Лекция 1

8 февраля

Цель курса «Архитектура ЭВМ и язык ассемблера»

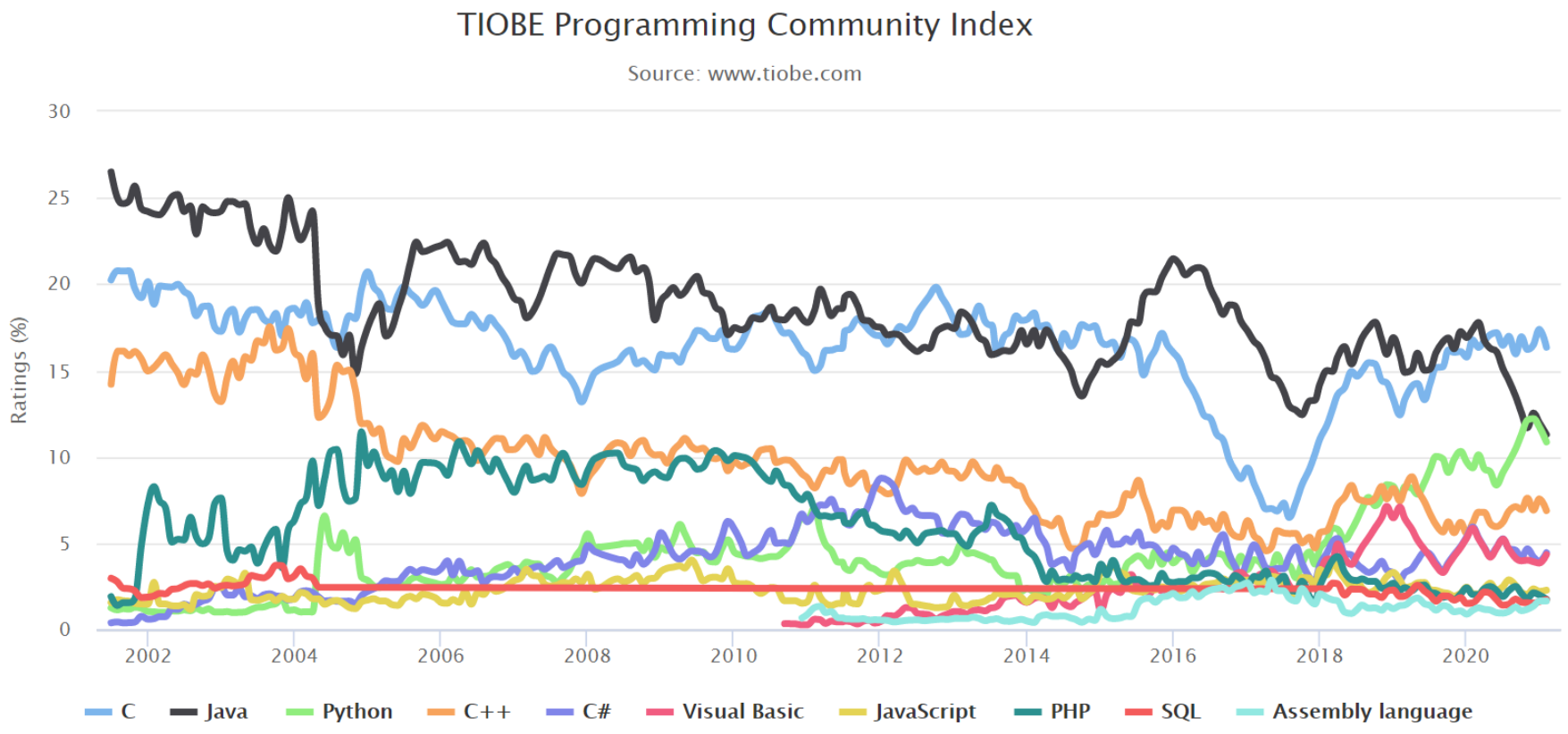
1. Формирование связного представления об организации современных вычислительных систем.
   - Аппаратура, системные программы, прикладные программы
2. Понимание взаимосвязей между архитектурными решениями на уровнях аппаратуры ЭВМ и ПО.
3. Понимание факторов, влияющих на качественные и количественные характеристики ЭВМ, производительность и безопасность всей вычислительной системы в целом.

- Язык Си как средство демонстрации …
  - того, как работает аппаратура …
  - и того, как следует использовать аппаратуру для реализации языка программирования
- Результат
  - Понимание причин хорошей/плохой производительности программы
  - Понимание причин ошибок
  - Основа для других курсов: ОС, сети, компиляторы, параллельная обработка данных …

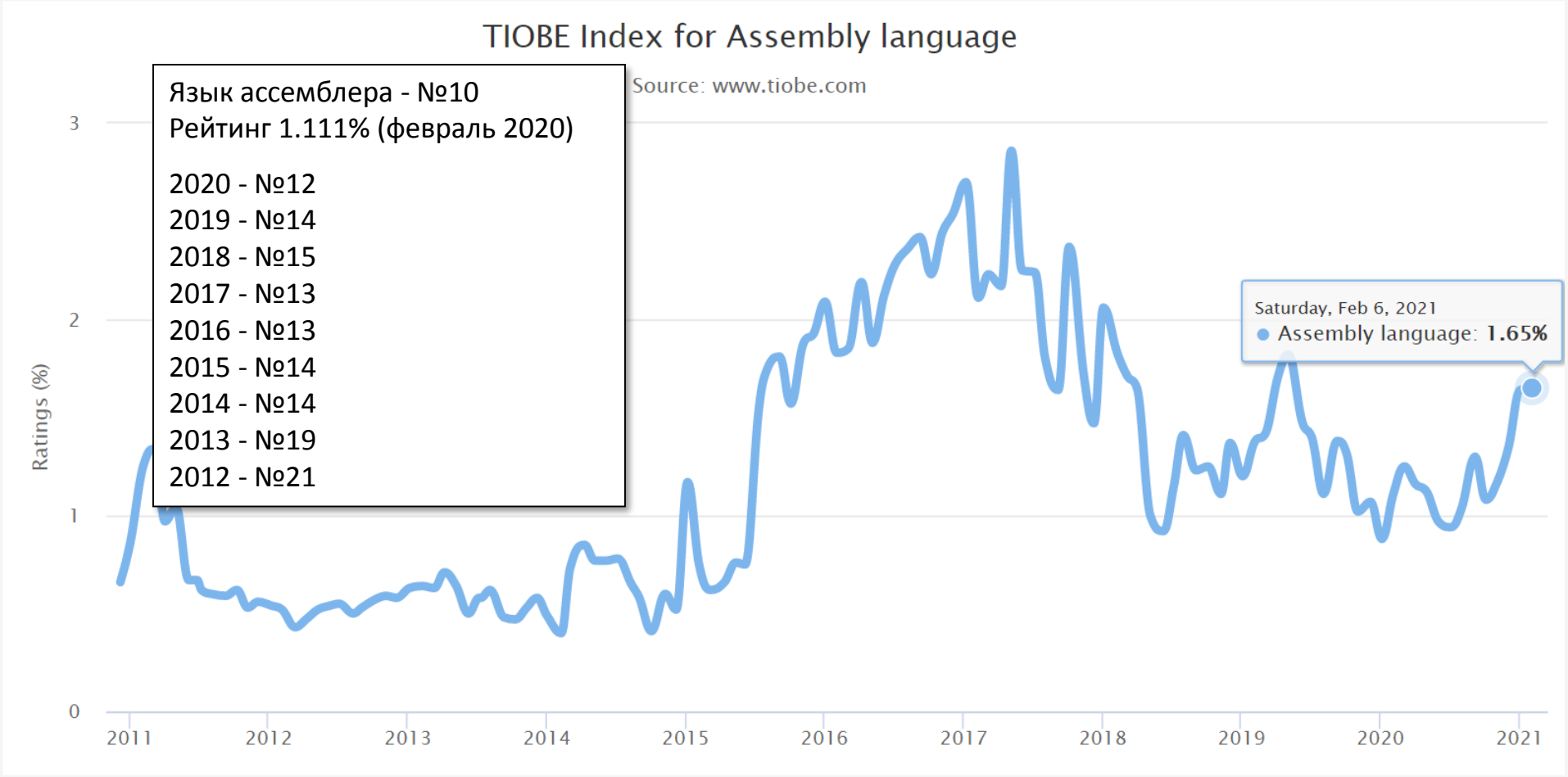
Проблема №1 Свойства чисел

- $X^2 \geq 0$?
  - float – всегда да
  - int ?
    - 40000 * 40000 = 1600000000
    - 50000 * 50000 = ?
- (x + y) + z = x + (y + z) ?
  - signed/unsigned int – всегда да
  - float
    - (1e20 + -1e20) + 3.14 → 3.14
    - 1e20 + (-1e20 + 3.14) → ??
- Свойства типов данных могут быть использованы для причинения вреда

Проблема №2 Язык ассемблера



Проблема №2 Язык ассемблера



Язык ассемблера - №10
Рейтинг 1.111% (февраль 2020)

2020 - №12
2019 - №14
2018 - №15
2017 - №13
2016 - №13
2015 - №14
2014 - №14
2013 - №19
2012 - №21

Компилятор Си составит программу ничуть не хуже

Проблема №3 Отладка

- Ошибки работы с памятью
  - Выход за пределы массива
  - Некорректные указатели
  - malloc/free
- Неопределенное поведение

```
double fun(int i) {
  volatile double d[1] = {3.14};
  volatile long int a[2];
  a[i] = 1073741824;
  return d[0];
}
```

```
fun(0)  →  3.14
fun(1)  →  3.14
fun(2)  →  3.1399998664856
fun(3)  →  2.0000061035156
fun(4)  →  3.14, а потом segmentation fault
```

Проблема №4 Производительность
Асимптотическая сложность

```
void copyij(int src[2048][2048], int dst[2048][2048]) {
  int i,j;
  for (i = 0; i < 2048; i++)
    for (j = 0; j < 2048; j++)
      dst[i][j] = src[i][j];
}
```

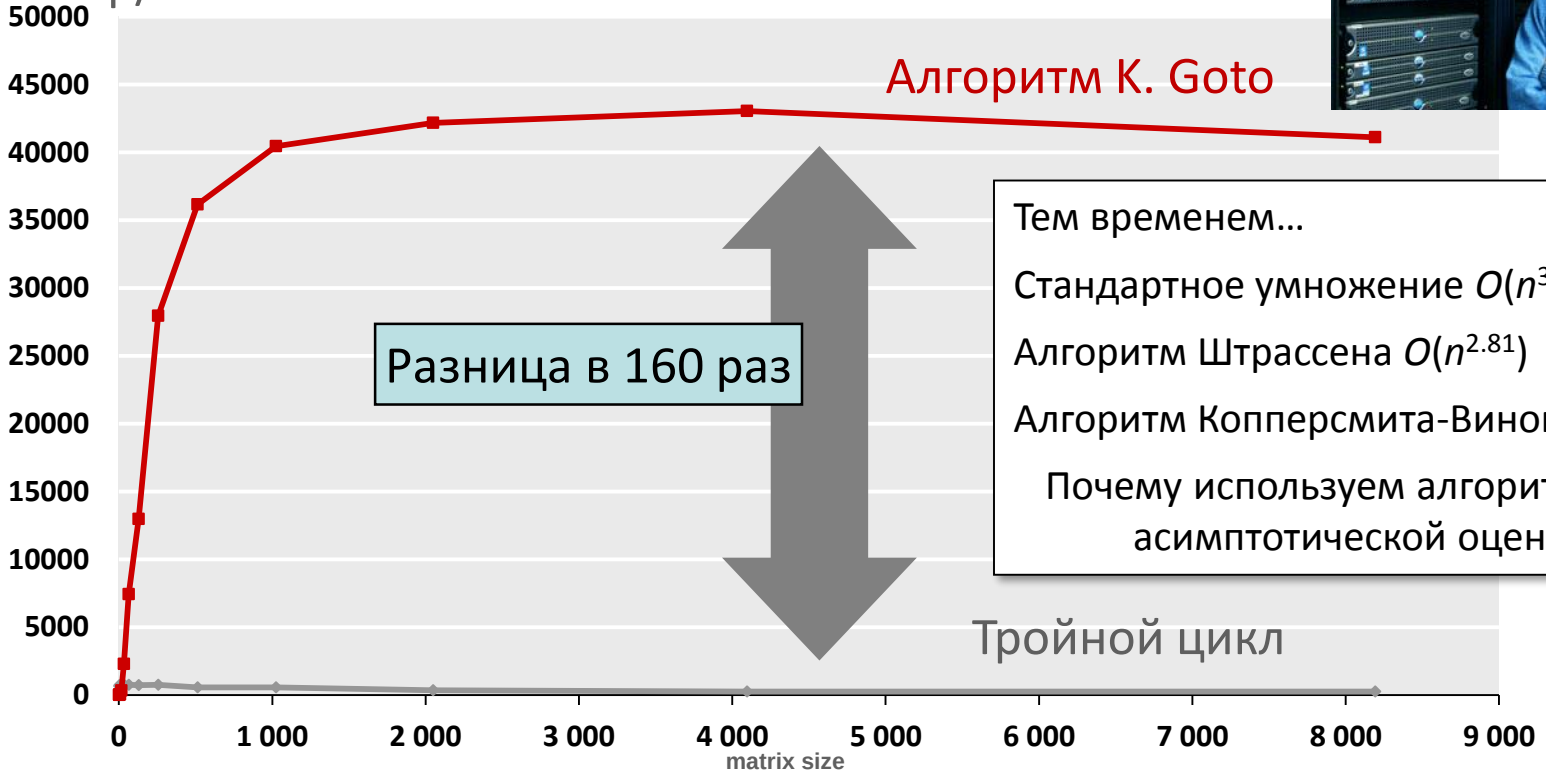
- Память имеет многоуровневую, иерархическую структуру
- Скорость работы программы зависит от порядка выполняемых действий

```
void copyji(int src[2048][2048], int dst[2048][2048]) {
  int i,j;
  for (j = 0; j < 2048; j++)
    for (i = 0; i < 2048; i++)
      dst[i][j] = src[i][j];
}
```

Медленней ~20 раз

Проблема №4 Производительность
Асимптотическая сложность

Умножение матриц 2 x Core 2 Duo 3 GHz (double)
Библиотека GotoBLAS



Тем временем…

Стандартное умножение $O(n^3)$

Алгоритм Штрассена $O(n^{2.81})$

Алгоритм Копперсмита-Винограда $O(n^{2.38})$

Почему используем алгоритм с худшей асимптотической оценкой???

В точности одинаковая сложность ($2n^3$)

Kazushige Goto and Robert A. van de Geijn. 2008. Anatomy of high-performance matrix multiplication. ACM Trans. Math. Softw. 34, 3, Article 12 (May 2008)

Проблема №5 Безопасность

- Оптимизации компилятора способны нарушать безопасность кода

```
memset(password, '\0', len);
free(password);
```

-O3 ⇒

```
free(password);
```

- Язык высокого уровня не специфицирует существенные для безопасности аспекты

```
int a[2], b;
memset(a, 0, 12);
```

- Соответствие исходного и исполняемого кода
  - Необходим доступ к системе сборки и конфигурационным файлам
- Исходный код может быть просто недоступен

G. Balakrishnan, T. Reps, D. Melski, and T. Teitelbaum. WYSINWYX: What You See Is Not What You eXecute. In Verified Software: Theories, Tools, Experiments, Bertrand Meyer and Jim Woodcock (Eds.). 2005. Lecture Notes In Computer Science, Vol. 4171. Springer-Verlag, Berlin, Heidelberg 202-213. DOI=http://dx.doi.org/10.1007/978-3-540-69149-5_22

Зачем сейчас нужен язык ассемблера?

- Язык ассемблера позволяет понимать поведение машины

- Системное программное обеспечение создается людьми, в том числе выпускниками ВМК
  - Компиляторы и многие другие инструменты разработки программ
  - Операционные системы
  - Средства защиты информации
- Отладка ошибок разработчиком программы
- Настройка производительности программы
  - Почему оптимизация программы компилятором не дает ожидаемого результата
- Безопасность программного обеспечения (software security, cyber security)
  - Сертификация программ
  - Средства защиты
  - Вредоносный код: ботнеты, вирусы/черви, руткиты, …

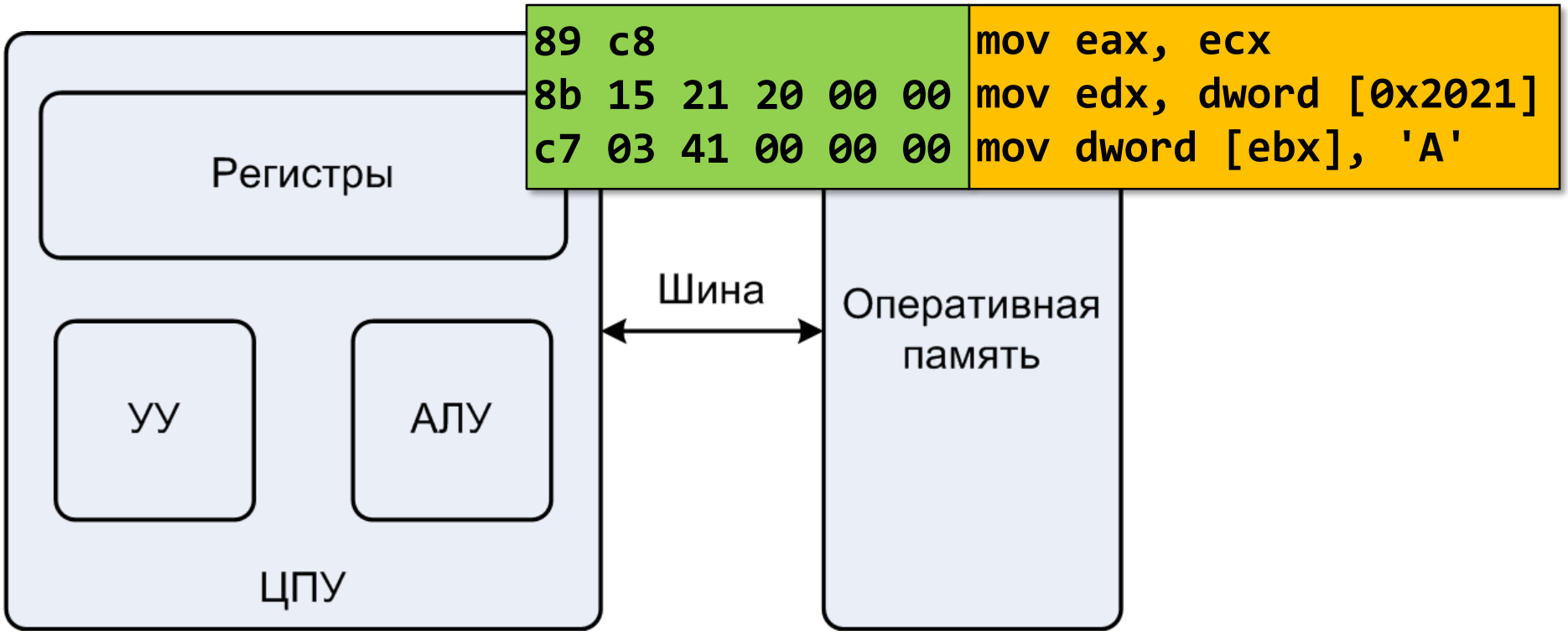
История первых компьютеров

- Чарлз Бэббидж, Ада Лавлайс 1820 – 1833
- Конрад фон Цузе
  - Z1 1938г., Z2 1939 г., Z3 1941г., Z4 1945г.
- Экерт и Моучли
  - ENIAC 1946 г.
  - фон Нейман
    - First Draft of a Report on the EDVAC, 30 июня 1945
- Атанасов и Берри
  - ABC 1942 г.
- Mark I
  - 1949 г.
- Исаак Брук
  - М-1 1951 г.
- Сергей Лебедев
  - МЭСМ 1950 г.
  - БЭСМ-1 1952 г.

Применение первых компьютеров
- задачи математической физики
- шифрование/дешифрование сообщений
- управление (военной) техникой

Современное применение компьютеров
- хранение информации и предоставление удобного доступа для работы с ней
- игры!!!
- передача информации
- управление технологическими процессами
- работа во всем, что нас окружает …

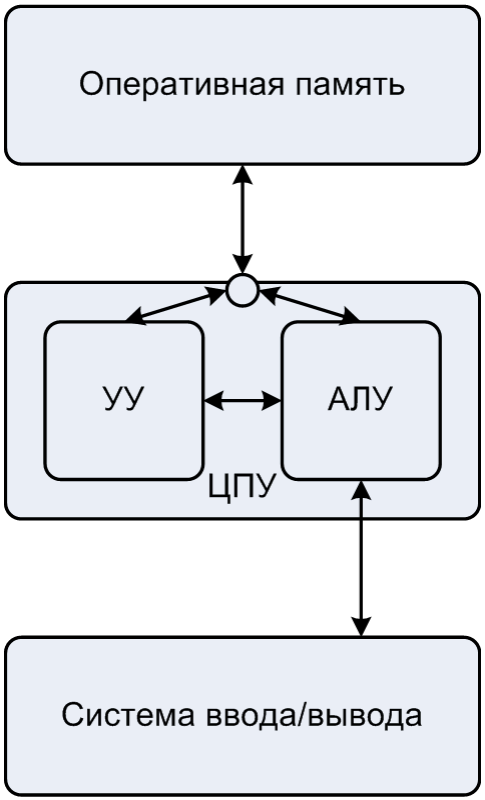
Архитектура фон Неймана



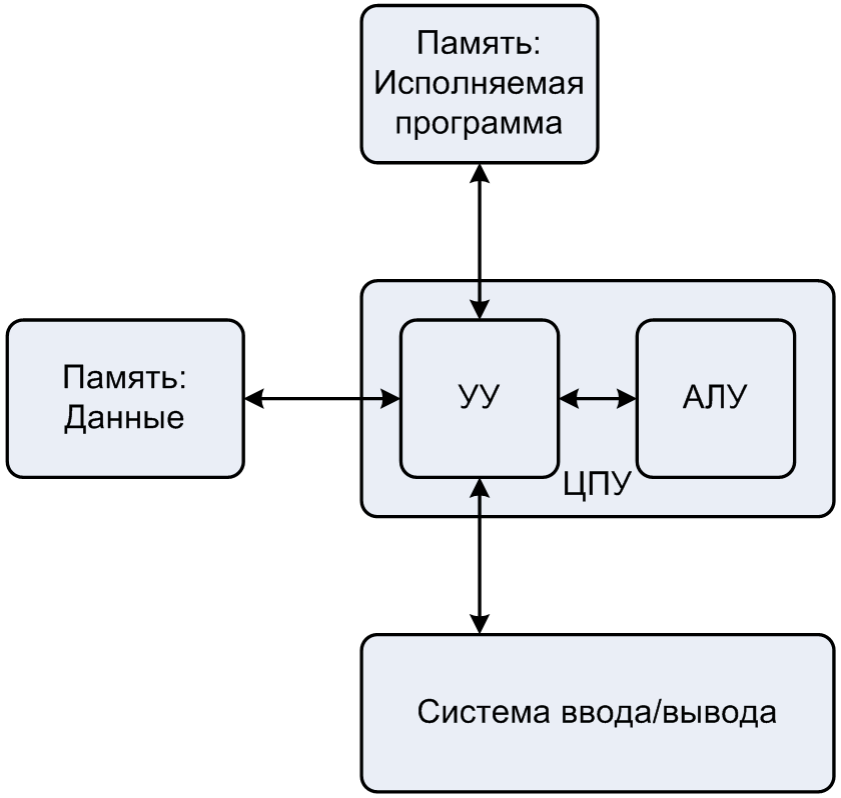
1. Двоичное кодирование информации
2. Неразличимость команд и данных
3. Адресуемость памяти
4. Последовательное выполнение команд

Вспоминаем Си-машину…

Единственный вариант архитектуры?



Еще один вариант изображения архитектуры фон Неймана…



Гарвардская архитектура

Вспоминаем Си-машину…

(Модельный) цикл работы ЭВМ

1. **Извлечение инструкции из памяти**
   Используя текущее значение *счетчика команд,* процессор извлекает некоторое количество байт из памяти и помещает их в *буфер команд*
2. **Декодирование команды**
   Процессор просматривает содержимое *буфера команд* и определяет *код операции* и ее *операнды*
   Длина декодированной команды прибавляется к текущему значению *счетчика команд*
3. **Загрузка операндов**
   Извлекаются значения операндов
   Если операнд размещен в ячейках памяти – вычисляется *исполнительный адрес*
4. **Выполнение операции над данными**
5. **Запись результата**
   Результат может быть записан в том числе и в *счетчик команд* для изменения естественного порядка выполнения

Промежуточные итоги: ключевые термины и понятия

- Регистр, имя регистра, значение регистра, разрядность регистра, машинное слово
- Память, ячейка памяти, адрес ячейки памяти, время доступа к памяти
- Цикл, тактовая частота, счетчик команд
- Машинная команда, код операции, операнд, адресность команды
  Исполнительный (действительный) адрес
  англ. *effective (executive) address*
- Способ адресации
  прямая, непосредственная, косвенная