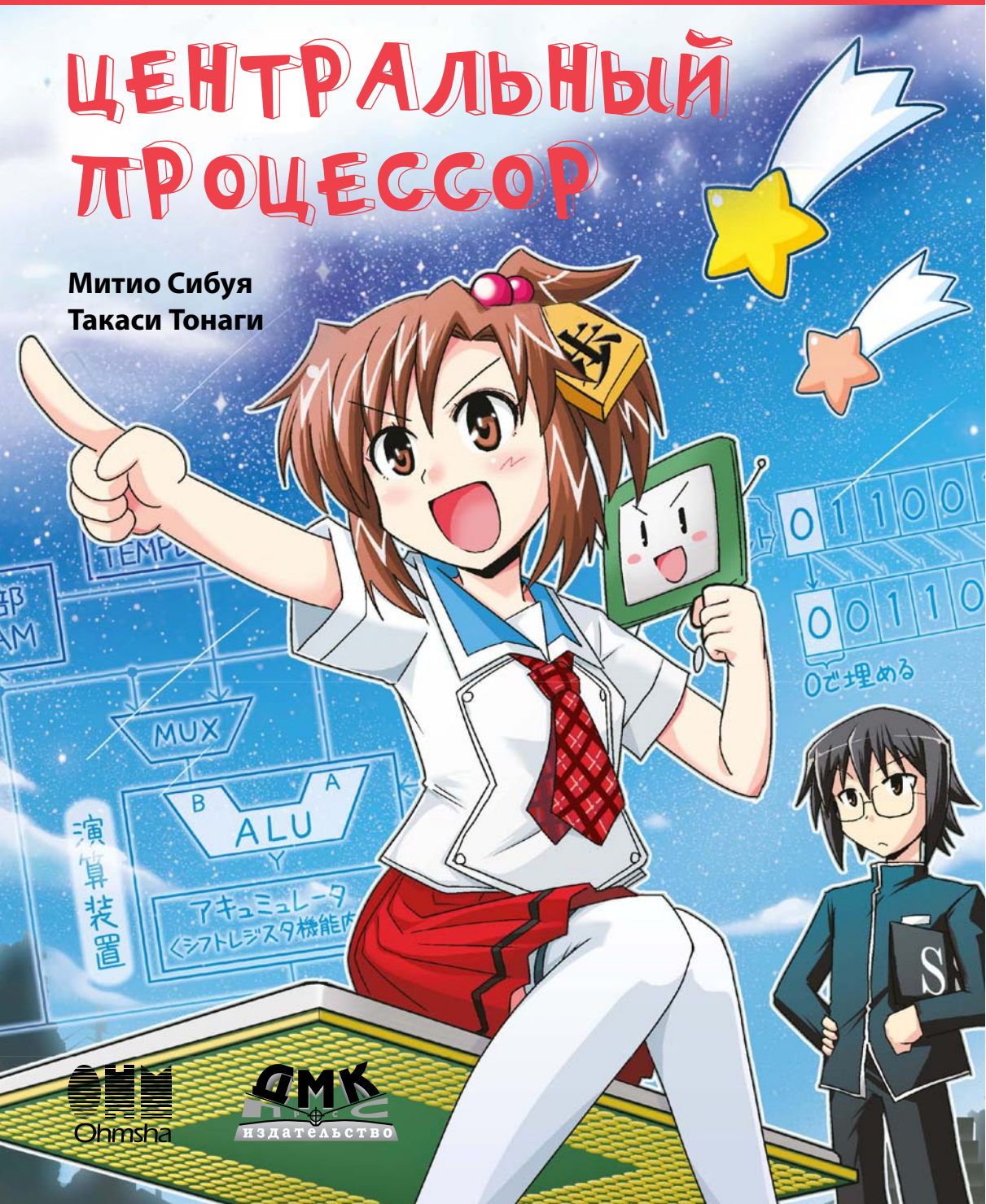
ЗАНИМАТЕЛЬНАЯ ИНФОРМАТИКА

МАНГА

ЦЕНТРАЛЬНЫЙ ПРОЦЕССОР

Митио Сибуя
Такаси Тонаги



Ohmsha
ДМК ПРЕСС
издательство

Занимательная информатика

Центральный процессор

Манга

マンガでわかる
CPU

渋谷　道雄／著
十凪　高志／作画
オフィスsawa／制作

Ohmsha

ОБРАЗОВАТЕЛЬНАЯ МАНГА

ЗАНИМАТЕЛЬНАЯ ИНФОРМАТИКА

ЦЕНТРАЛЬНЫЙ ПРОЦЕССОР

МИТИО СИБУЯ
ТАКАСИ ТОНАГИ

ПЕРЕВОД С ЯПОНСКОГО
КЛИОНСКОГО А. Б.

ДМК ПРЕСС
ИЗДАТЕЛЬСТВО

Москва
ДМК Пресс,
2017

УДК 004.318
ББК 32.971.32-04
С34

Сибуя Митио

С34 Занимательная информатика. Центральный процессор. Манга / Сибуя Митио (автор), Тонаги Такаси (худож.); пер. с яп. Клионского А. Б. — М.: ДМК Пресс, 2017 — 250 с.: ил. — (Серия «Образовательная манга»). — Доп. тит. л. яп.

ISBN 978-5-97060-507-3

Кацураги Аюми, чемпионка по японским шахматам сёги, встречает таинственного незнакомца, который предлагает ей сыграть партию с компьютером. Кто одержит верх в этом поединке — человек или машина? И какую тайную цель преследует загадочный программист?

В книге просто и доступно объясняются основы вычислительной техники, рассказывается об устройстве классического центрального процессора (ЦПУ), принципах его работы и областях применения.

УДК 004.318
ББК 32.971.32-04

Original Japanese edition
Manga de CPU (The Manga Guide to CPU)
By Michio Shibuya (Author), Takashi Tonagi (Illustrator) and
Office sawa (Producer)
Published by Ohmsha, Ltd.
3-1 Kanda Nishikicho, Chiyodaku, Tokyo, Japan

Книга «Занимательная информатика. Центральный процессор» Митио Сибуя и Такаси Тонаги подготовлена и издана по договору с Ohmsha Ltd.

ISBN 978-4-274-05061-9 (яп.)
ISBN 978-5-97060-507-3

ПРЕДИСЛОВИЕ

С тех пор как в 50-е годы XX века компьютеры нашли коммерческое применение, информационные технологии (IT) продолжают приковывать к себе пристальное внимание. Основным узлом компьютеров являются центральные процессоры, или центральные процессорные устройства (ЦПУ, англ. CPU), с приходом XXI века ставшие намного быстрее и миниатюрнее благодаря новым технологиям проектирования и изготовления микросхем. Сейчас ЦПУ используются практически во всех бытовых электроприборах. Теперь они не только в персональных компьютерах (ПК), смартфонах и планшетах, которые мы используем каждый день, но и в кондиционерах, холодильниках, стиральных машинах.

ЦПУ современных компьютеров содержат много усовершенствованных функций, поэтому я решил не рассматривать их в этой книге. По этой же причине не будут затронуты вопросы архитектуры компьютеров, хотя в настоящее время в данной области ведутся активные разработки. Но, поняв основные принципы, положенные в основу первых ЦПУ и актуальные по сей день, вы получите общее представление о центральных процессорах и поймёте, как они могут выполнять программы.

Можно использовать такую аналогию. Прошёл не один десяток лет с тех пор, как автомобили перестали быть предметом роскоши и вошли в повседневную реальность. Но хотя теперь они окружают нас повсюду, мало кто задумывается, почему они ездят, каковы принципы работы двигателя внутреннего сгорания, устройство коробки передач и т. п. Я слышал, что в 1950-х годах на экзаменах в автошколах будущим водителям задавали вопросы об устройстве двигателя; сейчас в этом нет необходимости. В наши дни только очень любознательные люди стремятся понять, как работает «начинка» того или иного технического приспособления.

Думаю, что понимание принципов работы ЦПУ, понемногу ставших неотъемлемой частью нашей жизни, поможет читателям не только удовлетворить любопытство, но и расширить свой кругозор.

По случаю выхода этой книги в свет хочу выразить благодарность г-же Савако Савада из Office sawa, придумавшей увлекательный сюжет, а также художнику манги г-ну Такаси Тонаги.

Митио Сибуя

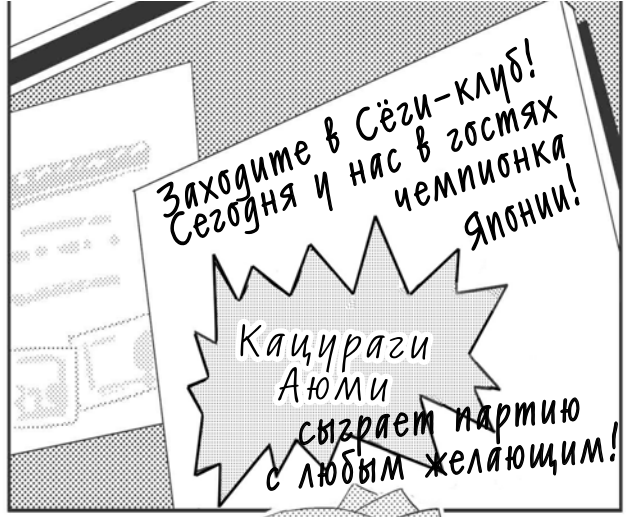
Ноябрь 2014 г.

СОДЕРЖАНИЕ

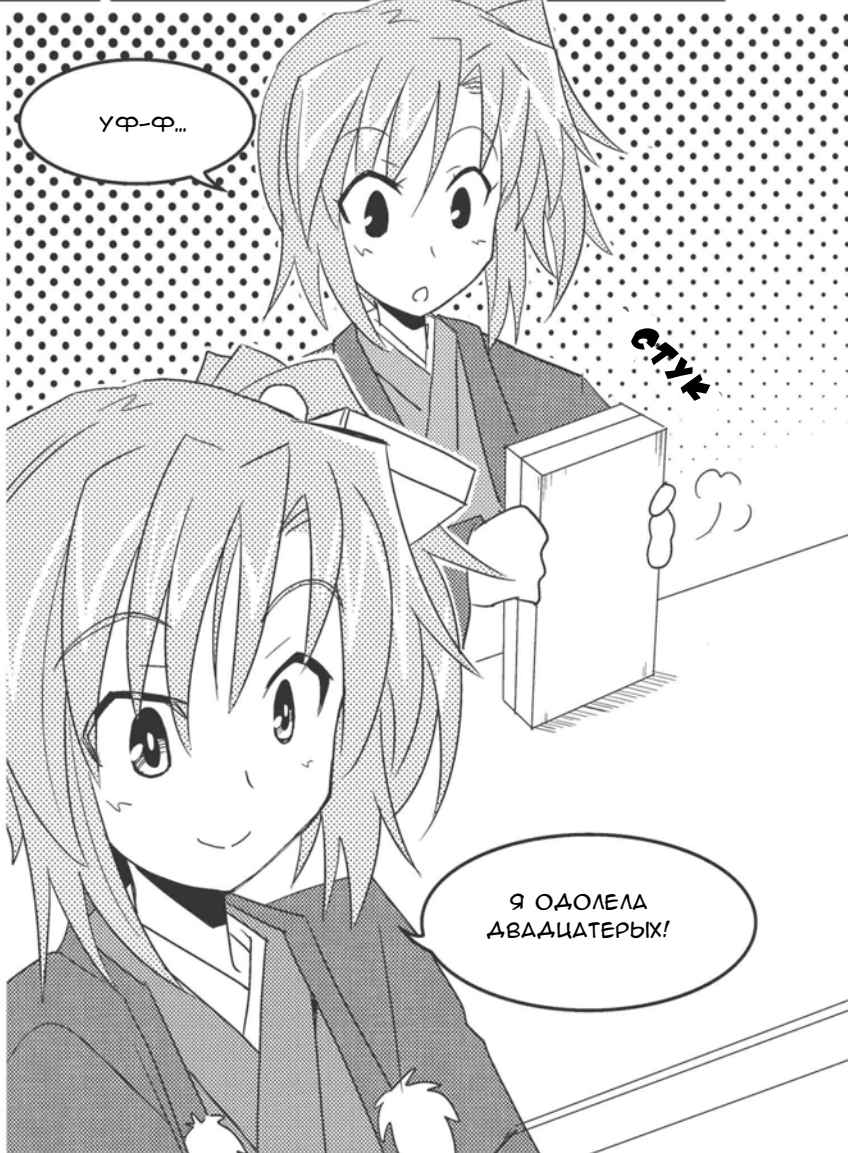
ГЛАВА 1

ЧТО ДЕЛАЕТ ЦПУ?

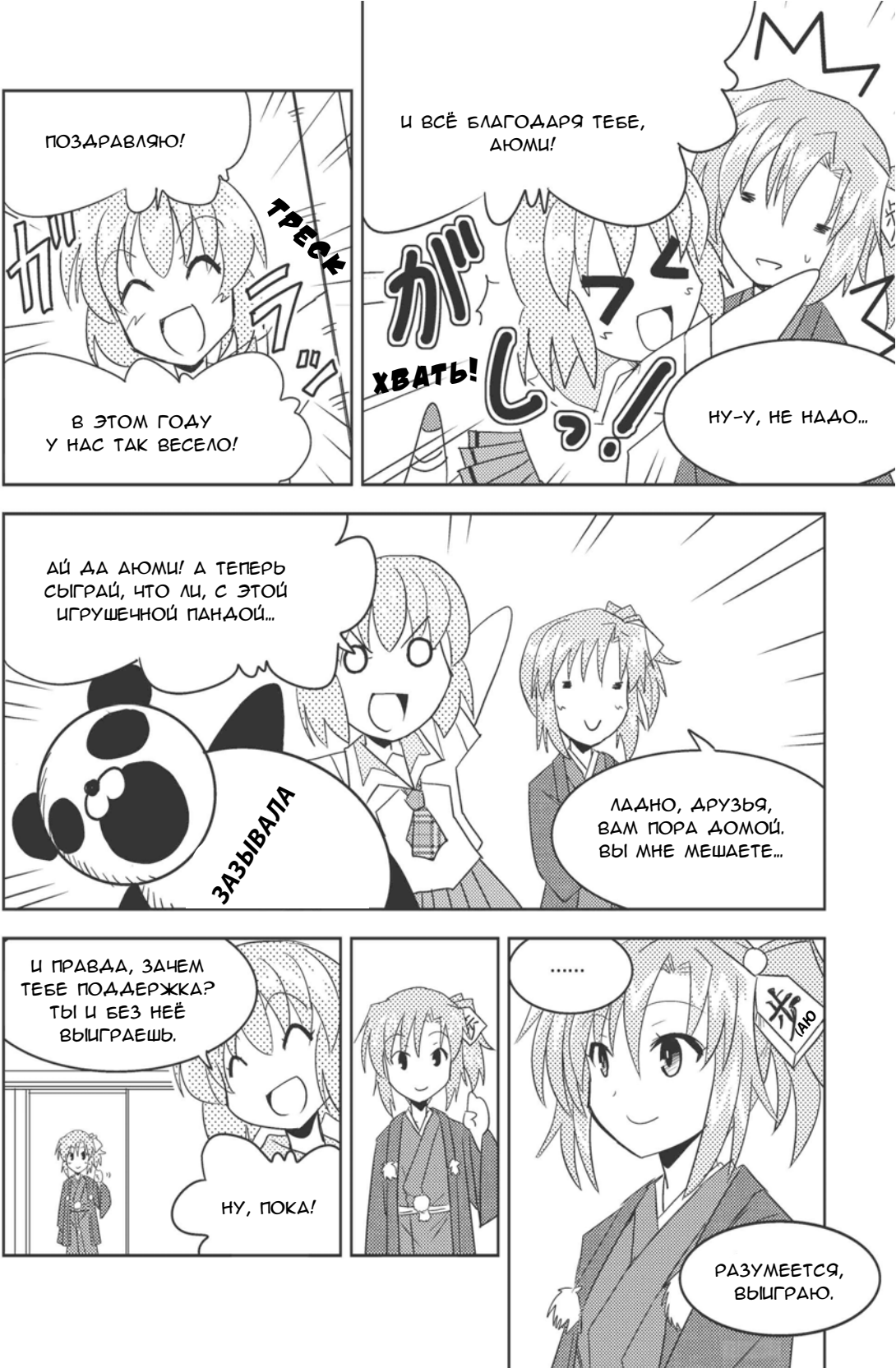

БАХ
БАХ
ФЕСТИВАЛЬ КУЛЬТУРЫ


Заходите в Сёги-клуб!
Сегодня у нас в гостях чемпионка Японии!
Кацураги Аюми сыграет партию с любым желающим!


АЙ ДА АЮМИ!
ВОТ ЭТО СИЛА!

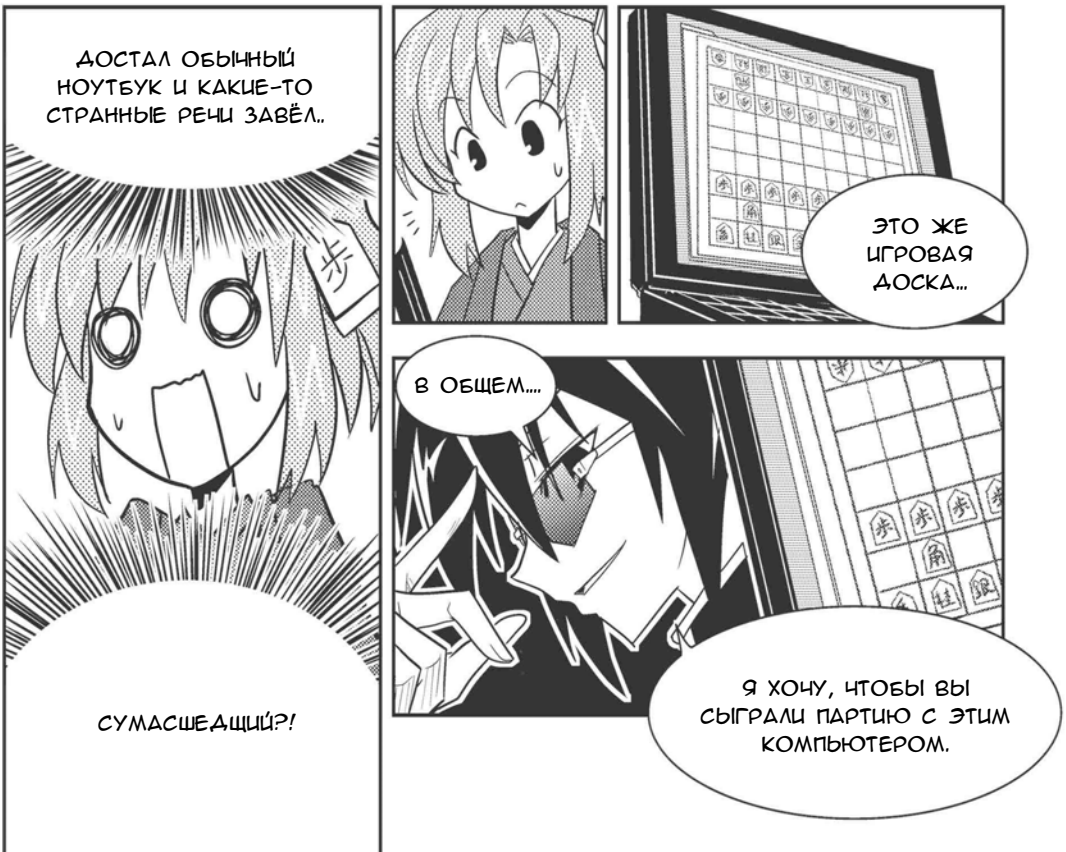

УФ-Ф...
СТУК
Я ОДОЛЕЛА ДВАДЦАТЕРЫХ!


клуб!
в гостях
чемпионка
Японии!
Кацураги
Аюми
сыграет партию
любым желающим!!


ПОЗДРАВЛЯЮ!
ТРЕСК
В ЭТОМ ГОДУ У НАС ТАК ВЕСЕЛО!
И ВСЁ БЛАГОДАРЯ ТЕБЕ, АЮМИ!
ХВАТЬ!
НУ-У, НЕ НАДО...
АЙ ДА АЮМИ! А ТЕПЕРЬ СЫГРАЙ, ЧТО ЛИ, С ЭТОЙ ИГРУШЕЧНОЙ ПАНДОЙ...
ЗАЗЫВАЛА
ЛАДНО, ДРУЗЬЯ, ВАМ ПОРА ДОМОЙ. ВЫ МНЕ МЕШАЕТЕ...
И ПРАВДА, ЗАЧЕМ ТЕБЕ ПОДДЕРЖКА? ТЫ И БЕЗ НЕЁ ВЫИГРАЕШЬ.
НУ, ПОКА!
.....
АЮ
РАЗУМЕЕТСЯ, ВЫИГРАЮ.


ДЕЙСТВИТЕЛЬНО, Я ОЧЕНЬ СИЛЬНА.
ХОТЯ, СКОРЕЕ, МОИ СОПЕРНИКИ ОЧЕНЬ СЛАБЫ... СКУКОТИЩА...
СТУК
ПРОСТИТЕ...
МОГУ ЛИ Я ПРЕДЛОЖИТЬ ВАМ СЫГРАТЬ?
А... КОНЕЧНО.
С УДОВОЛЬСТВИЕМ.
НУ, ТОГДА...


А-ХА-ХА-ХА!
СЛОВНО СТРЕМИТЕЛЬНЫЙ МЕТЕОР НА НОЧНОМ НЕБЕ...
SS
Shooting Star
ТАДАМ
А-ХА-ХА-ХА!
...SHOOTING STAR - В ЧЁРНОМ БЛЕСКЕ ЛАКИРОВАННОГО КОРПУСА!!


ДОСТАЛ ОБЫЧНЫЙ НОУТБУК И КАКИЕ-ТО СТРАННЫЕ РЕЧИ ЗАВЁЛ..
СУМАСШЕДШИЙ?!
ЭТО ЖЕ ИГРОВАЯ ДОСКА...
В ОБЩЕМ....
Я ХОЧУ, ЧТОБЫ ВЫ СЫГРАЛИ ПАРТИЮ С ЭТИМ КОМПЬЮТЕРОМ.


ВЫ ПРЕДЛАГАЕТЕ МНЕ ПОИГРАТЬ В КОМПЬЮТЕРНУЮ ИГРУ?!
ХАЛЯВО ВРЕМЯ ТРАТИТЬ...
ХИ-ХИ!. ЭТО НЕ ПРОСТО КОМПЬЮТЕРНАЯ ИГРА.


НА МОЁМ НОУТБУКЕ SHOOTING STAR УСТАНОВЛЕНА ПРОГРАММА, КОТОРУЮ Я САМ НАПИСАЛ.
ТАДАМ!
И ОНА СИЛЬНЕЕ ВАС!!!


О'КЕЙ! НЕ ПОНИМАЮ, ЗАЧЕМ ВАМ ЭТО НУЖНО, НО ВЫ ЯВНО МЕНЯ НЕДООЦЕНИВАЕТЕ.
ゴゴゴゴゴ
ВЫЗЫВ
МНЕ ДОСТАТОЧНО ВЫИГРАТЬ У ВАШЕГО SHOOTING STAR? ЛАДНО, ДАВАЙТЕ.


YOU LOSE
SS
Я ПРОИГРАЛА?
НЕ МОЖЕТ БЫТЬ!
ДАЖЕ ЧЕМПИОНКА КАЦУРАГИ АЮМИ В ПОДМЁТКИ НЕ ГОДИТСЯ МОЕМУ КОМПЬЮТЕРУ!
ПОХОЖЕ, ИГРА ОКОНЧЕНА.
КАК ЛЕГКО МОЙ ЧЁРНЫЙ ЛАКИРОВАННЫЙ ДРУГ ОДОЛЕЛ ВАС! СМИРИТЕСЬ ЖЕ С ТЕМ, ЧТО ВЫ НАМНОГО СЛАБЕЕ! А-ХА-ХА-ХА!!!


НЕ... НЕ ВЕРЮ!
ЧТОБЫ Я ПРОИГРАЛА ТАКОМУ ТИПУ!
ТАКОМУ?
Я, ТАКАЯ УМНАЯ И ХОРОШАЯ – ЭТОМУ ПРИДУРКУ?!
О ДА, КОНЕЧНО, ТЫ ХОРОШАЯ...
НЕТ, ПОГОДИ...
ЗНАЧИТ, Я НЕ ПРОИГРАЛА!
Я ЖЕ ИГРАЛА С ТВОЕЙ ПРОГРАММОЙ, А НЕ С ТОБОЙ САМИМ.
МОШЕННИК! ЧТО, ОБИДЕЛСЯ? ТОГДА ПОПРОБУЙ САМ СРАЗИТЬСЯ СО МНОЙ!


КАЦУРАГИ АЮМИ, ТЫ НЕ УМЕЕШЬ ПРОИГРЫВАТЬ!
БАМ
ДА, ТЕБЯ ПОБЕДИЛ НЕ Я, А КОМПЬЮТЕР.
НО ЭТО ОЗНАЧАЕТ, ЧТО ТВОЕМУ МОЗГУ ДАЛЕКО ДО МОЗГА КОМПЬЮТЕРА – ЦПУ.
ЭТО ДОКАЗАННЫЙ ФАКТ!

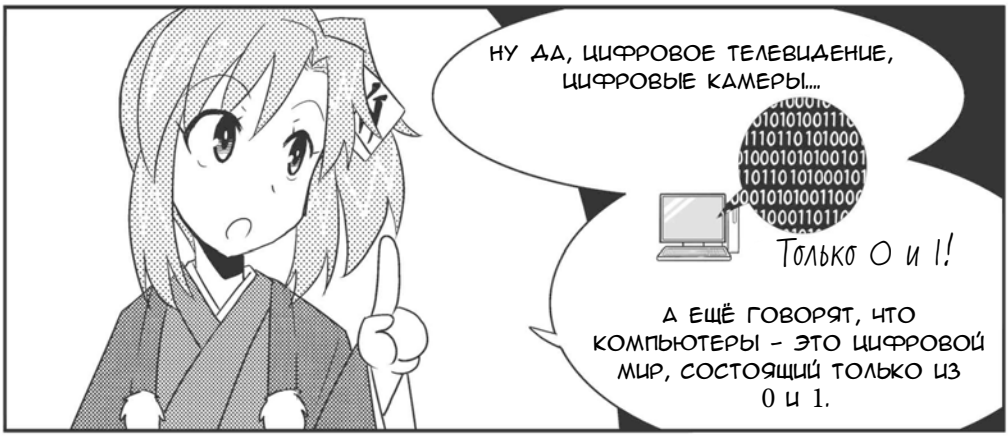

КУДА УЖ ТЕБЕ ДО МЕНЯ, ОГИНО Ю, ТАЛАНТЛИВОГО ПРОГРАММИСТА...
...УМЕЛО ИСПОЛЬЗУЩЕГО ЕГО ВОЗМОЖНОСТИ.


.....?
ЦПУ?


ЦПУ – ЭТО ДЕТАЛЬ В КОМПЬЮТЕРЕ.
ТИПА ТАКОЙ.
CPU
НО ПОЧЕМУ ЭТО "МОЗГ" КОМПЬЮТЕРА?
ЧТО ВООБЩЕ ДЕЛАЕТ ЦПУ?
ХЕ... ОБ ЭТОМ ТЕБЕ ЗНАТЬ РАНОВАТО.
НУ, ЕСЛИ УЖ ОЧЕНЬ ХОЧЕШЬ, ПОКЛОНИСЬ МНЕ ТРИ РАЗА...
ЕСЛИ НЕ СКАЖЕШЬ, ТВОЯ ДРАГОЦЕННОСТЬ НЕЧАЯННО УПАДЁТ НА ЗЁМЛЮ ~♪
КАК МЕТЕОР!
ЧЁРНАЯ ЗВЕЗДА!
ЧТО ЗА ДИКОСТЬ!!
В ДВУХ СЛОВАХ НЕ ПОЛУЧИТСЯ... А ТЫ ПОНЯТЬ-ТО СМОЖЕШЬ?
НЕ ТВОЯ ЗАБОТА! ДАВАЙ РАССКАЗЫВАЙ!
НЕ СМЕЙ С НИМ ГРУБО ОБРАЩАТЬСЯ!
ОСТАВЬ НОУТБУК В ПОКОЕ!

Компьютер работает с любой информацией


В ПОСЛЕДНИЕ ГОДЫ ИДЁТ **ОЦИФРОВКА** ДАННЫХ: МУЗЫКУ, ГРАФИКУ, ФОТОСНИМКИ И ДРУГУЮ ИНФОРМАЦИЮ ПРЕДСТАВЛЯЮТ НУЛЯМИ И ЕДИНИЦАМИ.
ТЕПЕРЬ ЛЮБЫЕ ДАННЫЕ – МУЗЫКУ, ИЗОБРАЖЕНИЯ, ВИДЕО – МОЖНО ОБРАБОТАТЬ НА КОМПЬЮТЕРЕ!

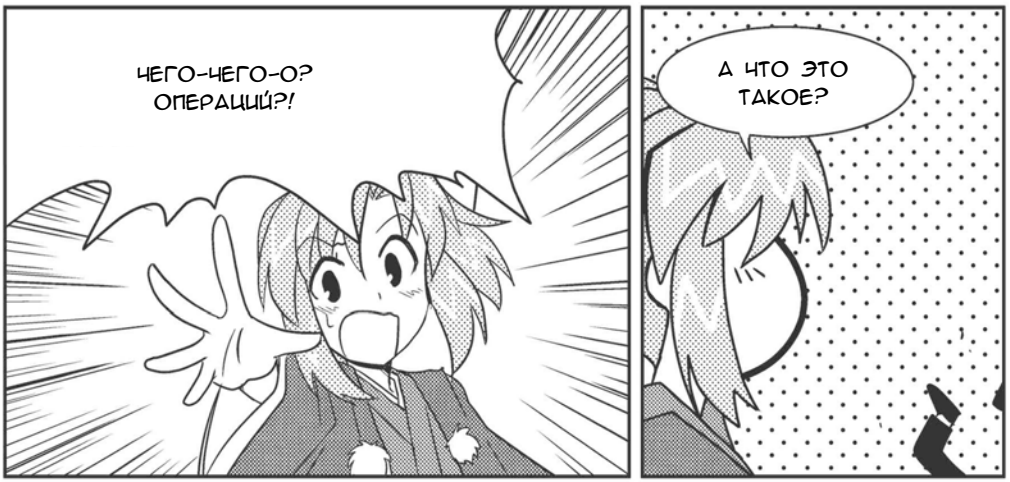

НУ ДА, ЦИФРОВОЕ ТЕЛЕВИДЕНИЕ, ЦИФРОВЫЕ КАМЕРЫ....
Только 0 и 1!
А ЕЩЁ ГОВОРЯТ, ЧТО КОМПЬЮТЕРЫ – ЭТО ЦИФРОВОЙ МИР, СОСТОЯЩИЙ ТОЛЬКО ИЗ 0 И 1.


НО ЧТО В ЭТОМ ХОРОШЕГО?
НУ КАК ЖЕ. ОЦИФРОВКА ИНФОРМАЦИИ – ЭТО НОВЫЕ ВОЗМОЖНОСТИ!
ЦИФРОВЫЕ ТЕХНОЛОГИИ ОЧЕНЬ ПОМОГАЮТ НАМ В СОВРЕМЕННОЙ ЖИЗНИ.

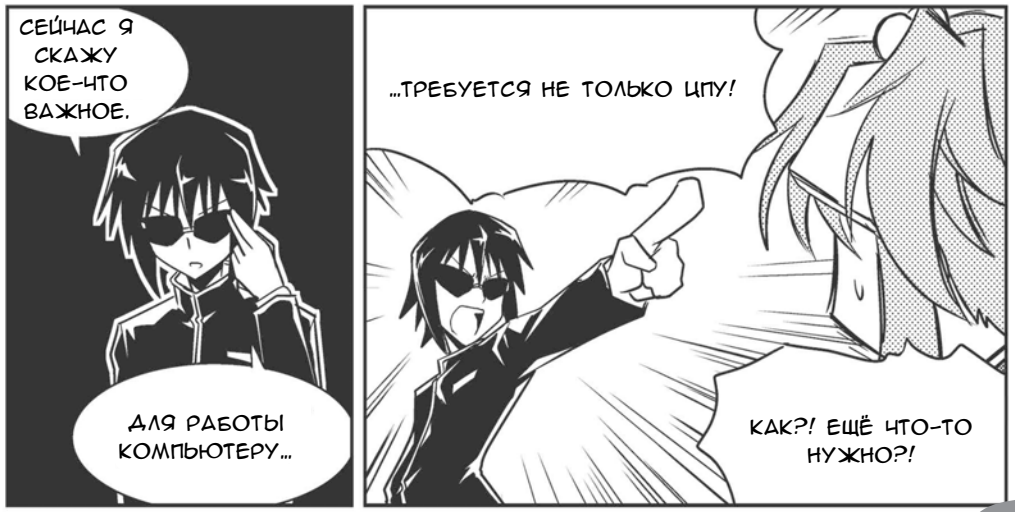

АГА!
ЭТО ВСЁ ТОЖЕ БЛАГОДАРЯ ЦИФРОВЫМ ТЕХНОЛОГИЯМ?
Подключив компьютер к сети, читаем новости, смотрим видео.
Редактируем на компьютере фотографии, снятые цифровой камерой, и посылаем их по электронной почте.
Редактируем!
e-mail
Цифровая камера
Копируем!
Скачав в сети музыку, копируем звуковые данные на плеер.
ИМЕННО! МЫ НАСЛАЖДАЕМСЯ ЦИФРОВОЙ ЖИЗНЬЮ!
ЦЕНТРОМ ЭТИХ ЦИФРОВЫХ ТЕХНОЛОГИЙ ЯВЛЯЕТСЯ "МОЗГ" КОМПЬЮТЕРА – ЦПУ.
ЦПУ
НАКОНЕЦ-ТО МЫ ПОДОШЛИ К ЦПУ! И ЧТО ЖЕ ОН ДЕЛАЕТ?

Центр компьютера — центральный процессор

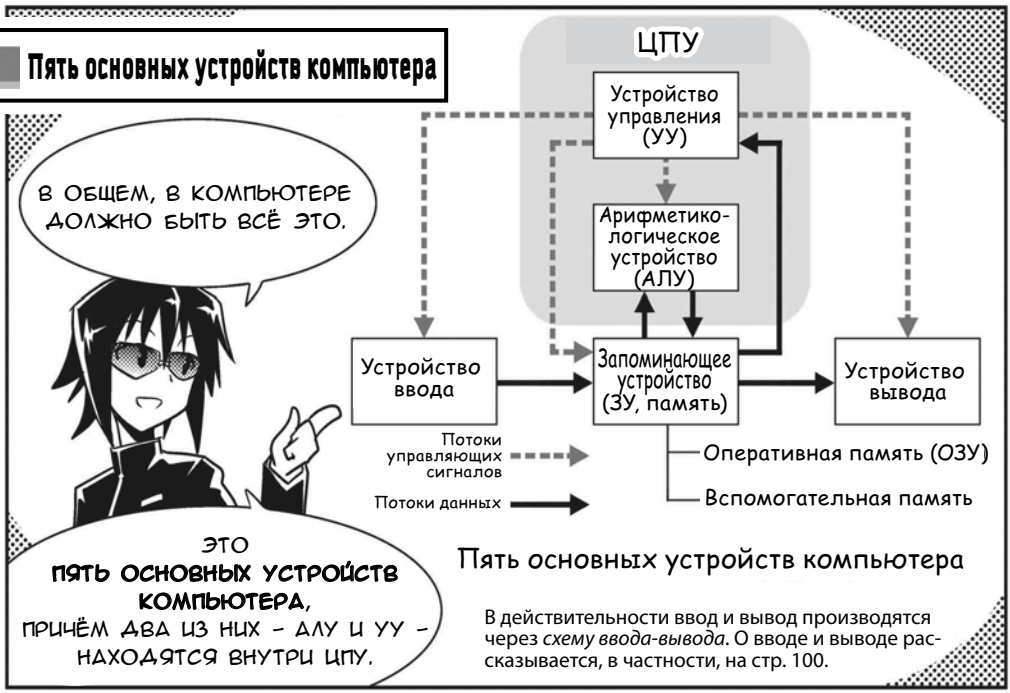
* Синонимы — «центральный процессор» (ЦП), «микропроцессор». — *Прим. перев.*



Некоторые современные ЦПУ содержат математический сопроцессор («модуль операций с плавающей точкой», англ. Floating Point Unit, FPU), способный выполнять также умножение и деление; однако эта книга посвящена только самым базовым функциям ЦПУ, поэтому математические сопроцессоры в ней не рассматриваются.



Пять основных устройств компьютера



Пять основных устройств компьютера

В действительности ввод и вывод производятся через *схему ввода-вывода*. О вводе и выводе рассказывается, в частности, на стр. 100.


НАЧНЁМ С **УСТРОЙСТВА ВВОДА**. ОНО ВВОДИТ В КОМПЬЮТЕР ДАННЫЕ И КОМАНДЫ ИЗВНЕ.
В ПЕРСОНАЛЬНЫХ КОМПЬЮТЕРАХ (ПК) ЭТО, НАПРИМЕР, КЛАВИАТУРА, МЫШЬ.


УСТРОЙСТВО ВЫВОДА, НАПРОТИВ, ВЫВОДИТ ДАННЫЕ ИЗ КОМПЬЮТЕРА НАРУЖУ.
ХОРОШИЙ ПРИМЕР – МОНИТОР И ПРИНТЕР, ПОДКЛЮЧЁННЫЕ К ПК.

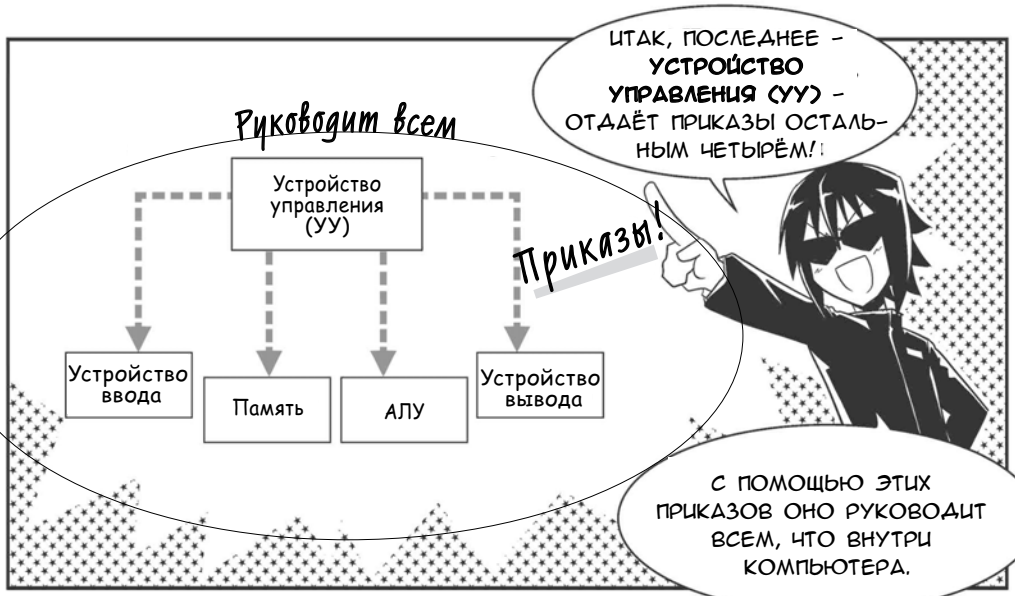

ДА, КЛАВИАТУРОЙ И МОНИТОРОМ Я ПОЛЬЗУЮСЬ.
АРИФМЕТИКО-ЛОГИЧЕСКОЕ УСТРОЙСТВО (АЛУ) ВЫПОЛНЯЕТ ОПЕРАЦИИ (ВЫЧИСЛЕНИЯ).
ГОВОРЯЩЕЕ НАЗВАНИЕ, НЕ ТАК ЛИ?


НО НЕЛЬЗЯ ЗАБЫВАТЬ ВОТ О ЧЁМ!
АЛУ МОЖЕТ РАБОТАТЬ ТОЛЬКО ВО ВЗАИМОДЕЙСТВИИ С **ПАМЯТЬЮ (ЗУ)** И **УСТРОЙСТВОМ УПРАВЛЕНИЯ (УУ)**.


ПАМЯТЬ? УСТРОЙСТВО УПРАВЛЕНИЯ?
ЭТО ДВА ОСТАЛЬНЫХ... НО ЧТО ОНИ СОБОЙ ПРЕДСТАВЛЯЮТ?

* Про вспомогательную память см. стр. 115.

* При операциях также используются, например, регистры — запоминающие устройства внутри ЦПУ. О них пойдёт речь на стр. 70.


Руководит всем
Устройство управления (УУ)
Устройство ввода
Память
АЛУ
Устройство вывода
Приказы!
Итак, последнее – Устройство управления (УУ) – отдаёт приказы остальным четырём!!
С помощью этих приказов оно руководит всем, что внутри компьютера.


Как прораб на стройке?

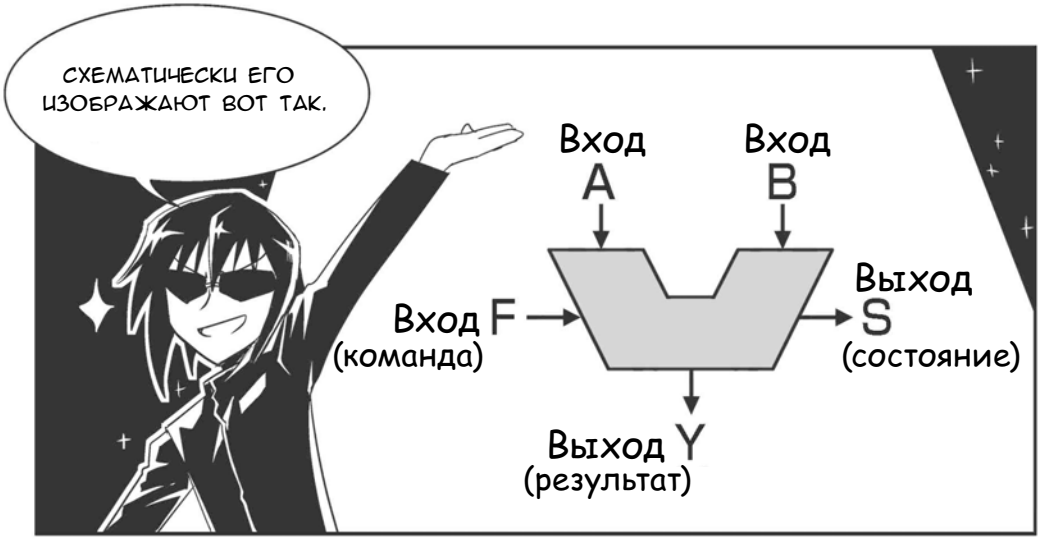

Да, вроде того. Программы (наряды на работы) находятся в памяти.


Устройство управления считывает программу из памяти, интерпретирует её...
Программа, считанная из памяти
Приказы!
Устройство управления
Принесите оттуда данные!
Сложите это и то!
Сохраните результат вон там!
...и отдаёт устройствам приказы, нужные для выполнения работ.


ЗНАЧИТ, УСТРОЙСТВО УПРАВЛЕНИЯ НУЖНО ДЛЯ ПРАВИЛЬНОГО ВЫПОЛНЕНИЯ НАРЯДОВ НА РАБОТЫ...
...ТО ЕСТЬ ПРОГРАММ!
ИМЕННО. ИТАК, Я РАССКАЗАЛ О ВСЕХ ПЯТИ УСТРОЙСТВАХ.
Вывод
Ввод
УУ
ЗУ
АЛУ
УРА! ТЕПЕРЬ Я ПОНИМАЮ СМЫСЛ ТОЙ СХЕМЫ...
(См. стр. 16)
ЦПУ
Устройство управления (УУ)
Арифметико-логическое устройство (АЛУ)
Устройство ввода
Запоминающее устройство (ЗУ, память)
Устройство вывода
УСТРОЙСТВА ОБМЕНИВАЮТСЯ ДАННЫМИ, ПОЛУЧАЮТ ПРИКАЗЫ!
НО ПРИДЁТСЯ РАЗБИРАТЬСЯ ЕЩЁ С ПОТОКАМИ ДАННЫХ И УПРАВЛЯЮЩИХ СИГНАЛОВ?
ХМ...
А-ХА-ХА! Я РАД, ЧТО ТЫ ПОНЯЛА!
ИТАК, ПЕРЕХОДИМ К СЛЕДУЩЕЙ ТЕМЕ!
ЧТО-ТО ОН РАЗВЕСЕЛИЛСЯ...
А-ХА-ХА-ХА
И ОБЪЯСНЯЕТ ТАК ПОДРОБНО...

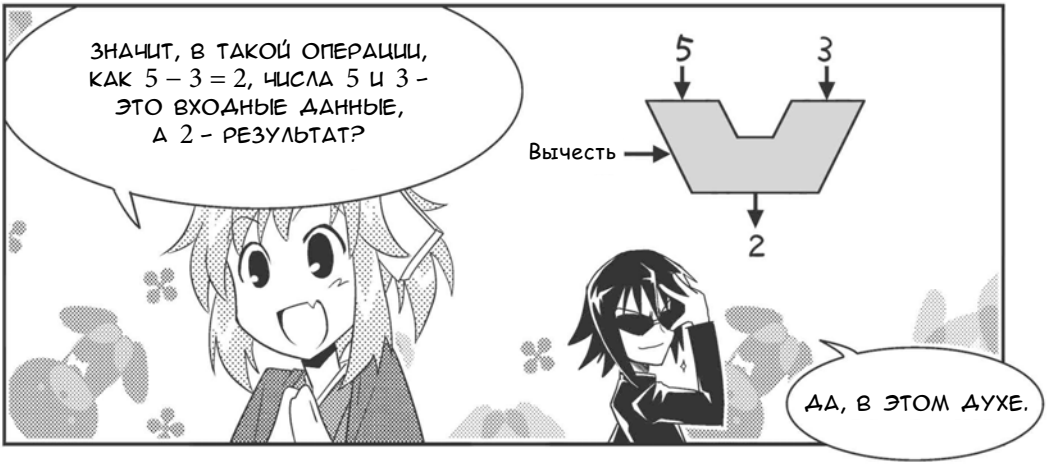
АЛУ — центральная часть ЦПУ

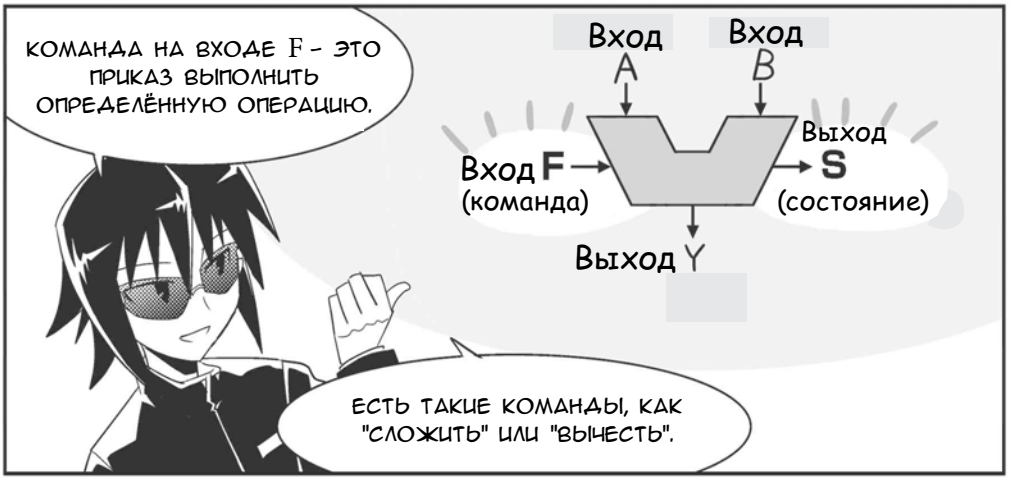
(См. стр. 15)


Схематически его изображают вот так.
Вход A
Вход B
Вход F (команда)
Выход S (состояние)
Выход Y (результат)


Что это? Не то тарелка, не то буква V...


Смысл схемы прост.
На входы A и B подаются два числа. Это **входные данные**.
Y
A
B
Результат появляется на выходе Y.


Значит, в такой операции, как 5 – 3 = 2, числа 5 и 3 – это входные данные, а 2 – результат?
5
3
Вычесть
2
Да, в этом духе.


КОМАНДА НА ВХОДЕ F – ЭТО ПРИКАЗ ВЫПОЛНИТЬ ОПРЕДЕЛЁННУЮ ОПЕРАЦИЮ.
Вход A
Вход B
Выход S
Вход F (команда)
(состояние)
Выход Y
ЕСТЬ ТАКИЕ КОМАНДЫ, КАК "СЛОЖИТЬ" ИЛИ "ВЫЧЕСТЬ".


А СОСТОЯНИЕ НА ВЫХОДЕ S – ЭТО ПРИЗНАКИ РЕЗУЛЬТАТА ОПЕРАЦИИ.
ОНО, НАПРИМЕР, ПОКАЗЫВАЕТ, ЧТО РЕЗУЛЬТАТ ИМЕЕТ ЗНАК "ПЛЮС" ИЛИ "МИНУС".

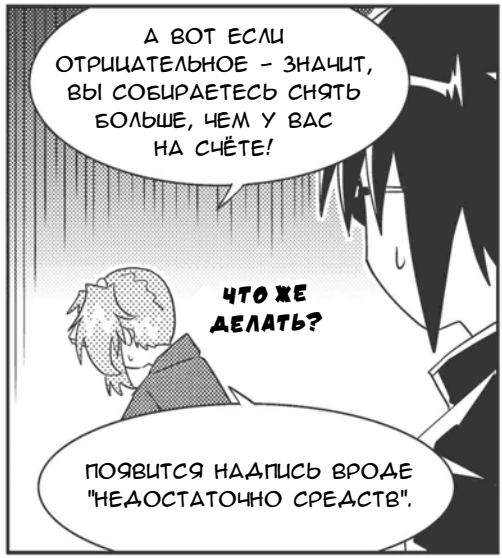

5
3
(команда) Вычесть
Состояние
Знак "плюс"
2
ЗНАЧИТ, В СЛУЧАЕ 5–3=2 СОСТОЯНИЕ ПОКАЖЕТ, ЧТО РЕЗУЛЬТАТ ИМЕЕТ ЗНАК "ПЛЮС"...


НО ЗАЧЕМ ВООБЩЕ НУЖНА ИНФОРМАЦИЯ О ПОЛОЖИТЕЛЬНОМ ИЛИ ОТРИЦАТЕЛЬНОМ РЕЗУЛЬТАТЕ?

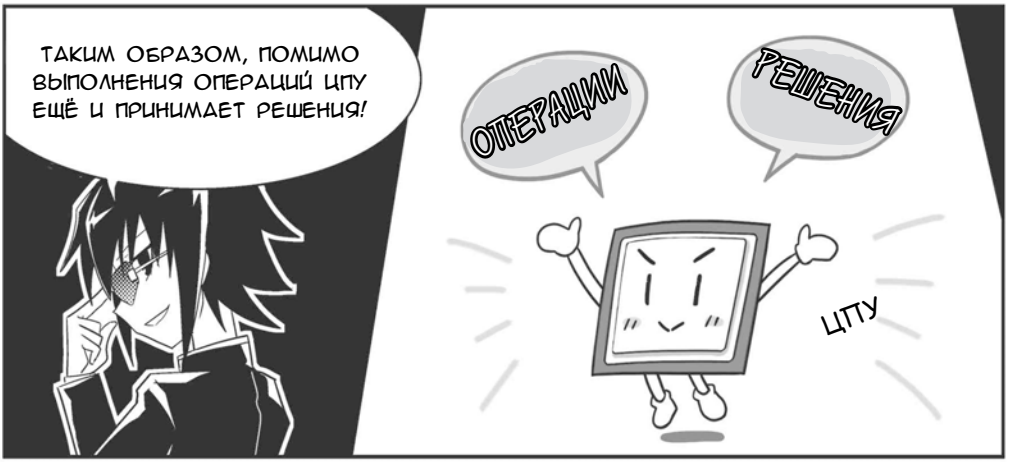

ХОРОШИЙ ВОПРОС! ВООБЩЕ СОСТОЯНИЕ НУЖНО ДЛЯ ПРИНЯТИЯ РЕШЕНИЯ О СООТВЕТСТВИИ ЗАДАННЫМ УСЛОВИЯМ.
ЗАДАННЫМ УСЛОВИЯМ? РЕШЕНИЯ?

ЦПУ выполняет операции и принимает решения


"ПЯТЬ МИНУС ТРИ" – ЭТО ВСЕГО ЛИШЬ ОПЕРАЦИЯ ВЫЧИТАНИЯ, НО ЗНАК РЕЗУЛЬТАТА – "ПЛЮС" ИЛИ "МИНУС" – МОЖЕТ БЫТЬ ОПРЕДЕЛЯЮЩИМ!
ВЕДЬ ЕСЛИ РЕЗУЛЬТАТ – ПОЛОЖИТЕЛЬНОЕ ЗНАЧЕНИЕ, ТО СНЯТЬ ДЕНЬГИ УДАСТСЯ...


А ВОТ ЕСЛИ ОТРИЦАТЕЛЬНОЕ – ЗНАЧИТ, ВЫ СОБИРАЕТЕСЬ СНЯТЬ БОЛЬШЕ, ЧЕМ У ВАС НА СЧЁТЕ!
ЧТО ЖЕ ДЕЛАТЬ?
ПОЯВИТСЯ НАДПИСЬ ВРОДЕ "НЕДОСТАТОЧНО СРЕДСТВ".


Результат со знаком «ПЛЮС»
Результат со знаком «МИНУС»
Получите наличные!
Недостаточно средств на счёте.
残高不足です
ГР-Р!..
ВЕРНО. ДРУГИМИ СЛОВАМИ, НА ОСНОВАНИИ ЗНАКА РЕЗУЛЬТАТА ВЫЧИТАНИЯ...
БОЖЖ ЦПУ
...ЦПУ ПРИНИМАЕТ РЕШЕНИЕ О ТОМ, ХВАТИТ ЛИ СРЕДСТВ НА СЧЁТЕ КЛИЕНТА, И ВЫБИРАЕТ ВАРИАНТ ДАЛЬНЕЙШИХ ДЕЙСТВИЙ.


ТАКИМ ОБРАЗОМ, ПОМИМО ВЫПОЛНЕНИЯ ОПЕРАЦИЙ ЦПУ ЕЩЁ И ПРИНИМАЕТ РЕШЕНИЯ!
ОПЕРАЦИИ
РЕШЕНИЯ
ЦПУ


ВОТ КАК? ЕСЛИ ПРАВИЛЬНО ЗАДАТЬ ПРОГРАММУ (НАРЯД НА РАБОТЫ), ТО ЦПУ И ОПЕРАЦИИ ВЫПОЛНИТ, И РЕШЕНИЯ ПРИМЕТ...
ПОВТОРЯЯ ЭТИ ДВА ДЕЙСТВИЯ, КОМПЬЮТЕР МОЖЕТ МНОГОЕ.


ДА. ОН СПОСОБЕН РАБОТАТЬ БЫСТРЕЕ ЧЕЛОВЕКА И ОБРАБАТЫВАТЬ ИНФОРМАЦИЮ ЛУЧШЕ... В ОБЩЕМ, ОБСТАВЛЯЕТ НАС КОЕ В ЧЁМ!
ХИ-ХИ-ХИ
НАПРИМЕР, В ИГРЕ СЁГИ... ИЗВИНИ, ЧТО ВСПОМНИЛ.


ДА. ТЕПЕРЬ Я КОЕ-ЧТО ЗНАЮ ПРО ЦПУ.
М-М...
НО ОСТАЛОСЬ ЕЩЁ МНОГО НЕПОНЯТНОГО! Я ЖДУ ПРОДОЛЖЕНИЯ!


...ПРАВДА?

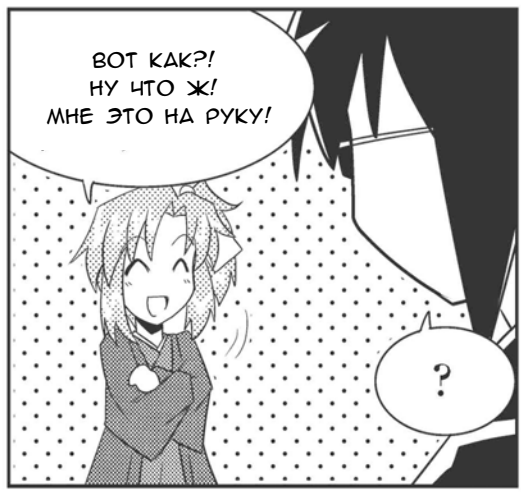

Я ХОЧУ УЗНАТЬ ОБО ВСЁМ ЭТОМ ПОБОЛЬШЕ.
КОНЕЧНО, МНЕ НЕМНОГО ОБИДНО, ЧТО Я ПРОИГРАЛА...


НА САМОМ ДЕЛЕ ЧЕРТОВСКИ ОБИДНО!!! СЕГОДНЯ НОЧЬЮ ДАЖЕ УСНУТЬ НЕ СМОГУ!
А-А-А!!


ЦПУ ТЕПЕРЬ МОЙ ЗАКЛЯТЫЙ ВРАГ!
А ВРАГА, КАК ГОВОРИТСЯ, ЛУЧШЕ ЗНАТЬ В ЛИЦО!


А МНЕ ПОНРАВИЛОСЬ ПРО КОМПЬЮТЕР РАССКАЗЫВАТЬ.


НА САМОМ ДЕЛЕ ПРОСТО ЗАХОТЕЛОСЬ УМОМ И ЗНАНИЯМИ БЛЕСНУТЬ! А-ХА-ХА!


ВОТ КАК?! НУ ЧТО Ж! МНЕ ЭТО НА РУКУ!
?


У ТЕБЯ, НАВЕРНОЕ, И ДРУЗЕЙ-ТО НЕТ...
НЕУЖТО Я ПРОИЗВОЖУ ТАКОЕ ЖАЛКОЕ ВПЕЧАТЛЕНИЕ?!
ВОТ И ЗАХОТЕЛОСЬ С ЛЮДЬМИ ПОГОВОРИТЬ, ДА? СИДИШЬ, НАВЕРНОЕ, ДНЯМИ И НОЧАМИ В ЧЕТЫРЁХ СТЕНАХ СО СВОИМИ ПРОГРАММАМИ...
ПРОДОЛЖАЙ УЧИТЬ МЕНЯ. ЭТО ТВОЙ ШАНС ПОВЫСИТЬ КОММУНИКАБЕЛЬ-НОСТЬ.
А ЕСЛИ ОТКАЖЕШЬСЯ, ТО КОМПЬЮТЕР НЕ ОТДАМ!
ХИ-ХИ-ХИ
МОЙ НОУТБУК ПОПАЛ В ЗАЛОЖНИКИ?!
ОБЕЩАЮ СЛУШАТЬ ВНИМАТЕЛЬНО!
ЭЙ! УЖЕ ВСЁ РЕШИЛА?! Я ЕЩЁ НЕ СКАЗАЛ "ДА"!

ДОПОЛНИТЕЛЬНАЯ ИНФОРМАЦИЯ

◆ Что такое информация?

Уже много лет назад в повседневный лексикон вошли такие слова, как «информация», «информационные технологии» (Information Technology, IT). Часто они имеют отношение к Интернету и компьютерным технологиям, однако не являются исключительно компьютерными терминами.

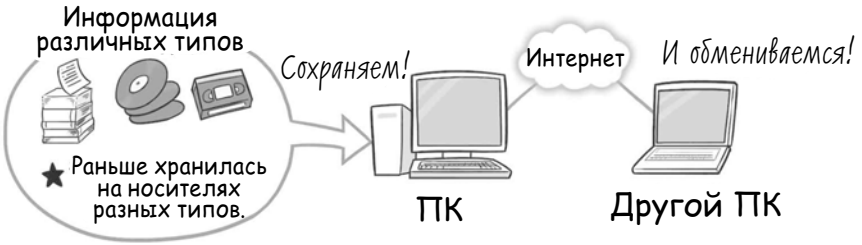
Что же такое информация? Можно дать следующее упрощённое определение: «всё, что окружает нас, воздействуя на наши органы чувств и порождая определённое восприятие».

Любые явления природы, произведения искусства (картины, фотографии, музыка, художественная литература), публицистика, новости и развлекательные передачи по радио и телевидению — всё это является информацией. Кроме радио и телевидения, всё вышеперечисленное существовало и до того, как человек научился использовать электричество. Распространяясь в социуме, информация влияет на нашу жизнь.

Особо ценится информация, полезная для индивида или определённой группы; всё остальное отсеивается. То, что не является ценной информацией, называется *шумом* (помехами). Чтобы шум не мешал воспринимать важную информацию (сигнал), нужны специальные меры, повышающие соотношение «сигнал/шум».

С давних времён больше всего ценилась информация, помогающая отдельному человеку или группе людей добывать себе пропитание. Во времена войн за плодородные территории важную информацию представляли новости об исходе сражений. В наши дни ценятся данные об урожаях зерна, иногда даже используемые для спекуляций с фьючерсами на бирже. Как следствие, оказываются важными и долгосрочные прогнозы погоды... Умелое распоряжение информационными ресурсами приносило человеку прибыль задолго до появления Интернета! Так, Кинокуния Бундзаэмон, известный купец эпохи Эдо, преуспевал в торговле, эффективно используя важные сведения. Другими словами, информация существовала и широко использовалась в докомпьютерную эру.

Тогда в чём же особая ценность взаимосвязи информации и современных цифровых технологий? Самое ценное свойство *оцифровки* (см. стр. 12) в том, что и текстовой, и звуковой (музыкальной), и визуальной информацией теперь можно обмениваться по цифровым линиям связи, в частности по Интернету, сохраняя информацию разных типов на одном и том же носителе (например, на жёстком диске) в цифровом формате. Персональные компьютеры, подключённые к одной линии связи, могут обмениваться оцифрованной информацией и обрабатывать её в соответствии с назначением.



Анализируя сочетания информации различных типов, изобретатели создают новые форматы данных, появление которых никто не мог предугадать. По мере роста объёмов информации это становится выгоднее, чем просто использовать разнотипные данные по отдельности.

Технологии связи и передачи информации развились благодаря прогрессу в электротехнике и электронике, а также распространению телеграфной и телефонной связи, радио- и телевещания. Так, до июля 2011 года практически все регионы Японии были переведены на телевещание в цифровом формате, с технической точки зрения представляющее собой сочетание технологий цифровой связи и сжатия изображений.

Центральным звеном цифровых технологий является ЦПУ, выполняющее различные вычисления.

◆ В чём разница между цифровой и аналоговой информацией?

Представление информации различных типов — текста, звука, изображений и видео, с помощью цифр 0 и 1 называют *оцифровкой*. И осуществляет её ЦПУ посредством операций.

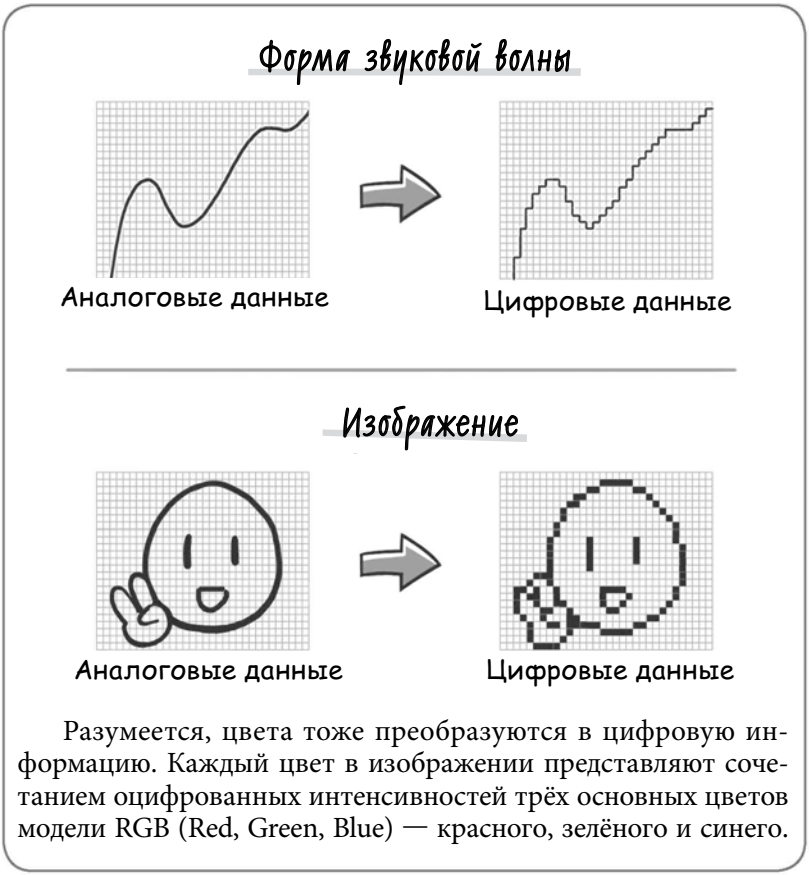
Антонимом слова «*цифровой*» является слово «*аналоговый*». В чём отличие этих двух типов данных? Разницу легко увидеть на примере термометров. В аналоговом термометре есть тонкая трубка со спиртом или ртутью и шкала с делениями, позволяющими увидеть изменение объёма жидкости при тепловом расширении. Цифровой термометр имеет датчик, преобразующий температуру в напряжение, величина которого пересчитывается в значение температуры, отображаемое на цифровом индикаторе.

Таким образом, слово «аналоговый» указывает на непрерывно изменяющуюся величину, а «цифровой» — на *дискретную величину* (т. е. величину, которая может изменяться только скачками).

Слово digital («цифровой») дословно переводится как «пальцевой» (в смысле «подсчитываемый на пальцах»), поэтому многие полагают, что информация в компьютере существует только в виде целых чисел. Но это не так. Ведь десятичные дроби конечной разрядности — это тоже цифровые значения.

Другими словами, цифровая величина может принимать дискретные значения, кратные минимальной базовой единице и ограниченные разрядностью.

Итак, чем же отличается оцифрованная информация (*цифровые данные*) от изначальной аналоговой информации (*аналоговых данных*)? Взгляните на рисунок ниже, в общих чертах показывающий это отличие.



Разумеется, цвета тоже преобразуются в цифровую информацию. Каждый цвет в изображении представляют сочетанием оцифрованных интенсивностей трёх основных цветов модели RGB (Red, Green, Blue) — красного, зелёного и синего.

Таким образом, когда информация преобразуется из аналоговой в цифровую, её объём сокращается. Конечно, сокращение должно быть настолько незначительным, чтобы человеческий глаз или ухо не смогли отличить оцифрованные данные от исходных.

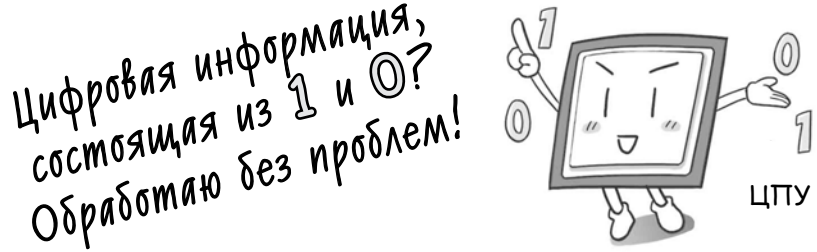
Сокращение объёма информации для последующего сохранения на цифровых носителях или передачи по цифровым линиям связи называют *сжатием данных*, а возврат её в первоначальное состояние — *восстановлением данных*.

Чтобы сократить объём информации, отбросив части, без которых качество звука, изображения или видео заметно не ухудшится, используют специальные *технологии сжатия данных*. При этом объём информации уменьшается без возможности восстановления. Такие методы называют *сжатием с потерями*.

А вот сжатую текстовую информацию нужно восстанавливать полностью, поскольку восстановленный текст должен быть точной копией первоначального. В этом случае используются методы *сжатия без потерь*.

Как бы там ни было, ЦПУ может выполнять над цифровой информацией арифметические и логические операции, что может использоваться и для сжатия, и для восстановления данных.

Таким образом, благодаря оцифровке любая информация становится объектом вычислений.



На вопрос о том, в чём смысл цифровых технологий, некоторые отвечают: «Чтобы создать мир, состоящий из нулей и единиц». Однако на самом деле смысл оцифровки в том, чтобы представленная нулями и единицами информация приносила нам пользу в жизни!

Стоит также отметить, что одним из достоинств цифровой информации по сравнению с аналоговой является устойчивость к внешним помехам (шуму), возникающим в процессе передачи по линиям связи.

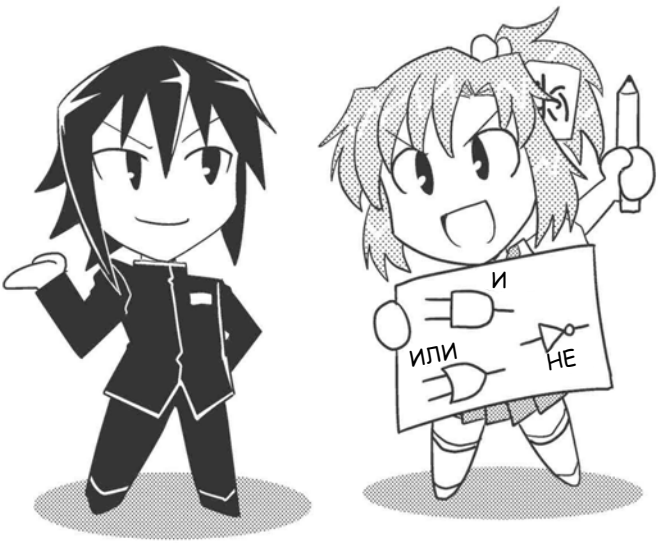
Для абстрактного выражения цифровой информации используют цифры 0 и 1, но в реальности внутри ЦПУ используются либо электрические сигналы с напряжениями так называемых «низкого» и «высокого» уровней, либо состояния отсутствия и протекания электрического тока.

Так что по сути ничего бы не изменилось, если бы в книгах вместо цифр 0 и 1 использовались значки ○ и ●.

Помните о том, что цифры 0 и 1, используемые в качестве «кирпичиков» цифровой информации, — это просто *символические обозначения*, а не реальные величины.

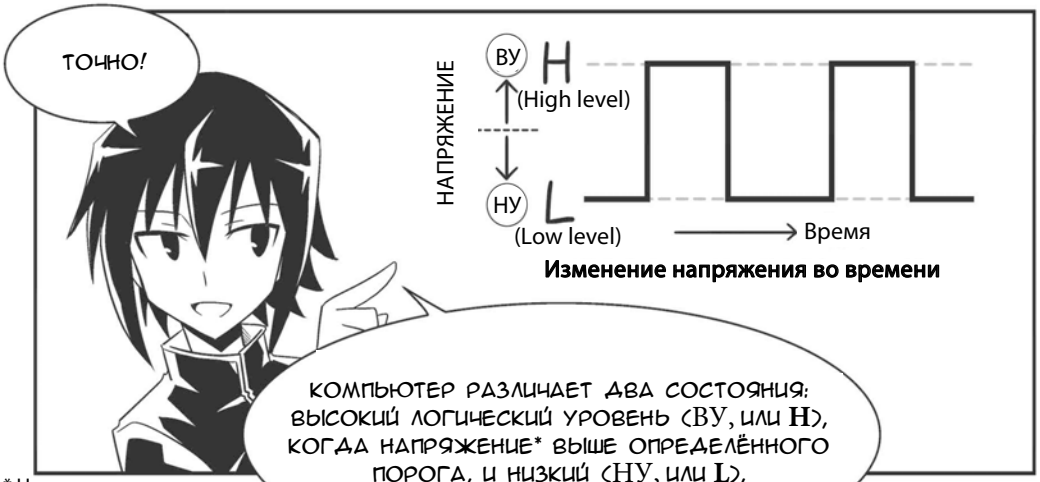
ГЛАВА 2

ЦИФРОВЫЕ ВЫЧИСЛЕНИЯ

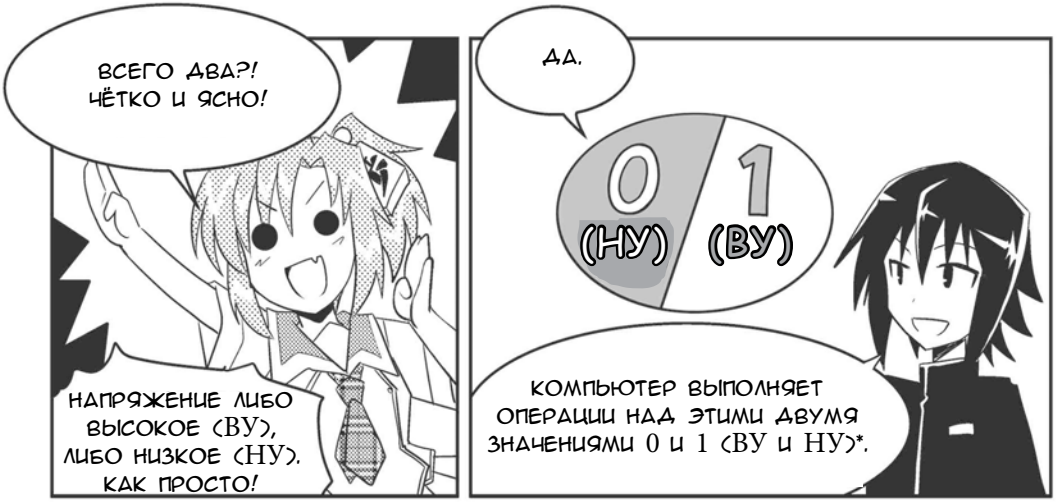


2.1. МИР КОМПЬЮТЕРА – ДВОИЧНЫЕ ЧИСЛА

0 и 1 – два взаимоисключающих состояния

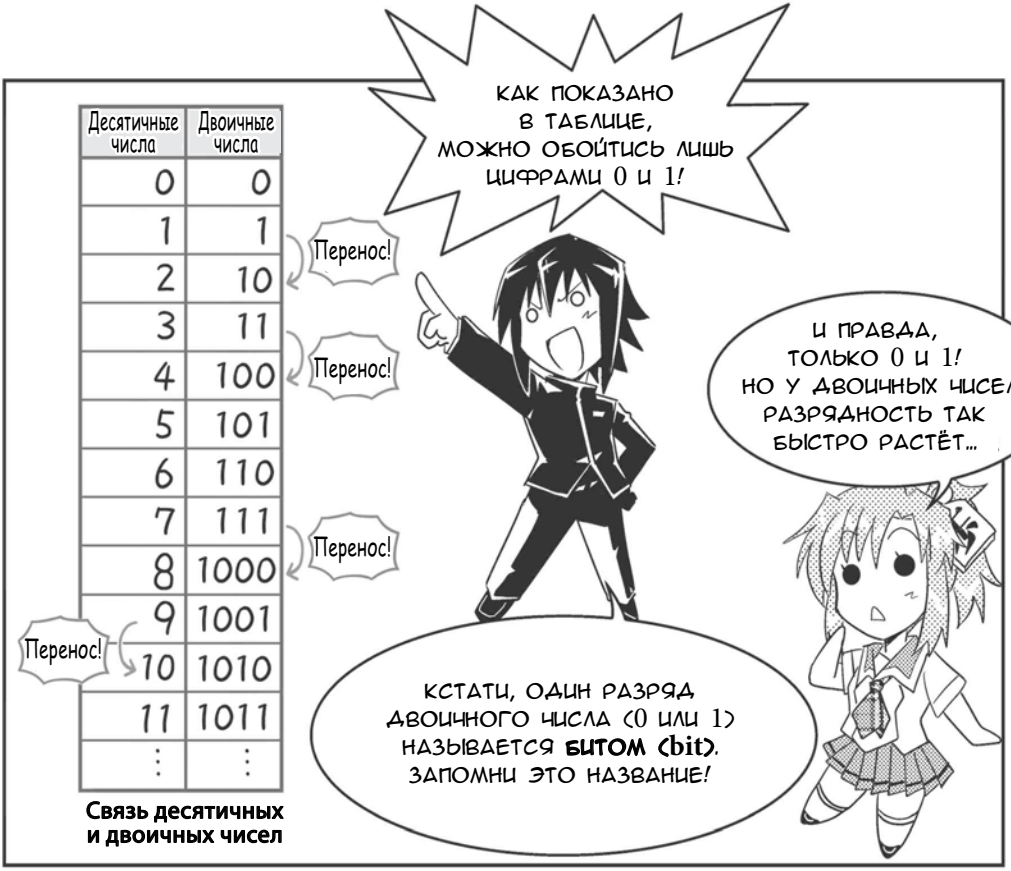


* Напряжение — это мера действия силы, вызывающей электрический ток.



* Хотя в этой книге используется так называемая *положительная логика* (0 = НУ, 1 = ВУ), существует также и *отрицательная логика* (0 = ВУ, 1 = НУ). Выбор логики зависит от разработчика конкретной системы.

Десятичные и двоичные числа

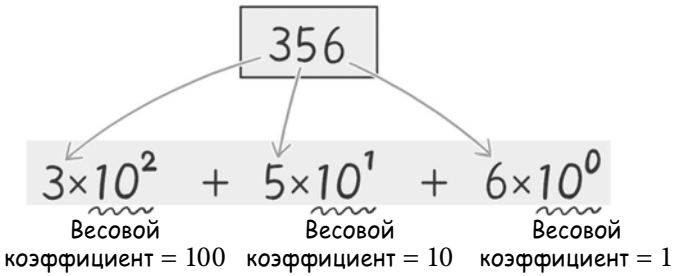

Десятичные числа
Двоичные числа
0 0
1 1
2 10
3 11
4 100
5 101
6 110
7 111
8 1000
9 1001
10 1010
11 1011
Перенос!
Перенос!
Перенос!
Перенос!
Связь десятичных и двоичных чисел
КАК ПОКАЗАНО В ТАБЛИЦЕ, МОЖНО ОБОЙТИСЬ ЛИШЬ ЦИФРАМИ 0 И 1!
И ПРАВДА, ТОЛЬКО 0 И 1! НО У ДВОИЧНЫХ ЧИСЕЛ РАЗРЯДНОСТЬ ТАК БЫСТРО РАСТЁТ...
КСТАТИ, ОДИН РАЗРЯД ДВОИЧНОГО ЧИСЛА (0 ИЛИ 1) НАЗЫВАЕТСЯ БИТОМ (bit). ЗАПОМНИ ЭТО НАЗВАНИЕ!


Один бит
1001
ЧЕТЫРЕ РАЗРЯДА – ЧЕТЫРЕ БИТА...
ЧЕТЫРЁХРАЗРЯДНОЕ ДВОИЧНОЕ ЧИСЛО – ЭТО 4 БИТА. НАПРИМЕР, ДЛЯ ДВОИЧНОГО ЧИСЛА 1001 (ДЕСЯТИЧНОЕ 9) НУЖНО ЧЕТЫРЕ БИТА...


ПРОДОЛЖУ РАССКАЗ О ДВОИЧНЫХ ЧИСЛАХ! ТЫ ГОТОВА К ПУТЕШЕСТВИЮ В МИР НУЛЕЙ И ЕДИНИЦ?..
ДРЫГ!!
ДА!
СЛОВНО ОЖИЛ...

Двоичное представление

Итак, изучим основы *двоичной системы счисления*! Для начала поразмышляем о *десятичной системе счисления*, которую все мы обычно используем.

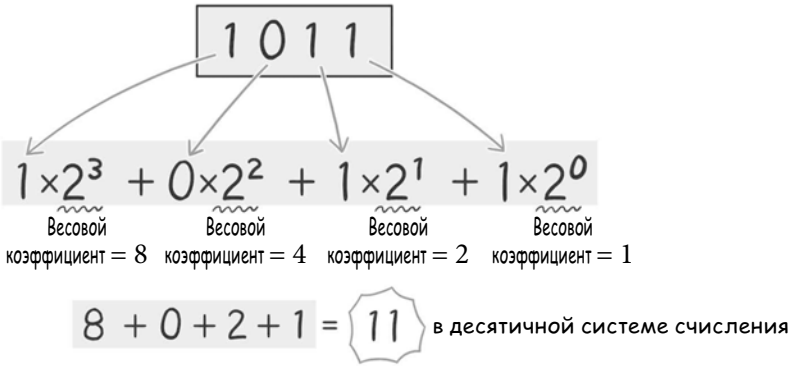


Любое число в степени 0 даёт 1.
Например, $10^0 = 1$, $2^0 = 1$.

На рисунке выше показан состав *десятичного числа* 356. Можно увидеть, что каждый разряд имеет свой вес.

Ясно! Это легко увидеть на примере монет.
356 иен состоят из трёх 100-иеновых ($10^2 = 100$), пяти 10-иеновых ($10^1 = 10$) и шести 1-иеновых ($10^0 = 1$) монет.

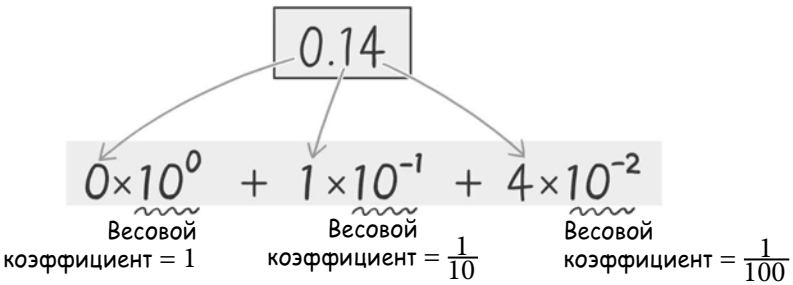
В самом деле. Итак, теперь применим это к *двоичным числам*. Вес каждого разряда расчитывается так же, только вместо 10 нужно использовать 2. Это показано на следующем рисунке.



Хм... Хотя на самом деле таких монет не существует в природе... Но предположим, что где-то имеют хождение странные монеты номиналом 8 иен ($2^3 = 8$), 4 иены ($2^2 = 4$) и 2 иены ($2^1 = 2$), а также обычные 1-иеновые ($2^0 = 1$) монеты. У каждого может быть только по одной монете каждого номинала. А разряды означают наличие (1) или отсутствие (0) соответствующей монеты.

Поэтому двоичное число 1011 в десятичной системе означает 8+0+2+1=11. Один раз поймёшь это, а потом всё легко!

Кстати, эта логика распространяется и на разряды после десятичной точки. Взгляни на рисунок ниже.



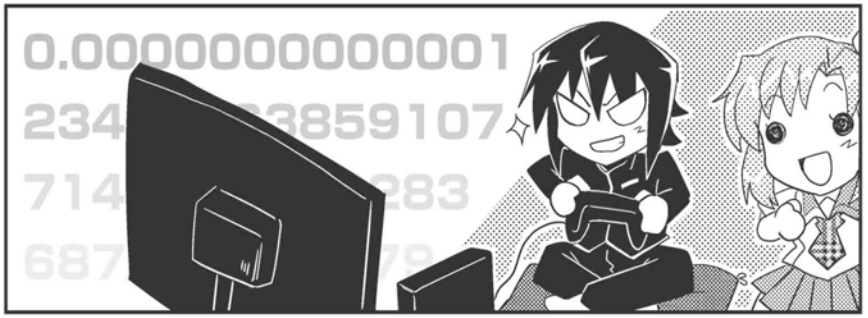
Весовые коэффициенты разрядов после десятичной точки будут $10^{-1} = 0.1$, $10^{-2} = 0.01$ и так далее.

Хм... Значит, и к двоичным числам должна быть применима подобная логика, только весовые коэффициенты разрядов после двоичной точки будут другие:

$2^{-1} = 0.5$, $2^{-2} = 0.25$, $2^{-3} = 0.125$ и так далее.

Считать лень, но идея понятна!

Представления с фиксированной и плавающей точками



Сейчас я научу тебя важным словам. На самом деле для представления чисел в компьютере используются *фиксированная* и *плавающая точки*.

Для вычислений над очень малыми числами вроде 0.00000000000000...001 или, наоборот, огромными типа 1000000000000000... намного удобнее использовать плавающую точку.

Хм... Чем же они отличаются?

Например, «сто миллионов» в десятичной системе записывается 100 000 000, но написать 10^8 и удобнее, и красивее. Это называется *экспоненциальной записью*. Она имеет вид степени 10^n, где показатель n называют *порядком*. Представление с плавающей точкой основано на этой экспоненциальной записи.

С другой стороны, представление с фиксированной точкой основано на форме записи, которую мы обычно используем. В случае целых чисел десятичную точку* ставят справа от самого младшего разряда. Связь между двумя этими формами записи выглядит вот так:

С фиксированной точкой	С плавающей точкой
123. (Десятичная точка)	1.23×10^2
1230000.	1.23×10^6
0.00000123	1.23×10^{-6}

* В ряде стран принято использовать запятую в качестве десятичного разделителя. — *Прим. ред.*

Вот оно что! Представление с фиксированной точкой может потребовать очень много разрядов, если число очень большое или, наоборот, маленькое. Но если точка плавающая, то можно просто изменить порядок! Как же это удобно!

Да, в этом духе. Мы взяли для примера десятичные числа, но в мире компьютера все числа двоичные, и это нужно учитывать. Вот пример представления с плавающей точкой в компьютере:

Значение мантиссы приведено для примера

$$1.69 \times 2^{n}$$

мантисса; порядок

Пример представления с плавающей точкой в компьютере

Число 1.69 в данном случае выбрано произвольно. Конечно, в компьютере все части этого числа представляются в двоичном формате, но здесь для наглядности я использовал десятичную форму записи. Однако в любом случае мантисса должна быть не меньше 1, но меньше 2.

Ясно! Это представление позволяет компьютеру легко работать с очень большими или очень маленькими числами.

Да! Кроме того, скорость вычислений над числами с плавающей точкой связана с производительностью компьютера (ЦПУ).

Подробнее об этом будет рассказано на стр. 139.

Для обычных *научно-технических расчётов* обычно хватает 15 разрядов, но в некоторых случаях используются 30. А, например, в современных технологиях шифрования могут использоваться целые числа, имеющие не менее 300 разрядов.

Кстати, для игровых приставок, выполняющих сложную обработку изображений с большой скоростью, тоже важны вычисления с плавающей точкой.

Да уж... Я так считать в уме точно не смогу.

Обидно, конечно, уступать компьютеру, но технический прогресс — классная штука!

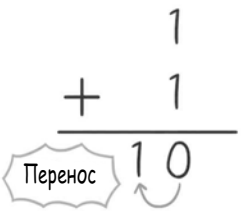
Сложение и вычитание двоичных чисел

Наконец-то мы подошли к важной теме — «операции над двоичными числами»!

Для начала поговорим о сложении... Во-первых, при сложении двух однобитовых чисел возможны следующие случаи:

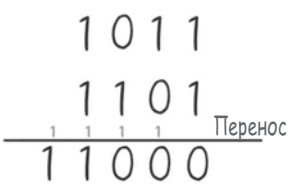
0+0=0、 0+1=1、 1+0=1、 1+1=10

Ух ты, как просто! В последней операции 1 + 1 = 10 младший разряд обнуляется, а 1 переносится в старший разряд.



Да. Освоишь однобитовое сложение — и с многобитовым тоже проблем не будет! Рассмотрим, например, операцию сложения $(1011)_2 + (1101)_2$, которая сопровождается последовательным переносом*.

* ($)_2$ означает двоичное число, а ($)_{10}$ — десятичное.



Хм... значит, достаточно помнить о переносе. Да, складывать двоичные числа довольно просто. Или это просто я такая умная?..

Ладно. Теперь про *вычитание*. Здесь требуется изменить знак вычитаемого, используя такое понятие, как *дополнительный код*.

Сложить с дополнительным кодом — то же самое, что вычесть! Правда, круто?!

Ты слишком увлёкся... Я что-то перестала понимать, что ты говоришь.

Хм... Тогда попробую объяснить на примере десятичных чисел. Например, вычесть 15 — это то же самое, что прибавить –15, правильно?

Теперь представь, что у нас нет знака «минус». Что делать? Нельзя ли –15 выразить как-нибудь по другому?

Ну, этого даже я не знаю... Рассказывай, воображала!

Тогда взгляни на эти две операции сложения.

***Операция* А:**

$15 + (-15) = 0$

$$\begin{array}{r} 15 \\ +(-15) \\ \hline 0 \end{array}$$

***Операция* Б:**

$15 + (85) = 100$

$$\begin{array}{r} 15 \\ +(85) \\ \hline 100 \end{array}$$

Игнорируем!

Сравнивая только два разряда, видим, что результат операции **А** — 0, результат операции **Б** — 00.

Значит, если рассматривать только два разряда, результат операции 15 + (–15) будет абсолютно тот же, что и результат операции 15 + (85)!

Вот это да-а! В самом деле, численно 0 и 00 ничем не отличаются! Но как быть с единицей в результате 100 операции **Б**?

Если это двухразрядная операция, то нас интересуют только два разряда! Перенос из старшего разряда — *переполнение* — нас не волнует! Считай, что мы его не видели!

Ничего себе! Ну и отговорки пошли! Возможно ли такое?!

Хи-хи-хи... Удивлена?

В этом случае говорят, что 85 является *дополнением* 15 до 100. *Дополнительный код* — это такое дополнение, при сложении которого с исходным числом все разряды обнуляются, происходит перенос из старшего разряда (*арифметическое переполнение*).

И этот дополнительный код используют как отрицательное число. Например, в нашем случае 85 равносильно –15 с точки зрения результата.

Например, операцию вычитания 9647 – 1200 = 8447 можно выполнить путём сложения 9647 + 8800 = 18447, игнорируя перенос из старшего разряда. Как видишь, четыре младшие разряда совпадают.

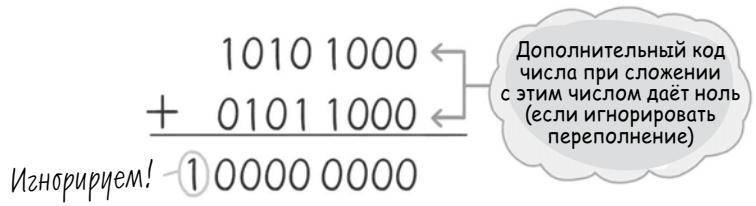
Используя число 8800 — дополнение числа 1200 до 10000, мы получили тот же результат, что и в операции вычитания!

Ого, как смело!

Значит, можно вычитать, складывая с этим дополнительным кодом... Действительно удобно. Но применим ли этот трюк к двоичным числам?

Это не трюк, а красивая теория! Чистая логика!!

Ладно, покажу, как это делается с двоичными числами.



Складываем два двоичных числа, игнорируя переполнение. Если получим 0 во всех разрядах результата, то одно число является дополнительным кодом (или *дополнением до двух**) для другого. Его можно использовать для замены вычитания сложением.

Ясно... Но найти дополнительный код для двоичного числа, наверное, непросто...

Нет, есть простой способ! Достаточно сделать следующее.

* Укоренившийся термин «дополнение до двух» (англ. two's complement) не совсем корректен, так как речь идёт о дополнении до степени двух (до 2^N для N-разрядного двоичного числа). — *Прим. перев.*

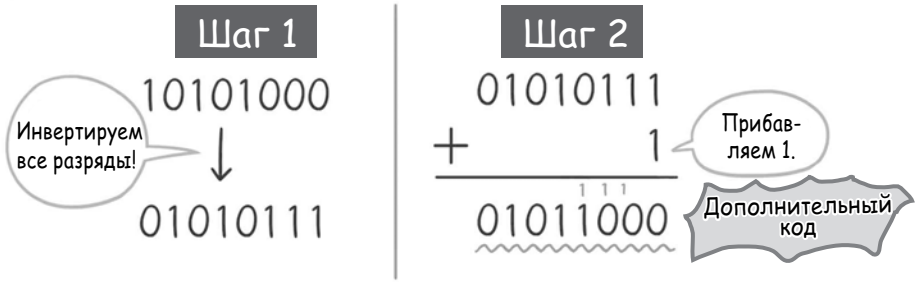
Находим дополнительный код двоичного числа!

Шаг 1 Инвертируем все разряды исходного числа (получаем его *обратный код*, или так называемое *дополнение до единицы*).

Шаг 2 Прибавляем к обратному коду число 1.

Дополнительный код готов – заменяем вычитание сложением!

Попробую ка я сама найти дополнительный код для нашего примера. Вроде бы ничего сложного.

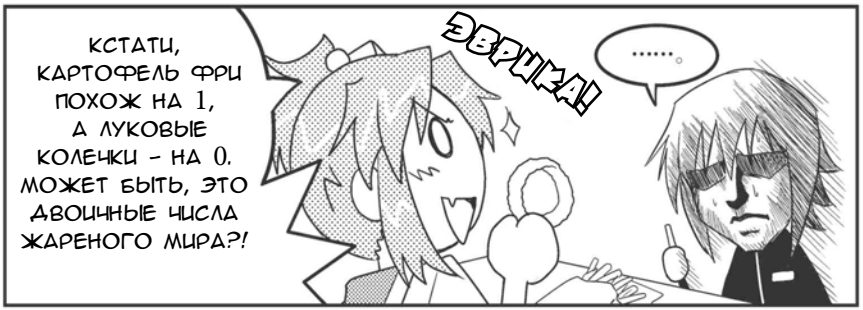


Подобная логика используется и в компьютере — точнее сказать, в АЛУ — для выполнения арифметических операций (сложения, вычитания).

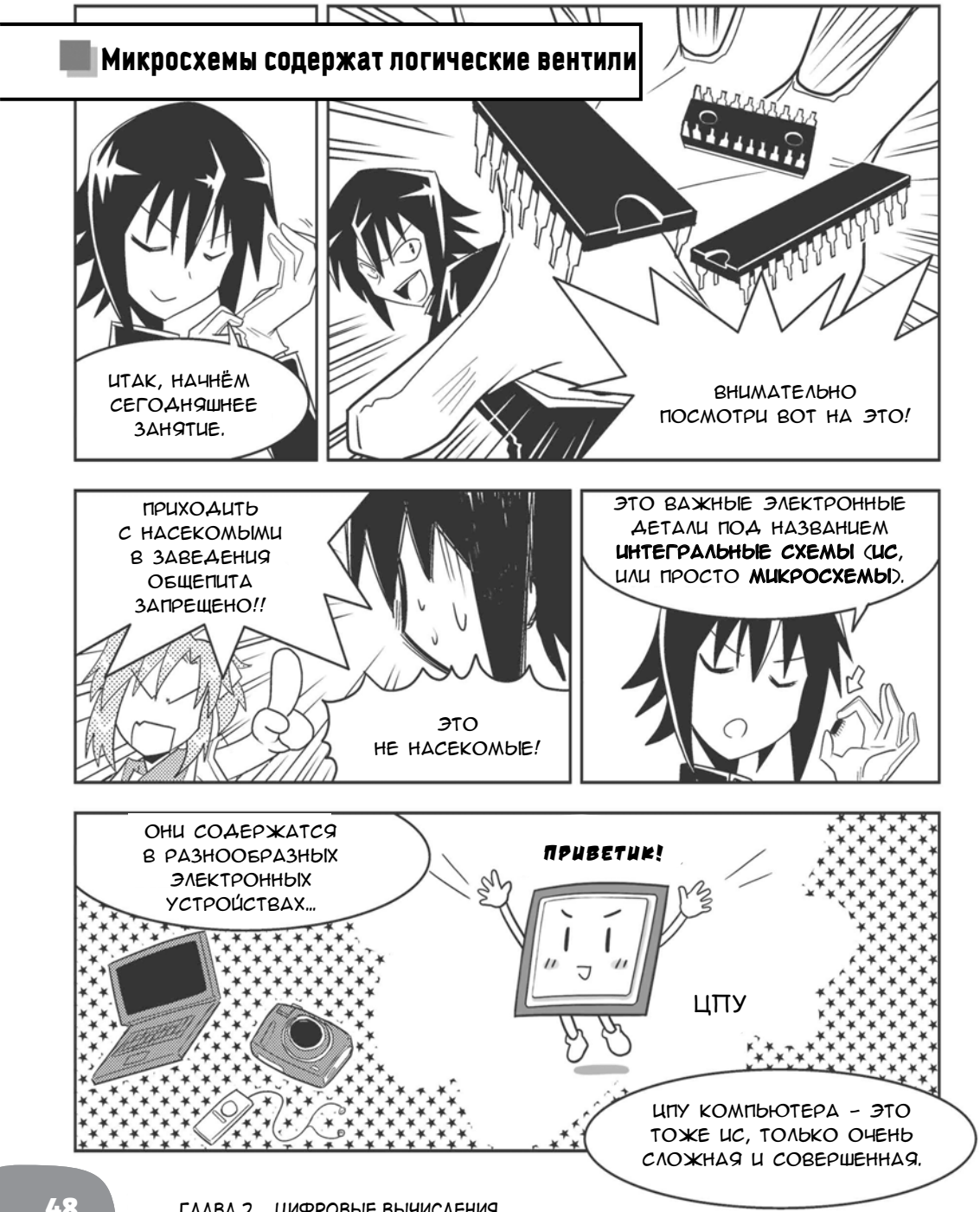
Правда, на самом деле АЛУ сначала выполняет сложение с обратным кодом, а единицу прибавляет уже к результату. То же самое, только в другом порядке, так ведь?

Поскольку компьютер ограничен всего двумя цифрами — 0 и 1, операции очень просты, и выполняет он их исключительно быстро!

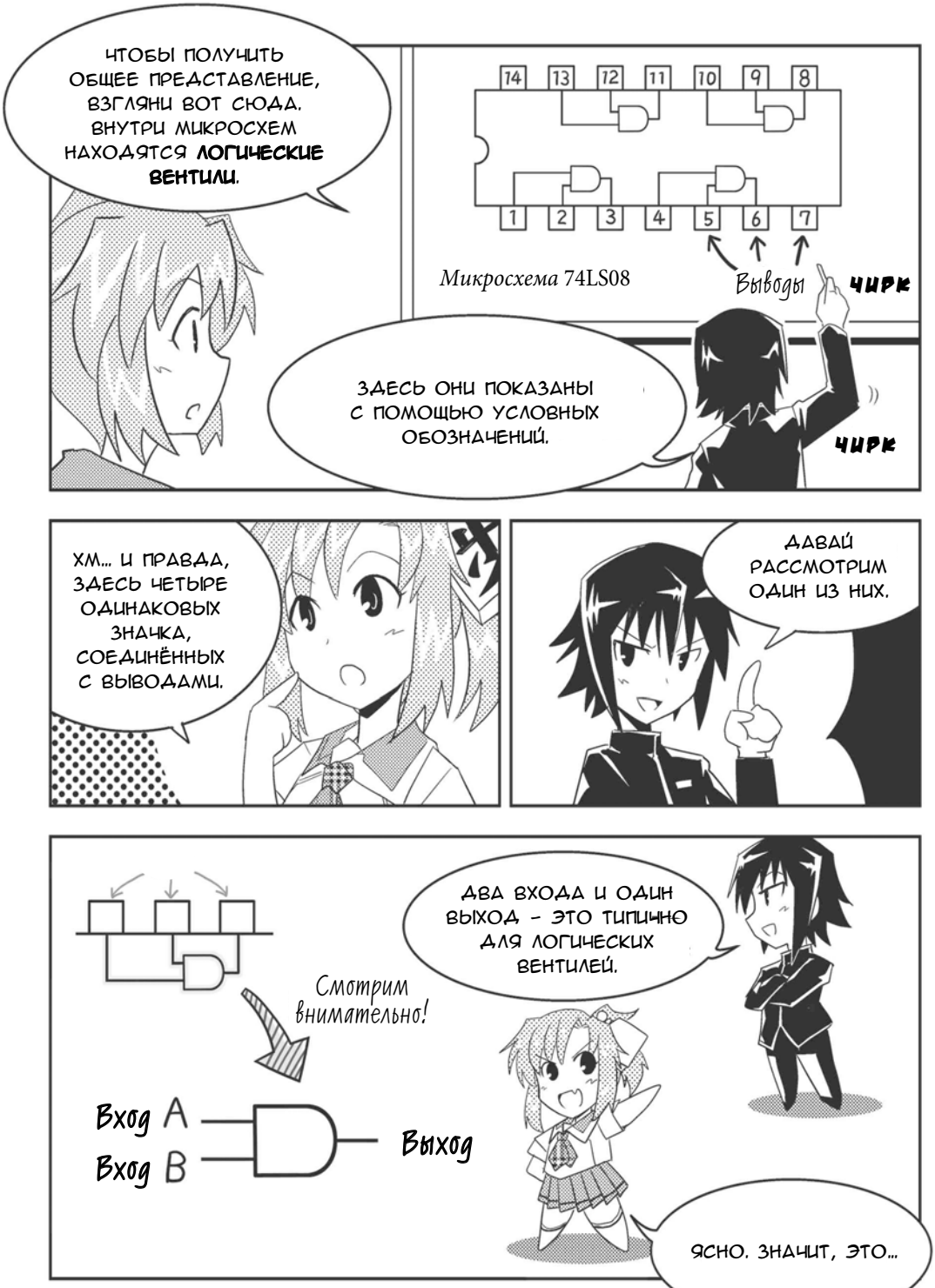
Ясно! Это, похоже, одно из преимуществ двоичных чисел.

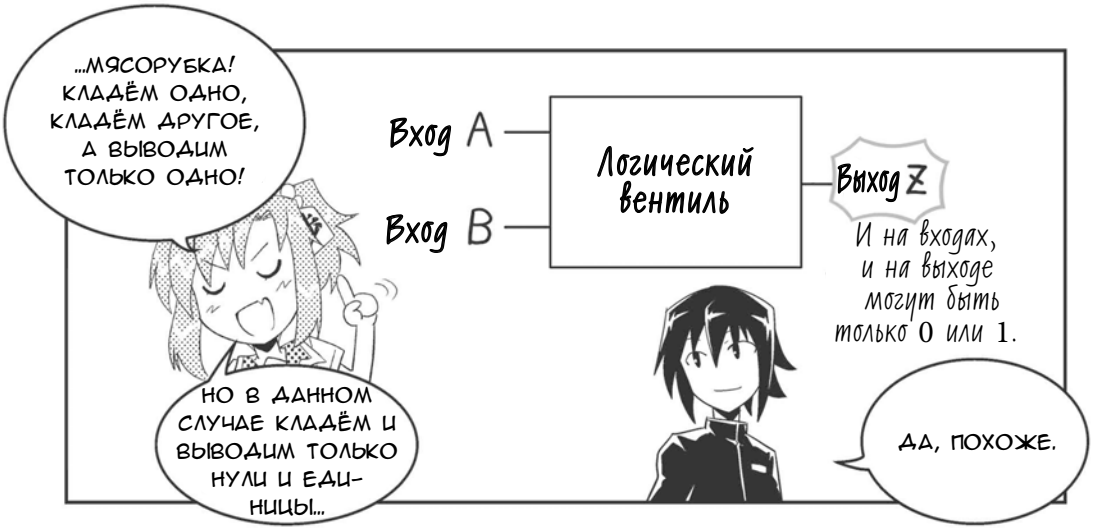


2.2. ЧТО ТАКОЕ ЛОГИЧЕСКИЕ ОПЕРАЦИИ?

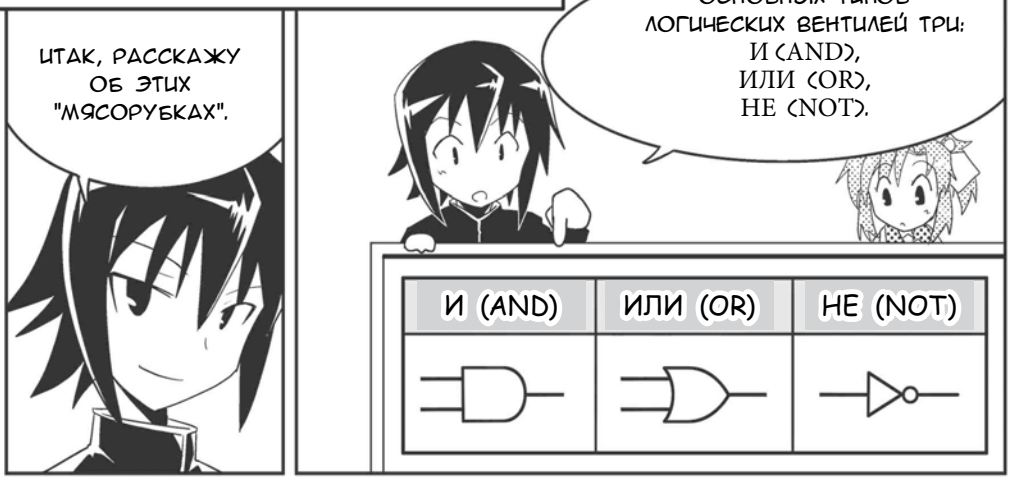



НО ЧТО ЭТО ЗА СЕРЕБРИСТЫЕ НОЖКИ У ЭТИХ НАСЕКОМ... ТО ЕСТЬ МИКРОСХЕМ?
КАЖДЫЙ ВЫВОД МИКРОСХЕМЫ – ЭТО ПУТЬ, ПО КОТОРОМУ ТЕЧЁТ ТОК.
IC
Вывод
1 0110010
ПО НИМ ВХОДЯТ И ВЫХОДЯТ ЦИФРОВЫЕ (ЭЛЕКТРИЧЕСКИЕ) СИГНАЛЫ 0 ИЛИ 1 (НУ ИЛИ ВУ).
ВЫХОДИТ, ОНИ НЕ ПРОСТО ДЛЯ КРАСОТЫ?..
ТЕПЕРЬ О ВАЖНОМ.
Логические операции!
А-А!!
ВНУТРИ МИКРОСХЕМЫ ВЫПОЛНЯЮТСЯ ЛОГИЧЕСКИЕ (ЦИФРОВЫЕ) ОПЕРАЦИИ НАД НУЛЯМИ И ЕДИНИЦАМИ!
ЛОГИЧЕСКИЕ? ЭТО, НАВЕРНОЕ, ПОСЛОЖНЕЙ, ЧЕМ СЛОЖЕНИЕ ИЛИ ТАМ ВЫЧИТАНИЕ...
НО У МЕНЯ ХОРОШЕЕ ЛОГИЧЕСКОЕ МЫШЛЕНИЕ! Я СПРАВЛЮСЬ!
НЕ НАПРЯГАЙСЯ – ЛОГИЧЕСКИЕ ОПЕРАЦИИ НА САМОМ ДЕЛЕ ОЧЕНЬ ПРОСТЫ И ПОНЯТНЫ.
Я СМОГУ!

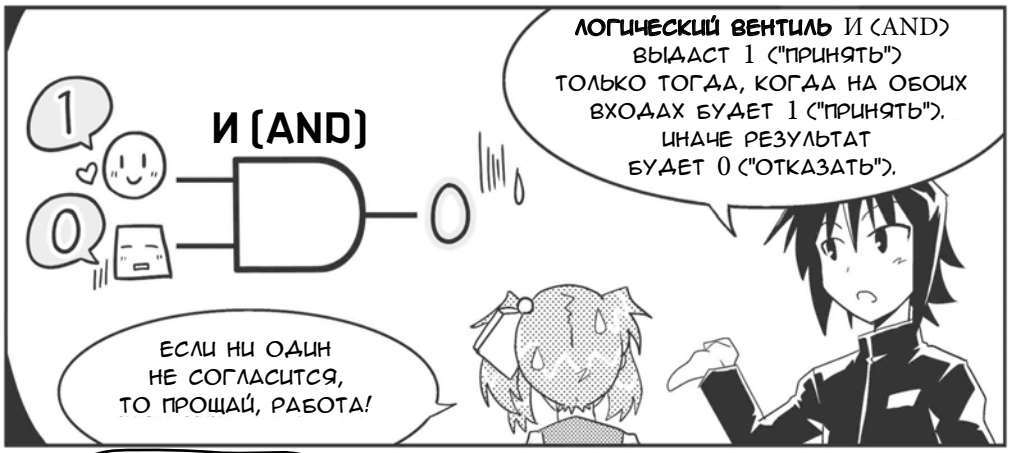

ЧТОБЫ ПОЛУЧИТЬ ОБЩЕЕ ПРЕДСТАВЛЕНИЕ, ВЗГЛЯНИ ВОТ СЮДА. ВНУТРИ МИКРОСХЕМ НАХОДЯТСЯ ЛОГИЧЕСКИЕ ВЕНТИЛИ.
14 13 12 11 10 9 8
1 2 3 4 5 6 7
Микросхема 74LS08
Выводы
ЧИРК
ЗДЕСЬ ОНИ ПОКАЗАНЫ С ПОМОЩЬЮ УСЛОВНЫХ ОБОЗНАЧЕНИЙ.
ЧИРК
ХМ... И ПРАВДА, ЗДЕСЬ ЧЕТЫРЕ ОДИНАКОВЫХ ЗНАЧКА, СОЕДИНЁННЫХ С ВЫВОДАМИ.
ДАВАЙ РАССМОТРИМ ОДИН ИЗ НИХ.
Смотрим внимательно!
Вход A
Вход B
Выход
ДВА ВХОДА И ОДИН ВЫХОД – ЭТО ТИПИЧНО ДЛЯ ЛОГИЧЕСКИХ ВЕНТИЛЕЙ.
ЯСНО. ЗНАЧИТ, ЭТО...



Три основных вентиля (И, ИЛИ, НЕ)




Пусть результат может быть 1 ("Принять") или 0 ("Отказать").
А два входа – это мнения двух сотрудников, проводящих собеседование.
Два входа
1
Принять
0
Отказать
Ой!


И (AND)
1
0
0
Логический вентиль И (AND) выдаст 1 ("Принять") только тогда, когда на обоих входах будет 1 ("Принять"). Иначе результат будет 0 ("Отказать").
Если ни один не согласится, то прощай, работа!

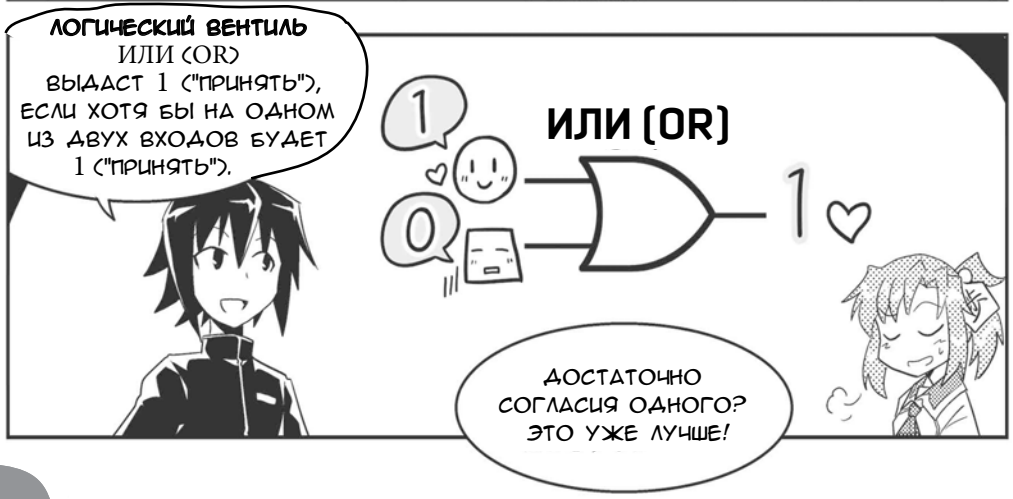

Логический вентиль ИЛИ (OR) выдаст 1 ("Принять"), если хотя бы на одном из двух входов будет 1 ("Принять").
ИЛИ (OR)
1
0
1
Достаточно согласия одного? Это уже лучше!

Таблица истинности, диаграмма Венна


ВОТ!!!
ЗАРУБИ СЕБЕ НА НОСУ!
Таблица истинности логического вентиля И (AND)
A
B
Z
Входы
Выход
A B Z
0 0 0
1 0 0
0 1 0
1 1 1
Нули на входах A и B, ноль на выходе
Единица на входе A, ноль на входе B, ноль на выходе
Ноль на входе A, единица на входе B, ноль на выходе
Единицы на входах A и B, единица на выходе
ТЫК!
ОГО! ВСЕ ВОЗМОЖНЫЕ КОМБИНАЦИИ! КАК УДОБНО!!


A
B
РАЗОБРАТЬСЯ С ЛОГИЧЕСКИМИ ВЕНТИЛЯМИ ПОМОЖЕТ ТАКЖЕ ДИАГРАММА ВЕННА.
КАЖЕТСЯ, МЫ ИЗУЧАЛИ ЭТО В СРЕДНЕЙ ШКОЛЕ.


УГУ... ВАЖНО, ЧТО НА ДИАГРАММЕ ВЕННА ТОЖЕ ПОКАЗАНЫ ДВА СОСТОЯНИЯ.
СЕРЫЙ ЦВЕТ ВНУТРИ ПРЯМОУГОЛЬНИКА ОЗНАЧАЕТ 1 НА ВЫХОДЕ, БЕЛЫЙ – 0.

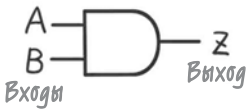
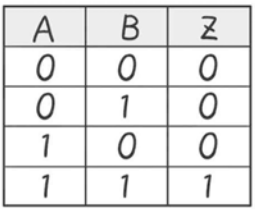
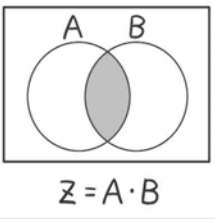

НУЛИ И ЕДИНИЦЫ ВЫРАЖЕНЫ ОЧЕНЬ НАГЛЯДНО!
Закрашена (1) только область, общая для A и B.
A
B
Внутри прямоугольника – мир нулей и единиц!


ТОЧНО. ИТАК, ОПИШУ ТРИ ОСНОВНЫХ ЛОГИЧЕСКИХ ВЕНТИЛЯ ЭТИМИ МЕТОДАМИ.

Логические вентили И (AND), ИЛИ (OR) и НЕ (NOT)

Итак, подытожим всё, что мы узнали про три основных логических вентиля. Вот их условные обозначения, таблицы истинности и диаграммы Венна!

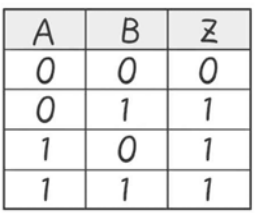
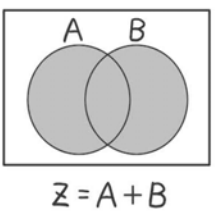
Вентиль И (AND) (схема логического умножения)

Обозначение	Таблица истинности	Диаграмма Венна
		

A	B	Z
0	0	0
0	1	0
1	0	0
1	1	1

Логический вентиль *И (AND)* даёт на выходе Z результат 1 в случае, когда на оба входа A и B поданы значения 1. Его можно записать формулой [Z = A · B]. Операция логического умножения обозначается, например, точкой посередине (·) или знаком ∩.

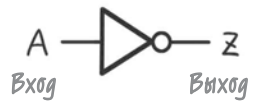
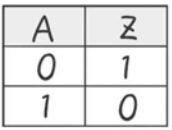
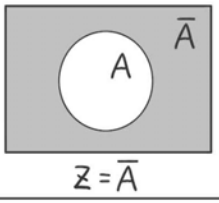
Вентиль ИЛИ (OR) (схема логического сложения)

Обозначение	Таблица истинности	Диаграмма Венна
		

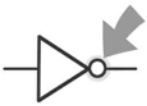
A	B	Z
0	0	0
0	1	1
1	0	1
1	1	1

Логический вентиль *ИЛИ (OR)* даёт на выходе Z результат 1 в случае, когда хотя бы на один из входов A или B подано значение 1. Его можно записать формулой [Z = A + B]. Операция логического сложения обозначается, например, знаком + (плюс) или ∪.

Вентиль НЕ (NOT) (схема логического отрицания)

Обозначение	Таблица истинности	Диаграмма Венна
		

Логический вентиль *НЕ (NOT)* даёт на выходе Z результат, противоположный значению на входе A. Его можно записать формулой $[Z = \overline{A}]$. Операцию логического отрицания обозначают чертой вверху.



Этот белый кружок означает инверсию 0 и 1! (изменение на противоположное значение)

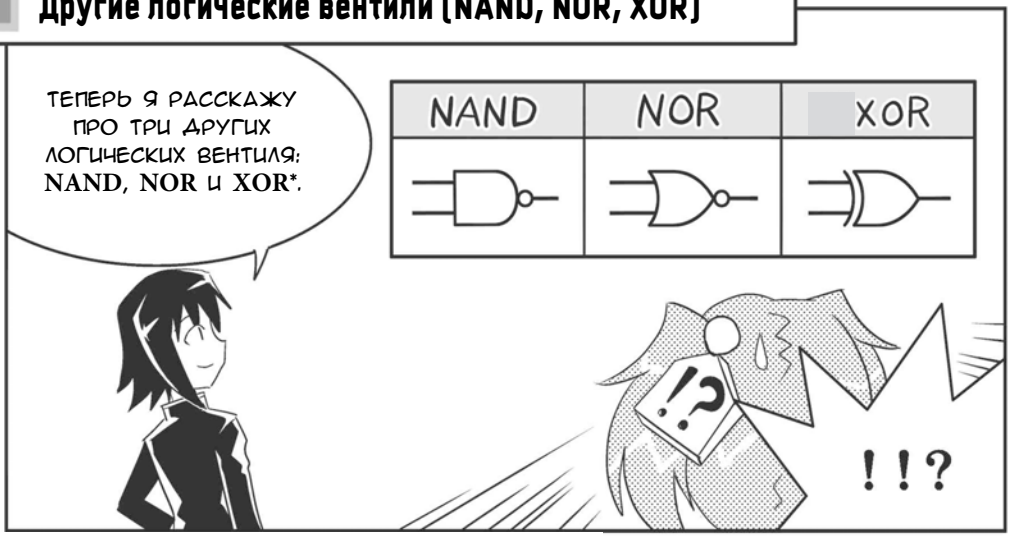
Значит, логические вентили можно обозначать также формулами $[Z = A \cdot B]$, $[Z = A + B]$ и $[Z = \overline{A}]$?! Как много способов!

Да, немало. Здесь нужно кое-что пояснить. Логические вентили И (AND) и ИЛИ (OR) я описал на примере схем с двумя входами: A и B, но на самом деле входов может быть и больше.

В подобных случаях логический вентиль И (AND) даст на выходе результат 1 тогда, когда на *все входы* поданы значения 1. С другой стороны, логический вентиль ИЛИ (OR) даст на выходе результат 1 тогда, когда *хотя бы на один из входов* подано значение 1.

Ну а *сигнальные линии* — это просто провода, по которым передаются цифровые сигналы 0 или 1, так?

Другие логические вентили (NAND, NOR, XOR)

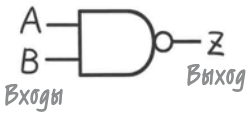
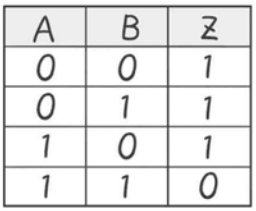
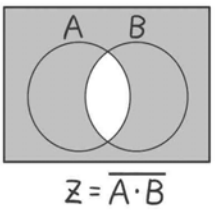


* XOR иногда обозначают EXOR или EOR.

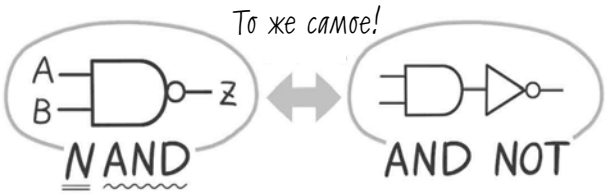
Логические вентили NAND, NOR и XOR

Итак, опишу другие основные логические вентили. Хотя на самом деле это всего лишь комбинации уже известных нам И (AND), ИЛИ (OR) и НЕ (NOT)!

Вентиль NAND (схема логического умножения с отрицанием, элемент Шеффера)

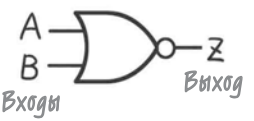
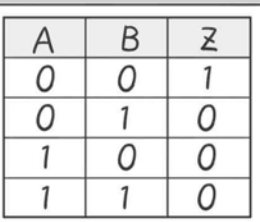
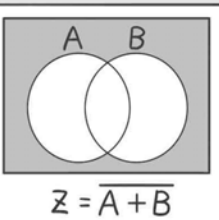
Обозначение	Таблица истинности	Диаграмма Венна
		

A	B	Z
0	0	1
0	1	1
1	0	1
1	1	0

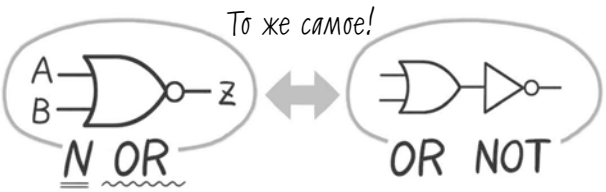


Логический вентиль *NAND (И-НЕ)* составлен из логических вентилей AND и NOT. Результат на выходе NAND получается отрицанием результата на выходе AND. Этот логический вентиль также выражается формулой [$Z = \overline{A \cdot B}$].

Вентиль NOR (схема логического сложения с отрицанием, элемент Пирса)

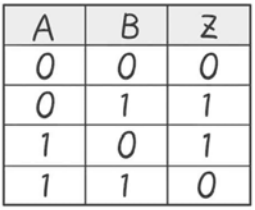
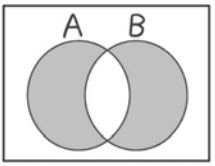
Обозначение	Таблица истинности	Диаграмма Венна
		

A	B	Z
0	0	1
0	1	0
1	0	0
1	1	0

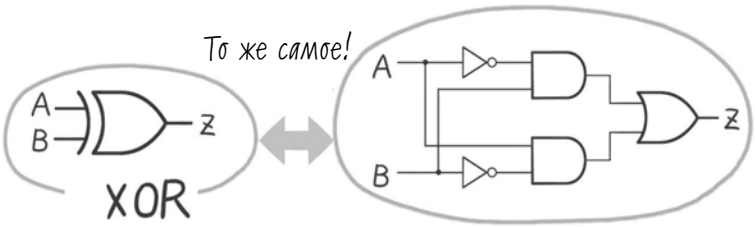


Логический вентиль *NOR (ИЛИ-НЕ)* составлен из логических вентилей OR и NOT. Результат на выходе NOR получается отрицанием результата на выходе OR. Этот логический вентиль также выражается формулой [$Z = \overline{A + B}$].

Вентиль XOR (элемент «исключающее ИЛИ», схема сложения по модулю 2)

Обозначение	Таблица истинности	Диаграмма Венна
		

A	B	Z
0	0	0
0	1	1
1	0	1
1	1	0



Логический вентиль *XOR* (исключающее ИЛИ) даёт на выходе результат 1 тогда, когда на оба входа, A и B, поданы разные значения. Этот логический вентиль записывается формулой [$Z = A \oplus B$].

Он работает точно так же, как приведённая выше схема, составленная из логических вентилей AND, OR и NOT. Кстати, «X» («EX») в его названии — это сокращение от exclusive («исключающий»).

Значит, им можно заменить схему, состоящую из нескольких вентилей? Как полезно! Ты был прав — это стоило изучить.

Законы де Моргана

Отвлекусь от темы, но не ласкают ли твоё ухо такие слова, как «законы», «правила»? Признаюсь, я испытываю приятное волнение каждый раз, когда их слышу! Расскажу-ка я о важных законах!

Это *законы де Моргана**, без которых немыслимы логические операции!

* Их также называют *правилами де Моргана*.

Законы де Моргана

$$\overline{A \cdot B} = \overline{A} + \overline{B}$$

$$\overline{A + B} = \overline{A} \cdot \overline{B}$$

Ну и наелась я, однако... Ничего не поделаешь, слишком уж фастфуд вкусный. ~♪

Не убегай от разговора! Согласен, формулы на первый взгляд кажутся сложными. Но, по сути, это правила замены логического умножения (AND) логическим сложением (OR) или наоборот. Понятно?

И правда! Там, где в левой части стоит точка (знак умножения) (AND), в правой части — «плюс» (OR), и наоборот. Можно вот так нарисовать?

Верно! Это означает, что законы де Моргана можно использовать для построения эквивалентных схем с целью упрощения, например как на следующем рисунке.

И слева и справа — логический вентиль NAND!

И слева и справа — логический вентиль NOR!

Замена схем по законам де Моргана

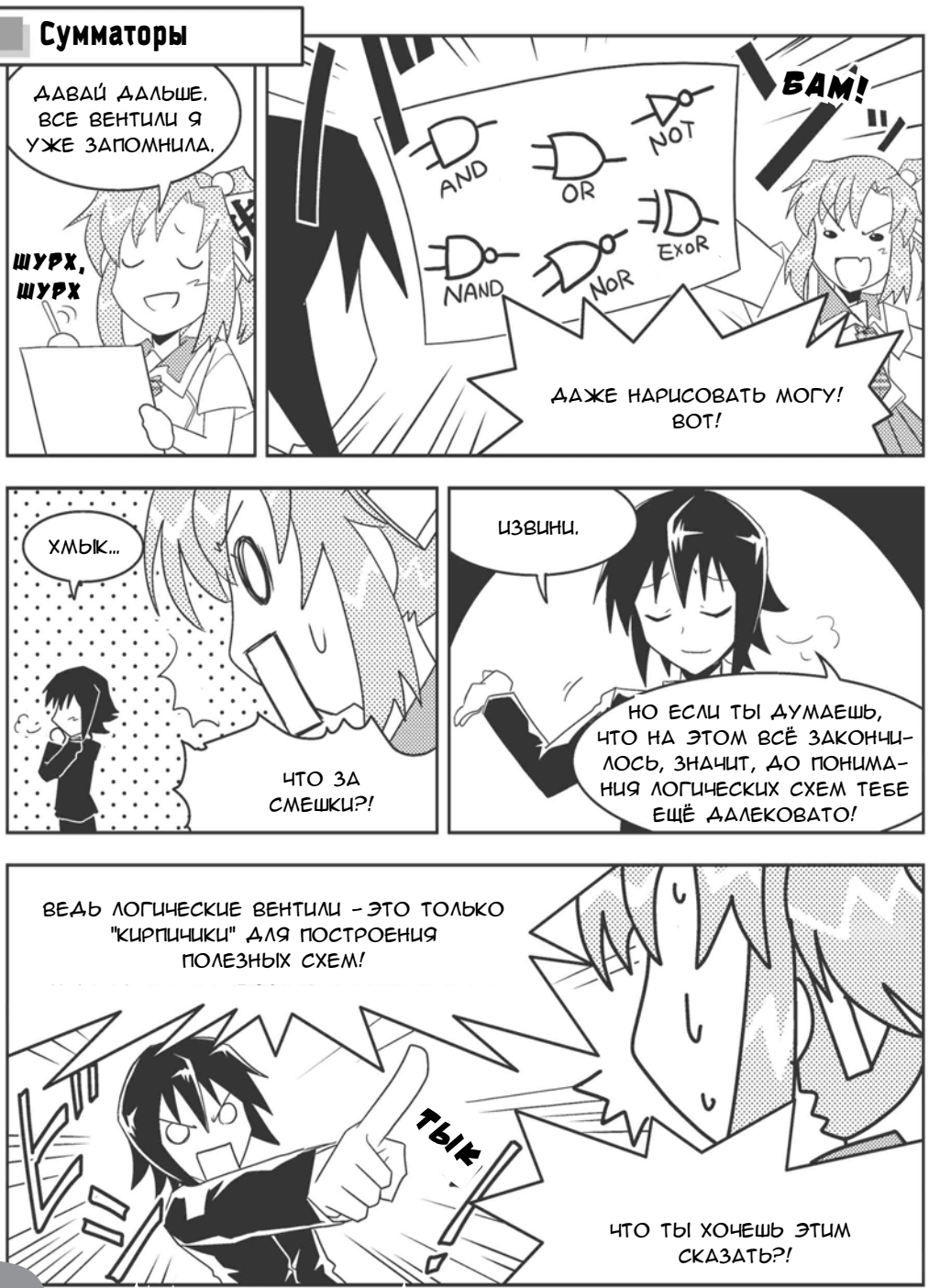
Ого, и правда! Однако схемы справа и слева слишком уж разные... Это не страшно?

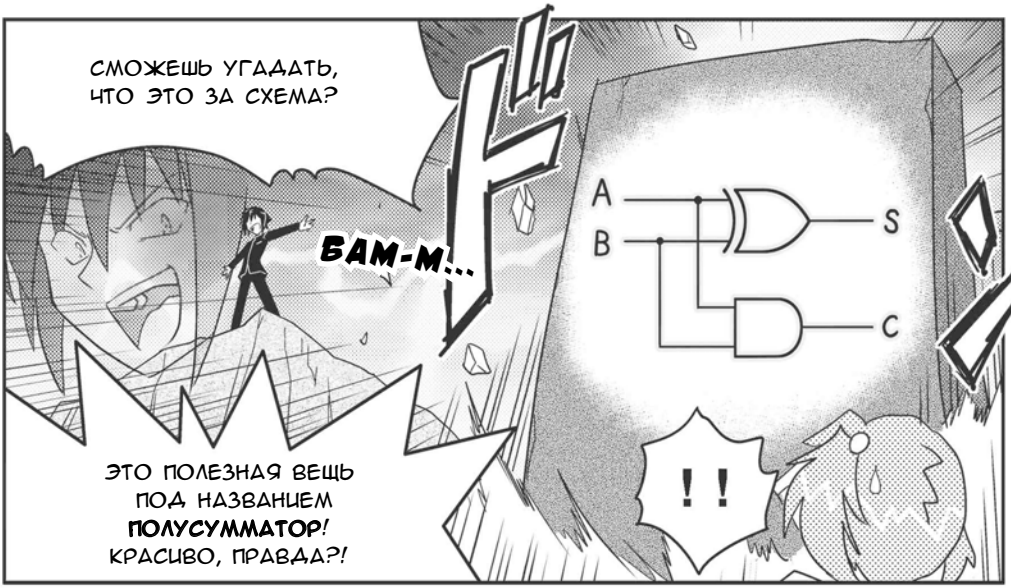
Внешне они, может быть, и отличаются, но в принципе суть одно. Ведь логические (цифровые) схемы — это мир, в котором только нули и единицы, поэтому, перевернув всё вверх тормашками, мы получим по сути то же самое! Законы основаны на этой особенности.

Вот как? Значит, всё можно заменять, как вздумается? Такие законы мне по душе — запомню их обязательно!

2.3. Схемы, выполняющие операции

Сумматоры




СМОЖЕШЬ УГАДАТЬ, ЧТО ЭТО ЗА СХЕМА?
БАМ-М...
A
B
S
C
ЭТО ПОЛЕЗНАЯ ВЕЩЬ ПОД НАЗВАНИЕМ **ПОЛУСУММАТОР**! КРАСИВО, ПРАВДА?!
!!


ЭТА ПРОСТАЯ СХЕМА НУЖНА ДЛЯ СУММИРОВАНИЯ...
...ТО ЕСТЬ АРИФМЕТИЧЕСКОГО СЛОЖЕНИЯ.
ДА-А, КРАСИВО!.. В НЕЙ ИСПОЛЬЗОВАНЫ ЛОГИЧЕСКИЕ ВЕНТИЛИ AND И XOR, ВЕРНО?


НО КАК ОНА МОЖЕТ СКЛАДЫВАТЬ?
НУ ТАК И БЫТЬ, ВЫСЛУШАЮ ТВОЁ ОБЪЯСНЕНИЕ!
А ПОВЕЖЛИВЕЙ НЕЛЬЗЯ?

Полусумматор

Итак, расскажу про полусумматор. Тебе это уже должно быть понятно. Вспомни про сложение однобитных чисел.

0+0=0、0+1=1、1+0=1、1+1=10

Это можно изобразить вот так, в виде таблицы истинности.

Однобитные числа подаются на входы A и B. Кроме того, результат состоит из двух разрядов, поэтому если на выходе, например, 1, то мы пишем 01.

A	B	Выход
0	+ 0	= 00
0	+ 1	= 01
1	+ 0	= 01
1	+ 1	= 10 (Есть перенос в старший разряд)

↑ Младший разряд

Ничего не замечаешь? Посмотри внимательно на часть, выделенную серым!

Если смотреть только на младший разряд результата... Это же таблица истинности логического вентиля XOR (стр. 59)! У него на выходе 1 только тогда, когда на входах A и B разные значения.

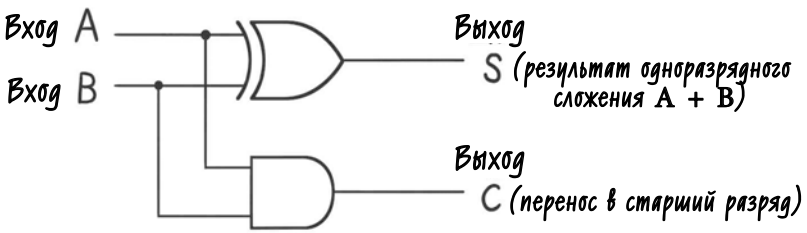
Верно. А теперь посмотри на старший разряд результата.

```
A   B    Выход
0 + 0 =  0 0
0 + 1 =  0 1
1 + 0 =  0 1
1 + 1 =  1 0   (Перенос в старший разряд)
         ↑ Старший разряд
```

Ух ты, это же таблица истинности логического вентиля AND (стр. 55)! У него на выходе 1 только тогда, когда на обоих входах единицы.

Значит... соорудив из логических вентилей XOR и AND схему с двумя выходами для старшего и младшего разрядов, мы сможем складывать однобитные числа!

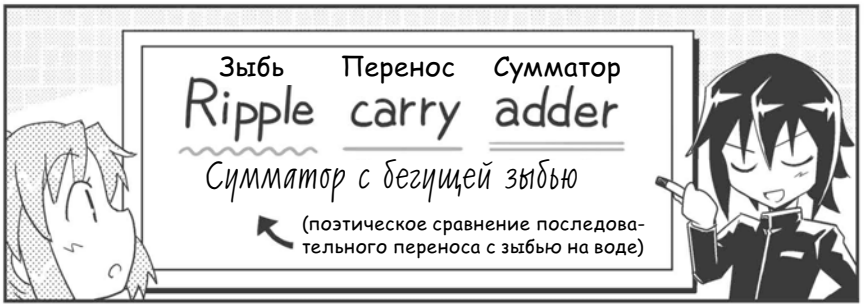
Всё просто, когда улавливаешь суть, не так ли? Для младшего разряда используется *выход S*, а для старшего разряда, то есть для переноса в старший разряд, — *выход C*. Здесь буква S означает sum («сумма»), а C — carry («перенос»).



Полусумматор

Такие схемы с двумя входами и двумя выходами — это полусумматоры. Они позволяют складывать два однобитных числа.

Полный сумматор, сумматор с последовательным переносом



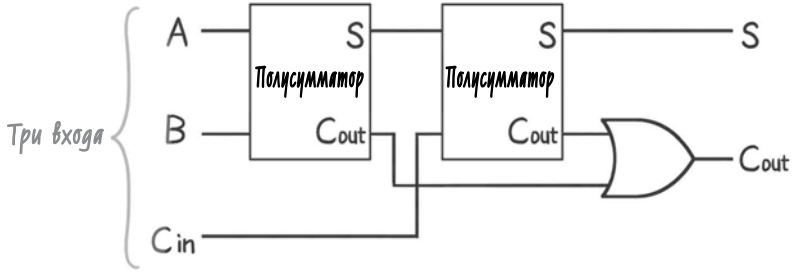
Да, полусумматор оказался очень простым, когда я поняла принцип! Но вот я думаю...

Да, *выход для переноса в старший разряд* есть, но как его использовать дальше? Ведь входа для *переноса из младшего разряда* нет! Значит, с его помощью можно складывать только однобитные числа. Какой-то он недоделанный!

Молодец, верно заметила. Действительно, полусумматор не может принимать перенос из младшего разряда, поэтому способен складывать только однобитные числа. В общем, недоделанный — хоть ругай его, хоть нет!

Зачем мне его ругать?

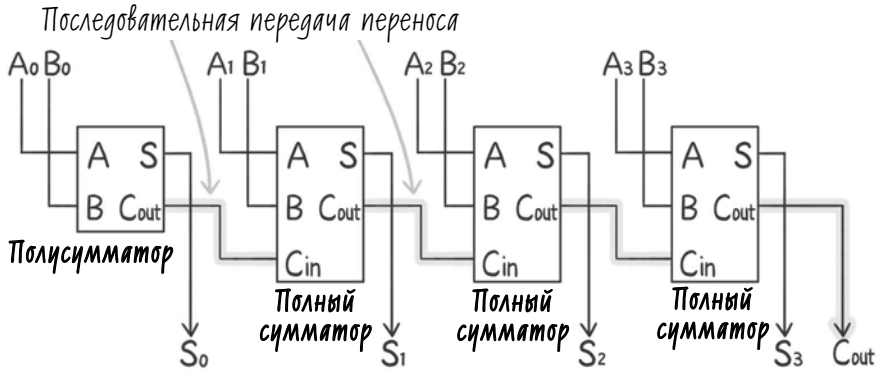
В любом случае недооценивать его нельзя! Ведь из двух таких полусумматоров можно собрать *полный сумматор*, с дополнительным входом для переноса из младшего разряда. Вот тебе рисунок. Здесь полусумматоры для наглядности изображены в виде прямоугольников... Итак, у полного сумматора есть три входа и два выхода.



Полный сумматор

В самом деле, использованы два полусумматора. C_{in} означает вход для переноса, а C_{out} — выход для переноса, так?

Верно! Подключив выходы для переноса в старший разряд ко входам для переноса из младшего разряда, получим *сумматор с последовательным переносом* (ripple carry adder). Он может складывать многоразрядные числа. Вот, например, четырёхбитный.



Сумматор с последовательным переносом

Ясно. Выходы переноса подключены ко входам переноса следующего разряда. Значит, перенос будет распространяться от младших разрядов к старшим.

Сумматоры с последовательным и параллельным переносом

Что-то похожее на этот последовательный перенос, я кажется, уже видела... А, точно! Считая в столбик, мы добавляем единицу к следующему разряду, когда сумма больше 10.

Да. Но есть одна проблема. Если разрядов много, то будет накапливаться *задержка распространения*, и поразрядное сложение с последовательным переносом займёт очень много времени. Более старшим разрядам придётся ждать информации о переносе из более младших.

Задержка распространения переноса по разрядам

Работа сумматора с последовательным переносом

И правда, процесс может затянуться. А ведь сложение приходится выполнять часто... Что же делать?

Хи-хи! Для решения этой проблемы изобрели *сумматор с параллельным переносом* (Carry Lookahead Adder). В нём есть схема ускоренного переноса, одновременно вычисляющая перенос для всех разрядов*.

* В случае вычитания вычисляется заём для всех разрядов.

Работа сумматора с параллельным переносом

Здорово! Старшие разряды тоже не скучают!

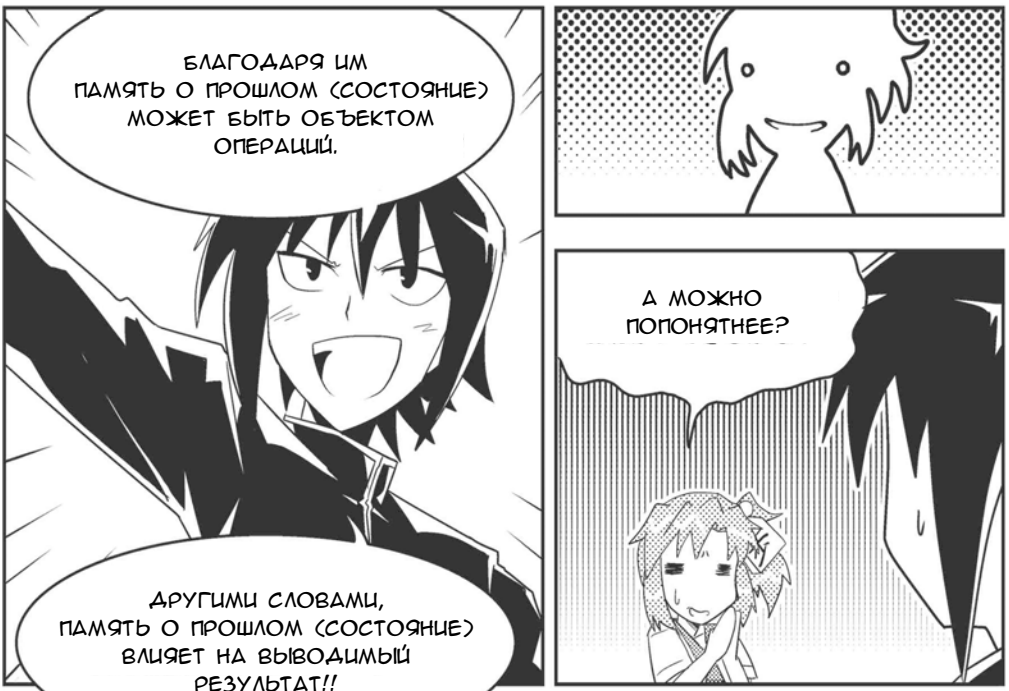
Да. Решение о том, есть ли перенос в соответствующий разряд или нет, принимается для всех разрядов одновременно. Недостаток здесь в том, что требуется дополнительная схема, зато скорость сложения (вычитания) значительно возрастает.

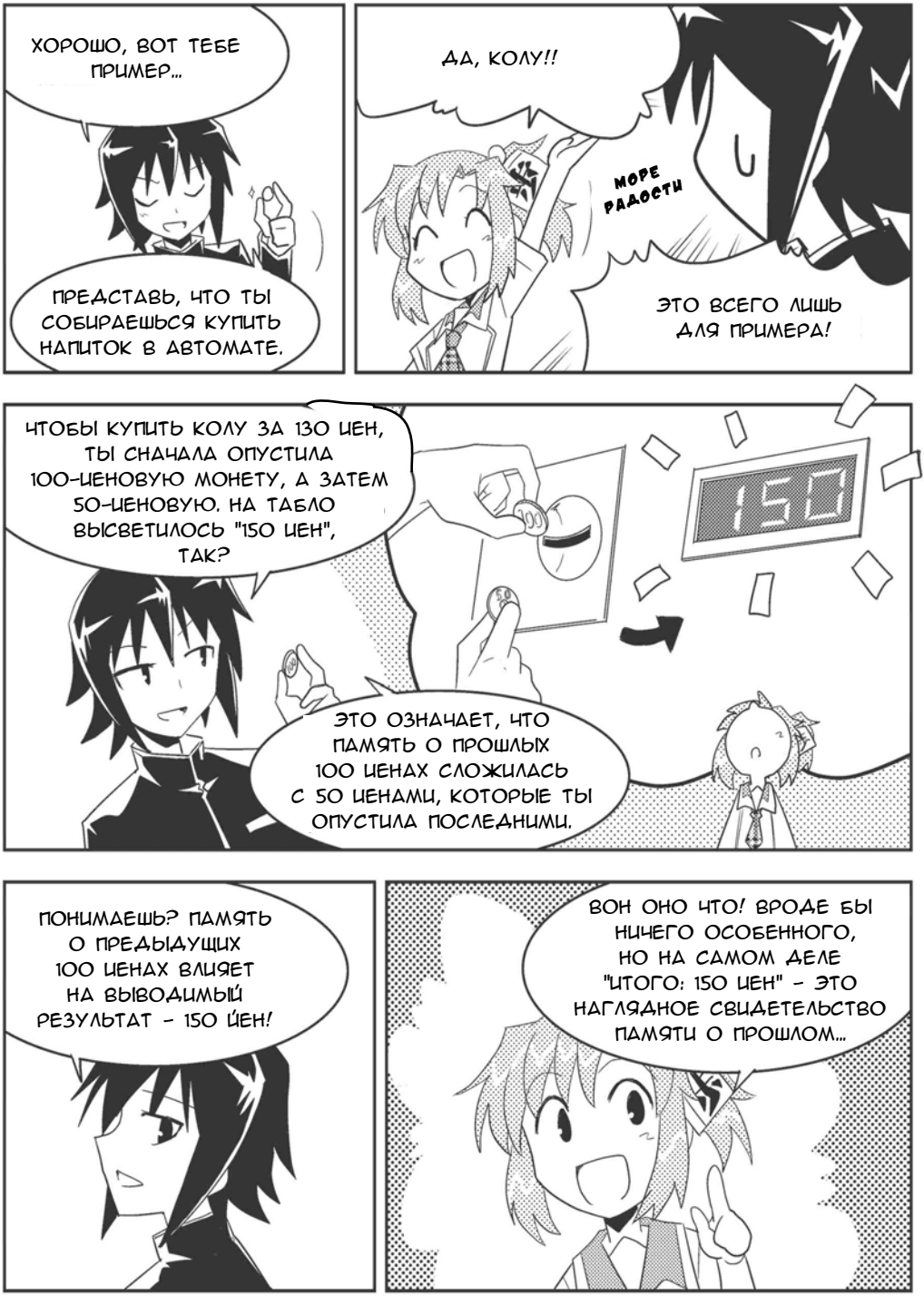
Ого! Ради ускорения вычислений идут на всевозможные хитрости... Я-то думала, что сумматор — это что-то такое медлительное, но оказалось, что реальные сумматоры могут быть очень даже быстрыми.

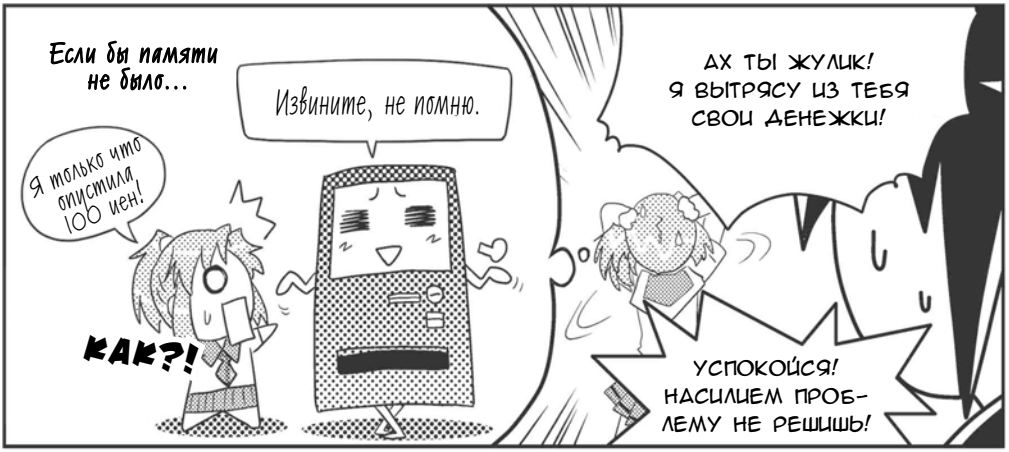
2.4. ЗАПОМИНАЮЩИЕ СХЕМЫ


РЕ-ГИС-ТРЫ? ХМ. ВПЕРВЫЕ СЛЫШУ.
А КАКИЕ ОНИ ИЗ СЕБЯ?
НУ, ЕСЛИ ИСПОЛЬЗОВАТЬ БЫТОВОЕ СРАВНЕНИЕ, РЕГИСТР – ЭТО ЧТО-ТО ВРОДЕ ОТРЫВНОГО БЛОКНОТА.
98
+34
В ПРОЦЕССЕ ВЫЧИСЛЕНИЙ СЮДА ЗАПИСЫВАЮТ ЗНАЧЕНИЯ!


С РЕГИСТРОМ ПРОЩЕ, ЧЕМ С ПАМЯТЬЮ, НО ОН ПРЕДНАЗНАЧЕН ДЛЯ ВРЕМЕННОГО ЗАПОМИНАНИЯ.
ЗАПОМИНАЮЩИЕ УСТРОЙСТВА ТОЖЕ БЫВАЮТ РАЗНЫМИ...


БЛАГОДАРЯ ИМ ПАМЯТЬ О ПРОШЛОМ (СОСТОЯНИЕ) МОЖЕТ БЫТЬ ОБЪЕКТОМ ОПЕРАЦИЙ.
ДРУГИМИ СЛОВАМИ, ПАМЯТЬ О ПРОШЛОМ (СОСТОЯНИЕ) ВЛИЯЕТ НА ВЫВОДИМЫЙ РЕЗУЛЬТАТ!!
А МОЖНО ПОПОНЯТНЕЕ?


ХОРОШО, ВОТ ТЕБЕ ПРИМЕР...
ПРЕДСТАВЬ, ЧТО ТЫ СОБИРАЕШЬСЯ КУПИТЬ НАПИТОК В АВТОМАТЕ.
ДА, КОЛУ!!
МОРЕ РАДОСТИ
ЭТО ВСЕГО ЛИШЬ ДЛЯ ПРИМЕРА!
ЧТОБЫ КУПИТЬ КОЛУ ЗА 130 ИЕН, ТЫ СНАЧАЛА ОПУСТИЛА 100-ИЕНОВУЮ МОНЕТУ, А ЗАТЕМ 50-ИЕНОВУЮ. НА ТАБЛО ВЫСВЕТИЛОСЬ "150 ИЕН", ТАК?
150
ЭТО ОЗНАЧАЕТ, ЧТО ПАМЯТЬ О ПРОШЛЫХ 100 ИЕНАХ СЛОЖИЛАСЬ С 50 ИЕНАМИ, КОТОРЫЕ ТЫ ОПУСТИЛА ПОСЛЕДНИМИ.
ПОНИМАЕШЬ? ПАМЯТЬ О ПРЕДЫДУЩИХ 100 ИЕНАХ ВЛИЯЕТ НА ВЫВОДИМЫЙ РЕЗУЛЬТАТ – 150 ЙЕН!
ВОН ОНО ЧТО! ВРОДЕ БЫ НИЧЕГО ОСОБЕННОГО, НО НА САМОМ ДЕЛЕ "ИТОГО: 150 ИЕН" – ЭТО НАГЛЯДНОЕ СВИДЕТЕЛЬСТВО ПАМЯТИ О ПРОШЛОМ...


Если бы памяти не было...
Извините, не помню.
Я только что опустила 100 иен!
КАК?!
АХ ТЫ ЖУЛИК! Я ВЫТРЯСУ ИЗ ТЕБЯ СВОИ ДЕНЕЖКИ!
УСПОКОЙСЯ! НАСИЛИЕМ ПРОБЛЕМУ НЕ РЕШИШЬ!


Сравниваем память (состояние) настоящего с памятью (состоянием) прошлого.
Пример:
Сегодня продали 3 яблока, а вчера — 2.
3 > 2
Сегодняшняя выручка больше вчерашней.
ЯБЛОКО
Пересчитываем предыдущий результат на основе новых введённых условий.
Вчера общий объём продаж составил 6 яблок, а сегодня продали 3 яблока.
6 + 3 = 9
Общий объём продаж — 9 яблок.
УРА!
ЧТОБЫ ОТ КОМПЬЮТЕРА БЫЛ ТОЛК, КАК ОТ АВТОМАТА ПО ПРОДАЖЕ НАПИТКОВ, ЕМУ ОБЯЗАТЕЛЬНО НУЖНЫ ЗАПОМИНАЮЩИЕ СХЕМЫ.
ВОТ ТАКИХ ИНСТРУКЦИЙ МНОГО В ПРОГРАММАХ (НАРЯДАХ НА РАБОТУ).


ДА, Я ПОНЯЛА, ЧТО ЗАПОМИНАЮЩИЕ СХЕМЫ ОЧЕНЬ ВАЖНЫ.
ПОЙДУ-КА В САМОМ ДЕЛЕ КОЛЫ КУПЛЮ...
НАДО ЖЕ! КАК ОНА ВНУШАЕМА.

Основа запоминающих схем — триггер

Всё-таки трудно представить, как работают эти запоминающие схемы! К тому же я слышала, что механизмы памяти человеческого мозга очень сложны...

С компьютерами дело обстоит проще. Ведь это мир нулей и единиц, так? Другими словами, для компьютера «запомнить» — значит просто сохранить состояние 0 или 1.

Я уже говорил, что 0 и 1 — это разные уровни напряжения (НУ и ВУ); см. стр. 37. Поэтому, чтобы запомнить, например, единицу, нужно сохранять состояние 1, сколько бы времени ни прошло. В таком случае говорят, что произошло *защёлкивание данных* (latch).

Ясно. Но нам ведь не нужно защёлкивать эту единицу навсегда! Через какое-то время может понадобиться нуль запомнить. Надо уметь менять состояние, когда мы захотим, так?

Верно! Представь, что ты нажала на выключатель. Свет зажёгся, и в комнате будет светло до тех пор, пока ты не нажмёшь ещё раз. Тогда свет выключится и опять станет темно, пока ты опять не нажмёшь. Так же должна работать и запоминающая схема. Нам нужно свободно переключать состояния 0 и 1 путём изменения каких-либо условий — так называемых *условий триггера*.

После сохранения значения 0 или 1 в какой-то момент потребуется изменить его на противоположное, то есть инвертировать. Запоминающие схемы обязаны это уметь!

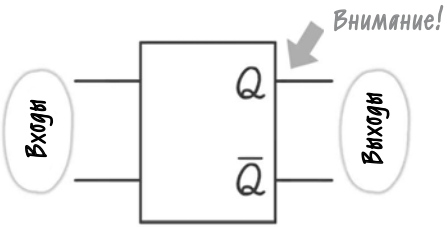
Угу. Но не многого ли мы хотим? И сохранять значение 0 или 1, и инвертировать его, когда нам вздумается...

Да, мы хотим многого, и добиться этого помогает триггерная схема. Это основа запоминающих схем! Кстати, по-английски она называется «флип-флоп» (flip-flop, FF)* — «перекидное устройство».

Флип-флоп... смешно звучит! Но что она делает?

* О происхождении названия см. стр. 80.

Она на самом деле очень нужна! Для понятности я здесь изобразил её в виде прямоугольника. Один такой прямоугольник позволяет запомнить данные длиной в один бит (0 или 1).



Входы никакими буквами не помечены, поэтому что они разные в разных типах триггерных схем.

Так... Есть входы... Есть выходы Q и $\overline{Q}$...

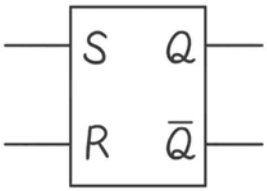
Да. Обрати внимание на выход Q! Именно он всё время находится в состоянии 1 или 0. Кстати, выход $\overline{Q}$... отрицает выход Q (например, когда на выходе Q значение 1, на выходе $\overline{Q}$ будет 0). Этот $\overline{Q}$ удобен на этапе проектирования схемы, а сейчас можешь не обращать не него внимания.

Ясно. А как эта схема устроена? Я имею в виду не прямоугольник, а то, что в реальности!

Главное, не торопись. На самом деле существует несколько типов триггеров, которые устроены и работают по разному. На этот раз я познакомлю тебя с тремя из них: RS-, D- и T-триггерами.

RS-триггер

Так... Начнём с RS-триггера. На прямоугольнике есть обозначения R и S. Это что, Rise & Sushi — рис и суши?!



(Другое название – «RS-защёлка». Иногда буквы R и S в названии меняют местами: «SR-триггер»)

А вот и нет! R — это reset (сброс), а S — set (установка). Особенностью данной схемы является наличие двух входов: сброса и установки.

Суть в том, что если единицу подать на вход S, то на выходе Q будет 1, а если на вход R, то на выходе Q будет 0.

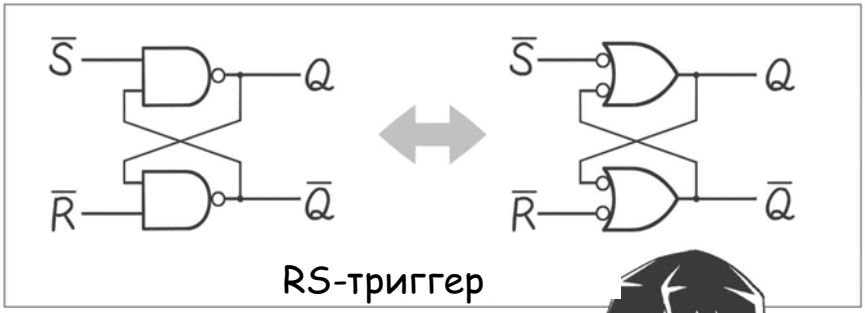
Кроме того, если состояние выхода Q один раз определится, оно уже не будет изменяться, даже если значение на этом входе изменить на ноль. Его можно инвертировать, только подав единицу на другой вход.

Ясно... В общем, состояние триггера зависит от того, на какой из входов была подана последняя единица: если на вход S, то в триггере хранится 1, а если на вход R, то хранится 0.

Теперь немного о сложном. Реальный триггер выглядит так, как показано на следующем рисунке. По законам де Моргана (см. стр. 60) его можно выразить двумя способами: логическими вентилями NAND или NOR.

Ой, какая странная форма! Два вентиля NAND (или NOR) как будто образуют «восьмёрку».

Ага! Выход каждого из этих вентилей соединён с одним из входов другого.



Входы		Выходы		Режим работы
$\overline{S}$	$\overline{R}$	Q	$\overline{Q}$	
1	1	Без изменений		Состояние выхода не меняется
0	1	1	0	Установка
1	0	0	1	Сброс
0	0	1	1	Запрещено

Эта «восьмёрка» и есть та часть, которая запоминает, то есть «защёлкивает» 1 или 0. Её можно назвать главной особенностью запоминающих схем.

Запутанная схемка, однако! Только таблица истинности поможет понять, как она работает.

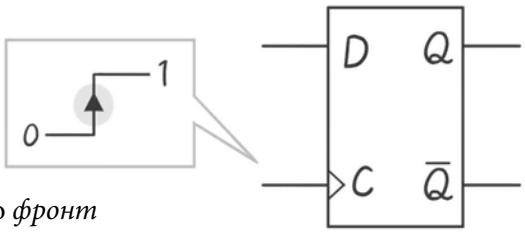
Так... «*Без изменений*» — это когда поддерживается текущее состояние выхода: Q = 1 или Q = 0. Но что значит «*Запрещено*» в самом низу?

А, это? При установке или сбросе нельзя допускать, чтобы на обоих входах одновременно был НУ (0), иначе на обоих выходах Q и $\overline{Q}$ будет ВУ(1) до тех пор, пока на один из входов не будет подан ВУ. Но по логике работы схемы на выходах должны быть противоположные значения. Чтобы избежать такой неприятной ситуации...

Понятно. Разработали что-то вроде правил дорожного движения для схемы.

D-триггер, тактовый сигнал

Итак, у нас на очереди D-триггер! Вход обозначен буквой D. Но что такое этот треугольничек с буквой C?

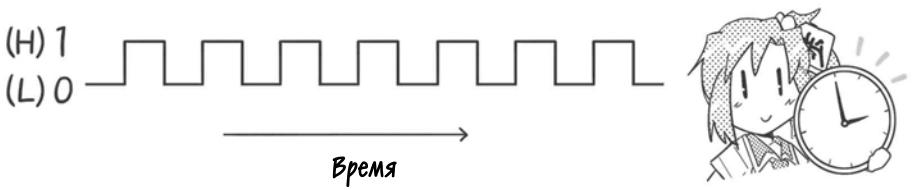


Это *фронт* (изменение уровня сигнала)

Буква D означает data — данные, а треугольничком обозначен *передний фронт*. Буква C означает clock — *тактовый сигнал*.

Передний фронт? Непонятно. И что такое тактовый сигнал? По английски clock — это, кажется, «часы»...

Да! На самом деле внутри компьютера постоянно присутствует периодический цифровой сигнал, необходимый для синхронизации работы схем. Это и есть тактовый сигнал! Как часы отсчитывают время, так и он периодически изменяется между НУ и ВУ (нулём и единицей). Эти «часы» работают независимо от того, что у нас там на входах или на выходах схем.



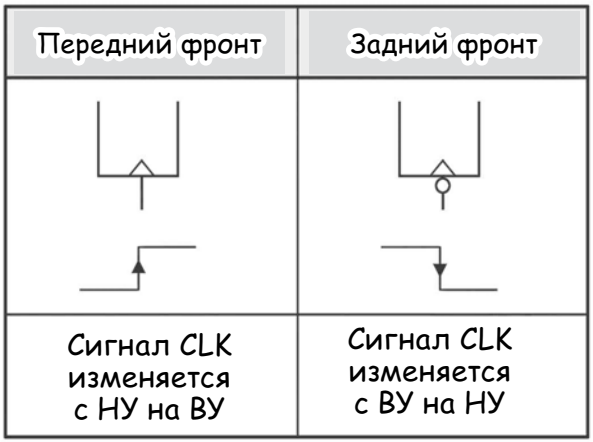
Тактовый сигнал

Ясно. Он тикает, как часы. Как мы проводим день по часам, так и для работы схем они нужны.

Да. Тактовый сигнал может требоваться для того, чтобы схема заработала. Кроме того, «сигналом к началу работы» служит часть тактового сигнала, которая называется *передним фронтом*. Посмотри на этот рисунок!

Ого! Этот тактовый сигнал весь испещрён стрелочками.

Кстати, именение тактового сигнала с НУ на ВУ (с 0 на 1) — это *передний фронт*, а с ВУ на НУ (с 1 на 0) — *задний фронт*.

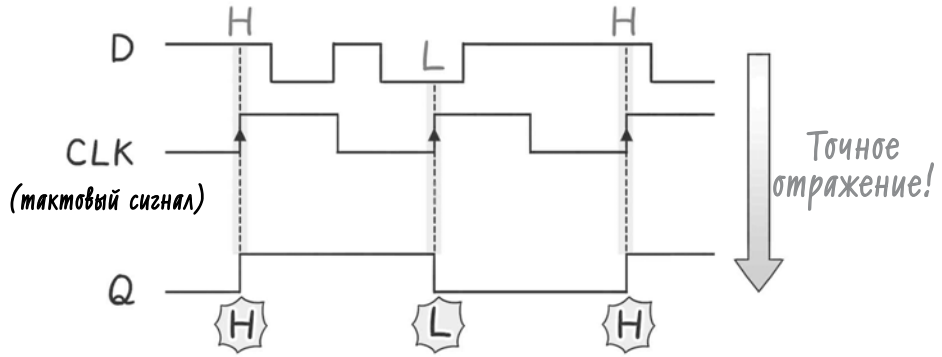


Ха-ха, наконец-то я поняла! Передний или задний фронт —это как звонок, по которому выполняется какое-либо действие. Ну, например, урок в школе начинается или, наоборот, заканчивается.

Точно! Ты привела замечательный пример!

Итак, вернёмся к нашему разговору. В моменты прихода переднего фронта тактового сигнала значение на выходе Q становится таким же, как на входе D.

Это легко понять при взгляде на эту временную диаграмму. *Временные диаграммы* показывают изменение каждого сигнала во времени.



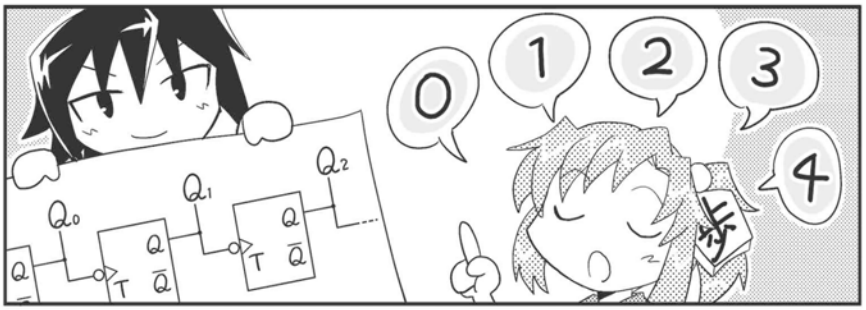
Без переднего фронта вход D никак не влияет на выход Q!

Ясно. Временная диаграмма всё прекрасно объясняет. В общем, особенность D-триггера в том, что он работает синхронно с передним фронтом тактового сигнала.

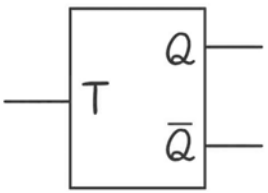
Хм... Схемы, как и современные люди, не могут жить без часов!

Почему по-английски триггеры называют «флип-флоп»?
Слово flip-flop имитирует шлепки, резкие звуки во время ходьбы в сандалиях или при перебросе планки качелей. В запоминающих схемах тоже происходит «переброс» (инверсия) значений. Может быть, потому триггеры так и назвали.

T-триггер, счётчик

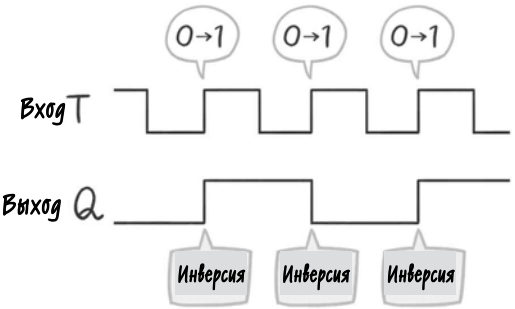


Последним у нас T-триггер... Как, всего один вход?! Это, случаем, не ошибка?



Ха-ха-ха! Ты за кого меня принимаешь?!

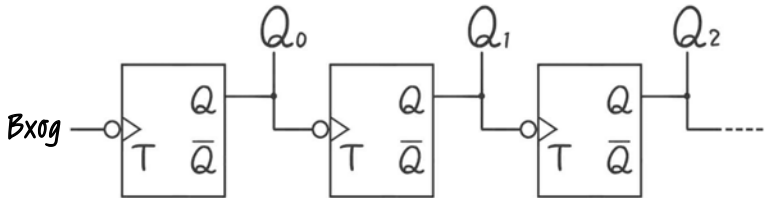
T-триггер имеет всего один вход и очень простую конструкцию. При каждом изменении уровня сигнала на входе T (в зависимости от конкретного триггера — с 0 на 1 или с 1 на 0) значение на выходе Q инвертируется. Вот временная диаграмма.



Передние фронты (есть также типы, реагирующие на задние фронты)

Да, он действительно прост и понятен! Настоящая запоминающая схема, хоть и с одним входом.

Кстати, инверсию 0 и 1 называют по английски toggle («тумблер», «переключатель с двумя состояниями»). На самом деле буква T в названии идёт отсюда! Далее, если T-триггеры соединить в цепочку, как показано внизу, то получится схема счётчика*.



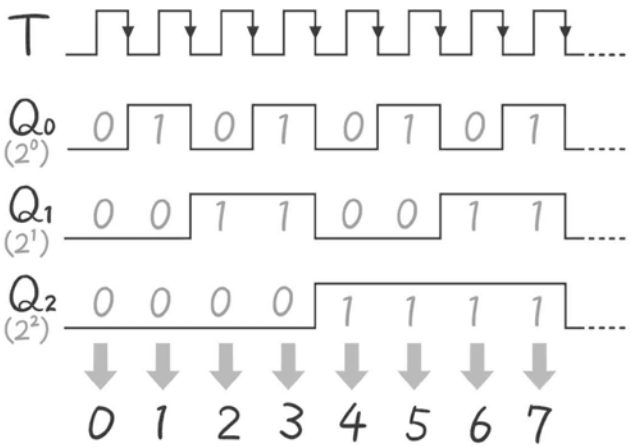
Эта схема состоит из T-триггеров, работающих по заднему фронту, и выполняет функцию прямого счёта, как видно из нижеприведённой временной диаграммы.

Схема счётчика

* В этой схеме при смене уровня сигнала на входе с ВУ на НУ происходит инверсия выхода первого триггера, которая затем распространяется слева направо по каскадам счётчика с временной задержкой. Эту схему называют асинхронным счётчиком. Однако в обычных логических схемах, в том числе и в ЦПУ, используются синхронные счётчики, в которых выходы каждого из каскадов изменяются одновременно с изменением сигнала на входе.

Хм... И как же он считает?

Вернись к временной диаграмме T-триггера на предыдущей странице. Инверсия на выходе происходит в два раза реже, чем на входе, так? Поэтому если соединить в цепочку много T-триггеров, то получится временная диаграмма, приведённая ниже.



Если записать значения (0 или 1) на каждом из выходов по порядку слева направо Q_2, Q_1, Q_0, мы получим двоичное число, которое будет увеличиваться на 1 при каждом изменении входа (в данном случае — при появлении на нём заднего фронта сигнала). Интересно, не правда ли?

Ого, и правда! Q_2 — это разряд двоичного числа с весовым коэффициентом $2^м$ =4, Q_1 — разряд с весовым коэффициентом $2^1 = 2$, Q_0 — разряд с весовым коэффициентом $2^0 = 1$.

Сначала двоичное число равно 000 (0), затем 001 (1), затем 010 (2), затем 011 (3) и так далее. Эта конструкция действительно позволяет выполнять прямой счёт!

Да. Например, если в схеме три прямоугольника (Т-триггера), то есть она может выразить восемь ($2^3 = 8$) чисел, то с её помощью можно считать от 0 до 7.

Кстати, счётчик можно сделать и из триггеров других типов, например D-триггеров. Кроме того, проявив изобретательность, можно сделать также и схему, выполняющую обратный счёт.

Та-ак! *Прямой счёт* — это 0, 1, 2, 3... *Обратный счёт* — это ...3, 2, 1, 0. Да, может пригодиться.

Ладно, закончим на этом с триггерами. Как я уже говорил вначале, триггеры — это основа запоминающих схем. Например, и память компьютера (ОЗУ), и регистры внутри процессора тоже построены на основе триггеров. И, как ты только что увидела, они используются даже в счётчиках.

Ха-ха! Триггеры — основа различных устройств. И по-английски их название так мило звучит!


СПАСИБО ЗА СЕГОДНЯШНИЙ УРОК! Я УЗНАЛА МНОГО НОВОГО!
ХМ... СЕГОДНЯ Я ДАЛ ТЕБЕ ТОЛЬКО НАЧАЛЬНЫЕ ЗНАНИЯ.
ЗАПОМНИ ВСЁ ЭТО ХОРОШЕНЬКО.
НЕ БЕСПОКОЙСЯ. Я ЖЕ ТАК УМНА, И У МЕНЯ УДИВИТЕЛЬНАЯ ПАМЯТЬ!
УДИВИТЕЛЬНАЯ ПАМЯТЬ...
А МОЖЕШЬ ВСПОМНИТЬ ВСЕХ, С КЕМ ТЫ ИГРАЛА В СЁГИ?
Э-Э... ВИДИШЬ ЛИ...
САМУРАЙ НЕ ОБЯЗАН ПОМНИТЬ ВСЕХ НЕГОДЯЕВ, КОТОРЫХ ОН ЗАВАЛИЛ.
НУ ВОТ, ПОШЛИ ОТГОВОРКИ...
3F 終日 禁
HAMBURG
ДА, НЕ ПОМНЮ, И С ЭТИМ НИЧЕГО НЕ ПОДЕЛАЕШЬ!

ДОПОЛНИТЕЛЬНАЯ ИНФОРМАЦИЯ

◆ Современные методы проектирования схем (CAD, FPGA)

Для проектирования современных интегральных микросхем, включая ЦПУ, используются методы, напоминающие разработку компьютерных программ. Они позволяют воспроизвести схему с помощью так называемого *языка описания аппаратуры* — HDL (Hardware Description Language). Раньше использовали описанные в этой главе обозначения логических вентилей, но в последнее время они практически вышли из употребления благодаря развитию систем автоматизированного проектирования (САПР, англ. CAD: Computer Aided Design). Сейчас эти обозначения используют разве что для проектирования очень простых схем.

Несмотря на это, знания обозначений логических вентилей помогут вам понять, например, принципы работы системы в целом или отдельных её функций.

На заре развития ЦПУ в ходу были интегральные схемы, содержащие всего по нескольку логических элементов, таких как AND, OR, NOT и т. п. С их помощью создавали испытательные модели и системы оценки параметров, необходимые для проектирования интегральных схем с более сложными функциями, например микросхем ЦПУ. На этих моделях проверяли правильность работы схем, соответствие требуемым характеристикам. При наличии отклонений измеряли сигналы в линиях, соединяющих логические вентили, и вносили необходимые изменения в конструкцию.

Однако сейчас такие электронные детали с функциями отдельных логических вентилей на стадии разработки практически не используются — их место заняла интегральная схема под названием *программируемая пользователем вентильная матрица*, или *программируемая логическая матрица* (ПЛМ, англ. FPGA: Field Programmable Gate Array).

FPGA (ПЛМ) — это очень удобная микросхема, которую можно изменять с помощью компьютерных программ.

Внутри ПЛМ находится множество встроенных логических блоков (вентилей), в каждом из которых содержится *таблица перекодировки* (look-up table) — настраиваемая таблица истинности, задающая поведение вентиля. В продаже имеются ПЛМ с различным количеством таблиц перекодировки: от нескольких сотен до миллиона и более. ПЛМ также позволяют программно задавать линии связи микросхемы с внешними устройствами.

Всё это позволяет использовать ПЛМ такой же степени интеграции в качестве принципиально разных микросхем для всевозможных систем.

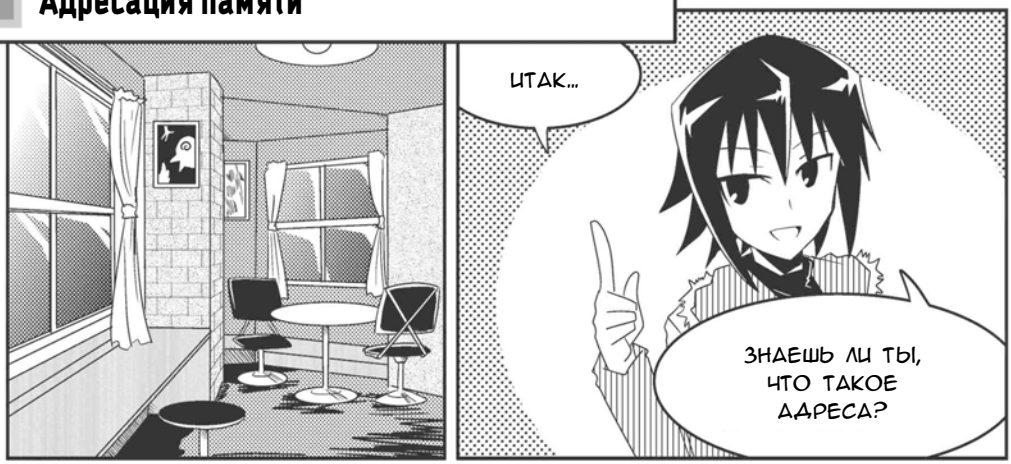
Разумеется, имеются и ПЛМ, содержащие функциональные блоки ЦПУ. Хотя с точки зрения снижения себестоимости, конечно, выгоднее спроектировать специальную микросхему, выполняющую требуемые функции, и запустить её в массовое производство, в последнее время цены на ПЛМ тоже снизились. С другой стороны, разработка специальной микросхемы требует времени, поэтому в случае, если не планируется выпускать её в течение достаточно длительного срока в количестве не менее нескольких сотен тысяч штук, бывает выгоднее использовать вместо неё ПЛМ.

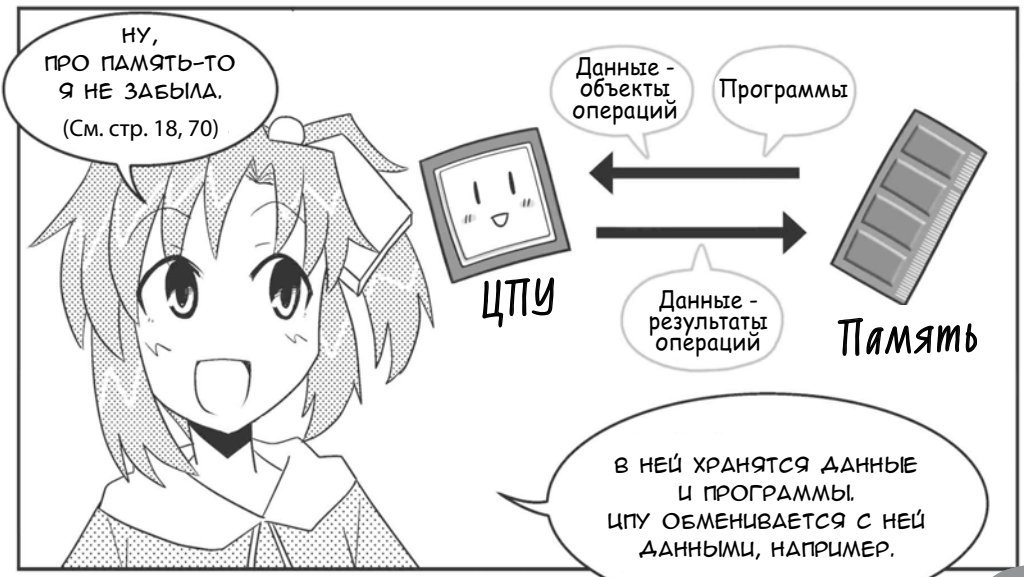
ГЛАВА 3

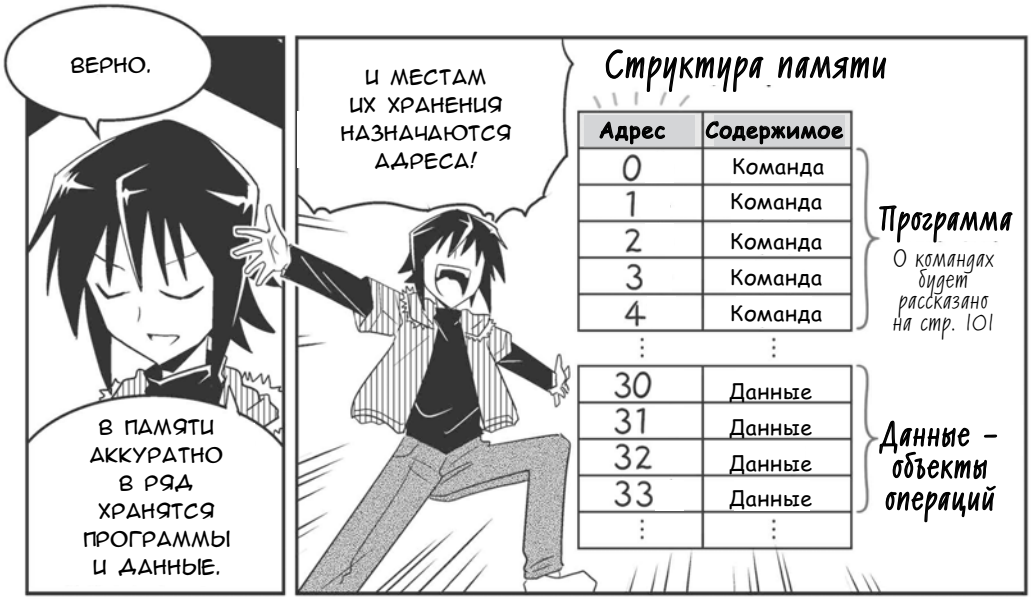
УСТРОЙСТВО ЦПУ

3.1. РАЗЛИЧНЫЕ СВЕДЕНИЯ ПРО ПАМЯТЬ И ЦПУ

Адресация памяти






ВЕРНО.
В ПАМЯТИ АККУРАТНО В РЯД ХРАНЯТСЯ ПРОГРАММЫ И ДАННЫЕ.
И МЕСТАМ ИХ ХРАНЕНИЯ НАЗНАЧАЮТСЯ АДРЕСА!
Структура памяти
Адрес
Содержимое
0 Команда
1 Команда
2 Команда
3 Команда
4 Команда
30 Данные
31 Данные
32 Данные
33 Данные
Программа
О командах будет рассказано на стр. 101
Данные – объекты операций


И ПРАВДА! ВСЁ АККУРАТНО ЛЕЖИТ В СВОИХ ЯЧЕЙКАХ, ПОМЕЧЕННЫХ АДРЕСАМИ!
ЭТА ОБЛАСТЬ, НАХОДЯЩАЯСЯ ПОД НЕПОСРЕДСТВЕННЫМ УПРАВЛЕНИЕМ ЦПУ...
...НАЗЫВАЕТСЯ ПРОСТРАНСТВОМ АДРЕСОВ (ПРОСТРАНСТВОМ ПАМЯТИ)! ЗАПОМНИ ХОРОШЕНЬКО!


Адрес
Содержимое
0 Команда
1 Команда
2 Команда
3 Команда
4 Команда
30 Данные
31 Данные
32 Данные
33 Данные
Пространство адресов (памяти)
Подробно об этом будет рассказано на стр. 119
ХМ... ПОД НЕПОСРЕДСТВЕННЫМ УПРАВЛЕНИЕМ...
ТО ЕСТЬ ЦПУ МОЖЕТ И ВЫЗЫВАТЬ ДАННЫЕ ОТТУДА, И ЗАПИСЫВАТЬ ИХ КАК ВЗДУМАЕТСЯ?


Я ПРЕДСТАВЛЯЮ ЭТО ТАК: ЦПУ С НАГЛЫМ ЛИЦОМ УЗУРПАТОРА!!!
ЦПУ
А-ХА-ХА-А!
Кручу как хочу...
Пространство адресов
(пространство памяти)
Адрес
Содержимое
0
1
2
3
4
30
31
32
33
Команда
Данные
БЕДНОЕ ПРОСТРАНСТВО...


НО ЗАЧЕМ НУЖНЫ ЭТИ АДРЕСА?
С НОМЕРКАМИ ЧТО, УДОБНЕЕ?

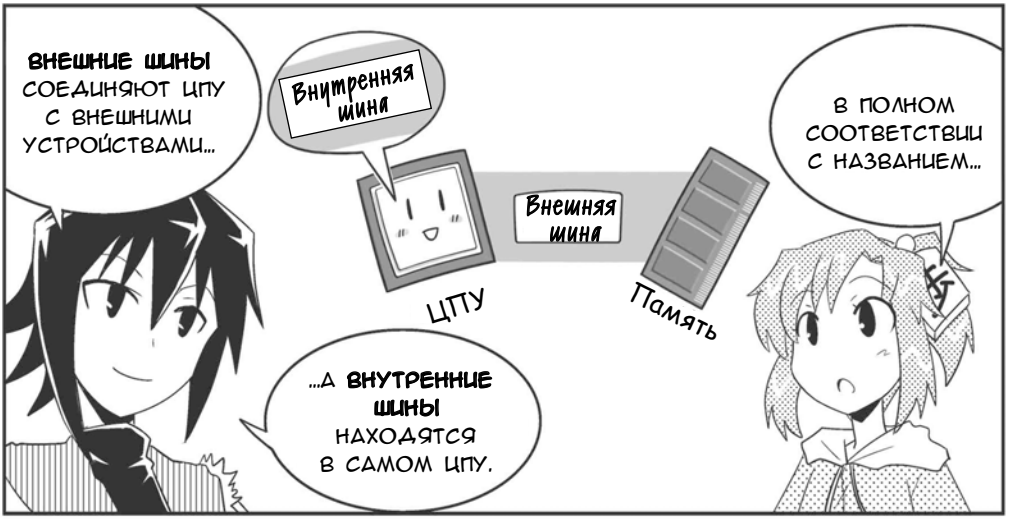

ХМ... НЕУЖЕЛИ НЕПОНЯТНО?
К ДАННЫМ МОЖНО ОБРАЩАТЬСЯ ПО ИХ НОМЕРАМ.

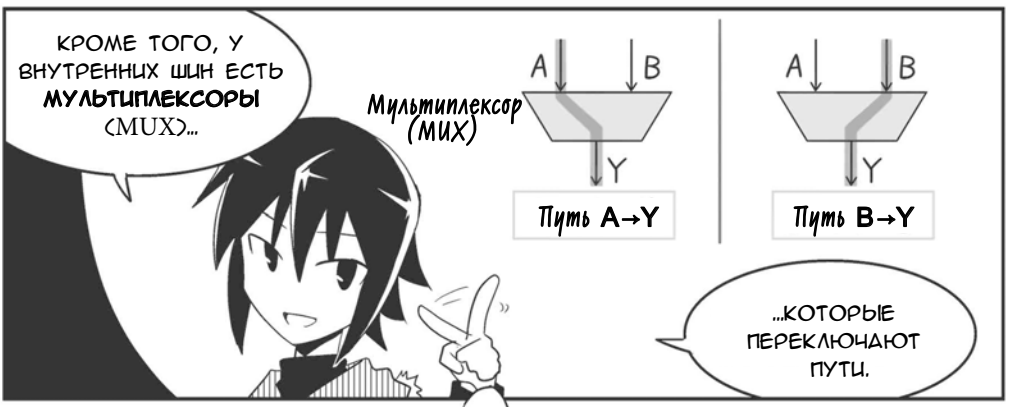

НА САМОМ ДЕЛЕ, КОГДА ЦПУ ВЫЗЫВАЕТ ИЗ ПАМЯТИ ДАННЫЕ ИЛИ ПРОГРАММЫ (ИЛИ КОГДА ОН СОХРАНЯЕТ ДАННЫЕ)...
...ОН ПЕРЕДАЁТ ИХ АДРЕС!
83-й!
Слушаюсь!
ЦПУ
Передача адреса
Память
ПРОСТО ПЕРЕДАЁТ АДРЕС!


ОБРАЩАТЬСЯ ПО НОМЕРАМ – ЭТО, КОНЕЧНО, ПРОСТО И ПОНЯТНО...
...НО ЭТО СУХОЙ, МАШИННЫЙ ЯЗЫК.
ТАК ЭТО И ЕСТЬ МАШИНА.

Шина – это путь данных

* Изначально так назывался общественный транспорт на конной тяге.

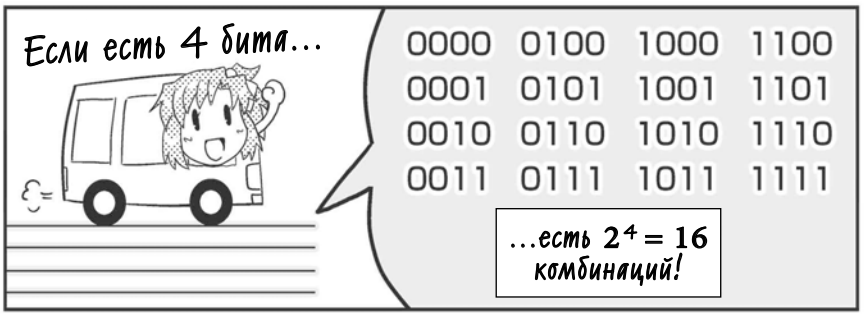

ВНЕШНИЕ ШИНЫ СОЕДИНЯЮТ ЦПУ С ВНЕШНИМИ УСТРОЙСТВАМИ...
Внутренняя шина
Внешняя шина
ЦПУ
Память
В ПОЛНОМ СООТВЕТСТВИИ С НАЗВАНИЕМ...
...А **ВНУТРЕННИЕ ШИНЫ** НАХОДЯТСЯ В САМОМ ЦПУ.


КРОМЕ ТОГО, У ВНУТРЕННИХ ШИН ЕСТЬ **МУЛЬТИПЛЕКСОРЫ** (MUX)...
Мультиплексор (MUX)
A
B
Y
Путь A→Y
Путь B→Y
...КОТОРЫЕ ПЕРЕКЛЮЧАЮТ ПУТИ.


МУЛЬТИПЛЕКСОР ПОЗВОЛЯЕТ УПРОСТИТЬ РАЗВОДКУ ВНУТРЕННЕЙ ШИНЫ ДАННЫХ ЦПУ.
И ЭТО ОЧЕНЬ УДОБНО!


ДЕЙСТВИТЕЛЬНО, КАК ДОРОГИ И АВТОБУСНЫЕ МАРШРУТЫ!
ЧЕМ ПРОЩЕ, ТЕМ УДОБНЕЕ!

Ширина шины и битность

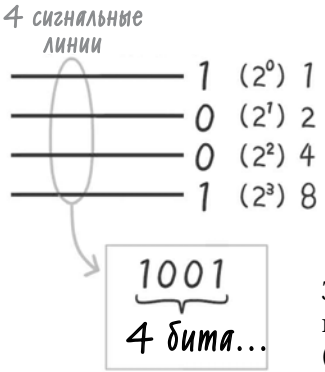


Остановимся на шине подробнее. Я говорил, что шина — это путь данных, но в действительности это пучок сигнальных линий.

А, про сигнальные линии я слышала (см. стр. 56)! Это провода, по которым передаются электрические сигналы 0 или 1 (ВУ или НУ).

Верно! И от количества этих линий зависит разрядность числа, которое можно представить.

Например, у нас есть 4 сигнальные линии. Назначив каждой из них свой весовой коэффициент, мы можем выразить четырёхразрядное (четырёхбитное) двоичное число.



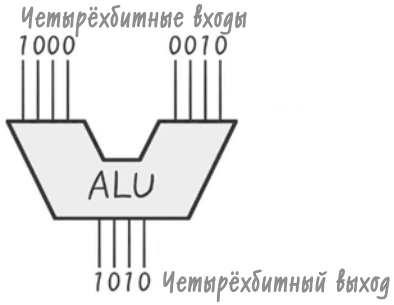
Заодно вспомним про двоичные числа! (см. стр. 39)

Ясно. Число сигнальных линий — это *битность* (количество битов). Например, четырьмя сигнальными линиями (4 битами) можно выразить $2^4 = 16$ чисел от 0 (0000) до 15 (1111).

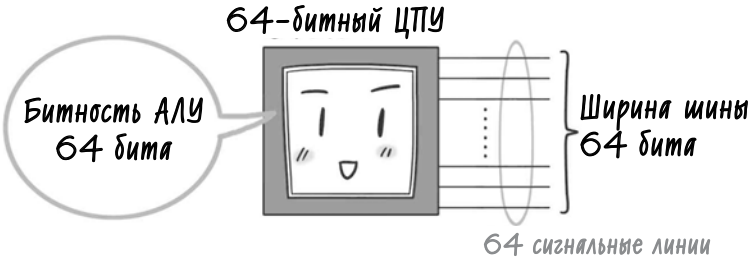
Кажется, я поняла кое-что важное! Чем больше сигнальных линий (чем выше битность), тем лучше, не так ли? Ведь чем бóльшие числа мы сможем представить, тем больше данных мы сможем обработать за один раз.

Ха-ха, верно заметила! И число этих сигнальных линий (битов) называется *шириной шины*. Чем она больше (то есть чем выше битность), тем выше производительность ЦПУ.

Например, на рисунке внизу показано АЛУ, выполняющее четырёхбитную (четырёхразрядную) операцию. Ясно, что ширина шин, соединённых с АЛУ, тоже равна 4 битам.



Здесь для простоты представлен 4-битный АЛУ, однако в последнее время их делают, например, 64-битными. В связи с этим шины данных тоже часто имеют ширину 64 бита, так как удобнее, если битность АЛУ совпадает с шириной шины данных. Именно поэтому часто можно услышать про 64-битные ЦПУ.



Битность ЦПУ и ширина шины данных могут не совпадать. Например, в некоторых 16-битных процессорах примерно 1982 года выпуска использовались 16-битные АЛУ, хотя шина данных была 8 бит. В этих ЦПУ данные считывались из памяти в два приёма.

Раз от ширины шины зависит производительность ЦПУ... чем шире шина данных, тем лучше!

По поводу ширины шины данных хорошо бы усвоить следующее.

Ширина внешней шины данных между ЦПУ и памятью определяет, сколько битов данных можно прочитать или записать за один раз.

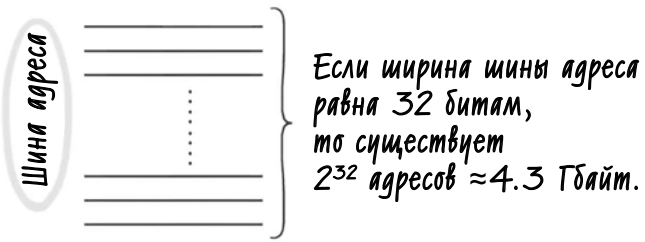
Кроме того, ширина внутренней шины данных, подключённой ко входу АЛУ внутри ЦПУ, определяет, сколько битов можно обработать в один приём.

Ясно! Я поняла, что ширина шины данных — это очень важно!

Итак, после ширины шины данных мы переходим к ширине шины адреса. Она определяет битность (разрядность) адресов, которые мы можем использовать! А также размер пространства адресов (см. стр. 90).

Размер пространства адресов? То есть сколько в нём адресов? Например, если битность равна 4, то адресов будет $2^4 = 16$, так?

Да. Пусть ширина шины адреса равна 32 битам. Следовательно, всего существует $2^{32} = 4\,294\,967\,296$ адресов. В таком случае говорят, что размер пространства адресов равен 32 битам, или примерно 4.3 гигабайтам (Гбайт).



Кроме того, от ширины шины адреса (размера пространства адресов) зависит, какой объём памяти можно использовать*.

* Про объём памяти и ширину шины адреса рассказано на следующей странице.

Значит, эта ширина шины адреса тоже важна. Чем шире, тем лучше!

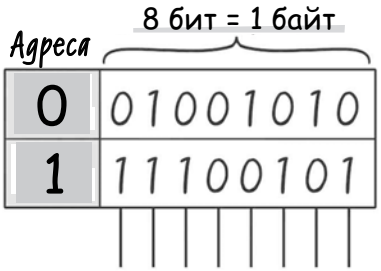
Не жадничай... Хотя это нормальная реакция.

Хорошо иметь широкие шину данных и шину адреса — в разумных, конечно, пределах.

Объём памяти и ширина шин

Посмотрим, как объём памяти связан с шириной (разрядностью) шин. Вот простой пример.

Пусть объём содержимого одного адреса равен *1 байту* (это единица измерения объёма данных, равная 8 битам).

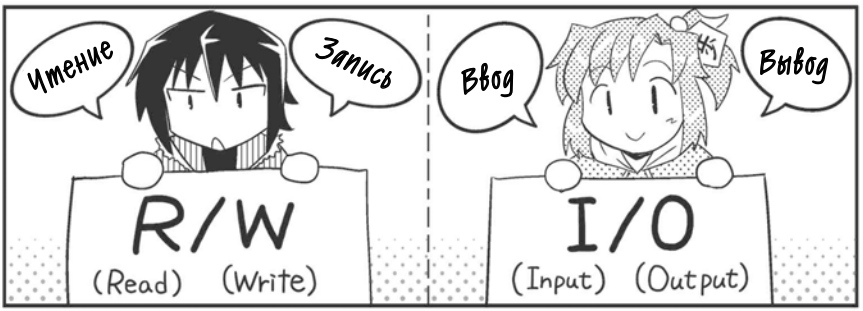


Для передачи данных объёмом 8 бит в один приём требуется 8-битная шина данных

Если размер адресного пространства (ширина шины адреса) равен 12 бит, то количество адресов равно $2^{12} = 4096$. Следовательно, максимальный объём памяти, который мы можем использовать, составит:

$4096 \cdot 1$ байт = 4096 байт ≈ 4 Кбайт.

Управление чтением/записью, управление вводом/выводом



Теперь расскажу о сигналах управления. Тебе знакома аббревиатура R/W?

Red & White... Новогодний песенный конкурс? Поздравляю!

А я поздравляю тебя с очередной глупой шуткой!

На самом деле это очень важная аббревиатура, имеющая отношение к ЦПУ. Прежде всего, R (read) / W (write) означает «чтение/запись».

Чтение — это извлечение сохранённых данных.

Запись — это сохранение данных.

Кстати, используется также другая аббревиатура — L/S: load (загрузка) / store (сохранение).

Разница между read/write и load/store

Можно сказать, что выражение *read/write* относится к аппаратному обеспечению, а *load/store* — к программам. R/W означает электрические сигналы управления памятью (прочитать/записать); при этом не важно, куда поместить прочитанные или откуда взять записываемые данные.

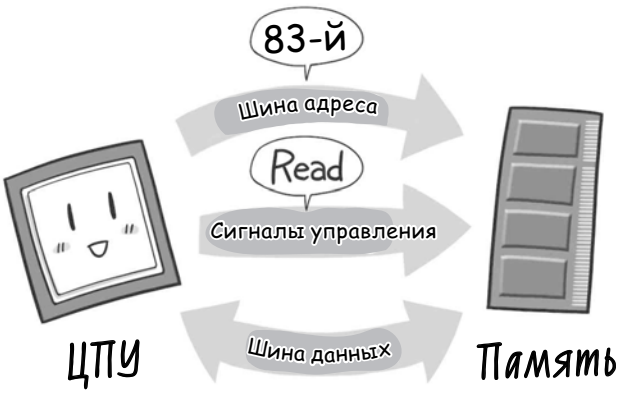
С другой стороны, load подразумевает, что прочитанные из памяти данные загружаются в регистры, а store указывает на перемещение данных из регистров в память, то есть акцент делается на потоки данных.

В этом я уже немного разбираюсь. ЦПУ обменивается данными с памятью, так? Обращаясь к данным, передавая данные результатов операций...

Да, ты правильно понимаешь! В процессе обмена ЦПУ отдаёт памяти приказы: прочитать (read) или записать (write). Вроде таких: «извлеки данные!», «сохрани данные!». Это и есть *сигналы управления* чтением/записью (управление R/W).

Только что я говорил о шине адресов и шине данных, но есть ещё и эти важные сигналы управления*!

* Шину для передачи сигналов управления называют *шиной управления*.



Хм... Значит, если, например, нужно прочитать данные по 83-му адресу, то выдаётся сигнал управления Read (прочитать), и по шине адреса передаётся число 83! А для отправки действительных данных используется шина данных.

Всё верно! На тебя можно положиться! Перехожу к следующему объяснению. Опять начну с вопроса: понимаешь ли ты смысл аббревиатуры I/O?

Ну, наверное, «И. О.» — исполняющий обязанности!

Здесь тебе не викторина «Отгадай слово»! Это сокращение означает I (input) / O (output) — ввод/вывод!

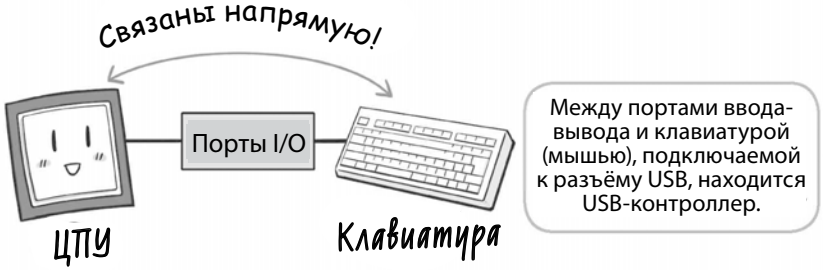
Ввод — это передача данных извне внутрь компьютера. *Вывод* — передача данных из компьютера вовне.

Ага, поняла. Клавиатура или мышь — это устройства ввода, а монитор или принтер — устройства вывода. Input и output!

Верно. И сигналы управления такими внешними устройствами называются *управлением вводом/выводом (сигналами I/O)*. Запомни ещё один термин: *порты ввода-вывода (порты I/O)*. Как морские порты нужны для обмена товарами — импорта и экспорта, так и эти порты обеспечивают обмен данными с внешними устройствами.

Через порты ввода-вывода ЦПУ напрямую связан со внешними устройствами*, например с клавиатурой! Взгляни на этот рисунок.

* Кстати, монитор обычно не связан с ЦПУ напрямую (см. стр. 113).

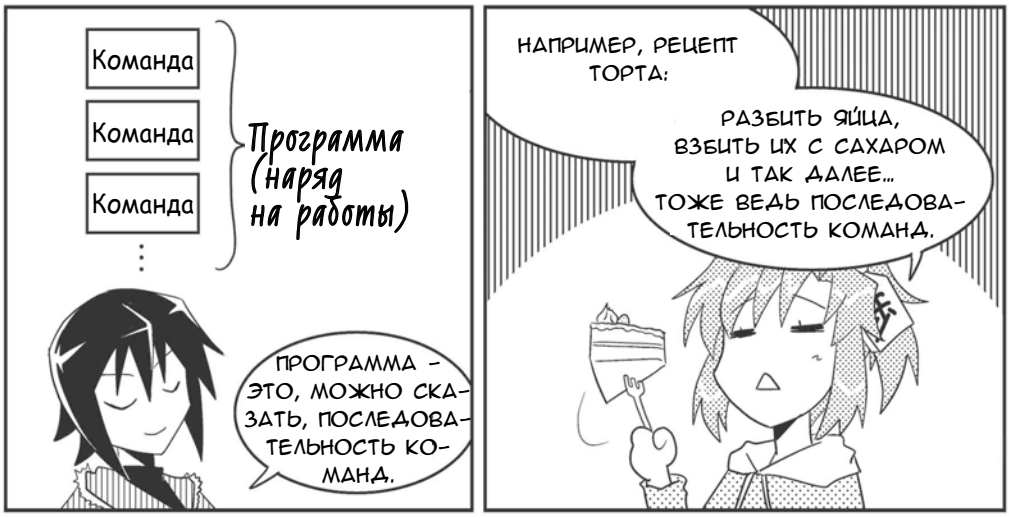


Порты... Действительно, ворота во внешний мир!

ЦПУ и память тоже связаны через порты: порт адреса и порт данных, которые подключены к шине адреса и шине данных!

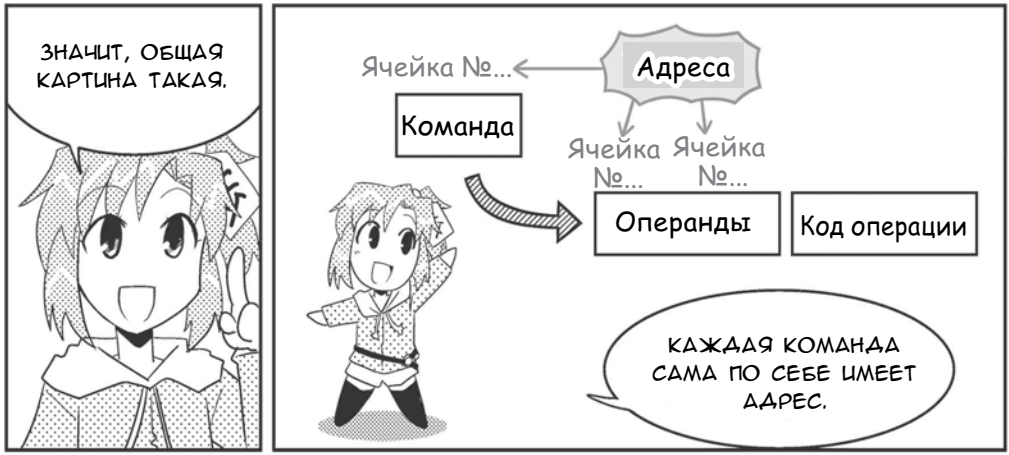
Полезно изучить концептуальную схему ЦПУ (стр. 106), на которой показаны порт адреса, порт данных, управление R/W и управление I/O.

Команды состоят из кода операции и операндов



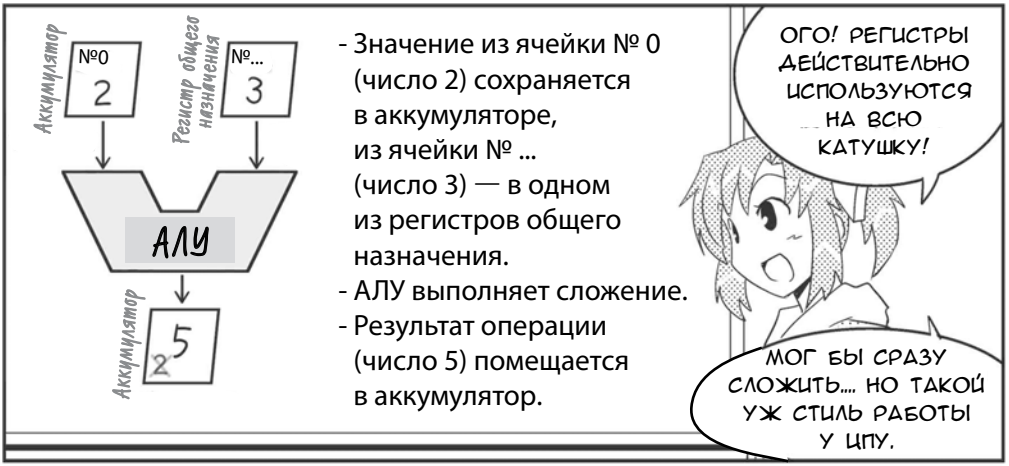
О типах команд пойдёт речь в главе 4.

Иногда операнды могут находиться в регистрах, например аккумуляторе, регистрах специального назначения, о чём будет рассказано позже.



Для операций используются регистры — аккумулятор и другие

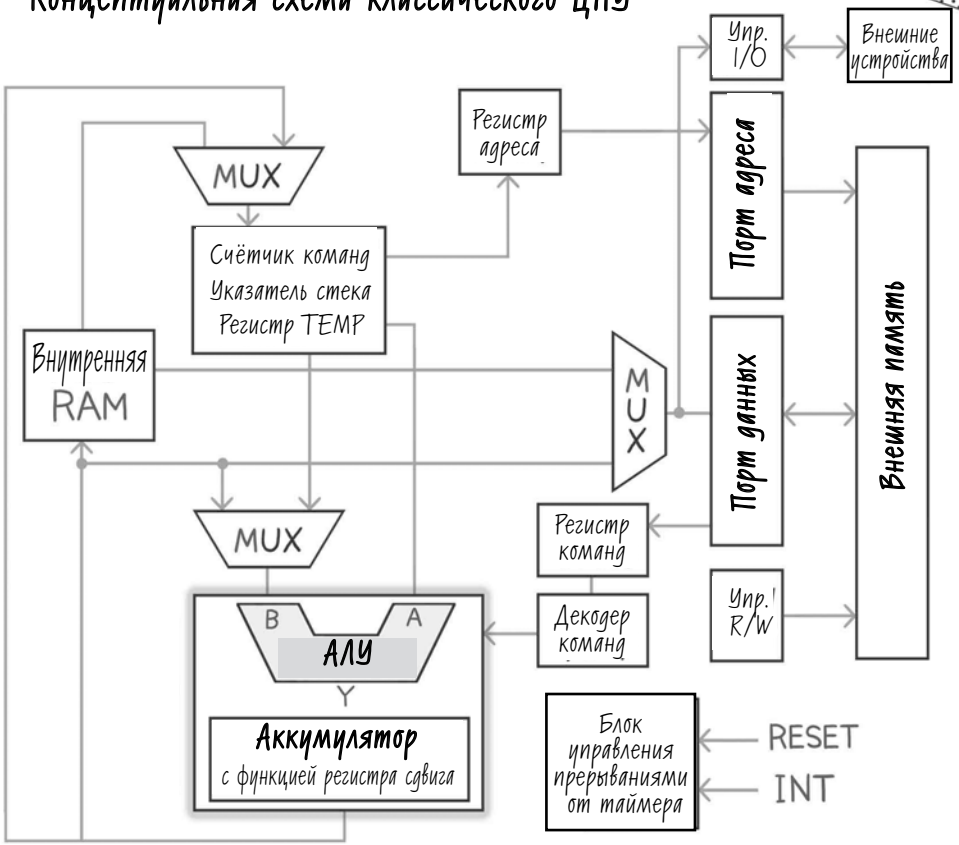

ТАК... РЕГИСТРЫ – ЭТО "ОТРЫВНЫЕ БЛОКНОТЫ" ВНУТРИ ЦПУ.
(См. стр. 71)
Note
ВОТ ТАКИЕ.
ВЕРНО. ВЗГЛЯНИ НА ЭТИ ДВА РЕГИСТРА.
Аккумулятор
В него один за другим записываются результаты вычислений.
Для расчётов
Регистры общего назначения
Могут использоваться в различных целях
Что угодно
АККУМУЛЯТОР – ЭТО КАК БЛОКНОТ ДЛЯ РАСЧЁТОВ. ОН ВСЕГДА ИСПОЛЬЗУЕТСЯ В ОПЕРАЦИЯХ.
РЕГИСТРЫ ОБЩЕГО НАЗНАЧЕНИЯ МОЖНО ИСПОЛЬЗОВАТЬ ПО-РАЗНОМУ – НАПРИМЕР, ДЛЯ ВЫЧИСЛЕНИЙ.
ПРИ ВЫПОЛНЕНИИ ОПЕРАЦИЙ ЭТИ БЛОКНОТЫ ИСПОЛЬЗУЮТСЯ НА ВСЮ КАТУШКУ!
ЧТО ЭТО ЗНАЧИТ?
НАПРИМЕР, КОМАНДА "СЛОЖИТЬ СОДЕРЖИМОЕ ЯЧЕЕК №... И №..." ВЫПОЛНЯЕТСЯ ТАК.
ДАННЫЕ ПО КРАЙНЕЙ МЕРЕ ОДИН РАЗ СОХРАНЯЮТСЯ В РЕГИСТРАХ.



* Про декодирование рассказывается на стр. 109.

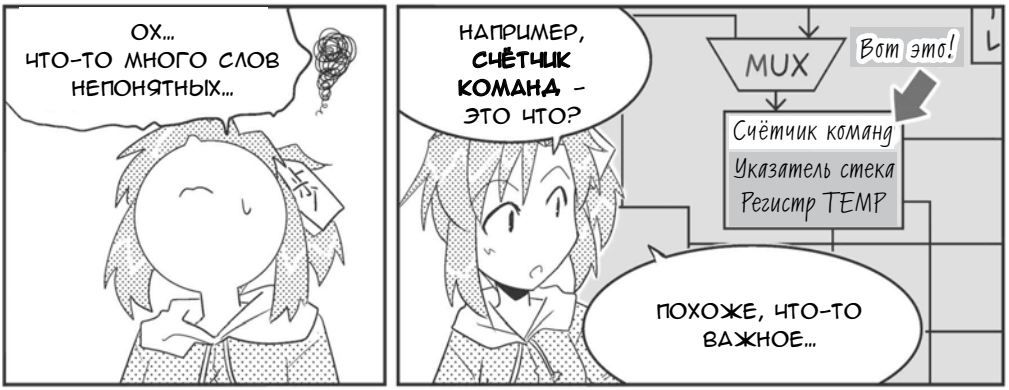
3.2. ОБРАБОТКА КОМАНД В ЦЕНТРАЛЬНОМ ПРОЦЕССОРЕ

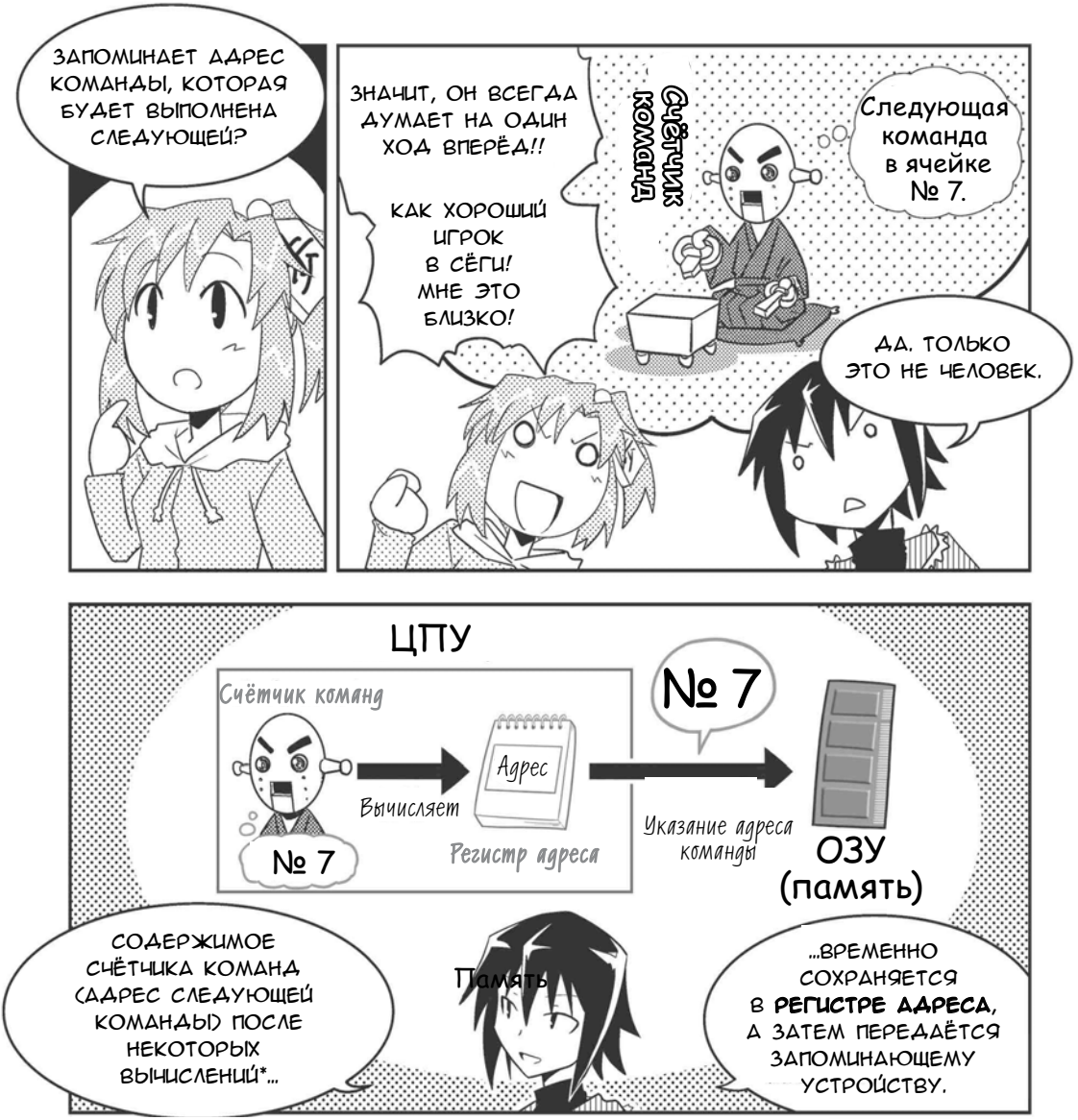
Концептуальная схема классического ЦПУ



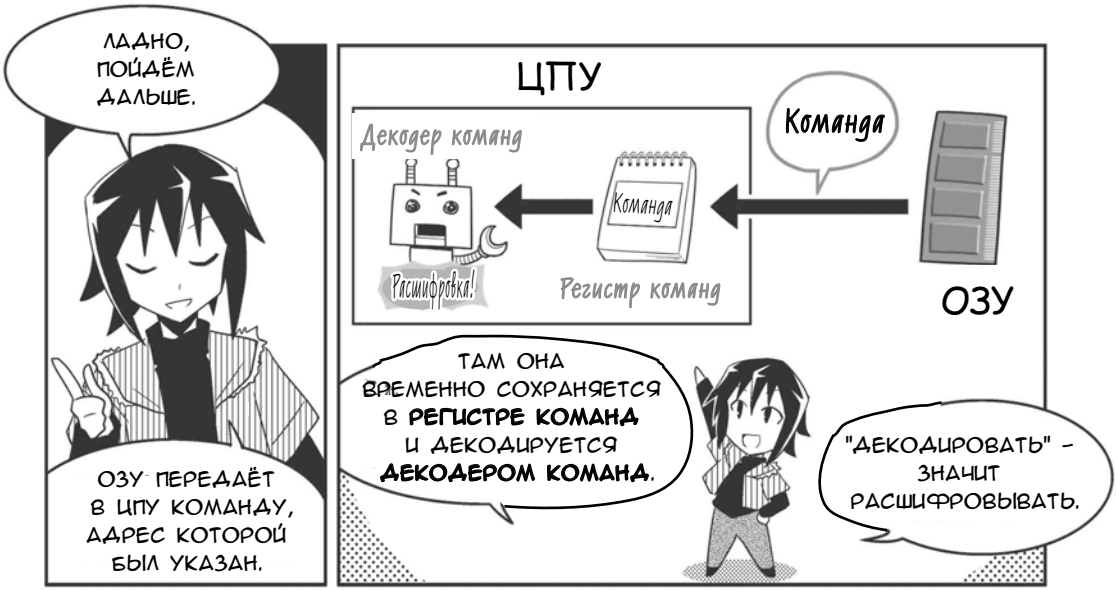
Шины на схеме для наглядности показаны одинарными линиями.

Обработка команд в ЦПУ





* В некоторых случаях адрес вычисляется, о чём будет рассказано на стр. 175, 177.


ЛАДНО, ПОЙДЁМ ДАЛЬШЕ.
ОЗУ ПЕРЕДАЁТ В ЦПУ КОМАНДУ, АДРЕС КОТОРОЙ БЫЛ УКАЗАН.
ЦПУ
Декодер команд
Команда
Расшифровка!
Регистр команд
Команда
ОЗУ
ТАМ ОНА ВРЕМЕННО СОХРАНЯЕТСЯ В РЕГИСТРЕ КОМАНД И ДЕКОДИРУЕТСЯ ДЕКОДЕРОМ КОМАНД.
"ДЕКОДИРОВАТЬ" – ЗНАЧИТ РАСШИФРОВЫВАТЬ.


ЗАЧЕМ ДЕКОДИРОВАТЬ? НЕЛЬЗЯ ЛИ ИСПОЛЬЗОВАТЬ ЕЁ В ТОМ ВИДЕ, В КАКОМ ОНА ХРАНИТСЯ В ПАМЯТИ?

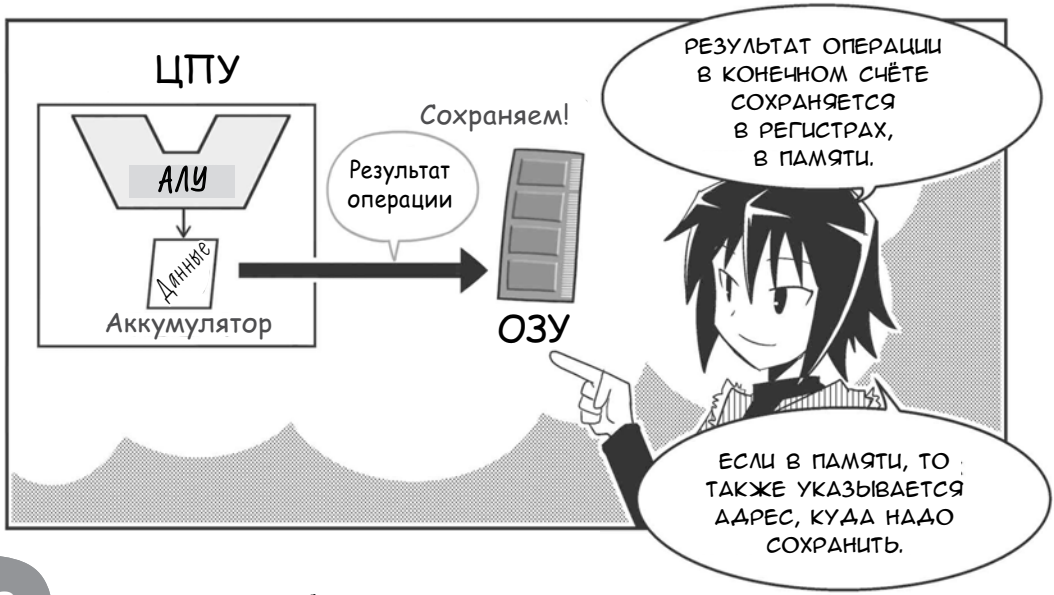

ХМ!
НА САМОМ ДЕЛЕ МАШИННЫЙ ЯЗЫК КОМАНД, ХРАНЯЩИХСЯ В ОЗУ, ОТЛИЧАЕТСЯ ОТ СИГНАЛОВ УПРАВЛЕНИЯ ЦПУ.


Декодер команд
Коммутирует (переключает) устройства внутри ЦПУ так, чтобы выполнилась операция, заданная командой, прочитанной из памяти.
ВОТ ЭТО ДА-А... КАК МНОГО ВСЯКИХ ПРОЦЕДУР.
歩
ПОЭТОМУ ДЕКОДЕР КОМАНД ДОЛЖЕН ПОДГОТОВИТЬ ЦПУ К ВЫПОЛНЕНИЮ ПРОЧИТАННОГО КОДА КОМАНДЫ.

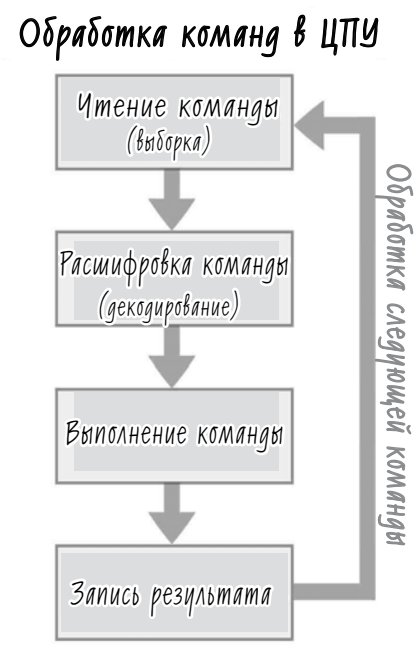

И КОГДА КОМАНДА ДЕКОДИРОВАНА... ВОТ, СМОТРИ!
ТА-ДАМ!
Делай вот это
Код операции
вот с этим.
Операнды
СТАНОВЯТСЯ ИЗВЕСТНЫ ОПЕРАНДЫ И КОД ОПЕРАЦИИ!!


А ЧТО БУДЕТ ДАЛЬШЕ, ТЫ УЖЕ ГОВОРИЛ!
(См. стр. 103)


АЛУ ВЫПОЛНЯЕТ ОПЕРАЦИЮ, ИСПОЛЬЗУЯ... НУ-У... АККУМУЛЯТОР?
ВЕРНО.


ЦПУ
АЛУ
Данные
Аккумулятор
Сохраняем!
Результат операции
ОЗУ
РЕЗУЛЬТАТ ОПЕРАЦИИ В КОНЕЧНОМ СЧЁТЕ СОХРАНЯЕТСЯ В РЕГИСТРАХ, В ПАМЯТИ.
ЕСЛИ В ПАМЯТИ, ТО ТАКЖЕ УКАЗЫВАЕТСЯ АДРЕС, КУДА НАДО СОХРАНИТЬ.

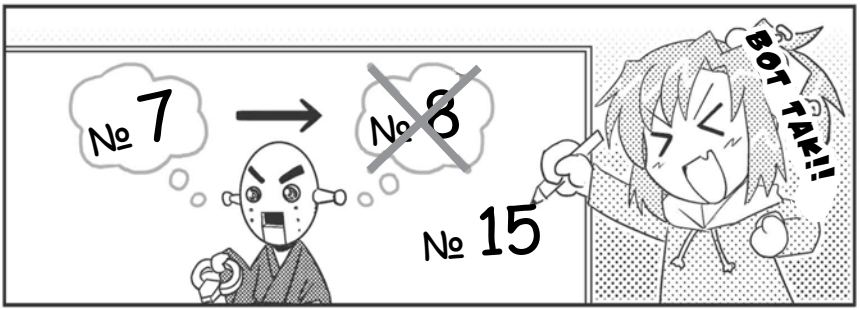

И ТОГДА ЭТУ КОМАНДУ МОЖНО СЧИТАТЬ ВЫПОЛНЕННОЙ?
УГУ. ПОДВЕДЁМ ИТОГИ.
ВЫПОЛНЯЯ КОМАНДЫ (ПРОГРАММУ), ЦПУ ПОВТОРЯЕТ СЛЕДУЮЩИЕ ДЕЙСТВИЯ.


Обработка команд в ЦПУ
Чтение команды (выборка)
Расшифровка команды (декодирование)
Выполнение команды
Запись результата
Обработка следующей команды


ОДНУ КОМАНДУ ВЫПОЛНИЛ – И СРАЗУ СЛЕДУЮЩАЯ? НЕЛЕГКО ЭТО...


НО ОТМЕТИТЬ ВСЁ РАВНО НАДО! ЗА ВЫПОЛНЕННУЮ КОМАНДУ – ДО ДНА! ♪
ПОСТОЙ. НАШ УРОК ЕЩЁ НЕ ОКОНЧЕН.

Счётчик команд позволяет изменять порядок выполнения



М-м... Однако в той схеме ЦПУ (см. стр. 106) ещё так много непонятных слов...

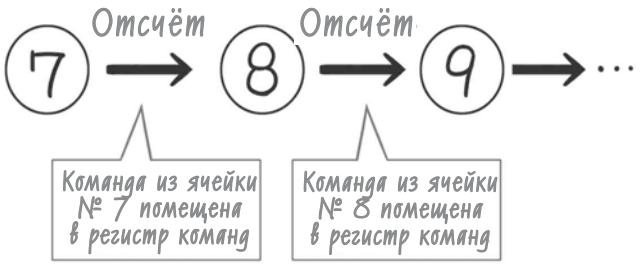
Не торопись. Взгляни на неё ещё раз после следующего занятия. А сейчас я лучше объясню тебе про счётчик команд (или сокращённо PC, от английского Program Counter).

А-а, это тот, который всегда думает на шаг вперёд... то есть всегда помнит адрес следующей команды!

Кстати, раз он называется счётчиком (см. стр. 82), значит, считает числа, так?

После команды из ячейки № 7 выполняется команда из ячейки № 8, затем — № 9 и так далее. В общем, адрес, который в нём хранится, должен постоянно увеличиваться.

Да, принцип такой. Кстати, отсчёт (увеличение номера на единицу) производится сразу же после того, как очередная команда помещается в регистр команд. Как на этом рисунке.



Однако здесь есть важный момент! Дело в том, что за № 7 вовсе не обязательно последует № 8.

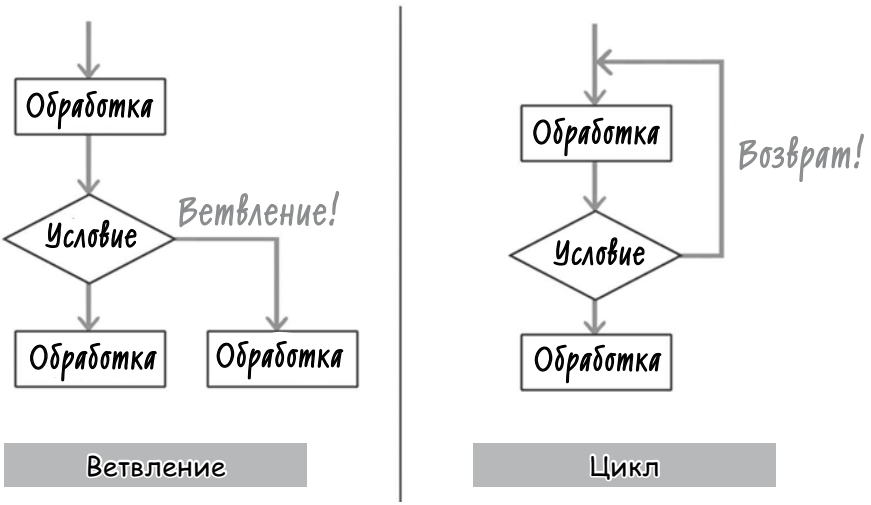
Счётчик команд содержит адрес команды, которая должна выполняться следующей.

Но есть вероятность, что после № 7 он перепрыгнет вперёд, например на № 15, или, наоборот, назад — скажем, на № 3.

Как?! Зачем ему перепрыгивать?!

Хи-хи. Это важный момент — слушай внимательно!

Дело в том, что программа может содержать, например, условные ветвления и циклы, когда при выполнении определённого условия адрес следующей команды изменяется — выполняется переход. Взгляни на эти схемы.



Ага! Это как в том примере с банкоматом (см. стр. 26). Проверяется условие «нехватка средств» и принимается решение о дальнейших действиях. Или, если ввести неправильный пароль, выскочит сообщение об этом и вернётся предыдущий экран.

Да, банкомат — очень хороший пример. А для выполнения этих ветвлений, циклов, достаточно изменить значение счётчика команд на адрес перехода.

Вот как? Изменяя адрес в счётчике команд, выбираем другую команду! Чтобы программа выполнялась так, как нам нужно!

Кроме того, битность счётчика команд (а значит, и битность содержащегося в нём адреса) обычно равна битности шины адреса, то есть битности адресного пространства (см. стр. 96).

Программистам, пишушим приложения под Windows и другие операционные системы персональных компьютеров, не нужно думать о битности счётчика команд. Дело в том, что в этих операционных системах используется метод так называемой виртуальной памяти, избавляющий программиста от необходимости принимать решения о доступе (чтении или записи) к реальной, физической памяти. Устройство, связывающее адреса виртуальной памяти с адресами физической, называется *блоком управления памятью* (Memory Management Unit, MMU).

Разумеется, это так просто! Кстати, счётчик команд способен запоминать только один предстоящий шаг, а игрок в сёги просчитывает партию на много ходов вперёд! Значит, между счётчиком команд и человеком всё-таки есть маленькое отличие...

И ты называешь это «маленьким отличием»?!

3.3. РАЗЛИЧНЫЕ ЗАПОМИНАЮЩИЕ УСТРОЙСТВА

Сравнение жёсткого диска и ОЗУ

Ты раньше не говорил про эту серебристую маленькую коробочку. Что же такое этот жёсткий диск (внешняя память)?

Сейчас быстренько расскажу. Попутно я буду сравнивать его с оперативной памятью (ОЗУ).

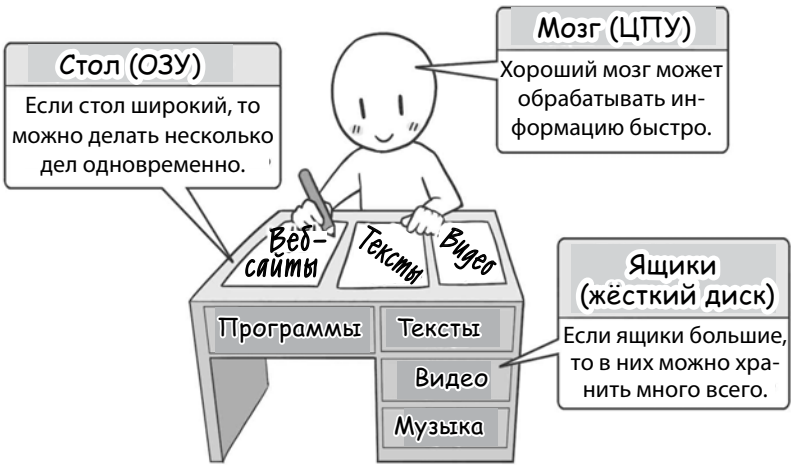
При выключении питания данные в памяти компьютера исчезают, но данные на жёстком диске сохраняются!

Поэтому и операционные системы для управления компьютером (Windows, к примеру), и всякие программы, и данные, которые мы создали или скачали (текст, изображения и прочее), — всё это хранится на жёстком диске.

Что?! Я про это ничего не знала! Ты же всё время говорил, что данные — объекты операций и программы хранятся в оперативной памяти (ОЗУ).

На самом деле часть данных просто копируется с жёсткого диска в память после включения питания.

Ладно! Главное, что все данные надёжно сохраняются на жёстком диске даже после выключения питания. Взгляни на этот рисунок.



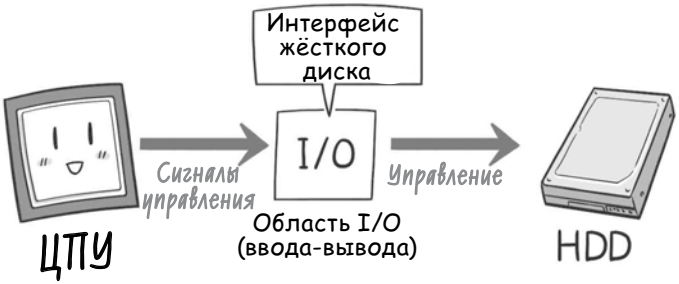
Роль ЦПУ, памяти и жёсткого диска

Здесь показана роль ЦПУ, памяти и жёсткого диска. Можно провести такую аналогию: память — это стол, а жёсткий диск — ящики стола. Теперь поняла разницу между ними?

Ого, они действительно отличаются. Чем больше память, тем легче обрабатывать данные, а чем больше жёсткий диск, тем больше данных можно сохранить.

Теперь о втором отличии.

ЦПУ может обращаться к памяти напрямую, а к жёсткому диску — не может!



ЦПУ не может напрямую управлять жёстким диском!

ЦПУ только передаёт сигналы управления интерфейсу жёсткого диска, находящемуся в области I/O (о ней мы узнаем на стр. 121). Непосредственное управление осуществляет интерфейс!

Работая на компьютере, мы можем свободно обращаться к данным на жёстком диске, поэтому, наверное, трудно представить, что он не связан с ЦПУ напрямую. Однако на самом деле всё именно так, как показано на рисунке.

Другими словами, жёсткий диск не является частью адресного пространства (пространства памяти), которым ЦПУ управляет напрямую!

Вот как? Значит, общаться непосредственно с ЦПУ могут только память и порты ввода-вывода (см. стр. 100). Теперь я понимаю, почему мы уделяли памяти столько внимания!

Теперь о третьем отличии!

Быстродействие памяти выше, чем жёсткого диска!

Если сравнить разные запоминающие устройства компьютера по ёмкости (объёму сохраняемых данных) и быстродействию, то мы получим вот такую пирамиду.

В общем... чем ближе к ЦПУ, тем выше быстродействие, но меньше ёмкость. Чем дальше от ЦПУ, тем ниже быстродействие, но больше ёмкость. Так?

Да! Например, регистры имеют высокое быстродействие, но малую ёмкость. Их можно сравнить с маленькими карманными блокнотиками.

Ясно... Я хорошо поняла, чем память отличается от жёсткого диска. Хотя и то, и другое запоминающее устройства, у них совершенно разные функции, соответствующие их особенностям.

Кстати, в таких компьютерах, как современные ноутбуки, вместо жестких дисков (HDD), содержащих движущиеся части, используются так называемые *твердотельные накопители* (Solid State Disks, SSD) — запоминающие устройства на полупроводниках, выполняющие точно такие же функции, что и HDD, но более устойчивые к механическим воздействиям, например вибрациям.

Области RAM, ROM, I/O

Итак, теперь я объясню про пространство адресов (памяти). Помнишь ли ты, что это такое?

А-а, разумеется. Пространство под прямым управлением ЦПУ... (см. стр. 90).

Адрес	Содержимое
0	Команда
1	Команда
2	Команда
3	Команда
4	Команда
⋮	⋮
30	Данные
31	Данные
32	Данные
33	Данные
⋮	⋮

Пространство адресов
(пространство памяти)

Правильно. Но если точнее, адресное пространство — это все внешние запоминающие устройства, которыми управляет ЦПУ.

Все?.. А что, есть ещё какие-то кроме ОЗУ?

Угу. Сейчас я скажу кое-что важное. На самом деле внешняя память, соответствующая пространству адресов, бывает двух типов: RAM (оперативное запоминающее устройство, ОЗУ), доступная и для чтения, и для записи, и ROM (постоянное запоминающее устройство, ПЗУ) — постоянная память, доступная только для чтения. Соответственно, в пространстве адресов имеются две области: RAM и ROM.

См. стр. 132

Какие-то RAM и ROM — что ещё за новости?!

Та-ак, разберёмся... ROM — обычная оперативная память (ОЗУ), доступная для чтения и записи, данные в которой исчезают при выключении питания...

Но что такое ROM? Память, хранящая данные даже после выключения питания, но доступная только для чтения? Принадлежащая пространству адресов? Что это за штуковина такая?

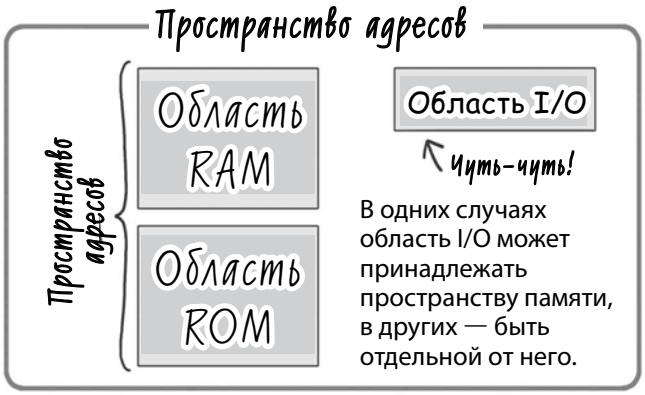
Ты, наверное, не знаешь, но в материнских платах компьютеров есть микросхема ROM под названием BIOS, хранящая программу «первых шагов» ЦПУ после включения питания.

Вот как? Да-а, без тех самых первых шагов компьютер — всего лишь железный ящик. Эту программу, которую нельзя забыть, нужно хранить в ROM!

Что такое BIOS?

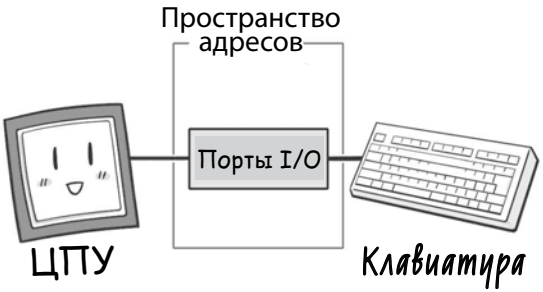
В ROM хранится самая базовая программа под названием BIOS (Basic Input/Output System — базовая система ввода-вывода), которая проверяет исправность различных устройств компьютера сразу же после включения питания, а затем загружает с жёсткого диска операционную систему (например, Windows).

Кроме того, есть также область I/O, хотя она очень мала по сравнению с областями RAM и ROM.



Я хорошо помню, что значит I/O. Это Input/Output, то есть ввод-вывод (см. стр. 100).

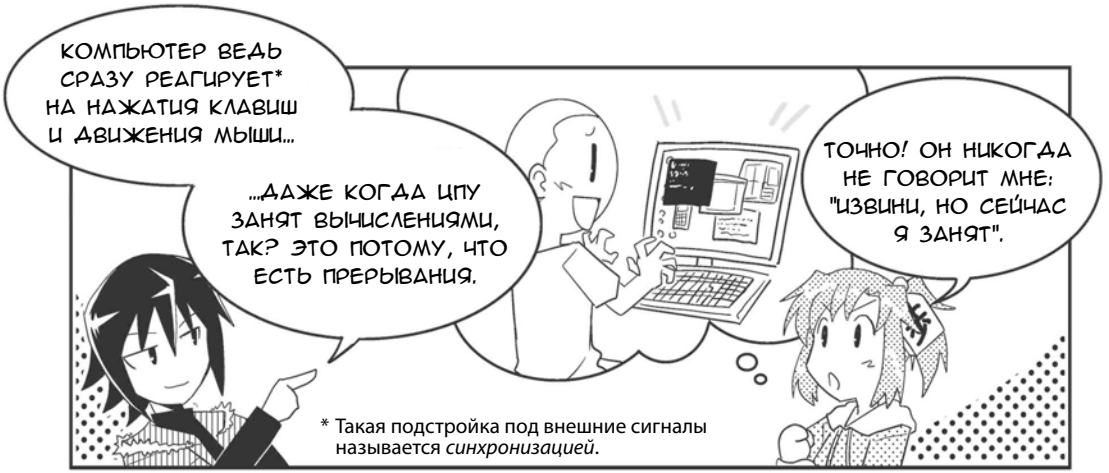
Верно. В этой области I/O расположены порты I/O (порты ввода-вывода). Как я уже говорил, через эти порты ЦПУ напрямую связан с внешними устройствами (клавиатурой, например). Вот почему компьютер реагирует на нажатия клавиш.



Ого! Молодец, ЦПУ! Управлять внешними устройствами через пространство адресов очень удобно! Итак, теперь я знаю, что в пространстве адресов есть разные области!

3.4. Что такое прерывания?


ПОЛУЧАЕТСЯ, ЧТО НЕОЖИДАННЫЙ ЗВОНОК ПРЕРВАЛ ТВОЮ РАБОТУ.
А ЕСЛИ ОН К ТОМУ ЖЕ ВАЖНЫЙ – ХОРОШО, ЧТО ТЫ ОТВЛЕКЛАСЬ, НЕ ПРАВДА ЛИ?
ВОН ОНО ЧТО…
ДА, РАДИ ВАЖНОГО ЗВОНКА МОЖНО НА ВРЕМЯ ПРЕРВАТЬ РАБОТУ.
ВДРУГ ЭТО СТАРИЧОК, КОТОРЫЙ МНЕ ХОЧЕТ СОСТОЯНИЕ ЗАВЕЩАТЬ. ЗА ТО, ЧТО Я ЕМУ МЕСТО В ТРАНСПОРТЕ УСТУПИЛА.
ПО-ТВОЕМУ, ЭТО УДАЧНЫЙ ПРИМЕР?!!
НУ ЛАДНО… В ОБЩЕМ, ПРЕРЫВАНИЯ ПОЗВОЛЯЮТ…
Дело А
Готовка
Дело Б
Звонок
…ХОРОШО ДЕЛАТЬ НЕСКОЛЬКО ДЕЛ.


КОМПЬЮТЕР ВЕДЬ СРАЗУ РЕАГИРУЕТ* НА НАЖАТИЯ КЛАВИШ И ДВИЖЕНИЯ МЫШИ...
...ДАЖЕ КОГДА ЦПУ ЗАНЯТ ВЫЧИСЛЕНИЯМИ, ТАК? ЭТО ПОТОМУ, ЧТО ЕСТЬ ПРЕРЫВАНИЯ.
ТОЧНО! ОН НИКОГДА НЕ ГОВОРИТ МНЕ: "ИЗВИНИ, НО СЕЙЧАС Я ЗАНЯТ".
* Такая подстройка под внешние сигналы называется *синхронизацией*.


ИМЕННО БЛАГОДАРЯ ПРЕРЫВАНИЯМ...
...ЦПУ МОЖЕТ СОСРЕДОТОЧИТЬСЯ НА ОСНОВНОЙ ЗАДАЧЕ (ВЫЧИСЛЕНИЯХ, НАПРИМЕР).
ЕСЛИ БЫ ЕМУ ПРИХОДИЛОСЬ ЧАСТО ПРОВЕРЯТЬ, НАПРИМЕР, НЕ НАЖАТА ЛИ КЛАВИША...

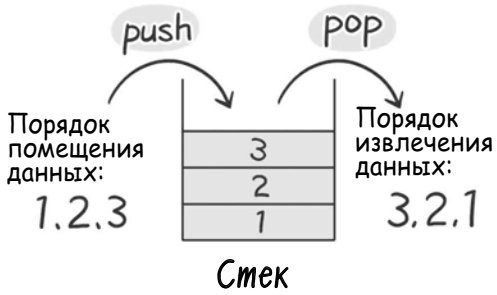

Без функции прерываний
С функцией прерываний
ЦПУ
Клавиатура
Пауза в вычислениях...
Не нажата ли клавиша?
Не нажата ли?
Не нажата ли?
Да не было ничего!
Ага!
Клавиша нажата!
...ПОЛУЧИЛОСЬ БЫ ВОТ ТАК!
ОГО! СУЩЕСТВЕННАЯ РАЗНИЦА В ЭФФЕКТИВНОСТИ...


ВАЖЕН ТАКЖЕ МЕХАНИЗМ ВОЗВРАТА К ПРЕЖНИМ ВЫЧИСЛЕНИЯМ...
...ПОСЛЕ ВЫПОЛНЕНИЯ ЗАДАЧИ ПРЕРЫВАНИЯ.
В ОБЩЕМ, НУЖНО ГДЕ-ТО ЗАПОМНИТЬ ПРОМЕЖУТОЧНЫЙ РЕЗУЛЬТАТ ВЫЧИСЛЕНИЙ, ЗНАЧЕНИЕ СЧЁТЧИКА КОМАНД.
Чтобы не забыть!
• 390
•Команда № 77
и т. п.
УГУ. ЭТО ПОНЯТНО.
ОБИДНО БЫЛО БЫ, ЕСЛИ БЫ В ПРОЦЕССЕ ПРИГОТОВЛЕНИЯ ОБЕДА ПРОДУКТЫ КУДА-ТО ИСЧЕЗЛИ...
...ИЛИ Я САМА ЗАБЫЛА БЫ, НА КАКОМ ЭТАПЕ ПРЕРВАЛАСЬ.
ВОТ ИМЕННО.
НУ А СЕЙЧАС...
...Я РАССКАЖУ О МЕХАНИЗМЕ...
...ПРЕ...
...ПРЕРЫВАНИЙ?!
ТЫ МЕНЯ ПРЕРЫВАЕШЬ!!!

Стек и его указатель

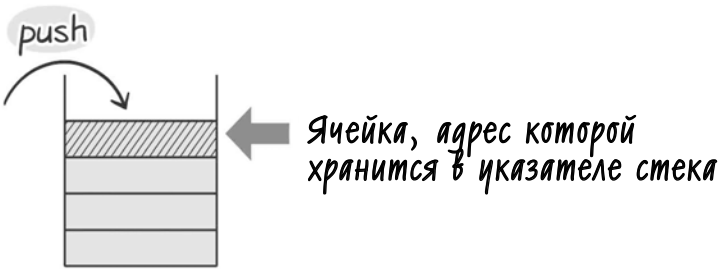
Итак, начнём! Чтобы по завершении прерывания вернуться к прежней задаче, нужно напоминание «не забыть!», о котором я только что говорил.

Для этого есть функция под названием *стек*. Под стек отводят часть оперативной памяти и используют её в качестве запоминающего устройства с немного необычным способом хранения данных. Взгляни на этот рисунок.



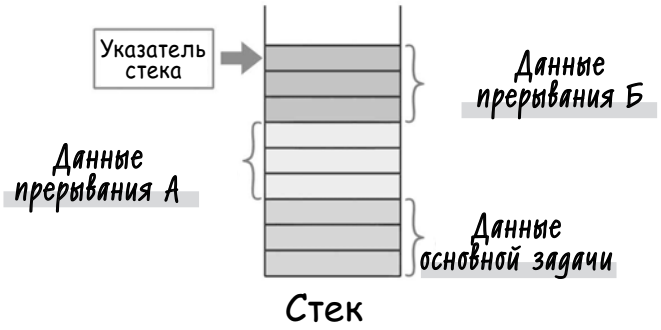
Какое странное запоминающее устройство!.. Хочется сравнить это со стопкой книг на столе: сначала укладываем книги одну на другую, а потом снимаем по порядку, начиная с самого верха. Сразу взять нужную книгу (нужные данные) не получится!

Точно. Есть регистр под названием *указатель стека* (Stack Pointer, SP), в котором запоминается адрес вершины стека, то есть адрес ячейки, в которую данные были помещены последними.



Вот оно что... Счётчик команд запоминает адрес ячейки со следующей командой, а указатель стека, значит, — адрес самой верхней ячейки стека.

При использовании стека нужно умело обращаться с указателем стека. Однако есть одна проблема... Если много прерываний возникают одно за другим, то в стеке накапливается очень много данных...



Конкретно говоря, в стек «временно эвакуируется» содержимое трёх регистров: аккумулятора, регистра состояния и счётчика команд. О регистре состояния см. стр. 162.

Ого! Так можно и запутаться!

Ты права. Если в программе есть ошибки (так называемые «баги», bugs), или если в ней не предусмотрена обработка множественных прерываний, указатель стека не справляется с ситуацией — программа может выйти из под контроля. Это говорит о том, что создатель программы не очень хорошо умеет обращаться с указателем стека...

А у тебя, Ю-кун, программы когда-нибудь из под контроля выходили?

Вообще говоря, концепция прерываний очень помогает в работе с ЦПУ, однако требует умелого обращения с указателем стека, чтобы программы не выходили из под контроля... На этом всё!

Ясно! Насколько я понимаю, у тебя тоже были проблемы с указателем стека.

А-ха-ха!.. Это была просто общая информация! Не фантазируй!

Приоритеты прерываний

Кхе-кхе... Соберёмся-ка с силами и изучим приоритеты прерываний. Представь, что во время готовки пришло прерывание от телефона, а потом, уже во время телефонного разговора, раздался звонок в дверь. Что ты будешь делать?

Какое неудачное стечение обстоятельств! С этим даже я не справлюсь. Не надо так много прерываний одновременно!

Хи-хи-хи... Трудно, да? В этом случае поможет такая штука, как *маска прерываний*, позволяющая отклонять прерывания! Это всё равно как маска, которой закрывают лицо...

Про установку маски прерываний будет рассказано на стр. 191.

Да, маска может защитить от многого!

Но расслабляться всё равно нельзя. Есть такое принудительное прерывание под названием *сброс* (reset), которое даже маской не запретишь! Приоритет сброса — самый высокий из всех причин прерываний. Для сигнала сброса есть особый вход, на который действие маски не распространяется...

Сброс! Да уж, с ним особо не поспоришь! Кнопка сброса и на игровой приставке есть... В общем, он возвращает устройство в первоначальное состояние.

Именно так. Кстати, при включении питания игровой приставки или компьютера загрузка происходит из начального состояния, не так ли? Причина в том, что сразу же после включения питания первым делом происходит сброс*.

Сброс приводит к инициализации программ. Другими словами, он возвращает внутренние микросхемы в их первоначальное состояние. Благодаря этому возможен нормальный запуск компьютера.

* О действии сброса рассказывается на стр. 138.

Вот как? Сброс мне казался каким-то насильственным решением, но как он важен, оказывается!..

Есть ЦПУ, где невозможно замаскировать ещё одно прерывание, которое хоть и не имеет такой же силы, как сброс, но обладает самым высоким приоритетом.

Вход для подобных немаскируемых (то есть «неизбежных») прерываний обозначается NMI (сокращение от Non Maskable Interrupt — «немаскируемое прерывание»). Его назначение может отличаться в различных системах, но в любом случае такой вход очень удобен при правильном использовании...

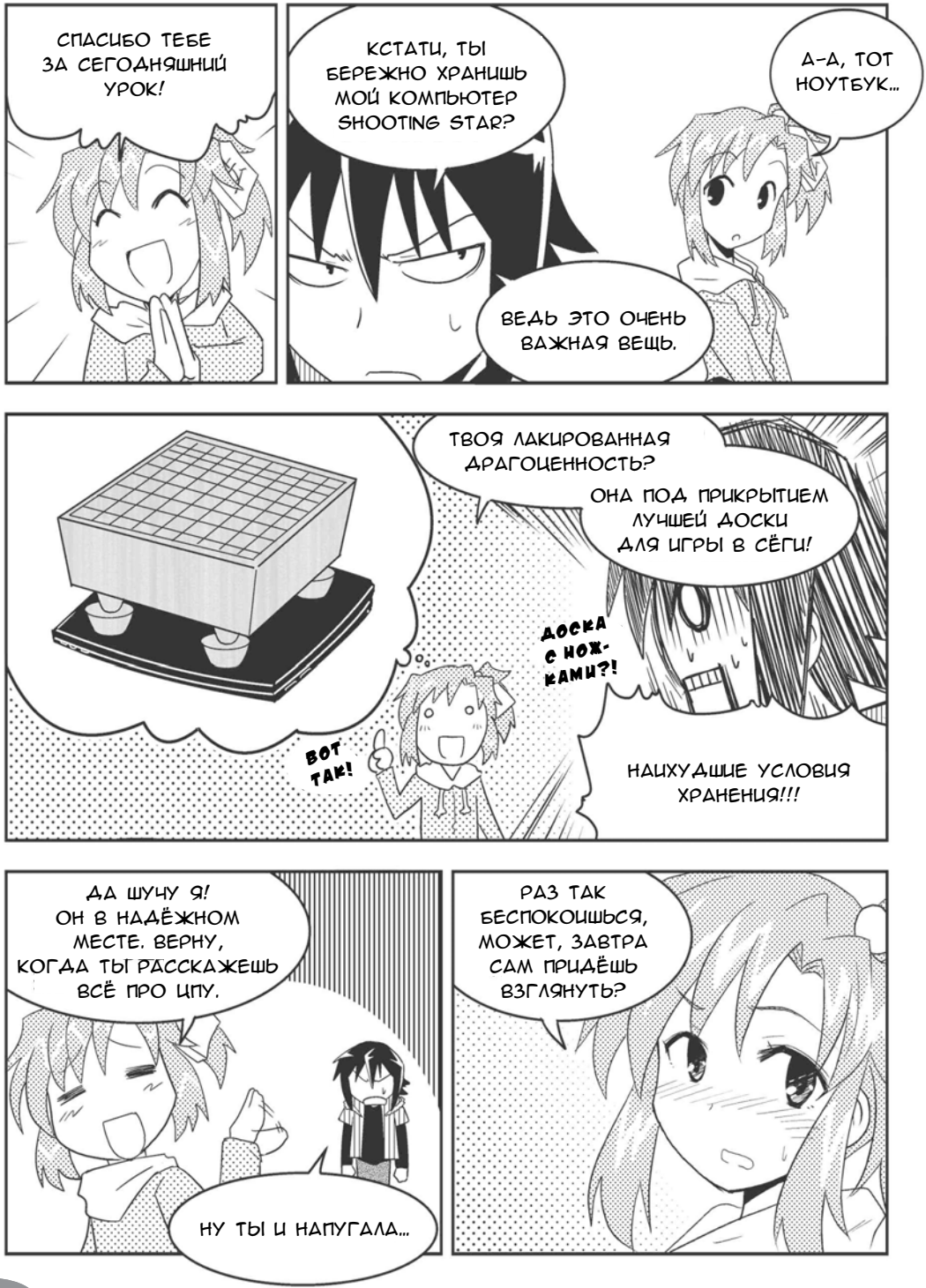
Да, использовать прерывания тоже можно по-разному. Как он глубок, этот мир прерываний!

Ещё есть такой механизм, как *прерывания от таймера**, позволяющий вызывать прерывания каждый раз, когда значение вычитающего счётчика (счётчика обратного счёта) становится равным нулю. Что-то вроде «Три, два, один, ноль — прерывание!». Это позволяет выполнять определённую программу через равные промежутки времени.

* О прерываниях от таймера пойдёт речь на стр. 136.

Ага! Я придумала хороший пример для прерываний от таймера. Каждый день я выполняю задачу под названием «сон». И каждые 24 часа ровно в 7 часов утра выполняется программа «звонок» — прерывание, нарушающее мой мирный сон!

Ах-ха-а, да это же обыкновенный будильник! Ну да...


Спасибо тебе за сегодняшний урок!
Кстати, ты бережно хранишь мой компьютер Shooting Star?
А-а, тот ноутбук...
Ведь это очень важная вещь.
Твоя лакированная драгоценность?
Она под прикрытием лучшей доски для игры в сёги!
Доска с ножками?!
Вот так!
Наихудшие условия хранения!!!
Да шучу я! Он в надёжном месте. Верну, когда ты расскажешь всё про ЦПУ.
Ну ты и напугала...
Раз так беспокоишься, может, завтра сам придёшь взглянуть?


ЧТО?
В ПОСЛЕДНЕЕ ВРЕМЯ Я ПЕРЕБРАЛА КАЛОРИЙ В ФАСТФУДЕ С ЭТИМИ ТОРТИКАМИ!
ДА И ДЕНЕГ НА КАФЕ УЖЕ ЖАЛКО! ЗАВТРА КАК РАЗ ВОСКРЕСЕНЬЕ, ДАВАЙ У МЕНЯ ДОМА ПОЗАНИМАЕМСЯ. А Я ЧАЕМ УГОЩУ!
ХМ-М... НУ, КАК СКАЖЕШЬ.

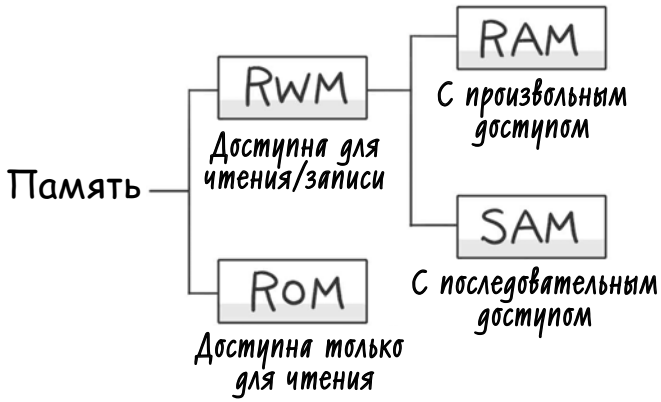

ТОЛЬКО ПРИБЕРИСЬ В СВОЕЙ КОМНАТЕ КАК СЛЕДУЕТ!
ПРЕДУПРЕЖДАЮ – НИ ОДНА ПЫЛИНКА НЕ ОСТАНЕТСЯ НЕЗАМЕЧЕННОЙ!!
БАМ!
Я ЧТО, УБОРЩИЦА?!
ЧТО ЭТО ТЫ ЗАДУМАЛ?! МЫ БУДЕМ ТОЛЬКО ИЗУЧАТЬ ЦПУ – И НИЧЕГО БОЛЬШЕ, ЯСНО???

ДОПОЛНИТЕЛЬНАЯ ИНФОРМАЦИЯ

◆ Типы памяти

ROM — это сокращение от Read Only Memory (память только для чтения). Содержимое этой памяти, не исчезающее даже при отключении питания, доступно только для чтения.

RAM — это Random Access Memory (память с произвольным доступом). Её содержимое доступно как для чтения, так и для записи, а слово Random («произвольный») означает, что обращаться можно к произвольной ячейке, указав её адрес. Может показаться, что RAM и ROM противоположны по назначению, но на самом деле это не так.



По вышеприведённой схеме видно, что противоположностью RAM является SAM (Sequential Access Memory — память с последовательным доступом). Это такие устаревшие накопители информации, как магнитные ленты или барабаны, в которых для чтения или записи блока данных в определённой позиции нужно было прочитать или перезаписать все предыдущие блоки.

В то же время память ROM изначально противопоставлялась RWM (Read/Write Memory) — памяти для чтения и записи.

Память, которая продолжает хранить данные даже после выключения питания, позволяет читать прежние данные и перезаписывать их после повторного включения, называется *энергонезависимой памятью*. Если же при выключении питания все сохранённые данные исчезают, то такую память называют *энергозависимой*.

В последнее время термин ROM стали употреблять в отношении энергонезависимой, а RAM — в отношении энергозависимой памяти.

* В отечественной литературе сокращению ROM соответствует ПЗУ (постоянное запоминающее устройство), а RAM — ОЗУ (оперативное запоминающее устройство). — *Прим. перев.*

◆ Порты I/O, GPU

Невозможно ввести информацию извне в ЦПУ или АЛУ, если к ним не присоединены устройства ввода-вывода (Input/Output: I/O).

Ввод — это не только набор текста на клавиатуре, но и поступление информации об электрических сигналах, возникающих, например, при нажатии кнопок. Кроме того, чтобы мы, пользователи, могли «общаться» с ЦПУ, результаты операций тоже нужно выводить в виде электрических сигналов, способных, скажем, зажечь светодиод.

Как и в случае с внешним ОЗУ, для подключения внешних устройств к внутренней шине используются «ворота», называемые портами ввода-вывода (портами I/O).

Основным устройством вывода привычных нам компьютеров является экран монитора, который на самом деле обычно не соединён с ЦПУ напрямую. Изображение на экране монитора создаётся с помощью вычислительной микросхемы под названием «*графический процессор*» (Graphics Processing Unit, GPU), предназначенной специально для *обработки графики*. ЦПУ для больших систем, в которых предполагается использование GPU, оснащаются специальными портами I/O для подключения GPU.

В системах с цветным ЖК-дисплеем (жидкокристаллическим дисплеем, англ. LCD), даже меньших, чем персональный компьютер, ЦПУ через порты I/O передаёт данные LCD-контроллеру, формирующему и выводящему изображение.

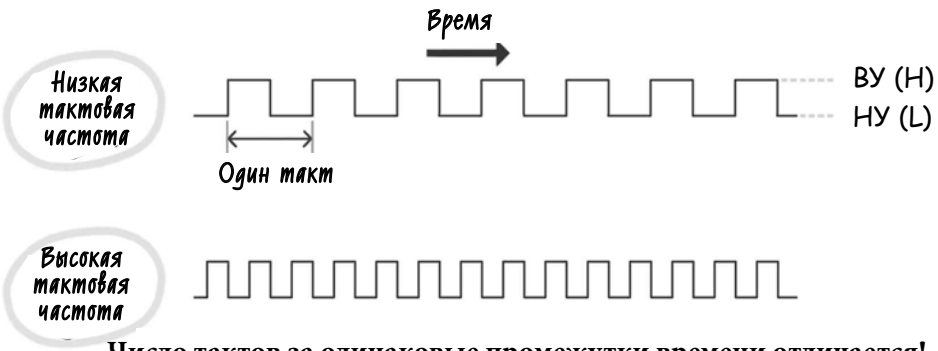
◆ Тактовая частота и её точность

Для функционирования ЦПУ требуется не только электропитание, но и тактовый сигнал (clock), который с определённой периодичностью изменяется с «низкого» уровня на «высокий» и обратно. *Тактовая частота* — это число периодов тактового сигнала (тактов) за 1 секунду.

Тактовый сигнал — это «биение сердца» ЦПУ, необходимое для работы внутренних схем (блоков*, защёлкивающих данные в АЛУ, увеличивающих значение счётчика команд и т. д.).

* Блоком называют составную часть устройства, служащую для выполнения определённой функции.

Единицей измерения тактовой частоты является *герц* (Гц); в герцах выражается число тактов в секунду. Например, тактовая частота 40 МГц означает, что за одну секунду повторяется 40 млн тактов. Эта характеристика очень важна с точки зрения производительности ЦПУ: ведь за каждый такт ЦПУ выполняет какое-либо действие, поэтому чем она больше, тем больше действий ЦПУ может выполнить за 1 секунду.



Число тактов за одинаковые промежутки времени отличается!

Другой характеристикой является *точность тактовой частоты*, которая показывает, насколько она соответствует определённому значению — например, 40 МГц. При использовании компьютеров в качестве средства связи возможно нарушение синхронизации, если опорные тактовые частоты в сигнальной линии связи перестанут совпадать.

◆ Тактовый генератор

Схема формирования тактового сигнала (генератор сигналов) называется *тактовым генератором*. Хотя обычно блок тактового генератора содержится внутри ЦПУ, можно также вводить в него внешний тактовый сигнал.

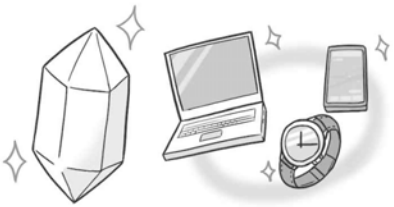
Схема генератора сигналов (кварцевый резонатор + усилитель + конденсаторы и сопротивления) реализуется путём подбора электронных компонентов, из-за разброса параметров которых возможно снижение точности тактовой частоты. Тактовый сигнал низкой точности может использоваться тогда, когда необходимости разгонять ЦПУ до предельной скорости, а также в случае, если обмен данными с другими устройствами не является важной задачаей.

С другой стороны, внешний генератор сигналов является отдельным устройством и может быть настроен с высокой точностью, поэтому хорошо подходит для ЦПУ, использующихся для управления аппаратурой связи и т. п., а также в тех случаях, когда используемый для связи сигнал требует повышенной точности.

Что такое кварцевый резонатор?

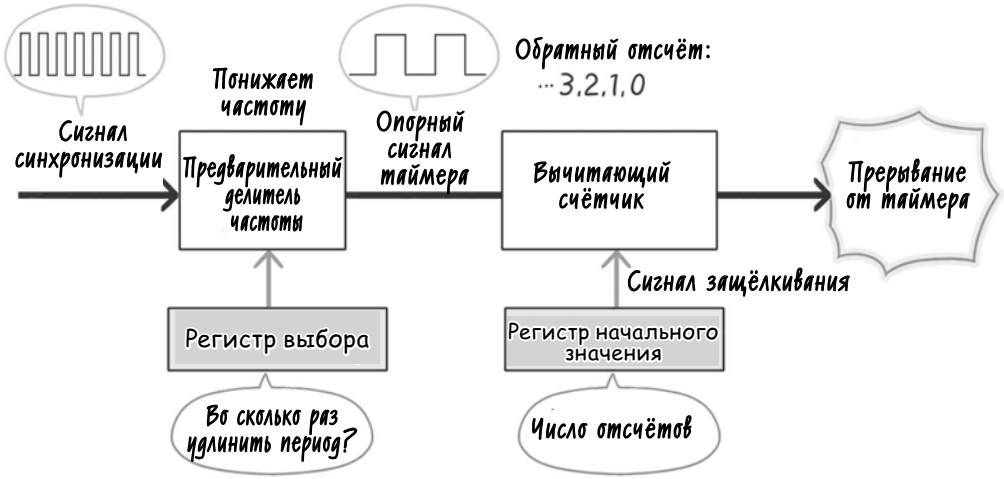
Тактовый генератор представляет собой усилитель, дополненный кварцевым резонатором, конденсаторами (электронными компонентами, накапливающими и отдающими электрическую энергию) и т. п. и способный вырабатывать сигнал определённой частоты. В кварцевом резонаторе содержится очень тонкая кварцевая пластинка, изготовленная из искусственного кварца, с двумя нанесёнными на неё электродами. При подаче на электроды напряжения пластинка изгибается, и если это напряжение сделать переменным, то можно вызвать в ней правильные колебания фиксированной частоты, что позволит с высокой точностью отсчитывать промежутки времени.

Кварцевые резонаторы используются в различных изделиях, требующих точного отсчёта промежутков времени: ПК, мобильных телефонах, кварцевых часах.



◆ Прерывания от таймера

Прерывания, вырабатываемые в моменты обнуления вычитающего счётчика, находящегося внутри ЦПУ, называют *прерываниями от таймера*.



Блок-схема прерывания от таймера

Порядок использования прерывания от таймера

Сначала нужно с помощью команды ЦПУ записать в регистр выбора число, указывающее, во сколько раз период опорного сигнала таймера будет больше периода сигнала синхронизации.

Затем нужно записать в регистр начального значения число отсчётов, через которое будет возникать прерывание.

Затем нужно с помощью команды ЦПУ отменить состояние «сброс таймера» (см. следующую страницу), что приведёт к запуску прерываний от таймера.

Это приведёт к подаче сигналов прерывания от блока таймера в схему управления ЦПУ через промежутки времени, равные произведению:

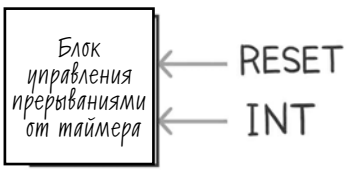
[Период сигнала синхронизации] × [Коэффициент предварительного делителя частоты] × [Начальное значение вычитающего счётчика].

В качестве сигнала синхронизации используется опорный тактовый сигнал ЦПУ, заставляющий его функционировать. На выходе предварительного делителя частоты формируется опорный сигнал таймера, период которого в целое число раз больше периода сигнала синхронизации и может составлять от нескольких миллисекунд до нескольких сотен микросекунд. Одному периоду опорного сигнала соответствует уменьшение на единицу значения вычитающего счётчика.

Задав начальное значение вычитающего счётчика в программе, можно выполнять требуемую программу через определённые промежутки времени. Например, в случае, если прерывания от таймера используются для получения мигающего сигнала, то есть включения и выключения светодиода через равные промежутки времени, вычитающий счётчик производит обратный отсчёт (периодически уменьшает своё содержимое на 1) даже во время выполнения других программ и при достижении нуля вызывает прерывание. Это позволяет более эффективно использовать возможности ЦПУ.

Далее, для изменения частоты прерываний достаточно командой ЦПУ изменить число отсчётов. Например, если в качестве начального значения вместо 100 использовать 50, то частота прерываний увеличится в два раза.

В заключение поговорим о блоке управления прерываниями от таймера на концептуальной схеме классического ЦПУ (стр. 106). Через вход INT в блок управления прерываниями вводятся команды от ЦПУ, а вход RESET («*сброс таймера*») нужен для запуска прерываний от таймера.



Если переключить вход «сброс таймера» в активное состояние, то таймер не будет функционировать, находясь в состоянии остановки. Если отменить состояние «сброс таймера», то таймер запустится и начнёт генерировать прерывания. Другими словами, вычитающий счётчик защёлкнет начальное значение и начнёт уменьшать его на единицу через промежутки времени, кратные в заданное число раз периоду сигнала синхронизации, при достижении нуля таймер сгенерирует сигнал прерывания. Одновременно с этим вычитающий счётчик вновь защёлкнет начальное значение и начнёт его уменьшать. Благодаря повторению этих действий вырабатываются периодические прерывания.

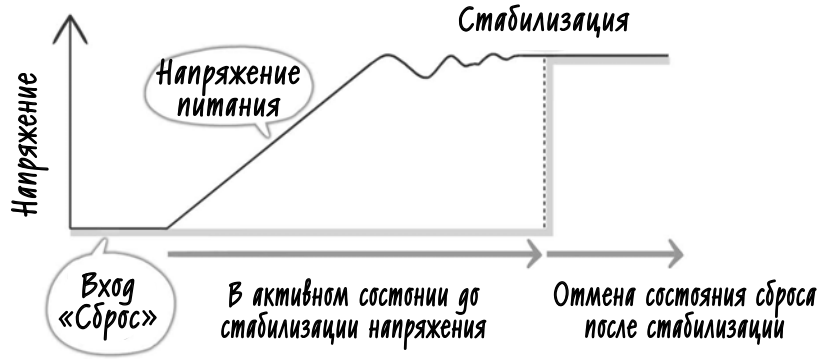
◆ Действие сброса

Сброс означает инициализацию программ и возврат внутренних схем ЦПУ в начальное состояние. Например, обнуляется значение счётчика команд, исчезают промежуточные результаты операций.

При включении питания компьютера первым делом выполняется сброс. Это необходимо для инициализации программ, чтобы они выполнялись в правильной последовательности.

Расскажу о порядке выполнения сброса. При подаче на вход «Сброс» низкого логического уровня (НУ) он переходит в *активное состояние*. Дело в том, что после включения питания компьютера требуется некоторое (небольшое) время для стабилизации напряжения питания. Запуск ЦПУ (начало выполнения программы) в это время может привести ко сбоям, поэтому до момента стабилизации напряжения вход «сброс» находится в активном состоянии и ЦПУ не запускается.

Другими словами, это состояние сброса обеспечивает защиту ЦПУ до момента стабилизации напряжения. Затем, после стабилизации напряжения, на вход «Сброс» подаётся высокий логический уровень (ВУ), отменяющий состояние сброса.



Связь напряжения питания и логического уровня на входе «Сброс»

Если по какой-либо причине ЦПУ выполняет непредусмотренное действие, то можно вручную подать на вход «Сброс» низкий логический уровень (т. е. перевести его в активное состояние), обеспечив тем самым инициализацию программ. Таким образом, механизм сброса незаменим для обеспечения нормальной работы компьютера.

◆ Определение производительности ЦПУ (значение FLOPS)

Производительность ЦПУ определяется тактовой частотой ЦПУ и скоростью выполнения операций. На основании этих характеристик даётся обобщённая оценка производительности.

От тактовой частоты ЦПУ зависит, как быстро могут работать логические схемы, используемые для вычислений АЛУ, а скорость вычислений показывает, как быстро могут выполняться операции, следующие одна за другой.

В классических ЦПУ блок АЛУ выполнял арифметические операции только над целыми числами, поэтому производительность лучше было оценивать не по скорости выполнения арифметических операций, а по числу миллионов команд, выполняемых за 1 секунду — MIPS (Million Instructions Per Second, мипс). Разумеется, и в те времена путём программных ухищрений выполняли операции над числами с десятичной точкой, однако современные ЦПУ имеют встроенное аппаратное обеспечение для выполнения операций с плавающей точкой, поэтому в наши дни для определения скорости выполнения арифметических операций используется единица MFLOPS (Million FLoating point number Operations Per Second, мегафлопс), показывающая, сколько миллионов операций с плавающей точкой выполняется за 1 секунду.

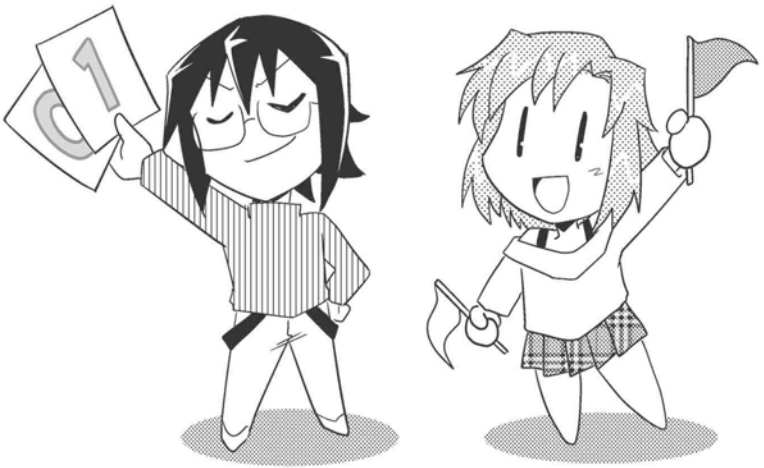
Например, производительность 1 MFLOPS означает, что ЦПУ может за 1 секунду выполнить 1 млн операций над числами с плавающей точкой, имеющими 15 значащих разрядов.

К сходным единицам относятся GFLOPS (гигафлопс) и TFLOPS (терафлопс).

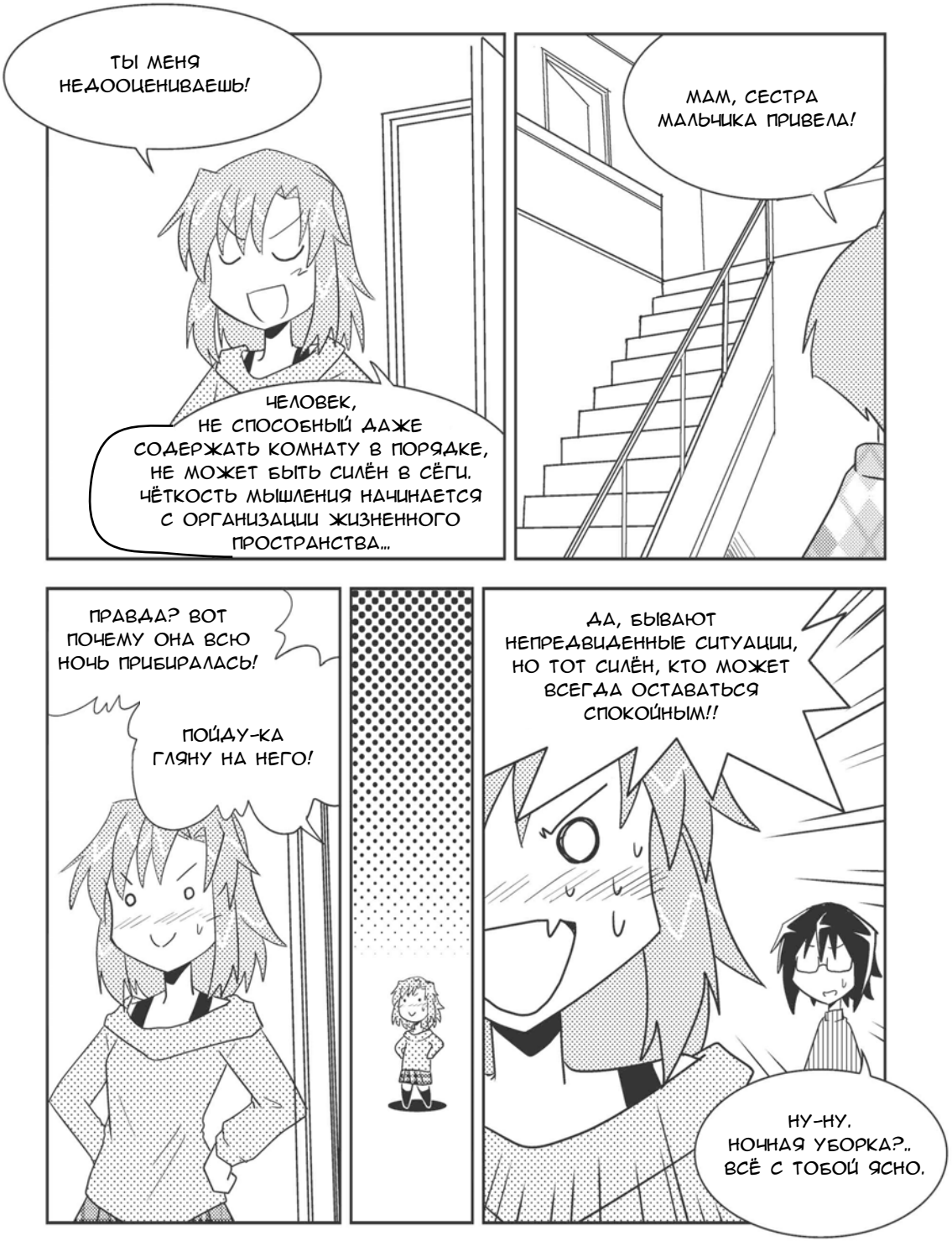
Производительность 1 GFLOPS означает возможность выполнения 1 млрд операций, а 1 TFLOPS — 1 трлн операций над числами с плавающей точкой, имеющими 15 значащих разрядов.

ГЛАВА 4

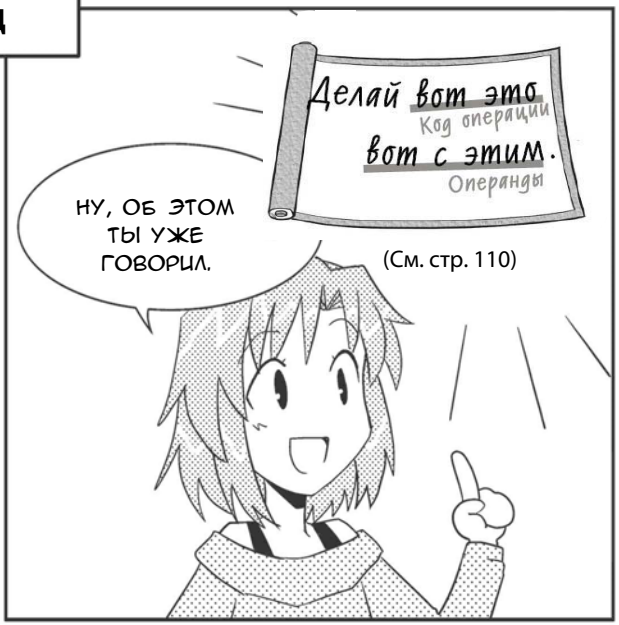
КОМАНДЫ ДЛЯ ВЫПОЛНЕНИЯ ОПЕРАЦИЙ



Ч.1. ТИПЫ КОМАНД


ТЫ МЕНЯ НЕДООЦЕНИВАЕШЬ!
ЧЕЛОВЕК, НЕ СПОСОБНЫЙ ДАЖЕ СОДЕРЖАТЬ КОМНАТУ В ПОРЯДКЕ, НЕ МОЖЕТ БЫТЬ СИЛЁН В СЁГИ. ЧЁТКОСТЬ МЫШЛЕНИЯ НАЧИНАЕТСЯ С ОРГАНИЗАЦИИ ЖИЗНЕННОГО ПРОСТРАНСТВА...
МАМ, СЕСТРА МАЛЬЧИКА ПРИВЕЛА!
ПРАВДА? ВОТ ПОЧЕМУ ОНА ВСЮ НОЧЬ ПРИБИРАЛАСЬ!
ПОЙДУ-КА ГЛЯНУ НА НЕГО!
ДА, БЫВАЮТ НЕПРЕДВИДЕННЫЕ СИТУАЦИИ, НО ТОТ СИЛЁН, КТО МОЖЕТ ВСЕГДА ОСТАВАТЬСЯ СПОКОЙНЫМ!!
НУ-НУ. НОЧНАЯ УБОРКА?.. ВСЁ С ТОБОЙ ЯСНО.

Различные типы команд

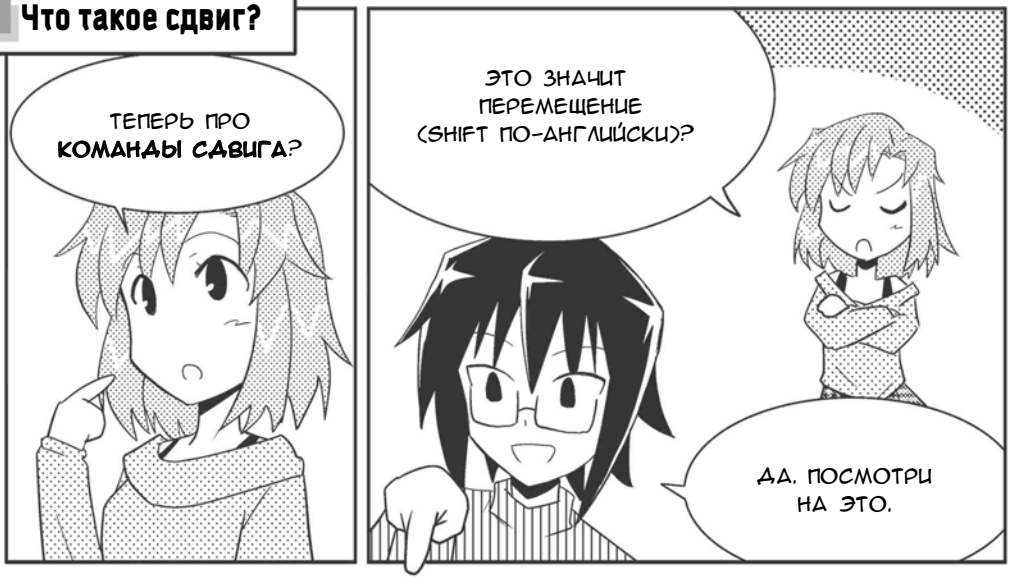


(См. стр. 110)

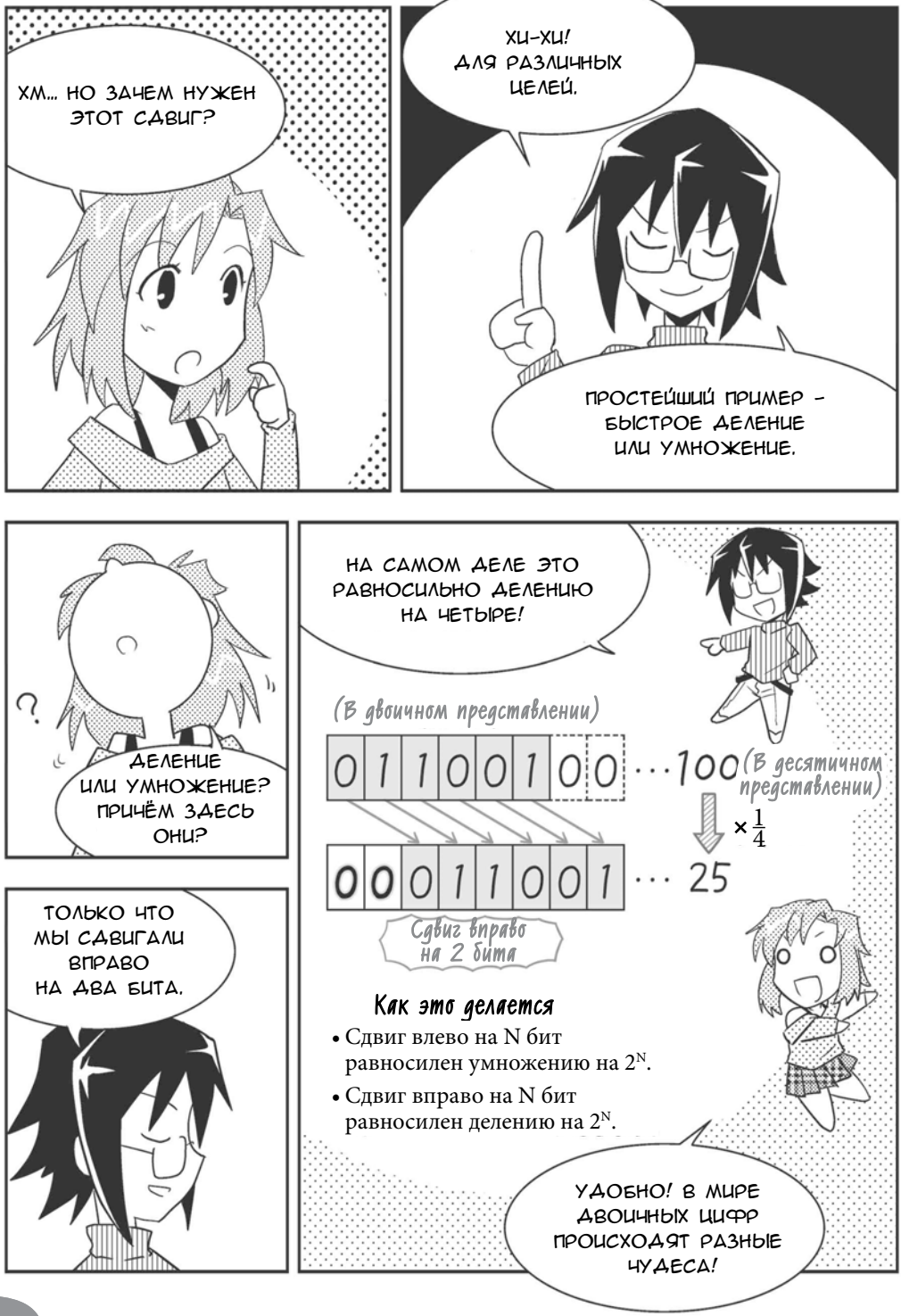

ДА! ВОТ ОНИ – РАЗНЫЕ ТИПЫ КОМАНД!
Различные команды (машинный язык)
Команды для операций
① Арифметические команды
② Логические команды
③ Команды сдвига
Другие команды
④ Команды пересылки данных
⑤ Команды ввода-вывода
⑥ Команды ветвления
⑦ Команды проверки условия (сравнения и т. п.)
СЕГОДНЯ Я РАССКАЖУ О НИХ ПО ПОРЯДКУ.
ОГО, СКОЛЬКО ИХ ТУТ!
НЕ ВОЛНУЙСЯ, ТЫ ВЕДЬ УЖЕ МНОГО ЗНАЕШЬ.
ПОНЯВ ЭТИ КОМАНДЫ, ТЫ БУДЕШЬ ЗНАТЬ, ЧТО ПРОИСХОДИТ ВНУТРИ ЦПУ.
ЯСНО… ТОГДА ОТВЕЧАЙ, КАКИЕ БЫВАЮТ КОМАНДЫ! ДАЮ 3 СЕКУНДЫ!
ЭТО НЕВЫПОЛНИМАЯ КОМАНДА!!!

Арифметические и логические команды

Что такое сдвиг?



В аккумуляторе есть функция сдвига.

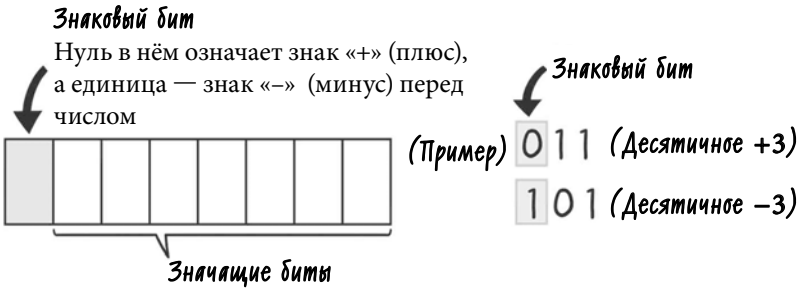

ХМ... НО ЗАЧЕМ НУЖЕН ЭТОТ СДВИГ?
ХИ-ХИ! ДЛЯ РАЗЛИЧНЫХ ЦЕЛЕЙ.
ПРОСТЕЙШИЙ ПРИМЕР – БЫСТРОЕ ДЕЛЕНИЕ ИЛИ УМНОЖЕНИЕ.
ДЕЛЕНИЕ ИЛИ УМНОЖЕНИЕ? ПРИЧЁМ ЗДЕСЬ ОНИ?
НА САМОМ ДЕЛЕ ЭТО РАВНОСИЛЬНО ДЕЛЕНИЮ НА ЧЕТЫРЕ!
(В двоичном представлении)
0 1 1 0 0 1 0 0 ... 100 (В десятичном представлении)
× 1/4
0 0 0 1 1 0 0 1 ... 25
Сдвиг вправо на 2 бита
Как это делается
• Сдвиг влево на N бит равносилен умножению на 2^N.
• Сдвиг вправо на N бит равносилен делению на 2^N.
ТОЛЬКО ЧТО МЫ СДВИГАЛИ ВПРАВО НА ДВА БИТА.
УДОБНО! В МИРЕ ДВОИЧНЫХ ЦИФР ПРОИСХОДЯТ РАЗНЫЕ ЧУДЕСА!

Знаковый бит для представления отрицательных чисел

Поговорим о сдвиге поподробней. Но перед этим я должен объяснить тебе, что такое знаковый бит.

Знаковый бит?

Знаковый бит — это самый старший бит. Если он равен 0, то это положительное число, а если 1 — отрицательное. На рисунке ниже самый старший бит *знаковый*, остальные *значащие*.



Так.. Двоичное 001 — это десятичное +3. Но почему двоичное 101 вдруг стало десятичным –3? Непонятно...

А ты вспомни про дополнительный код, используемый для представления отрицательных чисел в двоичной системе счисления (см. стр. 44)!

Чтобы получить дополнительный код:

① Инвертируем все биты.

② Прибавляем единицу к результату инверсии.

В самом деле, для числа 3 (011) противоположным по знаку будет –3 (101). Самый старший бит стал знаковым битом!

Три значащих бита выражают восемь значений от 0 до 7 (см. стр. 39), но если самый старший из них сделать знаковым, то они станут выражать восемь значений от –4 до 3.

Значение	Дополнительный код
3	0 1 1
2	0 1 0
1	0 0 1
0	0 0 0
-1	1 1 1
-2	1 1 0
-3	1 0 1
-4	1 0 0

Знаковый бит

Трёхбитные двоичные числа со знаком

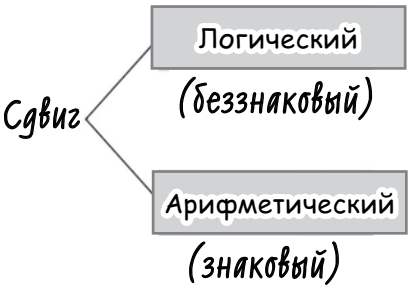
Хм... Но, например, есть двоичное число 101. Если считать, что оно имеет знаковый бит (число со знаком), это будет –3, а если считать, что знакового бита в нём нет (число без знака), это 5. Одно и то же двоичное число представляет разные значения. Так ведь запутаться можно!

Да, человек их действительно не отличит. Но не беспокойся — для различения таких значений в компьютере есть есть соответствующий механизм*.

* Существует так называемый флаг знака, показывающий смену знака результата операции (о нём будет рассказано на стр. 161). Проверяя этот флаг в программе, можно узнать, не было ли непредусмотренной смены знака. Правда, этот флаг есть не во всех ЦПУ. Если его нет, то необходимо обнаруживать непредусмотренную схему знака с помощью программных ухищрений.

Логический и арифметрический сдвиг

Наконец-то переходим к сдвигу. На самом деле он может быть либо *арифметическим*, либо *логическим*. Они отличаются друг от друга тем, есть или нет знаковый бит.

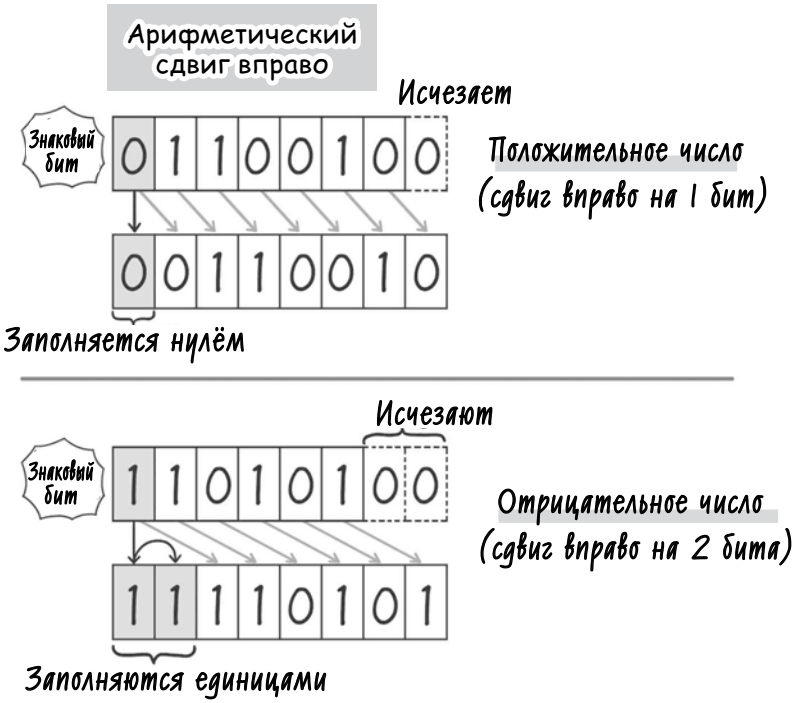


Значит, при логическом сдвиге нет знакового бита, а при арифметическом есть... Кажется, я понимаю.

Ведь в мире логики есть всего два значения – например, ВКЛЮЧЕНО и ВЫКЛЮЧЕНО. Поэтому там нет такого понятия, как «отрицательные числа» (знаковый бит).

Верно заметила! Логический сдвиг проще — это такой сдвиг, о котором я недавно рассказывал (см. стр. 147). А вот арифметический немного посложнее. Сейчас постараюсь объяснить.

Взгляни на схему. При арифметическом сдвиге вправо биты заполняются значением самого старшего (знакового) бита: если он 1, то присходит заполнение единицами, а если 0 — нулями. Таким образом, нужно учитывать знаковый бит.

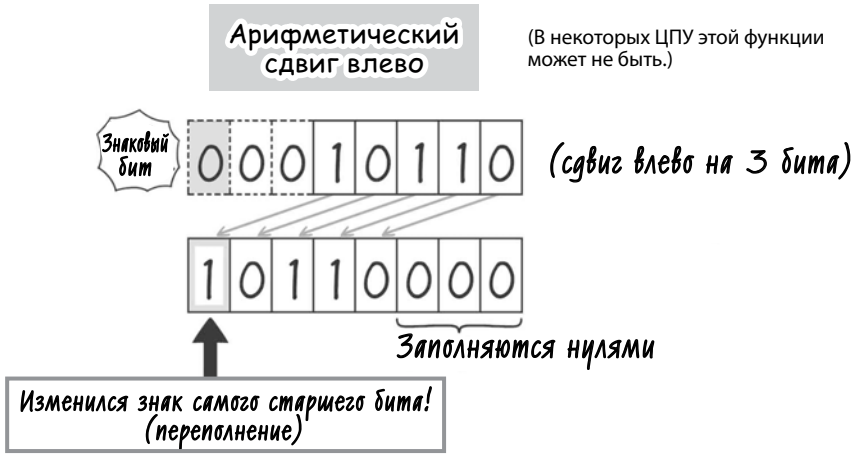


Вот как? При логическом сдвиге всегда нулями заполнялось, а при арифметическом надо ещё и о знаке думать...

Есть ещё один важный момент. Посмотри на рисунок на следующей странице. При сдвиге положительного значения влево в самом старшем бите может оказаться единица из значащего разряда.

Это проблема! Ведь число можно принять за отрицательное (так как знаковый бит равен 1).

Да. Сдвигом положительного числа влево на N битов мы хотели умножить его на 2^N, а получили изменение знака. Это называется *переполнением* (overflow). Подобно тому как вода выливается из переполненного ведра, результат операции может потеряться, если превысит установленную битность.



О ужас! Возникнет ошибка? Этого нельзя так оставлять... И вычислениям мешать будет.

То-то и оно. При возникновении переполнения установится *флаг (бит) переполнения* в регистре состояния (о нём я расскажу на стр. 162). Таким образом, ЦПУ «помнит» о том, что было переполнение.

Вот это да! История о неприятной ситуации, возникшей в аккумуляторе, тщательно записывается в другой регистр. Ошибка будет замечена!

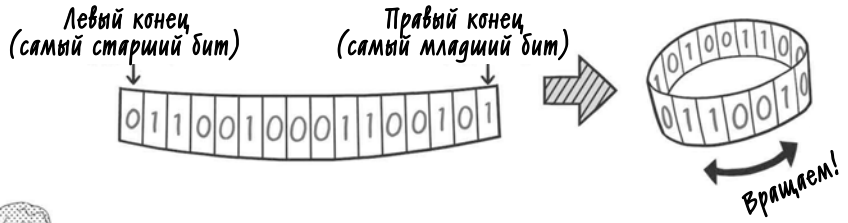
Переполнение и исчезновение порядка

Когда результат вычислений с плавающей точкой выходит за границы диапазона, установленного алгоритмом вычислений, возникает *переполнение* (overflow) или *исчезновение порядка* (underflow). Исчезновением порядка называется ситуация, когда результат — например, 0.000000000000...1 — настолько мал, что не может быть правильно представлен в компьютере.

В этой главе описываются только *целочисленные операции* над сравнительно короткими последовательностями битов, находящимися в аккумуляторе. Операции над целыми числами и над числами с плавающей точкой отличаются друг от друга как в концептуальном, так и в практическом смысле.

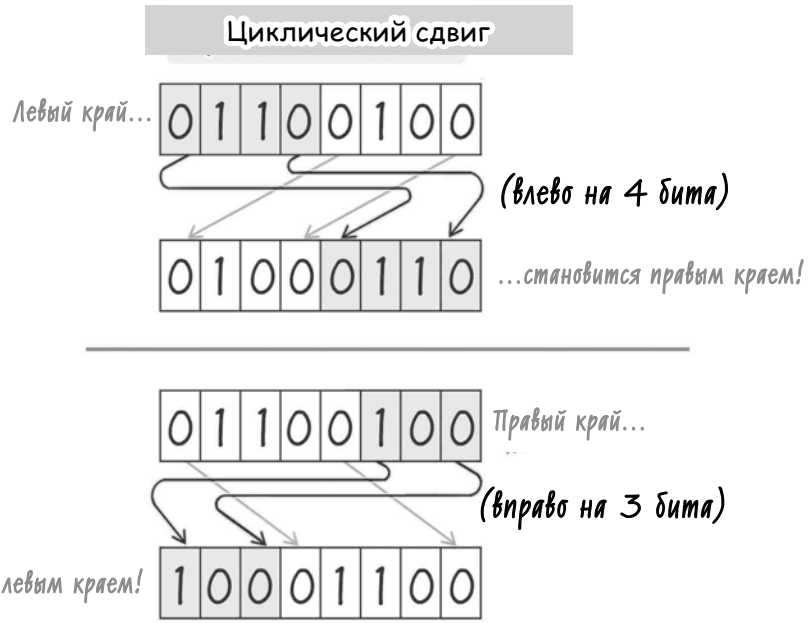
Циклический сдвиг

В завершение скажу о циклическом сдвиге. «Циклический» означает «двигающийся по кругу». Другими словами, можно представить, что последовательность битов в аккумуляторе соединена концами, образуя кольцо, которое вращается.



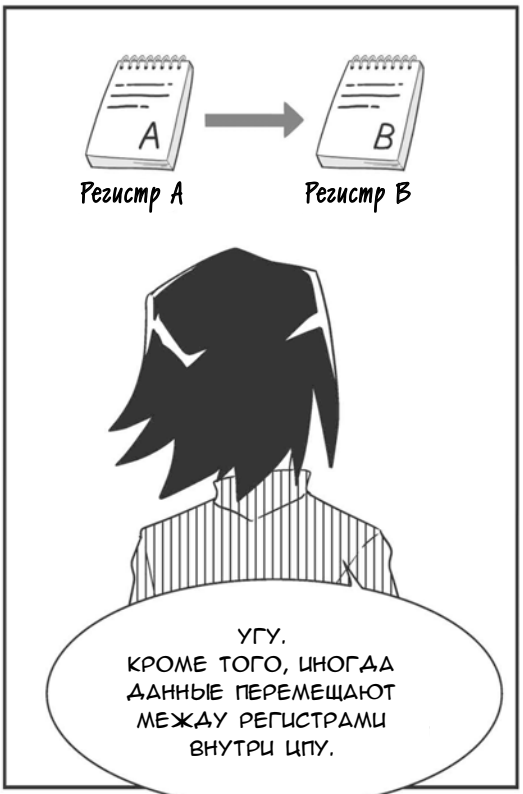
Ого! Концы ленты соединены. В самом деле, кольцо!

Действие циклического сдвига показано на рисунке ниже. Мне кажется, вышеприведённый пример с лентой хорошо его описывает.



Команды пересылки данных

(См. стр. 98)

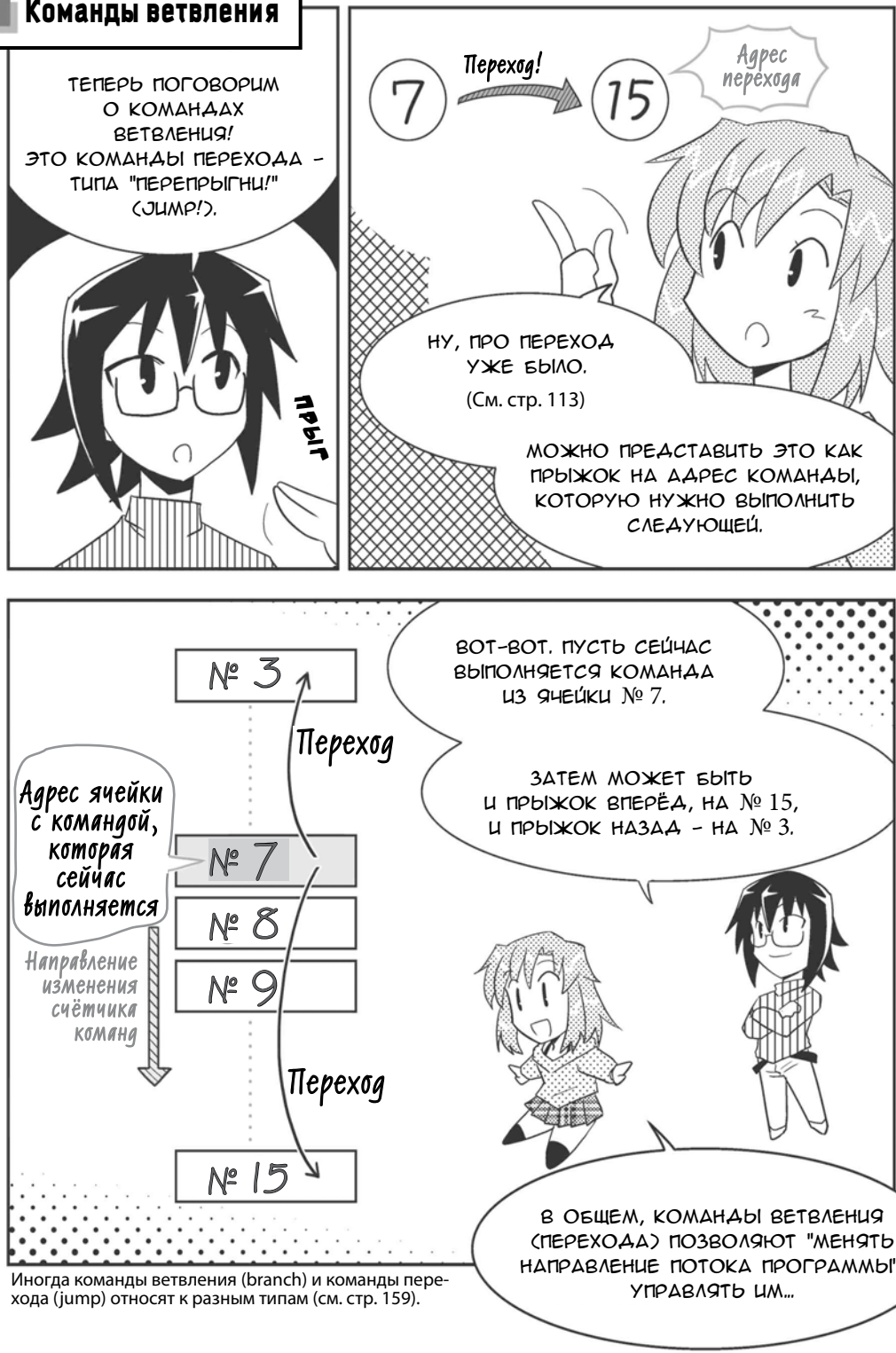


Команды ввода-вывода

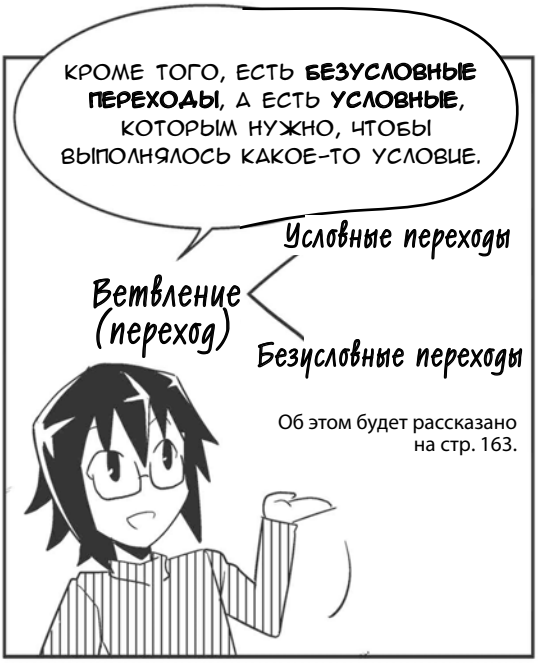
(См. стр. 100)

Передача данных может быть последовательной или параллельной, о чём будет рассказано на стр. 187.

Команды ветвления



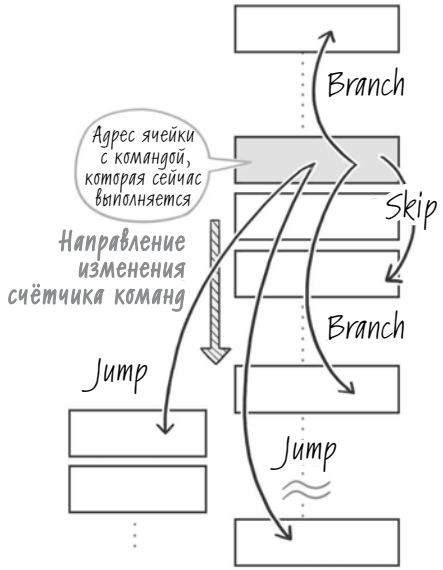
Иногда команды ветвления (branch) и команды перехода (jump) относят к разным типам (см. стр. 159).



Об этом будет рассказано на стр. 163.

Команды ветвления, перехода и пропуска

В зависимости от производителя ЦПУ команды ветвления (перехода) могут называться branch (ветвление), jump (переход) или skip (пропуск); единых правил использования этих терминов не существует. Однако в последнее время их различают согласно следующим определениям.



Определения Branch, Jump и Skip

- Branch: переход вперёд или назад на адрес, расположенный недалеко от выполняемой команды (короткий переход).
- Jump: переход на более дальний адрес по сравнению с командами Branch (длинный переход).
- Skip: выполнение или пропуск одной команды, расположенной сразу после той, которая выполняется сейчас.

Вот как? Разница в чувстве дистанции! Забавно!

К другим командам управления потоком программы относятся, в частности, STOP (Остановка) и SLEEP (Сон).

Про SLEEP будет рассказано на стр. 192.

В мнемонике (символьном обозначении команд — см. стр. 165) восьмибитных процессоров, доминировавших на заре эры ЦПУ, таких как i8080 фирмы Intel или Z80 фирмы Zilog, не было слова Branch. Кроме того, в однокристальном ЦПУ TMS 9900 (16-битном ЦПУ 1976 года выпуска) с той же архитектурой, что и у созданного в 1974 году миникомпьютера компании Texas Instruments (TI), слово Jump означало короткий переход, а Branch — переход по адресу, указанному, например, в регистре.

В популярных с недавних пор ЦПУ для компактной платы Arduino (серия MegaAVR фирмы Atmel) Jump обозначает команду безусловного изменения адреса выполняемой команды; имеется Skip, а также Branch — ветвление относительно адреса выполняемой команды.

Проверка условия и флаг состояния

И напоследок рассмотрим команды проверки условия (например, команды сравнения).

Вспомни пример с банкоматом.

Результат со знаком «ПЛЮС»

Результат со знаком «МИНУС»

Получите наличные!

Недостаточно средств на счёте.

ГР-Р!

(См. стр. 26)

Ох уж эта безжалостная проверка, когда твоя судьба зависит от результата операции сравнения снимаемой суммы с остатком на счёте!..

Да. При этом была выполнена команда сравнения двух значений на больше-меньше и принято решение о соответствии условиям.

Значение A

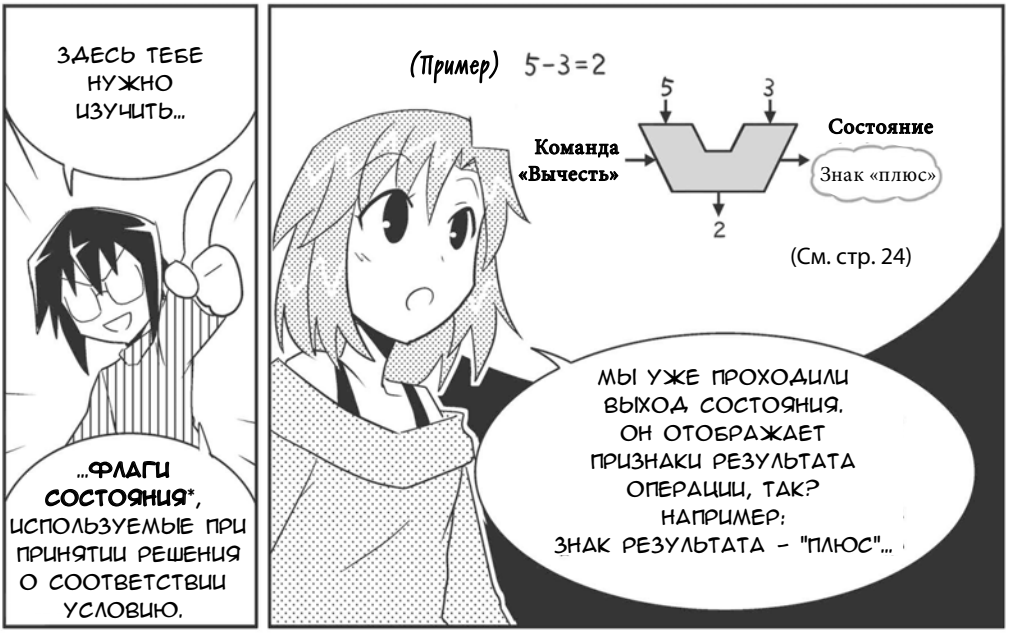
Значение B

Значение A больше? Или нет?

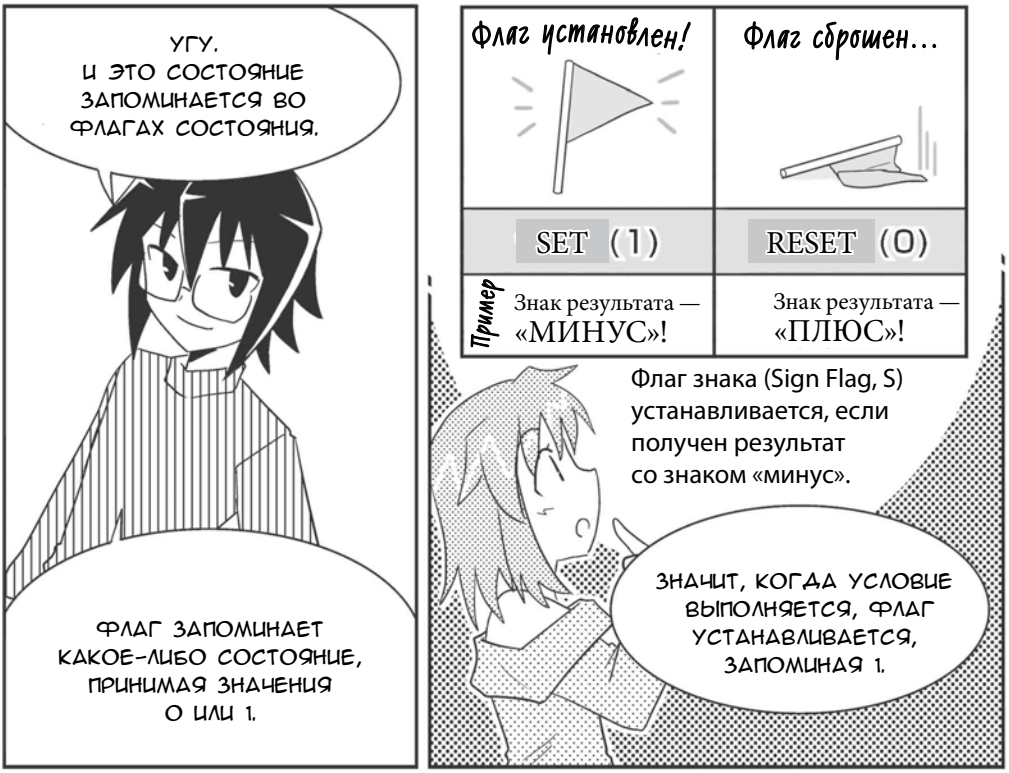
Сравниваю и принимаю решение!

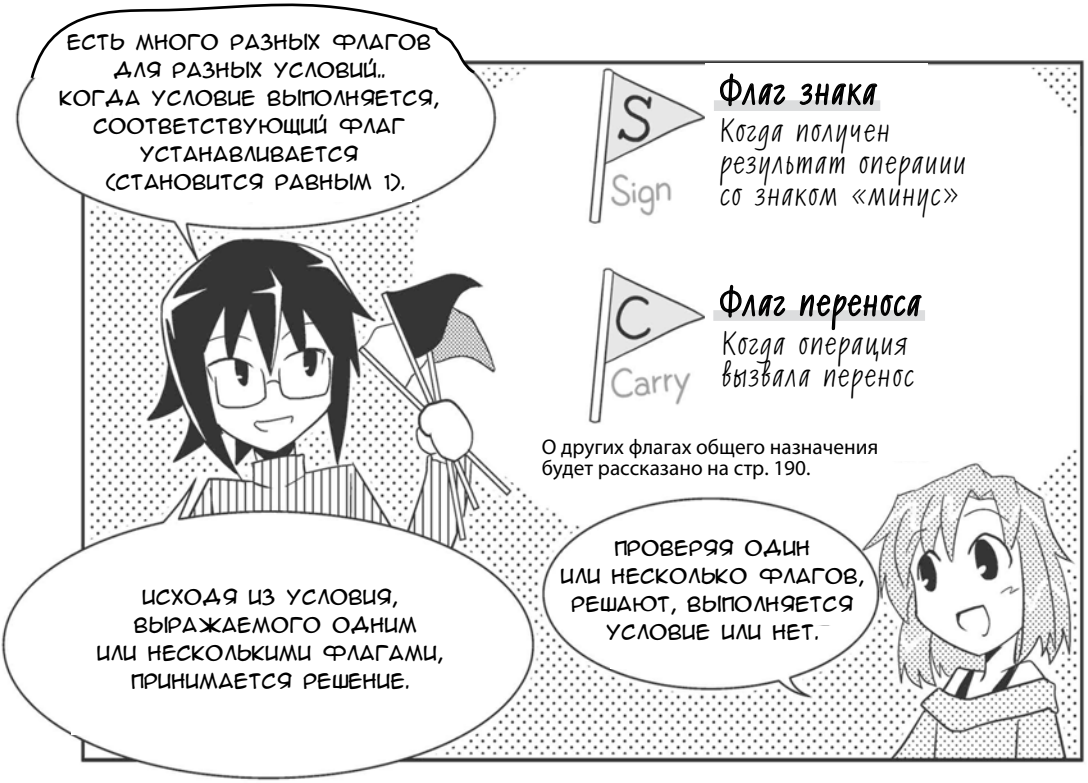
ЦПУ

У-у-у... суровое решение.



* Другое название — биты состояния.




ЕСТЬ МНОГО РАЗНЫХ ФЛАГОВ ДЛЯ РАЗНЫХ УСЛОВИЙ.. КОГДА УСЛОВИЕ ВЫПОЛНЯЕТСЯ, СООТВЕТСТВУЮЩИЙ ФЛАГ УСТАНАВЛИВАЕТСЯ (СТАНОВИТСЯ РАВНЫМ 1).
S
Sign
Флаг знака
Когда получен результат операции со знаком «минус»
C
Carry
Флаг переноса
Когда операция вызвала перенос
О других флагах общего назначения будет рассказано на стр. 190.
ПРОВЕРЯЯ ОДИН ИЛИ НЕСКОЛЬКО ФЛАГОВ, РЕШАЮТ, ВЫПОЛНЯЕТСЯ УСЛОВИЕ ИЛИ НЕТ.
ИСХОДЯ ИЗ УСЛОВИЯ, ВЫРАЖАЕМОГО ОДНИМ ИЛИ НЕСКОЛЬКИМИ ФЛАГАМИ, ПРИНИМАЕТСЯ РЕШЕНИЕ.

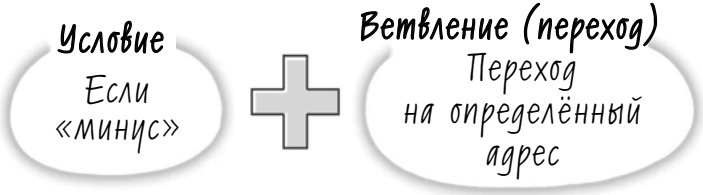

Регистр состояния
… … … … … C … S
Каждый бит означает какое-либо состояние.
Флаг переноса
Флаг знака
(1 или 0)
НАБОР ИЗ 8 ИЛИ 16 ФЛАГОВ НАЗЫВАЕТСЯ РЕГИСТРОМ СОСТОЯНИЯ.


ОГО! РЕГИСТР СОСТОЯНИЯ! ПРЯМО КАК СЫЩИК, КОТОРЫЙ ВСЁ ЗАПОМИНАЕТ.
НУ И ВИДОК У ТЕБЯ!

Соединим ветвление и проверку условия

Итак, мы изучили команды ветвления и команды проверки условия. Если объединить их, мы получим очень полезные команды.

Взять хотя бы такую команду, как Jump on Minus (переход, если «минус»)! Если значение в аккумуляторе отрицательное, то программа продолжит выполнение начиная с адреса перехода.



Значит, если выполнено определённое условие, то произойдёт переход на выбранный адрес... То есть можно изменять порядок выполнения программы в зависимости от условий.

Угу! Команды типа Conditional Jump (условный переход), Conditional Skip (условный пропуск), Conditional Branch (условное ветвление) объединяют ветвление и проверку условия. Они позволяют делать следующее:

Возможности условного перехода

① Выбор выполняемой программы в зависимости от условия.

② Отмена (пропуск) выполнения программы в зависимости от условия.

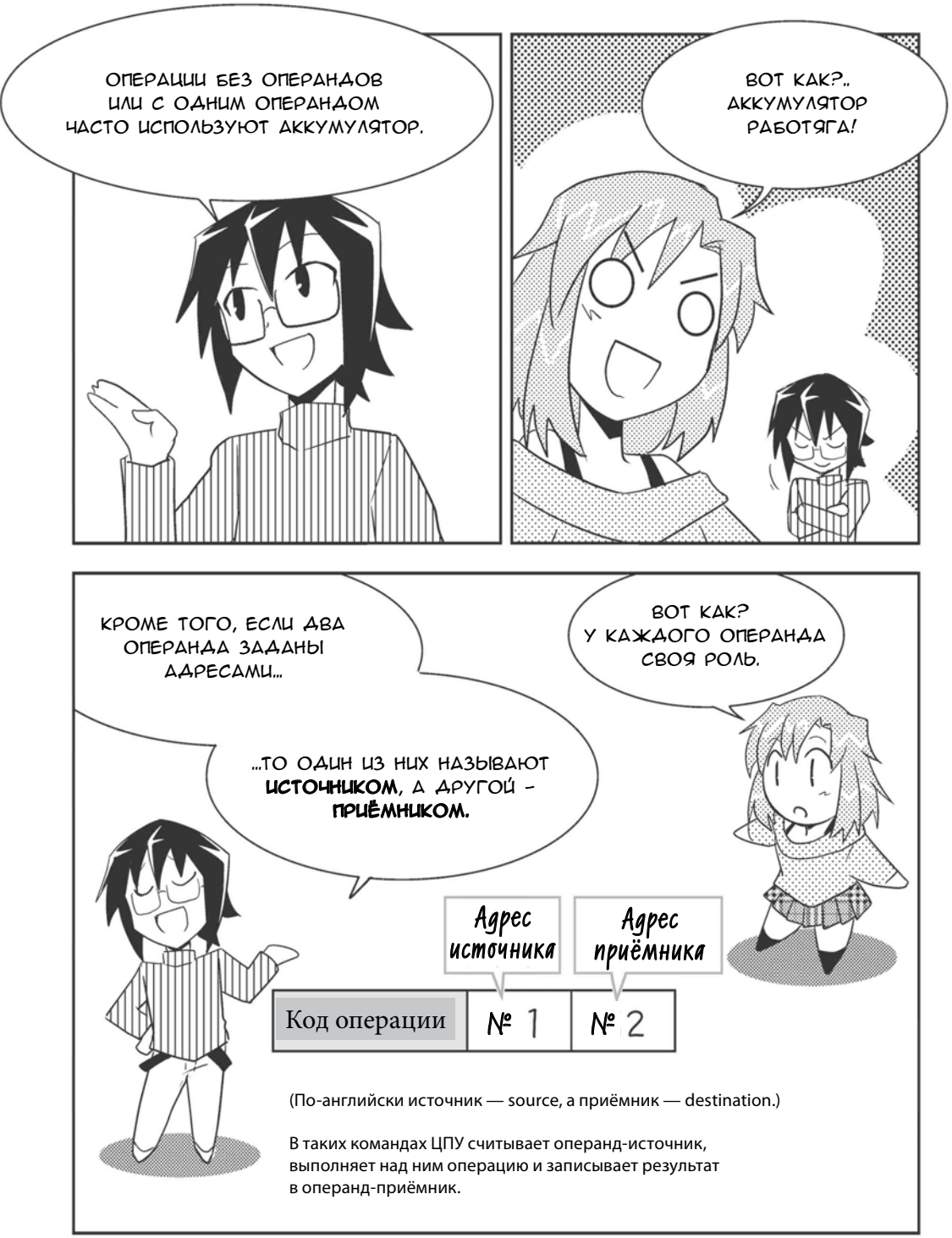
③ Установка или сброс битов портов вывода в зависимости от условия (например, при выполнении определённого условия можно зажигать или гасить светодиод, подключённый к порту вывода).

Ого! Такие команды действительно пригодятся как в ПК, так и в бытовой электронике.

Ч.2. ТИПЫ ОПЕРАНДОВ

Сколько операндов?

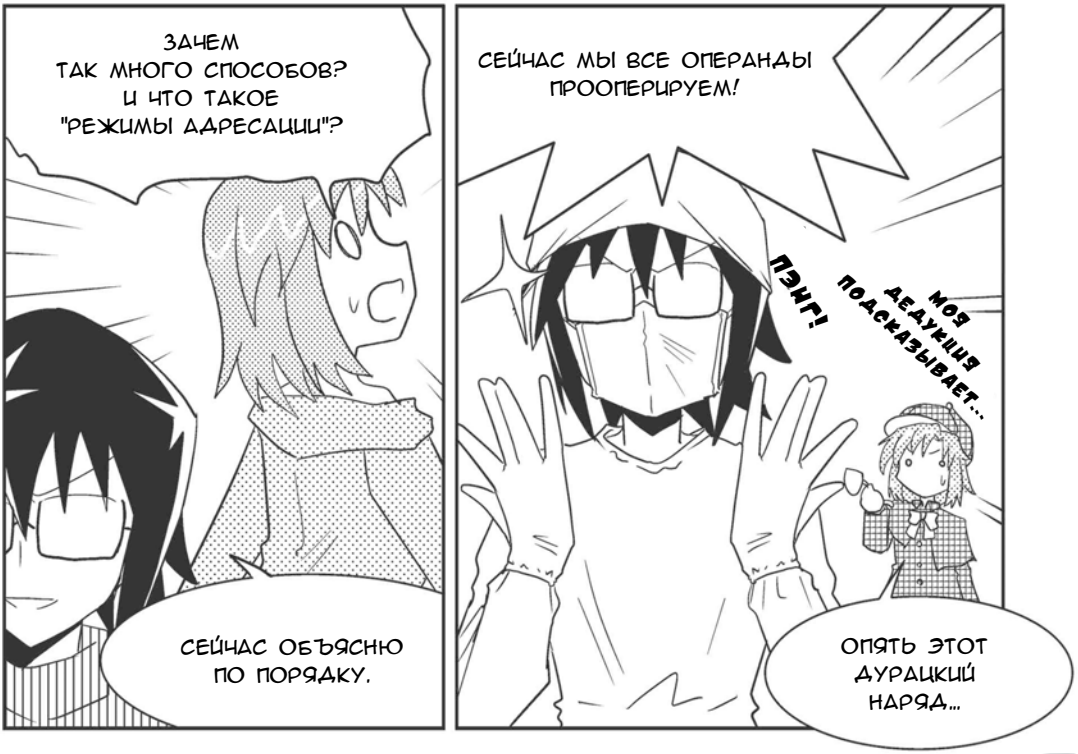
* Мнемоника — облегчение запоминания.



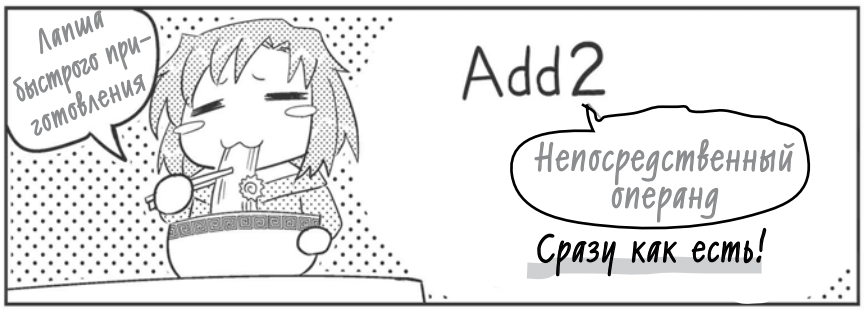
(По-английски источник — source, а приёмник — destination.)

В таких командах ЦПУ считывает операнд-источник, выполняет над ним операцию и записывает результат в операнд-приёмник.

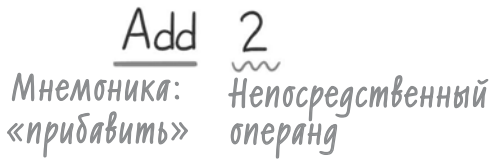

Итак, теперь о главном.
Методы указания операндов
• Непосредственные операнды
• Адресные ссылки
★ Режимы адресации
① Прямая адресация
② Относительная адресация
③ Косвенная адресация
④ Модификация адреса
Смотри! Операнды тоже задаются по-разному!


Зачем так много способов? И что такое "режимы адресации"?
Сейчас объясню по порядку.
Сейчас мы все операнды прооперируем!
Пэнг!
Моя дедукция подсказывает...
Опять этот дурацкий наряд...

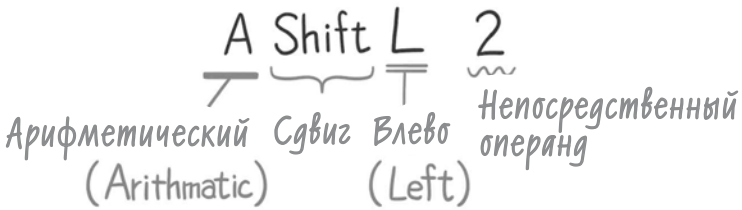
Непосредственные операнды



Начнём с непосредственных операндов. «Непосредственные» означает конкретные числа, используемые сразу как есть.



Это команда прибавить два к значению в аккумуляторе, так?



А это команда арифметического сдвига влево на 2 бита. Непосредственные операнды могут применяться не только в арифметических командах или командах сдвига, но и в других.

Да, с конкретными числами проще... Я запомнила!

Адресные ссылки

Ну, адресные ссылки — это когда указан адрес операнда? Например, ячейка № 1 или там № 2...

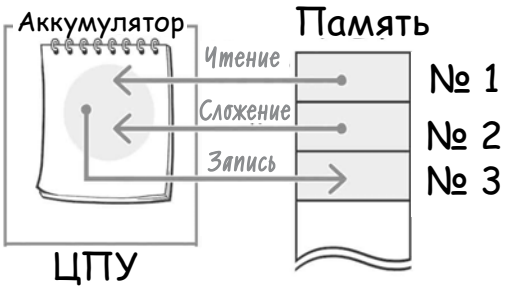
Да. Указывется адрес во внутренней* или внешней памяти, и над значением по этому адресу выполняется операция.

* На схеме классического ЦПУ на стр. 106 показано запоминающее устройство под названием «внутренняя RAM», к которому иногда тоже обращаются по адресам.

А как будут выглядеть, например, команды «сложить значения из ячеек № 1 и № 2 внешней памяти, а результат поместить в ячейку № 3»?

Да вот как.

```
LDA  Address1  (прочитать операнд из ячейки № 1 в аккумулятор)
ADD  Address2  (прибавить операнд из ячейки № 2 к аккумулятору)
STA  Address3  (записать результат из аккумулятора в ячейку № 3)
```



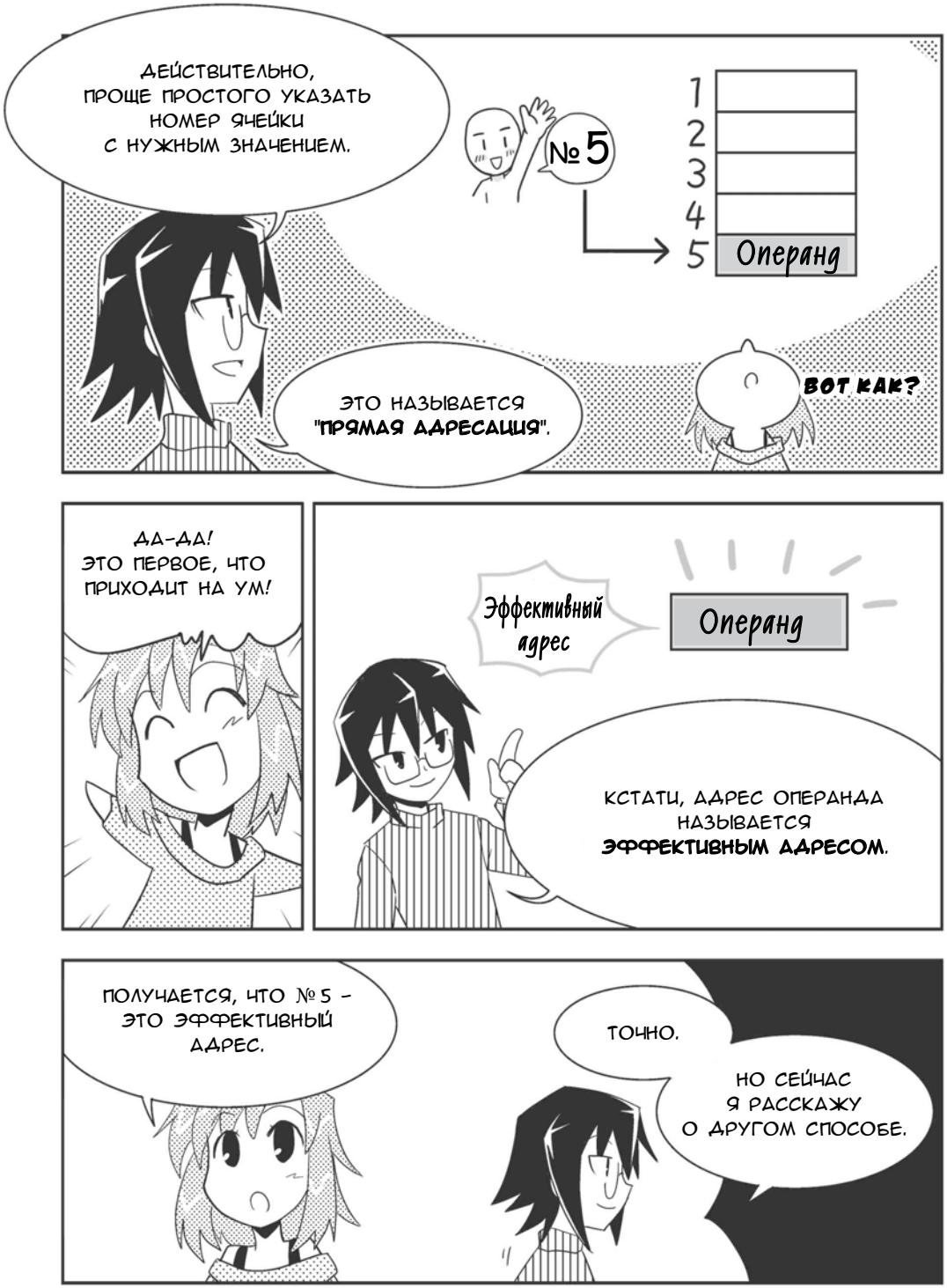
Ого! В вычислениях всегда используется аккумулятор, да?

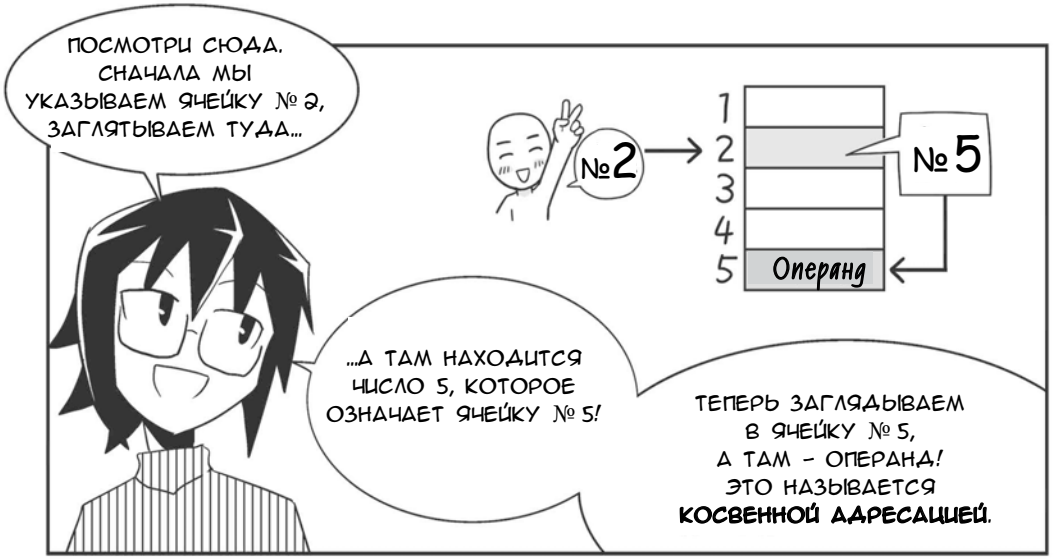
Именно. На это указывает буква A в *мнемоническом коде*. LDA означает Load to Accumulator (записать в аккумулятор), а STA — Store Accumulator (сохранить аккумулятор).

Что такое режимы адресации?

★ Режимы адресации

① Прямая адресация
② Относительная адресация
③ Косвенная адресация
④ Модификация адреса


ДЕЙСТВИТЕЛЬНО, ПРОЩЕ ПРОСТОГО УКАЗАТЬ НОМЕР ЯЧЕЙКИ С НУЖНЫМ ЗНАЧЕНИЕМ.
№5
1
2
3
4
5
Операнд
ЭТО НАЗЫВАЕТСЯ "ПРЯМАЯ АДРЕСАЦИЯ".
ВОТ КАК?
ДА-ДА! ЭТО ПЕРВОЕ, ЧТО ПРИХОДИТ НА УМ!
Эффективный адрес
Операнд
КСТАТИ, АДРЕС ОПЕРАНДА НАЗЫВАЕТСЯ ЭФФЕКТИВНЫМ АДРЕСОМ.
ПОЛУЧАЕТСЯ, ЧТО №5 – ЭТО ЭФФЕКТИВНЫЙ АДРЕС.
ТОЧНО.
НО СЕЙЧАС Я РАССКАЖУ О ДРУГОМ СПОСОБЕ.


ПОСМОТРИ СЮДА. СНАЧАЛА МЫ УКАЗЫВАЕМ ЯЧЕЙКУ № 2, ЗАГЛЯТЫВАЕМ ТУДА...
№2
1
2
3
4
5
№ 5
Операнд
...А ТАМ НАХОДИТСЯ ЧИСЛО 5, КОТОРОЕ ОЗНАЧАЕТ ЯЧЕЙКУ № 5!
ТЕПЕРЬ ЗАГЛЯДЫВАЕМ В ЯЧЕЙКУ № 5, А ТАМ – ОПЕРАНД! ЭТО НАЗЫВАЕТСЯ КОСВЕННОЙ АДРЕСАЦИЕЙ.


К ЧЕМУ ТАКИЕ СЛОЖНОСТИ?!
ЯСНО! ЭТО КАК ЗАВЕЩАНИЕ И СПРЯТАННОЕ СОКРОВИЩЕ!
ХВАТИТ ГЛУПОСТЕЙ!

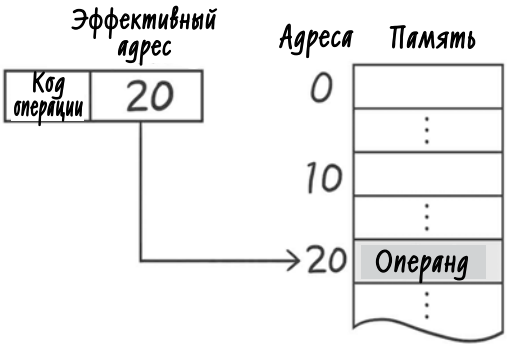

ЭТОТ СПОСОБ ИНОГДА ПОЛЕЗЕН. ЕСЛИ АДРЕС ОПЕРАНДА СЛИШКОМ ДЛИННЫЙ, ВРОДЕ № 9999...9...
Плохо!
Дли-и-и-инный...
Код операции
Адрес операнда
...
Длина поля операндов ограничена!
...ТО ОН НЕ УМЕСТИТСЯ В ПОЛЕ ОПЕРАНДОВ..


ЯСНО!..
КОСВЕННАЯ АДРЕСАЦИЯ ПОЗВОЛЯЕТ ЗАДАТЬ В ПОЛЕ ОПЕРАНДОВ КОРОТКИЙ АДРЕС ДЛИННОГО АДРЕСА ОПЕРАНДА.
Ограничения!
Код операции
Допустимые режимы адресации
КРОМЕ ТОГО, КОД ОПЕРАЦИИ ИНОГДА НАКЛАДЫВАЕТ ОГРАНИЧЕНИЯ НА ВОЗМОЖНЫЕ РЕЖИМЫ АДРЕСАЦИИ.
КАКИЕ-ТО ОБСТОЯТЕЛЬСТВА?
УДИВИТЕЛЬНО...
РАЗНООБРАЗИЕ РЕЖИМОВ АДРЕСАЦИИ ПОЗВОЛЯЕТ ЦПУ ВЫПОЛНЯТЬ СЛОЖНЫЕ ПРОГРАММЫ.
Я, ТАЛАНТЛИВЫЙ ПРОГРАММИСТ, ЗНАЮ ИХ ВСЕ. НО ДЛЯ ТЕБЯ ОНИ, НАВЕРНОЕ, БУДУТ СЛОЖНОВАТЫ.
НИЧЕГО СЛОЖНОГО! ГОВОРИ! Я СРАЗУ ЖЕ ПОЙМУ!
ХОРОШО. ПРИВЕДУ ОПИСАНИЯ ВСЕХ РЕЖИМОВ АДРЕСАЦИИ.

Сводка режимов адресации

① Прямая адресация

В поле операндов содержится эффективный адрес (адрес ячейки, содержащей операнд). Иногда этот режим называют *абсолютной адресацией*.

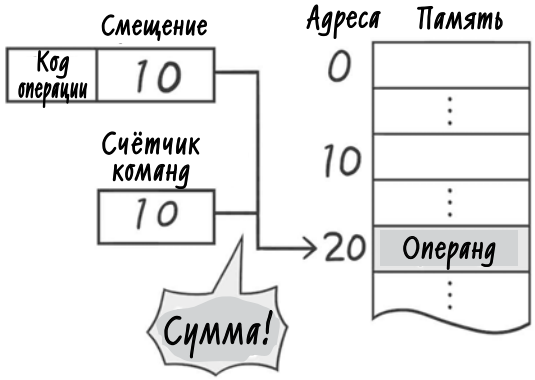


Ещё подробней!

В некоторых ЦПУ длины одной команды не хватает для представления всего пространства адресов, поэтому поле операндов удлиняют на несколько байтов. В 16-битных ЦПУ длина команды (код операции + поле операндов) обычно равна 16 битам (2 байтам), но иногда её удлиняют до 4 или 8 байт, расширяя поле операндов.

② Относительная адресация

В этом режиме эффективный адрес операнда получают прибавлением смещения в поле операнда к текущему значению счётчика команд.



Ещё подробней!

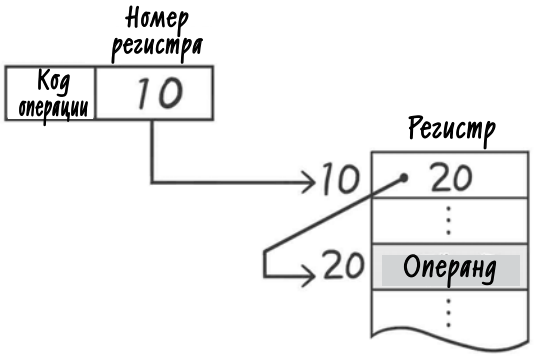
Обычно этот режим применяется в командах ветвления (перехода). Он используется только для небольших изменений адреса, например в командах условного ветвления, так как диапазон адресации ограничен количеством значащих битов поля операнда.

В качестве базового адреса используется текущее значение счётчика команд, указывающее на адрес следующей команды.

Для относительной адресации может быть использован не только счётчик команд, но и один из других регистров.

③ Косвенная адресация

В этом методе в поле операндов указывается номер регистра, содержащего эффективный адрес операнда.

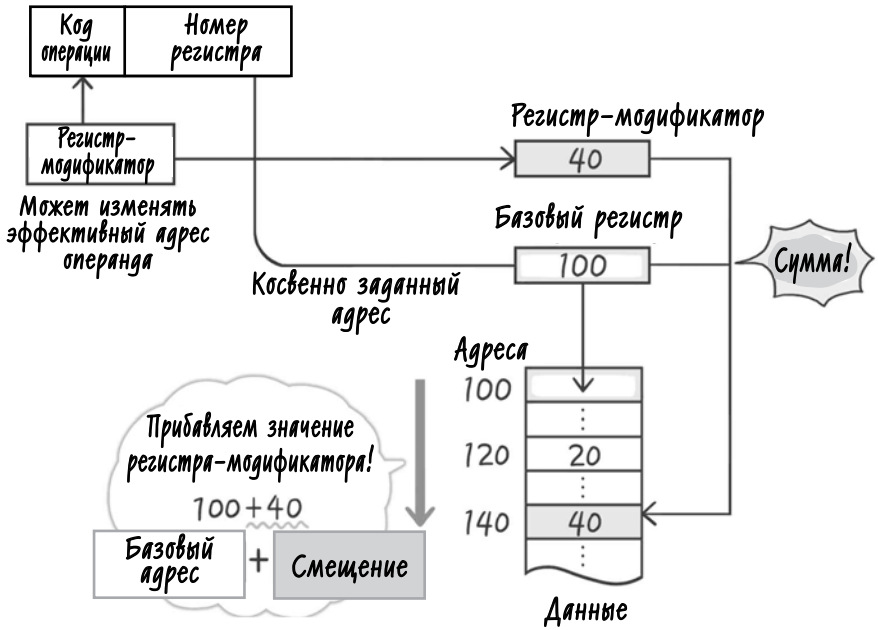


Ещё подробней!

Этот метод, используемый совместно с модификацией адреса, тесно связан с массивами в языке С и ему подобных языках программирования. Во многих случаях полученный эффективный адрес используется в качестве базового для режима модификации адреса.

④ Модификация адреса

В этом режиме эффективный адрес равен сумме базового адреса и смещения в *регистре-модификаторе*. Базовый адрес может содержаться в счётчике команд, одном из регистров или в поле операндов команды.



Ещё подробней!

Обычно регистр-модификатор называют *индексным регистром*, а регистр, хранящий базовый адрес, — *базовым регистром*.

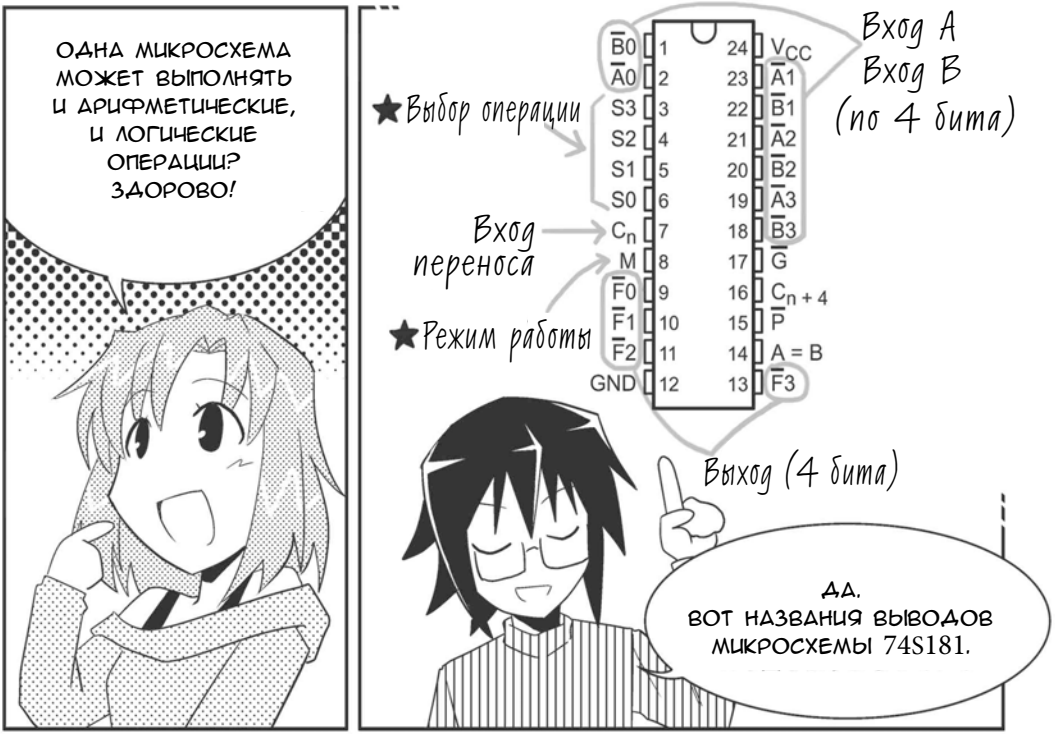
Во многих ЦПУ эффективный адрес получают сложением значений в базовом и индексном регистрах.

Этот режим адресации удобен тем, что позволяет извлекать из набора данных значения по их номерам.

Ч.3. КАК АЛУ ВЫПОЛНЯЕТ ОПЕРАЦИИ?

Заглянем внутрь АЛУ

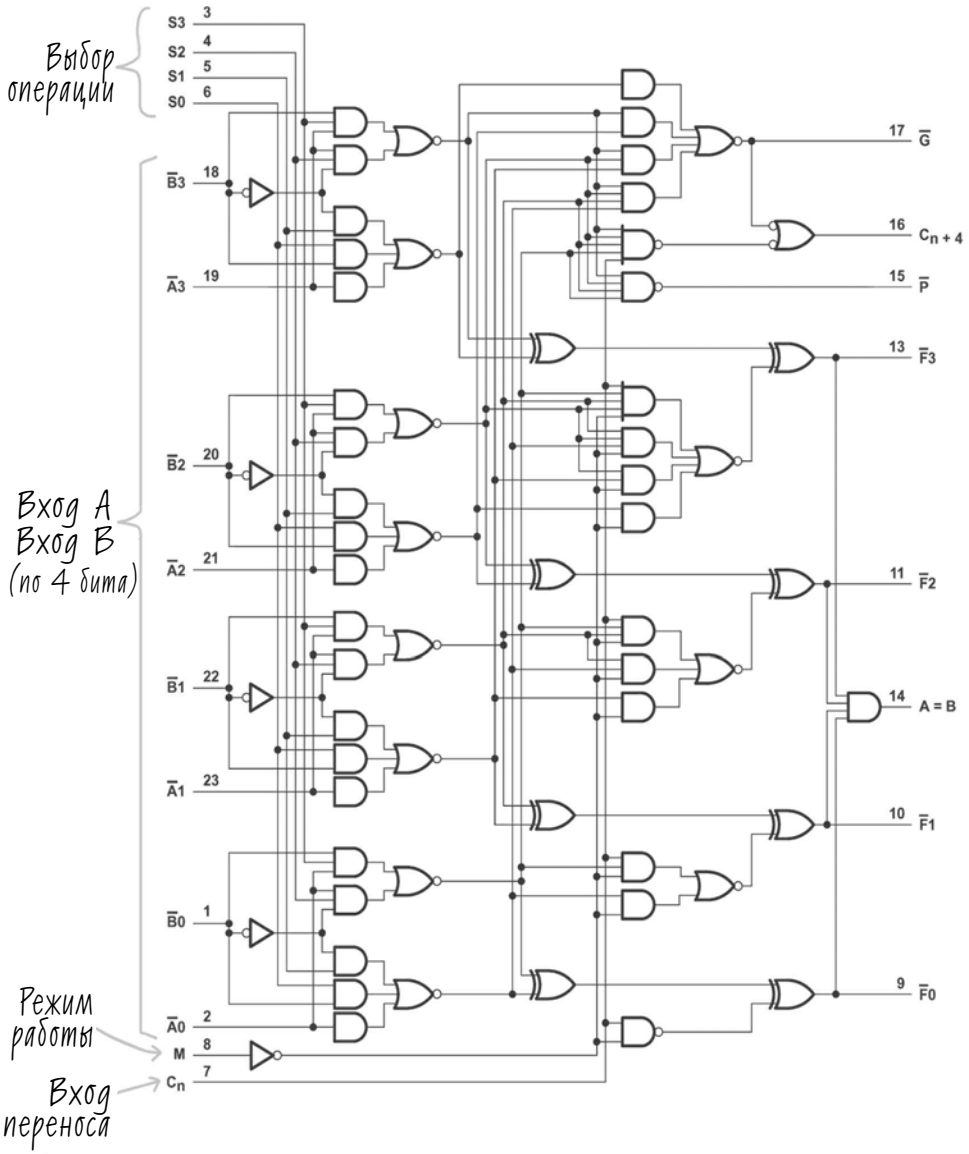
* В мини-компьютере компании TI (см. стр. 159) использовались четыре микросхемы 74S181. В те времена он считался быстродействующим мини-компьютером и использовался даже, например, в авиационных тренажёрах. Выпускался также упрощённый вариант микросхемы — 74S381.


ОДНА МИКРОСХЕМА МОЖЕТ ВЫПОЛНЯТЬ И АРИФМЕТИЧЕСКИЕ, И ЛОГИЧЕСКИЕ ОПЕРАЦИИ? ЗДОРОВО!
Выбор операции
Вход переноса
Режим работы
Вход A
Вход B
(по 4 бита)
B0 1
A0 2
S3 3
S2 4
S1 5
S0 6
Cn 7
M 8
F0 9
F1 10
F2 11
GND 12
24 VCC
23 A1
22 B1
21 A2
20 B2
19 A3
18 B3
17 G
16 Cn + 4
15 P
14 A = B
13 F3
Выход (4 бита)
ДА. ВОТ НАЗВАНИЯ ВЫВОДОВ МИКРОСХЕМЫ 74S181.


ВЫВОД "РЕЖИМ РАБОТЫ" ОПРЕДЕЛЯЕТ ТИП ОПЕРАЦИИ – АРИФМЕТИЧЕСКАЯ ИЛИ ЛОГИЧЕСКАЯ.
ВЫВОД "ВЫБОР ОПЕРАЦИИ" ОПРЕДЕЛЯЕТ КОНКРЕТНУЮ ОПЕРАЦИЮ.
ЯСНО. ЭТО КАК РЕЖИМЫ В КОНДИЦИОНЕРЕ – ЛИБО ОХЛАЖДЕНИЕ, ЛИБО ОБОГРЕВ.
ТЕПЕРЬ ПРИВЕДУ ПРИНЦИПИАЛЬНУЮ СХЕМУ И ТАБЛИЦУ ОПЕРАЦИЙ ЭТОГО АЛУ. И НА ЭТОМ МЫ СЕГОДНЯ ЗАКОНЧИМ.

Принципиальная схема 74S181

(*На основе технического паспорта компании Texas Instruments с изменениями*)



М-да-а! Сложная схема, но можно увидеть входы A и B, а также четыре вывода «Выбор операции» S_0 — S_3 и вывод «Режим работы» M.

Угу... А ещё вход переноса C_n.

Таблица операций 74S181

Выбор операции				Сигнал с активным высоким уровнем		
				M = H	M = L; Арифметические операции	
S3	S2	S1	S0	Логические операции	C_n = H (без переноса)	C_n = L (с переносом)
L	L	L	L	F = $\overline{A}$	F = A	F = A PLUS 1
L	L	L	H	F = $\overline{A + B}$	F = A + B	F = (A+ B) PLUS 1
L	L	H	L	F = $\overline{A}$B	F = A + $\overline{B}$	F = (A + $\overline{B}$) PLUS 1
L	L	H	H	F = 0	F = MINUS 1 (2's COMPL)	F = ZERO
L	H	L	L	F = $\overline{AB}$	F = A PLUS A$\overline{B}$	F = A PLUS A$\overline{B}$ PLUS 1
L	H	L	H	F = $\overline{B}$	F = (A + B) PLUS A$\overline{B}$	F = (A + B) PLUS A$\overline{B}$ PLUS 1
L	H	H	L	F = A ⊕ B	F = A MINUS B MINUS 1	F = A MINUS B
L	H	H	H	F = A$\overline{B}$	F = A$\overline{B}$ MINUS 1	F = A $\overline{B}$
H	L	L	L	F = $\overline{A}$ + B	F = A PLUS AB	F = A PLUS AB PLUS 1
H	L	L	H	F = $\overline{A \oplus B}$	F = A PLUS B	F = A PLUS B PLUS 1
H	L	H	L	F = B	F = (A + $\overline{B}$) PLUS AB	F = (A + $\overline{B}$) PLUS AB PLUS 1
H	L	H	H	F = AB	F = AB MINUS 1	F = AB
H	H	L	L	F = 1	F = A PLUS A†	F = A PLUS A PLUS 1
H	H	L	H	F = A + $\overline{B}$	F = (A + B) PLUS A	F = (A + B) PLUS A PLUS 1
H	H	H	L	F = A + B	F = (A + $\overline{B}$) PLUS A	F =(A + $\overline{B}$) PLUS A PLUS 1
H	H	H	H	F = A	F = A MINUS 1	F = A

† Каждый из битов сдвигается в следующую старшую позицию.

Обозначения в формулах приводятся на стр. 55–59.

MINUS и PLUS — это арифметические операции вычитания и сложения соответственно. Знаки +, $\overline{\ }$ и ⊕ в таблице означают логические операции. Показаны также дублирующие или избыточные операции, возникшие как «побочные эффекты» на этапе проектирования микросхемы.

Серым фоном в таблице выделены важные выводы схемы. Во-первых, M — это вывод «Режим работы». Если M = H, то это режим логических операций, а если M = L — арифметических.

Арифметические операции могут выполняться с переносом или без него. Если C_n = H, то переноса нет, а если C_n = L, значит, он есть.

Имеются четыре вывода S «Выбор операции»... Разные сочетания их уровней (всего 16) позволяют выбирать разные операции!

Объясню про самые важные коды операций из этой таблицы*. Для удобства 16 операциям назначены коды от 0 до 15, но в ЦПУ других типов для тех же операций могут использоваться другие коды.

	Логические операции	Арифметические операции	
		Без переноса	С переносом
0	$F = \overline{A}$	F = A	F = A PLUS 1
1	$F = \overline{A + B}$	F = A + B	F = (A+ B) PLUS 1
2	$F = \overline{A}B$	$F = A + \overline{B}$	$F = (A + \overline{B})$ PLUS 1
3	F = 0	F = MINUS 1 (2's COMPL)	F = ZERO
4	$F = \overline{AB}$	$F = A$ PLUS $A\overline{B}$	$F = A$ PLUS $A\overline{B}$ PLUS 1
5	$F = \overline{B}$	$F = (A + B)$ PLUS $A\overline{B}$	$F = (A + B)$ PLUS $A\overline{B}$ PLUS 1
6	$F = A \oplus B$	F = A MINUS B MINUS 1	F = A MINUS B
7	$F = A\overline{B}$	$F = A\overline{B}$ MINUS 1	$F = A\,\overline{B}$
8	$F = \overline{A} + B$	F = A PLUS AB	F = A PLUS AB PLUS 1
9	$F = \overline{A \oplus B}$	F = A PLUS B	F = A PLUS B PLUS 1
10	F = B	$F = (A + \overline{B})$ PLUS AB	$F = (A + \overline{B})$ PLUS AB PLUS 1
11	F = AB	F = AB MINUS 1	F = AB
12	F = 1	F = A PLUS A	F = A PLUS A PLUS 1
13	$F = A + \overline{B}$	F = (A + B) PLUS A	F = (A + B) PLUS A PLUS 1
14	F = A + B	$F = (A + \overline{B})$ PLUS A	$F =(A + \overline{B})$ PLUS A PLUS 1
15	F = A	F = A MINUS 1	F = A

* Операции, выделенные серым, будут объясняться ниже.

Важные арифметические команды

<Код операции: 6>

Без переноса . . . Сначала из A вычитается B, затем из результата вычитается 1.

С переносом . . . Из A вычитается B.

<Код операции: 9>

Без переноса . . . Вычисляется сумма A и B.

С переносом . . . Сначала вычисляется сумма A и B, затем к результату прибавляется 1.

Важные логические команды

<Код операции: 1> . . . NOR (A, B)

Сначала выполняется побитовое ИЛИ (OR) над A и B, затем результат подвергается побитовому отрицанию (NOT). Другими словами, выполняется операция NOR над парами соответствующих битов A и B.

<Код операции: 3> . . . ZERO

Результат F равен 0 вне зависимости от A и B.

<Код операции: 4> . . . NAND (A, B)

Сначала выполняется побитовое И (AND) над A и B, затем результат подвергается побитовому отрицанию (NOT). Другими словами, выполняется операция NAND над парами соответствующих битов A и B.

<Код операции: 5> . . . NOT (B)

Выполняется побитовое отрицание (NOT) операнда B. Другими словами, все биты B инвертируются.

<Код операции: 6> . . . XOR (A, B)

Выполняется побитовое исключающее ИЛИ (XOR) над A и B.

<Код операции: 9> . . . XNOR (A, B)

Сначала выполняется побитовое исключающее ИЛИ (XOR) над A и B, затем результат подвергается побитовому отрицанию (NOT).

<Код операции: 10> . . . (B)

Результат операции F равен B.

<Код операции: 11> . . . AND (A, B)

Выполняется побитовое логическое И (AND) над A и B.

<Код операции: 12> . . . ONEs

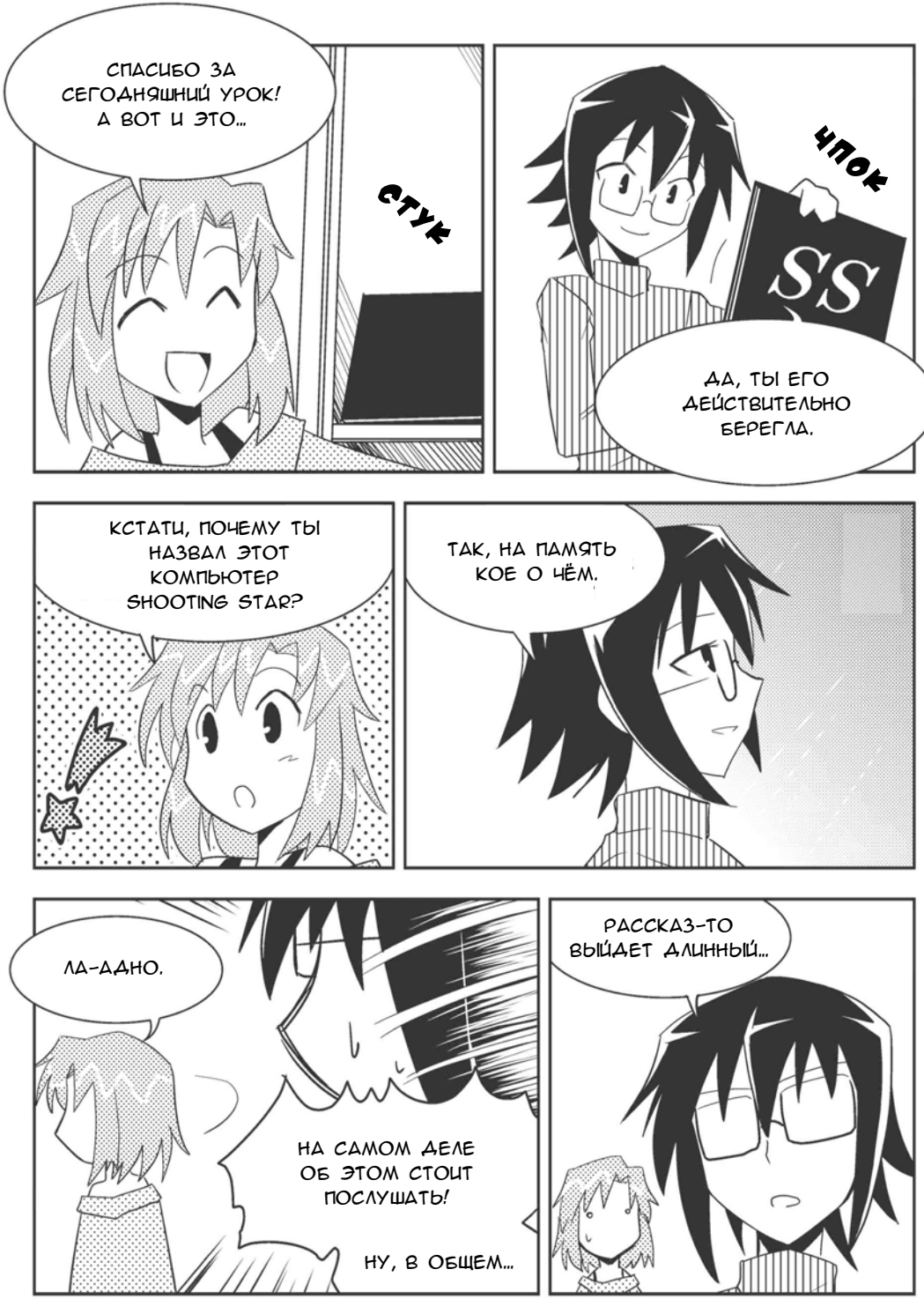
Все биты результата F устанавливаются в 1 независимо от A и B.

<Код операции: 14> . . . OR (A, B)

Выполняется побитовое ИЛИ (OR) над A и B.

<Код операции: 10> . . . (A)

Результат операции F равен A.


СПАСИБО ЗА СЕГОДНЯШНИЙ УРОК! А ВОТ И ЭТО...
СТУК
ЧПОК
SS
ДА, ТЫ ЕГО ДЕЙСТВИТЕЛЬНО БЕРЕГЛА.
КСТАТИ, ПОЧЕМУ ТЫ НАЗВАЛ ЭТОТ КОМПЬЮТЕР SHOOTING STAR?
ТАК, НА ПАМЯТЬ КОЕ О ЧЁМ.
ЛА-АДНО.
НА САМОМ ДЕЛЕ ОБ ЭТОМ СТОИТ ПОСЛУШАТЬ!
НУ, В ОБЩЕМ...
РАССКАЗ-ТО ВЫЙДЕТ ДЛИННЫЙ...

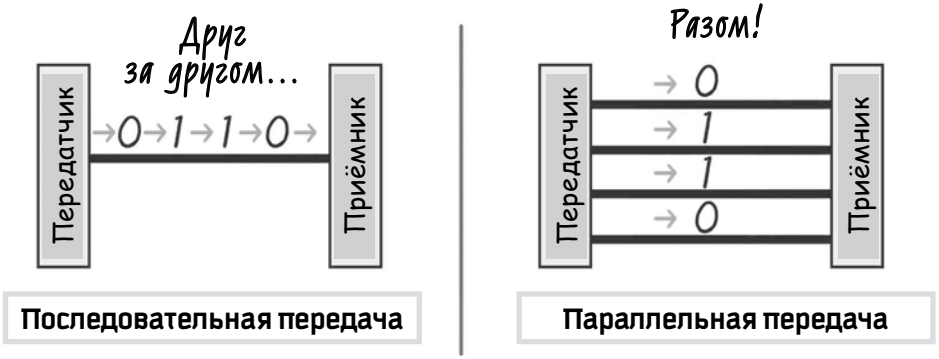

Я ДОЛГО ЖИЛ ЗА ГРАНИЦЕЙ. МЫ ПЕРЕЕХАЛИ, ПОТОМУ ЧТО ОТЕЦ НАШЁЛ ТАМ РАБОТУ.
В ЯПОНИЮ ВЕРНУЛИСЬ НЕДАВНО.
ПРОДОЛЖАЙ... ЛЮБЛЮ ВСЯКИЕ БАЙКИ!
ПОНИМАЮ...
ЭТО НЕ "БАЙКА"! ЭТО ЧИСТАЯ ПРАВДА!
НУ, ЕСЛИ ПРАВДА...
ЖИЗНЬ ЗА ГРАНИЦЕЙ. ОДИНОЧЕСТВО. ТОСКА ПО РОДИНЕ.
МОЯ МИЛАЯ ЯПОНИЯ...
ЮНОША СОЗДАЁТ ТРАДИЦИОННУЮ ИГРУ.
НЕВОЗМОЖНО СДЕРЖАТЬ СЛЁЗ!..
А-А-А!
НУ ТЫ И НАВООБРАЖАЛА!
ТУК-ТУК...
АЮМИ, ПОЗНАКОМЬ МЕНЯ СО СВОИМ ГОСТЕМ ♪
ЩЁЛК


……!!
МАМА!
ТЫ ТОЛЬКО НИЧЕГО
НЕ ПОДУМАЙ!
ОН ПРОСТО…
А-А, ЭТО ТЫ, Ю-КУН!
ДАВНО НЕ ВИДЕЛИСЬ!
ПОКЛОН
ИЗВИНИТЕ, ЧТО
НЕ ПИСАЛ ВАМ.
ДА, Я СЛЫШАЛА,
ЧТО ТЫ ЖИЛ
ЗА ГРАНИЦЕЙ.
ЛЕТ ДЕСЯТЬ ПРОШЛО
С ТЕХ ПОР,
КАК ВЫ С АЮМИ
ИГРАЛИ ВМЕСТЕ…
ОЙ…
ЧТО?!
ПОЧЕМУ?!
НЕ МОЖЕТ
БЫТЬ!

ДОПОЛНИТЕЛЬНАЯ ИНФОРМАЦИЯ

◆ Последовательная и параллельная передача

Существует два метода передачи данных (цифровых сигналов): *последовательный* (Serial) и *параллельный* (Parallel).



При последовательной передаче биты передаются один за другим, а при параллельной по нескольку бит передаются за один раз. Кстати, USB (USB-память, USB-подключение) расшифровывается как Universal Serial Bus (универсальная последовательная шина) и подразумевает последовательную передачу.

Регистр сдвига и параллельно-последовательное преобразование

Функциональные блоки логических схем иногда оснащают *регистром сдвига*. Он не выполняет функции аккумулятора внутри ЦПУ, а просто выполняет сдвиг.

Наиболее широко этот регистр применяется в функциональных блоках, сдвигающих параллельные (например, 8-битные) данные на 1 бит с определённой периодичностью, извлекая при этом крайний правый бит и формируя данные для последовательной передачи.

(Считать ли такие блоки для последовательной передачи частью ЦПУ или же относить их к устройствам I/O? Мнения по этому вопросу могут различаться, но здесь мы отнесём эти функциональные блоки, отвечающие за передачу информации, к устройствам ввода-вывода и не будем рассматривать их как часть самого ЦПУ.)

◆ Обзор основных регистров

Регистры выполняют разнообразные функции и являются незаменимой частью ЦПУ. Ниже приведены функции основных регистров (правда, в названии некоторых из них нет слова «регистр»).

- **Аккумулятор**

Регистр, запоминающий результат операции АЛУ. Он позволяет сразу же использовать этот результат для следующей операции. Название «аккумулятор» происходит от слова *accumulation* (накопление), так как с его помощью можно накапливать (суммировать) результаты последовательных вычислений.

- **Регистр команд, декодер команд**

Защёлкивает машинную команду, прочитанную из памяти, и переводит её в код операции, интерпретируя, какую именно операцию и над какими операндами нужно выполнить.

Кстати, количество кодов ограничено числом операций АЛУ, которое очень мало (например, вышеописанная IC 74S181 выполняет 16 арифметических и 16 логических операций), поэтому обычно используют внутреннюю программу управления АЛУ, позволяющую комбинировать операции, быстро выполняя их в непрерывном потоке.

- **Регистр состояния**

В зависимости от результата очередной операции ЦПУ можно изменять порядок выполнения программы или осуществлять ввод-вывод. При этом для принятия решения используются флаги (по сути, однобитные маркеры). Эти флаги объединяют в 8-битном или 16-битном регистре состояния. Есть много разных флагов, и с ними я познакомлю вас на стр. 190.

- **Регистры-модификаторы (базовый регистр, индексный регистр)**

Используются в некоторых режимах адресации. Базовый регистр содержит базовый адрес, от которого отсчитываются смещения операндов, указываемые в командах.

В индексный регистр можно записать константу, с помощью которой ЦПУ будет вычислять эффективные адреса операндов, складывая её с текущими значениями счётчика команд.

- **Регистр TEMP (регистр временного запоминания)**

Используется для временной "эвакуации" и запоминания данных в процессе работы ЦПУ. Некоторые ЦПУ содержат по одному такому регистру в нескольких функциональных блоках. (Хотя в этой книге регистр TEMP не рассматривается, он изображён на Концептуальной схеме классического ЦПУ на стр. 106).

Нижеприведённые устройства тоже можно назвать регистрами. Как было показано на стр. 82, счётчики по своей конструкции относятся к регистрам.

- **Счётчик команд (Program Counter, PC)**

Регистр, запоминающий адрес (номер ячейки) следующей команды. Имеется в любых ЦПУ.

- **Указатель стека (Stack Pointer, SP)**

Необходим при использовании стека. Запоминает адрес (номер ячейки) вершины стека.

Основные флаги состояния

В ЦПУ есть флаги (биты) состояния, которые устанавливаются или сбрасываются в зависимости от результата операции. На основе одного или нескольких условий, выражаемых флагами, ЦПУ принимает различные решения, обеспечивающие ветвление программы и т. п.

- **Флаг нуля (Zero Flag, флаг Z)**

Устанавливается, если результат операции, содержащийся в аккумуляторе, равен нулю. При отсутствии в ЦПУ компаратора (блока сравнения) также используется в качестве флага EQ (equal, «равно») для операций сравнения, который устанавливается при равенстве сравнимаемых значений.

- **Флаг знака (Sign Flag, флаг S)**
  или флаг отрицательного результата (Negative Flag, флаг N)

Устанавливается, если результат операции, содержащийся в аккумуляторе, является отрицательным числом.

- **Флаг переноса (Carry Flag, флаг C)**
  или флаг переполнения (Overflow Flag, флаг OV)

Устанавливаются, если при арифметическом сложении произошли перенос из старшего разряда или переполнение.

- **Флаг заёма (Borrow Flag, флаг B)**

Устанавливается, если при вычитании произошёл заём. Вместо этого флага для проверки на заём часто используют флаг C. Если после вычитания флаг C сброшен (т. е. равен 0), значит, имел место заём.

- **Флаг «больше, чем» (Greater Than Flag, флаг GT)**

Устанавливается, если в результате операции сравнения получилось «больше». Соответствует математическому знаку > (больше).

- **Флаг «меньше, чем» (Less Than Flag, флаг LT)**

Устанавливается, если в результате операции сравнения получилось «меньше». Соответствует математическому знаку < (меньше).

- **Флаг нечётности (Oddity Flag, флаг OD)**

Показывает, что результат операции содержит нечётное число единиц.

- **Маска прерываний**

Определяет, какие прерывания будут обрабатываться. Установка бита означает, что соответствующее прерывание запрещено (замаскировано).

- **Флаг прерывания (Interrupt Flag, флаг I)**

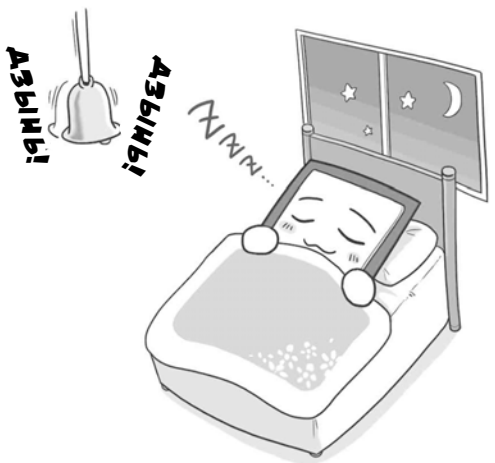
Показывает, что возникло прерывание, даже в тех случаях, если соответствующее прерывание запрещено.

◆ Команда SLEEP

Среди команд, управляющих потоком программы, кроме команд перехода (JUMP) есть и такие команды, как STOP и SLEEP.

Команда SLEEP останавливает работу ЦПУ, то есть приостанавливает выполнение программы до момента появления на входе каких-либо данных (до возникновения прерывания). Подобная функция имеется также в операционных системах персональных компьютеров.

Эта команда может использоваться для понижения рабочей частоты ЦПУ или приостановки выполнения программы в целях энергосбережения. Для возврата в нормальный режим работы необходимо вызвать прерывание путём, например, нажатия кнопки на каком-либо устройстве, что приведёт к запуску программы обработки прерывания.

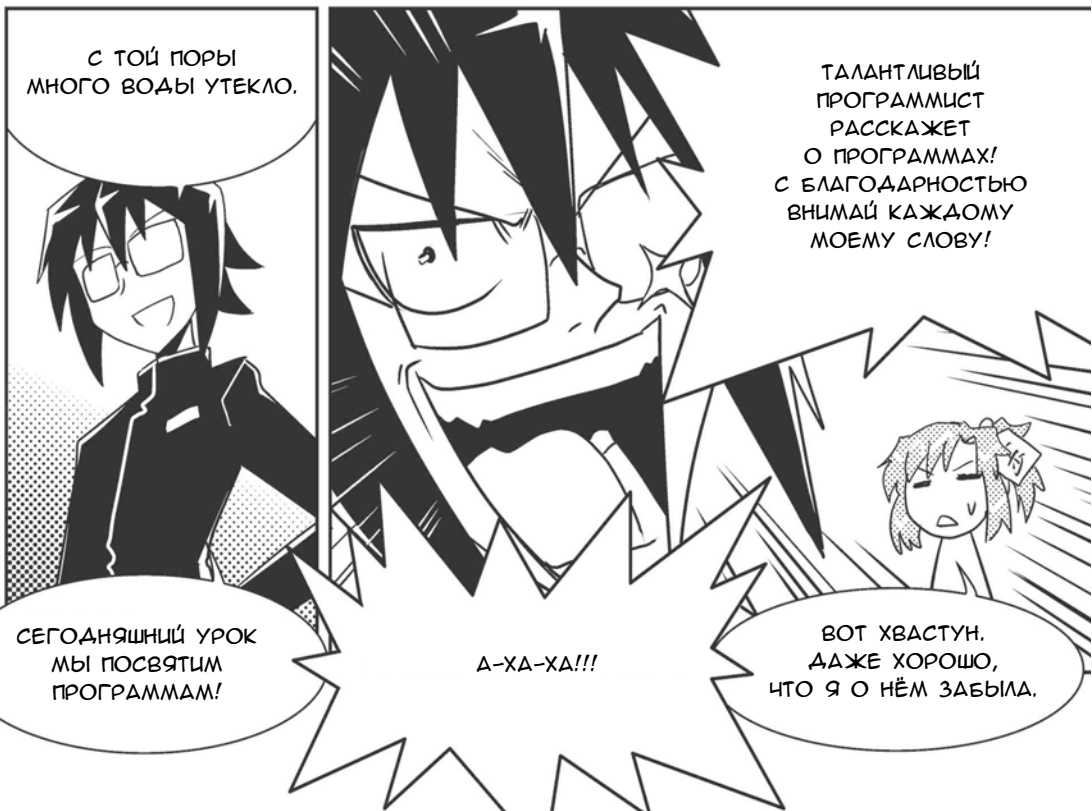


ГЛАВА 5

ПРОГРАММЫ

5.1. АССЕМБЛЕР И ЯЗЫКИ ВЫСОКОГО УРОВНЯ


БЫТЬ ТОГО НЕ МОЖЕТ!!!
ДА НЕ КРИЧИ ТАК!
В ОБЩЕМ, Я ТЕБЯ СОВЕРШЕННО НЕ ПОМНЮ.
НУ, ПОДУМАЕШЬ...
Ю-КУН...


С ТОЙ ПОРЫ МНОГО ВОДЫ УТЕКЛО.
ТАЛАНТЛИВЫЙ ПРОГРАММИСТ РАССКАЖЕТ О ПРОГРАММАХ! С БЛАГОДАРНОСТЬЮ ВНИМАЙ КАЖДОМУ МОЕМУ СЛОВУ!
СЕГОДНЯШНИЙ УРОК МЫ ПОСВЯТИМ ПРОГРАММАМ!
А-ХА-ХА!!!
ВОТ ХВАСТУН. ДАЖЕ ХОРОШО, ЧТО Я О НЁМ ЗАБЫЛА.

Что такое ассемблер?


Языки для написания программ называются **языками программирования**.
Например, язык C
Мнемонические коды
Последовательность нулей и единиц
Языки высокого уровня
Ассемблер
Машинный язык
Хорошо понятные для человека
Хорошо понятные для ЦПУ
Они бывают разных типов. Язык с использованием мнемонических кодов называется **ассемблером**.


Хм... языки высокого уровня, ассемблер и машинный язык...

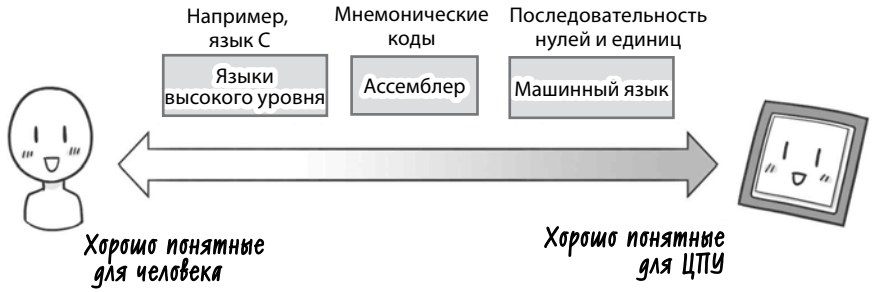

Высокий уровень – это прекрасно!
Дорогие автомобили! Престижные отели!


Нет, в данном случае "высокий уровень" имеет другой смысл.
Имеется в виду – "хорошо понятный человеку", не зависит от особенностей ЦПУ...


А про ассемблер и машинный язык...
...я сейчас расскажу.

Особенности ассемблера и языков высокого уровня

Здесь я расскажу об ассемблере, который в большей степени понятен ЦПУ, и о языках высокого уровня, которые более понятны человеку.



Хм... Я немного запуталась.

Очевидно, что машинный язык как последовательность нулей и единиц должен быть понятен ЦПУ и совершенно непонятен человеку (см. стр. 144).

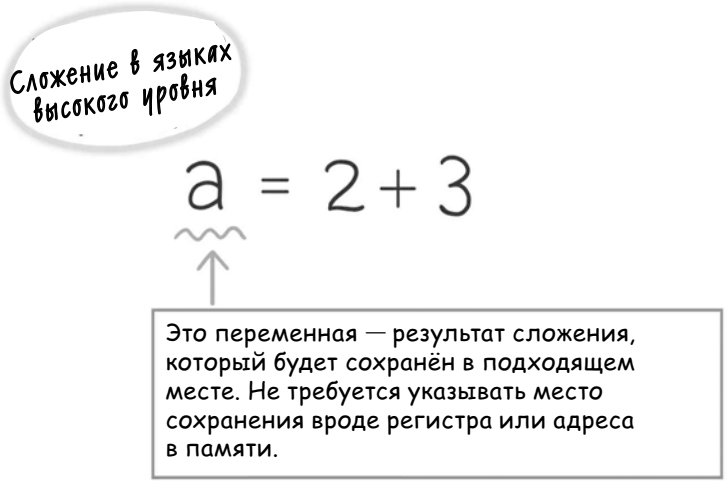
Но ассемблер — это же мнемонические коды, поэтому и человек должен их понимать, наверное? Например, ADD — «сложить» — английское название команды... В таком случае что представляют собой «языки высокого уровня», хорошо понятные человеку?!

Вопрос вполне логичный. Действительно, ассемблер до известной степени можно понять.

Однако ассемблер понятен тебе сейчас, когда ты изучила устройство ЦПУ и узнала, что такое аккумулятор и другие регистры, адреса, типы команд и тому подобное!

А вот когда ты используешь язык высокого уровня, можно вообще не задумываться о регистрах, адресах, командах и прочем.

Например, если тебе нужно сложить 2 и 3, можно просто написать «a = 2 + 3»!



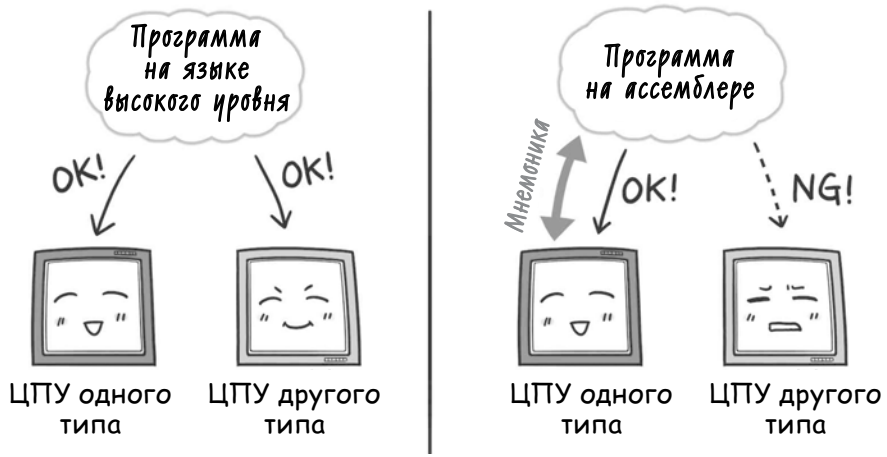
Ну и дела! Это же совсем не тот способ сложения, который мы изучали раньше (см. стр. 105, 169)! Так можно давать ЦПУ задания на языке, который интуитивно понятен человеку! Об устройстве ЦПУ знать необязательно! Просто революция! Торжество гуманизма!

Хи-хи-хи, вот ты и поняла прелесть языков высокого уровня! Они используются для разработки больших программ*.

* Про это мы узнаем на стр. 202.

У языков высокого уровня есть и другие достоинства, о которых я сейчас расскажу.

Взгляни на рисунок ниже — сейчас я его поясню. Программу на языке высокого уровня можно использовать для ЦПУ самых разных типов. С другой стороны, программу на ассемблере можно использовать только для ЦПУ того типа, для которого она была написана. Дело в том, что мнемоника отражает «как есть» собственные команды данного типа ЦПУ, поэтому перенос ассемблерной программы на ЦПУ других типов будет сопряжён с трудностями.



Различие программ на языке высокого уровня и на ассемблере

Угу. Я поняла, что язык высокого уровня и понятней, и удобней. Но в чём тогда достоинства ассемблера?

Зачем мы изучали ассемблер, устройство ЦПУ? Неужели всё это было не так уж и нужно?

Наверное, можно сказать, что язык высокого уровня — это современные технологии, а ассемблер — устаревшие...

Нет, это не так! Ассемблер используется и в наши дни, особенно в случаях, когда требуется скоростная обработка данных: ведь он позволяет использовать ЦПУ на полную мощность.

Ведь ассемблер — это язык собственных команд ЦПУ, выраженных с помощью мнемоники, поэтому он легко и эффективно переводится (транслируется) на машинный язык.

Легко перевести на машинный язык!

С другой стороны, чтобы перевести программу на языке высокого уровня на машинный язык, её нужно откомпилировать. И такая программа, изначально написанная на «простом человеческом языке», может выполняться медленнее*, хотя результат вычислений, конечно, тоже будет правильным. То есть время ЦПУ будет использоваться неэффективно. Другими словами, при использовании языка высокого уровня возможности ЦПУ не всегда реализуются в полной мере.

При переводе с языка высокого уровня на машинный язык программа вынуждена проделывать «лишнюю работу»

* Несмотря на это, современные ЦПУ обладают высоким быстродействием, поэтому некоторое замедление выполнения программ во многих случаях не создаёт проблем при использовании в системах, предназначенных для взаимодействтвия с человеком: персональных компьютерах и т. п.

Ясно! Ассемблер позволяет избавиться от лишнего и использовать ЦПУ на все сто! Это классный язык, который и в наши дни находит применение!

О разработке больших программ

Программы для персональных компьютеров — например, для табличных вычислений, создания презентаций, работы с текстом — называются *приложениями*. На их разработку уходит очень много времени и усилий, и занимаются этим большие коллективы программистов. Языки, с помощью которых можно осуществить эту цель, называются *языками разработки программ*, или *высокоуровневыми языками программирования*. Это такие языки, как C и подобные ему: C++ («си-плюс-плюс»), C# («си-шарп»).

При разработке программ на таких языках высокого уровня программисты не задумываются ни об отдельных командах ЦПУ (об уровне машинного кода), ни об упомянутых в предыдущей главе режимах адресации.

Высокоуровневая разработка предполагает создание исходного кода программы и его *компиляцию* в машинный код, который ЦПУ может выполнить. В процессе компиляции режимы адресации и прочее автоматически оптимизируются, поэтому разработчикам исходного кода не нужно беспокоиться о наборах команд определённых ЦПУ, режимах адресации, использовании разнообразных регистров.

Однако язык ассемблера (практически уровень машинного кода) используется для написания программ для небольших устройств. В этом случае невозможно написать правильную программу, если не разбираться досконально в особенностях работы ЦПУ, режимах адресации. Язык ассемблера позволяет записать собственные команды ЦПУ в виде мнемонических кодов и ассемблировать их, то есть автоматически преобразовать в машинный (двоичный) код с помощью специальной программы.

Хотя для разработки *операционных систем* (ОС) типа Windows в последнее время стали использоваться C-подобные высокоуровневые языки разработки, некоторые участки кода, особенно требующие высокой скорости выполнения, разрабатываются с использованием ассемблера. То же самое касается и критичных по скорости участков кода приложений.

На ассемблере иногда частично пишутся программы вроде специальных симуляторов, чтобы немного повысить скорость вычислений.

Кроме того, программы для компактных микрокомпьютеров иногда полностью пишутся на ассемблере, без использования, например, языка C.

Чем программа отличается от исходного кода?

Чуть раньше (см. стр. 196) я назвал программу «исходным кодом». Понятия это очень близкие, но, строго говоря, между ними есть отличия.

Так-так! Давай-ка поподробнее!

Итак, *программа* — это полный набор указаний компьютеру, как выполнять работу.

С другой стороны, *исходный код* — это изначальная структура программы, с помощью которой человек может задавать и редактировать порядок её выполнения. В последнее время стало возможным генерировать части исходного кода автоматически с помощью технологий *исскуственного интеллекта* (Artificial Intelligence, AI).

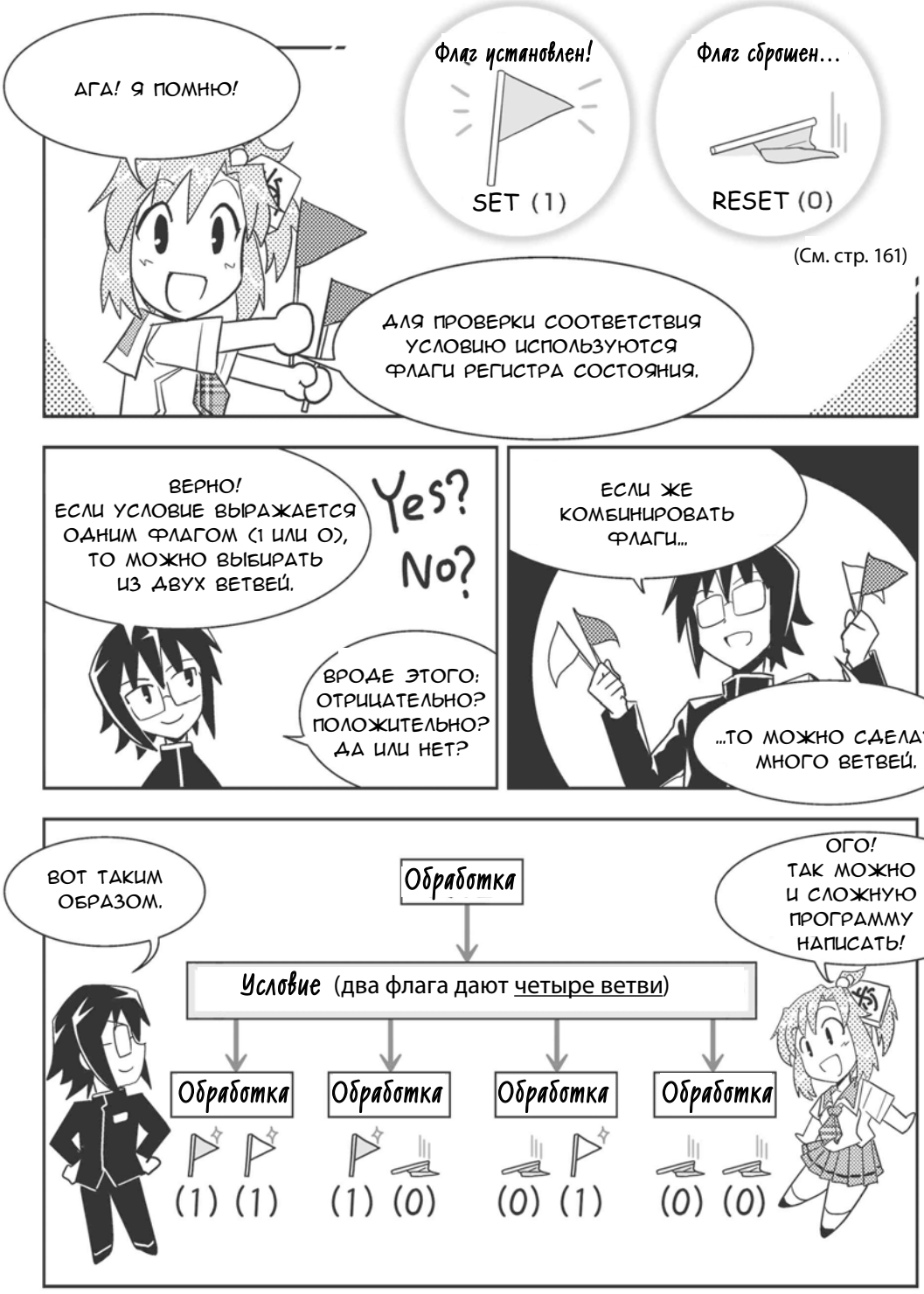
Угу... кажется, понимаю. *Программа* — это полный наряд на работы. А *исходный код* — это данные (набор символов, выражающий указания компьютеру), составленные, например, человеком.

Кстати, иногда можно услышать слово «исходная программа». Это можно считать синонимом понятия «исходный код».

5.2. ОСНОВНЫЕ СВЕДЕНИЯ О ПРОГРАММАХ

Что могут проверки условий и переходы?

(См. стр. 113, 163)


Ага! Я помню!
Флаг установлен!
SET (1)
Флаг сброшен...
RESET (0)
(См. стр. 161)
Для проверки соответствия условию используются флаги регистра состояния.
Верно! Если условие выражается одним флагом (1 или 0), то можно выбирать из двух ветвей.
Yes?
No?
Вроде этого: отрицательно? положительно? да или нет?
Если же комбинировать флаги...
...то можно сделать много ветвей.
Вот таким образом.
Обработка
Условие (два флага дают четыре ветви)
Обработка
Обработка
Обработка
Обработка
(1) (1)
(1) (0)
(0) (1)
(0) (0)
Ого! Так можно и сложную программу написать!


А ЕСЛИ ИСПОЛЬЗОВАТЬ УСЛОВНЫЕ ПЕРЕХОДЫ...
Цикл повторяется, пока выполняется условие.
Обработка
Условие
No
Yes
Когда условие перестало выполняться, идём дальше!
...МОЖНО КРАСИВО ЗАДАТЬ ПОВТОР ОДНОГО И ТОГО ЖЕ ДЕЙСТВИЯ!
ЭТО НАЗЫВАЕТСЯ ЦИКЛОМ (LOOP).
КАК УДОБНО!
МОЖНО НЕ ОТДАВАТЬ ПО МНОГУ РАЗ ОДНИ И ТЕ ЖЕ КОМАНДЫ...
НУ ВОТ И ВСЁ САМОЕ ОСНОВНОЕ О ПРОГРАММАХ.
КАК?! ТОЛЬКО И ВСЕГО?!!
НУ, КОНЕЧНО, ДЛЯ НАПИСАНИЯ ПРОГРАММ...
...ПОТРЕБУЮТСЯ ГОРАЗДО БОЛЕЕ ГЛУБОКИЕ ЗНАНИЯ.
ОДНАКО...
?


...КАК БЫ СЛОЖНА НИ БЫЛА ПРОГРАММА И КАКОЙ БЫ ЯЗЫК ПРОГРАММИРОВАНИЯ НИ ИСПОЛЬЗОВАЛСЯ...
...ПРОВЕРКА УСЛОВИЯ И ПЕРЕХОД ВСЕГДА БУДУТ ОЧЕНЬ ВАЖНЫМИ ПОНЯТИЯМИ!
ПОЛУЧАЕТСЯ, ПРОВЕРКА УСЛОВИЯ И ПЕРЕХОД – ЭТО ОСНОВА И ВЫСШИЙ СМЫСЛ ПРОГРАММИРОВАНИЯ.
ДА! ПОКА ЗАПОМНИ ТОЛЬКО ЭТО.
ЕСЛИ ЗАХОЧЕШЬ УЗНАТЬ О ПРОГРАММАХ БОЛЬШЕ, ТЕБЕ ПОНАДОБИТСЯ ПЕРЕЧИТАТЬ ГОРЫ КНИГ И ПЕРЕЛОПАТИТЬ МОРЕ ИСХОДНЫХ КОДОВ!
И НИКАК ИНАЧЕ! ТАКОВ ПУТЬ СТАНОВЛЕНИЯ ТАЛАНТЛИВОГО ПРОГРАММИСТА!!
ВСПЫХ!
КАКОЙ ВЫСОКИЙ СЛОГ!
ЗА ОДИН ДЕНЬ ЭТОТ ПУТЬ НЕ ПРОЙТИ.
ВШ-Ш
ПЕРЕГРЕЛСЯ?!

Что бы поручить компьютеру?

Что ж, сегодня мы изучили программы... И теперь я хотел бы, чтобы ты вспомнила об оцифровке информации (см. стр. 12).

В современном обществе разнообразная информация, например музыка, фотографии, изображения, превращена в цифровые данные и может быть обработана на компьютере.

Ага, мы об этом говорили вначале... Удивительно, если вдуматься! Раз любую информацию можно обрабатывать на компьютере, мы можем сделать столько всего полезного, просто написав хорошую программу!

Да, так оно и есть.

То, чего раньше и представить было нельзя, в наши дни быстро воплощается в жизнь.

Например, в системах безопасности используются программы для распознавания лица, в которых различные особенности человеческого лица (расстояние между глазами, размеры и положение рта, носа и т. д.) обрабатываются и запоминаются компьютером. Программа оцифровывает данные о человеческом лице, и компьютер может распознавать его.

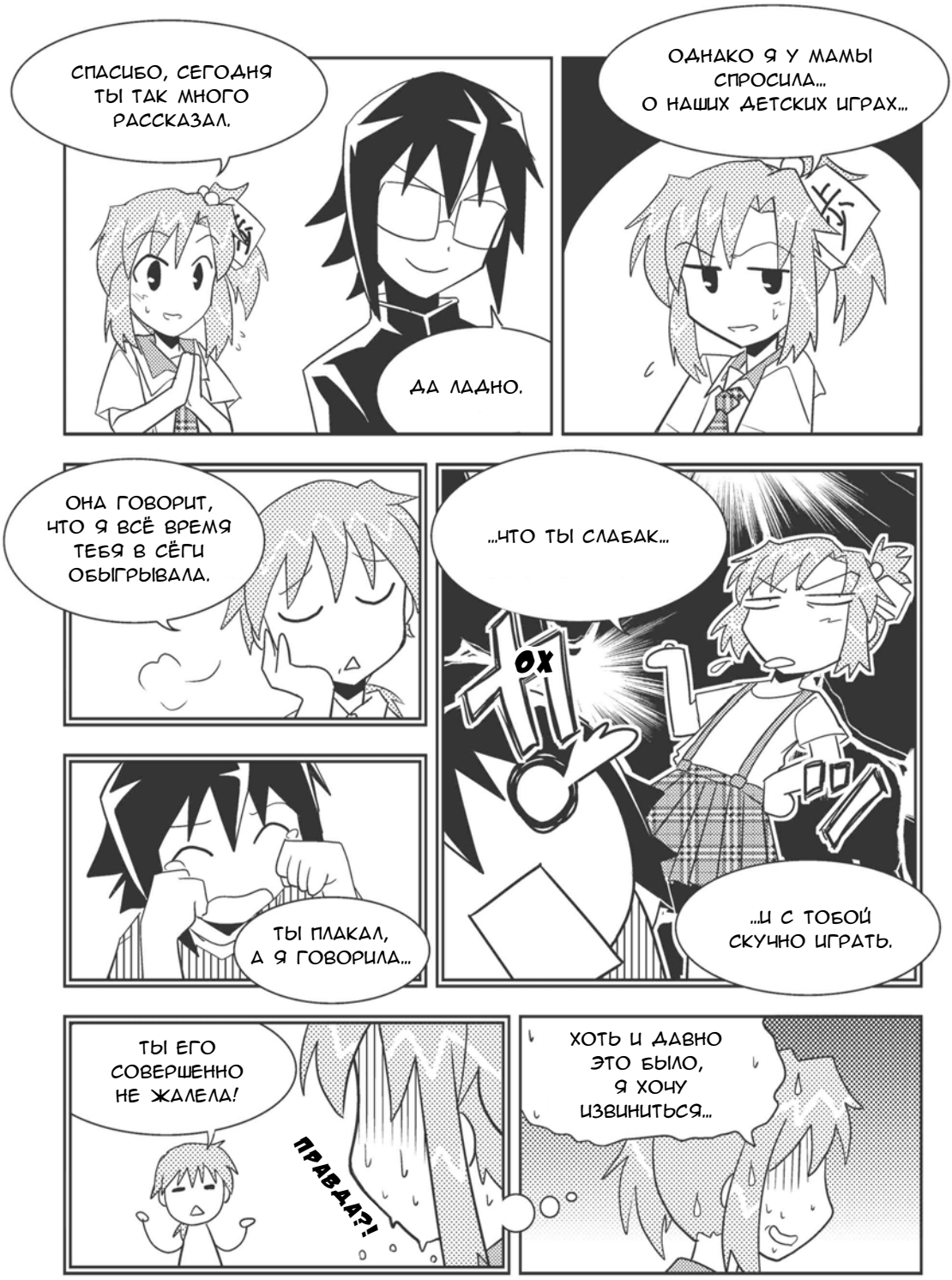
Да... Компьютер узнаёт человека в лицо... Просто фантастика какая-то — даже немного страшновато! Но хорошо, если это поможет предотвращать преступления.

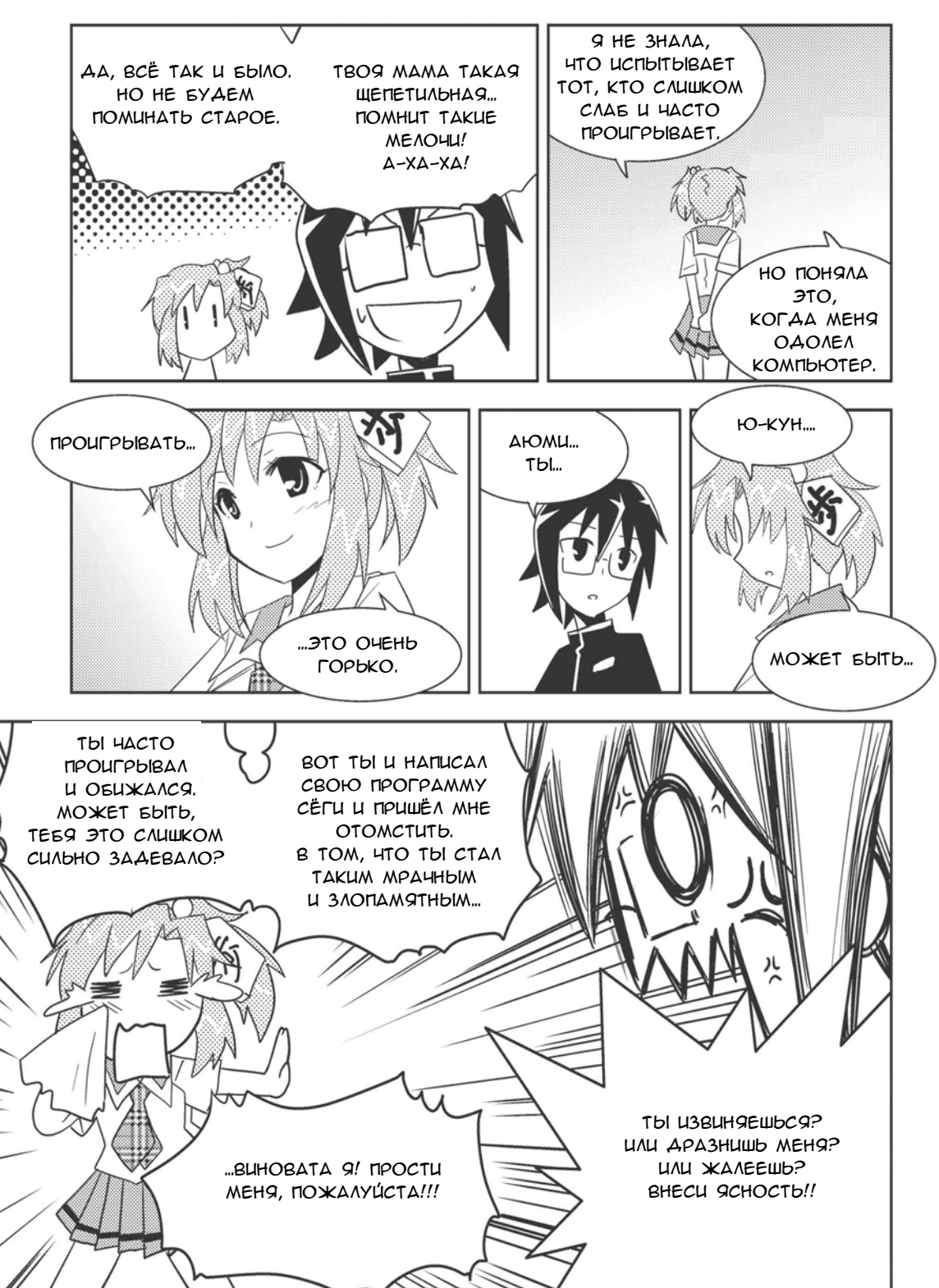
Что ж, разработка хороших программ — весьма интересное занятие. Вот что я, например, хотела бы поручить компьютеру?

Прогнозы изменения цен на акции, исхода скачек... Какую-нибудь программу автоматического зарабатывания денег...

Ах, не будем о твоих личных амбициях!

Что бы я хотел поручить компьютеру? Какая программа была бы удобна и принесла бы пользу людям? Размышлять об этом и мечтать — вот что главное для программиста!


СПАСИБО, СЕГОДНЯ ТЫ ТАК МНОГО РАССКАЗАЛ.
ДА ЛАДНО.
ОДНАКО Я У МАМЫ СПРОСИЛА... О НАШИХ ДЕТСКИХ ИГРАХ...
ОНА ГОВОРИТ, ЧТО Я ВСЁ ВРЕМЯ ТЕБЯ В СЁГИ ОБЫГРЫВАЛА.
...ЧТО ТЫ СЛАБАК...
ОХ
ТЫ ПЛАКАЛ, А Я ГОВОРИЛА...
...И С ТОБОЙ СКУЧНО ИГРАТЬ.
ТЫ ЕГО СОВЕРШЕННО НЕ ЖАЛЕЛА!
ПРАВДА?!
ХОТЬ И ДАВНО ЭТО БЫЛО, Я ХОЧУ ИЗВИНИТЬСЯ...


ДА, ВСЁ ТАК И БЫЛО. НО НЕ БУДЕМ ПОМИНАТЬ СТАРОЕ.
ТВОЯ МАМА ТАКАЯ ЩЕПЕТИЛЬНАЯ... ПОМНИТ ТАКИЕ МЕЛОЧИ! А-ХА-ХА!
Я НЕ ЗНАЛА, ЧТО ИСПЫТЫВАЕТ ТОТ, КТО СЛИШКОМ СЛАБ И ЧАСТО ПРОИГРЫВАЕТ.
НО ПОНЯЛА ЭТО, КОГДА МЕНЯ ОДОЛЕЛ КОМПЬЮТЕР.
ПРОИГРЫВАТЬ...
...ЭТО ОЧЕНЬ ГОРЬКО.
ДЮМИ... ТЫ...
Ю-КУН....
МОЖЕТ БЫТЬ...
ТЫ ЧАСТО ПРОИГРЫВАЛ И ОБИЖАЛСЯ. МОЖЕТ БЫТЬ, ТЕБЯ ЭТО СЛИШКОМ СИЛЬНО ЗАДЕВАЛО?
ВОТ ТЫ И НАПИСАЛ СВОЮ ПРОГРАММУ СЁГИ И ПРИШЁЛ МНЕ ОТОМСТИТЬ. В ТОМ, ЧТО ТЫ СТАЛ ТАКИМ МРАЧНЫМ И ЗЛОПАМЯТНЫМ...
ТЫ ИЗВИНЯЕШЬСЯ? ИЛИ ДРАЗНИШЬ МЕНЯ? ИЛИ ЖАЛЕЕШЬ? ВНЕСИ ЯСНОСТЬ!!
...ВИНОВАТА Я! ПРОСТИ МЕНЯ, ПОЖАЛУЙСТА!!!


ЛАДНО, ДОЛЖНА ПРИЗНАТЬ...
Я ПРОИГРАЛА, КАК БЫ ГОРЬКО ЭТО НИ БЫЛО...
...А ТЫ СМЫЛ С СЕБЯ ТОТ ПОЗОР.
ХЕ-ХЕ...
ТЕПЕРЬ МЫ КВИТЫ!
ЗАБУДЬ О ДЕТСКИХ ОБИДАХ, МРАЧНЫЙ ЮНОША!
НУ ВОТ, ОПЯТЬ ТЫ ЯЗВИШЬ.
ХМ-М... НУ ЛАДНО. СЛЕДУЮЩИЙ УРОК - ПОСЛЕДНИЙ.
ТАК ЧТО ПРИНОСИ SHOOTING STAR.
SHOOTING STAR... А ЧТО ЭТО?
?
ДЕЛАЕШЬ ВИД, ЧТО ЗАБЫЛА?!

ДОПОЛНИТЕЛЬНАЯ ИНФОРМАЦИЯ

◆ Где хранятся программы?

В микроконтроллерах (глава 6) для небольших устройств программы обычно хранятся в ROM.

В персональных компьютерах самая базовая программа хранится в ROM, и ЦПУ с её помощью загружает операционную систему с жёсткого диска в RAM. Также и приложения выполняются после того, как ЦПУ загружает их в RAM.

На заре развития ЦПУ несколько десятилетий назад использовалась маленькая программа под названием *резидентный монитор*, представлявшая собой миниатюрную предшественницу операционных систем, и с её помощью разрабатывали программы на ассемблере.

В настоящее время разработка программ на ассемблере тоже ведётся с помощью персональных компьютеров.

Производители ЦПУ предоставляют *средства разработки на ассемблере*, предназначенные для конкретных моделей ЦПУ. С помощью этих средств программу разрабатывают на персональном компьютере, записывают её в ROM с помощью специального устройства (ROM-writer), подключённого к компьютеру, и встраивают эту ROM в устройство, для которого была написана программа.

Кроме того, в наши дни программу, написанную на компьютере, можно непосредственно записывать в энергонезависимую память (см. стр. 132), находящуюся внутри устройства. Это позволяет быстро проверять правильность работы устройства, не производя каждый раз запись в ROM.

Программа хранится в ROM (или в энергонезависимой памяти)!

О ROM и RAM см. стр. 119.
О энергонезависимой памяти см. стр. 132.

Кстати, методику записи программы непосредственно на устройство ROM, подключенное к ЦПУ, называют *внутрисхемным программированием*.

Этапы запуска программы

Представим, что программа, разработанная для выполнения определённых действий, записана в ROM (или в энергонезависимую память). Как при этом будет работать ЦПУ после включения питания устройства?

Во-первых, сразу же после включения питания ЦПУ не делает ничего, так как запуск до стабилизации напряжения источника питания может привести к внутренним сбоям (механизм сброса описывается на стр. 138).

Чтобы обеспечить это бездействие, вход «Сброс» ЦПУ находится в *активном состоянии* (обычно ему соответствует низкий логический уровень). Если в ЦПУ встроен тактовый генератор, то в это время он должен начать работу.

Тактовые генераторы обычно начинают генерировать сигнал до того, как напряжение питания ЦПУ стабилизируется на установленном уровне. Когда это происходит — другими словами, когда напряжение достигает уровня, гарантирующего нормальную работу ЦПУ, — после временной задержки, указанной в техническом паспорте, производится отмена состояния сброса. Теперь наконец-то можно сказать, что ЦПУ готов к считыванию первой строки программы. После отмены состояния сброса ЦПУ первым делом считывает вектор сброса.

Вектор сброса — это первая или последняя ячейка пространства адресов, находящегося под управлением ЦПУ; в ней записан адрес первой команды, которая будет выполняться.

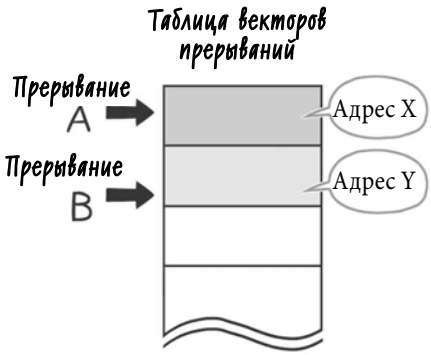
Концептуальная схема вектора сброса

Сброс обладает наивысшим приоритетом среди всех остальных прерываний, поэтому если по какой-либо причине вывод сброса становится активным, то ЦПУ сразу же возвращается в начальное состояние независимо от того, какая задача выполнялась до этого.

Стартовав с адреса, записанного в векторе сброса, программа наконец-то начинает обработку данных путём выполнения, например, операций.

Кстати, в ЦПУ имеются различные прерывания, и для каждого типа прерываний назначается адрес, с которого стартует программа его обработки. Часть памяти, содержащую эти адреса, называют *таблицей векторов прерываний*.

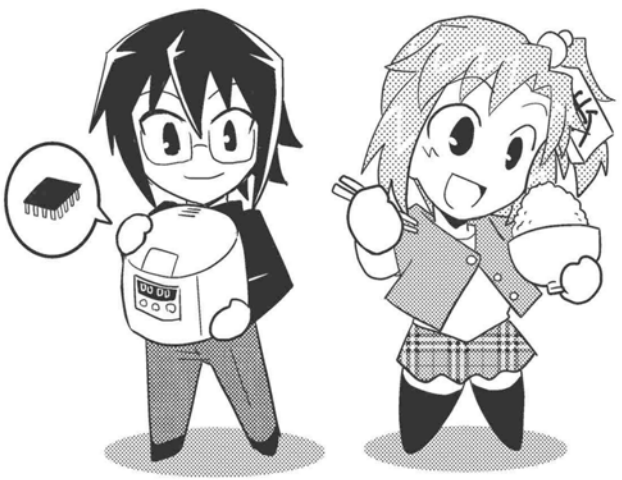
Заглянув в таблицу векторов прерываний, мы бы увидели: «Если пришло прерывание A, то стартуй с адреса X!» «Если пришло прерывание B, то стартуй с адреса Y!» — и так далее...



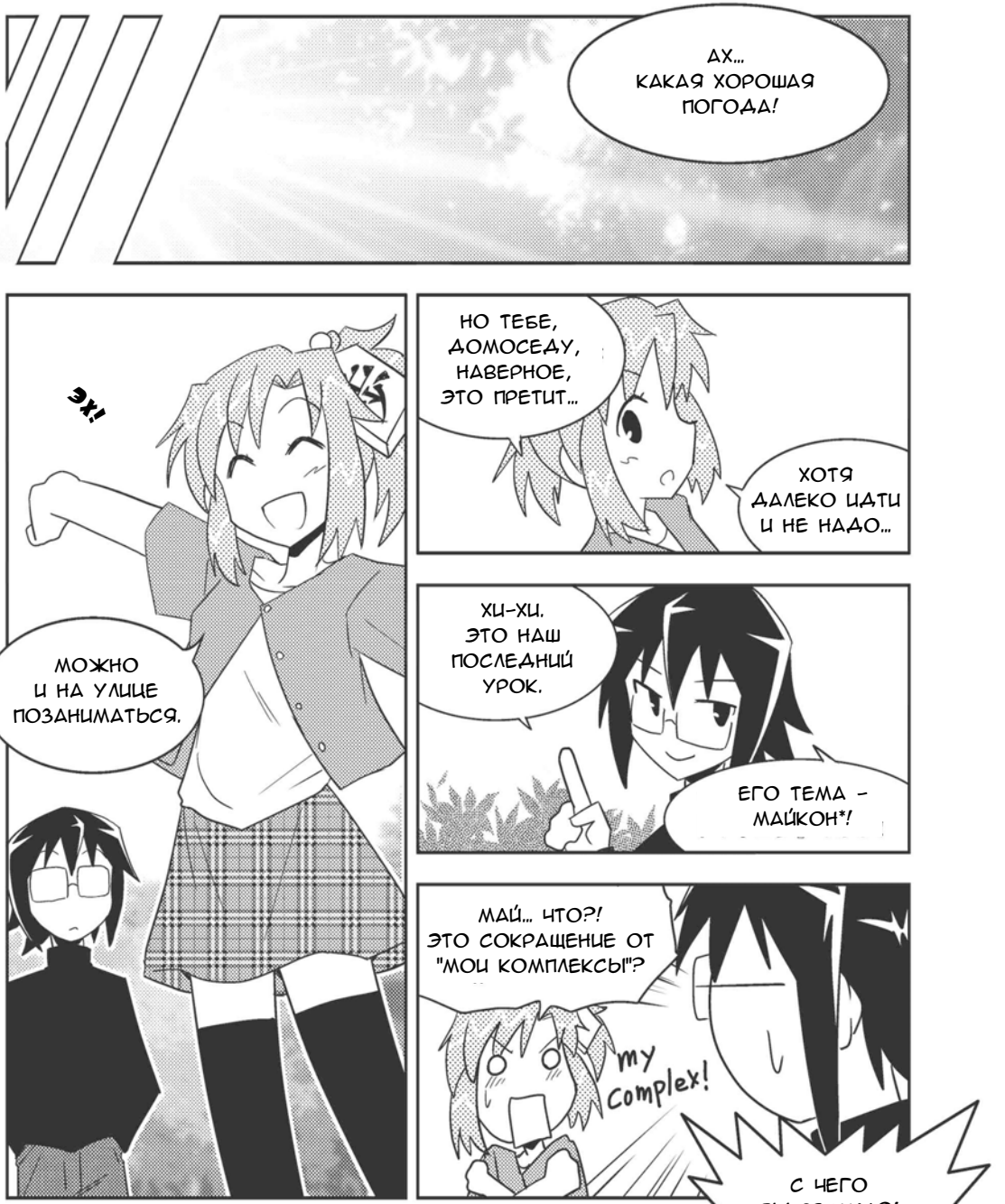
Концептуальная схема таблицы векторов прерываний

ГЛАВА 6

МИКРОКОНТРОЛЛЕРЫ



6.1. ЧТО ТАКОЕ МИКРОКОНТРОЛЛЕР?

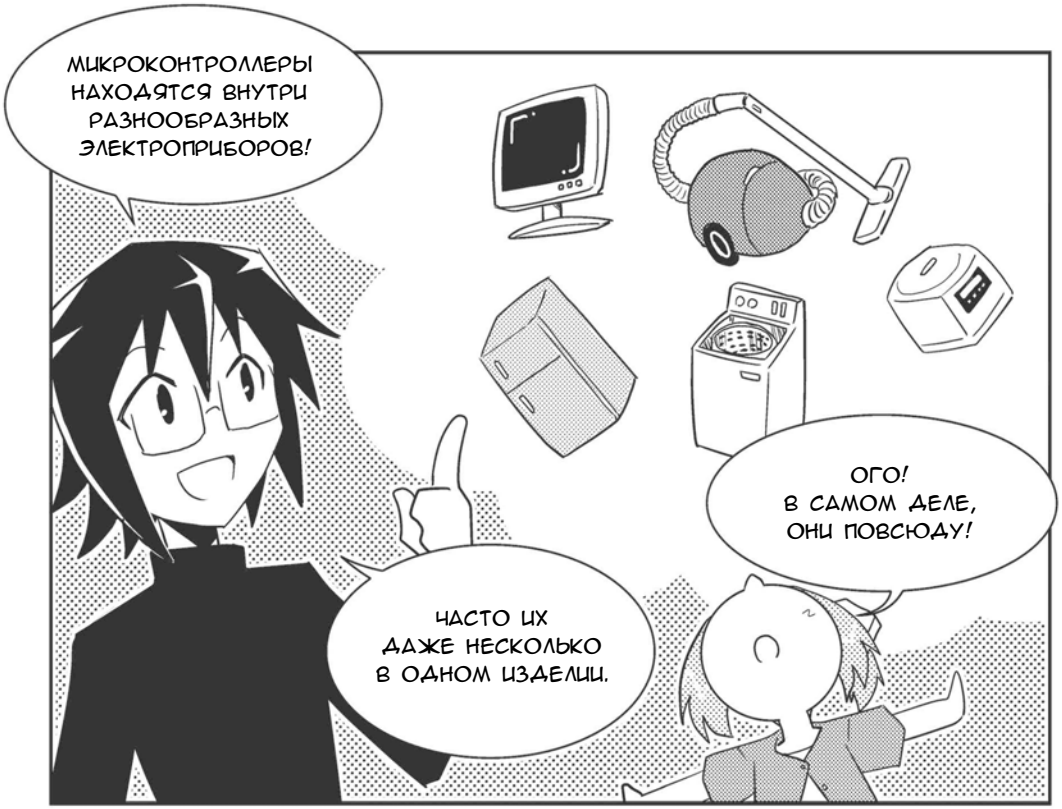


* Японское название микроконтроллера. — *Прим.перев.*

Микроконтроллеры находятся внутри разных изделий

Иногда «майконами» называют микрокомпьютеры (micro computer). См. также стр. 224.

* Бывают и микроконтроллеры прямоугольной формы, как микросхемы, показанные на стр. 48.



Функции микроконтроллера


Микроконтроллер
Функция памяти (RoM , RAM)
Функция ЦПУ
Функция управления I/O
И всё это в одной микросхеме!
ПРЕДСТАВЬ СЕБЕ! В МАЙКОНАХ ЕСТЬ ТРИ ФУНКЦИИ: ПАМЯТЬ (ROM, RAM), ЦПУ И УПРАВЛЕНИЕ I/O!
ПОЭТОМУ ИХ ЕЩЁ НАЗЫВАЮТ ВСТРАИВАЕМЫМИ КОНТРОЛЛЕРАМИ.

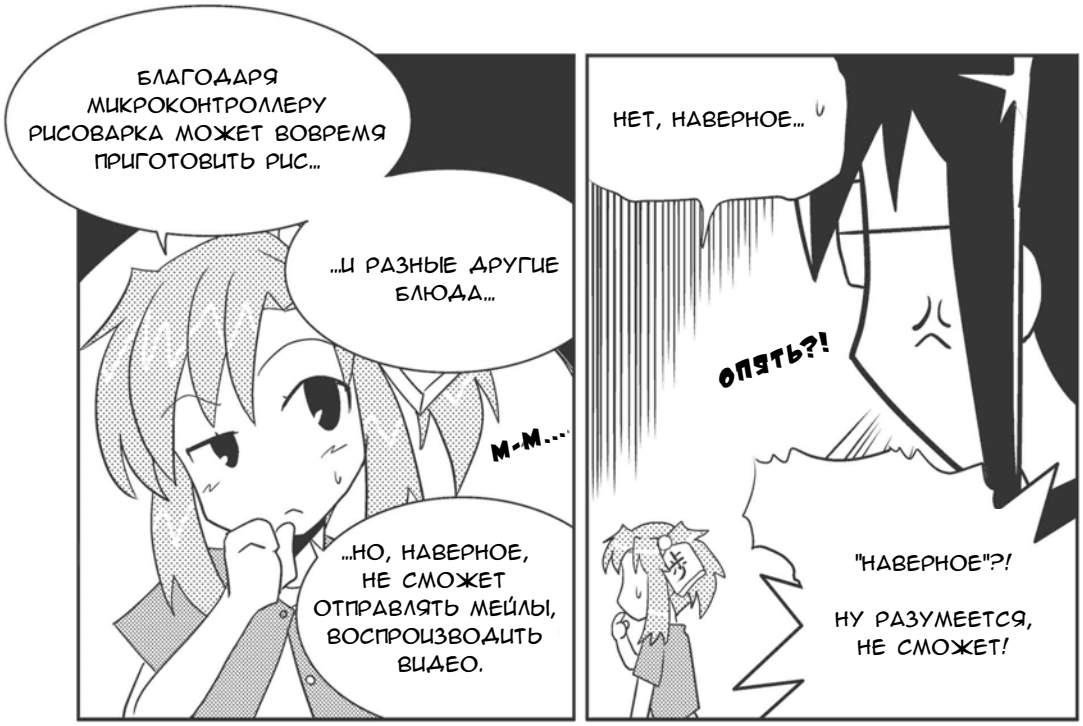

НИЧЕГО СЕБЕ! ТРИ В ОДНОМ!
ХИ-ХИ! ЗДОРОВО, ПРАВДА?..


ОДИН МИКРОКОНТРОЛЛЕР ДЕЛАЕТ ВСЮ РАБОТУ – ОТ ХРАНЕНИЯ ПРОГРАММ ДО ВЫПОЛНЕНИЯ ОПЕРАЦИЙ И ОБРАБОТКИ ВВОДА-ВЫВОДА.


ЗНАЧИТ, МИКРОКОНТРОЛЛЕР – ЭТО ПОЛНОЦЕННОЕ УСТРОЙСТВО УПРАВЛЕНИЯ ТЕХНИКОЙ.
ПОЛУЧАЕТСЯ, ЧТО...
...МИКРОКОНТРОЛЛЕРЫ В НЕКОТОРОМ СМЫСЛЕ УДОБНЕЙ, ЧЕМ ПЕРСОНАЛЬНЫЕ КОМПЬЮТЕРЫ?
В ОБЩЕМ, ДА.


НО ПРОИЗВОДИТЕЛЬНОСТЬ ЦПУ ПЕРСОНАЛОК И МАЙКОНОВ – ЭТО НЕБО И ЗЕМЛЯ!
Пример функций ЦПУ микроконтроллера
Управление температурой
Например, поддерживать температуру 70 °C
Майкон
Управление таймером
Например, начать работу в 6:00


СКАЖЕМ, МИКРОКОНТРОЛЛЕР РИСОВАРКИ ПОДДЕРЖИВАЕТ ФУНКЦИИ УПРАВЛЕНИЯ ТЕМПЕРАТУРОЙ И ТАЙМЕРОМ...
ДА, В САМОМ ДЕЛЕ...
...НО НЕ МОЖЕТ ДЕЛАТЬ РАЗНУЮ СЛОЖНУЮ РАБОТУ, КАК ЦПУ ПЕРСОНАЛКИ.

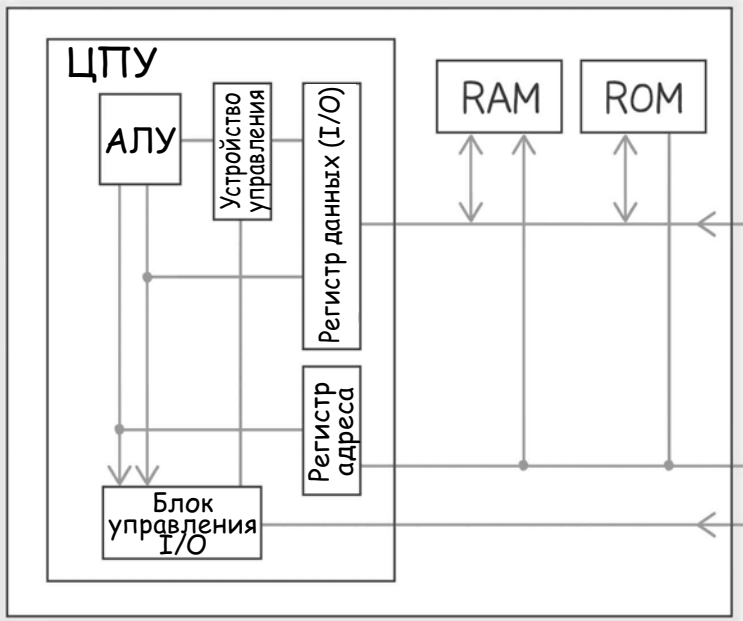

Благодаря микроконтроллеру рисоварка может вовремя приготовить рис...
...и разные другие блюда...
м-м...
...но, наверное, не сможет отправлять мейлы, воспроизводить видео.
Нет, наверное...
опять?!
"Наверное"?!
Ну разумеется, не сможет!


В общем, функции микроконтроллера ограничены его назначением.
Функции ограничены!
Сравнительно дешёвый!
МАЙКОН
Тем не менее бывают высокопроизводительные и дорогие микроконтроллеры.
Поэтому он дешевле, чем ЦПУ персонального компьютера.


КСТАТИ, ВСЕ ФУНКЦИИ СОДЕРЖАТСЯ В ОДНОМ КРИСТАЛЛЕ (ЧИПЕ),...
one chip!
ПОЭТОМУ ИХ ЕЩЁ НАЗЫВАЮТ ОДНОКРИСТАЛЬНЫМИ МИКРОКОНТРОЛЛЕРАМИ.
ХОРОШО. ПРО ОСОБЕННОСТИ МАЙКОНОВ Я ПОНЯЛА.
В ОДНОЙ МАЛЕНЬКОЙ МИКРОСХЕМЕ ПОМЕЩАЕТСЯ НАСТОЯЩЕЕ УСТРОЙСТВО УПРАВЛЕНИЯ ТЕХНИКОЙ!
НИЧЕГО ОБЩЕГО С МОИМИ КОМПЛЕКСАМИ!
ДА И НЕ МОЖЕТ БЫТЬ НИЧЕГО ОБЩЕГО!
ТРЕСК
УРА!
ДУМАЕШЬ, ЧТО ВЕСЬ МИР ВОКРУГ ТЕБЯ ОДНОЙ ВЕРТИТСЯ?!

Устройство майкона

Концептуальная схема микроконтроллера



История микроконтроллеров

С «майконами» (микроконтроллерами) связано много занимательных историй. Приведём здесь одну из них.

Первые электронные вычислительные машины занимали помещение размером с волейбольную площадку и содержали АЛУ, построенные на электронных лампах. Это было в 40-х годах XX века, в разгар второй мировой войны. Говорят, что в Англии велись работы над большой ЭВМ для чтения немецких шифровок, но всё это держалось в тайне — совсем не так, как в наше время конкуренции на рынке компьютеров! Поэтому точно неизвестно, была ли ЭВМ ENIAC, созданная в США в 1946 году, первой в мире или нет.

В 1947 году, когда изобрели транзистор, большие надежды возлагались на замену ламповых ЭВМ полупроводниковыми. А с 1958 года, когда появились первые микросхемы, возникла тенденция миниатюризации ЭВМ.

Несмотря на это, первый 16-битный мини-компьютер на четырех микросхемах 74S181, о котором рассказывалось в главе 4, появился лишь в 1972 году. Его размеры без устройств ввода-вывода составляли 60 см в ширину, 30 см в высоту и 40 см в глубину. Поддерживалось пространство адресов в 16 килослов (kiloword, 1 kW = 16 бит). В переводе на современный язык это 32 килобайта (Кбайт, KB). В наши дни даже объём карт памяти SD может составлять 32 Гбайта, что в 1 млн раз больше!

Со второй половины 1970-х годов, когда фирма Intel сообщила о создании универсального *однокристального ЦПУ*, началось их распространение по всему миру и удешевление.

Пришла эпоха, когда однокристальные ЦПУ стали доступными даже для любителей: в 1976 году был выпущен конструктор обучения программированию в машинных кодах на базе однокристального ЦПУ (TK-80 компании NEC).

В те времена в Японии окончание «-кон» означало «компьютер», и было много радиолюбителей, называвших «майконами» (от английского My Computer: «мой компьютер») свои домашние компьютеры.

Однако в процессе развития ЦПУ сначала появились *одноплатные контроллеры* (One Board Controller) размером в лист формата A4, содержащие ЦПУ, память и устройство управления I/O и способные управлять техникой, а затем и однокристальные контроллеры. Незаменимая для ЦПУ память (ROM, RAM) и порты I/O были встроены с ЦПУ в одну микросхему, что позволило одной микросхеме делать всю работу: хранить программы, выполнять операции, обратывать ввод и вывод.

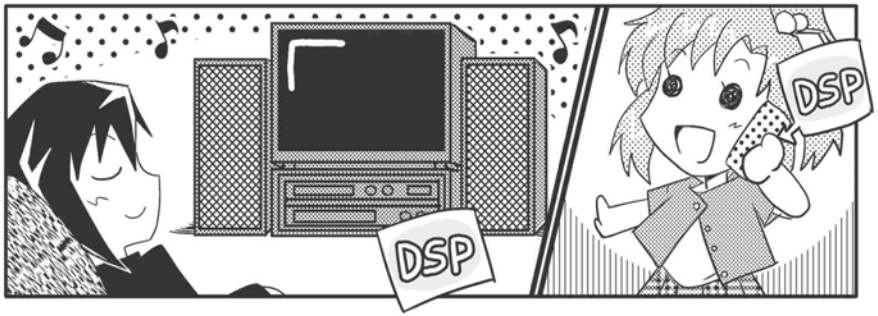
Это были маленькие, размером всего с одну микросхему контроллеры, поэтому их называли «микроконтроллеры» (micro controller); в Японии это название сократилось и получилось «майкон».

После этого во всех малогабаритных товарах широкого потребления, в игрушках и тому подобном стали использоваться только *однокристальные микроконтроллеры* (one chip micro controller), хотя возросшие требования, например, к объёму памяти поддерживают также спрос на одноплатные компьютеры.

В общем майкон подразумевает объединение функций ЦПУ, памяти и управления I/O, поэтому означает не «мой компьютер», а «микроконтроллер».

В последние годы майконами стали называть просто ЦПУ в широком смысле, что создаёт путаницу в понятиях.

Что такое DSP?



Воспользуемся шансом — раз уж мы изучили микроконтроллеры, поговорим и о DSP.

О DSP? Это ещё что такое? Ты замучишь меня этими английскими сокращениями!

Как и ЦПУ, DSP — это микросхема для выполнения операций, но её быстродействие выше, чем даже у ЦПУ.

Головной мозг DSP — это умножитель-аккумулятор, поэтому его конёк — умножать и складывать одновременно!

Вот как?! Но что хорошего в том, чтобы умножать и складывать одновременно? Какая в этом польза?

На самом деле для обработки звукового цифрового сигнала требуется выполнять много операций одновременного умножения-суммирования.

Звукового... Например, от мобильного телефона?

Мой аналоговый голос преобразуется в цифровой сигнал и передаётся собеседнику. Наверное, приходится его обрабатывать так...

Верно! В старых моделях мобильных телефонов был DSP. И в наши дни DSP тоже используется широко, например в аудиоаппаратуре: в цифровых фильтрах или для улучшения звуковых эффектов.

Ого! Он ближе нам, чем можно подумать!

В последнее время, когда научились встраивать в один кристалл RAM большого объёма, разрабатываются даже DSP уровня микроконтроллера.

Кстати, DSP — это абревиатура от Digital Signal Processor (*цифровой сигнальный процессор*). Он называется так потому, что тесно связан с *обработкой цифровых сигналов*, в частности звуковых.

Цифровой сигнальный процессор
Digital Signal Processor
Устройство для обработки цифровых сигналов

Вот как? Это его умение одновременно умножать и складывать, видимо, помогает цифровой сигнал обрабатывать. ЦПУ, конечно, важны, но и про DSP забывать не стоит. Запомню как следует!

ДОПОЛНИТЕЛЬНАЯ ИНФОРМАЦИЯ

DSP и умножитель-сумматор

В процессе развития ЦПУ понадобилось повысить скорость выполнения арифметических вычислений, особенно умножения и деления.

Как мы видели раньше, АЛУ — мозг ЦПУ — мог выполнять только арифметические операции сложения и вычитания. Поэтому умножение в таком классическом АЛУ означает многократное сложение, а деление — многократное вычитание.

По мере внедрения компьютеров в научно-технические расчёты возрастала потребность в быстром умножении, поэтому разрабатывались электронные схемы, быстро выполняющие умножение с плавающей точкой. Вершиной этих разработок стал DSP — цифровой сигнальный процессор.

Почему его так назвали? Дело в том, что для фильтрации звукового сигнала нужно выполнять быстрое преобразование Фурье — БПФ (Fast Fourier Transform, FFT), в котором много операций одновременного умножения и сложения. Поэтому и назвали цифровым сигнальным процессором (DSP) схему, «мозгом» которой является сумматор-аккумулятор и которая способна одновременно выполнять умножение и сложение.

Сразу же после появления DSP стала распространяться цифровая мобильная связь. В ней стали использовать основанные на DSP устройства под названием *вокодеры* (voice encoder decoder — *кодировщик-декодировщик голоса*), в которых звуковой сигнал оцифровывался, фильтровался и сжимался перед передачей и восстанавливался в прежнем виде после приёма.

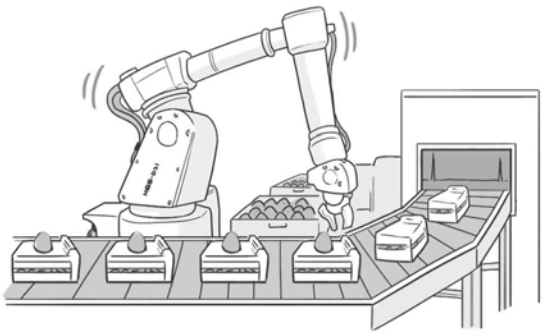
Затем, когда RAM большого объёма научились размещать в одном кристалле, появились DSP-микроконтроллеры, способные быстро обрабатывать звуковые данные.

Использование в промышленном оборудовании

ЦПУ, микроконтроллеры, DSP и тому подобные устройства применяются в различной окружающей нас технике.

Однокристальные микроконтроллеры содержатся почти во всех современных электронных настенных или наручных часах, будильниках; по нескольку таких микросхем используется в холодильниках, кондиционерах, стиральных машинах и другой бытовой технике. Разумеется, микроконтроллеры используются также в блоках управления телевизорами, кондиционерами, передающих и обрабатывающих информацию о нажатии кнопок на пульте инфракрасного дистанционного управления этими бытовыми приборами.

Кроме того, в промышленном оборудовании, которым нужно в какой-либо форме управлять (в роботах на автоматизированных линиях, автоматических конвейерах и т. п.), обязательно используются микроконтроллеры, ЦПУ или DSP.



Не только бытовая, но и промышленная техника...

Вплоть до настоящего времени масштабы производства и области применения микросхем определённых типов, например ЦПУ или однокристальных микроконтроллеров, содержащих ЦПУ и блок управления I/O, определялись технологическими особенностями производства полупроводников, а также балансом себестоимости и рыночной цены этих изделий.

Однако в последнее время благодаря развитию технологий изготовления IC стала использоваться микросхема под названием ПЛМ — программируемая логическая матрица, или программируемая пользователем вентильная матрица (англ. Field Programmable Gate Array, FPGA). О ней уже говорилось на стр. 85.

ПЛМ позволяет потребителю самому спроектировать логическую схему и воплотить её в качестве аппаратного обеспечения.

В одной микросхеме ПЛМ содержится от нескольких тысяч до миллиона таблиц перекодировки (look-up table). На начальном этапе использования она содержит память со списком таблиц перекодировки и линии разводки, которые, однако, ещё не подключены. Записав в неё с помощью специальных средств информацию о требуемой разводке, пользователь получает логическую микросхему, которую он сам спроектировал.

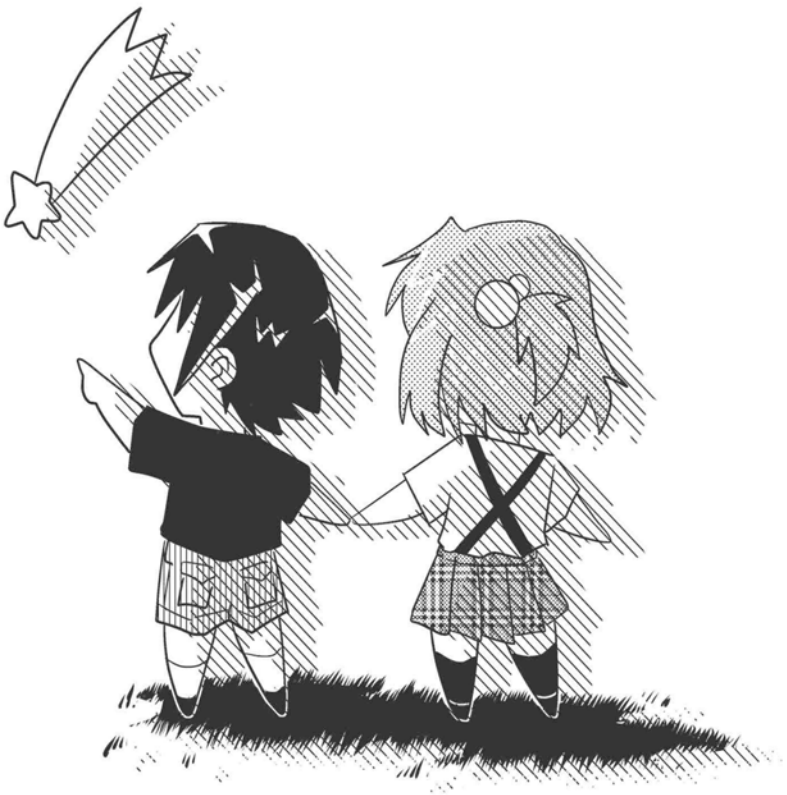
Разработку можно производить с помощью обычного персонального компьютера, а специальные средства для записи в ПЛМ можно подключать, например, через порт USB, что позволяет легко изготавливать большие логические схемы.

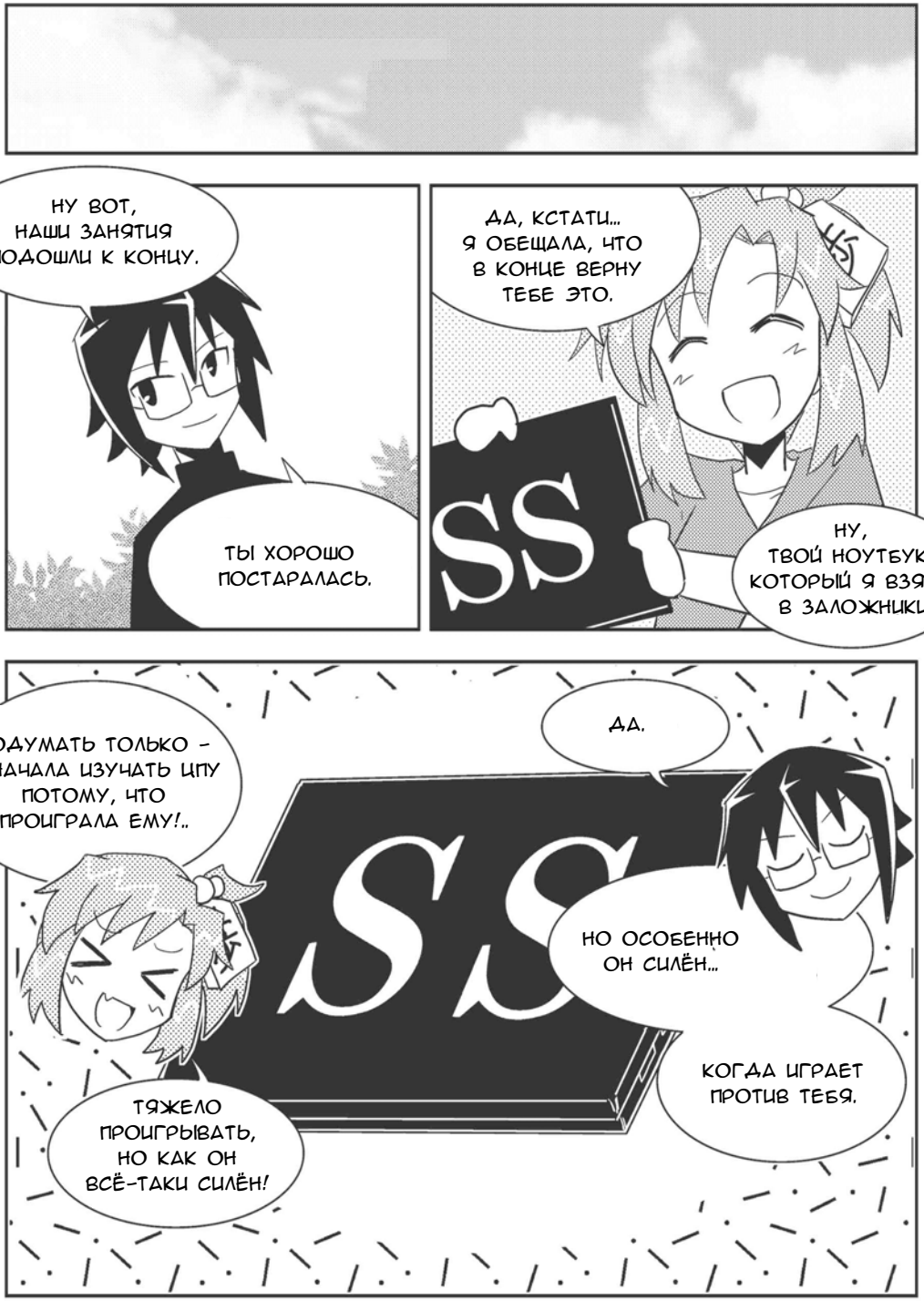
В прежние годы, даже несмотря на наличие ПЛМ, для ЦПУ приходилось использовать отдельную микросхему, однако в последнее время появилась возможность встраивать в ПЛМ и функцию ЦПУ в том числе. Это можно сделать двумя методами. В первом из них с помощью средств разработки в ПЛМ записывают информацию о разводке, позволяющей реализовать функции существующего ЦПУ вместе с его периферийной логикой, а в другом методе функции определённого ЦПУ встраивают в микросхему в качестве аппаратного обеспечения, отдельного от вентильной матрицы.

Как бы там ни было, можно сказать, что ЦПУ в качестве отдельных микросхем будут постепенно выходить из употребления.

Но подобно тому, как появление DSP не отменяет необходимости в функциях контроллера для управления блоками DSP, отличными от умножителя-аккумулятора, так и базовые принципы ЦПУ остаются важными, несмотря на всё большую интеграцию логических схем, осуществляемую производителями интегральных схем.

ЭПИЛОГ



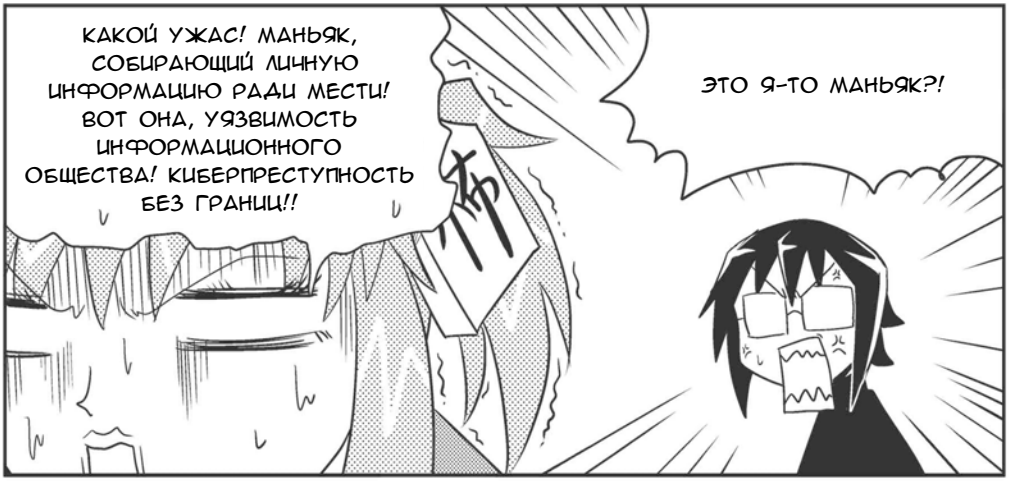

НУ ВОТ, НАШИ ЗАНЯТИЯ ПОДОШЛИ К КОНЦУ.
ТЫ ХОРОШО ПОСТАРАЛАСЬ.
ДА, КСТАТИ... Я ОБЕЩАЛА, ЧТО В КОНЦЕ ВЕРНУ ТЕБЕ ЭТО.
НУ, ТВОЙ НОУТБУК, КОТОРЫЙ Я ВЗЯЛА В ЗАЛОЖНИКИ.
ПОДУМАТЬ ТОЛЬКО – Я НАЧАЛА ИЗУЧАТЬ ЦПУ ПОТОМУ, ЧТО ПРОИГРАЛА ЕМУ!..
ТЯЖЕЛО ПРОИГРЫВАТЬ, НО КАК ОН ВСЁ-ТАКИ СИЛЁН!
ДА.
НО ОСОБЕННО ОН СИЛЁН...
КОГДА ИГРАЕТ ПРОТИВ ТЕБЯ.


НЕ ПОНЯЛА. ЧТО ТЫ ХОЧЕШЬ ЭТИМ СКАЗАТЬ?
НУ, ПРОСТО… ЕСТЬ У ВАС В КЛУБЕ ОДНА ЧЕРЕСЧУР ОБЩИТЕЛЬНАЯ ДЕВУШКА – ТВОЙ СТАРШИЙ ТОВАРИЩ.


ОНА В БЛОГЕ НЕ ТОЛЬКО ЗАПИСИ ХОДОВ, НО И ЛИЧНУЮ ИНФОРМАЦИЮ РАЗМЕЩАЕТ.
В целях защиты персональных данных лицо одноклубницы Кацураги Аюми размыто.
ДА! Я ЗНАЮ, КТО ЭТО!!!


ПЕРЕД ВОЗВРАЩЕНИЕМ В ЯПОНИЮ Я ВСПОМНИЛ О ТЕБЕ И ПОГУГЛИЛ ПО ИМЕНИ.
ИНФОРМАЦИЯ НАШЛАСЬ СРАЗУ ЖЕ.


ФОТОГРАФИИ С ПОСЛЕДНЕГО ТУРНИРА, ЗАПИСИ ХОДОВ…


КАКОЙ УЖАС! МАНЬЯК, СОБИРАЮЩИЙ ЛИЧНУЮ ИНФОРМАЦИЮ РАДИ МЕСТИ! ВОТ ОНА, УЯЗВИМОСТЬ ИНФОРМАЦИОННОГО ОБЩЕСТВА! КИБЕРПРЕСТУПНОСТЬ БЕЗ ГРАНИЦ!!
ЭТО Я-ТО МАНЬЯК?!

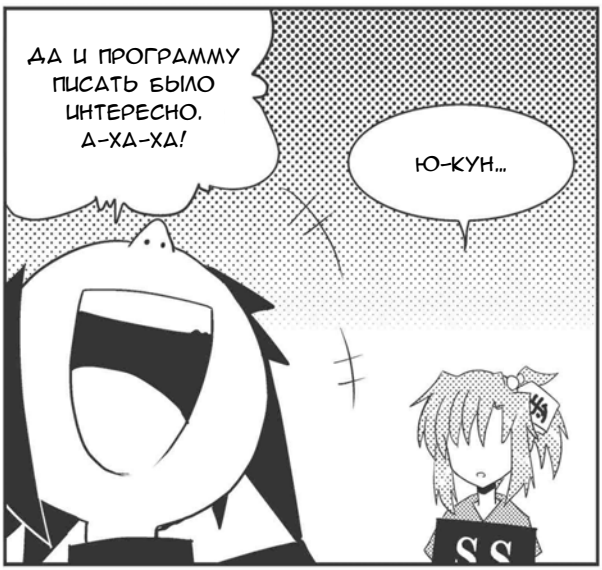

ТАК ВОТ, НА ФОТО У ТЕБЯ, ПОБЕДИТЕЛЬНИЦЫ, БЫЛ ТАКОЙ СКУЧАЮЩИЙ ВИД...


И, УВИДЕВ ТВОЁ ГРУСТНОЕ ЛИЦО, Я...


...ПОНИМАЮ! ТЫ ВСПОМНИЛ ТОТ ПОЗОР БЕСКОНЕЧНЫХ ПОРАЖЕНИЙ! И РЕШИЛ ОТОМСТИТЬ, НАПИСАВ "ПРОГРАММУ ИГРЫ ПРОТИВ КАЦУРАГИ АЮМИ", ТАК?!
ПРЕКРАТИ НАКОНЕЦ!!!

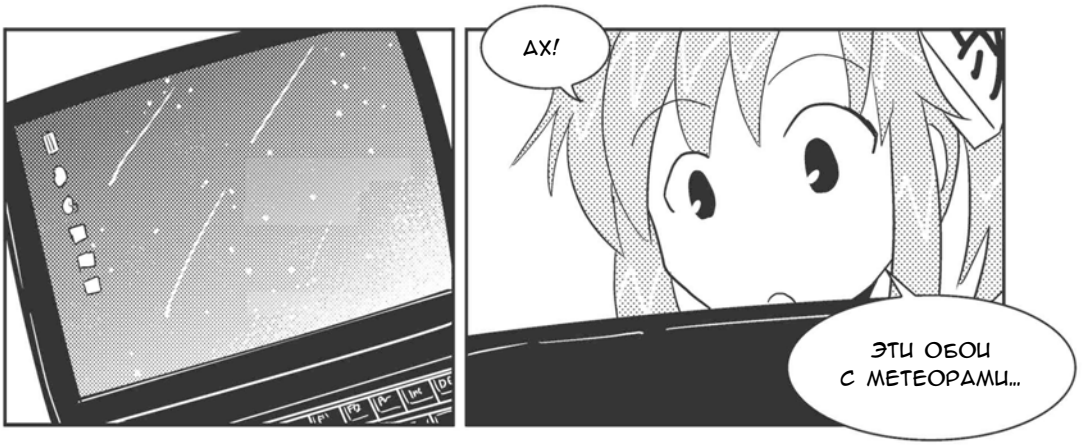

ЛАДНО. КАК БЫ ТАМ НИ БЫЛО, ЗАБИРАЙ СВОЙ SHOOTING STAR.
SS
ТЯЖЕЛО БЫЛО ПРОИГРЫВАТЬ, НО Я МНОГОМУ НАУЧИЛАСЬ.
ОСТАВЬ ЕГО СЕБЕ.
ЧТО?!
ЭТОТ НОУТБУК СИЛЁН В ИГРЕ ПРОТИВ ТЕБЯ.
И ТЕБЕ ОН, НАВЕРНОЕ, БУДЕТ ПОЛЕЗЕН.
ИСПОЛЬЗУЙ ЕГО КАК ЛЕКАРСТВО ОТ СКУКИ И ТРЕНАЖЁР.
Я С САМОГО НАЧАЛА СОБИРАЛСЯ ТЕБЕ ЕГО ПОДАРИТЬ.


ЗНАЧИТ...
...ЭТО БЫЛО НЕ ТОЛЬКО....
...ОРУЖИЕ МЕСТИ...
...НО И ПОДАРОК, КОТОРЫЙ ТЫ МНЕ ГОТОВИЛ?
ДУМАЙ КАК ХОЧЕШЬ.
И МОЖЕШЬ МНЕ НИЧЕМ НЕ ОТПЛАЧИВАТЬ. У МЕНЯ ЭТИХ НОУТБУКОВ ПОЛНО!


ДА И ПРОГРАММУ ПИСАТЬ БЫЛО ИНТЕРЕСНО. А-ХА-ХА!
Ю-КУН...


ЧТО?
ЕЩЁ КАКИЕ-ТО ВОПРОСЫ?


Я ТАК РАДА!
СПАСИБО!
БУДУ ЕГО БЕРЕЧЬ!
... !
ТУ-ТУМ
АХ...
ТЕПЕРЬ ОН МОЙ!
ВСПЫХ
ЩЁЛК
ПОПРОБУЮ ВКЛЮЧИТЬ!

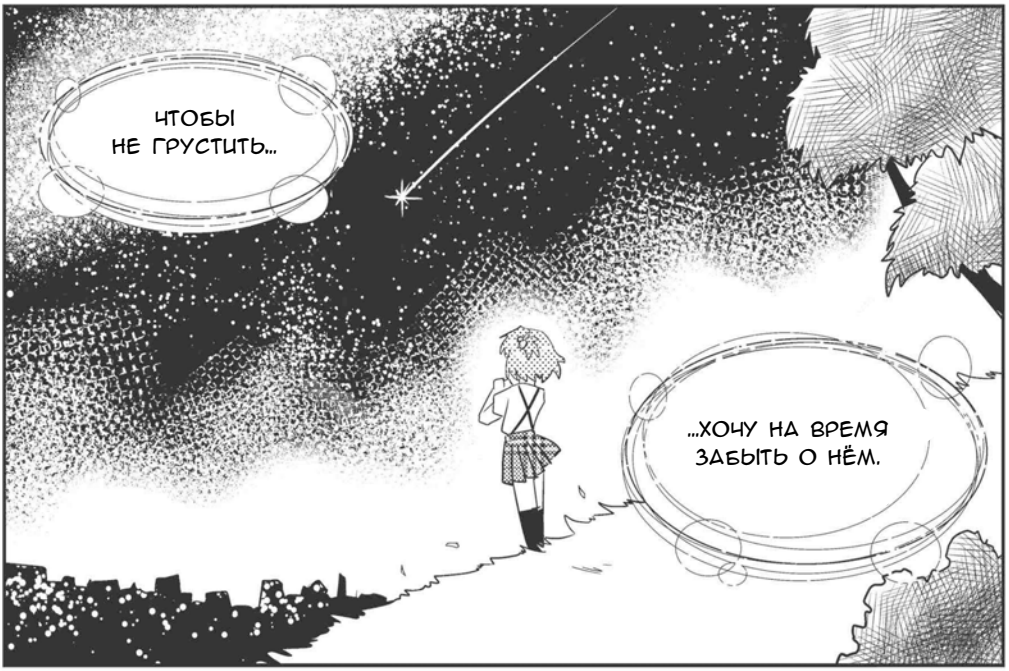

АХ!
ЭТИ ОБОИ С МЕТЕОРАМИ...


КАЖЕТСЯ, Я ВСПОМИНАЮ...
КОГДА-ТО ДАВНО Я ХОДИЛА В ПАРК С МАЛЬЧИКОМ, КОТОРЫЙ УЕЗЖАЛ ЗА ГРАНИЦУ... ПОСМОТРЕТЬ НА ЗВЕЗДОПАД...


НАКОНЕЦ-ТО ТЫ ВСПОМНИЛА!
И Я ЗАГАДАЛА ЖЕЛАНИЕ НА ПАДАЮЩУЮ ЗВЕЗДУ:
"ХОЧУ ПОЛУЧИТЬ В НАСЛЕДСТВО ОГРОМНОЕ СОСТОЯНИЕ!"
ОХ ТЫ И ЖАДНА!.. БОЛЬШЕ НИЧЕГО НЕ ПОЖЕЛАЛА?!


ЭТО Ж БЫЛ ЗВЕЗДОПАД.
Я МНОГО ЧЕГО ЗАГАДАЛА.
НУ НАПРИМЕР?


ТАК...
Ю-КУН ЗАВТРА УЕЗЖАЕТ ДАЛЕКО-ДАЛЕКО. МНЕ БУДЕТ ГРУСТНО БЕЗ НЕГО.


ЧТОБЫ НЕ ГРУСТИТЬ...
...ХОЧУ НА ВРЕМЯ ЗАБЫТЬ О НЁМ.


ОЙ!!!
ЭТО ЧТО?!
Я ВСПОМНИЛА!!!
ВПРОЧЕМ, НЕВАЖНО! ЭТО СЕКРЕТ.
А САМ-ТО ТЫ ЧТО ЗАГАДАЛ?!
… ! !
НУ, КАК БЫ ЭТО СКАЗАТЬ…
Я ХОЧУ КОГДА-НИБУДЬ ВЕРНУТЬСЯ СЮДА…
…И СНОВА ПОВСТРЕЧАТЬСЯ С ДЮМИ.
ЧТОБЫ… ЧТОБЫ Я НАУЧИЛСЯ ЛУЧШЕ ИГРАТЬ В СЁГИ. ДА!
ХА-ХА! В ЭТОМ НИ ЗВЁЗДЫ, НИ БОГИ ТЕБЕ ПОМОЧЬ НЕ СМОГУТ!
Я ВЕДЬ ПРОИГРАЛА ЦПУ, А НЕ ТЕБЕ!!!
ТА-АК! НАЧИНАЕТСЯ!


АГА, ОБИДЕЛСЯ?
ТОГДА ПОПРОБУЙ САМ
СРАЗИТЬСЯ СО МНОЙ!
ДУРАЧИНА!
ЧТО? ОПЯТЬ
ЗА СТАРОЕ?!
КАЦУРАГИ АЮМИ,
ТЫ НЕ УМЕЕШЬ
ПРОИГРЫВАТЬ!
КОНЕЦ

Послесловие. Тенденции современных ЦПУ

В этой книге рассмотрено базовое устройство классических примитивных ЦПУ, которые сейчас используются лишь как ЦПУ микроконтроллеров простейших электронных устройств: игрушек, пультов управления кондиционерами для дома и т. п. Так как в условиях стремительного технического прогресса информация очень быстро устаревает, в манге представлены только самые базовые принципы ЦПУ, чтобы она как можно дольше могла служить подспорьем для изучающих вычислительную технику.

Вопросы, касающиеся архитектуры компьютеров, здесь не затронуты, однако в заключение нелишне будет поговорить о современных тенденциях в этой области. Устройство сложных современных ЦПУ трудно выразить простым языком и с помощью упрощённых схем, таких как приведённая в начале книги схема ЦПУ, поэтому опишем только основные моменты.

В широко используемых ЦПУ (например, современных ПК) ради повышения скорости выполнения программ используются различные хитрости. Так, достаточно давно стала использоваться *предварительная выборка команд* (упреждающая выборка, предвыборка; англ. instruction prefetch), при которой следующая команда считывается (выбирается) из памяти в ЦПУ ещё до того, как закончится выполнение текущей, что позволяет сократить время, затрачиваемое на выборку команд из памяти. Если к тому же заранее декодировать эту предварительно выбранную команду и подготовить ЦПУ к её выполнению, то скорость выполнения ещё более возрастёт.

Другая хитрость заключается в том, что каждый из функциональных блоков ЦПУ (выборки команд, декодирования, выполнения операции, записи/чтения ячеек памяти, записи данных в регистры и т. д.) временно запоминает своё состояние и передаёт его следующему блоку сразу же, как тот освободится, что уменьшает «время простоя» функциональных блоков и создаёт видимость параллельного выполнения команд. Это называется *конвейерной архитектурой*.

Кроме того, в давно проводившихся исследованиях выяснилось, что определённые коды операций имеют тенденцию выполняться над операндами строго определённых типов. Это позволило упростить коды операций, создав ЦПУ с архитектурой RISC (Reduced Instruction Set Computer — компьютер с сокращённым набором команд).

Раньше опасались, что сокращение набора команд приведёт к увеличению количества команд, необходимых для выполнения сложных вычислений, и к замедлению выполнения программ, но опасения оказались беспочвенными. Архитектура RISC упрощает ЦПУ на аппаратном уровне и сокращает время выполнения одной команды, что позволяет увеличить тактовую частоту. В результате скорость выполнения большинства программ на ЦПУ архитектуры RISC, наоборот, увеличивается.

Архитектура RISC стала использоваться в самых разнообразных устройствах, а существовавшую прежде архитектуру стали называть CISC (Complex Instruction Set Computer — компьютер со сложным (полным) набором команд). Сам термин CISC вошёл в употребление как антоним RISC и просто указывает на ЦПУ прежней архитектуры. Другими словами, какой-то особой «архитектуры CISC» в действительности не существует.

Современные микросхемы ЦПУ, например производства фирмы Intel, содержат несколько ядер и реализуют распределённую обработку данных на определённых этапах выполнения программы. Здесь мы не будем подробно рассматривать это революционное новшество в архитектуре компьютеров, так как оно выходит за рамки базового устройства ЦПУ. Отметим только, что в некоторых случаях сложных вычислений бывает выгодно выполнять команды не в том порядке, в котором они содержатся в программе. В подобных случаях для повышения скорости обработки вычисления распределяются между несколькими ядра ЦПУ, которые работают параллельно и независимо друг от друга, но могут при необходимости обмениваться данными или координировать работу над одной и той же вычислительной задачей.

Для реализации возможностей таких ЦПУ необходимы также программные ухищрения, например операционные системы, координирующие выполнение машинных команд и управляющие обращениями к памяти.

Итак, в этой манге рассматривалась лишь самая элементарная архитектура однокристальных ЦПУ. Но стоит надеяться, что книга поспособствует развитию исследований в области базовых концепций ЦПУ, в том числе и в Японии.

ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ

Латиница

А

К

Л

М

Н

О

П

Р

С

• Автор
Сибуя Митио

В 1971 году закончил кафедру электроники технологического факультета университета Токай. Работал исследователем в области ЯМР и других областях. С 1979 года в течение 12 лет занимался разработкой, проектированием и организацией производства изделий МОП в международной компании по производству полупроводниковых приборов. После этого работал в технических отделах японской компании по продаже полупроводниковых приборов, международной компании по производству ИС, отвечая за проектирование и разработку микросхем. В мае 2007 года вступил в должность инженера по технической поддержке компании Sankyosha Co., Ltd. и в настоящее время является специальным консультантом данной компании.

• Сценарий
Савада Савако

• Художник
Тонаги Такаси

• Оформление
Office sawa

Книги издательства «ДМК Пресс» можно заказать в торгово-издательском холдинге «Планета Альянс» наложенным платежом, выслав открытку или письмо по почтовому адресу: 115487, г. Москва, 2-й Нагатинский пр-д, д. 6А.

При оформлении заказа следует указать адрес (полностью), по которому должны быть высланы книги; фамилию, имя и отчество получателя. Желательно также указать свой телефон и электронный адрес.

Эти книги вы можете заказать и в интернет-магазине: **www.alians-kniga.ru**.

Оптовые закупки: тел. **(499) 782-38-89**.

Электронный адрес: **books@alians-kniga.ru**.

Сибуя Митио (автор), Тонаги Такаси (художник)

Занимательная информатика.
Центральный процессор. Манга

Главный редактор *Д. А. Мовчан*
dmkpress@gmail.com
Перевод с японского *А. Б. Клионский*
Корректор *О. В. Готлиб*
Вёрстка *А. Б. Клионский*
Обложка *А. Г. Мовчан*

Формат 70×100 1/16.
Гарнитура Anime Ace. Печать офсетная.
Усл. п. л. 19. Тираж 500 экз.

Веб-сайт издательства ДМК Пресс: www.dmkpress.com

ОБРАЗОВАТЕЛЬНАЯ МАНГА

ЗАНИМАТЕЛЬНАЯ ИНФОРМАТИКА

ЦЕНТРАЛЬНЫЙ ПРОЦЕССОР

КАЦУРАГИ АЮМИ, ЧЕМПИОНКА ПО ЯПОНСКИМ ШАХМАТАМ СЁГИ, ВСТРЕЧАЕТ ТАИНСТВЕННОГО НЕЗНАКОМЦА, КОТОРЫЙ ПРЕДЛАГАЕТ ЕЙ СЫГРАТЬ ПАРТИЮ С КОМПЬЮТЕРОМ. КТО ОДЕРЖИТ ВЕРХ В ЭТОМ ПОЕДИНКЕ – ЧЕЛОВЕК ИЛИ МАШИНА? И КАКУЮ ТАЙНУЮ ЦЕЛЬ ПРЕСЛЕДУЕТ ЗАГАДОЧНЫЙ ПРОГРАММИСТ?

В КНИГЕ ПРОСТО И ДОСТУПНО ОБЪЯСНЯЮТСЯ ОСНОВЫ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ, РАССКАЗЫВАЕТСЯ ОБ УСТРОЙСТВЕ КЛАССИЧЕСКОГО ЦЕНТРАЛЬНОГО ПРОЦЕССОРА (ЦПУ), ПРИНЦИПАХ ЕГО РАБОТЫ И ОБЛАСТЯХ ПРИМЕНЕНИЯ.

Интернет-магазин: www.dmkpress.com
Книга-почтой: orders@alians-kniga.ru
Оптовая продажа: “Альянс-книга“.
(499)782-3889. books@alians-kniga.ru

ДМК ПРЕСС
ИЗДАТЕЛЬСТВО
www.дмк.рф

ISBN 978-5-97060-507-3