Классификация параллельных аппаратных архитектур по Флинну

Ниже приведена классификация аппаратных архитектур по Флинну.

- **SISD, Single Instruction stream over a Single Data stream**
  - ОКОД – вычислительная система с одиночным потоком команд и одиночным потоком данных. На рисунке 1 наглядно представлена схема архитектуры.

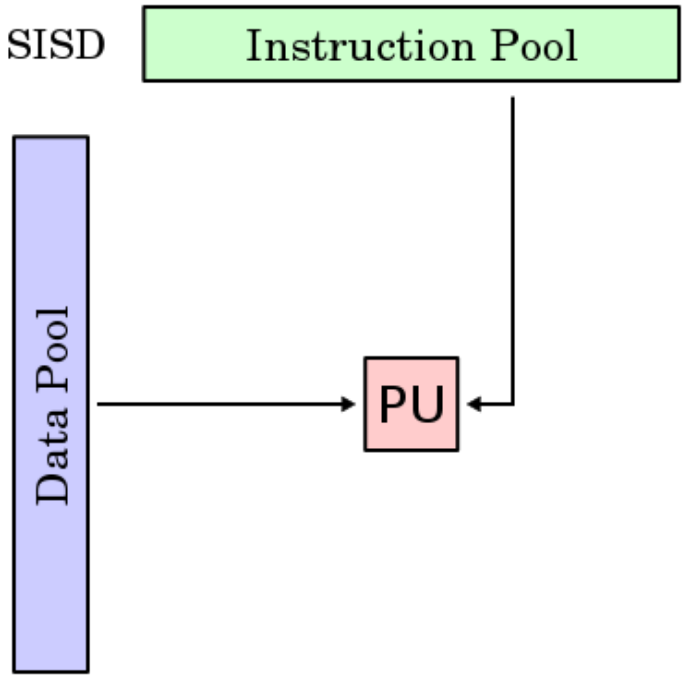

Рисунок 1. SISD архитектура

- **SIMD, Single Instruction, Multiple Data**
  - ОКМД – вычислительная система с одиночным потоком команд и множественным потоком данных. На рисунке 2 наглядно представлена схема архитектуры.

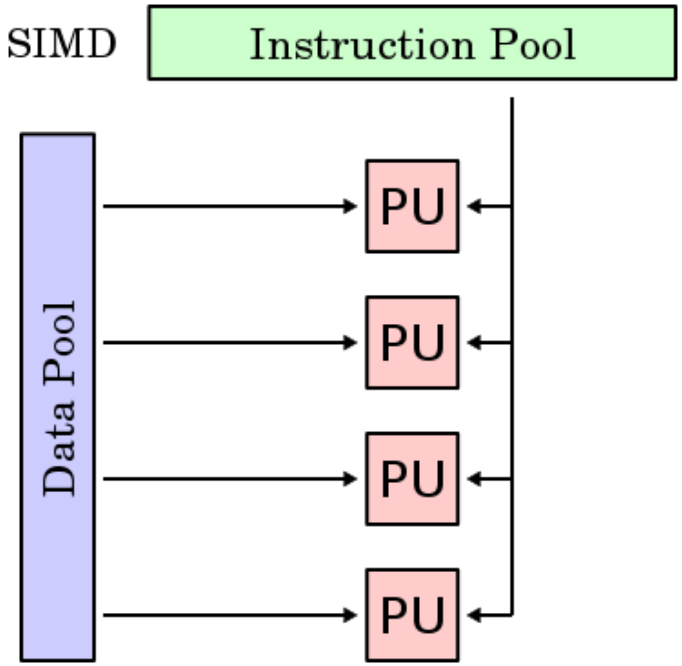

Рисунок 2. SIMD архитектура

1. Одна операция применяется сразу к нескольким элементам массива данных
2. Параллельные операции выполняются на всех доступных вычислителях
3. Обработкой данных управляет единственная программа

4. Пространство имён является глобальным
5. Используются специализированные языки программирования и/или библиотеки, предназначенные для конкретной архитектуры

- **MISD, Multiple Instruction Single Data**
  - МКОД – вычислительная система со множественным потоком команд и одиночным потоком данных. На рисунке 3 наглядно представлена схема архитектуры.

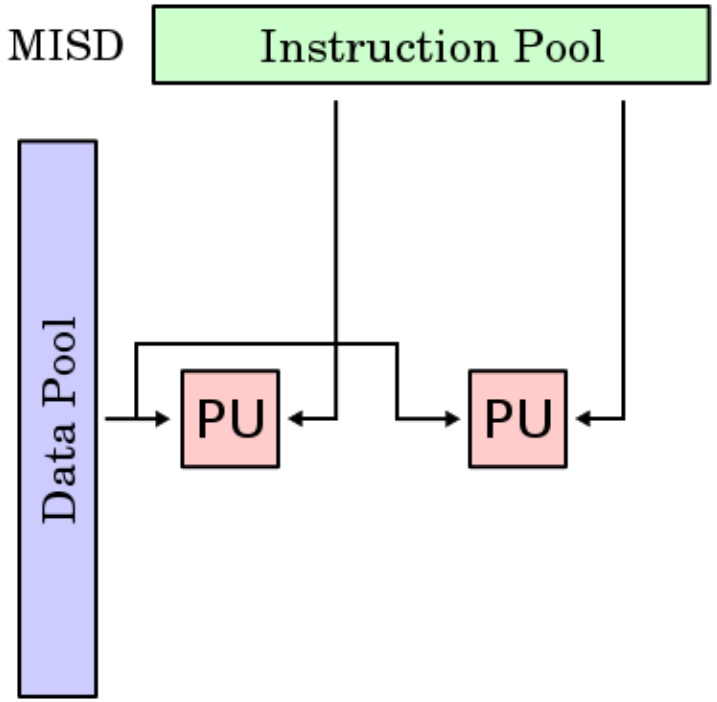

Рисунок 3. MISD архитектура

- MIMD, Multiple Instruction Multiple Data
  - МКМД – вычислительная система со множественным потоком команд и множественным потоком данных. На рисунке 4 наглядно представлена схема архитектуры.

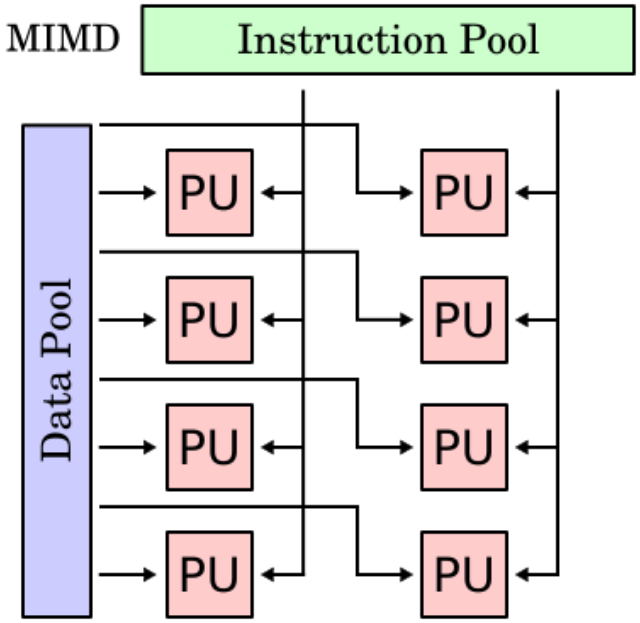

Рисунок 4. MIMD архитектура

1. Задача разбивается на несколько самостоятельных подзадач
2. Для каждой подзадачи разрабатывается отдельная программа
3. Программы подзадач выполняются на отдельных вычислителях
   a. Для этого используются обычные языки программирования

4. Программы подзадач должны обмениваться информацией
    a. Для этого используются функции специализированной библиотеки
5. Преимущество
    a. Теоретическая возможность достижения наивысшего быстродействия, гибкость

- Дополнительная информация http://en.wikipedia.org/wiki/Flynn%27s_taxonomy

1.1 Характеристики последовательных программ

Ниже приведены основные характеристики последовательных программ.

- Неоптимальное использование аппаратных средств
  - Невысокая производительность
- Использование традиционных (стандартных) средств программирования
  - Языков программирования
  - Библиотек функций/классов
  - Средств отладки и тестирования
- Инвариантность к возможностям параллелизма аппаратных средств и платформ
  - Относительно лёгкая переносимость

На рисунке 5 представлена схема последовательной программы.

Рисунок 5. Схема последовательной программы

1.2 Характеристики параллельных программ

Ниже приведены основные характеристики последовательных программ.

- Оптимизация использования аппаратных средств
  - Возможность преодолеть ограничение производительности последовательных программ
- Использование средств программирования, не используемых при разработке последовательных программ
  - Языков программирования
  - Библиотек функций/классов
  - Средств отладки и тестирования
- Повышенная сложность проектирования и разработки
- Ориентация на возможности параллелизма конкретных аппаратных средств и платформ
  - Переносимость затруднена или невозможна

На рисунке 6 представлена схема параллельной программы.

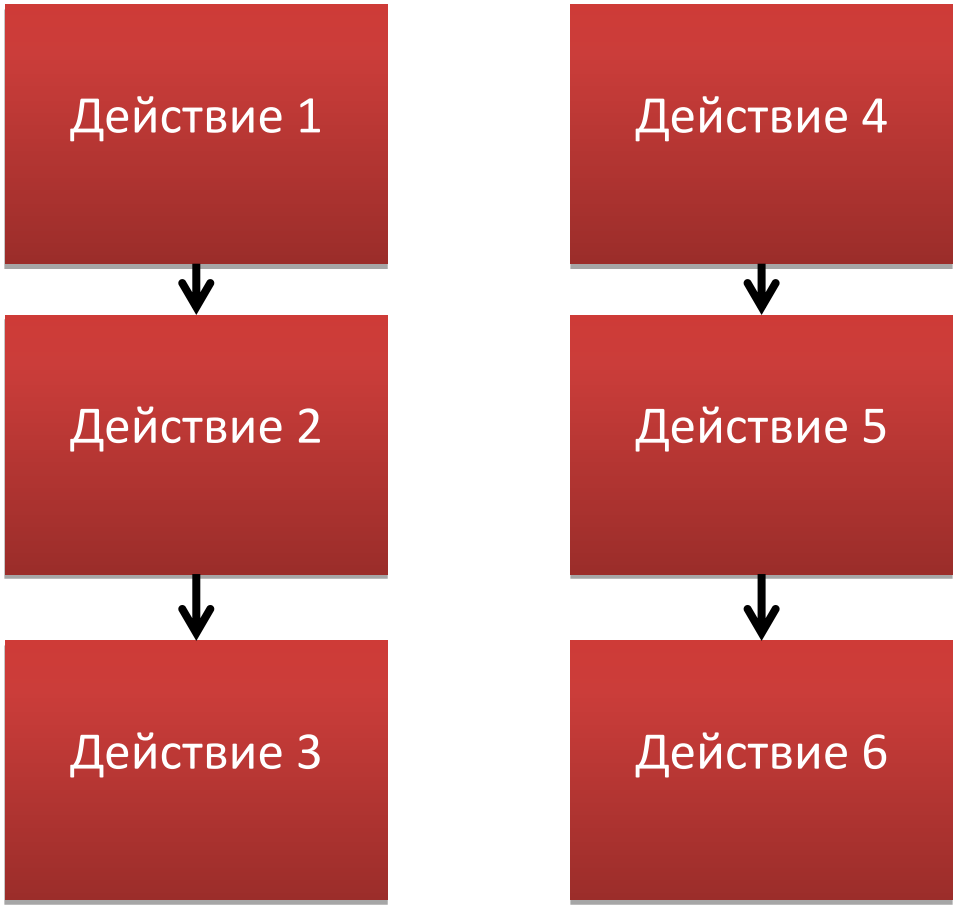

Рисунок 6. Чистая параллельная программа

1.3 Особенности разработки параллельных программ

Основные особенности разработки параллельных программ.

- Управление параллельно выполняющимися действиями
- Обеспечение совместного использования общих ресурсов параллельно выполняющимися действиями
- Необходимо выявлять и устранять ошибки, характерные только для параллельных программ
  - Взаимные блокировки
  - Гонки
- Выявленные ошибки трудно повторить
  - Часто проявляются архитектурные ошибки
  - Не всегда возможно использовать отладчик
- Необходимо обеспечивать масштабируемость и балансировку загрузки аппаратных средств

На рисунке 7 представлена схема реальной программы с элементами параллелизма.

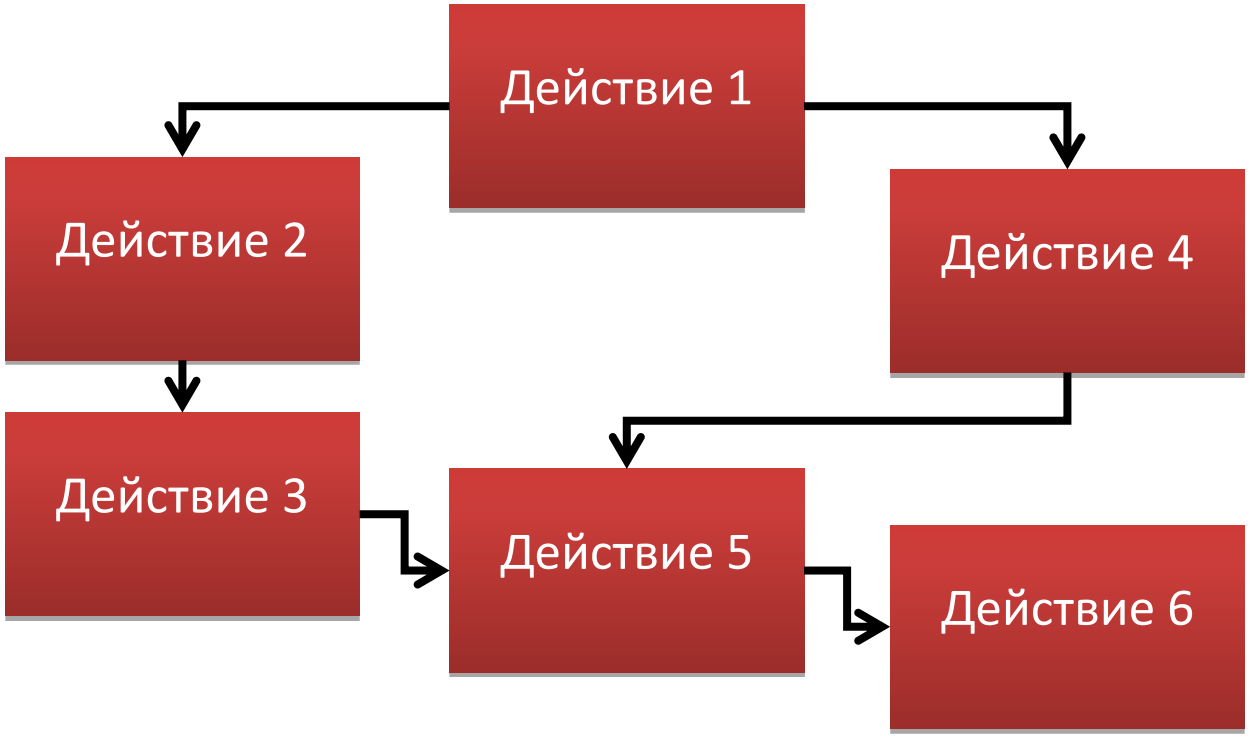

Рисунок 7. Реальная программа с элементами параллелизма

Понятия процессов и потоков. Основные определения

Поток – последовательность команд программы, выполняющихся одна за другой в детерминированной последовательности.

Основной принцип потока – несколько потоков могут выполняться параллельно (псевдопараллельно).

Поток (поток управления, задача, нить, thread) – одна из параллельно (асинхронно) выполняющихся ветвей процесса

- Особенности потоков
  - В процессе присутствует единственный главный поток
  - Все потоки одного процесса работают в едином адресном пространстве
    - Общие переменные и код
    - Нет необходимости использовать специальные средства взаимодействия
    - Каждый поток имеет собственный стек
    - Каждый поток имеет собственное состояние

Процесс – выполнение пассивных инструкций компьютерной программы на процессоре.

Процесс – совокупность взаимосвязанных и взаимодействующих действий, преобразующих входные данные в выходные (ISO 9000:2000).

Процесс порождается при запуске программы.

Состав процесса:

- Главный поток
- Дополнительные потоки – необязательно
- Память данных
- Память программ

Все потоки в рамках одного процесса выполняются в едином адресном пространстве, то есть используют общие переменные. Каждый поток имеет собственный стек. Поток может порождать другие потоки.

Процесс (process) – совокупность действий процессора и необходимых ресурсов для обеспечения выполнения инструкций программы

- Состав процесса
  - Области памяти данных и программ
  - Стек
  - Отображение виртуальной памяти на физическую память

- Состояние

Типичные состояния процессов:

- Остановлен
  - Процесс не использует процессор
- Терминирован
  - Процесс закончился, но ещё не удалён
- Ожидает
  - Процесс ждёт событие
- Готов
  - Готов к выполнению, но ожидает освобождения процессора
- Выполняется
  - Процесс выполняется процессором

На рисунке 8 представлен график состояний процесса.

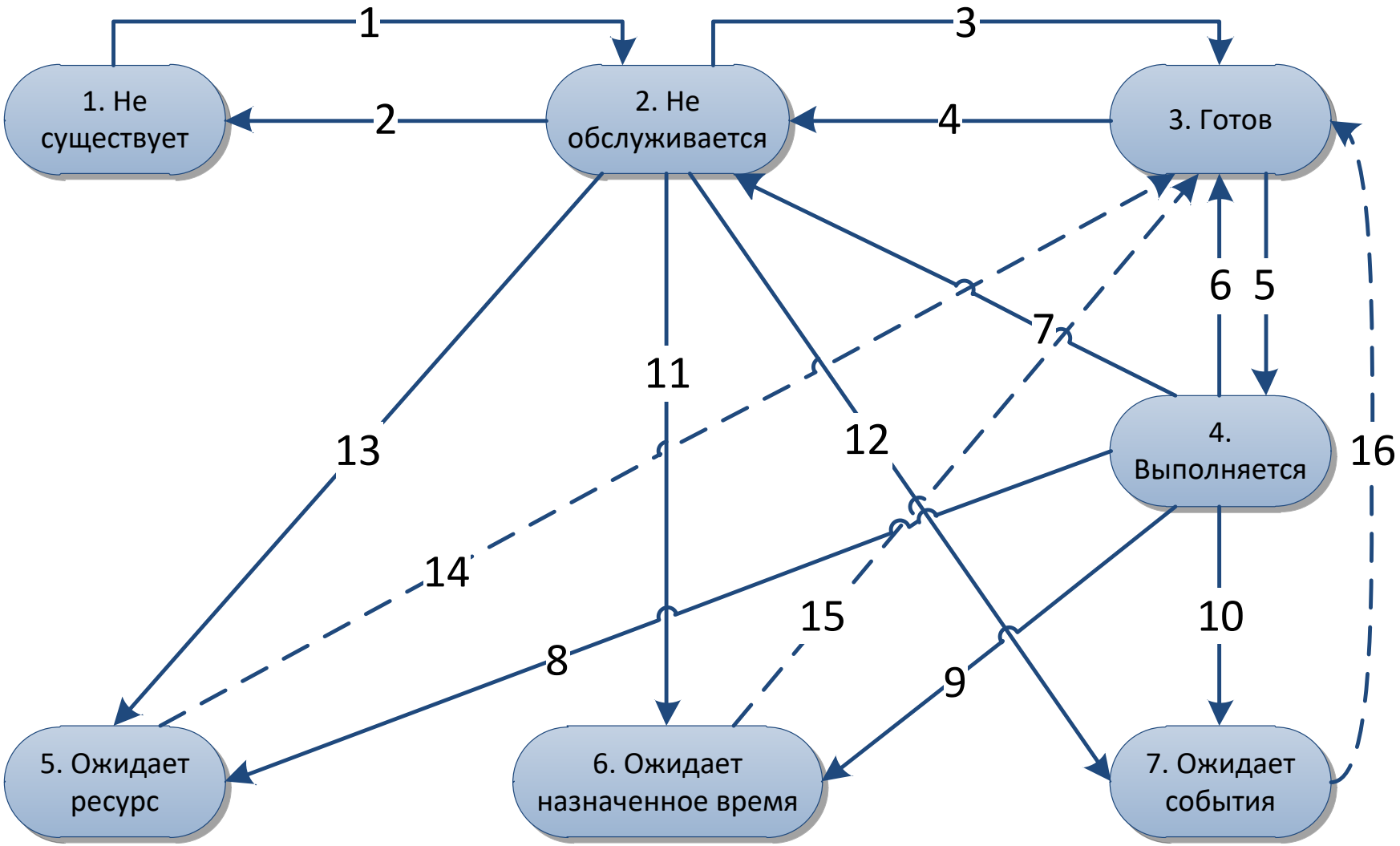

Рисунок 8. График состояний процесса

Межпроцессное взаимодействие – способ передачи информации между процессами.

- Виды межпроцессного взаимодействия
  - Разделяемая память
  - Семафоры
  - Сигналы
  - Почтовые ящики

Событие – оповещение процесса со стороны ОС о возникновении межпроцессного взаимодействия.

Примеры событий:

- Принятие семафором требуемого значения
- Поступление сигнала
- Поступление сообщения в почтовый ящик

Ресурс – объект (устройство, память), необходимый процессу или потоку для выполнения заданных действий

Приоритет – целое число, определяющее важность каждого процесса или потока в ОС:

- Чем больше приоритет у процесса или потока, тем больше процессорного времени ему будет выделено

Виды ресурсов по природе:

- Аппаратные ресурсы
  - Процессор
  - Область памяти
  - Периферийные устройства
  - Прерывания
- Программные ресурсы
  - Программа
  - Данные
  - Файлы
  - Сообщения

Виды ресурсов по характеристикам:

- Активные
  - Способны изменять и создавать информацию (процессор)
- Пассивные
  - Способны хранить информацию (память)
- Локальные
  - Относятся к одному процессу
  - После завершения процесса автоматически удаляются

- Разделяемые
  - Относятся к нескольким процессам
  - Удаляются только после окончания использования их последним процессом
- Постоянные
  - Для работы требуют операций «захватить» и «освободить»
- Временные
  - Для работы требуют операций «создать» и «удалить»

Виды разделяемых ресурсов:

- Некритичные
  - Безопасно могут быть использованы одновременно несколькими процессами и потоками
  - Примеры: жёсткий диск в целом, сетевая карта, видеокарта
- Критичные
  - Безопасно могут быть использованы в один момент времени только одним процессом или потоком
  - Примеры: разделяемая память при её модификации

Типы взаимодействия процессов:

- Независимые процессы

- Процессы не используют разделяемые ресурсы

- Сотрудничающие процессы
  - Процессы разделяют канал коммуникации: один пишет, другой читает
  - Процессы работают по очереди: один работает, второй ожидает завершения работы первого
- Конкурирующие процессы
  - Процессы используют совместно разделяемый ресурс
  - Процессы используют критические секции
  - Процессы используют взаимные исключения (мьютексы)

Критическая секция – участок программного кода, который допускается выполнять только единственным потоком (процессом)

Взаимное исключение (мьютекс, mutual exclusion, mutex) – способ синхронизации потоков за счёт использования захвата совместно используемого ресурса, также называемого мьютексом

- Если мьютекс занят, то при попытке его захвата поток переходит в состояние ожидания
- Как только мьютекс освобождается, ранее ожидавший поток высвобождается, а мьютекс вновь считается захваченным

Безопасное взаимодействие – целостность информации и неделимость действий при взаимодействии обеспечиваются операционной системой

Небезопасное взаимодействие – целостность информации и неделимость действий при взаимодействии обеспечиваются приложением

Разделяемая память – область памяти, одновременно доступная для нескольких процессов.

- Это базовый вид взаимодействия процессов, к которому сводятся все остальные виды взаимодействия
- Разделяемая память может отображаться на разные области виртуальной памяти, поэтому нужно преобразовывать указатели

Операции:

- Создать – создаётся объект (файл), недоступный для использования
- Подсоединить – созданный объект разделяемой памяти присоединяется к адресному пространству процесса, после этого разделяемой памятью можно пользоваться для обмена данными
- Отсоединить – объект разделяемой памяти отсоединяется от адресного пространства процесса
- Удалить – процесс сообщает ОС о том, что больше не будет использовать разделяемую память, реально объект разделяемой памяти будет удалён после окончания его использования последним процессом

Семафоры - это объект синхронизации, задающий количество процессов и/или потоков, имеющих одновременный доступ к разделяемому ресурсу

- По сути это безопасный счётчик
- Значение счётчика может быть меньше 0 – это значит, что несколько потоков/процессов ожидают освобождения семафора
- Виды
  - Двоичные (булевские)
  - Счётные

Операции с семафорами:

- Взять (Get) – уменьшает значение счётчика на k, если в счётчике значение не меньше k, иначе поток блокируется
- Вернуть (Put) – увеличивает значение счётчика на k
- Попробовать взять (TryGet) – то же, что и Get, но не блокирует поток
- Проверить (Test) – возвращает значение счётчика
- Блокировать (Lock) – «обнулить», если значение счётчика больше 0
- Разблокировать (Unlock) – вернуть столько, сколько сняли при блокировке.

Mutex (mutual exclusion) – механизм синхронизации, предназначенный для устранения недостатков семафоров:

- Захват семафора после его случайного разблокирования всегда удачен и может привести к повреждению данных
- Семафор является объектом ОС, поэтому для доступа к нему необходимо переключать задачи

- Состав мьютекса
  - Булевский семафор
  - Идентификатор потока, захватившего разделяемый ресурс
- Мьютексы хранятся в памяти процесса (локальные) или в разделяемой памяти процессов (глобальные)
- Операции
  - Захватить (Lock) – если мьютекс захвачен другим потоком, текущий поток блокируется
  - Освободить (Unlock) – работает только для потока, являющегося владельцем мьютекса
  - Попробовать захватить (TryLock) – то же, что и Lock, но текущий поток не блокируется

Переключение задач в многозадачных ОС. Многозадачность

Многозадачность (multitasking) – свойство ОС или среды исполнения обеспечивать возможность параллельной (или псевдопараллельной) обработки нескольких процессов.

Настоящая многозадачность возможна только в системах с несколькими процессорами (ядрами).

Типы многозадачности:

- Процессная – планировщик ОС может управлять процессами
- Поточная – планировщик ОС может управлять процессами и потоками

Многозадачность называется **невытесняющей**, когда

- Загружаются в память два или более приложений
- Процессорное время предоставляется только основному приложению
  - Второе приложение становится фоновым
- Для выполнения фонового приложения его необходимо активизировать

Многозадачность называется **кооперативной (совместная),** когда

- Для выполнения следующей программы текущая программа должна объявить о готовности отдать процессор

Преимущества такого вида многозадачности:

- Не нужно защищать совместно используемые ресурсы
- Легко преобразовать обычные программы в параллельные

Недостаток такого вида:

- Выполняющаяся задача может не отдать процессор и остальные задачи не смогут выполниться
- Трудно организовать многопоточную архитектуру ввода-вывода в ядре ОС в случае:
  - Одна задача инициировала операцию ввода-вывода и ждёт её завершения
  - В это время следует переключиться на другую задачу

Многозадачность называется **вытесняющей,** когда

- Для выполнения следующей программы ОС принудительно приостанавливает выполнение текущей программы
- Планировщик задач
- Эффективное использование приоритетов
- Преимущества:
  - Повышение надёжности
  - Возможность реализовать многопоточный ввод-вывод в ядре ОС

    - Возможность использования многоядерных и многопроцессорных систем
- Недостатки:
    - Усложняется разработка приложений
    - Возникает требование реентерабельности
    - Необходимость защиты совместно используемых ресурсов

В приложении 1 можно ознакомиться с примерами параллельных программ.