A deterministic mathematical model of the transmission of information II

By E. GESZTELYI (Debrecen)

Dedicated to Professor B. Barna on his 80th birthday

Introduction

This paper is the immediate continuation of the paper [2]. We showed there that the storage and the transmission of information are two aspects of the one and the same thing. Any transmission of information can be thought of as a storage of information in the channel and any storage of information in a memory goes together with some transmissions of information. Namely, both the reading of the content of a memory and the writing of some content in a memory is a move of information. Thus, the well known Shannon's scheme of the transmission of information (see Fig.1. of [2]) can be explained as a writing of the information in the channel and as a reading of information from the channel.

In the antiquity, the transmission of information was realized by means of messengers or later sometimes by carrier pigeons or more later by mail-carriage. The unique possibility was a writing on the part of the source and obtaining the writing from the carrier of information, a reading on the part of destination. Originally, Shannon's scheme of transmission of information referred to the telegraph. Clearly, also the telegraphy is connected with certain writings and readings. Moreover, in the inside of a computer or between computers the transmission of information goes together with a writing and reading. But in this case not persons but processors do the reading automatically such that the content of a memory (or memory cell) will be copied in another memory (or memory cell). We shall give mathematical descriptions for this situations and it will be shown in the next chapters.

1. An example for a memory over a set

First of all, we remind the abstract notion of the memory introduced previously in [1] and [2]. By a memory over a set $\mathbf{S}$, we mean a triple $\mathbf{M}=(\mathbf{S}, \mathbf{C}, s)$ where $\mathbf{S}$ is a nonvoid set, $\mathbf{C}$ is a set of some partitions of $\mathbf{S}$, and $s \in \mathbf{S}$ is a variable on $\mathbf{S}$, $\mathbf{S}$ is called the set of possible states of $\mathbf{M}$, the current value of s is regarded as the instant state of $\mathbf{M}$. $\mathbf{C}$ is called the core of $\mathbf{M}$ and any partition $P \in \mathbf{C}$ is said to be a memory cell of $\mathbf{M}$. We suppose that the blocks of all partitions $P \in \mathbf{C}$ are marked with the elements of

some set Γ_P. The content of the cell P will be an element of Γ_P depending on the instant state s. Namely by the content of P at state s, we mean the index of that block of P which contains s. Thus, if $P=\{B_k|k\in\Gamma_P\}$ and $(P)_s$ denotes the content of P at state s then

$$(P)_s = k \Leftrightarrow s\in B_k. \tag{1.1}$$

If P is finite then we can take $\Gamma_P=\{0, 1, \ldots, |P|-1\}$ and so, we can store numbers in the memory cell P.

The memory **M** is said to be finite if **C** is a finite set of finite partitions of **S**, while the state set **S** does not need to be finite.

Example 1. We give the following finite memory $\mathbf{M}=(\mathbf{S}, \mathbf{C}, s)$ where the set **S** is a lower segment of nonnegative hexadecimal numbers

$$\mathbf{S} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$$

and $\mathbf{C}=\{A, B\}$ where

$$A = \{\{0, 1, 2, 3, \}_0, \{4, 5, 6, 7\}_1, \{8, 9, A, B\}_2, \{C, D, E, F\}_3\}$$

and

$$B = \{\{0, 4, 6, C\}_0, \{1, 5, 9, D\}_1, \{2, 6, A, E\}_2, \{3, 7, B, F\}_3\}.$$

The core of the memory is shown on Fig. 1.

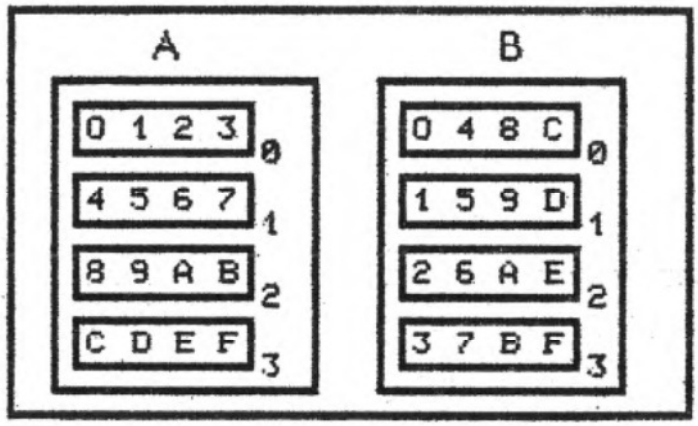


Fig. 1

We see that $|A|=|B|=4$ and it is easy to check that $A\cdot B$ is the finest partition of **S** and so $|A\cdot B|=|\mathbf{S}|=16=|A|\cdot|B|$. This shows that **C** is independent, the size of **M** is 16 and so the storing capacity of **M** is $\text{cap}\,(\mathbf{M})=\log_2 16=4$ bits.

2. Processors

If we define a memory on the state set of an automaton then we get a model of a device called processor in the practice. As mentioned in the introduction, the processors are devices, which are able to the automatical reading the content of a memory and the automatical writing in a memory. An automaton is a device which can change the instant state by influence of inputs only according to a given rule described by a function. Now if an automaton and a memory have a common state set then the content of the memory can be changed by entering the input signals in the automaton making possible the automatical reading and writing in the memory.

Definition 2.0. A tuple $\mathscr{P}=(\mathbf{S}, \mathbf{X}, \delta, \mathbf{C}, s)$ is said to be a processor if $\mathbf{A}=(\mathbf{S}, \mathbf{X}, \delta)$ is an automaton (with state set $\mathbf{S}$, input alphabet $\mathbf{X}$, transition function δ), and $\mathbf{M}=(\mathbf{S}, \mathbf{C}, s)$ is a memory on $\mathbf{S}$ (with core $\mathbf{C}$ and instant state s). The instant state of $\mathbf{M}$ is regarded as the instant state of both the automaton A and the memory $\mathbf{M}$. □

The elements of the input alphabet of the automaton of a processor $\mathscr{P}$ are called instructions for $\mathscr{P}$. A sequence $p=x_1 \ldots x_k \in \mathbf{X}^+$ of instructions is called an external program for the processor. It is tacitly supposed the existence of a power which puts the new instant state $s'=\delta(s, x)$ in force instead of the old state s when the instruction x is given as an input for the automaton of the processor $\mathscr{P}$ being in state s. In practice, such powers are the clock generators and drivers for processors. But, we do not deal with the modelling of these devices. Similarly, we shall avoid the questions of the refreshing of memories ([4]).

Thus, the execution of a command is identical with the setting of a new instante state. This calls forth certain alterations in the content of the memory. So, a processor is a device which operates on the content of its memory cells, in full accordance with the determination of processors in systems programming [5].

The processor $\mathscr{P}=(\mathbf{S}, \mathbf{X}, \delta, \mathbf{C}, s)$ can be written symbolically as a couple $\mathscr{P}=(\mathbf{A}, \mathbf{M})$ where $\mathbf{A}=(\mathbf{S}, \mathbf{X}, \delta)$ is the automaton of the processor and $\mathbf{M}=(\mathbf{S}, \mathbf{C}, s)$ is the memory of $\mathscr{P}$. We say in this case that the processor is an automaton $\mathbf{A}$ provided with a memory. Sometimes we regard a processor as a memory provided with an automaton and we shall express this by writing $\mathscr{P}=(\mathbf{M}, \mathbf{A})$.

Next, we give an example for a processor which has the memory given in example 1.

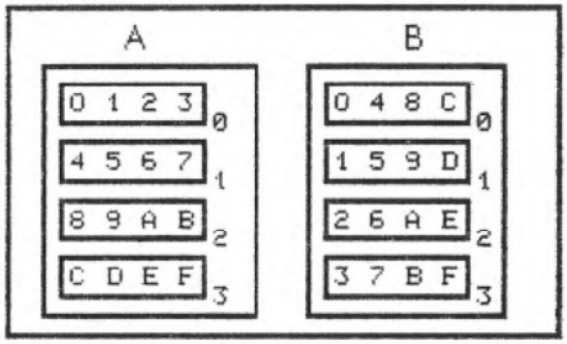
Example 2. Let $\mathscr{P}$ be the processor $\mathscr{P}=(\mathbf{S}, \mathbf{X}, \delta, \mathbf{C}, s)$ where

$$\mathbf{S} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$$

$$\mathbf{X} = \{x_0, x_1, x_2, y_0, y_1, y_2, y_3, z\}.$$

The core $\mathbf{C}$ is identical with the core of the memory given in example 1 (see Fig. 1. and 2.). The state transition function is given by the table shown on Fig. 2.

As mentioned above, if we give the instruction $x \in \mathbf{X}$ for the processor $\mathscr{P}$ which is in state s then this is equivalent to the entering x as an input for the automaton $\mathbf{A}=(\mathbf{S}, \mathbf{X}, \delta)$ which is in state s also. So, the new instant state $s'=\delta(s, x)$ will be in force. But, this alteration of the instant state causes an alteration in the content of some memory cells. This is called the meaning of the instruction.



← TABLE OF THE δ FUNCTION →																	THE MEANING OF INSTRUCTIONS expressed	
INSTRUCTIONS = ELEMENTS OF INPUT SET X	ELEMENTS OF S=STATE SET OF A																by Z80 mnemonics	in BASIC language
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F		
x_0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	NOP	PAUSE
x_1	0	5	A	F	0	5	A	F	0	5	A	F	0	5	A	F	LD A,B	LET A=B
x_2	0	0	0	0	5	5	5	5	A	A	A	A	F	F	F	F	LD B,A	LET B=A
y_0	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3	LD A,0	LET A=0
y_1	4	5	6	7	4	5	6	7	4	5	6	7	4	5	6	7	LD A,1	LET A=1
y_2	8	9	A	B	8	9	A	B	8	9	A	B	8	9	A	B	LD A,2	LET A=2
y_3	C	D	E	F	C	D	E	F	C	D	E	F	C	D	E	F	LD A,3	LET A=3
z	0	5	A	F	4	9	F	3	8	D	2	7	C	1	6	B	ADD A,B	LET A=A+B

Fig. 2

We indicated on Fig. 2. what do the instructions. For the sake of shortness we expressed the meaning of instructions in both *Z*80 assembly language and BASIC language.

Thus for example, we can easy check that on the effect of the instruction x_2, the cell B will obtain the content of the cell A independently of the previous content of B, while the content of A remains unchanged:

$$\forall s \in \mathbf{S}\colon (B)_{sx2} = (A)_s \wedge (A)_{sx2} = (A)_s \tag{2.1}$$

where $(A)_s$ is the content of A at state s (see (1.1)) and we used the short algebraic notation $sx = \delta(s, x)$.

We can formulate this situation more generally:

Definition 2.1. Let $\mathscr{P} = (\mathbf{S}, \mathbf{X}, \delta, \mathbf{C}, s)$ be a processor and let $A, B \in \mathbf{C}$ be some registers such that $\Gamma_A = \Gamma_B$, moreover, let $x \in \mathbf{X}$ be an input signal of the automaton of $\mathscr{P}$ (i.e. an instruction of $\mathscr{P}$). The instruction x is called a loading from register A

to register B if

(2.2) $\quad \forall S \in \mathbf{S}\colon (B)_{sx} = (A)_s \quad \text{and} \quad \forall R \in \mathbf{C}[R \neq B \Rightarrow (R)_{sx} = (R)_s].$

The register A is called the source register whilst B is called the destination register. The meaning of the instruction x is: Move the content of the source register A to the destination register B and we write the mnemonic *LD B, A*. □

Definition 2.2. Let $\mathscr{P} = (\mathbf{S}, \mathbf{X}, \delta, \mathbf{C}, s)$ be a processor and $A = \{A_v \subseteq \mathbf{S} | v \in \Gamma_A\}$ be a register of $\mathscr{P}$. An instruction $x \in \mathbf{X}$ is called an immediate loading of $n \in \Gamma_A$ into the register A, if

(2.3) $\quad \forall s \in \mathbf{S}\colon (A)_{sx} = n \quad \text{and} \quad \forall R \in \mathbf{C}[R \neq A \Rightarrow (R)_{sx} = (R)_s].$ □

Definition 2.3. The instruction $x \in \mathbf{X}$ is called an addition if there exist registers $A, B \in \mathbf{C}$ such that

$$\Gamma_A = \{0, 1, \ldots, |A|-1\} \supseteq \Gamma_B = \{0, 1, \ldots, |B|-1\}$$

and

$$\forall s \in \mathbf{S}\colon (A)_{sx} = (A)_s + (B)_s \pmod{|A|} \quad \text{and} \quad \forall R \in \mathbf{C}[R \neq A \Rightarrow (R)_{sx} = (R)_s].$$ □

Remark 2.1. One can think that the above definitions are not new. But it seems, the novelty is hidden in the fact that we rest on precize mathematical notions. The corresponding definitions of assembly language instructions use the notion of a processor in heuristical sense from the point of view of mathematics, moreover for want of the notion of state, the above definitions cannot be formulated in the usual assembly language descriptions.

Let $\mathbf{M} = (\mathbf{S}, \mathbf{C}, s)$ be an independent memory on a finite set $\mathbf{S}$. It may happen that there exist different states $s', s'' \in \mathbf{S}$ $(s' \neq s'')$ such that

(2.4) $\quad \forall R \in \mathbf{C}\colon (R)_{s'} = (R)_{s''}.$

Then, we say that the memory $\mathbf{M}$ is not economically defined on $\mathbf{S}$. Clearly, in the contrary case when $\mathbf{M}$ is economically defined, on $\mathbf{S}$, it follows from (2.4) that $s' = s''$.

We gave in [3] the following characterization:

Let $\mathbf{M}$ be an independent memory on the finite set $\mathbf{S}$. $\mathbf{M}$ is economically defined on $\mathbf{S}$ iff the storing capacity of $\mathbf{M}$ is the largest possible on $\mathbf{S}$:

(2.5) $\quad \operatorname{cap}(\mathbf{M}) = \log_2 |\mathbf{S}|.$

On the basis of this result, we prove the following lemma.

Lemma 2.1. *Let* $\mathbf{M}$ *be an independent memory on a finite set* $\mathbf{S}$. $\mathbf{M}$ *is economically defined on* $\mathbf{S}$ *iff the volume of the core of* $\mathbf{M}$ *is the finest possible partition of* $\mathbf{S}$.

PROOF. By the volume of the core $\mathbf{C} = \{R_1, \ldots, R_n\}$ we mean the product $R = R_1 \ldots R_n$. Thus, we have to show that (2.5) is is equivalent to

(2.6) $\quad R_1 \ldots R_n = \{\{s\} | s \in S\}.$

Since ([3]) $\operatorname{cap}(\mathbf{M}) = \log_2 |R_1 \ldots R_n|$, (2.5) is equivalent to

(2.7) $\quad |R| = |R_1 \ldots R_n| = |\mathbf{S}|.$

But (2.7) is valid if and only if (2.6) holds. ∎

Definition 2.4. Let $\mathscr{P}=\{\mathbf{S}, \mathbf{X}, \delta, \mathbf{C}, s\}$ be a processor and $x\in\mathbf{X}$ be an instruction of $\mathscr{P}$. (i) A procedure resulting in $s'=\delta(s, x)$ for all $s\in\mathbf{S}$ will be called a state level description of the instruction x and we shall speak in brief of an SL-procedure. (ii) A procedure will be called a content level description of the instruction x (briefly a CL-procedure for x) if for all registers $R\in\mathbf{C}$, it gives the alteration of the content of R which operation performed means the execution of the instruction x.

For example a table for the function $\delta(s, x)$ contains state level descriptions of all instructions $x\in X$. The descriptions of the meaning of instructions can provide examples for content level descriptions of instructions.

Theorem 2.1 *Let $\mathscr{P}=(\mathbf{S}, \mathbf{X}, \delta, \mathbf{C}, s)$ be a processor and $x\in\mathbf{X}$ be an instruction of $\mathscr{P}$. If the memory of $\mathscr{P}$ is economically defined on $\mathbf{S}$ then the state level descriptions of x are equivalent to the content level descriptions of x in the sense that any state level description of x involves a content level description of x and inversely, any content level description involves a state level description of x.*

PROOF. Let $sx=\delta(s, x)$ be given by some SL-procedure. If $\mathscr{P}$ receives x in state s then $(R)_s$ is the content of an arbitrary register R of $\mathscr{P}$. The execution of the instruction x means that the new instant state sx bacomes effective instead of s. Then the new content of R will be $(R)_{sx}$. This is a description of the alteration of the content of an arbitrary register of $\mathscr{P}$, i.e. this is a description of that operation on the content of registers which performed means the execution of the instruction x. So, we have obtained a content level description of x.

Now, given a CL-procedure supplying for all $v\in\{1, \ldots, n\}$ the new content j_v of the register $R_v=\{B_k^{(v)}\,|\,k\in\Gamma_{R_v}\}$ which follows on effect of the instruction x from the previous content of the registers of C. Let $s\in\mathbf{S}$ be an arbitrary state and suppose that $\mathscr{P}$ receives x. Then in this state $j_v=(R_v)_{sx}$ and so, according to the definition of the content of registers we have

$$\forall v\in\{1, \ldots, n\}\colon\ sx\in B_{j_v}^{(v)},\quad \text{i.e.,}\quad sx\in B_{j_1}^{(1)}\cap\ldots\cap B_{j_n}^{(n)}. \tag{2.11}$$

Since $\mathbf{M}$ is independent and it is economically defined on $\mathbf{S}$, the volume of the core of $\mathbf{M}$ is the finest possible partition of $\mathbf{S}$ see (Lemma 2.1). Thus, sx is uniquely determined by (2.11). This is an SL-procedure to obtain $sx=\delta(s, x)$. ∎

Finally, we give a nontrivial example for a processor with a big number of states, such that the instructions can be described by CL-procedures, only. This processor will be the world-famous logical game: the Rubik's cube. Naturally, the inventor E. Rubik — professor of the polytechnical university Budapest — was not aware of the fact that he created a processor neither in abstract nor in practical sense. The abstract notion of processor did not exist that time, and the practical notion of processors was known as a thing which is in strong connection with electronic digital computers. The next example, can be regarded as the first abstract description of the Rubik's cube as a processor.

Example 3. There are two ways of the definition of a processor $\mathscr{P}=(\mathbf{A}, \mathbf{M})$.

1. First, we give an automaton $\mathbf{A}=(\mathbf{S}, \mathbf{X}, \delta)$ and then, we define a memory $\mathbf{M}=(\mathbf{S}, \mathbf{C}, s)$ on the state set $\mathbf{S}$ of $\mathbf{A}$.

2. First, we construct a memory $\mathbf{M}=(\mathbf{S}, \mathbf{C}, s)$ on a set $\mathbf{S}$ and then we give a set $\mathbf{X}$ and a function $\delta: \mathbf{S}\times\mathbf{X}\to\mathbf{S}$ in order to define the automaton $\mathbf{A}=(\mathbf{S}, \mathbf{X}, \delta)$.

Now, if it is intended to define the state transition function by CL-procedures for the instructions then we must choose only the 2nd way, since the CL-procedures operate on the contents of registers.

Next, we give the state set $\mathbf{S}$. Let H be an arbitrary set of 6 elements. For the sake of simplicity, the elements of H will be denoted by the numbers $1, \ldots, 6$:

$$H = \{1, 2, 3, 4, 5, 6\}. \tag{2.12}$$

Let $\mathbf{S}$ be the Cartesian product of H with itself 54 times:

$$\mathbf{S} = \overset{1}{H}\times\ldots\times\overset{54}{H} = H^{54}. \tag{2.13}$$

Thus, any element $s\in\mathbf{S}$ will be an 54-dimensional vector:

$$s = (s_1, s_2, \ldots, s_{54}) \quad (s_k\in H,\ k = 1, \ldots, 54). \tag{2.14}$$

Now, we define the memory $\mathbf{M}=(\mathbf{S}, \mathbf{C}, s)$ on $\mathbf{S}$ with a core

$$\mathbf{C} = \{P_1, P_2, \ldots, P_{54}\} \tag{2.15}$$

containing 54 memory cells where the partitions P_k $(k=1, \ldots, 54)$ are generated by that functions $f_k: \mathbf{S}\to H$ for which

$$f_k(s) = s_k, \quad s = (s_1, \ldots, s_{54})\in\mathbf{S}. \tag{2.16}$$

We see that each partition belonging to $\mathbf{C}$ contains exactly 6 blocks. For all $k\in\{1, \ldots, 54\}$, let the blocks $f_k^{-1}(i)$ be marked by colours such that the blocks $f_k^{-1}(1)$, $f_k^{-1}(2)$, $f_k^{-1}(3)$, $f_k^{-1}(4)$, $f_k^{-1}(5)$ and $f_k^{-1}(6)$ will be of colour white, yellow, blue, pink green and red, respectively. In other words, we take the generator functions $g_k(s)=$ $=\Phi(f_k(s))$ instead of the generator functions $f_k(s)$ where Φ is the function for which

$$\Phi(1) = \text{"White" briefly } W$$

$$\Phi(2) = \text{"Yellow" briefly } Y$$

$$\Phi(3) = \text{"Blue" briefly } B$$

$$\Phi(4) = \text{"Pink" briefly } P$$

$$\Phi(5) = \text{"Green" briefly } G$$

$$\Phi(6) = \text{"Red" briefly } R.$$

Moreover if the content of some register $P_v\in\mathbf{C}$ is "white" then we say that the register P_v is of colour white, if $(P_v)_s=Y$ then we say that P_v is of colour yellow and so on.

In this manner we have defined the memory $\mathbf{M}=(\mathbf{S}, \mathbf{C}, s)$ where the state set $\mathbf{S}$ is given by (2.13) and the core $\mathbf{C}$ is of the form (2.15) with memory cells P_v generated by the functions f_v.

We shall show that **M** is independent and it is economically defined on **S**. Indeed, the volume of **C** is

$$P_1 \ldots P_{54} = \{f_1^{-1}(i_1) \cap \ldots \cap f_{54}^{-1}(i_{54}) | s = (i_1, \ldots, i_{54}) \in H^{54} (= \mathbf{S})\} =$$

$$= \{f_1^{-1}[f_1(s)] \cap \ldots \cap f_{54}^{-1}[f_{54}(s)] | s \in \mathbf{S}\} = \{\{s\} | s \in \mathbf{S}\}.$$

Thus, we see on the basis of Theorem 3.2 of [3] that **M** is independent and by Lemma 2.1, that it is economically defined on **S**.

Now, we shall define an automaton $\mathbf{A} = (\mathbf{S}, \mathbf{X}, \delta)$. Let the set **X** of instructions be defined as follows:

$$\mathbf{X} = \{W, Y, B, P, G, R\}. \tag{2.17}$$

As mentioned above, the state transition function $\delta(s, x)$, will be given by content-level descriptions of the instructions $x \in \mathbf{X}$. So, on the basis of Theorem 2.1, for each $x \in \mathbf{X}$, we can obtain the function $\delta(s, x) = \delta_k(s)$: $\mathbf{S} \to \mathbf{S}$.

In order to make easier the content level descriptions, we shall change the denotations of the registers $P_1, \ldots, P_{54}$.

For $k \in \{1, \ldots, 54\}$ let the numbers n, m be determined such that

$$k - 1 = 9n + m \quad \text{where} \quad n, m \in \{0, 1, \ldots, 8\}. \tag{2.18}$$

Then for P_k, we use the denotation

$$P_k = \Phi(n+1)m. \tag{2.19}$$

Thus for example

$$P_1 = W\emptyset, \quad P_2 = W1, \ldots, P_{10} = Y\emptyset, \ldots, P_{54} = R8.$$

If A and B are registers then in case of "CL-instructions" we shall use the standard denotation $A := B$ instead of LD A, B or LET $A = B$, etc. $A := B$ will be a "CL-instruction" of a CL-procedure describing an instruction $x \in \mathbf{X}$ on content's level.

For the instructions $x \in \mathbf{X}$, we shall use the mnemonics shown on Fig. 3. The Rubik's cube is shown on Fig. 4, while the CL-procedures describing the the instructions x are given by a table shown on Fig 5. Here Z is an auxiliary register for tempo-

Instruction	Instruction's name (mnemonic)	Meaning of the instruction's name
W	RRW	Rotate Right the side of color White
Y	RRY	Rotate Right the side of color Yellow
B	RRB	Rotate Right the side of color Blue
P	RRP	Rotate Right the side of color Pink
G	RRG	Rotate Right the side of color Green
R	RRR	Rotate Right the side of color Red

Fig. 3.

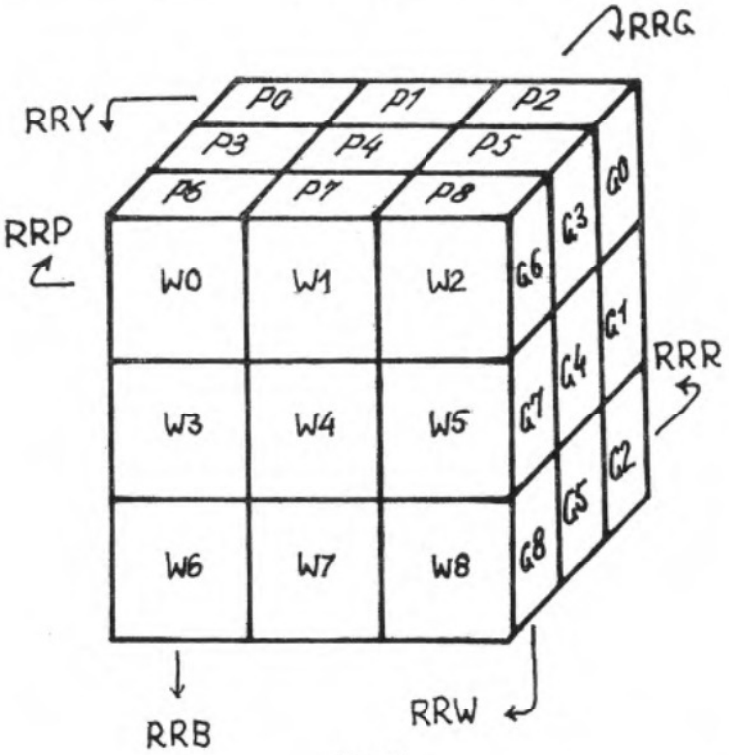


Fig. 4.

CL-procedure for RRW	CL-procedure for RRY	CL-procedure for RRB	CL-procedure for RRP	CL-procedure for RRG	CL-procedure for RRR
Z:=W6	Z:=Y6	Z:=B6	Z:=P6	Z:=G6	Z:=R6
W6:=W8	Y6:=Y8	B6:=B8	P6:=P8	G6:=G8	R6:=R8
W8:=W2	Y8:=Y2	B8:=B2	P8:=P2	G8:=G2	R8:=R2
W2:=W0	Y2:=Y0	B2:=B0	P2:=P0	G2:=G0	R2:=R0
W0:=Z	Y0:=Z	B0:=Z	P0:=Z	G0:=Z	R0:=Z
Z:=W3	Z:=Y3	Z:=B3	Z:=P3	Z:=G3	Z:=R3
W3:=W7	Y3:=Y7	B3:=B7	P3:=P7	G3:=G7	R3:=R7
W7:=W5	Y7:=Y5	B7:=B5	P7:=P5	G7:=G5	R7:=R5
W5:=W1	Y5:=Y1	B5:=B1	P5:=P1	G5:=G1	R5:=R1
W1:=Z	Y1:=Z	B1:=Z	P1:=Z	G1:=Z	R1:=Z
Z:=G6	Z:=B2	Z:=Y8	Z:=Y2	Z:=Y0	Z:=Y6
G6:=P6	B2:=P2	Y8:=R8	Y2:=B8	Y0:=P8	Y6:=G8
P6:=B6	P2:=G2	R8:=W0	B8:=W2	P8:=W8	G8:=W6
B6:=R6	G2:=R2	W0:=P0	W2:=G0	W8:=R0	W6:=B0
R6:=Z	R2:=Z	P0:=Z	G0:=Z	R0:=Z	B0:=Z
Z:=G7	Z:=B1	Z:=Y5	Z:=Y1	Z:=Y3	Z:=Y7
G7:=P7	B1:=P1	Y5:=R5	Y1:=B5	Y3:=P5	Y7:=G5
P7:=B7	P1:=G1	R5:=W3	B5:=W1	P5:=W5	G5:=W7
B7:=R7	G1:=R1	W3:=P3	W1:=G3	W5:=R3	W7:=B3
R7:=Z	R1:=Z	P3:=Z	G3:=Z	R3:=Z	B3:=Z
Z:=G8	Z:=B0	Z:=Y2	Z:=Y0	Z:=Y6	Z:=Y8
G8:=P8	B0:=P0	Y2:=R2	Y0:=B2	Y6:=P2	Y8:=G2
P8:=B8	P0:=G0	R2:=W6	B2:=W0	P2:=W2	G2:=W8
B8:=R8	G0:=R0	W6:=P6	W0:=G6	W2:=R6	W8:=B6
R1:=Z	R0:=Z	P6:=Z	G6:=Z	R6:=Z	B6:=Z

Fig. 5.
Content level descriptions of instructions

rary preservation of the content of some registers of the processor. It is necessary only in case of simulation of the CL-procedures on a sequencial machine. In case of the real Rubik's cube, any execution for an instruction is performed through a set of parallel excahanges of the content of registers. Any register is a source and at the same time it is a destination too. Considering the final result, the above two ways of execution don't differ.

In this paper, we have described the transmission of information in the inside of a processor. In the next paper (in the part III), we shall deal with the transmission of information between processors.

References

[1] J. J. DONOVAN, Systems programing, *Mc Graw-Hill Book Company*.
[2] E. GESZTELYI, A deterministic mathematical model of the transmission of information I. *Publ. Math.* (Debrecen).
[3] E. GESZTELYI, A mathematical theory of processors I. *MTA SZTAKI Közlemények* **37**/1987 pp. 21—61.
[4] F. GÉCSEG, Products of automata, *Springer-Verlag* 1986.
[5] L. A. LEVENTHAL, Z80 Assembly language programming, *Osborn et Ass. Inc. Berkeley*, California.
[6] G. LUECKE, J. P. MIZE, W. N. CARR, Semiconductor Memory Design and Application, *Mc Graw-Hill Book Company* (1980).

(Received December, 1988)