# Mathematical model of computer viruses

*Ferenc Leitold,*

Hunix Ltd., Hungary

fleitold@hunix.hu

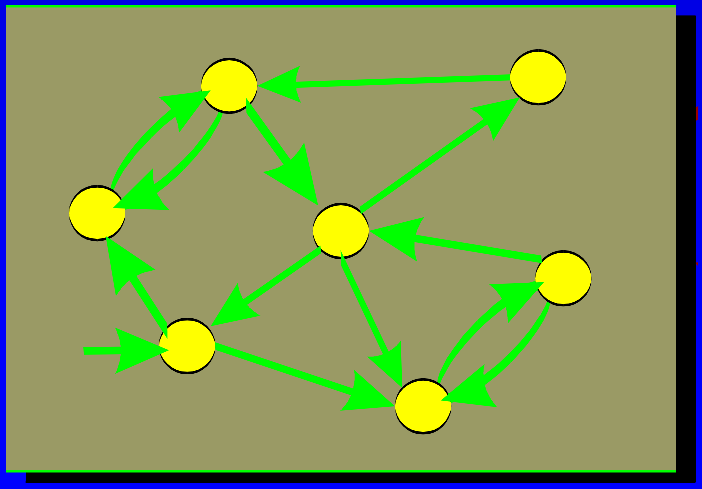
# Table of contents

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

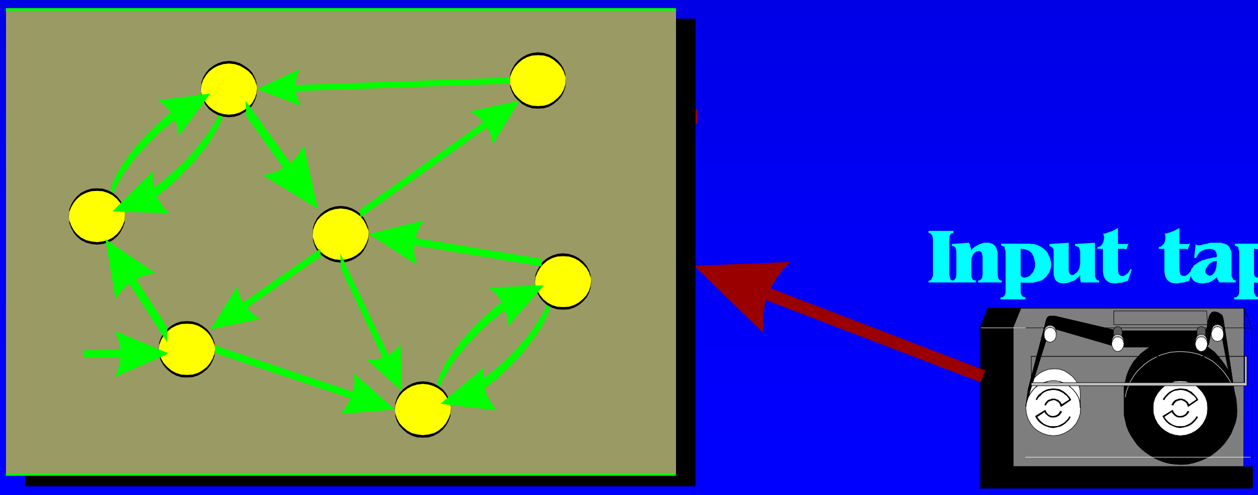
# Turing Machine

# Turing Machine

Finite automata



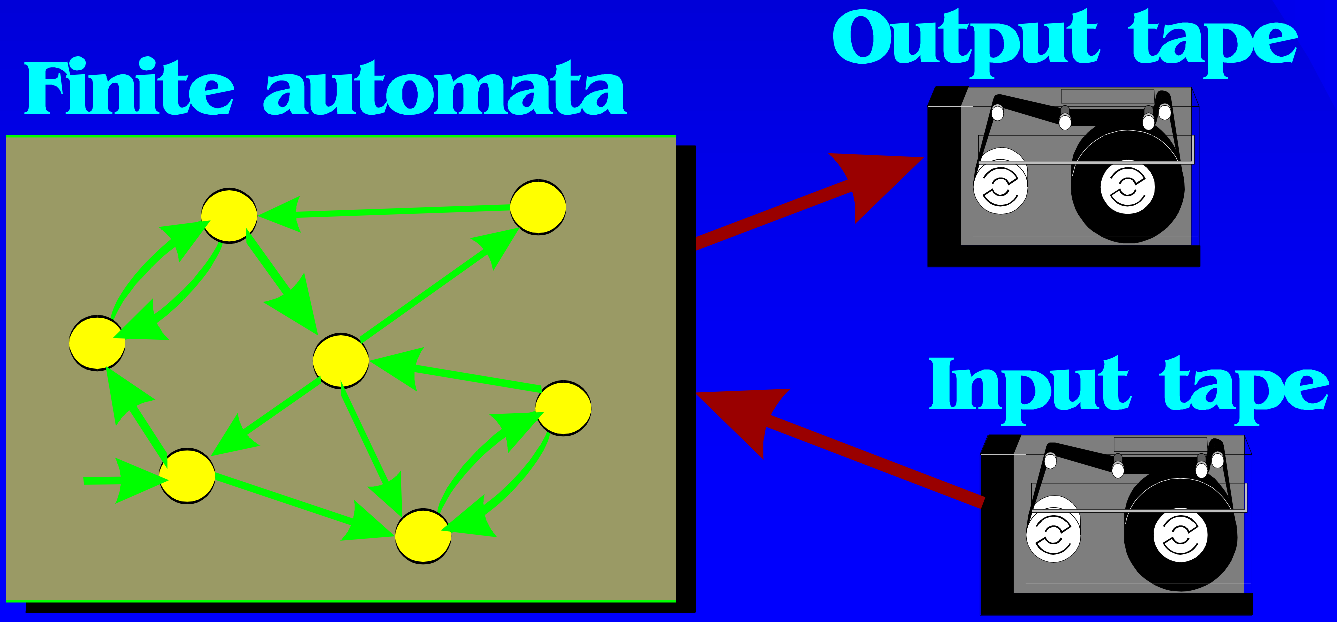
# Turing Machine



Finite automata

Input tape

# Turing Machine



Output tape

Finite automata

Input tape

# Turing Machine

Hungarian VirusBuster™

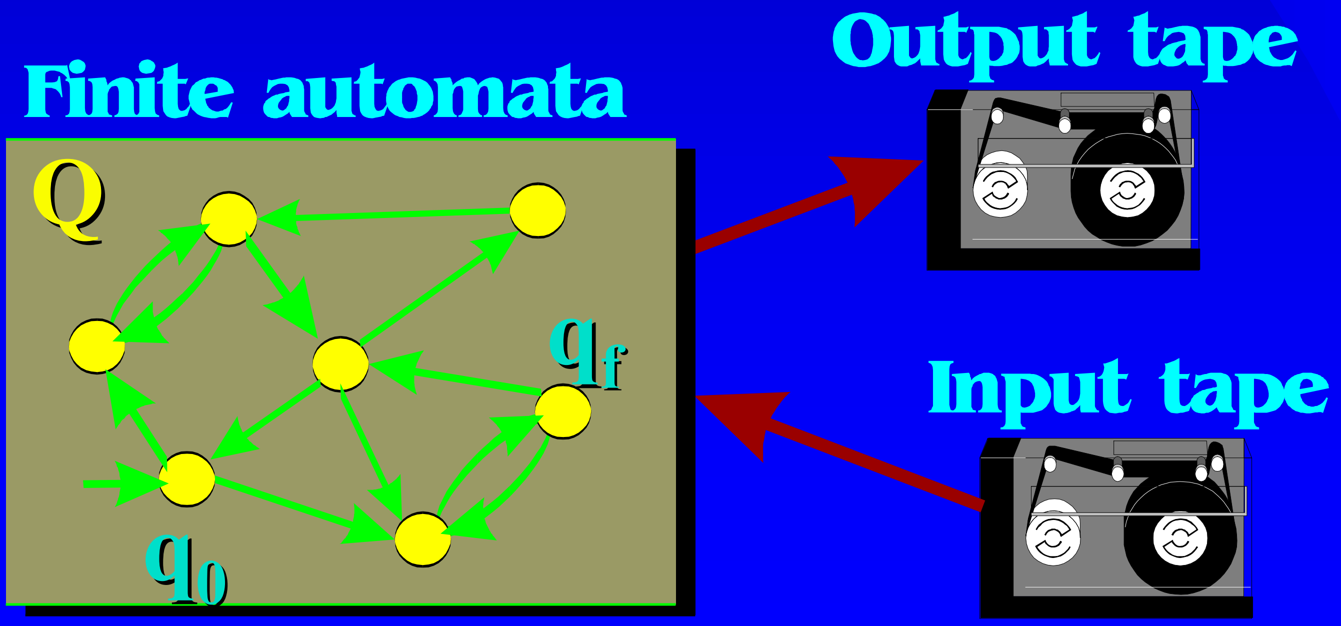
$$T = < Q, S, I, \delta, b, q_0, q_f >$$

S: tape symbols

I: input symbols, $I \subset S$

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

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

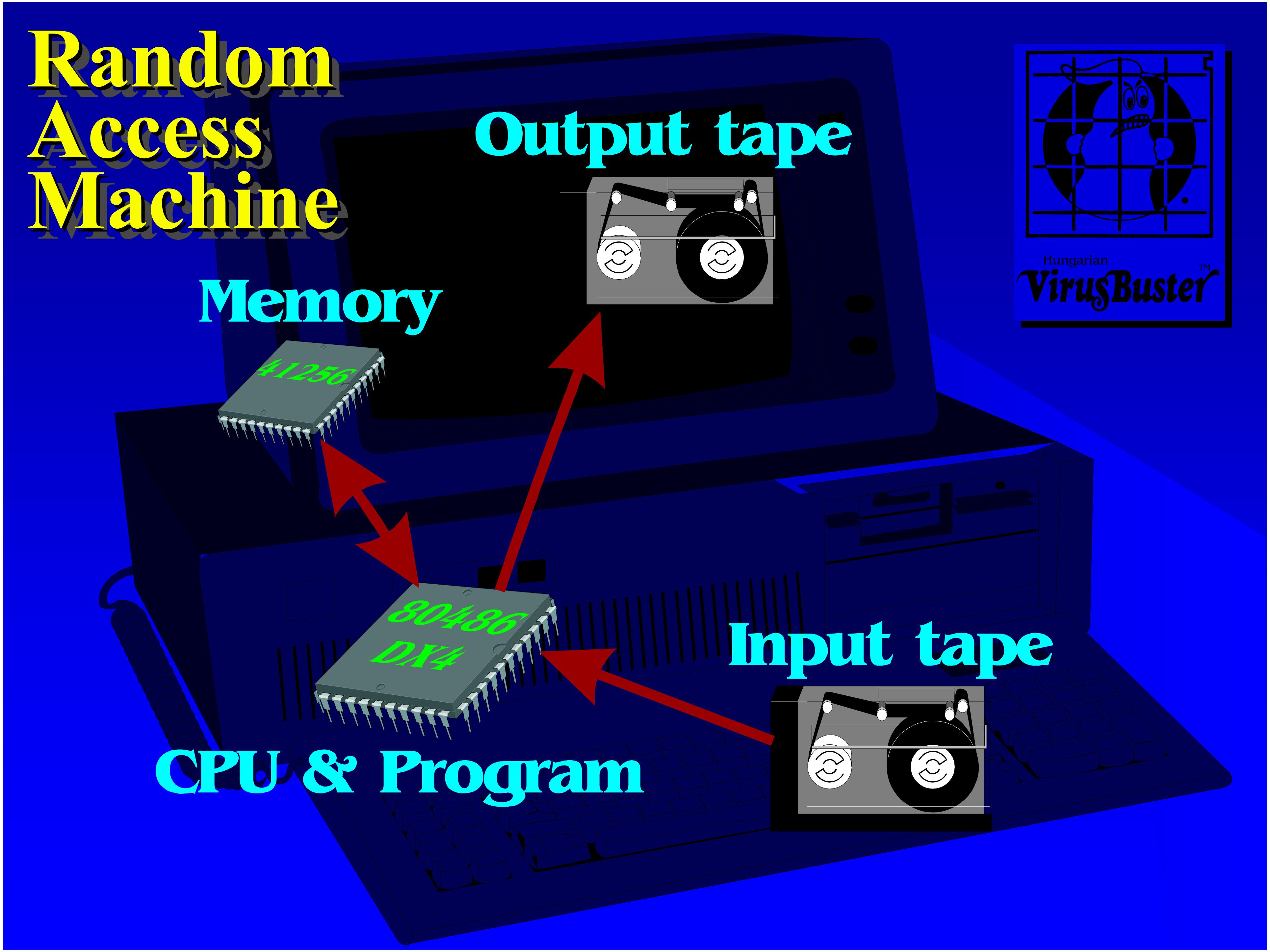


Finite automata

Output tape

Input tape

# Random Access Machine



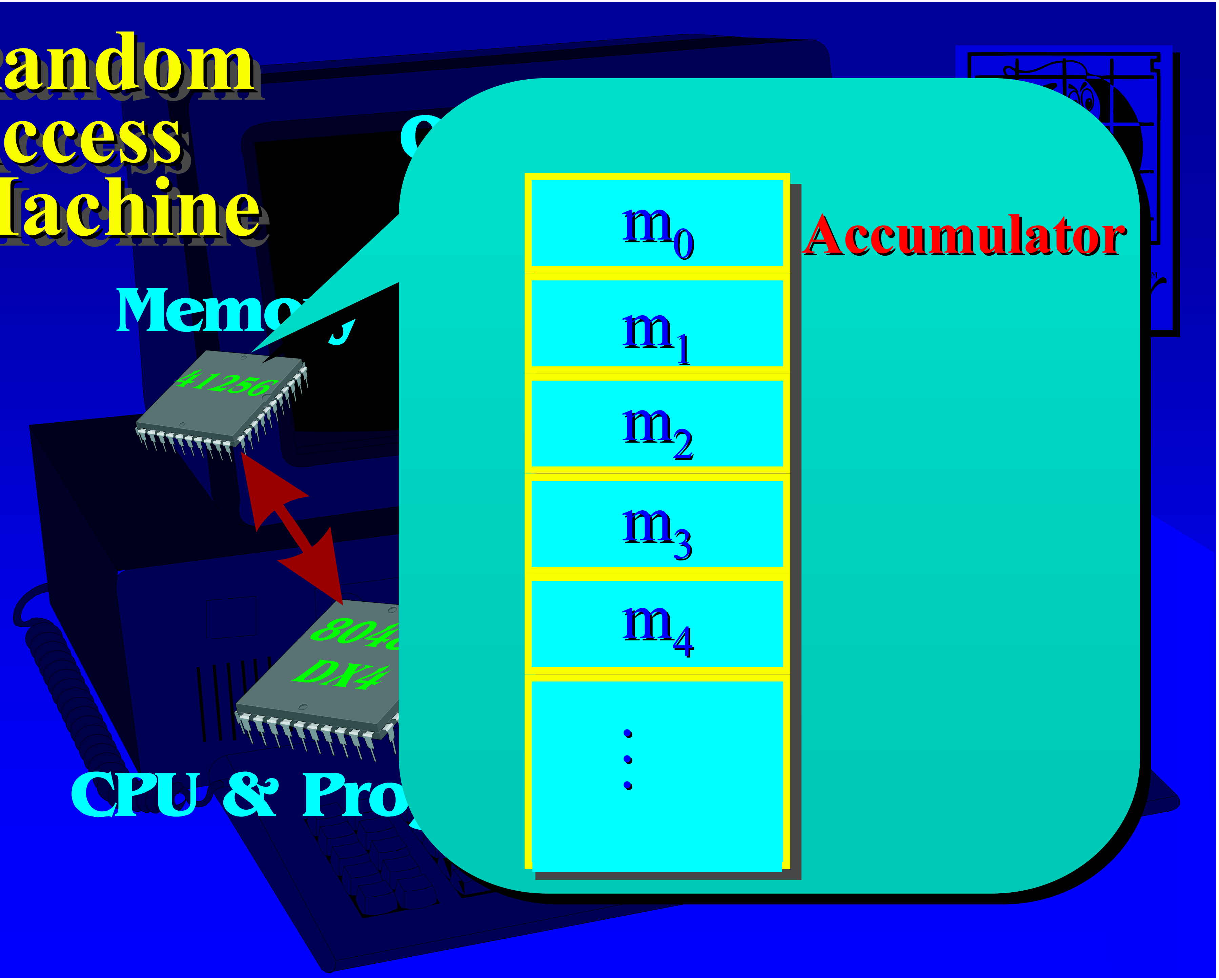
Output tape

Memory

Input tape

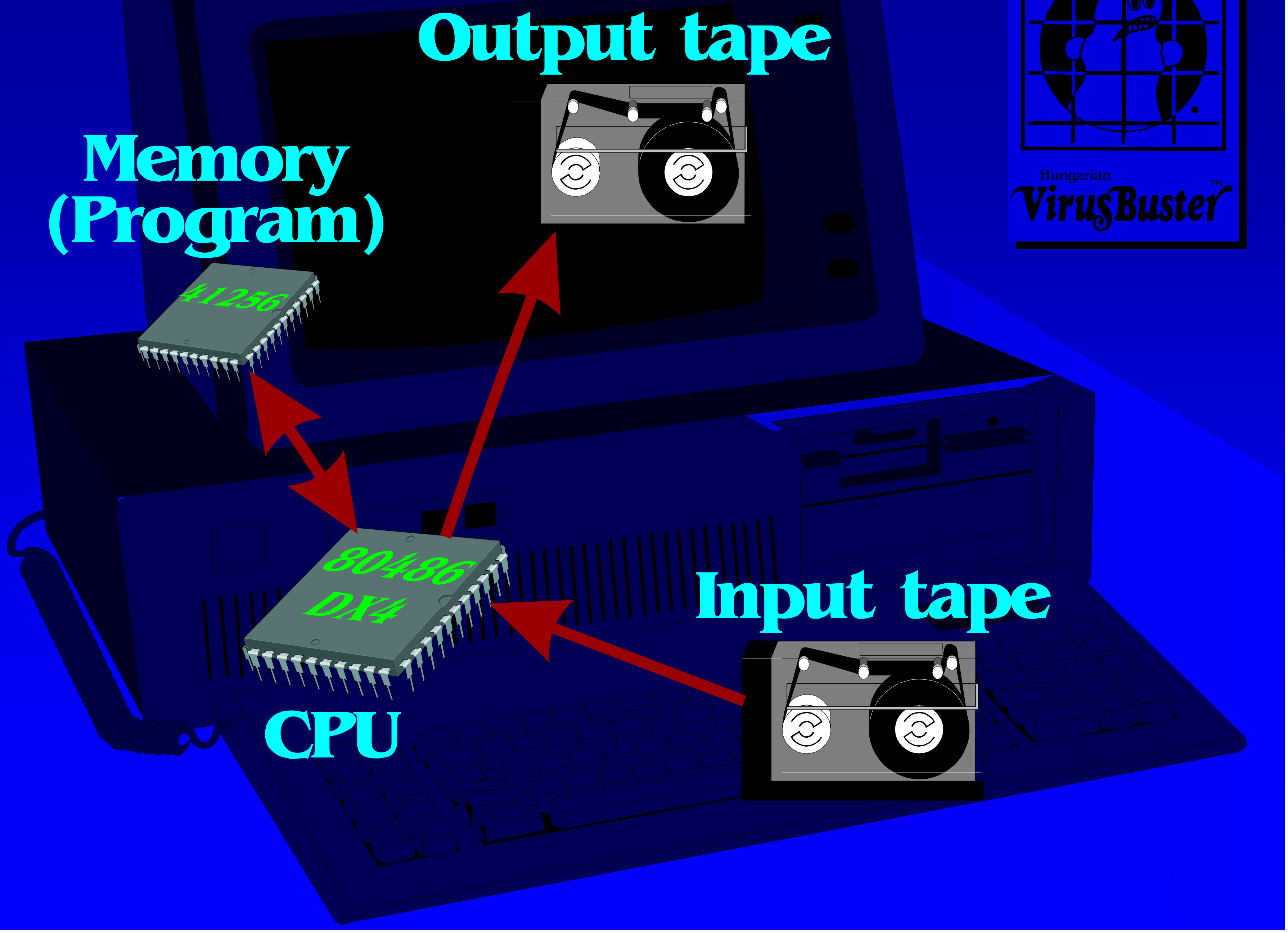
CPU & Program

# Random Access Machine



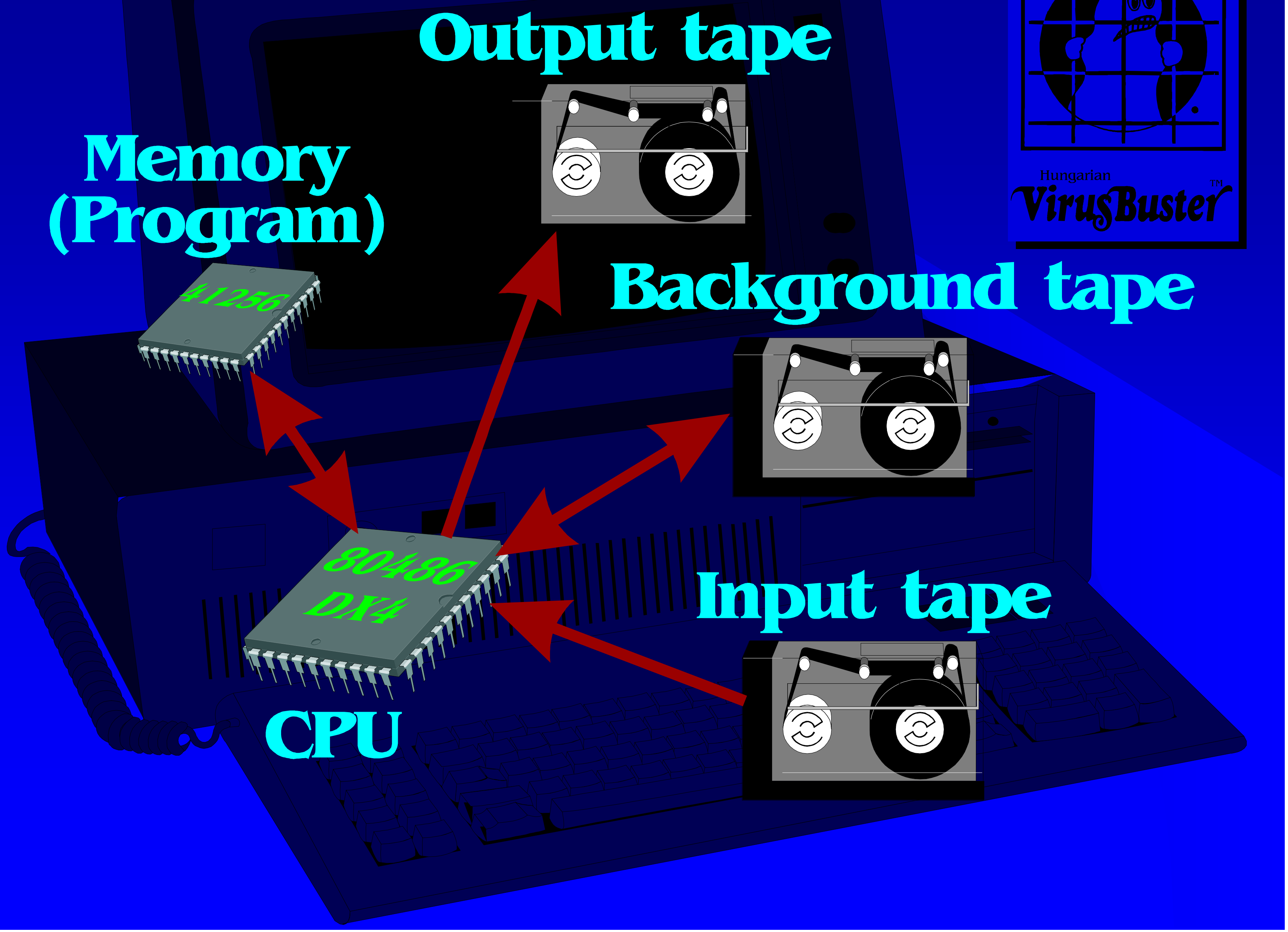
| | |
|---|---|
| $m_0$ | Accumulator |
| $m_1$ | |
| $m_2$ | |
| $m_3$ | |
| $m_4$ | |
| ⋮ | |

# RASPM



Hungarian VirusBuster™

# RASPM with ABS



Output tape

Memory (Program)

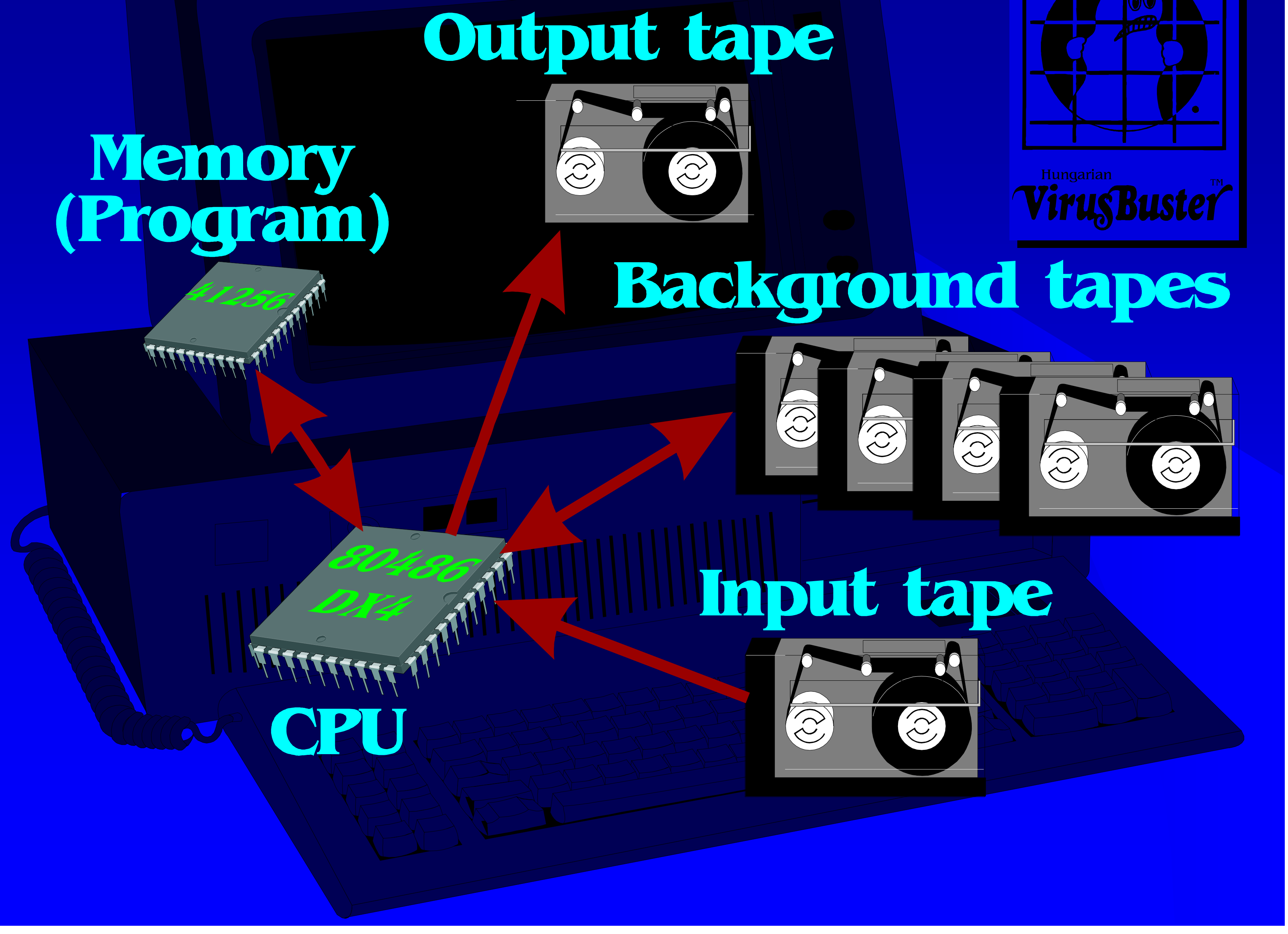
Background tape

Input tape

CPU

Hungarian VirusBuster™

# RASPM with SABS



Memory (Program)

Output tape

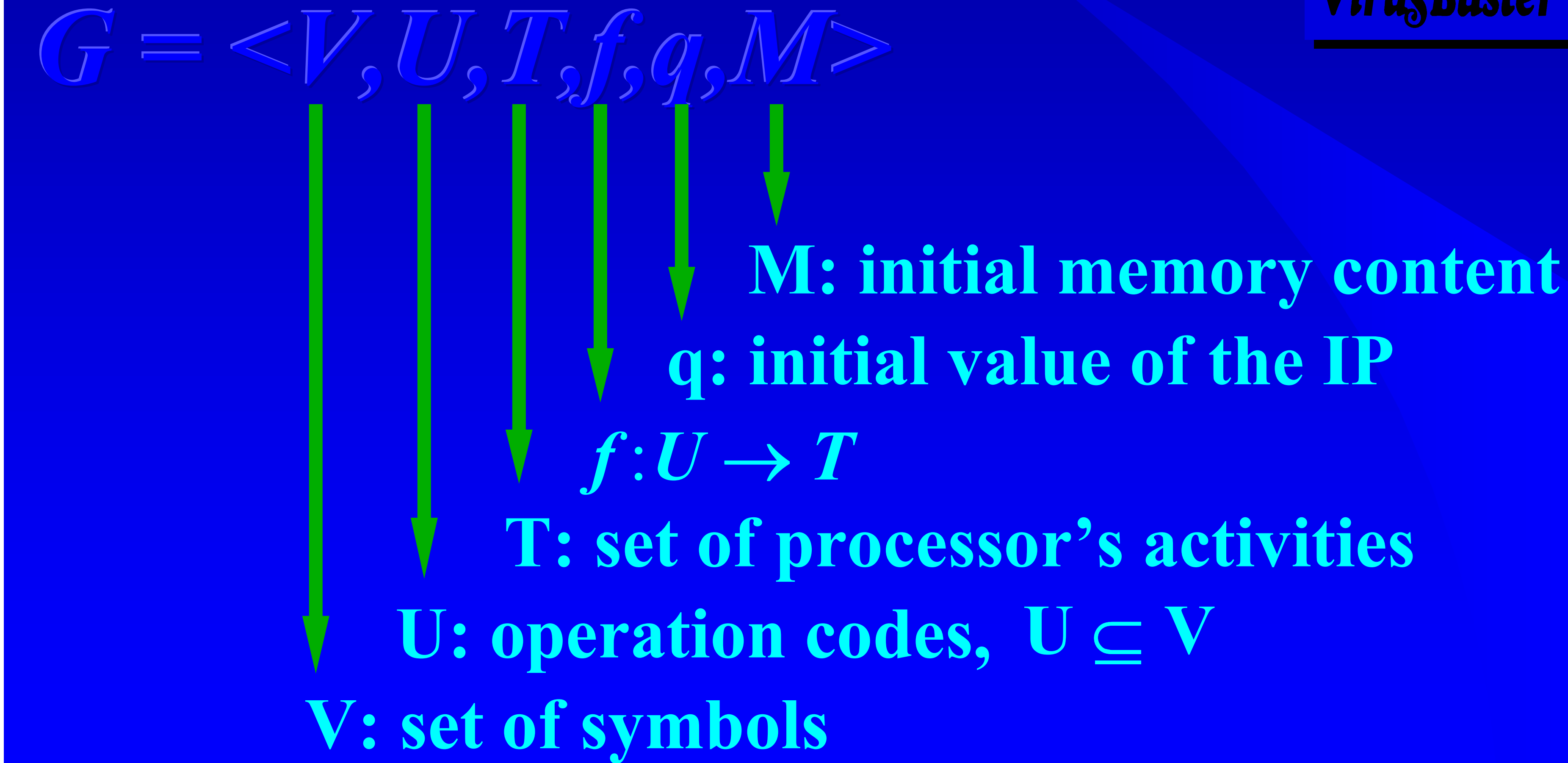
Background tapes

Input tape

CPU

Hungarian VirusBuster™

# RASPM with ABS definition



$G = \langle V,U,T,f,q,M \rangle$

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

Hungarian VirusBuster™

- 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

# Operating Systems under RASPM with ABS

- The OS is in the initial memory (M)

# Operating Systems under RASPM with ABS

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

# Operating Systems under RASPM with ABS

Hungarian VirusBuster™

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

# 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

# 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

# Comparing RASPM with ABS-es

Hungarian VirusBuster™

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

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

# Comparing RASPM with ABS-es

Hungarian VirusBuster™

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

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

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

# 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

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

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

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

# 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 instruction sets (activities)
- different sets of operation codes
- different operation codes

# Comparing RASPM with ABS-es

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

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

# 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

Hungarian VirusBuster™

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

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

$$V_1 \neq V_2$$

- different symbols
- different tape formats

# Computer virus

# Computer virus

Hungarian VirusBuster™

- a (part of) program

# Computer virus

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

# Computer virus

Hungarian VirusBuster™

- 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

# Virus spreading modes

- machine specific

# Virus spreading modes

- 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

# What can we do with this mathematical model ?

# What can we do with this mathematical model ?

- Examining virus detection problem

# What can we do with this mathematical model ?

- Examining virus detection problem
- Examining searching techniques

# What can we do with this mathematical model ?

- Examining virus detection problem
- Examining searching techniques
- Examining polymorphic viruses

# What can we do with this mathematical model ?

- Examining virus detection problem
- Examining searching techniques
- Examining polymorphic viruses
- Examining multiplatform viruses

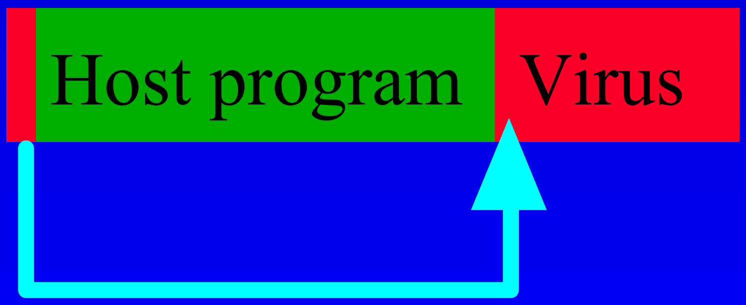
# General virus detection problem

## 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.

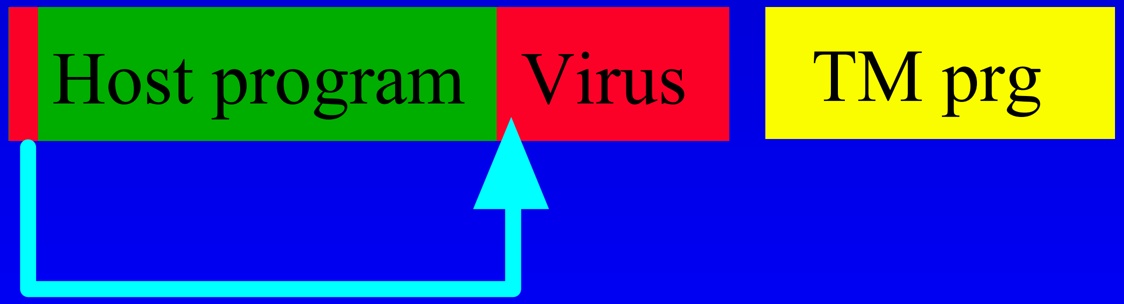
# General virus detection problem

## Proof:



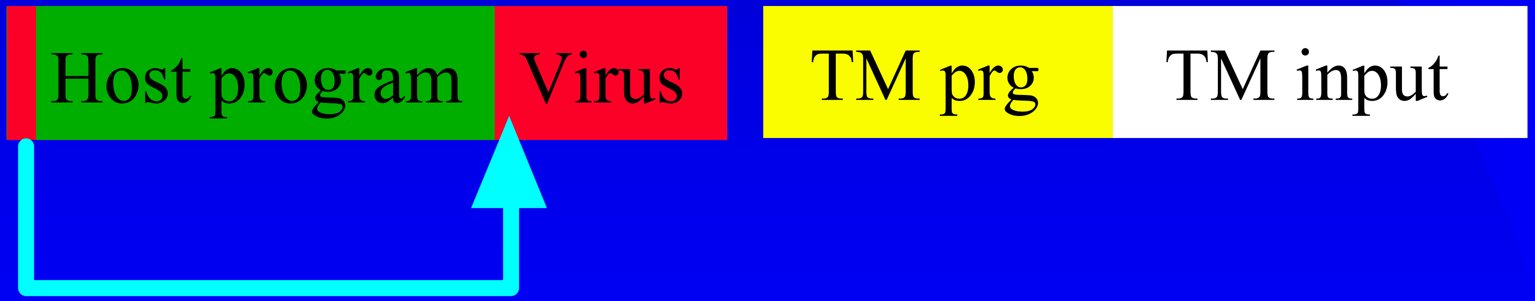
# General virus detection problem

## Proof:



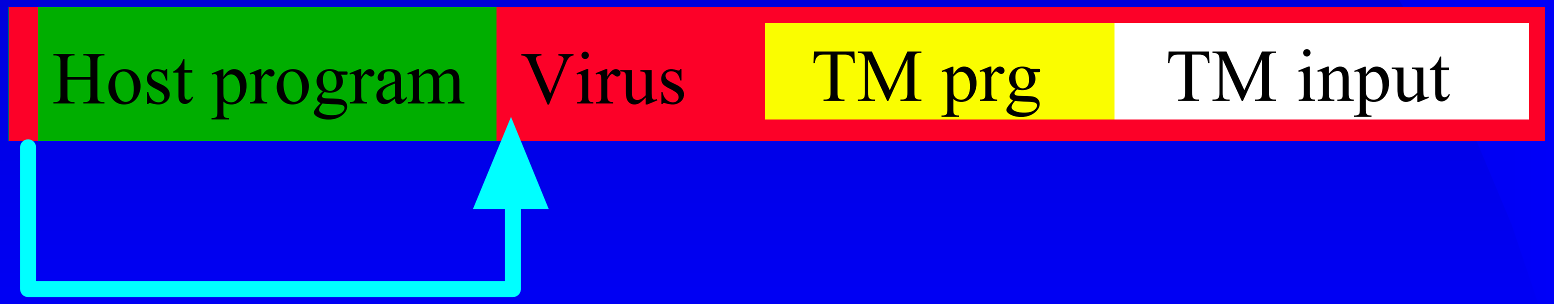
# General virus detection problem

## Proof:



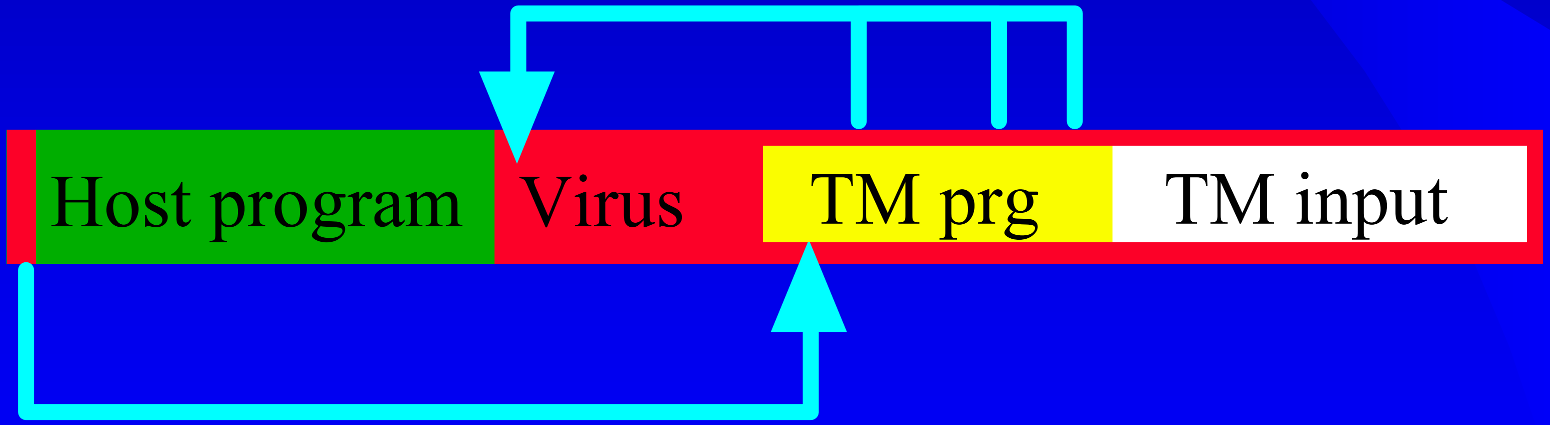
# General virus detection problem

## Proof:



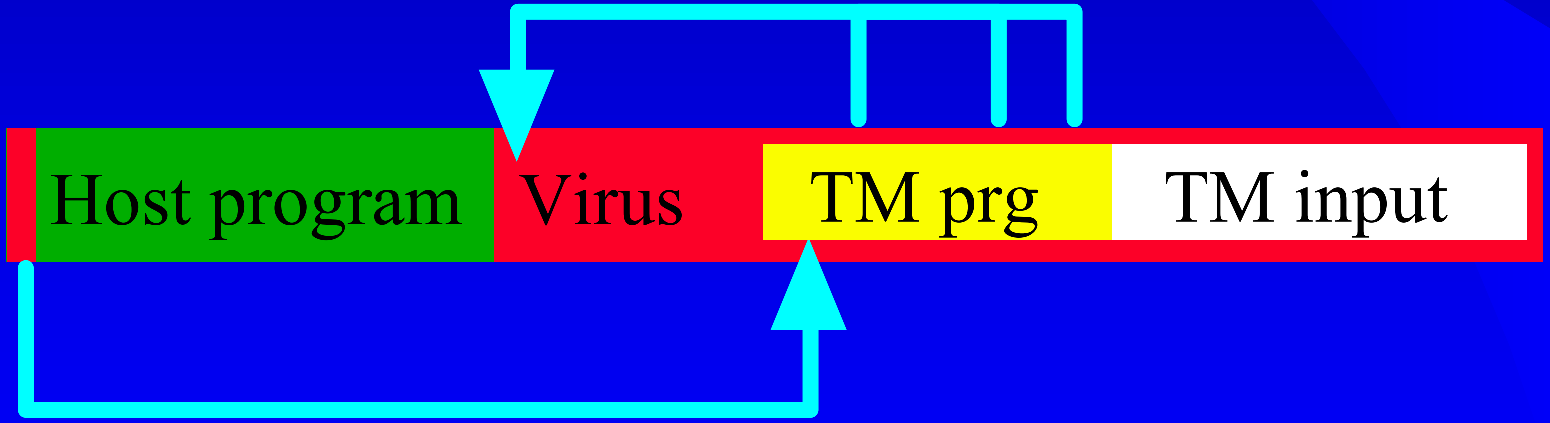
# General virus detection problem

Proof:



# General virus detection problem

## Proof:



Virus detection problem ➡ TM halting problem

“An anti-virus has its limit,
thanks to Turing,
and a virus can find those limits,
exploit them,
thanks to Darwin.”

from the Giant Black Book of Computer Viruses

# Searching technique questions

# 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

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

# Sequence searching algorithm

# 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: $p \approx \dfrac{L \cdot M}{n^{N}}$

# 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

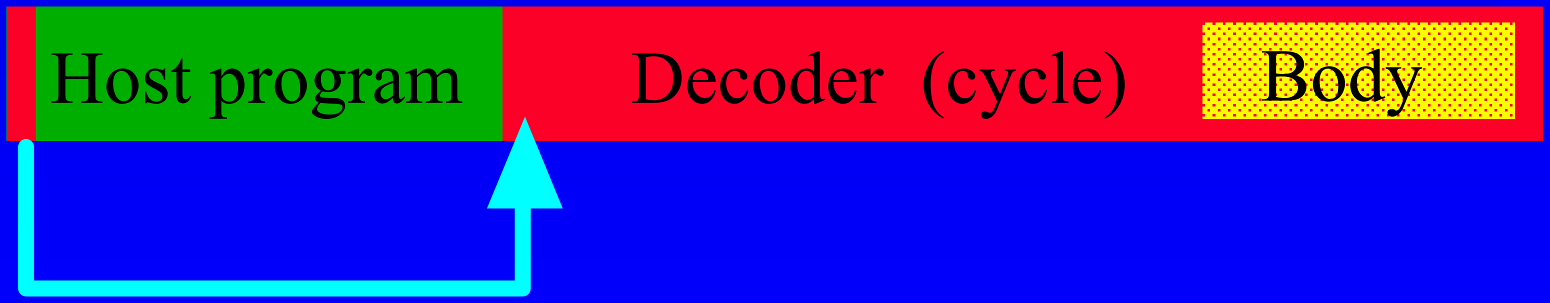
Hungarian VirusBuster™

# “Heuristic” algorithm

- for known viruses

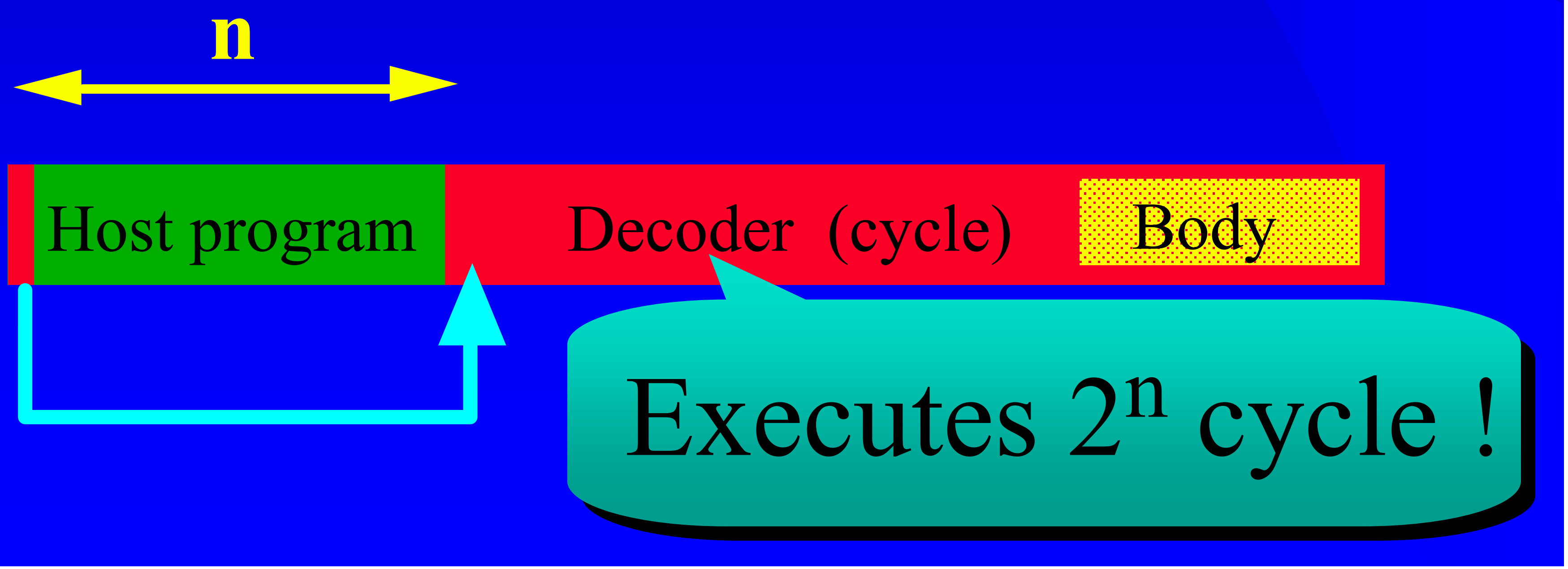
# “Heuristic” algorithm

- for known viruses
- expense criteria:



# “Heuristic” algorithm

- for known viruses
- expense criteria: NP

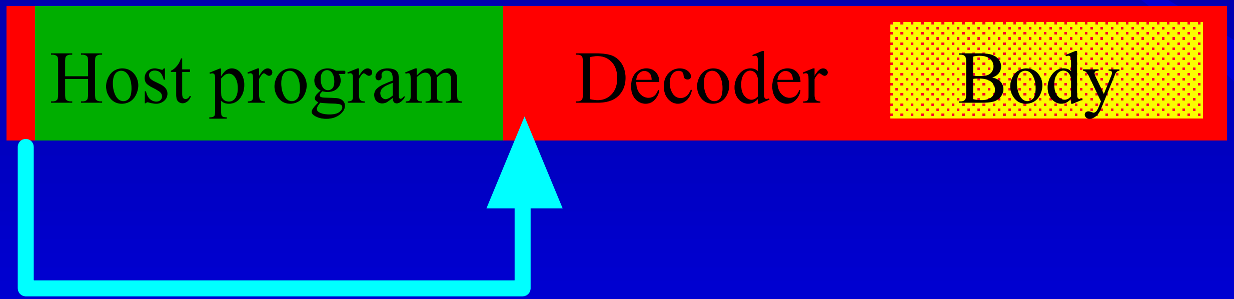


Executes $2^n$ cycle !

# How can we measure the power of polymorphism ?

# How can we measure the power of polymorphism ?

Hungarian VirusBuster™



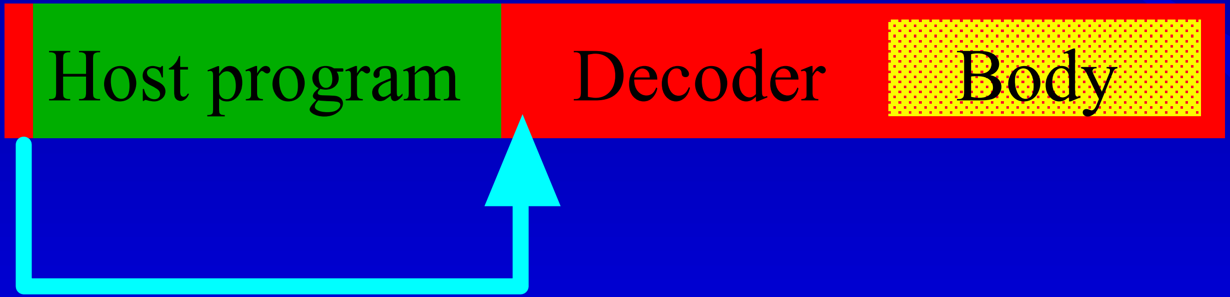
Host program

Decoder

Body

# How can we measure the power of polymorphism ?

Hungarian VirusBuster™



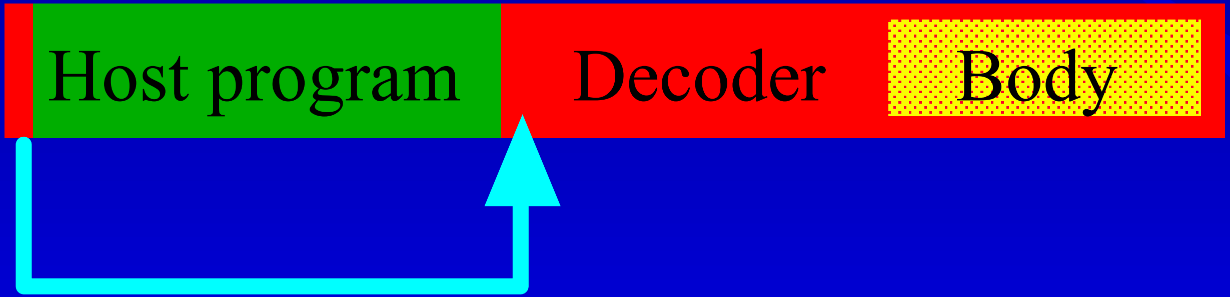
Host program

Decoder

Body

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

# How can we measure the power of polymorphism ?



Host program

Decoder

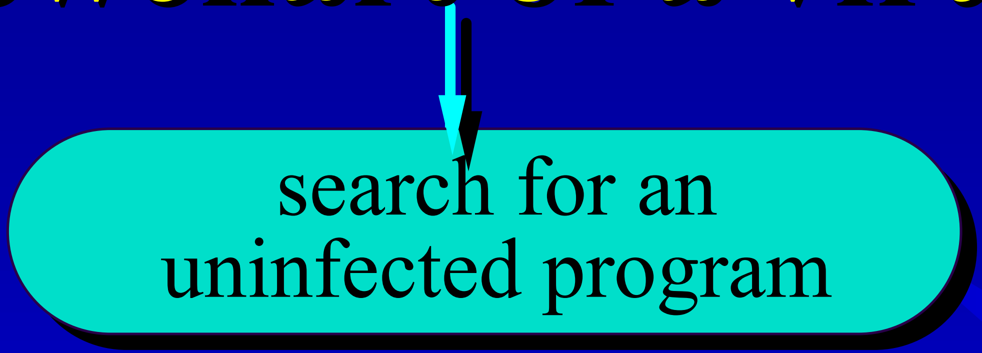
Body

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

$$\beta = \text{number of variants of the decoders}$$

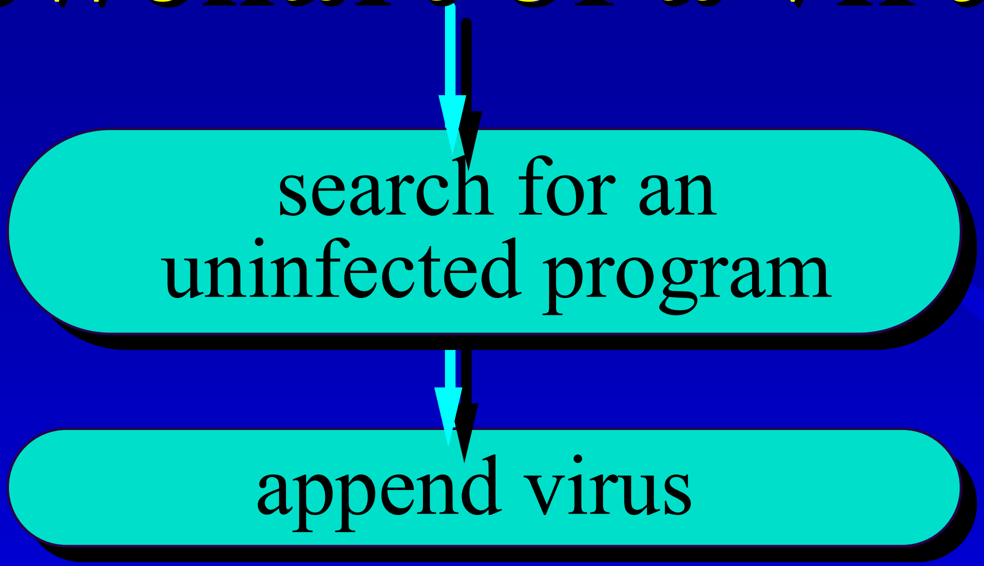
# Flowchart of a virus

# Flowchart of a virus



search for an uninfected program

# Flowchart of a virus

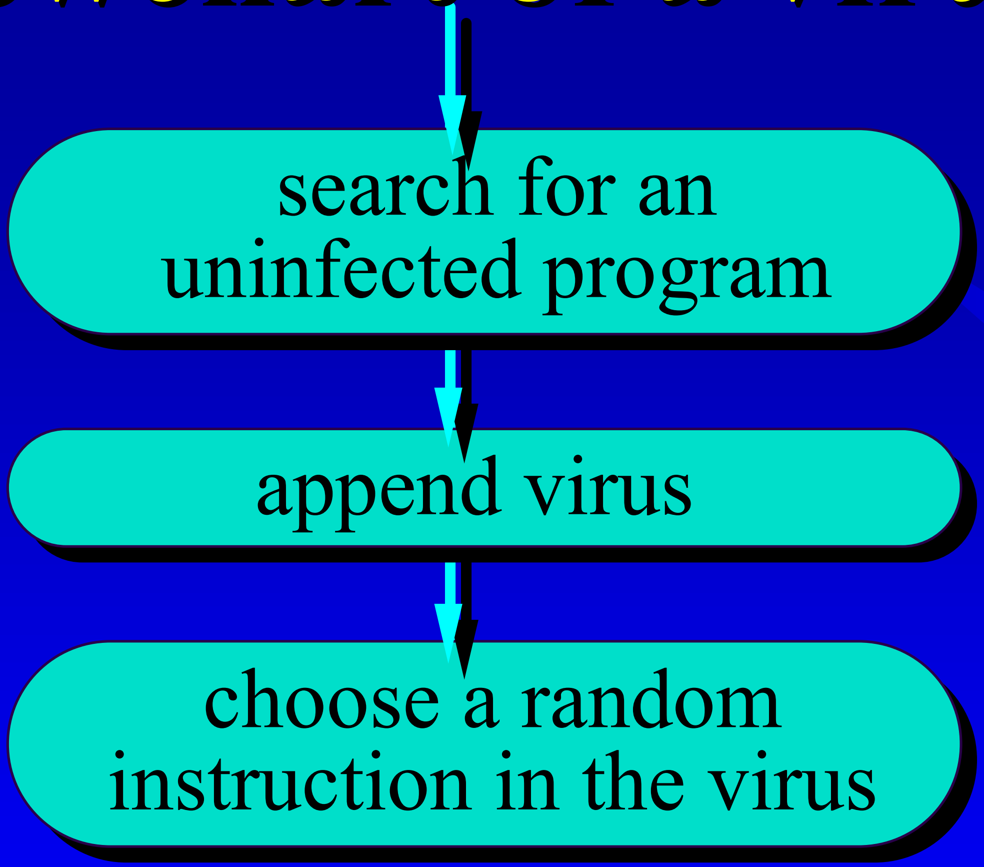


search for an uninfected program

append virus

Hungarian VirusBuster™

# Flowchart of a virus



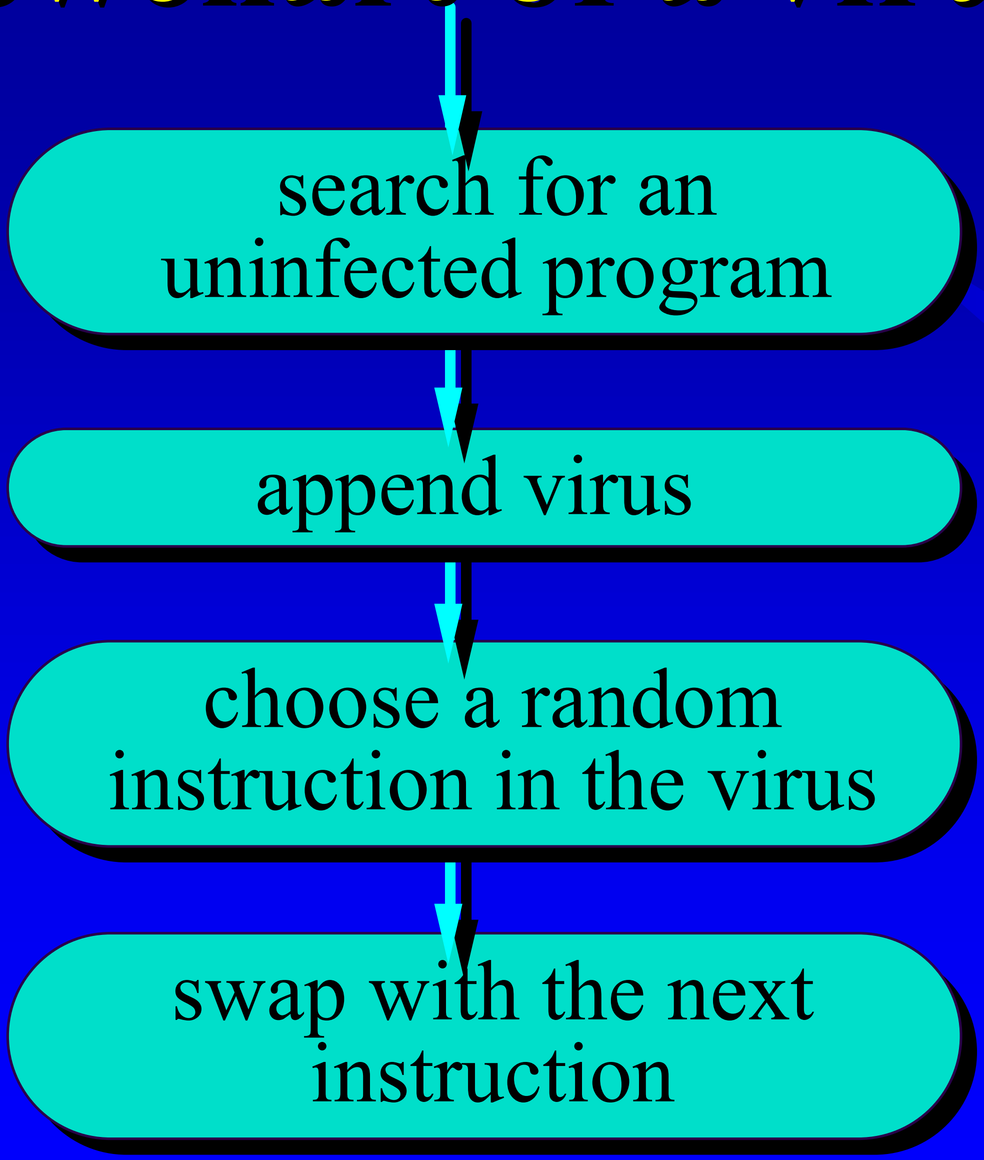
search for an uninfected program

append virus

choose a random instruction in the virus

Hungarian VirusBuster™

# Flowchart of a virus



search for an uninfected program

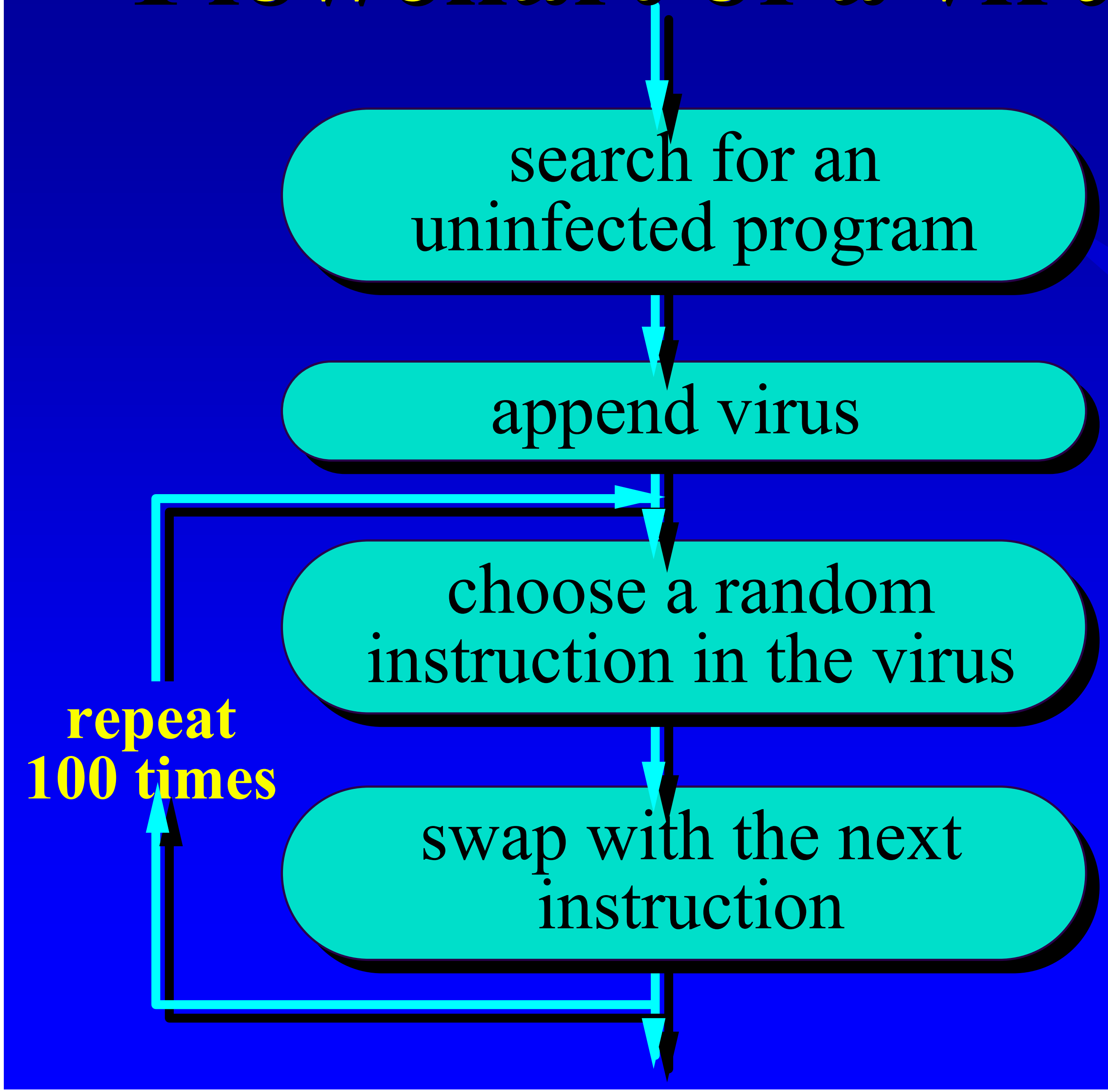
append virus

choose a random instruction in the virus

swap with the next instruction

Hungarian VirusBuster™

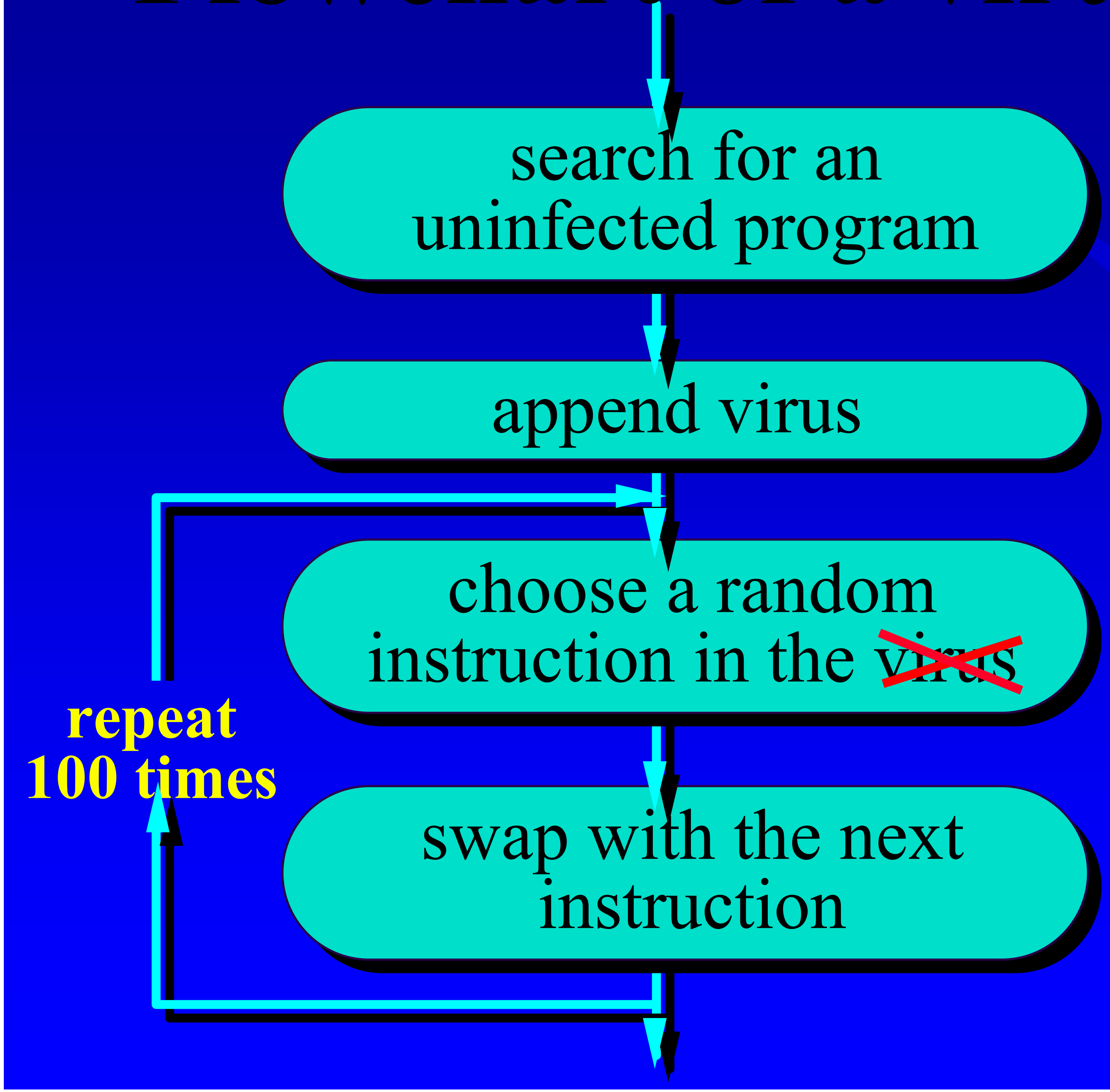
# Flowchart of a virus



search for an uninfected program
append virus
choose a random instruction in the virus
swap with the next instruction
repeat 100 times

# Flowchart of a virus

Hungarian VirusBuster™



search for an uninfected program

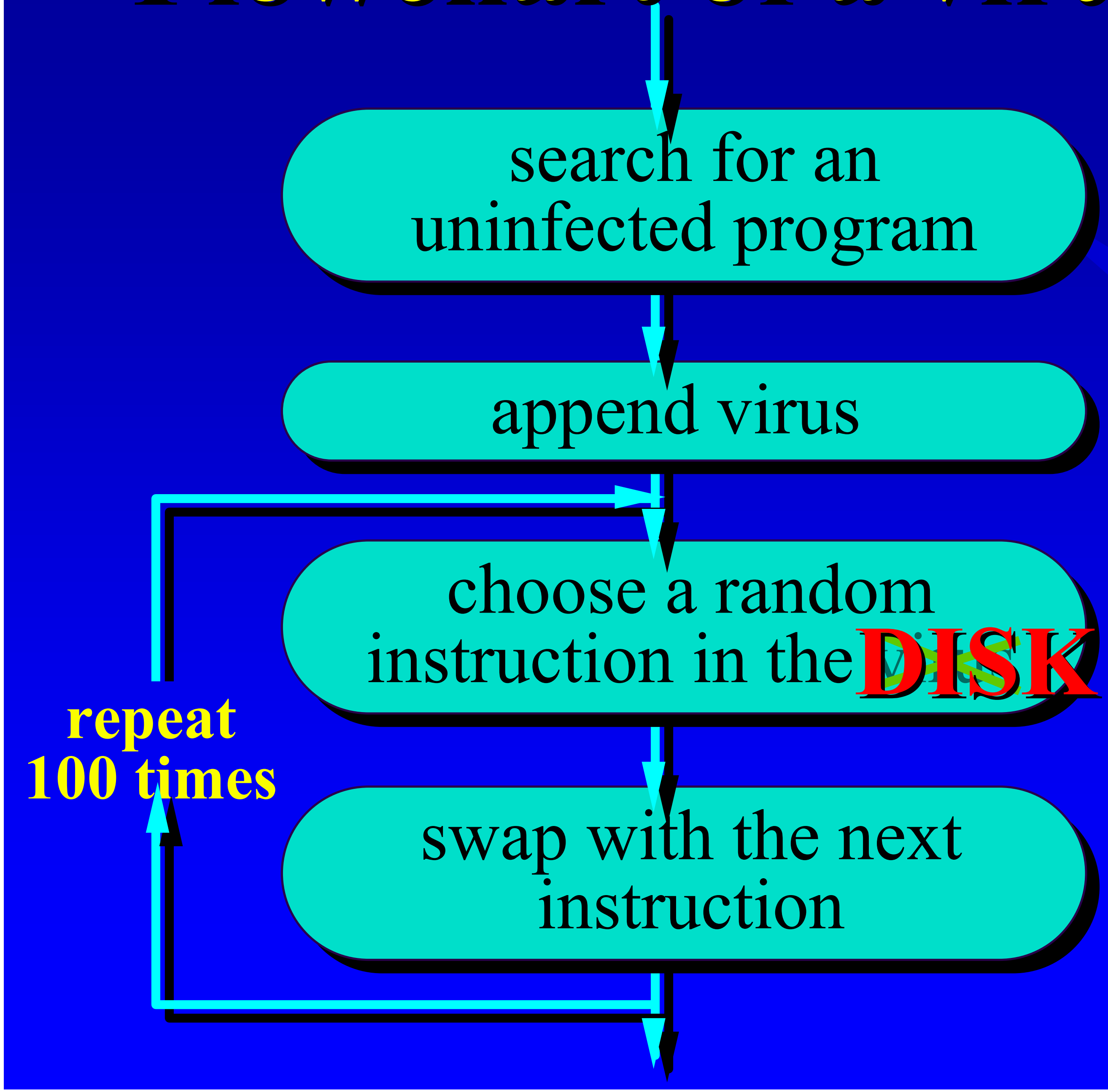
append virus

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,<br>Disk corruption |

Hungarian VirusBuster™

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

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

# 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

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

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

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

Hungarian VirusBuster™

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

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

Conditions:

$U_1$ ✋ $U_2 \neq 0$

**- The virus code can be the same.**

# Multiplatform viruses

Hungarian VirusBuster™

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

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

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.**

Hungarian
VirusBuster™

# ?