Суперскалярные процессоры

1 Структура процессора

2 Стратегии выдачи и завершения команд

3 Переименование регистров

4 Переупорядочивание команд (окно команд)

- **Знать:** понятие суперскалярного процессора и его структуру (типовую, с двумя конвейерами и со специализированными блоками); три категории стратегий выдачи и завершения команд; назначение и идея переименования регистров; переупорядочивание команд с помощью централизованного и распределенных окон команд.
- **Уметь:**.
- **Помнить:**
- **Литература:** Цилькер Б.Я., Орлов С.А. Организация ЭВМ и систем. Учебник для вузов. – СПб.: Питер, 2004. – 668 с. (с. 453-476) .

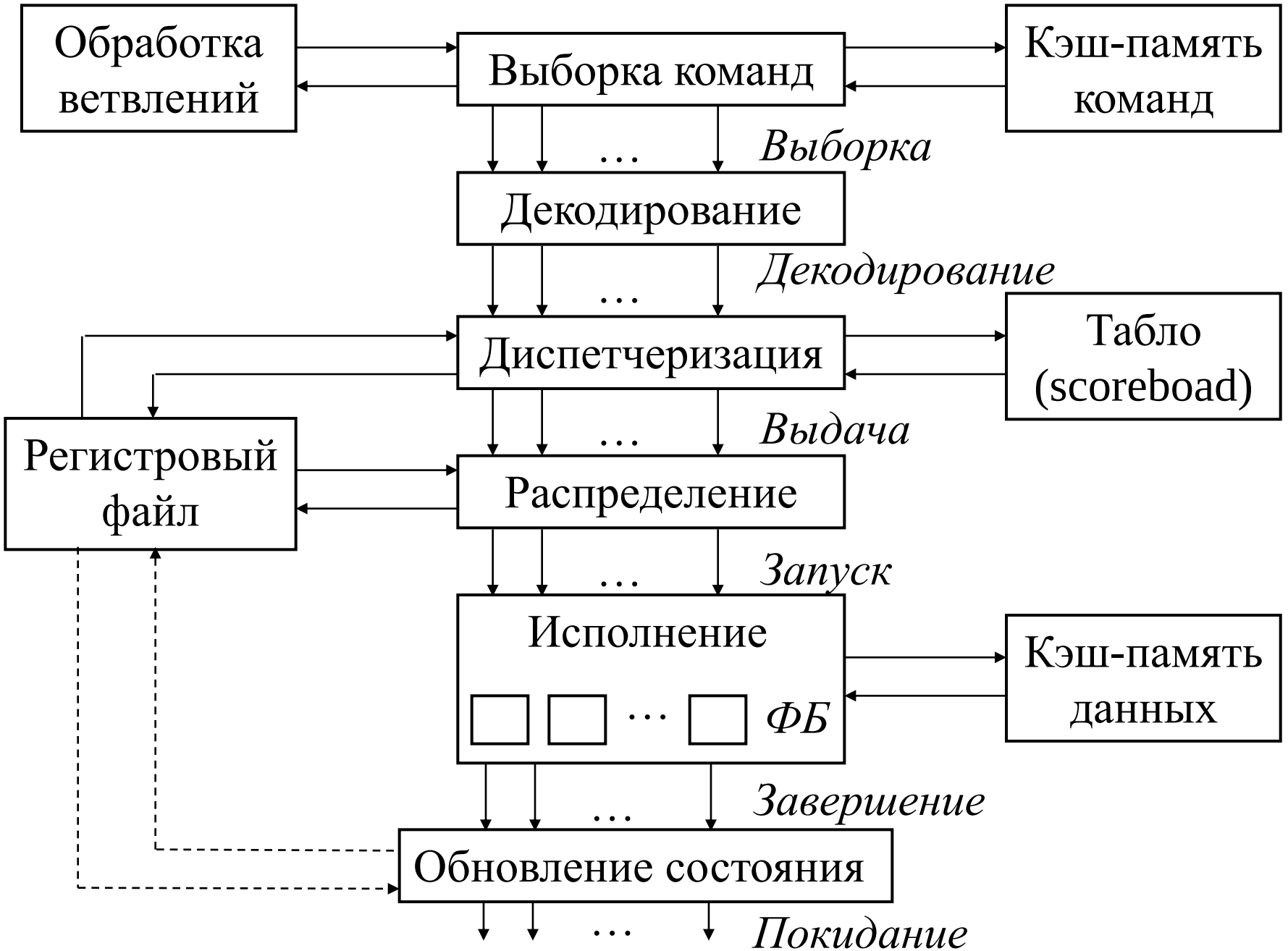
1 Структуры процессоров

- Основной объем вычислений обычно приходится на скалярные вычисления, то есть на обработку одиночных операндов, таких, например, как целые числа. Для подобных вычислений параллелизм реализуется значительно сложнее, чем для векторных операций, но тем не менее возможен. Примером могут служить суперскалярные процессоры (ССПР).
- *Суперскалярным* называется процессор, который одновременно выполняет более одной скалярной команды. Это достигается за счет включения в состав процессора нескольких самостоятельных функциональных (исполнительных) блоков, каждый из которых отвечает за свой класс операций и может присутствовать в процессоре в нескольких экземплярах.

Структура типичного ССПР: блоки выборки и декодирования команд

- Процессор включает в себя шесть блоков: выборки команд, декодирования команд, диспетчеризации команд, распределения команд по функциональным блокам (ФБ), исполнения и обновления состояния.
- *Блок выборки команд* извлекает команды из основной памяти через кэш-память команд. Этот блок хранит несколько значений счетчика команд и обрабатывает команды условного перехода.
- *Блок декодирования* расшифровывает код операции. В некоторых процессорах (микропроцессорах фирмы Intel) блоки выборки и декодирования совмещены.

Структура ССПР

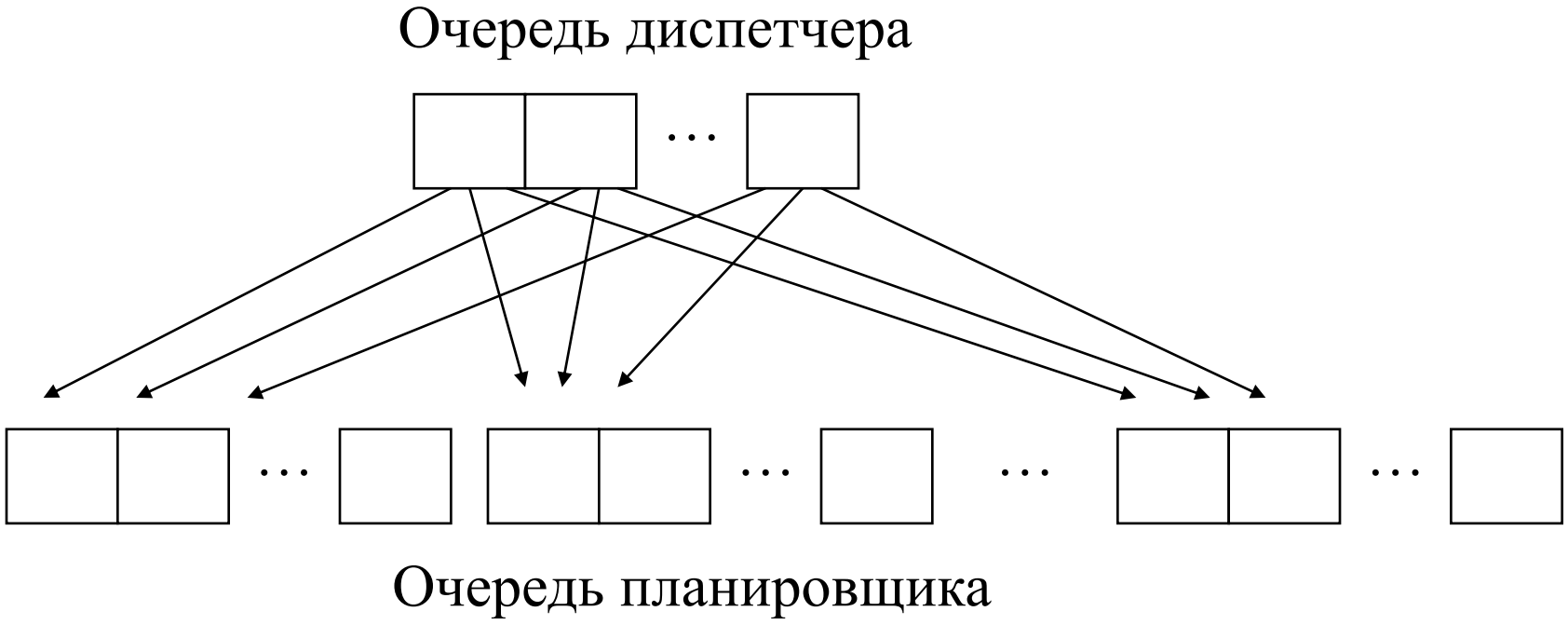


Блоки диспетчеризации и распределения

- Эти блоки взаимодействуют между собой и в совокупности играют в ССПР роль контроллера трафика, они хранят очереди декодированных команд.
- Очередь блока распределения часто рассредоточивается по нескольким самостоятельным буферам – *накопителям команд*, – предназначенным для хранения команд, которые уже декодированы, но еще не выполнены.
- Каждый *накопитель команд* связан со своим функциональным блоком (ФБ), поэтому число накопителей обычно равно числу ФБ, но если в процессоре используется несколько однотипных ФБ, то им придается общий накопитель.

Очереди диспетчера и планировщика

- По отношению к блоку диспетчеризации накопители команд выступают в роли виртуальных функциональных устройств. Оба вида очередей показаны ниже на рисунке. В некоторых процессорах они объединены в единую очередь.



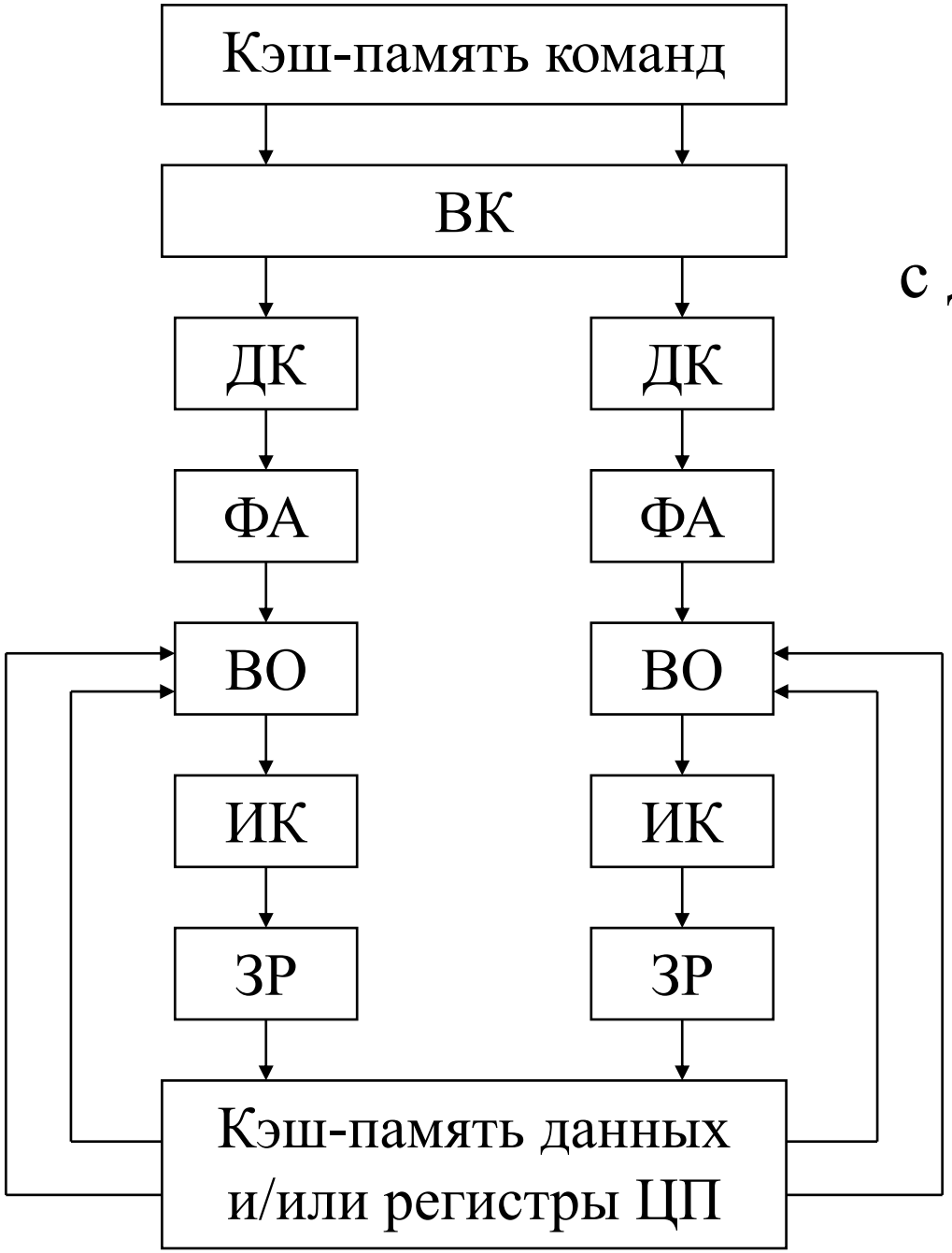
- В дополнение к очереди, блок диспетчеризации хранит также список свободных функциональных блоков, называемый ***табло***. Табло используется для отслеживания состояния очереди распределения. Один раз за цикл блок диспетчеризации извлекает команды из своей очереди, считывает из памяти или регистров операнды этих команд, после чего в зависимости от состояния табло, помещает команды и значения операндов в очередь распределения. Эта операция называется *выдачей команд*.
- Блок распределения в каждом цикле проверяет каждую команду в своих очередях на наличие всех необходимых для ее выполнения операндов и при положительном ответе начинает выполнение таких команд в соответствующем функциональном блоке.

Блоки исполнения и обновления состояния

- *Блок исполнения* состоит из набора функциональных блоков.
- Примерами ФБ могут служить целочисленные операционные блоки, блоки умножения и сложения с плавающей запятой, блок доступа к памяти.
- Когда исполнение команды завершается, ее результат записывается и анализируется *блоком обновления состояния*, который обеспечивает учет полученного результата теми командами в очередях распределения, где этот результат выступает в качестве одного из операндов.

ССПР с двумя конвейерами

- Суперскалярность предполагает параллельную работу максимального числа исполнительных блоков, что хорошо сочетается с конвейерной обработкой. При этом желательно, чтобы конвейеров было несколько (два или три).
- Подобный подход реализован, например, в микропроцессоре Intel Pentium, где имеются два конвейера, каждый со своим АЛУ (см. рисунок ниже). Здесь в отличие от стандартного конвейера, в каждом цикле необходимо производить выборку более чем одной команды. Соответственно память должна допускать одновременное считывание нескольких команд и операндов.

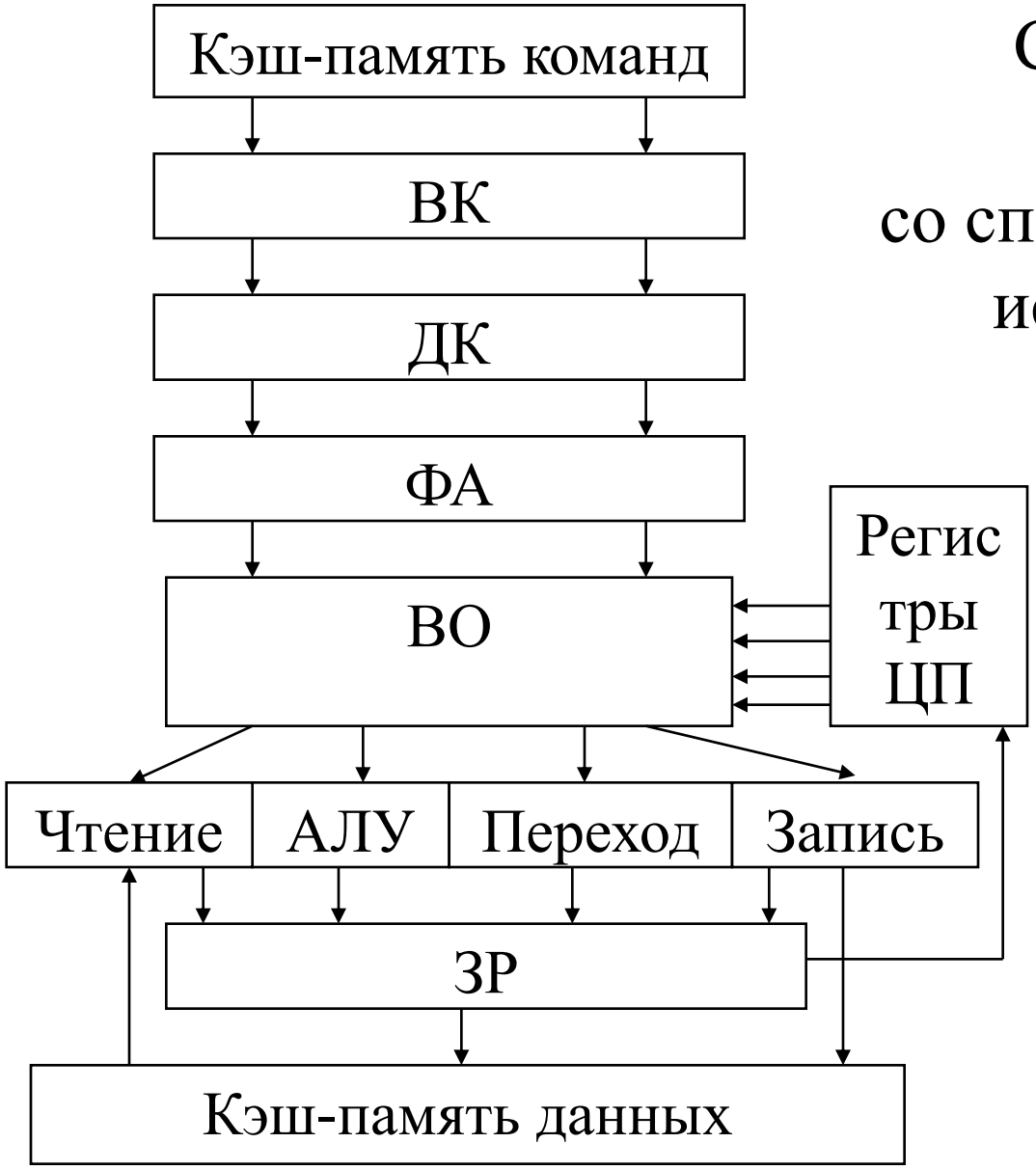


Суперскалярный процессор с двумя конвейерами

ССПР со специализированными исполнительными блоками

- Более интегрированный подход к построению суперскалярного конвейера показан на рисунке ниже.
- Здесь блок выборки (ВК) извлекает из памяти более одной команды и передает их через ступени декодирования команды и вычисления адресов операндов в блок выборки операндов (ВО). Когда операнды становятся доступными, команды распределяются по соответствующим исполнительным блокам. При этом операции "Чтение", "Запись" и "Переход" реализуются самостоятельными исполнительными блоками.
- Подобная форма процессора используется в микропроцессорах Pentium II и Pentium III. В Athlon (AMD) 3 конвейера.

Суперскалярный процессор со специализированными исполнительными блоками



Повышение производительности

- По разным оценкам, применение суперскалярного подхода приводит к повышению производительности в пределах от 1,8 до 8 раз.
- В процессорах некоторых ЭВМ реализованы как суперскалярность, так и суперконвейеризация.
- Такое совмещение имеет место в микропроцессорах Athlon и Duron фирмы AMD.
- Причем это совмещение охватывает не только конвейер команд, но и блок обработки чисел в форме с плавающей запятой.

2 Стратегии выдачи и завершения команд

- В режиме параллельного выполнения нескольких команд процессор должен определить, в какой очередности следует:
  - выбирать команды из памяти;
  - выполнять эти команды;
  - позволять командам изменять содержимое регистров и ячеек памяти.
- Для достижения максимальной загрузки всех ступеней своих конвейеров суперскалярный процессор должен варьировать все перечисленные виды последовательностей, но так, чтобы получаемый результат был идентичен результату при выполнении команд в порядке, определенном программой.

Категории стратегий

- Суперскалярый процессор обязан учитывать все виды зависимостей и конфликтов.
- В самом общем виде стратегии выдачи и завершения команд можно сгруппировать в такие категории:
  - упорядоченная выдача и упорядоченное завершение;
  - упорядоченная выдача и неупорядоченное завершение;
  - неупорядоченная выдача и неупорядоченное завершение.
- Рассмотрим эти варианты на примере суперскалярного процессора с двумя конвейерами.

Пример суперскалярного процессора

- Процессор способен одновременно выбирать и декодировать две команды, причем передача обеих команд на декодирование производится одновременно. В состав процессора входят три отдельных функциональных блока (ФБ) и два устройства, обеспечивающих запись результата.
- Предполагается существование следующих ограничений на выполнение программного кода из шести команд (I1-I6):
  - I1 требует для своего выполнения двух циклов (тактов) процессора;
  - I3 и I4 имеют конфликт за обладание одним и тем же ФБ;
  - I5 зависит от значения, вычисляемого командой I4;
  - I5 и I6 конфликтуют за обладание одним и тем же ФБ.

Упорядоченная выдача и упорядоченное завершение

Цикл	ДК		ИК			ЗР	
1	I1	I2					
2	I3	I4	I1	I2			
3	I3	I4	I1				
4		I4			I3	I1	I2
5	I5	I6			I4		
6				I5		I3	I4
7				I6			
8						I5	I6

I1 требует для своего выполнения двух циклов (тактов) процессора.
I3 и I4 имеют конфликт за обладание одним и тем же ФБ.
I5 зависит от значения, вычисляемого командой I4.
I5 и I6 конфликтуют за обладание одним и тем же ФБ.

- В примере производится одновременная выборка и декодирование двух команд.
- Чтобы принять очередные команды, процессор должен ожидать, пока освободятся обе части ступени декодирования.
- Для упорядочивания завершения выдача команд приостанавливается, если возникает конфликт за общий функциональный блок или если функциональному блоку для формирования результата требуется более чем один такт процессора.
- Задержка в примере составляет 8 тактов.
- Такая стратегия применялась в первых процессорах типа Pentium, сейчас она практически не встречается, но ее используют в качестве точки отчета при сравнении различных стратегий.

Упорядоченная выдача и неупорядоченное завершение

Цикл	ДК		ИК			ЗР	
1	I1	I2					
2	I3	I4	I1	I2			
3		I4	I1		I3	I2	
4	I5	I6			I4	I1	I3
5		I6		I5		I4	
6				I6		I5	
7						I6	

Стратегия при заданных условиях допускает завершение команды I2 до окончания команды I1. Это позволяет команде I3 завершиться на один такт раньше, вследствие чего результаты выполнения команд I1 и I3 записываются в одном и том же такте.

- При неупорядоченной выдаче в любой момент времени в стадии исполнения может находиться любое число команд, а степень параллелизма ограничена только числом функциональных блоков.
- По сравнению с предыдущей стратегией, возможность неупорядоченного завершения команд привела к сокращению времени выполнения шести команд на один цикл процессора.

Неупорядоченная выдача и неупорядоченное завершение

- Неупорядоченная выдача развивает предыдущую концепцию, разрешая процессору нарушать предписанный программой порядок выдачи команд на исполнение.
- Чтобы обеспечить неупорядоченную выдачу команд, в конвейере необходимо максимально развязать ступени декодирования и исполнения.
- Это обеспечивается с помощью буферной памяти, называемой окном команд. Каждая декодированная команда сначала помещается в окно команд.
- Процессор продолжает выборку и декодирование команд вплоть до полного заполнения буфера.

- Выдача команд из буфера на исполнение определяется не последовательностью их поступления, а мерой готовности. Иными словами, любая команда, для которой уже известны значения всех операндов, при условии что функциональный блок, требуемый для ее исполнения свободен, немедленно выдается из буфера на исполнение.
- На рисунке ниже изображено окно команд, но оно не является дополнительной ступенью конвейера.
- Стратегии неупорядоченной выдачи и неупорядоченного завершения также свойственны ранее рассмотренные ограничения. Команда не может быть выдана, если она приводит к зависимости или конфликту. Разница заключается в том, что к выдаче готово больше команд, и это позволяет уменьшить вероятность приостановки конвейера.

Использование окна команд

Цикл	ДК		Окно	ИК			ЗР	
1	I1	I2						
2	I3	I4	I1, I2	I1	I2			
3	I5	I6	I3, I4	I1		I3	I2	
4			I4, I5, I6		I6	I4	I1	I3
5					I5		I4	I6
6							I5	

В примере возможна выдача команды I6 до выдачи команды I5 (в примере I5 зависит от I4, а I6 – нет). Таким образом, сберегается один такт, как в ступени исполнения команды (ИК), так и в ступени записи результата (ЗР), при этом экономится один такт.

3 Переименование регистров

- Применение стратегий с неупорядоченной выдачей и завершением команд ведет к тому, что запись в регистры может происходить также неупорядоченно и отдельные команды, обратившись к какому-то регистру, вместо нужного получат "устаревшее " или "опережающее" значение.
- Пример:
  - I1: MUL R2, R3, R1 (R2:=R3xR1)
  - I2: ADD R0, R1, R2 (R0:=R1+R2)
  - I3: SUB R2, R0, R1 (R2:=R3-R1)
- При неупорядоченных выдаче/завершении возможны ситуации, приводящие к неверному результату.

В рассматриваемом примере могут быть следующие неверные результаты:

- команда I2 была выполнена до того, как I1 успела записать в регистр R2 свой результат, т.е. I2 использовала "старое" содержимое R2.
- команда I3 исполнена раньше, чем I1, поэтому неверный результат будет получен в I2, а по завершении цепочки из трех команд в R2 останется результат I1 вместо результата I3.
- (Рассматриваемый пример)
  - I1: MUL R2, R3, R1 (R2:=R3xR1)
  - I2: ADD R0, R1, R2 (R0:=R1+R2)
  - I3: SUB R2, R0, R1 (R2:=R3-R1)

- Нарушение порядка выполнения команд ведет к неправильному результату. Вводя новый регистр R2a, получим иную последовательность:
- I1: MUL R2, R0, R1 (R2:=R3xR1)
- I2: ADD R0, R1, R2 (R0:=R1+R2)
- I3: SUB R2a, R3, R1 (R2a:=R3-R1)
- Здесь возможность конфликта устранена. Такой метод известен как *переименование регистров*.
- Основная идея переименования регистров состоит в том, что каждый новый результат записывается в один из свободных в данный момент дополнительных регистров, при этом ссылки на заменяемый регистр во всех последующих командах соответственным образом корректируются.

- Программист, составляющий программу, имеет дело с именами логических регистров.
- Число физических регистров аппаратного регистрового файла (АРФ) обычно больше числа логических.
- "Лишние" регистры АРФ используются в процедуре переименования регистров для временного хранения результатов до момента разрешения конфликтов по данным, после чего значение из регистра временного хранения переписываются на свое "штатное" место.
- В некоторых процессорах "лишние" регистры в АРФ отсутствуют, а для поддержки переименования предусмотрены специальные структуры, например, буфер переименования.

4 Переупорядочивание команд (окно команд)

- В основе переупорядочивания команд лежит использование окна команд – буферной памяти, куда помещаются все команды, прошедшие декодирование, и переименование регистров (переименование выполняется только для тех команд, которые записывают свои результаты в регистры).
- Окно команд обеспечивает отсрочку передачи команд на исполнение до момента готовности операндов, а также нужную очередность завершения команд и загрузки результатов в регистры АРФ.
- Известны два варианта окна команд – *централизованное и распределенное*.

Централизованное окно

- ***Централизованное окно*** команд реализуется в виде так называемого *табло*.
- Табло представляет собой буферное запоминающее устройство, в котором хранится некоторое количество последних извлеченных из памяти и декодированных команд, а также текущая информация о доступности ресурсов, привлекаемых для их исполнения.
- Функциями табло являются оперативное выявление команд, для исполнения которых доступны все необходимые операнды и ресурсы, и выдача таких команд на исполнение в соответствующие функциональные блоки.

Реализация табло

- Физически табло реализуется на основе ассоциативной памяти.
- Каждой команде выделяется одна ячейка, состоящая из следующих полей:
  - поля операции (ОС), где хранится дешифрированный код операции;
  - двух полей операндов (OS1/IS1, OS2/IS2) размещающих значения операндов (OS1, OS2), если они известны, либо, информацию о том, откуда эти операнды должны быть получены (IS1, IS2);
  - поля результата (R), указывающего регистр, куда должен быть помещен результат выполнения данной команды;
  - поля битов достоверности (VS1, VS2).

Пример заполнения табло

Команда	ОС	R	O_{S1}/I_{S1}	O_{S2}/I_{S2}	V_{S1}	V_{S2}
1	ОП	ИД	ЗН	ИД	1	0
2	ОП	ИД	ЗН	ИД	1	0
3	ОП	ИД	ИД	ЗН	0	1
4	ОП	ИД	ЗН	ЗН	1	1
5	ОП	ИД	ИД	ЗН	0	1

В табло также хранится текущая информация о доступности функциональных блоков.

В примере для команд I1, I2, I3 и I5 известны значения одного из операндов и они вынуждены ожидать значения второго операнда. Команде I4 известны оба операнда, и она может быть выдана, если свободен соответствующий функциональный блок

Распределенное окно команд

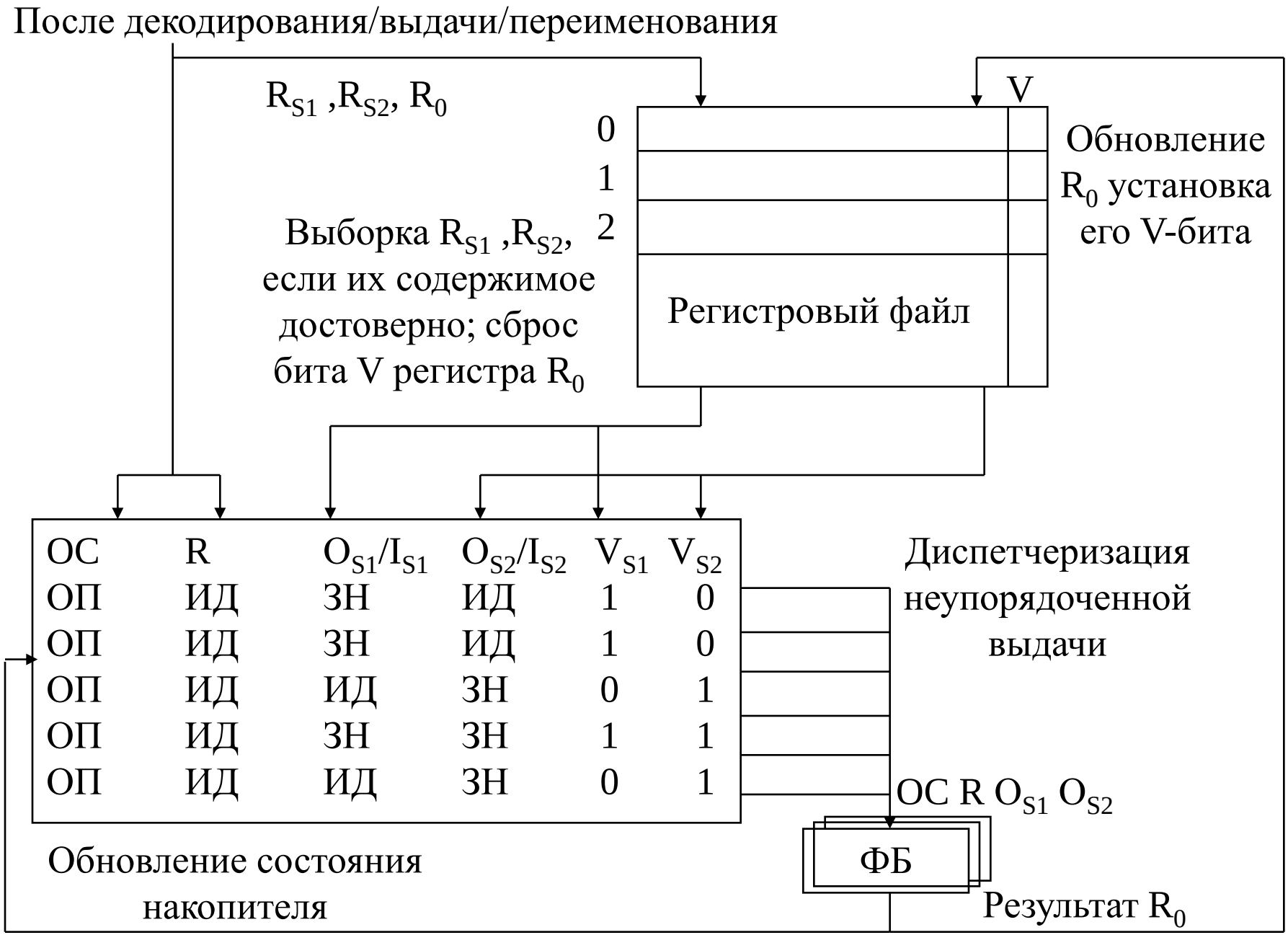
- В этом случае на входе каждого функционального блока размещается буфер декодированных команд, называемый накопителем команд (или схемой резервирования).
- После выборки и декодирования команды распределяются по накопителям тех функциональных блоков, где команда будет исполняться.
- Логика работы каждого накопителя аналогична централизованному окну команд. При этом не требуется, чтобы операнд был обязательно занесен в отведенный для него регистр – он может быть ускоренно передан прямо в накопитель команд для немедленного использования или буферизирован там для последующего использования.

Работа накопителя команд

- Одновременно могут в среднем исполняться 1-3 команды, а иногда и 5-6 команд. Типичный накопитель рассчитан на 1-3 команды.
- После передачи команд в накопитель там, производится проверка на наличие команд, для исполнения которых есть все необходимые операнды.
- Поиск таких команд выполняется путем анализа битов VS1 и VS2. Если у команды оба бита в единичном состоянии, то она готова к выдаче в ФБ.
- При наличии в накопителе только одной команды она сразу выдается в ФБ. Если готовых команд несколько, из них в ФБ пересылается наиболее "старая", то есть поступившая в накопитель первой.

Завершение выполнения команды

- После завершения выполнения команды ее результат совместно с идентификатором регистра результата (R_0) выдается в регистровый файл и в накопитель для обновления их содержимого.
- В ходе обновления в регистровом файле вычисленное значение заносится в R_0, а бит достоверности регистра результата устанавливается в единицу.
- С этого момента значение R_0 доступно в качестве операнда для последующих команд.
- В накопителе производится поиск идентификатора R_0 в полях OS1/IS1, OS2/IS2 всех команд и их замена на вычисленное значение. Одновременно состояние бита VS1 (VS2) заменяется на единицу.
- Далее выполняется очередной поиск готовых к исполнению команд и их выдача в ФБ.


После декодирования/выдачи/переименования
R_{S1} ,R_{S2}, R_0
V
0
1
2
Обновление R_0 установка его V-бита
Выборка R_{S1} ,R_{S2}, если их содержимое достоверно; сброс бита V регистра R_0
Регистровый файл
ОС R O_{S1}/I_{S1} O_{S2}/I_{S2} V_{S1} V_{S2}
ОП ИД ЗН ИД 1 0
ОП ИД ЗН ИД 1 0
ОП ИД ИД ЗН 0 1
ОП ИД ЗН ЗН 1 1
ОП ИД ИД ЗН 0 1
Диспетчеризация неупорядоченной выдачи
ОС R O_{S1} O_{S2}
ФБ
Обновление состояния накопителя
Результат R_0