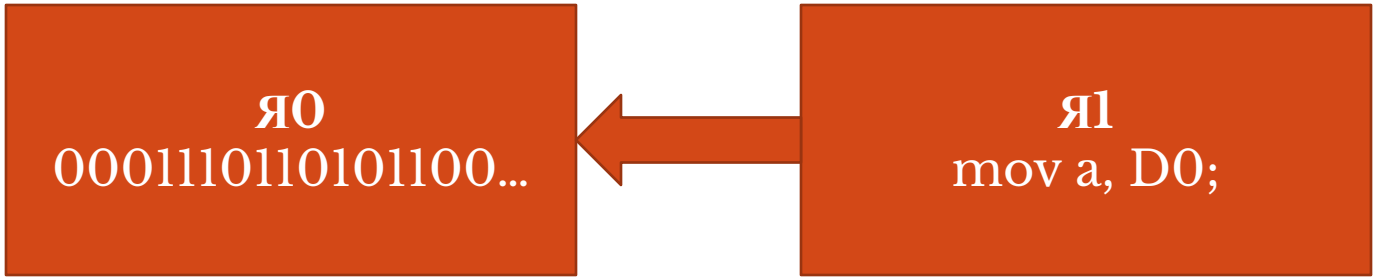
Архитектура ЭВМ и периферийных устройств

Основы

Языки, уровни и виртуальные машины



Первый способ выполнения программы, написанной на языке Я 1, — замена каждой команды на эквивалентный набор команд в языке Я 0. В этом случае компьютер выполняет новую программу, написанную на языке Я 0, вместо старой программы, написанной на Я 1. Эта технология называется **трансляцией.**

Второй способ — написание программы на языке Я 0, которая берет программы, написанные на языке Я 1, в качестве входных данных, рассматривает каждую команду по очереди и сразу выполняет эквивалентный набор команд языка Я 0. Эта технология не требует составления новой программы на Я 0. Она называется **интерпретацией, а программа, которая осуществляет интерпретацию, называется интерпретатором.**

Обычно гораздо проще представить себе существование гипотетического компьютера или **виртуальной машины, для которой машинным языком является язык** Я 1, чем думать о трансляции и интерпретации. Назовем такую виртуальную машину М 1, а виртуальную машину с языком Я 0 — М 0.

Рис. 1.1. Многоуровневая машина

Между языком и виртуальной машиной существует важная зависимость. У каждой машины есть какой-то определенный машинный язык, состоящий из всех команд, которые эта машина может выполнять. В сущности, машина определяет язык. Сходным образом язык определяет машину, которая может выполнять все программы, написанные на этом языке. Машину, задающуюся определенным языком, очень сложно и дорого сконструировать из электронных схем, но мы можем представить себе такую машину.

Современные многоуровневые машины

Рис. 1.2. Компьютер с шестью уровнями. Способ поддержки каждого уровня указан под ним. В скобках указывается название поддерживающей программы

Уровень 0

Уровень 0 — аппаратное обеспечение машины. Его электронные схемы выполняют программы, написанные на языке уровня 1. Ради полноты нужно упомянуть о существовании еще одного уровня, расположенного ниже уровня 0. Этот уровень не показан на рис. 1.2, так как он попадает в сферу электронной техники и, следовательно, не рассматривается в этой книге. Он называется **уровнем физических устройств.** На этом уровне находятся транзисторы, которые являются примитивами для разработчиков компьютеров. Объяснять, как работают транзисторы, — задача физики.

На самом нижнем уровне, **цифровом логическом уровне,** объекты называются **вентилями.** Хотя вентили состоят из аналоговых компонентов, таких как транзисторы, они могут быть точно смоделированы как цифровые средства. У каждого вентиля есть одно или несколько цифровых входных данных (сигналов, представляющих 0 или 1). Вентиль вычисляет простые функции этих сигналов, такие как И или ИЛИ. Каждый вентиль формируется из нескольких транзисторов. Несколько вентилей формируют 1 бит памяти, который может содержать 0 или 1. Биты памяти, объединенные в группы, например, по 16,32 или 64, формируют регистры. Каждый регистр может содержать одно двоичное число до определенного предела. Из вентилей также может состоять сам компьютер.

Уровень 1

Следующий уровень — **микроархитектурный уровень.** На этом уровне можно видеть совокупности 8 или 32 регистров, которые формируют локальную память и схему, называемую **АЛУ (арифметико-логическое устройство).** АЛУ выполняет простые арифметические операции. Регистры вместе с АЛУ формируют **тракт данных,** по которому поступают данные. Основная операция тракта данных состоит в следующем. Выбирается один или два регистра, АЛУ производит над ними какую-либо операцию, например сложения, а результат помещается в один из этих регистров.

На некоторых машинах работа тракта данных контролируется особой программой, которая называется **микропрограммой.** На других машинах тракт данных контролируется аппаратными средствами.

Уровень 2

Второй уровень мы будем называть **уровнем архитектуры системы команд.** Каждый производитель публикует руководство для компьютеров, которые он продает, под названием ≪Руководство по машинному языку≫ или ≪Принципы работы компьютера Western Wombat Model 100X≫ и т. п. Такие руководства содержат информацию именно об этом уровне. Когда они описывают набор машинных команд, они в действительности описывают команды, которые выполняются микропрограммой-интерпретатором или аппаратным обеспечением. Если производитель поставляет два интерпретатора для одной машины, он должен издать два руководства по машинному языку, отдельно для каждого интерпретатора.

Уровень 3

Следующий уровень обычно гибридный. Большинство команд в его языке есть также и на уровне архитектуры системы команд (команды, имеющиеся на одном из уровней, вполне могут находиться на других уровнях). У этого уровня есть некоторые дополнительные особенности: набор новых команд, другая организация памяти, способность выполнять две и более программ одновременно и некоторые другие. При построении третьего уровня возможно больше вариантов, чем при построении первого и второго.

Новые средства, появившиеся на третьем уровне, выполняются интерпретатором, который работает на втором уровне. Этот интерпретатор был когда-то назван операционной системой. Команды третьего уровня, идентичные командам второго уровня, выполняются микропрограммой или аппаратным обеспечением, но не операционной системой. Иными словами, одна часть команд третьего уровня интерпретируется операционной системой, а другая часть — микропрограммой. Вот почему этот уровень считается гибридным. Мы будем называть этот уровень **уровнем операционной системы.**

Уровень 4

Между третьим и четвертым уровнями есть существенная разница. Нижние три уровня конструируются не для того, чтобы с ними работал обычный программист. Они изначально предназначены для работы интерпретаторов и трансляторов, поддерживающих более высокие уровни. Эти трансляторы и интерпретаторы составляются так называемыми **системными программистами,** которые специализируются на разработке и построении новых виртуальных машин. Уровни с четвертого и выше предназначены для прикладных программистов, решающих конкретные задачи.

Четвертый уровень представляет собой символическую форму одного из языков более низкого уровня. На этом уровне можно писать программы в приемлемой для человека форме. Эти программы сначала транслируются на язык уровня 1, 2 или 3, а затем интерпретируются соответствующей виртуальной или фактически существующей машиной. Программа, которая выполняет трансляцию, называется **ассемблером.**

Уровень 5

Пятый уровень обычно состоит из языков, разработанных для прикладных программистов. Такие языки называются **языками высокого уровня.** Существуют сотни языков высокого уровня. Наиболее известные среди них — BASIC, C, C++, Java, LISP и Prolog. Программы, написанные на этих языках, обычно транслируются на уровень 3 или 4. Трансляторы, которые обрабатывают эти программы, называются **компиляторами.** Отметим, что иногда также используется метод интерпретации. Например, программы на языке Java обычно интерпретируются.

Рис. 1.2. Компьютер с шестью уровнями. Способ поддержки каждого уровня указан под ним. В скобках указывается название поддерживающей программы

Процессоры

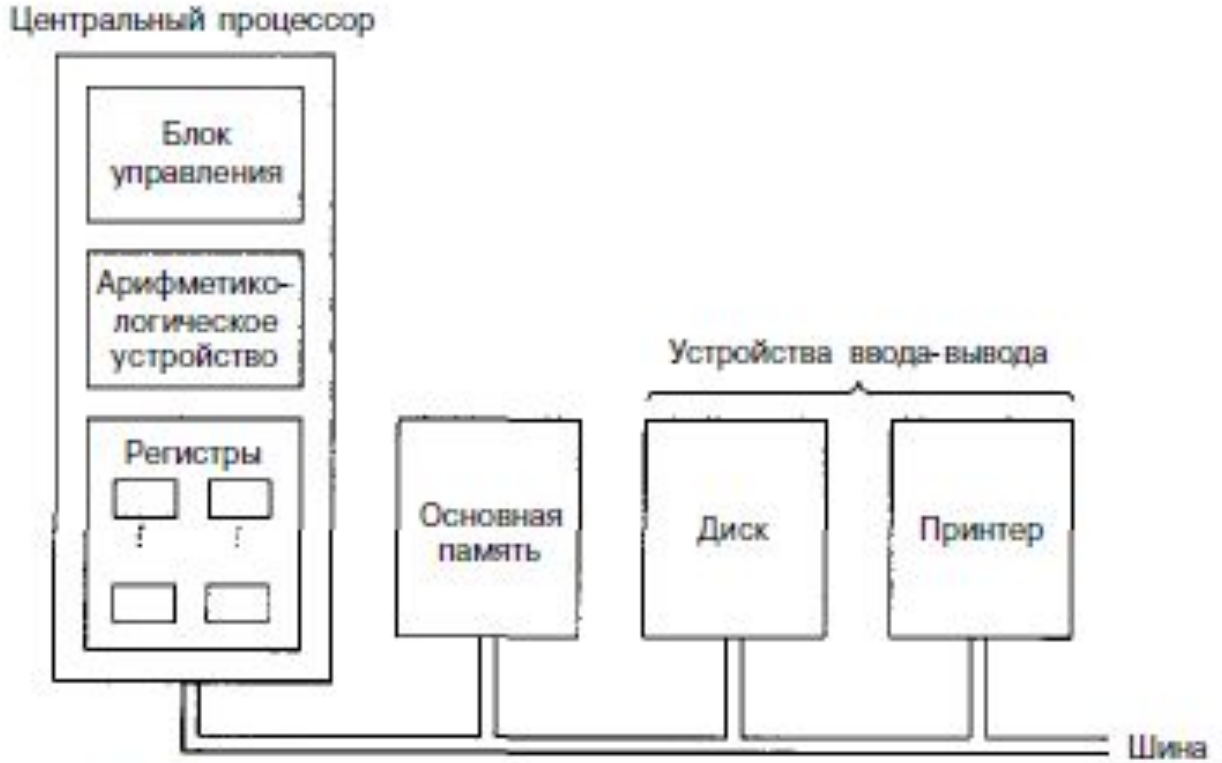


Рис. 2.1. Схема устройства компьютера с одним центральным процессором и двумя устройствами ввода-вывода

На рис. 2.1 показано устройство обычного компьютера. **Центральный процессор —** это мозг компьютера. Его задача — выполнять программы, находящиеся в основной памяти. Он вызывает команды из памяти, определяет их тип, а затем выполняет их одну за другой. Компоненты соединены **шиной,** представляющей собой набор параллельно связанных проводов, по которым передаются адреса, данные и сигналы управления. Шины могут быть внешними (связывающими

Процессор состоит из нескольких частей. Блок управления отвечает за вызов команд из памяти и определение их типа. Арифметико-логическое устройство выполняет арифметические операции (например, сложение) и логические операции (например, логическое И).
Внутри центрального процессора находится память для хранения промежуточных результатов и некоторых команд управления. Эта память состоит из нескольких регистров, каждый из которых выполняет определенную функцию.

Самый важный регистр — **счетчик команд,** который указывает, какую команду нужно выполнять дальше. Название ≪счетчик команд≫ не соответствует действительности, поскольку он ничего не считает, но этот термин употребляется повсеместно1.
Еще есть **регистр команд,** в котором находится команда, выполняемая в данный момент.

Внутреннее устройство процессора

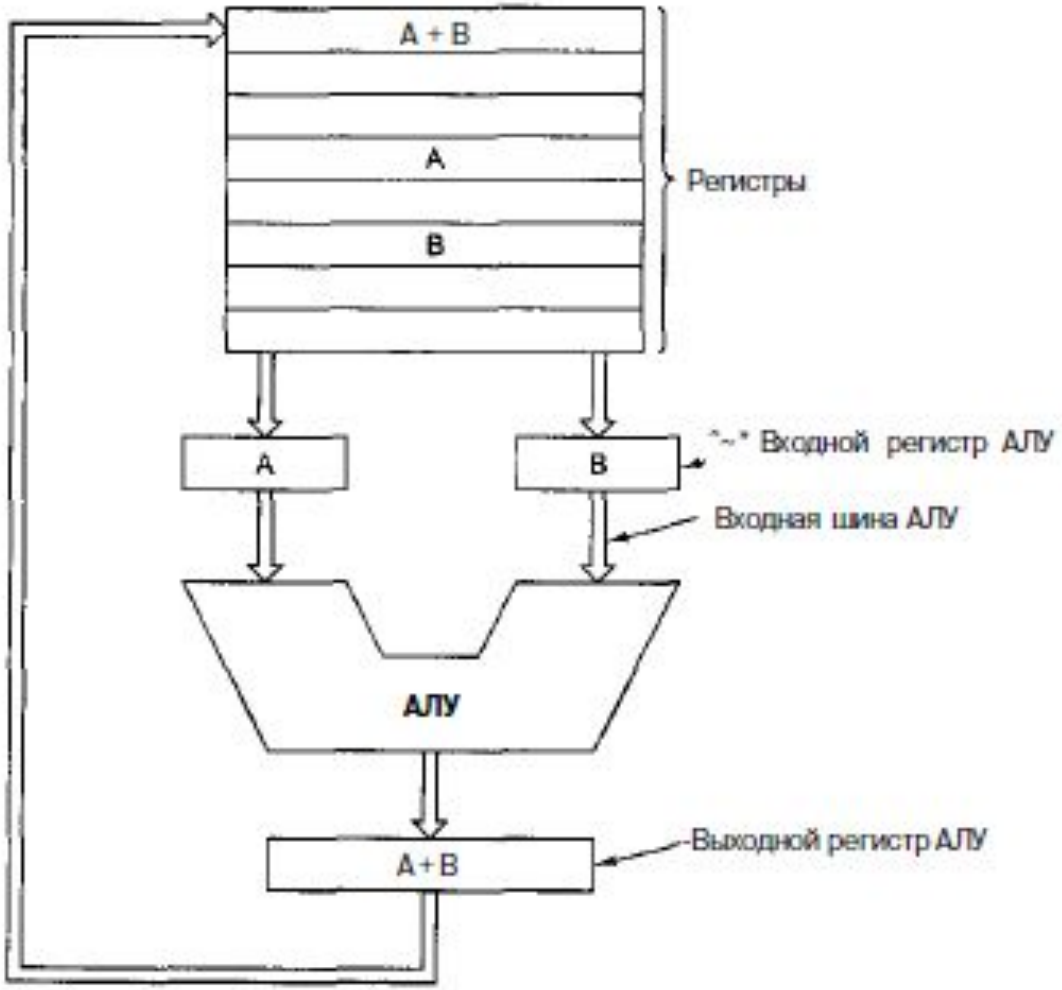


Рис. 2.2. Тракт данных в обычной фон-неймановской машине

Выполнение команд

Центральный процессор выполняет каждую команду за несколько шагов:
1) вызывает следующую команду из памяти и переносит ее в регистр команд;
2) меняет положение счетчика команд, который теперь должен указывать на следующую команду;
3) определяет тип вызванной команды;
4) если команда использует слово из памяти, определяет, где находится это слово;
5) переносит слово, если это необходимо, в регистр центрального процессора;
6) выполняет команду;
7) переходит к шагу 1, чтобы начать выполнение следующей команды.
Такая последовательность шагов **(выборка—декодирование—исполнение)** является основой работы всех компьютеров.

Микропрограммирование

Первые компьютеры содержали небольшое количество команд, и эти команды были простыми. Но поиски более мощных компьютеров привели, кроме всего прочего, к появлению более сложных команд. Вскоре разработчики поняли, что при наличии сложных команд программы выполняются быстрее, хотя выполнение отдельных команд занимает больше времени. В качестве примеров сложных команд можно назвать выполнение операций с плавающей точкой, обеспечение прямого доступа к элементам массива и т. п.

Сложные команды были лучше, потому что некоторые операции иногда перекрывались. Какие-то операции могли выполняться параллельно, для этого использовались разные части аппаратного обеспечения. Для дорогих компьютеров с высокой производительностью стоимость этого дополнительного аппаратного обеспечения была вполне оправданна. Таким образом, у дорогих компьютеров было гораздо больше команд, чем у дешевых. Однако развитие программного обеспечения и требования совместимости команд привели к тому, что сложные команды стали использоваться и в дешевых компьютерах, хотя там во главу угла ставилась стоимость, а не скорость работы.

Но как построить дешевый компьютер, который будет выполнять все сложные команды, предназначенные для высокоэффективных дорогостоящих машин? Решением этой проблемы стала **интерпретация.**

К концу 70-х годов интерпретаторы стали применяться практически во всех моделях, кроме самых дорогостоящих машин с очень высокой производительностью (например, Cray-1 и компьютеров серии Control Data Cyber). Использование интерпретаторов исключало высокую стоимость сложных команд, и разработчики могли вводить все более и более сложные команды,

Хотя самые первые 8-битные микропроцессоры были очень простыми и содержали небольшой набор команд, к концу 70-х годов даже они стали разрабатываться с использованием интерпретаторов. В этот период основной проблемой для разработчиков стала возрастающая сложность микропроцессоров. Главное преимущество интерпретации заключалось в том, что можно было разработать простой процессор, а вся сложность сводилась к созданию интерпретатора. Таким образом, разработка сложного аппаратного обеспечения замещалась разработкой сложного программного обеспечения.

RISC и CISC

RISC — это сокращение от Reduced Instruction Set Computer — компьютер с сокращенным набором команд. RISC противопоставлялся CISC (Complex Instruction Set Computer — компьютер с полным набором команд).

В итоге Intel разработала «гибрид», состоящий из RISC-ядра, напрямую выполняющего простые команды и интерпретатора, берущего на себя обработку «сложных» команд. Такая система была менее производительна, по сравнению с «чистым» RISC, однако обеспечивала совместимость с софтом, написанным для CISC и давала ощутимый прирост производительности по сравнению с последним.

Принципы современных архитектур

Все команды непосредственно выполняются аппаратным обеспечением.

Компьютер должен начинать выполнение большого числа команд.

Команды должны легко **декодироваться.**

К памяти должны **обращаться только команды загрузки и сохранения.**

Должно быть большое количество регистров.

Кэширование

Уже много лет известно, что главным препятствием высокой скорости выполнения команд является их вызов из памяти. Для разрешения этой проблемы разработчики придумали средство для вызова команд из памяти заранее, чтобы они имелись в наличии в тот момент, когда будут необходимы. Эги команды помещались в набор регистров, который назывался **буфером выборки с упреждением.** Таким образом, когда была нужна определенная команда, она вызывалась прямо из буфера, и не нужно было ждать, пока она считается из памяти.

Конвейер

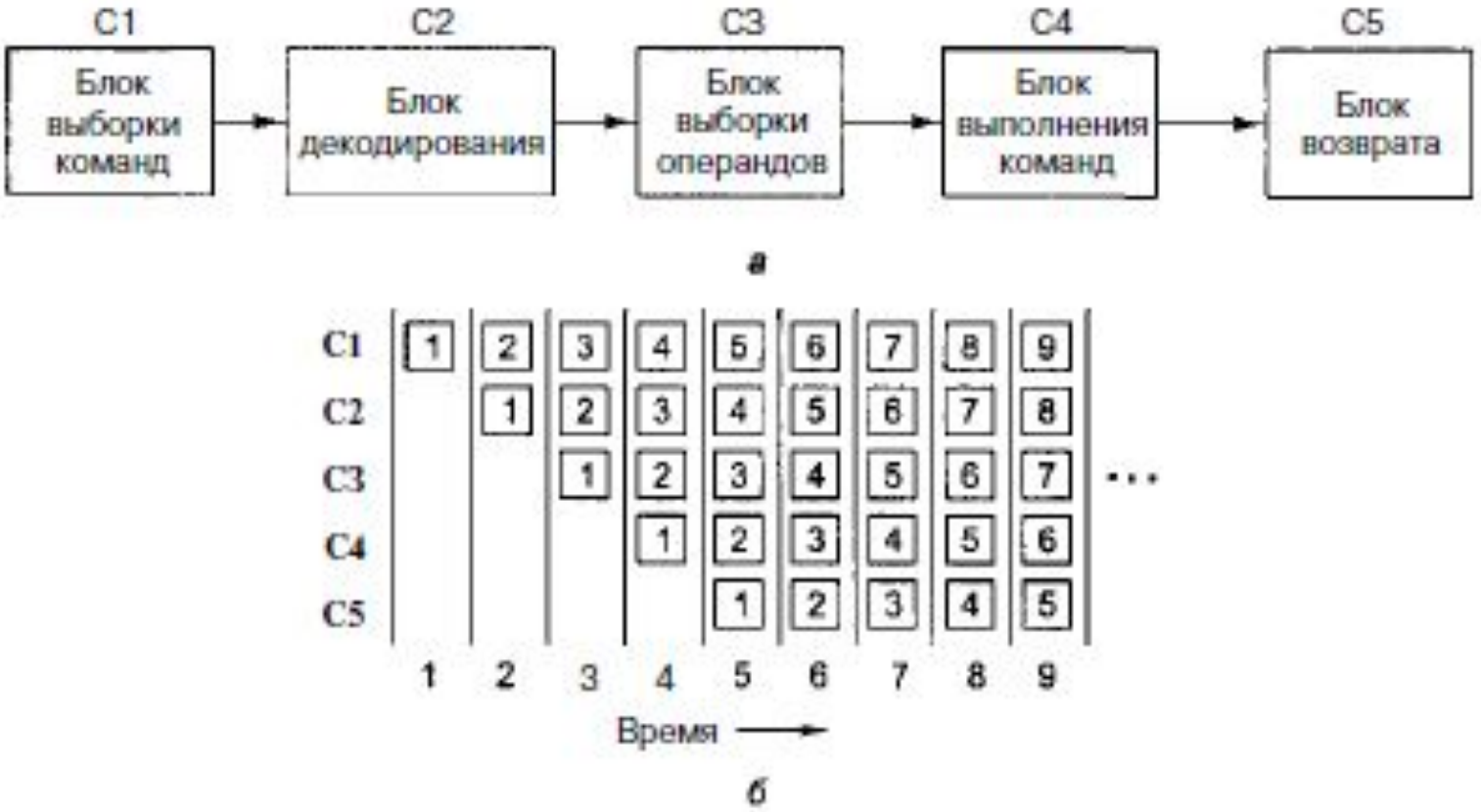


Рис. 2.3. Конвейер из 5 стадий (а); состояние каждой стадии в зависимости от количества пройденных циклов (б). Показано 9 циклов

На рис. 2.3, *б мы видим, как действует конвейер во времени. Во время цикла 1* стадия С1 работает над командой 1, вызывая ее из памяти. Во время цикла 2 стадия С2 декодирует команду 1, в то время как стадия С1 вызывает из памяти команду 2. Во время цикла 3 стадия С3 вызывает операнды для команды 1, стадия С2 декодирует команду 2, а стадия С1 вызывает третью команду. Во время цикла 4 стадия С4 выполняет команду 1, С3 вызывает операнды для команды 2, С2 декодирует команду 3, а С1 вызывает команду 4. Наконец, во время пятого цикла С5 записывает результат выполнения команды 1 обратно в регистр, тогда как другие стадии работают над следующими командами.

Суперскаляры

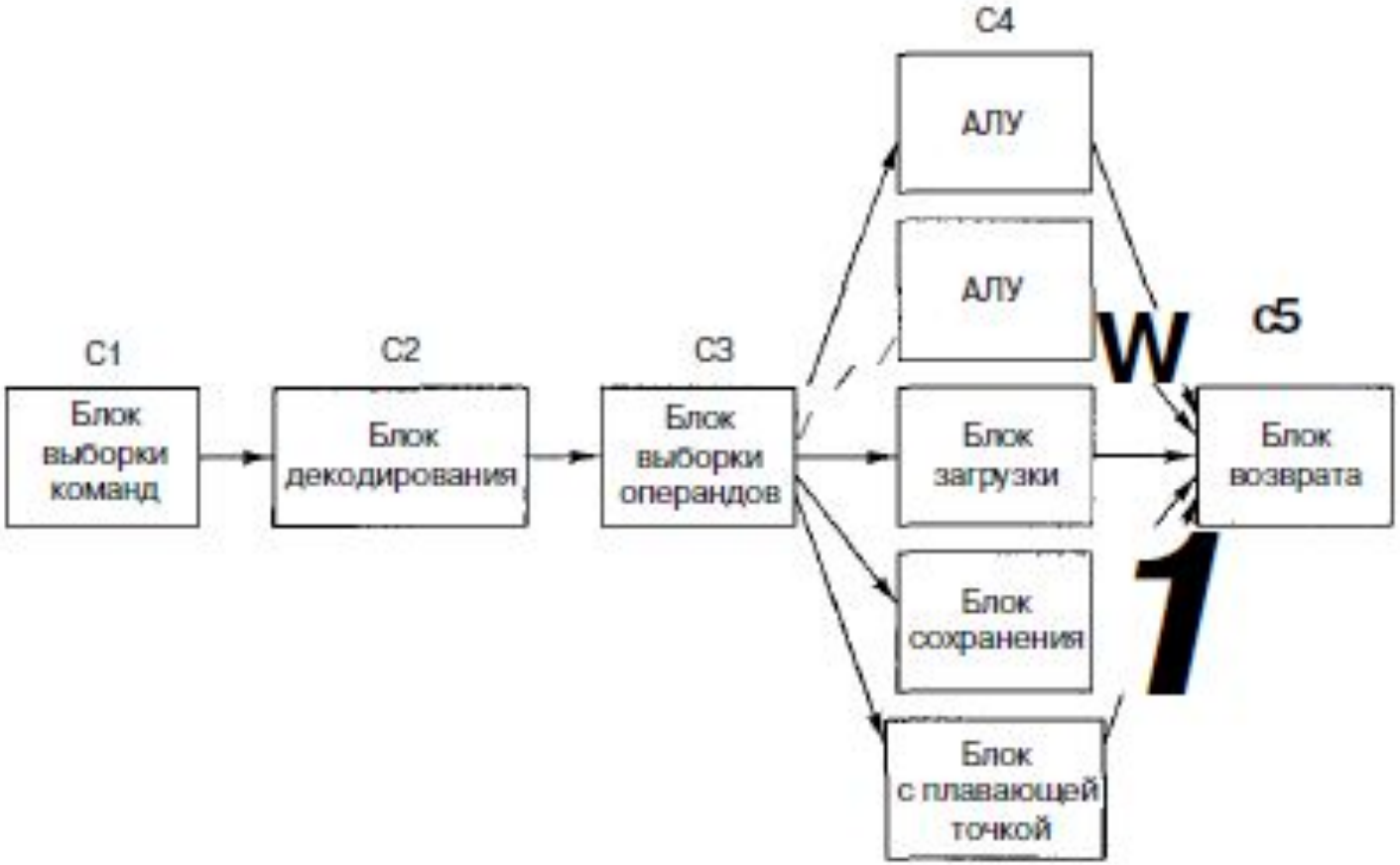


Рис. 2.5. Суперскалярный процессор с пятью функциональными блоками

Векторный компьютер

Массивно-параллельный процессор (array processor) состоит из большого числа сходных процессоров, которые выполняют одну и ту же последовательность команд применительно к разным наборам данных.

Для программистов **векторный процессор (vector processor)** очень похож на массивно-параллельный процессор (array processor). Как и массивно-параллельный процессор, он очень эффективен при выполнении последовательности операций над парами элементов данных. Но, в отличие от первого (array processor), все операции сложения выполняются в одном блоке суммирования, который имеет конвейерную структуру.

Мультипроцессорные системы

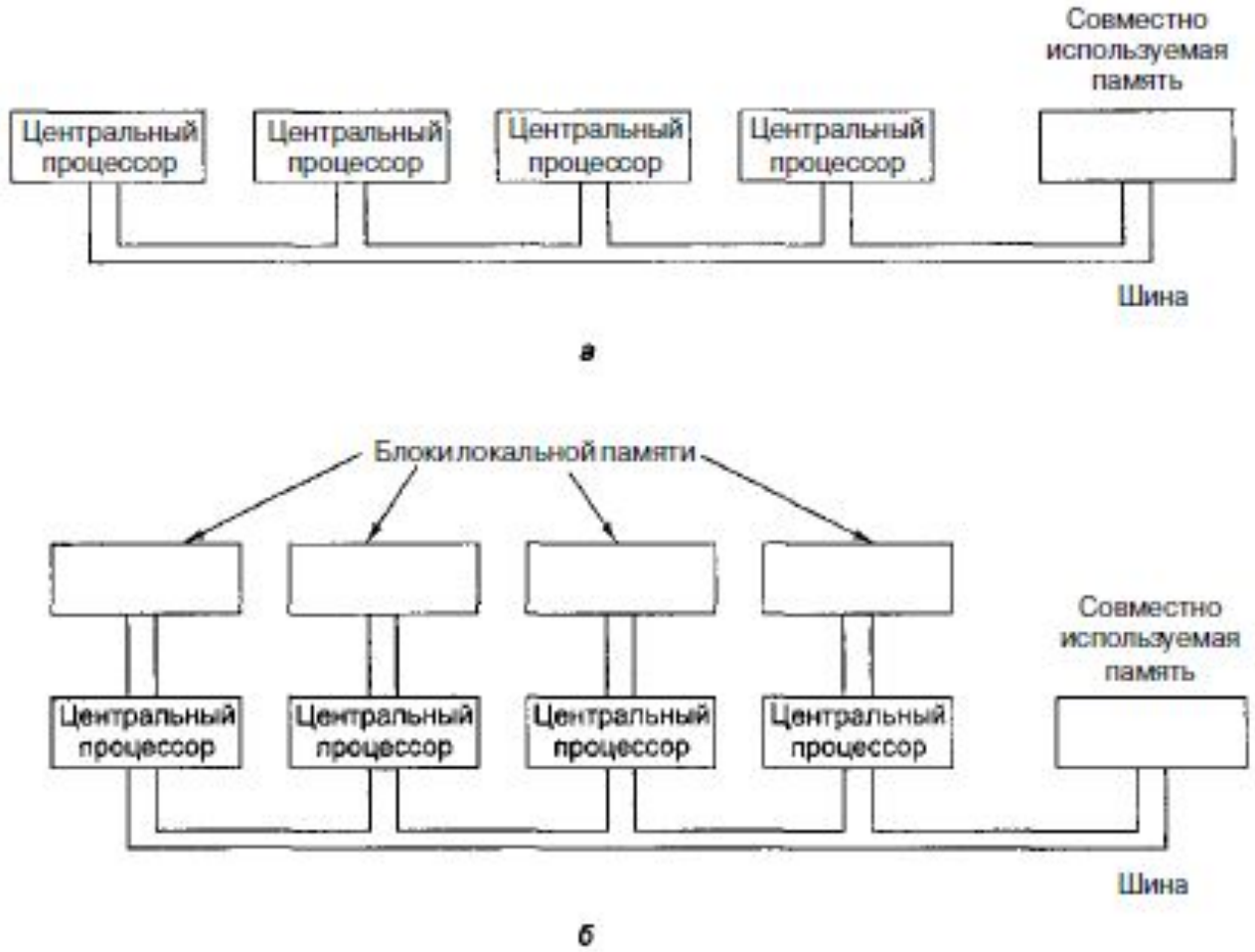


Рис. 2.7. Мультипроцессор с одной шиной и одной общей памятью (а); мультипроцессор, в котором для каждого процессора имеется собственная локальная память (б)