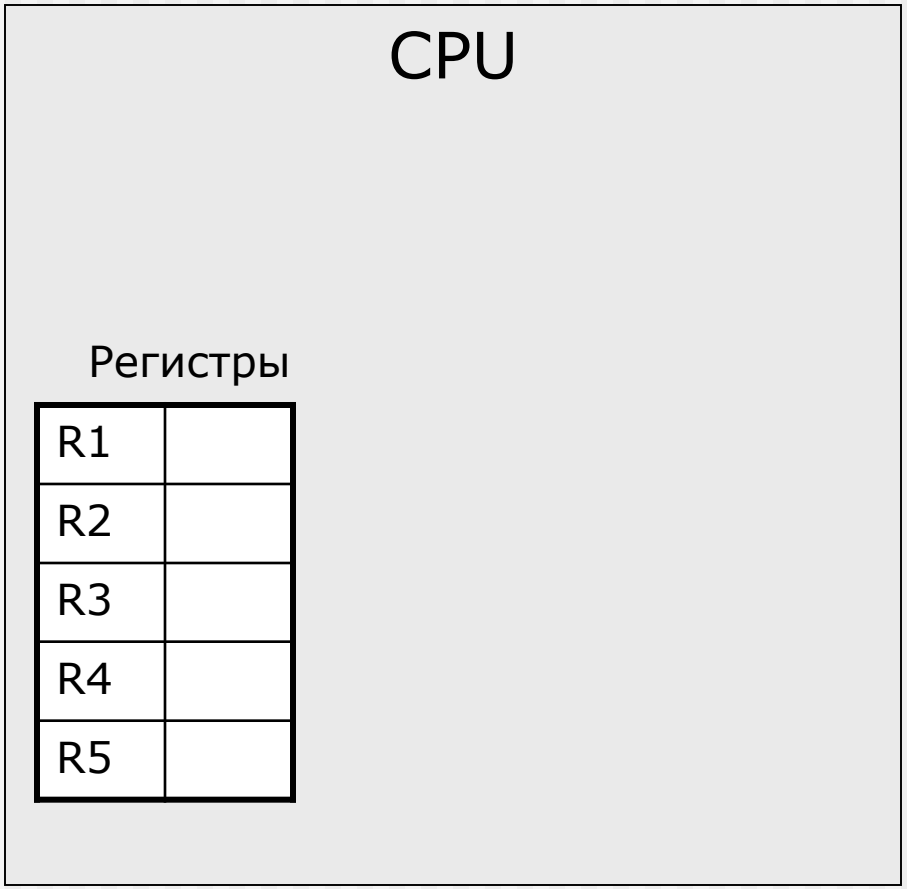
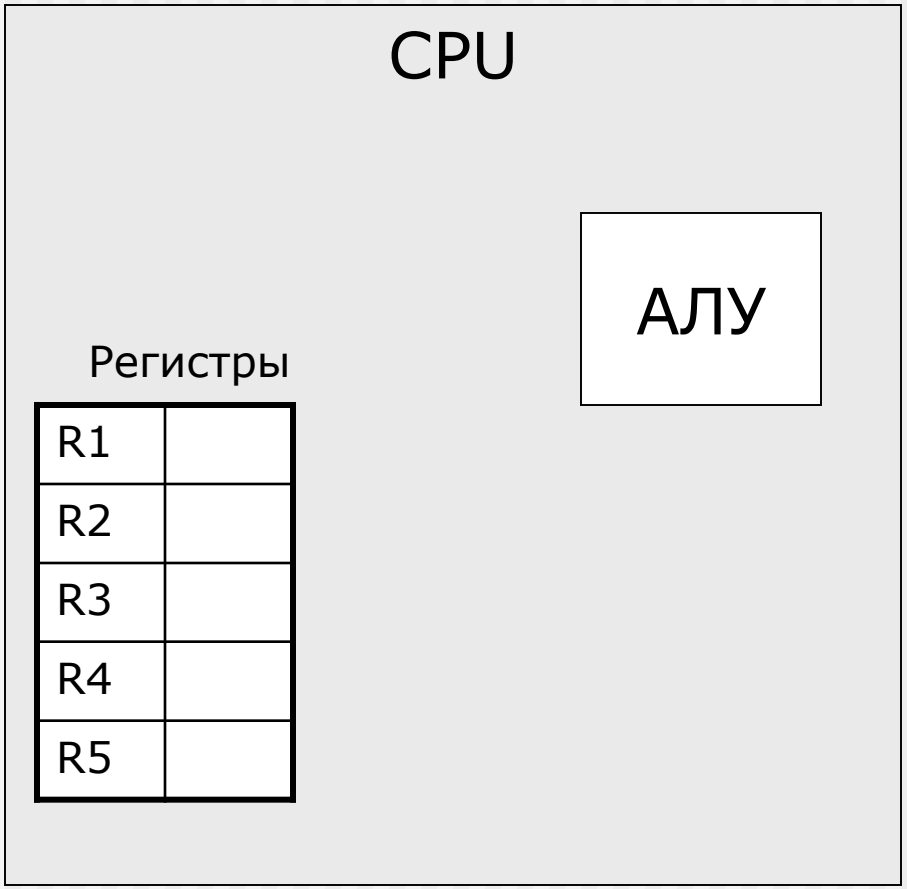
ОПЕРАЦИОННЫЕ СИСТЕМЫ

Несколько слов об архитектуре ЭВМ

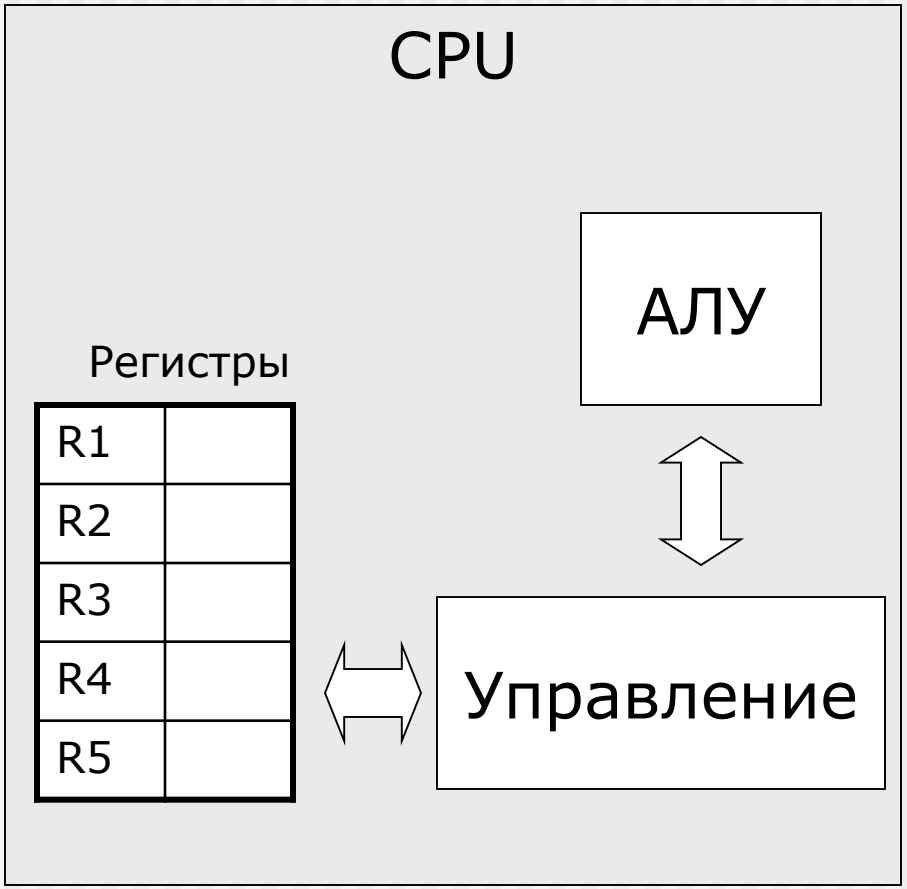
Введение в архитектуру ЭВМ



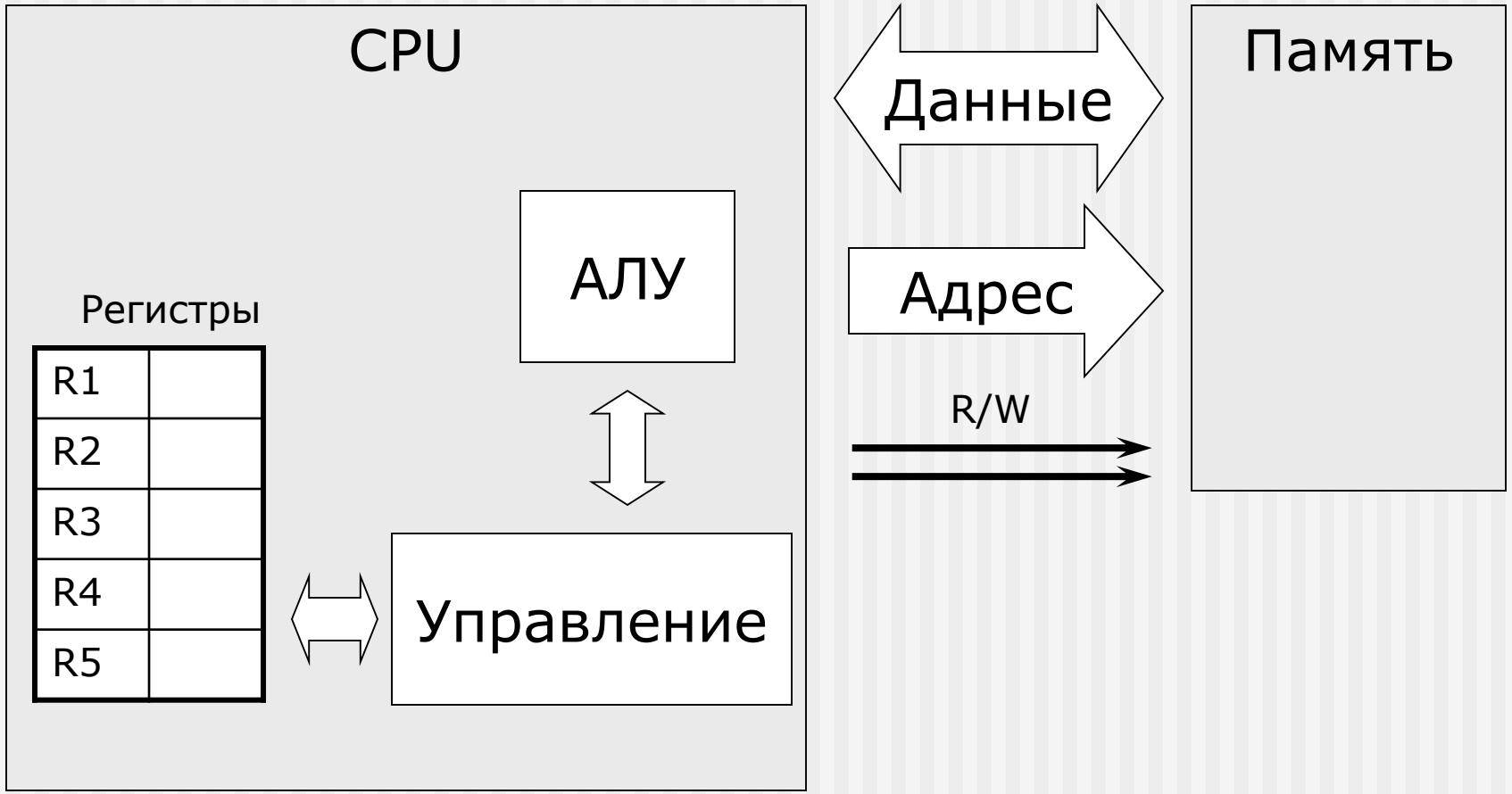
Введение в архитектуру ЭВМ



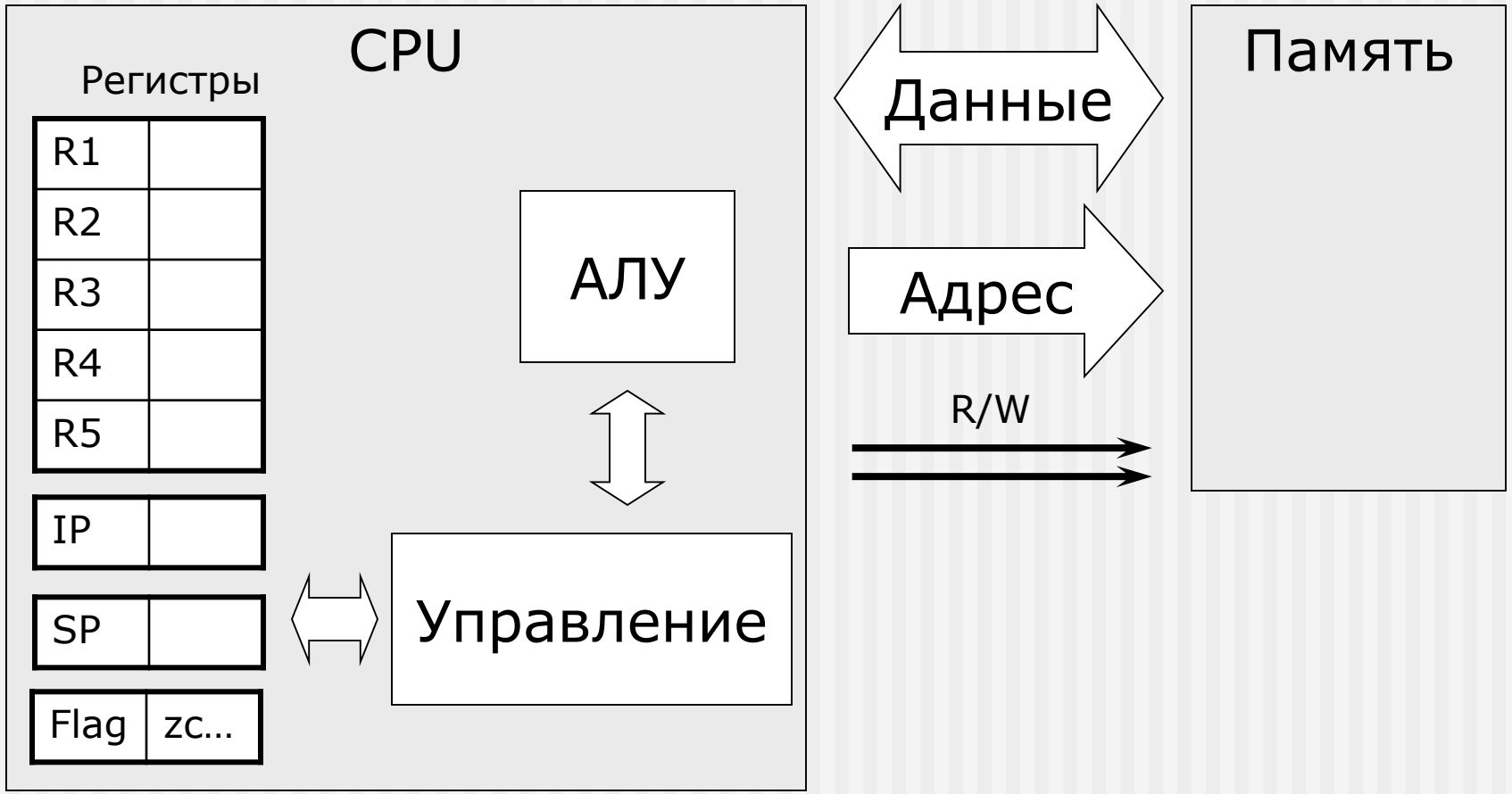
Введение в архитектуру ЭВМ



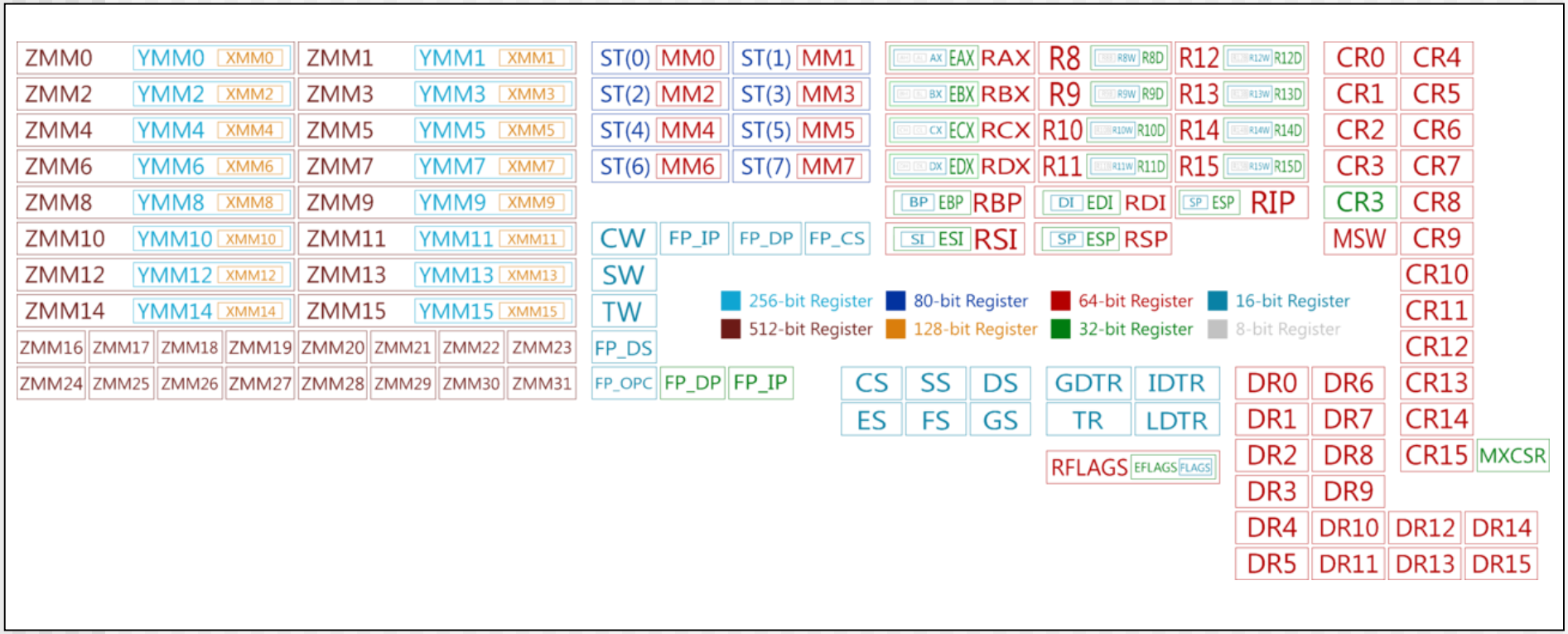
Введение в архитектуру ЭВМ



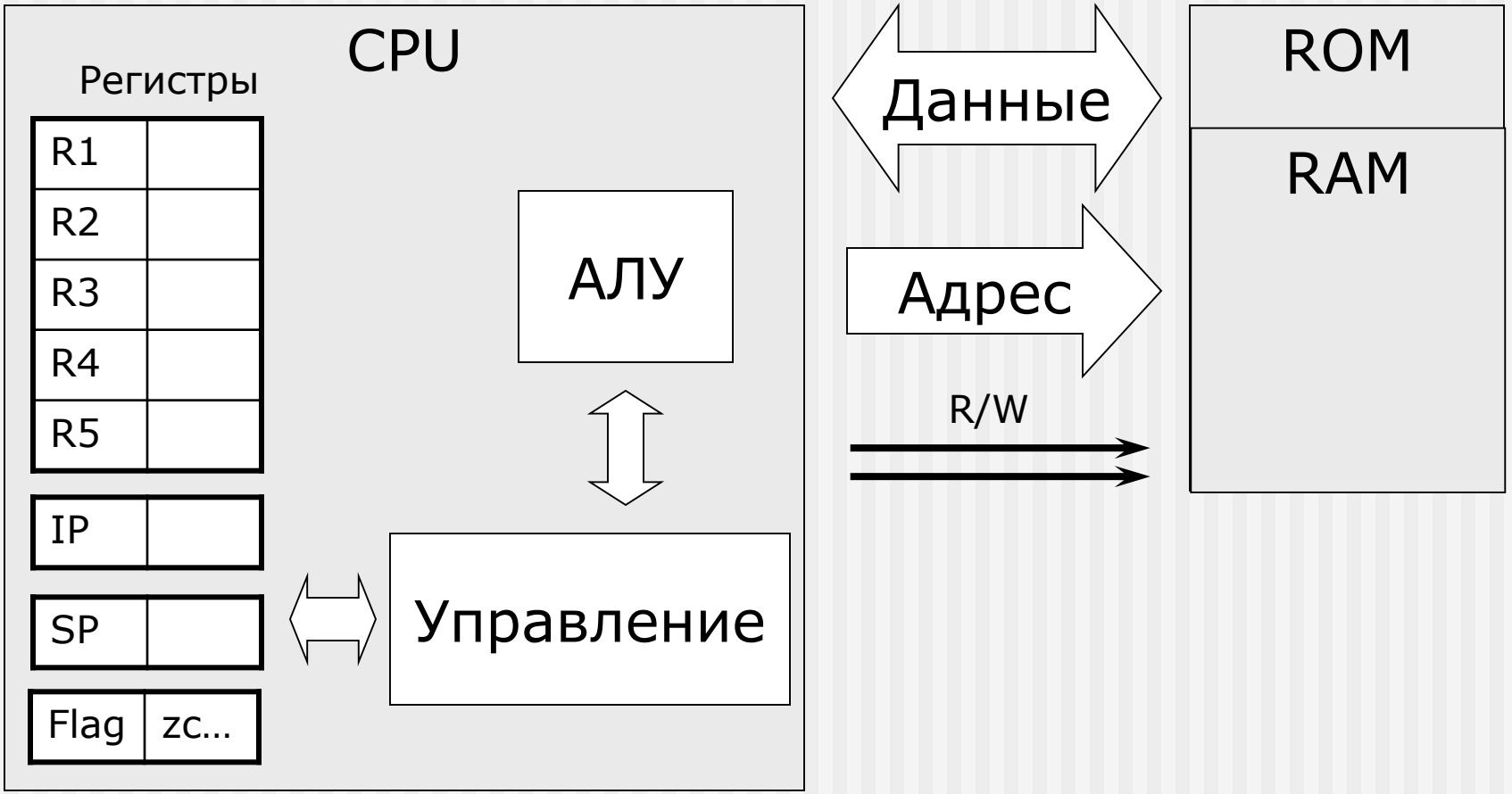
Введение в архитектуру ЭВМ



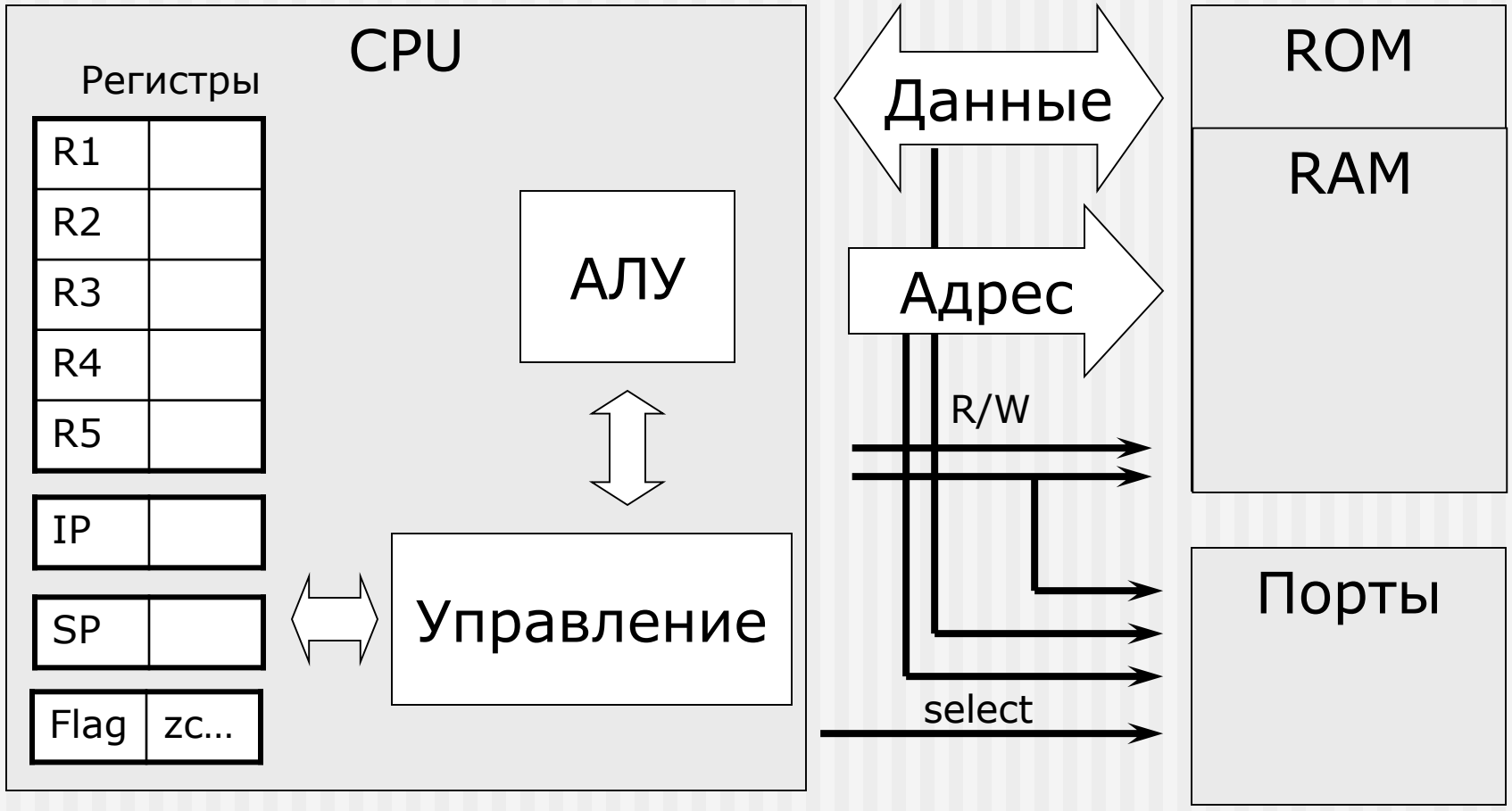
Регистры современного процессора архитектуры x86



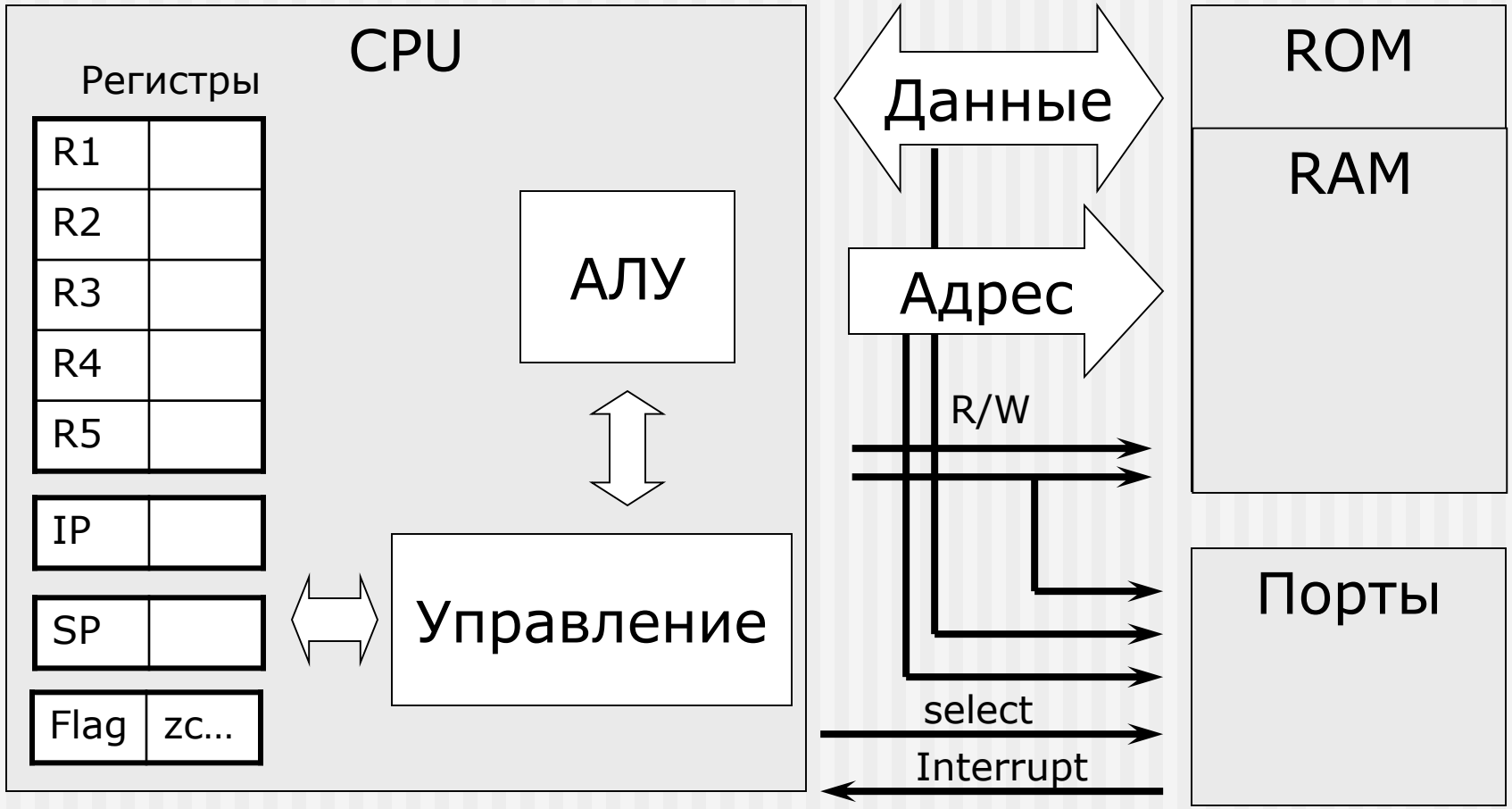
Введение в архитектуру ЭВМ



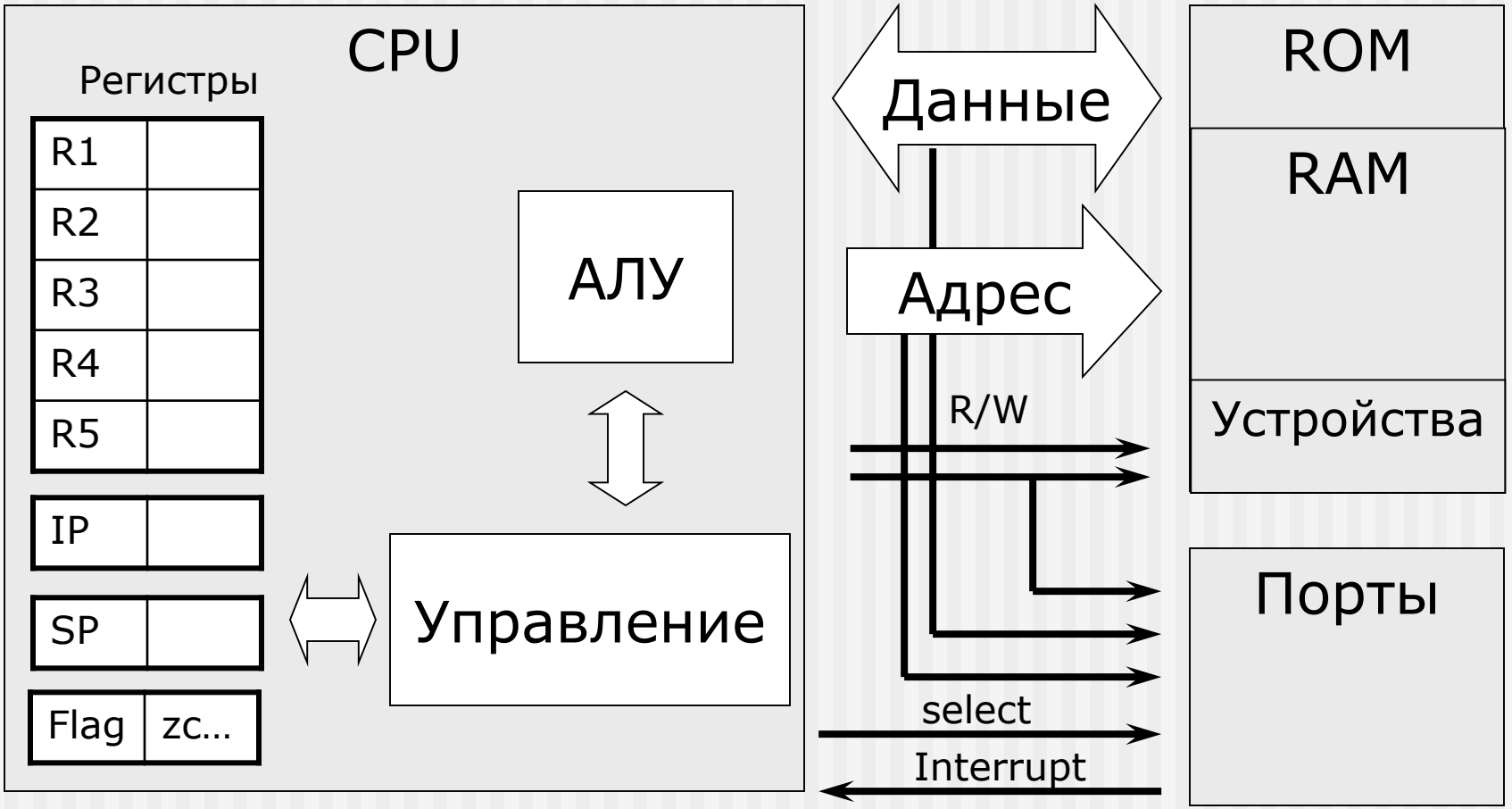
Введение в архитектуру ЭВМ



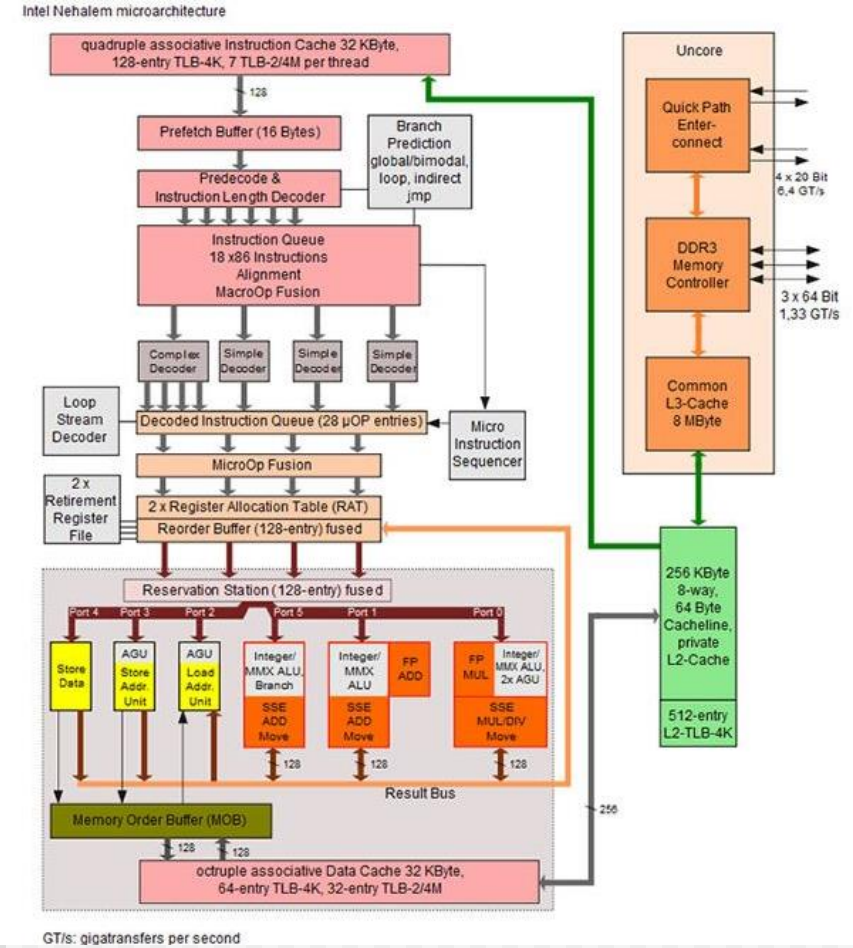
Введение в архитектуру ЭВМ

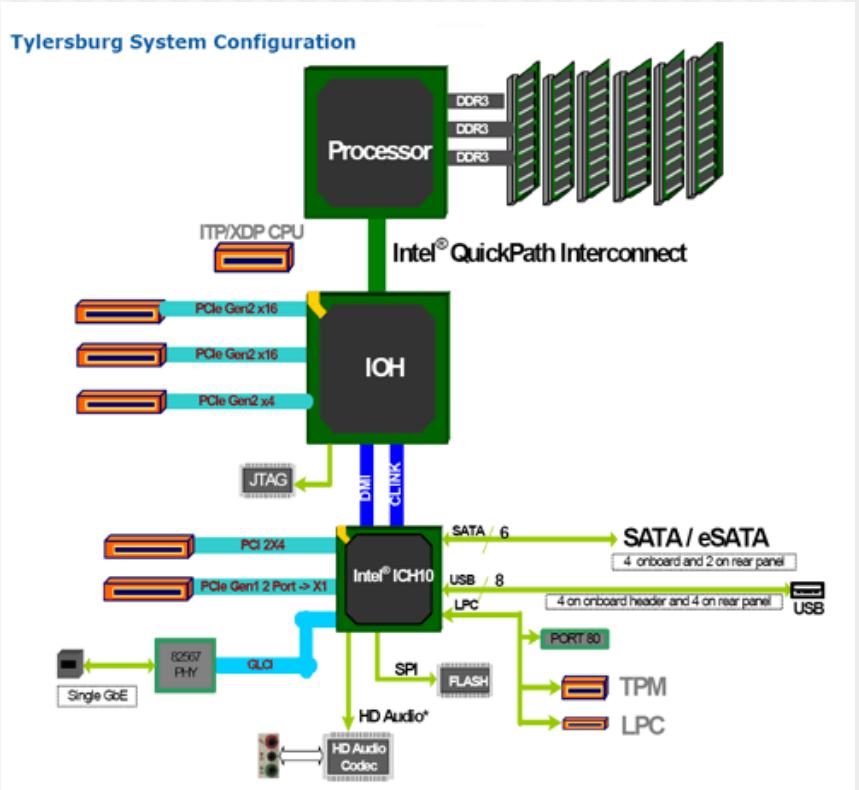


Введение в архитектуру ЭВМ



Введение в архитектуру ЭВМ





Уязвимость Meltdown

- Современные процессоры и ОС позволяют сделать так, что:
  - каждый процесс работает в виртуальном адресном пространстве, как будто он запущен один на компьютере;
  - для разных областей памяти могут быть установлены разные права на чтение/запись.
- Для ускорения вызовов ОС в часть виртуального адресного пространства каждого процесса отображена память ядра ОС, закрытая к доступу из процесса.
- Для ускорения работы:
  - осуществляется кеширование обращений к оперативной памяти;
  - процессор выполняет много инструкции одновременно (заранее).
- Оказалось, что эти ускорения позволяют полностью и незаметно обойти защиту на чтение любой памяти из адресного пространства процесса, в том числе ядра ОС.

Уязвимость Meltdown

1. Создаётся массив M размером 256*4096 элементов. Убеждаемся, что массив не закеширован.
2. Выполняем как транзакцию:
    1. a = байт из памяти ядра
    2. b = M[a*4096]
3. Проверяем время доступа к элементам $M[i * 4096], i \in 0{:}\,255$.
4. Число из памяти ядра = i для которого время доступа было минимально.

Транзакция при ошибке выполнения одной из команд завершается тихо и не меняет значения переменных.

Не должна сработать, но выполняется заранее процессорами Intel перед проверкой прав доступа.

Выполняется заранее. Результат не сохраняется из-за ошибки в предыдущей команде. Но процессор успел обратиться к M[a*4096] и эта память теперь закеширована.

Уязвимость Spectre

1. Обманывает блок предсказания ветвлений и заставляет процессор спекулятивно выполнять код в чужом процессе.
2. Результат получает так же, замеряя время доступа к памяти.
3. Позволяет читать память любых процессов, работающих на одном ядре с нападающим.
4. Сложнее использовать:
    1. надо найти в атакуемом процессе подходящий код;
    2. надо иметь возможность закинуть ему данные (через файлы, ввод с клавиатуры, …);
    3. …
5. Есть на всех быстрых процессорах.