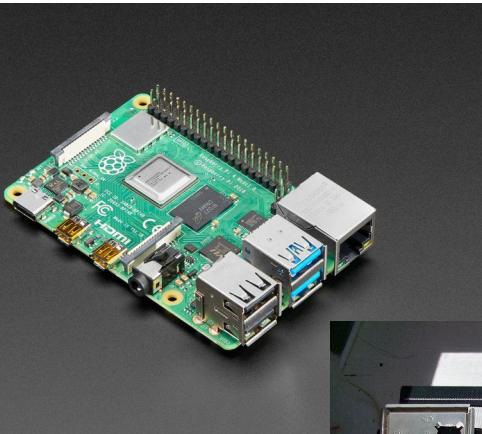


Архитектура вычислительных систем. Основные понятия



АРХИТЕКТУРА КОМПЬЮТЕРА

6-Е ИЗДАНИЕ



Э.ТАНЕНБАУМ Т.ОСТИН

PH PTR

ПИТЕР

Э.Таненбаум, Т. Остин.
Архитектура компьютера. 6-е изд. –
СПб.: Изд. Питер, 2017. – 816 с.

Рэндал Э. Брайант, Дэвид Р. О'Халларон
Компьютерные системы: архитектура и
программирование. 3-е изд.
– М.: ДМК Пресс, 2022. – 994 с.

Рэндал Э. Брайант
Дэвид Р. О'Халларон

Компьютерные системы

Архитектура и программирование

Третье издание



Сара Л. Харрис, Дэвид Харрис
Цифровая схемотехника и архитектура
компьютера: RISC-V –
М.: ДМК Пресс, 2021. – 810 с.



Дэвид М. Харрис
Сара Л. Харрис

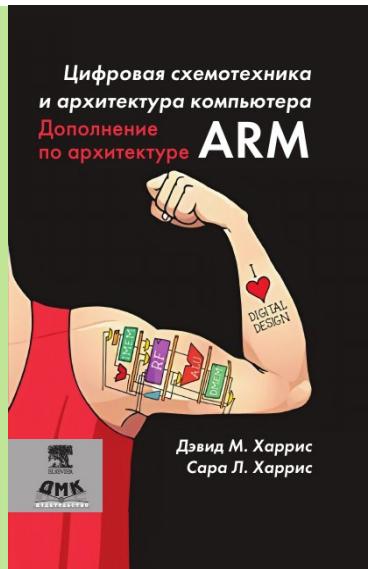


Дэвид М. Харрис, Сара Л. Харрис
Цифровая схемотехника и архитектура
компьютера. – М.: ДМК Пресс, 2018. – 792 с.

Цифровая схемотехника
и
архитектура компьютера



Цветное издание



Дэвид М. Харрис, Сара Л. Харрис
Цифровая схемотехника и архитектура
компьютера. Дополнение по архитектуре ARM –
М.: ДМК Пресс, 2019. – 356 с.

<https://uneex.org/LecturesCMC/ArchitectureAssembler2024>

О UNIX

Кто мы такие

Пользователи, программисты и администраторы UNIX-подобных систем. Поскольку семинар организован [факультетом ВМК МГУ](#), среди нас много выпускников, сотрудников и студентов ВМК.

Что нас интересует

UNIX-подобные системы. Как с их помощью решать задачи разного уровня от облегчения работы с системой до научных проблем и от чисто прикладных до глубоко внутрисистемных.

Зачем нам семинар

Обмен знаниями. Есть мнение, что сведения можно передавать в электронном виде, то есть излагать факты и считывать их. Однако понять что-либо намного проще при встрече и разговоре.

Почему UNIX

Это никем не зарегистрированное слово, отражающее специфику семинара и язык дискуссии. Выглядит эстетично.

Объявления

2024-02-11	Сетевые протоколы в Linux
2024-02-11	Совместная разработка на Python
2024-02-11	Архитектура и язык ассемблера RISC-V
2023-09-11	Разработка программного обеспечения для GNU/Linux
2023-09-03	Язык программирования Python
2023-02-13	Сетевые протоколы в Linux
2023-02-07	Совместная разработка на Python

Сайт по процессору RISC-V

<https://riscv.org/>

The screenshot shows the official website for RISC-V International. At the top, there is a navigation bar with social media icons (Twitter, LinkedIn, YouTube, RSS, etc.) and links for Languages, Tech Meetings, Community Meetings, Working Groups Portal, and Join. Below the navigation is the RISC-V logo. The main content area features a large banner with a colorful circuit board background. On the left side of the banner, the word "Announcements" is displayed in large white text. Below it, a section titled "RISC-V International Newsletter - May/June 2023" is described with the text: "Dive into the latest updates with our technical, marketing and academic groups, submit your next mentorship proposal, and check out our YouTube channel for more resources. RISC-V thrives on collaboration and community involvement." A yellow "READ MORE" button is located at the bottom left of the banner. At the very bottom of the page, there is a call-to-action button that says "Join RISC-V International".

Join RISC-V International

RARS - RISC-V Assembler and Runtime Simulator

<https://github.com/TheThirdOne/rars>

The screenshot shows the RARS 1.1 interface with the following components:

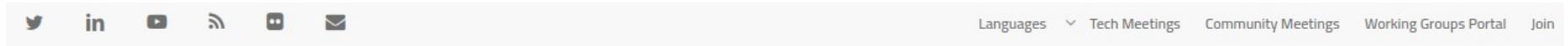
- File Menu:** File, Edit, Run, Settings, Tools, Help.
- Toolbar:** Includes icons for Open, Save, Run, Stop, Break, and various assembly instructions.
- Run Speed Control:** A slider set to "Run speed at max (no interaction)".
- Assembly Editor:** Displays the assembly code for "bottles.s".
- Registers View:** A table showing the state of 32 registers (zero through t6) and floating-point numbers (s0 through s11). The PC register shows the value 0x00400054.
- Messages View:** A scrollable window showing the output of the program execution, which prints the lyrics of "99 Bottles of Beer" repeatedly.
- Bottom Status Bar:** Shows "Line: 1 Column: 1" and "Show Line Numbers" checked.

```
File Edit Run Settings Tools Help
/home/benjamin/Documents/workspace/rars/examples/bottles.s - RARS 1.1
Run speed at max (no interaction)

Edit Execute bottles.s printf.s printnum.s printstr.s
1 .globl main
2 .data
3 template:
4     .string "%d bottles of beer on the wall. %d bottles of beer. Take one down pass it around. %d bottles of beer on the wall\n"
5
6 .text
7 main:
8     # Save state to free up $0, $1, and $a for use
9     addi    sp, sp, -16
10    sw      $1, 4(sp)
11    sw      $0, 8(sp)
12    sw      $a, 12(sp)
13
14
15
16     # Call printf(template, x, x, x-1) for 0 < x < 100
17    li      $0, 99
18 loop:
19    mv      a1, $0
20    mv      a2, $0
21    addi   $0, $0, -1
22    mv      a3, $0
23    la      a0, template
24    call    printf
25    bnez   $0, loop
26
27     # Load state
28    lw      ra, 12(sp)
29    lw      $0, 8(sp)
30    lw      $1, 4(sp)
31    addi   sp, sp, 16
32
33     # Exit (B3) with code 0
34    li      a0, 0
35    li      a7, 93
36    ecall
37
38
39
40
Line: 1 Column: 1 Show Line Numbers
Messages Run I/O
Clear
99 bottles of beer on the wall. 99 bottles of beer. Take one down pass it around. 99 bottles of beer on the wall
98 bottles of beer on the wall. 98 bottles of beer. Take one down pass it around. 97 bottles of beer on the wall
97 bottles of beer on the wall. 97 bottles of beer. Take one down pass it around. 96 bottles of beer on the wall
96 bottles of beer on the wall. 96 bottles of beer. Take one down pass it around. 95 bottles of beer on the wall
95 bottles of beer on the wall. 95 bottles of beer. Take one down pass it around. 94 bottles of beer on the wall
94 bottles of beer on the wall. 94 bottles of beer. Take one down pass it around. 93 bottles of beer on the wall
93 bottles of beer on the wall. 93 bottles of beer. Take one down pass it around. 92 bottles of beer on the wall
92 bottles of beer on the wall. 92 bottles of beer. Take one down pass it around. 91 bottles of beer on the wall
91 bottles of beer on the wall. 91 bottles of beer. Take one down pass it around. 90 bottles of beer on the wall
90 bottles of beer on the wall. 90 bottles of beer. Take one down pass it around. 89 bottles of beer on the wall
89 bottles of beer on the wall. 89 bottles of beer. Take one down pass it around. 88 bottles of beer on the wall
88 bottles of beer on the wall. 88 bottles of beer. Take one down pass it around. 87 bottles of beer on the wall
87 bottles of beer on the wall. 87 bottles of beer. Take one down pass it around. 86 bottles of beer on the wall
86 bottles of beer on the wall. 86 bottles of beer. Take one down pass it around. 85 bottles of beer on the wall
85 bottles of beer on the wall. 85 bottles of beer. Take one down pass it around. 84 bottles of beer on the wall
84 bottles of beer on the wall. 84 bottles of beer. Take one down pass it around. 83 bottles of beer on the wall
```

Specifications

<https://riscv.org/technical/specifications/>



About RISC-V ▾ Membership ▾ RISC-V Exchange **Technical** ▾ News & Events ▾ Community ▾ Q

Specifications

The RISC-V instruction set architecture (ISA) and related specifications are developed, ratified and maintained by [RISC-V International contributing members](#) within the RISC-V International [Technical Working Groups](#). Work on the specification is [performed on GitHub](#), and the GitHub [Issue mechanism](#) can be used to provide input into the specification.

Архитектура компьютерных систем

КАК СОБРАТЬ СОВРЕМЕННЫЙ КОМПЬЮТЕР
ПО ВСЕМ ПРАВИЛАМ



Нисан Ноам, Шокен Шимон.

Архитектура компьютерных систем. Как собрать современный компьютер по всем правилам — Москва : Эксмо, 2023. — 496 с.

Сайт книги:

<https://www.nand2tetris.org/>

Создание современного компьютера с нуля:
от Nand до Tetris (проектный курс)

<https://www.coursera.org/learn/build-a-computer>

Build a Modern Computer from First Principles:
Nand to Tetris Part II (project-centered course)

<https://www.coursera.org/learn/nand2tetris2>

- Хотя программисты редко пишут программы непосредственно на машинном языке, изучение низкоуровневого программирования — необходимое условие для полного и глубокого понимания того, как работают компьютеры. Кроме того, глубокое понимание низкоуровневого программирования помогает программисту писать более качественные и эффективные программы высокого уровня. Наконец, довольно увлекательно наблюдать на практике за тем, как самые сложные программные системы оказываются, по сути, потомками простых инструкций, каждая из которых задумана выполнять побитовую операцию на аппаратном уровне.

Определения архитектуры (A) ВС

Архитектура компьютера - концептуальная модель компьютерной системы, воплощённая в её компонентах, их взаимодействии между собой и с окружением, включающая также принципы её проектирования и развития.

Аспекты реализации (например, технология, применяемая при реализации памяти) не являются частью архитектуры. [Таненбаум]

- A. - **абстрактное представление ВС**, отражающее её структурную, схемотехническую и логическую организацию.
- A. - **множество взаимосвязанных компонент ВС**, включающих: программное обеспечение (software), аппаратное обеспечение (hardware), алгоритмическое обеспечение (brainware), специальное микропрограммное обеспечение (firmware) + система (структура), поддерживающая слаженное функционирование перечисленного.
- A. - **абстрактное многоуровневое представление физической системы с точки зрения программиста** с закреплением функций за каждым уровнем и установлением интерфейса между уровнями.

Структура (от лат. *structūra* - “строение”) - внутреннее устройство чего-либо. Внутреннее устройство связано с категориями целого и его частей.

Вычислительная система — компьютер, вычислительная машина (ВМ) в самом общем понимании

Мотивы понимания различных архитектур

Многие классы задач требуют одновременного понимания и использования разнообразных по уровню компьютерных архитектур.

**Часто в компаниях перебрасывают программистов с одного проекта на другой.
При этом изменяются характеристики архитектур, используемых в проектах.**

Некоторые направления предметной области изменяются за счет автоматизации, внедрения ИИ, отсутствия спроса и по другим причинам. Это ведет к необходимости переориентации на другие задачи, зачастую меняющие уровень используемой компьютерной архитектуры.

Незнание методов и подходов других архитектурных уровней зачастую ведет к неэффективному решению поставленной задачи за счет неправильно выбора инструментов, которые соответствуют текущим знаниям разработчика, но не соответствуют архитектурному уровню решаемой задачи.

- Несоответствие вниз.
- Несоответствие вверх.

Многоуровневое восприятие архитектур ВС



[Таненбаум]

Реверс-инжиниринг встраиваемых систем

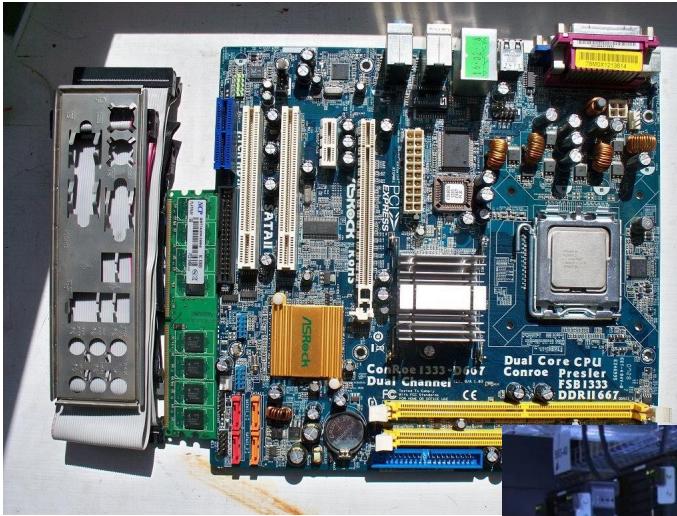
Алексей Усанов



Усанов А. Е.
Реверс-инжиниринг встраиваемых
систем. – М.: ДМК Пресс, 2023. – 296 с.

Оценка	Комментарий	На примерах Было дано задание: сделать космический корабль, готовый к запуску	Аналог в 5- балльной системе	Шкала в шакалах
10	Существенно превосходит ожидания, студент сделал значительно больше, чем от него требовал руководитель	Космический корабль уже долетел до точки назначения	5++	
9	Превосходит ожидания, студент сделал значительно больше, чем от него требовал руководитель	Космический корабль уже выходит на орбиту	5+	
8	Студент сделал все, что от него требовалось в полном объеме	Космический корабль готов к запуску	5	
7	Студент сделал почти все, что от него требовалось	Космический корабль к запуску готов, но еще не покрашен	4+	
6	В работе студента есть недочеты и/или задание сделано не полностью	В космическом корабле нет пульта управления	4	
5	Сделано мало либо в работе студента есть значительные недочеты и/или ошибки	Заказчику представлен только двигатель космического корабля	3+	
4	Сделано очень мало либо в работе студента есть очень значительные недочеты и/или ошибки	Заказчику представлена схема корабля и несколько винтиков	3	
1-3	Студент сделал критически недостаточно либо сделал не то, что требовалось	Сделали катамаран	Неуд.	
0	Выставляется, если студент не сдал документацию и/или недобросовестно работал над проектом, по мнению руководителя	Нет даже схемы космического корабля	Неуд.	

Классификација архитектур ВС



Критерии классификации архитектур ВС

Уровень восприятия

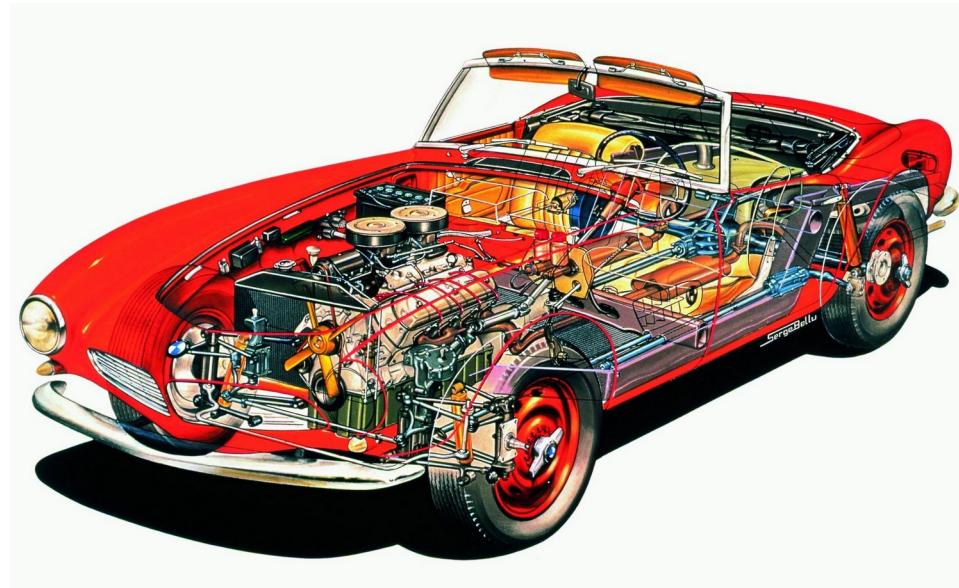
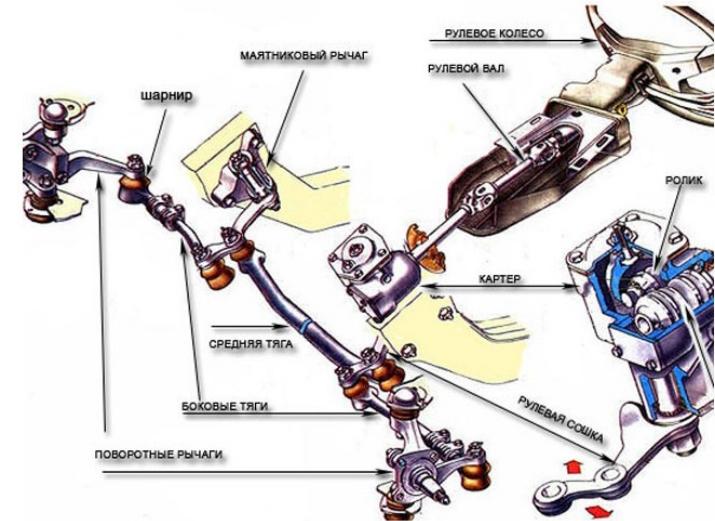
Предметная ориентация (специализация)

Парадигма (стиль) программирования

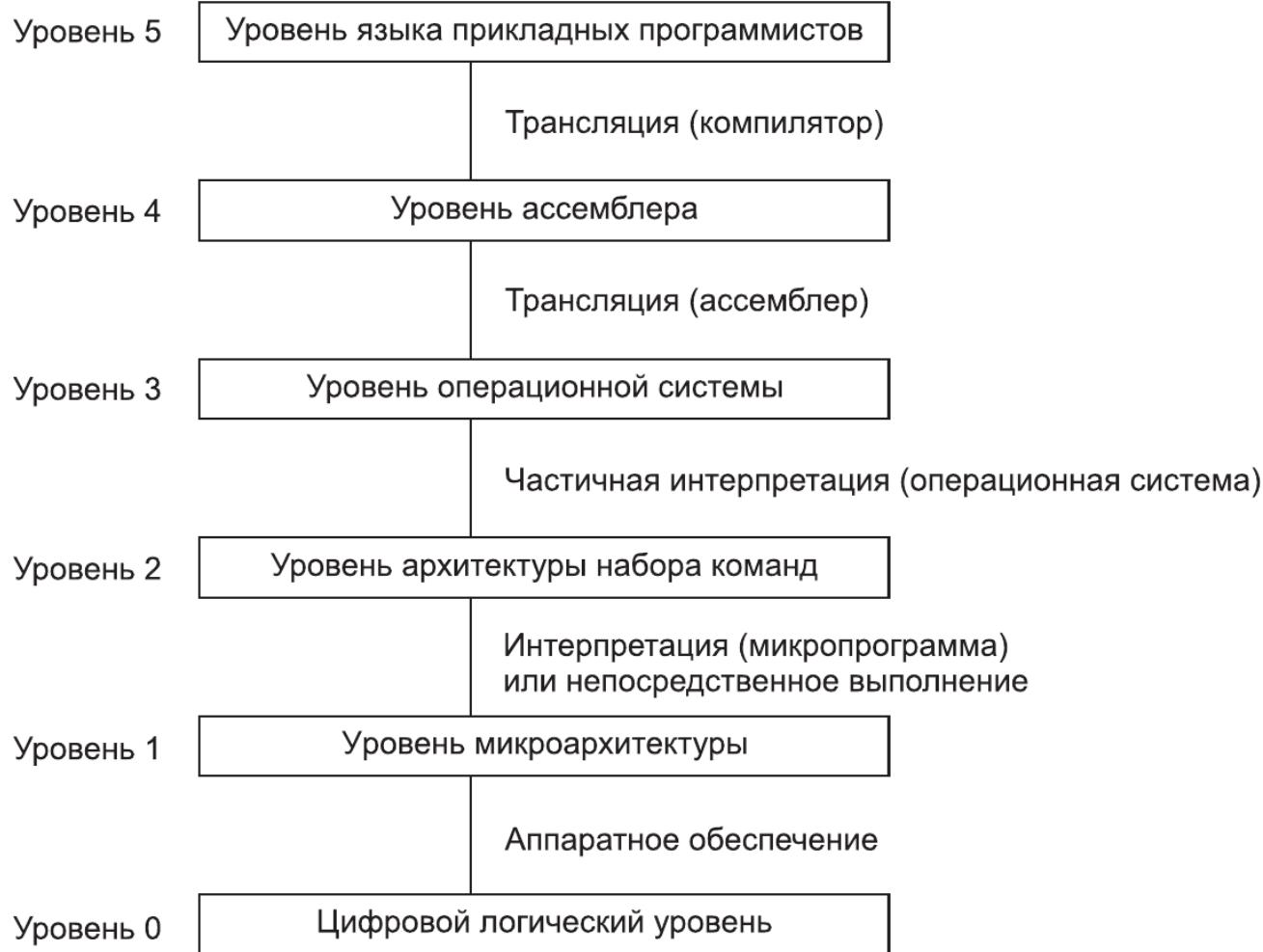
Принципы организации вычислений

Реализация архитектурного уровня

Уровни конструкции автомобиля и его узлов

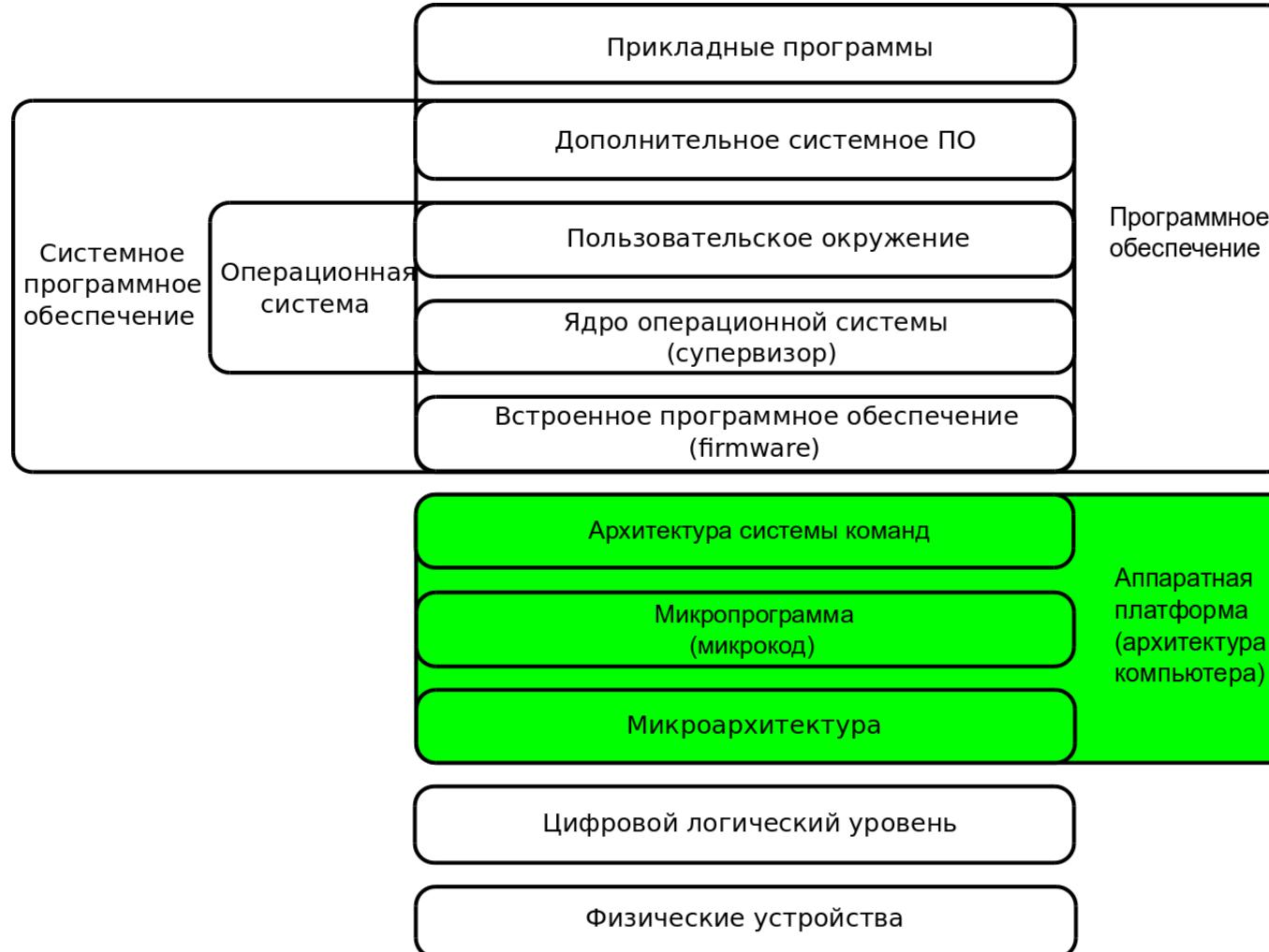


Уровень восприятия архитектуры ВС



[Таненбаум]

Уровень восприятия архитектуры ВС

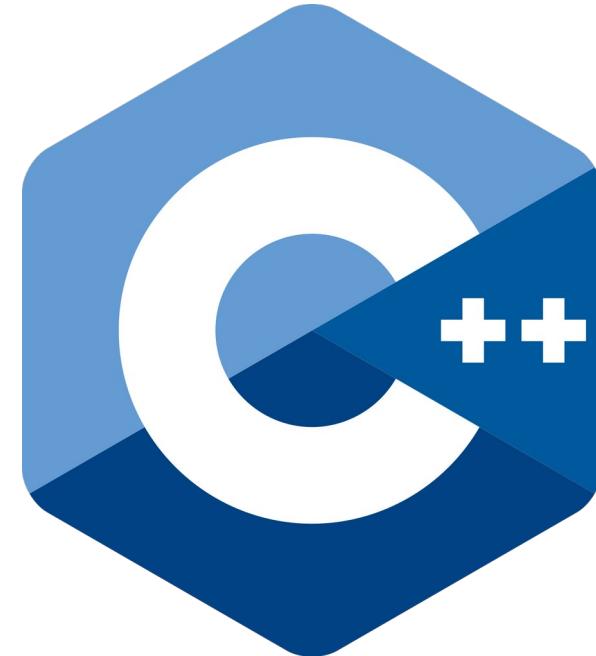


[Википедия]

Детализация уровней восприятия

- Цифровой логический уровень
- Уровень микроархитектуры
- **Уровень архитектуры набора команд**
- Уровень операционной системы
- Уровень ассемблера
- Уровень промежуточных языков (команд) – (LLVM, IL, JavaVM...)
- **Уровень языков ориентированных на системное программирования (C, C++, Rust...)**
- Уровень универсальных языков прикладного программирования (Java, Kotlin, C#, Python, Go, JS...)
- Уровень предметно-ориентированных языков (Norma, ANTLR, GPSS, Prolog, Cmake...)
- Уровень Low-code
- Уровень No-code

Пример. Проявление многоуровневости в C++



Прикладной уровень => библиотеки

- Стандартная библиотека (универсальность)
- Boost (универсальность)
- SFML (специализация)
- SDL (специализация)

Системный уровень => (уровень данного языка)

- Генераторы компиляторов (специализация)
- Встроенный ассемблер (уровень команд)

Параллелизм (специализация)

- Pthread (системный уровень)
- OpenMP(прикладной уровень)
- MPI (прикладной уровень)

Предметная ориентация архитектур ВС

Универсальные ВС

- **решение различных задач с примерно одинаковой эффективностью**

Предметно (проблемно) ориентированные ВС

- **эффективное решение задач определенного класса**

Специализированные ВС

- **решение задач достаточно узкого класса или одной задачи**

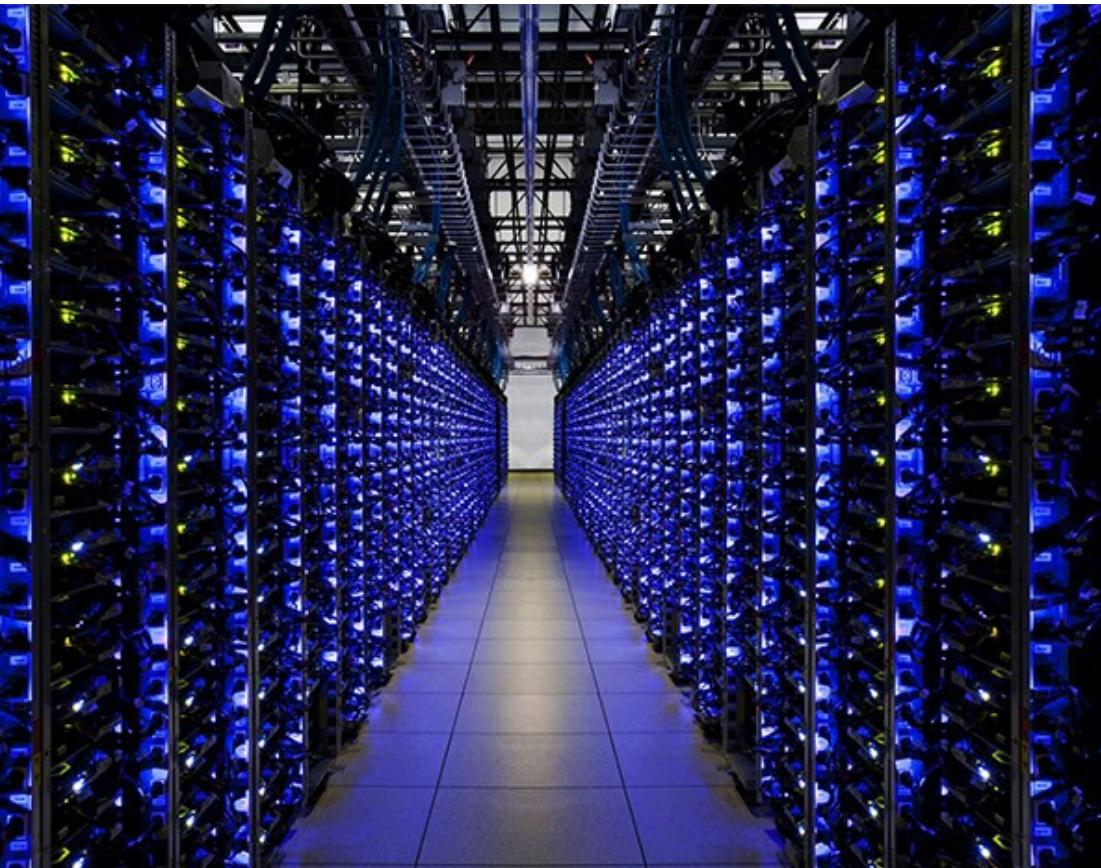
Ориентация (решение конкретных задач)



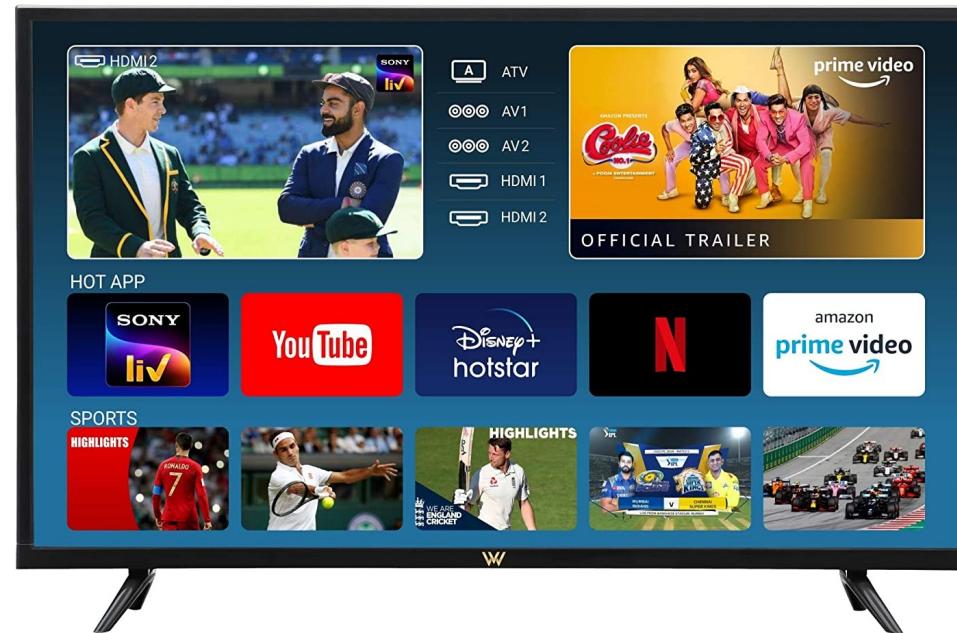
Примеры универсальных ВС



Примеры проблемно-ориентированных ВС



Примеры специализированных ВС



Парадигма программирования

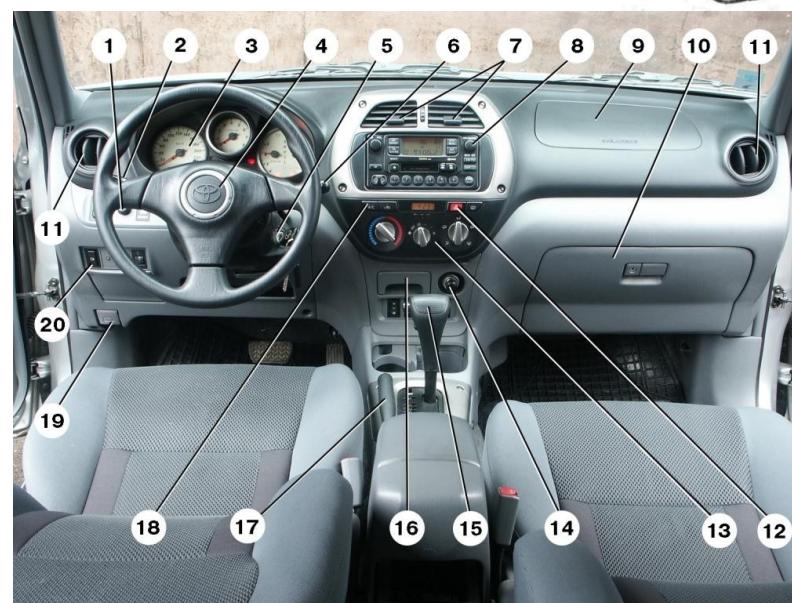
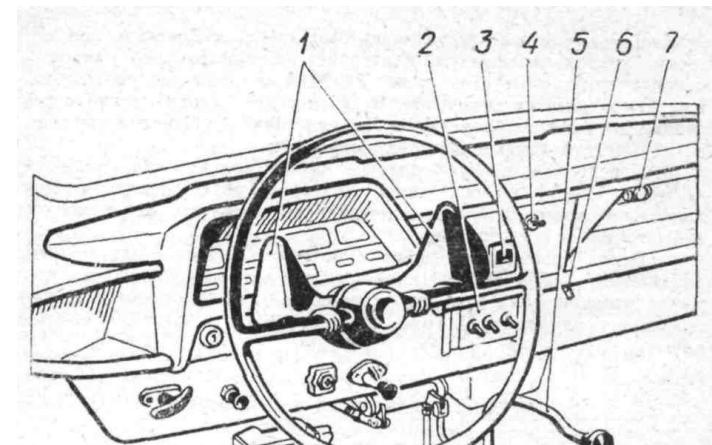
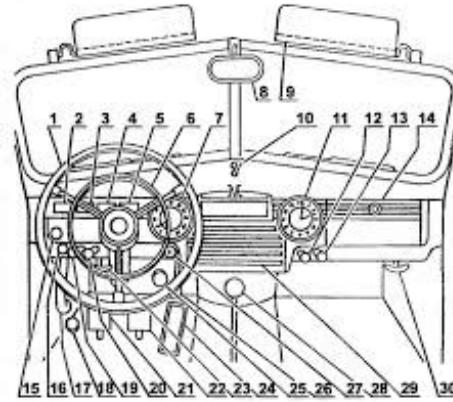
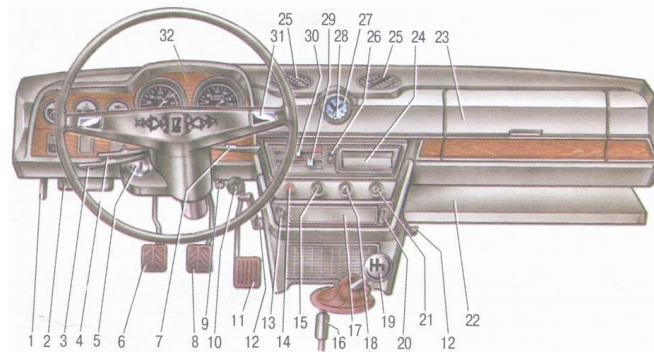
Поддержка однозначности вычислений

- *бестиповая архитектура*
- *статически типизированная архитектура*
- *динамически типизированная архитектура*

Организация управления вычислениями

- *последовательные вычисления*
- *параллельные вычисления*

Уровень управления автомобилем



Принципы организации вычислений

Дискретные компьютеры
Аналоговые компьютеры
Квантовые компьютеры



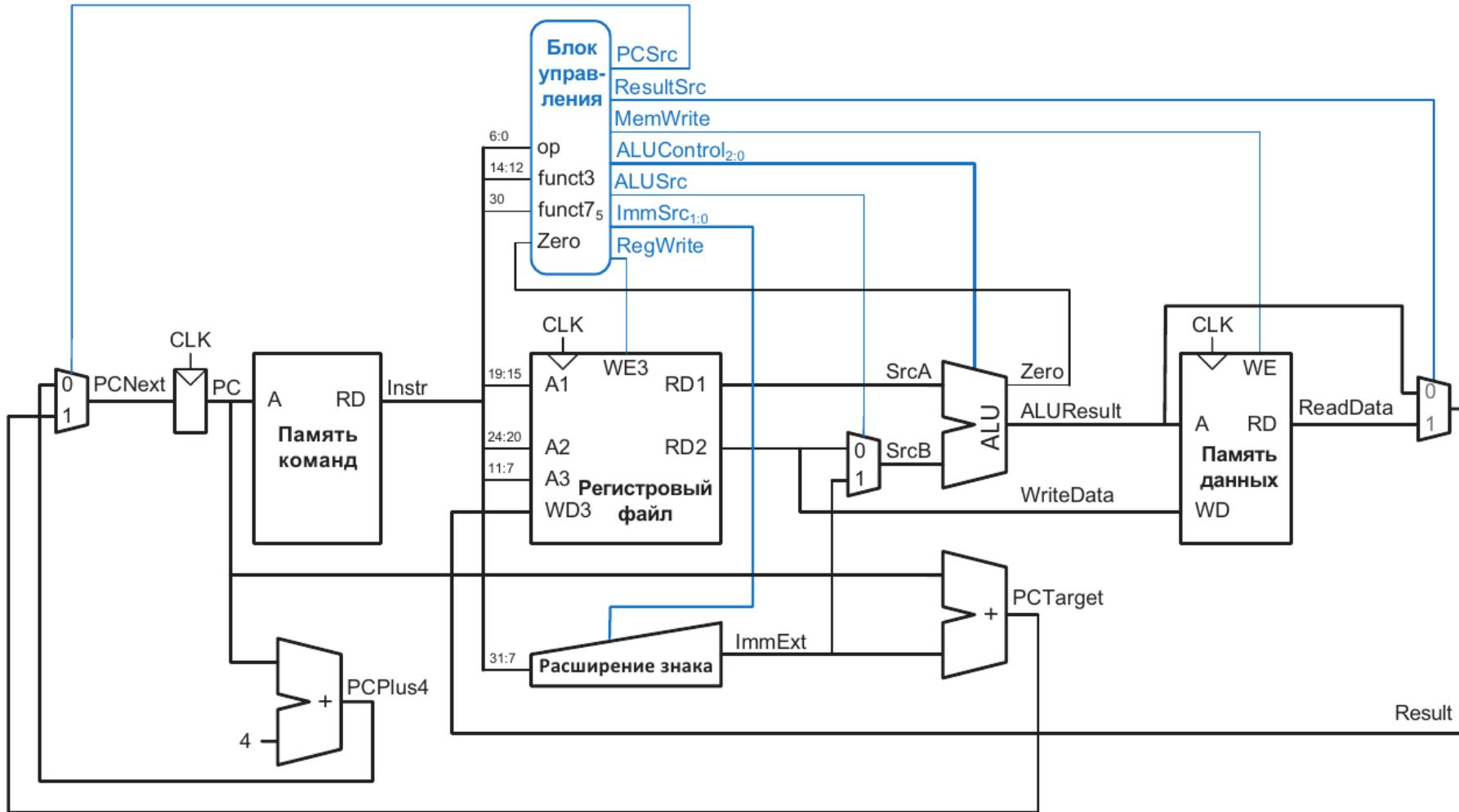
Реализация архитектурного уровня

Аппаратные решения

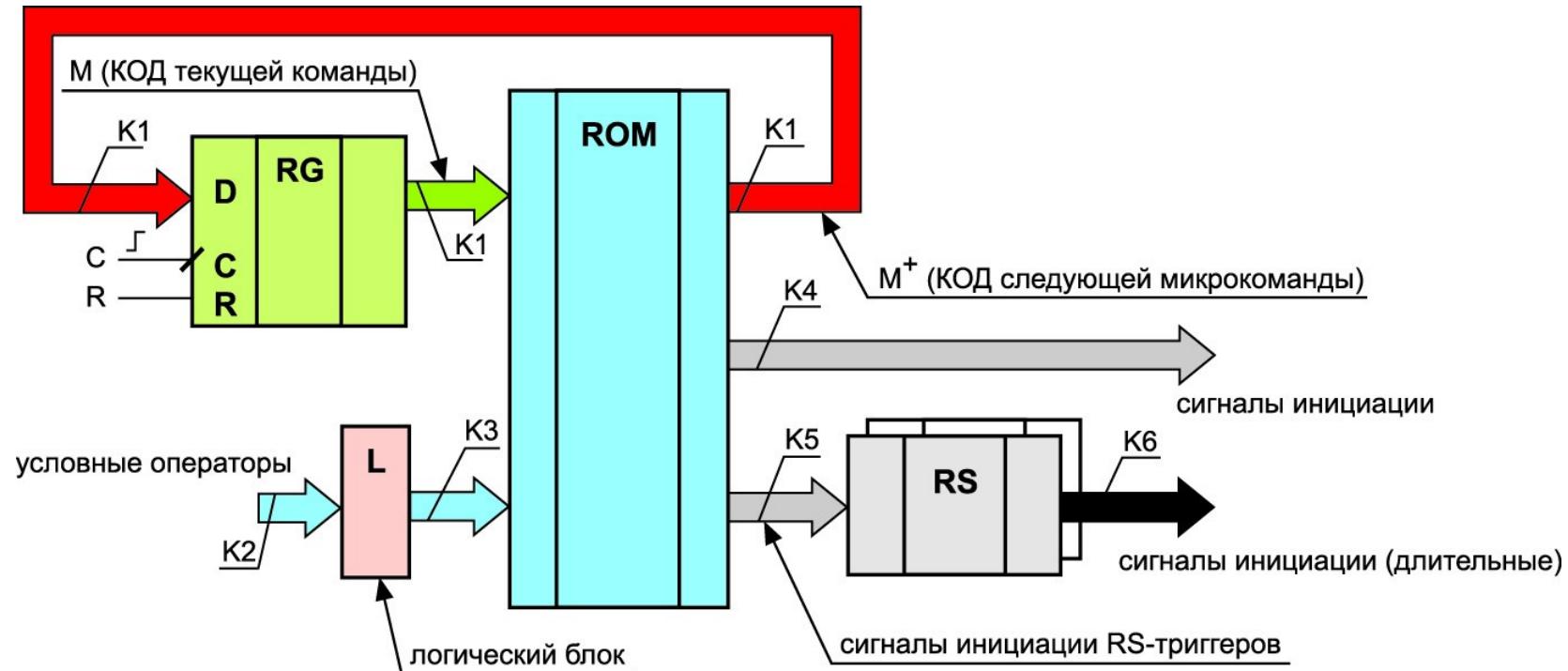
Микропрограммные решения

Программные решения (эмуляция)

Аппаратное решение



Микропрограммное решение



Эмуляция аппаратных решений



Аппаратно несуществующие архитектуры



CODE EDITOR [F1]

```
function end_arrow()
    nt:t
    score=0
    tl=60
end

function game_over()
    score=0
end

Create games
on your
mobile with
TIC-80
tunewave screen()
local
if then c=33 end
then c=35 end
if btr then c=2 end
if btr then c=7 end
Line 40/92 col 1
```

size 1465

The screenshot shows a terminal window titled "CODE EDITOR [F1]" displaying a script for the TIC-80 interpreter. The script includes functions for drawing arrows and ending the game, and a main loop for creating games on mobile devices using the TIC-80 interpreter. It also includes a "tunewave screen()" command and local variable declarations. The bottom status bar indicates "Line 40/92 col 1". To the right of the terminal are two small icons: a blue smartphone-like device with a smiley face and a teal laptop with a curly brace icon on its screen.

<https://tic80.com/>

<https://www.lexaloffle.com/pico-8.php>

