


Научная статья
УДК 004.31(075.8)
DOI: [10.34759/trd-2022-126-19](https://doi.org/10.34759/trd-2022-126-19)

ПОДХОД К ПЛАНИРОВАНИЮ ЗАГРУЗКИ ПРОЦЕССОРОВ МУЛЬТИПРОЦЕССОРНЫХ СИСТЕМ КРИТИЧЕСКОГО НАЗНАЧЕНИЯ

Юлия Васильевна Соколова¹, Евгений Владимирович Леун²,
Павел Вячеславович Примаков³, Сергей Юрьевич Самойлов⁴
^{1,2,3,4}Научно-производственное объединение им. С.А. Лавочкина,
Химки, Московская область, Россия

¹syuv@laspacespace.ru

²leunev@laspacespace.ru

³primakovpv@laspacespace.ru

⁴khsm@laspacespace.ru

Аннотация. Статья посвящена мультипроцессорным системам. Описывается подход загрузки процессоров, который возможно применить в системах реального времени, т.е. таких систем, которые реагирует на события в среде вне системы или в рамках требуемых временных ограничений. Поэтому в этой статье предлагается метод, основанный на планировании загрузки процессоров в многопроцессорных системах, что повышает производительность мультипроцессорных систем и увеличивает коэффициент готовности.

Ключевые слова: мультипроцессорная система, планирование, загрузка, высокая готовность, назначение, метод, алгоритм, расписание

Для цитирования: Соколова Ю.В., Леун Е.В., Примаков П.В., Самойлов С.Ю.

Подход к планированию загрузки процессоров мультипроцессорных систем критического назначения // Труды МАИ. 2022. № 126. DOI: [10.34759/trd-2022-126-19](https://doi.org/10.34759/trd-2022-126-19)

Original article

APPROACH TO PLANNING THE LOAD OF PROCESSORS OF CRITICAL MULTIPROCESSOR SYSTEMS

Yulia V. Sokolova¹, **Evgeny V. Leun²**, **Pavel V. Primakov³**, **Sergei Yu. Samoilov⁴**

^{1,2,3,4}Lavochkin Research and Production Association,

Khimki, Moscow region, Russia

¹syuv@laspacespace.ru

²leunev@laspacespace.ru

³primakovpv@laspacespace.ru

⁴khsm@laspacespace.ru

Abstract. One of the priority areas in economically developed countries is the security of critically important, especially complexly organized systems. Examples of such systems are a variety of robotic production; nuclear power plant control systems; onboard computing systems; groups of unmanned aerial, land and water robots; global navigation satellite system (GLONASS); large software systems of high importance and many other systems.

The main problem of building computing systems at all times remains the task of ensuring their long-term functioning. This task has three components: reliability, availability and serviceability.

Of particular relevance is the use of computer systems for managing critical objects operating in real time.

The main difference between real-time operating systems and any other operating systems is the guarantees for the start or end time of processes that are provided by real-time systems.

In the event of a failure, such systems are subject to high requirements for operability, non-failure operation, safety, security, etc. Obviously, the most important thing is to minimize the time and hardware costs required for the response of a multiprocessor system to an emergency situation.

One of the options for solving this issue may be planning the load of processors in multiprocessor systems. In this case, you can avoid simultaneous loading of several processors by one task (program, subroutine, algorithm, file, etc.) and, at the same time, schedule the queue of incoming tasks in such a way that they are served simultaneously. This allows you to reduce unplanned downtime and at the same time increase its availability along with increased speed.

The article is devoted to multiprocessor systems. The issue of compiling a plan for loading processors in them is touched upon. It is supposed to use the so-called real-time systems.

Keywords: multiprocessor system, scheduling, loading, high availability, assignment, method, algorithm, schedule

For citation: Sokolova Yu.V., Leun E.V., Primakov P.V., SamoiloV S.Yu. Approach to planning the load of processors of critical multiprocessor systems. *Trudy MAI*, 2022, no. 126. DOI: [10.34759/trd-2022-126-19](https://doi.org/10.34759/trd-2022-126-19)

Введение

Одной из наиболее важных тем в развивающихся странах является безопасность критически важных систем, особенно сложноорганизованных систем. Примером таких систем является роботизированные производства, системы управления атомными электростанциями; бортовые вычислительные системы; группировки беспилотных летательных, сухопутных и водных роботов; глобальная навигационная спутниковая система (ГЛОНАСС); крупные программные комплексы высокой значимости и многие другие системы.

Главной проблемой при построении вычислительных систем всегда была задача обеспечения ее долгосрочной жизнеспособности, которая имеет три основных составляющих, а именно: надежность, готовность и удобство обслуживания.

Повысить надежность возможно на основе принципа предотвращения неисправностей, снижая интенсивность отказов и сбоев применяя электронные схемы и компоненты с высокой и сверхвысокой степенью интеграции, а также снижая уровень помех, облегчая режим работы схемы, обеспечивая тепловой режим их работы и совершенствуя методы сборки аппаратуры. Единицей измерения надежности является среднее время наработки на отказ.

Снижение времени простоя системы (повышение готовности) предполагает подавление отказов и сбоев при работе системы, с помощью корректировки ошибок, а также средств автоматического восстановления вычислительного процесса после сбоев, включая резервное копирование процессоров и резервное программное обеспечение. На этой основе реализуются различные типы отказоустойчивых архитектур.

Использование вычислительных систем особенно актуально для управления критически важными объектами в режиме реального времени.

Основное различие между операционной системой реального времени и любой другой операционной системой заключается в гарантии на время начала или завершения процесса, предоставляемой системой реального времени.

Различают системами жесткого и мягкого реального времени. Только системы жесткого реального времени предоставляют гарантию обслуживания. Системы мягкого реального времени могут только гарантировать, что у критических задач будет максимум ресурсов, но они не могут гарантировать, что задачи будут выполнены в течение заданного периода времени.

На практике очень сложно обеспечить жесткое реальное время. Даже если процессорное время может быть запланировано, то при поступлении новых задач невозможно обеспечить бесконфликтное использование других вычислительных ресурсов компьютера.

Однако самым важным для операционной системы реального времени является планирование загрузки процессоров. Для других ресурсов гарантируется,

что конфликты между ними невозможны в силу специфики решаемых задач, либо поддерживается избыток критических ресурсов (например, память).

В связи с этим, в статье предлагаются подход к планированию загрузки процессоров в системах критического назначения, который обеспечивает увеличение производительности в многопроцессорных системах и повышает уровень коэффициента готовности.

Постановка задачи

Мультипроцессорная система [1-3], состоящая из P процессоров, производит вычисления $X = \{x_1, x_2, \dots, x_j, \dots, x_n\}$ заданий x , по определенному алгоритму описываемого вектором сложности $W = \|w_i\|_n$, где $N = n = |x|$, $i \in X$. Выполняемые задачи i буферизуются в очереди $\leq Q$, где Q – суммарный объем заданий (выраженный в битах, байтах и т.д.). Очередь может заполняться задачами в произвольный момент времени. Задания передаются на обработку процессором в момент времени $r_i \geq 0$ и инициализируются значениями w_i .

Мультипроцессорная система состоит из процессоров, обладающих одинаковыми характеристиками (тактовыми частотами, производительностью и архитектурой). Процессоры не имеют приоритетов и равнозначны, т.е. не прерываются на выполнение других задач [4-5]. Данная система определяется множеством $Pr = \{P_1, P_2, \dots, P_r, \dots, P_p\}$, где P_β – процессоры мультипроцессорной системы, причем $(P_\beta = \overline{1, \lambda})$ [6-7].

Для каждой задачи $x_i \in X$ должен быть определен момент начала выполнения S_j такой, что $S_j \geq r_j$ и такой что $S_j + w_i \leq S_j$, где S_i, S_j – моменты начала выполнения

других заданий, причем $i, j \in X, i \neq j$. Поиск расписания загрузки процессоров, аналитически может быть описан следующим отображением [8-9]:

$$\alpha_q = \left\{ x_{q_1}, x_{q_2}, \dots, x_{q_n} \right\} \rightarrow \left\{ p_{11}, p_{12}, \dots, p_{1n} \right\}, \quad (1)$$

где $q=1, \dots, n$, символ « \rightarrow » это операция планирования заданий множества X на соответствующее множество процессоров Pr .

С учётом [2,3]:

$$\sum_{j=1}^n r_j \rightarrow \min \quad (2)$$

фрагменты программ предлагается параллельно распределять на процессоры мультипроцессорной системы для их независимого исполнения без учета других операторов.

Предлагаемый метод может быть разделен на несколько шагов:

1. Задание матрицы $PrX = \|p_{\beta, x_j}\|$, где $X = \{x_1, x_2, \dots, x_j, \dots, x_n\}$ – множество заданий требующих составления плана загрузки процессоров, а $Pr = \{P_1, P_2, \dots, P_{\beta}, \dots, P_{\lambda}\}$ – множество процессоров мультипроцессорной системы.

2. Поиск в строке и столбце $Pr X_{\beta\alpha}$ минимального элемента, с последующим вычитаем его из прочих элементов матрицы.

3. Поиск нуля в строке $Pr X_{\beta\alpha}$ нуля и вычеркивание остальных нулей в строке матрицы. После чего поиск нуля производится в столбце $Pr X_{\beta\alpha}$ и так же вычеркиваются остальные нули в столбце и т.д. для всех элементов $Pr X_{\beta\alpha}$.

4. Осуществляется проведение минимального количества горизонтальных и вертикальных пересечений через отмеченные нули и поиск минимума среди не зачеркнутых прямыми чисел и вычитание его из этих чисел.

5. На пересечении прямых осуществляется прибавление минимального числа.

6. Пока не будет найден ноль, в $Pr X_{\beta\alpha}$ осуществляем повторение п. 2-5.

7. Если $Pr X_{\beta\alpha} = 0$, то на процессор j назначается подпрограмма i .

8. Осуществляется повторение п. 2–7 пока не будут найдены остальные назначения подпрограмм на процессоры мультипроцессорной системы до выставления признака готовности $Exec$.

9. Проводится анализ битов доступности поля $Work$ признака $Exec$. При $Work = 1$, то читается номер свободного процессора и п.2, в противном случае осуществляется поиск нуля в столбце $Pr X_{\beta\alpha}$ и вычеркивание остальных нулей в столбце.

Результаты и их обсуждение

На основе предложенного метода можно уменьшить время планирования параллельного выполнения подпрограмм между процессами в многопроцессорных системах, был разработан алгоритм планирования загрузки в многопроцессорных системах, состоящий из следующих этапов [10-11].

1. Задается матрица заданий $Pr X_{\beta\alpha}$, где $\alpha = \overline{1, P}$ – количество заданий мультипроцессорной системы, а $\beta = \overline{1, P}$ – количество процессоров.

2. Осуществляется поиск в строке $\min_j^P \text{Pr } X_{\beta\alpha}$ с последующим вычитанием этого

значения из всех элементов строки и поиск в столбце $\min_i^P \text{Pr } X_{\beta,\alpha}$ и дальнейшим вычитанием этого значения из всех элементов столбца.

3. Для всех строк $\text{Pr } X_{\beta\alpha}$ произвести поиск минимального элемента и его запоминание. Если есть еще нули в строке, то вычеркнуть их.

4. Для всех столбцов $\text{Pr } X_{\beta\alpha}$ поиск нуля и его запоминание. Если есть еще нули в столбце, то вычеркнуть их.

5. Если в $\text{Pr } X_{\beta\alpha}$ не обнаружено нулей, то выделить минимальное количество горизонтальных и вертикальных пересечений через отмеченные нули и перейти на п.6, иначе п. 2 - 4.

6. Осуществляется поиск среди не зачеркнутых прямыми чисел минимального значения и вычитание его из этих чисел и дальнейшее прибавление минимального числа к числам, стоящим на пересечении этих прямых.

7. Пока в $\text{Pr } X_{\beta\alpha}$ не появится хотя бы один ноль, осуществлять повторение п. 2-6.

8. Если $\text{Pr } X_{\beta\alpha} = 0$, то на процессор j назначается подпрограмма i .

9. Осуществлять повторение п. 2–8 пока не будут найдены остальные назначения подпрограмм на процессоры мультипроцессорной системы.

На первом этапе задается исходная матрица $\text{Pr } X_{\beta\alpha}$, где α – строки, задающие множество заданий, для которых необходимо найти расписание загрузки процессоров, а столбцы – множество процессоров, соответственно.

На втором этапе в каждой строке исходной матрицы происходит поиск минимального значения и выполнение вычитание найденного значения из всех элементов соответствующей строки.

Далее осуществляется поиск минимального значения в строке и его запоминание. При появлении в строке нулей, производится их вычеркивание.

Если количество нулей равно начальному количеству подпрограмм, то поиск завершен. Строка обозначает номер подпрограммы, а столбец – процессор, на который назначается эта подпрограмма.

Если количество нулей не равно начальному количеству подпрограмм, то необходимо выполнить п. 5 и п. 6. Т.е. провести минимальное число прямых, которые проходят через все нули таблицы и выполнить поиск минимального значения, который не проходит ни через одну прямую. Потом производится вычитание найденного минимума из всех чисел, через которые не проходит ни одна прямая, и его суммирование со всеми значениями, лежащими на пересечении этих двух прямых. Если после выполнения п. 6 не появилось количество подпрограмм равно количеству нулей, то необходимо повторить п. 2–7 пока не выполнится данное условие.

Дальнейшим направлением исследования является разработка функциональной схемы загрузки процессоров и исследование его загруженности. Ниже предложена планируемая структурная схема соответствующего устройства (рис. 1) [19,20].

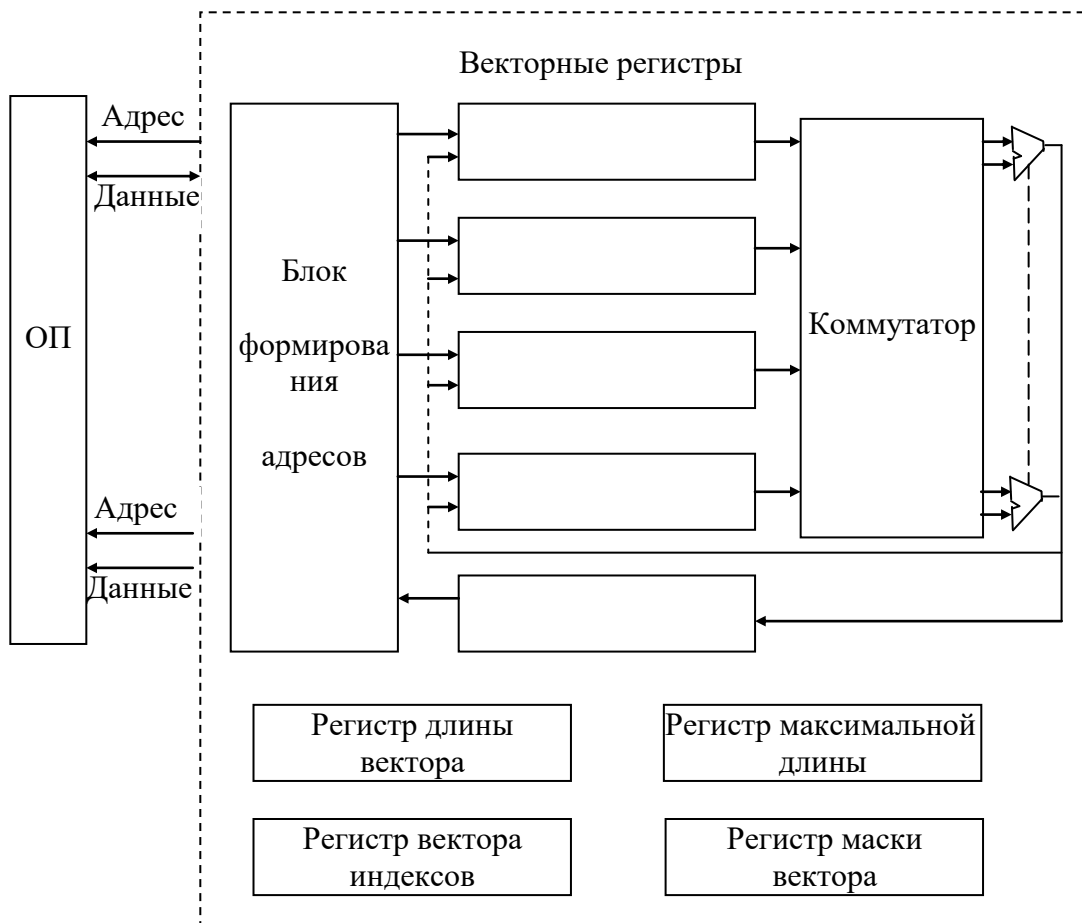


Рисунок 1 - Структурная схема устройства планирования загрузки процессоров

Обработка всех k компонентов (вектора-операндов) задается векторной командой. (рис. 1). Как правило, АЛУ состоит из отдельных блоков сумматора и умножителя. Например, иногда блок для вычисления обратной величины операции деления $\frac{X}{Y}$ реализуется в виде $X\left(\frac{1}{Y}\right)$. Каждый из таких блоков также конвейеризирован. Кроме того, состав векторной вычислительной системы обычно дополняют скалярным сопроцессором, который позволяет параллельно выполнять векторные и скалярные команды.

При использовании векторного процессора должны выполняются следующие шаги:

- чтение инструкции,

- декодирование инструкции,
- получение N чисел,
- их сложение и умножение,
- сохранение результата [12-18].

Приведённый выше алгоритм, демонстрирует что скорость выполнения увеличилась примерно в 2 раза, что подтверждает преимущество использования предлагаемого векторного процессора.

Следовательно, с помощью предложенного алгоритма и средств планирования загрузки процессоров, можно ускорить решение задач планирования нагрузки, уменьшить общую коммуникационную задержку и повысить общую производительность вычислительных систем в системах критического назначения.

Список источников

Список источников

1. Цилькер Б.Я. Организация ЭВМ и систем. - СПб.: Питер, 2007. – 668 с.
2. Оре О. Теория графов. - М.: Наука, 1968. - 352 с.
3. Кормен Т.М. и др. Алгоритмы: построение и анализ. Алгоритмы для работы с графами. Ч. VI. – М.: Издательский дом "Вильямс", 2006. – 1296 с.
4. Stallings W. Computer organization and architecture, Prentice-Hall, 1999, 881 p.
5. Левин И.И., Штейнберг Б.Я. Сравнительный анализ эффективности параллельных программ для различных архитектур параллельных ЭВМ // Искусственный интеллект. 2001. № 3. С. 234-242.

6. Слюсар В. Многоядерная архитектура: проблемные аспекты // Электроника: Наука, технология, бизнес. 2007. № 1 (75). С. 92-97
7. John L. Hennessy, David A. Patterson. Computer Architecture: A Quantitative Approach, Morgan Kaufmann, 2003, 883 p.
8. Микушин А.В., Сажнев А.М., Сединин В.И. Цифровые устройства и микропроцессоры. - СПб.: БХВ-Петербург, 2010. – 832 с.
9. Бусурин В.И., Медведев В.М., Карабицкий А.С., Гроппа Д.В. Алгоритмы анализа цифровой информации для оптимизации контроля систем управления // Труды МАИ. 2017. № 97. URL: <https://trudymai.ru/published.php?ID=87277>
10. Басов Р.Г., Борзов Д.Б. Распараллеливание загрузки процессоров в мультипроцессорных системах // XIV Международная научно-техническая конференция «Оптико-электронные приборы и устройства в системах распознавания образов, обработки изображений и символьной информации» (Курск, 16-19 мая 2017): сборник материалов. – Курск: Юго-Западный государственный университет, 2018. С. 64-66.
11. Басов Р.Г., Борзов Д.Б., Локтионова О.Г. Программа моделирования загрузки мультипроцессорной системы // Свидетельство о государственной регистрации программ на ЭВМ №2019616927 РФ, 30.05.2019.
12. Трахтенгерц Э.А. Введение в теорию анализа и распараллеливания программ ЭВМ в процессе трансляции - М.: Наука, 1981. - 254 с.
13. Tyutlyaeva E.O., Konyukhov S.S., Odintsov I.O., Moskovsky A.A. Seismic Processing Performance Analysis on Different Hardware Environment // Seismic

Processing Performance Analysis on Different Hardware Environment, 2017, vol. 4, no. 3, pp. 80 - 90. DOI: [10.14529/jsfi170305](https://doi.org/10.14529/jsfi170305)

14. Хокни Р., Джессхоуп К. Параллельные ЭВМ. Архитектура, программирование и алгоритмы. - М.: Радио и связь, 1986. - 392 с.

15. Каляев А.В., Левин И.И. Модульно-наращиваемые многопроцессорные системы со структурно-процедурной организацией вычислений: монография. - М.: Янус-К, 2003. - 379 с.

16. Борзов Д.Б., Чернецкая И.Е. Проектирование процессора ЭВМ. – Курск: Юго-Западный государственный университет, 2020. – 199 с.

17. Momose S. SX-Aurora TSUBASA. Brand-new Vector Supercomputer // SC'17 Supercomputer Forum, 2017. URL: <https://www.osp.ru/os/2018/01/13053934>

18. Кузьминский М. Из ускорителей в процессоры // Открытые системы СУБД. 2016. № 3. С. 4–6. URL: <https://www.osp.ru/os/2016/03/13050252>

19. Маркарян А.О., Чурков И.С. Задачи управления в системе принятия решений при отказах автоматизированных рабочих мест // Труды МАИ. 2020. № 113. URL: <http://trudymai.ru/published.php?ID=118150>. DOI: [10.34759/trd-2020-113-10](https://doi.org/10.34759/trd-2020-113-10)

20. Набатов А.Н., Веденяпин И.Э., Мухтаров А.Р. Применение онтологического подхода к процессу проектирования информационной системы // Труды МАИ. 2018. № 102. URL: <http://trudymai.ru/published.php?ID=99177>

References

1. Tsil'ker B.Ya. *Organizatsiya EVM i system* (Organization of computers and systems), Saint Petersburg, Piter, 2007, 668 p.

2. Ore O. *Teoriya grafov* (Theory of Graphs), Moscow, Nauka, 1968, 352 p.
3. Kormen T.M. et al. *Algoritmy: postroyeniye i analiz. Algoritmy dlya raboty s grafami. Ch. VI.* (Algorithms: construction and analysis. Algorithms for working with graphs), Moscow, Izdatel'skii dom "Vil'yams", 2006, 1296 p.
4. Stallings W. *Computer organization and architecture*, Prentice-Hall, 1999, 881 p.
5. Levin I.I., Shteinberg B.Ya. *Iskusstvennyi intellekt*, 2001, no. 3, pp. 234-242.
6. Slyusar V. *Elektronika: Nauka, tekhnologiya, biznes*, 2007, no. 1 (75), pp. 92-97.
7. John L. Hennessy, David A. Patterson. *Computer Architecture: A Quantitative Approach*, Morgan Kaufmann, 2003, 883 p.
8. Mikushin A.V., Sazhnev A.M., Sedinin V.I. *Tsifrovyye ustroystva i mikroprotsessory* (Digital devices and microprocessors), Saint Petersburg, BKhV-Peterburg, 2010, 832 p.
9. Busurin V.I., Medvedev V.M., Karabitskii A.S., Groppa D.V. *Trudy MAI*, 2017, no. 97.
URL: <https://trudymai.ru/eng/published.php?ID=87277>
10. Basov R.G., Borzov D.B. *XIV Mezhdunarodnaya nauchno-tekhnicheskaya konferentsiya «Optiko-elektronnyye pribory i ustroystva v sistemakh raspoznaniya i obrabotki izobrazhenii i simvol'noi informatsii»: sbornik materialov*, Kursk, Yugo-Zapadnyi gosudarstvennyi universitet, 2018, pp. 64-66.
11. Basov R.G., Borzov D.B., Loktionova O.G. *Svidetel'stvo o gosudarstvennoi registratsii programm na EVM №2019616927 RF*, 30.05.2019.
12. Trakhtengerts E.A. *Vvedeniye v teoriyu analiza i rasparallelivaniya programm EVM v protsesse translyatsii* (Introduction to the theory of analysis and parallelization of computer programs in the translation process), Moscow, Nauka, 1981, 254 p.

13. Tyutlyaeva E.O., Konyukhov S.S., Odintsov I.O., Moskovsky A.A. Seismic Processing Performance Analysis on Different Hardware Environment, *Seismic Processing Performance Analysis on Different Hardware Environment*, 2017, vol. 4, no. 3, pp. 80 - 90. DOI: [10.14529/jsfi170305](https://doi.org/10.14529/jsfi170305)
14. Khokni R., Dzheskhoup K. *Parallel'nye EVM. Arkhitektura, programmirovaniye i algoritmy* (Parallel computers. Architecture, programming and algorithms), Moscow, Radio i svyaz', 1986, 392 p.
15. Kalyaev A.B., Levin I.I. *Modul'no-narashchivaemye mnogoprotsessornye sistemy so strukturno-protsedurnoi organizatsiei vychislenii* (Modularly expandable multiprocessor systems with structural and procedural organization of computations), Moscow, Yanus-K, 2003, 379 p.
16. Borzov D.B., Chernetskaya I.E. *Proektirovaniye protsessora EVM* (Computer processor design), Kursk, Yugo-Zapadnyi gosudarstvennyi universitet, 2020, 199 p.
17. Momose S. SX-Aurora TSUBASA. Brand-new Vector Supercomputer, *SC'17 Supercomputer Forum*, 2017. URL: <https://www.osp.ru/os/2018/01/13053934>
18. Kuz'minskii M. *Otkrytye sistemy SUBD*, 2016, no. 3, pp. 4–6. URL: <https://www.osp.ru/os/2016/03/13050252>
19. Markaryan A.O., Churkov I.S. *Trudy MAI*, 2020, no. 113. URL: <http://trudymai.ru/eng/published.php?ID=118150>. DOI: [10.34759/trd-2020-113-10](https://doi.org/10.34759/trd-2020-113-10)
20. Nabatov A.N., Vedenyapin I.E., Mukhtarov A.R. *Trudy MAI*, 2018, no. 102. URL: <http://trudymai.ru/eng/published.php?ID=99177>

Статья поступила в редакцию 25.08.2022

Статья после доработки 10.09.2022

Одобрена после рецензирования 02.10.2022

Принята к публикации 12.10.2022

The article was submitted on 25.08.2022; approved after reviewing on 02.10.2022;
accepted for publication on 12.10.2022