Иммунитет как результат эволюции ЭВМ

En Immunity as the Result of the Computers' Evolution

V. A. Konyavskiy,
PhD (Engineering, Grand Doctor),
Head of Information Security Department
Moscow Institute of Physics and Technology
konyavskiy@gospochta.ru

The article introduces computers' evolution history, which makes scantiness of all traditional compensating for vulnerabilities approaches to be vivid. Cheap and simple ones are not effective, and effective are expensive and difficult in use as they are the devices changing base computers' architecture. The alternative, offered in the article is new architecture, that became natural evolution stage, prepared by the most famous Russian and World scientists' and engineers' works.

Keywords: computers' architecture, Turing machine, von Neumann architecture, Harvard architecture, New Harvard architecture, Setun', MIR Computer, ПС-2000, Elbrus

В статье описана история эволюции ЭВМ, которая иллюстрирует ограниченность всех традиционных подходов к компенсации их уязвимостей. Простые и дешевые подходы неэффективны, а эффективные – сложны и дороги, поскольку представляют собой устройства, корректирующие базовую архитектуру ЭВМ. Предлагаемая в статье альтернатива – новая архитектура, являющаяся закономерным этапом эволюции, подготовленным исследованиями и разработками известнейших российских и мировых ученых и корпораций

Ключевые слова: архитектура компьютера, машина Тьюринга, архитектура фон Неймана, гарвардская архитектура, новая гарвардская архитектура, «Сетунь», ЭВМ МИР, ПС-2000, «Эльбрус»

Валерий Аркадьевич Конявский,
доктор технических наук, заведующий кафедрой «Защита информации», факультет радиотехники и кибернетики
Московский физико-технический институт
konyavskiy@gospochta.ru

Электронно-вычислительные машины (ЭВМ, компьютеры) предназначены для выполнения программ – различных алгоритмов, описанных на языках программирования. Здесь слово «различных» практически означает «любых», то есть компьютер выполняет любые программы. Однако заранее определенные функции выполняют не только компьютеры – например, это делают также конечные автоматы. Отличие между ними – именно в том, что компьютер исполняет «любые» программы, по сравнению с конечным автоматом он несравнимо более гибок, универсален.

Любой компьютер – это реализация (более или менее близкая) идеи «машины Тьюринга» [1].

Понятия «машина Тьюринга» и «алгоритм», «вычислимость» неразрывно связаны, определяются одно через другое. Само существование абстрактного «исполнителя», такого, как машина Тьюринга, вселяет уверенность во всемогущество человека. Действительно, любая (точнее, рекурсивная, что и есть практически любая) задача может быть решена, если для этого имеется достаточно ресурсов (памяти и времени).

Возможно, завораживающая простота формулировок и спровоцировала разработку универсальных вычислительных машин (средств вычислительной техники, СВТ; персональных ЭВМ, ПЭВМ), которые частично (с конечной памятью) моделируют машину Тьюринга, давая нам псевдонеограниченные возможности и толкая на экстенсивный путь

развития. Не хватает памяти – что за проблема – добавим. Не хватает времени – увеличим тактовую частоту, количество ядер, виртуализируем ресурсы, наконец.

Эта позиция многие годы «паровозом» тащила за собой развитие информационных технологий. Емкость обычных локальных дисков, например, за два десятилетия выросла от десятков килобайт до сотен гигабайт и часто уже измеряется терабайтами, а памяти так и не хватает. Тактовые частоты от килогерц достигли гигагерц, а производительности не хватает. Зато индустрия ИТ стала едва ли не определяющей современный уровень экономического развития. Гигантские суммы инвестиций – плата за технический прогресс и универсальность решений.

На машине Тьюринга можно [1] моделировать любую другую вычислительную машину (любого «исполнителя») – поэтому говорят о полноте машины Тьюринга.

С указанными выше ограничениями (конечностью памяти) на универсальной машине (УМ) можно, в свою очередь, моделировать машину Тьюринга. Такие УМ называют полными по Тьюрингу (Turing complete).

Полные по Тьюрингу УМ, таким образом, как минимум, должны выполнять элементарные операции, свойственные машине Тьюринга, а именно: перемещать управляющее устройство (головку чтения/записи) влево и вправо по ленте, читать и записывать в ячейки символы некоторого конечного алфавита. Линейность памяти и последовательность выполнения операций – базовые характеристики машины Тьюринга.

УМ потенциально может самообучаться, достигая уровня, при котором человек, ведущий беседу, не сможет определить, кто его собеседник – человек или компьютер (тест Тьюринга). Конечно, это блестящие перспективы, само существование которых делает необычайно привлекательной сферу компьютерных технологий!

Но всегда ли возможность самообучения является плюсом? Серьезные опасения, например, может вызвать возможность бесконтрольного самообучения контроллеров АСУ ТП АЭС, железнодорожного транспорта, непрерывных производств и т. д. Вряд ли эти контроллеры должны блистать интеллектом вместо того, чтобы абсолютно точно выполнять возложенные на них функции. Видимо поэтому для решения таких задач применяют обычно конечные автоматы, которые наряду с контекстно-свободными грамматиками, примитивно-рекурсивными функциями являются примерами неполных по Тьюрингу формализмов.

Таким образом, есть много задач, которые *можно* решить и не на универсальном «исполнителе». Более того, есть много задач, которые *нужно* решать не на универсальном, а на специализированном «исполнителе». Так, неотчуждаемая возможность «читать и записывать», которая делает операцию копирования в компьютере имманентной, полностью противоречит, как минимум, задачам защиты информации. Необходимое для «универсальности» свойство становится неприемлемым в конкретных условиях.

Пытаясь защититься от вредоносных хакерских программ, человечество уже более 60 лет разрабатывает продукты, традиционно относимые к области защиты информации – средства идентификации, аутентификации, авторизации, контроля целостности, антивирусное ПО, криптографические средства и т. д. Их использование отчасти приносит положительный эффект, но очень и очень «отчасти». Действуя в рамках пусть универсальной, но одной формальной модели, мы неизбежно натолкнемся на ее неполноту, в точном соответствии с теоремой Геделя о неполноте.

Становится очевидным, что искать уязвимости только в ПО явно недостаточно.

Действительно, если УМ выполняет любые программы, то, очевидно, она выполнит и вредоносную программу. Это не зависит от используемого в ней программного обеспечения, а определяется ее архитектурой. Универсальность компьютера обеспечивается архитектурно, самой «конструкцией» машины Тьюринга как мыслимой в абстракции, так и реализованной на практике. Способность выполнять вредоносные программы – это базовая, системная, архитектурная уязвимость всех компьютеров, построенных как машина Тьюринга. Уязвимость – оборотная сторона универсальности. Машина Тьюринга архитектурно уязвима. Архитектурно уязвимы и все виды компьютеров, которые мы используем, потому что они разрабатывались так, чтобы быть максимально универсальными. Этой уязвимостью мы платим за универсальность наших компьютеров. Мы эксплуатируем компьютеры, а хакеры эксплуатируют уязвимость.

Поскольку архитектуру нельзя изменить программным путем, то никакие программные средства не помогут нам надежно защититься от хакеров. Игра «кто кого» продолжается уже много лет, давая работу сотням тысяч специалистов по информационной безопасности, но не спасая нас от потерь.

Как же быть?

Если уязвимость заключается в архитектуре, то и совершенствовать нужно архитектуру.

Классическими являются две архитектуры – архитектура фон Неймана [2] и гарвардская архитектура [3]. Примером первой являются практически все настольные компьютеры, примером второй – практически все планшетные компьютеры и телефоны.

Архитектура компьютера, о котором пойдет речь ниже, отличается от обеих упомянутых архитектур. Однако данный вариант новой или измененной (модифицированной) архитектуры – не первый случай отступления от классики. Рассмотрим некоторые из них.

Известны [4] принципы организации вычислительного процесса фон Неймана, а именно:

П1. Использование двоичной системы счисления в вычислительных машинах.

П2. Программное управление ЭВМ. Работа ЭВМ контролируется программой, состоящей из набора команд. Команды выполняются последовательно.

П3. Память компьютера используется для хранения не только дан-

ных, но и программ. При этом и команды программы, и данные кодируются в двоичной системе счисления, то есть способ их записи одинаков, поэтому в определенных ситуациях над командами можно выполнять те же действия, что и над данными.

П4. Ячейки памяти ЭВМ имеют адреса, которые последовательно пронумерованы. В любой момент можно обратиться к любой ячейке памяти по ее адресу.

П5. Возможность условного перехода в процессе выполнения программы. Несмотря на то, что команды выполняются последовательно, в программах можно реализовать возможность перехода к любому участку кода.

Анализ истории развития вычислительной техники показывает, что большинство из упомянутых выше принципов неоднократно нарушались и зачастую это приносило неожиданные положительные результаты. Рассмотрим некоторые исторические примеры.

В 1958 году под руководством Н. П. Брусенцова был разработан опытный образец ЭВМ «Сетунь» [5], а в 1961 году состоялся ее запуск в серию. Особенностью этой вычислительной машины было использование троичной системы счисления с коэффициентами (1, 0, –1). Главное преимущество троичного представления чисел перед фоннеймановским (двоичным) представлением заключается в том, что оно позволяет реализовать естественное представление натурального ряда чисел со знаком.

В любой системе с нечетным количеством цифр можно создать «симметричную» систему чисел со знаком. Троичная система – самая простая для технической реализации из числа подобных систем. При этом особенности вычислений дают ощутимое преимущество в скорости операций и их энергоемкости. Так, в троичном сумматоре перенос в следующий разряд возникает в 8 ситуациях из 27, а в двоичном – в 4 из 8. Кроме этого, упрощается операция умножения: умножение на –1 просто инвертирует множимое. Такое упрощение вычислений не исчерпало себя, и вполне можно ожидать появления новых решений, основанных на троичной системе счислений.

Таким образом, в ЭВМ «Сетунь» очевидным образом нарушен П1.

В 1967 году на выставке в Лондоне демонстрировалась ЭВМ МИР-1, которая тогда произвела фурор. Важной особенностью этой ЭВМ был язык высокого уровня, непосредственно исполняемый машиной.

Язык «Аналитик» представлял собой набор конструкций для описания вычислительных схем и развивался в направлении алгебраических спецификаций таковых. Разработчики [5] реализовали все стандартные рекурсивные функции, включая их в состав языка[1]. Настоящий, завершенный язык, пригодный для описания вычислений, функций и процедур не может состоять только из набора рекурсивных функций, и по совету академика А. А. Дородницына пришлось включить в него оператор перехода, завершив создание языка, и заодно констатировав, что простое отрицание принципов фон Неймана не обязательно приводит к успеху. В то же время нельзя эти принципы считать догмами: например, принцип последовательного исполнения команд целесообразно исповедовать только в самых простых случаях. Использование мультикомандного принципа зачастую приводит к намного более эффективным решениям.

В 1974 году на конгрессе IFIP В. М. Глушков высказал мнение [6] о том, что только разработка принципиально новой – не-неймановской – архитектуры вычислительных систем позволит решить проблему построения суперЭВМ с неограниченным ростом производительности. Такие решения были найдены Глушковым и положены в основу оригинальной структуры высокопроизводительной ЭВМ, названной им макроконвейером.

Суть принципа макроконвейерной обработки данных заключается в следующем: ЭВМ содержит не один, а много процессоров, и каждому процессору на очередном шаге вычислений дается такое задание, которое позволяет ему на протяжении длительного времени работать автономно, без взаимодействия с другими процессорами.

Исполняемый язык высокого уровня и принцип макроконвейерной обработки нарушают принципы фон Неймана (П2, П3), что не мешает основанным на них решениям быть эффективными.

Отметим, что рассмотренные выше идеи позднее были использованы в макроконвейерных многостековых сопроцессорах безопасности «Аккорд-КПД» и «Аккорд-СБ» (рис. 1) [7]. Они использовались в организациях кредитно-финансовой системы страны для того, чтобы в огромном потоке финансовых документов каждый из них был «подписан» в процессе создания и проверен на всех этапах обработки. Устанавливались они в большие быстродействующие ЭВМ и преследовали основную задачу – добиться высокой производительности и линейного ее увеличения при установке нескольких сопроцессоров, не занимая ресурсов основной вычислительной машины, которая решала функциональные задачи. Шина при этом оставалась довольно медленной. Вот здесь и пригодился принцип макроконвейера: вначале загружалась задача, перестраивалась архитектура вычислителя, потом отправлялся некоторый массив данных, и пока он обрабатывался, в стек загружались следующие данные.

В 1961 году Burroughs Corporation [8] выпустила двухпроцессорный компьютер с виртуальной памятью B5000. Его уникальными для своего времени особенностями были, в частности, адресация на основе дескрипторов, а также использование языка высокого уровня («Алгол») в качестве входного, как и ранее, в ЭВМ «МИР».

[1] *Здесь уже проявилась ориентация авторов на «последовательный отказ от хорошо известных принципов фон Неймана», в частности, это касается принципа максимальной простоты системы команд. Естественно, аппаратная интерпретация языка высокого уровня требовала ощутимой динамической «перестройки» ЭВМ в процессе исполнения задач.*

При дескрипторной (теговой) организации памяти каждое слово содержит не только информационную, но и управляющую часть – тег элемента. Использование тегов позволяет резко снизить количество ошибок, выбирая (ограничивая) перед исполнением нужный вариант операции и контролируя операнды. При такой организации «лампольсины» никогда не произведутся, так как невозможно сложение «лампочек» и «апельсинов». Нарушение принципов П2 и П3 дает положительный эффект.

Идеи, заложенные в ЭВМ В5000 и «МИР», принципиально важны. При их создании впервые был опробован механизм динамического изменения структуры ЭВМ в соответствии с исполняемой программой и показано, что при разработке ЭВМ необходимо понимать, какие программы будут на ней исполняться. И обратное – создавая программное обеспечение, необходимо понимать архитектуру компьютера.

Использование «Алгола» в качестве управляющего языка и теговая организация памяти впоследствии стали также отличительными признаками ЭВМ «Эльбрус», которая разрабатывалась коллективом академика В. С. Бурцева, говорившего по этому поводу [9]: «При решении сложных задач возникает проблема быстрого перераспределения вычислительных ресурсов и коммутации информационных потоков. Из-за этого эффективность использования одного процессора снижается до уровня 5–10 % и ниже. Всё это происходит потому, что по-прежнему применяется фоннеймановская схема вычислительного процесса. Нам удалось уйти от фоннеймановской структуры обработки информации – в нашей машине отдельные операции, даже скалярные на определенном интервале времени, могут выполняться независимо одна от другой. Вычислительные ресурсы распределяются аппаратно».

В 1972–1975 годах в Институте проблем управления РАН под руководством академика И. В. Прангишвили была разработана много-

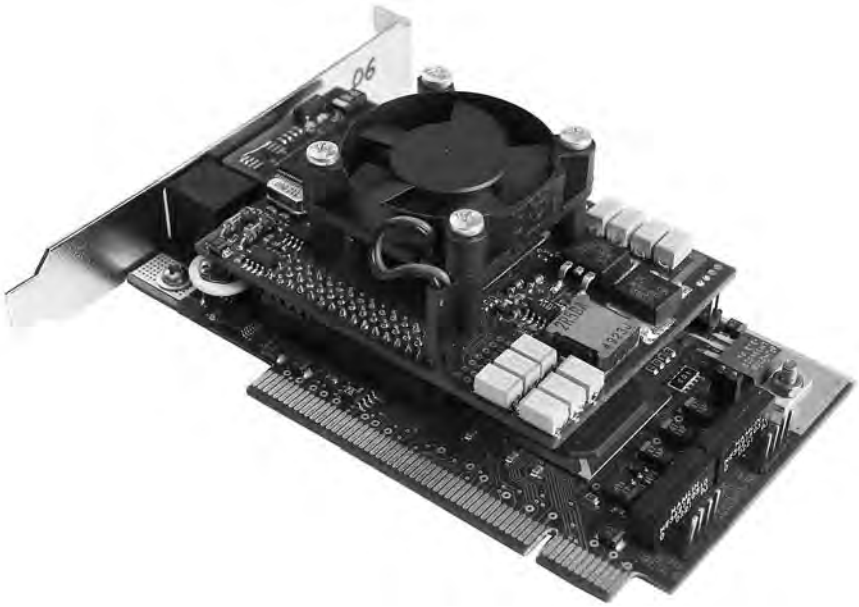

Рис. 1. «***Аккорд-СБ***»

процессорная машина с одним потоком команд и многими потоками данных – ПС-2000 [10]. Управление состоянием ПС-2000 осуществлялось с помощью ЭВМ СМ-2М. Уже в то время говорили, что аббревиатура ПС в названии комплекса означает «перестраиваемые структуры».

Дальнейшим развитием этих идей стала машина ПС-3000 (ее серийный выпуск начался в 1979 году), в которой в полной мере были реализованы идеи динамической перестраиваемости структуры. В основе архитектурных принципов организации МВК ПС-3000 [11] лежала динамическая перестраиваемость ее структуры согласно текущим требованиям параллельных вычислительных процессов. Перераспределение ресурсов осуществлялось как программно, так и аппаратно, оптимизируя структуру комплекса под текущую задачу.

Эти и многие другие примеры показывают, что принципы организации вычислительного процесса по фон Нейману многократно ставились под сомнение ведущими мировыми разработчиками ЭВМ, и часто «не зашоренный» подход давал положительные результаты, хотя эксперименты обходились недешево. Верно и обратное: ориентация на универсализм (организацию вычислительного процесса по принципам фон Неймана) на предыдущем этапе развития техники придала мощный импульс информационным технологиям, но одновременно надолго остановила продуктивные исследования в области компьютерных архитектур.

В последние годы техника совершила огромный скачек вперед, но это почти никак не отразилось на архитектурах компьютеров. Наиболее заметное изменение в этой области – опережающий рост решений на базе гарвардской архитектуры. Если несколько лет назад объемы продаваемых процессоров типов x86 и ARM соотносились как 80:20, то сегодня уже 50:50, и эта тенденция усиливается[2]. Таким образом, можно предполагать, что сложились условия для размышлений над усовершенствованием архитектуры.

При разработке компьютера главное – понять, какая часть функций должна быть реализована аппаратно, а какая – программно. Правильный выбор этого соотношения позволил ПЭВМ с архитектурой фон Неймана многие годы занимать лидирующее положение в мире вычислительных систем. Однако сейчас практически достигнут предел эффективности данного технического решения, и для совершенствования компьютеров требуется устранить имеющиеся уязвимости. В аппаратную часть нужно включать то, что

[2] *Оценка дана акад. Б. А. Бабаяном (INTEL) в беседе с автором, июнь 2015 года.*

снижает стоимость, редко изменяется, расширяет возможности и используется постоянно. Кроме этого, сейчас совсем не выглядит экзотической мысль о том, что в процессе работы структура компьютера может динамически изменяться. Или, например, вначале структура может быть такой, как конечный автомат, а потом, на следующем этапе, стать «универсальным исполнителем» по Тьюрингу.

Отличительной особенностью архитектуры фон Неймана является то, что команды и данные в ней не разделяются, а передаются по единому каналу (рис. 2).

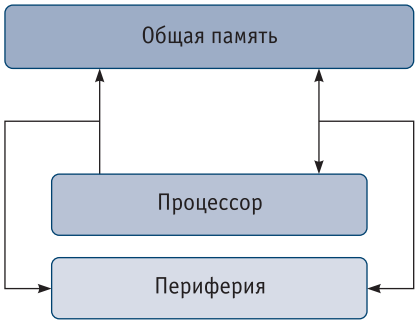

Рис. 2. ***Архитектура фон Неймана***

Гарвардская архитектура предполагает наличие разных каналов для команд и данных (рис. 3).

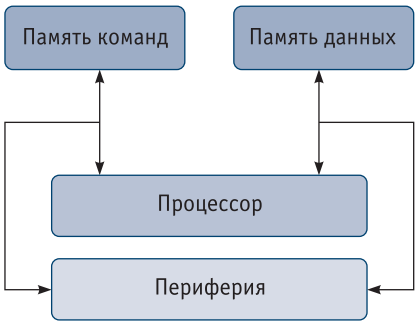

Рис. 3. ***Гарвардская архитектура***

Такая схема взаимодействия требует более сложной организации процессора, но обеспечивает более высокое быстродействие, так как потоки команд и данных становятся не последовательными, а параллельными, независимыми.

Однако и в случае компьютера фоннеймановского типа, и компьютера с гарвардской архитектурой организация потоков команд и данных таковы, что архитектурная уязвимость присуща каждому из них. Гибкость, универсальность в обоих случаях обеспечивается возможностью изменения последовательности команд и данных (двунаправленные стрелки от процессора к памяти на рис. 3) – независимо от того, в одной памяти они лежат, или разделены. В свою очередь, возможность изменения последовательности команд и данных создает и возможность для несанкционированного вмешательства вредоносного ПО – в этом и заключается основная архитектурная уязвимость.

На использовании этой уязвимости основаны практически все современные хакерские атаки, которые в основном сводятся к атаке на «перехват управления». Схема атаки обычно выглядит так:

1) вредоносное ПО (ВрПО) внедряется на компьютер и размещается в его оперативной памяти;

2) внедряется и размещается в оперативной памяти вредоносный обработчик прерываний;

3) ВрПО и обработчик прерываний записываются в долговременную память;

4) с помощью любого доступного механизма, например с помощью DDOS-атаки, вызывается прерывание;

5) внедренный ранее обработчик прерываний срабатывает и передает управление ВрПО;

6) ВрПО выполняет свою функцию, например, реализует разрушающее программное воздействие.

Здесь 1–3 – это шаги по подготовке атаки, 4 – инициирование атаки, 5 и 6 – собственно использование архитектурной уязвимости.

Для обезвреживания шагов 1 и 2 обычно используются антивирусные программы. Иногда это бывает полезным, но только иногда, поскольку невозможно с помощью антивирусных программ выявить все ВрПО. Более того, специалистам известны конструкции ВрПО, которые точно нельзя обнаружить. Можно даже сказать, что компьютерные вирусы и в целом ВрПО удается обнаружить только в силу их несовершенства. В общем случае всегда можно разработать такое ВрПО, которое не может быть обнаружено с помощью антивирусных программ сигнатурного поиска, эвристических анализаторов и поведенческих блокираторов.

Блокирование последствий выполнения шага 3 выполняется при последующей загрузке с помощью механизмов контроля целостности – по сути, ревизоров, определяющих, есть ли изменения в составе данных. Иногда эта проверка выполняется с помощью тех же наборов антивирусных программ, но это слабое решение, так как проверка должна выполняться до загрузки ОС, а программы, в том числе и антивирусные, работают под управлением ОС.

Генерация события на шаге 4 частично блокируется с помощью специальных средств анализа трафика, устанавливаемых как в сети, так и на клиентских компьютерах. Важно то, что пока нет средств, позволяющих гарантированно блокировать эту уязвимость.

Негативные последствия шагов 5 и 6 блокируются с помощью механизмов контроля запуска задач (процессов, потоков). Это очень эффективные механизмы, но реализующие их средства, во-первых, довольно дороги, а во-вторых, для их настройки нужно быть специалистом в области компьютерных технологий и информационной безопасности.

Поскольку некоторые из перечисленных функций безопасности должны выполняться до загрузки операционной системы, то реализовать их можно только с помощью сложного устройства и нельзя реализовать программно.

Примером такого устройства является СЗИ НСД «Аккорд» [12]. Это аппаратный модуль доверенной загрузки с программным комплексом разграничения доступа. Он выполняет все необходимые контрольные функции, его программная часть контролирует, в частности, и запуск задач (рис. 4). «Аккорд» предназначен для работы на компьютерах с процессором x86. Напомним, что архитектура таких компьютеров очень близка к классической архитектуре фоннеймановского типа.

Эффективность СЗИ НСД «Аккорд» связана с тем, что он блокирует уязвимости, связанные с нарушением целостности, и создает доверенную среду для работы программных

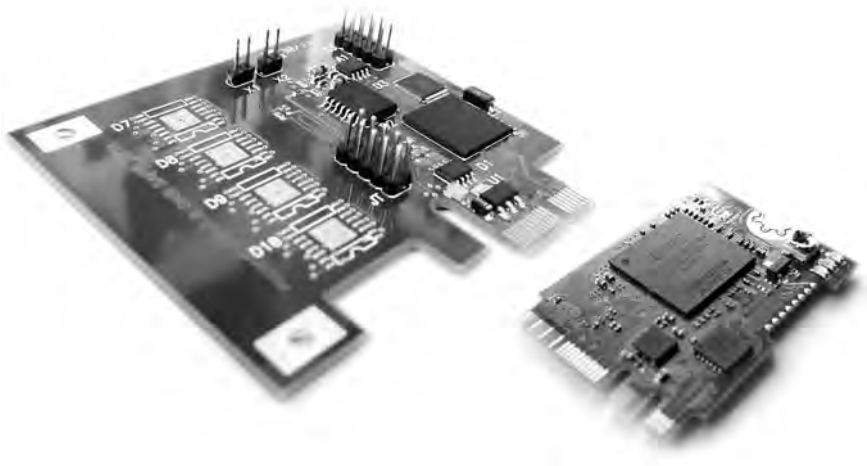

Рис. 4. ***СЗИ НСД «Аккорд-GX» и «Аккорд-GXm.2»***

средств, обеспечивающих защиту компьютера на шагах 1–6.

Несмотря на большую распространенность, цена СЗИ НСД «Аккорд» довольно высока, а его настройка – дело для профессионалов. Конечно, он хорош для корпоративного применения, но, видимо, слишком сложен для частного. Сложность его связана именно с фоннеймановской архитектурой защищаемого компьютера: нужно добавить неизменяемую память, разделить потоки команд и данных, исполнить контрольные процедуры в доверенной среде до запуска ОС и др.

Однако в компьютерах, использующих гарвардскую архитектуру, потоки команд и данных уже разделены. Возникает вопрос: нельзя ли использовать это обстоятельство для упрощения и удешевления защитных механизмов? Ведь нужно только сделать память неизменяемой (тогда не будет необходимости использовать сложные механизмы контроля целостности программ и данных до старта ОС), а контрольные процедуры в этом случае можно исполнять под управлением проверенной и неизменяемой ОС.

Эти функции легко реализовать, если обеспечить движение команд и данных только в одном направлении – из памяти в процессор (рис. 5). Очевидно, что такая архитектура обеспечит неизменность ОС, программ и данных.

Если мы вернемся к схеме атаки, описанной выше, то увидим, что шаг 3 не может быть выполнен, поэтому и сама атака (шаги 5 и 6) тоже не исполнится. Такой компьютер приобретет значительный «противовирусный иммунитет», так как вредоносное ПО не будет фиксироваться на компьютере.

Недостатком при этом будет потребность в доработке практически всего существующего ПО, так как его разработчики не ограничивают себя в использовании операций записи в память, которая тем самым становится необходимым условием для работы практически всех программ. Однако этого можно избежать, если предложенную архитектуру дополнить блоками сеансовой памяти, в которой и будут исполняться программы (рис. 6).

Таким образом, архитектура компьютера будет отличаться на разных этапах – сначала она такая, как на рис. 5, а потом такая, как на рис. 6. Фактически, архитектура изменяется от этапа начальной загрузки к этапу функционирования. Совместив эти рисунки, получаем изменяемую архитектуру гарвардского типа (рис. 7).

Предложенная нами архитектура, получившая название «новой гарвардской», отличается тем, что в ней используется память, для которой установлен режим «только чтение». При загрузке команды данные размещаются в сеансовой памяти, в которой и исполняются. Начальная загрузка и копирование кодов в сеансовую память могут выполняться как последовательно, так и параллельно, поскольку суть разделения этапов от этого не меняется (рис. 8).

Конечно, приведенная здесь схема описана условно, и в реальных

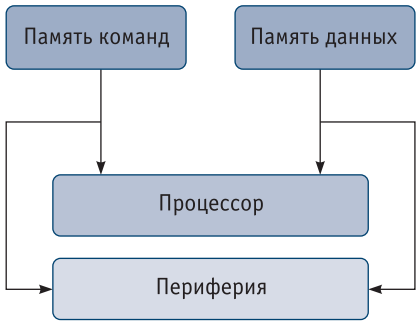

Рис. 5. ***Гарвардская архитектура с памятью RO***

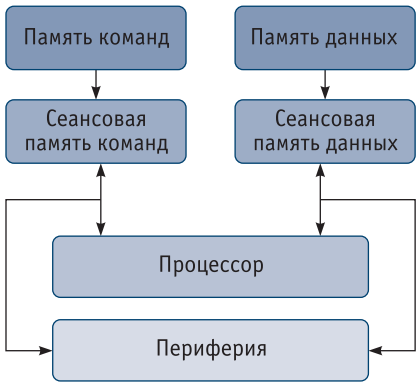

Рис. 6. ***Гарвардская архитектура с сеансовой памятью***

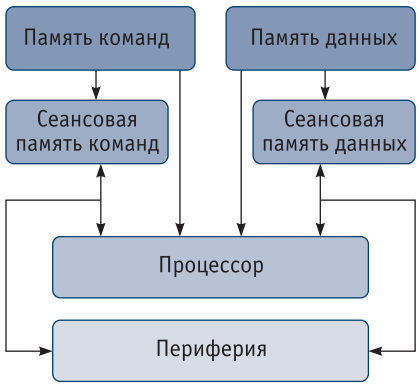

Рис. 7. ***Новая гарвардская архитектура***

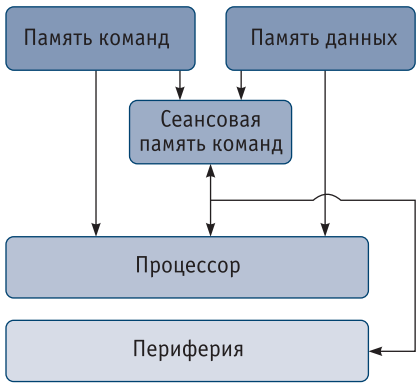

Рис. 8. ***Новая гарвардская архитектура с общей сеансовой памятью***

компьютерах всё немного сложнее. Однако можно уверенно сказать, что владельцы таких компьютеров (рис. 9) чувствуют себя намного защищеннее от атак хакеров.

Рис. 9. ***Компьютер МКТ-card, созданный на базе новой гарвардской архитектуры***

Новая архитектура характеризуется динамической изменяемостью, что обеспечивает защищенность, эффективность и неизменность операционной системы, «противовирусный иммунитет», а также не препятствует возможному применению адаптированных стандартных ОС и всего написанного для них ПО.

В архитектуре этого компьютера мы нарушили несколько принципов фон Неймана, и в первую очередь – П4 и П5. Действительно, П4 не выполняется, так как память команд и память данных не доступны на запись, да и нумерацию ячеек этой, а также сеансовой памяти, нельзя считать «последовательной». Кроме того, естественно, условный переход возможен в пределах сеансовой памяти и невозможен в защищенной памяти – в этом заключается нарушение принципа П5.

И что же мы получили взамен?

Основные преимущества: высокий уровень «противовирусного иммунитета», возможность создания и поддержки доверенной среды, возможность использования всего ранее наработанного ПО.

Важно, что на основе описанной архитектуры можно создавать компьютеры для всех видов информационного взаимодействия, при которых доверенность и защищенность последнего критически важна – от ДБО [13] и защищенных облаков [14, 15] до Интернета вещей. Сейчас на основе описанной архитектуры разработаны и серийно выпускаются 7 типов компьютеров: МКТ, МКТ+, МКTrusT, MKcard, MKcard-long, AQ-MK, TrusTPAD. Их особенности описаны в [16–22], а сами компьютеры – в [23]. Разработка новых видов компьютеров продолжается.

На предложенную архитектуру можно посмотреть и с метафизической точки зрения. Нетрудно заметить, что динамическая изменяемость компьютера, выполненного по новой гарвардской архитектуре, реализуется «во времени». Действительно, сразу после включения компьютер не соответствует требованиям машины Тьюринга, он работает скорее как конечный автомат, выполняя контрольные процедуры, после завершения которых он изменяется и становится универсальным.

Но если есть изменяемость «во времени», почему бы не быть и изменяемости «в пространстве»? Такие компьютеры тоже уже запатентованы и разработаны. В следующих публикациях я расскажу о них. ■

ЛИТЕРАТУРА

1. *Википедия – свободная энциклопедия [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Машина_Тьюринга/.*
2. *Википедия – свободная энциклопедия [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Архитектура_фон_Неймана/.*
3. *Википедия – свободная энциклопедия [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Гарвардская_архитектура/.*
4. *Планета информатики. Принципы фон Неймана (Архитектура фон Неймана) [Электронный ресурс]. – Режим доступа: http://www.inf1.info/machineneumann/.*
5. *Малиновский Б. Н. История вычислительной техники в лицах. – К.: «КИТ», ПТОО «А.С.К.». – 1995. – 384 с., ил.*
6. *Glushkov V. M., Ignatyev M. B., Myasnikov V. A., Torgashev V. A. Recursive Machines and Computing Technology. IFIP Congress 1974: 65–70.*
7. *Сайт компании ОКБ САПР [Электронный ресурс]. – Режим доступа: http://www.okbsapr.ru/dosug.html.*
8. *Википедия – свободная энциклопедия [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Burroughs_Corporation/.*
9. *Журнал «Электроника: НТБ» [Электронный ресурс]. – Режим доступа: http://www.electronics.ru/issue/2000/4/1/.*
10. *Виртуальный компьютерный музей [Электронный ресурс]. – Режим доступа: http://www.computer-museum.ru/histussr/11-1.htm.*
11. *Домашнее радио: страничка радиолюбителя [Электронный ресурс]. – Режим доступа: http://housea.ru/index.php/computer/50255/.*
12. *Конявский В. А. Управление защитой информации на базе СЗИ НСД «Аккорд». – М.: Радио и связь. – 1999. – 325 с., ил.*
13. *Конявский В. А. Защищенный микрокомпьютер МК-TRUST – новое решение для ДБО // Национальный банковский журнал. – 2014. – № 3.*
14. *Конявский В. А., Акаткин Ю. М. Мы не доверяем облаку или облако нам? // Information Security/Информационная безопасность. – 2014. – № 1.*
15. *Акаткин Ю. М., Конявский В. А. Безопасный доступ к корпоративным облачным приложениям // Information Security/Информационная безопасность. – 2014. – № 1.*
16. *Конявский В. А., Степанов В. Б. Компьютер типа «тонкий клиент» с аппаратной защитой данных // Патент на полезную модель № 118773. 2012. Бюл. № 21.*
17. *Конявский В. А. Компьютер с аппаратной защитой данных от несанкционированного изменения // Патент на полезную модель № 137626. 2014. Бюл. № 5.*
18. *Конявский В. А. Мобильный компьютер с аппаратной защитой доверенной операционной системы // Патент на полезную модель № 138562. 2014. Бюл. № 8.*
19. *Конявский В. А. Мобильный компьютер с аппаратной защитой доверенной операционной системы от несанкционированных изменений // Патент на полезную модель № 139532. 2014. Бюл. № 11.*
20. *Конявский В. А. Мобильный компьютер с аппаратной защитой доверенной операционной системы // Патент на полезную модель № 147527. 2014. Бюл. № 31.*
21. *Конявский В. А., Акаткин Ю. М. Мобильный компьютер с аппаратной защитой доверенной операционной системы от несанкционированных изменений // Патент на полезную модель № 151264. 2015. Бюл. № 9.*
22. *Конявский В. А. Рабочая станция с аппаратной защитой данных для компьютерных сетей с клиент-серверной или терминальной архитектурой // Патент на полезную модель № 153044. 2015. Бюл. № 18.*
23. *Сайт компании Trusted Cloud Computers [Электронный ресурс]. – Режим доступа: http://www.trustedcloudcomputers.ru/.*