ZIBELINE INTERNATIONAL PUBLISHING

Journal of Mechanical Engineering Research and Developments (JMERD)

DOI : http://doi.org/10.26480/jmerd.03.2019.19.23

ISSN: 1024-1752

CODEN : JERDFO

CrossMark

COMPUTER WITH CHANGEABLE ARCHITECTURE

Valery A. Konyavsky*, Gennady V. Ross

Laboratory of Semantic Analysis and Integration, Plekhanov Russian University of Economics, 36 Stremyanniy Lane, Moscow, Russian Federation
****Corresponding Author Email: konyavskiy@gospochta.ru***

ARTICLE DETAILS

Article History:

Received 01 February 2019
Accepted 14 March 2019
Available online 28 March 2019

ABSTRACT

The relevance of this article is that, despite the rapid development of computer technologies, it has not been yet invented the architecture that did not have vulnerabilities for hacker attacks. The purpose of the paper is to find a way of creation a computer with changeable architecture. The authors used the comparative method for finding features of the classical architectures: von Neumann and Harvard, which are great for everyday tasks, but too vulnerable to virus attacks. The result of the research was the theoretical development of a new architecture based on the classical ones. The novelty of the research lies in the fact that the authors developed a new technology of computer architecture, which is protected from viruses. This article will be useful to anyone who is just interested in or specializes in improving computer architecture to prevent hacker attacks.

KEYWORDS

Virus, hacker, Von Neumann architecture, Harvard architecture, “new Harvard” architecture

1. INTRODUCTION

Nowadays, computers are widely used in education and healthcare, scientific organizations and banks, practically at any work and at home. These computational tools are also part of information-measuring, medical-diagnostic, control, electronic, telecommunication and other technical systems.

A modern computer is a universal multifunctional electronic automatic device for working with information. Computers in modern society have undertaken a large part of the work related to information. By historical standards, computer-based information processing technologies are still very young and at the very beginning of their development, but even today they are transforming or replacing traditional processes of information processing.

Alongside with huge advantages, computers bring challenges. And almost all of these challenges are connected with the possibility to interfere with the operation of your computer from the outside. Fighting hackers has become a "pain in the neck" for many people, and the word "virus" is often associated with a computer rather than medicine. The failure to fight hackers makes us look for the reasons not in the software as it is done today, but in something deeper.

Computers are designed to run programs – various algorithms written in programming languages. The word “various” means almost “all” here. The computer runs any program. Predefined functions can be performed not only by computers – for example, finite-state machines can perform them too. The difference is that the computer executes "any" program. The computer is incomparably more flexible and versatile compared to the finite-state machine.

2. THE TURING MACHINE AS MODEL OF MODERN COMPUTER

Any computer is (more or less exact) implementation of the idea of a "Turing machine" [1]. The concepts "Turing machine" and "algorithm", computability is inseparably linked, defined by one another. The mere existence of an abstract “executor” such as the Turing machine makes us feel confident about human omnipotence. Indeed, any problem (or rather, recursive one, i.e. almost any) can be solved if there are enough resources (memory and time).

Perhaps it was amazing simplicity of wordings that provoked the development of general-purpose computers (universal computers, computer equipment, personal computers, PC), which partially (with finite memory) simulate the Turing machine, giving us pseudo-unlimited possibilities and pushing us into the way of extensive development. If the memory is insufficient – no problem – we’ll add some. If the time is insufficient – we’ll increase the clock frequency, the number of cores, or virtualize the resources.

For many years, this position has dragged the development of information technology behind it like a "locomotive". For example, the capacity of common local disks has grown from tens of kilobytes to hundreds of gigabytes over the last two decades, and is often measured by terabytes, but the memory still lacks. Beginning from kilohertz, clock frequencies have reached gigahertz, but the performance still lacks. However, the IT industry has become one of the industries determining the current level of economic development. Huge amounts of investments are the payment for technological progress and universality of solutions.

Using the Turing machine, one can model any other computer (any "executor"), so they say about the completeness of the Turing machine [1]. In its turn, taking into account the above-mentioned restrictions (finiteness of the memory), the Turing machine can be modeled on the general-purpose computer. Such general-purpose computers are called Turing complete.

Thus, Turing complete general-purpose computers should at least perform elementary operations peculiar to the Turing machine, namely, to move the control unit (read-write head) to the left or to the right along the tape, to read and to write characters of a finite alphabet in cells. Linearity of the memory and consistency of operations are the basic characteristics of the Turing machine.

General-purpose computers are potentially capable of self-learning, reaching a level at which a discoursing person will not be able to guess whom he is speaking to – a man or a computer (the Turing test). Of course, these are splendid prospects, the very existence of which makes the sphere of computer technology very attractive. But is the ability of self-learning always a plus? For example, serious concerns can emerge if controllers of automated process control system at nuclear power plants, in railway transport or continuous production will uncontrolledly learn. It

is unlikely that these controllers should be intelligent, and not accurately perform their functions. Apparently, therefore finite-state machines are usually used to solve such problems. Finite-state machines, context-free grammars, primitive recursive functions are examples of Turing incomplete formalisms.

Thus, there are many problems that can be solved not by a general-purpose "executor". Moreover, there are many problems that should be solved not by a general-purpose, but by a specialized "executor". Thus, the inalienable opportunity "to read and to write", which makes the copy operation in the computer immanent, totally contradicts, at least, the tasks of information security. The property, which is required for "general-purposefulness" gets unacceptable in specific circumstances.

Trying to protect itself from malicious hacker programs, the humanity has (for more than 60 years) been developing programs that are traditionally attributed to the sphere of information security – means of identification, authentication, authorization, integrity control, anti-virus software, cryptographic tools and so on. Use of such means brings some positive effect, but very poor one. Acting within a universal, but a formal model, we will inevitably face its incompleteness – in full accordance with the Gödel's incompleteness theorem.

It becomes obvious that searching for vulnerabilities only in software is not enough. Indeed, if a general-purpose computer runs any program, it will obviously run a malware. It does not depend on its software; it is defined by its architecture. General-purposefulness of a computer is ensured by its architecture, the very "construction" of the Turing machine, both, in the abstract form and in practice. The ability to run malicious programs is a basic, systemic, architectural vulnerability of all computers built like the Turing machine. Vulnerability is the reverse side of general-purposefulness. The Turing machine is vulnerable in its architecture. All kinds of computers used are vulnerable because they have been designed to be as general-purposeful, as possible. This vulnerability is a price for general-purposefulness of our computers. We exploit computers, and hackers exploit this vulnerability. Since the architecture cannot be changed by software, no software will help us to ensure reliable defense against hackers. The "tug of war" has been going on for many years, giving jobs to hundreds of thousands of information security experts, but not saving us from losses.

3. THE VULNERABILITY OF THE CLASSICAL COMPUTER ARCHITECTURES

If vulnerability lies in the architecture, we need to improve the architecture. There are two classic architectures – the Von Neumann architecture and the Harvard architecture [2,3]. Almost all personal computers can serve as examples of the former one, and almost all tablet computers and phones can serve as examples of the latter one.

The computer described below is a computer whose architecture differs from both the von Neumann architecture and the Harvard architecture. However, the computer described below, or rather, a new or changed or modified architecture, is not the first deviation from classics. Let's consider some of them.

We know the von Neumann principles of computing process organization, namely [4]:

P1. Use of the binary system in computers.

P2. Operation control of the computer with the help of a sequence of commands (programs).

P3. Use of the computer memory to store both data and programs. The data and the programs are stored in a single memory in the same form, and the same operation can be performed in relation to commands as to the data.

P4. Serial numbering (addressing) of all computer storage locations, and the possibility to randomly access any storage location using its address. In other words, the entire memory of the computer is located in the same address space.

P5. Provision of a conditional branch to any part of the code while running a program, despite the fact that the commands are executed one by one.

Note that P4 and P5 are not new principles, but in fact it is a partial description of the Turing machine. The analysis of the computer development history shows that some of these principles have been repeatedly violated, which often led to unexpectedly positive results. Some historical examples are summarized in Table 1.

Table 1: Some computers deviating from the Von Neumann principles

Principle	Example of violation	Peculiarity
P1	Computer "Setun" [5]	Ternary number system
P2	V.M. Glushkov's m*acroconveyor computer* [6] Computer "B5000" produced by Burroughs Corporation [7]	Independent command flows
P3	Computer "MIR"-1 Computer "B5000" produced by Burroughs Corporation All computers built based on the Harvard architecture	Computer language High-level commands and data are stored separately

These and many other examples show that computing process organization principles of Von Neumann (at least P1-P3) are not dogmas, and they have been repeatedly questioned by leading global developers, and an open-minded approach has yielded positive results, although the experiments have been rather expensive. The contrary thereof is also true – focusing on general-purposefulness (computing process organization in accordance with the Von Neumann principles) at the previous stage of technological development gave a powerful impetus to information technology, but at the same time stopped productive research in the field of computer architectures for a long time. It should be also noted that the principles that go back to the ideas of the Turing machine (P4 and P5) have never been violated, or we just could not find examples thereof.

Technology has made a huge breakthrough over the recent years, but it has not almost affected the computer architecture. The most outstanding change here is the outrunning growth of solutions based on the Harvard architecture. While a few years ago, the ratio of x86 and ARM processors sold was 80:20, today it is 50:50, and this trend is increasing . There are now conditions to reflect on improving the architecture.

While developing a computer, one must understand what functions are to be performed by hardware, and what functions are to be performed by software. The correct choice of this ratio has allowed PCs built on the von Neumann architecture to occupy a leading position during many years. However, the maximum effectiveness of this technical solution has been already reached; extensive development (increase of productivity, memory, reduction of dimensions, etc.) does not meet social needs any longer. A computer deprived of the vulnerabilities that hamper the development of information exchange may become a driver for the development. When developing a new computer, one must include in the hardware things that reduce its cost, rarely change, expand its capabilities and are always used. Moreover, it does not seem impossible now that the computer structure can dynamically change in the process of operation. Or, for example, at first the structure may be as that of a finite-state machine, and then at the next stage it can become a Turing "general-purpose" machine.

A distinctive feature of the von Neumann (Figure 1) architecture is that commands and data are not divided, they are transferred through a single common channel.

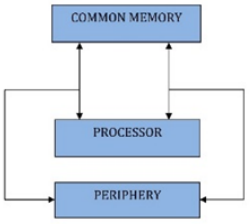

Figure 1: Von Neumann architecture

The Harvard architecture (Figure 2) provides for availability of separate channels for commands and data.

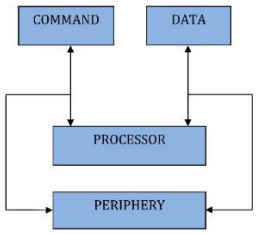

Figure 2: Harvard architecture

Such a scheme requires a more complex organization of the processor, but ensures greater performance, since flows of commands and data are not consecutive, but parallel, independent. However, both in the case of a Von Neumann computer and in the case of a computer with Harvard architecture the organization of command and data flows is such that both computers are vulnerable in their architecture. The flexibility and the general-purposefulness in both cases are ensured by the possibility to change the sequence of commands and data (double-headed arrows from the processor to the memory), irrespective of whether they are stored in a single or separate memory. In its turn, the possibility to change the sequence of commands and data creates a possibility of unauthorized interference of malicious software – this is the main architectural vulnerability.

4. THE USAGE OF VULNERABILITIES IN CLASSIC ARCHITECTURE DURING THE HACKER ATTACKS

Such vulnerability is a basis for almost all modern hacker attacks which mostly boil down to the "takeover" attack. The attack usually looks like as follows:

s1) malicious software is introduced and installed in the main memory;

s2) a malicious interrupt handler is introduced and installed in the main memory;

s3) the malicious software and the interrupt handler are recorded in the long-term memory;

s4) an interrupt is caused using any available mechanism such as a DDOS attack;

s5) the preinstalled interrupt handler gets activated and passes control to the malicious software;

s6) the malicious software performs its function, for example, destroys information.

S1-s3 – are here steps to prepare for an attack, s4 is initiating the attack, s5 and s6 are the very use of architectural vulnerability.

To neutralize s1 and s2 steps antivirus programs are usually used. Sometimes it is useful, but only sometimes, because it is impossible to detect all malware with the help of anti-virus software. Moreover, experts know some strictures of the malware that cannot be detected at all. One could even say that computer viruses and generally malware can be detected only because of their defects. In general, you can always develop a malware that cannot be detected by signature search anti-virus software, heuristic analyzers and behavior blockers.

S3 step cannot be prevented in standard architectures. One can only block the aftereffects. One can block the aftereffects of s3 while starting the system using integrity control mechanisms, which are in fact auditors determining whether there is a change in data composition or not. Sometimes this test is performed using the same sets of anti-virus software, but it is a poor solution, because the test should be performed prior to start of the OS, but programs, including antivirus ones run under the OS control.

Event generation at s4 is partially blocked with the help of special traffic analysis tools installed both in the network and on client computers. It is important that there are no means yet to certainly block this vulnerability.

Negative effects of s5 and s6 steps are blocked using task (process, flow) initiation control tools. These are very effective mechanisms, but tools implementing them are quite expensive, and to set them one needs to be an expert in computer technology and information security.

5. THE ARCHITECTURE "NEW HARVARD"

Since some of these security functions should be performed before starting the operating system, they cannot be implemented by software, but only with the help of a complex device. Data security tool protecting from unauthorized access (DST PUA) "Accord" is an example of such a device [8]. This is a trusted startup hardware module with a software complex of access control. It performs all the necessary control functions, and its software part controls, in particular, tasks initiation. Accord is designed to work on x86-processor computers. Note that the architecture of such computers is very similar to the classical Von Neumann architecture. The effectiveness of DST PUA Accord (Figure 3) is connected with the fact that it blocks vulnerabilities associated with integrity violation and creates a trusted environment to run software that protects your computer at s1-s6 steps.

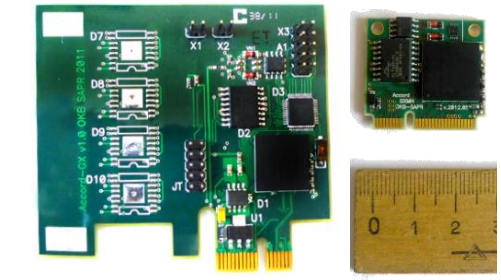

Figure 3: DST PUA Accord

Despite its popularity, the price of DST PUA Accord is quite high, and its setting can be done only by professionals. Of course, it is the best solution for companies, but apparently it is too complex for private use. Its complexity is due to the von Neumann architecture of the computer protected – you need to add a read-only memory, divide the flows of commands and data, carry out control procedures in a trusted environment before starting the operating system, and so on. However, computers built on the Harvard architecture already have separated flows of commands and data. If so, is it possible to use this fact to simplify and reduce the cost of security mechanisms? It is only necessary to make the memory read-only (then there would be no need to use complex mechanisms to monitor programs and data integrity before the start of the operating system), and in this case control procedures can run under a trusted and read-only operating system. These functions can be easily performed if one ensures movement of commands and data only in one direction – from the memory to the processor (Figure 4). Obviously, this architecture will ensure invariability of the OS, programs and data.

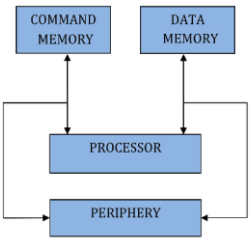

Figure 4: Harvard architecture with RO memory

If we return to the above attack scheme, we see that s3 step cannot be executed, so the attack itself (s5 and s6 steps) cannot be executed too. Such a computer will acquire a significant "viral immunity" as malware will not be recorded on the computer. A disadvantage of this is that you will have to improve almost all the software, because developers of existing software widely use the memory write operations. To run, almost all programs need the possibility to write. To be able to use all previously developed software without any modifications, the proposed architecture should be supplemented with session memory blocks, in which programs will run.

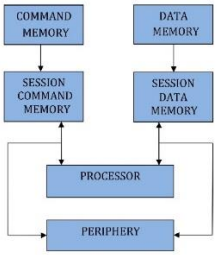

Figure 5: Harvard architecture with session memory

Thus, the computer architecture will differ at various stages – at first, it will be as shown in figure 5, and then it will be as shown in figure 6. In fact, the architecture changes from the start-up to the operation stage. Combining two pictures, we will get a variable architecture of the Harvard type.

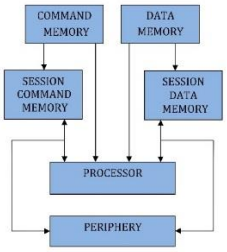

Figure 6: New Harvard architecture

The proposed architecture has been named "new Harvard" architecture. It differs from others by its read-only memory. At the start-up stage commands and data are located in the session memory, where they are executed. The initial start-up and copying of codes to the session memory may be performed both, consecutively or in parallel – the essence of separation does not change (Figure 7, 8).

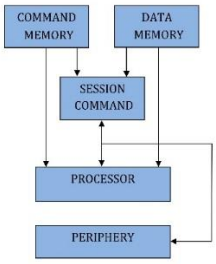

Figure 7: New Harvard architecture with a common session memory

Figure 8: A modular computer MKT card, New Harvard architecture

Of course, this scheme is just theoretical, and it is much more complex in real computers. However, we can say for sure that owners of such computers feel much more protected from hacker attacks. The new architecture is characterized by dynamic variability, which maintains effectiveness and at the same time ensures sufficient security, invariability of the operating system, "viral immunity", use of adapted standard operating systems and all the software developed for them.

It is important that the above architecture can be used to create computers for all kinds of information exchange where the trusted environment and security are important – from remote banking (RB) and secure "clouds" to the "Internet of Things" [9-11]. 7 types of computers are now serially produced on the basis of the above architecture – MKT, MKT+ MKTrusT, MKcard, MKcard-long, AQ-MK, TrusTPAD [12-19]. Their features are described in, and the computers themselves are described in. New types of computers are being developed now [20-24].

6. CONCLUSION

Developing the architecture of this computer, we have violated several von Neumann principles, and first of all – P4 and P5. Indeed, P4 is violated, since the command memory and the data memory are not available for writing, and numbering of the cells of this memory and the session memory cannot be considered "consecutive". And, of course, a conditional branch is possible within the session memory and impossible in the protected memory, so P5 is also violated. And what do we get in return? There are two main advantages – a high level of "viral immunity" and a possibility to create and maintain a trusted environment and to use all previously developed and tested software.

REFERENCES

[1] Ebbinhaus, G.D., Jacobs, K., Man, F.K., Hermes, G. 1972. Turing machines and recursive functions. Mir, Moscow.

[2] Ashraf, M.A., Hanafiah, M.M. 2018. Sustaining life on earth system through clean air, pure water, and fertile soil. Environmental Scince & Pollution Research.

[3] Aspray, W. 1990. John von Neumann and the origins of modern computing. MIT Press, Cambridge.

[4] Dastani, M., Panahi, S., Sattari, M. 2019. Webometrics Analysis Of Iranian Universities About M edical Sciences' Websites Between September 2016 And March 2017. Acta Informatica Malaysia, 3(1), 07-12.

[5] Cohen, B. 2000. Howard Aiken, Portrait of a computer pioneer. The MIT Press, Cambridge.

[6] Abugalia, A. 2019. Effect Of Corona On The Wave Propagation Along Overhead Transmission Lines. Acta Electronica Malaysia, 3(1), 06-09.

[7] Von Neumann Principles (Von Neumann Architecture). http://www.inf1.info/machineneumann

[8] Tian, X.C., Li, Q.H., He, C.S., Cai, Y.G., Zhang, Y., Yang, Z.G. 2018. Design and experiment of_reciprocating double Track Straight Line Conveyor. Acta Mechanica Malaysia, 2(2), 01-04.

[9] Malinovsky, A. 1995. History of computer engineering in persons. KIT, PTOO A.S.K., Kyiv.

[10] Adrian, C., Abdullah, R., Atan, R., Jusoh, Y.Y. 2018. Theoretical Retical Aspect in Formulatting Assesment Model Of Big Data Analytics Environment. Acta Mechanica Malaysia, 1(1), 16-17.

[11] Glushkov, V.M., Ignatyev, M.B., Myasnikov, V.A., Torgashev, V.A. 1974. Recursive Machines and Computing Technology. IFIP Congress, 65-70.

[12] Martin, I.L. 2012. Too far ahead of its time: Barclays, Burroughs and real-time banking. IEEE Annals of the History of Computing, 34(2), 5-19.

[13] Konyavsky, V.A. 1999. Data security management using DST PUA Accord. Radio i Svyaz, Moscow.

[14] Konyavsky, V.A. 2014. Secure microcomputer MK-TRUST – a new solution for remote banking. National Banking Journal, 3, 105.

[15] Konyavsky, V.A., Akatkin, Y.M. 2014. Do not we trust the cloud or it does not trust us? Information Security, 1, 28-29.

[16] Akatkin, Y.M., Konyavsky, V.A. 2014. Secure access to corporate cloud applications. Information Security, 1, 23.

[17] Konyavsky, V.A., Stepanov, V.B. 2012. Thin client computer with hardware data security tools: utility model patent, 118773(21).

[18] Konyavsky, V.A. 2014. Computer with hardware data protection from unauthorized change: utility model patent, 137626(5).

[19] Konyavsky, V.A. 2014. Mobile computer with hardware protection of

the trusted operating system: utility model patent, 138562(8).

[20] Konyavsky, V.A. 2014. Mobile computer with hardware protection of the trusted operating system from unauthorized change: utility model patent, 139532(11).

[21] Konyavsky, V.A. 2014. Mobile computer with hardware protection of the trusted operating system: utility model patent, 147527(31).

[22] Akatkin, Y.M., Konyavsky, V.A. 2015. Mobile computer with hardware protection of the trusted operating system from unauthorized change: utility model patent, 151264(9).

[23] Konyavsky, V.A. 2015. Workstation with hardware data security tools for computer networks with client server or terminal architecture: utility model patent, 153044(18).

[24] Trusted Cloud Computers. http://www.trustedcloudcomputers.ru