АРХИТЕКТУРА ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

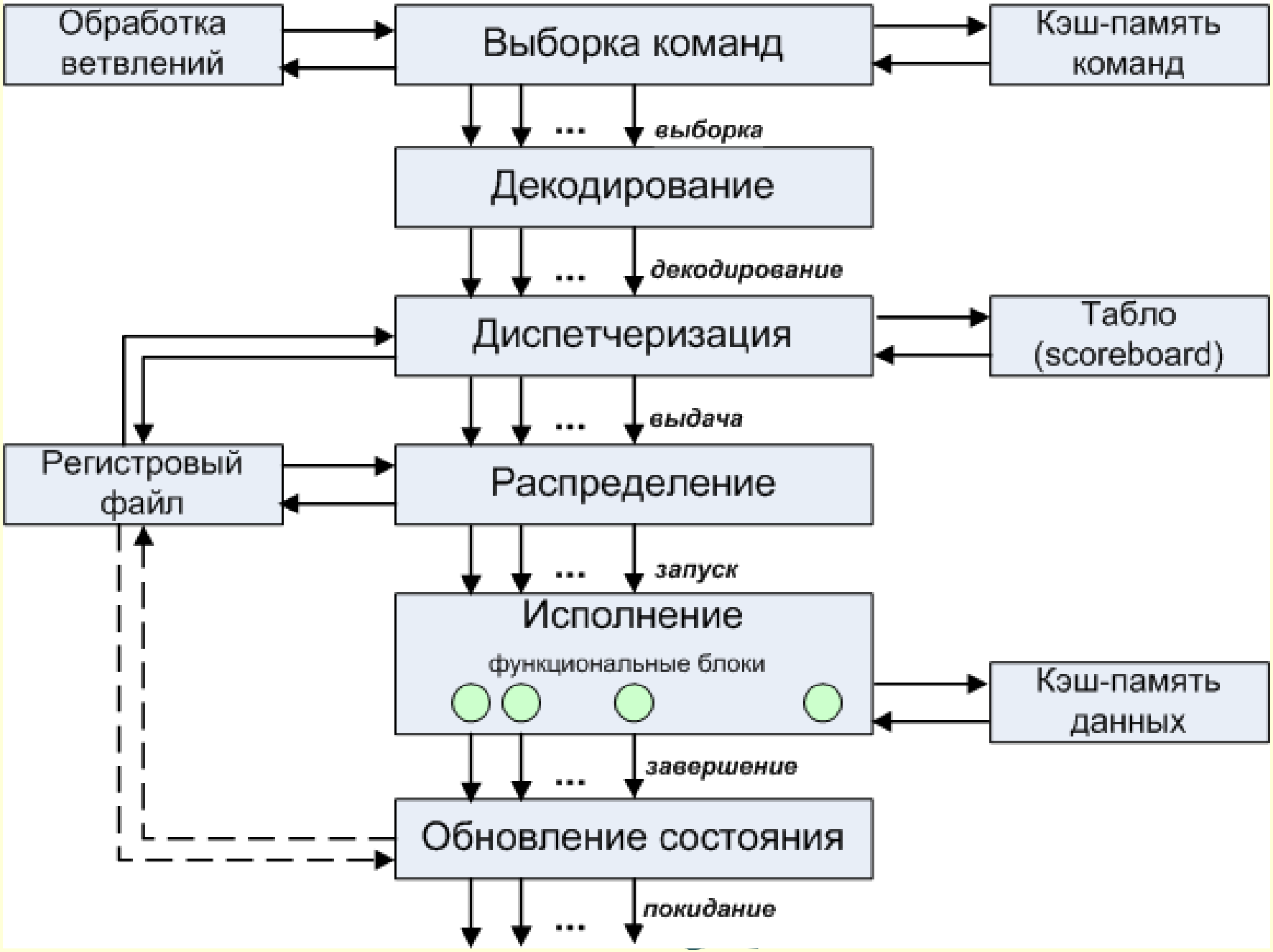
Лекция 5:

Суперскалярные ВС

Суперскалярная архитектура

- Мотивация – поиск решений, повышающих производительность
  - в предыдущих лециях рассматривались конвейеры команд и арифметики
  - матричные и векторные процессоры: доп. параллелизм при обработке массивов
    - не решается проблема скалярных операций (это существенный объём вычислений)
- Суперскалярный подход – выполнение нескольких скалярных команд параллельно
  - дублирование функциональных блоков

Суперскалярная архитектура

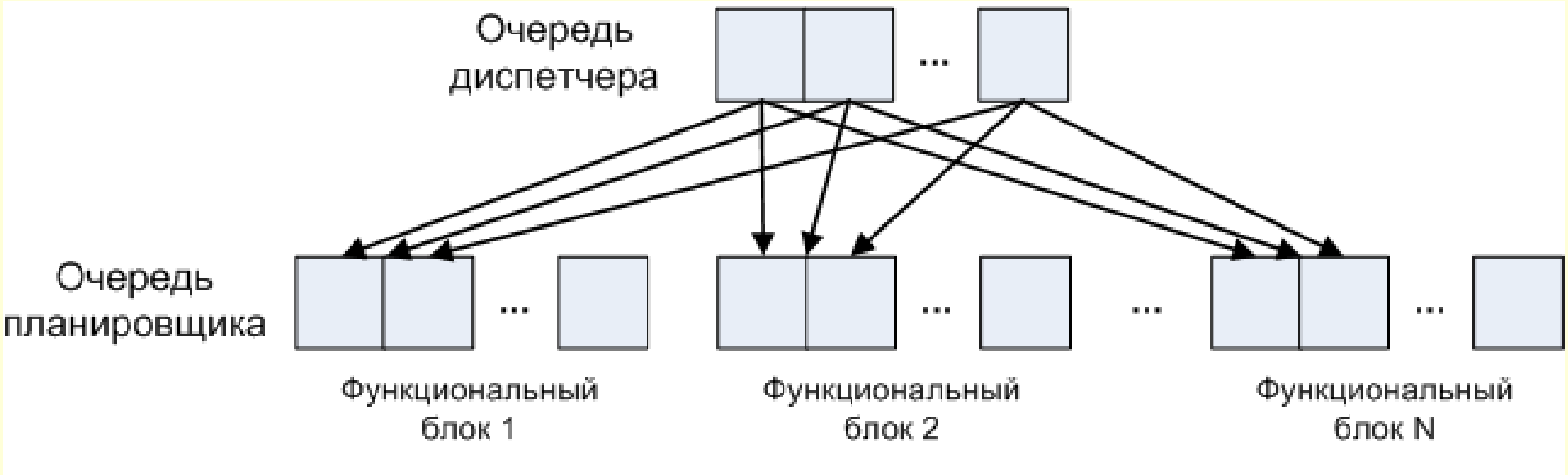


Суперскалярная архитектура

- Блок выборки команд
  - извлекает команды из ОП через кэш-память команд
  - хранит несколько значений счётчика команд
  - обрабатывает команды УП
- Блок декодирования
  - расшифровывает код операции
  - в некоторых процессорах объединён с блоком выборки

Суперскалярная архитектура

- Блоки диспетчеризации и распределения
  - хранят очереди команд
    - табло для слежения за очередью распределения
    - очередь блока распределения часто разделена на накопители команд (каждый привязан к ФБ)
  - управляют потоками команд

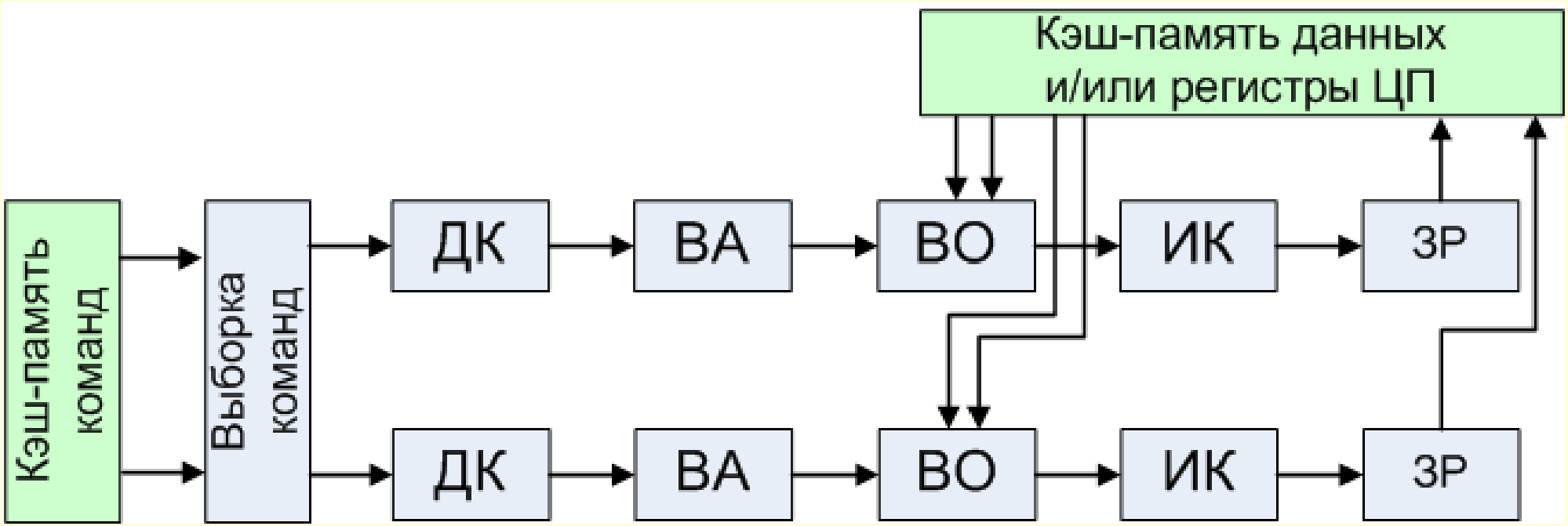


Суперскалярная архитектура

- Блок исполнения
  - состоит из функциональных блоков
    - АЛУ, доступ к памяти
- Блок обновления состояния
  - запись результата
  - доставка результатов командам, находящимся в очередях распределения

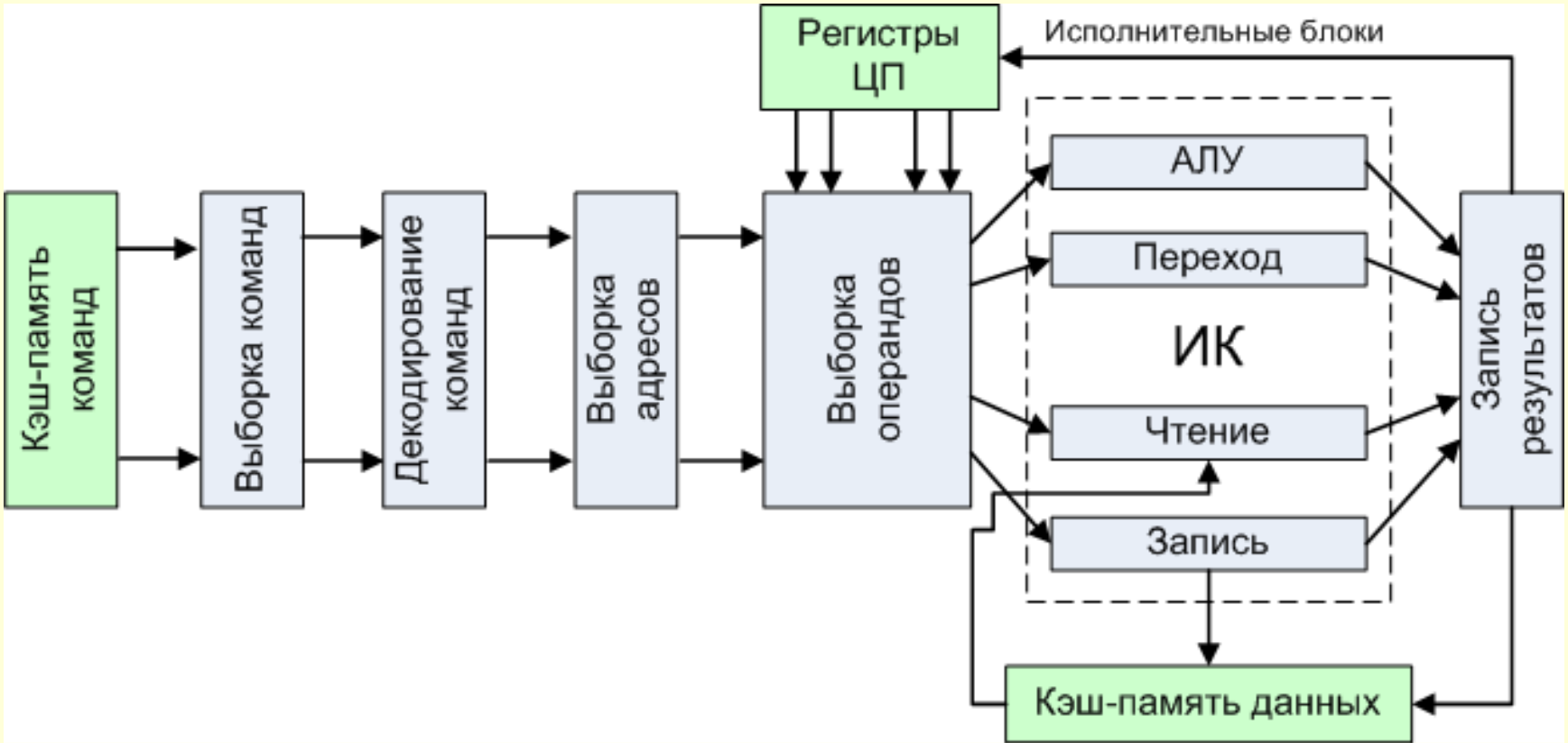
Суперскалярная архитектура

- Суперскалярность предполагает параллельную работу ФБ
  - желательно иметь 2-3 конвейера
  - пример реализации – Intel Pentium



Суперскалярная архитектура

- Более интегрированный подход:
  - выборка более одной команды
  - распределение по блокам по мере доступности операндов
  - Чтение/Запись/Переход – самостоятельные блоки
  - Примеры реализации: Intel Pentium II+, AMD Athlon



Особенности реализации

Несколько функциональных блоков + несколько конвейеров команд:

- Проблема последовательности поступления команд
  - неупорядоченная выдача (out-of-order issue)
    - порядок поступления на исполнительные блоки отличается от предписанной
- Проблема последовательности завершения команд
  - неупорядоченное завершение (out-of-order completion)

Особенности реализации

Неупорядоченное завершение

MUL R1, R2, R3

ADD R4, R5, R6

- умножение потребует больше тактов

Конфликт по функциональному блоку

- На блок может претендовать несколько команд

ADD R1,R2,R3

SUB R4, R5, R6

- зависимости по данным нет, но АЛУ может быть только одно

Стратегии выдачи и завершения команд

- Процессор определяет очерёдность:
  - выборки команд из ОП
  - выполнения этих команд
  - изменения командами содержимого регистров и памяти
- Цель – оптимальная загрузка конвейера
- Стратегии:
  - упорядоченные выдача и завершение
  - упорядоченная выдача, неупорядоченное завершение
  - неупорядоченная выдача и неупорядоченное завершение

Стратегии: пример

- На вход процессора поступает программа из 6 команд I1-I6:
  - I1 исполняется 2 такта
  - между I3 и I4 конфликт за общий ФБ
  - I5 зависит от результата I4
  - I5 и I6 конфликтуют за ФБ

Стратегии: пример

Упорядоченная выдача и упорядоченное завершение

- самая простая в реализации стратегия
- первые Intel Pentium

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

Стратегии: пример

Упорядоченная выдача и неупорядоченное завершение

- Один из конвейеров может продолжать работу при заторе в другом
- Гарантируется нормальный порядок записи

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	

Стратегии: пример

Неупорядоченная выдача и неупорядоченное завершение

Слабже связь между декодированием и исполнением:

- буферная память (окно команд)
- декодер заполняет буфер
- выдача из буфера по мере готовности

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		I5		I4	I6
6							I5	

Аппаратная поддержка суперскалярных операций

Задачи:

- устранить зависимость по данным (чтение/запись после записи)
  - решение – **переименование регистров**
- обеспечить результат вычислений, идентичный последовательному исполнению
  - **переупорядочивание команд** (откладывание исполнения)

Переименование регистров

Исходный код:

MUL R2, R0, R1

ADD R0, R1,R2

SUB R2, R0, R1

Вводим новые регистры, устраняя конфликт:

MUL R2, R0, R1

ADD **R0a**, R1, R2

SUB **R2a**, **R0a**, R1

Переименование регистров

- Основная идея
  - запись результатов в свободные регистры
  - коррекция ссылкок на заменяемый регистр
  - программист работает с именами логических регистров
  - число физических регистров отличается (в большую сторону) от логических
    - либо используют буфер переименования
  - таблицы подстановок (lookup table) для динамического отображения логических регистров на физические

Переименование регистров

- Буфер переименования
  - ассоциативное ЗУ или набор регистров
  - ячейки содержат 5 полей
    - флаг «вход занят» (БП недоступен)
    - номер логического регистра, заменяемого данным БП
    - значение – текущее содержимое регистра
    - флаг «значение достоверно» (если значение не вычислено, ЗД = 0)
    - флаг «последнее переименование» - актуальный экземпляр ссылки на регистр

Переименование регистров

Записи АРФ:

- флаг «значение достоверно» (вычислено)

Переупорядочивание команд

- Окно команд – буферная память для декодированных команд
  - отсрочка исполнения команд до готовности операндов
  - контроль очерёдности исполнения
- Два варианта:
  - централизованное окно команд
  - распределённое окно команд

Централизованное окно команд

- Реализуется в виде **табло** (scoreboard)
  - буферное ЗУ
  - хранит N последних команд и информацию о ресурсах для их исполнения
  - выявляет команды, готовые к исполнению
- Для каждой команды запись содержит:
  - код операции
  - значения операндов (2 поля операндов)
    - или информацию об их источнике
  - поле результата
  - поле битов достоверности

Централизованное окно команд

Содержимое табло:

Команда	Поле операции	Поле результата	Поле операнда 1	Поле операнда 2	Поле битов достоверности	
					ЗД1	ЗД2
1	ОП	ИД	ЗН	ИД	ЗН	ИД
2	ОП	ИД	ЗН	ИД	ЗН	ИД
3	ОП	ИД	ИД	ЗН	ИД	ЗН
4	ОП	ИД	ЗН	ЗН	ЗН	ЗН
5	ОП	ИД	ИД	ЗН	ИД	ЗН
			...			

ОП - дешифрированный код операции; ИД - идентификатор регистра;

ЗН - значение операнда; ЗД - значение достоверно

Централизованное окно команд

- На очередном такте в табло может быть готово к выдаче несколько команд
- Необходимо сохранять заданную программой последовательность исполнения
  - метод 1: **стек диспетчеризации**
    - добавление новых команд в верх табло
    - при выдаче команды – сдвиг вниз для остальных
  - метод 2: **блок обновления регистров**
    - табло работает по принципу FIFO-очереди
    - общий сдвиг таблицы вниз (проще логика работы)

Распределённое окно команд

- На входе каждого ФБ – буфер декодированных команд
  - накопитель команд (схема резервирования – reservation station)
  - логика аналогична окну централизованному команд
- Команды распределяются по схемам резервирования тех ФБ, на которых будут выполняться
- Операнды могут передаваться прямо в накопитель или буферизированы в нём для последующего использования

Буфер восстановления последовательности

- **Задачи**
  - Поддержка правильной последовательности исполнения команд при параллельных ФБ
  - Распределение команд по схемам резервирования
- **Структура - кольцевой буфер**
  - голова – указатель на свободный вход
  - хвост – команда, первой подлежащая удалению из БВП
  - в ячейках – состояние команд (выдана, выполняется, завершена)
- **Команды покидают буфер строго упорядоченно**
  - запись в регистры и память разрешена только командам, покинувшим БВП

АРХИТЕКТУРА ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

Лекция 6:

Архитектуры с полным и сокращённым набором команд

Система команд

- Семантический разрыв (СР):
  - операции, характерные для языков высокого уровня, отличаются от машинных команд
- Подходы к преодолению разрыва
  - архитектура с полным набором команд
    - CISC (Complex Instruction Set Computer)
  - архитектура с сокращённым набором команд
    - RISC (Reduced Instruction Set Computer)
  - архитектура с командными словами сверхбольшой длины
    - VLIW (Very Long Insturction Word)

Архитектура CISC

- Решение проблемы СР за счёт усложнения системы команд
  - сложные команды, аналогичные операторам ЯВУ
- Характерные черты архитектуры:
  - малое число регистров общего назначения
  - много сложных машинных команд
  - разнообразие способов адресации операндов
  - множество форматов команд различной разрядности
  - команды, совмещающие вычисления с обращением к памяти

Архитектура CISC

- Недостатки:
  - сложность архитектуры (в т.ч. устройства управления)
    - негативно сказывается на производительности
  - сложно организовать эффективный конвейер
- Закономерности
  - доля сложных команд в программах мала, но отнимает бОльшую часть управляющей памяти
  - частый вызов процедур, реализующих операторы ЯВУ
  - в среднем, много локальных переменных и немного параметров при вызове процедур (до 6)
  - до половины операций – присваивание (пересылка значения)

Архитектура RISC

- Главная цель – эффективный конвейер команд
  - равномерность потока команд
  - высокая загрузка ЦП
  - в идеале, любой этап цикла за 1 такт

Архитектура RISC

- Решения
  - стандартная длина команды (равна ширине шины данных)
  - простота декодирования и управления
    - меньше команд, форматов данных и видов адресации
  - минимум команд с доступом к памяти
    - к памяти обращаются только команды «чтение»/«запись»
    - остальные команды – «регистр-регистр»
    - много регистров общего назначения (от 32)

Архитектура RISC

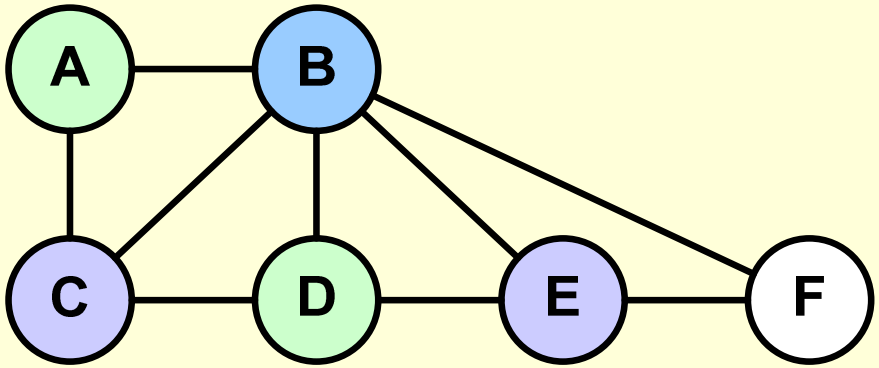
- Характерные черты архитектуры:
  - 75-100% команд занимают 1 цикл
  - однословная длина всех команд
  - малое числок команд (до 128)
  - малое число форматов команд (до 4)
  - малое число способов адресации (до 4)
  - выделенные операции чтения/записи в память
  - большой процессорный файл регистров

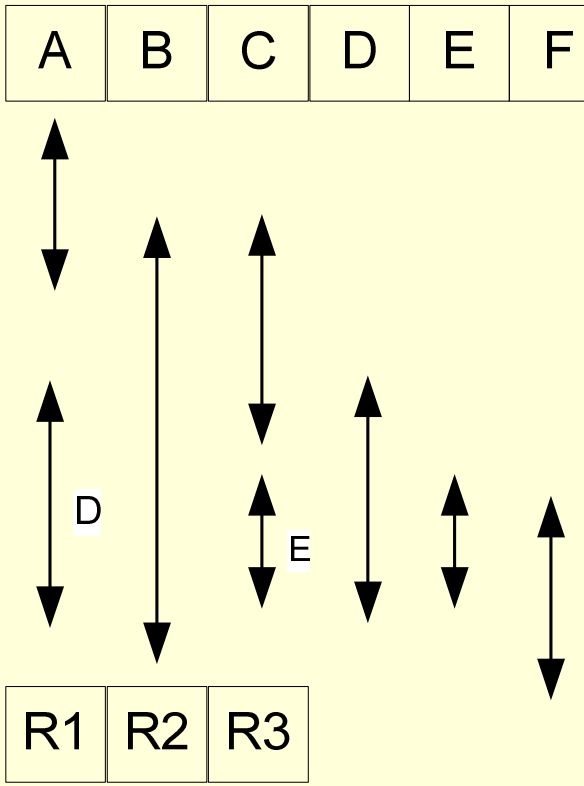
Архитектура RISC

- Для эффективного использования большого числа РОН требуется оптимизация
  - программная
    - на этапе компиляции
  - аппаратная

Программная оптимизация

- Максимальное задействование РОН на этапе компиляции
  - бесконечное число виртуальных регистров
  - при оптимизации излишки отображаются на память
- Определение, какие данные эффективнее помещать в регистры
  - раскраска графа





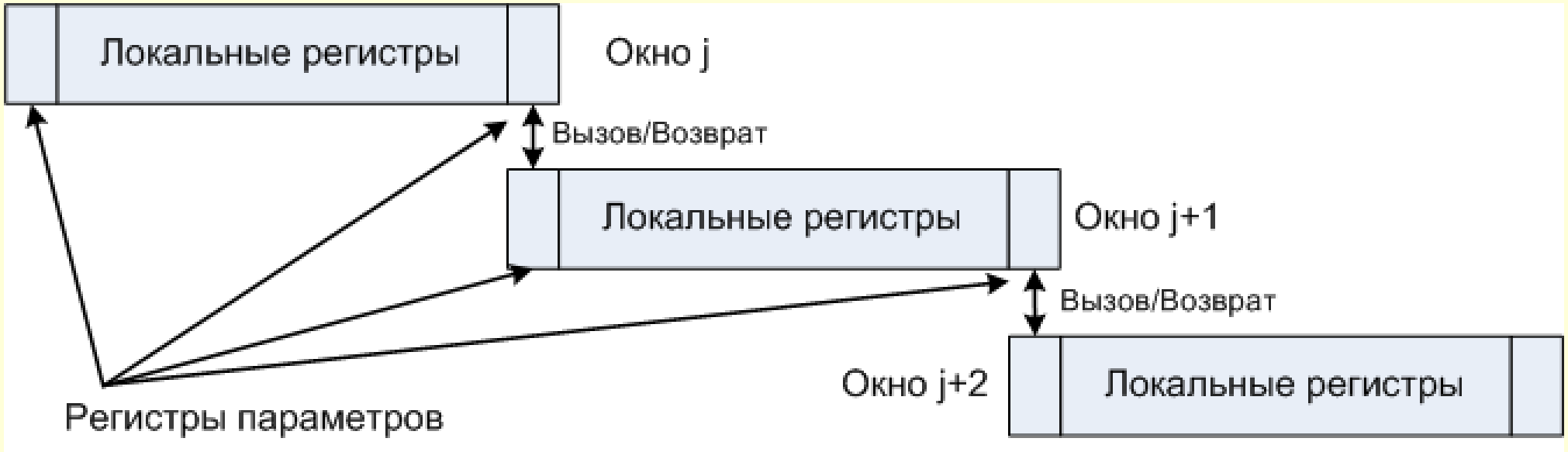
Аппаратная оптимизация

- Цель – сократить затраты на вызов процедур
  - много локальных переменных и констант
- Регистровые окна
  - задача – упростить вызов процедур
  - регистровый файл разбивается на группы (окна) равного размера (по 32 регистра)
    - отдельное окно глобальных переменных
    - отдельные окна каждой процедуре

Аппаратная оптимизация

Регистровые окна перекрываются:

- Правое поле предыдущего окна служит левым полем следующего
- Среднее поле – локальные переменные и константы
- Циклический буфер окон ограничивает вложенность



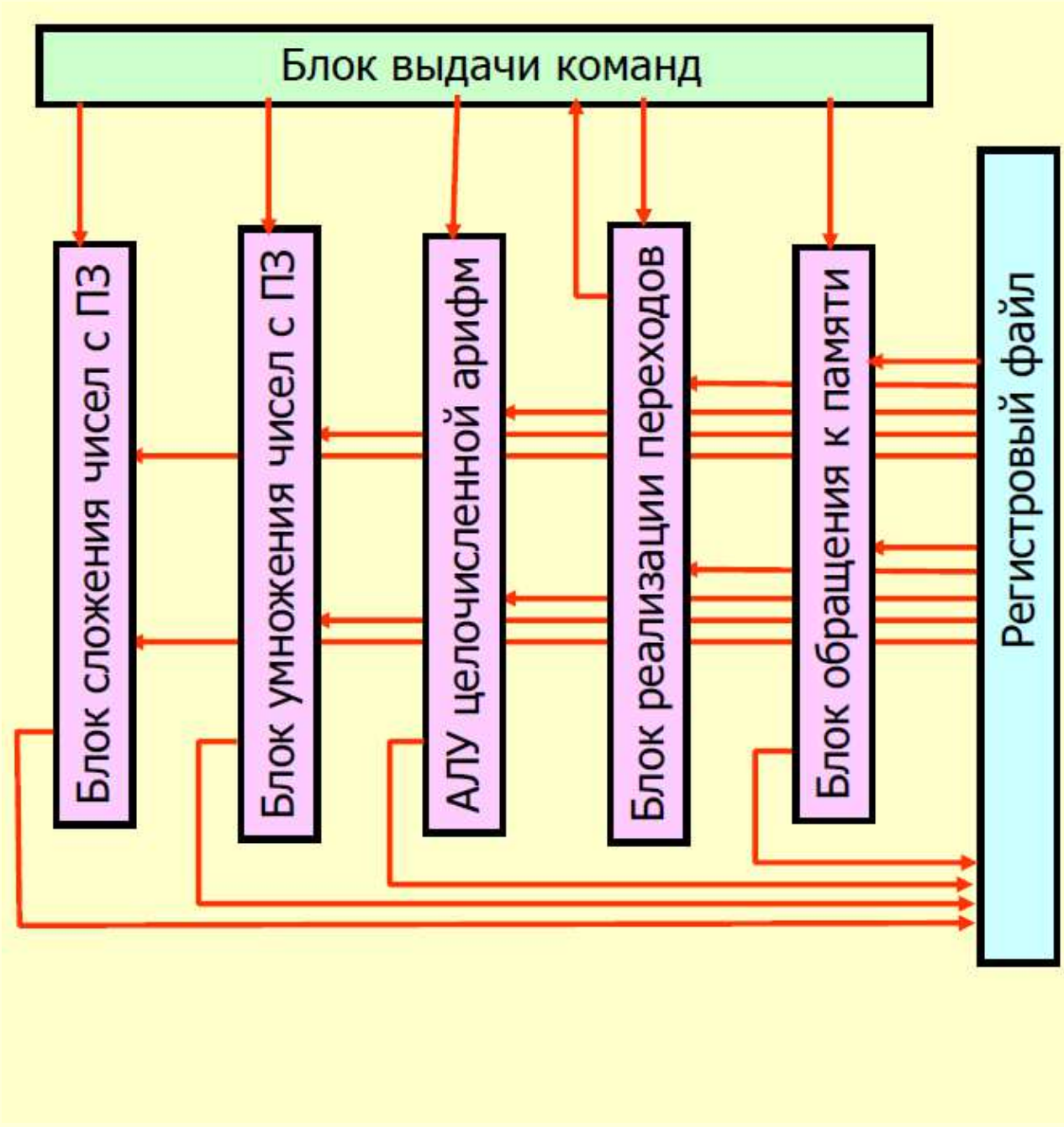
Преимущества и недостатки RISC

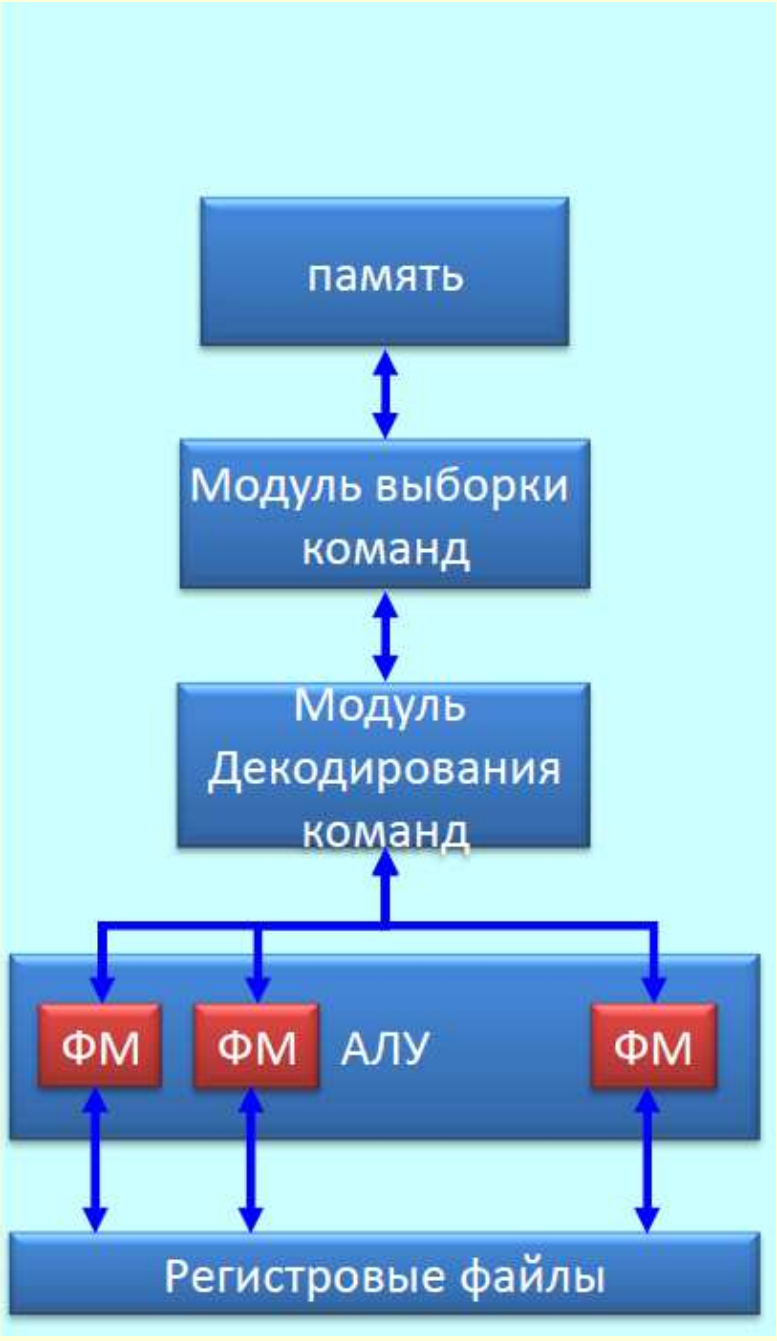
- Простая и надёжная структура устройств управления
- Хорошее быстродействие
  - унификация набора команд
  - упрощение передачи параметров в процедуры
- Малый набор команд
  - увеличение кода программы
  - больше время доступа к регистрам

Архитектура VLIW

- Архитектура с **командными словами сверхбольшой длины** или со **сверхдлинными командами** (VLIW, Very Long Instruction Word)
- Базируется на RISC-архитектуре
  - несколько простых команд объединяются в одну сверхдлинную и выполняются параллельно
    - число объединяемых команд равно числу ФБ
    - в одну сверхдлинную команду входят только команды, исполняемые разными ФБ
- 3-20 полей в команде (до 1024 бит)
- Оптимизация возлагается на компилятор

Архитектура VLIW





Архитектура VLIW

- VLIW – статическая архитектура
  - конфликты исключаются на этапе компиляции
  - можно существенно упростить аппаратуру процессора
    - высокое быстродействие
- Применение
  - большинство цифровых сигнальных процессоров

Архитектура EPIC

- **Вычислительные системы с явным параллелизмом команд** (EPIC - Explicitly Parallel Instruction Computing)
- Развивают идеи VLIW
  - IA-64 (Intel и Hewlett-Packard)
    - Intel Itanium
  - Команды упаковываются в сверхдлинную команду – связку (bundle) длиной 128 разрядов
    - 3 команды и шаблон зависимостей между командами

Архитектура EPIC

- Поле каждой команды в связке состоит из
  - 13-разрядного поля кода операции
  - 6-разрядного поля предикатов (номер одного из 64 регистров предиката)
    - для маркировки ветви исполнения программы
    - при вычислении условия перехода регистр получает значение 1 или 0
    - сохраняются только результаты команд нужной ветви
  - три 7-разрядных поля: два операнда и результат (РОН или регистр с плавающей запятой)

Особенности архитектуры EPIC

- Большое число регистров
- Масштабируемость архитектуры до большого числа ФБ
- Явный параллелизм в машинном коде
  - зависимости выявляет компилятор
- Предикация
  - маркировка и параллельное исполнение команд из разных ветвей
- Предварительная загрузка
  - данные из медленной памяти загружаются заранее

Особенности архитектуры EPIC

Преимущества:

- Упрощение аппаратуры, быстродействие
- Масштабируемость

Недостатки:

- Требуются продвинутые компиляторы
- Недостаточная гибкость программ