Ingénierie des Systèmes d'Information
Vol. 28, No. 4, August, 2023, pp. 951-958
Journal homepage: http://iieta.org/journals/isi

Accelerating Code Assembly: Exploiting Heterogeneous Computing Architectures for Optimization

Maksym Karyonov

Department of Information Communications and Software Engineering, O.S. Popov Odesa National Academy of Telecommunications, Odesa 65023, Ukraine

Corresponding Author Email: maksymkaryonov2@ukr.net

https://doi.org/10.18280/isi.280415

Received: 21 March 2023
Revised: 20 July 2023
Accepted: 17 August 2023
Available online: 31 August 2023

Keywords:
computer technology, software development, informatics, digital technologies, typical block diagram, programming

ABSTRACT

Amid rapid technological advancements, the efficient optimization of software code assembly and compilation is paramount to the swift and reliable functioning of high-performance computing systems. This study investigates the potential for boosting code assembly speed by exploiting various computing architectures. The adopted methodology encompasses system analysis, examination of diverse computer system architectures, and the application of optimization and resource management techniques to enhance the assembly and compilation of program codes effectively. The paper delves into the evolution of computer architecture and underscores the importance of machine code, elucidating their impacts on IT development. Key areas of study include mobile object tracking, cache memory-based architectures, and GPU inference mechanisms for neural networks. The criticality of expertise, security, and contextual understanding when adopting these technologies is also emphasized. The findings from this study could catalyze the inception of novel code assembly technologies, thereby optimizing computing efficiency and expediting software development. Consequently, these advancements could diminish the time required for program creation and launch, thereby elevating industry productivity. The practical significance of this research stems from its potential application in accelerating code assembly.

1. INTRODUCTION

The impetus for this research arises from an inadequately explored realm - accelerating code assembly utilizing diverse computer architectures. Despite continuous efforts in the evolution of modern computer system architecture and structural advancements, a knowledge lacuna persists in pinpointing the optimal strategies for program code optimization. The objective of this research is to critically appraise and cultivate contemporary methods to expedite code assembly, keeping in mind the assortment of existing architectural alternatives.

The multitude of presently available architecture options necessitates exploration into avenues for code assembly acceleration. Such an investigation holds potential to significantly augment the efficacy of contemporary computers, manifesting in enhanced data processing speed, expedited information transmission and reception, and reduced command response time. An essential facet of this study involves identifying potential limitations and shortcomings associated with the utilization of specific computer architectures in the process of program code assembly.

Liu et al. [1] underscore the necessity for a symbiotic relationship between hardware and software developers to accelerate modern computing machine code assembly through architecture alteration. They observe that while hardware acceleration systems are predominantly driven by software developers, the latter often face challenges in system construction due to substantial disparities in equipment concept comprehension and design tool application.

Harris and Harris [2], in their exploration of the complexities of digital design and computer architecture construction, point out the vast array of existing computer architectures. They posit that the selection of a specific architecture determines the nature of data processing and the subsequent transformation methodologies. They further suggest that the employment of varied architectures presents additional opportunities to accelerate code assembly processes.

Hennessy and Patterson [3] highlight the choice of computer architecture types in their collaborative research. They present two pivotal factors in modern computer architecture: memory hierarchy and parallelism, and suggest that computer code may be viewed as the simplest programming language or as a primitive computer program, be it assembled or compiled.

Maroun et al. [4] investigated the principles of time-predictable compilation using one-way code in their research. They propose that the real-time architecture of the Patmos instruction set offers a considerable degree of predictability. Further, they note that one-way machine code is generated by a compiler that produces high-performance programs.

The topic is further expanded by Ying et al. [5] who investigated the prospects of extending memory page management to prevent code pointer leakage. They note that while code pointers are common in low-level languages, their exposure without adequate protection significantly escalates the risk to computer system security.

Singh and Singh [6], in their review of malware detection

principles in executable files, highlight an increase in code acceleration options over the past decade. They propose that the employment of diverse computer architectures to expedite program code is a promising approach.

The study is structured into five sections, commencing with an "Introduction" that delineates the research problem, provides the context for the study, identifies a knowledge gap, and establishes the primary study objective. The subsequent section, "Materials and Methods", details the methodology employed in the research, exploring various technological aspects including system architectures, code assembly, IoT edge computing, wireless devices, and multi-core systems. The "Results" section elucidates the findings of their investigation into different types of computing machine architectures, providing an in-depth explanation of machine code, computer architecture, and their foundational role in all computing technology. The "Discussion" section offers an extensive analysis and review of multiple research studies on the significance of software optimization, computing system architecture, and their role in code assembly acceleration and performance improvement. The "Conclusion" section encapsulates the study's findings, focusing on the impact of varied computer architectures on system performance, particularly in terms of code compilation speed. The primary objective of this research is to examine the actual potentials and prospects for accelerating code assembly through the use of varied computer architectures.

2. MATERIALS AND METHODS

In this scientific study, it was used the method of system analysis, combined with the analysis of various types of architectures of computing machines and the code assembly options. Identification of the most common types of architectures has become the main goal of the study, which can be effectively used to speed up the process of code assembly, as well as to establish the characteristic features of architectures of various types that can be used for this purpose.

To achieve this goal, it was studied the various materials, including descriptions of existing types of architectures of computing machines and the code assembly options, as well as materials containing information on methods of system analysis. As a result, it has been identified the most efficient types of architectures that can be used to accelerate code assembly.

In the context of the rapidly evolving software development industry, the improvement of performance and optimization of processes are the key factors for achieving success. The suggested recommendations, based on the characteristics of architectures, can significantly improve performance in the process of code assembly. These findings can be used for further research and development of new methods of process optimization, which in turn will increase the efficiency of software development and ensure its competitiveness. This method was applied to analyze the information system that included several components such as databases, servers, client applications, and so on. With the help of system analysis, it was possible to identify the key components of the system and determine the interactions between them. This has made it possible to better understand the processes occurring in the system, identify the possible problems and bottlenecks that may affect its operation, as well as develop recommendations for optimizing the system to increase its efficiency and reliability. In the course of the study, it was also analyzed various types of architectures of computing systems and their characteristics were determined. In addition, the options for code assembly were studied and their advantages and disadvantages were identified. For the study, it was used materials from open sources, such as scientific articles and publications in journals. The method of system analysis was applied to evaluate the characteristics of various architectures of computing systems, as well as to identify factors affecting the code assembly. This method does not require experiments or surveys. The analysis involved several steps, including gathering information on architectures and code assembly options, evaluating their characteristics, and identifying the factors influencing code assembly performance.

To determine the characteristics of architectures of the computing system that can be effectively used to accelerate the code assembly, the method of system analysis was used. This method was chosen due to its ability to analyze complex systems and reveal their properties and characteristics. In the course of the study, it was analyzed various types of architectures of computing systems, such as SIMD (Single Instruction Multiple Data), MIMD (Multiple Instruction Multiple Data), SISD (Single Instruction Single Data), MISD (Multiple Instruction Single Data), and others. It also discussed the options for code assembly, such as compilation at runtime, compilation at load time, and compilation at assembly time. The advantages and disadvantages of each of the options of code assembly are determined depending on the type of architecture of the computing system.

The reliability of the study is guaranteed on the quality and accuracy of the materials used, which were sourced from open publications and scientific articles. Alternative approaches could involve conducting experiments, surveys, or employing specialized methodologies to analyze specific aspects of computing architectures and code assembly. Validity factors can be addressed by cross-referencing information from multiple sources, ensuring consistency and coherence of results. Mitigation strategies for challenges faced could include systematic literature reviews, consulting domain experts, and maintaining transparency in the analysis methodology and decision-making process.

3. RESULTS

Machine code is the basis of the work of computers and programming. This is a low-level language understood directly by the processor, rather than a human. It is a set of commands defined by a particular computing machine and it is subject to interpretation by firmware or processor. Machine code can be considered as the simplest programming language and as an elementary level of perception of computer programs. Some programmers create computer programs directly in machine code if the code is too cumbersome and the process of manually managing the processor and its resources is labor-intensive. Computer architecture defines the fact how the machine processes large amounts of information using the principles of its transformation into data, following the sequence of interaction between software and technical means. Each specific type of computing machine has its own individual structure. (“A Comparison of Build Systems” is a study that compares different build systems, including their performance on different computer architectures).

Figure 1 shows a block diagram of the typical architecture

of computing machines of the first and second generations. It shows the main components of such machines and their relationship, including the processor, RAM, input/output devices and other components. Machine code and computer architecture are the basic elements on which all computing technology is based. Without them, it is impossible to create effective programs and devices that are used in everyday life. Therefore, knowledge of these concepts is essential for anyone working in the field of computers and information technology.

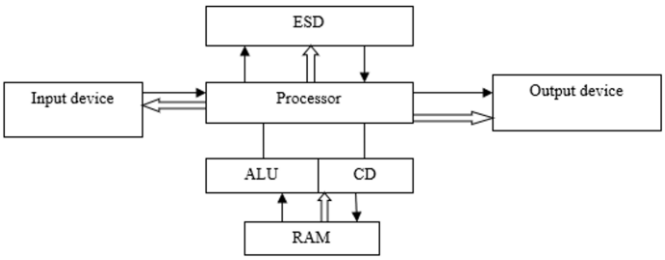

Figure 1. Architecture (typical) of computing machine [7]
Note: ESD – external storage device; RAM – random-access memory; ALU – arithmetic logic unit; CD – control device

Figure 1 shows the classical architecture of computing machines, developed according to the von Neumann principle. In this architecture, the processor and memory interact through a common bus, which transfers information flows and control signals from the processor to memory and vice versa. The control device performs the function of isolating the necessary memory cell, which contains information regarding the next program instruction in the queue; for this purpose, the control device provides for the placement of a special reader – a program counter. However, as the amount of data and the frequency of memory access grew, the typical von Neumann architecture began to limit the capabilities of computing machines. The sharp increase in overhead costs for information transfer has led to the need to develop new architectures.

In response to this problem, the designers of computing machines began to create architectures that were optimized to work with large amounts of data and allow more efficient use of computing resources. Backbone architecture is one such architecture, shown in Figure 2. In the backbone architecture, the processor and memory are connected by a network of backbones (rather than a common bus), which allows data to be transferred in parallel and reduces the time for information transfer. This architecture also provides an opportunity to parallel processing of data and reduces delay in the data transfer process, allowing making computations faster. Thus, the development of new computer architectures makes it possible to increase their performance and efficiency in working with large amounts of data.

According to the data presented in Figure 2, the backbone is a connecting link of all working sections of the computing machine in this architectural solution (also called “bus”). It includes three types of buses: data highway (used to receive and transmit incoming information), control backbone (used to control the reception and transmission of information and control this process) and address backbone (used to track the final directions of sending information arrays). The presence of controllers in the backbone architecture of computing machines is their key difference from analogs with a typical von Neumann architecture. The controller performs the functions of a specialized processor that controls the operation of external devices. It is equipped with an autonomous command system, and if necessary, it can receive a task to perform data exchange from the central processor. In this case, the central processor does not participate in the data exchange. The modern personal computer has a logical organization of its hardware components. In this case, all elements of the system interact with each other through the backbone (Figure 2), which combines three types of buses (control, data, and addresses). The command system is one of the key components of the architecture of the personal computer. The command system of modern computers includes the following: the commands for input and output of data (designed for data exchange with peripheral devices), commands for performing specific actions (logical and arithmetic operations), control commands (divided into commands for unconditional and conditional transitions, as well as access to subprograms) and commands of information redirection (provide copying of information in different parts of the system).

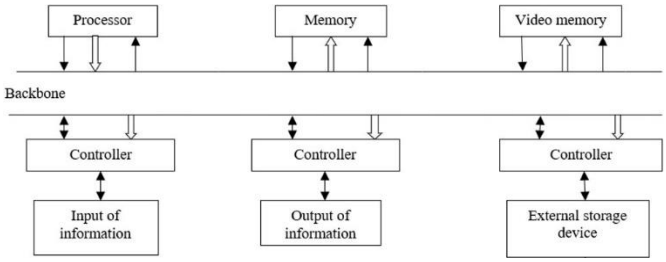

Figure 2. Backbone architecture of computing machine [7]

M. Flynn's classification was introduced in 1966, and it refers to computer architecture. This classification refers to the main types of processor architecture, namely to single-threaded, multi-threaded, single-core and multi-core processors. Since then, many changes and improvements have taken place in the field of processor architecture, and some of them may not fit into M. Flynn's original classification. M. Flynn's classification is the basis for describing various architectures of computing systems and is an important tool for analyzing and comparing different types of processors and computer systems. However, it should be noted that M. Flynn's classification still remains fundamental for many developers and specialists in the field of processor architecture. Despite the emergence of new technologies and architectures, the basic principles of M. Flynn's classification are still applied in modern processors. In addition, it should be noted that the development of new architectures of processors and computers is an active area of research and development, and new ideas and technologies appear regularly. Perhaps in the future, there will be a need to supplement or expand the classification of M. Flynn in order to consider the new types of processors and architectures.

According to M. Flynn's classification, presented in 1966 (“Understanding Flynn's Taxonomy and Its Relevance in Modern Computing”, “Flynn's taxonomy”), there are the following types of architectures of computing systems:

•MIMD (Multiple Instruction Multiple Data) is a type of architecture of computing systems in which several processors, working independently, execute a set of instructions on a set of data, and each of them can be processed separately. MIMD systems can be implemented both as clusters and as multi-core processors that can work on different parts of the program simultaneously. Each processor in such systems may have its own set of instructions and can process data from its own

memory. This allows efficient solving of problems that can be divided into independent subtasks. This approach is widely used in parallel computing, where different processes can work on different aspects of the same task. This allows getting high productivity and reduces the time required to solve complex tasks. However, the need for effective management and synchronization of such systems can significantly complicate their development and maintenance. The MIMD architecture is used in various fields such as scientific research, design, machine learning, graphics, multimedia, and others. It allows effective parallelizing of the execution of tasks and improving the system performance, which is especially important in the rapidly growing complexity and volume of processed data.

•MISD (Multiple Instruction, Single Data) is an architecture of a multiprocessor computing system in which several processors perform independent tasks using a common set of data. However, in this architecture, all processors execute different instructions, instead of executing the same instruction on different data sets. In the MISD architecture, each processor receives the same set of data from the shared memory and executes its own unique instruction on them. Thus, each processor works independently, but the computing results must match between all processors. MISD systems are widely used for cryptographic computations, such as performing multiple decryptions of the same encrypted message. In general, the MISD architecture is not very popular for now, because it has few advantages compared to other types of architectures. However, it remains interesting for some specific applications where high computational accuracy and reliability are required. X-ray computed tomography (CT) scanner is one of the examples of a system that uses the MISD architecture. Each slice of the body is processed by several processors, and each of them executes its instruction on the same data. Then, the computing results of all processors are combined to create a body slice image.

•SISD (Single Instruction Single Data) is a type of architecture of computing system in which one instruction is executed on one data flow. This means that in the SISD system, only one command is executed simultaneously, which processes one piece of data. In the SISD-system, the central processing unit executes instructions sequentially, applying them to each data element. All operations are performed sequentially, and no operation can start until the previous one is completed. This means that there is no parallelism in SISD-systems, which limits their performance. In general, the SISD-architecture is the least efficient of all architecture types, but it remains important in some specialized areas where high computational accuracy and error tolerance are required. The SISD-architecture is used in simple computing devices such as calculators, some electronic devices for control and navigation, and in older computers that do not have parallel processing capability. However, the SISD-architecture is currently used in computer systems where the results' accuracy is of paramount importance, for example in scientific computing or signal processing. In these systems, the processor executes one instruction on one data element, by providing a high computational accuracy, but a low processing speed.

•SIMD (Single Instruction Multiple Data) is a type of architecture of computer system where each processor executes the same instruction on different sets of data. In this case, each processor runs independently, but all of them receive the same instruction. SIMD systems have high performance for tasks that can be easily divided into many identical computations on large data sets. However, the SIMD-architecture may be ineffective for complex tasks that cannot be easily divided into many identical computations on large data sets. In addition, the need to use specialized instructions for performing SIMD-operations can lead to restrictions in the choice of programming languages and technologies. The use of graphics processing units (GPU) to perform parallel computing is one of the most common examples of SIMD-architecture. For example, the GPU processor can be used to compute physics effects in video games or to process graphic images. SIMD-systems are also widely used in the field of signal processing, where it is necessary to perform many of the same computations on large amounts of data, for example, in the processing of images and sound. SIMD-systems can also be used in scientific and engineering computations such as numerical modeling and data analysis.

4. DISCUSSION

In the study by Dvorak and Pergl [8], the researchers have discussed the problem of rapid technological changes and they have shown that modern IT companies should apply the latest engineering technologies in order to keep up with the progress of information technology. It has been noted that software solutions quickly become obsolete, so the ability of software to evolve and adapt quickly is essential. According to scientists, the search for ways to accelerate the code is one of the key aspects in this context. It has also been noted that the ability to change software within a short period of time allows for faster adaptation to changing market requirements. The study has shown that using the latest engineering technologies can help companies improve their performance and stay competitive in the rapidly changing world of information technology [9]. It is worth agreeing with the fact that it is necessary to constantly adapt to rapidly changing market requirements in the modern world of information technology. New technologies and innovations in the IT sector can significantly affect the performance of companies and their competitiveness. The author also agrees with the fact that finding ways to accelerate the code and improve the efficiency of software is an important aspect in this context. However, it is important to understand that both the latest engineering technologies and also modern methods for project and team management can help companies succeed and remain competitive.

Lagadec and Plantec [10] have reviewed in their scientific work the various technological advances in systems based on the Smalltalk and they have emphasized that modern programmable systems with a full interactive development environment can be expanded promptly and adjusted to specific contexts. This study notes that the implementation of such technological solutions requires efficient code assembly. This, in turn, determines the need to find new and more efficient methods for optimizing the code assembly process. Such methods can provide higher performance and acceleration of the work of systems, which is an important factor in modern information technology. In general, such technological advances can increase the efficiency of various systems and provide a better quality of work [11-13]. The theme outlined in the study regarding the optimization of process of the code assembly is indeed important and relevant in information technologies. New and more efficient optimization techniques can significantly improve system

performance and speed up programs. However, it should be noted that such technological solutions can be difficult to develop and require high qualifications of specialists. In addition, the effectiveness of the application of new technologies may depend on specific tasks and contexts, so it is not always possible to achieve the declared performance. In general, the study by Lagadec and Plantek emphasizes the importance of finding new methods for optimization and applying modern technological advances to improve the quality of work of systems [10].

In the scientific work by Wang et al. [14-16] regarding the study of prospects for the development of highly reliable computers using a reliable architecture of a set of instruction and emulators, the researcher draws attention to the importance of the architecture of computing machines and the speed of code assembly for the reliability and durability of functioning of modern computing technology. According to the researcher, the acceleration of code assembly can be successfully implemented through the use of various architectural solutions in computer technology when strictly following the program instructions. However, this requires high professional skills of the service personnel of enterprises and organizations that use this technique, and their ability to navigate freely in the most complex architectures of computing machines. The highly reliable computers with increased exploitation duration and a high degree of reliability can be the result of the acceleration of the code assembly. The author agrees with the fact of the study that the architecture of computing machines plays an important role in the performance and reliability of computers, and acceleration of the code assembly can lead to improved system efficiency. However, it is also important to pay attention to the aspects of security and data protection when developing computers and technologies, but security should not be sacrificed in favor of performance and the speed of code assembly [17-19]. In addition, along with the architectural solutions, software testing and quality control, as well as fixing vulnerabilities and updating the system, are important factors in improving the reliability of computers.

In the scientific study by Wu et al. [20], it was considered the methods of mobile object tracking based on the Internet of Things (IoT) edge computing. It is stated here that the use of multiple IoT devices is required for the efficient identification of mobile objects. The use of the infrastructure of edge computing (that allows offloading high-energy computing devices) is a key factor for a successful solution to such problems. However, to achieve the desired results, it is required to accelerate the code assembly by means of changing the architecture of these computing devices. It has also been noted that in order to implement such solutions, it is necessary to have highly qualified personnel, which are able to work effectively with various computer architectures. In general, the proposed methods can be useful for solving the tasks of tracking mobile objects in the IoT environment [21]. Methods for tracking mobile objects based on edge computing in the Internet of Things, proposed in the study by Wu et al. [20] can be useful for solving tasks in this area. However, the effective identification of mobile objects requires the use of multiple IoT devices and the use of edge computing infrastructure. In addition, it is stated in the study that in order to achieve the desired results, it is required to accelerate the code assembly by changing the architecture of these computing devices and the availability of highly qualified personnel.

Jiang et al. [22] emphasize in their research work on the development of edge computing themes with energy consumption that wireless devices have significant capabilities to provide a wide range of services to users. Including data segregation, real-time local analytics, as well as transmission relaying. To achieve wider functionality of such devices, the researchers consider it necessary to perform the high-quality assembly of the program code and the competent selection of architectural solutions. At the same time, it is noted that a significant reduction in the power consumption of these devices can be achieved through the use of edge computing technology and the use of various optimization algorithms. This allows for significantly increasing the efficiency and performance of these devices and reducing energy costs [23, 24].

The opinion on this study can be expressed in the following way: the study presents an interesting topic and contains valuable information about capabilities of the wireless devices and the optimization of their power consumption, which can be useful for the development of more efficient and productive devices.

In the joint scientific work of Uddin et al. [25], it is discussed the main aspects of modeling of micro-threaded multi-core architectures of computing machines based on the cache memory. Scientists point to the difficulty of modeling systems with a large number of cores that obey different instructions. It is noted that building the architecture of computing machines and the quality and speed of code assembly are important factors for the effective functioning of such systems. Moreover, it has been pointed out the need to use software tools that can facilitate the task of modeling micro-threaded multi-core architectures, as well as simplify the process of developing and debugging applications for such systems. In this work, it is also considered the possibility of improving the performance of multi-core systems by means of optimizing the operation of cache memory and the control of data flow [26, 27]. This study is an important contribution to the development of the field of multi-core system modeling and may be useful for the development of more efficient and productive computing systems in the future.

The group of research scientists represented by He et al. [28] considered in the joint study the issues of efficient inference mechanisms on GPU for a binary neural network. According to the researchers, neural networks have become increasingly powerful and popular in mobile computing in recent years. This determines the high relevance of building a high-quality architecture of computing machines and devices that require significant computing resources. However, significant differences in the architecture can make it impossible to transfer the implementation of technologies to mobile devices, which leads to a decrease in their performance. In their work, scientists pay attention to the efficiency of inference mechanisms on the GPU, which can significantly accelerate the process of learning and predicting results for neural networks. The results of the study can be used in the future to create more efficient architectures of computing devices. The study conducted by a group of scientists-researchers represents an important contribution to the development of efficient GPU inference mechanisms for a binary neural network. The results of the work can be useful for creating more efficient architectures of computing devices and improving the performance of mobile computations. However, it should be considered that the effectiveness of the inference mechanism may depend on the specific architecture and requirements of the application, so additional research and adaptation of

methods for specific tasks may be required.

Castello et al. [29] in their joint research work explored the general methods for optimizing deep training on multi-core ARM processors. Scientists believe that the platform for distributed training of deep neural networks is an effective tool for high-precision networks with complex architecture. However, increasing the number of cores in a processor leads to difficulties in designing effective methods of optimization for deep training [30]. In their work, the researchers have considered various optimization methods and have proposed several recommendations that can improve the efficiency of deep neural network training on multi-core ARM processors. In addition, scientists drew attention to the importance of changing the architecture of computing machines and acceleration of the code assembly to improve the performance of multi-core ARM processors [31, 32]. The research carried out by the scientists is an important contribution to the development of methods for optimizing deep training on multi-core ARM processors. Recommendations proposed by the researchers can help improve the efficiency of training deep neural networks on such processors. However, it is worth considering that this study is limited to the use of ARM processors and it does not consider other platforms and architectures.

Crookes [33] in his research work considered the prospects for building a highly efficient computer architecture for image processing. According to the study, in the future, the use of standard microprocessor technology will have significant prospects in building high-speed computers for image processing. To accelerate the code assembly process, various architectures of computing systems will be used. Crookes also notes that there are many approaches to building the architecture of computing machines for image processing. Some of these approaches use specialized hardware such as graphics processing units, while others use standard microprocessors. In general, the study led to the conclusion that the use of standard microprocessor technology in combination with various architectures of computer systems is an effective approach to building computing machines for image processing [34]. This approach can be used to increase the speed and accuracy of image processing. A study by Crookes [33] provides information on the prospects for building a highly efficient computer architecture of computing machines for image processing. It is considered various approaches to building such an architecture and it is noted that the use of standard microprocessor technology has significant prospects in this area. However, despite this, it is also necessary to consider other approaches that can use specialized hardware, such as graphics processors. The study also highlights the importance of accelerating the code assembly process to increase the efficiency of computing machines. This study contains interesting information about the prospects for building a high-performance computer architecture for image processing; however, to fully understand this area, it is necessary to consider the diversity of approaches and architectural solutions.

The reviewed studies have practical implications for computing and offer insights for the development of new tools, methodologies, and techniques. They emphasize the importance of efficient code assembly and architectural choices in improving system performance, reliability, and efficiency. They highlight the need for IT companies to adopt the latest engineering technologies and optimize code assembly processes to keep up with rapid technological changes and remain competitive. They also emphasize the significance of architectural solutions in accelerating code assembly and enhancing the performance of computing systems. In addition, the researchers explore specific domains such as IoT edge computing, mobile object tracking, power consumption optimization, multi-core architectures, neural network inference mechanisms, and image processing. They provide recommendations for leveraging edge computing infrastructure, optimizing code assembly, and considering different architectural approaches to improve system functionality, energy efficiency, and accuracy.

5. CONCLUSIONS

Advancements in computer architecture, particularly the shift from von Neumann to backbone architecture, have led to more efficient data processing and use of computational resources. Also, M. Flynn's classification continues to be a fundamental tool in understanding, comparing, and implementing different architectures of computing systems, despite the rapid evolution in the field. For combating software obsolescence, accelerating code assembly process emerges as a crucial strategy, which, however, necessitates personnel adept in complex machine architectures. The adoption of edge computing and architectural modifications, particularly in IoT environments and wireless devices, can enhance performance and energy efficiency. The importance of efficient GPU inference mechanisms and deep learning optimization on multi-core ARM processors is underlined for improving the performance in deep learning and artificial neural networks. Even though standard microprocessors have high-speed image processing potential, combining them with different system architectures or specialized hardware like GPUs can lead to improved performance.

In the modern world, there are a huge number of different architectures of computing machines that are used to process and analyze large amounts of information. Each of these architectures has its own characteristics and can be effectively used depending on the specific task. As part of the study, it was analyzed various types of architectures of computing machines and their application in accelerating the code compilation. One of the main goals of the study was to find out what types of architectures can be most effectively used to accelerate the process of compiling code. During the study, certain architectural solutions were identified that can help increase the performance of computers and speed up the process of code compilation.

This is one of the main conclusions of the study: the choice of a particular computer architecture can have a great impact on performance when performing various tasks, including acceleration of the process of code compilation. For example, some types of architectures, such as multi-core processors, can provide significant performance gains and reduce the time of the code compilation. However, for each specific task, it is necessary to choose the architecture that will be most effective. For example, the architecture with a large number of cores can be more efficient for the tasks connected with processing a large amount of data; while the tasks requiring high computational accuracy may require a more complex architecture with a large amount of cache memory and high speed. Thus, the choice of a specific architecture of computing machines is a key factor affecting the performance when performing various tasks, including the acceleration of code

compilation. Further research in this area will allow the development of more efficient computer architectures that will increase productivity and acceleration of the process of software development.

Building on the results of the article, future research can delve into advanced and unconventional architectures such as quantum and neuromorphic computing, seeking their potential benefits in performance and energy efficiency. Further exploration of different compilation techniques in relation to specific architectures is also warranted. The role of artificial intelligence optimizations, particularly with GPU inferences and deep learning, could provide more insights into reducing training time and enhancing AI systems' efficiency. An investigation into optimizing different architectures for edge computing and IoT could yield valuable results. Research into software-hardware co-design, power efficiency, and the influence of non-technical factors like team expertise and economic considerations can also provide a more comprehensive understanding of system performance.

REFERENCES

[1] Liu, Y., Li, Y., Qi, Z., Guan, H. (2019). A scala based framework for developing acceleration systems with FPGAs. Journal of Systems Architecture, 98: 231-242. https://doi.org/10.1016/j.sysarc.2019.08.001

[2] Harris, S., Harrris, D. (2021). Digital Design and Computer Architecture, RISC-V Edition. Burlington: Morgan Kaufmann.

[3] Hennessy, J., Patterson, D. (2017). Computer Architecture, 6 Edition. Burlington: Morgan Kaufmann.

[4] Maroun, E., J., Schoeberl, M., Puschner, P. (2021). Compiling for time-predictability with dual-issue single-path code. Journal of Systems Architecture, 118: 102230. https://doi.org/10.1016/j.sysarc.2021.102230

[5] Ying, J., Hou, R., Zhao, L., Yuan, F., Zhao, P., Meng, D. (2022). CPP: A lightweight memory page management extension to prevent code pointer leakage. Journal of Systems Architecture, 130: 102679. https://doi.org/10.1016/j.sysarc.2022.102679

[6] Singh, J., Singh, J. (2021). A survey on machine learning-based malware detection in executable files. Journal of Systems Architecture, 112: 101861. https://doi.org/10.1016/j.sysarc.2020.101861

[7] Aryal, H. (2016). Computer Organization and Architecture. https://nitsri.ac.in/Department/Electronics%20&%20Communication%20Engineering/Chapter1-Introduction.pdf.

[8] Dvořák, O., Pergl, R. (2022). Tackling rapid technology changes by applying enterprise engineering theories. Science of Computer Programming, 215: 102747. https://doi.org/10.1016/j.scico.2021.102747

[9] Yang, Z., Yung, S., Bodeveix, J.P.., Filali, M., Wang, T., Zhou, Y. (2021). Multi-task Ada code generation from synchronous dataflow programs on multi-core: Approach and industrial study. Science of Computer Programming, 207: 102644. https://doi.org/10.1016/j.scico.2021.102644

[10] Lagadec, L., Plantec, A. (2014). Preface to the special issue on advances in Smalltalk based systems. Science of Computer Programming, 215: 102747. https://doi.org/10.1016/j.scico.2014.07.004

[11] Hartmann, C., Häublein, K., Reichenbach, M., Fey, D. (2018). IPAS: A design framework for analysis, synthesis and optimization of image processing applications for heterogenous computing architectures. Journal of Real-Time Image Processing, 14(3): 549-564. https://doi.org/10.1007/s11554-016-0587-x

[12] Ginters, E. (2019). Augmented reality use for cycling quality improvement. Procedia Computer Science, 149: 167-176. https://doi.org/10.1016/j.procs.2019.01.120

[13] Ginters, E., Barkane, Z., Vincent, H. (2010). Systems dynamics use for technologies assessment. In the 22th European Modeling & Simulation Symposium (EMSS 2010), Fes, Morocco, pp. 357-363.

[14] Wang, S.P. (2021). Design high-confidence computers using trusted instructional set architecture and emulators. High-Confidence Computing, 1(2): 100009. https://doi.org/10.1016/j.hcc.2021.100009

[15] Kale, A.P., Sonawane, S., Wahul, R.M., Dudhedia, M.A. (2022). Improved genetic optimized feature selection for online sequential extreme learning machine. Ingenierie des Systemes d'Information, 27(5): 843-848. https://doi.org/10.18280/isi.270519

[16] Gao, J., Ismail, N., Gao, Y. (2022). Computer big data analysis and predictive maintenance based on deep learning. Ingenierie des Systemes d'Information, 27(2): 349-355. https://doi.org/10.18280/isi.270220

[17] Kashtanov, S.F., Polukarov, Y.O., Polukarov, O.I., Mitiuk, L.O., Kachynska, N.F. (2021). Specifics of modern security requirements for software of electronic machine control systems. INCAS Bulletin, 13(S): 87-97. https://doi.org/10.13111/2066-8201.2021.13.S.9

[18] Nass, O., Kamalova, G., Shotkin, R., Rabcan, J. (2021). Analysis of methods for planning data processing tasks in distributed systems for the remote access to information resources: Topic: Communication and control systems and networks. In 2021 International Conference on Information and Digital Technologies (IDT), Zilina, Slovakia, pp. 273-276. https://doi.org/10.1109/IDT52577.2021.9497583

[19] Bapiyev, I., Kamalova, G., Yermukhambetova, F., Khairullina, A., Kassymova, A. (2021). Neural network model of countering network cyber attacks using expert knowledge. Journal of Theoretical and Applied Information Technology, 99(13): 3179-3190.

[20] Wu, Y., Tian, P., Cao, Y., Ge, L., Yu, W. (2022). Edge computing-Based mobile object tracking in internet of things. High-Confidence Computing, 2(1): 100045. https://doi.org/10.1016/j.hcc.2021.100045

[21] Sandra, L., Trisetyarso, A., Ramadhan, A., Abdurachnan, E., Lumbangaol, F., Isa, S.M. (2021). Social network analysis algorithms, techniques and methods. In 2021 International Conference on Advanced Mechatronics, Intelligent Manufacture and Industrial Automation (ICAMIMIA), Surabaya, Indonesia, pp. 182-189. https://doi.org/10.1109/ICAMIMIA54022.2021.9807748

[22] Jiang, C., Fan, T., Gao, H., Shi, W., Liu, L., Cerin, C., Wan, J. (2020). Energy aware edge computing: A survey. Computer Communications, 151: 556-580. https://doi.org/10.1016/j.comcom.2020.01.004

[23] Culler, D., Singh, J.P., Gupta, A. (1998). Parallel Computer Architecture. Burlington: Morgan Kaufmann.

[24] Bondarenko, I.N., Galich, A.V., Slipchenko, N.I., Troitski, S.I. (2012). Cone-shaped resonator the high-

order mode oscillation trasducers. In 2012 22nd International Crimean Conference" Microwave & Telecommunication Technology", Sevastopol, Ukraine, pp. 565-567.
[25] Uddin, I., Poss, R., Jesshope, C. (2014). Cache-based high-level simulation of microthreaded many-core architectures. Journal of Systems Architecture, 60(7): 529-552. https://doi.org/10.1016/j.sysarc.2014.05.003
[26] Mittal, S., Verma, G., Kaushik, B., Khanday, F.A. (2021). A survey of SRAM-based in-memory computing techniques and applications. Journal of Systems Architecture, 119: 102276. https://doi.org/10.1016/j.sysarc.2014.05.003
[27] Stepanchuk, O., Bieliatynskyi, A., Pylypenko, O. (2019). Modelling the bottlenecks interconnection on the city street network. In International Scientific Siberian Transport Forum, Novosibirsk, Russia, pp. 889-898. https://doi.org/10.1007/978-3-030-37919-3_88
[28] He, S., Meng, H., Zhou, Z., Liu, Y., Huang, K., Chen, G. (2021). An efficient GPU-accelerated inference engine for binary neural network on mobile phones. Journal of Systems Architecture, 117: 102156. https://doi.org/10.1016/j.sysarc.2021.102156
[29] Castello, A., Barrachina, S., Dolz, M.F., Quintana-Orti, E.S., Juan, P.S., Tomas, A.E. (2022). High performance and energy efficient inference for deep learning on multicore ARM processors using general optimization techniques and BLIS. Science of Computer Programming, 125: 102459. https://doi.org/10.1016/j.sysarc.2022.102459
[30] Bondarenko, I.N., Lavrinovich, A.A. (2007). Investigation of the thin-film high-temperature superconductivity coplanar line. Telecommunications and Radio Engineering (English translation of Elektrosvyaz and Radiotekhnika), 66(7): 597-605. https://doi.org/10.1615/TelecomRadEng.v66.i7.30
[31] Ungurean, I., Gaitan, N.C. (2020). A software architecture for the industrial internet of things-A conceptual model. Sensors (Switzerland), 20(19): 5603. https://doi.org/10.3390/s20195603
[32] Aviv, I., Barger, A., Kofman, A., Weisfeld, R. (2023). Reference Architecture for Blockchain-Native Distributed Information System. IEEE Access, 11: 4838-4851. https://doi.org/10.1109/ACCESS.2023.3235838
[33] Crookes, D. (1999). Architectures for high performance image processing: The future. Journal of Systems Architecture, 45(10): 739-748. https://doi.org/10.1016/S1383-7621(98)00035-6
[34] Aviv, I., Gafni, R., Sherman, S., Aviv, B., Sterkin, A., Bega, E. (2023). Infrastructure From Code: The Next Generation of Cloud Lifecycle Automation. IEEE Software, 40(1): 42-49. https://doi.org/10.1109/MS.2022.3209958