Performance Evaluation 69 (2012) 267–273

Contents lists available at SciVerse ScienceDirect

Performance Evaluation

journal homepage: www.elsevier.com/locate/peva

An information-theoretic approach to estimate the capacity of processing units

Boris Ryabko *

Siberian State University of Telecommunications and Information Sciences and Institute of Computational Technology of Siberian Branch of Russian Academy of Science, Russian Federation

ARTICLE INFO

Article history:
Received 4 September 2010
Received in revised form 12 February 2012
Accepted 29 February 2012
Available online 10 March 2012

Keywords:
Processor capacity
Processor instructions
Processor memory
Entropy
Information Theory

ABSTRACT

We suggest a concept of processor capacity and entropy efficiency, which give a possibility to compare processors with different sets of instructions, different kinds of memory and different numbers of cores. A simple method of estimation of the processor capacity is presented. The suggested methods can be generalized to devices similar to computer processors (like processors of mobile telephones, etc.)

1. Introduction

We consider a processor with a certain set of instructions and several kinds of memory and address the problem of how its capacity can be estimated theoretically. More precisely, we consider the following questions: What is the processor capacity, if we know the execution time of each instruction and the speed of each kind of memory? What is the processor efficiency if the processor is used for solving problems of a certain kind (say, matrix multiplications)? To the best of our knowledge, computer science does not give answers to these questions. In fact, nowadays the efficiency of different processors can be compared only experimentally based on benchmarks.

In this paper we suggest concepts of processor capacity and entropy efficiency, which give a possibility to compare theoretically different processors. It is natural to use estimations of the processor capacity at the design stage. The paper contains some examples of such estimations that are intended to illustrate some possibilities of the suggested approach. In particular, it gives a possibility to estimate the influence on the processor capacity of such parameters as speed and size of memory, number of cores, number of instructions, etc.

The suggested approach is based on the concept of Shannon entropy, the capacity of a discrete noiseless channel and some other ideas of Information Theory.

2. Capacity

2.1. The main concept and definitions

First we briefly describe the main point of the suggested approach and give definitions. For a start, we will consider the simplified variant of a processor, which consists of a set of instructions I and an accessible memory M. It is important to

* Tel.: +7 3832061303; fax: +7 3832698203.
E-mail address: boris@ryabko.net.

0166-5316/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.peva.2012.02.002

note that any instruction $x \in I$ contains not only its name (say, JUMP), but memory addresses and indexes of registers. For example, all instructions JUMP which deal with different memory addresses are contained in I as different instructions.

A processor task P determines a certain sequence of instructions $\bar{x}(P) = x_1 x_2 x_3, \ldots, x_i \in I$. For example, if the program P contains a loop which will be executed ten times, then the sequence $\bar{x}$ will contain the body of this loop repeated ten times. It is worth noting that we do not suppose that all possible sequences of instructions are allowable. Generally speaking, there can be sequences of instructions which are forbidden. For example, it is possible that some pairs of instructions are forbidden, etc. In other words, it is possible that sequences of instructions should satisfy some limitations (or some rules). We define the set of all allowable sequences of instructions by S_C and consider two different sequences of the processor instructions from S_C as two different processor tasks. So, any task can be presented as a sequence of instructions from S_C. The opposite is also true: any sequence of instructions from S_C can be considered as a processor task. Indeed, using a so-called assembly language any sequence of instructions from S_C can be presented as a computer program, see, for example, [11].

Let us denote the execution time of an instruction x by $\tau(x)$. For the sake of simplification, we suppose that all execution times $\tau(x), x \in I$, are integers and the greatest common divisor of $\tau(x), x \in I$, equals 1. (This assumption is valid for many processors if the time unit equals a so-called clock rate and there are instructions whose execution time is one unit, i.e. $\tau(x) = 1$; see [2,3,5,7,10,11].) In this paper this assumption gives a possibility to use lim instead of lim sup, when the capacity will be defined.

Then the execution time $\tau(\bar{x})$ of a sequence of instructions $\bar{x} = x_1 x_2 x_3 \ldots x_t$ is given by

$$\tau(\bar{x}) = \sum_{i=1}^{t} \tau(x_i).$$

The key observation is as follows: the number of different tasks, whose execution time equals T, is equal to the size of the set of all sequences of instructions, whose execution time equals T, i.e.

$$\nu(T) = N(T), \tag{1}$$

where $\nu(T)$ is the number of different problems, whose execution time equals T, and

$$N(T) = |\{\bar{x} : \tau(\bar{x}) = T\}|. \tag{2}$$

Hence,

$$\log \nu(T) = \log N(T). \tag{3}$$

(Here and below T is an integer, $\log x \equiv \log_2 x$ and $|Y|$ is the number of elements of Y if Y is a set, and the length of Y if Y is a word.) In other words, the total number of tasks executed in time T is equal to (2). Based on this consideration we give the following definition.

Definition 1. Let there be a processing unit with a set of instructions I and let $\tau(x)$ be the execution time of an instruction $x \in I$. The processor capacity $C(I)$ is defined as follows:

$$C(I) = \lim_{T \to \infty} \frac{\log N(T)}{T} \tag{4}$$

bits per time unit. (Here $N(T)$ is defined in (2).)

Claim 1. *The limit (4) exists if I is finite, execution times $\tau(x), x \in I$ are integers and the greatest common divisor of $\tau(x), x \in I$, equals 1.*

Proof is given in Appendix.

For example, let there be two processing units Γ_1 and Γ_2 which have identical sets of instructions I_1 and I_2, but their execution time are different in such a way that the speed of the first processing unit is twice more than that of the second one, i.e. $\tau_1(x) = \tau_2(x)/2$ for any $x \in I$ (where τ_i corresponds to the processing unit Γ_i.) From definition (4) we immediately obtain that the capacity of the first processing unit is twice more than the second one, i.e. $C(I_1) = 2\,C(I_2)$. Apparently, this equation is quite natural.

Formal definitions are given above and now we can try to explain the main idea of the suggested approach. Let us imagine that there is a processor which can execute N different sequences of instructions during, say, one hour. Roughly speaking, this processor can execute N^2 sequences of instructions during two hours, because, if s_1 and s_2 are one-hour sequences, the combined sequence $s_1 s_2$ is a two-hour one. (We did not take into account a few extra two-hour sequences which have one instruction whose execution starts at the end of the first hour and finishes at the beginning of the second one. The point is that the Claim 1 shows that such exceptions do not change the limit behavior of logarithm of the number of sequences.) Analogously, approximately N^k sequences can be executed during k hours. So, the number of possible sequences (and solvable tasks) grows exponentially as a function of the time t, thus $\log N(t)/t$ (or the limit of this value) appears to be an adequate measure of the processor capacity. Thus, the number of possible sequences grows exponentially with the rate that is asymptotically equal to the capacity of the processor.

It is worth noting that this situation is typical for Information Theory, where, for example, the capacity of a lossless channel is defined by the rate of asymptotic growth of the number of allowed sequences of basic symbols (letters), whose length can be different (see [9]).

2.2. Methods for estimating the processor capacity

The processor capacity $C(I)$ can be estimated in different situations by different methods. In particular, a stream of instructions generated by different tasks can be described as a sequence of words created by a formal language, or the dependence between sequentially executed instructions can be modeled by Markov chains, etc. Seemingly the most general approach is to define the set of admissible sequences of instructions as a certain subset of all possible sequences. More precisely, the set of admissible sequences G is defined as a subset $G \subset A^\infty$, where A^∞ is the set of one-side infinite words over the alphabet A: $A^\infty = \{x : x = x_1x_2\ldots\}$, $x_i \in A$, $i = 1, 2, \ldots$. In this case the capacity of G is deeply connected with the topological entropy and can be calculated by methods of the theory of dynamical systems in some particular cases. We do not consider this approach in details, because it seems to be difficult to use it for solving applied problems which require a finite description of the set of admissible sequences.

The simplest estimate of the processor capacity can be obtained if we suppose that all sequences of the instructions are admissible. In other words, we consider the set of instructions I as an alphabet and suppose that all sequences of letters (instructions) can be executed. In this case the method of calculation of the lossless channel capacity, given by Shannon in [9], can be used. It is important to note that this method can be used for upper-bounding the processor capacity for all other models, because for any processor the set of admissible sequences of instructions is a subset of all words over the "alphabet" I. In this paper we use only this estimate because, on the one hand, our goal is to explain the main ideas of the approach and, on the other hand, this model is quite natural because almost all instructions of a processor can be used in any order. Moreover, this model is quite flexible and gives a possibility to take into account some special cases. For example, if there is a pair of particular instructions which can be used only together in a certain order, then they can be considered as one instruction whose execution time is the sum of times of the combined instructions.

Let, as before, there be a processor with a set of instructions I whose execution time is $\tau(x)$, $x \in I$, and all sequences of instructions are allowed. In other words, if we consider the set I as an alphabet, then all possible words over this alphabet can be considered as admissible sequences of instructions for the processor. The question we consider now is how one can calculate (or estimate) the capacity (4) for this case. The solution is suggested by Shannon [9] who showed that the capacity $C(I)$ is equal to the logarithm of the largest real solution X_0 of the following equation:

$$X^{-\tau(x_1)} + X^{-\tau(x_2)} + \cdots + X^{-\tau(x_s)} = 1, \tag{5}$$

where $I = \{x_1, \ldots, x_s\}$. In other words, $C(I) = \log X_0$. Note that the root can be calculated by a so-called bisection method which is the simplest root-finding algorithm. This method will be used in all following examples.

3. Examples and possible applications

3.1. One-core processors

As an example we briefly consider the MMIX computer suggested by Knuth [4]. The point is that its processor is described in detail and can be considered as a model of a modern processor. The MMIX computer has 256 general-purpose registers, 32 special-purpose ones and 2^{64} bytes of virtual memory [4]. The MMIX instructions are presented as 32-bit words and in this case the "processor alphabet" consists of almost 2^{32} words (almost, because some combinations of bits do not make sense). In [4] the execution (or running) time is assigned to each instruction in such a way that each instruction takes an integer number of υ, where υ is a unit that represents the clock cycle time. Besides, it is assumed that the running time depends on the number of memory references (*mems*) that a program uses. For example, it is assumed that the execution time of each of the *LOAD* instructions is $\upsilon + \mu$, where μ is an average time of memory Ref. [4]. If we consider υ as the time unit and define $\hat{\mu} = \mu/\upsilon$, we obtain from the description of MMIX [4] and (5) the following equation for finding an upper bound on the MMIX capacity:

$$2^{24}\left(\frac{139}{X} + \frac{32}{X^2} + \frac{5}{X^3} + \frac{17}{X^4} + \frac{3}{X^5} + \frac{4}{X^{10}} + \frac{2}{X^{40}} + \frac{4}{X^{60}}\right) + 2^8 2^{64}\left(\frac{46}{X^{1+\hat{\mu}}} + \frac{2}{X^{1+20\hat{\mu}}} + \frac{46}{X^{2+2\hat{\mu}}}\right) = 1. \tag{6}$$

Let us explain how this equation is obtained. There are 139 different instructions whose execution time is 1 and operands can be any 24-bit word, that is why there is the summand $2^{24}\frac{139}{X}$ in (6). Then, there are 32 instructions whose execution time is 2 and operands are 24-bit words and the summand $2^{24}\frac{32}{X^2}$ corresponds to those instructions, etc. The value $\hat{\mu}$ depends on the realization of MMIX and is larger than 1 for modern processors [4]. Now we can estimate the capacity for different $\hat{\mu}$. For example, if $\hat{\mu}$ is small (say, $\hat{\mu} = 1$), the capacity is about 35 bits per time unit (It is the clock cycle time in our case). If $\hat{\mu}$ is large ($\hat{\mu} = 5$), the capacity is around 31, 5 bits per time unit.

This examples show that the Eq. (5) can be used for estimation of the processor capacity and the capacity of processors is mainly determined by the subsets of instructions whose execution time is minimal.

It is natural to use estimations of the processor capacity at the design stage. We consider examples of such estimations that are intended to illustrate some possibilities of the suggested approach.

First we consider a processor, whose design is close to the MMIX. Suppose a designer has decided to use the MMIX set of registers. Suppose further, that he/she has a possibility to use two different kinds of memory, such that the time of one

reference to the memory and the cost of one cell are τ_1, d_1 and τ_2, d_2, correspondingly. It is natural to suppose that the total price of the memory is required not to exceed a certain bound D. As in the example with MMIX we define $\hat{\mu}_1 = \tau_1/\upsilon$, $\hat{\mu}_2 = \tau_2/\upsilon$, where, as before, υ is a unit that represents the clock cycle time.

As in the case of MMIX, we suppose that there are instructions for writing and reading information from a register to a cell. The set of these instructions coincides with the corresponding set of the MMIX. If we denote the number of the memory cells by S, then the number of the instructions which can be used for reading and writing, is proportional to S. Having taken into account that MMIX has 2^8 registers and the Eq. (6), we can see that the designer should consider two following equations

$$\left(2^{24}\left(\frac{139}{X}+\frac{32}{X^2}+\frac{5}{X^3}+\frac{17}{X^4}+\frac{3}{X^5}+\frac{4}{X^{10}}+\frac{2}{X^{40}}+\frac{4}{X^{60}}\right)\right)+\left(2^8 S_i\left(\frac{46}{X^{1+\hat{\mu}_i}}+\frac{2}{X^{1+20\hat{\mu}_i}}+\frac{46}{X^{2+2\hat{\mu}_i}}\right)\right)=1 \qquad (7)$$

for $i = 1, 2$, where $S_i = D/d_i$, i.e. S_i is the number of cells of the i-th kind of memory, $i = 1, 2$. The designer can calculate the maximal roots for each equation ($i = 1, 2$) and then he/she can choose that kind of memory for which the solution is larger. It will mean that the processor capacity will be larger for the chosen kind of memory. For example, suppose that the total price should not exceed 1 ($D = 1$), the prices of one cell of memory are $d_1 = 2^{-30}$ and $d_2 = 2^{-34}$, whereas $\hat{\mu}_1 = 1.2$, $\hat{\mu}_2 = 1.4$. The direct calculation of the Eq. (7) for $S_1 = 2^{30}$ and $S_2 = 2^{34}$ shows that the former is preferable, because the processor capacity is larger for the first kind of memory.

Obviously, this model can be generalized for different set of instructions and different kinds of memory. In such a case the considered problem can be described as follows. We suppose that there are instructions $\mu_i^w(n)$ for writing information from a special register to n-th cell of i-th kind of memory ($n = 0, \ldots, n_i - 1$, $i = 1, \ldots, k$,) and similar instructions $\mu_i^r(n)$ for reading. Moreover, it is supposed that all other instructions cannot directly read or write to the memory of those kinds, i.e. they can write to and read from the registers only. (It is worth noting that this model is quite close to some real processors.) Denote the execution time of the instructions $\mu_i^w(n)$ and $\mu_i^r(n)$ by $\dot{\tau}_i$, $1 = 1, \ldots, k$.

In order to get an upper bound of the processor capacity for the described model we, as before, consider the set of instructions as an alphabet and estimate its capacity. From (5) we obtain that the capacity is $\log X_0$, where X_0 is the largest real solution of the following equation:

$$\sum_{x\in I^*} X^{-\tau(x)} + R\left(\frac{2n_1}{X^{\dot{\tau}_1}}+\frac{2n_2}{X^{\dot{\tau}_2}}+\cdots+\frac{2n_k}{X^{\dot{\tau}_k}}\right)=1, \qquad (8)$$

where I^* contains all instructions except $\mu_i^r(n)$ and $\mu_i^w(n)$, $1 = 1, \ldots, k$, R is a number of registers. (The summand $\frac{2n_i}{X^{\dot{\tau}_i}}$ corresponds to the instructions $\mu_i^w(n)$ and $\mu_i^r(n)$.)

Let us suppose that the price of one cell of the ith kind of memory is d_i whereas the total cost of memory is limited by D. Then, from the previous equation we obtain the following optimization problem:

$$\log X_0 \longrightarrow \textit{maximum},$$

where X_0 is the maximal real solution of the Eq. (8) and

$$d_1 n_1 + d_2 n_2 + \cdots + d_k n_k \le D; \qquad n_i \ge 0, \quad i = 1, \ldots, k.$$

The solution of this problem can be found using standard methods and used by processor designers.

3.2. Multi-core processors

Let us briefly consider multi-core processing units, whose descriptions can be found in [11]. First we consider one general question. Let there be a multi-core processing unit with l, $l \ge 1$, cores. The following claim shows that, generally speaking, the capacity of this processing unit equals the sum of capacities of corresponding one-core processing units.

Claim 2. *Let there be a multi-core processing unit with l, $l \ge 1$, cores $I_1, \ldots, I_l$, such that any core can be run as an individual processor independently on other cores, i.e. any instruction of j-th core I_j can be used independently of instructions executed by other cores. Then the capacity of this multi-core processing unit is the sum of capacities of one-core processors corresponding to individual cores $I_1, \ldots, I_l$. In other words, if we consider a processing unit with 1 core I_j (i.e. with the same set of instructions and all kinds of memories of the original multi-core processor), then*

$$C\left(\bigotimes_{j=1}^{l} I_j\right) = C(I_1) + C(I_2) + \cdots + C(I_l), \qquad (9)$$

where $C(\bigotimes_{j=1}^{l} I_j)$ is the capacity of the multi-core processing unit, $C(I_j)$ is a capacity of one-core processor with the set of instructions I_j and all kinds of memory of the considered multi-core processing unit.

Proof. Having taken into account the obvious equality $N_{multi}(T) = N_1(T)\, N_2(T) \ldots N_l(T)$, we obtain (9) from the definitions (2) and (4).

In particular, the claim shows that the capacity of l-core processing unit with l identical processors should be l times more than the capacity of one processor.

Let us consider an example. Let there be a multi-core processing unit with memory, which can be used by all processors. More precisely, we suppose that there is a l-core processing unit with l identical processors. All cores have an identical set of instructions $I_j, j = 1, \ldots, l$, and several kinds of memory. For any core there are r_j registers and m_j cells of an individual memory (which can be used by j-th core only) and, besides, there are M cells of memory, which can be used by all cores (the shared memory). First we estimate the capacity of one-core processing unit and then use the Claim 2. In order to avoid insignificant details, we consider a model similar to a previous example. In this case we denote by τ_m and τ_M the time of one reference to the memory m_j ($j = 1, \ldots, l$) and M. As in the previous example we define $\hat{\mu}_m = \tau_m/\upsilon$, $\hat{\mu}_M = \tau_M/\upsilon$, where, as before, υ is a unit that represents the clock cycle time. Analogically to (8) we can present an equation for calculation of capacity for one-core processor as follows:

$$\sum_{x \in I^*} X^{-\tau(x)} + r_j \left(\frac{2m_j}{X^{\hat{\mu}_m}} + \frac{2M}{X^{\hat{\mu}_M}} \right) = 1. \tag{10}$$

It is clear that the larger m_j and M, the larger a solution X_0 of (10) and, hence, the larger capacity C_1. Taking into account that the capacity C_l of l-core processing unit equals $l\, C_1$, we can see that the impact of the size of the shared memory (M) is much larger that the impact of the individual memories $m_j, j = 1, \ldots, l$.

So, we can see that the capacity of a multi-core processing unit is proportional to the number of cores and the shared memory increases the capacity much larger than the individual memory. □

4. Entropy efficiency

A processing unit can be used for solving tasks of different kinds and it is possible that the maximal capacity cannot be achieved for a certain kind of tasks. For example, one computer can be a Web server, another can be used for solving differential equations, etc. In principle, it is possible that the processor uses only a small part of instructions when it is solving tasks of a certain kind, that is why it cannot achieve the maximal capacity. In order to model this situation we suggest the following approach: there is an information source which generates a sequence of tasks in such a way, that the processor begins to solve each next task as soon as the previous task is finished. We will not deal with a probability distribution on the sequences of the tasks, but consider sequences of processor instructions, determined by sequences of the tasks, as a stochastic processes. In what follows we will consider the model when this stochastic process is stationary and ergodic, and we will define the processor efficiency for this case.

The definition of the entropy efficiency will be based on results and ideas of information theory, which we introduce in what follows. Let there be a stationary and ergodic process $z = z_1, z_2, \ldots$ generating letters from a finite alphabet A (the definition of stationary ergodic process can be found, for ex., in [1]). The n-order Shannon entropy and the limit Shannon entropy are defined as follows:

$$h_n(z) = -\frac{1}{n+1} \sum_{u \in A^{n+1}} P_z(u) \log P_z(u),$$
$$h_\infty(z) = \lim_{n \to \infty} h_n(z) \tag{11}$$

where $n \geq 0$, $P_z(u)$ is the probability that $z_1 z_2 \ldots z_{|u|} = u$ (this limit always exists, see [1,9]). We will consider so-called i.i.d. sources. By definition, they generate independent and identically distributed random variables from some set A. Now we can define the entropy efficiency.

Definition 2. Let there be a processing unit with a set of instructions I and let $\tau(x)$ be the execution time of an instruction $x \in I$. Let this computer be used for solving such a sequence of tasks, that the corresponding sequence of the instructions $z = z_1 z_2, \ldots, z_i \in I$, is a stationary ergodic stochastic process. Then the entropy efficiency is defined as follows:

$$c(I, z) = h_\infty(z) \Big/ \sum_{x \in I} P_z(x) \tau(x), \tag{12}$$

where $P_z(x)$ is the probability that $z_1 = x$, $x \in I$.

Informally, the Shannon entropy is a quantity of information (per letter), which can be transmitted and the denominator in (12) is the average execution time of an instruction.

The entropy efficiency can be estimated basing on statistical estimations of instruction frequencies and such an estimation gives a possibility to investigate the influence of different parameters on the entropy efficiency.

It is easy to see that the entropy efficiency (12) is maximal, if the sequence of instructions $x_1 x_2, \ldots, x_i \in I$ is generated by an i.i.d. source with probabilities $p^*(x) = X_0^{-\tau(x)}$, where X_0 is the largest real solution to the Eq. (5), $x \in I$. Indeed, having

taken into account that $h_\infty(z) = h_0(z)$ for i.i.d. source [1] and the definition of entropy (11), the direct calculation of $c(I, p^*)$ in (12) shows that $c(I, p^*) = \log X_0$ and, hence, $c(I, p^*) = C(I)$.

It will be convenient to combine the results about the processor capacity and the entropy efficiency in the following statement.

Claim 3. *Let there be a processing unit with a set of instructions I and let $\tau(x)$ be the execution time of $x \in I$. Suppose that all sequences of instructions are admissible programs. Then the following equalities are valid:*

(i) *The processor capacity $C(I)$ (4) equals $\log X_0$, where X_0 is the largest real solution to the Eq. (5).*
(ii) *The entropy efficiency (12) is maximal if the sequences of instructions are generated by an i.i.d. source with probabilities $p^*(x) = X_0^{-\tau(x)}$, $x \in I$.*

So, we can see that the processor capacity is the maximal value of the entropy efficiency, and this maximum can be attained for a certain class of computer problems. This fact can be useful for applications. For example, if the entropy efficiency is much less than the processor capacity, it could mean that either a different processor would be a better choice for the considered problems or a specialized processor may be in order. In other words, information about the processor capacity and the entropy efficiency can be useful for processor designers and manufacturers.

5. Conclusion

We have suggested a definition of the processor capacity and the entropy efficiency as well as a method for their estimation. It can be suggested that this approach may be useful in the design stage when developing computers and similar devices.

It would be interesting to analyze the "evolution" of processing units from the point of view of their capacity. The preliminary analysis shows that the development of the RISC processors (see [8]), the increase in quantity of the registers and some other innovations (see [2,3,5,7,10,11]), lead to the increase of the capacity of processing units. Moreover, such methods as using cache memory can be interpreted as an attempt to increase the entropy efficiency.

Acknowledgments

The author was supported by the Russian Foundation for Basic Research (grant no. 09-07-00005).

Appendix

Proof of Claim 1. We will consider stationary subsets [6], because their capacity (or combinatorial entropy [6]) can be defined analogously (4). Let $G \subset A^\infty$, where A is an alphabet. The definition of a stationary subset G is as follows: take any word from G and cross its first letter out. If the obtained set equals G, then G is called stationary. The following proposition is due to Krichevsky [6]:

Proposition 1. *Let G be a stationary set and $G(s)$ be the set of all s-letter prefixes of all words from G, where s is an integer. Then $\lim_{s\to\infty} \log |G(s)|/s$ exists.*

The proof can be found in [6].

Let, as before, S_C be a set of all allowable sequences of instructions from I. We denote the subset of all infinite sequences of S_C by S_C^∞. Let us represent a set of all instructions as $I = \{x_1, x_2, \ldots, x_N\}$ and define a new alphabet I^* as follows: $I^* = \{x_1^1, x_1^2, \ldots, x_1^{\tau(x_1)}, x_2^1, x_2^2, \ldots, x_2^{\tau(x_2)}, \ldots, x_N^1, x_N^2, \ldots, x_N^{\tau(x_N)}\}$. We suppose that the length of all letters from I^* equals 1 and define a subset of allowable sequences S_C^* as follows: $x_{i_1}^1 x_{i_1}^2 \ldots x_{i_1}^{\tau(x_{i_1})} x_{i_2}^1 x_{i_2}^2 \ldots x_{i_2}^{\tau(x_{i_2})} \ldots \in S_C^*$ if and only if $x_{i_1} x_{i_2} \ldots \in S_C$. In words, any letter (or instructions) from I, whose length is $\tau(x)$, is presented as a sequence of $\tau(x)$ letters, whose length is 1. Having taken into account that I is finite, it is easy to see that if the $\lim_{T\to\infty} \log |S_C^*(T)|/T$ exists, then the limit (4) exists, too.

Let A be an alphabet and G is a subset of one-side infinite words over A, i.e. $G \subset A^\infty$. We define the set G^{-1} as follows: take all words from G and cross their first letters out. The obtained set of words is G^{-1} and, by definition, $G^{-s} = (G^{-(s-1)})^{-1}$, $G^0 = G$, $s > 0$. Let $\tau^* = \prod_{x\in I} \tau(x)$ and $\hat{S}_C^* = \bigcup_{j=0}^{\tau^*} (S_C^*)^{-j}$. From those definitions we immediately obtain that $\hat{S}_C^*$ is a stationary set and, according to proposition, there exists $\lim_{T\to\infty} |\log \hat{S}_C^*(T)|/T$. Taking into account the definitions of τ^* and $\hat{S}_C^*$, we can see that for any integer T

$$\log |\hat{S}_C^*(T)| \le \log |S_C^*(T)| \le \log(\tau^* |\hat{S}_C^*(T)|).$$

The $\lim_{T\to\infty} |\log \hat{S}_C^*(T)|/T$ exists, hence, $\lim_{T\to\infty} |\log(\tau^* |\hat{S}_C^*(T)|)/T$ exists. From this and last inequalities we can see that $\lim_{T\to\infty} \log |S_C^*(T)|/T$ exists. As we mentioned above, it means that the limit (4) exists, too.

The claim is proven. □

References

[1] T.M. Cover, J.A. Thomas, Elements of Information Theory, Wiley, 2006.
[2] V.C. Hamacher, Z.G. Vranesic, S.G. Zaky, Computer Organization, McGraw-Hill, 2002.
[3] J.L. Hennessy, D.A. Patterson, Computer Architecture: A Quantitative Approach, Morgan Kaufmann Publishers, 2008.
[4] D.E. Knuth, The art of computer programming, volume 1, Fascicle 1, MMIX: A RISC Computer for the New Millennium, 2005.
[5] C.E. Kozyrakis, D.A. Patterson, A new direction for computer architecture research, Computer 31 (11) (1998) 24–32.
[6] R. Krichevsky, Universal Compression and Retrieval, Kluver Academic Publishers, 1993.
[7] D.A. Patterson, J.L. Hennessy, Computer Organization and Design: The Hardware/Software Interface, Morgan Kaufmann, 2004.
[8] C.H. Sequin, D.A Patterson, Design and Implementation of RISC I, University of California, Berkeley, 1982.
[9] C.E. Shannon, A mathematical theory of communication, Bell Sys. Tech. J. 27 (1948) 379–423. pp. 623–656.
[10] W. Stallings, Computer Organization and Architecture: Designing for Performance, Prentice-Hall, 2009.
[11] A.S. Tanenbaum, Structured Computer Organization, Prentice Hall PTR, 2005.

Boris Ryabko received the M.S. degree from Novosibirsk state university in 1971, Ph.D. degree from Institute of Mathematics, Novosibirsk in 1981 and Dr.Sci. degree from Inst. of Problems of Information Transmission, Moscow, 1989. He has been Professor of applied mathematics and computer science since 1986. Now he is with the Siberian State University of Telecommunication and Information Sciences. His research interests include Applied Mathematics, Information and Coding Theory, Cryptography and Mathematical Biology. He published more than 150 scientific articles and 6 books in these fields.