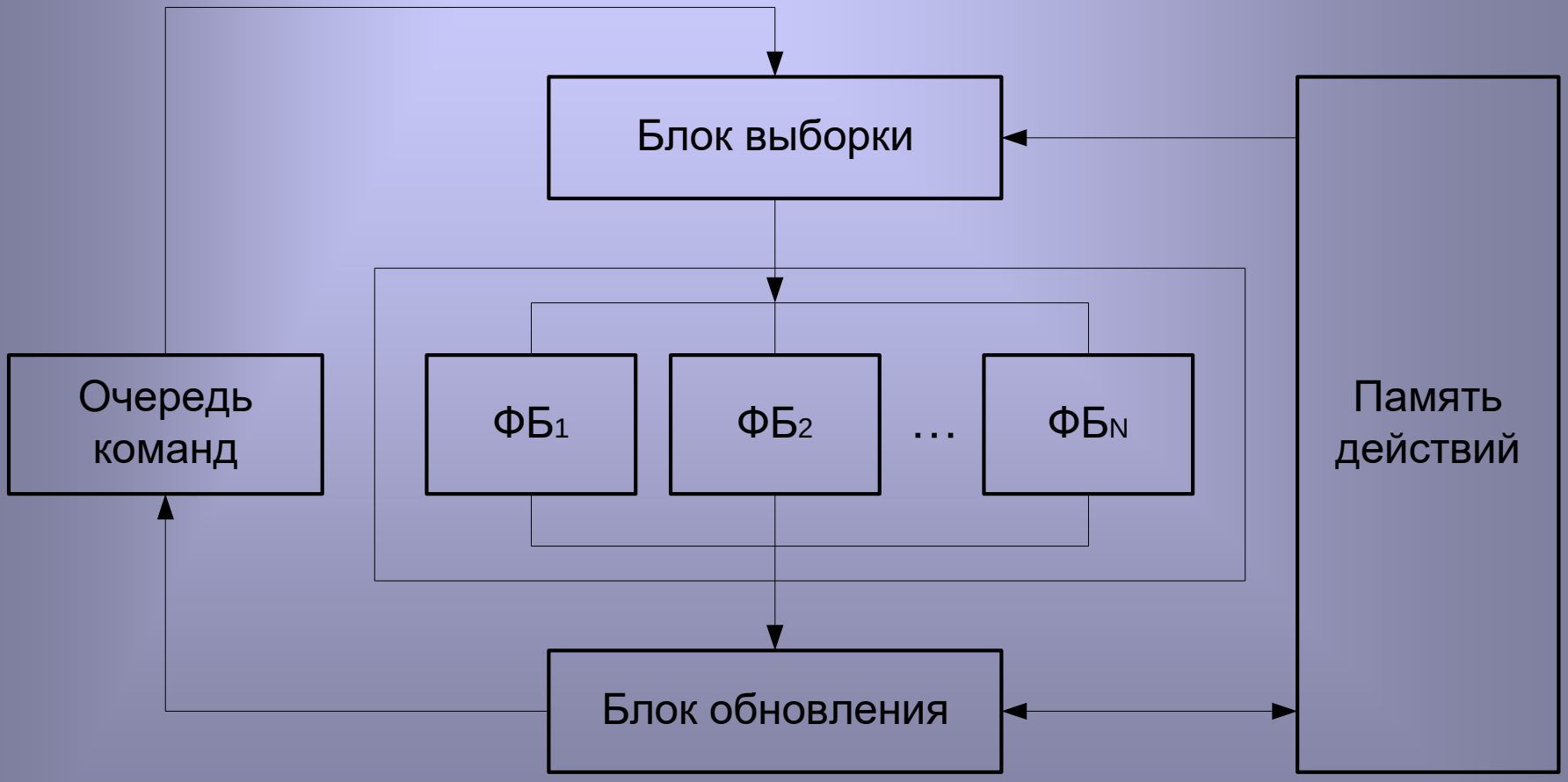
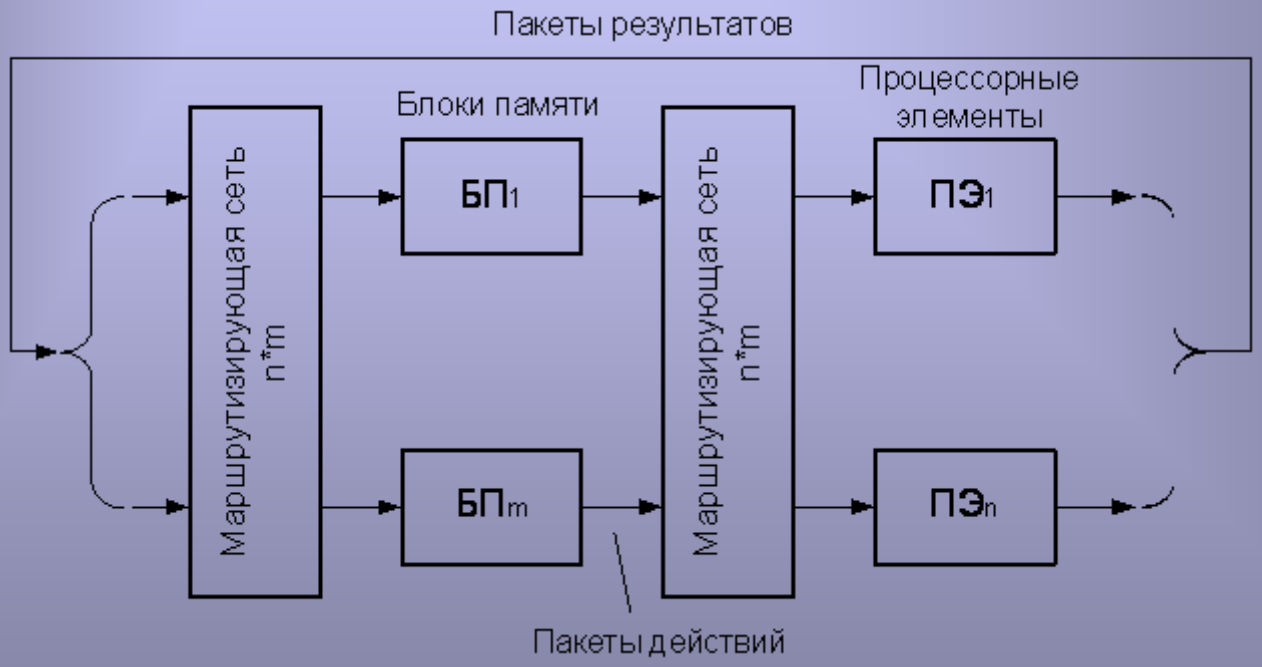


Рисунок 7. Структура процессорного элемента типовой статической ПС

Статическая потоковая архитектура по Деннису

Предложена Деннисом в 1975 году.

- *пакеты действий* (activity templates) в виде: Код операции • Операнды • Адресат.
- *Пакет результата* имеет вид: Значение • Адресат.



LAU System, TI`s Distributed Data Processor, DDMI Utah Data Driven Machine, Nec Image Pipelined Processor, Hughes Dataflow Multiprocessor.

Динамические ПВС. Архитектура с помеченными токенами

Каждый токен содержит *тег* (фишку): адрес команды+информация, определяющая вычислительный контекст.

Правило активирования вершины : вершина активируется, когда на всех ее входных дугах присутствуют токены с идентичны-ми тегами.

Примеры: SIGMA-1, PATTSY Processor Array Tagged-Token System, NTT`s Dataflow Processor Array System, DDDP Distributed Data Driven Processor, SDFA Statelles Data-Flow Architecture.

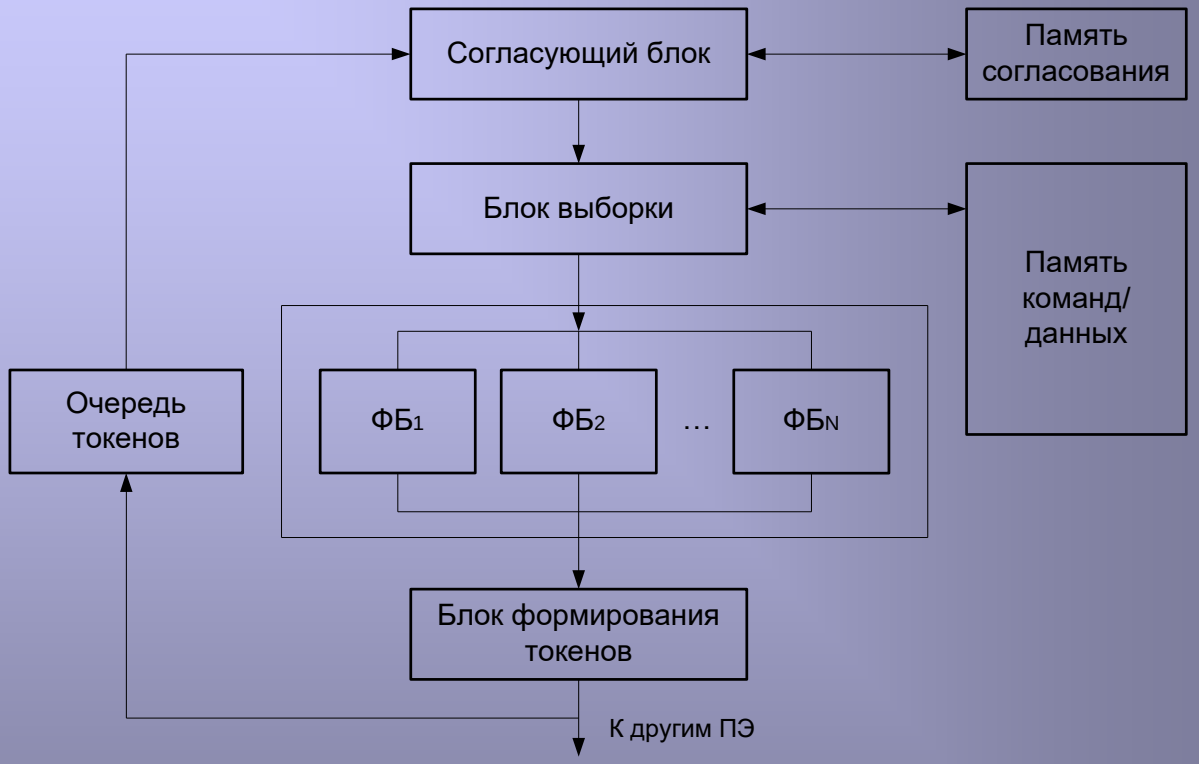


Рисунок 10. Структура процессорного элемента типовой ПС с помеченными токенами

Динамические ПВС. Архитектура с явно адресуемыми токенами

Любое вычисление полностью описывается *указателем команды* (IP, Instruction Pointer) и *указателем кадра* (FP, Frame Pointer).
Этот кортеж <FP, IP> входит в тег токена, а сам токен выглядит следующим образом: Значение • FP.IP.

Команды, реализующие потоковый граф, хранятся в памяти команд и имеют формат: Код операции • Индекс в памяти кадров • Адресат.

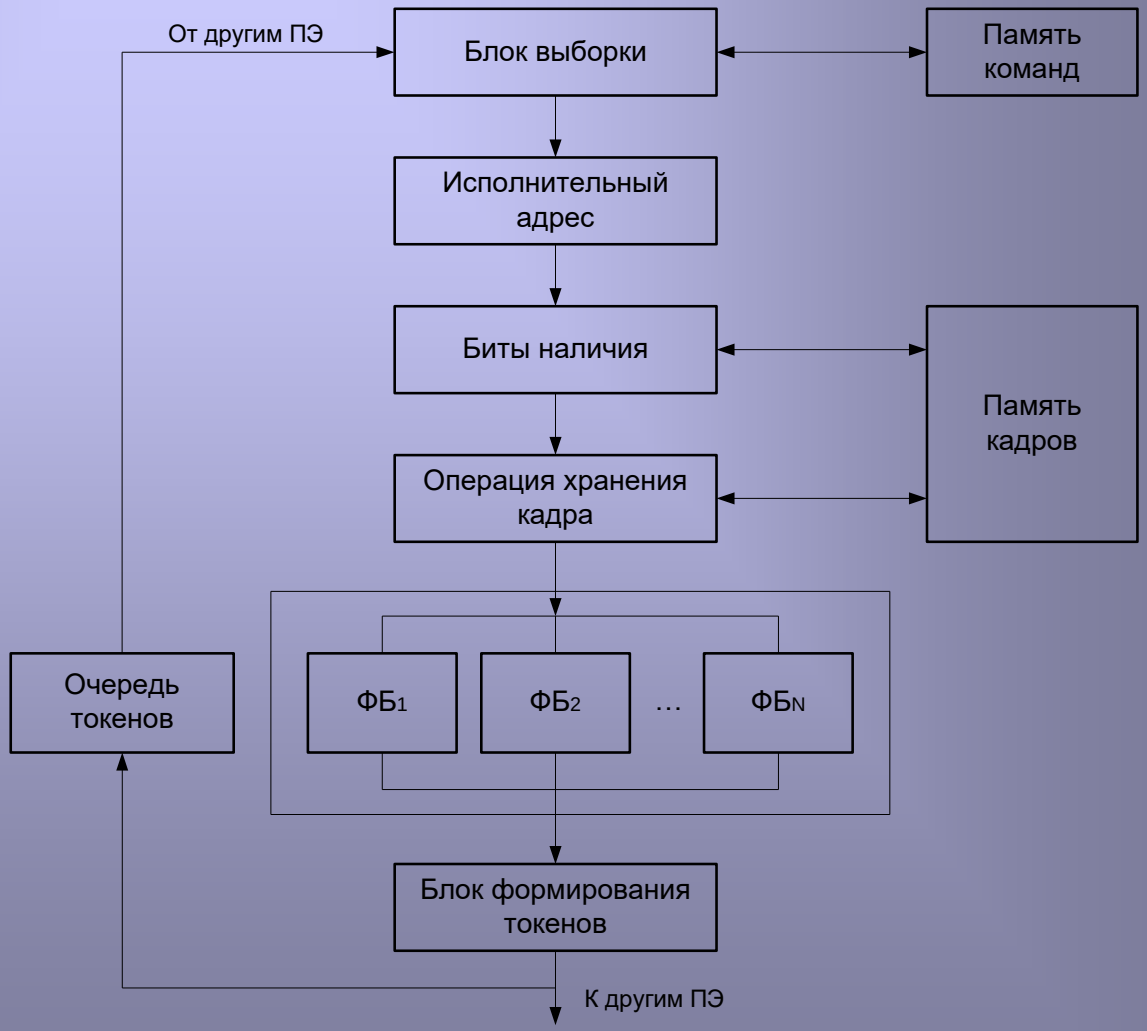


Рисунок 11. Структура процессорного элемента типовой ПС с явной адресацией токенов

Новые принципы организации вычислительных процессов высокого параллелизма

В отделе «Проблем построения информационно-вычислительных систем высокого параллелизма» Института проблем информатики Российской академии наук под руководством академика В.С. Бурцева были разработаны новые принципы организации вычислительного процесса и на их основе создана вычислительная система с автоматическим распределением ресурсов (ВСАРР), использующая гибридную модель вычислений, управляемых потоком данных, с динамически формируемым контекстом.

Всеволод Сергеевич Бурцев.
(11.02.1927 – 14.06.2005)

Под руководством В. С. Бурцева впервые в СССР в 1953 г. осуществлен съем данных с радиолокационной станции в дискретном виде, что впоследствии было широко использовано при создании системы противоракетной обороны (ПРО). Созданный молодым коллективом В. С. Бурцева вычислительный комплекс был включен в полигонную систему ПРО, и при испытаниях впервые в мире было осуществлено уничтожение баллистической ракеты (1961 г.). За рубежом такой опыт был повторен спустя более чем 20 лет.

В. С. Бурцев был Главным конструктором многопроцессорных вычислительных комплексов "Эльбрус-1", "Эльбрус-2" и 5Э26, принятых на вооружение и действующих в системах ПРО, ПСО, ЦККП, С-300 и др.

Основные технические предпосылки необходимости и возможности реализации процессоров, в которых используются новые принципы организации вычислительных процессов

1. Фактически достигнут физический предел увеличения производительности одного микропроцессора.

2. Основным методом увеличения производительности вычислительных средств (ВС) становится метод распараллеливания вычислительных процессов.

3. На математическом уровне задача распараллеливания вычислительного процесса успешно решена: векторно-матричные вычисления.

4. Фон-неймановский принцип организации вычислительного процесса, который сводится к последовательному выполнению команд, не может эффективно работать с задачами, обладающими высоким параллелизмом.

5. Какой бы ни была архитектура вычислительных систем, программист должен решать две задачи (решить которые ему не под силу):

a) распределить вычислительные процессы по процессорам или машинам так, чтобы они были равномерно загружены;

b) обеспечить синхронизацию вычислительного процесса так, чтобы данные одного процессора поступали на вход другого в соответствии с алгоритмом, определенным задачей.

6. Как правило, с увеличением числа микропроцессоров или машин в комплексе, эффективное их использование падает.

Производитель ВС, как правило, рекламирует максимальную производительность (Pmax), которая вычисляется, как Pmax = P*N и умалчивает о реальной производительности P реал, которая, не может быть подсчитана по формуле P*N.

В течение 10-ти лет велись работы над решением проблемы увеличения производительности одного процессора при решении задач обладающих высоким параллелизмом.

Работы в 2000 году закончились построением макета, показывающим возможность создания процессора, производительностью выше 1терафлоп в секунду. В настоящее время в связи с бурным развитием микроэлектроники можно говорить о построении такого микропроцессора на одном или нескольких кристаллах.

Несколькими зарубежными фирмами решен вопрос изготовления ассоциативной памяти объемом в 128 тыс. 72-х разрядных слов (ключей), необходимой для построения процессора на новых принципах.

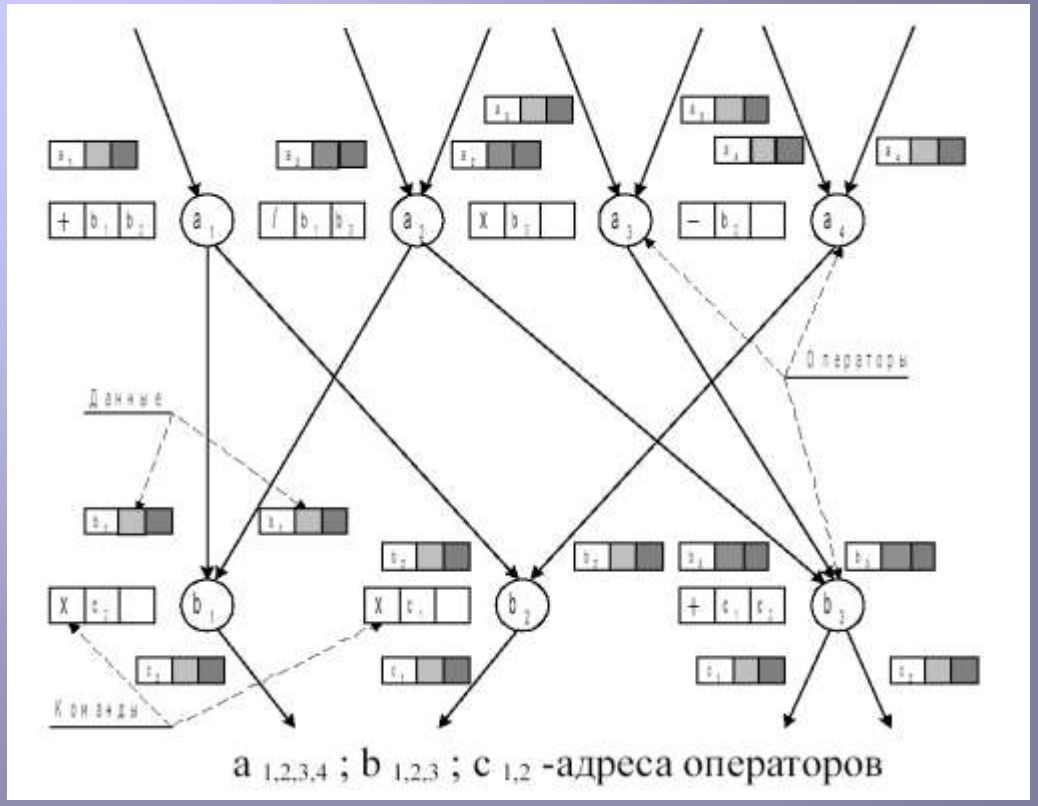
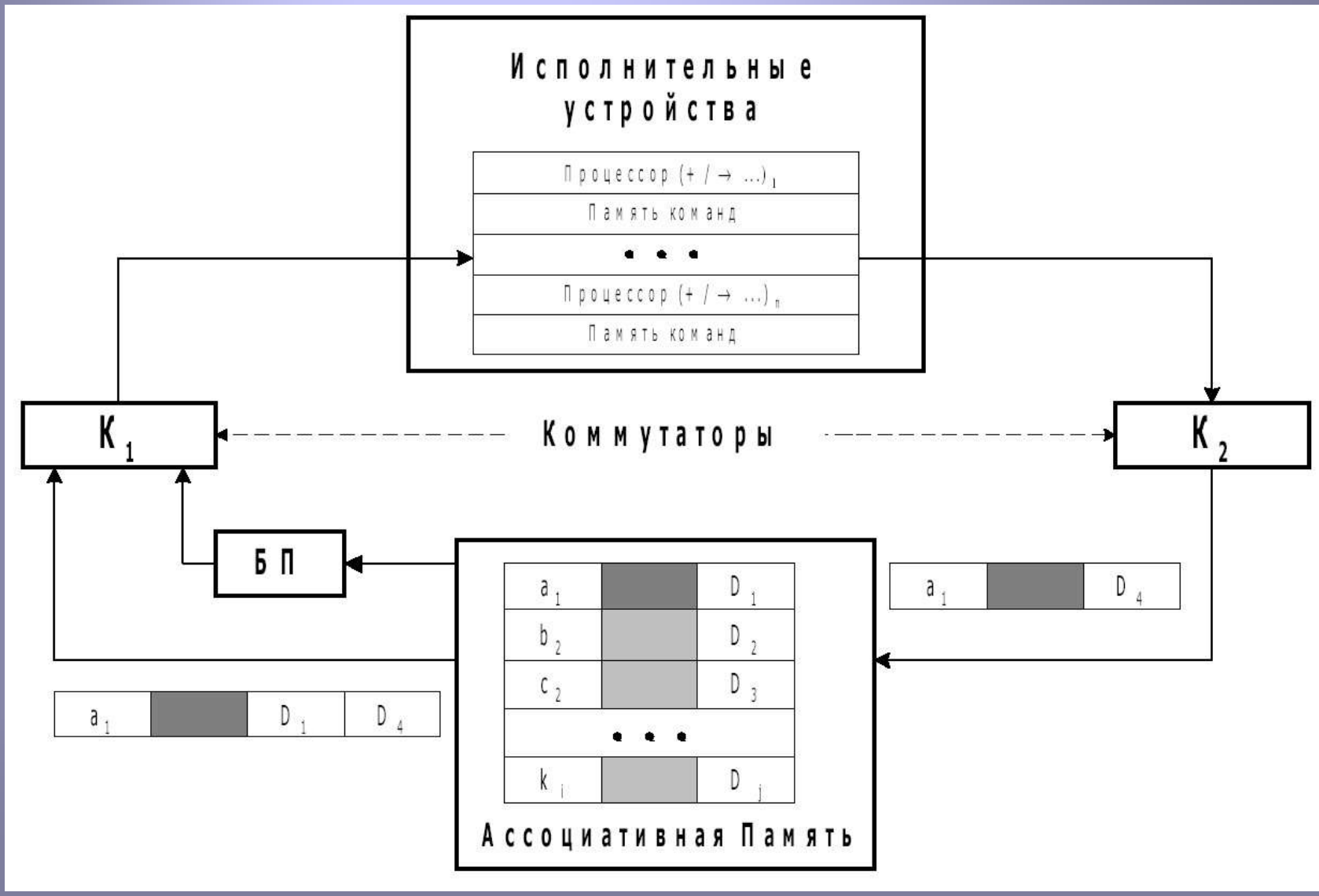


Рисунок 12. Граф вычислительного процесса

Схема высокопроизводительного процессора параллельного счета новой архитектуры



Особенности рассматриваемой архитектуры

1. На аппаратном уровне реализуется практически полный параллелизм вычислительных процессов выполняемой задачи (1011 - 1013 оп/с в одном чипе).

2. Программист исключен из распределения ресурсов вычислительных средств.

3. Обеспечивается высокая структурная надежность и технологичность системы.

4. Из АП на исполнительные устройства поступают не связанные между собой пары данных для исполнения операций.

5. АП берет на себя функцию адресной арифметики и целый ряд других функций, управленческого характера, существенно упрощая написание программ и повышая производительность процессоров: от 1.5 до 5 раз, в зависимости от задачи.

6. Процессор, при работе в масштабе реального времени практически не нуждается в традиционном регистре прерываний, реагируя практически мгновенно на сигналы, приходящие в масштабе реального времени.

Моделирование и реализация потоковой машины на вычислительном кластере

Имеется реализованная на персональной ЭВМ программная модель ОСВМ, позволяющая успешно исследовать выполнение задач небольшого размера. Для повышения скорости моделирования ОСВМ использовались кластера ТКС-7.

ТКС-7 – восьмипроцессорная система на базе Pentium III 800 МГц с коммуникационной сетью SCI 2D тор, использующей адаптеры Dolphin D311/312.

Передача данных между блоками осуществляется с помощью межпроцессных пересылок функциями MPI.

На кластере ТКС-7 реализована упрощенная модель ОСВМ, были решены задачи нахождения максимального числа и перемножения матриц.

Программный комплекс системного сопровождения для ВСАРР

СПО ВСАРР должно:

1. поддерживать многозадачный многопользовательский режим работы ВС,
2. удаленный доступ к ВС,
3. механизмы интеграции ВС в традиционную вычислительную среду и т.д.

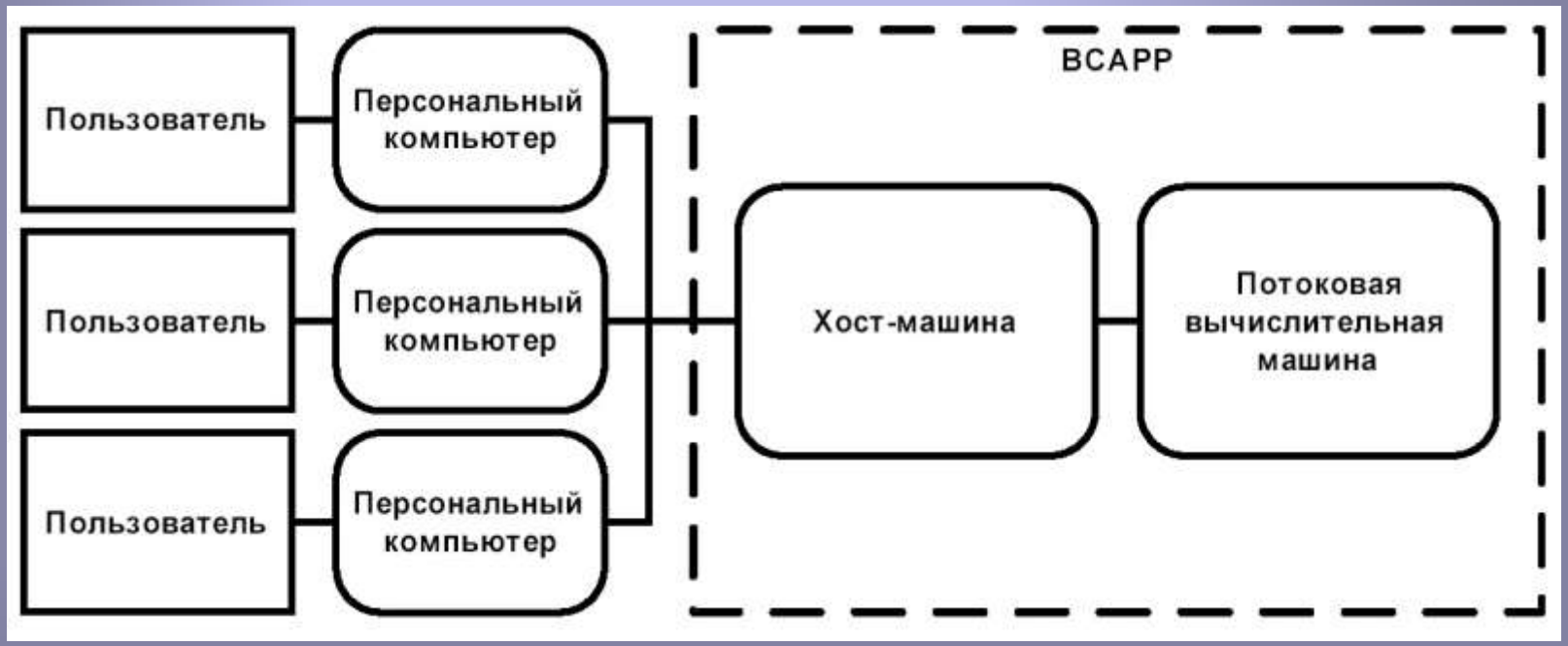


Рисунок 14. Схема доступа пользователей к ВСАРР

Программный комплекс системного сопровождения для ВСАРР

Данный программный комплекс должен предоставлять пользователям следующие основные возможности:

☑ Доступ к ВСАРР с помощью локальной или глобальной сети со своего персонального компьютера.

☑ Хранение файлов с потоковыми программами и файлов с исходными данными и результатами **вычислений** в персональном рабочем пространстве и обмен информацией с другими пользователями.

☑ Запуск потоковых программ на ВСАРР с определенными исходными данными и получение результатов вычислений.

☑ Программный комплекс должен включать в себя три части:

- Резидентная часть – выполняется на потоковой вычислительной машине, включает в себя обработчики нештатных ситуаций и различные библиотеки.
- Операционная часть – выполняется на хост-машине, осуществляет обработку различных видов пользовательских запросов и контролирует использование ресурсов хост-машины и потоковой вычислительной машины.
- Клиентская часть – выполняется на персональном компьютере пользователя, организует взаимодействие между пользователем и операционной частью программного комплекса.

Структура программного комплекса

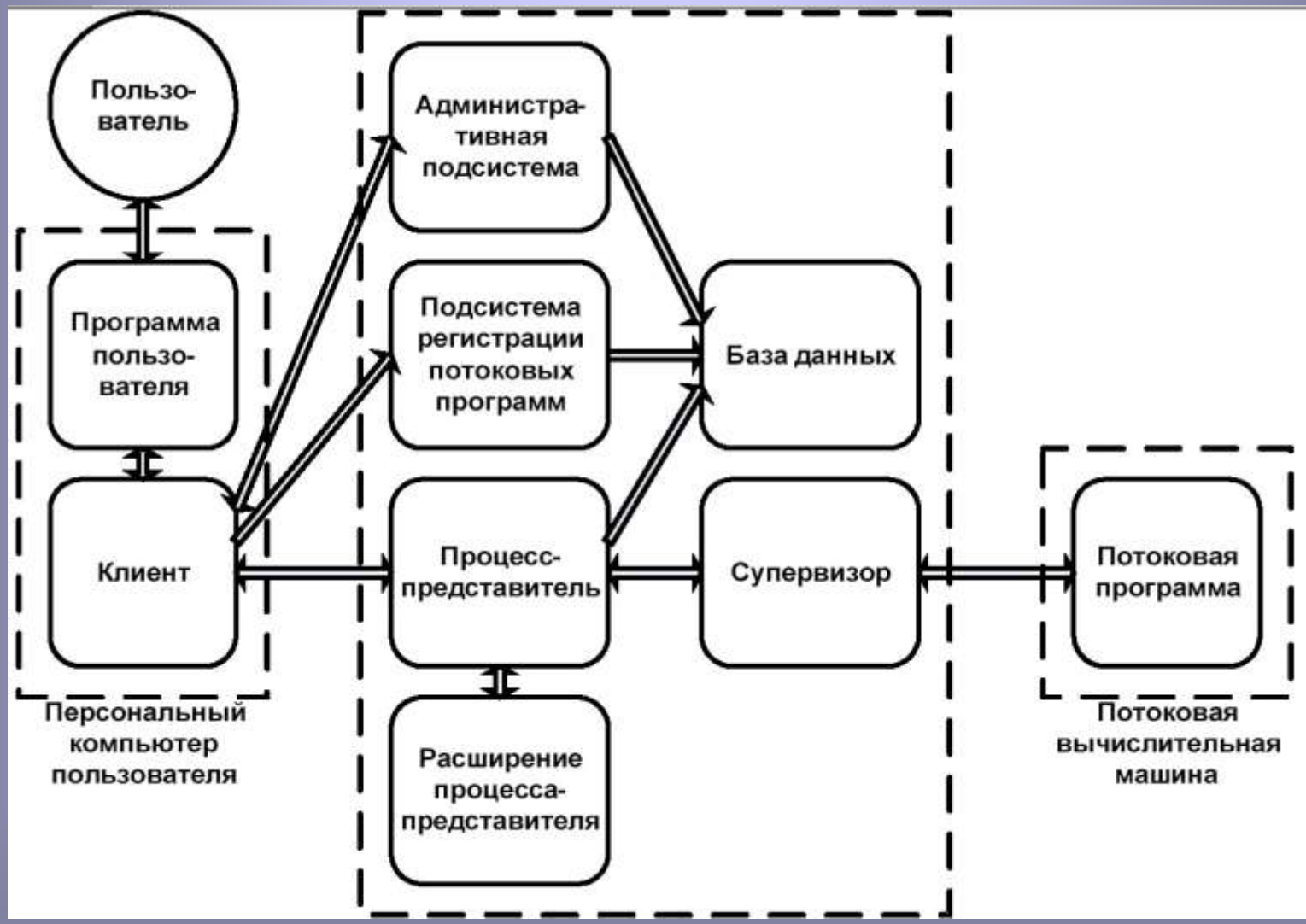


Рисунок 15. Структура программного комплекса

Язык программирования для модели вычислений, основанной на принципе потока данных DCF

Язык DCF (Data-Control Flow) проектируется как расширение языка последовательного программирования Си специальными возможностями для представления параллелизма в модели dataflow.

Нить (*thread*) – это набор инструкций, выполняемых последовательно, как в модели фон Неймана, а сами нити могут выполняться параллельно и запускаются при готовности своих входных аргументов, как в модели dataflow.

Неформально, возможности языка Си предлагается использовать для программирования вычислений, составляющих тела нитей, а специальные операторы и описания, составляющие его расширение – для описания dataflow-механизмов взаимодействия нитей.

Основные термины и понятия языка DCF

Нить - цепочка операций, выполняемых в модели вычислений фон Неймана. При этом различные нити могут выполняться параллельно. Запуск нити происходит при появлении готовых значений всех входных параметров нити.

Например:

thread Func(int N, float Var) { … }

Можно запустить нить, послав токены, содержащие значения всех аргументов, на имя функции-нити.

Запуск нитей, синхронизация нитей и передача значений между ними осуществляется с помощью токенов - единиц данных для dataflow управления.

Токен - объект, содержащий тег, идентифицирующий возможного получателя токена, и данные некоторых типов языка Си: int, float, char, адрес и др.

Содержит два компонента: имя функции-назначения (функции-нити или request-функции), для которой предназначен токен, и некоторую характеристику, называемую цветом.

Токены порождаются при выполнении специальной стандартной функции.

Структура программы

Программа на языке DCF - набор функций, некоторые из которых имеют спецификатор thread.

С помощью любой функции-нити можно породить произвольное количество экземпляров нити.

В программе среди описаний функций-нитей обязательно должно быть описание функции-нити с именем main. Запуск программы состоит в активации этой функции-нити. Функция-нить main может иметь аргументы, через которые программа получает параметры из внешней среды выполнения. Повторная активация функции-нити main запрещена.

Таким образом, в любой момент времени выполнения программы происходит параллельное выполнение множества нитей, запущенных с помощью активации функций-нитей.

Программа завершается, когда завершаются все нити.

Заключение

Проекты ПВС:

☑ Манчестерскую ВС (Манчестерский университет),
☑ Tagged Token,
☑ Monsoon (Массачусетский технологический институт),
☑ Sigma,
☑ EMS,
☑ EMC-4 (Тсукуба, Япония),
☑ RAPID (проект Sharp - Mitsubishi - университет Осаки, Япония) и др.

Как утверждает ряд авторов, многие интересные проекты Data Flow не были по достоинству оценены производителями вычислительной техники, в силу нетрадиционности подхода.

С другой стороны, потоковые вычисления нашли применение в современных суперскалярных процессорах.

В качестве примеров таких микропроцессоров можно привести HP PA-8000 и Intel Pentium Pro.

Так что системы, управляемые потоком данных, так или иначе, находят применение в современной вычислительной технике.