The object of the study is the phenomenon of an extreme increase in the time of program code execution at certain sizes of data processed by it. The problem to be solved was to verify the general nature of the phenomenon for different equipment.

The evolution of modern computing technology, its RAM often takes place in an extensive way – by increasing the number of structural elements. Problems can manifest themselves in the fact that periodic processes in the code begin to exhibit a resonance effect, which leads to different indicators of data processing time, the sizes of which are multiples and non-multiples of the block structures. The work is studied the influence of the dimensionality of data blocks on the speed of execution of the cycle that iterates them. The tools of differential regression analysis are used. Experiments were carried out on equipment with different architecture, type and amount of RAM, running different operating systems. In all of them resonant effects were revealed. It led to differences in the average code execution time by 1.6–3.6 times, and the time of memory access operations increased up to 136 times. Special attention was drawn to the fact that the increase in operating time was found for structures whose size is a power of two multiple (2N), specifically for the values 512 and 1024. These dimensions are present in many types of tasks, in particular, cryptographic purposes or stream-based data processing. Following the recommendations given in the paper can help identify time delays in applications, and improve the performance of applications by eliminating them

Keywords: program performance, computer memory (RAM), operation time, linear regression, differential analysis

UDC 004.62
DOI: 10.15587/1729-4061.2025.322989

ESTIMATION OF SOFTWARE STRUCTURES DIMENSION INFLUENCE ON DATA PROCESSING TIME INCREASING

Yevhen Danylets
PhD, Associate Professor*
Dmytro Korchevskyi
Doctor of Pedagogical Sciences*
Serhii Novak
PhD, Associate Professor*
Denys Samoilenko
Corresponding author
PhD, Associate Professor*
E-mail: denniksam@gmail.com
Mykola Sulima
PhD*
*Department of Information Technologies and Fundamental Study
Odesa Technological University "STEP"
Sadova str., 3, Odesa, Ukraine, 65023

Received 27.11.2024
Received in revised form 23.01.2025
Accepted date 10.02.2025
Published date 28.02.2025

How to Cite: *Danylets, Y., Korchevskyi, D., Novak, S., Samoilenko, D., Sulima, M. (2025). Estimation of software structures dimension influence on data processing time increasing. Eastern-European Journal of Enterprise Technologies, 1 (9 (133)), 24–34. https://doi.org/10.15587/1729-4061.2025.322989*

1. Introduction

The speed of computing systems has been and remains one of the most relevant indicators of the quality of their work. The speed of program code execution is influenced by a large number of factors. These include both hardware indicators such as the clock frequency of the processor or the bit rate of the RAM, and the complexity of the program algorithms used in the code. The total code execution time can also be affected by dimensional characteristics – the volume of data structures processed by the program. Moreover, it is not about the total volume of data, the dependence on which seems obvious, but about the size of single, relatively small structures, blocks into which the processed data is divided.

The cluster principle of organizing the RAM of computing devices (computers, smartphones, tablets, etc.) can be considered the basis for the fact that the specified dependence will not be monotonic. It is expected that there will be a different effect for structures whose sizes are multiples and not multiples of the memory cluster. Moreover, it seems logical to assume that structures that are completely embedded in memory clusters are more optimal and faster to process. Accordingly, when graphically displaying the dependence of the operating time on the size of the structure, certain "failures" are expected – local minima of the operating time for multiples of the cluster of values. However, published experimental studies from time to time allow to identify cases that indicate completely opposite phenomena – deviation of time indicators [1–6].

In addition to the general structure, RAM can differ in manufacturing technologies, manufacturer, and access to it can be regulated by different hardware environments, operating systems. All this can affect the time indicators of application programs with data at different sizes of their structures in different ways.

It is considered relevant to conduct a series of studies on the influence of the dimensionality of program structures on the speed of their processing with the involvement of performers with different architectures, hardware and software, type and size of RAM, as well as its manufacturer.

2. Literature review and problem statement

In [1], research was conducted, part of which was to identify the influence of the size factor on the speed of program execution. The influence of different ways of placing blocks in the performer's memory on this time was studied. At the same time, the work focused on analyzing the total size of the available memory and comparing it with the amount of data

used in the program. At the same time, a detailed impact of the block dimension was not carried out. Presumably, due to the lack of a direct goal of considering the impact of the block dimension on the time graphs in the work, extreme time deviations were not registered.

A different approach to the problem was taken in [2]. A detailed analysis of the time indicators of operation in the memory of competing parallel tasks was carried out. Modern (at the time of writing the article) types of microcircuits (DDR4) were selected for the study and specialized equipment was used to conduct the study. However, the indicators obtained in the work were ultimately expressed in units of internal cycles. On the one hand, this is universal, but on the other hand, it does not provide for the possibility of differences in the actual completion time of different cycles. Also, the focus of the study was precisely the effects of task competition, the factor of the influence of the dimensionality of tasks on time indicators was not studied.

In work [3], deviations in time indicators were found when modeling real systems using computational ones. The study measured that the deviation can reach a scale factor of 2. This deviation is explained by the hardware features of the system used for modeling. However, the fact of the influence itself was not studied in detail, as were the features associated with the dimensionality of the data blocks of the models.

The fact of differences in the processing time of different data blocks was studied in work [4] from the point of view of information security in relation to potential data leakage. At the same time, the aim of the work was focused on issues of an economic nature and a detailed analysis of the influence of the dimensionality of the blocks on time was not carried out. A direct assumption was used regarding the increase in running time for larger data sizes.

The very fact of the influence of dimensionality on code execution time was demonstrated in [5]. It was found that at certain block sizes, an extreme increase in program execution time is observed. A hypothesis was expressed regarding the prevalence of the detected behavior. At the same time, only one type of executor was studied in detail in the work, and at present it can be stated that the technology used in the work is outdated. The work also used an imperfect mathematical research apparatus that did not separate the contributions of various factors to the total program execution time.

In [6], on the contrary, a detailed study of the ultra-new type of STT-MRAM RAM was conducted. In the process, a deviation was found in the key time indicators of its operation. This can be considered indirect evidence that the dependence on the dimensionality of data blocks remains in new microcircuits. However, the influence of the dimensionality of information units on these indicators was not studied in the work. The work is also focused on a separate type of memory chips, which justifies the feasibility of comparative analysis with other technologies.

Trends in new RAM technologies are disclosed in [7]. A thorough study of various types and areas of use of modern developments has been conducted. However, the time indicators characterizing the mentioned technologies did not include information on time deviations associated with dimensional factors.

All this gives grounds to argue that it is advisable to conduct a study devoted to the analysis of the influence of the dimensionality of data blocks on the deviation of time indicators of program code execution for modern equipment with different hardware environments – processor, operating system, etc. It is also considered advisable to create a universal testing methodology for different performers with different hardware components in order to identify critical indicators of data dimensionality and their targeted avoidance.

3. The aim and objectives of the study

The aim of the study is to experimentally test the hypothesis of the existence of exceptional sizes of program structures, which lead to an extreme increase in the time of their program processing. If the hypothesis is confirmed, this will make it possible to avoid an increase in the execution time of programs by purposefully correcting the sizes of their program structures.

To achieve the aim, the following objectives were set:

– to implement a methodology for determining the exec cution time of program instructions with different sizes of structures, to implement tools to confirm or refute the stated hypothesis;

– to conduct a series of experiments measuring the exec cution time of programs depending on the dimensionality of program structures, to use different hardware and software equipment; to investigate the nature of the hypothesis;

– to assess the degree of influence of the dimensionality of structures on the execution time of programs.

4. Materials and methods

The object of the study was the phenomenon of extreme increase in the time of program code execution at certain sizes of data processed by it. The hypothesis of the study is the inevitability of the existence of exceptional sizes of data structures, which lead to an extreme increase in the time of their program processing, for any electronic computing equipment. It is assumed that the number, size and density of exceptional sizes may differ in different equipment, the very fact of the presence of these structures is subject to verification. The hypothesis is verified by the inductive method, using a limited number of experimental observations on the available equipment. As a simplification, the time of code execution in the work is considered to be the time measured by software means. Additional control of physical time by certified measuring means was not carried out.

Focusing on the maximum practicality of the results obtained, a number of various computing devices were selected for conducting experiments. The main selection criterion was the prevalence of this type of device in the field of modern applied programming. Accordingly, the equipment under study included personal computers, in particular, in the form factor of laptops, tablets and smartphones. According to advertising restrictions, the brands of manufacturers and models of devices are not given in the work, only the characteristics of their processors (CPU), random access memory (RAM) and operating system (OS) are noted. The corresponding data are summarized in Table 1. To refer to a specific device in the text of the work, the parameter “Ref” is used, also included in the table.

In order to determine the time of operations that are close to the basic processor operations, a method based on differential analysis and linear regression was used.

Table 1

Characteristics of the experimental equipment

Ref	CPU	RAM	OS
1-L	AMD A6-5200 APU 800MHz	Hynix Semiconductor DDR3L-800 (8Gb)	Kali Linux 2024.3
1-W			Windows 10, x64
2-L	Intel(R) Core(TM) i9-13900H 2.60 GHz	Hynix Semiconductor LPDDR5-6400 (16Gb)	Kali Linux 2024.3
2-W			Windows 11, x64
3-A	Samsung Exynos 7870 1.6 GHz ARM Cortex-A53	Samsung LPDDR3-1600 SDRAM eMMC 5.1 (3Gb)	Android 10
4-A	Mediatek Helio G80 MT6769V 2.0 GHz ARM Cortex-A75	Hynix Semiconductor LPDDR4X-2133 eMMC 5.1 (4Gb)	Android 11
5-W	Intel(R) Core(TM) i3-7100U 2.40GHz	Samsung DDR4-2400T (4Gb)	Windows 10, x64
6-W	Intel(R) Core(TM) i5-1035G4 1.50 GHz	Samsung DDR4-2666 (16Gb)	Windows 10, x64
7-M	Intel(R) Core(TM) i5-8210Y 1.60 GHz	Hynix Semiconductor LPDDR3-2133 (8Gb)	macOS Sonoma 14.7.1
8-M	Apple Inc TSMC M1 3.2 GHz	Hynix Semiconductor LPDDR4X-4266 (16Gb)	macOS Ventura 13.4.1
9-X	Intel(R) Xeon(R) Gold 6342 2.80 GHz	Samsung DDR4-3200 (1024Gb)	Windows Server 10

The multiple repetition method used in the previous work [5] takes into account all components in the total operation time – computational (mathematical), management (organization of cycles) and, in fact, memory tracing. Correct conclusions about individual components can be obtained only when ensuring the invariance of others. That is, the analysis of the impact of memory tracing operations on the total time can be objectively carried out only with the same number of cycle repetitions, as well as with a constant number of mathematical operations within each of the cycles. This limits the study of the behavior of algorithms when other parameters change.

To implement differential comparison, two code blocks were compiled, differing only in the memory tracing operation. The maximum correspondence of the blocks was observed in terms of the total number of operations, in particular the operations of index dereferencing of the pointer (px[]), the number of its reads and writes, as well as mathematical calculations. Table 2 shows fragments of the codes of the two program blocks.

Table 2

Control and reference program blocks

Control block	Reference block

for (int j = 1; j < jump_count; j+=1) {
shift = jump_step * j;
px[shift] = px[shift] + px[shift];
}
	for (int j = 1; j < jump_count; j+=1) {
	shift = jump_step * j;
	px[1] = px[1] + px[1];
	}

The blocks were executed sequentially, within the same operational flow, which additionally created similar execution conditions for them. The only difference built into the algorithms is the offset value, which is counted from the beginning of the array: px[shift] in the control block versus px[1] in the reference block. The value "1" was chosen to additionally equalize the operational complexity, since using the offset "0" would not involve pointer arithmetic when calculating the offset. The variable offset value "shift" in the reference block is not used, but is calculated in the same way as in the control block.

The semantics of the control block consists in sequentially performing operations to read and write elements of the array "px" that are at a distance of "jump_step" from each other. Problems of this type can arise in matrix algorithms, especially if the matrix is given by a one-dimensional data array. Similar situations are also typical in spectral analysis problems during transitions between multiple harmonics of the spectrum.

The difference in execution times of the control and reference code blocks was used as the variable under study. Its regression analysis tools allow, firstly, to distinguish between the constant and dynamic components that affect the difference in code execution time. Taking into account the fact that the compiled code can be slightly changed by compiler optimization tools, it was assumed that there is a potential for the remainder of service operations even in the difference in execution times of the specified blocks. However, these operations should be a constant value that can be detected by regression tools.

Secondly, regression analysis allows to control the reliability of the obtained data by calculating the correlation coefficient. This indicator detects data deviations (misses) that inevitably arise in multitasking operating environments such as personal computers or servers. Launching system processes, especially those with high priority, can change the environment of application code execution and, as a result, affect its execution time. However, high-priority processes usually have an impulse nature, which leads to distortion of the results only for individual experimental points. Such deviations will lead to a decrease in the correlation coefficient of the obtained data, due to which they can be detected and leveled. In the framework of this work, when detecting a reduced correlation of the data, the entire series of measurements was repeated in full.

The nature of the dependence revealed by regression analysis does not play a fundamental role in the task, since it is set at the stage of setting up the experiment. The linear regression variant was chosen as the most studied and effective in data analysis.

Accordingly, a linear change in the number of repetitions of program blocks was provided. The number of points forming the sample for correlation analysis was chosen for reasons of maintaining statistical reliability at the level of 0.95. When assessing the correlation bias at the level of $(n-1)/n$, $n=20$ was chosen.

Scaling of variables before conducting correlation analysis was carried out in order to ensure greater sensitivity of the correlation coefficient. The coordinates of the points processed by regression differ significantly in magnitude. Thus, with an expected time of one processor operation at the level of 1 ns, the ratio of the number of repetitions to the obtained time (in seconds) will be close to 10^9. Scale factors are used both for the measured time and for the number of repetitions. In the following program fragment, they correspond to the variables "x_scale" and "y_scale":

```
for (int k = 1; k <= regr_count; ++k) {
      start = clock();
      for (int i = 0; i < regr_step * k; ++i) { Control block }
      end = clock();
     time1 = (double)(end - start) / CLK_TCK;

      start = clock();
     for (int i = 0; i < regr_step * k; ++i) { Reference block }
      end = clock();
     time2 = (double)(end - start) / CLK_TCK;

      x = regr_step * k * x_scale ;
      y = (time1 - time2) * y_scale;
}
```

The results of the code execution were stored in a file and processed after its completion.

As a mathematical apparatus for processing experimental data, the linear regression method was used, adapted to computational problems. Formulas that use the deviation of measured values from their average values look mathematically simpler, however, they require data processing in two stages. In the first, the average values are calculated, in the second, the deviations. For practical use, the following algorithm was implemented for one pass of the program loop.

To formalize further expressions, let's denote as x the normalized number of iterations of the code repetition loop, as y the normalized difference in execution times of the studied and reference loops. It is assumed that between the values y and x, represented by discrete experimental points y_i and x_i, there should be a linear dependence in the form:

$$y = a_N + b_N x.$$

The index N additionally emphasizes the fact that the dependence coefficients are calculated separately for each value of the array tracing step N ("jump_step" in the program fragments). In this case, the values of a_N and b_N are defined as [8]:

$$a_N = \frac{s_y s_{xx} - s_x s_{xy}}{n_r s_{xx} - s_x^2}, \quad b_N = \frac{n_r s_{xy} - s_x s_y}{n_r s_{xx} - s_x^2}, \qquad (1)$$

where n_r – the number of steps of the regression analysis (the number of experimental points, "regr_count" in the program fragments). The following notations are also used in the formulas:

$$s_x = \sum_{i=0}^{n_r} x_i, \quad s_y = \sum_{i=0}^{n_r} y_i, \quad s_{xy} = \sum_{i=0}^{n_r} x_i y_i,$$

$$s_{xx} = \sum_{i=0}^{n_r} x_i x_i = \sum_{i=0}^{n_r} x_i^2, \quad s_{yy} = \sum_{i=0}^{n_r} y_i y_i = \sum_{i=0}^{n_r} y_i^2.$$

The value of the linear correlation coefficient between y and x using the introduced notations will be:

$$r_{xy} = \frac{n_r s_{xy} - s_x s_y}{\sqrt{\left(n_r s_{xx} - s_x^2\right)\left(n_r s_{yy} - s_y^2\right)}}. \qquad (2)$$

(1) and (2) were chosen for reasons of optimization of the number of calculations. They do not contain references to the average values of the quantities x or y, which allows to carry out calculations in one stage, together with the main experiments. More common in the literature expressions involving average values require data storage and two stages of processing – in the first the average values are determined, in the second the dependence coefficients themselves are calculated. This fact is emphasized because different approaches can lead to different final values of a_N and b_N, which differ within the error limits due to the limited sample size. Insignificant deviations (about 1 %) are possible when processing the data with other algorithms. With the assumed statistical reliability at the level of 0.95, such deviations will not affect the general conclusions.

5. Results of the study of the influence of structure size on execution time

5. 1. Methodology for determining the execution time of program instructions

At the beginning of the measurements, an analysis was conducted of the dependence of the execution times of the control and reference blocks, as well as their difference, on the parameter "jump_count", which determines the number of elements of the array being iterated. In practical terms, this parameter can correspond to the total number of elements in the array or its cluster. The obtained data are presented in Fig. 1.

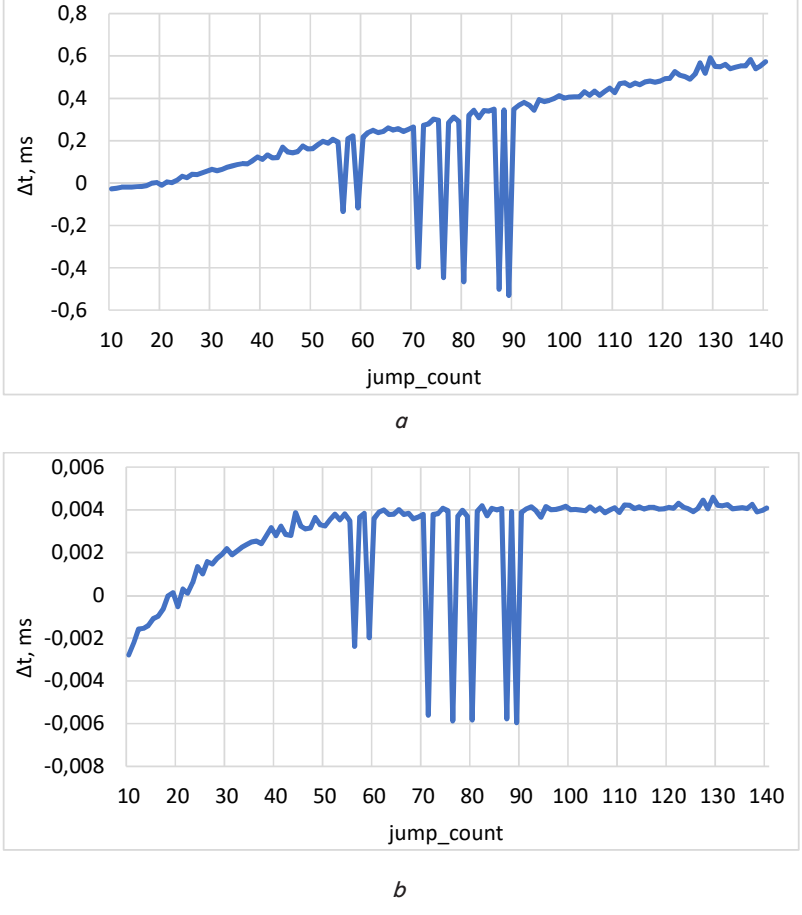


Fig. 1. Graphs of the dependence of the time difference of the control and reference code blocks on the parameter "jump_count": *a* – absolute value; *b* – normalized to a single execution

As can be seen from Fig. 1, for values of "jump_count" less than 90, unstable indicators of the specific difference time are observed. For larger values, the indicators stop changing and demonstrate a constant level. This may be caused by a high contribution of measurement errors for small numbers of repetitions, however, the given dependence in itself justifies the relevance for a separate study. Focusing on the task of studying the influence of the dimensionality of data blocks on time, it was decided to use the value from the beginning of the stability region: jump_count=100. This value was used for all subsequent experiments with all equipment.

The general method for determining the execution time of program instructions, taking into account the above, can be formulated as follows:

1. The initial value of "regr_step" is set (usually in the range of 10000-100000) and "jump_count" (equal to 100 for all experiments). The control and reference code blocks (Table 2) are executed according to the scheme given in the listing in Section 4. The execution time of each block should be about 5–50 ms in order to reliably measure the time by hardware means. If the execution time differs significantly, the value of "regr_step" varies. For different executors, the values of "regr_step" may differ due to different speeds.

2. The execution times of the control and reference blocks are sequentially measured for each regression step (regr_step, 2regr_step, ... 20regr_step), and the coefficients (1) are calculated based on the results of the series.

3. The value of "N" (corresponding to the program variable "shift") is increased by one. Repeat p. 2.

4. Repeat p. 3 for values of "N" from the range 1–1050. The values of the obtained coefficients a_N are plotted on the graph depending on "N".

The implementation of the method was performed using the C++ language, which allowed the execution of the same code on different hardware platforms with different operating systems. The data that is iterated in different steps is implemented in the form of an array of type "double", placed in the external memory for the program – "heap":

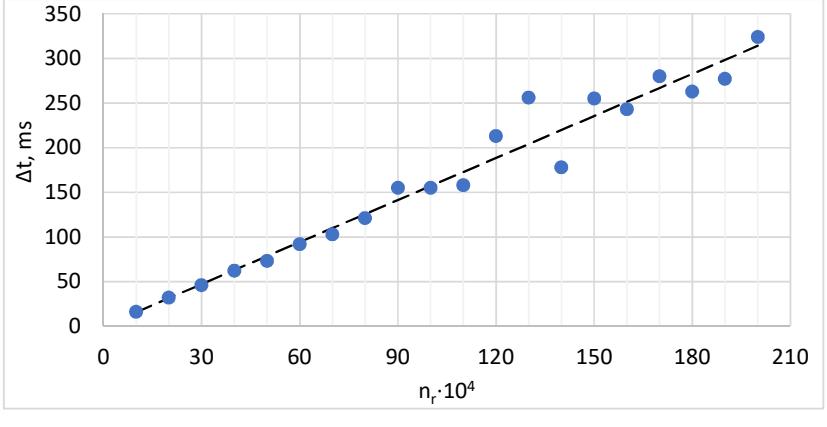
```
double*px=new double[1050*100],
```

where 1050 – the maximum iteration step (N), 100 – "jump_count", the number of iteration steps. For other experimental settings, these values can be changed.

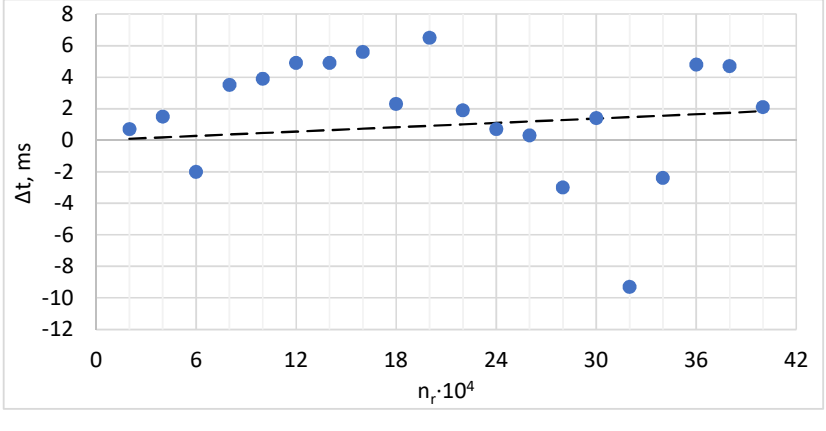
A separate experiment (item 2) was carried out in the form of a series of measurements with linearly distributed values, which specify the number of code block repetitions, with subsequent regression analysis. As an example, Table 3 shows a series of experimental points obtained for the W-2 equipment (Table 1). As can be seen from Table 3, the choice of the value of the "regr_step" parameter affected the scaling of the variables and allowed them to be brought to the same order of magnitude. This increases the sensitivity of the correlation coefficient to deviations in any coordinate and increases the confidence in the conclusions obtained through it. For other equipment, other values of the "regr_step" parameter were used, which took into account the speed of the device and were selected experimentally.

The expressions applied to the data from Table 3 (for N=127) allowed to obtain the values a_{127}=0.000034, b_{127}=1.57, r_{xy}=0.995 for the 5-W equipment and a_{127}=0.000677, b_{127}=0.046, r_{xy}=0.276 for the 2-W equipment. The line with the corresponding coefficients together with the experimental points are shown in Fig. 2.

The linear correlation coefficient allows to assess the degree of confidence in the obtained regression results. Thus, in Fig. 2, *a*, the visual arrangement of the points and the correlation coefficient close to unity (r_{xy}=0.995) indicate a high reliability of the calculated value b and confirmation of the hypothesis of the difference in program execution times. For Fig. 2, *b*, the situation is the opposite, which is expressed by a much smaller coefficient (r_{xy}=0.276). Accordingly, the result for *b* obtained in this series of measurements cannot be considered reliable, and the assumption regarding the different execution times of the control and reference blocks is not confirmed. A similar analysis was carried out for all experimental series.



a



b

Fig. 2. Experimental points from Table 1 and their regression lines: *a* – 5-W equipment; *b* – 2-W equipment

Table 3

Results of a series of measurements for 5-W and 2-W equipment with regr_count=20, jump_count=100, jump_step=127

r	$n_r \cdot 10^4$	t_1, ms	t_2, ms	Δt, ms	$n_r \cdot 10^4$	t_1, ms	t_2, ms	Δt, ms
1	10	54	38	16	2	4.9	4.2	0.7
2	20	109	77	32	4	9.9	8.4	1.5
3	30	162	116	46	6	10.6	12.6	-2
4	40	215	153	62	8	20.2	16.7	3.5
5	50	266	193	73	10	24.9	21	3.9
6	60	323	231	92	12	30	25.1	4.9
7	70	373	270	103	14	34.2	29.3	4.9
8	80	428	307	121	16	39.1	33.5	5.6
9	90	501	346	155	18	40	37.7	2.3
10	100	538	383	155	20	48.4	41.8	6.5
11	110	616	458	158	22	47.9	46.1	1.9
12	120	709	496	213	24	50.9	50.2	0.7
13	130	770	514	256	26	54.7	54.4	0.3
14	140	773	595	178	28	55.5	58.6	-3
15	150	832	577	255	30	64.2	62.8	1.4
16	160	859	616	243	32	57.7	67	-9.3
17	170	936	656	280	34	68.7	71.1	-2.4
18	180	961	698	263	36	80.1	75.3	4.8
19	190	1019	742	277	38	84.4	79.7	4.7
20	200	1097	773	324	40	85.8	83.7	2.1

From a physical point of view, both regression coefficients (a and b) have a time dimension on the nanosecond scale. For a reliable series of measurements, the value b=1.57 ns corresponds to the difference in execution times of one control and one reference block. Taking into account the fact that 100 iterations of the array are measured in one block, it is possible to estimate the average difference time of one operation as 0.0157 ns or 15.7 picoseconds.

Having supplemented the above algorithm with a point on the analysis of measured data, a general implementation of the method for determining the execution time of program instructions was obtained, designed to identify critical data sizes and assess the degree of confidence in them.

5. 2. Experimental results of measuring code execution time for different devices

Measurements of the execution times of the control and reference blocks, as well as their difference, depending on the data size (N) with all other parameters unchanged, were carried out for the equipment shown in Table 1. The results were processed according to the method given in section 5. 1 and for each series the coefficient b (specific difference time between code blocks) was determined. The summarized data are presented in the form of graphs in Fig. 3. When conducting experiments for all types of equipment, the same code specified in the previous section was used. In order to shorten the reference to the equipment, the field "Ref" from Table 1 was used.

For the 8-M equipment, which has a laptop form factor, the study was performed in two modes: 8-M-A when operating from the power supply, and 8-M-B when operating from the built-in rechargeable battery. The graphs are presented one below the other for greater convenience of visual comparison.

From the preliminary analysis of the obtained results in Fig. 3, it is immediately possible to conclude that the general nature of the phenomenon of extreme growth of code execution time at certain data sizes, detected for all experiments, is confirmed. The extrema have a pronounced local nature, that is, the growth of time is observed at a separate point, and not in its vicinity. Typical values of the dimension N for blocks for which the growth of time is observed are 128, 256 and 512, which corresponds to an integer power of the number 2^k. For individual graphs (Fig. 3, a–e, k), extrema are also observed for N=64, 192, 384, etc., which corresponds to the form 2^k+2^{k-1}. The execution time is restored to the average value after passing the extreme point.

The situation presented in [5] with multiple extrema on the dependence graph was manifested for 1-W/1-L equipment (Fig. 3, a, b) with DDR3 and 3-A (Fig. 3, e) – LPDDR3 memory. For other equipment, the number of extreme points is much smaller, but does not completely disappear. However, for one of the most modern 2-W/2-L equipment (Fig. 3, c, d) with DDR5, the number of extrema increases again, but with a reduced amplitude compared to the main ones.

The "step" effect, which consists in a smooth increase in the average difference execution time with increasing N, is observed for Android OS (3-A, 4-A), MacOS (7-M), server 9-X, as well as for the oldest 1-L/1-W equipment. For the graphs of equipment 2-L, 2-W, 5-W, 6-W, the "steps" are not observed and the background of the graph is horizontal.

The number of extrema and their amplitude is different on different graphs (Fig. 3), however, the most pronounced extrema are traced for N=512 and N=1024, other positions are either less pronounced or absent on individual graphs. In general, the hypothesis of the existence of exceptional sizes of program structures, which lead to an extreme increase in the time of their program processing, can be considered confirmed.

Experiments conducted on the same equipment under the control of different operating systems, indicate some differences in the general dependence graphs. For the pair (1-L)-(1-W) (Fig. 3, a, b) there are almost no differences. It is only possible to state that under the control of the Linux OS, the number of intermediate extrema of the execution time slightly increases compared to the Windows OS. This is especially noticeable in the right part of the graphs (for larger values of N). At the same time, the situation for the pair of graphs (2-L)-(2-W) (Fig. 3, c, d) is the opposite: under the control of the Linux OS, the number of local extrema decreases, especially noticeable in the left part of the graph. Also, for the second pair under the control of the Linux OS, a slightly smaller average difference time is observed (the graph lies closer to the abscissa axis), however, at the extreme points the value is larger than that for the Windows OS. Accordingly, it is impossible to note the obvious influence of the OS on the nature of program execution, however, the differences found can be considered as an update for further research in this area.

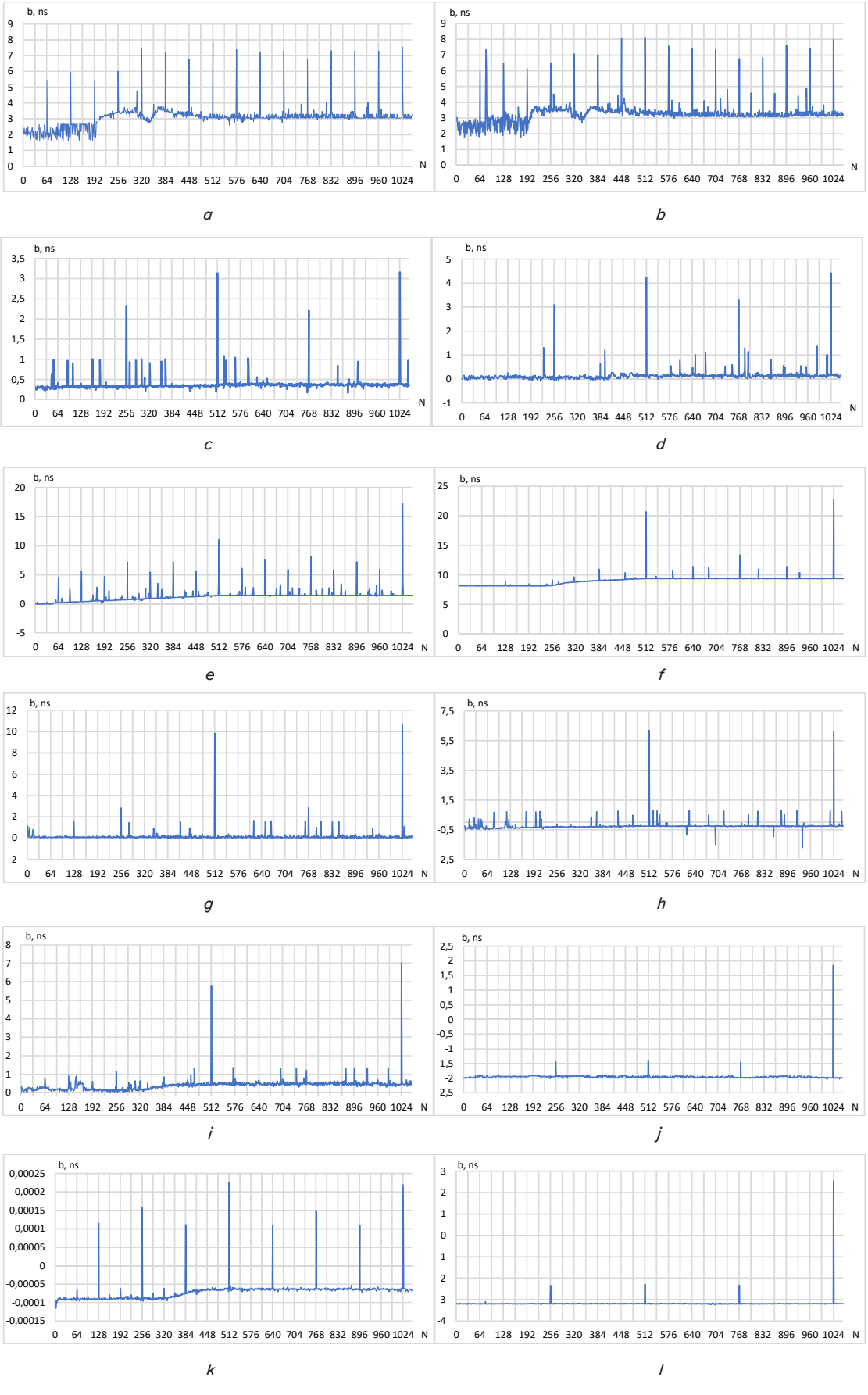


Fig. 3. Experimental graphs of the dependence of the regression coefficient b for the differential code execution time (in nanoseconds) on the size of the data block N for the equipment: *a* – 1-L; *b* – 1-W; *c* – 2-L; *d* – 2-W; *e* – 3-A; *f* – 4-A; *g* – 5-W; *h* – 6-W; *i* – 7-M; *j* – 8-M-A; *k* – 9-X; *l* – 8-M-B (Table 1)

5. 3. Assessment of the degree of influence of the dimensionality of structures on the execution time of programs

Estimating the specific influence of this difference requires a regression analysis for both the difference time and the execution times of the blocks themselves. After analyzing the data from Table 3 (part 1), it was obtained: for the reference block b_{ref}=5.49 ns, for the control block b_{ctr}=3.92 ns. Accordingly, the specific influence (calculated on the average value) will be:

$$\delta b = \frac{2b}{b_{ref} + b_{ctr}} \approx 0.33,$$

which is similar to the estimate of the deviation of 33 %.

For the second part of Table 3, the small correlation coefficient (r_{xy}=0.276), as well as the visual arrangement of the data (Fig. 1, *b*) do not allow to state a guaranteed dependence in the execution time of two different code blocks. The sign of the difference changes at different points, not even allowing to state that its nature is unambiguous. Accordingly, the assessment of the influence of the time difference was not carried out. This fact is the basis for the statement about the absence of an unambiguous manifestation of the difference in the execution time of program blocks for their arbitrary size. It seems logical to separate the means of analyzing the results with the presence and absence of the specified phenomenon.

Detailed data on all points of all series of experiments are not given within the text of the work. The deviation values indicated in Table 3 are typical for all cases, the difference lies in the absolute time associated with the speed of the equipment. Some of the series have a high correlation coefficient, which confirms the hypothesis with the deviation of the execution time of the control block at the level of 30–60 % (except for extreme points). The other part does not show a guaranteed dependence at a low correlation coefficient. This is explained in more detail in Section 6 of the work.

The graphs of Fig. 3, *h–l*, which demonstrate a stable negative average value of the difference time, only for individual extreme points acquiring a positive value, attract special attention.

Nevertheless, for all experiments, extreme points *N*=512 and *N*=1024 were found. At these points, the linear correlation coefficient confirms the hypothesis of the deviation of the execution time. In relative terms, the difference from the average value is at least 2 times greater (equipment 4-A), which correlates with some conclusions of the work [3]. For most situations, an increase of tens of times is observed, and for equipment 6-W, 8-M and 9-X it generally changes its character, deviating in the opposite direction from the average value.

Table 4 summarizes the results for the last points of the linear regression series for different equipment near *N*=512 (for values 511, 512 and 513). These results correspond to the longest measurements and allow to estimate the execution time of the blocks themselves, and not just the difference between them. The absolute values of the data of different series were chosen for reasons of regression normalization, therefore they are not directly comparable with each other.

A more detailed analysis shows that the increase in the block operation time can vary from 1.6 times (3-A) to 3.6 times (4-A, 5-W). The difference time has a wider range of coefficient changes - from 1.9 (4-A) to 132 (5-W). Moreover, for the data of 6-W, 9-X and partly for 8-M, in addition to the increase in time, a change in its sign is observed from negative to positive. The effect of the change in sign is visually noticeable on the corresponding graphs in Fig. 3. The effect of the extremum in such experiments is better observed in the change in the absolute execution time of the blocks (t_1, t_2) in Table 4. With a relatively constant time value for the reference block (t_2), the time of the control block (t_1) demonstrates an increase at a value of 512. In relative terms, the growth coefficient is 1.36 for 8-M and 6.18 for 9-X.

Table 4

Results of direct time measurement near point *N*=512

r	t_1, ms	t_2, ms	Δt, ms	t_1, ms	t_2, ms	Δt, ms
Equip:	1-L			2-L		
511	66.0	177.8	111.9	5.4	89.1	83.7
512	157.0	269.0	112.0	169.6	253.4	83.8
513	60.7	172.6	111.9	6.4	90.1	83.7
Equip:	3-A			4-A		
511	76.1	430.1	354.0	374.9	656.1	281.2
512	217.9	571.8	353.9	726.9	1007.4	280.4
513	72.9	426.7	353.9	377.3	657.8	280.5
Equip:	5-W			6-W		
511	15.0	783.0	768.0	–46.0	521.0	567.0
512	1992.0	2763.0	771.0	1263.0	1826.0	563.0
513	–15.0	785.0	800.0	–52.0	510.0	562.0
Equip:	7-M			8-M-A		
511	15.0	227.4	212.4	–1.98	1.57	3.55
512	365.7	559.6	193.8	–1.37	2.13	3.50
513	39.6	232.1	192.5	–1.98	1.57	3.55
Equip:	8-M-B			9-X		
511	–3.19	2.51	5.71	-0.064	0.056	0.120
512	–2.27	3.43	5.70	0.227	0.346	0.119
513	–3.19	2.52	5.71	–0.062	0.057	0.119

The power source can also affect the overall results. Thus, a comparison between graphs 8-M-A and 8-M-B shows a change in the background time of the graph from approximately –2 ns to –3 ns. However, the general appearance of the graphs does not change significantly and is not reflected in the positions of the extrema. The coefficient of increase in time in the extrema also does not change. This is most likely caused by a change in the processor clock frequency in the direction of decrease when working with a battery, which can be compared with the effect of the scale factor of the graph on the ordinate axis.

Also, from Table 4 it can be seen that the extrema have a clearly pronounced local character, that is, the magnitude of the operating time returns to the previous level, as before the increase, at the next point after the increase. In other words, the execution time of a program with a block dimension of *N*=512 may differ from that for *N*=513 by several times, and an unusual conclusion is that for a larger value of N the execution time will be smaller.

6. Discussion of the results regarding the hypothesis of the existence of exceptional sizes of data structures, which lead to an extreme increase in the time of their program processing

The implementation of the method for measuring the execution time of the program code has shown a number of

features, which are shown in Fig. 1. In Fig. 1, *b* (normalized time of a single operation), three areas with different behavior of the graph can be visually distinguished. In the first area, the time increases evenly, in the second it demonstrates unstable behavior, and in the third it reaches a constant value. The first part can be explained by the contribution of the high-speed, but small-sized processor cache. With the growth of the volume of data being processed, its contribution decreases, which leads to a gradual increase in the total code execution time. The third area demonstrates the expected behavior and does not require explanation. While the second part of the graph with the number of repetitions (jump_count) in the range of 55–90 has a number of extrema, the nature of which requires a separate study. In order to eliminate this effect, further studies were conducted outside the instability region. When using the method in practice on equipment that differs significantly from Table 1, it is recommended to conduct similar studies before performing the main measurements.

Single series of measurements of the difference time of the control and reference code blocks showed that not all series have the same behavior. The most typical situations are shown in Fig. 2 and Table 3 and can be divided into two groups: (Fig. 2, *a*) the execution time of the control block always has the same nature of deviations from the reference and (Fig. 2, *b*) the nature of the deviations is not deterministic. The same nature is manifested for equipment with DDR3 memory, while non-deterministic – for DDR5. For DDR4, both types of behavior are observed on different equipment. However, at points with extreme time increases for all series, the behavior becomes deterministic – the execution time of the control block clearly exceeds the execution time of the reference one. This is confirmed by the high linear correlation coefficient in the series at the level of 0.995–0.999.

The results of different series, grouped by type of equipment, are shown in Fig. 3 and demonstrate several facts that require more detailed discussion. The distinct effects of changing the code execution time for different block sizes, identified in [5], are partially confirmed. The main hypothesis regarding the existence of exceptional sizes of data structures that lead to an extreme increase in the time of their software processing is fully confirmed for all experiments.

The numerical values of the deviation of the code execution time with different data block sizes (1.6–3.6 times) corw relate with the deviations in the system simulation time given in [3] (by 2 times). This, firstly, can be proposed as a possible explanation of the nature of the deviations and, secondly, expands the practical significance of the obtained results on the problem of system simulation.

The appearance of extrema on the graphs of the dependence of the execution time on the block sizes was found for all types of equipment. The independence of the essence of the effect from the operating system and processor allows to suggest that the nature of the effect comes from the RAM. Of the potential causes of the phenomenon stated in [5], preference should be given to the increased frequency of RAS (Row Address Strobe) signals. The duration of RAS is 2–5 times longer than CAS (Column Address Strobe) signals, the supply of which is accompanied by each memory read [9]. The mags nitude of the difference in the duration of the RAS and CAS signals also correlates with the obtained values of the time deviation, which may indicate the correctness of the assumption.

At the same time, the role of the control bytes of the virtual address space, characteristic of the Windows operating system [5], can be considered insignificant due to the manifestation of deviation for different operating systems.

The largest number of extreme points is observed for the oldest (among the selected) technology DDR3. The smallest is for DDR4. But for DDR5, an increase in the number of extremes is again observed, however, with a smaller amplitude. This fact makes additional studies of devices with this type of memory relevant.

Another "step" effect, which consists in a smooth increase in the average "background" difference time, is observed irregularly. It is not possible to unambiguously determine the characteristics of the equipment by which this effect manifests itself or, conversely, is absent. The absence of a "step" is characteristic of DDR5, while for DDR3 and LPDDR4X this phenomenon manifests itself. For DDR4 technology, both types of graphs are observed. It can be assumed that this may be associated with the multi-channel memory operation mode and buffer sizes of different levels. However, to confirm the assumption, separate studies with appropriate selection of experimental equipment characteristics are also required. At the moment, the lack of a reliable explanation for the "step" effect can be attributed to the shortcomings of this study.

The results of direct time measurement, partially presented in Table 4, made it possible to assess the degree of influence of an unsuccessful choice of block size with an extreme size on the total code execution time. The data vary in a wide range. For the difference time of the control and reference blocks, the deviation coefficient is in the range from 1.6 to 3.6 for different equipment. For the absolute execution time of the control block, the time increase coefficient belongs to a much wider range of 1.9–132. It is not excluded that other types of computing equipment can expand the obtained limits of the coefficients when conducting similar experiments.

In any case, the algorithmic solution proposed in the work allows for practical software diagnostics of existing equipment and determining its characteristic effects and their parameters. This will allow avoiding potential deterioration of program performance by adjusting the block sizes used to feed data. Or, if it is impossible to change the software, justify hardware changes. The objectivity of the obtained data is ensured by the reliable mathematical apparatus of differential analysis with linear regression.

Also, in terms of the practical significance of the obtained results, it can be noted that the block sizes corresponding to extreme points (2^k bits) are typical for cryptography problems [10]. Elements of precisely such sizes are the main data structures in hashing or block encryption algorithms. Moreover, the algorithms themselves are often based on repeated repetition of the same type of block operations, forming cycles similar to those considered in the work. Algorithms that process streaming data (audio, video, and other streams) also have similar behavior [11]. The phenomenon of time deviation was first observed in matrix multiplication operations, which extends the practical use of the results to algorithms of the corresponding type. It is considered advisable to conduct mandatory studies on the influence of the size of the data representation structures on the running time of algorithms of this type.

The fact of the recovery of the code running time after passing extreme points can be used as a recommendation to avoid the corresponding dimensions of the structures or the sequence of their placement. For example, when iterating an array in steps multiple of 513, the increase in running time

is not observed on any of the considered equipment. Accordingly, ignoring the last (513^{th}) element will allow operating on data with a dimension of 512, but without increasing the running time. As a result, it is possible to recommend increasing the dimension of the structure that turned out to be extreme, in order to avoid an increase in the code execution time, while maintaining the proper information capacity.

The conducted studies are limited by electronic computing equipment capable of executing imperative-type software with direct access to the performer's RAM during its operation. The maximum amount of RAM requested in all experiments was 6,720,000 bits (~820 KB), the executable code had a size of about 300 KB. Although the methodology can be modified and for smaller amounts of memory, individual studies were not conducted.

A promising development of this study may be conducting experiments with specialized equipment samples. First, to increase the number of devices with the latest memory types, in particular, DDR5. Second, to focus on the availability and parameters of processor cache memory of various levels and its contribution to the resulting program runtime. If there is a need to study performers with small amounts of RAM (less than 1 MB), it will be necessary to review the numerical parameters of the methodology. The theoretical part of the experiments described in the work is considered sufficient and does not require changes.

7. Conclusions

1. A methodology for studying the operating time of prop gram code fragments has been implemented, its implementation in C++ is given. The methodology is a combination of the mathematical apparatus of differential analysis and linear regression. Applying the methodology to iteration operations of dynamic arrays of type double allowed to identify situations in which the factor of the size or step of iterations has a confirmed influence. The greatest influence was found for RAM with DDR3 and LPDDR4X technologies, while for DDR5 the influence is much less common; for DDR4 the data differ in different devices. For none of the experiments was a complete absence of deviation registered, which confirmed the hypothesis of the existence of exceptional sizes of program structures that lead to an extreme increase in their processing time.

2. Experiments conducted for different equipment have shown a different behavior of the dependence of the difference time on the data dimension. The most cases were registered when an extreme increase in time was observed when processing data with a dimension multiple of powers of two 2^k or their combinations 2^k+2^{k-1}. Most often, deviations appear for DDR3 and some DDR4 models. For DDR5, the deviation appears at sizes that are multiples of 512. For all considered technologies, extremes of the increase in operating time were found for step sizes of 512 and 1024 array elements.

3. The magnitudes of deviations vary from 1.6 to 3.6 times (for difference time) and in some experiments reach a factor of 132 for absolute time. However, the increase is local in nature and the average operating time is restored after passing the extreme point. This allowed to formulate a recommendation to avoid program structures with a dimension that is a multiple of the extreme in cycles, especially when the number of iterations approaches or exceeds 100. Adjustment by increasing the size of the structure by one unit while ignoring the last element will lead to the leveling of the detected effect. The difference in the positions of the extreme points for different equipment gives grounds to recommend conducting preliminary studies by running software on it that implements the methodology described in the work.

Conflict of interest

The authors declare that they have no conflict of interest regarding this study, including financial, personal, authorship or other, which could affect the study and its results presented in this article.

Financing

The study was conducted without financial support.

Data availability

The manuscript has no related data.

Use of artificial intelligence

The authors confirm that they did not use artificial intelligence technologies when creating the presented work.

References

1. Cheng, G., Wan, Z., Ding, W., Sun, R. (2023). Memory Allocation Strategy in Edge Programmable Logic Controllers Based on Dynamic Programming and Fixed-Size Allocation. Applied Sciences, 13 (18), 10297. https://doi.org/10.3390/app131810297
2. de Lecea, A. F., Hassan, M., Mezzetti, E., Abella, J., Cazorla, F. J. (2023). Improving Timing-Related Guarantees for Main Memory in Multicore Critical Embedded Systems. 2023 IEEE Real-Time Systems Symposium (RTSS), 265–278. https://doi.org/10.1109/rtss59052.2023.00031
3. Over, A., Strazdins, P., Clarke, B. (2005). Cycle Accurate Memory Modelling: A Case-Study in Validation. 13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 85–96. https://doi.org/10.1109/mascots.2005.22
4. Fletchery, C. W., Ren, L., Yu, X., Van Dijk, M., Khan, O., Devadas, S. (2014). Suppressing the Oblivious RAM timing channel while making information leakage and program efficiency trade-offs. 2014 IEEE 20th International Symposium on High Performance Computer Architecture (HPCA), 213–224. https://doi.org/10.1109/hpca.2014.6835932
5. Samoilenko, D. (2011). Memory tracing influence on algorithm complexity. Electrotechnic and computer systems, 4 (80), 209–212. Available at: https://eltecs.op.edu.ua/index.php/journal/article/view/907

6. Asifuzzaman, K., Verdejo, R. S., Radojković, P. (2022). Performance and Power Estimation of STT-MRAM Main Memory with Reliable System-level Simulation. ACM Transactions on Embedded Computing Systems, 21 (1), 1–25. https://doi.org/10.1145/3476838
7. Worlanyo Gbedawo, V., Agyeman Owusu, G., Komla Ankah, C., Ibrahim Daabo, M. (2023). An Overview of Computer Memory Systems and Emerging Trends. American Journal of Electrical and Computer Engineering, 7 (2), 19–26. https://doi.org/10.11648/j.ajece.20230702.11
8. Simple Linear Regression. Available at: https://www.ncl.ac.uk/webtemplate/ask-assets/external/maths-resources/statistics/regression-and-correlation/simple-linear-regression.html
9. Mukundan, J., Hunter, H., Kim, K., Stuecheli, J., Martínez, J. F. (2013). Understanding and mitigating refresh overheads in high-density DDR4 DRAM systems. ACM SIGARCH Computer Architecture News, 41 (3), 48–59. https://doi.org/10.1145/2508148.2485927
10. Sousi, A.-L., Yehya, D., Joudi, M. (2020). AES Encryption: Study & Evaluation. Rafik Hariri University. Available at: https://www.researchgate.net/publication/346446212_AES_Encryption_Study_Evaluation
11. Laghari, A. A., Shahid, S., Yadav, R., Karim, S., Khan, A., Li, H., Shoulin, Y. (2023). The state of art and review on video streaming. Journal of High Speed Networks, 29 (3), 211–236. https://doi.org/10.3233/jhs-222087