

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 ...

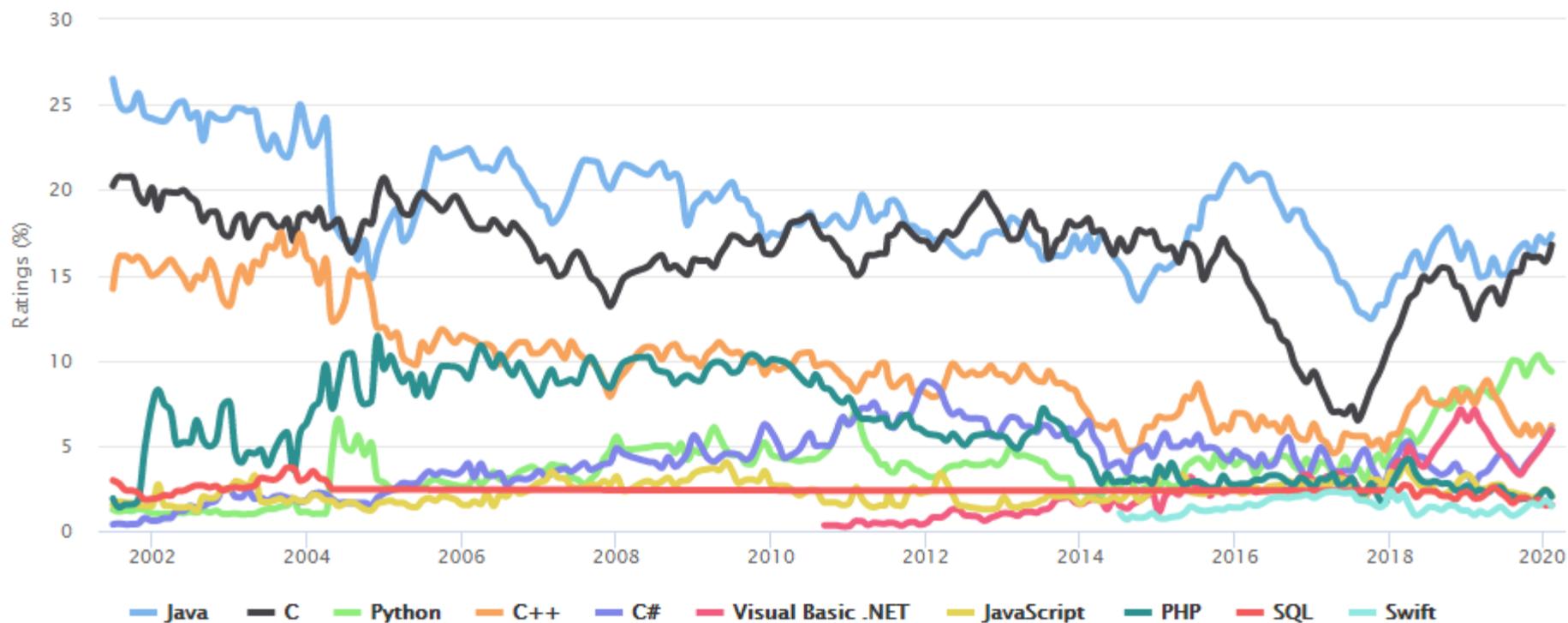
Проблема №1 Свойства чисел

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

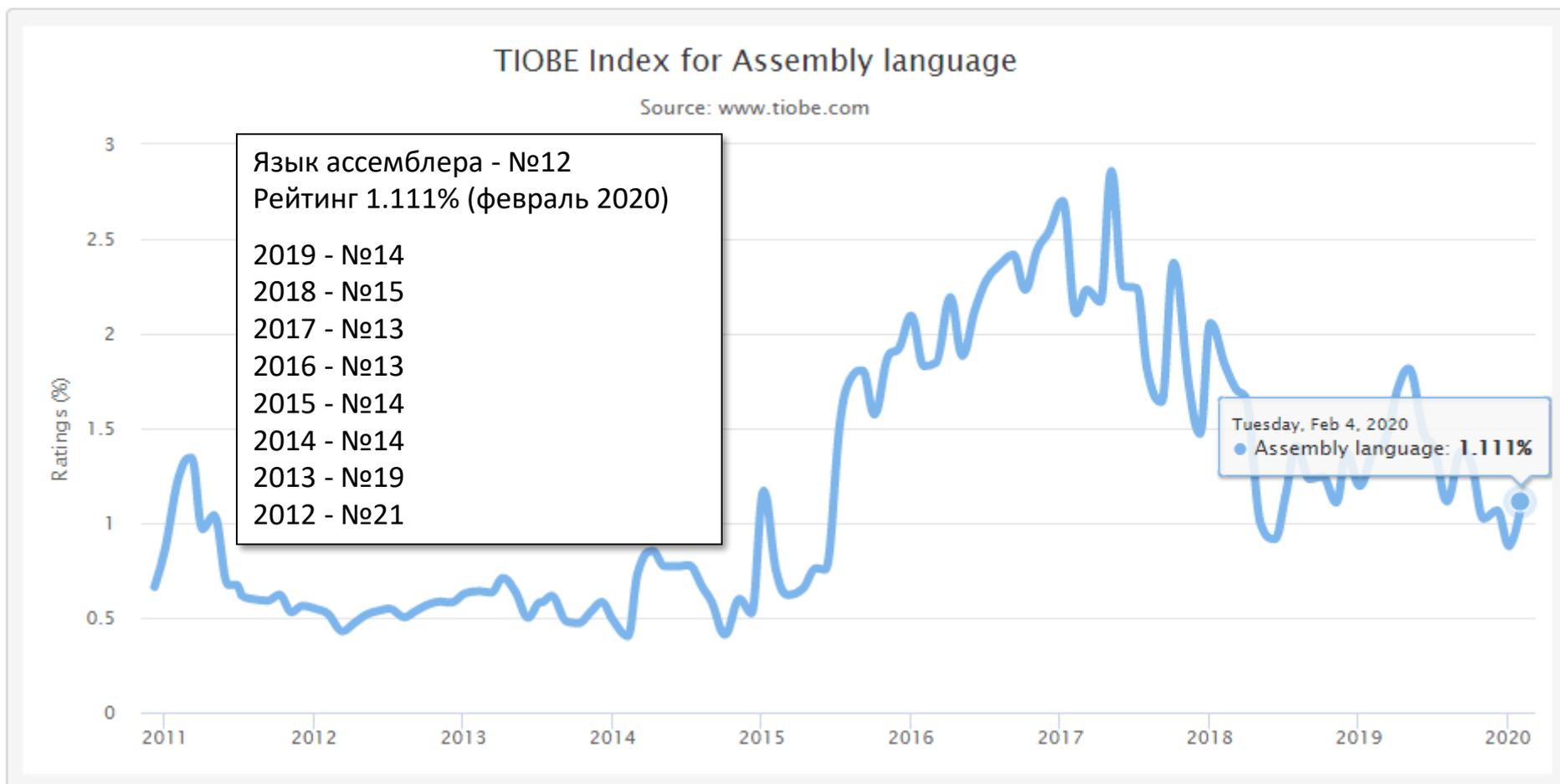
Проблема №2 Язык ассемблера

TIOBE Programming Community Index

Source: www.tiobe.com



Проблема №2 Язык ассемблера



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) → 2.00000061035156
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

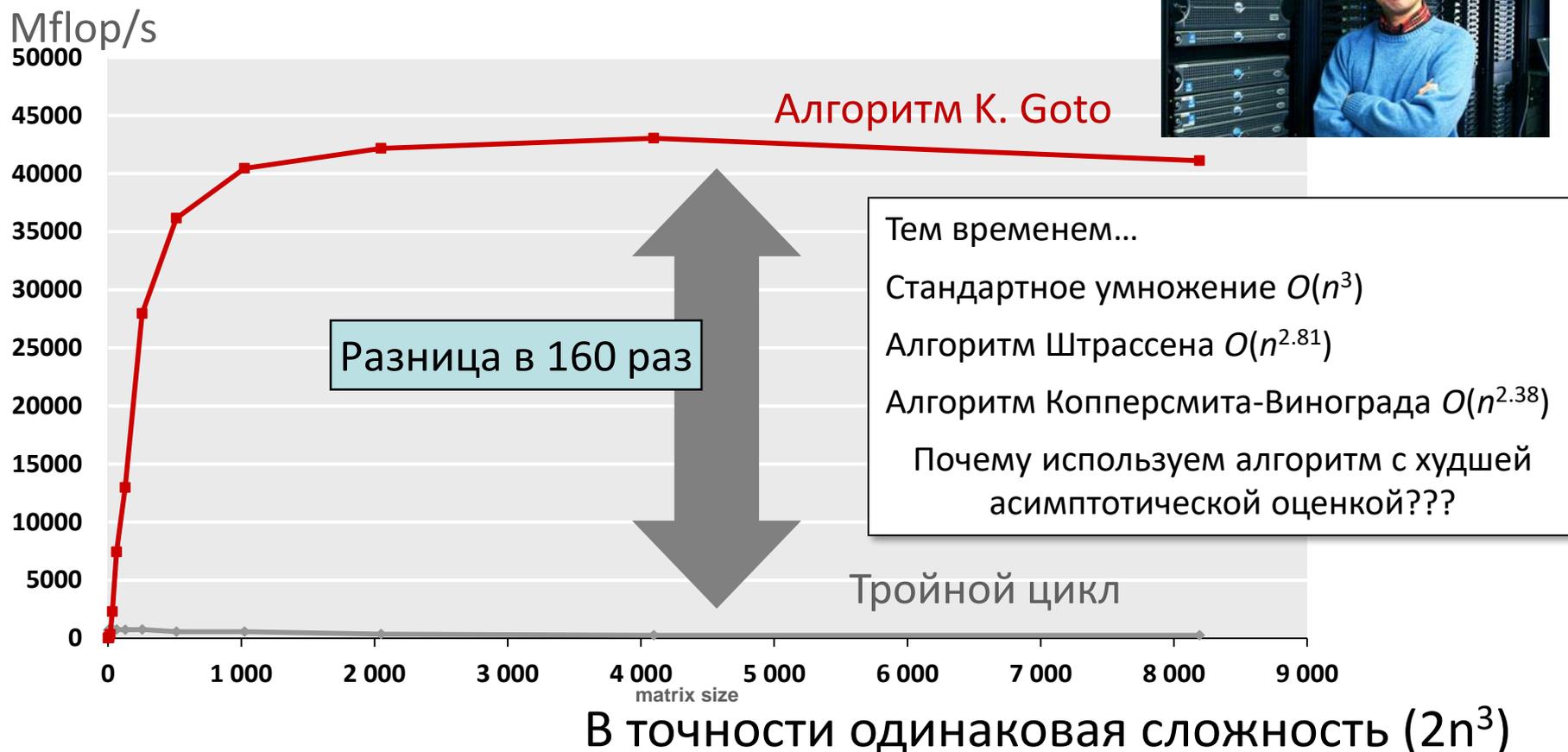
```
void copyji(int src[2048][2048], int dst[2048][2048]) {  
    int i,j;  
    for (j = 0; j < 2048; j++)  
        for (i = 0; i < 2048; i++)  
            dst[i][j] = src[i][j];  
}
```

Медленней ~20 раз

Проблема №4 Производительность

Асимптотическая сложность

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



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);           -03
free(password);                         => free(password);
```

- Язык высокого уровня не специфицирует существенные для безопасности аспекты

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

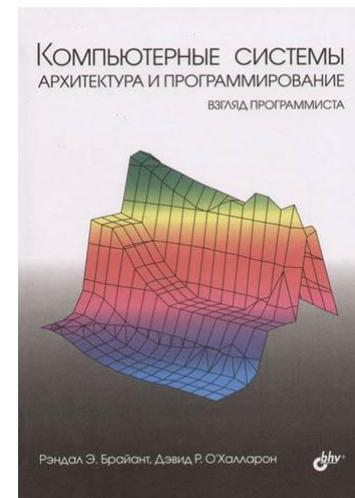
- Соответствие исходного и исполняемого кода
 - Необходим доступ к системе сборки и конфигурационным файлам
- Исходный код может быть просто недоступен

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, ...

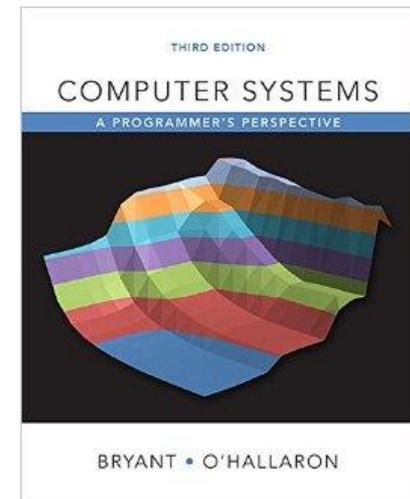
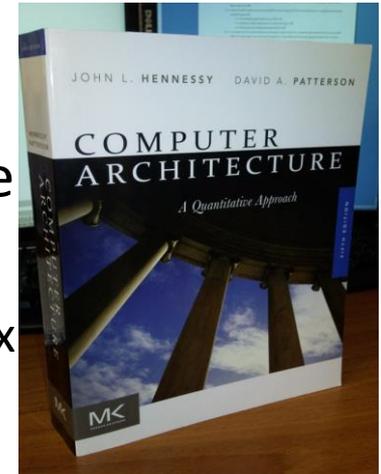
Литература

- Рэндал Э. Брайант, Дэвид О'Халларон. Компьютерные системы: архитектура и программирование (Computer Systems: A Programmer's Perspective). Издательство: БХВ-Петербург, 2005 г. — 1186 стр.
- Э. Таненбаум, Т. Остин. Архитектура компьютера. 6-е изд. Издательство: Питер, 2016. — 816 стр.
- Е.А. Кузьменкова, В.С. Махнычев, В.А. Падарян. Семинары по курсу «Архитектура ЭВМ и язык ассемблера»: учебно-методическое пособие. Часть 1. Издательство: МАКС ПРЕСС, 2014, 78 стр.
- Е.А. Кузьменкова, В.А. Падарян, М.А. Соловьев. Семинары по курсу «Архитектура ЭВМ и язык ассемблера»: учебно-методическое пособие. Часть 2. Издательство: МАКС ПРЕСС, 2014, 100 стр.



Дополнительная (англоязычная) литература

- John L. Hennessy and David A. Patterson. Computer Architecture, 5th Edition: A Quantitative Approach. Morgan Kaufmann Publishers 2012.
 - Часть тем курса рассматривается в первых двух главах (~130 страниц)
- Randal E. Bryant, David R. O'Hallaron. Computer Systems: A Programmer's Perspective (3rd Edition). Pearson 2015. – 1120 p.



Из чего состоит курс

- Лекции
 - 48 часов / 24 лекции
 - Два письменных коллоквиума **c1** и **c2**
 - Письменный экзамен **ex**
- Семинары
 - 72 часа / 36 занятий
 - Аудитория
 - Машинный зал
 - Домашние задания
 - всего – 6 заданий
 - 4 задания сдаются в системе Ejudge
 - Зачет с оценкой
 - Комиссия: машинный зал
- Сайт поддержки
<http://asmcourse.cs.msu.ru/>

Итоговая оценка

$$0.2 * c1 + 0.2 * c2 + 0.6 * ex$$

- удовлетворительно $\geq 40\%$
- хорошо $\geq 60\%$
- отлично $\geq 80\%$

Пропуск коллоквиума – штраф -5%.

Ограничение на количество посылок задачи:

- простые задачи – 16,
- обычные задачи – 24,
- повышенной сложности – 32.

На сайте поддержки курса в разделе «Экзамены и контрольные» детально описаны правила выставления оценок за лекционный курс и практикум

http://asmcourse.cs.msu.ru/?page_id=1020

План лекций

- Введение на примерах
 - 5 лекций
- Архитектура набора команд, производительность и безопасность программ
 - 10 лекций
- Система программирования (языка Си)
 - 3 лекции
- Архитектура ЭВМ
 - 5 лекций

Ориентировочное время проведения коллоквиумов

1. 11 или 18 марта, зависит от переноса праздничных дней
2. 15 или 22 апреля

История первых компьютеров

- Чарлз Бэббидж, Ада Лавлейс 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
 - M-1 1951 г.
- Сергей Лебедев
 - МЭСМ 1950 г.
 - БЭСМ-1 1952 г.

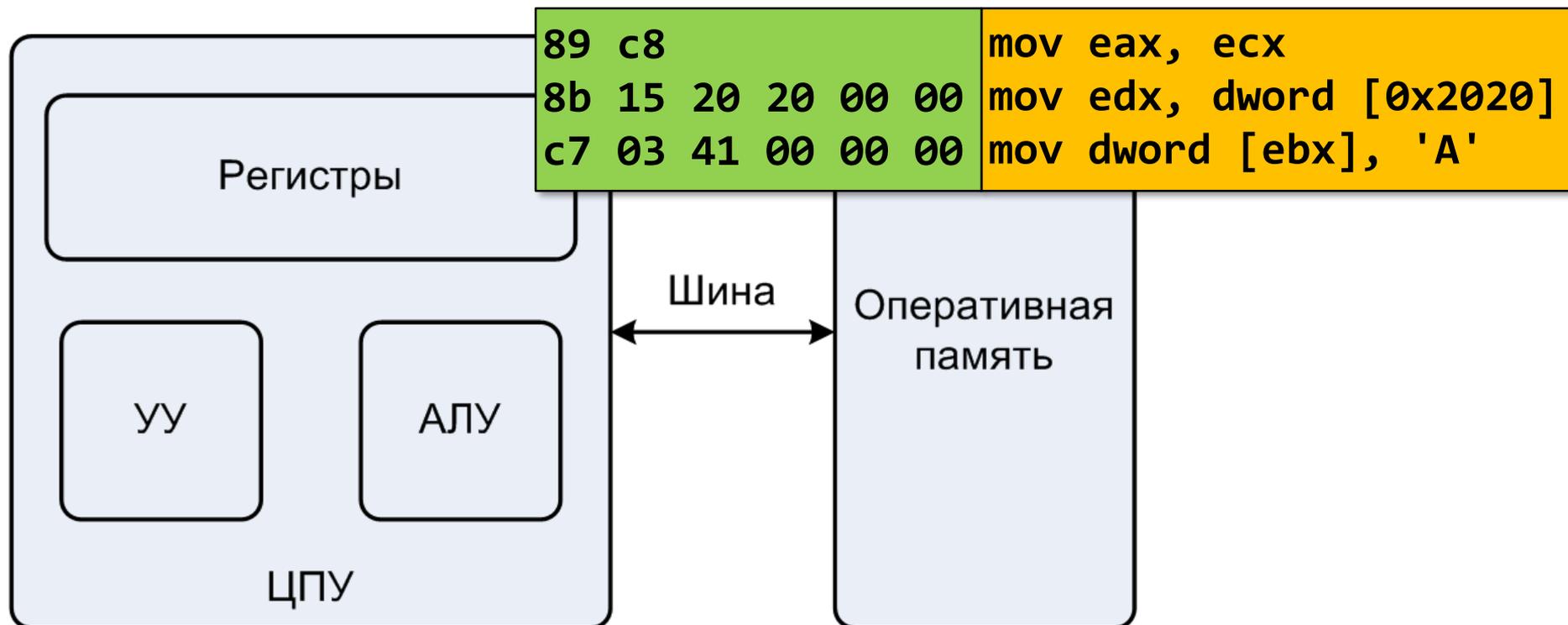
Применение первых компьютеров

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

Современное применение компьютеров

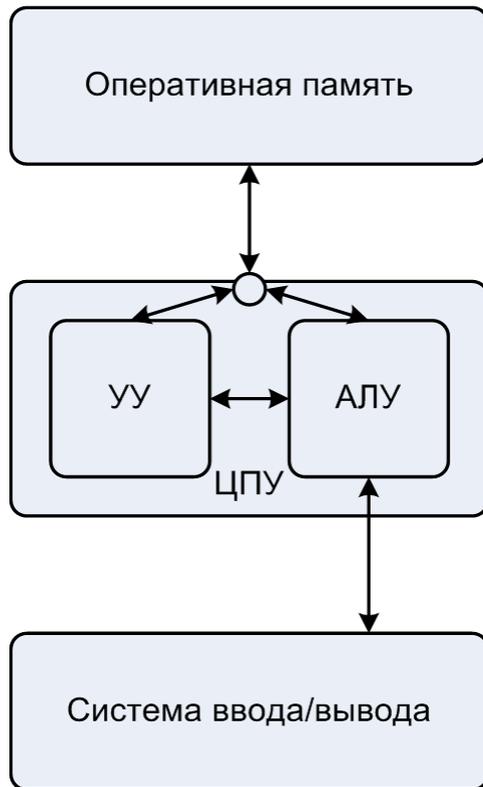
- хранение информации и предоставление удобного доступа для работы с ней
- игры!!!
- передача информации
- управление технологическими процессами
- работа во всем, что нас окружает ...

Архитектура фон Неймана

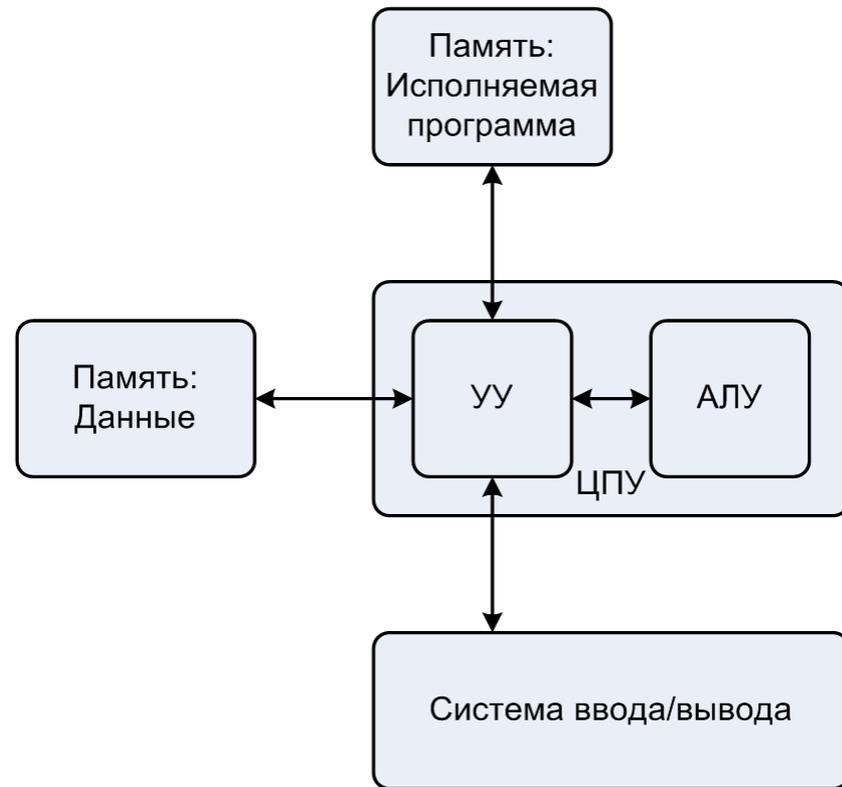


1. Двоичное кодирование информации
2. Неразличимость команд и данных
3. Адресуемость памяти
4. Последовательное выполнение команд

Единственный вариант архитектуры?



Еще один вариант
изображения архитектуры
фон Неймана...



Гарвардская архитектура

(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