Lectures

Tags | Operation systems

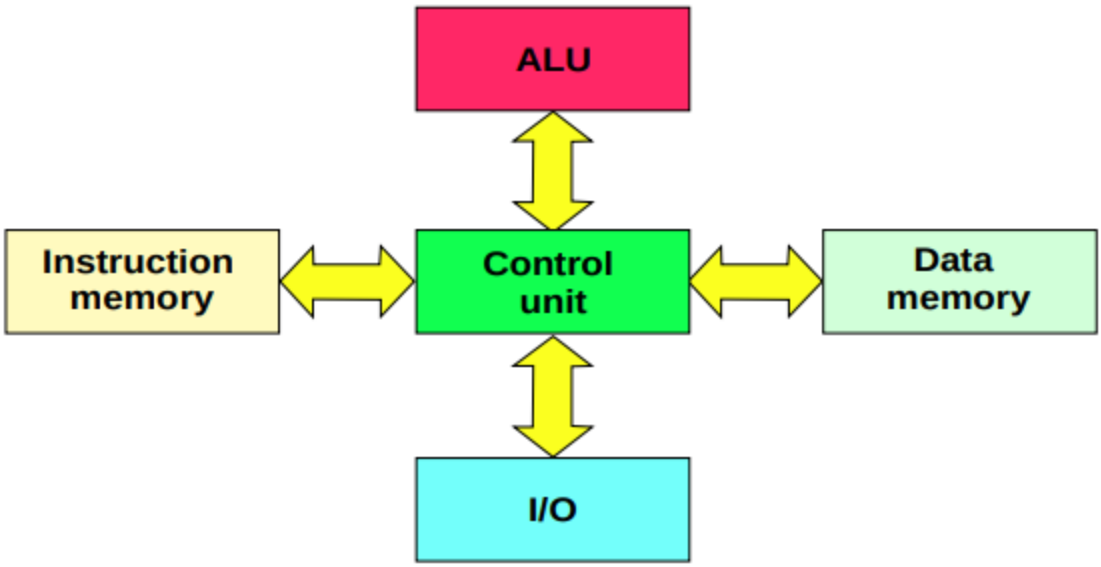
Course preview.

За основу взят курс Столинкса (Stallings William), он пишет хорошо. Две лабы по 20 баллов, рубежка на 20, две микроконтрольных по 5 баллов и экзамен на 30. Вопросов на данный вопрос ~110, но раздует до ~140 примерно. Мораль: надо ботать теорию много… Первая лаба на монитоинг и профилирование, вторая на написание модуля для ядра ОС, который позволяет обойти некоторые ее барьеры.

Лекция #1. Архитектура компьютерных систем.

Традиционно, архитектура компьютерных систем делятся на два вида: Гарвардская и фон-Неймановская:

Гарвардская архитектура:

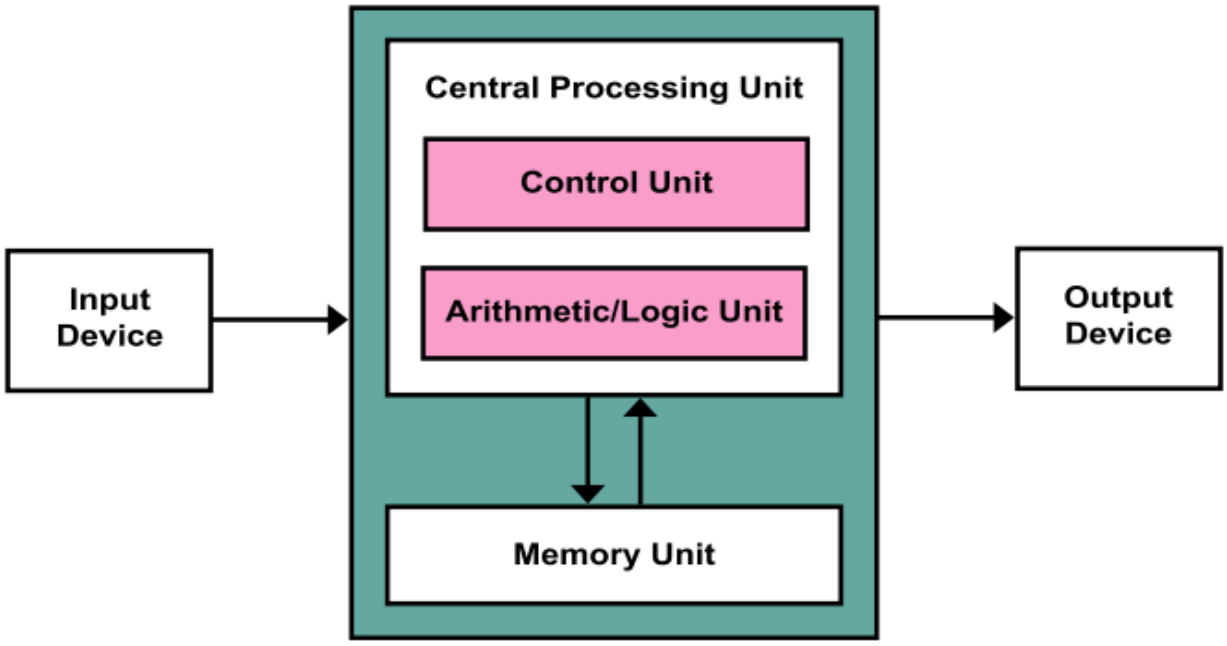


Гарвардская архитектура

Основные особенности:

1. Память данных и память команд разделены. Это физически разные устройства.
2. Каналы “общения” данных и команд с CU тоже физически разделены. Это физически разные шины.
3. В архитектуре фон Неймана процессор в каждый момент времени может либо читать инструкцию, либо читать/записывать единицу данных из/в памяти. Оба действия одновременно происходить не могут, поскольку инструкции и данные используют один и тот же поток (*шину*). В компьютере с использованием гарвардской архитектуры процессор может считывать очередную команду и оперировать памятью данных одновременно и без использования кэш-памяти. Исходя из физического разделения шин команд и данных, разрядности этих шин могут различаться и *физически* не могут пересекаться.

Архитектура фон Неймана:



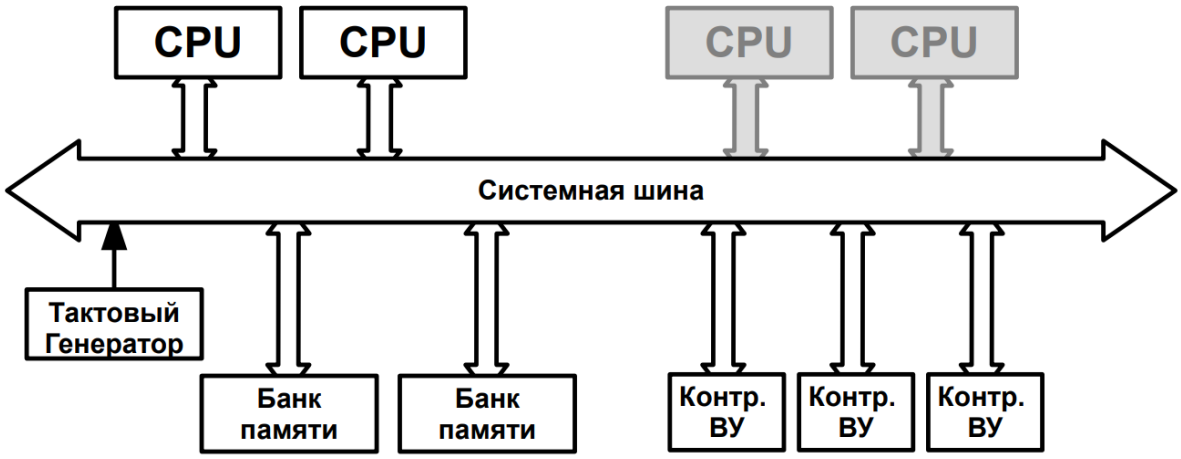
Архитектура фон Неймана

Основные особенности:

1. Принцип однородности памяти. Данные и команды хранятся в одном memory unit.
2. Принцип адресуемости памяти. Вся память - это набор “пронумерованных” ячеек, и процессор имеет доступ к каждой из них.
3. Принцип программного управления. Вычисления представлены в виде программы, которая состоит последовательности команд.
4. Принцип двоичного кодирования. И данные, и информация кодируются двоичными 0 или 1.

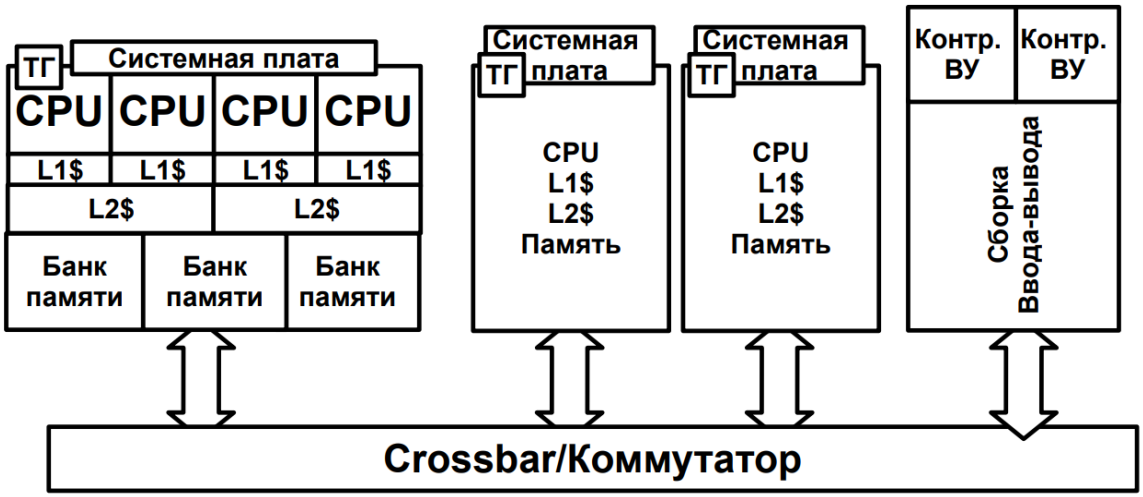
Итак, фон-неймановская архитектура строится вокруг памяти. И с памятью работать можно по-разному, на основе чего также возникает два вида архитектуры.

1. UMA — uniform memory access. Все устройства являются одноранговыми, подкючены к одной общей шине и лазают по этой шине в память. То есть процессор, обращаясь к памяти, пройдет столько же системных уровней, как, например, контроллер ВУ. Очевидно, что задержки в таком случае будут одинаковые, очевидно также, что достоинство данной системы - это ее простота. Но возникает проблема с шиной. Допустим, процессора два. Один загрузил себе в кеш страницу инструкций, чтобы с ними быстрее работать, а второй процессор использует шину для работы с ВУ, например. Все ок. Но добавление процессоров, или другое использование тех двух приводит к взаимной блокировки процессоров, одному приходится ждать, пока с шины уйдет другой. Неэффективно.



Архитектура UMA

2. NUMA — non-uniform memory access. Процессоры были разбиты на различные системные платы, на каждой из которых теперь располагаются кеши 1-2, возможно 3 уровня и банк памяти. То есть далеко ходить по шине в банк память больше не нужно. Однако можно сходить и на другую системную плату, чтобы памяти было побольше, чем есть на одной плате. Однако в таком случае нужно быть готовым к тому, что разные куски одной программы пройдут за разное время, к примеру, итерация по части массива пройдет быстро, а по части — долго, потому что часть элементов хранится в банке памяти на другой системной плате. Роль шины стал выполнять crossbar-коммутатор. Коммутатор он, собственно, потому что к нему подключаются разные системные платы.



Архитектура NUMA