Введение в программирование

Гусев Илья, Булгаков Илья

Московский физико-технический институт

Москва, 2018

Содержание

Структура курса

1. Алгоритмы:
   - 3 семестра
   - Экзамены в 1 и 3
   - 1 ак. час лекций в неделю
2. C++:
   - 2 семестра
   - Экзамен во 2
   - 1 ак. час лекций в неделю
3. Практика
   - 3 семестра,
   - Во всех зачёты
   - 2 ак. часа семинаров в неделю

Итого: 3 экзамена, 3 зачёта, 4 ак. часа в неделю на протяжении 3 семестров

Правила зачёта

Оценки

- 1 модуль - 25 балла
- 2 модуль - 25 балла
- 3 модуль - 25 балла
- 4 модуль - 25 балла
- 20 баллов за задачи и 5 баллов за ответ на контрольной
- Сроки сдачи ограничены для каждого модуля
- Критерий приёма задачи: Яндекс.Контест + review кода
- Review кода = style + правильность алгоритма + функциональная декомпозиция

Правила зачёта

Оценки

- 0-39 баллов - 1 (неуд)
- 40-49 баллов - 2 (неуд)
- 50-56 баллов - 3 (удовл)
- 57-65 баллов - 4 (удовл)
- 66-71 баллов - 5 (хор)
- 72-77 балла - 6 (хор)
- 78-83 баллов - 7 (хор)
- 84-88 баллов - 8 (отл)
- 89-93 балла - 9 (отл)
- 94-100 баллов - 10 (отл)

Правила зачёта

Посещаемость, списывание, кодстайл

- Про посещаемость: она напрямую не влияет на оценку
- Про списывание: любое дублирование чужого кода штрафуется -5 и незачётом по задаче

Правила зачёта

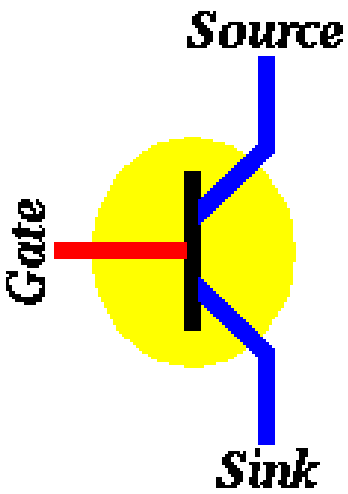
Кодстайл

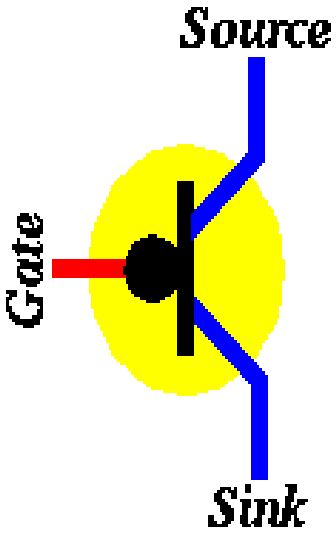
- Понятные названия переменных (однобуквенные только как счётчики в циклах или заданные в условии задачи)
- Не должно быть утечек памяти
- Грамтоная декомпозиция на функции. В main должен находиться ввод данных, вызов функции, которая решает задачу, вывод результата. Ввод/вывод производится только в main, решение - только в отдельном наборе функций
- Переменные должны объявляться по месту использования (уже давно >= C99)
- Глобальными переменными пользоваться нельзя
- Остальное: https://goo.gl/Ztu6td

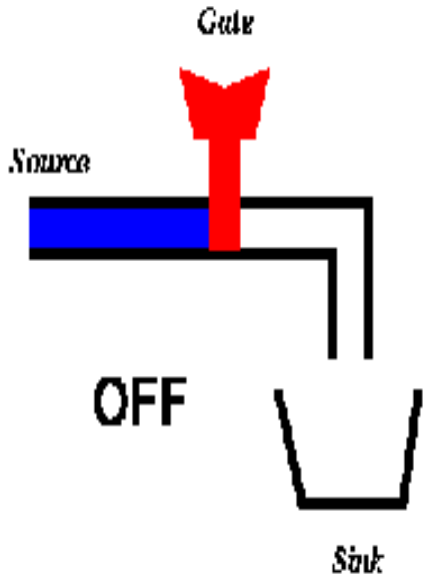
Транзисторы → Логические элементы

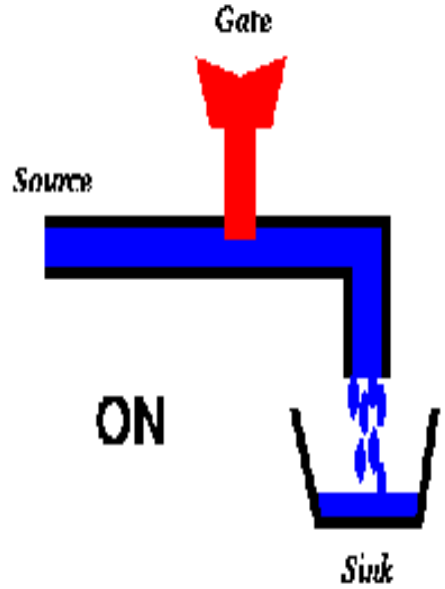
- http://www.cs.bu.edu/~best/courses/modules/Transistors2Gates/
- Reddit topic: https://bit.ly/2CdBk3u
- Реализация «Тетриса» в игре «Жизнь»: https://habr.com/post/338584/

Транзисторы → Логические элементы



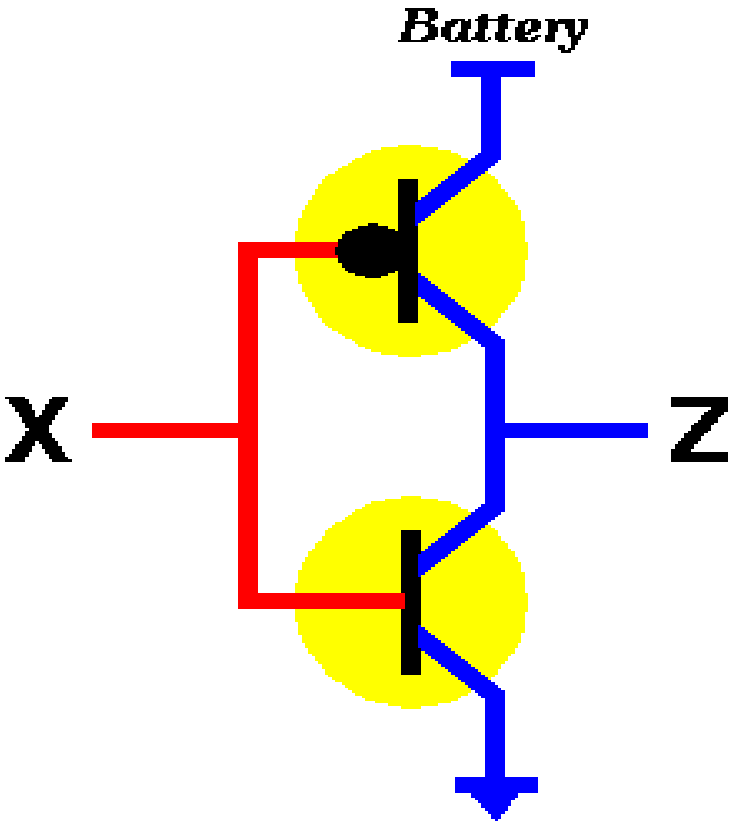






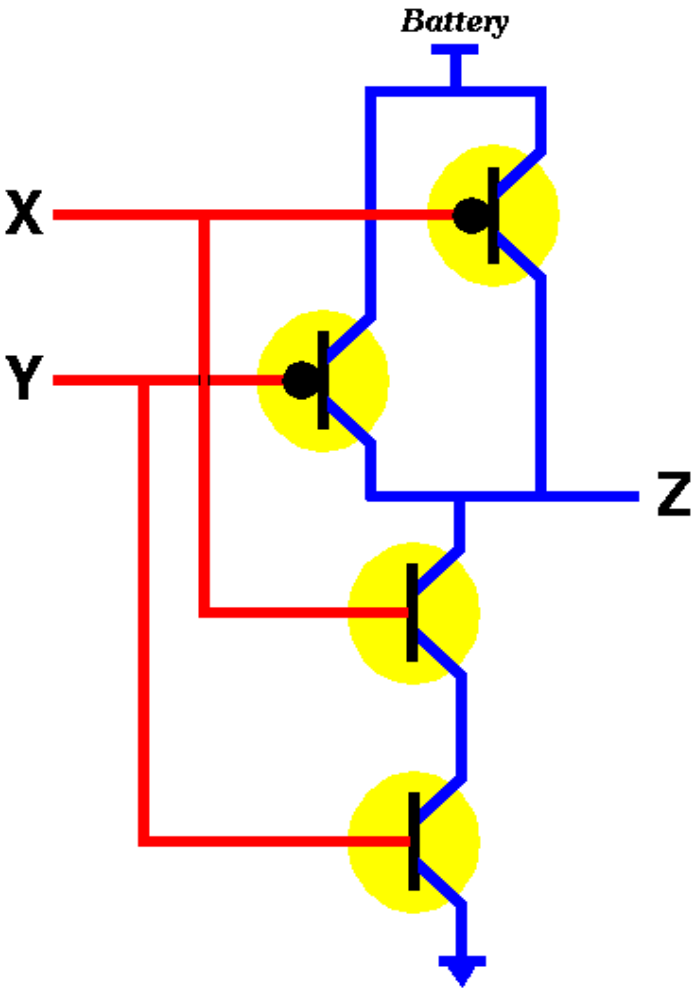
Транзисторы → Логические элементы

NOT



Транзисторы → Логические элементы

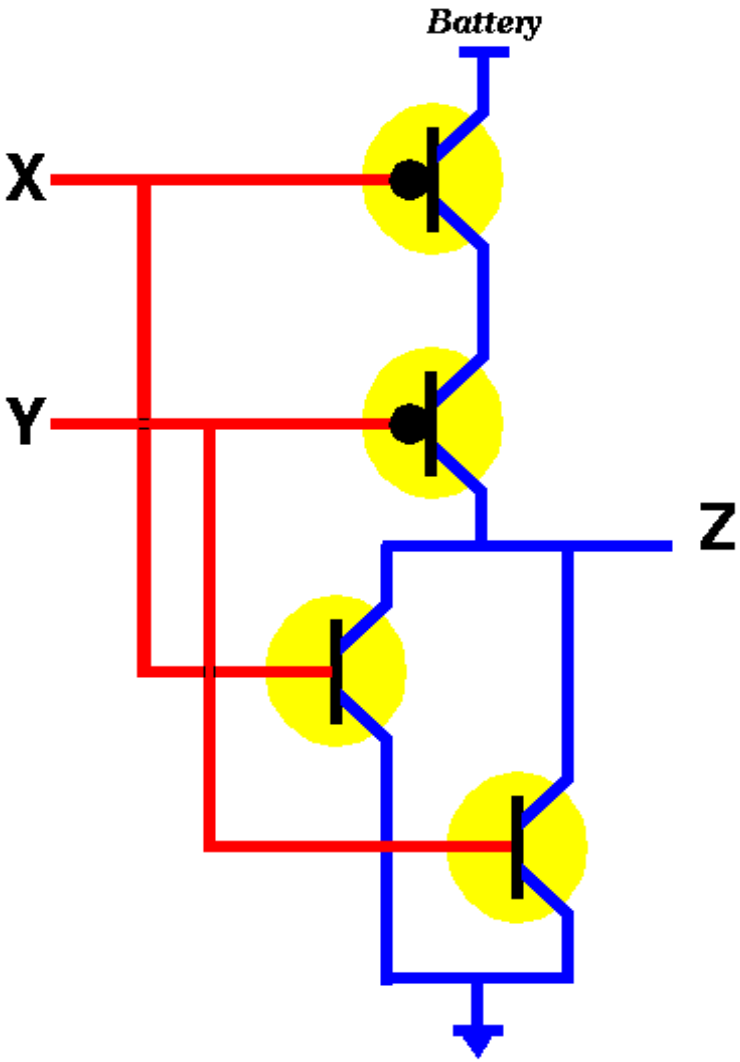
NAND



X	Y	Z
0	0	1
0	1	1
1	0	1
1	1	0

Транзисторы → Логические элементы

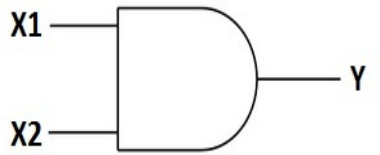
NOR



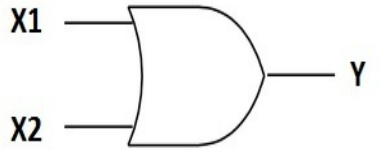
X	Y	Z
0	0	1
0	1	0
1	0	0
1	1	0

Логические элементы → Сумматор

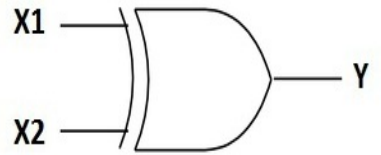
AND, OR, XOR



Вход X1	Вход X2	Выход Y
0	0	0
1	0	0
0	1	0
1	1	1



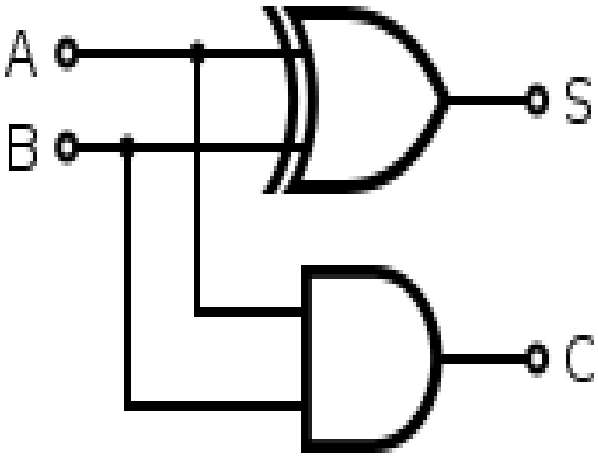
Вход X1	Вход X2	Выход Y
0	0	0
1	0	1
0	1	1
1	1	1

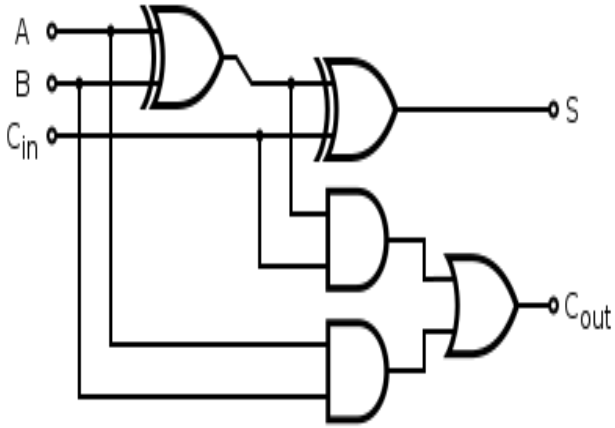


Вход X1	Вход X2	Выход Y
0	0	0
1	0	1
0	1	1
1	1	0

Логические элементы → Сумматор

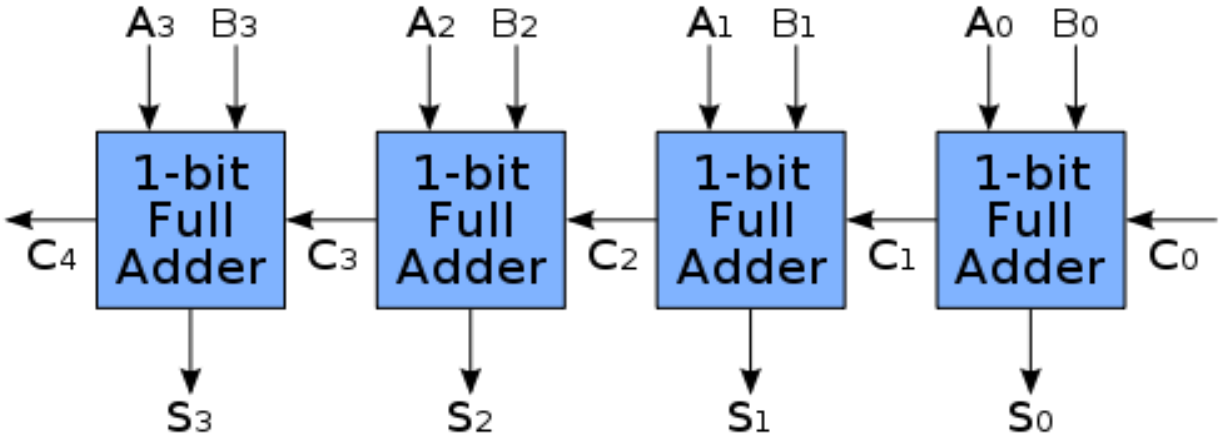
Half and full adder



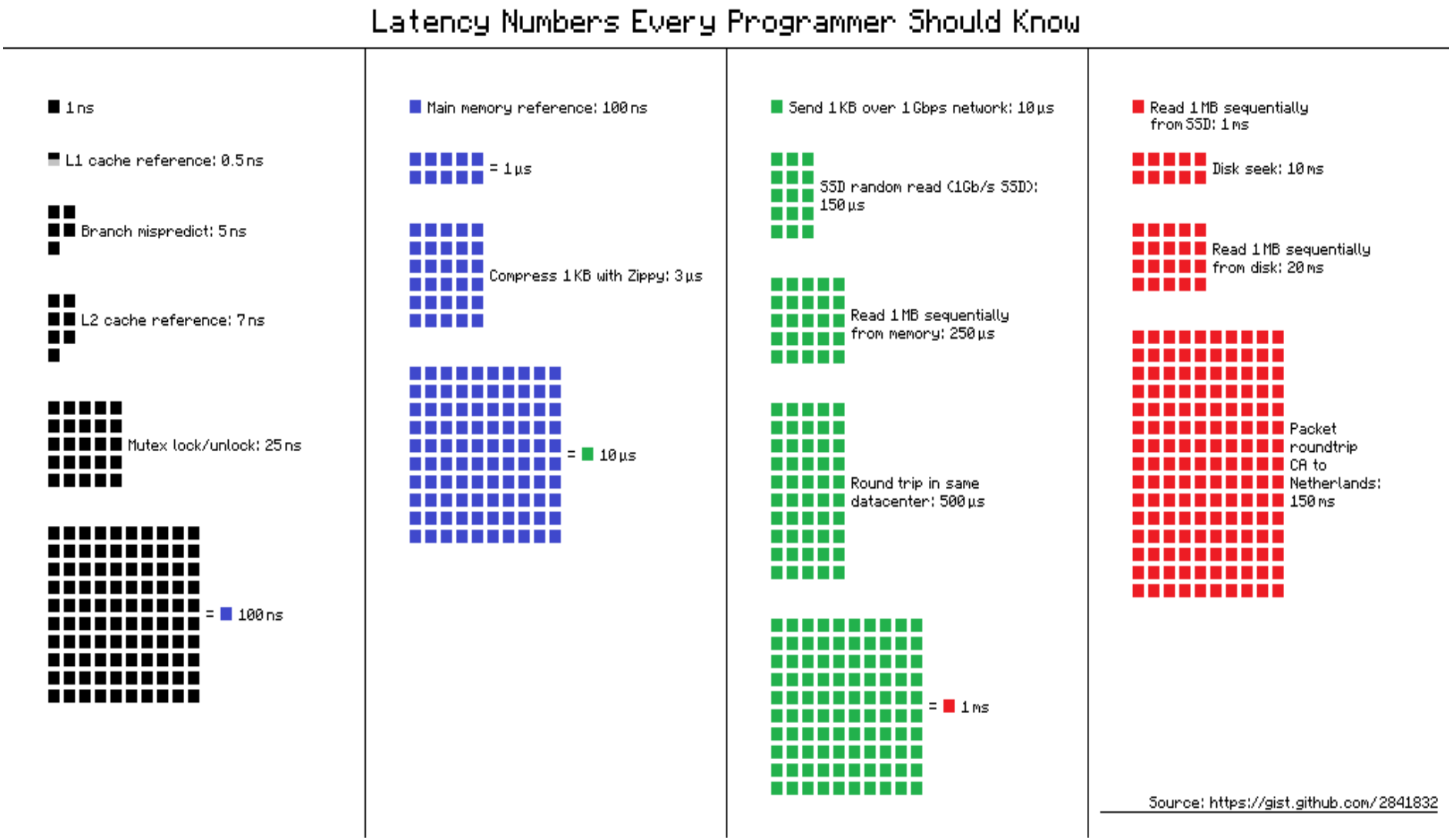


Логические элементы → Сумматор

Линейный каскадный сумматор



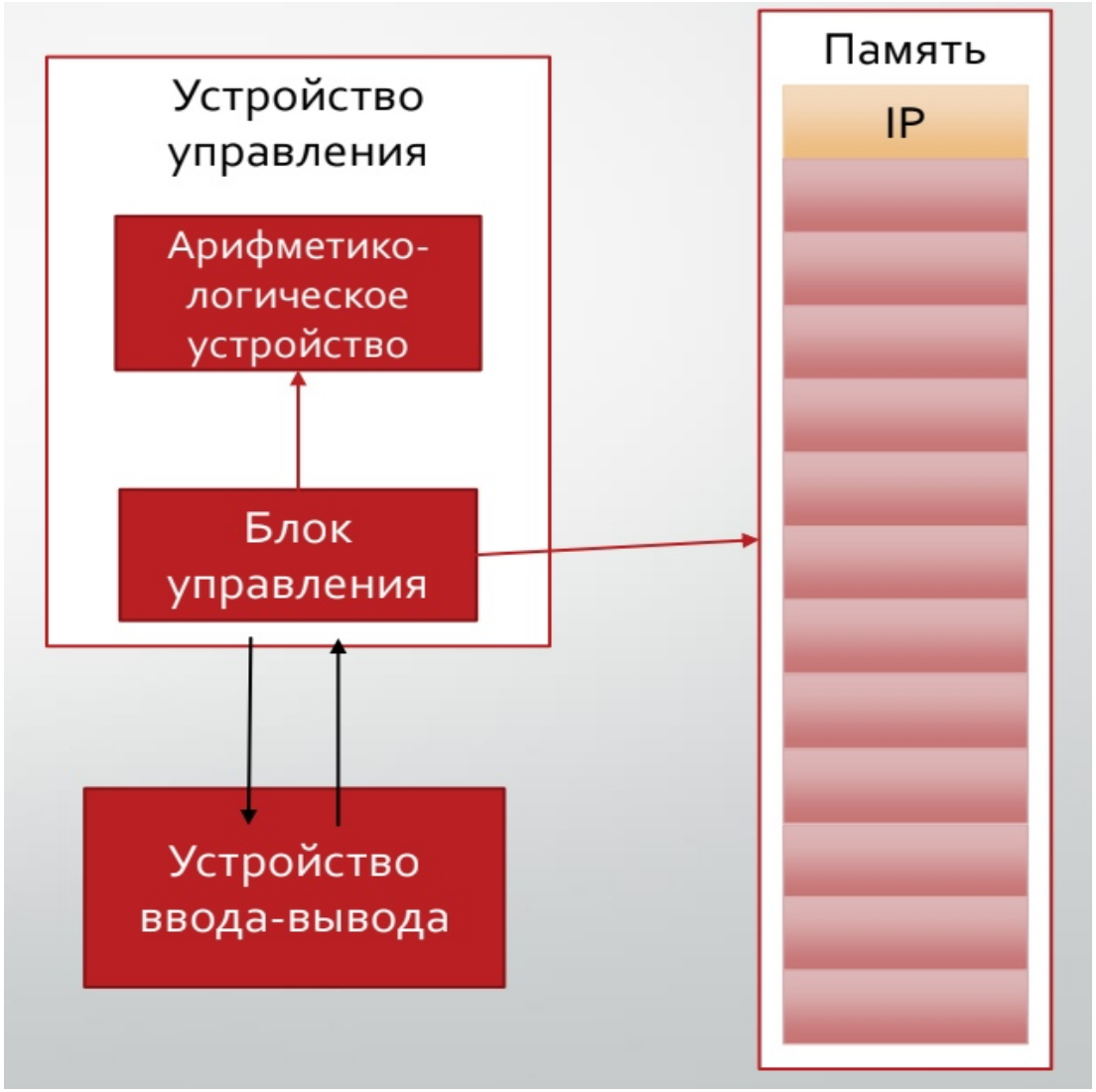
Память


Latency Numbers Every Programmer Should Know
1 ns
L1 cache reference: 0.5 ns
Branch mispredict: 5 ns
L2 cache reference: 7 ns
Mutex lock/unlock: 25 ns
= 100 ns
Main memory reference: 100 ns
= 1 µs
Compress 1 KB with Zippy: 3 µs
= 10 µs
Send 1 KB over 1 Gbps network: 10 µs
SSD random read (1Gb/s SSD): 150 µs
Read 1 MB sequentially from memory: 250 µs
Round trip in same datacenter: 500 µs
= 1 ms
Read 1 MB sequentially from SSD: 1 ms
Disk seek: 10 ms
Read 1 MB sequentially from disk: 20 ms
Packet roundtrip CA to Netherlands: 150 ms
Source: https://gist.github.com/2841832

Архитектура фон Неймана

- Работа компьютера контролируется программой, состоящей из набора команд
- Принцип последовательного выполнения: после выполнения текущей команды IP (instruction pointer) автоматически указывает на следующую
- Принцип однородности: в памяти хранятся и данные, и команды
- Принцип адресуемости: память состоит из ячеек, каждая имеет уникальный «адрес»
- Использование двоичной системы счисления * (https://habr.com/post/166679/)

Архитектура фон Неймана



Архитектура фон Неймана

Тип данных	Семантика
Address	Адрес ячеек памяти
Value	Значение, лежащее в памяти
state ∈ State : Address -> Value	Функция текущего состояния
operator ∈ Command: State -> State	Функция изменения состояния
decode : Value -> Command	Расшифровка команды. Выполняет блок управления.
Command	Множество всех инструкций, которые умеет выполнять АЛУ
stop ∈ Command	Инструкция остановки
ip ∈ Address	Регистр текущей команды

Принцип хранимой программы: $\forall c \in Command, \exists v \in Value : decode(v) = c$
Текущая операция: $operation = decode(state(state(ip)))$

Архитектура фон Неймана

Пример

«ASSEMBLER»	Коды в памяти	Смысл
[00] : IP = 12	00 00 00 0C	Выделенная ячейка с указателем на текущую команду
[04] : AX = 0	00 00 00 00	Специальная переменная для сложения, обмена данными
[08] : IN = 0	00 00 00 00	Пользовательская переменная для ввода
[0C] : INP IN	03 00 00 08	Кодируем команду ввода
[10] : MOV AX, IN	00 04 00 08	Копируем значение в другую переменную
[14] : ADD AX, IN	01 04 00 08	Удваиваем значение. Результат в AX
[18] : OUT AX	04 00 00 04	Выводим удвоенное значение
[1C] : STOP	FF 00 00 00	Конец программы

Архитектура фон Неймана

Пример

«ASSEMBLER»	Коды в памяти	Смысл
[00] : IP = 12	00 00 00 0C	Выделенная ячейка с указателем на текущую команду
[04] : AX = 0	00 00 00 00	Специальная переменная для сложения, обмена данными
[08] : IN = 0	00 00 00 00	Пользовательская переменная для ввода
[0C] : INP IN	03 00 00 08	Кодируем команду ввода
[10] : MOV AX, IN	00 04 00 08	Копируем значение в другую переменную
[14] : ADD AX, IN	01 04 00 08	Удваиваем значение. Результат в AX
[18] : OUT AX	04 00 00 04	Выводим удвоенное значение
[1C] : STOP	FF 00 00 00	Конец программы

Кодируем команду

Архитектура фон Неймана

Пример

«ASSEMBLER»	Коды в памяти	Смысл
[00] : IP = 12	00 00 00 0C	Выделенная ячейка с указателем на текущую команду
[04] : AX = 0	00 00 00 00	Специальная переменная для сложения, обмена данными
[08] : IN = 0	00 00 00 00	Пользовательская переменная для ввода
[0C] : INP IN	03 00 00 08	Кодируем команду ввода
[10] : MOV AX, IN	00 04 00 08	Копируем значение в другую переменную
[14] : ADD AX, IN	01 04 00 08	Удваиваем значение. Результат в AX
[18] : OU	Кодируем адрес приёмника 4	Выводим удвоенное значение
[1C] : STOP	FF 00 00 00	Конец программы

Архитектура фон Неймана

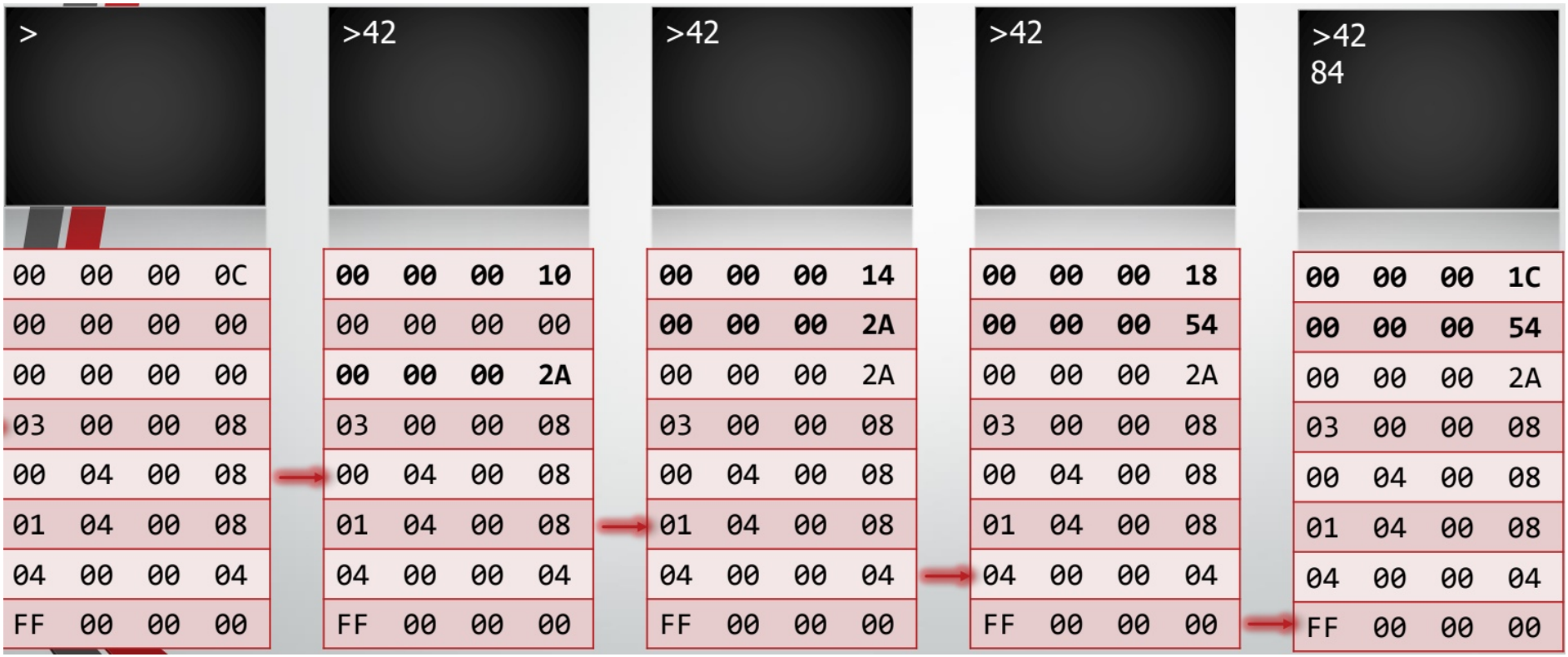
Пример

«ASSEMBLER»	Коды в памяти	Смысл
[00] : IP = 12	00 00 00 0C	Выделенная ячейка с указателем на текущую команду
[04] : AX = 0	00 00 00 00	Специальная переменная для сложения, обмена данными
[08] : IN = 0	00 00 00 00	Пользовательская переменная для ввода
[0C] : INP IN	03 00 00 08	Кодируем команду ввода
[10] : MOV AX, IN	00 04 00 08	Копируем значение в другую переменную
[14] : ADD AX, IN	01 04 00 08	Удваиваем значение. Результат в AX
[18] : OUT AX		Выводим удвоенное значение
[1C] : STOP		Конец программы

Кодируем адрес источника

Архитектура фон Неймана

Пример



Свойства языка

- Компилируемый
- Процедурный
- Статически типизируемый
- Возможность модификации памяти через указатели

Компилируемость

yallen@yallen-PC: ~

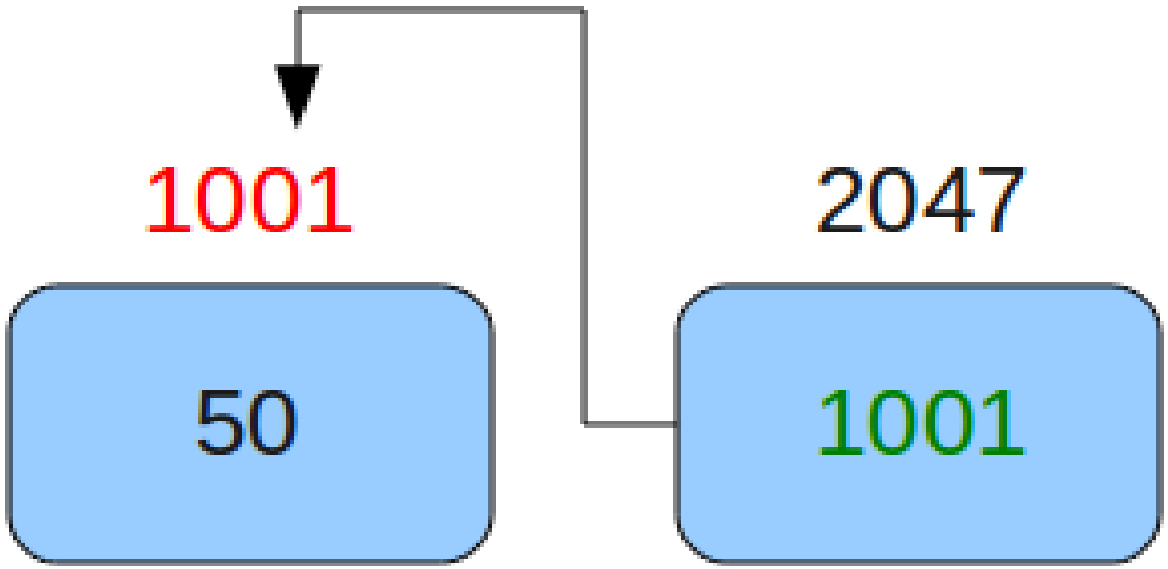
```
#include <stdio.h>

int main() {
    int a = 5;
    int b = 4;
    int c = a + b;
    printf("Addition is: %d\n", c);
    return 0;
}
~
~
~
~
~
~
~
~
~
~
~
~
~
~
                    1,1           Весь
```

yallen@yallen-PC: ~

```
        .file   "test.cpp"
        .section        .rodata
.LC0:
        .string "Addition is: %d\n"
        .text
        .globl  main
        .type   main, @function
main:
.LFB0:
        .cfi_startproc
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        subq    $16, %rsp
        movl    $5, -12(%rbp)
        movl    $4, -8(%rbp)
        movl    -12(%rbp), %edx
        movl    -8(%rbp), %eax
        addl    %edx, %eax
        movl    %eax, -4(%rbp)
        movl    -4(%rbp), %eax
        movl    %eax, %esi
        movl    $.LC0, %edi
        movl    $0, %eax
        call    printf
        movl    $0, %eax
        leave
        .cfi_def_cfa 7, 8
        ret
                 1,2-9    Наверху
```

Указатели



Полезные ссылки I


Т.Кормен, Ч.Лейзерсон, Р.Ривест, К.Штайн - Алгоритмы. Построение и анализ. - 2013, djvu
https://bit.ly/2wFzphU


Т.Кормен, Ч.Лейзерсон, Р.Ривест, К.Штайн - Алгоритмы. Построение и анализ. - 2013, pdf
https://bit.ly/2PpdqUc


Б.Керниган, Д.Ритчи - Язык программирования C - 2009, djvu
https://bit.ly/2PwcVb8


С.Липпман, Ж.Лажойе - Язык программирования C++. Базовый курс. - 2014, djvu
https://bit.ly/2LQhk6z


Working Draft, Standard for Programming Language C++
https://bit.ly/2PvGSIb

Полезные ссылки II


Викиконспекты
http://neerc.ifmo.ru/wiki/


Справка по C++
https://ru.cppreference.com/w/


C++ Super-FAQ
https://isocpp.org/faq


StackOverflow
https://stackoverflow.com/


Хабр
https://habr.com/