Mathematical Model of Computer Viruses

PhD. Ferenc Leitold,

Veszprém University - Veszprog Ltd., Hungary
fleitold@veszprog.hu

Table of contents

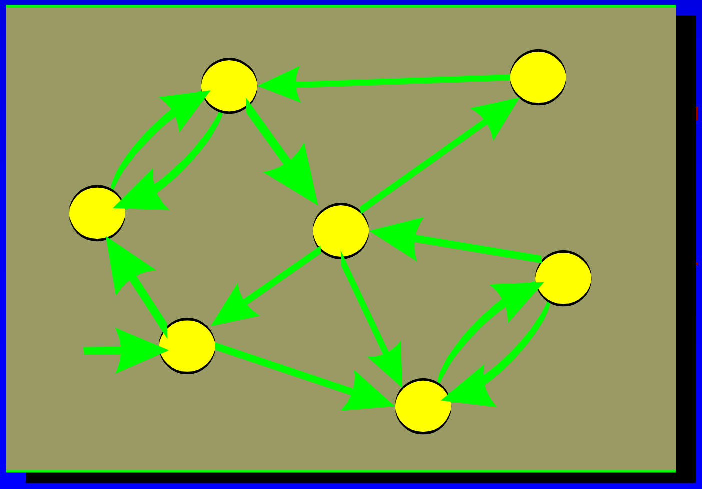
- Models of computation
- Operating system
- Virus definition
- What can we do with this mathematical model ?

Turing Machine

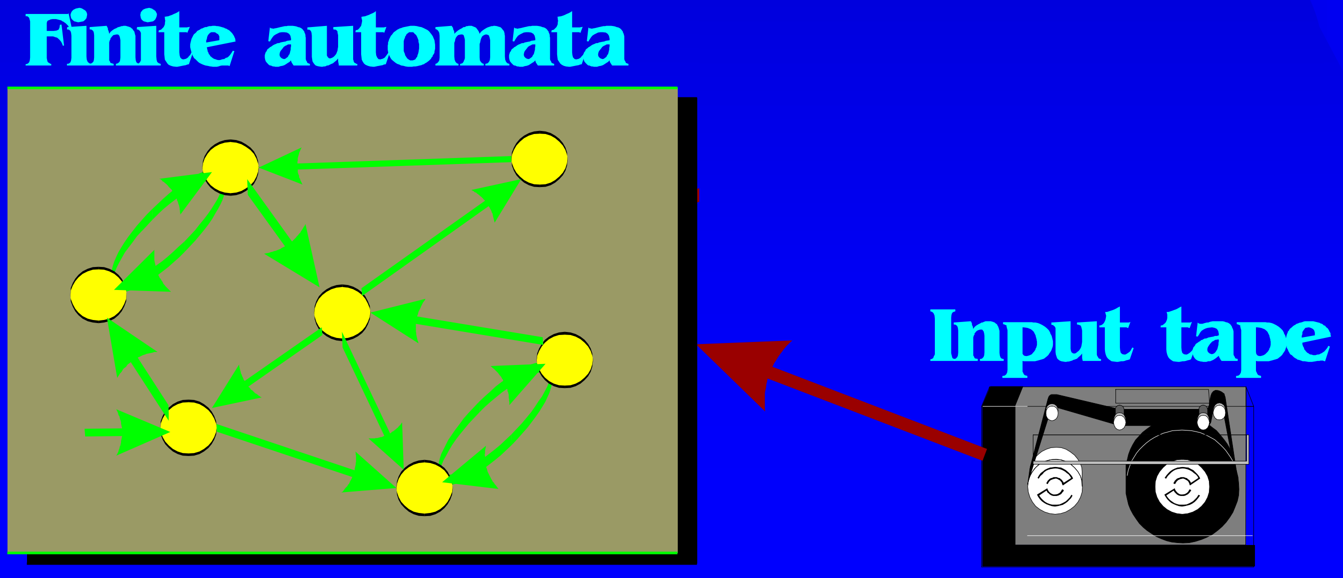
Turing Machine

VESZPROG

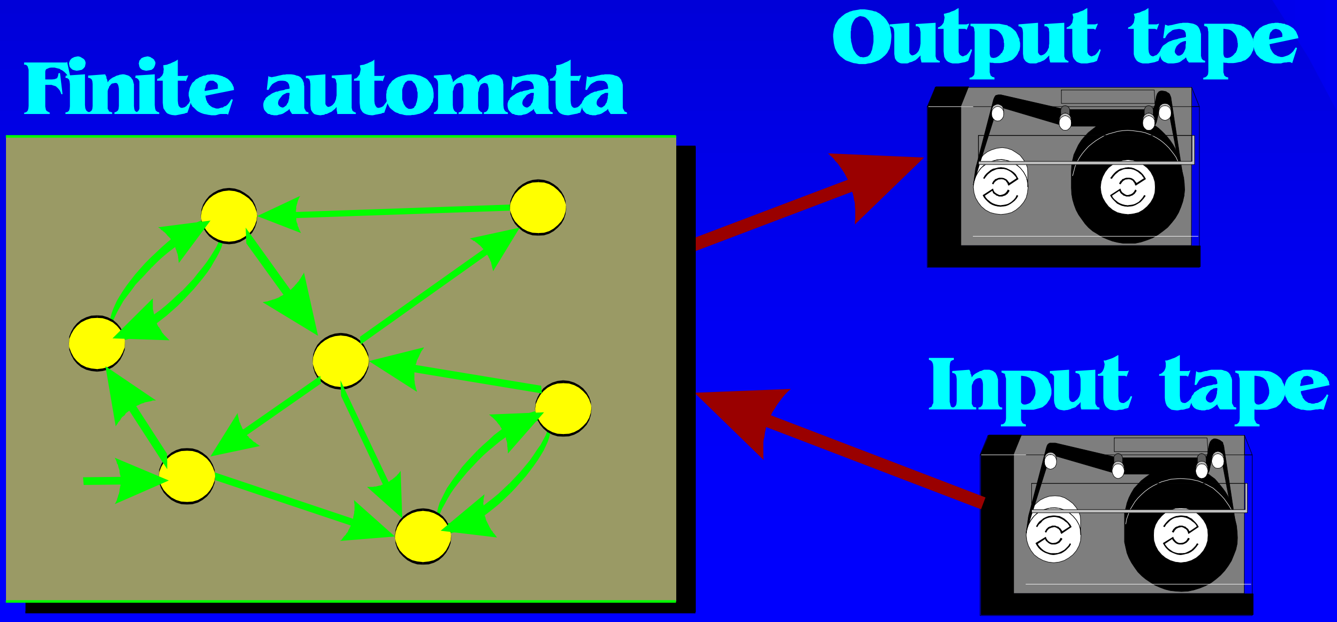
Finite automata



Turing Machine



Turing Machine



Turing Machine

VESZPROG

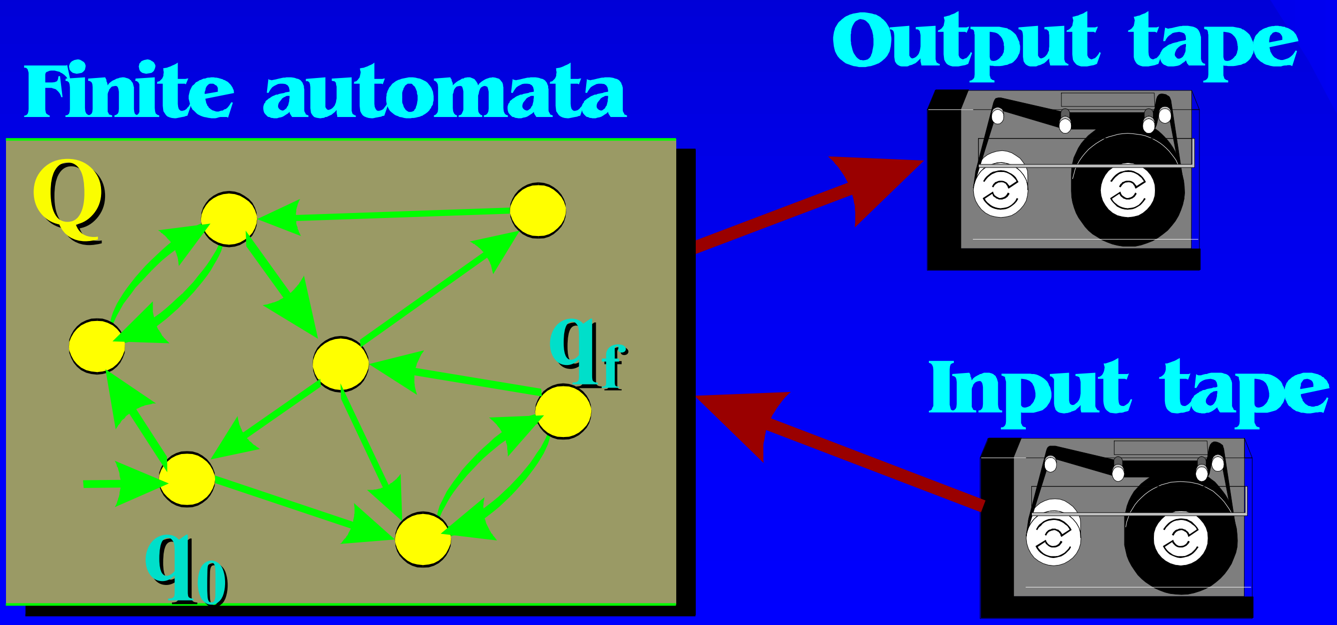
$$T =< \mathcal{Q}, S, I, \delta, b, q_0, q_f >$$

S: tape symbols

I: input symbols, $I \subset S$

b: blank symbol, $b \in S \setminus I$

δ: move function, $\delta : Q \times S \rightarrow Q \times S \times \{l, r, s\}$

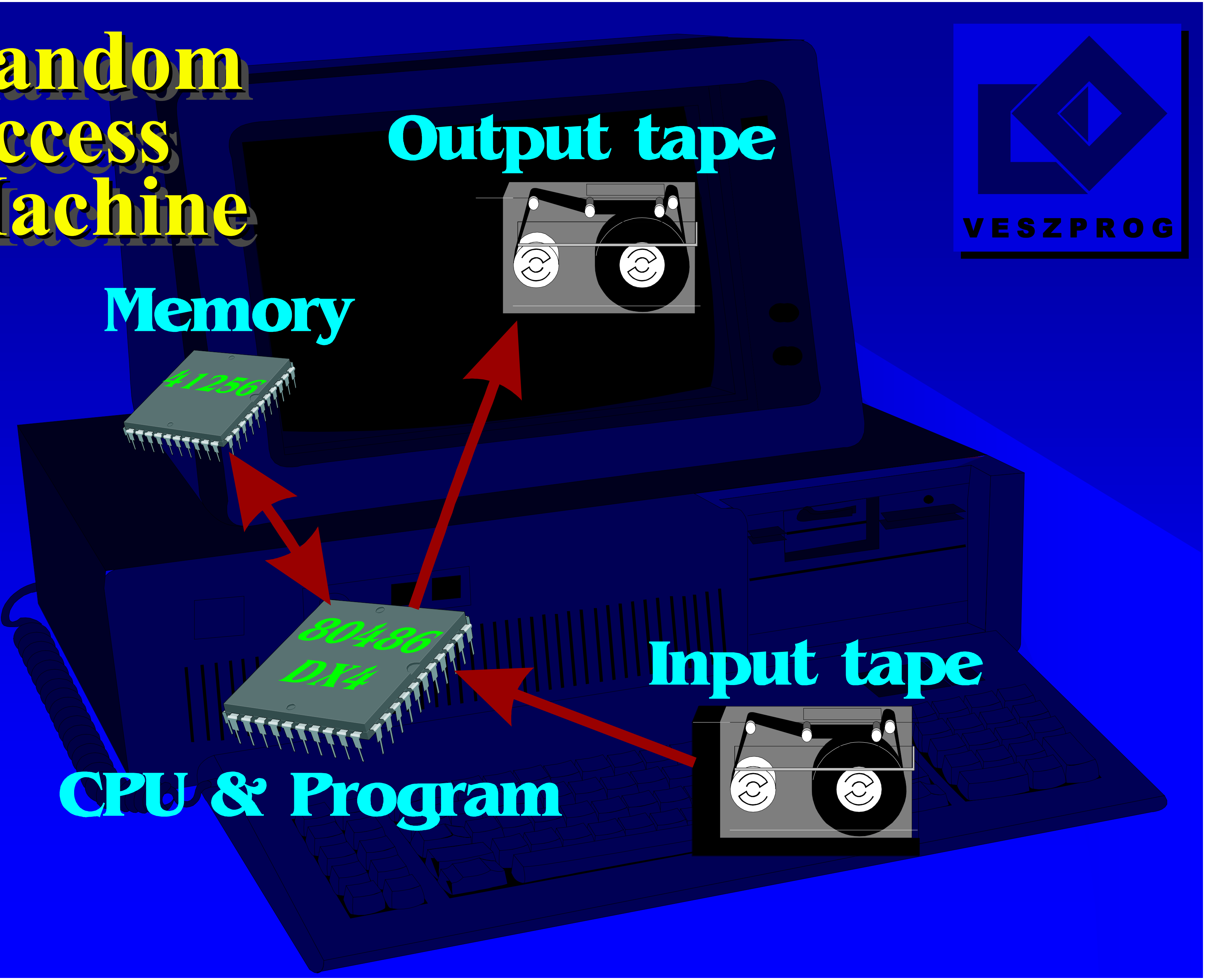


Finite automata

Output tape

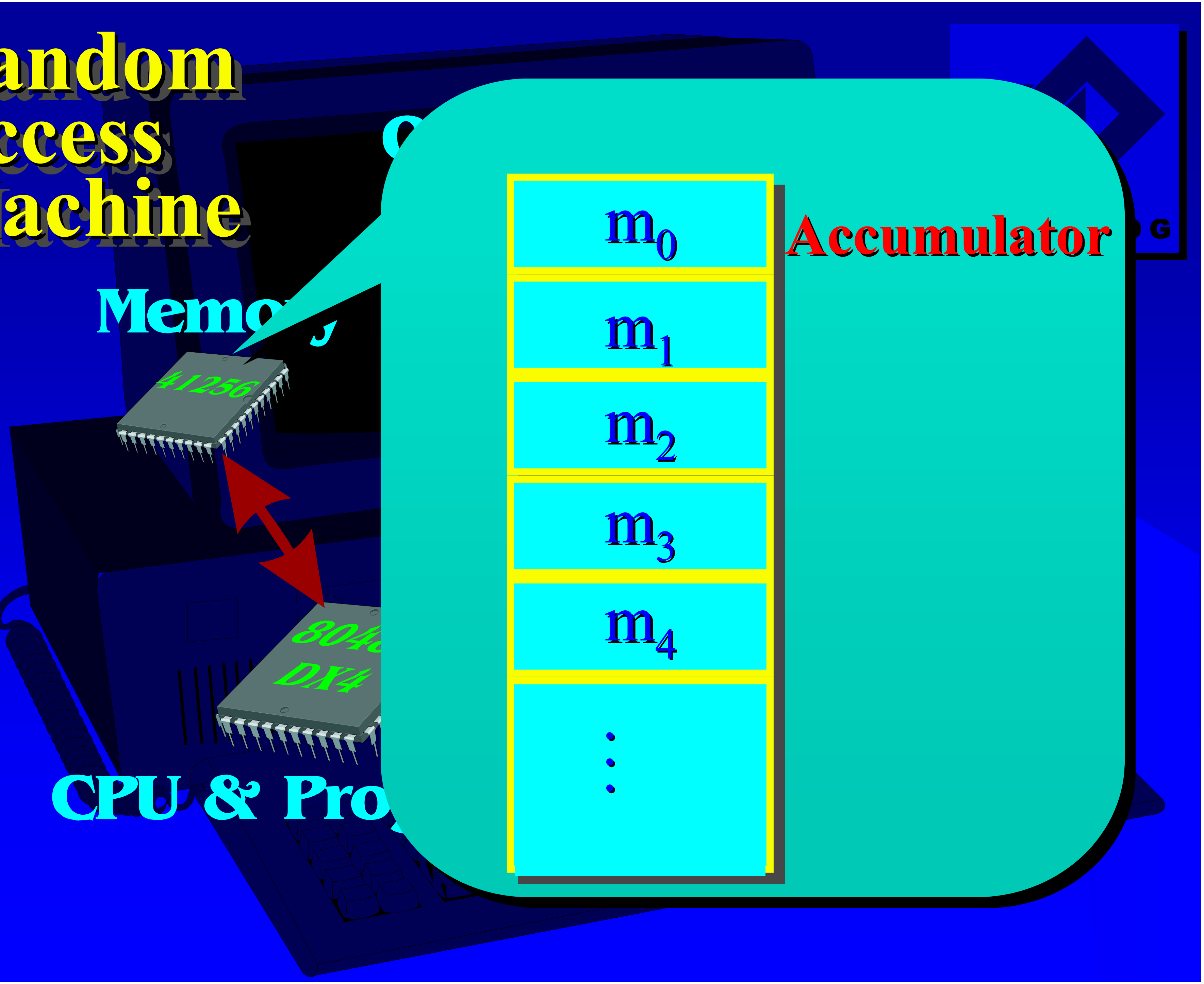
Input tape

Random Access Machine

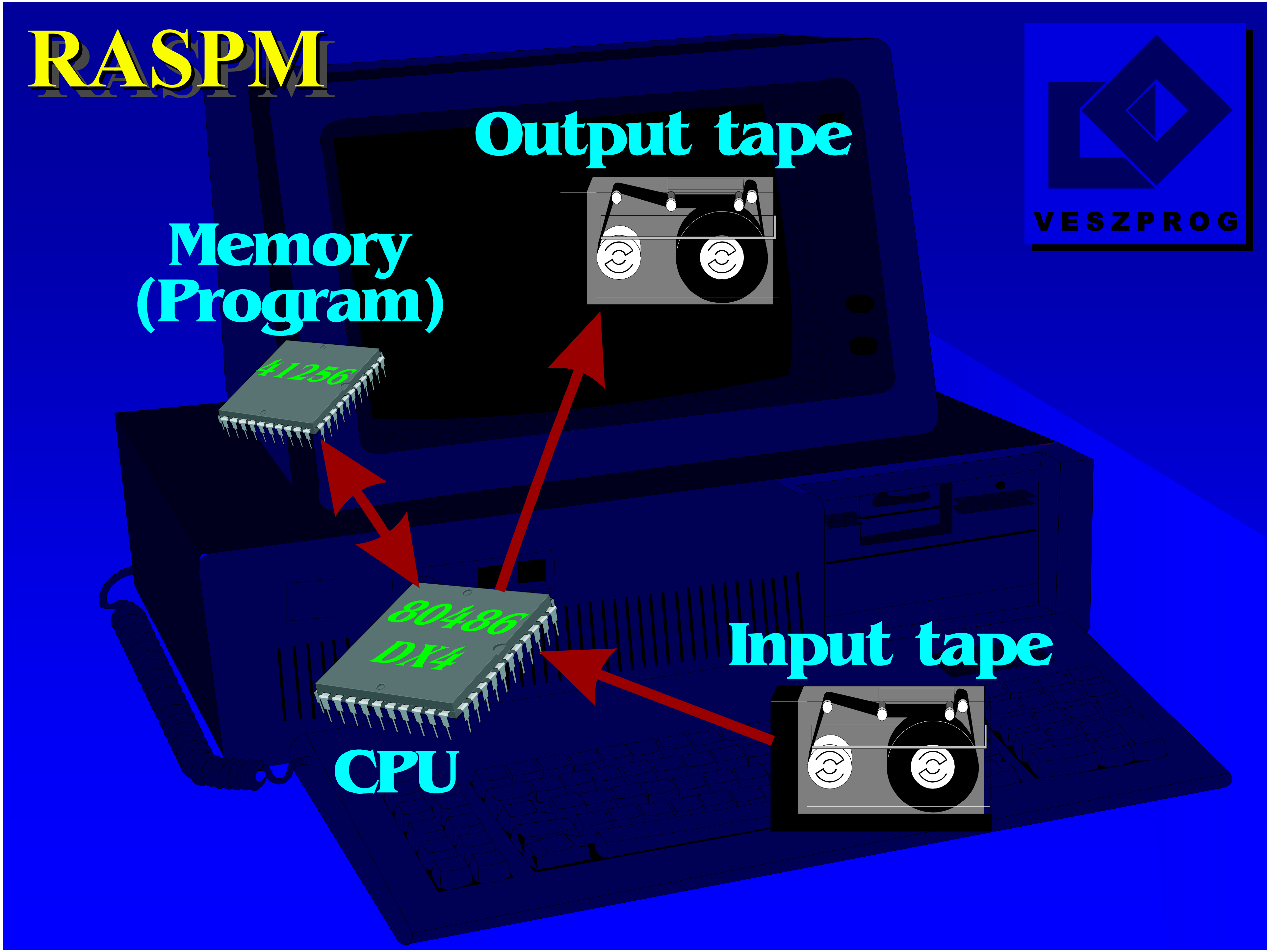


VESZPROG

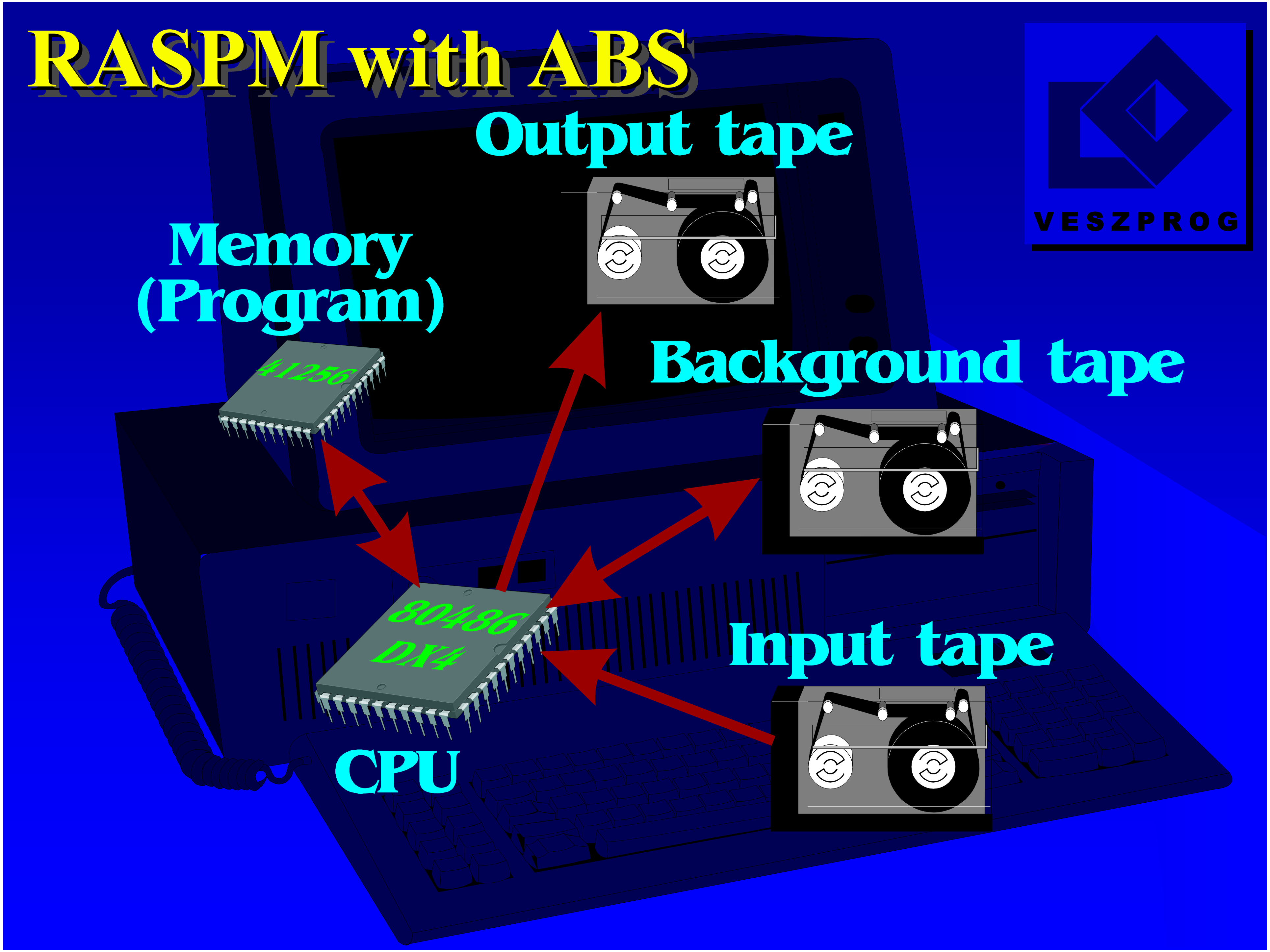
Random Access Machine



RASPM

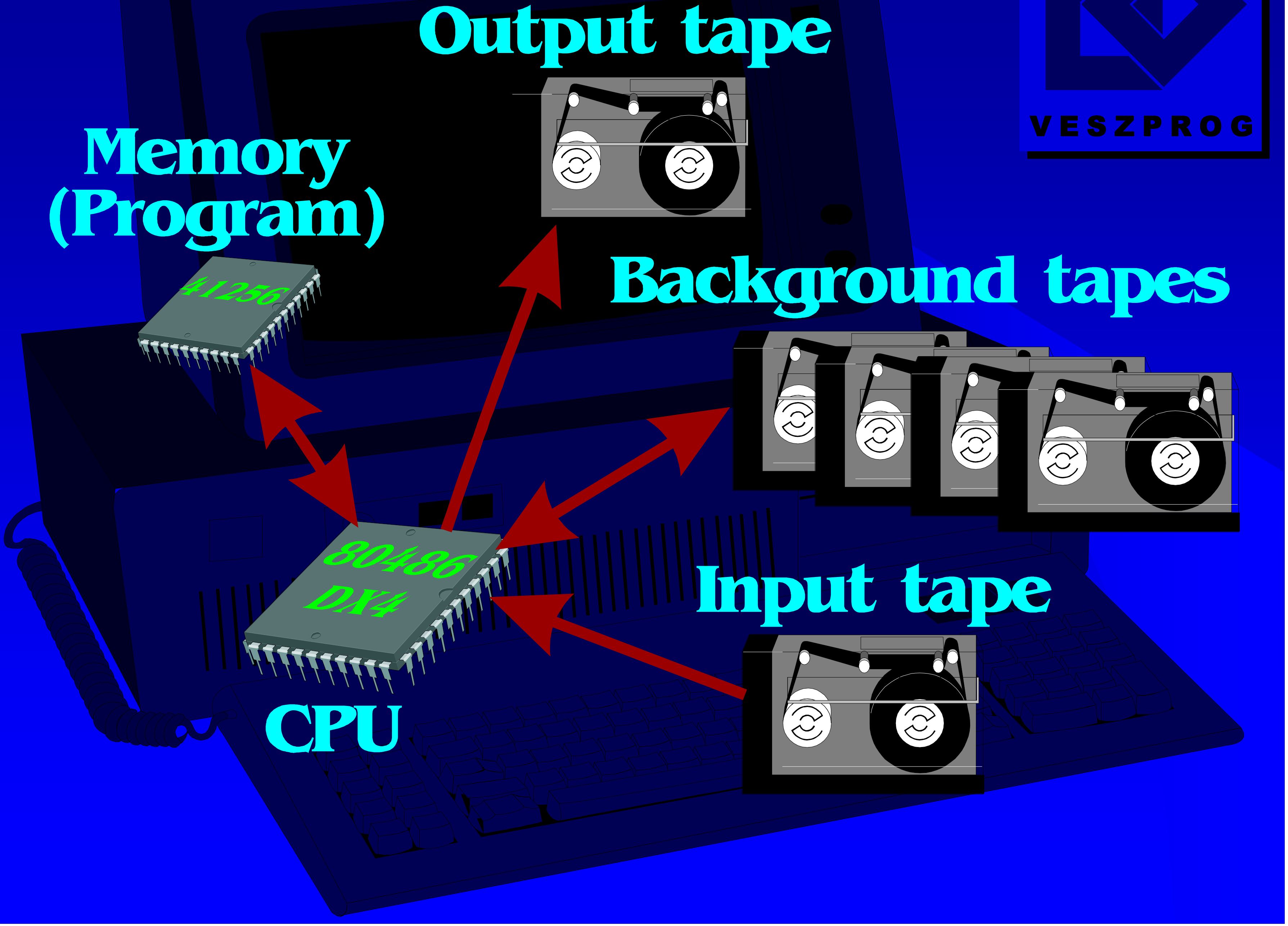

Output tape
Memory
(Program)
41256
80486
DX4
CPU
Input tape
VESZPROG

RASPM with ABS



RASPM with SABS

VESZPROG


Output tape

Memory (Program)

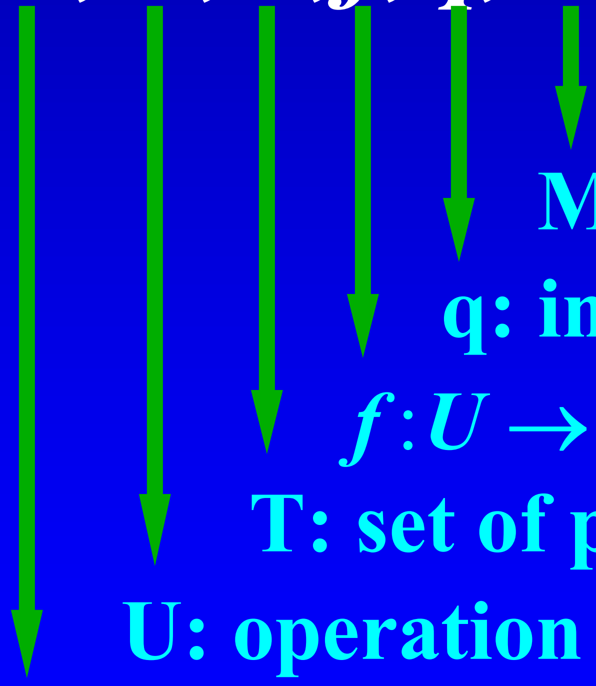
Background tapes

Input tape

CPU

RASPM with ABS definition

$$G = <V,U,T,f,q,M>$$



M: initial memory content

q: initial value of the IP

$f : U \rightarrow T$

T: set of processor’s activities

U: operation codes, $U \subseteq V$

V: set of symbols

Instruction set

- **move (LOAD, STORE)**
- **logical (AND, OR, XOR)**
- **arithmetic (ADD, SUB, MULT, DIV)**
- **branch (JUMP, JGTZ, JZERO)**
- **input/output tape handling (READ, WRITE)**
- **background tape handling (GET, PUT, SEEK, SETDRIVE)**

Operating System

VESZPROG

Operating System

- system of programs

VESZPROG

Operating System

- system of programs
- able to handle separate program or data files

Operating System

- system of programs
- able to handle separate program or data files
- able to make a specified program to run.

Operating Systems under RASPM with ABS

VESZPROG

Operating Systems under RASPM with ABS

- The OS is in the initial memory (M)

Operating Systems under RASPM with ABS

VESZPROG

- The OS is in the initial memory (M)
  → OS specific machine

Operating Systems under RASPM with ABS

- The OS is in the initial memory (M)
  → OS specific machine
- The OS is in the background tape

Operating Systems under RASPM with ABS

VESZPROG

- The OS is in the initial memory (M)
  → OS specific machine
- The OS is in the background tape
  → OS independent machine

Operating Systems under RASPM with ABS

VESZPROG

- The OS is in the initial memory (M)
  → OS specific machine
- The OS is in the background tape
  → OS independent machine
- The OS is in the input tape

Operating Systems under RASPM with ABS

- The OS is in the initial memory (M)
  → OS specific machine
- The OS is in the background tape
  → OS independent machine
- The OS is in the input tape
  → unusable

Operating Systems under RASPM with ABS

- The OS is in the initial memory (M)
  → OS specific machine
- The OS is in the background tape
  → OS independent machine
- The OS is in the input tape
  → unusable

Sample OS

VESZPROG

Comparing RASPM with ABS-es

$G_1=<V_1,U_1,T_1,f_1,q_1,M_1>$

$G_2=<V_2,U_2,T_2,f_2,q_2,M_2>$

Comparing RASPM with ABS-es

$$G_1=<V_1,U_1,T_1,f_1,q_1,M_1>$$

$$G_2=<V_2,U_2,T_2,f_2,q_2,M_2>$$

$$\{q_1,M_1\} \neq \{q_2,M_2\}$$

Comparing RASPM with ABS-es

$$G_1=<V_1,U_1,T_1,f_1,q_1,M_1>$$

$$G_2=<V_2,U_2,T_2,f_2,q_2,M_2>$$

$$\{q_1,M_1\} \neq \{q_2,M_2\}$$

- different operating systems
- different loader program

Comparing RASPM with ABS-es

$G_1=<V_1,U_1,T_1,f_1,q_1,M_1>$

$G_2=<V_2,U_2,T_2,f_2,q_2,M_2>$

Comparing RASPM with ABS-es

VESZPROG

$$G_1 = \langle V_1, U_1, T_1, f_1, q_1, M_1 \rangle$$

$$G_2 = \langle V_2, U_2, T_2, f_2, q_2, M_2 \rangle$$

$$\{f_1, T_1, U_1\} \neq \{f_2, T_2, U_2\}$$

Comparing RASPM with ABS-es

$$G_1=<V_1,U_1,T_1,f_1,q_1,M_1>$$

$$G_2=<V_2,U_2,T_2,f_2,q_2,M_2>$$

$$\{f_1,T_1,U_1\} \neq \{f_2,T_2,U_2\}$$

- **different activities**
- **different operation codes**

Comparing RASPM with ABS-es

$G_1 = < V_1, U_1, T_1, f_1, q_1, M_1 >$

$G_2 = < V_2, U_2, T_2, f_2, q_2, M_2 >$

Comparing RASPM with ABS-es

$$G_1=<V_1,U_1,T_1,f_1,q_1,M_1>$$

$$G_2=<V_2,U_2,T_2,f_2,q_2,M_2>$$

$$V_1 \neq V_2$$

Comparing RASPM with ABS-es

$$G_1=<V_1,U_1,T_1,f_1,q_1,M_1>$$

$$G_2=<V_2,U_2,T_2,f_2,q_2,M_2>$$

$$V_1 \neq V_2$$

- different symbols
- different tape formats

Computer virus

VESZPROG

Computer virus

VESZPROG

- a (part of) program

Computer virus

- a (part of) program
- it is attached to a program area

Computer virus

- a (part of) program
- it is attached to a program area
- it is able to link itself to other program areas

Computer virus

- a (part of) program
- it is attached to a program area
- it is able to link itself to other program areas
- it is executed when the host program area is to be executed

Virus spreading modes

VESZPROG

Virus spreading modes

- machine specific

Virus spreading modes

VESZPROG

- machine specific
- machine independent

Virus spreading modes

- machine specific
- machine independent
- operating system specific

Virus spreading modes

- machine specific
- machine independent
- operating system specific
- operating system independent

Virus spreading modes

- machine specific
- machine independent
- operating system specific
- operating system independent

- direct

Virus spreading modes

- machine specific
- machine independent
- operating system specific
- operating system independent

- direct
- indirect

Virus spreading modes

VESZPROG

- machine specific
- machine independent
- operating system specific
- operating system independent

- direct
- indirect

Sample virus

VESZPROG

What can we do with this mathematical model ?

VESZPROG

What can we do with this mathematical model ?

VESZPROG

- Examine the working mechanism of viruses

What can we do with this mathematical model ?

- Examine the working mechanism of viruses
- Examine the virus detection problem

What can we do with this mathematical model ?

- Examine the working mechanism of viruses
- Examine the virus detection problem
- Examine multiplatform viruses

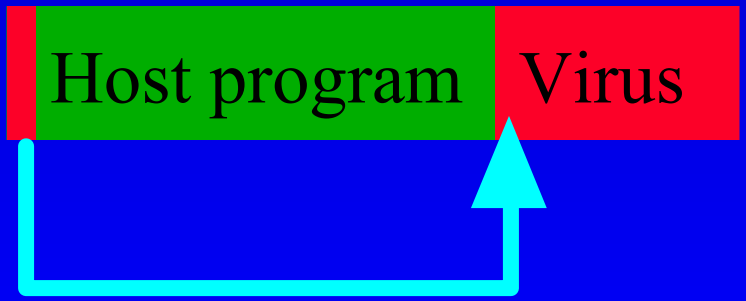
General virus detection problem

Theorem:

It is impossible to build a Turing Machine which could decide if an executable file in a RASPM with ABS contains a virus or not.

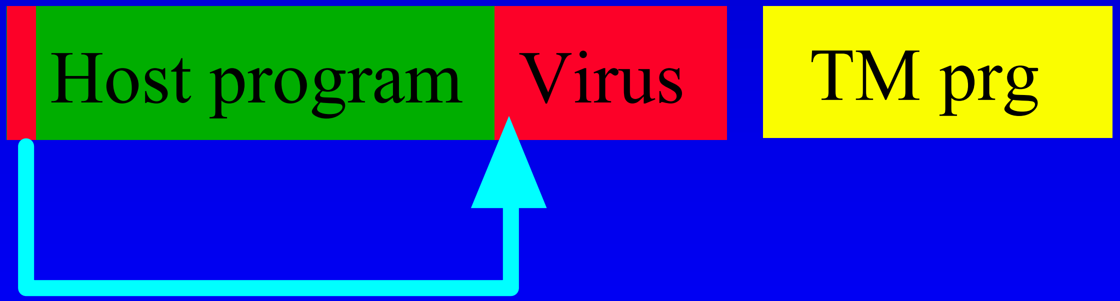
General virus detection problem

Proof:



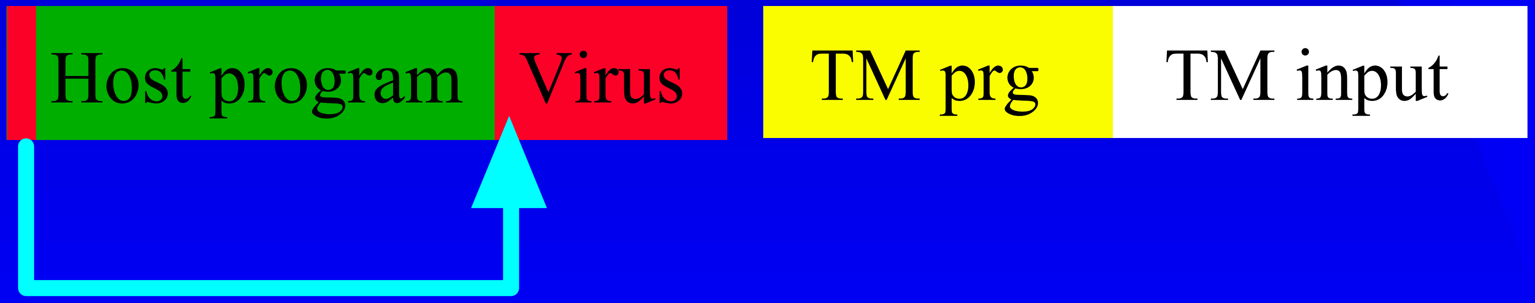
General virus detection problem

Proof:



General virus detection problem

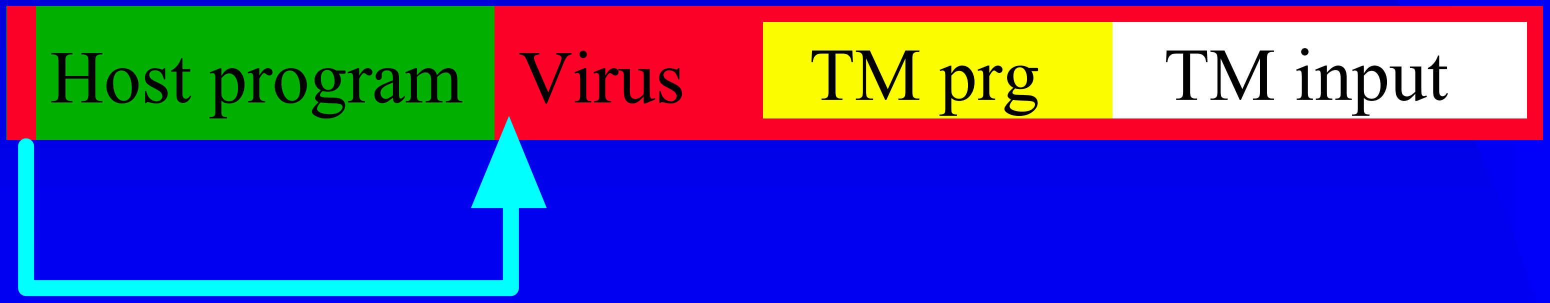
Proof:



General virus detection problem

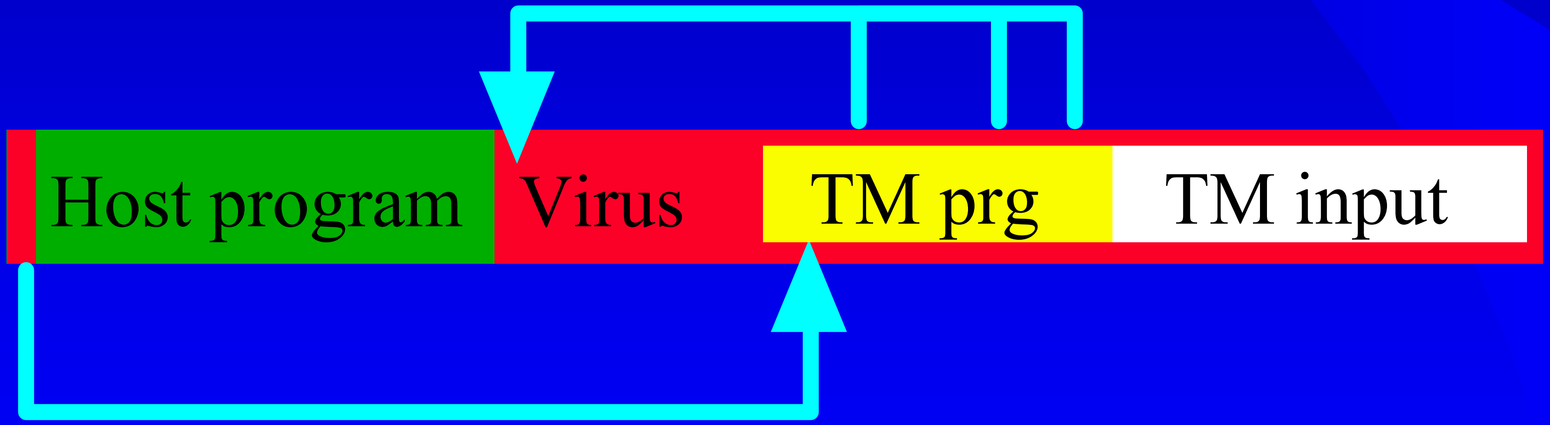
VESZPROG

Proof:



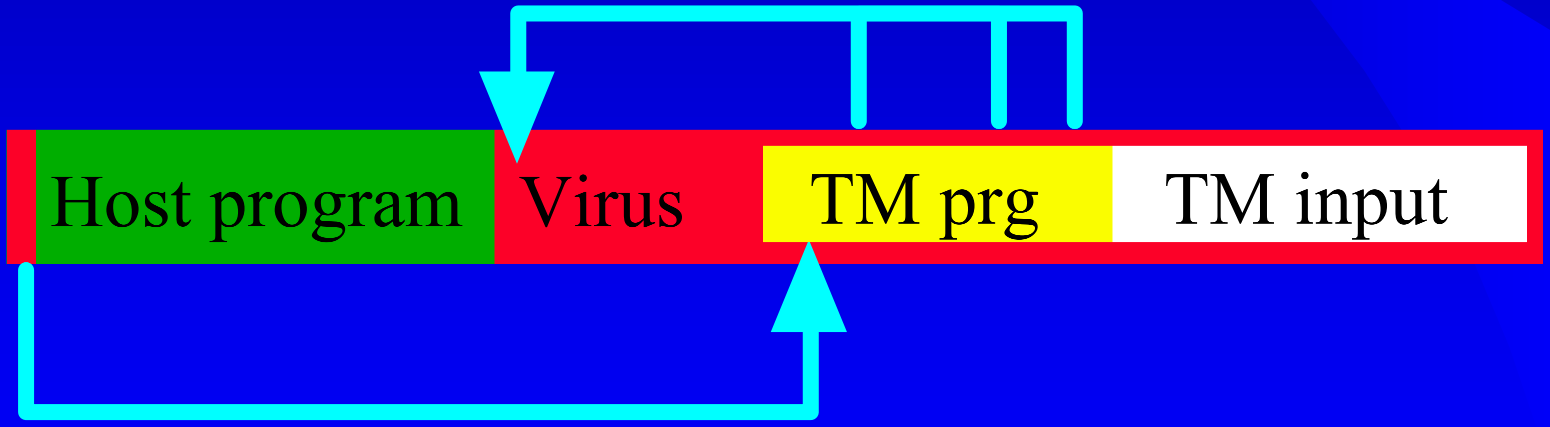
General virus detection problem

Proof:



General virus detection problem

Proof:



Virus detection problem ➡ TM halting problem

Multiplatform viruses

$$G_1=<V_1,U_1,T_1,f_1,q_1,M_1>$$

$$G_2=<V_2,U_2,T_2,f_2,q_2,M_2>$$

Multiplatform viruses

VESZPROG

$$G_1=<V_1,U_1,T_1,f_1,q_1,M_1>$$

$$G_2=<V_2,U_2,T_2,f_2,q_2,M_2>$$

Conditions:

V_1 🖐 $U_2 \neq 0$

U_1 🖐 $V_2 \neq 0$

G_1 has to know some operation codes of G_2

G_2 has to know some operation codes of G_1

Multiplatform viruses

$$G_1=<V_1,U_1,T_1,f_1,q_1,M_1>$$

$$G_2=<V_2,U_2,T_2,f_2,q_2,M_2>$$

Conditions:

U_1 ✋ $U_2 \neq 0$

- The virus code can be the same.

Multiplatform viruses

VESZPROG

$$G_1=<V_1,U_1,T_1,f_1,q_1,M_1>$$

$$G_2=<V_2,U_2,T_2,f_2,q_2,M_2>$$

Conditions:

U_1 🖐 $U_2 \neq 0$

- The virus code can be the same.

U_1 🖐 $U_2 = 0$

- The virus code must be different.

Future

VESZPROG

Future

- Examine general virus detection problem in limited cases:
  - Spreading under the model
  - Limit the time/space

Future

- Examine general virus detection problem in limited cases:
  - Spreading under the model
  - Limit the time/space
- Examine polymorphic techniques
  - Without coding/decoding
  - Changing instructions

VESZPROG

?

?

?

VESZPROG

?

?

VESZPROG

?

VESZPROG

?

?

?

Searching technique questions

VESZPROG

Searching technique questions

- For what kind of viruses can be used ?

Searching technique questions

- For what kind of viruses can be used ?
- What is the probability of false alarms ?

Searching technique questions

VESZPROG

- For what kind of viruses can be used ?
- What is the probability of false alarms ?
- What is the expense criteria ?

Sequence searching algorithm

VESZPROG

Sequence searching algorithm

- for non-polymorphic known viruses

Sequence searching algorithm

L: size of suspicious area
M: number of sequences
N: size of a sequence
n: number of values in one cell

- for non-polymorphic known viruses
- false alarms: $p \approx \frac{L \cdot M}{n^N}$

Sequence searching algorithm

L: size of suspicious area
M: number of sequences
N: size of a sequence
n: number of values in one cell

- for non-polymorphic known viruses
- false alarms: $p \approx \dfrac{L \cdot M}{n^N}$
- expense criteria: P, polynomial

$\leq L \cdot M \cdot N$ comparisions

“Heuristic” algorithm

VESZPROG

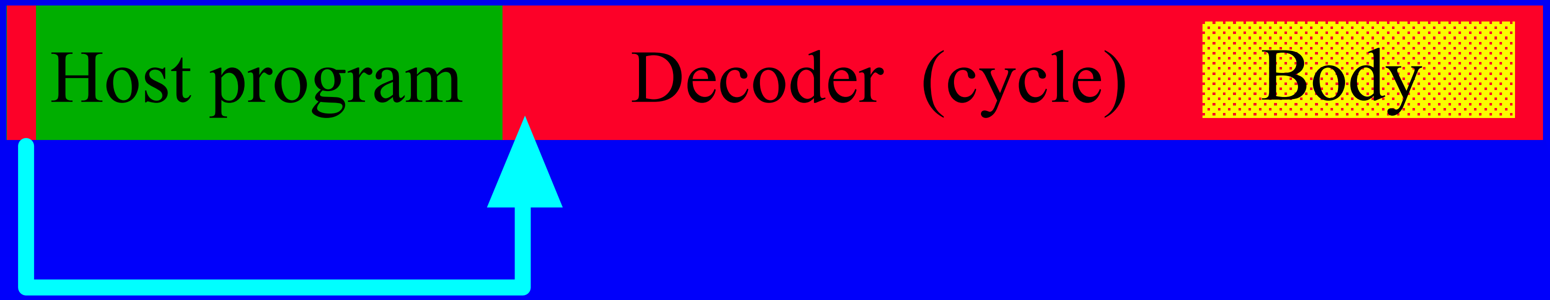
“Heuristic” algorithm

- for known viruses

“Heuristic” algorithm

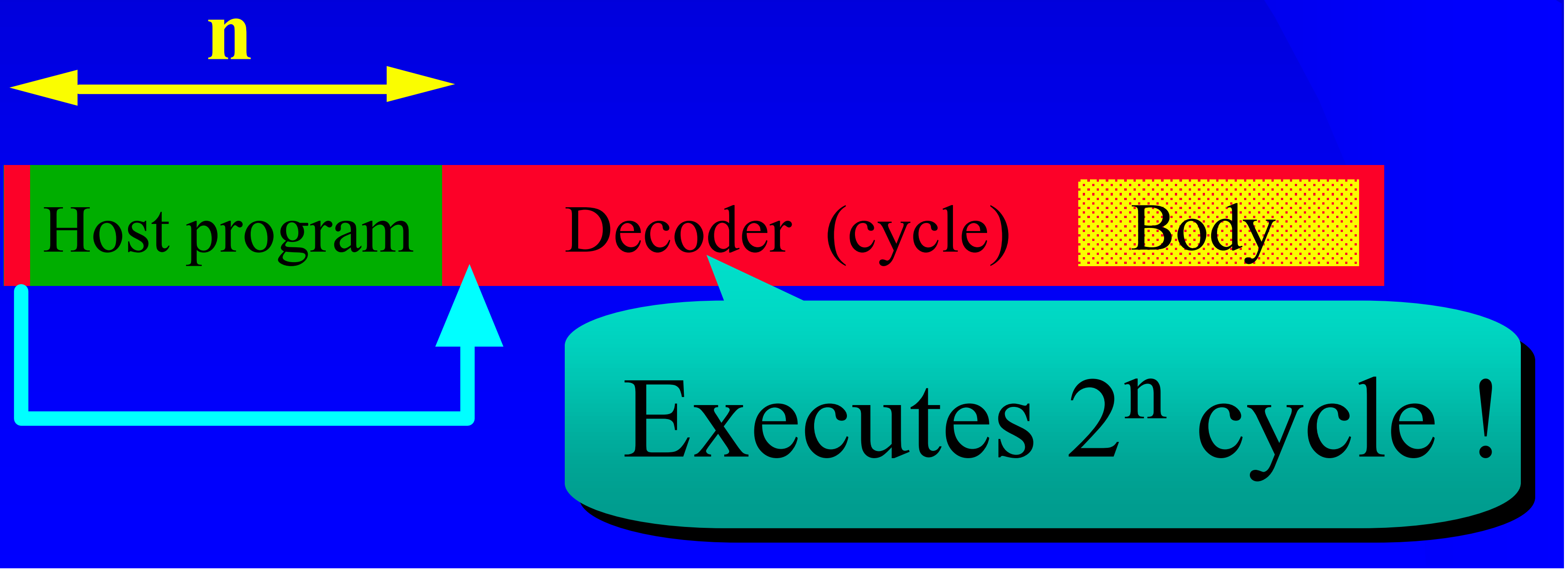
- for known viruses

- expense criteria:



“Heuristic” algorithm

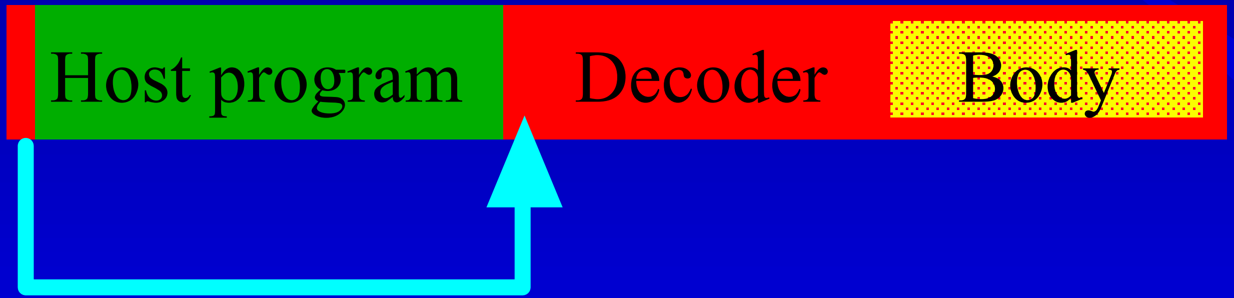
- for known viruses
- expense criteria: NP



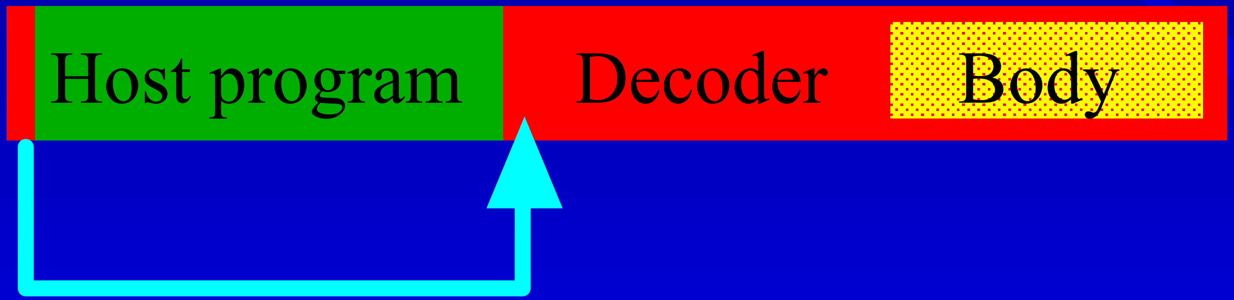
How can we measure the power of polymorphism ?

VESZPROG

How can we measure the power of polymorphism ?

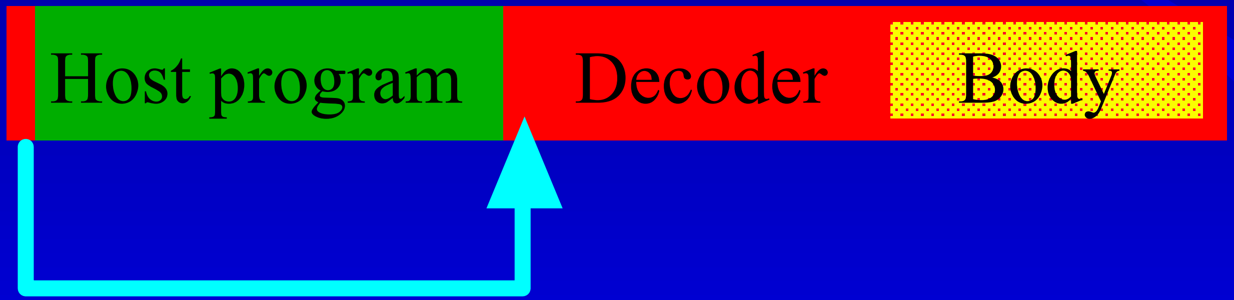


How can we measure the power of polymorphism ?



$$\alpha = \frac{\text{size of variable parts of the virus}}{\text{full size of the virus}}$$

How can we measure the power of polymorphism ?



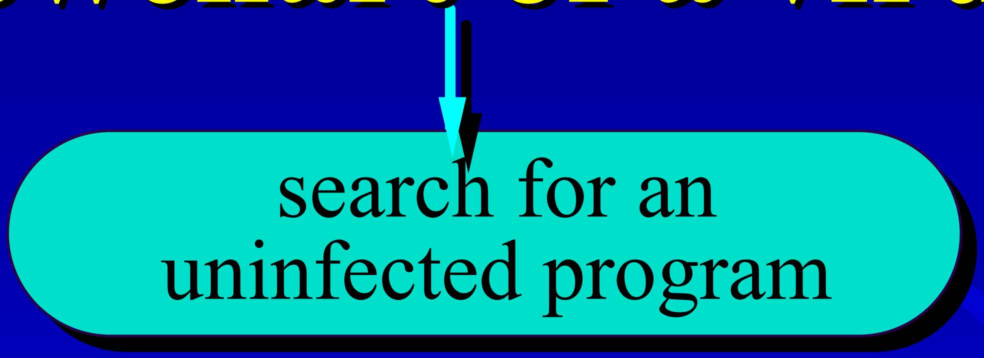
$$\alpha = \frac{\text{size of variable parts of the virus}}{\text{full size of the virus}}$$

β = number of variants of the decoders

Flowchart of a virus

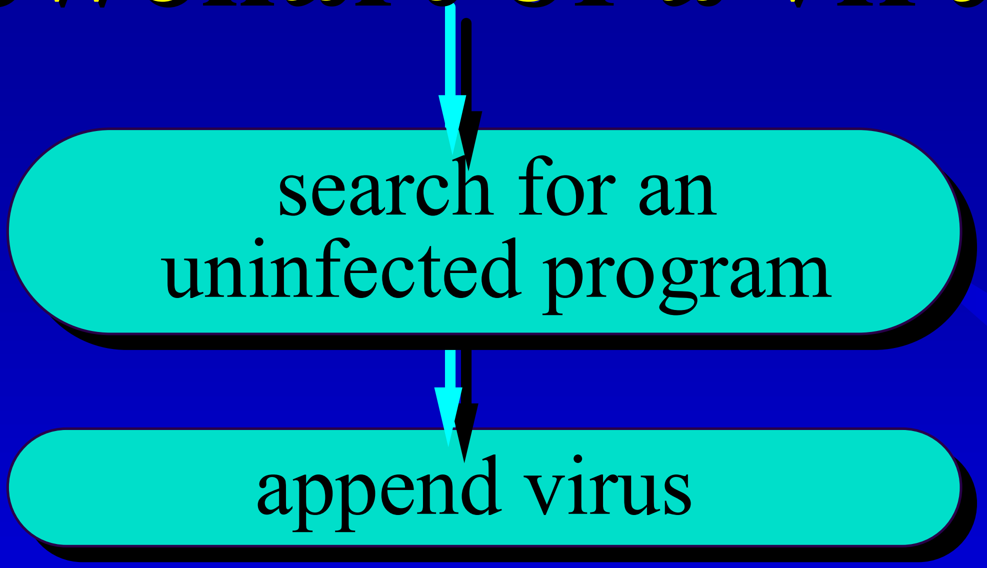
VESZPROG

Flowchart of a virus



search for an uninfected program

Flowchart of a virus

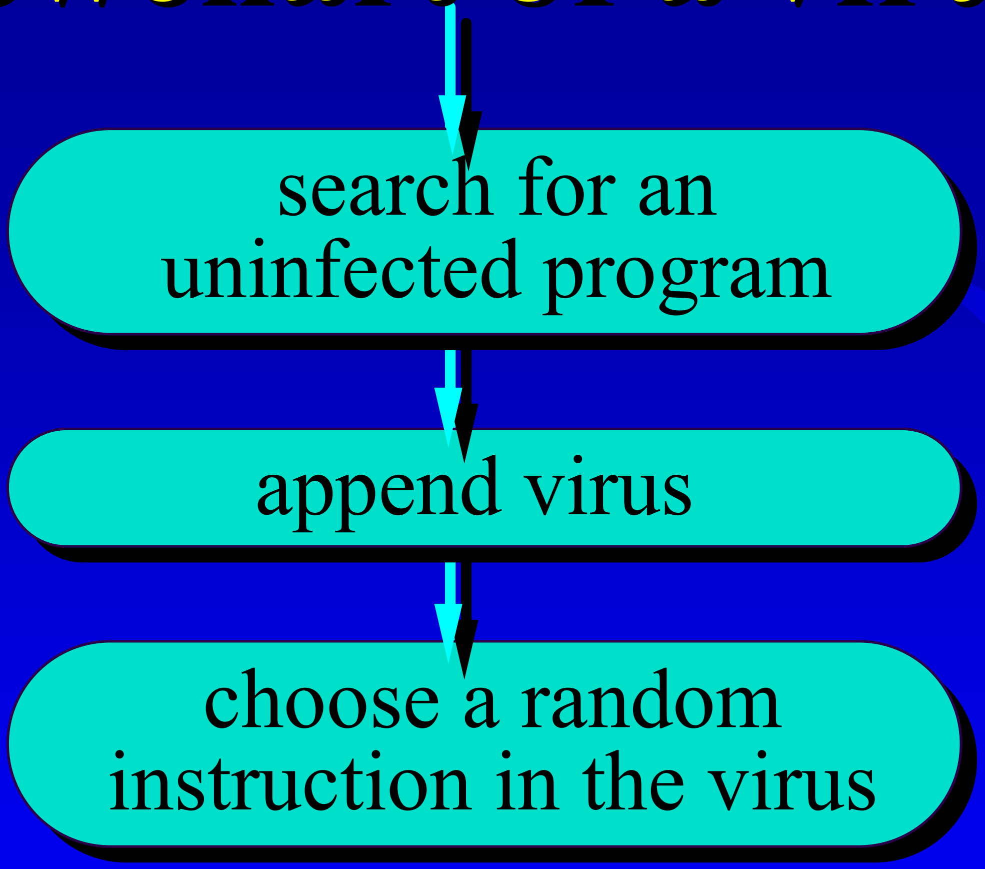


search for an uninfected program

append virus

Flowchart of a virus

VESZPROG

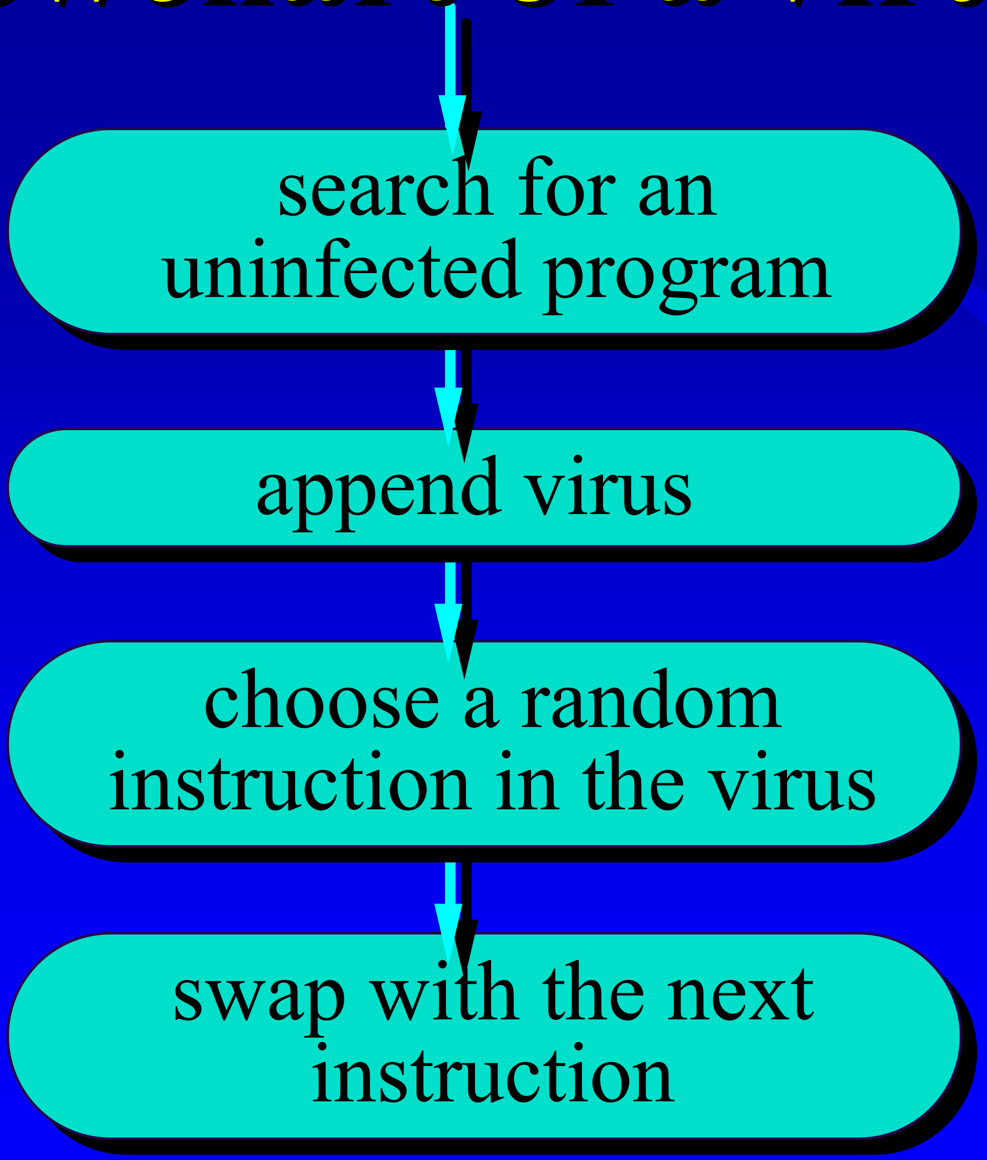


search for an uninfected program

append virus

choose a random instruction in the virus

Flowchart of a virus


search for an uninfected program

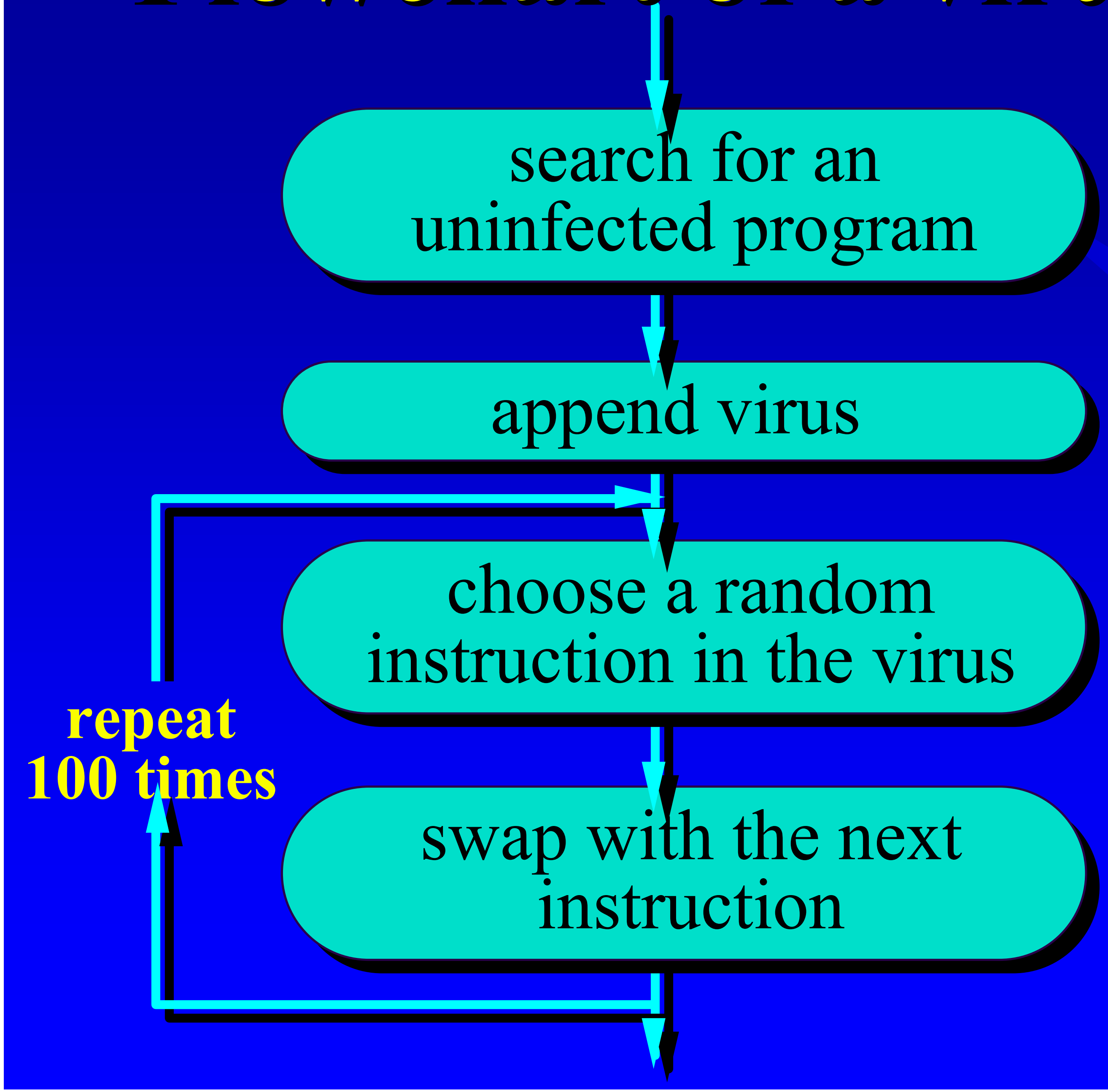
append virus

choose a random instruction in the virus

swap with the next instruction

VESZPROG

Flowchart of a virus



search for an uninfected program

append virus

choose a random instruction in the virus

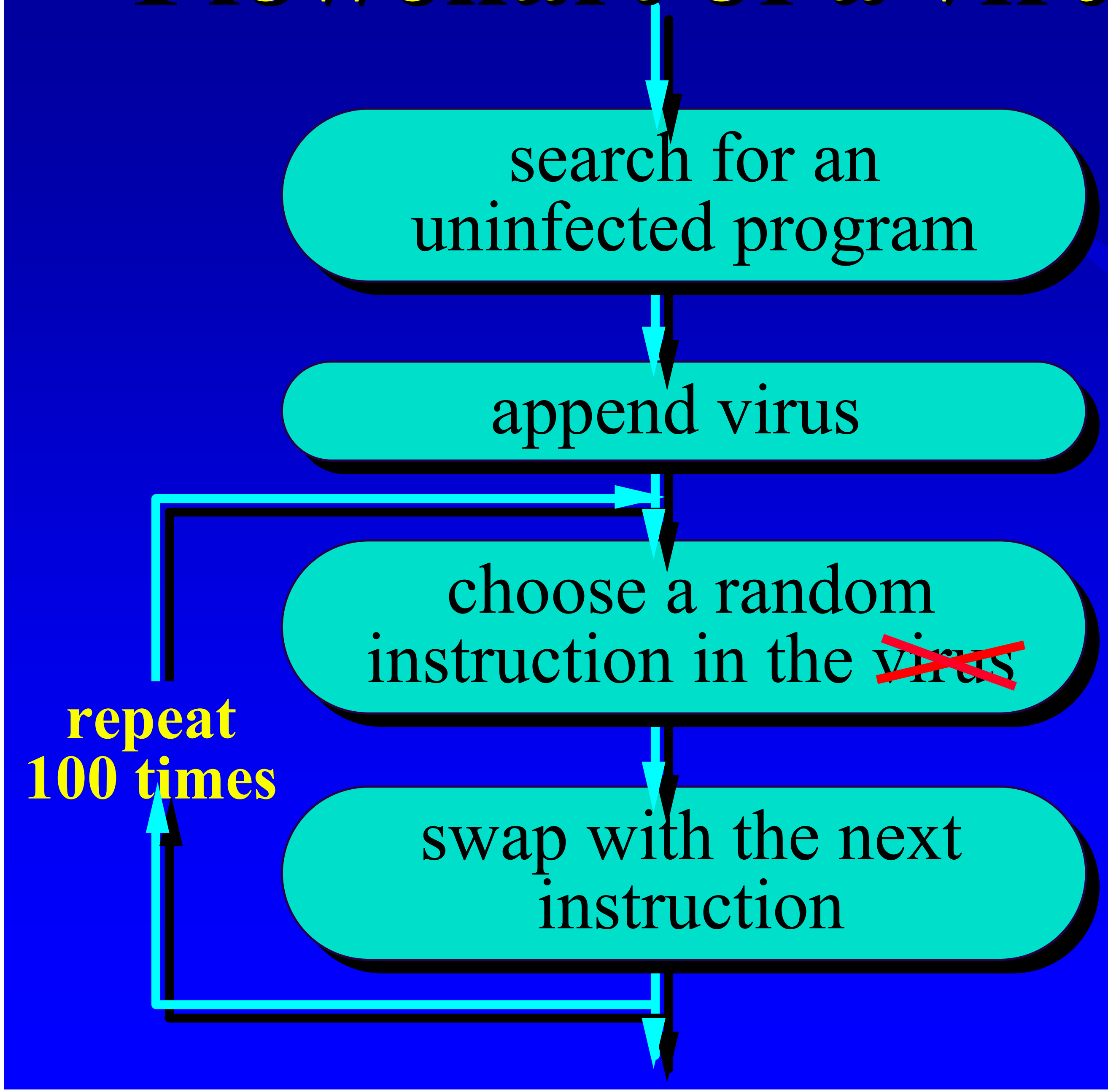
swap with the next instruction

repeat 100 times

VESZPROG

Flowchart of a virus

VESZPROG



search for an uninfected program

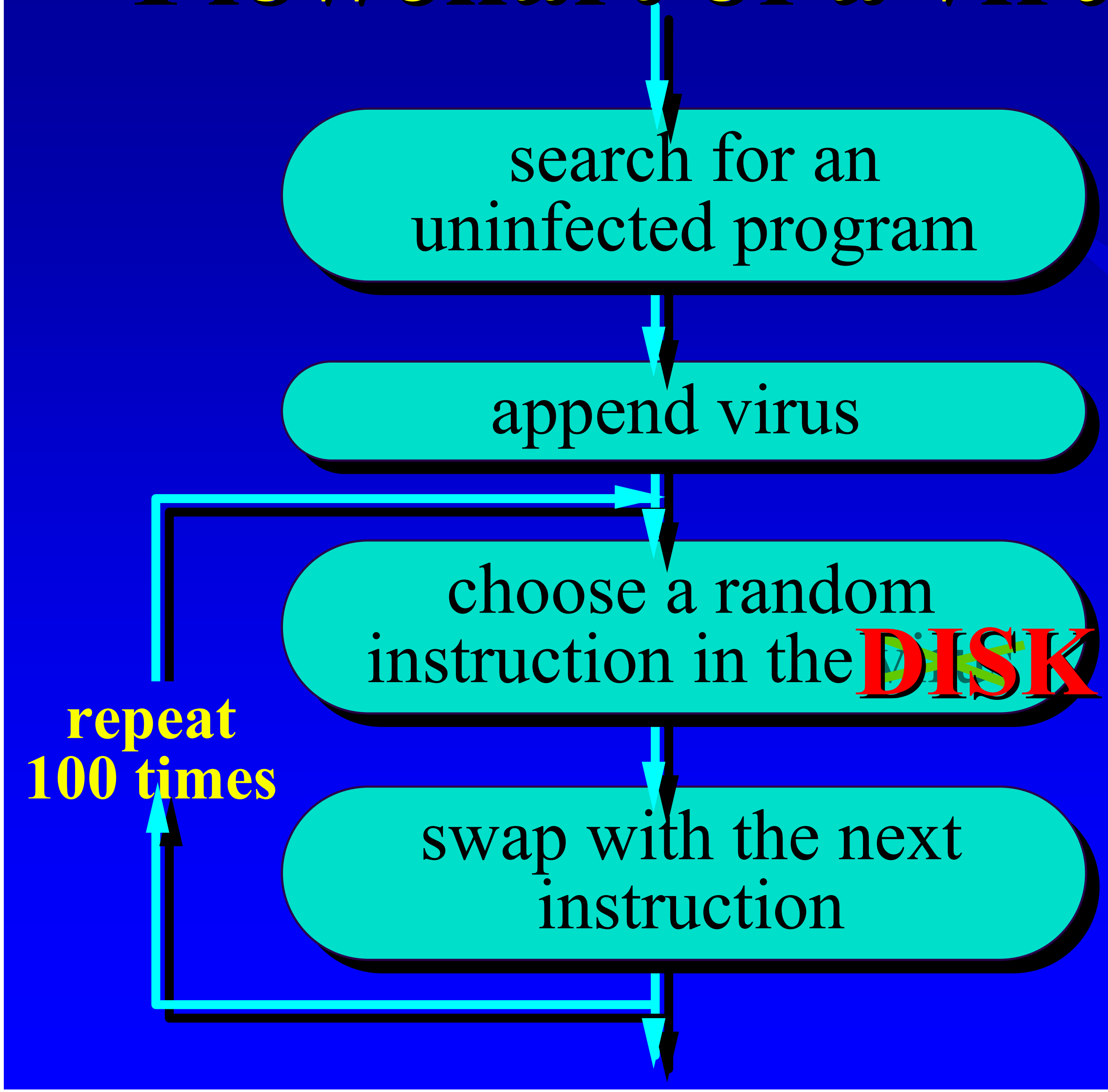
append virus

choose a random instruction in the ~~virus~~

swap with the next instruction

repeat 100 times

Flowchart of a virus



search for an uninfected program

append virus

choose a random instruction in the ~~virus~~ DISK

swap with the next instruction

repeat 100 times

Name:	RIPPER
Aliases:	Jack Ripper
Status:	Common
Origin:	Norway
Length:	1024 bytes (2 sectors)
Infect:	MBR, Boot sector
Other:	Resident, Stealth, Disk corruption

Name:	RIPPER
Aliases:	Jack Ripper
Status:	Common
Origin:	Norway
Length:	1024 bytes (2 sectors)
Infect:	MBR, Boot sector
Other:	Resident, Stealth, Disk corruption

VESZPROG

The virus swaps two words in the DOS write buffer. It occurs on a random basis of approximately 1 write in 1024 cases.

Name:	RIPPER
Aliases:	Jack Ripper
Status:	Common
Origin:	Norway
Length:	1024 bytes (2 sectors)
Infect:	MBR, Boot sector
Other:	Resident, Stealth, Disk corruption

VESZPROG

The virus swaps two words in the DOS write buffer. It occurs on a random basis of approximately 1 write in 1024 cases.

What can we do with this mathematical model ?

VESZPROG

- Examine the working mechanism of viruses
- Examine the virus detection problem
- Examine multiplatform viruses

What can we do with this mathematical model ?

- Examine the working mechanism of viruses
- Examine the virus detection problem
- Examine multiplatform viruses
- Examine new polymorphic virus types